PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

LARBI BEN M'HIDI UNIVERSITY OUM EL BOUAGHI
Faculty of Exact Sciences, Nature Sciences and Life
Department of Mathematics and Computer Science
Option: Engineering of Distributed System

Thesis

For the PhD degree obtention in Computer Science

Theme:

System level energy consumption optimization in network on chip (NoC) for multimedia applications

Presented in: **02 / 01 / 2017**

Presented by:
Belkebir Djalila.

Jury members:

Dr. Bourouis AbdElhabib (MCA)	Oum El Bouaghi University	President
Dr. Merah El Kamel (MCA)	Khenchela University	Examiner
Pr. Benmohammed Mohamed	Constantine 2 University	Examiner
Dr. Boutekkouk Fateh (MCA)	Oum El Bouaghi University	Supervisor

Academic Year 2016-2017

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE LARBI BEN M'HIDI OUM EL BOUAGHI
FACULTÉ DES SCIENCES EXACTES ET DES SCIENCES DE LA NATURE ET DE LA VIE
DEPARTEMENT DE MATHEMATIQUES ET D'INFORMATIQUE
Option : Ingénierie des Systèmes Distribuées

Thèse

Pour l'Obtention du diplôme de Doctorat troisième cycle en Informatique

Thème :

Optimisation de la consommation d'énergie au niveau système pour les applications multimédia sur les réseaux sur puce (NOC)

Soutenu le: **02 / 01 / 2017**

Présenté par :
Belkebir Djalila.

Membres de jury :

Dr. Bourouis AbdElhabib (MCA)	Université d'Oum El Bouaghi	Président
Dr. Merah El Kamel (MCA)	Université de khenchela	Examinateur
Pr. Benmohammed Mohamed	Université de Constantine 2	Examinateur
Dr. Boutekkouk Fateh (MCA)	Université d'Oum El Bouaghi	Rapporteur

Année Universitaire 2016-2017

Acknowledgements

The endless thanks go to ALLAH Almighty for all the blessings he has showered onto me, which has enabled me to write this last note in my research work.

I am deeply indebted to my research supervisor, Dr. Fateh Boutekkouk for presenting me such an interesting thesis topic. Each meeting with him added invaluable aspects to the implementation and broadened my perspective. He has guided me with his invaluable suggestions, lightened up the way in my darkest times and encouraged me a lot in the academic life.

Words can never be enough in expressing how grateful I am to those incredible parents who made this thesis possible and who have made me what I am. Thank you

ملخص

نظرا للتطور الواسع لتقنيات معالجة التكامل في العصرالحاضر، هذه الشبكات المستخدمة عن طريق الشرائح المتميزة بأدائها العالي و التطور مع نظام الشريحة هي التي اعتبرت كحل واعد لتحقيق صفقات سريعة ذات تكاليف منخفضة و حلول مرنة.

مع ذلك فأن تطبيقات الوسائط المتعددة الحالية المدمجة أصبحت أكثر تعقيدا بسبب تعدد الوظائف المدمجة، غالبا هاته التطبيقات مدمجة في الأجهزة المحمولة التي تفرض استهلاك منخفض للطاقة ، وفي هذا الصدد فان التركيز ليس قائم علي إدماج واحدة من تطبيقات الوسائط المتعددة في منصة تشغيل متعددة النواة بل في إدماج جميع التطبيقات في شريحة واحدة. و لهذا فان استراتيجيات الإدماج التي تسند مهام التطبيقات إلى المعالجين في الشريحة، عليها أن تكون لغرض التخفيض قدر المستطاع من استهلاك الطاقة مع احترام القيود التوقيتية للتطبيقات.

في الواقع, الخوارزميات الجينية مستخدمة على نطاق واسع لحل مشاكل النظام المعقد وذلك لسهولة استخدامها والتنفيذ بشكل عام و قدرتها للوصول إلى حلول مثالية لكن الخوارزمية المستوحاة من الكم التطوري فاقت الخوارزمية الكلاسيكية لقدرتها على حل مشاكل في وقت مزدوج و الذي اعتبر مستحيلا تحقيقها إلا في غضون مليون عام .الخوارزمية المستوحاة من الكم التطوري تمكنت من تحقيق توازن أفضل بين الاكتشاف و استغلال فضاء البحث من اجل حلول مثالية.

من ناحية أخرى، الآلات الخلوية هي أنظمة ديناميكية حيث أن واحدة من سماتها الرئيسية هي خاصية قواعد الانتقال المتميزة ببنية جدا بسيطة، لكن نتائج سلوكها العام يمكن أن تكون معقدة للغاية. أن بساطة الآلات الخلوية التي لها نموذج رياضي بسيط هو سبب استخدامها بقوة في النظام المركب خصوصا في شبكات الشرائح حيث يتم دمج عدد كبير من المعالجات على شريحة واحدة،و هذا الأخيرة عليها أن تتحمل ضغط عالي ،استهلاك الطاقة مع احترام القيود التوقيتية لأجل ذلك في أطروحتي درسنا كيف الخوارزميات الوراثية الكمية قادرة على تطوير الآلات الخلوية لتخفيض استهلاك الطاقة في شبكة الشريحة حيث أن كل قاعدة من قواعد الآلات الخلوية تمثل كروموزوم الذي يسمح بالبرمجة التلقائية للقواعد الجديدة و في صدد ذلك بيّنا أن الآلات الخلوية التطورية الكمية هي تقنية واعدة جدا بتحفيز اكتشاف القواعد الفعالة التي تقود لأحسن تابع أو دالة التقييم

الكلمات المفتاحية :
استهلاك الطاقة، وتطبيقات الوسائط المتعددة، الإدماج ، شبكات الشرائح ، الآلات الخلوية ، الكم والخوارزمية الجينية

Abstract

Today with the advent of new very-large-scale integration processing technologies, network on chip with its high performance and scalable alternative to the system on chip architecture, is considered as a promising solution to achieve faster time to market, reduced costs and flexible solutions. However, present day embedded multimedia applications are becoming more computational intensive due to the large number of integrated functions. Often, such applications are mapped onto mobile systems that need to operate with low energy consumption, but the focus is not on the implementation of one single application on the many-core platforms, but using a single-die that supports multi-application scenarios. Due to that the mapping strategies that map the application tasks on the chip should minimize the energy consumption of multiple parallel applications under timing constraints.

Indeed, genetic algorithms are widely used to solve problems in complex system for its easiest implementation and achievability of global optimum with a proper approximate solution, but quantum-inspired evolutionary algorithms have been surpassing the classical algorithms for their abilities to solve problems of polynomial-time that is considered it as impossible to be solved but with million years while their advantage is to balance between exploration and exploitation of the solution space and also obtain better solutions, even with a small population.

On the other hand, cellular automata are dynamic systems where one of their main characteristics is the transition rules that are represented by a very simple structure but the results of its overall behavior can be very complex. Cellular automata simplicity and rigorous mathematical model are the reason for being strongly favorites to be used in complex system, especially in regular network on chips, where a large number of processors are embedded onto a single die and should support high throughput, low latency and data communication under timing constraints. For that, in our thesis we study how quantum genetic algorithms can evolve cellular automata to optimize the dynamic energy consumption in regular network on chip at a system level where each transition rule represents a chromosome that is allowing an automatic

programming of the new transition rules. We show that quantum evolutionary cellular automata algorithm is a very promising technique that is stimulating the process of discovering the effective rules that leads to better fitness functions. In our work, we optimize the dynamic energy consumption in regular network on chip whereas the estimation of the energy consumption takes into account both dynamic and static.

Keywords:

Energy consumption, multimedia application, network on chip, cellular automata, heuristics, mapping, quantum genetic algorithm.

Résumé

Aujourd'hui, avec l'avènement de nouvelles technologies d'intégration à très grande échelle, les réseaux sur puce sont considérés comme une solution prometteuse, offrant des coûts réduits et une haute flexibilité et performance par rapport aux systèmes sur puce à bus. Cependant, les applications multimédia actuelles intégrées sont de plus en plus de calcul intensif en raison du grand nombre de fonctions intégrées. Ces applications sont souvent mappés sur des systèmes mobiles qui doivent fonctionner à faible consommation d'énergie, mais l'accent est non pas mis sur la mise en œuvre d'une seule application sur des plates-formes multi-cœurs, mais en utilisant une seule puce qui prend en charge des scénarios multi-applications ainsi que les stratégies du mapping projetant les tâches d'application sur la puce devraient réduire au maximum la consommation d'énergie sous contraintes temporelles.

En effet, les algorithmes génétiques sont largement utilisés pour résoudre des problèmes complexes pour leurs mises en œuvre faciles et la réalisabilité de l'optimum global avec une solution approchée correcte, mais les algorithmes évolutionnaires inspirés de la quantique ont surpassés les algorithmes évolutionnaires classiques pour leurs capacités de résoudre des problèmes complexes dans un temps polynomial est d'équilibrer entre l'exploration et l'exploitation de l'espace de solution et obtenir également de meilleures solutions, même avec une petite population.

De l'autre coté, les automates cellulaires sont des systèmes dynamiques où l'une de leurs principales caractéristiques est la règle de transition qui est représentée par une structure très simple, mais le résultat de leurs comportements globaux peut être très complexe. La simplicité des automates cellulaires et le modèle mathématique rigoureux sur lequel sont fondés sont les raisons d'être fortement favoris pour être utilisé dans les systèmes complexes et en particulier dans les réseaux sur puce, où un grand nombre de processeurs sont intégrés sur une seule puce qui devrait supporter un débit élevé, une faible latence et une communication de données sous contraintes temporelles et énergétiques. Pour cela, dans le contexte de notre thèse, nous étudions

comment les algorithmes génétiques quantiques peuvent évoluer les automates cellulaires pour optimiser la consommation d'énergie pour les applications multimédia sur réseau sur puce à topologie Mesh 2D où chaque règle de transition représente un chromosome qui permet une programmation automatique des nouvelles règles de transition. Nous montrons que les automates cellulaires évolutionnaires quantiques est une technique favorable qui stimule le processus de découverte de règles efficaces qui conduisent à une meilleure fonction de fitness. Dans notre travail, nous nous somme intéressés par l'optimisation de la consommation d'énergie dynamique bien que l'estimation prenne en compte la consommation d'énergies dynamique et statique.

Mot clés :

Consommation énergétique, applications multimédia, Réseaux sur puce (NoC), automate cellulaire, heuristique, mapping, algorithmes génétiques quantiques.

Notation and abbreviations

Notation	*Details*
ITRS	International Technology Roadmap for Semiconductors
CA	cellular automaton
QECA	quantum evolutionary cellular automata
MMS	multimedia system
auto-indust	automotive industrial
NoC	Network on Chip
SoC	System on Chip
IC	integrated circuit
IP	intellectual property
DSM	deep submicron-meter
NI	NETWORK INTERFACE
VCs	virtual channels
CDG	channel dependency graph
MIT	Massachusetts Institute of Technology
MIPS	Microprocessor Without Interlocked Pipeline Stages
BE	best-effort
GT	guaranteed throughput
TDN	Temporally Disjoint Networks
SPIN	Scalable Programmable Integrated Network
CHAIN	CHip Area Interconnect
MANGO	Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces
GS	guaranteed services
MoC	Models of Computation
KPN	Kahn process networks
SDF	Synchronous Data Flow
J	joule
mks	meter-kg-second
PTM	Predictive Technology Model
RTL	Register-transfer level
HLS	High-level synthesis
MCBCG	Model Checking Based Sequential Clock Gating
PMU	Power Management Unit

Notation	*Details*
CMP	Chip Multi-processor
DVFS	Dynamic Voltage and Frequency Scaling
TFETs	Tunnel Field Effect Transistors
PNC	power-efficient network calculus-based
ABB	adaptive body biasing
SCC	Single-Chip Cloud Computer
VFI	Voltage/Frequency Islands
OP	operating point
VHSIC	Hardware Description Language
TLM	transaction-level modeling
TG	task graphs
CTG	communication task graphs
CWG	communication weight graphs
CRG	communication resources graph
ATG	annotated task graphs
SDFG /ADFG	synchronous and asynchronous data flow graphs
PN	Petri Networks
KPN	Kahn Process Networks
DOL	Distributed Operation Layer
MILP	mixed-integer linear programming
PE	processing element
VNS	Variable neighborhood search
EAs	Evolutionary algorithms
LISP	Locator/identifier separation protocol
C-NOT	Controlled-NOT gate
Qubit	Quantum bit
CI	initial configuration
NP	Nondeterministic polynomial time (Class of computational problems for which a given solution can be verified as a solution in polynomial time by a deterministic Turing machine.
NP-hard	Class of problems which are at least as hard as the hardest problems in NP. Problems that are NP-hard do not have to be elements of NP; indeed, they may not even be decidable.
DAGs	directed acyclic graphs
CTG	communication task graphs
CAN	controller area network
PWM	pulse width modulation
IIR	infinite impulse response
iDCT	filter and an inverse discrete cosine transform
FIR	finite impulse response
FFT	fast fourier transform
iFFT	inverse fast fourier transform
APCGs	Application Characterization Graphs
ID	destination identity
RRA based deadlock	based deadlock factor randomized routing algorithm
QGA	Quantum genetic algorithm

Contents

Abstract....i
Anotation and abbreviations vi
General introduction....1
1. Problem definition....2
2. Contributions 5
3. Thesis overview....6

Chapter I Network on chip....7

1. Introduction....7
2. Embedded system 8
2.1. The concept of the system 8
2.2 Embedded system definition 8
2.3 Embbeded system types....8
2.3.1 General computing 9
2.3.2 Control system....9
2.3.3 Signal processing....9
2.3.4 Communication and networking....9
2.4 Main characteristic....9
2.4.1 Dedicated functions....9
2.4.2 Real-time operation 10
2.4.3 Monolithic developments.... 10
2.4.4 Real-time operting system.... 10
2.4.5 Reactive operation.... 10
2.4.6 Configurability.... 10
2.4.7 Streamlined protection mechanisms 10
2.4.8 Direct use of interrupts 10
2.5 Real-time system.... 11
2.5.1 Real-time scheduling techniques.... 11
a. Rate monotonic (RM).... 12
b. Deadlock monotonic (DM) 12
2.6 Embedded system different architectures.... 12
2.6.1 System on chip 12
a. System on chip interconnections.... 13
a.a. Point to point.... 13
a.b. Standard bus.... 13
a.c. The hierarchical bus.... 14

a.d. Crossbar 16
2.6.2 Network on chip (NoC) 16
2.6.3 Comparaison of embedded system architectures 17

3. NoC main components 18

3.1 NoC layered communication 18
3.2 NoC main components 19
3.2.1 Node 20
3.2.2 The network interface (NI) 20
3.2.3 The routers 20
3.2.4 The links 20
3.3 NoC main characteristic 21
3.3.1 Topology 21
a. Spin 21
b. Cliché 22
c. Torus 23
d. Octagon 24
e. BFT 25
3.3.2 Switching perotocol 26
a. Circuit switching 26
b. Packet switching 27
b.a. Store and forward (SAF) 27
b.b. Virtual cut-through (VCT) switching 28
b.c. Wormhole switching 29
c. Comparaison of switching techniques 30
c.a. Buffering and virtual channels 30
c.b. Comparaison of different switching technique 31
3.3.3 Routing protocol 32
a. Deterministic routing algorithms 33
b. Adaptive routing algorithms 34
c. Minimal adaptive routing 35
d. Non-minimal adaptive routing 36
e. Turn model 37
f. Randomized routing algorithms 38
g. Comparaison of routing algorithms 40

4. NoC simulator 40

4.1 NS-2 40
4.2 NOXIM 41
4.3 DARSIM 41
4.4 SUNFLOOR – 3D SUNFLOOR 41
4.5 ORION 2.0 41

5. NoC examples 42

5.1 Æthereal 42
5.2 Nostrum 42
5.3 Spin 43
5.4 Chain 43
5.5 Mango 43

6. Conclusion 44

Chapter II Model of computation in multimedia applications and NoC energy efficiency and delay model 45

1. Introduction 45

2. Models of computation 46

2.1. Process-oriented MoC 46
2.1.1 Kahn Process networks 47
2.1.2 Process calculi 48
2.1.3 Synchronous data flow 49
a. Definition 49
b. Formal model 49
c. Scheduling policies over SDF 50

3. Network on chip delay model 51

3.1. NoC latency 52
3.2. Computation delay 53
3.3. Communication delay 53
3.4. The average number of packets in the system 54
3.5. Network on chip average throughput 54

4. Network on chip power model 54

4.1. Link power model 56
4.2. Router power model 57

5. Power reduction techniques 58

5.1. Techniques for reducing dynamic power 58
5.1.1 Gate sizing 58
5.1.2 Clock gating 59
5.1.3 Sequentiel clock-gating 59
5.1.4 System level simulation guidedhls approach to generate clock-gated RTL 60
5.1.5 Voltage and frequency scaling 60
a. Design-time voltage and frequency scaling 61
b. Static voltage and frequency scaling 62
c. Dynamic voltage and frequency scaling 62
5.2. Techniques for reducng short circuit power 64
5.3. Techniques for reducing leakage power 64
5.3.1 Multiple supply voltage 64
5.3.2 Multiple threshold voltage 65
5.3.3 Adaptive body biasing 65
5.3.4 Power gating 66
5.3.5 Segment gating 66
5.4. Comparison between different power reduction techniques 67

6. Conclusion 67

Chapter III Heuristic optimization algorithms based network on chip 68

1. Introduction 68

2. Problems and proposed contributions 69

2.1. Routing and switching strategy 69
2.2. Task migration 71
2.2.1 Definition of task migration 71
2.2.2 Reasons for task migration 71
2.3. Dynamic voltage/frequency scaling 72

3. Heuristic optimization methods for nocs ... 73

3.1. Preliminaries ... 73
3.1.1 Objective function ... 74
3.1.2 Search space ... 74
3.1.3 Types of solutions ... 74
3.1.4 Constraints ... 74
3.1.5 Heuristics ... 74
3.2. Characteristics of heuristic optimization methods ... 75
3.2.1 Generation of new solutions ... 75
3.2.2 Treatment of new solutions ... 75
3.2.3 Limitation of the search space ... 75
3.2.4 Prior knowledge ... 75
3.2.5 Computational complexity ... 76

4. Mapping and scheduling techniques in noc ... 76

4.1. Key factors on task mapping ... 76
4.1.1 Target architecture ... 76
4.1.2 Abstraction level of the application specification ... 76
4.1.3 Figures of merit ... 76
4.1.4 Common domain semantic ... 77
4.1.5 Topologies ... 77
4.1.6 Optimization algorithms ... 77
4.2. Related work of mapping and scheduling techniques in NoC ... 77
4.2.1 Design-time mapping ... 78
4.2.2 Run-time mapping ... 82
4.2.3 Upcoming trends and open challenges in hybrid mapping ... 85

5. Classification of heuristic optimization methods ... 86

5.1. Trajectory methods ... 86
5.1.1 Tabu search ... 86
5.1.2 Variable neighbourhood search ... 86
5.2. Population based metaheuristics ... 87
5.2.1 Genetic algorithms ... 87
5.2.1.1 History ... 87
5.2.1.2 Definition ... 88
5.2.1.3 Working principle of genetic algorithms ... 90
a. Encoding technique in genetic algorithms ... 91
a.a. Binary encoding ... 91
a.b. Permutation encoding ... 92
a.c. Value encoding ... 92
a.d. Tree encoding ... 92
b. Selection techniques in genetic algorithms (GAS) ... 93
b.a. Roulette wheel selection ... 93
b.b. Rank selection method ... 94
b.c. Steady-state selection ... 95
5.2.1.4 Genetic algorithms operators ... 95
a. Crossover ... 95
a.a. Binary encoding crossover ... 95
a.b. Uniform crossover ... 96
a.c. Arithmetic crossover ... 96
a.d. Permutation encoding crossover ... 96
a.e. Value encoding crossover ... 97
a.f. Tree encoding crossover ... 97
b. Mutation ... 98

b.a. Binary encoding mutation 98
b.b Permutation encoding mutation 98
b.c. Value encoding mutation 99
b.d. Tree encoding mutation 99
5.2.1.5 Genetic algorithms issues 100

6. Cellular automata 101

6.1. History 101
6.2. Characteristics of cellular automata 102
6.3. Formal definition of cellular automata 102

7. Quantum computing 104

7.1. Prerequisites 104
7.1.1 Dirac's notation 104
7.1.2 Qubit 105
7.1.3 Quantum register 105
7.1.4 Quantum logic gates 105

8. Conclusion 107

Chapter IV Contributions, tests and experimentals 108

1. Introduction 108

2. Contribution in scheduling strategy and application model 108

2.1. Characteristics of scheduling policies 108
2.1.1 Target architecture 109
2.1.2 Application 109
2.1.3 Task model 110
2.1.4 Benchmarks' multimedia applications (case of study) 110
a. Automotive /industrial application 110
b. Multimedia system 112
2.2. The task workload 113
2.3. The task deadline 114
2.4. The first one reached its deadline 115
2.5. The minimum slack 116
2.6. The average response time 116

3. Contribution in routing and switching strategy 117

3.1. Bufferless wormhole routing 118
3.2. Based deadlock factor randomized routing algorithms 120

4. Contribution in task migration 121

4.1. CPU state 122
4.2. Task stack 123
4.3. Heap data 123
4.4. Program code 123
4.5. Task associations 123
4.6. Global variables 123

5. Contribution in dynamic voltage/frequency scaling 124

6. Quantum evolutionary cellular automata in network on chip 125

6.1. Cellular automata principles 126

6.2. Problem description 127
6.3. Solution description 129
6.4. Quantum evolutionary cellular automata lattice format 132
6.5. QECA concepts 133

7. Experiments and case of studies 140

7.1. Quantum evolutionary cellular automata results 141
7.2. Bufferless wormhole routing results 148
7.3. Randomized routing algorithm results 150
7.4. Dynamic voltage and frequency scaling results 156

8. Conclusion 157

General Conclusion 158

1. Conclusion 158

2. Perspectives 159

References 160

Annex A : Conferences And Journals Papers 167

Annex B : Platform description and some codes 169

List of figures

1. The essence of the embedded system 3
2. 2001, 2002 itrs prediction of static and dynamic power consumption trends..... 3
1.1. The embedded system diagram 8
1.2. Soft versus hard real-time temporal behavioral 11
1.3. Point to point connection 13
1.4. Bus topology 14
1.5. Hierarchical topology.....15
1.6. 2×3 mesh topologys 17
1.7. Noc communication protocol stacks. 18
1.8. Noc- based system 19
1.9. Switch architecture 20
1.10. Spin architecture 22
1.11. Cliche 23
1.12 Torus 24
1.13. 2d folded torus 24
1.14. Octagon 25
1.15. Bft architecture 26
1.16. Circuit switching technique 27
1.17. Store and forward switching technique 28
1.18. Virtual cut through switching technique 29
1.19. Wormhole switching technique 30
1.20. Double y-channel 2D mesh 35
1.21. +x sub-network and labeling 36
1.21. Six turns (solid arrows) allowed in west-first routing 37
1.23. A possible path from src to dst using 3-phase romm on a 2D mesh 39
2.1. Kahn process network example 48
2.2 Synchronous dataflow graph example 49
3.1. Example1: load representation on two CPU cores 71
3.2. Example2: load representation on two CPU cores 72
3.3. Taxonomy of mapping methodologies.....77
3.4. Block diagram representation of genetic algorithms 91
3.5. Binary encoding 92

3.6. Permutation encoding 92
3.7. Value encoding 92
3.8. Tree encoding 93
3.9. Roulette wheel method 94
3.10. Rank selection methods 94
3.11. Two point crossover 95
3.12. Uniform crossover 96
3.13. Arithmetic crossover 96
3.14. Permutation encoding crossover 97
3.15. Value encoding crossover 97
3.16. Tree encoding crossover 97
3.17. Binary encoding mutation 98
3.18. Permutation encoding mutation. 99
3.19. Value encoding mutation 99
3.20. Tree encoding mutation 99
3.21. The first row represents the current state of the cell, as well as that of its nearest neighbors on either side. this is the input to the transition function. the possible states are either white or black. the second row represents the output of the transition function, i.e., the state of the cell after applying the transition function. reading the diagram from the left, if the cell is in the state black and both neighbors are black as well, then the cell will be colored in white in the next time step. and if the right neighbor is white, then the cell will remain black, and so forth.....103
3.22. From top to bottom, left to right, the Figure s represent the evolution in time of the ca described in Figure .1. the initial configuration, shown in the top left image, is a single cell colored black, while all others are white. time flows downwards. each subsequent image depicts the current state of the automaton, and all stages since the initial one 104
4.1. Automotive/industrial task graph. 111
4.2. Multimedia system task graph. 112
4.3. Implementation of register based fifo buffers. 118
4.4. Wormhole flits format. (a) head-flit format, (b) body-flit format, (c) tail-flit format 119
4.5. The id bits of head flit. 119
4.6. Next-hop bits of head flit. 120
4.7 The combination of neighbours in ca based on NOC neighbours 127
4.8 Example1 of GA issues. 128
4.9 Example2 of GA issues. 129
4.10. The different mapping used in network on chip 2x2 mesh. (a) an example of the per-stage mapping (b) an example of the channel mapping 132
4.11. Cellular automata lattice format 133
4.12. Quantum genetic algorithm structure 134
4.13. Classical genetic algorithm structure 134
4.14. Example of an application graph mapped onto 2x2 mesh based noc, where the

mapping solutions represent the chromosome structure 135
4.15. Structure of the classical chromosome 136
4.16. Structure of the quantum chromosome 136
4.17. Measured chromosome 137
4.18. Polar plot of the rotation gate for q-bit individuals.....139
4.19. The average fitness value of each generation using GA in MMS 142
4.20. The average fitness value of each generation using ga in auto-indust.....143
4.21. The average fitness value of each generation using eca in MMS 143
4.22. The average fitness value of each generation using ECA in auto-indust. 144
4.23. The average fitness value of each generation using QECA in mms 144
4.24. The average fitness value of each generation using QECA in auto-indust 145
4.25. The deviation of each generation using MMS.....146
4.26. The deviation of each generation using ECA in auto-indust.....146
4.27. The deviation of each generation using QECA in MMS 147
4.28. The deviation of each generation using QECA in Auto-indust 147
4.29. Comparison of bufferless routing with registers versus bufferless routing 149
4.30. Effect of packet size on packet delay using per- stage mapping without channel mapping and per stage mapping with channel mapping and a random mapping.....150
4.31. Packet size on average communication using per-stage mapping without channel mapping and per-stage mapping with channel mapping and a random mapping effect 151
4.32. Effect of packet size on average network latency using per stage mapping without channel mapping and per stage mapping with channel mapping and a random mapping 151
4.33. Effect of packet size on average throughput using per stage mapping without channel mapping and per stage mapping with channel mapping and a random mapping 152
4.34. Packet injection rate versus average packet delay 152
4.35. Packet injection rate versus throughput 153
4.36. Average flit in the system at mean arrival rate = 0.5 and bandwidth = 50.....153
4.37. Buffer occupancy by the flit with mean arrival rate = 0.5 and bandwidth = 50.....154
4.38. Average mean service rate at mean arrival rate = 0.5 and bandwidth = 50.....154
4.39. Average packet in the system at mean arrival rate = 0.5 and bandwidth = 50.....155
4.40. Average packet in the system at mean arrival rate = 0.5 and bandwidth = 50.....155
4.41. Comparison of three mapping strategy based on the average network latency in the system at mean arrival rate = 0.5 156

List of figures for Annex B

B.1 The state of the buffer in one moment in time 175
B.2 The database of chromosomes in decimal format and in a binary format 176
B.3 The results of per_stage mapping of rule number 3 180
B.4 Code of some of the formula of Calcul_timer 180
B.5 Interface 1 of the application. 185
B.6 Example of how token are transmitted from task to another, where 1 means that the task proceed one token and 0 means that the task has no token in buffr in. 185
B.7 The set of paths took for each message sending while the first done is for the first path token using RRA while the second is while using the both bufferless wormhole and based deadlock_factor_RRA. 186
B.8 Interface 2 of the application. 186
B.9 Interface 3 of the application 187

List of tables

1.1. Comparison of embedded system architectures…….....18
1.2. Comparison of switching techniques in network on chip…….....32
2.1. The parameters notation…… 52
2.2. Parameters under 0.18um technology node…… 56
2.3. Comparison between different power reduction techniques.....67
4.1. Neighborhood look-up table 126
4.2. Look-up for quantum gates rotation. 139
4.3. Comparison between qeca, eca and ga in executing mms2 and auto-indust.....148
4.1. Results of executing mms2 and auto-indust with/without dynamic voltage and frequency scaling.....156

List of tables of Annex B

B.1 Code of sender_timer (Channel_timer). 170
B.2 Code of wormhole switching. 173
B.3 Code of buffer_timer. 174
B.4 Code of write action on the NoC buffer. 175
B.5 Code of read action on the NoC buffer. 177
B.6 Code of task_timer. 177
B.7 Code of task migration. 178
B.8 Code of dynamic voltage scaling. 178
B.9 Code of processor_allocation_timer using the scheduling strategy based workload. 179
B.10 Code of processor_allocation_timer using the scheduling strategy based minimum slack. 179
B.11 Code of CA_timer. 182
B.12 Code of GA_timer. 183
B.13 Code of calculating the switching factor. 184
B.14 Code of storing the best value in B(t). 184

GENERAL INTRODUCTION

1. Thesis context

An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today, and many architectures have been proposed such as System on Chip (SoC) and Network on Chip (NoC).

SoC designs provide the designers to integrate numbers of Intellectual Properties (IP) blocks together. But with the rapid advance of semiconductor technology, the complexity of such SoC increases as a result. By the end of this decade, a single chip will accommodate up to 1-billion transistors. Thus, the number of IPs in SoC designs can scale from a few dozens to several hundreds.

One of the challenges in the billion-transistor era is the communication infrastructure between heterogeneous core having different characteristics. The interconnection in current SoC based embedded system for connecting IPs is typically dedicated wires or shared buses. The dedicated wiring approach provides the best communication performance, but it design has poor reusability and scalability. Furthermore, the wire latency significantly affects the reliability of the systems when the system complexity increases and the feature size decreases. The shared

bus architecture provides a pool of bandwidth among all the cores in the system. However the shared bus limits the growth of the system complexity. To deal with the interconnect problems in SoC chips, a new technology has been developed. NoC with regular tile-based architectures has been proposed.

NoC technology is often called "a front-end solution to a back-end problem." As semiconductor transistor dimensions shrink and increasing amounts of IP block functions are added to a chip, the physical infrastructure that carries data on the chip and guarantees quality of service begins to crumble [183].

2. Problem definition

In modern application embedded systems, the performance constraints need to be satisfied according to the number of computational elements supported by the system. Old buses are no more a solution for streaming multimedia applications. The network on chip is the high performance and scalable alternative to the system on chip architecture. However, such systems are usually operated by self-source power like batteries, so to increase the operating time, a minimization of the energy consumption during their design is required.

Multimedia applications are soft real-time systems, where the application task graph is defined by its period and each task is defined by a deadline, to meet the timing constraints, the period of those systems mustn't be exceeded while the task's deadline can be missed with a given threshold in such a way that the quality of streaming multimedia are not degraded, and due to that, a scheduling algorithm is needed. Modern chips have several cores embedded on them and designed to achieve optimal power-performance. Each task running on these cores varies with performances such as energy consumption and latency, which are required to complete the execution of an assigned task. Two components constitute a network on chip power dissipation; dynamic switching power and static or the leakage power.

The following scheme represents the relative delay for local and global wires and for logic gates in the technologies of the near future as it is predicted by the International Technology Roadmap for Semiconductors (ITRS), that indicates the rapid increase in the interconnect performance.

In Figure 2, it has been seen that at processing technologies until 90 nm, the dynamic power is the major power consuming component, but at technologies below 90nm, this leakage power begins to build up, due to the fact that at lower technologies the supply voltage and thereby the threshold voltage is also reduced consequently. The increasing of the number of cores on a single chip leads to the issue of energy consumption, whereas the change of the

workload conditions can cause bungling of the power dissipation, resulting in the temperature hotspots. The hotspots can in turn degrade the performance and the lifetime of the chip.

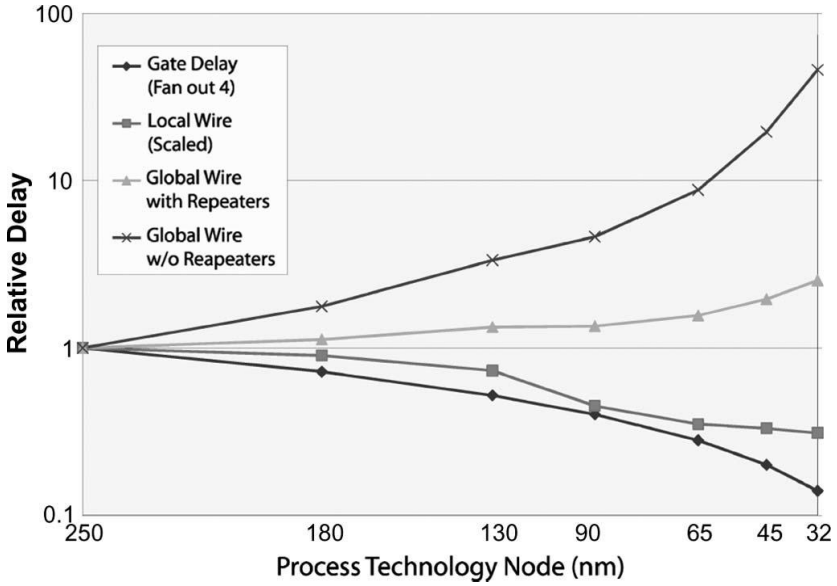


Figure 1. The essence of the embedded system.[49]

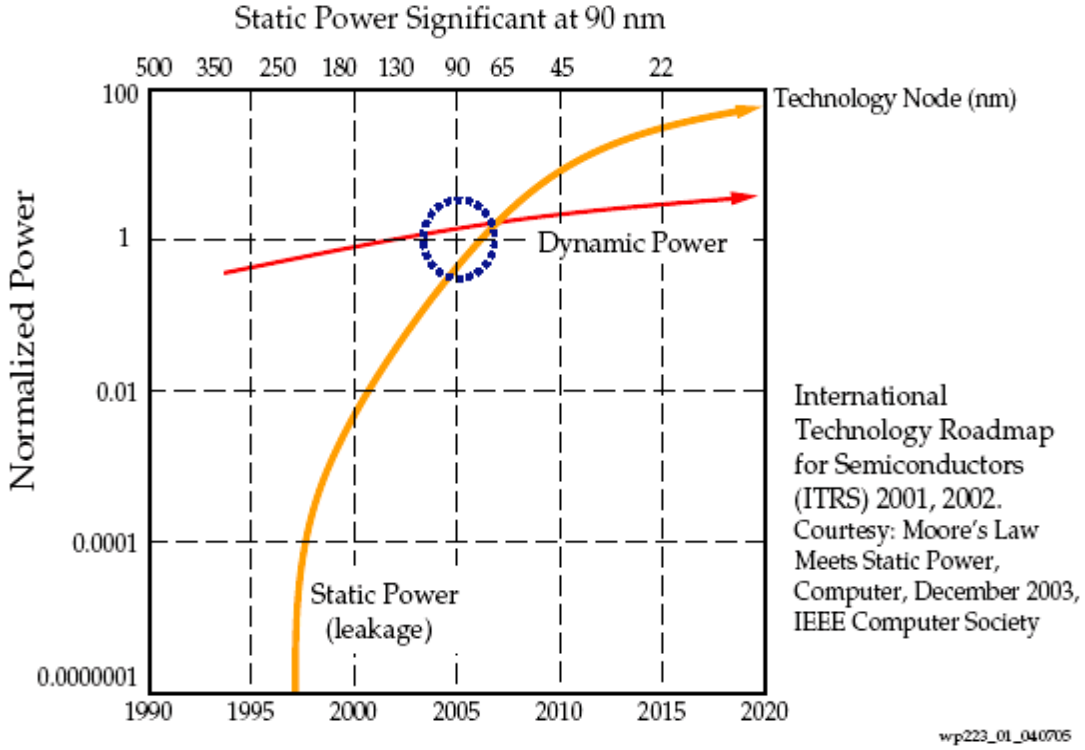


Figure 2. 2001, 2002 ITRS prediction of static and dynamic power dissipation trends.[49]

Therefore, the power management is a key issue towards the optimization of the performances in NoC. Many researches have been done in order to minimize the energy consumption at the system level of the self battery-operated devices such as efficient scheduling and dynamic voltage frequency scaling. Also the application mapping that determines the

assigning of multimedia application tasks to the network on chip tiles in such a way that the energy consumption is optimized is considered as non-deterministic polynomial-time hard (NP-hard) problem where the search space of the problem increases factorially with the system size and due to that an effective optimization algorithm is required. The choice of the most appropriate algorithm is a critical issue because an optimal mapping may enhance the network on chip performance up to 60%. For that many heuristic algorithms are designed to find a near optimal mapping such as list scheduling, clustering scheduling algorithms, or critical path-based heuristics were developed, but those algorithms are non- scalable and deterministic and that what make them not privilege to improve the system performance. Due to that stochastic global search technique that produce high quality solution are needed, for that the based bio-inspired methods has been used to solve scheduling problems such as genetic algorithms, neural networks, and simulated annealing. However, the use of genetic algorithms improve many advantages such as the easiest implementation and achievability of global optimum with a proper approximate solution, but they still the subject of intensive study to improve their performances for many problems such as using an old instance of populations or the complete population is impossible; general knowledge about previous solutions cannot be used. A new searching process must be started from the beginning by creating an initial random population of potential solutions. Also, another problem of using genetic algorithms is related to crossover and mutation where the chromosome represents one of the mapping solutions so the number of tasks and network on chip dimension are demanded before starting because their sizes represent the chromosome size so adding or removing tasks leads to chromosome change, and that will affect the results obtained from genetic algorithms operators (crossover and mutation). So instead of re-mapping many times, a fixed chromosome size that will be suitable for any application is needed. Also, after many crossovers and mutations, the new individuals will be adjusted to be suitable and accepted as a new mapping solution. And also the classical chromosomes are weak in representing the population diversity. From a side, quantum computing has been surpassing classical computers for its ability to solve problems of polynomial-time that classical computers considered it as impossible to be solved unless with million years. Recently, the growing theoretical and practical interest is devoted to research on quantum computing and quantum computers and due to that many quantum computing algorithms have been developed, such as Shors factorizing algorithm, were explored, quantum search algorithm and quantum genetic algorithm. And from another side, a cellular automaton (CA) presents a highly parallel and distributed system of locally interacting units which are able to produce a global behavior. CA can be considered as a model of naturally existing systems

produced by natural evolution. Such systems are capable of producing globally coordinated information processing, unguided by any global criterion or central control. Information processing capabilities of such systems are not explicitly represented in their components, but rather in their interconnections. These capabilities are more powerful than the ones done by elementary components or their combinations. For these reasons, CA has been used to model different physical and biological phenomena such as fluid flow, galaxy formation, avalanches, earthquakes, growth of stony corals, and other biological and physical pattern formations and in the recent year, CA have been used as a simulation model to solve the problem of embedded system such as allocation and scheduling of multimedia tasks.

3. Contributions

The main contribution of our thesis is related to the solving of the Multimedia applications mapping; the scheduling and the routing problems on regular network on chip based on quantum evolutionary cellular automata (QECA). Firstly an implementation of the classical genetic algorithm and the evolutionary cellular automata are done under the same assumption in order to make a comparison with the proposed solution that is applied on the mapping, and the scheduling of multimedia application tasks onto NoC tiles, and also, it is applied on the routing of the messages transmitted between those cores. The QECA method is based on the concept and principles of quantum computing, such as quantum bits, quantum gates and superposition of states. Thus, the mechanism of the QECA method can inherently treat the balance between exploration and exploitation where each Q-bit individual can represent and explore all possible states and drive it to exploit a single state. The use of quantum bit representation leads to better population diversity compared with the classical bit representations while the use of quantum gate drive the population towards the best solution.

Also in our research, we introduce a novel algorithm to avoid deadlock and live-lock in order to guarantee a greater throughput in randomized routing algorithm for 2D mesh network on chip. The proposed idea leads us to achieve low communication latency by minimizing the flits wait time in buffers and links. We called this algorithm a based deadlock factor randomized routing algorithm. This algorithm is implemented and the results obtained are compared with the old randomized routing algorithm. We had also proposed an idea to minimize the packet wait time in processor, we used the mapping per stage or in other way the mapping of tasks based on their priorities. This mapping ensures that each task is implemented onto a different processor to eliminate the waiting time. But in the case where there is unmapped task and all processors are allocated by other tasks and those tasks have the same priority, the occupancy of

each processor is computed and the unmapped task is mapped onto the processor that have a minimal occupancy.

We had proposed a new algorithm in dynamic voltage and frequency scaling that defines the voltage change point for the processor based on the allocated task workload. A change point is where the processor can change its voltage level during the allocated task execution time to achieve an end to end deadline and save the power.

4. Thesis overview

The thesis is organized as follows:

In chapter1, we will give an overview of activities in the field. We will first define the embedded system with its different architecture and in order to avoid the wide range of topics relevant to large scale IC design in general, we stated the motivation for network on chip and will give an introduction of the basic concepts.

Chapter 2 will put the light on understanding the concepts of energy-efficient embedded computing. In the first section, we will define the model of computation with many examples used in literature and then we will introduce simple models to estimate the latency of the system and energy consumption of different tasks. This gives the reader an indication of what are the components that consume the energy. Since the energy consumption depends very much on the considered technology, we assume a hypothetical 180 nm CMOS VLSI technology as the reference for the estimation. The last section of this chapter focuses on the system level based power reduction techniques.

In chapter 3, we will present a state of art of the optimization algorithms that are the most used in literature, and especially those based on network on chip.

In chapter 4, we will detail our proposed solution, and taking as case studies two multi-applications: the multimedia system (MMS) and the automotive industrial (auto-indust) on 2D mesh 5x5 network on chip.

We finish this manuscript with a conclusion that is divided into two parts, where the first is a conclusion that summarize the results obtained from discussing the ideas proposed in the thesis, and the second part represents the perspectives which will be dealt as future works.

CHAPTER I

NETWORK ON CHIP

1. Introduction

Network on Chip (NoC) provides an effective, reliable and flexible infrastructure for system modules based on data packet transmission scheme. It has become an effective solution to overcome the problems of global interconnection and communication in complex System on Chip (SoC) designs [1]. For NoC, the energy consumption, the latency and the routing are the important criteria to work on for achieving a better performance of NoC network communication. This chapter is organized in three sections. Section 2 introduces the embedded system concepts with different types and different architecture and the real-time embedded system. Section 3 defines the NoC with its switching techniques and routing algorithms. In section 4, we end the chapter by a conclusion throughout defining the most used developed NoC examples.

2. Embedded System

2.1 The concept of the system

What is a System? A system can be defined as a functionality of a set of programs that have structured behaviours, where the system inputs are analyzed in order to produce the system outputs.

2.2 Embedded system definition

According to Wayne Wolf, "An Embedded System is a computing system other than desktop computers". An embedded system is a combination of computer hardware (HW) with software (SW) that are embedded in order to perform a specific function (not computational) where the computer is a dedicated-purpose computer system designed to control or support the operation of a larger technical system.

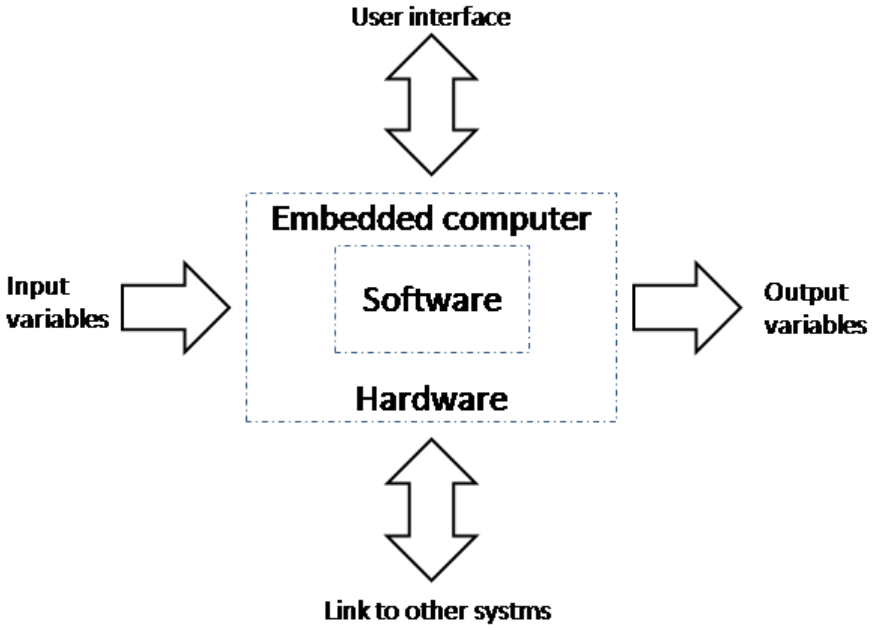


Figure 1.1. The embedded system diagram.

The embedded system was adopted in the industrial environment for its user's expectation satisfaction where many functions was achieved, such as the data compression or decompression in the field of transforming the information also in the field of monitoring and controlling the environments from different actions like processing and generated command by reading and displaying the system inputs.

2.3 Embedded system types

According to Koopman [3], there are 4 types of embedded systems, we distinguish:

2.3.1 General computing

- Applications similar to desktop computing, but in an embedded package.
- Video games, set-top boxes, wearable computers, automatic tellers.

2.3.2 Control system

- Closed-loop feedback control of real-time system.
- Vehicle engines, chemical processed, nuclear power, fight control.

2.3.3 Signal processing

- Computation involving large data streams.
- Radar, sonar, video compression.

2.3.4 Communication and networking

- Switching and information transmission.
- Telephone system, Internet.

2.4 Main characteristic

An embedded system is called independent if the needed materials such as processors, memory, I/O and programs that control the functionalities of those materials during any execution are available, also it should contains a power source. In another way, the embedded system should a self independent. In the following, we introduce many common characteristics that define all Embedded Systems.

2.4.1 Dedicated functions

An Embedded System can execute a dedicated function repeatedly so the use of general-purpose may not be possible, an example of embedded systems that are Microwave oven, Mobile phone, ATMs, Car braking systems, Automobile cruise controllers, Modem, Network cards and many more. Many constraints are related to the applications executed on embedded systems that must be considered in the design phase in order to implement a robust system, those constraints are:

- constraint of memory space
- constraint of power consumption

- real-time requirements
- Manufacturing cost.

2.4.2 Real-time operation

The embedded system computation depends on the timing constrained at which the result is delivered. In the literature, two techniques are used to satisfy those timing constraints:

2.4.3 Monolithic developments

The monolithic techniques are developed in low-level language, they are based on a single program that is responsible for the functionality of the system. The timing constraints are guaranteed via integrated the monolithic development program during the design phase.

2.4.4 Real-time operating system

RTOS provide to developers many techniques used in the real time programming such as: priority scheduling. RTOS allows high level techniques such as rate monotonic (RM), deadlock monotonic (DM).

2.4.5 Reactive operation

In any embedded system, the reactivity should be integrated in such a way that the worst-case conditions are considered during the execution in order to face any external events occurring during non predictable intervals.

2.4.6 Configurability

During execution, some functionality is only needed for that a system should be flexible so a specific application and hardware is provided while selecting only the necessary modules to be loaded.

2.4.7 Streamlined protection mechanisms

It requires limited protection because tested software can be assumed to be reliable

2.4.8 Direct use of interrupts

where:

- Allowing the user process to use interrupts directly.
- No need to go through interrupts service routines.
- Having efficient control over a variety of devices.

2.5 Real-time system

In technical control systems, each application is defined by a set of process or scenarios. According to [11], a process is the totality of activities in a system that influenced by each other and by external factors such as materials, energy, or transforming, transporting or storing the information. The basic element of a process is the task that represents the elementary and atomic entity of parallel execution. The task concept is fundamental for asynchronous programming which is concerned with the execution of a program in a computing system during the lifetime of a process. Control systems should be considered in terms of tasks with their natural properties that is called the real-time behaviour where each one is expressed by its deadline and its period and not by artificially assigned priorities.

Real-time systems are classified into soft real-time and hard real-time, the first allows a deadline miss that lead to a significant loss, whereas the deadline miss in hard real-time is catastrophic (the behaviour is shown in the following figure), for that a system behaviour predictability is needed in both classes. Predictability is often achieved by either static or dynamic scheduling of real-time tasks to achieve end to end deadlines. Static (or off line) scheduling is done at compile time, whereas dynamic scheduling (online) is done at the execution time where the scheduling decision are taken based on many tests done during the execution time of the task to achieve its end to end deadline.

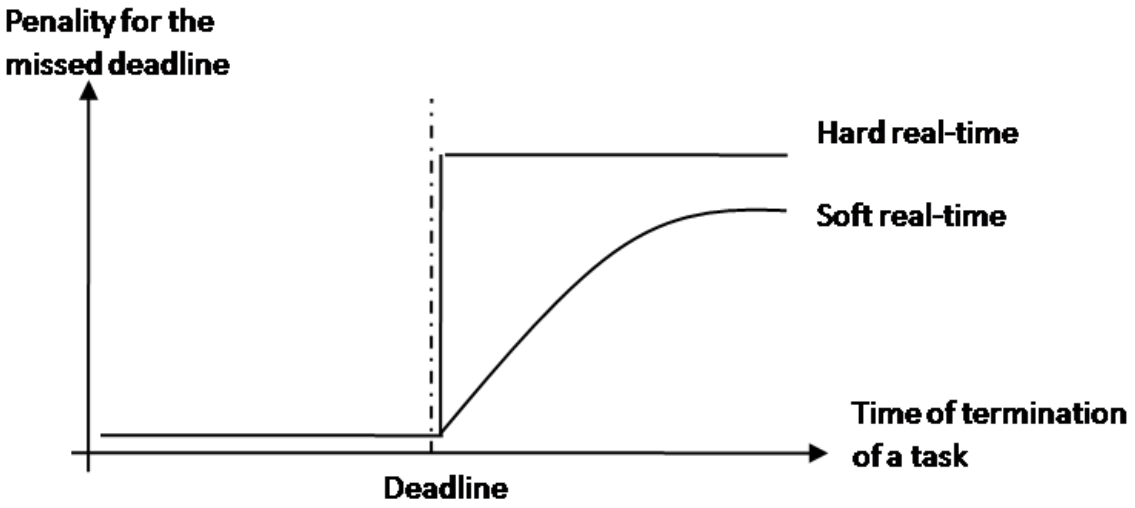


Figure 1.2. Soft versus hard real-time temporal behavioral.

2.5.1 Real-time scheduling techniques

The scheduling process is what determines the order of the execution of tasks; as we mentioned before that they can be classified into static-priority scheduling and dynamic-priority

scheduling, in the following we define few popular algorithms for scheduling processes with real-time constraints.

a. Rate monotonic (RM)

RM is an optimal static-priority scheduling technique where the priorities are assigned according to the defined period for that a task with a shorter period has a higher priority and the execution order start from the task with the shortest period. Real-time system is schedulable under RM if and only if:

$$\sum U_i \leq n\left(2^{1/n} - 1\right) \quad (1)$$

Ui is the processor occupancy by the task i and n in the number of tasks.

b. Deadlock monotonic (DM)

DM is an optimal dynamic priority scheduling technique that assigns to each task a priority based on the shorter task. During execution the scheduling order privilege the task with the earliest deadline. Real-time system is schedulable under EDF if and only if

$$\sum U_i \leq 1 \quad (2)$$

Where U_i is the processor occupancy by the task i

However, the rate monotonic scheduling technique has a simpler implementation, even on systems without explicit support for timing constraints (periods, deadlines), and offers a best predictability for the highest priority tasks whereas earliest deadline first is a full processor utilization with misbehaviour during overload conditions.

2.6 Embedded system different architectures

2.6.1 System on chip

System on Chip (SoC) [4] [5] is an integrated circuit (IC) that integrates all components of a computer or other electronic system into a single chip. It is a collection of all components and subcomponents of a system onto a single chip. SoC design allows high performance, good process technology, miniaturization, efficient battery life time and cost sensitivities. This revolution in design had been used by many designers of complex chips, as the performance, power consumption, cost, and size advantages of using the highest level of integration made available have proven to be extremely important for many designs.

SoC designed with new strategies to reuse the existing implemented productivities and it is named intellectual property (IP cores). Thus, the reuse of the IP cores with HW/SW co-design strategies are adopted as a technique to reduce the productivity gap, the design costs and the time-to-market.

a. System on chip interconnections

To satisfy the constraints time to market that is increase of the term of exigencies, different interconnections for SoC exist, in the following we illustrate some of them:

a. a Point to point

Point to point is the simplest; all to do is to connect two intellectual properties (IPs), so those functional blocks are linked together without any communication protocol. The interconnection requires that the IP blocs should have the same encapsulation format such as input/output interface.

This implementation enables a speed interconnection and fast integration. But the standard encapsulation limits the IP communications in case when indirect IPs need to communicate with private messages. Also, if one IP is blocked, the whole channel will be blocked.

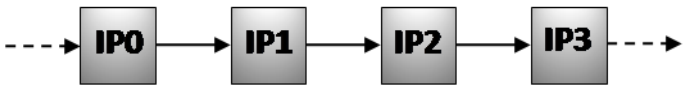


Figure 1.3. Point to point connection.

a. b Standard bus

Most SoC uses bus architecture or as known the shared media because it is the simplest in term of structure, but technology allows to connect all the IP blocs into a single communication media called bus; this architecture is based on the encapsulation of all the same functional bloc with a flexible communication protocol (not as point to point architecture) to integrate another norm or applications to SoC with a fast way, it just needs to add another IPs blocs with bus extending. The communication with this shared media is guaranteed by the bus arbiter, the following figure illustrates an example of a standard bus:

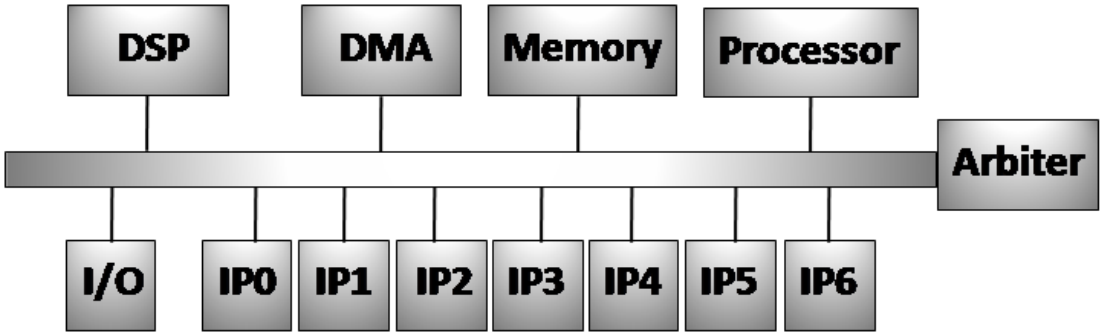


Figure 1.4. Bus topology.

