POPULAR DEMOCRATIC REPUBLIC OF ALGERIA
HIGHER EDUCATION AND SCIENTIFIC RESEARCH'SMINISTRY

University KASDI MERBAH OUARGLA
Faculty of new information technologies and communication
Department of computer and information technology

LMD Master 2018/2019
Option:

Dissertation

For the Master degree in network administration and security
Title:

Solving Task Migration Problems by Optimizing the Crossover and Mutation probabilities using tabu search

Presented in : 02 / 07 / 2019

Presented by:
. Benazia Meriem

Kemassi Imane

Jury members:

Dr.Chrite Abd El Hakim	Université KASDI MERBAH OUARGLA	President
Dr. Djediai Hemida	Université KASDI MERBAH OUARGLA	Examiner
Dr. Belekbir Djalila	Université KASDI MERBAH OUARGLA	Directed

Academic Year 2018-2019

ABSTRACT

Task migration (TM) is the act of transferring a process between two processing units, it is considered as an effective strategy to achieve load balancing and higher source utilization, in another way, it is a call to the operating system during run time.TM has been widely used in the distributed systems domain, but as a drawback, the migration time overhead may be long in point that it will increase network congestion and degrade the system performance, we aim to optimize those problems by using genetic algorithms (GA).

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 a 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.

Crossover is the process in which genes are selected from the parent chromosomes and new offspring is created, where the mutation is applied on one chromosome. The genetic algorithms' performance is largely influenced by crossover and mutation operators, the probabilities of applying those operators can be optimized by tabu search technique probabilities.

Keywords:

Task migration, Embedded Systems, Genetic algorithms, tabu search, crossover and mutation probabilities.

ملخص

ترحيل المهام هو نقل عملية بين وحدتي معالجة، و بتعريف آخر هو استدعاء لنظام التشغيل أثناء وقت التشغيل و أستخدم على نطاق واسع في مجال الأنظمة الموزعة ،برغم من هذه المحاسن إلا أن العيب فيها يكمن في طول وقت الترحيل مما يتسبب في زيادة ازدحام الشبكة و تقليل من أداء النظام. نحن نهدف إلى تحسين هذه المشاكل باستخدام الخوارزميات الجينية.

البحث في علامة تبويب هو إجراء عالي المستوى مصمما لإيجاد أو تحديد مجريات الأمور التي قد توفر حلا جيدا للتحسين ، بحيث تطبق خوارزمية البسيطة للبحث في علامة تبويب بحثا محليا في كل عملية تكرارية لتحديد أفضل حل ضمن قائمة تبويب و تعيينه كحل جديد ،وتعرف قائمة التبويب بأنها عبارة عن ذاكرة قصيرة المدى حيث يتم تخزين كروموسومات فيها.

يتأثر أداء خوارزميات الجينية إلى حد كبير بعملتي التبديل و التحويل. حيث أن التبديل هو عملية اختيار الجينات من الكروموسومات الأصل و إنشاء كروموسوم جديد، و يتم تطبيق التحويل على جيني واحد و تحسن احتمالات تطبيق التبديل و التحويل من خلال تحسين احتمالات البحث في التبويب.

الكلمات المفتاحية:

ترحيل المهام ، الأنظمة المدمجة ، الخوارزميات الجينية ، البحث عن علامة تبويب ، احتمالات التبادل والتحويل.

.

Table of content

Figures list

Tables list

GENERAL INTRODUCTION

1. Problem definition

Now a days, embedded system is inseparable part of our lives; in home, in office or on our moves (cars, phones, PC, ..), we are always surrounding by embedded system, because it is embedded as a part of a complete device system that includes hardware, such as electrical and mechanical components or a software, such as multimedia applications. A System on Chip (SOC) is a single chip that integrates multiple functionalities that are typically needed for a system into the single chip itself. As the SoC became more complex with the increasing number of low speed, medium speed, and high speed peripherals, this gave way to the design of on-chip interconnect IP required to facilitate effective communication amongst all IO blocks and internal blocks of a SoC, the on-chip interconnect is alternatively termed as Network-on-chip (NoC).

Multimedia applications are real-time systems, consists of tasks and messages between those tasks, The tasks are performed at the level of processors located at the NoC level where each task has a specific execution time defined as a deadline, 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 mapping and scheduling algorithms are needed, comes in so that NoC is fair to all, meets their need.