- Here some example of used bus in SoC:
- The "AMBA-AHB" (advance high- performance bus) of ARM society.
- The "Sonic SMART Interconnects" of Sonics society.
- The "PI-BUS" (peripheral Interconnect Byus)o of the European OMI (Open Microprocessor Initiative) constitute by Siemens, Philips, Matra-MHS, ARM and ST-INMOS.
- The "AVALON" ALTERA society.
- The CoreConnect of IBM.

The main disadvantage of this topology standard bus or as know by distributing media, is the sequential communication, in other way the constraint that the bus could not do but one communication at a time. Another disadvantage is bottleneck which is created when the number of communication increased the bus arbiter who is in charge of the communication; but also where the bandwidth constraints of many communication are very important, that the arbiter is predominant because it is who allow the communication inside the bus and also it has the charge to solve the conflicts problem (many requests at the same time). That arbitration limits the number of IP connected to bus to a lot of elements.

For more, this interconnect physical mode became an obstacle in SoC performance for now and for the future, the bus line is becoming longer with time because of the augmenting of IP bloc connected. We can see also with time a parasite that engenders a charge time raise which leads to reduce the frequency of the file may cause a clock late that cause a synchronization problem.

Considering all those problems with SoC needs and evolution all the time, bus topology is no more suitable to be used for integration in silicon.

a. c The hierarchical bus

For the problems mentioned in standard bus, a solution was shown to overdo the limits this topology has to face, and this solution is the hierarchical bus and its architecture is illustrated in the following schemes:

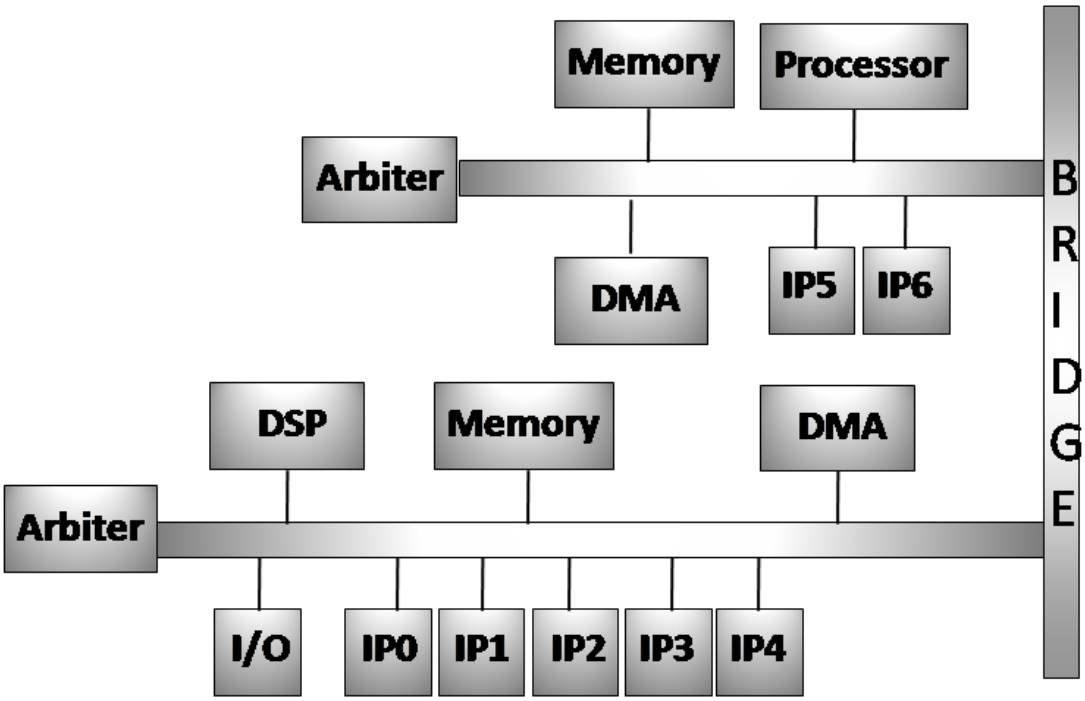


Figure 1.5. Hierarchical topology.

This new topology is based on the communication between many buses with bridges that link two buses with each other the main benefits of the hierarchical bus is the possibility of synchronising different tasks of SoC, by sharing them on buses. According to that, we can see a charge balance in communication the way that not overcharge just one bus allow a synchronization of application tasks.

The different segments of the hierarchical bus have a short length that the IP blocks are attached to; those IP blocks offer a weak capacity on each segment and allow a high frequency on each bus.

As well as, the transaction on those different buses' segments may be in parallel and in high frequency. In that way we offer a high quality of service into an application and a best global bandwidth.

However the access from bus to another implies a latency cost where the importance is to distribute the connected IP bloc and that's for the limit using of the bridge and also for synchronizing the bus tasks. For that, three types of standard bus that are constitute the hierarchical bus:

- Processor / Memory bus: charge in data transfer between processor and cache memory. They are very fast and offer a large bandwidth.
- Input/ Output bus: allows a communication interface.
- Classic bus: the classical bus bases are used here, we find in this bus a processor, a memory and an output/input witch don't need a large bandwidth.

These three categories provide a general sequence of tasks that should in most SoC applications that may be encountered. This constraint requires bandwidth between the application tasks.

Finally, the hierarchical bus consumes less than the distributed bus as the ability of elements connected to the bus is lower on each bus segment. This division into segments is a first step which will tend to network approach, but the bottleneck that can be observed at bridges becomes a limiting factor given the bandwidth needs in the future applications , this why topologies "crossbar " offer an interesting alternative for future SoC.

a. d Crossbar

An alternative offering a good compromise between bus topologies and those networks is the crossbar; in this case, all the functional blocks of the application are linked to each other through the crossbar. This has the advantage of enabling parallel communications (unlike the bus) and offers a large bandwidth to each communication because they are not shared with other communications as the case of the standard bus or hierarchical.

2.6.2 Network on chip (NOC)

The idea was talked about in the 90's, but actual research came in the new millennium. As SoCs grew in quantity of IP cores, bus architectures and crossbars were exposed of their deficiencies. Shared bus architectures led to resource contention and hierarchical bus architectures and crossbar designs generated complexity. Interconnection networks offer an attractive solution to this communication predicament and are becoming persistent in SoC systems. A well-designed interconnection network makes a well-organized use of limited communication means while providing low-latency, high bandwidth communication amongst different IPs with a minimum cost and low energy-dissipation. Undeniably, as system density and integration continued increasing, quite many designers discovered that it is more efficient to route packets, not wires. Utilizing an interconnection network in a SoC than a dedicated wiring permits 6 limited bandwidth to be shared so that it can be used resourcefully. In contrast, dedicated wiring is idle mostly. Using a network also administers regular, organized use of

communication resources, making SoCs easier to design, repair, and optimize. Figure 1.6 displays a 2×3 Mesh topology for NoC.

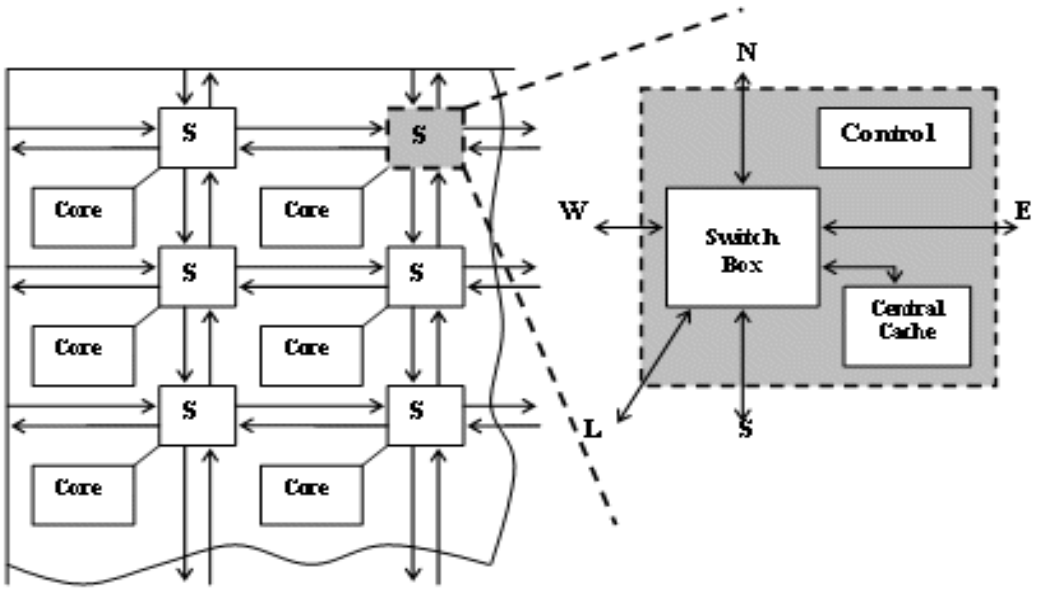


Figure 1.6. 2×3 Mesh topology.

Packet-based Network-on-Chip (NoC) has arisen as a remedy to the SoC design problem from a communication centric viewpoint. Moreover, arranging the interconnect logic uniformly throughout the chip rather than having buses as junction points has greatly eased floor planning of high-density chips.

2.6.3 Comparison of embedded system architectures

Each architecture has many pros as well as many cons, table 1.1 represents a comparison of SoC and NoC architectures adopted from [11]:

Cons	Bus	- Each unit attached adds parasitic capacitance, therefore electrical performance. - Bus timing is difficult in deep submicron-meter (DSM) process. - The bus arbiter is instance specific. - Bus testability is problematic and slow.
	NoC	- Internal network contention may cause large latencies. - Bus-oriented, IPs need smart wrappers. - Software needs clean synchronization in multiprocessor systems. - System designers need re-education for new concepts.

Pros	Bus	- Bandwidth is limited and shared by all units. - Bus latency is wire speed once arbiter has granted the control. - The concepts are simple and well understood by designers.
	NoC	- Only point-to-point one-way wires are used for all network sizes, thus local performance is not degraded when scaling. - Routing decisions are distributed. - The same router may be re-instantiated for all network sizes (communication reusability). - Aggregated bandwidth scales with the network size.

Table 1.1. Comparison of embedded system architectures.

3. NoC main components

3.1 NoC layered communications

Like general communication networks, NoC uses the 7 layers communication. However the architecture can be divided into different layers from bottom to top such as physical data link, network transport and application layer. A layer can be considered as a combination of soft and hard components performing some functionality. Each layer performs its task independent of the other layers and provides services to the upper layer and gets services from the lower layers. The way a layer performs its function is hidden from other layers.

Layers communicate with each other through standard interfaces. Each layer implements a different set of rules called protocols. Advantages of layered communication come at the cost of certain overheads. In NoC, mostly three layers are considered i.e. physical data link and network layer.

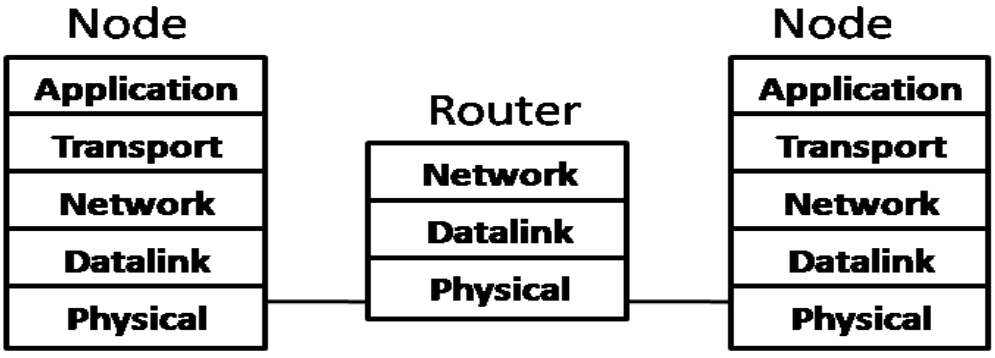


Figure 1.7. NoC communication protocol stacks.

- Physical layer delay with the actual transfer of data. It is responsible for clock signals for every connection, number of wires, control signals, electrical levels, medium of transfer etc.
- Data link layer: is concerned with flit formation, node to node communication, error detection and correction, flow control, encoding scheme, etc.
- Network layer performs packetzation, routing of packets from source to destination, resource addressing, and packet buffering and congestion control. It is also responsible for providing quality of service by addressing the issues of latency, throughput, jitter etc. when data arrives at this layer, if it is reached its final destination; it is formatted into packets and delivered to the transport layer. Otherwise, the data is pushed back to the lower layers to continue its route to the next hop; network layer maintains logical addressed of the nodes in the network.

3.2 NoC main components

In NoC- based system, nodes communicate through the routers, and are connected to it using network interface NI as depicted in figure 1.8. The NoC is itself an interconnection of routers and links; other modules such as arbiters and buffers are included in routers and links.

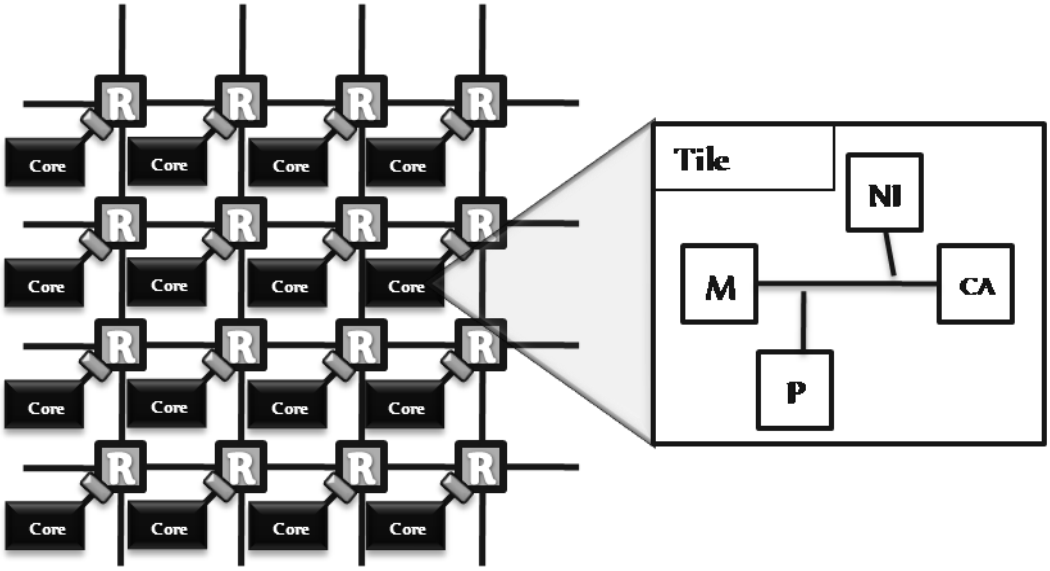


Figure 1.8. NoC- based system.

Each of the NoC based system elements is detailed subsequently:

3.2.1 Node

A node represents an element of the SoC that communicates with other elements; it can be a cluster of nodes when the NoC is used as a global interconnect or a simple element (general-purpose processor, DSP; memory block, peripheral etc).

3.2.2 The network interface (NI)

NIs are the interface between the protocol and the NoC IP blocks that are connected to the router. Their role is to separate the treatment (carried in IP) of the communications (managed by the network) A network adapter can be composed of two parts, ne that supports the network interface itself, the other performing the adaptation protocol witch called "wrapper".

3.2.3 The routers

A router has input and output ports connecting it to another component through links. The main role of the router is to route the incoming packets toward their destination in the topology. Additionally, the router has to perform arbitration among different port request. Therefore, the router is implemented by means of buffers to store the incoming/outgoing data, switches and arbitration logic. More complex routers could also include a network processor.

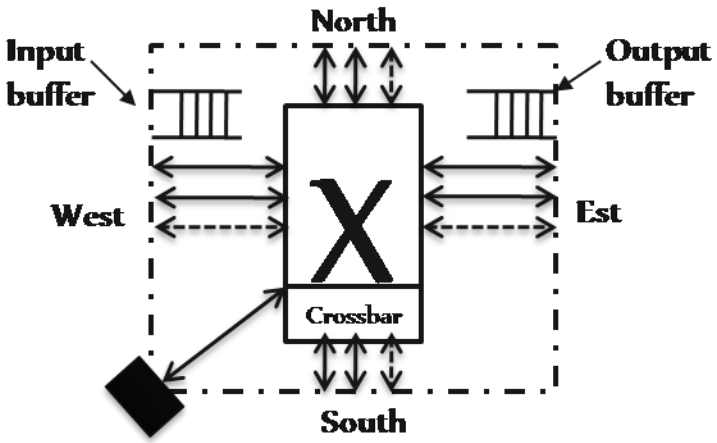


Figure 1.9. Switch architecture.

3.2.4 The links

The links are for goal to link the routers between them-or routers to NI. They offer a bandwidth for communications between the source and destination. It can have multiple virtual channels and can be unidirectional or bidirectional.

3.3 NoC main characteristic

Performance of NoC depends on various factors such as network topology, routing strategy and switching technique. However the performance evaluation metrics applied in NoC such as latency, throughput, power consumption, area overhead, and implementation complexity are mostly depending on each other and can't optimized separately. In this section, we will describe the main characteristic that should be considered in the implementation of NoC.

3.3.1 Topology

A topology has a major effect on all NoC parameters and due to that an ideal architecture should provide high throughput, low latency, low power consumption [15], and have small area requirements and implementation complexity [16]. Certainly, it is impossible to incorporate all of these features into the same system, because some of them contradict each other. That is why researchers always have to sacrifice some of the advantages of a certain architecture for the sake of gaining another one. For this reason none of the existing architectures offer the desired performance. Furthermore, the topology selection [12] [15] is also very significant in the moment of mapping the IP cores and the tasks along the network. In the following, a subset of the most popular ones that are classified based on two criteria:

- Regular or irregular.
- Direct or indirect.

Direct topologies are those that have at least one core/tile attached to each node, whereas the indirect topologies are networks that have a subset of nodes not connected to any core. In that case, these nodes only perform forwarding/relaying operations.

Also regular topologies assume each core has homogenous size, same functionality and communication requirements also they offer a predictable layout/floor-planning which helps the place and route.

a. SPIN

One of the proposed interconnect templates SPIN [17] (Scalable, Programmable, Integrated Network) uses fat-tree architecture (Figure 1.10. Black boxes are routers, white boxes are IP blocks). A fat-tree is a tree with routers on the nodes and IP blocks on the leaves. Every node has four leaves and the parent is replicated four times at any level of the tree. The number of parent port is equal to the number of child ports for every switch. The size of the network grows with a rate of $(N \ Log \ N)/8$ where N is the number of IP blocks. For N IP

blocks, the number of switches converges to $S = 3N/4$. There are as many parents as leaves so the network is non-blocking.

The term non-blocking comes from the area of Multistage Interconnection Networks (MIN), it means that it is always possible to establish a connection between any idle pair of input and output ports having no effect on existing connections [11] [27]. It is obvious that there must exist multiple paths between any given input and output ports in the network in order to be non-blocking. Supporting multiple paths leads to undesired growth of hardware complexity, power consumption, and high usage of on-chip space.

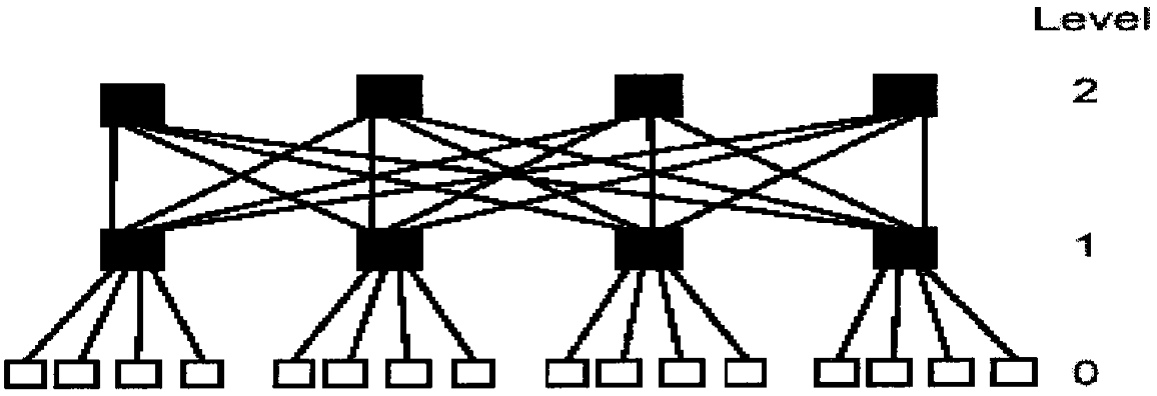


Figure 1.10. SPIN architecture [2].

b. CLICHE

CLICHE (Chip-Level Integration of Communicating Heterogeneous Elements) was proposed by Kumar et al. [8]. This architecture is the same as 2D mesh interconnection topology. There are as many switches as IP blocks. Each switch, except those at the edges, is connected to four neighbouring switches and one IP block. Local interconnections between IP blocks and switches are independent of the size of the network. The simplicity of the architecture allows for the division of the chip into processing regions. Different protocols may be used in local regions.

Routing is not complex, so the smaller size switches may be used and the network is scalable.

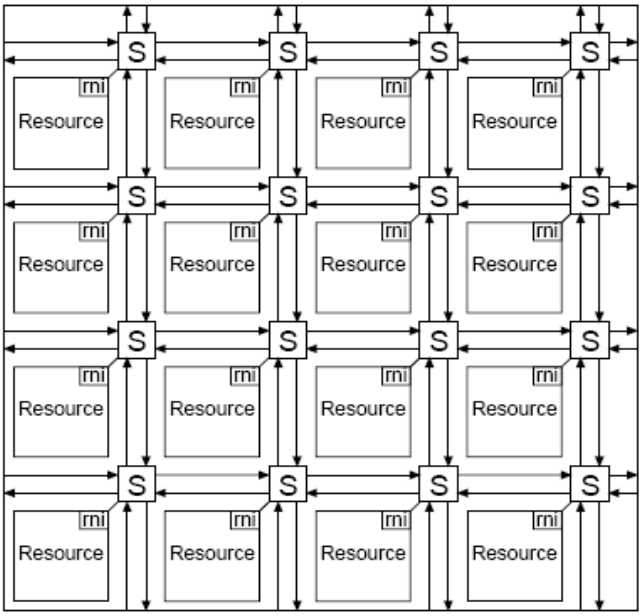


Figure 1.11. CLICHE [18].

c. Torus

Torus [7] architecture is the same as regular mesh. However, unlike the mesh, where edge switches are connected only to two neighbouring switches, the torus architecture uses wrap-around channels in order to connect the switches at the edges to the switches at the opposite edge. The number of switches is equal to the number of IP blocks and each switch has five ports.

Due to the long wrap-around channels the packet transmission delay may become significantly longer and require usage of repeaters. This can be avoided by folding the torus as it is shown in Figure 1.12. Folding is done by shifting all nodes in even rows to the right and all nodes in even positions of each row down, next connecting all the neighbouring nodes in newly gained rows and columns then pair-wise connecting edge nodes in rows and columns. Now wraparound links are significantly shorter and link propagation delays fit within a single clock cycle [31].

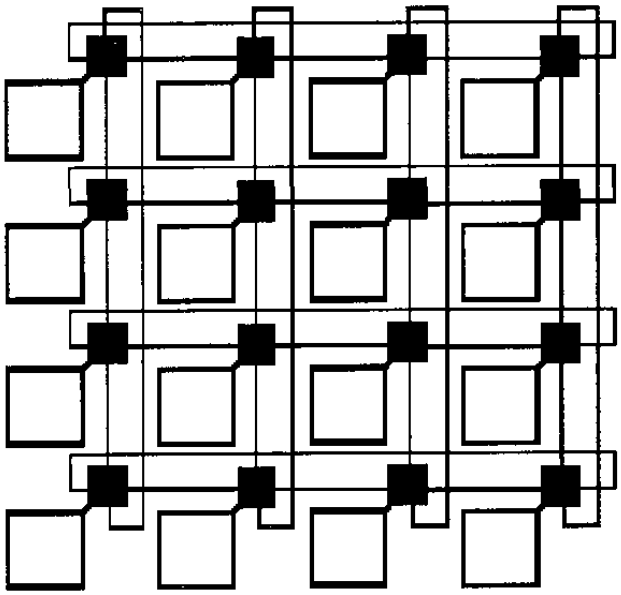


Figure 1.12. 2D Torus [2].

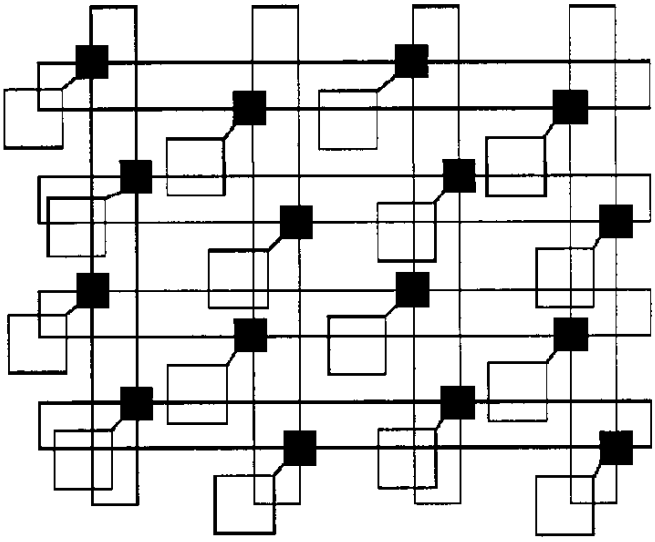


Figure 1.13. 2D Folded Torus[2].

d. Octagon

The Octagon [19] architecture has its own advantageous properties. each pair of nodes has a maximum two-hop path to communicate with each other. The basic model consists of eight IP blocks and 12 bidirectional links, as shown in Figure 1.14. The nodes are arranged in a ring and there is a central connection point in the center of a ring. Each node is also connected to the neighbouring nodes. The node consists of IP block and a switch. Each switch has three connection ports. Usually the architecture uses a simple shortest-path algorithm. The throughput

is higher than of the shared bus and crossbar interconnect if properly designed. This requires a development of good interconnection scheduler.

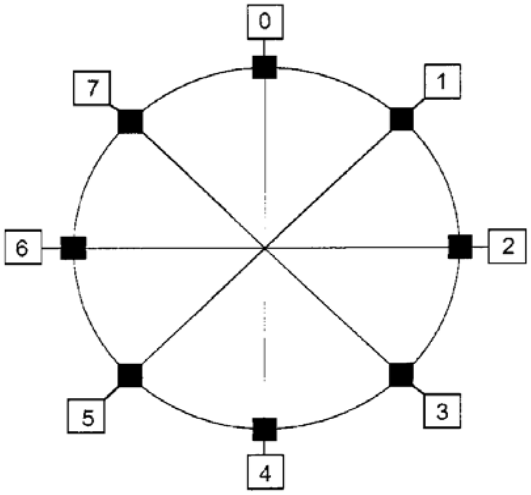


Figure 1.14. Octagon [2].

If we consider the scalability of the network, then while increasing the network size, the octagon is extended in multidimensional space. For a large number of nodes this architecture may significantly increase the wiring complexity.

e. BFT

Another proposed interconnect template is BFT (Butterfly Fat Tree) [9], [20]. In this tree based architecture, IP blocks are placed on the leaves and switches are placed at the vertices (Figure 1.15). Each switch has two parent ports and four child ports. In order to label the nodes, (l, p) coordinates are given to each node where l shows the level of the node and p shows the position of a node within that level. The address of a lower level is zero and the addresses of all IP's ranges from 0 to (N-1). There are N/4 switches at the first level and at the j^{th} level there are $N/2^{j+1}$ switches. The number of switches at each level reduces by 2. Unlike a simple mesh, where there is one switch for every four IP blocks, this topology requires one switch for every two IP blocks.

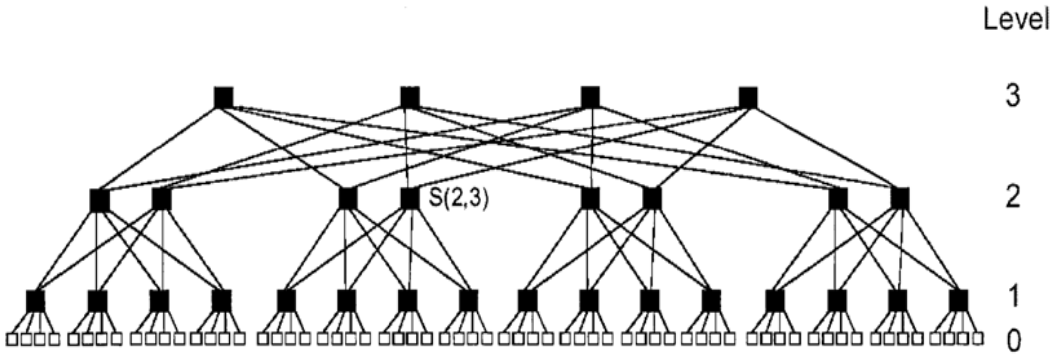


Figure 1.15. BFT Architecture [2].

3.3.2 Switching protocol

Switching techniques define the way and time of connections between input and output ports inside a switch that is enabled by the flow control i.e where the flow control is the synchronized transfer of a unit of data between a sender and a receiver and ensures the availability of sufficient buffering at the receiver to avoid the loss of data. Based on the adopted switching technique, a unit of transfer is selected where each transferred message is partitioned into fixed-length packets. Packets are individual routable units of data that contains the control bits that represent the header and the data bits as the body. The header contains destination information used by the routers to select the next port (next router) through the network. The packet as well is partitioned into fixed size units called flits whose represent the transfer unit across a link or across. There exist various switching techniques, but the most popular ones are Circuit Switching, Packet Switching, Virtual Cut-Through Switching and Wormhole Switching.

a. Circuit switching

In circuit switching [13] the physical path from source to destination is reserved for the entire duration of data transmission. This is realized by injecting the header flit into the network. The header flit contains the destination address and any additional control information. It moves toward the destination through intermediate routers reserving physical links it has passed. By the time it reaches the destination, the complete path has been reserved and the acknowledgement is sent back to the source.

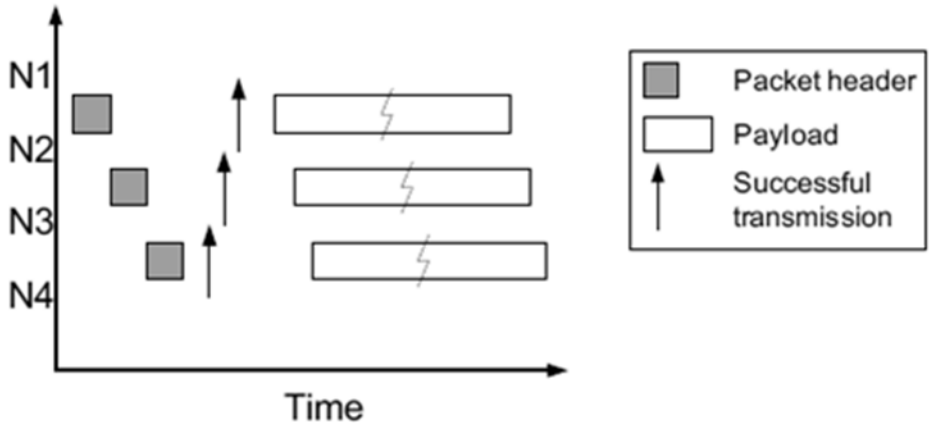


Figure 1.16. Circuit switching technique.

The reserved path may then be released by the destination or by the last bits of the message itself. This technique is advantageous when the messages are infrequent and long. In other words, it is useful when the message transmission time is long compared to the pass set up time. However, it may block other messages while reserving the entire path, thus causing unnecessary delays.

b. Packet switching

The packet switching [10] is a technique for purpose to forward the data to the next hop by using the routing information contained in the packet. In each switch the packets are queued or buffered, resulting in variable delay depending on congestion, routing algorithm, switch arbitration, etc. Packet switching has the following switching techniques:

b. a. Store and forward (SAF)

The message is divided into fixed-length blocks, called packets [13]. Unlike the circuit switching technique, which sets the path before sending a data, each packet is routed individually from source to destination. Each packet has routing and control information called packet header, which is used by intermediate routers to determine the packet's destination. Thus, the latency of a packet varies with the distance between source and destination. The longer the distance the greater is the latency.

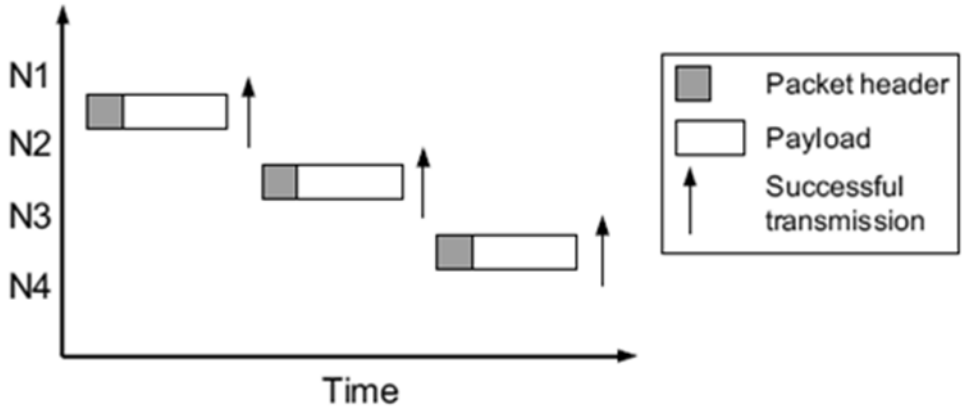


Figure 1.17. Store and Forward switching technique.

Packet switching is advantageous when messages are short and frequent. It also fully utilizes the communication link, while the circuit switching may keep the reserved path idle for some time. However, the storage requirements of the individual routers can become extensive if packet size becomes large and multiple packets must be buffered at a node.

b. b. Virtual cut- through (VCT) switching

In the packet switching technique, a packet must be received in its entirety before making any routing decisions. However, the size of a packet may be bigger than the width of a physical channel, so the transfer of a packet may take multiple cycles. The width of a physical channel is measured in bits and defines how many bits of information can be sent through the physical channel in parallel. The packet header is the first few bytes of a packet that can be received after the first few cycles and it contains the routing information of a packet. Rather than waiting for an entire packet to be received, the router can start forwarding the packet header and following data as soon as the routing decision is made and the output buffer is free. In the absence of blocking, the packet does not have to be buffered in the output buffer and can cut through directly to the input buffer of the next router before the current router receives the complete packet. This switching technique is called virtual cut-through (VCT) [13]. The difference of this technique from packet switching is that the packets do not always have to be buffered in the intermediate routers; they are buffered only if the packet is blocked. That is why at high network loads the virtual cut-through switching behaves just like packet switching. Only the packet header contains the routing information and the following data is simply forwarded along the same output channel as its predecessor. Therefore, transmission of different packets cannot be interleaved or multiplexed over the same physical channel.

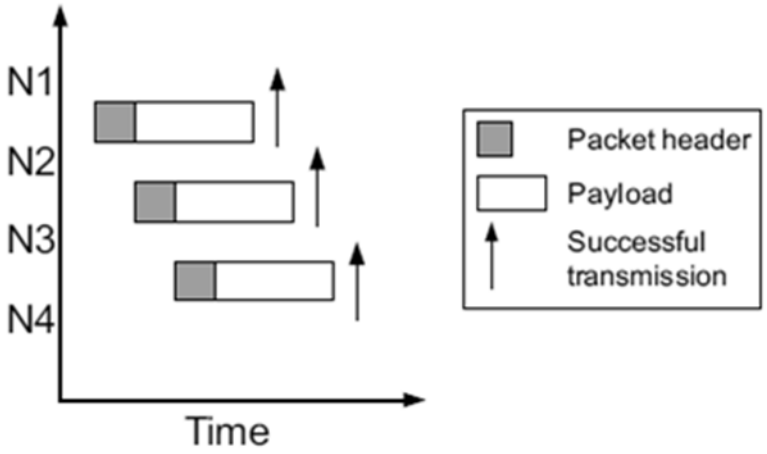


Figure 1.18. Virtual Cut Through switching technique.

Unlike circuit switching in VCT and SAF switching we do not have to reserve the whole path from source to destination since every packet contains routing information. A physical link between just two adjacent routers is reserved for the duration of packet transmission; it is released as soon as packet reaches the next router. In VCT and SAF switching, each packet is routed independently from others and packets with the same source and destination may take different paths. In circuit switching links are underutilized, because if the header flit is stalled all the physical links reserved so far cannot be used by anyone. VCT and SAF switching do not suffer from this kind of problems. As we said above, the circuit switching is good for lightly loaded networks with long messages while VCT and SAF switching are suitable for heavy loaded network configurations. It must be noted that, we can make the use of VCT's advantages over SAF only if the packet size is bigger than the width of the physical channel measured in bits, otherwise they perform similarly.

b. c. Wormhole switching

In wormhole switching [13] the message is divided into flits. This is done in order to decrease the buffer size at routers and to achieve much faster routers. The input and output buffers of a router are large enough to contain a few flits. A message is sent through the network at a flit level in a pipelined fashion. The header flit contains the routing information and builds a path in the network, which the other flits follow. In case of blocking VCT collects the following data in the current router, which requires bigger buffer size, while wormhole switching simply stops every flit in its current position. Thus, when a message is blocked it occupies several routers in the path constructed so far and a few flits need to be buffered at a router. As a result, there is no need for a local processor memory to buffer messages, which significantly reduces average message latency. However, only the header flit contains the

routing information and each incoming data flit is simply forwarded along the same output channel as the preceding data flit, which requires that the message must cross the channel in its entirety before it can be used by another message. This blocks the resources in case of stalled pipelines and makes the wormhole technique deadlock prone.

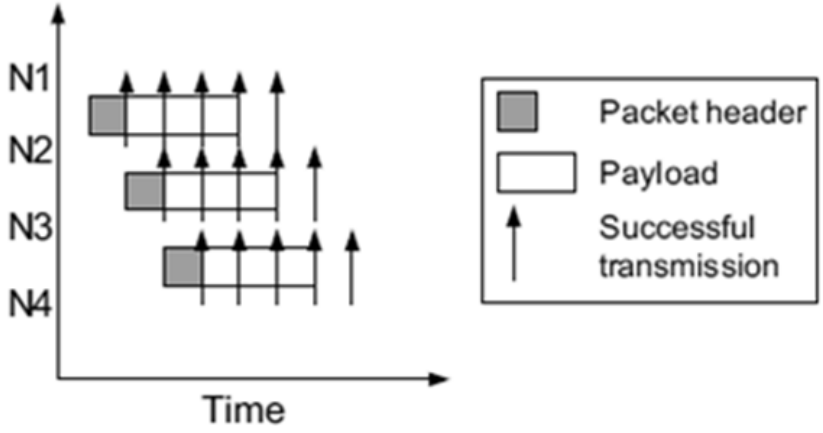


Figure 1.19. Wormhole switching technique.

The problem can be solved by adding some control logic that splits the physical channel into several virtual channels. They will have their own buffers, but share one physical medium, in other words, each port of the router will have several queues for storing incoming flits instead of a single queue. With such an organization, several buffers are associated with a single physical channel and from several virtual channels, it is like adding lanes to a street. Each of the queues is allocated to a certain packet, and contains flits of only that packet [30]. A physical channel is allocated to each of the virtual channels in flit-by-flit manner. The buffers sharing the same physical channel receive their flits alternately. If one of the packets is blocked the other one can advance further towards its destination.

c. Comparison of switching techniques

A comparison of different switching technique is done, but before presenting the most appropriates, the buffering and virtual channels in a network on chip are first presented because the amount of buffering resources in the network depends on the implemented switching technique.

c. a. Buffering and virtual channels

Buffers contribute on the main part of the area usage of the usage of the switch. Therefore, it is to try to reduce as much as possible the amount of buffering area without losing performance requirements [6] at the moment of design the switches.

There are two main aspects related to the buffering capabilities:

The buffers size: the width and the depth defines how many words are able to allocate on the switch.

The blocking between input buffer and output buffer: according to [32], the performance of queuing in output port is always less than the input port, and this is because in router strategy the head-of-the-line blocking problem is always facing the incoming packets that are buffered before packets whose output port is busy.

Also the increasing of buffer size is not a solution to avoid congestion [33] because not only the power and area of the chip is augmented but also the performance of both throughput and the system delays is degraded.

Another metric to improve NoC's routing is to combine with a switching technique strategy and a buffering scheme with the concept of virtual channels (VCs).

VCs [34] [35] are for purpose to:

- Reduce latencies and throughput of the system
- Avoid deadlocks
- Optimize wiring utilization (reducing leakage power and congestion).
- Increase network performance (minimizing the frequency).
- Moreover, the insertion of VCs enables to implement policies for allocating the physical channel bandwidth, which enables to support Quality of Service in applications.

c. b. Comparison of different switching techniques

In table 1.2, a brief summary of switching techniques is given (The term of energy consumption and power dissipated are defined in the next chapter).

Switching Technique	Per switch cost		Stalling	Energy consumption	Resource utilization
	Latency	Buffering			
Circuit switching	Packet	packet	At all nodes spanned by the packet	Significant leakage power	Poor
SAF switching	Packet	Packet	At two nodes between the link	Minimal leakage power comparing to Circuit switching	Good

VCT switching	Header	Header	At the local node	Reduce the energy consumed	Good
Wormhole switching	Header	flit	At all nodes spanned by the packet	Small buffer leads to reduce the static power and the Reduce the energy consumed	Moderate

Table 1.2. Comparison of switching techniques in network on chip.

3.3.3 Routing protocol

A routing algorithm determines a path for a packet to reach its destination. It must be decided within each intermediate router which output channels must be selected to be crossed by the incoming messages. There are various types of routing algorithms differentiated according to their key characteristics. In accordance with the place where the routing decision is made they may be grouped as centralized, source, and distributed routing algorithms. If an algorithm is centralized the path is chosen by a centralized controller, if it is source routed then the route is determined by the source router prior to sending a packet, in distributed algorithms the path is chosen in a distributed manner at the intermediate routers. According to the way how they choose a path routing algorithms are broadly classified as deterministic and adaptive algorithms.

Deterministic algorithms do not take into account network conditions when they take a decision that is why they always supply the same path from source to destination. But, it is not the case for adaptive ones in which network load, traffic conditions, information about available output channels are always taken into consideration. Each algorithm has a different impact on the network. Routing algorithms use a variety of metrics that affect the calculation of the optimal path for a message. Many properties of the interconnection network depend on the routing algorithm used because the complexity of an individual router has a significant impact on the complexity of the entire network. For example, if the routing algorithm is too complicated it will require extra hardware to realize the routing logic, moreover, it may take much more time to make a decision about the direction where the message should be sent to. It will in turn lead to increase of packet latency. Deadlock, livelock and starvation freedom are also among those properties. This property shows the ability to guarantee that packets will not block or wander across the network forever or permanently stop and never reach its destination.

- **Deadlock**

Deadlock is one of the situations that can postpone packet delivery indefinitely. It happens when a packet is requesting a resource that is held by another packet while holding the resource that is requested by another packet. There is a cyclic dependency between channels. Thus the packet may be blocked forever. Deadlock is the most difficult problem to solve. There are three strategies that can cope with deadlock: deadlock prevention, deadlock avoidance and deadlock recovery.

- **Livelock**

Livelock usually happens in adaptive routing schemes. It happens when a packet has been running forever in circular motion around its destination, because the channels that are required to reach the destination are occupied by other packets. Hot potato routing is an example of an algorithm that can cause a livelock. In this algorithm, whenever a desired channel is not available, a packet will pick any alternative available channel and move to the next switch instead of waiting. However, the alternative channels may misroute a packet around its destination. In order to remove livelock several techniques have been proposed such as minimal path, restricted non-minimal path, and probabilistic avoidance. Starvation: Starvation may happen when a resource that was requested by a packet is always granted to other packets. Thus, the packet stops permanently in traffic, never getting the resource it needs. Starvation can be avoided by using the correct resource assignment scheme.

a. Deterministic routing algorithms

Deterministic routing algorithms [13] always generate the same single routing path for a given pair of source and destination, usually choosing the shortest one. Only the addresses of current and destination nodes are used to compute the path. As messages with the same source and destination always use the same network path, they cannot use alternative paths to avoid blocked channels. If source routing is used, the path is computed at the source node without considering any information about traffic. Otherwise, if distributed routing is used, routers make unique decisions at the intermediate nodes, based on current and destination node addresses. In both cases channel status is not considered while computing the output channel to be used.

Deterministic algorithms should be progressive and profitable, which means that the header should move forward to reserving a new channel at each routing operation, under condition that the supplied channel always brings the packet closer to the destination. Thus deterministic routing algorithms use greedy algorithms, always choosing the shortest path.

The most popular deterministic algorithm is known as dimension-order routing. It is based on the idea that some topologies can be decomposed into several orthogonal dimensions, i.e. hyper cubes, meshes and tori. The distance between two nodes in these topologies is computed as the sum of the offsets in all dimensions. The algorithm reduces one of these offsets in each routing step. The offset of the current dimension must be equal to zero before the algorithm considers the offset of the next dimension.

Dimension–order routing is usually used for meshes and hypercubes. In 2D mesh it is called XY- or YX-routing depending on the dimension in which a packet travels first. The algorithm is deadlock-free for n-dimensional hypercubes and meshes, as their channel dependency graph (CDG) is acyclic. CDG is a directed graph where channels are represented by vertices and edges are pairs of channels connected with a routing function [29]. However, the CDG for some topologies has cycles. In order to remove cycles, physical channels may be split into virtual channels.

Most commercially available parallel machines usually use distributed deterministic routing as it is simple and fast. But distributed deterministic routing assumes that the traffic is uniform. In case of non uniform traffic the performance of distributed deterministic routing in terms of latency and throughput is very poor [13].

b. Adaptive routing algorithms

Adaptive routing algorithms do not restrict a message to a single path when travelling from the source to the destination. While making a decision the current network conditions are considered. This makes the routing more flexible and reduces unnecessary waiting time delays, providing better fault tolerance.

Adaptive routing algorithms contain two functions: routing and selection. Routing function gives a set of output channels based on the current node and destination node, Selection function selects the most appropriate output channel from that set.

The selection function always supplies with a free channel. Thus, messages can follow alternative ways instead of waiting for a busy channel. Non greedy algorithms, which can supply channels that send packets away from its destination, are also allowed.

Adaptive algorithms also allow backtracking technique, which enables the header to backtrack while releasing the previously reserved channels, thus systematically searching for appropriate path. For deterministic algorithms, the backtracking technique is useless as the message will go by the same path again. Adaptive routing algorithms increase routing flexibility, but the hardware becomes more complex and slower.

c. Minimal adaptive routing

Minimal routing algorithms always use the shortest path in order to reach the destination. While being adaptive they restrict the routing direction in some level. One of the examples of minimal adaptive routing algorithms is double Y-channel routing algorithm [21]. This algorithm divides the network into several sub-networks. The packet is sent via a particular sub-network according to the location of destination. The network is usually divided into +X sub-network and –X sub-network. Here Y dimension has a pair of channels and X dimension has unidirectional channel. If the location of the destination node is bigger than the location of source node, in other words, if dx > sx, than the packet is sent through +X sub-network. Otherwise –X subnetwork is used. If dx = sx then either sub-network can be used.

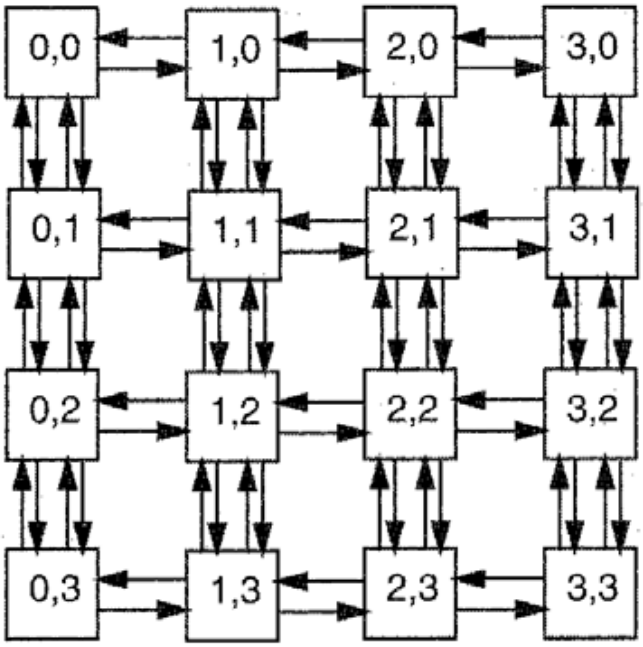


Figure 1.20. Double *Y*-channel 2D mesh [22].

The double Y-channel algorithm is minimal and fully adaptive, which means that the packet is delivered through any of the shortest paths. In order to avoid deadlock, channels should be ordered in an appropriate manner [21]. The packet that is sent to the destination should follow the channel in descending order, in other words, the channel label that was passed must be bigger than the channel label that is going to be passed.

This algorithm is impractical when n is large (n is the number of dimensions), due to the additional channel requirement.

d. Non-minimal adaptive routing

Unlike the minimal adaptive routing algorithm, where a packet searches only for the shortest path, non-minimal adaptive routing allows the packet to take a longer path if there is no available shortest path.

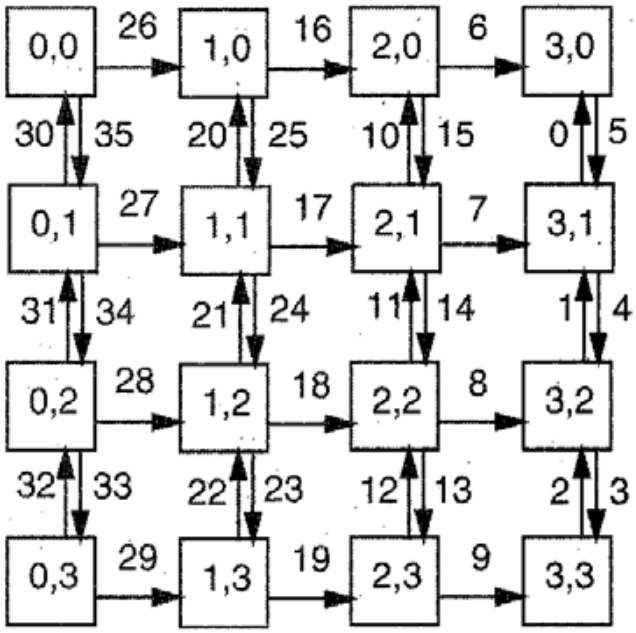


Figure 1.21. +*X* sub-network and labeling [22].

This technique is fully adaptive and can be achieved by few more additional channels. Two non-minimal routing algorithms were proposed by Dally et al. [23].

One of them is static dimension reversal routing algorithm, where every two adjacent nodes are connected by r pairs of channels. Here, the network is divided into r sub networks and the i^{th} sub-network consists of all the i^{th} pair channels. Each packet header stores additional value c which is initially set to zero. The packet can move in any direction in its own sub-network, but when the packet moves from high dimensional channel to low dimensional channel the c field is increased by one. If c reaches the value r-1 then the packet must switch into the deterministic dimension ordered routing algorithm. The packets can be routed also in reverse dimension order. Parameter r restricts the number of times it can happen.

Another algorithm is called a dynamic dimension reversal routing algorithm. Here, the channels are divided into two classes: adaptive and deterministic. At first, packets are sent through adaptive channels, moving in any direction. But, when a packet reaches a node, where all output channels are busy by packets with values of c that is smaller or equal to its own, it must switch to the deterministic channels. After entering the deterministic channels a packet cannot return to adaptive channels. The algorithm is deadlock-free.

e. Turn model

This algorithm was proposed by Glass and Ni [14] for n-dimensional meshes. It suggests a systematic approach to the development of adaptive routing without additional channels. The algorithm supports both minimal and non-minimal adaptive routing and is partially adaptive.

The packet, while travelling through the network, switches from one dimension to another, in order to reach the destination. Switching from one dimension to another is called the turn. If the packet changes its direction without moving to another dimension, then this is 180-degree turn. Usually physical channels are split into virtual channels and the packet may move from one virtual channel to another. If it moves from one virtual channel to another and is still in the same dimension and direction then this is a 0-degree turn. Turns can form a cycle. It is known that, deadlock occurs when the packet routes contain turns that form a cycle. The algorithm is based on the idea that, if there is no cyclic dependency between channels then deadlock cannot occur. So, the concept of the algorithm is to prohibit the smallest number of turns such that cycles are prevented. There are several steps that can be useful in developing maximal adaptive routing algorithms:

• Classify channels according to the direction in which they route packets.

• Determine all possible turns that can occur between one direction and another. 00 and 1800 turns are not considered.

• Identify the cycles that can be formed.

• Prohibit one turn in each cycle.

• 00 and 1800 turns cannot be prohibited as they are needed for non-minimal routing algorithms or if there are multiple channels in the same direction.

One of the examples for this algorithm is west first routing algorithm, where it first routes a packet west, if necessary, and then adaptively south, east and north.

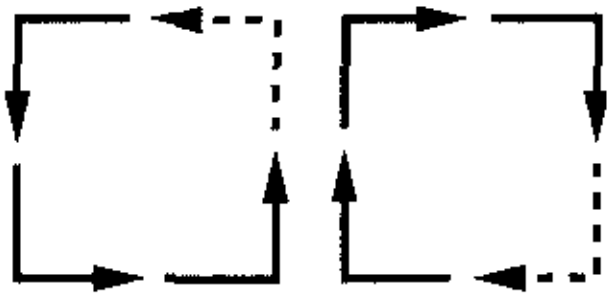


Figure 1.22. Six turns (solid arrows) allowed in west-first routing [22].

Figure 1.22 shows the turns that are prohibited. Thus, the packet can make only six turns and two turns to the west are not allowed. Therefore, to travel west, a packet must begin in that

direction. For minimal routing, the algorithm is fully adaptive if the destination is on the right side of the source, otherwise, it is deterministic. For non minimal routing the algorithm is adaptive in either case.

f. Randomized routing algorithms

Randomized routing algorithm is considered to be one of the most popular deterministic routing algorithms due to its simplicity of the implementation and the good performance, according to average packet delay and throughput metrics [25]. As we can see in Figure 1.23 not the whole available bandwidth of the network is used. Sometimes minimal constraints are relaxed to some extend and randomization is added for more efficient usage of the network bandwidth.

Valiant and Brebner [24] proposed one of the best known randomized algorithms named Valiant. This algorithm has two phases. In both of the phases it uses dimension-order routing. In the first phase a random node is selected, and a packet is sent there. In the second phase, it routes the packet from that random node to its destination. Valiant is non-minimal and tries to avoid congestion in the network.

Packets with the same source and destination are unlikely to take the same path, as they will have different intermediate nodes, which are selected randomly. Several sets of buffers are needed in order to avoid deadlocks. In a mesh topology, it requires two sets of independent buffers per communication link. In Valiant routing algorithm livelocks cannot occur, as the packet reaches its destination eventually, but it may take a longer path than actually required. The path taken is a drawback of this algorithm, because its length increases according to the topology being used. For example, in a mesh, the packet's path may be doubled. This means that the Valiant routing results in longer routing times. Hence, the demand for network bandwidth is also doubled.

Another routing algorithm which uses randomization is Randomized, Oblivious, Multi-phase, Minimal (ROMM) [25] algorithm. It inherits minimality of DOR, and randomization of Valiant. Minimality assures that the path taken by a packet will be minimal. Randomization is used to assure that packets with the same source and destination will not take the same path. These properties are combined to avoid congestion. This algorithm is oblivious, thus, making it easy to implement in contrast to adaptive algorithms which are more complex. As ROMM algorithm uses only minimal paths, it is constrained in randomization unlike Valiant which uses full randomization.

ROMM selects random nodes within the range of a minimal path a message is required to follow in order to reach the destination. It routes a packet in p-phases. A p-phase ROMM algorithm has p-1 randomly selected nodes Z1, Z2, …, Zp-1 between source and destination, such that, those Zi ‘s must be on the minimal path from source to destination.

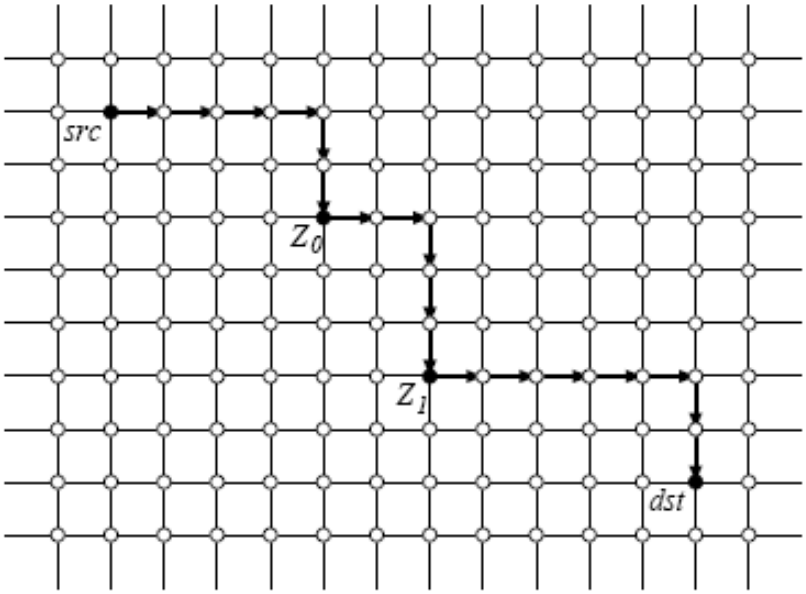


Figure 1.23. A possible path from *src* to *dst* using 3-phase ROMM on a 2D mesh.

Figure 1.23 illustrates a possible path from source to destination in a 3-phase ROMM. Here, the source node determines the path that would be taken by message using dimension-order routing and chooses a random node that lies within that path, in our case Z0. Then the message is routed from src to Z0 using dimension-order routing. Now, Z0 becomes a source node and chooses another random node that lies along the minimal path from Z0 to dist, in this case Z1, and send a message using dimension order routing. Z1 in its turn routes a message to dst again using dimension-order routing.

If p phases are used, then ROMM algorithm requires p-virtual channels for wormhole routed mesh network, in order to avoid deadlocks. The algorithm becomes very costly as p increases, as it increases memory usage adding virtual channels and complicates routing with additional logics required to manage virtual channels.

Another algorithm which allows several routes is O1TURN [26]. In O1TURN a network is partitioned into two virtual networks. One of them is XY-routed the other is YX-routed. It uses just one phase, unlike ROMM and Valiant. However, it also provides more than one path, because when the flit is injected into the network it may choose one of the networks, more precisely it is randomly sent into one of the virtual networks. According to [26] O1TURN

outperforms ROMM, Valiant and DOR under non-uniform traffic. But DOR is still better under uniform traffic pattern.

g. Comparison of routing algorithms

In this section we introduce general comparison of routing algorithms mentioned in previous sections as it is indicated in the literature. Deterministic algorithms in comparison to adaptive ones achieve higher throughput behaviour under the uniform traffic pattern in 2D mesh with the same number of VCs and have lower latencies at higher throughputs [13][14]. However, adaptive algorithms outperform them when used in 2D torus [13]. Deterministic algorithms suffer from channel under utilization while adaptive algorithms distribute the traffic more uniformly across the network and do not have such a shortcoming. Under non-uniform traffic pattern adaptive algorithms perform much better than deterministic ones in terms of throughput and latency. It is due to their ability to provide several routes between the same pair of source and destination. Performance evaluation of O1TURN, DOR, and ROMM algorithms and their comparison were presented in [26]. DOR outperforms O1TURN under uniform traffic when the network size is small (4x4). It achieves lower latency and higher throughput behaviour. When network size increases O1TURN behaves better. ROMM has the worst performance (greater latency, lower throughput) under uniform traffic. O1TURN is always better than DOR under non-uniform traffic pattern regardless of the network size. Under non-uniform traffic the performance of O1TURN is almost the same or better than the best of DOR or ROMM.

4. NoC simulator

The dedicated NoC tools vary depending on the purpose for which they are developed such as simulators. Two criteria are often addressed for simulators: the estimation of power dissipation and performance computing (throughput, latency, and reliability). These two criteria are crucial since NoC are an integral component of an embedded system. Because these systems are often subject to hard constraints of space, energy and execution time, a relevant estimation of the NoCs characteristics could be very helpful to the designer. We will show in the following NoCs synthesis or simulation tools we collected from the literature:

4.1 NS-2

NS-2 was first developed for prototyping and simulating ordinary computer networks. However, since NoCs shares many characteristics with classic networks, NS-2 was widely used

by many NoC researchers to simulate NoCs [36]. NS-2 is an open source, discrete event driven simulator and developed in C++ and OTcl. These modularity and availability have facilitated its spreading between researchers.

4.2 Noxim

This tool has been proposed by the Computer Architecture team at the University of Catania [37]. It is developed in SystemC language. It allows the user to define 2D mesh NoC architecture with various parameters including: network size, buffer size, packet size, routing algorithm and injection rate of packets. Noxim allows the evaluation of NoCs in terms of throughput, latency and power consumption.

4.3 DARSIM

DARSIM is a NoC simulator, which was developed at the Massachusetts Institute of Technology (MIT). This tool allows the simulation of mesh NoC architectures of 2 and 3 dimensions. It offers a multitude of NoC simulation configurations with various parameters. This includes two generation modes of data generation [38]: a) Trace-driven injection, which involves the injection of packets into the network and monitors their spatial and temporal evolution. b) MIPS Simulation mode: each node can be configured as a Microprocessor Without Interlocked Pipeline Stages (MIPS) with its own memory.

4.4 SunFloor - 3D SunFloor

SunFloor is a support tool for NoC design. It can be used at earlier design phases to synthesize the most appropriate topology with these constraints as input (Model, Energy and Space, Design Objectives). From these data, SunFloor generates a system specification ready to be translated into comprehensive architecture, usually in SystemC language and by the intervention of a second tool which is xpipesCompiler [39].

4.5 ORION 2.0

ORION 2.0 is the successor of the version proposed by a team from Princeton University in 2003 [28]. It is a simulator dedicated primarily to the estimation of power and space for NoCs architectures. Among the improvements compared to the first version we find the support for new semiconductor technology through models of transistors and capacitances upgraded from industry.

5. NoC Examples

In this section, we briefly recapitulate a complete compilation of existing NoCs, there are many more, rather the purpose of this section is to address a representative set: ÆTHEREAL, NOSTRUM, SPIN, CHAIN, MANGO, and XPIPES:

5.1 ÆTHEREAL

TheÆTHEREAL, developed at Philips, is a NoC that provides guaranteed throughput (GT) alongside best-effort (BE) service. In the ÆTHEREAL the guaranteed services pervade as a requirement for hardware design and also as a foundation for software programming. The router provides both GT and BE services. All routers in the network have a common sense of time, and the routers forward traffic based on slot allocation. Those sequences of slots implement a virtual circuit. GT traffic is connection-oriented, and in early router instantiations, did not have headers as the next hop was determined by a local slot table. In recent versions, the slot tables have been removed to save area, and the information is provided in a GT packet header. The allocation of slots can be set up statically, during an initialization phase, or dynamically, during runtime. BE traffic makes use of non reserved slots and of any slots reserved but not used. BE packets are used to program the GT slots of the routers. With regard to buffering, input queuing is implemented using custom-made hardware FIFOs to keep the area costs down. The ÆTHEREAL connections support a number of different transaction types, such as read, write, acknowledged write, test and set, and flush, and, as such, it is similar to existing bus protocols. In addition, it offers a number of connection types including narrowcast, multicast, and simple.

5.2 NOSTRUM

The work of researchers at KTH in Stockholm has evolved from a system-level chip design approach. Their emphasis has been on architecture and platform-based design targeted towards multiple application domains. They have recognized the increasing complexity of working with high-density VLSI technologies and hence highlighted the advantages of a grid-based, router-driven communication media for on-chip communication. Also the implementation of guaranteed services has also been a focus point of this group. In the NOSTRUM NoC, guaranteed services are provided by so called looped containers. These are implemented by virtual circuits, using an explicit time division multiplexing mechanism which they call Temporally Disjoint Networks (TDN).

5.3 SPIN

The SPIN network (Scalable Programmable Integrated Network) implements a fat-tree topology with two one-way 32-bit data paths at the link layer. It is proven that, for any given amount of hardware, a fat tree can simulate any other network built from the same amount of hardware with only a polylogarithmic slowdown in latency. This is in contrast to, for example, two-dimensional arrays or simple trees which exhibit polynomial slow down when simulating other networks and, as such, do not have any advantage over a sequential computer.

In SPIN, packets are sent via the network as a sequence of flits each of size 4 bytes. Wormhole routing is used with no limit on packet size. The first flit contains the header, with one byte reserved for addressing, and the last byte of the packet contains the payload checksum. There are three types of flits; first, data, and last. Link-level flow control is used to identify the flit type and act accordingly upon its content.

5.4 CHAIN

The CHAIN network (CHip Area Interconnect) developed at the University of Manchester is interesting in that it is implemented entirely using asynchronous, or clockless, circuit techniques. It makes use of delay insensitive 1-of-4 encoding, and source routes BE packets. An easy adaption along a path consisting of links of different bit widths is supported. CHAIN is targeted for heterogeneous low-power systems in which the network is system specific. It has been implemented in a smart card which benefits from the low idle power capabilities of asynchronous circuits.

5.5 MANGO

The MANGO network (Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces), developed at the Technical University of Denmark, is another clockless NoC, targeted for coarse-grained GALStype SoC. MANGO provides connectionless BE routing as well as connection-oriented guaranteed services (GS). In order to make for a simple design, the routers implement virtual channels (VCs) as separate physical buffers. GS connections are established by allocating a sequence of VCs through the network. While the routers are implemented using area efficient bundled-data circuits, the links implement delay insensitive signal encoding. This makes global timing robust because no timing assumptions are necessary between routers.

6. Conclusion

NoC occupies a wide spectrum of research, ranging from highly abstract software related issues, across system topology to physical level implementation. In this chapter, we have given an overview of activities in the field. We first defined the embedded system with its different architecture and in order to avoid the wide range of topics relevant to large scale IC design in general, we stated the motivation for NoC and given an introduction of the basic concepts.

CHAPTER II

MODELS OF COMPUTATION IN MULTIMEDIA APPLICATIONS AND NOC ENERGY EFFICIENCY AND DELAY MODEL

1. Introduction

In the past, the number of instructions executed in a computer system in one unit of time (e.g., MIPS or FLOPS) was the important indicator of measuring the performance of a system. In the future, the number of instructions executed per unit of energy (e.g., a joule) will be of equal importance.

However NoCs are composed of several processing elements, e.g. processors, memories, links and various other communication interfaces. In order to automate the design process, system specification has to provide clear and unambiguous semantics as well as sufficient expressive power to represent the vast majority of embedded system applications. Models of Computation (MoC) and system design languages provide the foundation for defining the NoC

behavior. There are several requirements imposed on the behavior modeling of system level design process for NoCs. In order to identify relevant requirements and constraints for embedded system specification, formal models of computation are described and evaluated according to the expressive power, supported features, complexity and analyzability.

The objective of this chapter is to put the light on understanding the concepts of energy-efficient embedded computing. It is divided into two sections. In the first section, we define the models of computation for multimedia applications with many examples used in NoC and then we introduce in the simple models to estimate the latency of the system and energy consumption of different tasks. This gives the reader an indication of where energy is dissipated and what are the mechanisms of energy dissipation. Since energy consumption depends very much on the considered technology, we assume a hypothetical 180 nm CMOS VLSI technology as the reference for the estimation. The last section focuses on the power reduction techniques based system level.

2. Models of computation

Model of Computation (MoC) is what represent the system behavior, requirements and constraints on a high level, and this representation is described in the formal manner by means of mathematical functions, set-theoretical notations or combinations thereof. However, two basic types can be differentiated and processed based- and state based- MoC.

2.1 Process-oriented MoC

Process oriented MoC [40] describes the system behavior in the form of processes that communicate with each other by messages that are transmitted in channels or shared memories. Process oriented MoC describes the concurrency behavior of different applications in a system and due to that it is the suitable for implementation of the multi-core platform. The functionality of a process is typically given in the imperative form, defined in a programming language, e.g. C, C++. However, the consequent statements of the communicated processes could be represented in a form of activity diagrams or flow charts and for that the ordering between different processes is based upon the dataflow dependencies between those processes, making the process based MoC suitable for high level specification at the beginning of the design flow.

Notions inter process concurrency and communication mechanisms require a deadlock and determinism analysis. Deadlock refers to a well known concept of circular dependencies between processes, while determinism refers to the predictability of the program behavior. A deterministic MoC produces the same output for the same input set. Such behavior is highly

desirable in order to perform system behavior validation, but fully deterministic model can lead to over specification. More specifically, it is desirable for a global system comprised out of several processes to produce the same outputs for a given set of inputs, but the order their execution should remain arbitrary. In order to cope with these constraints and requirements, different process based MoC were proposed throughout the years. Some of the most prominent process based MoC include process networks, dataflow models and process calculi.

Process networks [41] are the most common example of a process oriented model of computation. Essentially, process networks are composed out of concurrent processes that communicate through buffers. Communication is asynchronous on the server side, where the sent messages are being accumulated in buffers. On the receiver side, communication is blocked, because the receiver cannot proceed with the computation until the required inputs are read. Process networks are convenient for modeling data-flow dominated applications, which are inherently partitioned into separate tasks communicating via streams of data. Embedded multimedia, imaging and signal processing applications employ such dataflow model and they are usually modeled using process network mechanism.

Dataflow model [42] specification enforces the graphical representation of the system behavior, which is given in the form of a directed graph. Nodes of the graph correspond to the actors, where arcs represent the unbounded buffer channels between them. In comparison to a process employed in the process network MoC, actor corresponds to the atomic block of execution.

In this way, context switch operation is avoided and consequently runtime overhead is decreased. Actors execute, i.e. fire when all their inputs are available, performs some computation algorithm and produce outputs.

2.1.1 Kahn process networks

Kahn process networks (KPN) [43] are process based Model of Computation where the processes that are executed in parallel are independent of each other and the communications between them are unidirectional and point-to-point message passing channels. Such message passing channels require buffers as it is illustrated in figure 2.1. KPN are more faced to blocking as it is shown in figure 2.1 where the sending via channels can never block while the receiver processes always block until required input data from the channel are available. The blocking at the channel level requires a scheduling strategy to avoid deadlock and blocking and also those strategies can affect the memory requirements, for that two basic scheduling policies can be differentiated, demand driven scheduling and data driven scheduling.

- Demand driven scheduling strategy corresponds to the behavior where each process can run whenever its data is needed. Such behavior could lead to artificial deadlocks in the case of the local deadlock of a single process [40].
- Data driven scheduling runs processes whenever they are ready. Here, in the case of a local deadlock of a specific process, tokens would be accumulated on arcs creating a problem of memory consumption requirements. In the case of the buffer overflow condition, where buffers are full, scheduling policy needs to decide between complete and bounded execution. Complete execution refers to the condition where processes are run as long as they are ready. However, such behavior could lead to unbounded memory requirements, where the buffer size is increased according to the run-time requirements. Bounded execution limits the size of the buffer and senders are therefore blocked, until messages are received by the receiver process.

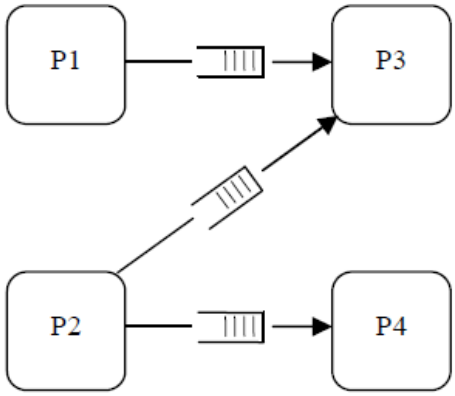


Figure 2.1. Kahn Process Network example. [40]

In general, KPN is suitable as a model of computation for the multi-core platform [44] for its determinism where the number of input results the same output regardless of the scheduling. The main drawback of the KPN as a model of computation is that they require a dynamic scheduling with runtime context switching as well as dynamic memory allocation. Dynamic memory allocation corresponds to the property where buffers for inter process communication are dependent on the underlying scheduling policy. Therefore, efficient and practical KPN realization of a multi-core system is difficult to achieve.

2.1.2 Process calculi

Process calculus [45] is a model of computation that represents the interactions, communications and synchronization mechanisms among concurrent processes in a high level formal description. Formal representation is given by a set of processes, composition rules and

axioms where the specific composition rules include parallel composition of processes, sequential composition or choice. Furthermore, the notion of recursion and replication are supported, and the communication between processes is restricted to the synchronous message passing. Process calculi models are suitable for analysis, equivalence checking, and formal verification for its formalization and restricted execution semantics.

2.1.3 Synchronous data flow

a. Definition

Synchronous Data Flow (SDF) specification is a specialization of the dataflow modeling paradigm [42]. In SDF, parallelism is represented explicitly, which makes such specification convenient for HW/SW Co synthesis. Given:

SDF, as a dataflow model of computation employ an actor as a basic unit of computation. Actors are atomic blocks of execution and SDF therefore lacks expensive runtime context switching operation. Furthermore, due to the restrictions imposed on the original dataflow architecture, where the number of tokens produced and consumed by the actor remains fixed, SDF can be scheduled statically.

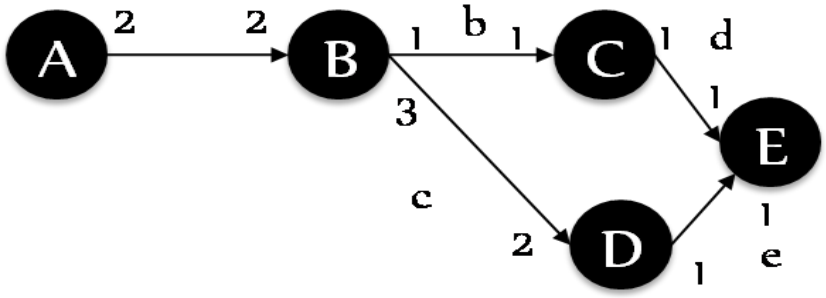


Figure 2.2. Synchronous dataflow graph example. [40]

b. Formal model

Formally, SDF is defined by a tuple (V, E, d, P, O, I) where:

V is the set of actors(tasks).

$E \subseteq V \times V$ is the set of directed edges.

$d : E \rightarrow \mathrm{N}$ is a function describing the number of initial tokens on an edge $(u, v) \in E$.

$P : V \rightarrow R^{+}$ is a function describing the flow on each edge.

$O : E \rightarrow N$ in a function describing the number of tokens produced on edge $(u, v) \in E$ by actor v on each execution

$I : E \rightarrow N$ in a function describing the number of tokens consumed on edge $(u, v) \in E$ by actor u on each execution

- The nodes in the SDF graph are called actors.
- Actors have well defined input/output behavior.
- Actors have the workload that is defined by the amount of work given in units of work.
- Actors produce and consume tokens.
- The edges represent dependencies.
- A token is a container in which amount of data can be stored.
- Tokens are consumed in the same order as they are produced, but a random access of data inside a token is allowed.
- An actor is enabled after a predefined number of tokens available on every input of an actor.
- An actor can fire (start its execution) after it is enabled. The number of token that must be available is specified next to the head of the data edges. The specified number of tokens is consumed from the input edges of the actor before the execution of an actor finishes that is within the response time of the actor.
- The number on the tail of an edge denotes the number of tokens an actor produces before the execution of the actor finishes.
- A self-edge of an actor is used to model that the previous execution must be finished before the next execution can start. This self-edge is given one initial token such that the next execution cannot start before the previous execution of the actor is finished.
- The response time of an actor is an upper bound on the time interval between the point in time that the actor is enabled ad that the point in time that the actor finishes its execution.

C. Scheduling policies over SDF

Even though the SDF model and its related models are widely adopted for algorithm specification in numerous DSP design environments, it has a couple of limitations to represent multimedia applications. First, the SDF model cannot express conditional execution of a block. Second, the model cannot use the shared data structure between blocks and pointer operations,

e.g. video frame blocks. Pointer operations to a shared global memory are prohibited by the SDF model to avoid side effects independently of node execution schedule.

Due to the explicitly parallelism representation of SDF, it is the most suitable to be used in our work as a model of computation as the best representing of the multimedia application dependencies. The scheduling in the mechanism that defines the task order during an execution, many scheduling policies are proposed in the literature, we cite them but consider only the most used between the SDF graph and network on chip architecture, in chapter 4 (session. 2), we detail those scheduling policies in order to define the mapping technique.

3. Network on chip delay model

In this section we present the mathematical delay model in a network on chip while we only consider during computing the transmission latency the wormhole switching technique that represent the adopted switching technique.

T	Number of multimedia application tasks.
C	Number of multimedia application channels.
NoC_{lat}	The network latency.
Ex_{ci}^{pj}	The execution time of task c_i on the processor p_j.
wt_{ci}^{pj}	The wait time of task c_i for the processor p_j.
n	The number of hops traversed by the packet from source to destination.
wl_j	The wait time of packet j for the link.
Com_j	The communication time , or the transmission time of packet j.
wb_j	The wait time on buffer of packet j.
R_j	The time it takes a router for transmission from the input port to the output port.
d(i)	The deadline of task i.
pr(i)	The priority of task i.
w(i)	The workload of task i.
$e_{i,j} \in E$	A communication channel between task i and task j.
$c_i \in C$	The task c number i.

$v(c_{i,j})$	The flow of the channel $c_{i,j}$.
$bw_{i,j}$	Or bw which is the bandwidth of the network, it can be uniform and non-uniform.
p_i	The processor i of NoC.
$l_{i,j}$	The NoC physical link that connects two nodes i and j.
f	The operating frequency.
$Nb.\Pr(i)$	The number of tasks that allocate the processor before task i.
Ex_{ck}^{pj}	The execution time of one of tasks c_k in p_j .
t_r	Time for routing decision for a packet.
t_w	Time spent by router for switching.
N_{flit}	The number of flits or the size of the packet
μ	The mean service rate
ρ	The traffic intensity
λ	The mean arrival rate
Avr_P	The average packet in the system
Z	The capacity of the buffer
Avr_{thr}	The average throughput of the system
Nb_{PE}	The number of processors in the system

Table 2.1. The parameters notation.

3.1 NoC latency

The time it takes to transmit, serve and execute all packets (or a flit) is defined by the sum of the execution time onto the allocated processors and communication time via links and it is given by:

$$\text{NoC}_{\text{lat}} = \sum_{i=1}^{T} \left(Ex_{ci}^{pj} + wt_{ci}^{pj} \right) + \sum_{j=1}^{C} \left((n-1) \times \left(wl_j + Com_j \right) + N \times (wb_j + R_j) \right) \quad (1)$$

The first part of the equation represents a computation time that is computed at the processor level and contains the execution time and the wait time of each task. The execution time is defined as the interval between the occupancy and the release of the processor, whereas

the wait time is defined as the interval between the arrival and the start of execution of the task. And they both are defined as:

3.2 Computation delay

The execution time Ex_{ci}^{pj} defined as the number of cycles it takes to execute task c_i on processor p_j and it is computed as follows:

$$Ex_{ci}^{pj} = \frac{w(i)}{f} \tag{2}$$

The wait time is the time it takes for the packet while waiting for the processor to be free before being executed and it is equal to the sum of the execution time of all waited tasks. If the packet is arriving and the processor is free, the wait time is null. The wait time of task on the processor is computed as:

$$wt_{ci}^{pj} = \sum_{k=1}^{Nb.\Pr(i)} Ex_{ck}^{pj} \tag{3}$$

3.3 The communication delay

The second part of the equation 1 represents the communication time that is defined as the time it takes for a packet to be transmitted from source to destination. It contains the link wait time, the buffer wait time, the routing decision time and the communication time. Those are defined as:

Link wait time is the time of waiting into the buffer in case of non-availability of channels (or virtual channels).

The wait time in the buffer is the time it takes for storing the flits (if needs) before transmitting it.

The routing decision time is the time that the router takes to transmit the packet from the input port to the output port. And it computed as:

$$R_j = t_r + t_w \tag{4}$$

The communication time is the time it takes to send the packet through the link and it is computed as follows:

$$Com_j = \frac{N_{flit}}{bw} \tag{5}$$

3.4 The average number of packets in the system

Given the mean arrival rate λ , the mean service rate is:

$$\mu = R_j + \left\lceil \frac{N_{flit}^{j}}{bw} \right\rceil \tag{6}$$

The occupancy of the buffer by the flit or the traffic intensity is calculated as follows:

$$\rho = \frac{\lambda}{\mu} \tag{7}$$

Given Z the capacity of the buffer, the average number of packets in the system Avr_P is given by:

$$Avr_P = \frac{(1-\rho)\rho^{Z}}{1-\rho^{Z+1}} \tag{8}$$

3.5 Network on chip average throughput

The average throughput Avr_{thr} is the rate at which the network can successfully deliver the injected packet is, and it is defined as:

$$Avr_{thr} = \frac{1}{Nb_{PE}} \times \sum_{j=1}^{C} N_{packet_j} \tag{9}$$

4. Network on chip power model

The concept of energy, initially introduced in the field of mechanics, refers to a scalar quantity that describes the capability of work that can be performed by a system. The intensity of the work, that is the energy used up per unit of time, is called power. Energy is thus the integral of power over time. There exist different forms of energy, such as potential energy, kinetic energy, thermal energy (heat), electrical energy, chemical energy, and radiation energy.

There exist many different units to measure the amount of energy. We will use the joule, which is the unit of energy in the mks (meter-kg-second) system. One joule (J) is defined as the amount of work done by a force of one Newton moving a mass of one kg for a distance of one meter. At the same time a joule is the amount of energy that is dissipated by the power of one electric watt lasting for one second.

The power model used to estimate the total power dissipated by a NoC for transferring a given packet is given by:

$$E_{NoC} = N_{task} \times E_{Ci}^{Pj} + N_{channel} \times E_{ei,j} \quad (10)$$

According to that model, each network component dissipates power to transmit a packet along the specified route, where: N_{task} is the number of tasks, $N_{channel}$ is the number of channels, E_{router} , E_{link} and E_{ci}^{pj} are the energy consumed during routing the packets in the router, transmitting it from source to destination and executing the task onto the appropriate processor respectively.

E_{Ci}^{Pj} is defined as the number of cycles it takes to execute c_i on processor p_j multiplied by the energy of one computational interval and calculated as:

$$E_{Ci}^{Pj} = r_j^i \times f_p \times E_{Cip}^{Pj} \quad (11)$$

r_j^i is the execution time of the task c_i on processor p_j , f_p is the operating frequency. E_{Cip}^{Pj} is energy of one computational interval equals to(with $\sigma_p = 0.2$)

$$E_{Cip}^{Pj} = V_{dd}^2 (1 + \sigma_p) \frac{\alpha_p}{2} C_p \quad (12)$$

The energy consumed due to sending the message $v(c_{i,j})$:

$$E_{ei,j} = NB\_F \times \frac{v(c_{i,j})}{f\_z} \times E_{flit} \quad (13)$$

$$E_{flit} = n_r \times E_{router} + (n_r - 1) \times E_{link} \quad (14)$$

$E_{ei,j}$ is energy consumed due to sending one message, E_{flit} is the energy consumed during sending one flit, f_z is the flit size, NB_F is the number of flits, E_{router} , E_{link} are the energy consumed by the flits while crossing the router and the link respectively. n_r is the number of hops traversed. The following parameters obtained from the Predictive Technology Model (PTM [46]), and [47][48][49].

Variable	Value
width or w (µm)	0.80
space or s (µm)	0.80

thickness or h(µm)	1.25
height or h(µm)	0.65
R_{wire} (Ω /mm)	30
$R_{gate} C_{gate}$ (ps)	10
Length or l (µm)	4
C_g (fF)	0.294492
C_C (fF)	0.203828
C_{total} (fF)	0.99664
C_{wire} (pF/mm)	0.3
K1	0.053
V_{bs}	0
K2	0,153
K4	1.63
V_{th1}	0,244
K5	3.65
K3	3.0×10^{-9}

Table 2.2. Parameters under 0.18um technology node.

4.1 Link power model

In order to calculate the energy consumed in an interconnect wire, a study of its physical characteristic, and its electrical behavior has been done [48] [49]. As the wire got longer, repeaters were used to minimize the energy consumption. The power equation for one gate driven wire is:

$$P_{link} = \frac{1}{2} CV^2 \alpha_l f + \tau \alpha VI_{short} f + VI_{bias,wire} + VI_{leak,gate} \quad (15)$$

The first component is the dynamic power consumption caused by charging and discharging of capacitive load on the gate's output, C is the load capacitance calculated as [50]:

$$C_{total} = MC_L + (M+1)C_c \quad (16)$$

$$C_g = \varepsilon \left[\frac{w}{h} + 2.04 \left(\frac{s}{s + 0.54h} \right)^{1.77} \left(\frac{t}{t + 4.53h} \right)^{0.07} \right] \quad (17)$$

$$C_c = \varepsilon\left[1.41\frac{t}{s}e^{-\frac{4s}{s+8.01h}} + 2.37\left(\frac{w}{w+0.31s}\right)^{0.28}\left(\frac{h}{h+8.96s}\right)^{0.76}e^{-\frac{2s}{s+6h}}\right] \quad (18)$$

C_l is the load capacitance and $(M+1)C_c$ the coupling capacitance of M neighboring wires. V is the supply voltage, α_l is the switching activity of the gate and f is the operating frequency of the system.

$\tau\alpha VI_{short}f$ is the short time period during I_{short} flow, and $I_{bias,wire}$, $I_{leak,gate}$ are the drain and source to body junction leakage currents in the NMOS device. Transistor subthreshold leakage is modeled by:

$$I_{short} = K_3 e^{K_4 V_{dd} K_5 V_{bs}} \quad (19)$$

V_{bs} is the voltage applied between the body and the source of the transistor. K_3, K_5 are constant fitting parameters, the other parameters depend on the technology used.

$$Wire_{lengh(l)} = 3\sqrt{\frac{R_{gate}C_{gate}}{R_{wire}C_{wire}}} \quad (20)$$

4.2 Router power model

The total energy consumption in a router for the transfer of one flit is given as follows:

$$E_{router} = E_{FIFO} + E_{Arbiter} + E_{Crossbar} \quad (21)$$

$E_{Arbiter}$ is the energy consumed due to setting up of a path for the packets to traverse from the input port to the output, and it is calculated as follows.

$$E_{Arbiter} = E_{Internal\ arbiter} + E_{switching\ arbiter} + E_{leakage\ arbiter} \quad (22)$$

$E_{crossbar}$ is the energy consumed due to routing the packets from input ports to the output, and it is calculated as follows:

$$E_{Crossbar} = E_{Internal\ Crossbar} + E_{Switching\ Crossbar} + E_{Leakage\ Crossbar} \quad (23)$$

E_{FIFO} is the energy consumed due to read and write accessing at each FIFO and it is computed as follow:

$$Energy_{FIFO} = P_{write} + P_{read} + P_{clk} + P_{\text{int}} \quad (24)$$

$$P_{write} = r_w P_{control} + \alpha P_{store} \quad (25)$$

Here, r_w is the rate at which write actions occur, $P_{control}$, P_{store} are the average power consumed at the control unit for one write action and the power consumed to store the data in the FIFO respectively.

$$P_{read} = r_r \; P_{control} + \alpha \; P_{retrieve} \tag{26}$$

Here, r_r is the rate at which read actions occur, $P_{control}$, $P_{retrieve}$ is the power consumed at the control unit and the power consumed to retrieve the data from the FIFO respectively.

P_{clk} Clock activity causes permanent constant power consumption due to switching activity.

P_{int} Which is the result of the internal short circuits occurring during switching of bits in the incoming data, the switching of read and write actions and clock activity. Therefore Pint can be calculated as follows:

$$P_{\text{int}} = r \; k_1 + k_2 \; \alpha + k_3 \tag{27}$$

Here, r is the rate of read and write actions, α is the amount of bit flips, the constants k_1 , k_2 are respectively the average internal power consumed for the control of read and write actions and due to bit changes in data and k_3 is due to the clock activity.

5. Power reduction techniques

Two components constitute a network on chip power dissipation: dynamic switching power and static, or leakage power, due to that, a classification of the optimization technique is done based on the type of dissipated power.

5.1 Techniques for reducing dynamic power

The equation $P = \frac{1}{2}\alpha C_l V^2 f$ represents the dynamic power of a circuit in which all the transistors switch once per clock cycle. Many researchers minimize or reduce the system energy by exploiting the characteristics of each element of the dynamic power equation.

5.1.1 Gate sizing

The power dissipated by a gate is proportional to its capacitive load C_l whose main components are:

- Output capacitance of the gate itself (due to parasitic.)
- The wire capacitance.
- To input capacitance of the gates in its fanout.

The output and input capacitances of gates are proportional to the gate size. Reducing the gate size reduces its capacitance, but increases its delay. Therefore, in order to preserve the timing behavior of the circuit, not all gates can be made smaller; only the ones that do not belong to a critical path1 can be slowed down. A critical path is with a slack equal to zero. The slack is the rest between the arrival time and the required time for a gate under a given delay.

The calculation of slack time can now be formulated. For an input vector v let $AT(g_j, v)$ be the arrival time of the gate g_i and $RT(g_j, v)$ be the required time of gate g_i under a given delay constraint. The time slack of gate with respect to the input vector is given by:

$$slack(g_i, v) = RT(g_i, v) - AT(g_j, v) \quad (28)$$

All gates with slack time greater than zero are candidates for down-sizing.

5.1.2 Clock gating

Clock signals are omnipresent in synchronous circuits. The clock signal is used in a majority of the circuit blocks, and since it switches every cycle, it has an activity factor of 1. Consequently, the clock network ends up consuming a huge fraction of the on-chip dynamic power. Clock gating has been heavily used in reducing the power consumption of the clock network by limiting its activity factor. Fundamentally, clock gating reduces the dynamic power dissipation by disconnecting the clock from an unused circuit block.

Clock power is a major component of total power consumption, which makes clock-gating a very important power saving technique. Register-transfer level (RTL) synthesis tools are capable of finding the clock-gating opportunities for a design, if the RTL model conforms to certain modeling guidelines. In [51] an algorithm for clock gating to automatically generate clock-gated RTL after High-level synthesis (HLS) is presented; since HLS is becoming increasingly important, there is a need to push such lower-level reduction techniques to a higher-level.

5.1.3 Sequentiel clock-gating

In traditional clock-gating reduction technique, clock-toggles and inconsequential computation of the registers could occur. For that a sequential clock-gating [52] aim at eliminating that problem providing power saving control information across clock boundaries. Usually sequential clock-gating opportunities are discovered manually based on certain

characteristics of a design (e.g. pipelining). Since the manual addition of sequential gating circuitry might change the functionality of the design, sequential equivalence checking is needed after such changes. Tools (Verification of optimizations), for sequential equivalence checking are expensive, and based on recent technologies. Therefore, it is desirable to automate the discovery of sequential clock-gating opportunities using already existing and proven technologies such as model checking and thereby a priori proving that the changes will not affect the required functionality. Model Checking Based Sequential Clock Gating (MCBCG) method, formally proves particular sequential dependencies of registers on other registers and logic, thus sequentially gating such registers will not require further validation

5.1.4 System level simulation guidedhls approach to generate clock-gated rtl

HLS tool is informed to insert clock-gating for specific parts of the design or the whole design statically and this technique is very time consuming. Also clock-gating may not save dynamic power all the time, especially for finer granularities. This savings is also dependent on the stimulus. In current state of the art methods it is very difficult because the power estimation is very time consuming and clock-gating is performed at a net list level, making a decision making task very difficult. Therefore, in [53] CoDel was proposed in order to allow hardware description at the algorithm level and thus dramatically reduces design time, Power analysis is used to compare the effectiveness of CoDeL's automated clock gating to automated clock gating using Synopsys tools. The results show that a combination of CoDeL and Synopsys clock gating provides 16% more power savings, on average, than Synopsys' automated clock gating alone. Finally, the CoDeL platform is compared to a modern DSP processor, and it is found that the CoDeL platform produces designs with somewhat slower run times, but dramatically lower power dissipation. Also in [54] propose a novel system-level design methodology, which utilizes a `relative power reduction model' that can help in predicting the impact of clock-gating on each register/bank quickly and accurately, by simulating the design at a cycle accurate transaction-level. As a result, our approach can automatically find the appropriate registers to clock-gate, guided by the system-level simulation.

5.1.5 Voltage and frequency scaling

Dynamic power is proportional to the square of the operating voltage. Therefore, reducing the voltage significantly improves the power consumption. Furthermore, since frequency is directly proportional to supply voltage, the frequency of the circuit can also be lowered, and thereby a cubic power reduction is possible. An example is shown that the Intel XScale® processor can dynamically operate over the voltage range of 0.7–1.75 V and at a frequency

range of 150–800 MHz. The highest energy consumption is 6.3 times of the lowest energy consumption. Voltage scaling can be performed in the interval V threshold, V normal. Since V normal is reduced as a device is scaled to lower dimensions, the range that is available for voltage scaling in sub-micron devices is reduced and voltage scaling becomes less effective. However, the delay of a circuit also depends on the supply voltage as follows:

$$\tau = k.C_l.\frac{V_{dd}}{(V_{dd} - V_{th})^2} \tag{29}$$

Where τ is the circuit delay, k is the gain factor, C_l is the load capacitance, V_{dd} is the supply voltage, and V_{th} is the threshold voltage. Thus, by reducing the voltage, we can achieve the power reduction, but the execution time increases. The main objective in achieving power reduction through voltage and frequency scaling is to obtain power reduction while meeting all the timing constraints.

Simple analysis shows that if there is a remaining time in execution time witch called slack, it would be better to run the processor as slow as possible, while meeting the timing constraints is more dynamic-power-efficient than executing as fast as possible and then idling for the remaining time. This is the main idea that is used in to exploit the hardware characteristic for reducing the power.

One other method to recover the lost performance is by scaling down the threshold voltage to the same extent as the supply voltage. This allows the circuit to deliver the same performance at a lower V_{dd} . However, smaller threshold voltages lead to smaller noise margins and increased leakage current. Furthermore, this cubic relationship holds only for a limited range of V_{th} scaling. The quadratic relationship between energy and V_{dd} deviates as V_{dd} is scaled down into the sub-threshold voltage level. In the sub-threshold region, while the dynamic power still reduces quadratically with voltage, the sub-threshold leakage current increases exponentially with the supply voltage. Hence dynamic and leakage power become comparable in the sub-threshold voltage region, and therefore, "just in time completion" is not energy inefficient. In practice, extending the voltage range below half V_{dd} is effective, but extending this range to sub-threshold operations may not be beneficial.

a. Design-time voltage and frequency scaling

One of the most common ways to reduce power consumption by voltage scaling is that during design time, circuits are designed to exceed the performance requirements. Then, the

supply voltage is reduced so as to just meet the performance constraints of the system. This is also called design-time voltage and frequency scaling. Design-time schemes scale and set the voltage and frequency, which remains constant (and therefore inefficient) for all applications at all times.

b. Static voltage and frequency scaling

Systems can be designed for several (typically a few) voltage and frequency levels and those levels can be switched at run time. In static voltage and frequency scaling, the change to a different voltage and frequency is pre-determined; this is quite popular in embedded systems. However, there are significant design challenges in supporting multiple voltages in CMOS design. Timing analysis for multiple voltage design is complicated as the analysis has to be carried out for different voltages. This methodology requires libraries characterized for the different voltages used. Constraints are specified for each supply voltage level or operating point. There can be different operating modes for different voltages. Constraints need not be same for all modes and voltages. The performance target for each mode can vary. Timing analysis should be carried out for all these situations simultaneously. Different constraints at different modes and voltages have to be satisfied.

c. Dynamic voltage and frequency scaling

The application and system characteristics can be dynamically analyzed to determine the voltage and frequency settings during execution. For example, adaptive scaling techniques make the decision on the next setting based on the recent history of execution. In follow a state of art of the evolution of DVFS.

- In 2007, Lee et al. emphasized on the need of a Power Management Unit (PMU) that controls the generation of the supply voltage and clock to fine-tune the speed of the target domain. The occupational level and performance required of each domain are generated to the PMU that decides the upgrade of frequency and voltage for each of the power domain. V/F line usage by the PMU causes the system to lock the V/F line. The lock essentially discourages multi-threading [55].
- Also in 2007, Malkowski et al. exploits the fact that the memory bound code can be power optimized for achieving energy saving. As soon as the application enters a memory bound phase, pre-fetchers are activated. In the memory bound phase, the execution time is dominated by the latency caused in the access to the main memory. Applications in the memory bound phase tend to be idle, waiting for the required set of data. Moreover, in the memory bound phase, a reduction in the

CPU voltage and frequency does not lower the performance. Therefore, power optimization can be achieved [56].

- In 2009, Beigne et al. proposed a fully power-aware locally-synchronous and globally-asynchronous NoC circuit for the implementation of DVFS mechanism. Adaptive design techniques are used for each of the synchronous NoC units. The main CPU is required to directly disable/enable the units for entering the power off mode and executing the reset phase. A multiple supply line for V/F decreases the voltage frequency transition delay. Therefore, the system becomes more time efficient [57].
- In 2011, Per-core DVFS if applied to chips achieves better application performance and control accuracy as compared to per-chip DVFS. It is known that DVFS can allow a cubic reduction in power density for each of the cores in a Chip Multi-processor (CMP) employing DVFS. The performance of the CMP can further be enhanced by minimizing the transition time of V/F scaling. Inter band Tunnel Field Effect Transistors (TFETs) cores are more energy-efficient at low frequencies as compared to the CMOS cores. Therefore, combining the two types of cores in a multi-core architecture and applying DVFS for V/F scaling, a thread migration scheme is proposed. Low frequency applications are migrated to the TFET cores, while high frequency applications are more efficiently handled by the CMOS core. The scheme achieves average dynamic energy and leakage energy savings of 17% and 30%, respectively. However, the scheme has an overhead of performance degradation of 1% [58].
- In 2012, Herbert et al. proposed a process variation parameter to shift work to efficient dies or cores. The work shifting can be divided into two levels: (a) among dies belonging to a given speed bin and (b) between VFI on a given die. The traditional DVFS schemes have neglected the variations in both the spatially-correlated within-die process and die-to-die, which if addressed can improve energy efficiency [59].
- In 2014, Wang et al. [60] proposed a power-efficient network calculus-based (PNC) method to minimize the power consumption of NoC. Based on the slack that a packet (message between cores are split into several packers) can be further delayed in the network without violating its deadline, where PNC method uses power-gating technique to reduce the active buffer size and uses voltage-frequency scaling technique to reduce the voltage-frequency of each voltage-

frequency island. With less active buffer units and lower voltage- frequency, the power consumption of NoC is reduced. Experimental results show that our PNC method can save at most 50% of the total power consumption.

- Also in 2014, Zhan et al. [61] proposed a methodology to minimize the energy consumption of NoC without violating the pre-specified latency deadlines of the real-time applications, where recognizing the distribution of slacks for different traffic streams, and assigns different voltages and frequencies to different routers to achieve NoC energy-efficiency, while meeting the deadlines for all packets. Furthermore, a feedback-control strategy is designed to enable dynamic frequency and voltage scaling on the network routers in conjunction with the energy optimization algorithm. It can flexibly improve the energy-efficiency of the overall network in response to sporadic traffic patterns at runtime.

5.2 Techniques for reducing short circuit power

Short circuit power is directly proportional to rise time and fall time on gates. Therefore, reducing the input transition times will decrease the short circuit current component. However, propagation delay requirements have to be considered while doing so. Short circuit currents are significant when the rise/fall time at the input of a gate is much larger than the output rise/fall time. This is because the short circuit path will be active for a longer period of time. To minimize the total average short circuit current, it is desirable to have equal input and output edge times. In this case, the power consumed by the short circuit current is typically less than 10% of the total dynamic power. An important point to note is that if the supply is lowered to below the sum of the thresholds of the transistors $V_{dd} < VT_n + |VT_p|$, the short circuit currents can be eliminated because both devices will never be on at the same time for any input voltage value.

5.3 Techniques for reducing leakage power

In the follow some of techniques to reduce leakage power are discussed.

5.3.1 Multiple supply voltage

The multiple supply system provides a high-voltage supply for high-performance circuits and a low-voltage supply for low-performance circuits. In a dual Vdd circuit, the reduced voltage (low-V_{dd}) is applied to the circuit on non-critical paths, while the original voltage

(high-V_{dd}) is applied to the circuit on critical paths. Since the critical path of the circuit is unchanged, this transformation preserves the circuit performance. To apply this technique, the circuit is typically designed using high-V_{dd} gates at first. If the propagation delay of a circuit path is less than the required clock period, the gates in the path are given low-V_{dd} . In an experimental setting [62], the dual V_{dd} system was applied on a media processor chip providing MPEG2 decoding and real-time MPEG1 encoding. By setting high-Vdd at 3.3 V and low-Vdd at 1.9 V, system power reduction of 47% in one of the modules and 69% in the clock distribution was obtained.

5.3.2 Multiple threshold voltage

Multiple V_{th} MOS devices are used to reduce power while maintaining speed. High speed circuit paths are designed using low-V_{th} devices, while the high-V_{th} devices are applied to gates in other paths in order to reduce subthreshold leakage current. Several design approaches have been proposed for dual-V_{th} circuit design. One approach builds the entire device using low-V_{th} transistors at first. If the delay of a circuit path is less than the required clock period, the transistors in the path are replaced by high-V_{th} transistors. The second approach allows all the gates to be built with high-V_{th} transistors initially. If a circuit path cannot operate at a required clock speed, gates in the path are replaced by low-V_{th} versions.

5.3.3 Adaptive body biasing

One efficient method for reducing power consumption is to use low supply voltage and low threshold voltage without losing performance. But an increase in the lower threshold voltage devices leads to increased sub threshold leakage and hence more standby power consumption. One solution to this problem is adaptive body biasing (ABB). The substrate bias to the n-type well of a pMOS transistor is termed V_{bp} and the bias to the p-type well of an nMOS transistor is termed Vbn. The voltage between V_{dd} and V_{bp} , or between GND and Vbn is termed V_{bb} . In the active mode, the transistors are made to operate at low-V_{dd} and low-V_{th} for high performance. The fluctuations in V_{th} are reduced by an adaptive system that constantly monitors the leakage current, and modulates V_{bb} to force the leakage current to be constant. In

the idle state, leakage current is blocked by raising the effective threshold voltage V_{th} by applying substrate bias V_{bb} . The ABB technique is very effective in reducing power consumption in the idle state, with the flexibility of even increasing the performance in the active state. While the area and power overhead of the sensing and control circuitry are shown to be negligible, there are some manufacturing-related drawbacks of these devices [63]. ABB requires either twin well or triple well technology to achieve different substrate bias voltage levels in different parts of the IC. Experiments applying ABB to a discrete cosine transform processor reported a small 5% area overhead. The substrate-bias current of V_{bb} control is less than 0.1% of the total current, a small power penalty.

5.3.4 Power gating

Power Gating is an extremely effective scheme for reducing the leakage power of idle circuit blocks. The power (V_{dd}) to circuit blocks that are not in use is temporarily turned off to reduce the leakage power. When the circuit block is required for operation, power is supplied once again. During the temporary shutdown time, the circuit block is not operational – it is in low power or inactive mode. Thus, the goal of power gating is to minimize leakage power by temporarily cutting-off power to selective blocks that are not active. Since the power gating transistors are rather large, the slew rate is also large, and it takes more time to switch the circuit on and off. This has a direct implication on the effectiveness of power gating. Since it takes a long time for the power-gated circuit to transition in and out of the low power mode, it is not profitable to power gate large circuits for short idle durations. This implies that either we implement power gating at a fine granularity, which increases the overhead of gating, or find large idle durations for coarse-grain power gating, which are fewer and more difficult to discover. In addition, coarse-grain power gating results in a large switched capacitance, and the resulting rush current can compromise the power network integrity. The circuit needs to be switched in stages in order to prevent this. Finally, since power gates are made of active transistors, the leakage of the power gating transistor is an important consideration in maximizing power savings.

5.3.5 Segment gating

The author in [64] finds that a holistic approach that includes links, portions of the crossbar, arbiters, and buffers yields an appreciable reduction in static power consumption over previous schemes. The author's approach was motivated by the observation that bandwidth is

abundant in on-chip communication interconnects, leading to underutilization of available network resources on many workloads. This presents an opportunity to turn off links with little or no impact on the dynamic energy and latency on the rest of the network. Along with the link, arbitration logic, switch capacity, and buffering at the output port of the upstream node and input port downstream may be powered down as well. There is potential to save leakage energy for up to 99% of total cycles.

5.4 Comparison between different power reduction techniques

Table 2.3 summarizes the research aiming at reducing the power consumption in NoC under timing constraints.

Works	Power Reduction	Other comments	Parameters to reduce
[51]	25%	Area 3%	Clock/ α
[52]	30%	Nothing	Clock/ α
[53]	16%	Area 20% Timing 20%	Clock/ α
[55]	63%	Nothing	V_{th} and f
[56]	39%	Nothing	V_{th} and f
[59]	9.2%	Throughput 5.7 % Power/ Throughput 7.7 %	V_{th} and f
[57]	22-86%	Leakage power saving	V_{th} and f
[60]	50%	Nothing	V_{th} and f

Table 2.3. Comparison between different power reduction techniques.

6. Conclusion

Multimedia applications have multi-application scenarios that are embedded onto multiprocessor where each task of the corresponding graph of the scenario is affected to one of the chip tiles, for that the application mapping determines the placement of IP cores into the NoC tiles in such a way that the performances are optimized. It is considered as NP-hard problem where the search space of the problem increases factorially with the system size and due to that an effective optimization algorithm is required. The next chapter is about the mapping strategies with the optimization algorithm that are the most defined in literature.

CHAPTER III

HEURISTIC OPTIMIZATION ALGORITHMS SPECIFIED IN NETWORK ON CHIP

1. Introduction

Optimization problems are concerned with finding the values for one or several variables that meet the best objectives without violating the constraints if it is considered as hard, and allows small violating if it is considered as soft. But before defining the optimization methods needed to solve the given problems, it might be helpful to find a classification for its size and its complexity.

The computational complexity of an optimization problem as well as optimization algorithms is given by $O(\bullet)$ which indicates the asymptotic time needed to solve the problem when it involves n variables and the problem size is determined by the number of these variables and the complexity becomes $O(n^2)$ and is therefore quadratic in n. A real improvement of the complexity can only be achieved when a better algorithm is found. This

applies not only for the polynomial problems where the number of computational increases factorially with the system size. This chapter is divided into three parts. In the first part, we talk about the problem to be dealt with, considering the routing algorithm, the switching strategy technique, the task migration, and the dynamic voltage/frequency scaling. In the second part, we talk about the heuristic optimization methods for NoCs, and in the last part we define genetic algorithm, cellular automata, and quantum computing.

2. Problems and proposed contributions

2.1 Routing and switching strategy

The design of network on chip without buffers can be a suitable solution for communication complexity, such as saving the energy consumed during the actions occurred in buffers, but it will cause a long latency and also an increased misrouting of packets. In this session, we focused on the design issues in using bufferless routing in NoC while motivated by the use of small FIFO registers; the idea is to use bufferless routing while guaranteeing minimal latency and greater throughput. The bufferless routing is a new randomized routing algorithm with a new format of wormhole flits that use a small FIFO registers. Also, we had proposed a randomized routing algorithm that works in two steps; the first is to determine the deadlock factors using a randomized routing algorithm while the second is to choose the direction with the minimum deadlock factor value.

The Buffers are important design factors of modern high performance in on-chip networks because they simplify the transmission and prevent packet loss, but also they consume a significant amount of energy that increases the complexity of the design.

In general, for NoC the memory represents one or many storage elements, can be DRAM or SRAM or registers. In [135] the author proved that DAMQSV and DAMQS are excellent schemes to optimize buffer management providing a good throughput when the network has a larger load. They can utilize significantly less buffer space without sacrificing the network performance. In [136], the author proposed a hybrid design of input buffers using both SRAM and STT-MRAM to hide the long write latency efficiently exploiting the benefits of STT-MRAM that is a near to zero leakage power with optimizing its drawback since the buffer scheme is more energy consumption and long latency. Simulation results enhance the throughput by 21% on average. In [137], it is shown that in on-chip interconnection networks can be designed with no buffers in order to minimize as possible the overall energy, but this technique results many disadvantages on the network latency with deadlock and livelock occurring. The bufferless routing requires a small packet injection rate with a maximum

network utilization where the latter is tracked by the use of Credit-based, once the downstream node detect free buffer, a packet injection is required. Hence, bufferless routing can be a promising approach for on-chip networks that primarily operate at a low packet injection rate.

The use of buffers allows a better routing decision for the head flit and due to that the misrouting is minimized as well, but a significant amount of energy is consumed in result of the read, write and the control actions, also a portion of energy is leaking which is the result of the internal short circuits occurring during switching of bits in the incoming data, the switching of read, write actions and clock activity.

The idea of bufferless routing has been to always route the flit into an output port without getting into the wait time where the head flit can decide which direction is the better. Each flit is routed independently of other flits through the network, and different flits from the same packet may take different paths. A rank is assigned to each flit in case when there is contention between multiple flits that are destined for a particular direction, the algorithm decides which one is high ranking.