The use of Modern chips in mobile or battery environment has given rise to the requirement of power savings and management. Reduced power consumption means higher battery life, and low execution time means faster performance. But, the problem of the increasing of the number of cores on a single chip leads to the issue of energy consumption and latencies so that number of nodes will be augmented as well, and the time of transmission throw the links will be bigger. Therefore, the power and time management are key issue towards the optimization of the performances in NoC.

Task migration is process that describe the movement of an executing task from one host processor (source) in distributed computing system to another (destination), in which processors contain a set of tasks and each task waits for its activation time for execution, so it may take a great time and effort of the processor to perform all tasks that are assigned to it. The process of transferring tasks from one processor to another is used to reduce the pressure on processors (the occupation rate) and reduce the time and power of execution tasks in such a way that the migration overheads won’t affect the overall NoC’ performances.

However, many heuristic algorithms are designed to find a near optimal mapping such as stochastic global search technique that produce high quality solution, for that, the based bio-inspired methods have been used to solve scheduling and mapping problems such as genetic algorithms ,The genetic algorithms’ performance is largely influenced by crossover and mutation operators, the probabilities of applying those operators can be controlled by tabu search technique.

2. Contributions

The genetic algorithm examines complex problems by developing a range of possibilities for possible solutions to these problems. Therefore, in our work, we will use it in order to optimize the NoC performances such as energy consumption under timing constraints with the combination of tabu search. Also, we will exploit the approach of migrating tasks from a current processor to another, under certain constrains that lead to improve the performances and minimize the migration overheads.

The process of the algorithms is divided into several stages, the most important is the selection and experimentation phase, which have the role to choose the position of tasks at the processors level and then to take the decision of transfer possibility or not, in which means, the decision of migrating the current task or not. The possibility of the transfer depends on some criterion that should be available on the destination node, those criterions represent the needs of the migrated task such as; the sufficient buffer space to store the incoming migrated data or the availability of the channels to transfer those data. The two processors are chosen in terms of the processor, Faster and less effort.

The tabu search is applying on chromosomes in both; selection phase or population phase (the creation of the first population),and each chromosome will be tested. If this chromosome increases power consumption and latency (ruin the improving of fitness function values), it will be placed in the search table to be avoided in each selection of the future generations. This means that whenever we choose a chromosome before the experiment, we go to the search table to make sure that it is not in the search table.

In NoC, some processors have a large number of tasks to execute, while others find some idle (no execution task). Therefore, these tasks bring migration to these processors to reduce pressure on them. On the other hand, there are tasks that cannot be performed until assigning them to an executable from another task. This may take time to reach, so it's time to leave it next to it to save execution time. However, this process must be routed. These tasks cannot be allowed to move randomly. Genetic solutions provide migration and an optimal schedule that includes solutions that cannot contribute to facilitating the migration process.

3. Thesis Overview

The thesis is organized as follows:

In chapter1, a description of the concepts and the main parts in the field on network on chip will be done. We will first define the embedded system with its different architecture; and then we will give an introduction of the basic concepts.

In chapter 2, we will put the light on understanding the model of energy-efficient and delay model of NoC based multimedia application. Also, we will describe the different system level-based power reduction techniques with a state of the art of the researches on the last decade with a comparison between them.

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

In chapter 4, we will detail our proposed solution, tests and experiments, with taking as case of study two benchmarks-based multimedia application task graphs: the multimedia system (MMS), the automotive industrial, and random task graph on 2Dmesh 5x5 network on chip.

We will finish this manuscript with a conclusion that summarizes the results obtained from discussing the proposed ideas.

CHAPTER I

NETWORK ON CHIP

1. Introduction

Network on Chip (NoC) is a new paradigm that replaces System on Chip (SoC) design. Increasing integration produces a situation where bus structure, which is commonly used in SoC, becomes blocked and increased capacitance poses physical problems. In NOC architecture, traditional bus structure is replaced with a network which is a lot similar to the Internet. Data communications between segments of chip are packetized and transferred through the network. The network consists of wires and routers. Processors, memories and other IP-blocks (Intellectual Property) are connected to routers. A routing algorithm plays a significant role on network's operation. Routers make the routing decisions based on the routing algorithm [104].