The traditional wormhole is where each message consists of several packets and the packet is encapsulated into several flits. The first flit processed the information about the destination that helps in setting up the path for the flits that are followed in a pipelined fashion. Once the transmission is completed the tail-flit free the channels occupied by the header flit. In bufferless wormhole, each flit needs to be head flit because in case of the head-flit is deflected, the entire packet would follow this head-flit and would be deflected, and that will have a negative impact on packet latency. One solution is to increase the width of the links, but that will increase the energy consumption of links [137].

Beside the bufferless routing shows many advantages in reducing complexity, energy consumption, simplifying and reducing the router Latency with area saving, but only under lower average injected traffic into the network, otherwise, if the network saturation point was passed, a live-lock and deadlock will absolutely occur. However, because our objective is to reduce the energy consumption under multimedia application timing constraints, the pure bufferless routing is not a promising solution, nor on optimizing our application performances, nor on respecting the wormhole routing, for that we proposed a new scheme of wormhole that is presented in chapter.4 session 3.

2.2 Task migration

2.2.1 Definition of task migration

Task migration [140] [141] is the act of transferring a process between two processing units, it is considered as an effective strategy to achieve load balancing and high resource utilization, in another way, it is a call to the operating system during run-time. The call can be made from both the source and the destination, but all migration callers will be treated as a generic source regardless of the layer. Task migration is usually started at selected checkpoints in the program, when a call is made and the migration is granted by the target, the operating system stops the currently running task by suspending it. The data associated with the task is stored and moved onto the target core by the migration mechanism. The operating system can, if the migration was successful, manually switch in a new task for processing. If the manual context switch was not made, the operating system will switch in the new task after the next system tick.

2.2.2 Reasons for task migration

Tasks are migrated between cores in order to achieve dynamic load distribution, ensure fault resilience, provide system administration, minimize energy consumption and thermal chip management [142],[143]. A fully loaded CPU core can migrate tasks to a lightly loaded core to increase the overall system performance. Performance improvements will show clearly in situations when a single core is loaded over its capacity as in the figure 3.1. A less loaded core can be handed some of the first core's tasks in order to increase the overall performance in the system.

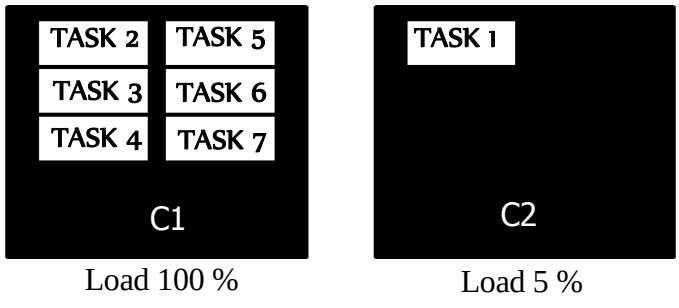


Figure 3.1. Example1: Load representation on two CPU cores.

Task migration can also be used to migrate all tasks to one single core in order to decrease the energy consumption, which is shown in Figure 3.2. A CPU core consumes more energy fully loaded than it does idle. An energy saving strategy comprises shutting down idle cores, and keeping them shutdown until they are needed. When a core is shut down it consumes the

least energy [147]. Two cores with light load could, combine the load into a single core with all the tasks running on it, while the other core could be shut down.

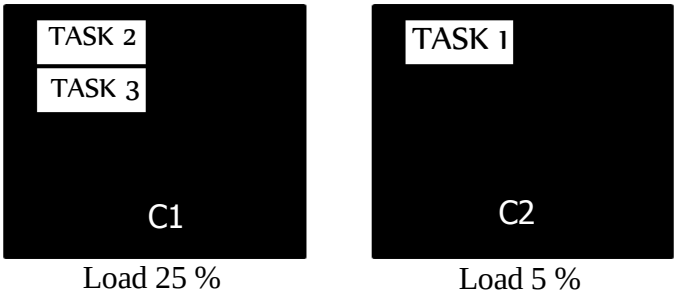


Figure 3.2. Example2: Load representation on two CPU cores.

2.3 Dynamic voltage/frequency scaling

An Inter-DVFS algorithm has two parts slack estimation and slack distribution. The slack estimation method for mixed task set varies from that of the periodic task set. Periodic tasks are known prior about their occurrence and their slack can be determined statically. Unlike periodic tasks, the arrival time of aperiodic tasks is not fixed and hence, they must be executed with full frequency in order to achieve better response times. The slack distribution method is simple and greedy in nature, as all the available slack time is provided for the next ready task.

The DVFS is an efficient technique that plays a key role in the power management; it saves the power by lowering the operating voltage and clock-speed of a processor in real-time [145] [146] [147] to meet the executed task timing constraint. By other way, the main characteristic used in the implementation of the DVFS is running the processor with the sufficient clock frequency and voltage that it required to be operated.

The implementation of the DVFS can be fragmented into two parts: (a) the adjustment of the required frequency and (b) the scaling of the applied voltage to a core on the basis of the workload being processed. In the modern CPU, the processor stalls when a data request is not entertained. The stall is an indication of the memory bound phase and is an area of focus pertaining to the energy conservation of the processors. The reduction in the power consumption of the electronic devices is desirable because it decreases the temperature of the devices and increases the time elapsed between the charges [148].

The DVFS is an operative technique used to improve the energy efficiency of the microprocessors. The DVFS achieves the energy efficiency by minimizing the applied voltage and clock-speed, when the workload decreases. When the applications are not sensitive to the frequency provided, for instance, in the memory bound state, the DVFS can be applied to reduce the power budget. As the voltage and frequency are assumed to be linearly dependent on

each other, reducing the voltage reduces the frequency. The phenomenal property of balancing the trade-off between power consumption and performance has made the DVFS, a de facto choice of the multi-core designs.

David et al. [149] exploited the above mentioned benefits of the DVFS in the design of the Single-Chip Cloud Computer (SCC). The power management using the Voltage/Frequency Islands (VFI) was performed in real-time and the result assured that the aforementioned approach is almost two times better, as compared to the peer energy-saving techniques.

Wang et al. [150] proposed an adaptive power control algorithm for the implementation of the DVFS. Moreover, the temperature control and cache re-sizing enhance the efficacy of the proposed algorithm. Nevertheless, no power cap is implemented to confine the maximum power drawn.

For the single-threaded applications, a control known, along with the DVFS are used to maximize the efficiency of the multi-core architecture. The thread packing specifies the quantity of threads running per core and uses the power cap mechanism to maximize the performance of the multi-threaded workloads with a varied number of cores. The thread reduction, coupled with the power cap mechanism, limits the maximum power that a particular core can use at any given instance of the time. Seo et al. [145] had also adopted the power capping mechanism using the DVFS to restrict the power consumption in the mobile devices and achieved a 10% saving in the energy consumption.

Jin et al. [148] accomplished 61.69% saving in the energy consumption, using the Hilbert Transform and the DVFS to implement the workload estimation model. A high accuracy and a negligible deadline miss ratio were attained. Nonetheless, the proposed design is only viable for the video encoding applications in the mobile devices.

Gorti and Somani [151] developed a thermal and performance management technique to improve chip lifetime and leverage application awareness. Chip heating is overcome by the intelligent selection of voltage-frequency (V, F) pair using the DVFS. The level of the (V, F) pair is chosen in accordance with the operating point (OP) of the core. Significant performance improvement and energy savings are obtained using the aforementioned scheme. Moreover, the reliability of the chip is increased by a factor of sixteen fold.

3. Heuristic optimization methods for NoCs

3.1 Preliminaries

Optimization is an intelligent selecting of the best alternative among a given set of options or in other way, it can be viewed as a decision making in order to optimize one or several

objectives according to the given scenarios. Optimization problems arise in various disciplines such as engineering design, manufacturing system, economics etc. Thus, in view of the practical utility of optimization problems there is a need for efficient and robust computational algorithms which can numerically solve with computers the mathematical models of medium as well as large size optimization problems arising in different fields. In this section, we will define some preliminaries related with the heuristic optimization.

3.1.1 Objective function

The objective function is in general in the form of an equation that need to be optimized and it is given based on specific constraints and based on the variables whose need to be minimized or maximized using nonlinear programming techniques.

3.1.2 Search space

The search space is in the form of a range or values, those values represent the variables or the space of all solutions that will be searched during optimization that are called search space.

3.1.3 Types of solutions

A solution to an optimization problem is the results of the objective function, it can be a feasible solution that satisfies all constraints or in other means; each point in the search space represents one feasible solution. Also, it can be an optimal solution that represents a feasible solution that provides the best objective function value. And the last one is the near-optimal solution that is also a feasible solution, but provides a superior objective function value, but not necessarily the best.

3.1.4 Constraints

Each constraint can be hard that is must be satisfied or it can be soft that is desirable to satisfy.

3.1.5 Heuristics

Heuristics are rules of searching or finding optimal or near-optimal solutions. Examples are FIFO, LIFO, earliest due date first, largest processing time first, shortest distance first, etc. Heuristics can be constructive while a solution is built piece by piece, or it can improvement where a better solution is searched based on primary used solution.

3.2 Characteristics of heuristic optimization methods

The main common characteristics of all heuristic optimization methods is that, they start with an initial solution, and iteratively produce new solutions by some generation rules and evaluate those new solutions, and in the final, we report the best solution found during the search process. The execution of the iterated search procedure is usually stopped or achieved in case of:

- No further improvement over a given number of iterations.
- When the required solution has been found.
- When the allowed CPU time (or other external limit) has been reached.

Heuristic optimization methods may differ substantially in their underlying concepts, the following is given the mains over them:

3.2.1 Generation of new solutions

A new solution can be generated by modifying the current solution or by building a new solution based on past experience or results. In doing so, a deterministic rule, a random guess or a combination of both can be employed.

3.2.2 Treatment of new solutions

In order to overcome local optima, heuristic optimization methods usually consider not only those new solutions that lead to an immediate improvement, but also some of those that are knowingly inferior to the best solution found so far. To enforce convergence, inferior solutions might either be included only when not being too far from the known optimum.

3.2.3 Limitation of the search space

Some methods exclude certain neighborhoods or regions or individuals to avoid cyclic search paths or spending too much computation time on supposedly irrelevant alternatives.

3.2.4 Prior knowledge

The prior knowledge is a guideline of making a good solution by storing previous generation or solutions; it can be incorporated in the choice of the initial solutions or in the search process. Though the inclusion of prior knowledge might significantly reduce the search space and increase the convergence speed.

3.2.5 Computational complexity

For heuristic optimization methods, the complexity depends on the costs of evaluating each solution, the number of iterations, and the population size.

4. Mapping and scheduling techniques in NoC

4.1 Key factors on task mapping

Many factors should be considered during the mapping of task onto NoC tiles, such as:

4.1.1 Target architecture

The target architecture is related to whether nodes in the NoC system are heterogeneous or homogeneous. Heterogeneity is the most common case, because this factor may improve system performance in the presence of different kinds of applications. Heterogeneity refers to having several kinds of nodes in the system (i.e., nodes may be different among them).

4.1.2 Abstraction level of the application specification

The abstraction level in which applications are described is a key factor in mapping tasks of such applications to the available resources. The first possible approach to this subject is to use Register Transfer Level, or RTL. RTL is a valuable tool for modeling and designing complex systems, and often relies on hardware description languages, such as VHDL (VHSIC Hardware Description Language) and Verilog. Such tools allow modeling a part of the NoC system such as the communication system [181]. The second reported approach is based on transaction-level modeling or TLM. Transactions are defined as the event of synchronization or data exchange among system modules. This approach is appealing because it allows performing a functional verification of the system, and the modeling is based on languages such as SystemC. TLM has been used successfully for synthesizing high speed MPSoC systems and for modeling the communications infrastructure of a NoC [181].

4.1.3 Figures of merit

This factor refers to the optimization criteria which must be considered along the optimization process related to the mapping stage. Such optimization can be viewed as a solutions space exploration, where each solution represents a single design choice with different values of the objective functions [181]. The task mapping process must find an acceptable solution within the space with allowable and optimized values for such functions. Among the most common figures of merit used for such optimization process, we may find: power consumption, delay time, mapping time, temperature, mean number of hops across the network, network contention, mean channel occupancy, bandwidth, and so on.

4.1.4 Common domain semantic

This is a medium level representation which combines information both from the high level application description and from the implementation platform. Among the plethora of

representations available for these purposes, graph-based approaches are the most common, with instances such as task graphs (TG), communication task graphs (CTG), communication weight graphs (CWG), communication resources graph (CRG), annotated task graphs (ATG), synchronous and asynchronous data flow graphs (SDFG and ADFG), and so on. Some other kinds of such medium– level representation are the Petri Networks (PN), and the Kahn Process Networks (KPN) [181].

4.1.5 Topologies

Topology refers to the way in which system nodes are physically interconnected. Topologies may be classified as either regular or irregular. Some instances of common topologies are meshes, torus, rings, and spidergon ones. Regular topologies are more constrained with respect to the distribution of the connections, which are generated by means of mathematical functions. Irregular Topologies are often the mixture of two or more regular forms, which leads to hybrid, hierarchical or totally irregular topologies.

4.1.6 Optimization algorithms

As already mentioned, the mapping stage relies on an optimization process, which searches along the solutions space, the design with a better tradeoff among the chosen figures of merit. The kind of optimization algorithm used for task mapping has a direct impact on the communications nature. For instance, off–line (static) optimization forces to having predictable communication assessments, whilst dynamic algorithms allow a more flexible communication scheme.

4.2 Related work of mapping and scheduling techniques in NoC

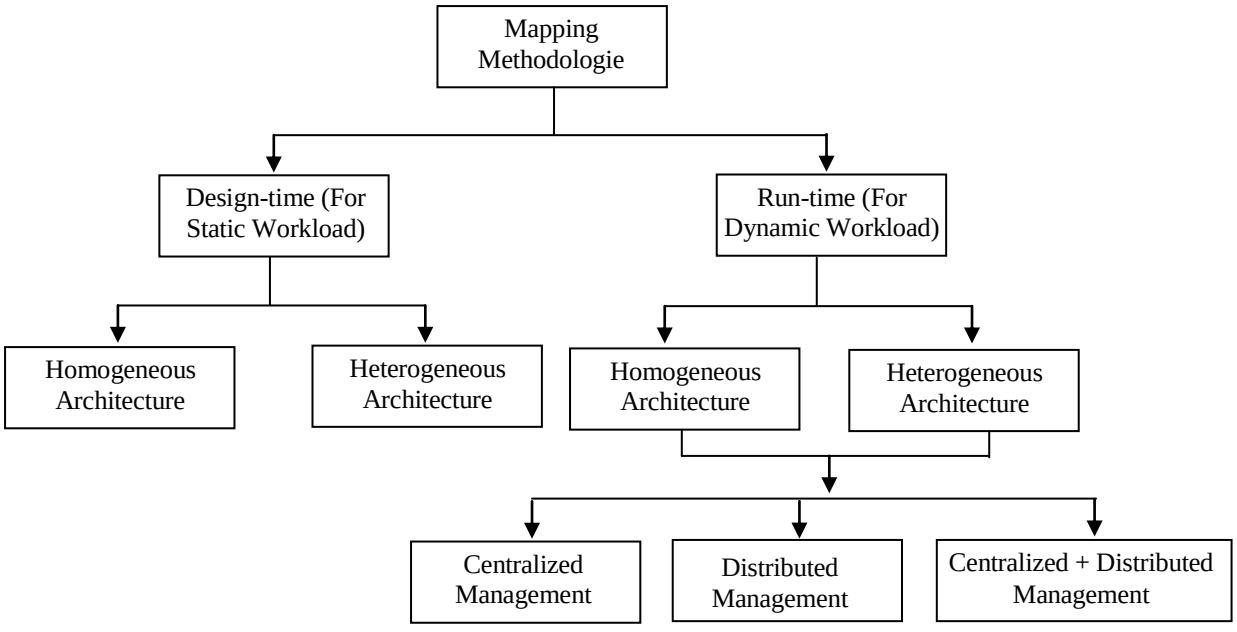


Figure 3.3. Taxonomy of Mapping Methodologies. [182]

There could be a number of taxonomies to classify the mapping methodologies, like target architecture based, optimization criteria based, workload based, etc. Broadly, the methodologies can be classified based on workload scenarios and other taxonomies can be included at some hierarchy in the classification as shown in Figure 3.3. For static and dynamic workload scenarios, the mapping methodologies perform optimization at design-time and run-time respectively, which has led them to classify as design-time and run-time methodologies respectively. The methodologies target either homogeneous or heterogeneous multi-core systems.

4.2.1 Design-time mapping

Design-time mapping methodologies have a global view of the system which facilitates in making better decision about using the system resources. As optimization is performed at design-time, the methodologies can use more thorough system information to make decisions. Thus, a better quality of mapping may be achieved as compared to the runtime mapping methodologies that are normally restricted to a local view where only the neighborhood of the task mapping is considered. Most of the mapping methodologies reported in the literature fall under design-time mapping, and aim at optimizing for difference performance metrics in order to fulfill the varying user demands. Different well established search approaches have been extensively used to develop design-time mapping methodologies in order to find optimal or near-optimal placement of tasks on platform cores towards improving the compute performance. In [151], a new parameter selection scheme for simulated annealing is proposed that sets task mapping specific optimization parameters automatically. The scheme bounds optimization iterations to a reasonable limit and defines an annealing schedule that scales up with the application and architecture complexity. The presented parameter selection scheme compared to extensive optimization achieves 90% goodness in results with only 5% optimization time, which helps large-scale architecture exploration where optimization time is important. The optimization procedure is analyzed with simulated annealing, group migration and random mapping algorithms using test graphs from the Standard Task Graph Set. Simulated annealing is found better than other algorithms in terms of both optimization time and the result. Simultaneous time and memory optimization method with simulated annealing is shown to speed up execution by 63% without memory buffer size increase. As a comparison, optimizing only the execution time yields 112% speedup, but also increases memory buffers by 49%. In [152], the author proposed a switch which employs the latency insensitive concepts and applies the round-robin scheduling techniques to achieve high communication resource utilization. Based on the assumptions of the 2D-mesh network topology constructed by the switch, its work

not only models the communication and the contention effect of the network, but develops a communication-driven task binding algorithm that employs the divide and conquer strategy to map applications onto the multiprocessor system-on-chip. The algorithm attempts to derive a binding of tasks such that the overall system throughput is maximized. To compare with the task binding without consideration of communication and contention effect, the experimental results demonstrate that the overall improvement of the system throughput is 20% during 844 test cases. In [153], the author proposed an evolutionary algorithm-based technique. To hide memory latency, prefetching is aggressively performed in the proposed technique. The experimental results show that our approach reduces the overlay overhead significantly compared to a non-optimized approach and the previous approach. In [154], the author addressed the problem of task mapping in the context of multiprocessor applications with stochastic execution times and in the presence of constraints on the percentage of missed deadlines.

In [155], the author proposed a methodology consisting of ILP formulation to explore efficient mappings. These searches based approaches provide efficient mapping solutions, but they have high computational costs for large scale problems such as applications with large number of tasks. Different pruning strategies have been incorporated to prune the search space, thereby reducing the computational costs. In [156], the author proposed a complete allocation and scheduling framework, where an MPSoC virtual platform is used to accurately derive input parameters, validate abstract models of system components and assess constraint satisfaction and objective function optimization. The optimizer implements an efficient and exact approach to allocation and scheduling based on problem decomposition. The allocation subproblem is solved through integer programming while the scheduling one through constraint programming. The two solvers can interact by means of no-good generation, thus building an iterative procedure which has been proven to converge to the optimal solution. In [157], the author proposed a decomposition based approach to speed up constraint optimization. They optimize for the schedule length or make-span.

In [158], the author proposed a complete algorithm based on Constraint Programming which solves the allocation and scheduling problem as a whole. He introduced a number of search acceleration techniques that significantly reduce run-time by aggressively pruning the search space without compromising optimality. The solver has been tested on a number of non-trivial instances and demonstrated promising run-times on SDFGs of practical size and one order of magnitude speed-up the fastest known complete approach. In [159], the author proposed a mapping framework called Distributed Operation Layer (DOL), which optimizes for

computation and communication time. They integrate an analytic performance analysis strategy into DOL to alleviate the modeling and analysis of systems. And in [160], the author considered the number of software pipeline stages to map streaming applications on SPM-based embedded multi-core system. The proposed method scales well over a wide range of cores and SPMs. In [161], the author presented a new KPN mapping algorithm that addresses communication and computation jointly. The algorithm is tested on two platforms with real applications and with randomly generated KPNs. We show that the algorithm finds solutions in situations where bare process mapping fails. It also reduced the average application makespan considerably when compared to previous heuristics.

In [162], the author presented a methodology to map multiple use-cases onto the NoC architecture, satisfying the constraints of each use-case. We present dynamic re-configuration mechanisms that match the NoC configuration to the communication characteristics of each use-case, also accounting for use-cases that can run in parallel. The methodology is applied to several real and synthetic SoC benchmarks, which result in a large reduction in NoC area (an average of 80%) and power consumption (an average of 54%) compared to traditional design approaches.

In [163], the author proposed a many-to-many core-switch mapping (mCSM) that allows a switch (core) to have multiple connections to its adjacent cores (switches). We also present decomposition methods that can obtain the suboptimal solutions with enhanced computational efficiency. Its work is the first to provide an exact mixed-integer linear programming (MILP) formulation for the complete CSM problems, including the optimal choice of core placements, switches for each core, and network interfaces for communication flows. Experiments with four random benchmarks show that 4:4 mCSM achieves 81.2% of energy savings and 2.5% of bandwidth savings compared with one-to-one mapping. They also show that, for one-to-one mapping, our optimal solutions obtained by the full MILP save 34.8% of energy consumption and 34.4% of bandwidth requirement compared with those from the existing algorithms. In [164], the author proposed compiler approach has four major steps: task scheduling, processor mapping, data mapping, and packet routing. In the first step, the application code is parallelized and the resulting parallel threads are assigned to virtual processors. The second step implements a virtual processor-to-physical processor mapping. The goal of this mapping is to ensure that the threads that are expected to communicate frequently with each other are assigned to neighboring processors as much as possible. In the third step, data elements are mapped to the memories attached to CMP nodes. The main objective of this mapping is to place a given data item into a node which is closer to the nodes that access it. The last step of our approach

determines the paths (between memories and processors) for data to travel in an energy efficient manner. In his paper, he describes the compiler algorithms we implemented in detail and present an experimental evaluation of the framework. In its evaluation, he tests the entire framework as well as the impact of omitting some of its steps. This experimental analysis clearly shows that the proposed framework reduces energy consumption of our applications significantly (27.41% on average over a pure performance oriented application mapping strategy) as a result of improved locality of data accesses. In [165], the author introduced a GA based approach that use Dynamic Voltage Scaling (DVS) to reduce the energy consumption by up to 51%. These methodologies show significant energy savings. Some methodologies that perform optimization for both energy consumption and compute performance are introduced in [103], [118] and [72]. In [166], the author proposed a mapping methodology that reduces power consumption by decreasing the energy consumption in communication while guaranteeing the required performance. They methodology provides an energy savings of 51%. In [167], the author developed an approach that effectively and efficiently allocates execution and storage slack to jointly optimize system lifetime and cost. While exploring less than 1.4% of the slack allocation design space, its approach consistently outperforms alternative slack allocation techniques to find sets of designs within 1.4% of the lifetime-cost Pareto-optimal front. In [168], the author proposed a thermal-aware system analysis method that produces mappings with a lower peak temperature of the system, leading to reliable system design. In [169], the author proposed an algorithm that determines a processor core allocation, level of system-level and processor-level structural redundancy, assignment of tasks to processors, floorplan, and schedule in order to minimize system failure rate and area while meeting functionality and timing constraints. Changes to the thermal profile resulting from changes in allocation, assignment, scheduling, and floorplan are modeled and optimized during synthesis, as is the impact of thermal profile of temperature-dependent failure mechanisms. The proposed techniques have the potential to substantially increase MPSoC system mean time to failure compared to area-optimized solutions. If power densities are high and the dominant lifetime failure mechanisms are strongly dependent on temperature. In [170], the author proposed a lifetime-aware task mapping methodology that produces mappings with higher lifetimes. These methodologies take preventive measures by performing reliability-aware mapping in order to reduce the occurrence of faults in the systems. Such preventive measures increase lifetime of systems.

Most of the design-time methodologies adopt search based approaches (e.g., GA, ILP, SA) that incur high computational costs. Thus, the evaluation time might not be acceptable for

large scale problems. However, they provide efficient mapping solutions for small scale systems within acceptable time. The evaluation time can be reduced by efficient pruning of the search space, but at the risk of missing the highest quality mapping solutions. The reliability-aware design-time methodologies increase the system lifetime, but they cannot overcome the faults incurred in the system.

Further, as the design-time methodologies find the placement of tasks at design-time, they are not suitable for run-time varying workloads in the systems and run-time changing environments. Such dynamic workload scenarios require remapping/ run-time mapping of applications. Even if these mapping methodologies are inadequate for the dynamic workload scenarios, they might be useful to find the initial task placement, or be optimized to be working at run-time.

4.2.2 Run-time mapping

In contrast to the design-time mapping, run-time mapping needs to account for the time taken to map each task as it contributes to overall application execution time. Furthermore, the tasks are mapped one by one, unlike the static case where all the tasks are mapped at once by looking globally at the system. Therefore, typically greedy heuristic algorithms are used for efficient run-time mapping in order to optimize performance metrics such as energy consumption, communication latency, execution time, etc. The run-time mapping has several requirements, advantages and issues & research challenge for different available mapping alternatives.

Requirement: The run-time mapping caters for dynamically workload scenarios where mapping of one or more already running applications may need to be reconsidered in case of the following requirements:

• Insertion of a new application into the system, which needs resources from the already executing applications.

• Modifying parameters of a running application.

• Killing a running application in order to free its occupied resources.

• Changing performance requirements of a running application.

This might need extra resources for performing extra functionality.

• When current mapping is not sufficiently optimal, it requires (re-)mapping.

Advantages: In addition to the suitability for dynamic workload scenarios, run-time mapping also offers a number of advantages. Some of them are as follows:

• Adaptability of the available resources: The available resources vary over time as the applications of the dynamic workload scenario enter at run-time.

• Ability to enable unforeseeable upgrades: It is possible to upgrade the system for new applications or changing standards that are not known at design-time, even after the delivery of the system to the end-user.

• Ability to avoid defective parts of a SoC: If one or more processing cores are not working properly after production of a SoC, then the defective cores can be disabled before the mapping process. Aging can lead to defective cores that are unforeseeable at design-time.

At run-time, the mapping of new applications to be supported onto a platform can be handled either by performing all the processing at the same time. The run-time platform manager handles the mapping of applications by taking the updated resources' status (Current System Status) into account. For on-the-fly mapping, efficient heuristics are required to assign new arriving tasks on the platform resources. These heuristics cannot guarantee for schedulability. However, these heuristics are platform independent since they do not use any platform specific analysis results computed in advance. Such heuristics lend well to map unknown applications (not available at design-time) on any platform.

For mapping using previously analyzed (DSE) results, the applications to be supported on a platform should be known at design-time. In such cases, light-weight heuristics are required to select the most efficient mappings for each application from the design-time (offline) analyzed mappings stored on the system (Mappings using different number of PEs). The selection is done subject to available system resources (extracted from Current System Status) and desired performance (User demands). The selected mapping is used to configure the platform. Compute intensive analysis is performed at design-time (Design-time DSE), facilitating for light-weight run-time platform manager that can configure the applications efficiently. In DSE, application and architecture description are taken as input and a number of mappings are produced. Such mapping methodologies have been referred to as hybrid mapping as they take the advantages of both design-time and run-time. The hybrid approach maps applications more efficiently than on-the-fly heuristics. However, flexibility in these approaches is limited, since all potential applications must be known in entirety at design-time and analysis results will be applied only to the analyzed platform. Therefore, design-time analysis needs to be repeated when the application sets or platform changes. Further, storing analysis results introduces additional memory overhead.

In [171], the author changes the thread-to-processor mapping at run-time based on the workload variation in order to optimize the performance. An improvement of 29% is achieved in the overall execution time. In [172], the presented heuristic offers additional advantage to trade-off execution time versus solution quality. In [173] the author proposed a variation-aware

task and communication mapping methodology for multiprocessor system-on-chips that use network-on-chip communication architecture so that the impact of parameter variations can be mitigated. His mapping scheme accounts for variation in both the processing cores and the communication links to ensure a complete and accurate model of the entire system. A new design metric, called performance yield and defined as the probability of the assigned schedule that meet the predefined performance constraints, is used to guide both the task scheduling and the routing path allocation procedure. An efficient yield computation method for this mapping complements and significantly improves the effectiveness of the proposed variation-aware mapping algorithm. Experimental results show that our variation-aware mapper achieves significant yield improvements over worst-case and nominal-case deterministic mapper.

In [174], the author performed adaptive resource management to maintain the QoS requirement of the application. In [175], the author proposed a run-time strategy for allocating the application tasks to the platform resources in homogeneous networks-on-chip (NoCs). As a novel contribution, he incorporated the user behavior information in the resource allocation process; this allows the system to better respond to real-time changes and adapt dynamically to user needs. Several algorithms are then proposed for solving the task allocation problem, while minimizing the communication energy consumption and network contention. If user behavior is taken into consideration, we observe about 60% of communication energy savings (with negligible and energy runtime overhead) compared to an arbitrary task allocation strategy.

In [176], the author presented a number of communication-aware run-time mapping heuristics for the efficient mapping of multiple applications onto an MPSoC platform in which more than one task can be supported by each processing element (PE). The proposed mapping heuristics examine the available resources prior to recommending the adjacent communicating tasks on to the same PE. In addition, the proposed heuristics give priority to the tasks of an application in close proximity so as to further minimize the communication overhead. His investigations show that the proposed heuristics are capable of alleviating network-on-chip congestion bottlenecks when compared to existing alternatives.

4.2.3 Upcoming trends and open challenges in hybrid mapping

The reported mapping methodologies provide three alternatives: design-time mapping, on-the-fly mapping and hybrid (design-time analysis and then run-time use) mapping. Design-time techniques have pre-dominated the reported literature. However, their inability to handle dynamic workload scenarios has led to the formulation of on-the-fly mapping methodologies. On-the-fly strategies surmount the limitation of handling dynamic workloads at run-time, but with the fallout of possible non-optimal mapping due to limited compute power at run-time.

Recently, the issues of design time and on-the-fly strategies have been addressed by developing hybrid mapping methodologies that attempt to incorporate the advantages of both. Hybrid strategies combine design space exploration of design-time techniques with the run-time management in order to select mapping configurations that are best suited to newly arriving applications.

They involve minimum computation at run-time, facilitating for light-weight run-time manager performing efficient mapping. Our experimental results have shown that runtime mapping gets speeded up by 93% when compared to state-of-the-art on-the-fly mapping methodologies [177]. Although the advantages of hybrid strategy seem promising, it comes with its own trade-offs due to the inherent pseudo dynamic nature and inability to handle new applications without available design-time exploration.

With no doubt, hybrid strategies seem to be followed in the field of mapping methodologies, but due to their nascent development and lack of in-depth examination, further development of design-time and on-the-fly mapping methodologies will continue hand-in-hand with hybrid strategies.

5. Classification of heuristic optimization methods

Optimization problems can be of different type such as multi objective optimization, multimodal optimization and combinatorial optimization. In this session, we will detail the combinatorial optimization. Many problems are basically divided into two categories; the first is where the solutions are programmed with real valued variable and the other is those where solution is programmed with discrete variable where the combinatorial optimization problems belong to.

In this section, we distinguished three main categories of optimization algorithm based on their efficiency of solving problems and the way of exploring the search space with dealing with the set of solutions. We characterized:

5.1 Trajectory methods

The trajectory methods are those working on a single solution. The term “trajectory methods” is used according the search that is done by these methods which is characterized by a trajectory in the search space. The following subsections will provide introductions of two types of trajectory methods:

5.1.1 Tabu search

Tabu search [75] was first proposed by Glover in 1986. Tabu search is a metaheuristic that guides a local search heuristic to escape from local minima and in the same time, to implement an exploration scheme. The simple tabu search algorithm applies an improving local search where at each iteration, the best solution among the list of neighborhoods is selected and remarked as a new current solution. A short-term memory is implemented as a tabu list where solution attributes are stored to avoid short term cycling. A term Aspiration criteria are defined which is allowed to include some unvisited solution of good quality which are excluded from allowed set. Starting from the basic search strategy, a number of developments and elaboration have been proposed over the years, and these include an introduction of intensification and diversification mechanisms where they are implemented via different forms of long term memories. Adaptive memories provide a mechanism allowing a diversity and intensity of the search in order to create more flexible search behaviors.

5.1.2 Variable neighborhood search

Variable neighborhood search (VNS) was proposed by Mladenovic and Hansen in 1997. VNS [90] is a method based on the idea of dynamically changing neighborhood within a local search with a descendent method to get from the neighbors to local optima exploring, systematically or at a random distant neighborhood of the incumbent solution. The method moves from this current solution to new one if there is an improvement in the solution. In this way, optimal variable is kept in the incumbent and obtains promising neighbors. VNS have three main cycles that consist of shaking, local search and move.

5.2 Population based metaheuristics

This type of metaheuristic explicitly works with a set of solutions that are merged, either implicitly or explicitly, to create new another solution. We only introduce the used methods in our implementation that is genetic algorithms and it as follows:

5.2.1 Genetic algorithms

5.2.1.1 History

Darwin’s theory of Evolution states that all life is related and has descended from a common ancestor. The theory presumes that complex creatures have evolved from more simplistic ancestors naturally over time. In a nutshell, as random genetic mutations occur within an organism's genetic code, the beneficial mutations are preserved because they aid survival -- a process known as "natural selection." These beneficial mutations are passed on to the next

generation. Over time, beneficial mutations accumulate and the result is an entirely different organism.

Genetic algorithms are adaptive heuristic search algorithm premised on the Darwin's evolutionary ideas of natural selection and genetic. The basic concept of genetic algorithms is designed to simulate processes in natural system necessary for evolution. As such, they represent an intelligent exploitation of a random search within a defined search space to solve a problem. First pioneered by John Holland in the 60's, GAs has been widely studied, experimented and applied in many fields in engineering world. Not only does genetic algorithm provide an alternative method to solving problem, it consistently outperforms other traditional methods in most of the problems link. Many of the real world problems which involve finding optimal parameters might prove difficult for traditional methods but are ideal for genetic algorithms.

5.2.1.2 Definition

Evolutionary algorithms (EAs) are population-based meta-heuristic optimization algorithms that use biology-inspired mechanisms and survival of the fittest theory in order to refine a set of solution iteratively.

Genetic algorithms are subclass of evolutionary algorithms where the individuals of the search space are binary strings or arrays of other elementary types. Genetic algorithms are computer based search techniques patterned after the genetic mechanisms of biological organisms that have adapted and flourished in changing highly competitive environment. The last decade has witnessed many exciting advances in the use of genetic algorithms to solve optimization problems in process control systems. Genetic algorithms are the solution for optimization of hard problems quickly, reliably and accurately. As the complexity of the real-time controller increases, the genetic algorithms applications have grown in more than equal measure, the pseudo code of genetic algorithm is defined in Algorithm 3.1.

One of the most fundamental principals in our world is the search for an optimal state. Optimization is the process of modifying the inputs or characteristics of a device, mathematical process to obtain minimum or maximum of the output. The input to the optimization process is the cost function, objective function or fitness function and the output is the fitness function of the system. Optimization is a primary tool, needed to tackle the unsolvable or hard problems. Optimization algorithms can be characterized into five categories. In the trial and error optimization, the processes affect the output without knowing about the constraints, responsible

to produce the output. A mathematical formula describes the objective function for optimization.

One dimensional optimization contains one variable and a problem having more than one variable requires multi-dimensional approach. As the number of dimensions increases, the process of optimization becomes difficult. Dynamic optimization is time dependent, whereas the static optimization is independent of time. The static problem is difficult to solve for finding the best solution, but the added dimension of time increases the challenge of solving dynamic problems. Discrete variable optimization contains only a finite number of possible values, whereas continuous variables have an infinite number of possible values. Variables often have limits or constraints. Constrained optimization incorporates variable equalities and inequalities into the cost function, whereas unconstrained optimization allows the variable to take any value. Many problems have been solved by the use of the GA. In [100], the author proposed a genetic algorithm to solve quadratic assignment problem. Experiment illustrates that genetic algorithms find a better solution than those of the best previously known heuristics. In [101], the author proposed genetic algorithm to solve the set partitioning problem when a basic genetic algorithm components such as fitness definition, parent selection and population replacement are modified and the experiment shows that genetic algorithm based heuristic is producing high quality solutions. In [102], the author proposed a genetic algorithm based heuristic for solving the generalized assignment problem where the GA heuristic includes a problem-specific coding of a solution structure, a fitness-unfitness pair evaluation function and a local improvement procedure and the experiment shows that the genetic algorithm heuristic is able to find high quality solutions. In [103], the author proposed genetic algorithm for solving the multidimensional knapsack problem where a heuristic based upon genetic algorithm is introduced and it is capable of obtaining high quality solutions to the problem. In [104], the author proposed genetic algorithm to solve the quadratic assignment problem where GA includes many greedy principles in its design. In [105], the author proposed genetic algorithm to solve vehicle routing problem with time window and three evolutionary algorithms were described and compared with the other metaheuristic algorithm. In [106], the author proposed a genetic algorithm to solve traveling salesman problem with precedence constraints where the key concept is the topological sort. Also a new crossover operation is introduced for the proposed genetic algorithm and the experiment shows that the proposed genetic algorithm produces an optimal solution and shows better performance compared to the traditional algorithm.

ALGORITHM 3.1. PSEUDO CODE FOR GENETIC ALGORITHM

```
Generate P random chromosome;
REPEAT
    Determine fitness of all chromosomes i=1..P;
    Determine probabilities Pi based on relative fitness;
    For number of reproduction;
        Randomly select two parents based on Pi;
        Generate two children by crossover operation on parents;
    END
    Insert offspring into the population;
    Apply mutation to all/some individuals;
UNTIL halting criterion is met;
```

In [107], the author proposed an improved genetic algorithm to solve traveling salesman problem where the new crossover operation, population reformulates operation, multi mutation operation, partial local optimal mutation operation and rearrangement operation are used to solve the problem. In [108], the author proposed an improved genetic algorithm for solving scheduling problem while the proposed approach is improved by a genetic algorithm using multipliers can be altered during the search process. In [109], the author proposed a genetic algorithm for solving traveling salesman problem where two mutation operations named inverted exchange and inverted displacement are combined to increase the performance of genetic algorithm. In [110], the author proposed genetic algorithm to solve flow shop scheduling problem where a genetic algorithm based heuristic is described to make span minimization on flow shop scheduling.

5.2.1.3 Working principle of genetic algorithms

The workability of genetic algorithms is based on the Darwinian's theory of survival of the fittest. Genetic algorithms may contain a chromosome, a gene, a set of the population, fitness, fitness function, breeding, mutation and selection. Genetic algorithms begin with a set of solutions represented by chromosomes, called a population. Solutions from one population are taken and used to form a new population, which is motivated by the possibility that the new population will be better than the old one. Further, solutions are selected according to their fitness to form new solutions, that is, offspring. The above process is repeated until some condition is satisfied. Algorithmically, the basic genetic algorithm is outlined as below:

Step I [Start] Generate random population of chromosomes, that is, suitable solutions for the problem.

Step II [Fitness] Evaluate the fitness of each chromosome in the population.

Step III [New population] Create a new population by repeating the following steps until the new population is complete.

a) [Selection] Select two parent chromosomes from a population according to their fitness. Better the fitness, the bigger chance to be selected to be the parent.

b) [Crossover] With a crossover probability, cross over the parents to form new offspring, that is, children. If no crossover was performed, offspring is the exact copy of parents.

c) [Mutation] With a mutation probability, mutate new offspring at each locus.

d) [Accepting] Place new offspring in the new population.

Step IV [Replace] Use new generated population for a further run of the algorithm.

Step V [Test] If the end condition is satisfied, stop, and return the best solution in current population.

Step VI [Loop] Go to step 2.

The genetic algorithms' performance is largely influenced by crossover and mutation operators. The block diagram representation of genetic algorithms is shown in Figure 3.4.

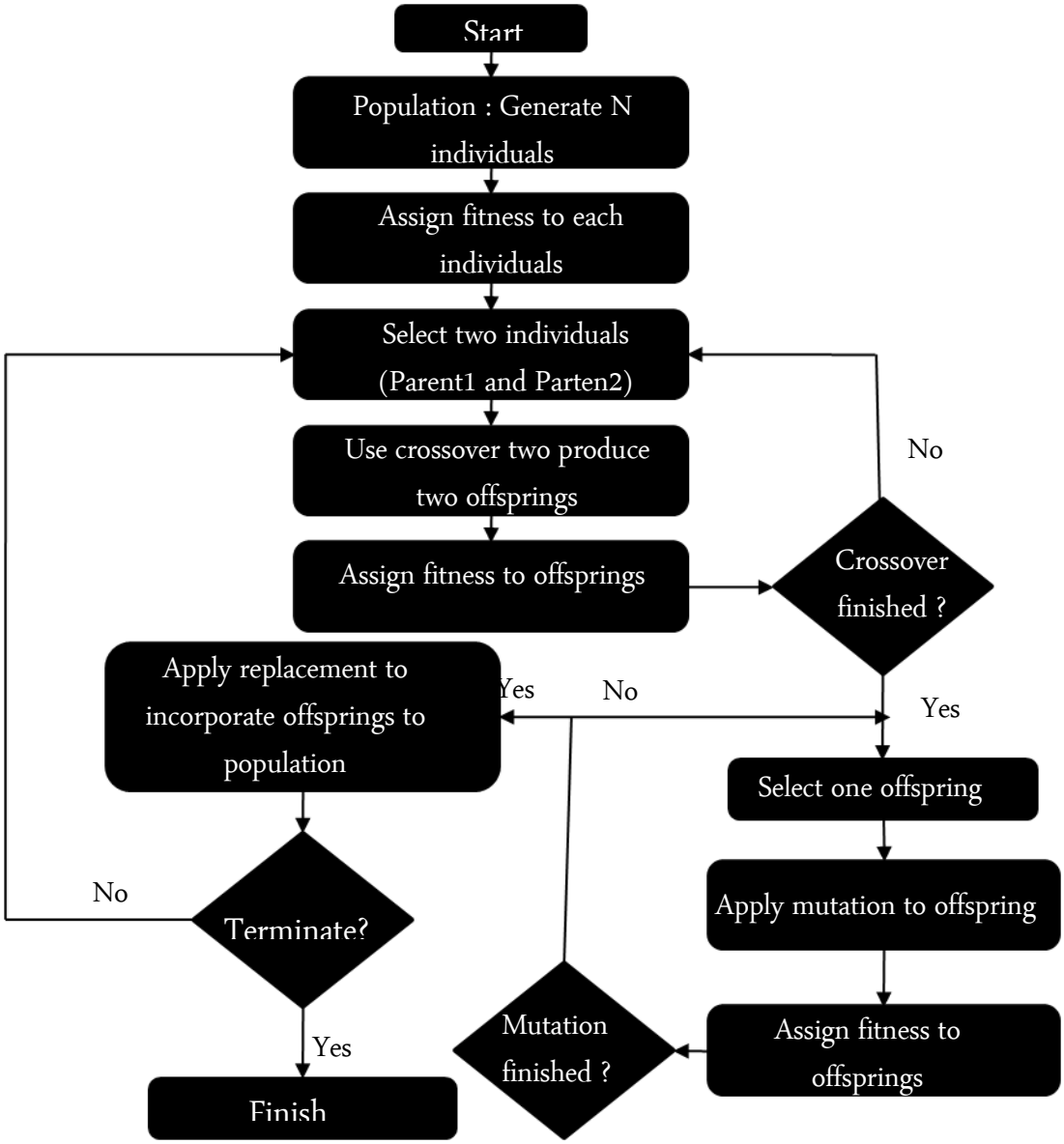


Figure 3.4. Block diagram representation of genetic algorithms.

a. Encoding technique in genetic algorithms

Encoding techniques in genetic algorithms are problem specific, which transforms the problem solution into chromosomes. Various encoding techniques used in genetic algorithms are binary encoding, permutation encoding, value encoding and tree encoding.

a. a Binary encoding

It is the most common form of encoding in which the data value is converted into binary strings. Binary encoding gives many possible chromosomes with a small number of alleles. A chromosome is represented in binary encoding as shown in Figure 3.5.

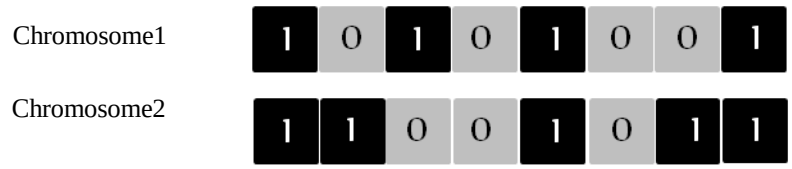


Figure 3.5. Binary encoding.

a. b Permutation encoding

Permutation encoding is best suited for ordering or queuing problems. Travelling salesman is a challenging problem in optimization, where permutation encoding is used. In permutation encoding, every chromosome is a string of numbers in a sequence as shown in Figure 3.6.

Figure 3.6. Permutation encoding.

a. c *Value encoding*

Value encoding can be form number, real number of characters to some complicated objects. Value encoding is a technique in which every chromosome is a string of some values and is used where some more complicated values are required. It can be expressed as shown in Figure 3.7.

Figure 3.7. Value encoding.

a. d Tree encoding

It is best suited technique for evolving expressions or programs, such as genetic programming. In tree encoding, every chromosome is a tree of some objects, functions or commands in programming languages.

Locator/identifier separation protocol (LISP) programming language is used for this purpose.LISP programs can be represented in a tree structure for crossover and mutation. In the tree encoding, the chromosomes are represented as shown in Figure 3.8.

There are no specific directions for using the type of encoding scheme in the specified problem rather, it depends upon the applicability and the requirements of the problem.

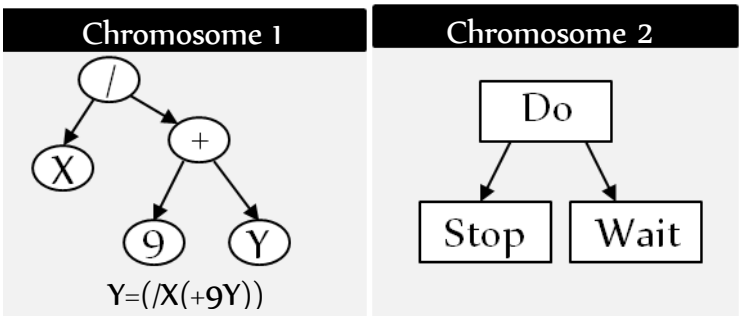


Figure 3.8. Tree Encoding.

b. Selection techniques in genetic algorithms (GAS)

Selection is an important function in genetic algorithms, based on an evaluation criterion that returns a measurement of worth for any chromosome in the context of the problem. It is the stage of genetic algorithm in which individual genomes are chosen from the string of chromosomes. The commonly used techniques for selection of chromosomes are Roulette wheel, rank selection and steady state selection.

b. a Roulette wheel selection

In this method the parents are selected according to their fitness. Better chromosomes, are having more chances to be selected as parents. It is the most common method for implementing fitness proportionate selection. Each individual is assigned a slice of the circular roulette wheel, and the size of the slice is proportional to the individual fitness of chromosomes, that is, bigger the value, larger the size of slice is. The functioning of the roulette wheel algorithm is described below:

Step 1 [Sum] Find the sum of all chromosomes' fitness in the population.

Step 2 [Select] Generate random number from the given population interval.

Step 3 [Loop] Go through the entire population and sum the fitness. When this sum is more than a fitness criteria value, stop and return this chromosome.

Figure 3.9 shows the roulette wheel for six individuals having different fitness values. The Sixth individual has a higher fitness than any other, it is expected that the Roulette wheel selection will choose the sixth individual more than any other individual.

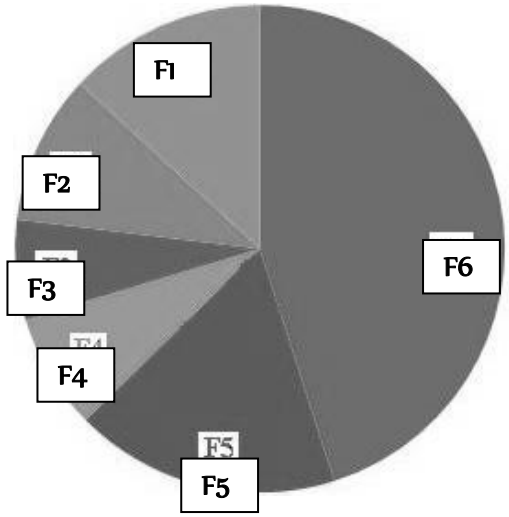


Figure 3.9. Roulette wheel method.

b. b Rank selection method

The application of the roulette wheel selection method is not satisfactory in genetic algorithms, when the fitness value of chromosomes differs very much. It is a slower convergence technique, which ranks the population by certain criteria and then every chromosome receives fitness value determined by this ranking. This method prevents quick convergence and the individuals in a population are ranked according to the fitness and the expected value of each individual depends on its rank rather than its absolute fitness.

The rank selection method is shown in Figure 3.10. For example, if the best chromosome fitness is 80 percent, its circumference occupies 80 percent of the roulette wheel and then other chromosomes will have minimum chances to be selected. On the other hand, the rank selection first ranks the population according to their fitness and then every chromosome receives ranking. The worst will have fitness 1, the second worst will have a fitness of 2, and the best one will have a fitness value n, where n is the number of chromosomes in the population.

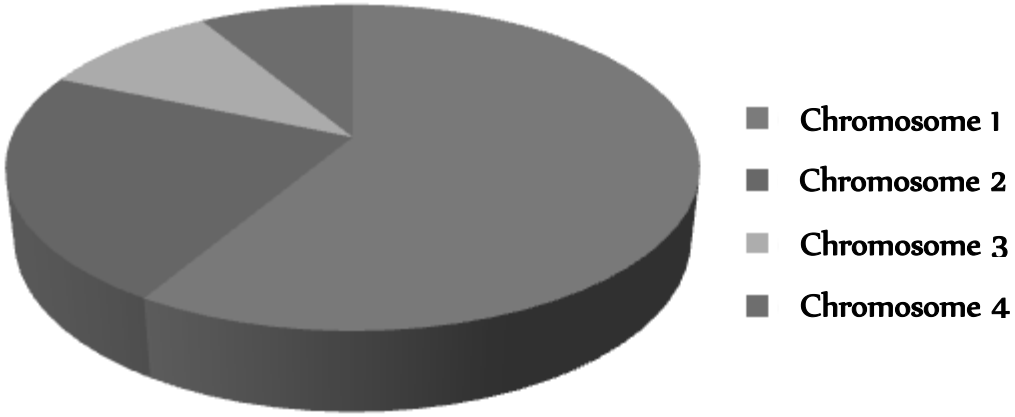


Figure 3.10. Rank selection methods.

b. c Steady-state selection

This method replaces few individuals in each generation, and is not a particular method for selecting the parents.

Only a small number of newly created offspring put in place of least fit individual. The main idea of steady-state selection is that bigger part of chromosome should retain to successive population.

5.2.1.4 Genetic algorithms operators

Genetic algorithms can be applied to any process control application for optimization of different parameters. Genetic algorithms use various operators viz. the crossover, mutation for the proper selection of optimized value. Selection of proper crossover and mutation technique depends upon the encoding method and as per the requirement of the problem.

a. Crossover

It is the process in which genes are selected from the parent chromosomes and new offspring is created. Crossover can be performed with binary encoding, permutation encoding, value encoding and tree encoding.

b. a Binary encoding crossover

In the binary encoding, the chromosomes may crossover at a single point, two points, uniformly or arithmetically. In single point crossover, a single crossover point is chosen and the data before this point are exactly copied from first parent and the data after this point are exactly copied from the second parent to create new offspring. Two parents in this method give two new offspring. The two point crossover is illustrated in Figure 3.11.

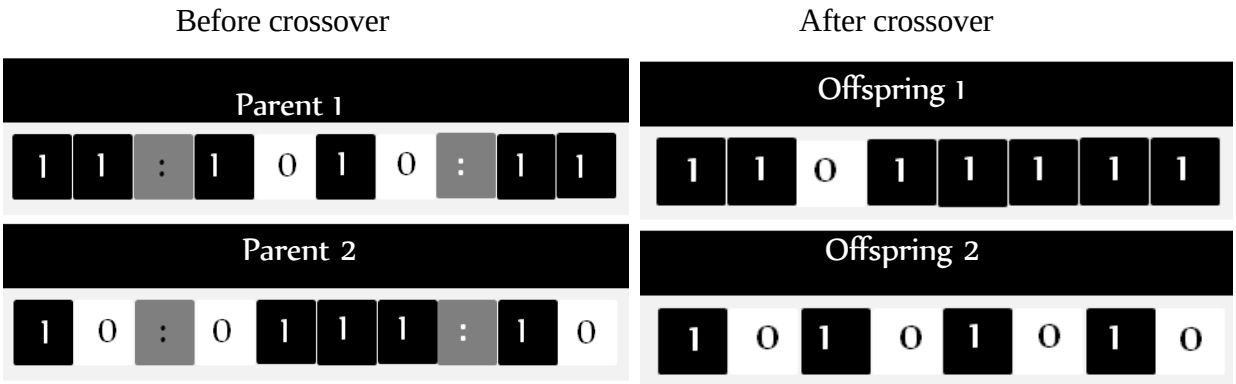


Figure 3.11. Two point crossover.

b. b Uniform crossover

In uniform crossover, data of the first parent chromosome and second parent chromosome are randomly copied, which is illustrated in Figure 3.12.

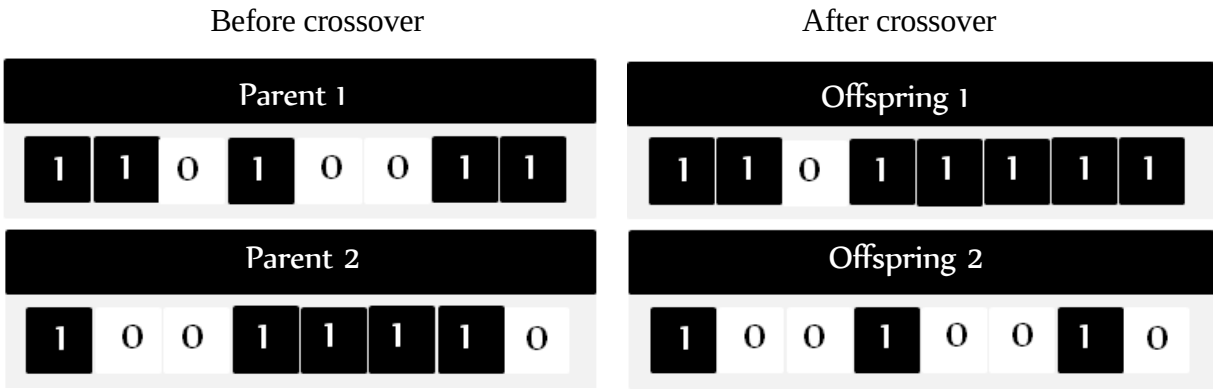


Figure 3.12. Uniform crossover.

b. c Arithmetic crossover

In arithmetic crossover, crossover of chromosomes is performed by AND and OR operators to create new offsprings as illustrated in Figure 3.13.

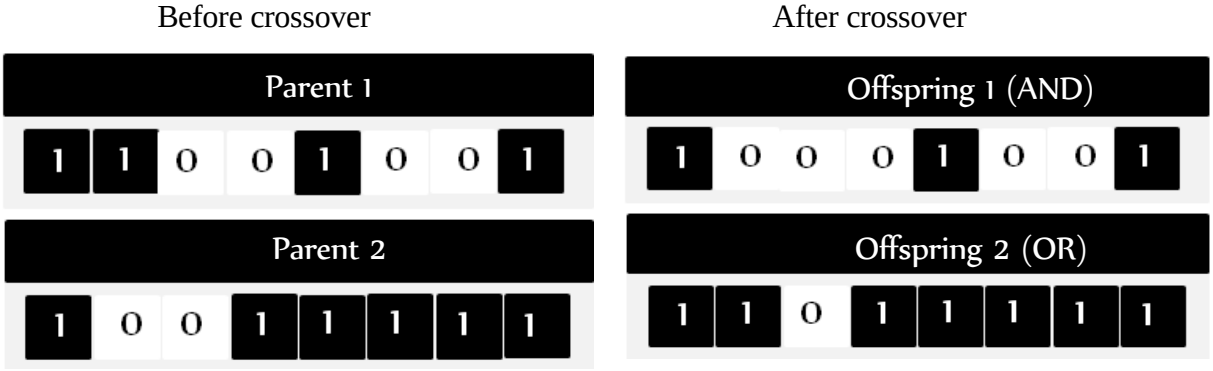


Figure 3.13. Arithmetic crossover.

b. d Permutation encoding crossover

In permutation encoding crossover, one crossover point is selected. The permutation is copied from first parent chromosome up to the point of crossover and the other parent chromosome is exactly copied to ensure that no number is left to be put in the offspring. Further, if the number is not yet in the offspring, it is added to the offspring chromosome. Travelling salesman problems and task ordering problems can be easily solved by permutation encoding. Figure 3.14 illustrates the single point crossover with permutation encoding.

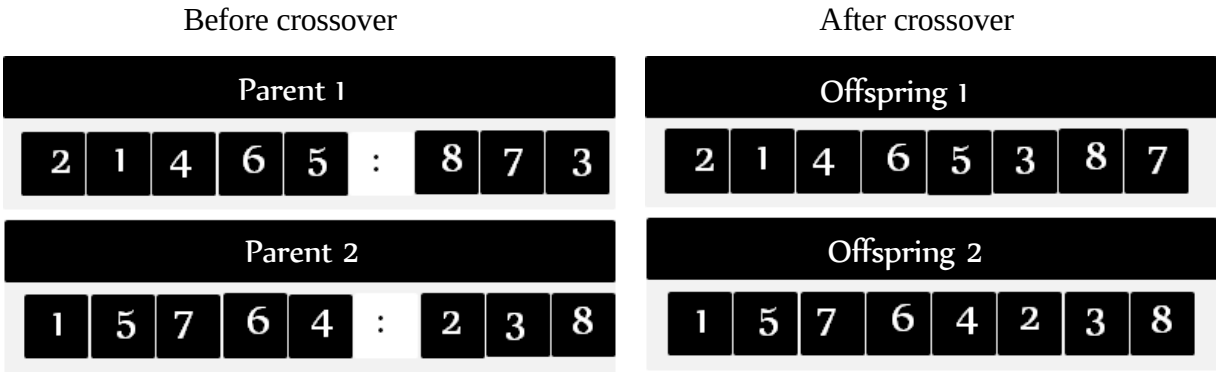


Figure 3.14. Permutation encoding crossover.

b. e Value encoding crossover

It can be performed at a single point, two point, uniform and arithmetic representation as in binary encoding technique. Figure 3.15 illustrates the single point crossover with value encoding.

Figure 3.15. Value encoding crossover.

b. f Tree encoding crossover

In this type of crossover, one point of crossover is selected in both parent tree chromosomes, which are divided at a point. The parts of the tree below crossover point are exactly exchanged to produce new offspring, which is illustrated in Figure 3.16.

The choice of the type of the crossover is strictly depends upon the problem.

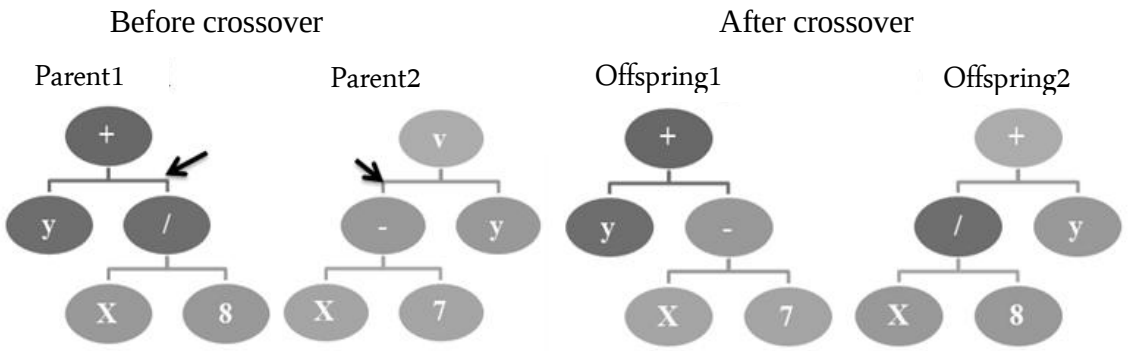


Figure 3.16. Tree encoding crossover.

b. Mutation

Premature convergence is a critical problem in most optimization techniques, consisting of populations, which occurs when highly fit parent chromosomes in the population breed many similar offspring in early evolution time. Crossover operation of genetic algorithms cannot generate quite different offspring from their parents because the acquired information is used to crossover the chromosomes. An alternate operator, mutation, can search new areas in contrast to the crossover. Crossover is referred as exploitation operator, whereas the mutation is exploration one. Like crossover, mutation can also be performed for all types of encoding techniques.

b. a Binary encoding mutation

In binary encoding mutation, the bits selected for creating new offspring are inverted, which is illustrated in Figure 3.17.

In binary encoding mutation, if the bit 1 is converted into bit 0, it decreases the numerical value of the chromosome, and is known as down mutation. Similarly, if the bit 0 is converted into bit 1, the numerical value of the chromosome increases and is referred as up mutation.

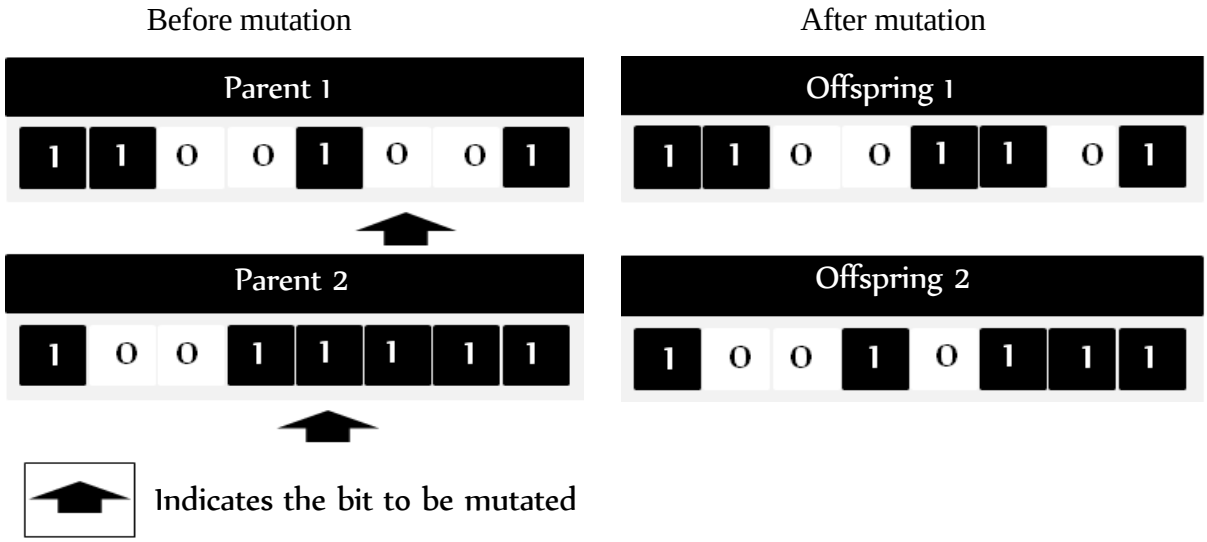


Figure 3.17. Binary encoding mutation.

b. b Permutation encoding mutation

In permutation encoding mutation, the order of the two numbers given in a sequence are exchanged as it is illustrated in Figure 3.18.

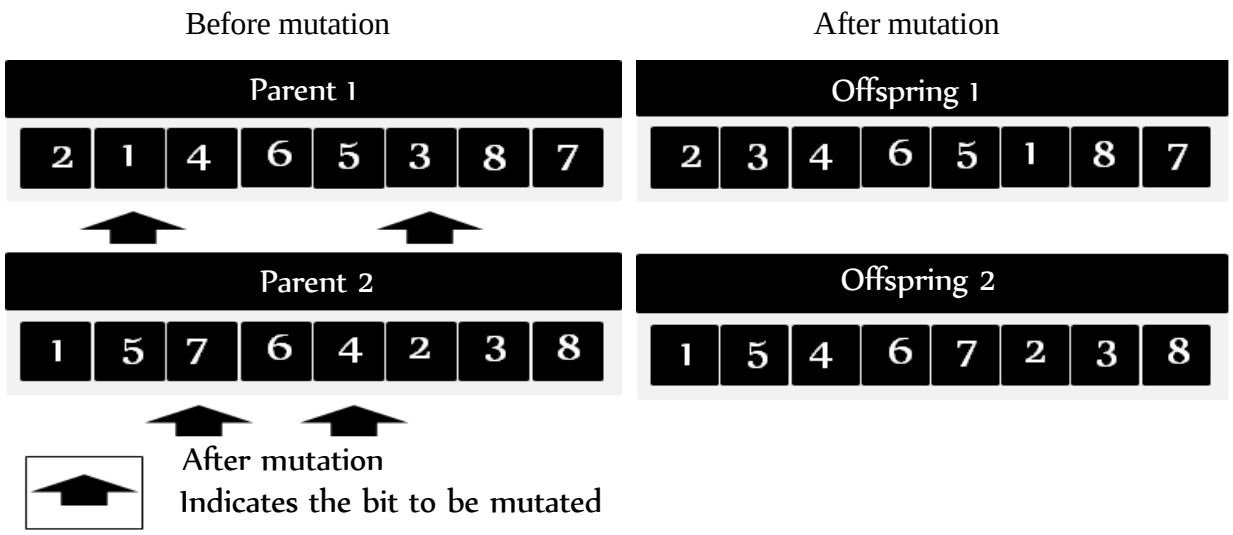


Figure 3.18. Permutation encoding mutation.

b. c Value encoding mutation

In value encoding mutation, a smaller numerical value is either added or subtracted from the selected values of chromosomes to create new offspring, which is illustrated in Figure 3.19.

Figure 3.19. Value encoding mutation.

b. d Tree encoding mutation

Tree encoding mutation, mutates the certain selected nodes of the tree to create new offspring, which is illustrated in Figure 3.20.

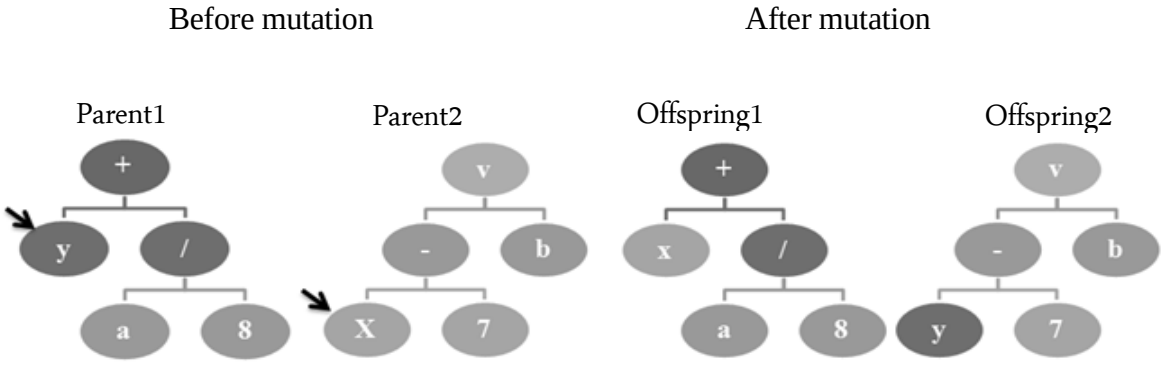


Figure 3.20. Tree encoding mutation.

5.2.1.5 Genetic algorithms issues

Genetic algorithms can be applied in complex non-linear process controllers for the optimization of parameters. Some issues are important to be considered for proper implementation of genetic algorithms to a plant to be optimized. Deciding on population size is an important issue while applying genetic algorithms. It is recommended by the researchers, that the population size should be of about 20 to 30 chromosomes. A very big population size consumes more time for finding optimum solution, which may deteriorate the performance of genetic algorithms.

Genetic algorithms may suffer from the problem of premature convergence due to improper selection of crossover rates. The higher crossover rate of about 85 percent to 95 percent is recommended to minimize premature convergence problems.

The low mutation rate of about 0.5 percent to 1 percent is generally recommended to obtain optimized results from genetic algorithm. Mutation is an artificial and forced method of changing the numerical value of the chromosome. Mutation should be avoided as far as possible because it is totally adhoc and random in nature. Small mutation rates prevent genetic algorithms from falling into local maxima or minima.

Deciding on selection method for selecting good chromosomes is another important issue while applying genetic algorithms for process control applications. Rank selection method and roulette wheel selection methods have shown good results over other methods of selection.

Genetic algorithms play an important role in process control applications for the optimization of parameters. Many researchers have contributed in this field. A broad review of genetic algorithm applications gives the directions to use this technique for the optimization of the process controllers.

According to [131], even GAs present many advantages, GA have also some weakness, limitations, and difficulties includes:

1. The problem of identifying the fitness function and definition of representation of the problem;

2. The problem of choosing the various parameters like the size of the population, mutation rate, crossover rate, the selection method and its strength;

3. Some time, premature convergence occurs and have trouble finding the exact global optimum;

4. Not effective for smooth unimodal functions, no effective terminator, and cannot use gradients;

5. Cannot easily incorporate problem specific information and Configuration is not straightforward;

6. Not good at identifying local optima, it needs to be coupled with a local search technique.

Even if GAs met these limitation, one can mitigate some by profiting its advantages like its possibility of parallelism and easy to discover global optimum by parallel search from multiple points in the space, hybridization with other different methods etc; or by identifying the problems that are solved easily by GAs (e.g. GA solve comfortably combinatorial optimization and transportation problems than smooth unimodal functions, for the later one can hybrid GA with other methods like Gradients etc.

6. Cellular automata

6.1 History

Cellular automata (CA) trace their beginning to the Los Alamos Laboratory in the 1940s, where mathematician John von Neumann was studying the concept of self-replicating robots. His idea was that moons or asteroids would be most efficiently mined by such automata, due to the exponential growth of their population. The cost of experimenting with such robots, however, was prohibitive. Stanislaw Ulam, also at Los Alamos, was working on the problem of crystal growth using a lattice model. At his suggestion von Neumann applied a lattice grid model also to the problem of self-reproducing automata. This model became what is now known as cellular automata.

What von Neumann invented with pen and paper was later popularized by the automaton known as the Game of Life, which is described above. Invented in the 1970s by John Conway, this model displays a wide variety of complex patterns despite its very simple rules for whether a given cell should ‘live’ or ‘die’. This simple CA caught the attention of researchers in a wide variety of fields, including computer science, physics, biology, economics, and mathematics.

In the 1980s Stephen Wolfram published a number of papers detailing his study of the universality of cellular automata and the complexity of their patterns. In 2002, after having studied cellular automata for decades, Stephen Wolfram published a 1280 page text on the subject of their simple rules and complex patterns. This fact, that cellular automata can display complex patterns despite having simple rules, is a reason that they are considered to be such a useful model type.

At the University of Siegen, Duchting and Vogelsaenger did some of the earliest research in the area of using cellular automata to model tumor growth. Their 1984 paper describes a

three-dimensional simulation of tumor growth and describes what its application to tumor treatment might be.

6.2 Characteristics of cellular automata

The following describe some terms and characteristics that shared by the most of cellular automata:

Discrete space: The space S which determines the dimension and the size of the configurations that is discrete.

Finite state set: The state set Q is always finite and typical Q = {0, 1} which defines a binary CA.

Cell/cell value: Each space-position i ∈ S identifies a cell which contains a cell-value Xi ∈ Q.

Configuration: A configuration X is the concatenation of all cell-values on the space S at time t, and therefore an element of Q^S.

Local function: The time evolution of the different cells is described by a local function f. Like the space of a CA, the time-parameter is also considered discrete.

Neighborhood scheme: The outcome of the local function at position i only depends on a finite set of cell-values in the vicinity of i. This set is defined by the neighborhood scheme N of the CA.

Global function: Given a space S (compatible with the neighborhood scheme), the local function f imposes a global function f_s on the set of configurations. This global function determines the space/time behavior of the cellular automaton on an initial configuration.

The most stringent and typical characteristic of the CA-model is the restriction that the local function does not depend on the time t or the place i: a cellular automaton has homogeneous space/time behavior. It is for this reason that CA is sometimes referred to as a shift-dynamical or translation invariant systems.

6.3 Formal definition of cellular automata

A cellular automaton is a discrete dynamical system where the space, time, and the states of the system are discrete. Each point in a regular spatial lattice is called a cell and can have any one of a finite number of states. The states of the cells in the lattice are updated according to a local rule. That is, the state of a cell at a given time depends only on its own state on one time step previously, and the states of its nearby neighbors at the previous time step. All cells on the

lattice are updated synchronously. Thus the state of the entire lattice advances in discrete time steps.

Formally, we define CA in the following way:

Definition 1. A Cellular Automaton is a 4-tuple $(L, \sum, N, f)$ consisting of a d-dimensional lattice of cells indexed by integers,$L = \mathbb{Z}^{\mathrm{d}}$, a finite set $\sum$ of cell states, a finite neighborhood scheme $N = \subseteq \mathbb{Z}^{\mathrm{d}}$, and a local transition function $f: \sum^{N} \to \sum$.

The transition function f simply takes, for each lattice cell position, $x \in L$, the states of the neighbors of x, which are the cells indexed by the set $x + N$ at the current time step, $\mathrm{t} \in \mathbb{Z}$ to determine the state of cell x at time $\mathrm{t} + 1$. There are two important properties of cellular automata that should be noted. Firstly, cellular automata are space-homogeneous, in that the local transition function performs the same function on each cell. Also, cellular automata are time-homogeneous, in that the local transition function does not depend on the time step t.

We may also view the transition function as the one that acts on the entire lattice, rather than on individual cells. In this view, we denote the states of the entire CA as a configuration $C \in \sum^{L}$ which gives the state of each individual cell. This gives us a global transition function which is simply a function that maps $F: \sum^{L} \to \sum^{L}$. Formally, we can express the evolution of CA by the formula:

$$Si\ (t+1) = f(\{\ Sj(t)\ \})\ o\grave{u}\ j \in Vi\ \wedge Vi \tag{5}$$

Figure 3.21 and Figure 3.22 illustrate a simple cellular automaton as presented in [130]. The lattice of this CA is the set of integers Z, i.e., the CA is one-dimensional. The neighborhood of each cell consists of the cell itself, along with its two nearest neighbors, one to each side. The cells, represented by boxes, have two possible states, in this the cases are represented by the box being either black or white. Figure 3.21 shows a pictorial representation of the CA transition function, as presented in [130]. Figure 3.22 gives a pictorial description of the automaton's evolution in time.

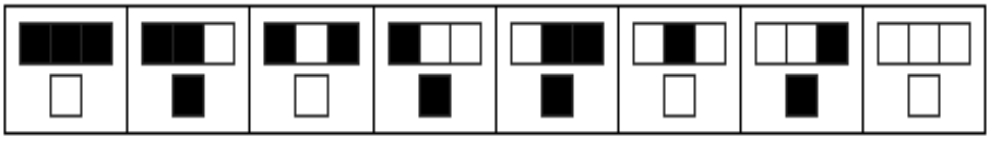


Figure 3.21. The first row represents the current state of the cell, as well as that of its nearest neighbors on either side. This is the input to the transition function. The possible states are either white or black. The second row represents the output of the transition function, i.e., the state of the cell after applying the transition function. Reading the diagram from the left, if the cell is in the state black and both neighbors are black as well, then the cell will be colored in white in the next time step. And if the right neighbor is white, then the cell will remain black, and so forth [130].

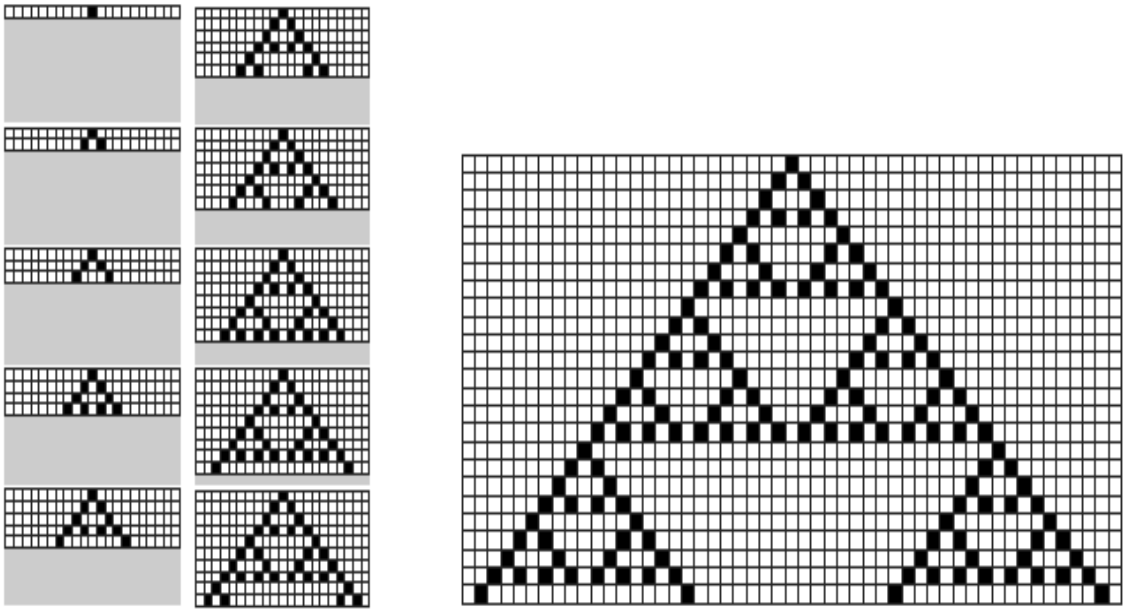


Figure 3.22. From top to bottom, left to right, the figures represent the evolution in time of the CA described in Figure 3.21. The initial configuration, shown in the top left image, is a single cell colored black, while all others are white. Time flows downwards. Each subsequent image depicts the current state of the automaton, and all stages since the initial one [130].

7. Quantum computing

The term "Quantum mechanics" is widely used in research, so what is it? The quantum mechanics is a mathematical framework or set of rules for the construction of physical theories.

Quantum computation and quantum information are the studies of the information processing task that can be accomplished using quantum mechanical systems

7.1 Prerequisites

We give here some important prerequisites about quantum computers that will be needed to understand the concepts of quantum computing.

7.1.1 Dirac's Notation

Dirac invented the "bra-ket" notation. It is very useful in quantum mechanics. The notation defines the "ket" vector, denoted by $|\psi\rangle$, $\langle\phi|$ being its conjugate transpose (also called Hermitian conjugate: the "bra"). The bracket" is then defined by $\langle\phi|\Psi\rangle$. The inner product is linear, and defined by:

$$\langle\phi|\psi\rangle = c \tag{6}$$

c is a number, if $c = \langle\phi|\psi\rangle$, the complex conjugate is

$$c^* = \langle\phi|\psi\rangle^* = \langle\phi|\psi\rangle \tag{7}$$

7.1.2 Qubit

A qubit is a quantum system in which the boolean states 0 and 1 are represented by a prescribed pair of normalized and mutually orthogonal quantum states labeled as $\{ |0\rangle, |1\rangle \}$. The $|0\rangle$ is called "ground state;" the $|1\rangle$ is the "excited state". As the most general electronic state is a superposition of the two basic states, we then have:

$$|\Psi_1\rangle = a|0\rangle + b|1\rangle \tag{8}$$

The two states form a "computational basis" and any other (pure) state of the qubit can be written as a superposition $\alpha|0\rangle + \beta|1\rangle$ for α and β such as $\alpha^2 + \beta^2 = 1$. Habitually, a qubit is a microscopic system, such as an atom, a nuclear spin, or a polarised photon.

7.1.3 Quantum Register

A collection of n qubits is called a quantum register of size n. For example, a quantum register of size 4 can store individual numbers such as 13: $|1\rangle \otimes |1\rangle \otimes |0\rangle \otimes |1\rangle = |1101\rangle = |13\rangle$, where $\otimes$ denotes the tensor product. It can also store the two of them simultaneously.

7.1.4 Quantum Logic Gates

Naturally, the quantum logic gate and quantum network is defined as follows:

***Definition of quantum logic gate*:** a quantum logic gate is a device which performs a fixed unitary operation on selected qubits in a fixed period of time.

Definition of quantum network: a quantum network is a device consisting of quantum logic gates whose computational steps are synchronized in time. A quantum gate acts on superpositions of different basis states of qubits, whereas classically this option is nonexistent.

Basic gates used in quantum computation are namely the Hadamard gate, the NOT gate, the CNOT (Controlled-NOT, also known as the XOR or the measurement gate) gate, the controlled phase-shift gate, the Toffoli gate and the Fredkin gate. In the following, will now define each one of them:

Hadamard Gate: the Hadamard gate is a single qubit gate H which performs the unitary operation known as the Hadamard transform whose action is the following:

$$|0\rangle \rightarrow |0\rangle + |1\rangle \tag{9}$$

$$|1\rangle \rightarrow |0\rangle - |1\rangle \tag{10}$$

NOT Gate: A single square root of NOT gate produces a completely random output with equal probabilities of the output being equal to 0 or 1. However two such gates linked

sequentially produce an output that is the inverse of the input, and thus behave in the same way as a classical NOT gate. We then have:

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{11}$$

$$|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{12}$$

Thus, an input of $|0\rangle$ leads to an equal and opposite amplitude of the output being $|0\rangle$ or $|1\rangle$. An input of $|1\rangle$ leads to an equal amplitude of the output of the gate being $|0\rangle$ and $|1\rangle$.

Controlled-NOT Gate: The Controlled-NOT gate (C-NOT) is a two-qubit gate, where the value of the first qubit (called control) determines what will happen to the second qubit (called target) qubit. More precisely, it flips the second qubit if the first qubit is $|1\rangle$ and does nothing if the control qubits is $|0\rangle$. Thus, the gate is represented by the following matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{13}$$

Written in the computational basis $\{ |00\rangle, |01\rangle, |10\rangle, |11\rangle \}$.This gate has attracted much interest in the field of quantum computation as it is reversible while requiring only 4 two-inputs.

Toffoli Gate: The Toffoli gate is a three-qubit gate where, if the first two bits are set, the third bit is flipped. It is defined by:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{14}$$

Universality of gates: Gates are called universal if they can be used to create any logic circuit, like the NAND (the conjunctive denial) gate in classical boolean-based circuits.

An extremely useful result of this universality is that any quantum computation can be done in terms of a C-NOT gate and a single-qubit gate, although, it might sometimes be more convenient to use other gates as well.

Also, a simple concatenation of the Toffoli gate and the C-NOT gate gives a simplified quantum adder, which is a good starting point for construction of full adders, multipliers, and more elaborate networks.

8. Conclusion

Optimization is an intelligent selecting of the best alternative among a given set of options or in other way, it can be viewed as a decision making in order to optimize one or several objectives according to the given scenarios. In this chapter we have defined the main used optimization algorithm in literature, and especially based network on chip. In the followed chapter, we will detail the proposed approach while our objective is to minimize the energy consumption in the network on chip, and that under timing constraints.

CHAPTER IV

CONTRIBUTIONS, TESTS AND EXPERIMENTS

1. Introduction

Because our objective is to minimize the energy consumption under the timing constraints, in this chapter we will detail the proposed solution while taking as case of study the mapping of two multi-applications: the multimedia system (MMS) and the automotive industrial (auto-indust) into 5x5 NoC based mesh.

2. Contribution in scheduling strategy and application model

2.1 Characteristics of scheduling policies

The scheduling of real-time tasks on a NoC architecture is where the correctness of execution not only depends on the logical result of computation but also on the time at which

the results are produced for that solving the scheduling problem is the process of mapping a given application onto a target architecture by repeating the following steps during the scheduling of each task:

1. Selecting which task of the application shall be considered.
2. Allocating this task to a resource.
3. Computation of start and execution times for the task.
4. Repeat these steps until all tasks are scheduled.

Note that the terms, 'mapping' and 'scheduling' are ambiguous. Scheduling is commonly used to describe all of the above mentioned steps as well as to describe the computation of start and execution times only.

The approach of solving the scheduling problem depends on the characteristics of both, the target architecture and the application that will be run on it.

2.1.1 Target architecture

The target architecture is either a uniprocessor or a multiprocessor platform. In the latter case the distinction between a homogeneous system, where all processing elements (PEs) have the same characteristics, and a heterogeneous system, with PEs that may have different specifications. In our implementation each tile is a uniprocessor where P_i represents a node of the network defined by its processor's global voltage or a set of voltage level.

Scheduling has to make sure that only one task can run on a resource at a time. If there is communication between tasks, resource constraints on the communication links have also to be taken into account, where each directed arc $l_{i,j} = (p_i, p_j)$ represents a physical unidirectional channel connecting two nodes p_i and p_j. The weight of the edge $l_{i,j}$ denoted by $bw_{i,j}$ represents the bandwidth capacity for the edge $l_{i,j}$ in flit/time.

2.1.2 Application

Any application that will run on the target architecture can be represented by a set of directed acyclic graphs (DAGs) D(T,C) in which the nodes or vertices depict the application tasks T and edges stand for intertask dependencies C. Each DAG has:

- At least one entry node (a node with no incoming dependencies).
- One or several exit nodes (nodes with no outgoing dependencies).
- A period equal or greater than the latest deadline of its task set T.
- The property to be either periodic or aperiodic.

2.1.3 Task model

$G(C, E)$: is a synchronous dataflow graph, with each vertex $c_i \in C$ represent a task has the following properties:

- $d(i)$ deadline which represents the time when t_i has to finish.
- $pr(i)$ the task priority level that is given based on the dependencies between tasks.
- An arrival time $Ar(ci)$,which stands for the time i becomes known to the scheduler.
- An array $R(ci)$, where the j-th element $c_i^j \in R_j$ gives the execution time of task ci if it is executed on the j-th processor in the architecture (NoC tile).
- $w(i)$ the task workload given in clock cycle unit determine the amount of work to be determined.
- Dependency Properties:

Each directed arc $e_{i,j} \in E$ characterizes the dependency between c_i and c_j. Dependencies are either control dependencies which indicate that c_j can not start its execution until c_i has finished, or data dependencies, which imply that c_i communicates with c_j. Each $e_{i,j}$ has associated with it $v(c_{i,j})$, which stands for the communication volume in bits from c_i to c_j.

2.1.4 Benchmarks' multimedia applications (Case of study)

The following describes a detailed definition of the used multimedia applications in our implementation, whereas in Figure 4.1 and Figure 4.2 illustrate the task graphs of those applications.

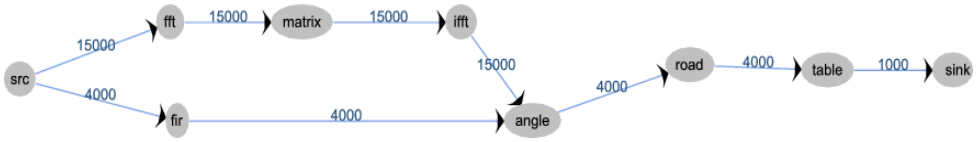
a. Automotive /industrial application

The first is the automotive/industrial application that is made of four communication task graphs (CTG): 0, 1, 2 and 3.

CTG 0 models an embedded automotive system that is composed of two controller area network (CAN) interfaces, a basic integer and floating point module and an actuator driven by pulse width modulation (PWM) signal.

CTG 1 describes an embedded automotive system that is composed of an infinite impulse response (IIR) filter and an inverse discrete cosine transform (iDCT) module.

CTG 2 models a system with: finite impulse response (FIR) filter, fast fourier transform (FFT) module, matrix arithmetic module, inverse fast fourier transform (iFFT) module, angle to time convertor, road speed calculation and table lookup and interpolation.



CTG 3 contains the following modules: pointer chasing (ptr), cache "buster" and tooth to spark.

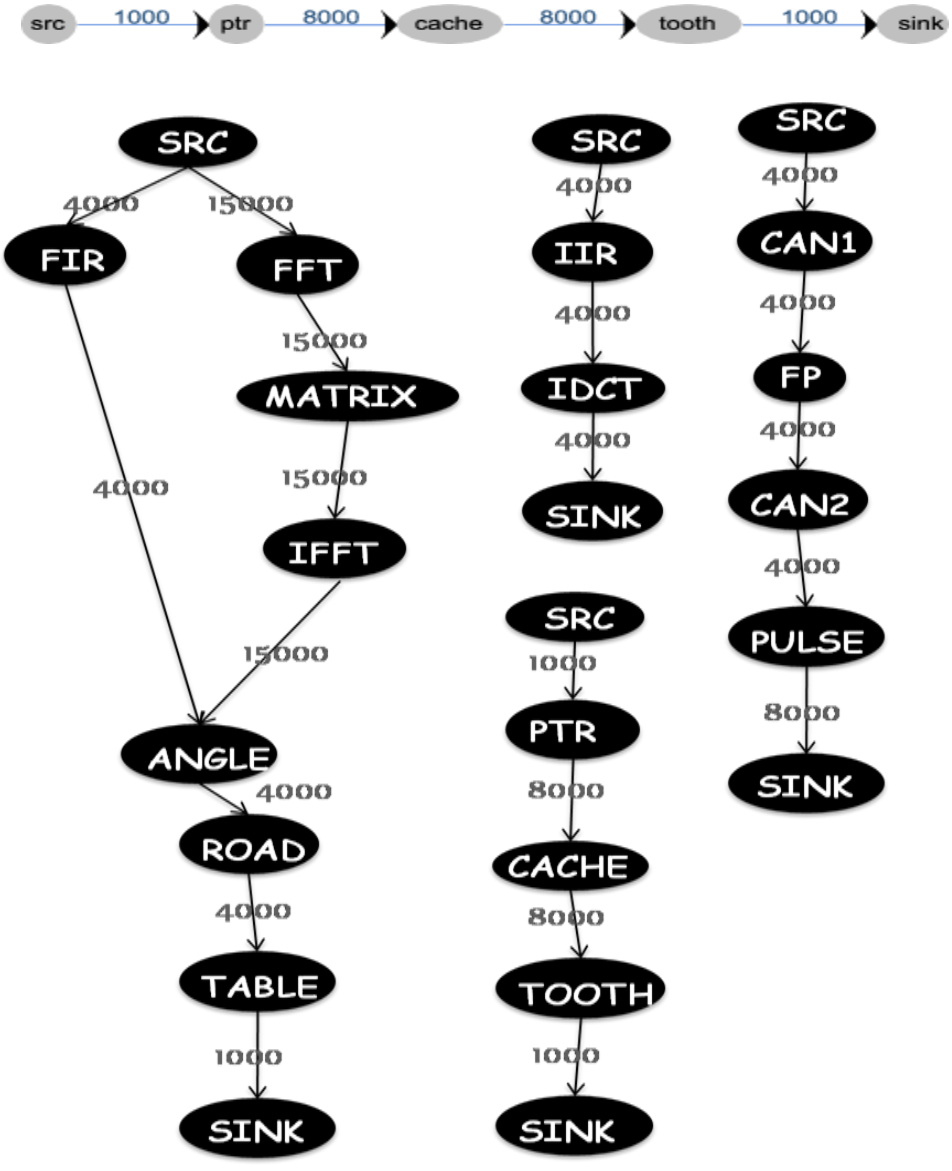


Figure 4.1. Automotive/industrial task graph [132].

b. Multimedia system (MMS)

A generic Multimedia System (MMS) was used in [132], [133] to test the performance of a Branch and Bound mapping algorithm on a real application. MMS is an audio video system. It contains an MP3 audio encoder, an MP3 audio decoder, an H.263 video encoder and an H.263 video decoder.

The MMS application was partitioned into 40 concurrent tasks. They were assigned to 16 cores in [132] and to 25 cores in [133]. We present next the Communication Task Graphs for the MMS application, which we derived from the Application Characterization Graphs (APCGs) [134]. The communications between tasks mapped on the same cores are ignored. Therefore, the used MMS CTGs are partial. They do not show all the communications between tasks, but all the communications required for building a 16 cores and 25 cores APCG are available.

CTG 0 was obtained from the MMS APCG from [132]. CTG 1 was obtained from the MMS APCG from [131]. We observe a single difference between the two CTGs: in CTG 0, task VLD (Variable Length Decoding) sends data to IDCT (Inverse Discrete Cosine Transform), while in CTG 1, VLD sends data to task IQ (Inverse Quantization).

Thus, we believe the two CTGs are essentially the same. However, in order to be compatible with previous research, we use both of them, exactly like in [132], [133].

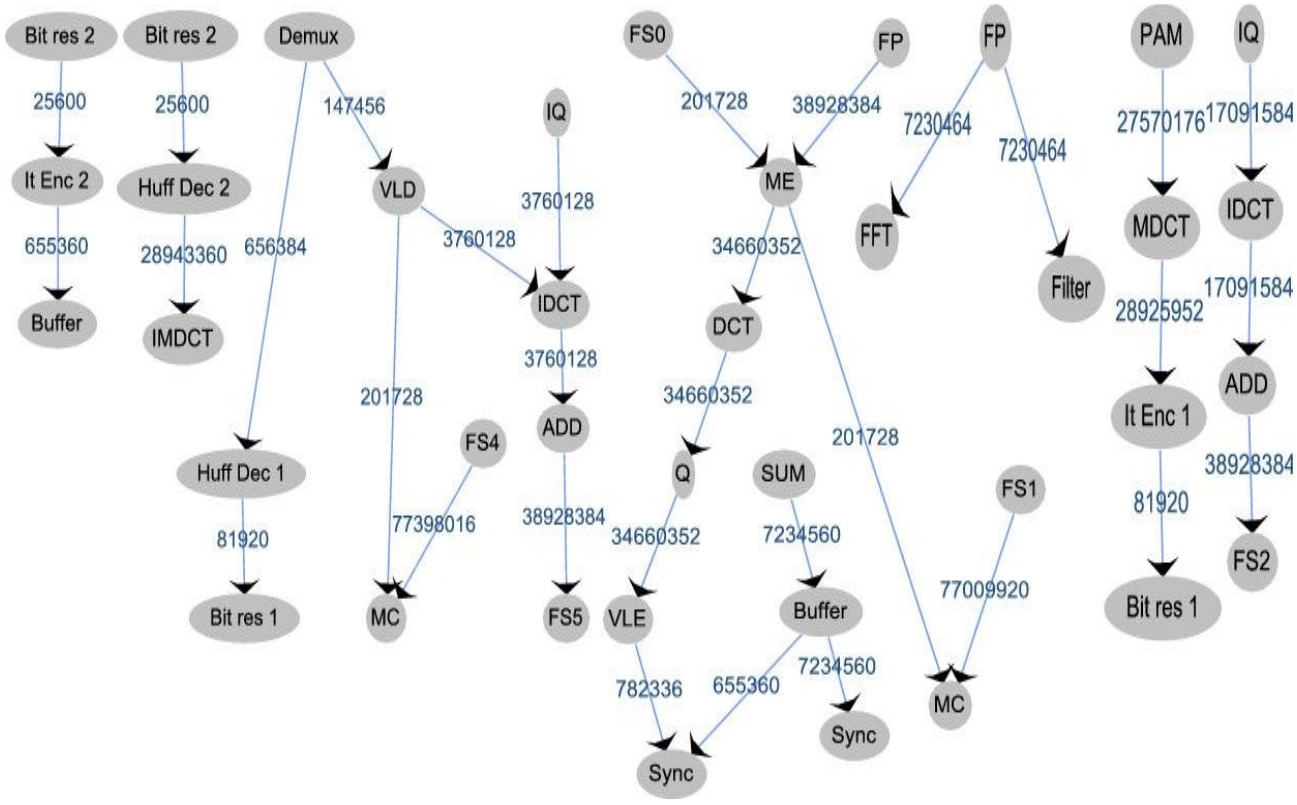


Figure 4.2. Multimedia system task graph [132].

We assume that each core has its own priority-ordered queue to schedule tasks, so for a set of tasks that are allocated to the same processor, a scheduling analysis for single processor can be applied to determine the task with the highest priority in each time. We used one of the following five scheduling analyses that are defined in Algorithm 4.1 (scheduling_based_task_workload on $core_i$, scheduling_based_task_deadline on $core_i$, scheduling_first_one_reached_its_deadline on $core_i$, scheduling_minimum_slack on $core_i$, scheduling_based_average_case_response_on $core_i$) , where each one is defined independently, i.e., if we choose to work with scheduling_based_task_workload, the scheduling of the tasks will be done based on the execution of Algorithm 4.2.. For that before starting each execution one of the scheduling analyses should be selected.

ALGORITHM 4.1. SCHEDULING ANALYSES
Given: $w_{c_i = c_{i_{workload}}}$, $d_{c_i = c_{i_{deadline}}}$, N: number of cores in NoC;
Find: priority ordered queue for each core;
FOR $i \leftarrow 0$ *to* N DO
Get scheduling technique
IF ($Core_{i - nb\_allocated\_task} \geq 0$) then activate scheduling_based_task_workload on $core_i$;
IF ($Core_{i - nb\_allocated\_task} \geq 0$) then activate scheduling_based_task_deadline on $core_i$;
IF ($Core_{i - nb\_allocated\_task} \geq 0$) then activate scheduling_first_one_reached_its_deadline on $core_i$;
IF ($Core_{i - nb\_allocated\_task} \geq 0$) then activate scheduling_minimum_slack on $core_i$.
IF ($Core_{i - nb\_allocated\_task} \geq 0$) then activate scheduling_based_average_case_response_on $core_i$.
END IF
ENDFOR

2.2 The task workload

Scheduling_based_task_workload is a scheduling strategy where each task should processed a workload that is defined by the amount of job that should be determined and it is given in clock cycle unit, and then the tasks are ordered based on their workload and the highest priority is given to the task with a bigger workload and the lowest priority is given to the task with the smallest workload and then the execution is done based on that order. And in case of more than one task are allocated to the same processor, the task with highest priority is chosen to be executed and the rest should wait in the current buffer until it will be free and on condition that there aren't any other high priority tasks, for that we thought of a way to eliminate the

buffer wait time (see figure 4.7 and session 3.1), and the task wait The algorithm is given as follows:

ALGORITHM 4.2. SCHEDULING_BASED_TASK_WORKLOAD

Given: $w_{c_x = c_{x_{workload}}}$, $Core_i$: the core to be scheduling.

FOR $k \leftarrow 0$ *To* *nb _ allocated _ task* *Do*

IF (c_k is activated) then *sort* $\uparrow (w_{c_k})$.

Assign priority to c_k .

ENDFOR

$k \leftarrow 0$;

WHILE ($Core_i$ still allocated)

IF(c_k is activated) THEN

IF(c_k is on prior) THEN Execute c_k

$k \leftarrow k + 1$;

ELSE (c_k is in wait_for_PE)

ENDIF

ELSE c_k is not released

ENDIF

ENDWHILE

2.3 The task deadline

In soft real-time system, the deadline is the amount of time where it is better for the task to finish its execution, and as it is said “better”, it is allowed to be passed but with a little threshold (the threshold is given in clock cycle unit before and it is determined before the execution start in such a way that, the multimedia application quality won’t be destroyed), Whereas in hard real-time system the task should finish its execution without considering the threshold. The hard real-time system is not considered since we are only concerned with the multimedia application. The tasks are ordered in such way that the highest priority is given to a task with the minimum deadline and the lowest priority is given to the task the maximum deadline. And in case of more than one task are allocated to the same processor and the letter one is free, the task with the highest priority is selected to be executed, and the others have to wait on the current buffer. The algorithm is given as follows:

ALGORITHM 4.3. SCHEDULING_BASED_TASK_DEADLINE

Given: $d_{c_x = c_{x_{deadline}}}$, $Core_i$: the core to be scheduling.

FOR $k \leftarrow 0$ *To* *nb _ allocated _ task* DO

IF (c_k is activated) THEN *sort* $\downarrow (d_{c_k})$;

Assign priority to c_k ;

ENDFOR
$k \leftarrow 0$;
WHILE ($Core_i$ still allocated)
IF (c_k is activated) THEN
IF (c_k is on prior) THEN Execute c_k
$k \leftarrow k + 1$;
ELSE (c_k is in wait_for_PE)
ENDIF
ELSE c_k is not released
ENDIF
ENDWHILE

2.4 The first one reached its deadline

Scheduling_first_one_reached_its_deadline is another scheduling strategy where the difference between each task workload and its deadline is calculated and the result defined by the remain for the task to reach the deadline (the remain is given by clock cycle unit). The highest priority is given to the minimum remain, and the lowest priority is given to maximum remain. The tasks are ordered based on the nearest workload to the deadline of each task without considering the actual time (it is static). And it is the same as before, if there was more than one task allocated to the same processor, only one task would be selected and the others should wait on the buffer of that current processor, till it will be free and of course with satisfying the condition of possessing the highest priority. The algorithm is given as follows:

ALGORITHM 4.4. SCHEDULING_FIRST_ONE_REACHED_ITS_DEADLINE
Given: $w_{c_x = c_{x_{workload}}}$, $d_{c_x = c_{x_{deadline}}}$, $Core_i$: the core to be scheduling.
FOR $k \leftarrow 0$ *To* $nb\_allocated\_task$ DO
IF (c_k is activated) THEN $sort \downarrow (w_{c_k} - d_{c_k})$.
Assign priority to c_k.
ENDFOR
$k \leftarrow 0$;
WHILE ($Core_i$ still allocated)
IF (c_k is activated)
IF (c_k is on prior) then Execute c_k
$k \leftarrow k + 1$;
ELSE (c_k is in wait_for_PE)
ENDIF
ELSE c_k is not released
ENDIF
ENDWHILE

2.5 The minimum slack

The minimum slack is defined by the minis of the task workload and task deadline with considering the actual time, so this strategy is dynamic, for that the task with the minimum slack, has the highest priority. The tasks are ordered based on the current priorities, but this order will change in each selection of a new task because the current time is considered; for that the priorities of the waited tasks on buffer are calculated after the end of the execution of current task on the processor. The algorithm is given as follows:

ALGORITHM 4.5. SCHEDULING_MINIMUM_SLACK

Given: $w_{c_x = c_{x_{workload}}}$, $d_{c_x = c_{x_{deadline}}}$, $Core_i$: the core to be scheduling, $Current\_time$: given in clock cycle.

WHILE ($Core_i$ still allocated)

FOR $k \leftarrow 0$ *To* $nb\_allocated\_task$ DO

IF (c_k is activated) THEN $sort \downarrow (Current\_time - (w_{c_k} - d_{c_k}))$;

Assign priority to c_k ;

ENDFOR

$k \leftarrow 0$;

IF (c_k is activated) THEN

IF(c_k is on prior) then Execute c_k

$k \leftarrow k + 1$;

ELSE (c_k is in wait_for_PE)

ENDIF

ELSE c_k is not released

ENDIF

$Current\_time++$;

ENDWHILE

2.6 The average response time

Scheduling_based_average_case_response_on $core_i$ that is defined in Algorithm 4.1 is a scheduling strategy where the average response time of each task as it is shown in Equation.2 is calculated, r represents the average response time. The highest priority is given to the task with the minimal value of r, and the lowest priority is with the maximum value of r. This strategy can be used as static or dynamic scheduling in such way that, if we calculated each task's r once and the values obtained are considered before determining the priorities before the first execution, this would be static scheduling. But if we calculated r in each task selecting while considering only the waited tasks in the buffer, this means that the finished tasks are not concerned with

calculating r of the waited tasks, and this would be dynamic scheduling. The equation of calculating r is given as follows:

$$r_i^{n+1} = c_{i\_workload} + \sum \forall c_j \in H\_priority \quad (Ci) \left\lceil \frac{r_i^n}{period} \right\rceil c_{j_{workload}} \qquad (2)$$

$H\_priority$: denotes the set of higher priority tasks running on the same core as task i.

3. Contribution in routing and switching strategy

In our implementation, we exploited the same ideas presented by [138] but with many differences. In this thesis, we used the internal block diagram of a 4-place FIFO buffer, while proposing a new scheme of coding the register in wormhole, and in the same time, the routing of messages inside the router follow the randomized routing algorithm and during the transmission, the head flit is the only to be buffered and the router always choose the first available direction, for that we thought of adding some metrics in the randomized routing to avoid the misrouting contentions like deadlock and livelock, and to optimize the energy consumption during transmission and minimal latencies with greater throughput. The energy in the buffer is computed as follows:

$$Energy_{FIFO} = P_{write} + P_{read} + P_{clk} + P_{\text{int}} \qquad (1)$$

$$P_{write} = r_w P_{control} + \alpha P_{store} \qquad (2)$$

Here, r_w is the rate at which write actions occur, $P_{control}, P_{store}$ are the average power consumed at the control unit for one write action and the power consumed to store the data in the FIFO respectively.

$$P_{read} = r_r P_{control} + \alpha P_{retrieve} \qquad (3)$$

Here, r_r is the rate at which read actions occur, $P_{control}, P_{retrieve}$ is the power consumed at the control unit and the power consumed to retrieve the data from the FIFO respectively.

P_{clk} Clock activity causes permanent constant power consumption due to switching activity.

P_{int} Which is the result of the internal short circuits occurring during switching of bits in the incoming data, the switching of read and write actions and clock activity. Therefore Pint can be calculated as follows:

$$P_{\text{int}} = r\ k_1 + k_2\ \alpha + k_3 \quad (4)$$

Here, r is the rate of read and write actions, α is the amount of bit flips, the constants k_1, k_2 are respectively the average internal power consumed for the control of read and write actions and due to bit changes in data and k_3 is due to the clock activity.

3.1 Bufferless wormhole routing

In our implementation, we exploited the same ideas presented by [138] but with many differences. In this thesis, we used the internal block diagram of a 4-place FIFO buffer, the scheme is illustrated in Figure 4.3. It is mainly composed of a FIFO control unit, a 1-to-4 channel input unit (similar to a de-multiplexer), a 4-to-1 channel output unit (similar to a multiplexer) and four registers to store the data. The size of the register is the size of head flit where the header contents only destination bytes ID bytes and Next-hop bytes.

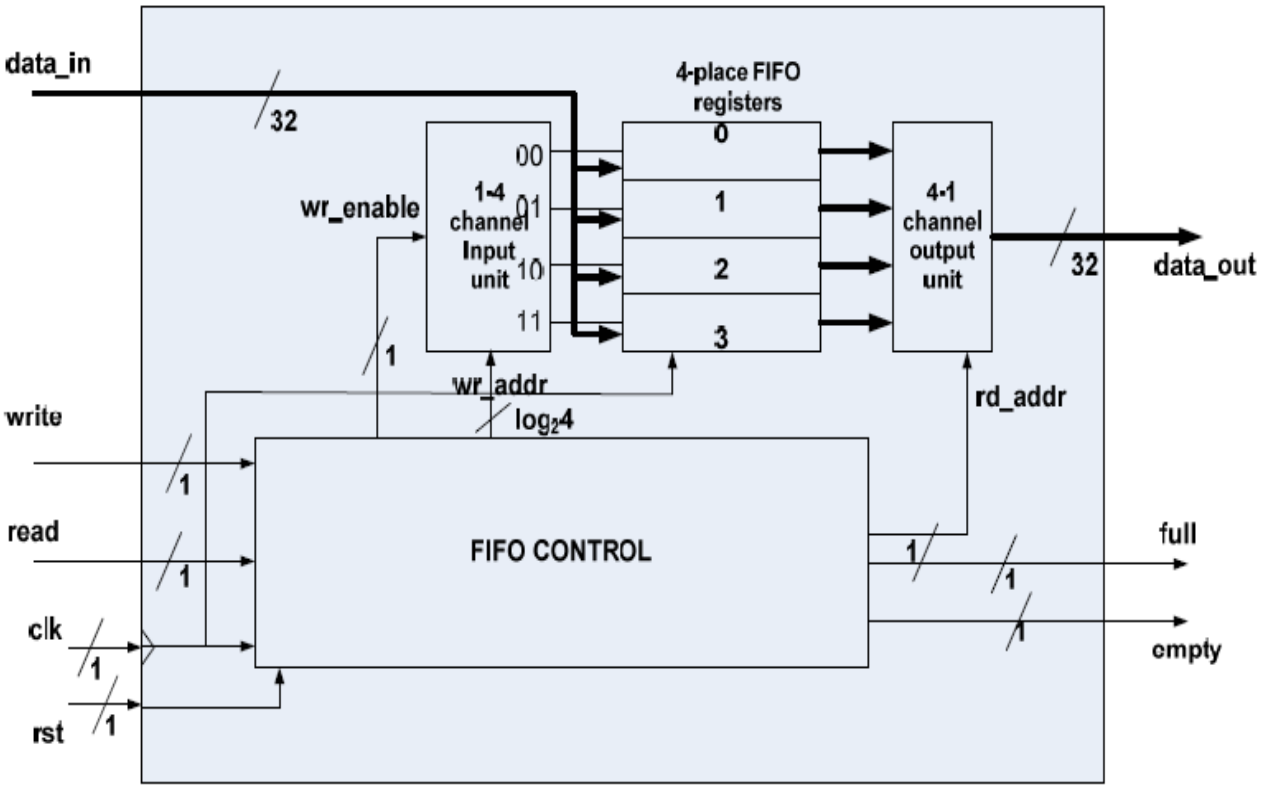


Figure 4.3. Implementation of Register based FIFO Buffers.