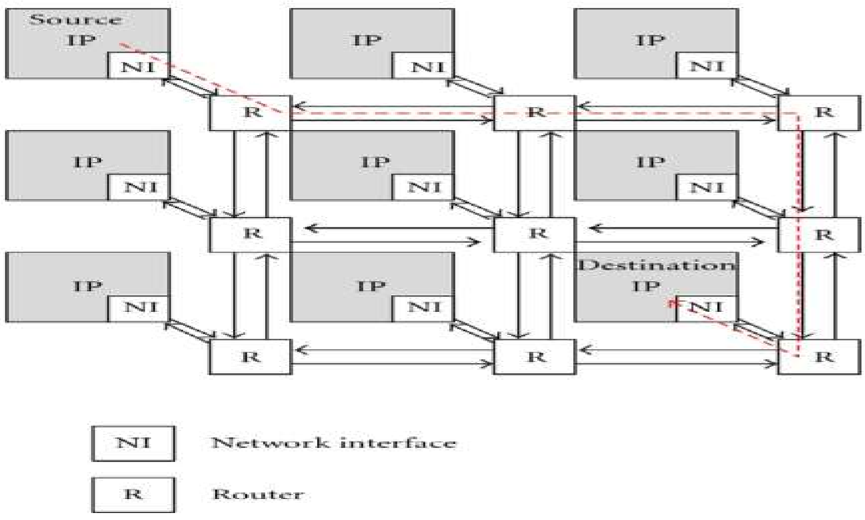


Figure 1.1: Network on Chip 2D-Mesh architecture.[89]

Networks on chip, a new interconnection concept made possible by the evolution of silicon technology, is becoming a preferred solution for simplifying the integration of complex components on systems-on-a-chip. In this chapter we will definite embedded system with a variance between different architectures of embedded system, present à different routing algorithm of NOC, the different main component and the layer communication of NOC

2. Embedded System

2.1 Embedded System Definition

There are many definition of embedded system ,we mention:

- An embedded system is a system that has software embedded into computer-hardware, which makes a system dedicated for an application (s) or specific part of an application or product or part of a larger system.
- An embedded system is one that has a dedicated purpose software embedded in a computer hardware.
- It is a dedicated computer-based system for an application(s) or product. It may be an independent system or a part of large system. Its software usually embeds into a ROM (Read Only Memory) or Flash.
- It is any device that includes a programmable computer but it is not itself intended to be a general-purpose computer.
- Embedded Systems are the electronic systems that contain a microprocessor or a microcontroller, but we do not think of them as computers– the computer is hidden or embedded in the system [82].

2.2 Real-Time System:

2.2.1 Real-Time Scheduling Techniques:

2.2.1.1 Rate Mono tonic (RM):

The concept of RM was firstly introduced by the authors in [17], and it was one of the first scheduling policies developed for real-time systems and it is still very widely used. We say that

RMS is a **static scheduling policy** because it assigns fixed priorities to processes. It turns out that these fixed priorities are sufficient to efficiently schedule the processes in many situations.

The theory underlying RMS is known as **rate-monotonic analysis (RMA)**. This theory, as summarized below, uses a relatively simple model of the system.

- All processes run periodically on a single CPU
- Context switching time is ignored.
- There are no data dependencies between processes.
- The execution time for a process is constant.
- All deadlines are at the ends of their periods.
- The highest-priority ready process is always selected for execution.

The major result of rate-monotonic analysis is that a relatively simple scheduling policy is optimal. Priorities are assigned by rank order of period, with the process with the shortest period being assigned the highest priority. This fixed-priority scheduling policy is the optimum assignment of static priorities to processes, in that, it provides the highest CPU utilization while ensuring that all processes meet their deadlines. [13]

Rate-monotonic scheduling is a static priority-based mechanism [17]. Priorities assigned to processes are inversely proportional to the length of the period. That is, the process with the shortest period is assigned the highest priority. Processes are executed in preemptive manner; at any time, the highest priority process with outstanding computation requirement is executed.

Amongst all the class of static priority scheduling schemes, it has been shown that rate monotonic priority assignment is optimal [17]. This implies that, if a given static priority scheduling algorithm can schedule a process system, the rate-monotonic algorithm is also able to schedule that process system. In the case of the rate-monotonic scheduling algorithm, optimality implies the imposition of constraints upon the process system and it includes the following characteristics:

- Fixedset of processes.
- All processes are periodic.

- All processes have deadline equal to period.
- One instance of a process must be complete before subsequent instances are run.
- All processes have known worst-case execution times.
- No synchronization is permitted between processes.
- All processes have an initial release at time 0.[71]

2.2.1.2 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

$$\sum U_i \leq 1$$

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 misbehavior during overload conditions.