Destination bytes: represent the destination to be reached with a given message, and it is encoded into 5 bytes (according to Mesh dimension 5x5=25 and the binary of 25 have 5 bytes), the destination bytes allows the flits to know if they arrive at the destination node.

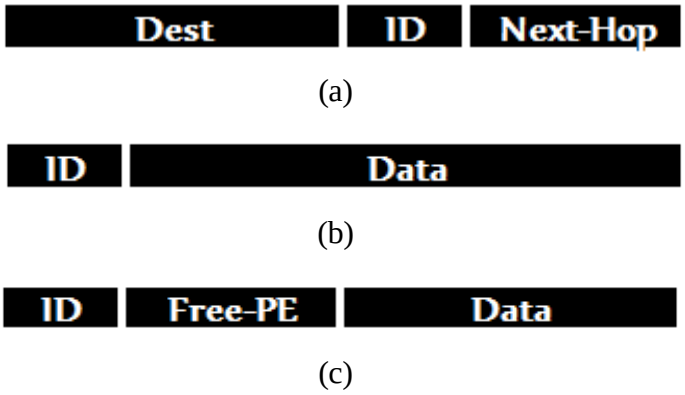


Figure 4.4. Wormhole flits format. (a) Head-flit format, (b) Body-flit format, (c) Tail-flit format.

ID bytes: In our application, we replaced the bytes of source and virtual channel by two bytes called destination identity (ID) in order to minimize the size of each register (the register has the size of a flit), in our encoding we only deleted the virtual channels bytes and we kept the virtual channels, and used them indirectly. The ID is in the form of (00, 01, 10, and 11 as illustrated in Figure 4.5) which indicates the current used registers, for example, if we are in the current register number "2", the ID would be "10" (we used 4 registers as mentioned before). First the arbiter verifies the number of the occupied destination's register on FIFO buffer and allocates the next one for the incoming head. Example: if there are 2 registers already in use ("00" and "01"), the next ID would be "10". This head's encoding minimizes the energy consumption comparing to the old flit encoding, due to the minimized read and the write actions of head flits, and also from the decreased transmitting of the head from node to another. The encoding of the body and tail flits are the same used in [139], but in place of virtual channel bits, we use ID bits.

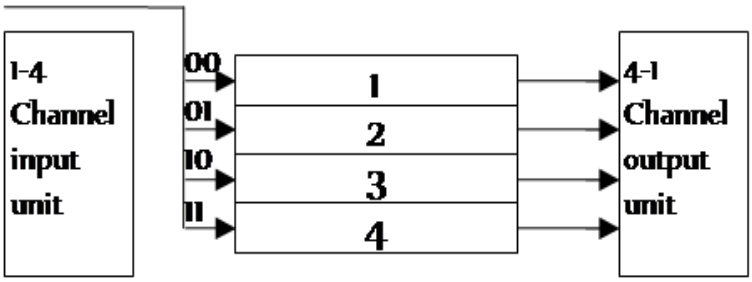


Figure 4.5. The ID bits of head flit.

Next-hop bytes: The head flit allocate a path for the incoming flits by leaving a trace on each buffer, the next-hop is in the form of 00, 01, 10 and 11 as it is shown in Figure 4.6, and it defines the output port to be crossed for the incoming flits. Once the destination is reached the tail-flit has to free the buffers, ID and next-hop.

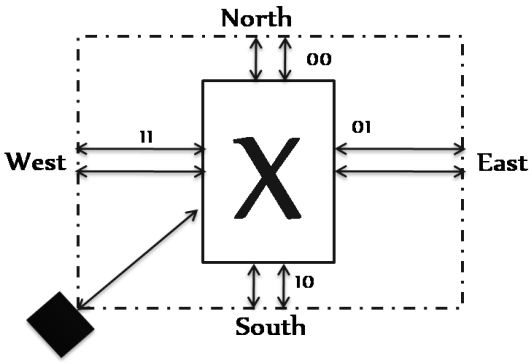


Figure 4.6. Next-hop bits of head flit.

3.2 Based deadlock factor randomized routing algorithms

The randomized routing algorithm is used for its simplicity and path allocation diversity, but unfortunately it is getting less used especially in the last decade for the reason that this algorithm is more affected by deadlock or live-lock. Another reason of non-using this algorithm is its long latency and much power needed; the router always chooses the next path randomly without considering the shortest path availability. As it has been mentioned before, during transmission the head flit is the only to be buffered and the router always choose the first available direction, for that we thought of adding some metrics in the randomized routing to avoid the misrouting contentions like deadlock and livelock, and to optimize the energy consumption during transmission and minimal latencies with greater throughput, the algorithm is defined as follows:

ALGORITHM 4.6. RANDOMIZED_ROUTING_ALGORITHM_WITH_DEADLOCK_FACTOR

Given: source, destination N: neighbours

WHILE ((Source != Destination) AND (loop <10)) DO

FOR N= (1..4) DO

Select randomly next hop x;

IF processor X ≠ free THEN increment deadlock probability;

ENDIF

IF buffer X = full THEN increment deadlock probability;

ENDIF

add saved probability to N

ENDWHILE

Allocate hop with minimum deadlock probability;

In classical randomized routing algorithm (RRA), the next path is randomly selected and checked for channel availability, if there is at least one free channel the head flit reserve the path for the incoming flits, otherwise the flits need to wait on the current hop buffers but the buffers may not have sufficient space to store the incoming flits. A turn deadlock could occur for flits and it would not have any choice but to backup and set another path. In our simulation we saw that the deadlock may occur many times for the incoming flits, and since we aim to minimize the latency and saving power, several backups and deadlocks are not a better option.

In based deadlock factor randomized routing algorithm (RRA based deadlock) the next path is randomly selected and the credit-based inside the node is activated and checked for channel and buffer availability otherwise the deadlock factor is incremented. We modified the randomized routing in order to find the next hop for the head flit based on the deadlock factor, this factor is incremented if the routing algorithm finds a deadlock, and the arbiter selects the next hop according to the minimum probability. We also save the probabilities of deadlock on each mapping between each source and destination and add it to the probability if there was a transmission between the same source and destination, but only on the current mapping that is mean for next mapping the probabilities are reinitialized to null.

With the use of VC (VCs number is head's ID) and randomized routing algorithm we can guarantee a diversity of paths and also forwarding flits without buffering.

Before adding the saved probabilities to the algorithm, the loop was 100, we tested for deadlock about 100 times, but now it is only 10 times and we achieved a fast routing decision.

4. Contribution in task migration

Task migration has been widely used in the distributed systems domain, but as a drawback, the timing overhead of migration is larger and may increase network congestion and degrade the system performance. Due to that, we migrate the task only toward one of the neighbors if the following conditions are satisfied:

$$\sum_{k=0}^{M} P_j\left(C_{k\_ext}\right) > \sum_{k=0}^{M} P_j\left(C_{k\_dead}\right) \tag{7}$$

$$\exists P_{j1} \in neighbour_{Pj}, \sum_{q=0}^{M1} P_{j1}\left(c_{q\_ext}\right) < \sum_{q=0}^{M1} P_{j1}\left(c_{q\_dead}\right) \tag{8}$$

If (Equation.7) and (Equation.8) are satisfied then migrate one of M to $P_{j_{neighbors}}$.

Where M, M1 are the number of allocated tasks to processor j and j1 during one period of time respectively. Equation.8 indicates that if the sum of execution time of tasks mapped to the

same processor leads to missing them deadline, and if one of neighbors of that processor executing M1 tasks without deadline missing, then migrate one of M1 tasks onto that neighbor. The interpretation of those equations is illustrated in Algorithm.4.7. The algorithm of the task migration is given as follows:

ALGORITHM 4.7. TASK_MIGRATION_ALGORITHM

```
FOR j = 0 to NUMBER_OF_PROCESSOR DO
 FOR k =0 to M DO
   SOM = SOM + P_j (C_k_ext);
   SOM1 = SOM 1+ P_j (C_k_dead );
 ENDFOR
 IF SOM > SOM1 THEN
  FOR j1 = 0 TO P_j _ neighbours DO
    FOR q = 0 to M1 DO
      SOM2 = SOM2 + P_j1 (c_q_ext);
      SOM3 = SOM3 + P_j1 (c_q_dead );
    ENDFOR
    IF SOM2 < SOM3 THEN Migrate S: one of P_j to P_j1;
                        Copy S_CPU_State to P_j1;
                        Copy S_STACK to P_j1;
                        Copy S_Heap_data to P_j1;
                        Copy S_Program_code to P_j1;
                        Copy S_Association to P_j1;
                        Copy S_Global_variables to P_j1;
    ENDFOR
ENDFOR
```

Assuming a migration is taking place between two CPU cores. The target core must have all the necessary information needed for resuming the migrated task, and all the information associated with the task e.g. communication between tasks. This information is sent to the target core either by means of temporal storage in shared memory or some other way of direct communication. The following information is needed to resume a migrated task:

4.1 CPU state

A task needs to know its CPU state, in both a context switch and in a task migration. The CPU state is easily read from the CPU status register and is stored in a temporary variable. This variable can be migrated onto the other CPU core.

4.2 Task stack

The task stack holds the variables and its values used in the task. The task stack has a similar structure as the system stack and is allocated in the memory heap. The task stack is fairly easy to migrate because the system knows where the stack begins and its size. A similar stack could then be allocated to the other core, and the variables could then be restored.

4.3 Heap data

Data allocated in the heap is difficult to migrate since the whole core uses the same heap. Variables associated with the relevant task must be somehow marked to inform the migration mechanism which data to migrate. Problems could eventually occur when determining the size of dynamically allocated arrays in the heap. Assuming there is a mechanism for linking heap data to tasks, the data could be migrated and re-allocated on the other core. The pointers to the data must be translated upon such a re-allocation.

4.4 Program code

Another difficult part to of migration is the executing code in the program memory. The program code is statically stored, and must also be marked with associations to the relevant task. The program counter in the system must continue exactly from the migration point relative to the program. The migration of the program code is also substantive when considering the run-time update, which would switch in-and-out program code while a task is running. The migrated code should, in summary, continue its process where the first core was stopped.

4.5 Task associations

Various associations exist between tasks, such as information exchange, child tasks, etc. A task is able to communicate with other tasks using message queues [144], which can only be used if the tasks are executing on the same core. Otherwise the interface of communication must change, so that the tasks communicate over some kind of inter-core communication. The task calls between cores must be handled either in a way that the tasks pass values between cores, or so that called tasks also migrate to the same core as the caller. This could of course pose problems with other tasks that call the same task.

4.6 Global variables

Several tasks could use the same global variables, which leads to a synchronization problem. This could possibly be a part of task associations, where the variable itself is not

difficult to migrate. For security reasons tasks that are selected for migration should preferably not share global variables with other tasks, since the communication using global variables could be broken or slowed down if one of the tasks is moved to another core.

5. Contribution in dynamic voltage/frequency scaling

A main idea of our DVS is the definition of a voltage change point for the processor based on the allocated task workload. A change point is where the processor can change its voltage level during the allocated task execution time to achieve an end to end deadline and save the power, the relationship between the frequency and the voltage are given in Equation.9 where V_{th} is the voltage threshold and $\alpha = 1.5$. Equation.10 represent the formula of calculating V_{th} and V_{th1}, K_1 and K_2 are parameters depend on the technology used. A multiple level of voltage is used for our DVS (0, 6 - 1, 0).

$$f = \frac{(V_{dd} - V_{th})^{\alpha}}{V_{dd}} \tag{9}$$

$$V_{th} = V_{th1} - K_1.V_{dd} - K_2.V_{bs} \tag{10}$$

The following algorithm define the proposed idea of scaling the voltage, we first calculate for each task its execution time by using the lower voltage level of the processor ($ET_{ci}(w_{ci})$) and the task workload, and then calculating the execution time by using a random voltage level (we generally used the middle) of the processor ($ET_{ci}(d_{ci})$) but this time with the task deadline. Then a difference between them is calculated and it is called a slack, if the slack less than 0 then that means that the low voltage level can't be used because it will lead to deadline miss, for that this voltage level would be rejected and the algorithm is restarted with a higher level than the initial ones. But, in the opposite case where the slack is higher than 0, a search for the voltage change point would be started. First the slack is transferred into clock cycle unit ($slack_{unit}$) , so if the $slack_{unit}$ is higher than the task execution time that is mean if the chosen voltage level leads to deadline miss, for that the initial chosen level will be incremented, else $slack_{unit}$ is the point where the processor can change its voltage level. And at last the task execution time is calculated based on that point. The algorithm of DVS is as follows:

ALGORITHM 4.8. DYNAMIC_VOLTAGE_SCALING

Given: $w_{ci} = c_{i\_workload}$, $d_{ci} = c_{i\_deadline}$, $x = one\ of_{(pj\_level(1..N))}$ $y = \left(p_{j_{level_0}} + p_{j_{level_n}}\right)/2$, $z = p_{j_{level\ 0}}$

Find: Slack= the change point (in millisecond)

Slack_unit = the change point (in clock cycle)

$ET_{ci}(w_{ci}) = execution_{time}(c_i\ on\ p_j - level\ (z)) = w_{ci} / f_{j_{level\ z}}$

$ET_{ci}(d_{ci}) = execution_{time}(c_i\ on\ p_j - level\ (y)) = d_{ci} / f_{j_{level\ y}}$

IF $ET_{ci}(w_{ci}) > ET_{ci}(d_{ci})$ THEN

$$slack = \frac{ET_{ci}(w_{ci}) - ET_{ci}(d_{ci})}{ET_{ci}(w_{ci}) \times 100\ \%};$$

$$slack_{unit} = \frac{slack \times w_{c_i}}{100\ \%};$$

$$slack_{unit} = slack_{unit} \times (p_{j_{level\ n}} - p_{j_{level\ x}});$$

IF $slack_{unit} > w_{ci}$ THEN

C_i will miss its deadline on p_j at level z;

Increment z level;

Increment x level;

Repeat DVS;

ELSE

$$ET_{c_i} = \frac{w_{c_i} - slack_{unit}}{f_{j_{level\ 0}}} + \frac{slack_{unit}}{f_{j_{level\ x}}}$$

ENDIF

ENDIF

6. Quantum evolutionary cellular automata in network on chip

The main contribution in this thesis is the use of a hybrid technique combining evolutionary genetic algorithm with cellular automata (CA) while using a different strategy of mapping of multimedia application onto 2D mesh based NoC in order to save power with respecting multimedia timing constraints. The first use of cellular automata (CA) with Genetic Algorithm was with Packard, then Mitchell and al where the optimization addressed the density classification problem [156][157], these discoveries are encouraging for the prospect of using GAs to automatically evolve computation for more complex tasks such as cryptography [158] and in more complex systems, the multiprocessor scheduling [159], [160] [161] and synchronization [162], [163], [164]. Moreover, evolving CAs with GAs gives us a tractable framework to study the mechanisms of an evolutionary process that might create complex coordinated behavior in natural decentralized distributed systems and that's what encourage its using in embedded system. In [155], CA was used to model and simulate real time tasks scheduling and allocation problems in multi-cores embedded systems. In this thesis, we will address the use of Evolutionary Cellular Automata (ECA) on Network on Chip where the objective is to find a set of rules that can successfully solve problems in a significant computational time. We had also proposed a new DVS algorithm defining a point where the processor can change its voltage level during the task execution time to achieve an end to end deadline and to save the energy of the execution.

6.1 Cellular automata principles

As it has been mentioned before in session 1, the finite automate is defined by the set of the following main characteristic variables:

$$\begin{cases} V : number\ \ of\ \ cell\ \ states \\ H : number\ \ of\ \ neighbours\ \ \ per\ \ each\ \ cell \\ f : transition\ \ \ rule \end{cases}$$

the transition rule (s) f defines a rule of updating cell i, and due to that it is considered as a very sensitive part, since CA is changed in time and in a synchronous manner based on those rules, for that a strict formulation of f is what guarantee the robustness of CA. The transition rule (s) is expressed in a table called "look-up table" which lists for each local neighborhood the state's update of the neighborhood's central cell where the neighborhood relation and the transition function are the same for every cell $\{0..n-1\}$, Formally, the evolution of CA at time $t+1$ is expressed by the formula:

$$\begin{cases} s_i(t+1) = f\left(\{ s_{j \in H(i)}(t) \} \right) \\ H(i): neighbours\ \ \ list\ \ of\ \ cell\ \ i \end{cases} \quad (15)$$

A cellular automaton is also defined by its lattice of N cells where each cell is in one of K possible states at time t. Each cell follows the same rule to update its state (see Table 3.1). The state s of a cell at a time $t+1$ depends on its state and the states of a number of neighboring cells at a time t. The CA starts with an initial configuration (CI) of cells and at each time step the states of all cells in the lattice are updated in a synchronous manner.

H(i)	0 000 0	0 000 1	0 001 0	0 001 1	0 010 0	….	1 011 1	1 111 1
New state	1	0	1	1	0	….	0	1

Table 4.1. Neighborhood look-up table.

An illustrative example of look-up table is given above, if one considers two-dimensional CA with two states and five neighbors state radius $H = 5$, the look-up table for updating cells is given as in Table 3.1.

In this work, we used the cellular automata with the quantum genetic algorithm in purpose of lowering the energy consumption in NoC based 2D mesh while respecting the soft real-time constraints of the multimedia application, where cells can have only two possible

states (0,and 1). The CA dimension is 2-D lattices. The number of neighbors that the transition rules take into account while updating each cell are $H = 5$ (4 neighbors + the cell in the middle). We choose $H = 5$ according to the structure of the 2D mesh NoC topology where each node is connected to four other neighbors' nodes (see Figure 3.23). According to this structure, we obtain $card\ (V)^h$ different neighbor configurations and in our case is $card\ (2)^5 = 32$, and each cell is updated based on those configurations. The cellular automata can have more than one transition rule, and each transition rule consist of a set of values that represent the updated value of each CA cells based on theirs (CA cells) neighbor configurations. The number of the transition rule that can be associated to each CA is the $card\ (V)^{card\ (V)^h}$, and in our case it is calculated as follows:

$card\ (2)^{card\ (2)^5} = 2^{32} = 4294967296$ Transition rule.

In the following figure, an example of how CA works is illustrated. The state of the cell is changed in time according to its previous state and its neighbors' states, so the corresponding decimal value is 28. The new state of the middle is the transition rule cell with position = 28.

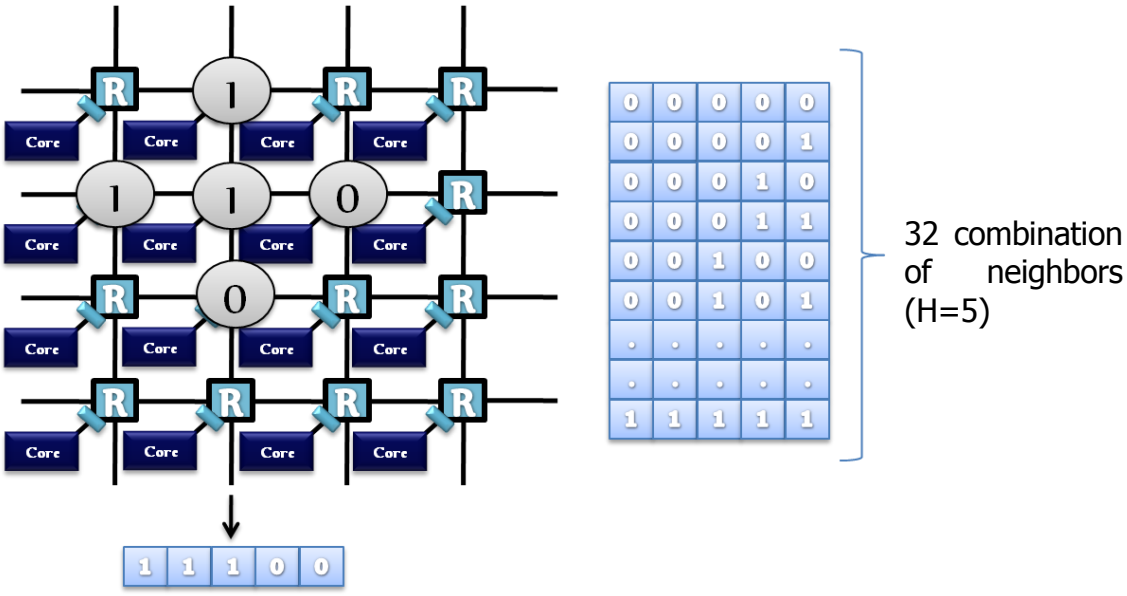


Figure 4.7. The combination of neighbours in CA based on NoC neighbours.

6.2 Problem description

In many researches done, the mapping problems used to be solved the most by genetic algorithm, but the main problem appears in the use of GA is after many CA operations between crossover and mutation, the new individuals will be adjusted to be suitable and accepted as a valid mapping solution that represents the chromosome, the following is an example illustrates

this issue, given a (2x2) processors and 7 tasks to be mapped, the structure of the chromosome is given in the following chapter.

As it is seen C6 and C4 from Chromosome2 are allocated to two processors P2 and P4, whereas C6 and C4 from Chromosome1 are not allocated to any processor. And adjustment phase is needed to adjust and modify the chromosome, so it can be accepted as a valid solution. And as the number of tasks augmented, this issue augmented as well.

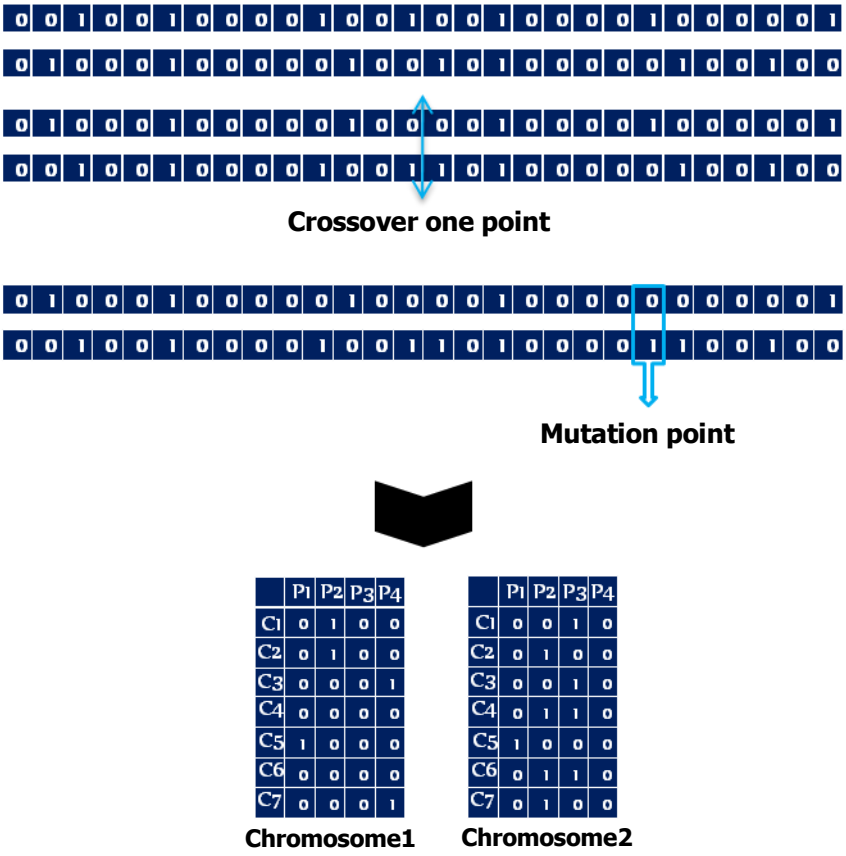


Figure 4.8. Example1 of GA issues.

The use of GAs still the subject of intensive study to improve their performance, general knowledge about the population is unachievable, after many generations (talking about NP hard complete problems where the search space is huge and the chromosome size is long and in each mutation and crossover, the new generation could be changed completely compared to the older generation) , the use of the old instance of populations or the complete population is impossible, for that a new searching process must be started from the beginning by creating an initial random population of potential solutions, but the probability of achieving the same results or using the exact same searched instance or a complete population is quite small, this problem is considered as the main problem in the use of GA.

Another issue of using GA is the dependence between multimedia application tasks and the chromosome structure; the chromosome represents one of the mapping solutions so the number of tasks and NoC dimension are demanded before starting because their sizes represent the chromosome size. Adding or removing tasks leads to chromosome change, and that will affect the results obtained from GA operators (crossover and mutation). So instead of restarting the mapping in each time, a fixed chromosome size that will be suitable for any application is needed.

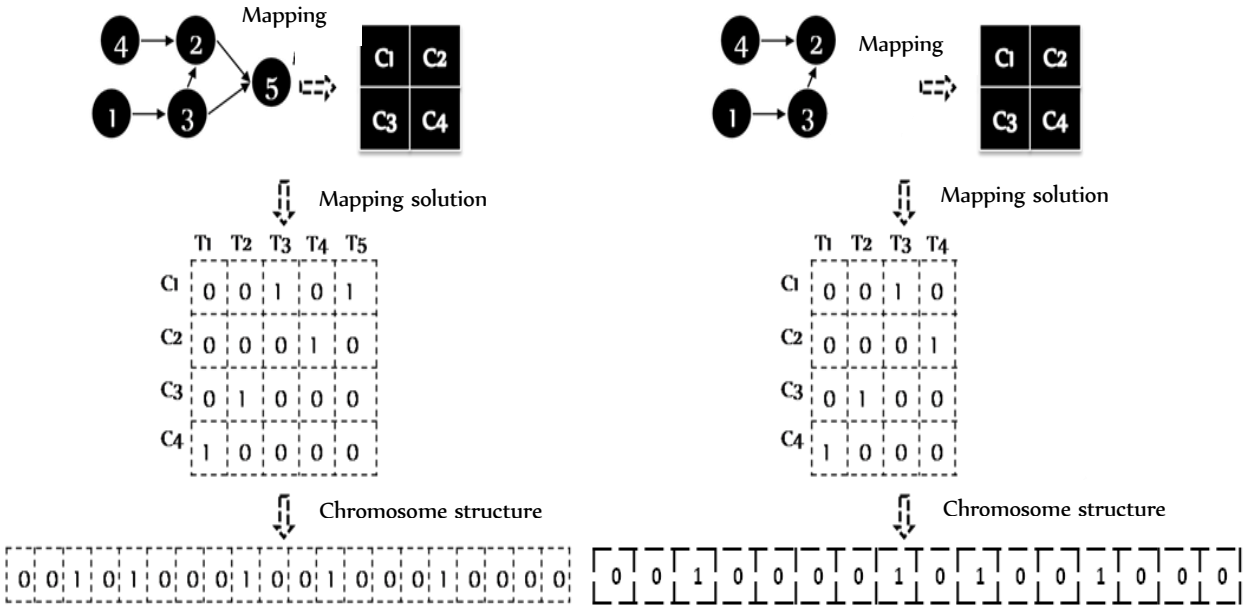


Figure 4.9. Example2 of GA issues.

Also a problem may appear is the stabilization of the results obtained that means after a specific number of iterations, the new obtained populations will leads the same results obtained from the older and due to that the algorithm will stop even with better results still could be achieved and it would look like blocked.

6.3 Solution description

Quantum computing has been surpassing classical computers for its ability to solve problems of polynomial-time that classical computers considered it as impossible to be solved but with a million years [165]. In Quantum Computers information is stored in the form of qubits represented by the superposition of the two bits sets with varying probability.

Recently, a growing theoretical and practical interest is devoted to research on quantum computing and quantum computers and due to that many quantum computing algorithms have been developed, such as Shors factorizing algorithm [165], were explored such as Grovers [166], [167], quantum search algorithm [166] and Quantum genetic algorithm (QGA). Research

on combining evolutionary computation and quantum computing started in the late 1990s. Quantum evolutionary computing is a branch of study on evolutionary computation and employs the certain principles of quantum mechanics, such as superposition, interference and uncertainty [168] [169]. Based on the concept and principles of quantum computing, such as quantum bits (Q-bits), quantum gates (Q-gates) and superposition of states, Han and Kim [170] developed a quantum-inspired evolutionary algorithm (QEA), which can achieve a better balance between exploration and exploitation of the solution space and also obtain better solutions, even with a small population, compared with the conventional EAs. Moreover, in QGA qubit encoding is used to represent the chromosome, and the evolutionary process is implemented by using quantum logic gate operation on the chromosomes. Now, much attention is paid to QGA because it has the characteristics of strong searching capability, rapid convergence, short computing time, and small population size, a great capability of global optimization and a good robustness.

It is informally shown that the quantum inspired genetic algorithm performs better than the classical counterpart for a small domain, In [170], a comparison is made between a classical genetic algorithm and a quantum inspired method for the travelling salesperson problem with the use of quantum crossover to optimize the problem. An improved QGA based on multi-qubit encoding and dynamically adjusting the rotation angle mechanism was presented to separate the blind sources [173].

In [171] [174], It is demonstrated that GQA are many effective and superior comparing to GA while the experiments are applied on the knapsack problem on [171] and with the use of quantum mutation to improve performances and also it is applied on the knapsack problem in [172] and on the infinite impulse response digital filter design in [174]. In [175], population catastrophe operation and violent vibration have been improved with the use of QGA. In [176] [157], a detailed description of quantum computing and quantum genetic algorithms was presented.

On the other hand, The CA presents a highly parallel and distributed system of locally interacting units which are able to produce a global behavior [177], [178], [176]. CA can be considered as a model of naturally existing systems produced by natural evolution. Such systems are capable of producing globally coordinated information processing, unguided by any global criterion or central control. Information processing capabilities of such systems are not explicitly represented in their components, but rather in their interconnections. These capabilities are more powerful than the ones done by elementary components or their combinations. For these reasons, CA has been used to model different physical and biological

phenomena such as fluid flow, galaxy formation, avalanches, earthquakes, growth of stony corals, and other biological and physical pattern formations. Also Packard took the first evolving CA with GA, then Mitchell and al where the optimization addressed the density classification problem [169] [170], these discoveries are encouraging for the prospect of using GAs to automatically evolve computation for more complex tasks such as cryptography [171] and in more complex systems, the multiprocessor scheduling [172], [179] [180] and synchronization [174], [175], [181]. Moreover, evolving CAs with GAs also gives us a tractable framework in which to study the mechanisms that an evolutionary process might create complex coordinated behavior in natural decentralized distributed systems and that's what encourage its using in embedded system.

The evolutionary cellular automata improves many advantages in the field of computer science, but the GA search space in CA lattices or the transition rules are increased according to the CA dimension and that leads to increasing the computational time (3D or 4D Cellular automata is hard to be updated by the GA in short time). Also the classical chromosome is weak in representing the population diversity and as mentioned before, since the use of quantum bit representation leads to better population diversity while the use of quantum gate drive the population towards the best solution.

We used a new strategy to map the application onto NoC architecture and it is called mapping based priority per stage that gives a strong first mapping. The main idea of this technique as it is illustrated in Figure 4.7 is to map the tasks with higher priority onto different processors in which mean the tasks that are executed in a parallel way are allocated each one to a processor that is different from another. Seeing Figure 4.7 (a), the application has 7 tasks implemented onto 2x2 NoC, the number of stages is 5, the first contains task number 1, stage2 contains task 2 and 3 till stage 5 that contains task 7.

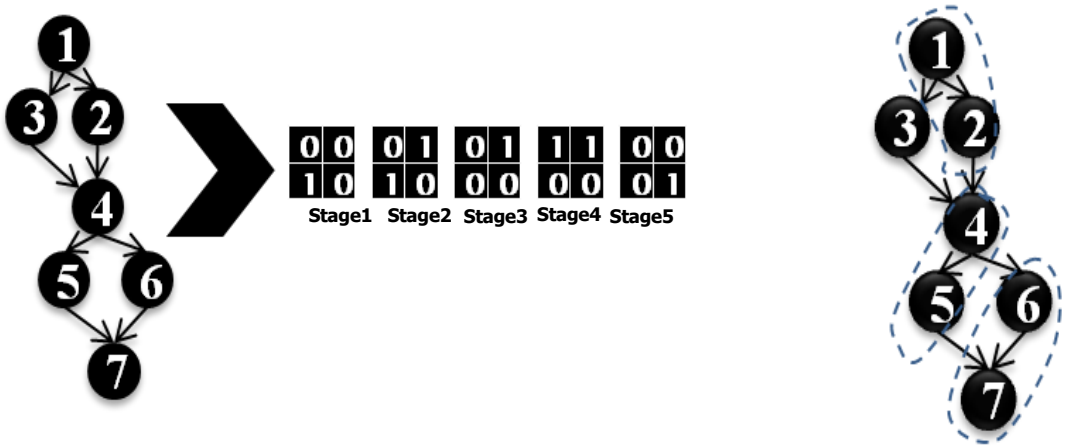
In this technique we have two cases; the first is when the number of tasks per stage is less than the number of processors by this we map the tasks randomly where each one occupies a different processor according to Equation.12. The second is the opposite of the first, is when the number of tasks per stage is bigger than the number of processors, we sort the tasks according to their workload, and we map the task into the processor that allocate the task with the lower workload as it is illustrated in Equation.13.

$$\forall\, stage_i \in STAGE \quad / c_j \in C ; \exists p_k \in P \,/\, p_k \subset stage_i$$

$$if \quad p_k = free \quad , c_j \rightarrow p_k \tag{12}$$

$$if \quad p_k \neq free \quad , \min = sort\ (w(j \in stage_i)),\ c_i \rightarrow p_{\min} \tag{13}$$

By this way, we minimized or in other cases eliminated the buffer wait time and the task wait time in case of two tasks are allocated to the same processor.



(a) (b)

Figure 4.10. The different mapping used in network on chip 2x2 Mesh. (a) An example of the per-stage mapping (b) an example of the channel mapping.

Another metric is used between each two stages, we call it the "Channel mapping" that maps two linked tasks to the same processor and it will look like the entire arc with its sink and source actors are allocated to that processor. A random mapping is used to map the other cores that didn't assign to any processor (like core3 in the figure 4.7 (b)) using (10) or (11).

6.4 Quantum evolutionary cellular automata lattice format

A CA consists of a lattice L of cells, in our case the CA consist of *STAGE* number of lattices that are updated in a synchronous and a parallel manner. The transition rules are applied onto those different stages. The first lattices are obtained from the first mapping of application onto NOC 2D mesh. The following figure represents an example of one lattice where each cell is itself contained 3 cells staring from left to right DVS cell, task cell and task migration cell. If the middle cell is 1 (Figure 4.7 (b)), it means that the processor is occupied by the task in the corresponding cell in the matrix in Figure 4.7 (a) otherwise the processor is free. If the left cell is 1 it means that the DVFS is applied onto that task, whereas the right cell indicates that the task migration is applied to that task as well. In the adjustment phase, we use the matrix in Figure 4.7 (b) because it allows us to verify if there un-allocated task.

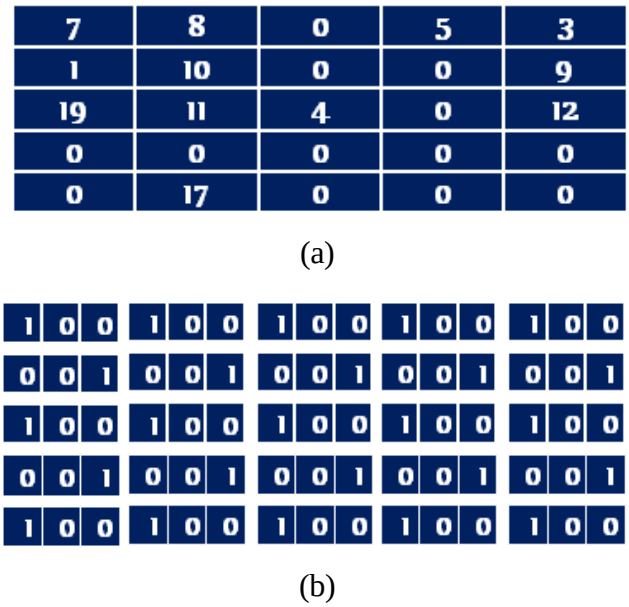


Figure 4.11. Cellular automata lattice format.

6.5 QECA concepts

Given: An application G(C, E) with C: set of tasks and E: set of channels between them. An NxN 2D mesh based NOC G(P, L) with P : set of cores and L is the set of physical links.

Find: an optimum CA lattice with its optimum set of rules that guarantee optimal energy consumption due to the mapping of an application tasks onto NoC processors.

The set of all possible configurations of the neighborhood $card\ (V)^h$ represents the correspondence table of the transition rules see Figure 3.23, whereas the set of all transition rules $card\ (V)^{card\ 'V)^h}$ represents the research space of the quantum evolutionary cellular automata. The quantum genetic algorithm is originally with the selection phase, crossover phase and mutation phase; but according to [177], those operations have a major effect on changing the probabilities of linear superposing of qubit states, but the use of quantum bit representation, it leads to better population diversity. Also the probabilities or the rate of applying mutation and crossover may reduce the performances of the quantum genetic algorithm. For that, it is better to operate without using selection, crossover and mutation and due to that the interference is the only genetic operator to apply.

In following, a detailed explication of how using QGA with CA to optimize NoC performances and they are compared with classical GA and ECA. The following figure illustrates the phases of how QECA works, whereas in Figure 4.10, the phases of both classical GA and ECA since they are following the same order but not the same implementation or chromosome representation.

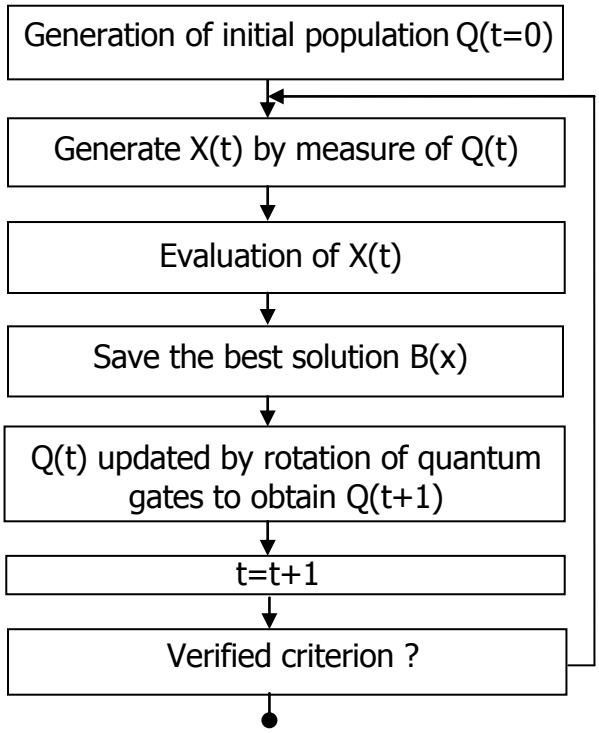


Figure 4.12. Quantum genetic algorithm structure.

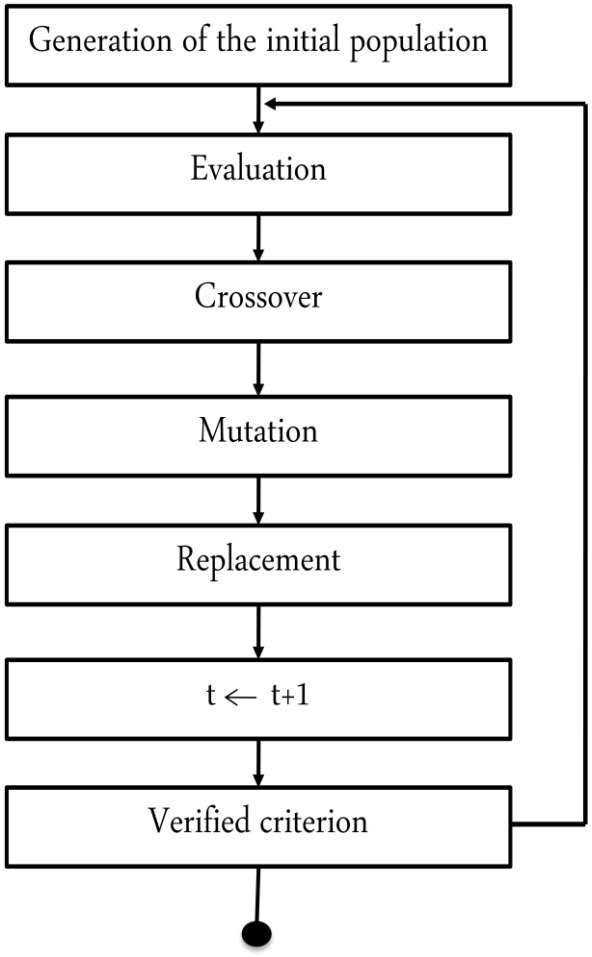


Figure 4.13. Classical genetic algorithm structure.

1) Problem Representation: The chromosome is used to represent the application mapping problem. In the classical genetic algorithm, the chromosome consists of a series of genes where each gene is assigned a binary value (1 or 0) with $gene\ (x, y)$ =1 indicates that $task\ (x)$ is allocated to $core\ (y)$ otherwise $gene\ (x, y)$ =0. Figure 4.12 shows an example of binary chromosome for a (2x2) mesh topology. A gene associated with a router is assigned a null value if no IP core is assigned to the router, each core can be allocated by many tasks whereas each task cannot be assigned to two cores. Whereas in evolutionary cellular automata, each transition rule represents a chromosome encoded in 32 genes (in our example), the locus represents the $gene\ (i)$ with its 4 neighbors, the corresponding new state of that gene is the state corresponding to the decimal value of the locus situated in the "correspondence table", the structure of the new chromosome is in Figure 4.13.

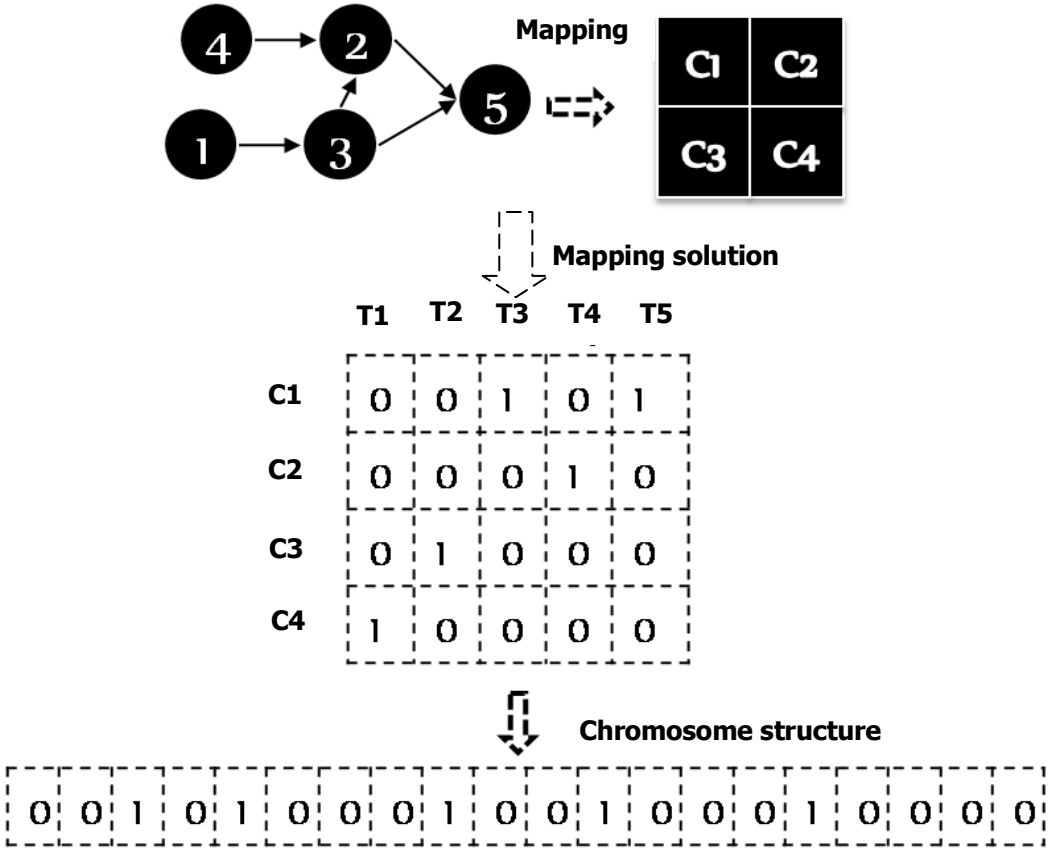


Figure 4.14. Example of an application graph mapped onto 2x2 mesh based NoC, where the mapping solutions represent the chromosome structure.

Figure 4.15. Structure of the classical chromosome [32 bits].

In QECA, we used the same chromosome representation as ECA but in place of the bit, a Q-bit is used. Q-bit is defined as the smallest unit of information. A Q-bit can be represented as:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{14}$$

Where α and β are real numbers whose represent the probabilities that the Q-bit found in the "0" and "1" states, respectively with $|\alpha|^2 + |\beta|^2 = 1$.

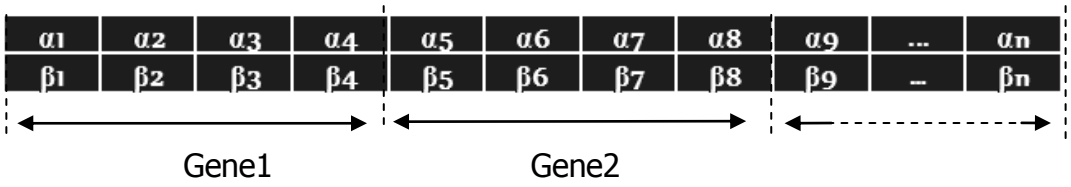


Figure 4.16. Structure of the quantum chromosome.

The state of a Q-bit may be "0", "1", or a linear superposition of the two:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{15}$$

Where $|0\rangle$ and $|1\rangle$ mean the states "0" and "1", respectively. However, a linear superposition of states with m Q-bits can be represented by Q-bit individual (Equation.16) for that a system with m Q-bits can represent 2^m states at the same time.

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \beta_3 & \dots & \beta_m \end{bmatrix} \tag{16}$$

with $|\alpha_i|^2 + |\beta_i|^2 = 1$, for $i = 1, 2, \dots, m$.

2) Population: The population is the main element of a genetic algorithm. Research has shown that the initial population may have an effect on the best fitness function value and these effects may last for several generations [152]. For a large NoC, the possible mapping space is extremely large which could slow down the convergence. Hence, a good initial population may

result in faster convergence. On the other hand, the population size influence is also another parameter to decide the coverage of mapping space. A population that is too large takes time to evolve whereas a population that is too small will lead to a local minimum. Population size is proportional to the application size. As it has is mentioned before in section VIII (b), the first mapping is based the channel mapping and the per-stage mapping and that enables the EGA and QECA to reduce the search space for low energy mapping with faster convergence instead of random exploration of the huge search space. In ECA, we generate an initial population of chromosomes, from 2^{32} transition rules, $P = 100$ individuals are chosen randomly and well distributed in the research space while considering the first mapping technique to allow diversity of solution. In QECA, We generated the same initial population from 2^{32} transition rules, and the same $P = 100$ individuals that are chosen randomly and well distributed in the research space, and adding to this Q(t=0) is generated. First, the generation counter is set to $t = 0$, then an initialization of the group of Q-bit individuals $Q(t)$ where $Q(t) = [q_1^t, q_2^t, ..., q_n^t]$, where n is the total number of Q-bit individuals and q_j^t is the j th Q-bit individual at a generation t which is defined as:

$$q_j^t = \begin{bmatrix} \alpha_{j1}^t & \alpha_{j2}^t & \alpha_{j3}^t & \cdots & \alpha_{jm}^t \\ \beta_{j1}^t & \beta_{j2}^t & \beta_{j3}^t & \cdots & \beta_{jm}^t \end{bmatrix} \quad (17)$$

Where $j = 1,2,..., n$ and m is the string length.

3) Measuring Chromosomes: this phase is only considered for QECA where X(t) is generated by the measuring of Q(t). In this step, each Q-bit is observed and measured from $Q(t)$ in order to extract a classic chromosome $X(t)$ that the evaluation of each quantum chromosome. For that, $X(t)$ is a group of binary solutions where $X(t) = [X_1^t, X_2^t, ..., X_n^t]$, and $X_{j1}^t = [x_{j1}^t, x_{j2}^t, ..., x_{jn}^t]$, where x_{ji}^t is a binary value that is determined as:

x_{ji}^t := get x_{ji}^t in [0,1] ;

if ($x_{ji}^t < |\beta_{ji}^t|^2$) x_{ji}^t =1; else x_{ji}^t =0;

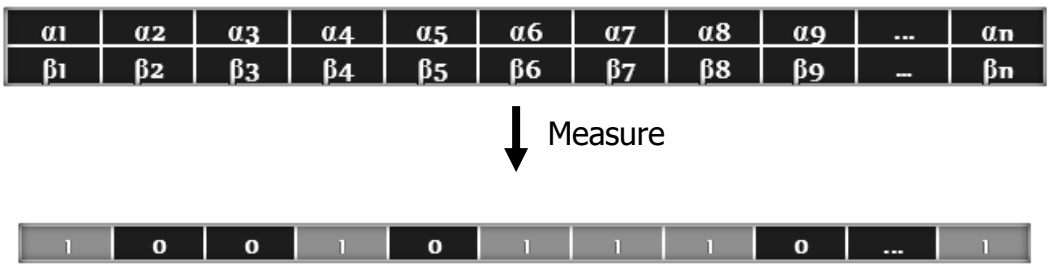


Figure 4.17. Measured chromosome.

4) Fitness function: Fitness function represents the desired optimization goal. The evaluation of each chromosome is through its evolution throughout M iterations. A note is assigned to individuals (chromosomes in ECA and in QECA) for their mapping solution. A best chromosome is with the best mapping solution that minimizes the energy consumption under timing constraints. The fitness function is given as:

$$F_{NoC} = \frac{\lambda \times E_{NoC} + (1-\lambda) \times L_{NoC}}{F\_\min} \times 100\ \% \tag{18}$$

In the equation, $F_{\min}$ is the minimum energy found and λ is a proportionality coefficient that is used to adjust the proportion of communication power consumption and delay in the cost function, and the value range is $0 < \lambda < 1$. So the optimization objective of the NoC map method in the thesis is to minimize (F_{NoC}).

The following steps 5 and 6 are not applied in the evolutionary cellular automata algorithm, but only on quantum evolutionary cellular automata.

5) Store the best solution among $X(t)$ into $B(t)$: $B(t)$ is a matrix that stores the best solution of QECA chromosome in the whole population.

6) Update $Q(t)$ using Q-gates: Quantum individuals are updated by using Q-gates. The population Q(t) is updated with a quantum gates rotation of qubits constituting individuals. The rotation gate $U(\Delta\theta_i)$ and the update operation are expressed as:

$$U(\Delta\theta_i) = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \tag{19}$$

$$\begin{bmatrix} \alpha_{ji}^{t} \\ \beta_{ji}^{t} \end{bmatrix} = U(\Delta\theta_i) \begin{bmatrix} \alpha_{ji}^{t-1} \\ \beta_{ji}^{t-1} \end{bmatrix} \tag{20}$$

$\Delta\theta_i$ is a rotation angle which determines the magnitude and direction of rotation. Figure 4.15 illustrates the polar plot of the rotation gate for Q-bit individuals.

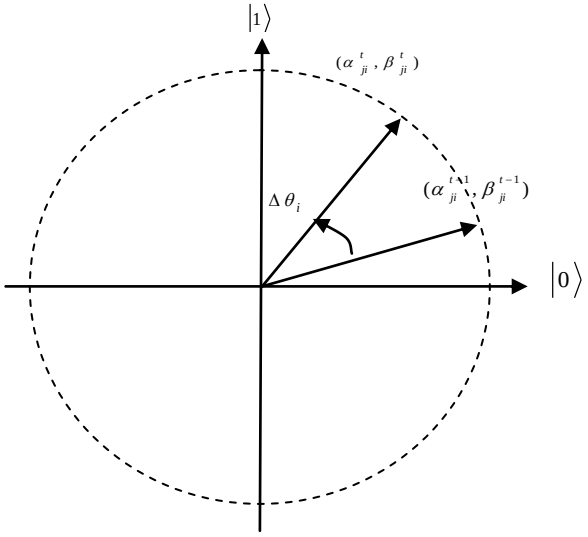


Figure 4.18. Polar plot of the rotation gate for Q-bit individuals.

x_{ji}^{t}	b_i^{t}	$f(X_j^{t}) > f(B^{t})$	$\Delta\theta_i$	$S(\alpha_{ji}^{t} \times \beta_{ji}^{t})$		
				> 0	< 0	$= 0$
0	0	0	$\Delta\theta_2$	-	+	±
0	0	1	$\Delta\theta_2$	-	+	±
0	1	0	$\Delta\theta 1$	-	+	±
0	1	1	$\Delta\theta_2$	-	+	±
1	0	0	$\Delta\theta 1$	+	-	±
1	0	1	$\Delta\theta_2$	+	-	±
1	1	0	$\Delta\theta_2$	+	-	±
1	1	1	$\Delta\theta_2$	+	-	±

Table 4.2. Look-up table for quantum gates rotation.

At generation t, the rotation angle $\Delta\theta_i$ is updated according to the criteria summarized in Table.2, where x_{ji}^{t} and b_i^{t} are the binary control variables in solution X_j^{t} and the best solution B^{t} of $B(t)$, respectively. $f(X_j^{t})$ and $f(B^{t})$ represent the objective function values of X_j^{t} and B^{t}. For example, when x_{ji}^{t} and b_i^{t} are 0 and 1, and $f(X_j^{t})$ is larger than $f(B^{t})$, the rotation angle $\Delta\theta_i$ is updated according to $S(\alpha_{ji}^{t} \times \beta_{ji}^{t})$ in Table.2 where $S(\alpha_{ji}^{t} \times \beta_{ji}^{t})$ is the sign of $\alpha_{ji}^{t} \times \beta_{ji}^{t}$.

In the last step of QECA, the best solution among $X(t)$ and $B(t-1)$ is stored to $B(t)$, and terminated if the stopping conditions are met; else generate a new population.

In an ECA, crossover, mutation and replacement are considered and they are applied just after step 4 where the fitness function for each chromosome is calculated, for that the following steps are considered only for ECA and they are as follows:

- Crossover: After selecting the new individuals, each two of them are coupled randomly or based on their fitness function to produce new offspring, and the fitter chromosomes are searched to form a new population. An example of a couple is when one of the individuals represents a mapping with high energy consumption, but with the best timing, whereas the second chromosome is with the minimum energy consumption, but with long latency, the crossover between them is done in order to search the middle fitness function between them. Crossover points are randomly set according to the nature randomization behavior of GA. Crossover is done at one point, or tow points. Two children chromosomes are generated from two selected parents. After the crossover between parents, if the same task is assigned to two cores, one of them is selected randomly and the other is set to "0" value. In the other case when there is an unmapped task, it is mapped to a core that is allocated by the task with the highest communication flow that is linked with the unmapped task. By that all individuals are labeled as valid chromosomes.

- Mutation: The crossover is applied between two parents, whereas the mutation uses one parent and creates one offspring. Each chromosome has a mutation probability which indicates if the chromosome is mutated or not. The probability is augmented or lowered in each generation. If the mutation probability of the chromosomes is always low and in the same, the fitness function is also small, we can observe the weak changing in the population. In this case we can change parameters such as the voltage of the processors, adding priority to some tasks that are prone to deadline missing. We can also change the scheduling mechanisms that are defined in section II (e).

Replacement: this phase is the last for ECA where new parents selected for replacing the older, we used a steady replacement where new chromosomes replace the old one regardless of their adaptation values, and the elastic replacement where the keeping individuals are with the best performances on the current generation.

7. Experiments and case of studies

In this section, we discuss the case of studies and experiments used to evaluate the influence of cellular automata on genetic algorithm and also on the quantum genetic algorithm.

As the communication patterns of these applications are different, the results produced have different characteristics, due to that we used the benchmark multi-application to test the efficiency of the algorithm and compare it with a classical genetic algorithm. We start by defining the communication task graphs and then we present the results obtained from the mapping of those applications onto the network on chip.

This thesis studies the effectiveness of evolving cellular automata with quantum genetic algorithm where considering the transition rules as the search space where each one represents a chromosome or an individual. The network topology is 5x5 mesh based NoC. We used the per-stage mapping with the channel mapping as the first mapping of the multimedia application onto the NoC architecture. The first mapping represents the lattices that are updated according to the CA transition rules. The lattices of each CA are transformed into mapping and each transition rule evaluated and assigned a fitness value based on the result obtained from each lattice updating. In CGA or ECA, the best transition rules with best fitness function values are selected for crossover and mutation, whereas in ECA, the population is updated using the quantum gates. The population size is fixed for 150. Crossover and mutation probabilities for CGA and ECA are not fixed, and each chromosome has a probability of mutation and crossover based on the algorithm defined in section VIII (D). The termination of GA is set to 800 generations, 300 generations for ECA and 150 for QECA, where $\Delta\theta_2 = 0.001\,\pi$ and $\Delta\theta_2 = 0.08\,\pi$ (The same as in [177]).

The number of ports per router is fixed to 4 ports and maximum link length is fixed to 4 four virtual channels, the scheduling analysis and the routing algorithm must take the number of ports of the router and the link length as a constraint. For 0.18um technology node the parameters are defined in Session 2 (Table.2). The router is operating at voltage range from 1.0v to 0.6v. The frequency of the system is calculated based on the operating voltage according to Equation.10.

7.1 Quantum evolutionary cellular automata results

Figure 4.20 and Figure 4.21 exhibit our results for the mapping of MMS and auto-indust respectively onto network on chip 5x5 mesh using the QECA, the metric chosen is the optimization of the fitness value in each generation, whereas in Figure 4.16, Figure 4.17, Figure 4.18 and Figure 4.19, and under the same assumption used in QECA, we tested the evolution of the fitness function in each generation using GA in auto-indust, GA in MMS, ECA in auto-indust and ECA in MMS respectively. A comparison of QECA, ECA and GA finesses function is done in order to demonstrate the efficiency of QECA as it is illustrated in the following

figures, where the results of the fitness function obtained from both ECA, and GA are disordered, sometimes it is low and other time it is high, that is because there isn't any criteria that would control nor the stabilization of the results, nor the increment of the fitness function values.

As it is seen in Figure 4.18 and Figure 4.19, the fitness function is over 300 generations and an optimization is achieved about 0.98 and 0.97 in both MMS and auto-indust respectively, so the time that the optimization takes is less than the time in GA that is about 460 generation where the fitness function optimized about 0.91on both MMS and auto-indust. Whereas in the figure Figure 4.20 and Figure 4.21 and under the same assumption, we can see that the fitness function is achieved about 0.99 on both MMS and auto-indust and only after 110 generations. So the optimization using QECA is efficient and accurate compared to GA and ECA.

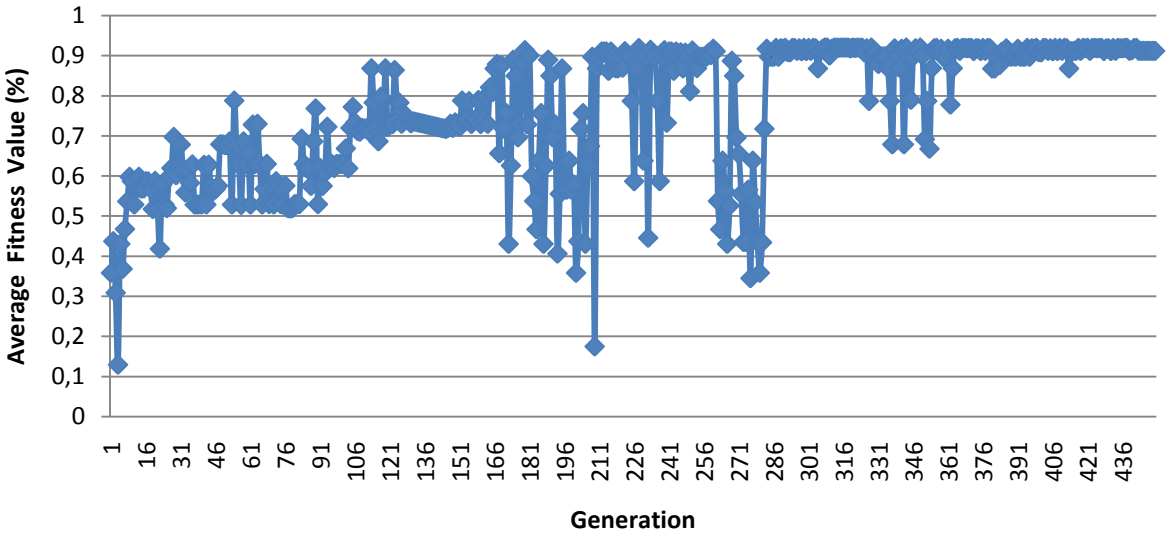


Figure 4.19. The average fitness value of each generation using GA in MMS.

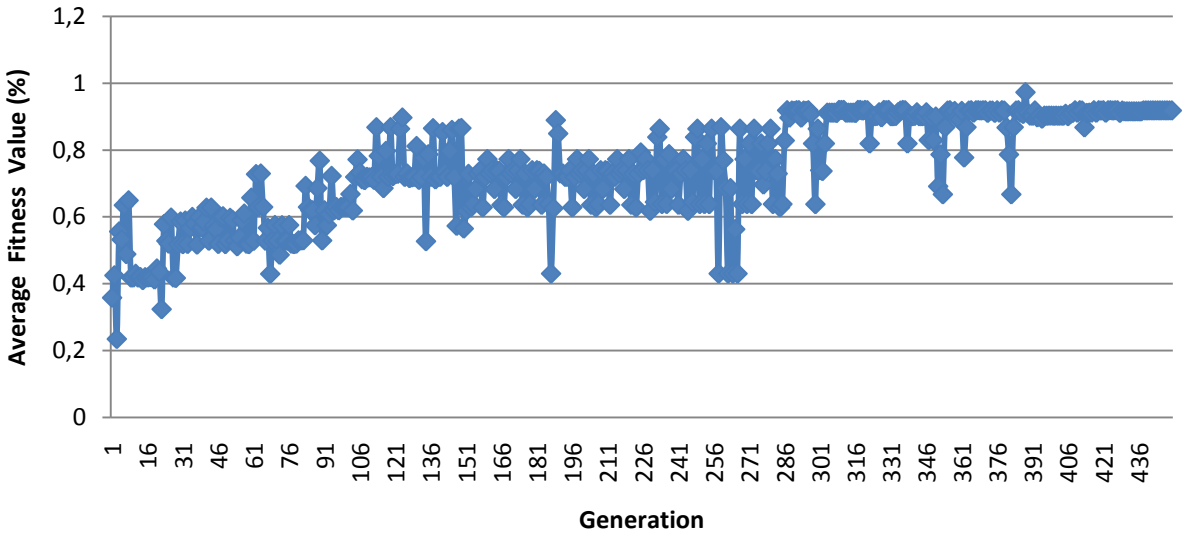


Figure 4.20. The average fitness value of each generation using GA in auto-indust.

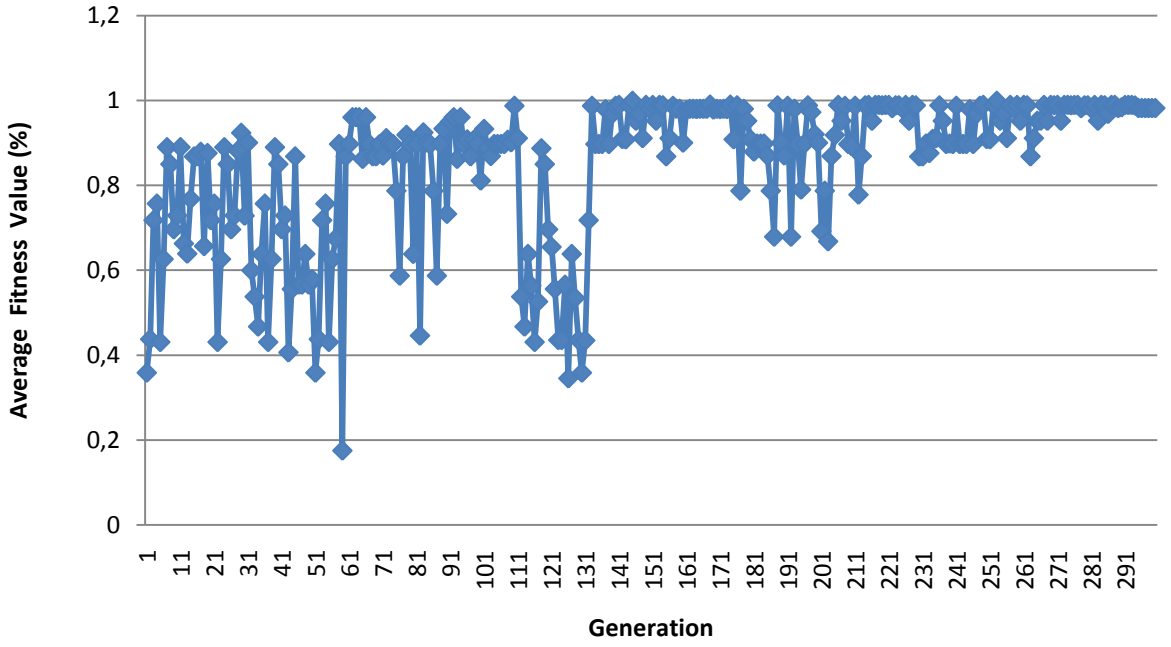


Figure 4.21. The average fitness value of each generation using ECA in MMS.

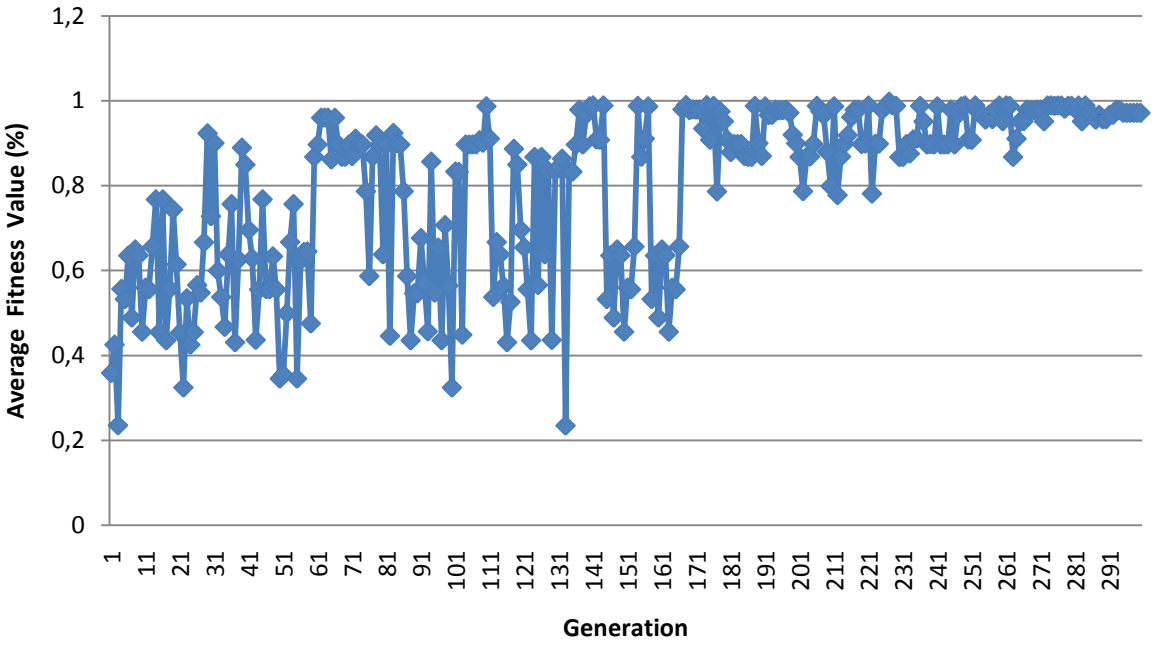


Figure 4.22. The average fitness value of each generation using ECA in auto-indust.

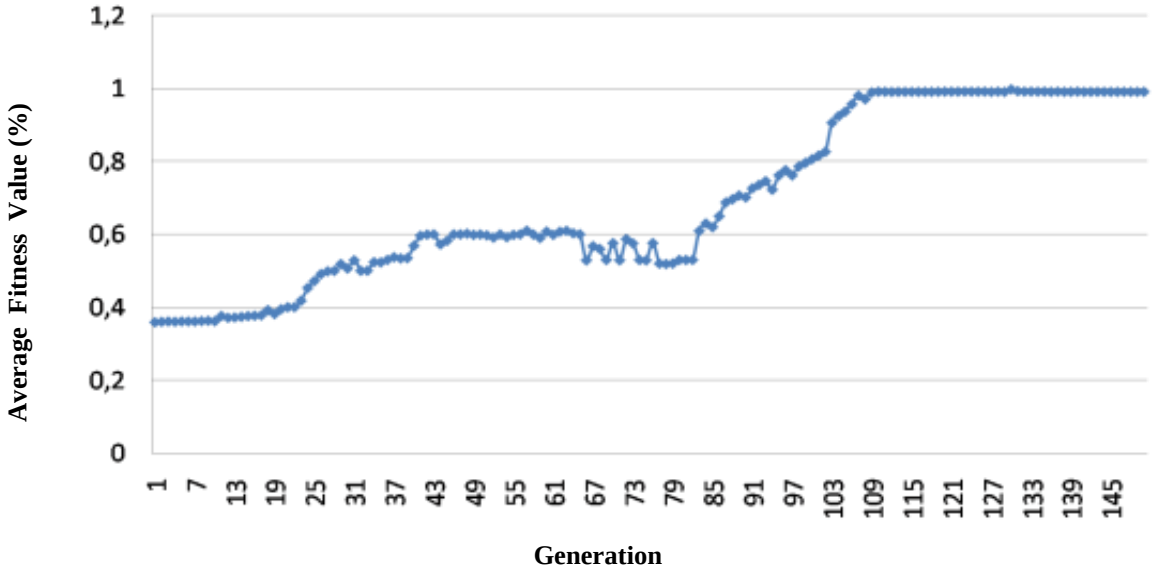


Figure 4.23. The average fitness value of each generation using QECA in MMS.

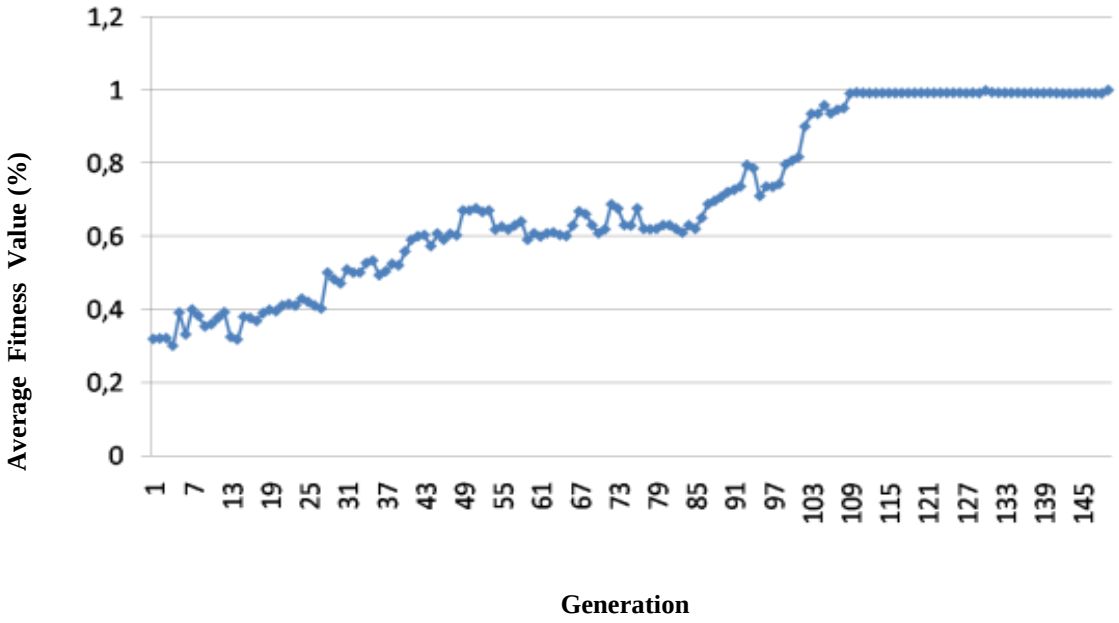


Figure 4.24. The average fitness value of each generation using QECA in auto-indust.

Another advantage whose appears in Figure 4.20 and Figure 4.21 is the stabilization and an increment of fitness values of the results obtained from QECA compared to GA and ECA, this increment is the result of using the quantum gates that have the ability to drive the population toward the best solution. By observing penalty methods, we can then confirm the effectiveness of QEGA's performance over ECA and GA and moreover, quantum algorithms have generally the ability to minimize the complexity of equivalent algorithms that run on classical computers.

In the following, another metric to prove the efficiency of QECA over ECA is the deviation of the fitness function. The deviation is considered as the gap between two fitness values that could influence the performance if it becomes too large over time in another way, it indicates that successive generations produce two vastly different fitness values. This gap helps us to take in consideration the role of the ECA operations (the mutation and the crossover) to guide the generation in the right direction, this guiding is done by controlling the probabilities of mutation and crossover. Whereas in QECA, the deviation of the fitness function is originally controlled by the use of quantum gate that drives the population towards the best solution. As it seen in the following figures (Figure 4.23 and Figure 4.24), huge gaps are remarked between fitness functions on both applications the MMS and the auto-indust, whereas in Figure 4.25 and Figure 4.26, we remarked a small gap between the first twenty (20) fitness functions and in the last a smooth incrementing of fitness values, this is the results of the use of quantum gates.

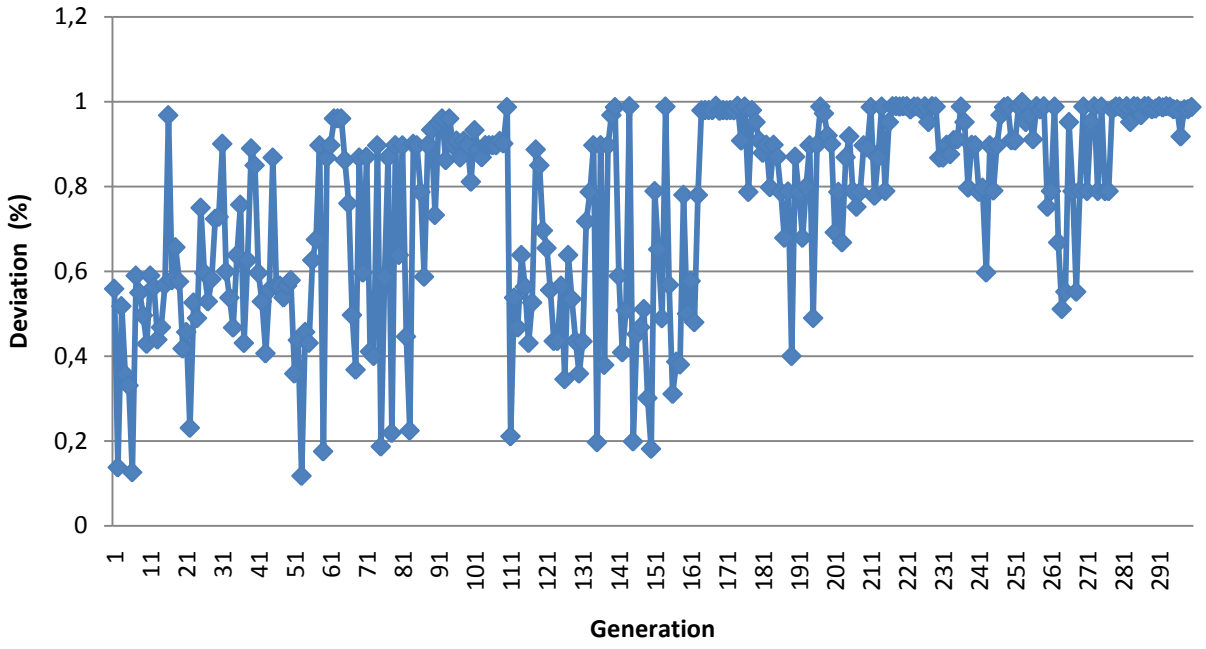


Figure 4.25. The deviation of each generation using MMS.

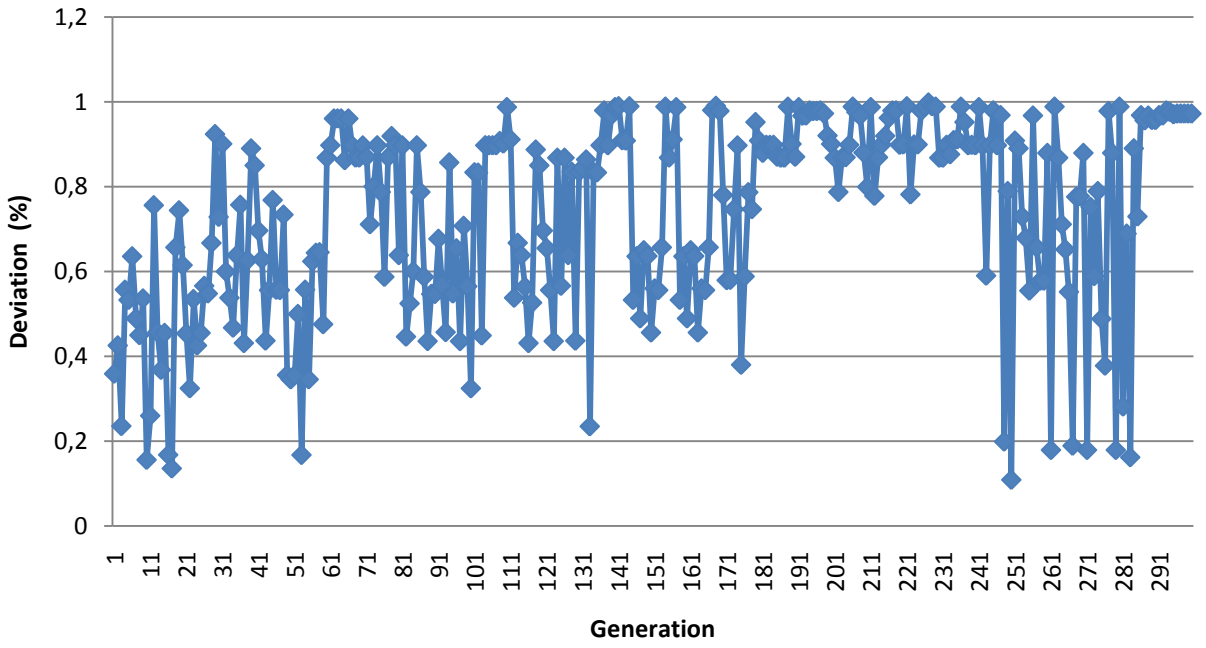


Figure 4.26. The deviation of each generation using ECA in auto-indust.

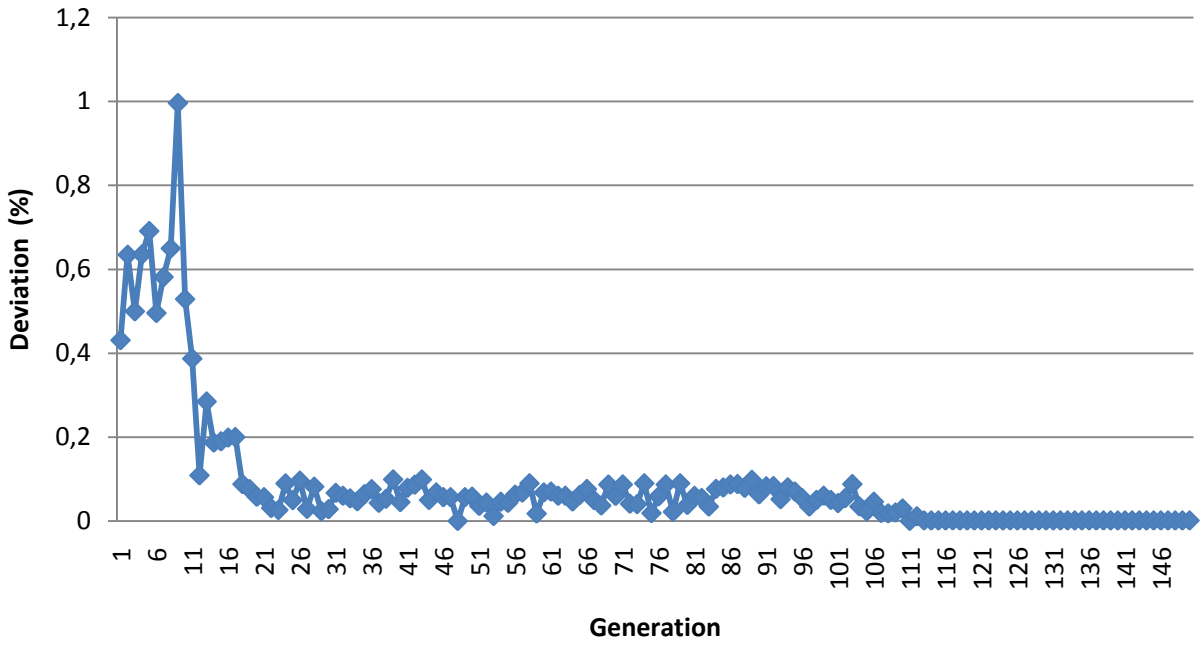


Figure 4.27. The deviation of each generation using QECA in MMS.

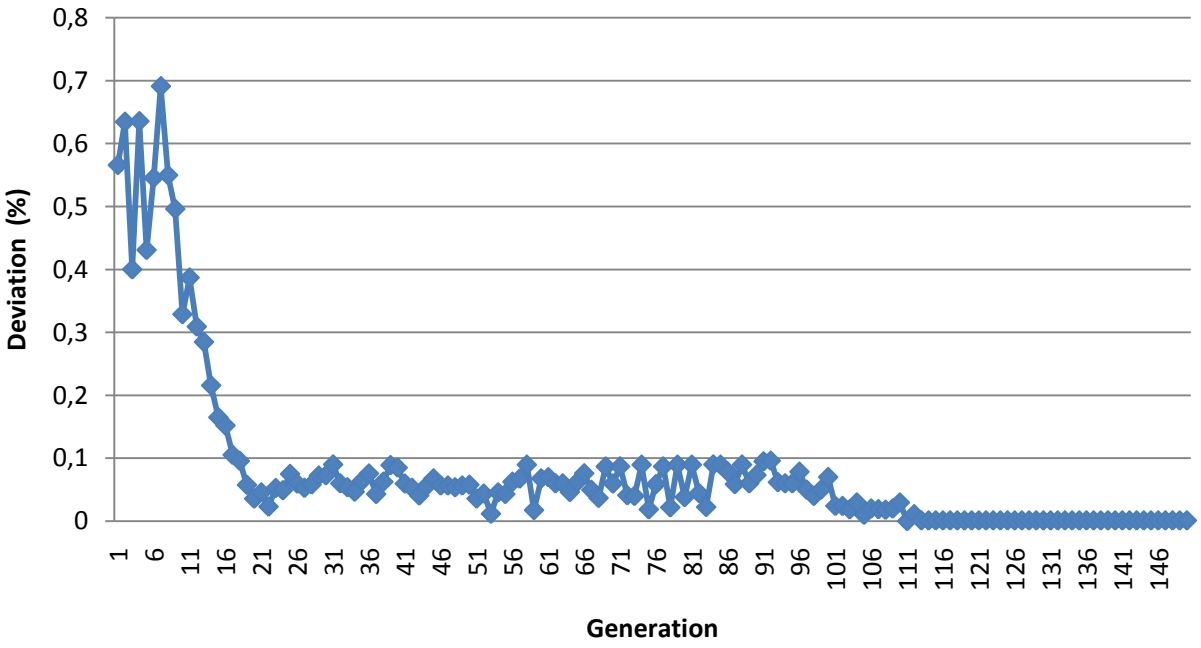


Figure 4.28. The deviation of each generation using QECA in auto-indust.

The results of overall energy consumption and latencies on GA, ECA and QECA are shown in the following table Table.3 where the energy consumption of MMS and auto-indust in ECA is 41.39% and 45.12% respectively compared with to the initial consumed energy, and also the latency is optimized about 40.01% and 38.56% on MMS and auto-indust respectively. Whereas in QECA, the energy consumption is much optimized seeing ECA, on both MMS and auto-indust about 66.90% and 69.98M respectively, also the latency is either optimized about

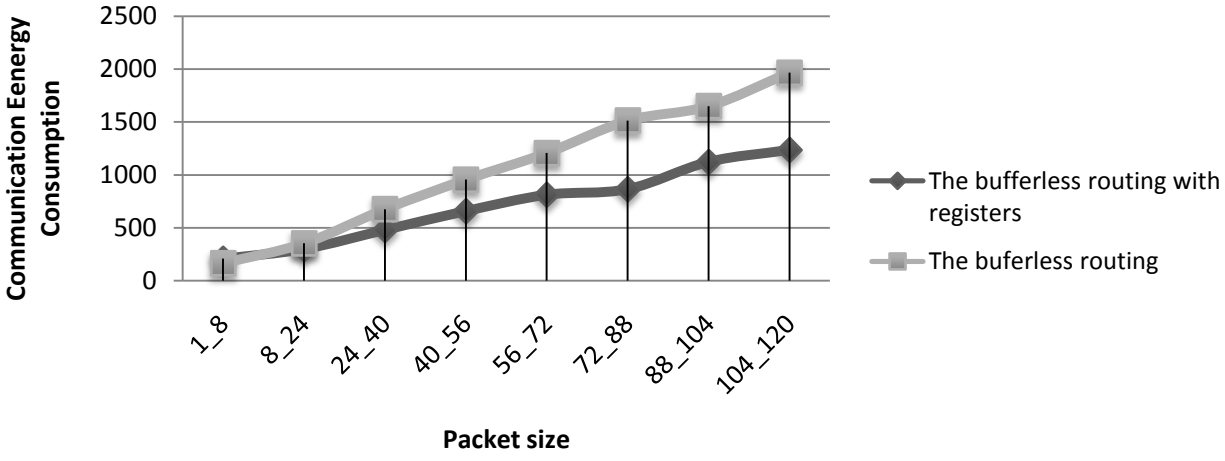
44.89% and 43.98% on both MMS and auto-indust respectively. Our proposed optimization algorithm that uses QECA is much efficiency comparing to ECA and GA algorithm and from other research, such as the work presented in [153], where the author proposed a multi-objective adaptive immune algorithm that is an evolutionary approach to optimize the latency and power consumption about 27.3% and 42.1% and the latency 29.3% and 36.1%, and in [154] the author proposed an application mapping technique that incorporates domain knowledge into the genetic algorithm (GA) to minimize the energy consumption of NoC communication and he achieved about 29% of NoC communication energy saving.

	Energy consumption		**Latency**	
	MMS2	**Auto-indust**	**MMS2**	**Auto-indust**
GA	30.23%	31.99%	20.66%	18.87%
ECA	41.39%	45.12%	40.01%	38.56%
QECA	66.90%	69.98%	44.89%	43.98%

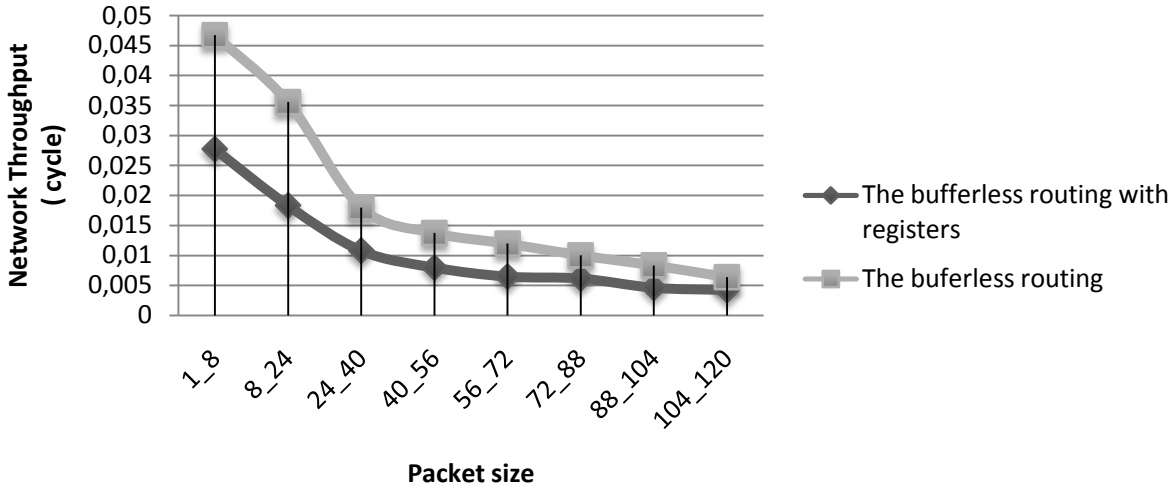
Table 4.3. Comparison between QECA, ECA and GA in executing mms2 and auto-indust.

7.2 Bufferless wormhole routing results

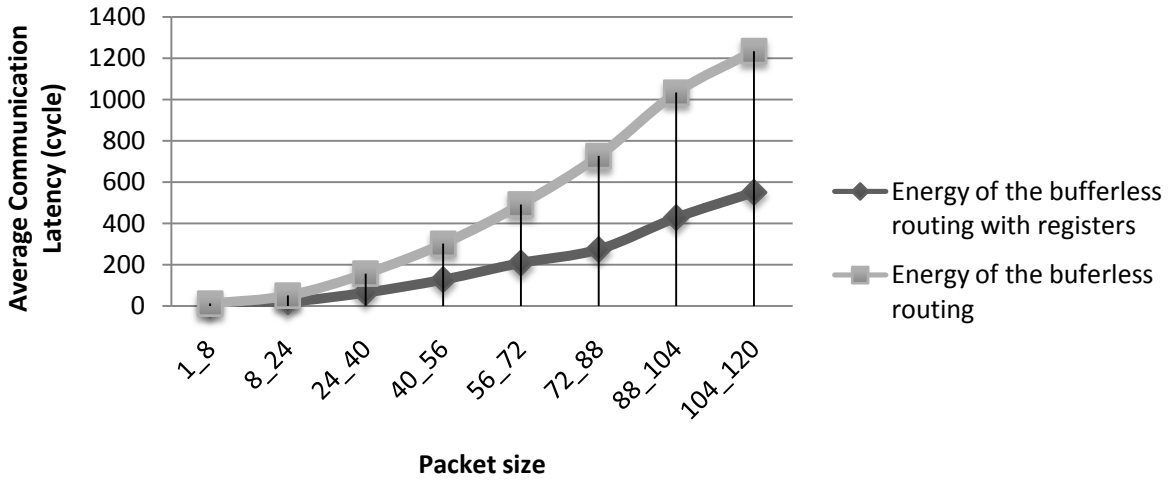
In order to perform a complete evaluation of the proposed routing algorithm, an implementation of the old randomized routing algorithm is employed using 2D mesh NoC with a size of 5x5, the router architecture utilized in the thesis consists of 4 virtual channels.



(a) : Communication energy consumption versus packet size.



(b) : Network throughput versus packet size.



(c) : Average communication latency versus packet size.

Figure 4.29. Comparison of Bufferless routing with registers versus bufferless routing.

In Figure 4.26, we simulated under the same assumption, a comparison of Bufferless routing with registers versus bufferless routing, and the results obtained, are from the mapping of MMS application onto NOC 5x5. In Figure 4.26 (a) we computed the communication energy consumption of both routing schemes, and as it is seen, under small packet size, the consumed energy by using the bufferless routing is less than the use of the other scheme. With the increasing number of packets, the consumed energy during communication from using bufferless routing with registers is increased as well, but this time it is less than the energy consumed using bufferless routing. The reason of this huge different increasing of bufferless routing is because the energy consumed from the links is getting much bigger with incrementing the number of packets and also because each flit represents a head flit and flits

from same packet always take different paths. The link utilization is incremented as the number of head flit increments and that requires huge power to transmit those head flits.

In Figure 4.26 (b) and Figure 4.26 (c), the throughput and average communication latency of both schemes are computed under varying the number of packets and as it is seen the bufferless routing shows weak results compared to bufferless routing with registers, the propagation and the transmission of flits take a different path, and take longer latency.

Bufferless routing shows great results in power saving as bout 33.07% on average and with the use of small FIFO registers to store only the head flits we obtained greater throughput and small latencies about 57.31% on average and 41.88% on average respectively.

7.3 Randomized routing algorithm results

In the Figures Figure 4.27, Figure 4.28, Figure 4.29, Figure 4.30, we tested the effect of packet size on average packet delay, average communication delay, average network latency and the average throughput using different mapping methods, a random mapping which the task assignments onto the processors are done with a random way, the second is the per stage mapping and the third we add the concepts of channel mapping to the per stage mapping. We simulated these tests under the same assumptions, tr=0.09ms, tw = 0.06, γ = 0.5, VC =4; buffer size = flit size , we generated a random task graph and a random mapping .

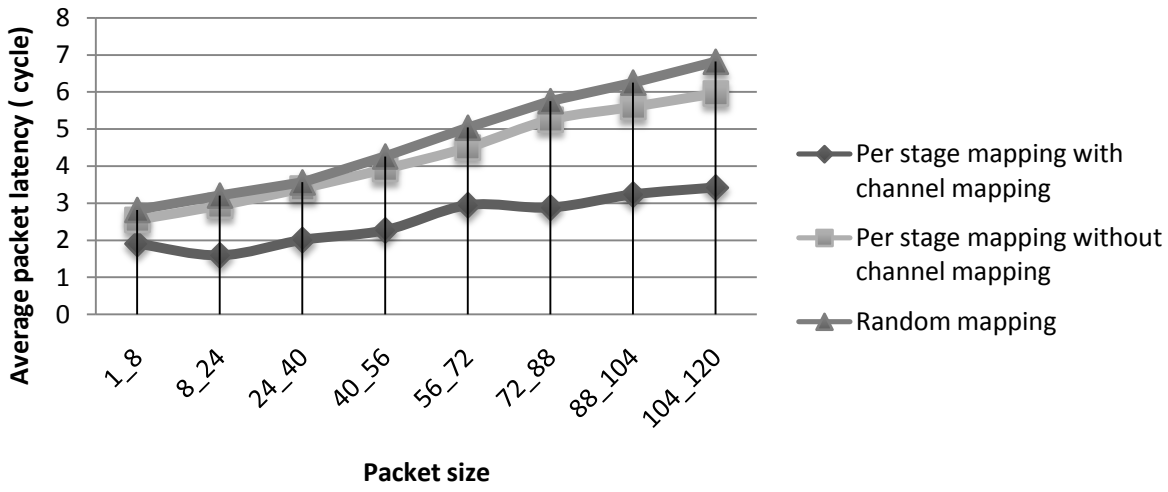


Figure 4.30. Effect of packet size on packet delay using Per- stage mapping without channel mapping and Per stage mapping with channel mapping and a Random mapping.

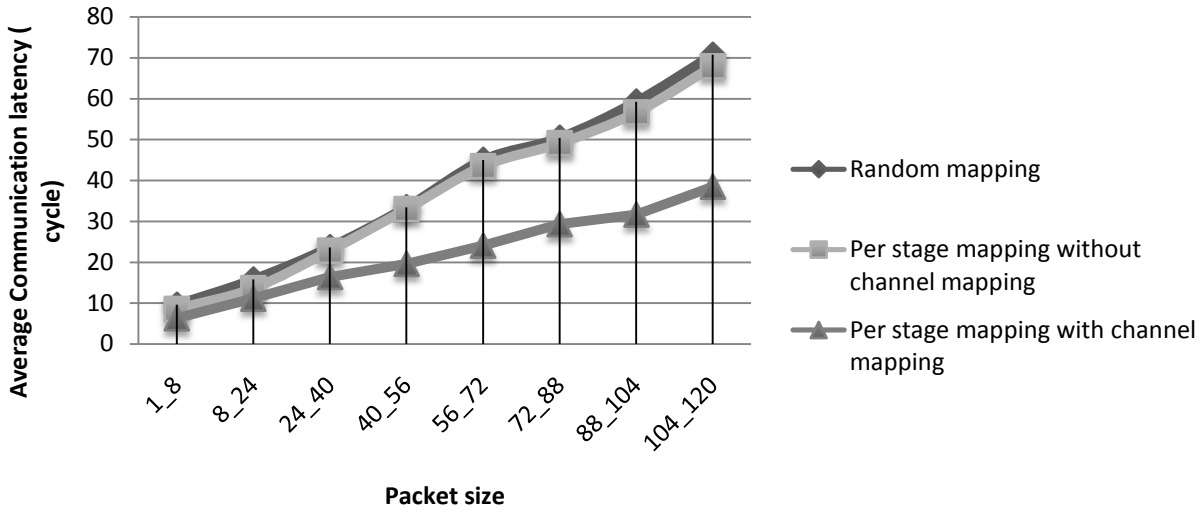


Figure 4.31. Effect of packet size on average communication using Per-stage mapping without channel mapping and Per-stage mapping with channel mapping and a Random mapping.

As we can see that by incrementing the number of flits the communication delay, computation delay and throughput are incremented as well. Also the use of the mapping per stage has improved better results compared to random mapping about 10%, in average packet delay and about 5% about in average network delay and about 5% in average communication and about 33% in the throughput, and also the mapping per stage with channel mapping improved much better results than the random mapping about 46% of the average packet latency and about 40% about in average network delay and about 42% in average communication and about 56% in the throughput.

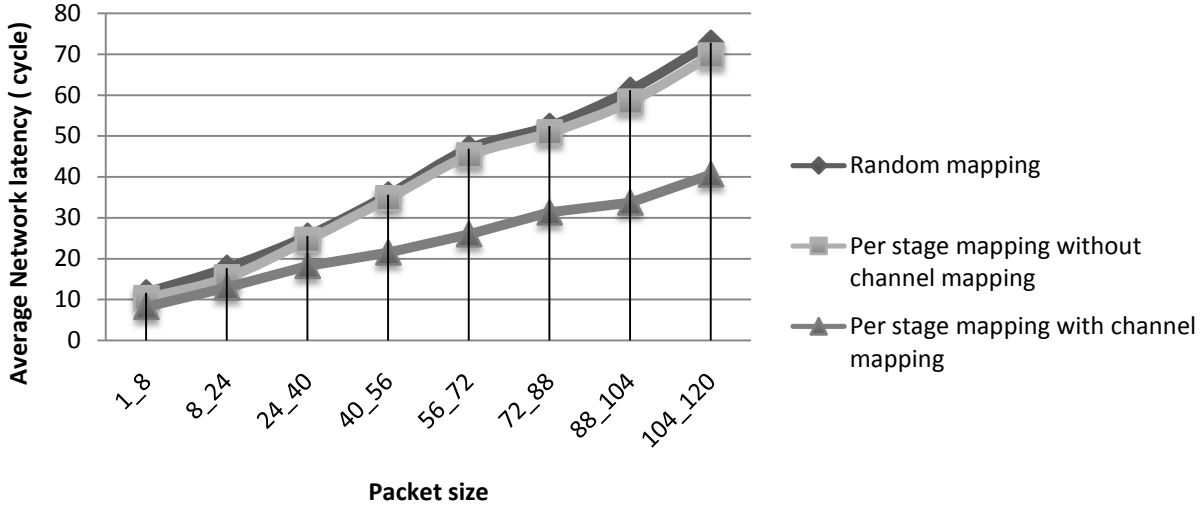


Figure 4.32. Effect of packet size on average network latency using Per stage mapping without channel mapping and Per stage mapping with channel mapping and a Random mapping.

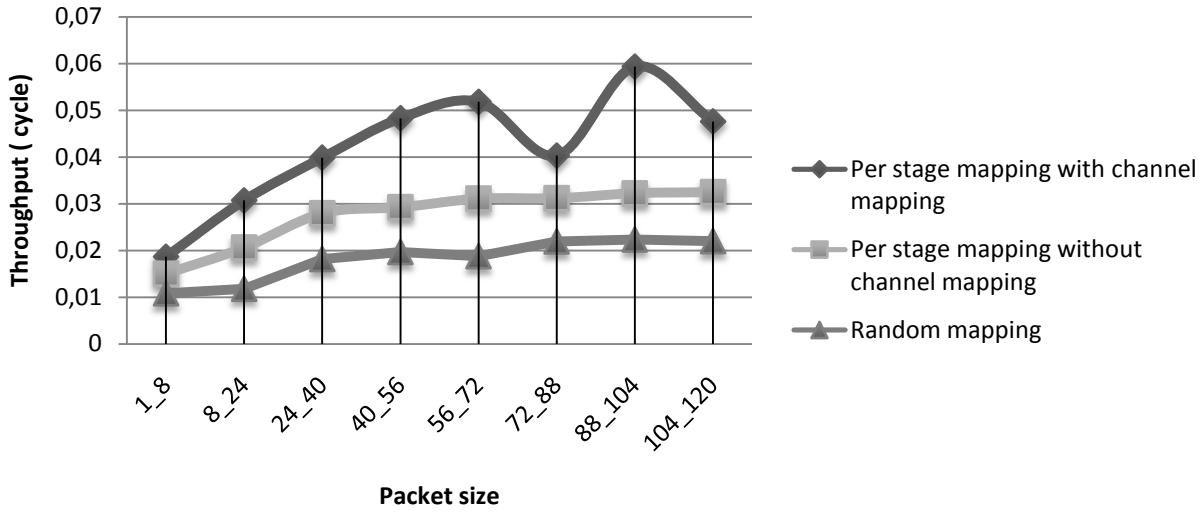


Figure 4.33. Effect of packet size on average throughput using Per stage mapping without channel mapping and Per stage mapping with channel mapping and a Random mapping.

The average communication delay is used as a metric to evaluate the routing algorithm. We first minimized the probability of deadlock and then eliminate as possible the wait time of packets and as it seen in Figure 4.28, the communication delay is much closer to the network delay, and that because of the transmission time has a major impact on the delay comparing to the packet delay as it is seen in Figure 4.27.

By varying the packet injection rate of different mapping, the average packet delays don't differ too much. In Figure 4.31 we estimated the average packet delay of mapping MMS onto 5x5 Mesh, the delays much closer because the mapping strategy that we used to minimize the buffer's wait time. The same thing in Figure 4.32, we estimated the throughput by varying the packet injection rate. As it is seen in both Figures, the per stage with channel mapping improves better results than the other by 37%.

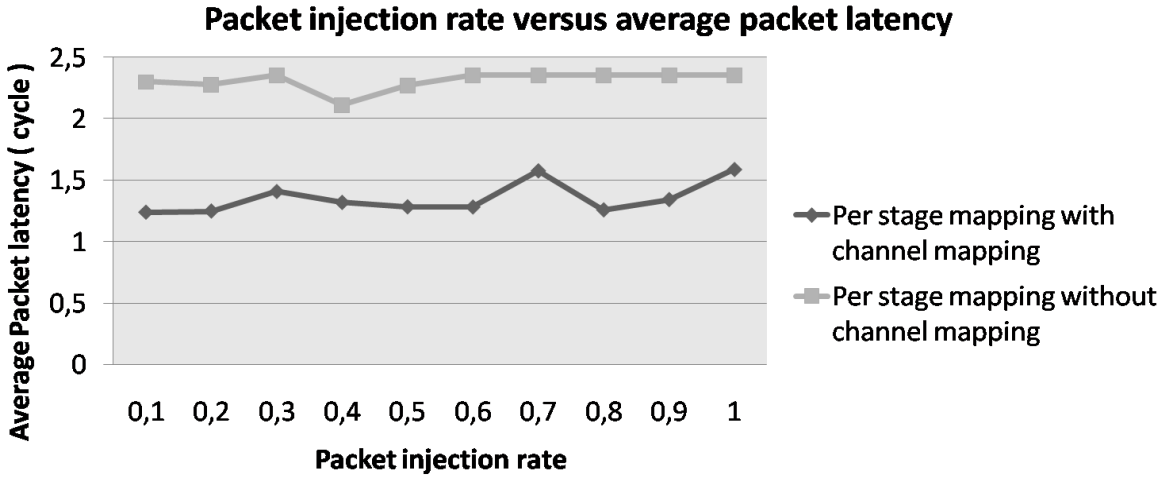


Figure 4.34. Packet injection rate versus average packet delay.

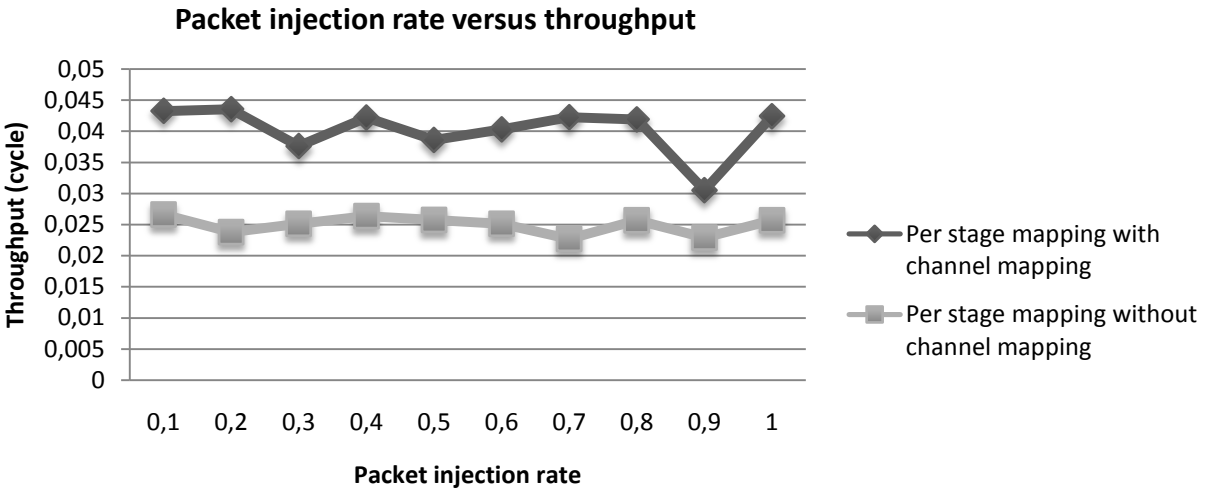


Figure 4.35. Packet injection rate versus throughput.

In Figure 4.33, Figure 4.34 and Figure 4.35, we tested the average flit in the system, the buffer occupancy by the flit and the average mean service rate by varying the router service time tw and tr under different size of virtual channels. Figure 4.33 shows that the average flit in the system is decreased by incrementing the router time decision and incrementing the number of VCs.

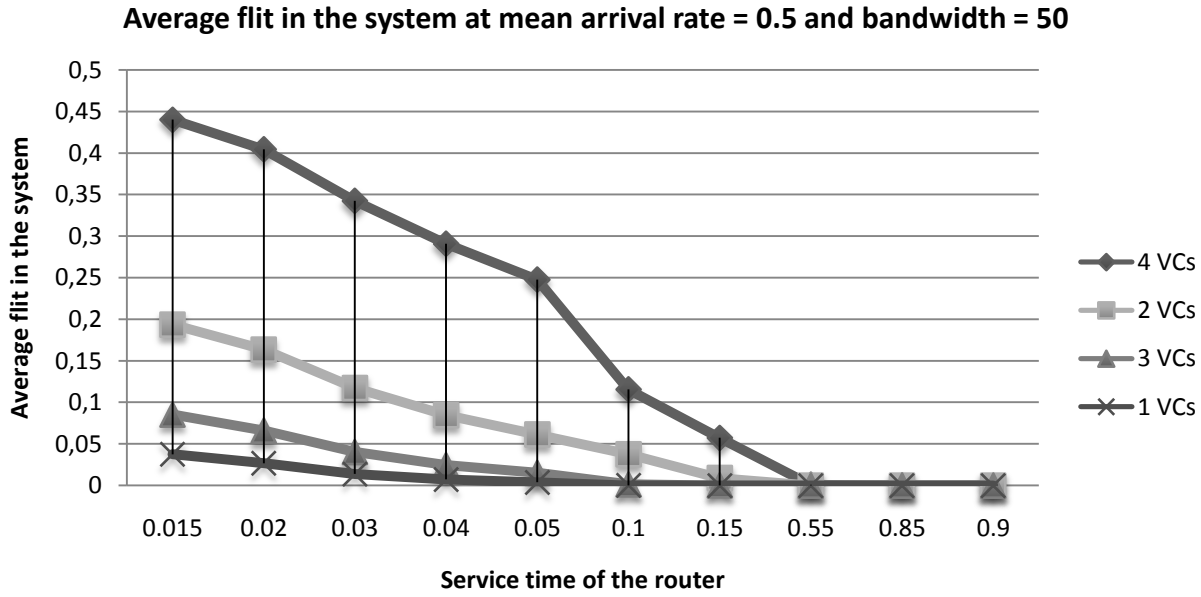


Figure 4.36. Average flit in the system at mean arrival rate = 0.5 and bandwidth = 50.

Figure 4.34 shows the buffer occupancy under varying the router time decision, it is seen that, the buffer occupancy by the flit is decreased by incrementing the router time decision. Beside the previous scheme, the results of using VCs are much closer to the results of using one

physical channel, this because the mapping strategy that we used to minimize the buffer' and link' wait time.