2.3 Embedded System Different Architectures

2.3.1 System on Chip (SoC):

System on Chip is a new form of embedded system, which integrates microprocessor, analog IP cores, digital IP core and memory (or off-chip memory controller interface) on a single chip. It is usually customized (CSIC) or a standard product for particular purposes (ASSP). It successfully integrate the hardware integrated circuits and embedded software, which can implement the computer system functions, on a silicon chip, and it belongs to the emerging cross-disciplinary for computer and microelectronics Soc represents a new generation of technology development of embedded processor, its design technology provides an opportunity for computer professionals to be involved in IC design area. And SOC design is still hot now in application design and other areas like microelectronics technology. [67]

2.3.2 Multiprocessor System-On-Chip (MPSOC):

The multiprocessor system-on-chip uses multiple CPUs along with other hardware subsystems to implement a system. A wide range of MPSOC architectures have been developed over the past decade. This paper surveys the history of MPSOCs to argue that they represent an important and distinct category of computer architecture. We consider some of the technological trends that have driven the design of MPSOCs. We also survey computer-aided design problems relevant to the design of MPSOCs.[109]

2.3.3 NoC:

Network-on-chip is an emerging alternative that overcomes the above-mentioned bottlenecks for integrating a large number of cores on a single SOC. NOC is a specific flavor of interconnection networks where the cores communicate with each other using a router-based packet-switched network [89].

2.3.4 Comparison of Embedded System Architectures:

	PROS	CONS
SOC	-An SoC consumes less power. -A smaller size means it is lightweight and of small size. - the cost of an SoC is small due to advancements in VLSI technology. -As mentioned in the first point, cabling is not much required and so the cost of cabling is conserved. -An SoC provides greater design security at hardware and firmware levels. -An SoC provides faster execution due to high speed processor and memory.	-Initial cost of design and development is very high. - Even a single transistor or system damage may prove to be very costly as the complete board has to be replaced, and its servicing is very expensive. -Integrating all systems on single chip increases complexity. -It is not suitable for power-intensive applications.

MPSOC	-MPSOC provide enoumous computational capacity in an energy efficient and cost efficient way. -computational performance. -reduction of physical units. -heterogeneity.	-MPSOC architectures were not designed with storing focuson safety and certification -MPSOC have serious drawbacks and limitation with respect to domain - Lacking of temporal determinism. - Error propagation and non-intended interference.
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.	- 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. -it is mush more costly to produce than a bus-based system due to the large overhead of area and the power consumption of router

Table 1.1**:** Comparison Of Embedded System Architectures

3. NoC Main Components

3.1 NoC Layered Communication

Network-On-Chip uses a layered communication approach, similar to the OSI model which is at the foundation of LANs and WAN, but it is simpler and not all layers are included. Thus, in NoC, mostly three layers are considered (i.e. physical data link and network layer).

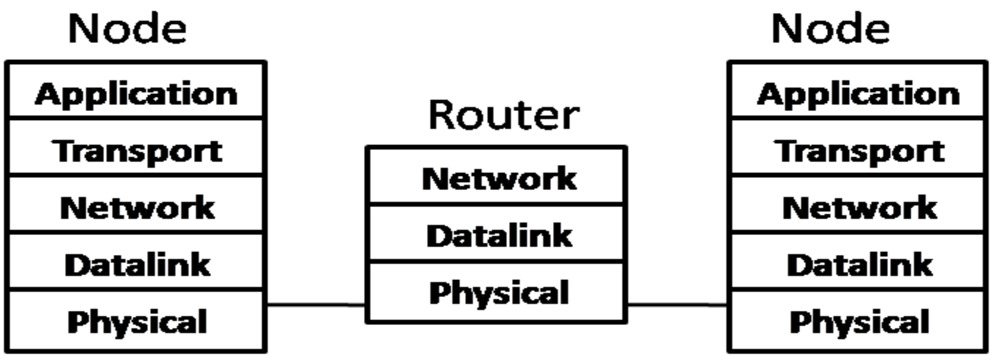


Figure 1.2: NoC Layered Communication

3.1.1 The Network Layer:

Switches are implemented in this layer. This layer hangs in charge also the topology of the network. Since addressing is strongly related to the topology, it is also treated in this layer. It defines the routing technique used for routing packets across the network and topology. It therefore routes packets from source to destination, resource addressing and packet buffering. It is also responsible for providing quality of service by addressing issues of latency, throughput, and jitter, etc. The Protocol Data Unit (PDU), which is the set of information exchanged between levels in the OSI system, used for communication between the entities of this layer is called a packet.[3]

3.1.2 The Data Link Layer:

It defines the bit exchange protocol on the links between the routers (eg, handle of hands, emission credit, etc.). It is therefore concerned with flitisation / package node to node communication, detection and correction of transmission error (bit of parity), flow control, schema encoding, etc. This layer ideally transmits data from one point to another, it could be omitted in a simulation. The omission is under the assumption that all data is transmitted without errors. Detection and Correction of the physical layer are implemented in the data link layer.[3]