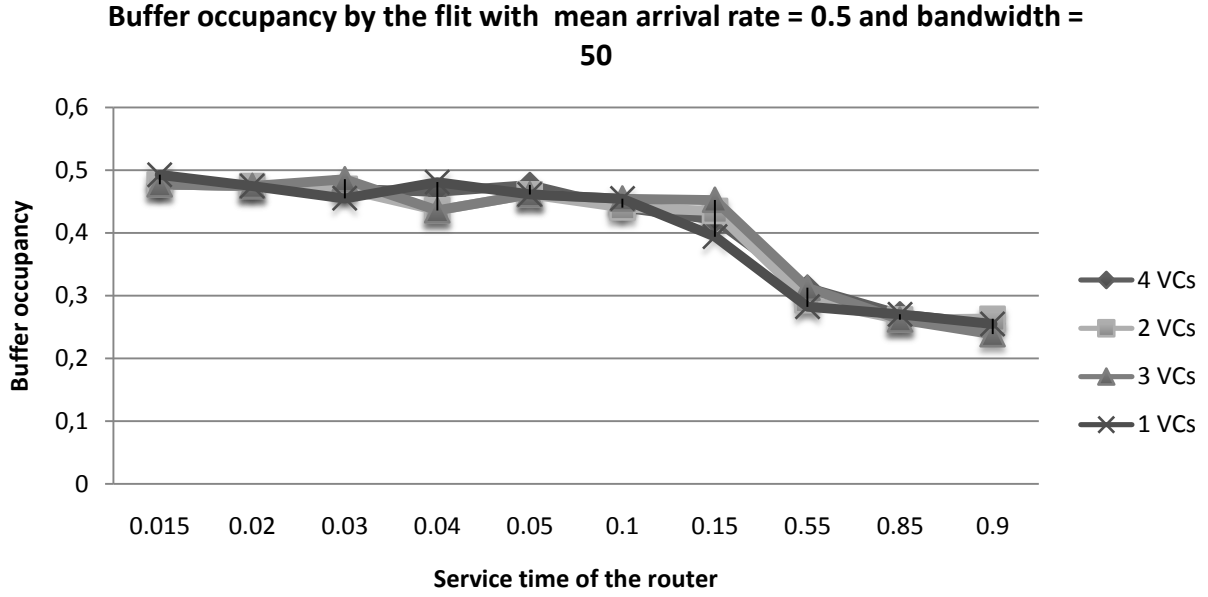


Figure 4.37. Buffer occupancy by the flit with mean arrival rate = 0.5 and bandwidth = 50.

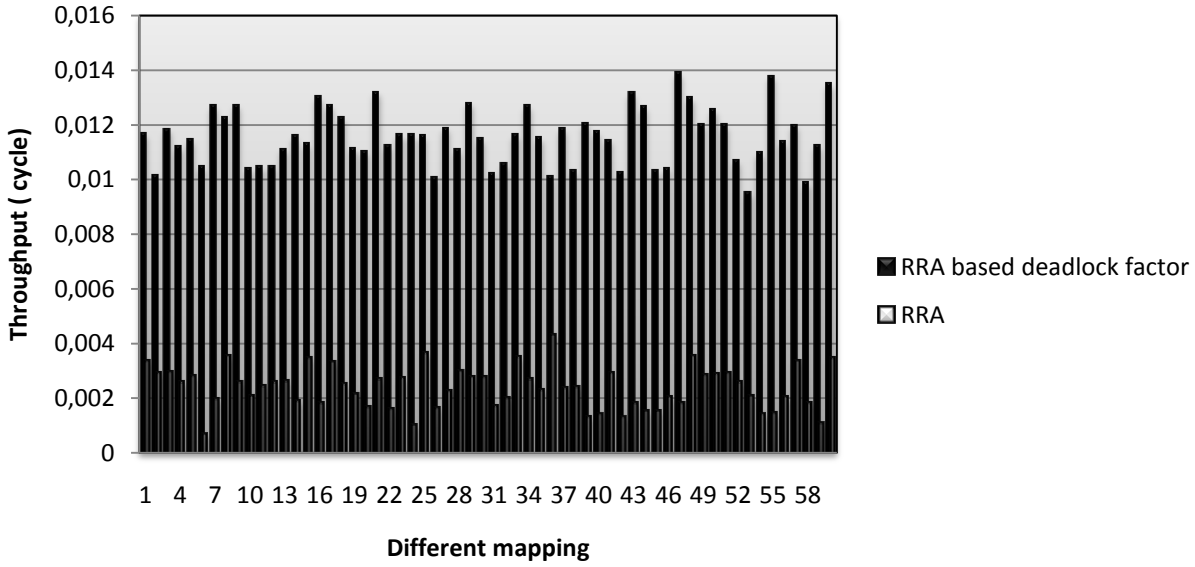


Figure 4.38. Average mean service rate at mean arrival rate = 0.5 and bandwidth = 50.

In order to test the efficiency of our algorithm, we implemented under the same assumption the randomized routing algorithm; Figure 4.35 represents the throughput of a system when mapping MMS onto 2-D Mesh. As it is seen the throughput of the system when using a randomized routing algorithm with deadlock factor is much better about 95.18% when using RRA.

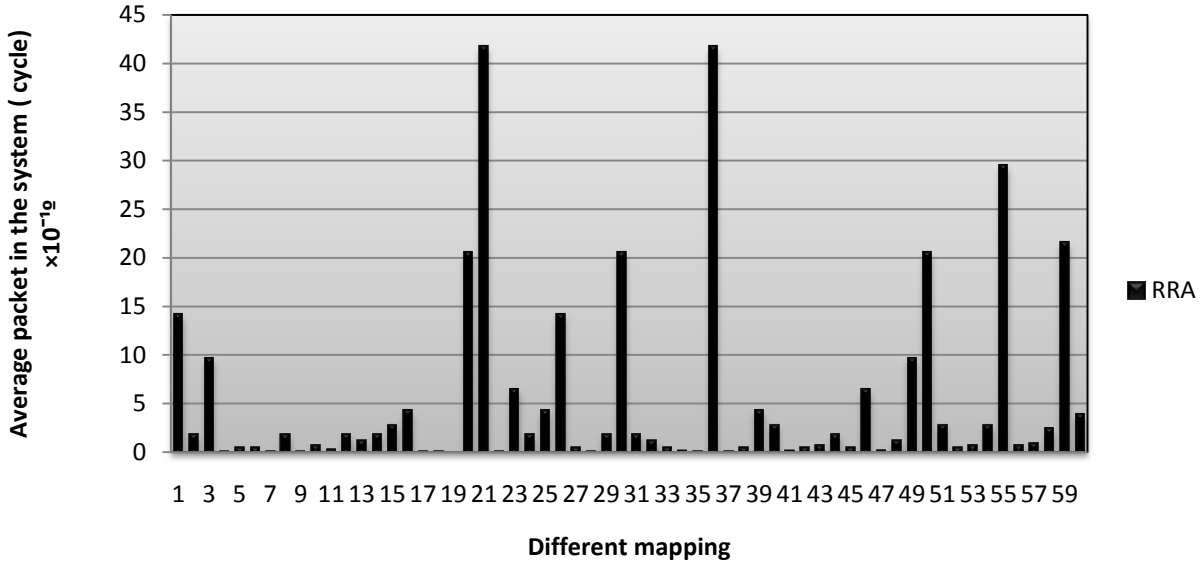


Figure 4.39. Average packet in the system at mean arrival rate = 0.5 and bandwidth = 50.

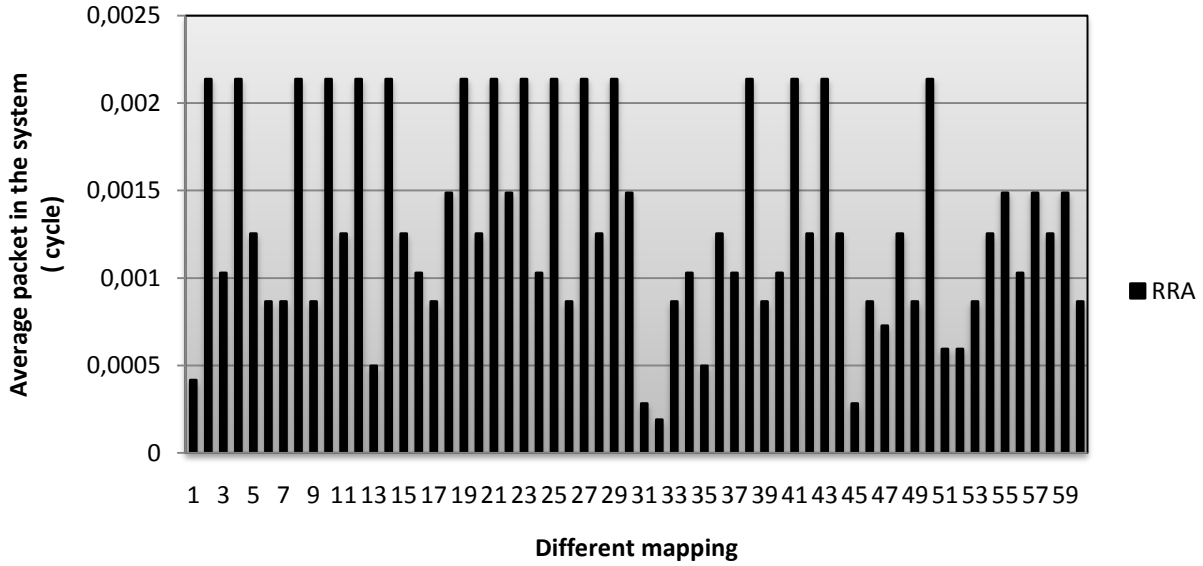


Figure 4.40. Average packet in the system at mean arrival rate = 0.5 and bandwidth = 50.

Figure 4.36 and Figure 4.37 represent the number of packets in the system of RRA and RRA with deadlock factor respectively, and as it is seen the avoiding of deadlock helps on serving more packets per cycle. The average packet of the system when using RRA with deadlock factor is much better about 99.99% than using RRA.

Figure 4.38 represents a comparison of the three mapping technique, a random mapping, per stage mapping and per stage with channel mapping, the comparison is done using MMS benchmark on NoC 5x5 mesh. We tested the average network latency of a system of different mapping. As it is seen the per stage with a channel mapping is more efficient than the others.

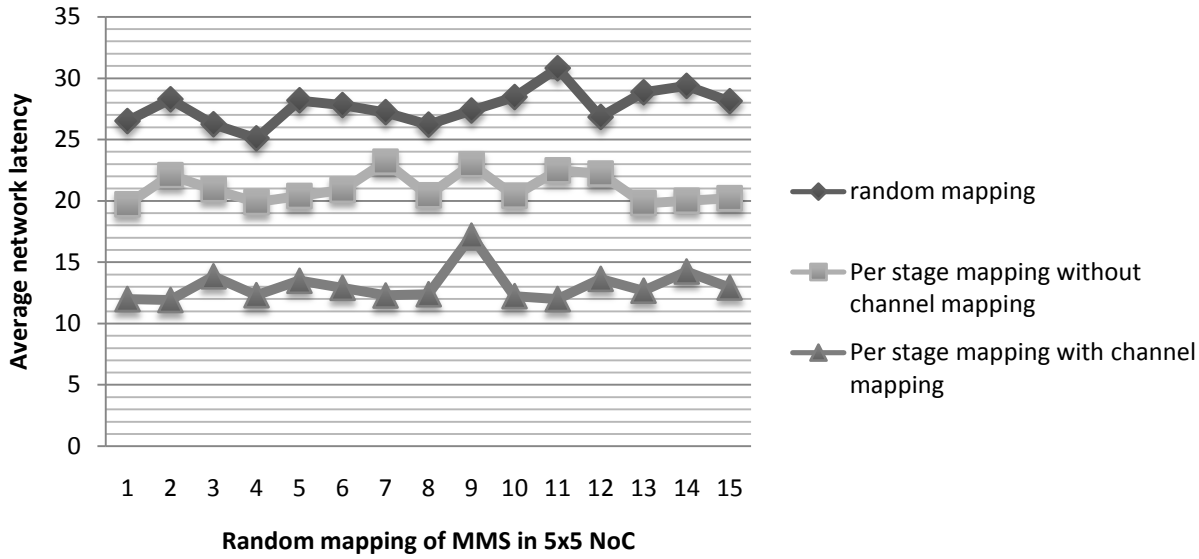


Figure 4.41. Comparison of three mapping strategy based on the average network latency in the system at mean arrival rate = 0.5.

7.4 Dynamic voltage and frequency scaling results

Table.4 represents the result obtained from using the proposed dynamic voltage and frequency scaling on both MMS and auto-indust with comparing it to none using of DVFS. In order to evaluate the algorithm three metrics are used that are the energy consumption, the latency of the system and an end to end deadline, the latter metric is related to the soft nature of real-time multimedia application and it is used to guarantee that the deadline is not missed, but with a small threshold. The results obtained are presented in the following table:

	Energy consumption		**Latency**		**End to end deadline**	
	MMS2	**Auto-indust**	**MMS2**	**Auto-indust**	**MMS2**	**Auto-indust**
DVFS	0.33	0.27	48.61	40.12	100%	100%
Non-DVFS	2.95	2.51	14.17	12.90	51.67%	49.19%

Table 4.4. Results of executing mms2 and auto-indust with/without dynamic voltage and frequency scaling.

As it is observed, the DVFS has a long latency comparing to Non-DVFS about 62.62% and 60.36% on MMS and auto-indust respectively, but the energy consumption is high in Non-DVFS about 88.81% and 89.24 % on MMS and auto-indust respectively. The end to end deadline in Non-DVFS is close to the half, so the tasks are executed with high voltage and end in short time, resulting in leakage power on idle processors, whereas in using DVFS, an end to end deadline is achieved without deadline missing and with optimizing the energy consumption.

8. Conclusion

Energy consumption and latencies are the most critical issues for advanced NoC designs using scaled technologies. In this chapter, we have presented a novel approach to optimize the network on chip performance using quantum computing in evolutionary cellular automata in order to guarantee better population diversity while driving the population towards the best solution. We compared classical GA, ECA, and shown that QECA algorithms are a very promising technique stimulating the process of discovering effective rules that leads to better fitness functions with better population diversity that is driven towards the best solution. We had also presented the efficiency of the proposed approach in the deadlock and livelock avoiding using a randomized routing algorithm and bufferless wormhole based registers. Also we had proposed dynamic voltage and frequency scaling algorithm that achieve an end to end deadline and optimize about 88.81% of the energy consumption.

GENERAL CONCLUSION

1. Conclusion

This thesis presents a novel method for solving the mapping, scheduling and the routing problems in regular network on chip for multimedia applications based on quantum evolutionary cellular automata. The method applies QECA to handling the multimedia application tasks mapping and scheduling problem. The QECA method is based on the concept and principles of quantum computing, such as quantum bits, quantum gates and superposition of states. Thus, the mechanism of the QECA method can inherently treat the balance between exploration and exploitation where each Q-bit individual can represent and explore all possible states and drive it to exploit a single state. The use of quantum bit representation leads to better population diversity compared with the classical bit representations while the use of quantum gate drive the population towards the best solution. The optimization achieved using QECA is efficient and accurate compared to GA and ECA where the fitness function is about 99% over 110 generations.

Also, we had proposed an idea considering the design issues in using bufferless routing in NoC while motivated by the use of small FIFO registers; we used the bufferless routing in order to achieve minimal latency and greater throughput by using a new randomized routing

algorithm with new format of wormhole flits that use small FIFO registers. The idea of the randomized routing algorithm works in two steps, the first is to determine the deadlock factors using a randomized routing algorithm while the second step is to choose the direction with the minimum deadlock factor value. We had also proposed a new mapping strategy in order to eliminate the buffer wait time. The routing algorithm achieves about 95.18% of the system throughput and both the mapping strategy with the routing algorithm achieved about 46% of the average packet delay.

In the context of power management, we proposed a dynamic voltage and frequency scaling algorithm that achieve 100% end to end deadline while searching the voltage changing point based on the task workload. Our algorithm saves the power by lowering the operating voltage and clock-speed of a processor in real-time to meet the executed tasks timing constraints.

2. Perspectives

In this thesis, we focused on the mapping of multimedia application task into regular 2D mesh network on chip while taking into account the possibility of task migration but only on one of the node neighbors, and also with applying the voltage and frequency scaling on each task (if conditions are satisfied) under 0,18 nm technology, where we had also solve the deadlock and livelock problems that are occurred in randomized routing algorithm with using new format of head flit in wormhole switching. In future work we attempt to:

- Using 3D mesh and also different technology (E.g. 0, 22 nm).
- Solve the mapping problem in multimedia application into irregular network on chip.
- Using another task migration strategy such as task migration through check-pointing or predictive task migration.
- Using another voltage and frequency scaling methods such as the partitioning DVFS.
- The optimization was targeting the energy consumption and latency of the system, another criteria is also important in irregular network on chip which is the chip area, we will apply the QECA but with three objectives (area, energy consumption and latency).

References

[1]. G. Fen, W. Ning, Q. Xiaolin and Z. Ying "Clustering-Based Topology Generation Approach for Application-Specific Network on Chip," San Francisco, USA. Vol II, October 2011.

[2]. P. P. Pandeet al. "Performance evaluation and design trade-offs for network-on chip interconnect architectures." IEEE Transactions on Computers, August 2005, Vol. 54, 8, pp. 1025-1040.

[3]. P. Koopman,"Embedded systems in the real world." Carnegie Mellon University, April 1998.

[4]. Rajesvari. R, Manoj. G, Angelin Ponrani. M "System-on-Chip (SoC) for Telecommand System Design," International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 3, March 2013.

[5]. David J Greaves. "System on Chip Design and Modelling ",University of Cambridge Computer Laboratory Lecture Notes. On 2011.

[6]. M. Coenen, S.Murali, A. Radulescu, K. Goossens, and G. De Micheli, "A Buffer-Sizing Algorithm for Networks on Chip Using TDMA and Credit-Based End-to-End Flow Control," in Proc. 4th international conference Hardware/software codesign and system synthesis CODES+ISSS' 06, Oct.22-25, 2006, pp.130-135.

[7]. W. J. Dally and B. Towles. "Route packets, not wires: On-chip interconnection," Proceedings of the 38th Design Automation Conference. June 2001.

[8]. S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani. "A network on chip architecture and design methodology," Proceedings of IEEE Computer Society Annual Symposium on VLSI, 2002. April 25- 26, 2002, pp. 105-112.

[9]. P. P. Pande, C. Grecu, A. Ivanov, R. Saleh. "Design of a switch for network on chip applications," Proceedings of the 2003 International Symposium on Circuits and Systems. ISCAS '03. May 2003, Vol. 5, pp. 217-220.

[10]. D. Siguenza- Tortosa and J. Nurmi, "Protea : A New Approach Network-on-Chip," Malaga (Spain), Tech. Rep. ,2002.

[11]. J. Duato, S. Yalamanchili, L. Ni. "Interconnection Networks: an Engineering Approach," On 2002.

[12]. L. Bononi and N.Concer, "Simulation and Analysis of Network on Chip Architecture: Ring, Spidergon and 2D Mesh," in Proc. Design, Automation and Test in Europe DATE'06, vol.2, Mar. 6-10, 2006, P.6pp.

[13]. P.Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in DATE '00: Proceedings of the conference on design, automation and test in Europe. New York, NY, USA: ACM, 2000, pp. 250-256.

[14]. C. J. Glass, L. M. Ni. "The Turn Model for Adaptive Routing," The 19th Annual International Symposium on Computer Architecture Proceedings. May 19-21, 1992, pp. 278-287.

[15]. S.Murali and G. De Micheli, "SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs," in Proc. 41st Design Automation Conference, 2004, pp.914-919.

[16]. M. Saldana, L. Shannon, and P. Chow, "The Routability of Multiprocessor Network Topologies in FPGAs," in SLIP '60: Proceedings of the 2006 internationnal workshop on System-level interconnect prediction, 2006, pp. 49-56.

[17]. P. Guerrier, A. Greiner. "A generic architecture for on-chip packet-switched interconnections," Design, Automation and Test in Europe Conference and Exhibition 2000. March 2000, pp. 250-256.

[18]. D. Marconett. "A Survey of Architectural Design and Implementation Tradeoffs in Network on Chip Systems," Davis : University of California.

[19]. F. Karim, A. Nguyen, S. Dey. "An interconnect architecture for networking systems on chips," IEEE Micro. Sept-Oct 2002, Vol. 22, 5, pp. 36-45.

[20]. R. I. Greenberg, G. Lee. "An improved analytical model for wormhole routed networks with application to butterfly fat-trees," Proceedings of the 1997 International Conference on Parallel Processing. 1997, pp. 44-48.

[21]. D. H. Linder, J. C. Harden. . "An adaptive and fault tolerant wormhole routing strategy for k -ary n-cubes," IEEE Transactions on Computers. January 1991, Vol. 40, 1, pp. 2-12.

[22]. L. M. Ni and P. K. McKinley. “A survey of wormhole routing techniques in direct networks,” February 1993,Vol. 26, 2, pp. 62-76.
[23]. W. J. Dally and H. Aoki. “Adaptive routing Using Virtual Channels,” MIT Laboratory for Computer Science. 1990. Technical report.
[24]. L. G. Valiant, G. J. Brebner. “Universal schemes for parallel communication,” Proceedings of the 13th annual ACM symposium on Theory of computing. 1981, pp. 263 - 277.
[25]. T. Nesson, S. L. Johnsson. “ROMM routing on mesh and torus networks. Proceedings of the 7th annual ACM symposium on Parallel algorithms and architectures. 1995, pp. 275 - 287.
[26]. D. Seo, A. Ali, W.-T. Lim, N. Rafique. “Near-optimal worst-case throughput routing for two-dimensional mesh networks,” Proceedings of 32nd International Symposium on C*omputer Architecture, 2005. ISCA '05. pp. 432 - 443.
[27]. L. Benini, G. De Michelli. “Networks on Chips: Technology and Tools,” Morgan Kaufmann Publishers, 2006.
[28]. H. Wang, et al., "Orion: A Power-Performance Simulator for Interconnection Networks," in 35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002, pp. 294 - 305.
[29]. W. J. Dally and C. L. Seitz. “Deadlock-Free Message Routing in Multiprocessor Interconnection Networks,” IEEE Transactions on Computers. May 1987, Vols. C-36, 5, pp. 547-553.
[30]. W. J. Dally. “Virtual-channel flow control. IEEE Transactions on Parallel and Distributed Systems,” March 1992, Vol. 3, 2, pp. 194-204.
[31]. P. P. Pande. “Networks on chip: Emerging interconnect infrastructures for MPSoC platforms,” PhD thesis, Dept. of Electrical and Computer Engineering, University of British Columbia. July 2005.
[32]. E. Rijpkema, K.G.W. Goosens, A.Radulescu, J.Dielissen, J.van Meerbergen,P.Wielage, and E.Waterland, “Trade Offs in the design of a Router with Both Guaranteed and Best- Effort Services for Network on Chip,” in Proc. Design, Automataion and Test in Europe Conference and Exhibition, 2003, pp. 350-355.
[33]. S. Kumar, A. Jantsch, J.-P.Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hermani, “A New on Chip Architecture and Design Methodology,” in Proc. IEEE Computer Society Annual Symposium on VLSI, Apr. 25-26, 2002, pp. 105-112.
[34]. R. Mulllins, A. West, and S. Moore, “Low-Latency Virtual-Channel Router for On-Chip Networks,” in Proc. 31st Annual International Symposium on Computer Architecture, June 19-23, 2004, pp.188-197.
[35]. A.Mello, L. Tedesco, N. Calazans, and F. Moraes, “Virtual Channels in Network on Chip: Implementation and Evaluation on Hermes NoC,” in Proc.th Symposium on Integrated Circuits and Systems Design, Sept. 4-7.2005, pp. 178-183.
[36]. M. Ali, et al., "Using the NS-2 Network Simulator for Evaluating Network on Chips (NoC)," in International Conference on Emerging Technologies, 2006, pp. 506 - 512.
[37]. Noxim website. Available: www.noxim.org .
[38]. M. Lis, et al., "DARSIM: a parallel cycle-level NoC simulator," in 6th Annual Workshop on Modeling, Benchmarking and Simulation, 2010.
[39]. A. Pullini, et al., "NoC Design and Implementation in 65nm Technology," in ACM/IEEE International Symposium on Networks-on-Chip, 2007, pp. 273-282.
[40]. Gajski, D. D., Gerstlauer, A., Abdi, S., Schirner, G., “Embedded System Design, Modeling, Synthesis, Verification.” Kluwer Academic Publishers, 2009
[41]. Davey, B. A. and Priestley, H. A., “Introduction to Lattices and Order,” Cambridge University Press, second edition, 2002.
[42]. Lee, E. A. and Messerschmitt, D. G., “Synchronous Data Flow,” Proceedings of the IEEE, vol. 75, no. 9, p 1235-1245, September, 1987.
[43]. Kahn, G., “The Semantics of a Simple Language for Parallel Programming,” In Proc. IFIP Congress74. North-Holland Publishing Co., 1974.
[44]. Milner, R., “A Calculus of Communicating Systems”, Springer, 1980.
[45]. Hoare, C. A. R., “Communication Sequential Processes,” Prentice Hall 1985.
[46]. Predictive Technology Model (PTM), Arizona State University, Available: http://ptm.asu.edu/, Last accessed: Dec. 2010.
[47]. M. Steven, Martin, F. Krisztian, M. Trevor, and D. Blaauw. “Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads.” pp. 721 – 72,. On 2002.
[48]. R. Ho. “on-chip wires : scaling and efficiency.” On August 2003.
[49]. S.Bhat, “energy models for network-on-chip components.” On Dec 2005.
[50]. Shyh-Chyi Wong, Winbond TSM, “An Extraction Method to Determine Interconnect Parasitic Parameters.” vol. 11, no. 4, on 2002.
[51]. S. Ahuja, W. Zhang, A. Lakshminarayana, and S. K. Shukla, “A Methodology for Power Aware High-level Synthesis of Co-Processors from Software Algorithms,” proceedings of International VLSI design Conference, India, 2010, pp. 282–287, January, 2010.
[52]. S. Ahuja and S. K. Shukla, “MCBCG: Model Checking Based Sequential Clock-Gating,” IEEE International workshop on High Level Design Validation and Test, pp. 20–25, November 2009.

[53]. N. Agarwal and N. Dimopoulos, "High-level fsmd design and automated clock gating with codel," Electrical and Computer Engineering, Canadian Journal of, pp. 31–38, 2008.
[54]. S. Ahuja,W. Zhang, and S. K. Shukla, "System level simulation guided approach to improve the efficacy of clock-gating," To appear in the IEEE International workshop on High Level Design Validation and Test, June 2010.
[55]. J. Lee, B. Nam and H. Yoo, "Dynamic voltage and frequency scaling (DVFS) scheme for multi-domains power management," IEEE Asian Solid-State Circuits Conference (ASSCC '07), Nov. 2007, pp. 360 – 363.
[56]. K. Malkowski, P. Raghavan, M. Kandemir, and M. J. Irwin, "Phase-aware adaptive hardware selection for power-efficient scientific computation," International Symposium on Low Power Electronics and Design (ISPLED'07), Aug 2007, pp. 403-406.
[57]. E. Beigne, F. Clermidey, H. Lhermet, S. Miermont, Y. Thonnart, X. Tran, A. Valentin, D. Varreau, P. Vivet, X. Popon and H. Lebreton, "An asynchronous power aware and adaptive NOC based circuit," IEEE Journal of solid-state Circuits, Vol. 44, NO. 4, April 2009, pp. 1167 – 1177.
[58]. X. Wang, K. Ma, and Y. Wand, "Adaptive power control with online model estimation for chip multiprocessors," IEEE Transactions on parallel and Distributed Systems, Vol. 22, No. 10, Oct. 2011, pp. 1681-1696.
[59]. S. Herbert, S. Garg, and D. Marculescu, "Exploiting process variability in voltage/frequency control," IEEE Transactions on VLSI systems, Vol. 20, No. 8, Aug 2012 pp. 1392-1404.
[60]. J. Wang, Y. Qian, J. Lu, B. Li, M.Zhu, W. Dou," Designing Voltage-Frequency Island Aware Power-Efficient NoC through Slack Optimization," Information Science and Applications (ICISA), 2014 International Conference, May 2014 pp 1 – 4.
[61]. J. Zhan, N. Stoimenov, J.Ouyang, L.Thiele,V.Narayanan, X.Yuan ,"Optimizing the NoC Slack Through Voltage and Frequency Scaling in Hard Real-Time Embedded Systems," IEEE Transactions on Design of Integrated Circuits and Systems,Vol.33 , Nov 2014 pp 1632 – 1643.
[62]. Ichiba, F., Suzuki, K., Mita, S., Kuroda, T., Furuyama, T.: Variable supply-voltage scheme with 95 In: ISLPED '99: Proceedings of the 1999 international symposium on Low power electronics and design, pp. 54–59. ACM, New York, NY, USA (1999).
[63]. Tsividis, Y.P.: Operation and Modeling of the MOS Transistor. McGraw-Hill, New York (1987)
[64]. C. Hale, G. Boris, W. Keckler; "Segment Gating for Static Energy Reduction in Networks-On-Chip." Dec 2009.
[65]. T.Simunic and S.Bo yd, Managing power consumption in networks on chips. Proceedings of Design, Automation, and Test in Europe (2002), pp.110–116.
[66]. L.Shang, L.-S.Peh, and N.K.Jha, Dynamic voltage scaling with links for power optimization of interconnection networks. Proceedings of International Symposium on High-Performance Computer Architecture (2003), pp.79–90.
[67]. J.Kim and M.Horo witz, Adaptive supply serial links with sub-1V operation and per-pin clock recovery. Proceedings of International Solid-State Circuits Conference (2002), pp.1403–1413.
[68]. sKirkpatrick, S., Gelatt, C.D., Vecchi, M.P. 1983.Optimization by simulated annealing. Sci.Vol. 220, pp. 671-680.
[69]. D. T. Connolly. 1990. An improved annealing scheme for the QAP. Eur. J. Oper. Res. Vol. 46, pp. 93-100.
[70]. P. J. M. Van Laarhoven, E. H. L. Aarts, and J. K. Lenstra. 1992. Job Shop Scheduling by Simulated Annealing. Oper. Res. vol.40,pp.113-125.
[71]. Alfonsas Misevicius.2003. A modified simulated annealing algorithm for the quadratic assignment problem. Informatica.vol.14, pp.497–514.
[72]. E. Rothberg. 2007. An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS. J. Comput. Vol. 19 pp. 534–541.
[73]. A. Al-khedhairi.2008. Simulated annealing metaheuristic for solving P-median problem. Int. J. Contemp. Math. Sciences vol.3. pp.1357-1365.
[74]. Xu Hao. 2010. Optimization Models and Heuristic Method Based on simulated annealing Strategy for Solving Travelling Solution Problem. Appl. Mech. Mater. vols (34-35),pp 1180-1184.
[75]. F. Glover.1986.Future paths for integer programming and links to artificial intelligence. Comput. & Oper. Res. Vol. 13, pp. 533-549.
[76]. Laguna, M., J.W. Barnes, F. W. Glover. 1991. Tabu search methods for a single machine scheduling problem. J. Intel. Manuf. Vol. 2, pp. 63-74.
[77]. Barnes, J. W and M. Laguna. Solve the multiple-machine weighted flow time problem using tabu search. IIE Trans.vol. 25, 1993, pp. 121-128.
[78]. Battiti, R. and G. Tecchiolli.1994.The Reactive Tabu Search. ORSA J Comput. Vol, 6, pp.126–140.
[79]. Laguna, M., J. P. Kelly, J. L. González Velarde, and F. Glover. 1995. Tabu search for the multilevel generalized assignment problem. Eur. J. Oper. Res. Vol. 82,pp. 176-189.
[80]. Carlton, W.and J. W. Barnes. Solving the Traveling Salesman Problem with Time Windows Using Tabu Search. IIE Trans. Vol. 28,1996, pp. 617-629.

[81]. Lokketangen, A., F. Glover. 1998. Solving zero-one mixed integer programming problems using tabu search. Eur. J. Oper. Res. vol. 106, pp. 624-658.
[82]. Nanry, W. P. and J. W. Barnes. 2000. Solving the pickup and delivery problem with time windows using reactive tabu search. Trans.Res. Part B vol.34, pp.107-121.
[83]. González-Velarde, J. L. and M. Laguna. 2002. Tabu search with simple ejection chains for coloring graphs. Ann. Oper. Res. Vol. 117, pp. 165-174.
[84]. Korycinski, D., M.M. Crawford, J.W. Barnes, and J. Ghosh. 2003. Adaptive feature selection for hyperspectral data analysis using a binary hierarchical classifier and tabu search. Proc. 2003 International Geoscience and Remote Sensing Symposium, Toulouse, France, pp. 297-299.
[85]. Scrich, C. R., V. A. Armentano and M. Laguna. 2004. Tardiness minimization on a flexible job shop: a tabu search approach. J. Intel. Manuf. vol.15,pp. 103-115.
[86]. Feo, T. A., & Resende, M. G. C. 1995. Greedy randomized adaptive search procedure. J. Global. Optimal. Vol. 6, pp. 109-133.
[87]. Atkinson, J.B .1998. A greedy randomized search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. J. Oper. Res. Soc. Vol.49, pp. 700–708.
[88]. Fleurent, C., Glover F. 1999. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. INFORMS J. Comput. Vol.11, pp. 198–204.
[89]. Binato, S., Faria, H. Jr., Resende, M.G.C.2001. Greedy randomized adaptive path relinking. Proceedings of the 4th Metaheuristics International Conference, pp. 393–397.
[90]. N. Mladenovic and P. Hansen. 1997. Variable Neighborhood Search. Comput. Oper. Res. vol. 24,pp. 1097-1100.
[91]. Hansen P and Mladenović N. 1997. Variable neighborhoods search for the p-median. Locat. Sci.Vol. 5, pp.207-226.
[92]. V.J. Rayward-Smith and A.S. Wade. 2000. Effective Local Search for the Steiner Tree Problem. In Advances in Steiner Trees ed. by Ding-Zhu Du, J. M.Smith and J.H. Rubinstein, Kluwer.
[93]. C. Avanthay, A. Hertz, and N. Zufferey. 2003. A Variable Neighborhood Search for Graph Coloring. Eur. J. Oper. Res. Vol. 151, pp. 379–388.
[94]. Puchinger, J. and G. R. Raidl. 2005. Relaxation Guided Variable Neighborhood Search. In Proceedings of the XVIII Mini EURO Conference on VNS, Tenerife, Spain
[95]. Consoli S, Darby-Dowman K, Mladenovic N and Moreno J. 2009. Variable neighborhood search for the minimum labelling Steiner tree problem. Ann. Oper. Res. Vol. 172, pp.71-96.
[96]. Fraser, Alex (1957). "Simulation of genetic systems by automatic digital computers. I. Introduction". Aust. J. Biol. Sci. 10: 484–491.
[97]. Bremermann, H.J. " The evolution of intelligence. The nervous system as a model of its environment", Technical report, no.1, contract no. 477(17), Dept. Mathematics, Univ. Washington, Seattle, July, 1958.
[98]. J. H. Holland. Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press, 1975.
[99]. Goldberg, David (2002). The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Norwell, MA: Kluwer Academic Publishers. ISBN 978-1402070983.
[100]. Tate David M., Smith Alice E. 1995. A Genetic Approach to the Quadratic Assignment Problem. Comput. Oper. Res. vol. 22, pp.73-83.
[101]. J. E. Beasley and P. C. Chu(1996) A genetic algorithm for the set covering problem. Eur. J. Oper. Res. Vol 94, pp. 392–404.
[102]. P.C, Chu and J. E. Beasley. 1997. A genetic algorithm for the generalized assignment problem. Comput. Oper. Res. vol.24, pp.17–23.
[103]. Chu, P.C. and J.E. Beasley.1998. A Genetic Algorithm for the Multidimensional Knapsack Problem. J Heuristics. Vol. 4, pp. 63-86.
[104]. R. K. Ahuja, J. B. Orlin and A. Tiwari. 2000. A descent genetic algorithm for the quadratic assignment problem. Comput. Oper. Res. vol. 27, pp. 917–934.
[105]. Olli Braysy. 2001. Genetic algorithms for the vehicle routing problem with time windows.Technical Report, 1/2001, University of Vaasa, Vaasa, Finland.
[106]. Chiung Moon et al. 2002. An efficient GA for the TSP with precedence constraints. Eur. J. Oper. Res. Vol. 140, pp. 606-617.
[107]. Omar M. Sallabi and Younis El-Haddad. 2009. An Improved Genetic Algorithm to Solve the Travelling Salesman Problem. World Acad. Sci. Eng. and Technol. vol.52, pp. 471-474.
[108]. Ou, G., Tamura, H., Tanno, K. and Tang, Z. 2010. A method of solving scheduling problems using an improved guided genetic algorithm. Electron. Comm. Jpn. vol.93, pp. 15–22.
[109]. Kusum Deep and Hadush Mebrathu. 2011. Combined Mutation Operators of Genetic Algorithm for the Travelling Salesman Problem. Int. J. Comb. Optim. Probl. Inform.Vol. 2, pp. 2-24.
[110]. Neelam Tyagi and Varshney R.G. 2012. A Model To Study Genetic Algorithm For The Flowshop Scheduling Problem. J. Inform. Oper. Manag. ISSN: 0976–7754 & E-ISSN: 0976–7762, Volume 3, pp-38-42.

[111]. Dorigo M, Maniezzo V, Colorni A. 1991. Positive feedback as a search strategy. Technical Report 91-016, Politecnico di Milano, Italy.
[112]. Colorni A., Dorigo M., Maniezzo V. & Trubian M. 1994. Ant system for job-shop scheduling. Belgian J. Oper. Res. Stat. and Comput. Sci. vol. 34, pp. 39-54.
[113]. Bullnheimer, B., Hartl, R.F., Strauss, C.1999. An improved ant system algorithm for the vehicle routing problem. Ann. Oper. Res.vol. 89, pp. 319–328.
[114]. E-G. Talbi, O. Roux, C. Fonlupt, and D. Robilliard. 2001. Parallel ant colonies for the quadratic assignment problem. Future. Gener. Comput.Syst. vol. 17,pp. 441–449.
[115]. Bell, J.E., McMullen, P.R. 2004.Ant colony optimization techniques for the vehicle routing problem. Adv. Eng. Inform. Vol. 1, pp. 41–48.
[116]. Frank Neumann, Carsten Witt. 2006. Ant Colony Optimization and the Minimum Spanning Tree Problem. Electron. Colloq. Comput. Complex. (ECCC) 13.
[117]. Ulrike Schneider. 2011. A Tabu search tutorial based on a real world scheduling problem. Cent. Eur. J. Oper. Res. vol. 19, pp. 467-493.
[118]. J. Kennedy and R. C. Eberhart. 1995. Particle swarm optimization. Proc. of IEEE Int Conf. on Neural Netw Piscataway, NJ, USA , pp. 1942-1948.
[119]. Ayed A. Salman, Imtiaz Ahmad, Sabah Al-Madani. 2002. Particle swarm optimization for task assignment problem. Microprocess and Microsyst. Vol.26, pp. 363-371.
[120]. KP Wang, L Huang, CG Zhou, W Pang. 2003. Particle swarm optimization for traveling salesman problem. Int. Conf. Mach. Learn. and Cybern.Vol. 3, pp.1583-1585.
[121]. Tasgetiren MF, Sevkli M, Liang YC, Gencyilmaz G. Particle swarm optimization algorithm for single machine total weighted tardiness problem. In: Proc. of the IEEE Congr Evol Comput, Portland,vol.2, 2004, pp.1412–1419.
[122]. Chen, A.L., G.K. Yang and Z.M. Wu. 2006. Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. J. Zhejiang University Sci. vol. 7, pp.607-614.
[123]. Lei Yuan, Zhendong Zhao. A Modified Binary Particle Swarm Optimization Algorithm for Permutation Flow Shop Problem. In Proceedings of the Sixth Int Conf Mach Learn Cyberns (ICMLC 2007), Hong Kong (19-22) August 2007, pages 902-907.
[124]. Sevkli, M., Guner, A. R. 2008. A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem. J. Artif. Evol. Appl. Vol. 2008, pp. 1-9.
[125]. Consoli, S., Moreno-Perez, JA., Darby-Dowman, K. and Mladenović N. 2010. Discrete particle swarm optimization for the minimum labelling Steiner tree problem. Nat. Comput. Vol. 9 pp. 29- 46.
[126]. Kunpeng KANG. 2012. A Fast Particle Swarm Optimization Algorithm for Large Scale Multidimensional Knapsack Problem. J. Comput. Inform. Syst. Vol. 8, pp.2709–2716.
[127]. Christian Blum, Jakob Puchinger, Günther R. Raidl, Andrea Roli. 2011.Hybrid metaheuristics in combinatorial optimization. A survey Appl. Soft. Comput. Vol.11, pp. 4135-4151.
[128]. C. Walshaw. 2004.Multilevel refinement for combinatorial optimization problems. Ann. Oper. Res. Vol.131, pp.325–372.
[129]. S. Wolfram, A New Kind of Science. Wolfram Media, 2002.
[130]. S.N.Sivanandam, S.N.Deepa, Introduction to Genetic Algorithms, Springer-Verlag Berlin, New York Heidelberg 2008, Library of Congress Control Number: 2007930221, ISBN 978- 3-540-73189-4
[131]. J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, vol. 24, no. 4, p. 551--562, 2005.
[132]. J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in Proceedings of the 2003 Asia and South Pacific Design Automation Conference, Kitakyushu, Japan, 2003, pp. 233-239.
[133]. R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," Trans. Comp.-Aided Des. Integ. Cir. Sys., vol. 28, no. 1, pp. 3-21, 2009.
[134]. Andrew J. Pagea, Thomas M. Keanea, Thomas J. Naughtonb, "Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system," Journal of Parallel and Distributed Computing, Vol. 70, no.7, pp. 758–766.
[135]. E.S.H. Hou, N. Ansari, R. Hong Ren, "A genetic algorithm for multiprocessor scheduling," IEEE Transactions on Parallel and Distributed Systems, Vol. 5, no.2,pp. 113 – 120, On 2002.
[136]. G. Amalarethinam, M. Josphin, "Dynamic Task Scheduling Methods in Heterogeneous Systems," International Journal of Computer Applications, Vol. 110, Iss. 6, On Jan. 2015.
[137]. T. Moscibroda, O. Mutlu. "A Case for Bufferless Routing in On-Chip Networks". In Proc. ISCA, USA, June 2009.
[138]. T. Li. "Estimation of Power Consumption in Wormhole Routed Networks on Chip." On 2005.

[139]. V. Nollet, et al, “Low Cost Task Migration Initiation in a Heterogeneous MP-SoC,” In: Design, Autom ation and Test in Europe Conference (DATE), pp. 252-253, on 2005.
[140]. S. Bertozzi, A. Acquaviva, D. Bertozzi, A. Poggiali, “Supporting Task Migration in Multi- rocessor Systems-onchip: a Feasibility Study,” In: Design, Automation and Test in Europe Conference (DATE), pp. 15-20, Munich, Germany, on 2006.
[141]. Stefano Bertozzi, Andrea Acquaviva, Davide Bertozzi, and Antonio Poggiali. Supporting task migration in multi-processor systems-on-chip: A feasibility study. In Design, Automation and Test in Europe, 2006. DATE ’06. Proceedings, pages 15–20. The EDA Consortium and EDAA : European Design and Automation Association and IEEE-CS DATC : The IEEE Computer Society, European Design and Automation Association, 2006.
[142]. L. Ganthel adn S. Layouni, M.E.A. Benkhelifa, F. Verdier, and S. Chauvet. Multiprocessor task migration implementation in a reconfigurable platform. In International Conference on Reconfigurable Computing and FPGAs, 2009.
[143]. Richard Barry. FreeRTOS Reference Manual: API functions and Configuration Options. Real Time Engineers Ltd, 2009.
[144]. Seo Y, Kim J, Seo E, “Effective analysis of DVFS and DPM in mobile devices,” Journal of Computer Science and Technology ,Vol. 27, no 4, pp. 781–90. On 2012.
[145]. D.Sengupta, R. Saleh, “Application-driven voltage-island partitioning for low-power system-on-chip design,” IEEE Transaction on Computer Aided Des Integrated Circuits Systems, vol.28, no..3, pp. 316–326, 2009.
[146]. Yu B, Dong S, Chen S, Goto S, “Voltage and level shifter assignment driven floor planning. IEICE,” Transaction Fundam Electronics Communication Computer Science, vol.92, pp. 2990-2997, 2009.
[147]. Jin X, Goto S, “Hilbert transform-based workload prediction and dynamic frequency scaling for power efficient video encoding,” IEEE Transactions of Computer Aided Des Integrated Circuits System ,Vol. 31, no. 5, pp. 649–61.2012.
[148]. David R, Bogdan P, Marculescu R, Ogras U, “Dynamic power management of voltage-frequency Island partitioned networks-on-chip using intel single chip cloud computer,” In: 5th IEEE/ACM international symposium on networks on chip, pp. 257–258, 2011.
[149]. Wang X, Ma K, Wand Y, “Adaptive power control with online model estimation for chip multiprocessors,” IEEE Transactions Parallel Distribution Systems, Vol. 22, no. 10, pp. 1681–1696, on 2011.
[150]. Gorti NPK, Somani AK, “Reliability aware dynamic voltage and frequency scaling for improved microprocessor lifetime,” ACM SIGOPS Operation System Revolution, Vol. 47 no. 3,pp. 10–7. On 2013.
[151]. H. Maaranen, K. Miettinen, and A. Penttinen, “On initial populations of a genetic algorithm for continuous optimization problems,” J. of Global Optimization, vol. 37, no. 3, pp. 405–436, Mar. 2007.
[152]. S. Johanna, S. Marius, J. Wang G.Guy, “A Multi-Objective Approach for Multi-Application NoC Mapping”. IEEE Circuits and Systems Latin American Symposium, pp.1-6,on 2011.
[153]. Y. Tei,, N. Shaikh-Husin,, Y. Hau, E. Supriyanto, M. Marsono, “Network-on-Chip Application Mapping based on Domain Knowledge Genetic Algorithm”. Advances in Robotics, Mechatronics and Circuits, pp.115-120,on 2014.
[154]. F.Boutekkouk, ―A Cellular Automaton Based Approach for Real Time Embedded Systems Scheduling Problem Resolution‖, on Artificial Intelligence Perspectives and Applications, Vol. 347, pp. 13-22, On 2015.
[155]. G. Ascia, V. Catania, M. Palesi, ―Multi-objective mapping for meshbased NoC architectures.‖ pp.182 – 188, Sept. 2004.
[156]. N.H. Packard (1988), ―Adaptation toward the edge of chaos,‖ In J.A.S.Kelso, A.J.Mandell, and M.F. Shlesinger, editors, Singapore, World Scientific on Dynamic Patterns in Complex Systems, pp. 293– 301.
[157]. S. Mirosáaw, S. Franciszek, ―Searching for efficient cellular automata based keys applied in symmetric key cryptography,‖ Annales UMCS Informatica Lublin-Polonia Sectio AI, pp. 49-60, On 2007.
[158]. F. Seredynski, ―Discovery with Genetic Algorithm Scheduling Strategies for Cellular Automata,‖ Proc. Parallel Problem Solving from Nature—PPSN V, A.E. Eiben, T. Back, M. Schoenauer, and H.- P. Schwefel, eds. pp. 643-652, on 1998.
[159]. F. Seredynski and C.Z. Janikow, ―Designing Cellular Automatabased Scheduling Algorithms,‖ GECCO-99: Proc. Genetic and Evolutionary Computation Conf., W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, eds., pp. 587-594, July 1999.
[160]. M. Mitchell, P.T. Hraber, and J.P. Crutchfield, ―Evolving cellular automata to perform computation: Mechanisms and impedients,‖Phys. D, vol. 75, p. 361–391, on 1994.
[161]. Carneiro, M.G., Oliveira, M.B, ―Synchronous cellular automata-based scheduler initialized by heuristic and modeled by a pseudo-linear,‖Natural Computing Journal, vol. 12, no. 3, pp. 339-351 , 2013.
[162]. D. Andre, F.H. Bennet III, and J.R. Koza, ―Discovery by Genetic Programming of a Cellular Automata Rule that Is Better than Any Known Rule for the Majority Classification Problem,‖ Proc. First Ann. Conf. Genetic Programming J.R. Koza, D.E. Goldberg, D.B. Fogel, and R.L. Riolo, eds., pp. 3-11, 1996.
[163]. R. Das, M. Mitchell, and J. Crutchfield, ―A Genetic Algorithm Discovers Particle-Based Computation in Cellular Automata,‖ Parallel Problem Solving from Nature-PPSN III, pp. 344-353, 1994.

[164]. P.W. Shor. Algorithm for quantum computation : discrete logarithms and factoring. IEEE, Proceedings of the 35th Annual Symposium on Foundation of Computer Science, Piscata-way, IEEE Press, 1994.
[165]. L. K. Grover. A fast quantum mechanical algorithm for database search. in Proc. of the 28th Annual ACM Symphosium on the Theory of Computing, pages 212–219, 1996.
[166]. L. K. Grover. Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Lett., 79(2):325–328, 1997.
[167]. A. Narayanan and M. Moore, "Quantum-inspired genetic algorithms," in Proc. 1996 IEEE Int. Conf. Evolutionary Computation, Piscataway, NJ, May 1996, pp. 61–66, IEEE Press.
[168]. K. H. Han and J. H. Kim, "Quantum-inspired evolutionary algorithm for a class of combi-natorial optimization," IEEE Trans. Evol. Comput, vol. 6, pp. 580–593, Dec. 2002.
[169]. Ajit Narayanan and Mark Moore. Quantum-inspired genetic algorithms. In Evolutionary Computation, 1996., proceedings of IEEE International Conference on, pages 61–66. IEEE, 1996.
[170]. Kuk hyun Han and Jong-Hwan Kim. Genetic quantum algorithm and its application to combinatorial optimization problem. In in Proc. 2000 Congress on Evolutionary Computa-tion. Piscataway, NJ: IEEE, pages 1354–1360. Press, 2000.
[171]. Kuk hyun Han, Kui hong Park, Chi ho Lee, and Jong hwan Kim. Parallel quantum-inspired genetic algorithm for combinatorial optimization problem. In in Proc. 2001 Congress on Evolutionary Computation. Piscataway, NJ: IEEE, pages 1422–1429. Press, 2001.
[172]. Zhenquan Zhuang Junan Yang, Bin Li. Research of quantum genetic algorith and its appli-cation in blind source separation. Journal of Electronics, 20(1):62–68, 2003.
[173]. Gexiang Zhang, Weidong Jin, and Laizhao Hu. A novel parallel quantum genetic algorithm. In Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on, pages 693–697. IEEE, 2003.
[174]. Gexiang Zhang, Weidong Jin, and Na Li. An improved quantum genetic algorithm and its application. In Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, pages 449–452. Springer, 2003.
[175]. Comparison of genetic algorithm and quantum genetic algorithm. http://ccis2k.org/iajit/PDF/vol.9,no.3/2107-6.pdf.
[176]. Zakaria Laboudi and Salim Chikhi. Comparison of genetic algorithm and quantum genetic algorithm. The international Arab Journal of Information Technology, 9:243, 2012.
[177]. Jun Zhi, Huaixiao Wang, Jianyong Liu and Chengqun Fu. The improvement of quan-tum genetic algorithm and its application on function optimization. Mathematical Problems in Engineering,, 2013:10 pages, 2013.
[178]. Bin Li and Zhenquan Zhuang. Genetic algorithm based-on the quantum probability repre-sentation. In Hujun Yin, Nigel M. Allinson, Richard T. Freeman, John A. Keane, and Simon J. Hubbard, editors, Intelligent Data Engineering and Automated Learning - IDEAL 2002, Third International Conference, Manchester, UK, August 12-14, Proceedings, volume 2412 of Lecture Notes in Computer Science, pages 500–505. Springer, 2002.
[179]. Donald A. Sofge. Proceedings of the second quantum interaction symposium (qi-2008), college publications. Prospective of algorithms for Quantum, UK, 2008.
[180]. R. Zhou and J Cao. Quantum novel genetic algorithm based on parallel subpopulation computing and its applications;. Artf. Intell. Rev.2009.
[181]. M. Sacanamboy, F. Bolaños, and R. Nieto, "A Primer for Mapping Techniques on NoC Systems", on Embedded Systems and Applications the WorldComp International Conference Proceedings. On dec.2014.
[182]. AK Singh, M Shafique, A Kumar, J Henkel," Mapping on multi/many-core systems: Survey of current and emerging trends, On Design Automation Conference. Pp.1-10. On.2013.
[183]. Wooseok Lee, "ArterisFlexNoCResilience Package". Design Automation for Embedded Systems Journal.On.2014."

Annex a: conferences and journals papers

The conferences and journals papers that are achieved in the context of the thesis:

1. Conferences papers

1. Djalila Belkebir and Fateh Boutekkouk, "Hybrid Mapping Strategy in Regular Network on Chip in Minimizing Power and Latency: Evolutionary Cellular Automaton" Conference ICCSA-1 Om el bouaghi, April 2016.
2. Djalila Belkebir and Fateh Boutekkouk, "Quantum Evolutionary Cellular Automata Mapping Optimization Technique Targeting Regular Network on Chip". In book: Automation Control Theory Perspectives in Intelligent Systems, Chapter: Automation Control Theory Perspectives in Intelligent Systems, Publisher: Springer International Publishing, Editors: Radek Silhavy, pp.129-140, January 2016.
3. Djalila Belkebir and Fateh Boutekkouk ,"An evolutionary cellular automaton for minimizing energy consumption in network on chip-based soft real-time embedded systems" 2015 3rd International Conference on Control, Engineering & Information Technology (CEIT), tlemcen, Algeria, May 2015.
4. Djalila Belkebir and Fateh Boutekkouk , "High Level Power Model in 2-D mesh Networkon-Chip", Journées d'étude sur les systèmes complexes (JESC'2015), April 2015.

2. Journals papers

1. Djalila Belkebir and Fateh Boutekkouk , "Two-steps into energy consumption optimization due to the mapping of multimedia application to network on chip architecture", Accepted in the journal: "International Journal of Intelligent Systems Technologies and Applications (IJISTA)" inderscience, 2015.

2. Djalila Belkebir and Fateh Boutekkouk, “Toward a new multi-objective optimization mapping strategy in network on chip” Accepted in the journal:”International Journal of High Performance Systems Architecture (IJHPSA)” inderscience , 2015.
3. Djalila Belkebir and Fateh Boutekkouk, “Evolutionary cellular automata: Hybrid mapping technique in network on chip architecture”, Accepted in the journal:”Journal of Information Processing Systems (JIPS)”, 2015.
4. Djalila Belkebir and Fateh Boutekkouk,“New routing strategy based bufferless routers” Accepted in the journal:”Journal of Information Processing Systems (JIPS)”, 2016.