3.1.3 The physical layer:

This is the lowest level layer of a NoC, it processes the actual data transfer. She is responsible for the clock signals for each connection, the number and nature of interconnection, control signals, electrical levels, the physical means by which will be transported the packets, as well as the way the data is transported (width of words carried in number of bits), etc. As this layer deals with electrical properties, simulation is best done in an analog simulator like Spice [81]. Most of them of the six layers described above, the smallest unit of time is a clock cycle. However, when simulating the physical layer, the time should be expressed in time physical. Co-simulation with physical time and clock cycles would be difficult to implement and expensive in calculation. [3]

3.2 NoC Main Components:

There are three main types of components, namely routers / switches, Nodes also known as Intellectual Property (IP), and Resource Network Interfaces:

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 such as: a general-purpose processor, a DSP (Digital Signal Processor), which is a microprocessor optimized for executing digital processing applications. the signal (filtering, signal extraction, etc.) as quickly as possible, a memory, a specific application hardware component, I / O controller, graphics controller, the mixed signal module, the unit radio frequency (RF) etc. Node should be implemented using the same technology as the network switch[12]. The designer can build own node or reuse commercial resources available from different component vendors.[3]

3.2.2 The Resource-Network Interface (RNI):

The RNI links a node to a network router. Therefore, it allows the node to send data to the router. The role of RNI in NOCs is the same as of the network card for the Internet [12]. The RNI consists of two parts, the dependent part resources and the independent portion of resources as shown in Figure 1.3. The part resource-independent is designed so that the RNI appears as another router connected to the router. The design of the independent part of the resources is common to all resources. If homogeneous PEs are used then the dependent part of resources can be reused and it will be the same for all resources, otherwise it is different for each resource.

The dependent part of the resources is responsible for the flitisation / deflitisation and the implementation of the encoding scheme. In the case of source routing, the RNT also contains routing tables and she is responsible for adding the full path in the header flit.[3]

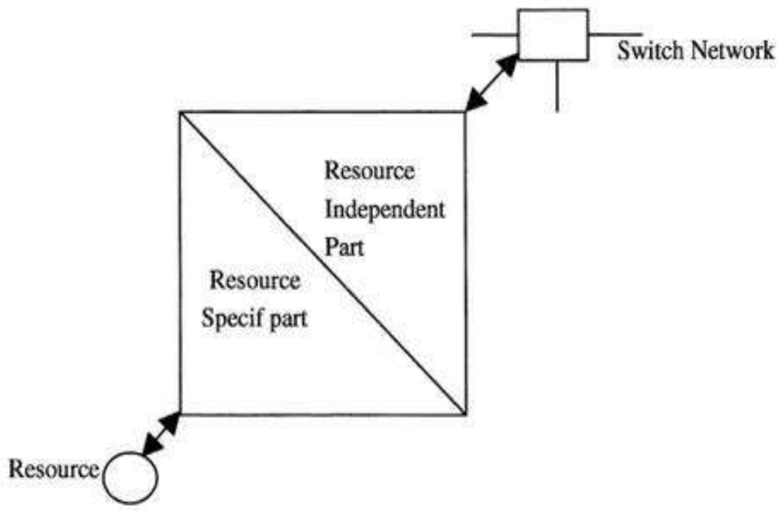


Figure 1.3 The Resource-Network Interface (RNI)[3]

3.2.3 Router:

Routers constitute the most important element in the "NoC" architecture. They allow the implementation of the routing strategy and the flow control algorithm. Additionally, the routers contain buffers to save coming packets. The routing algorithm implemented in the routers determinate destination of each entering packet. It usually implements simple minimal-path functions. [64]

Router also called switch. These components are in charge of forwarding data to the next tail. On the routers we can find implemented the routing protocol, buffer capabilities and the switching method. In general, the router component is composed of the next elements: Arbitrer, which its main task is to grant channels (selecting an input port and an output port) and route packets; Crossbar, of n input x n output ports that direct the input packet to the corresponding output port; and buffer or queue, if that is the case – as it is in the packet switching protocols – which is used to buffer incoming and outgoing data in the router. [28]

A NoC router is implemented using wormhole technique it consists of buffers, switches, and control units which are required to store and forward flits from the input ports to the desired output ports. The architecture is actually similar to that of modern routers, but with smaller area and buffer size .figure 1.4 shows a NoC 16 buffer slots per input port. The buffer slots are divided into four queues, and each queue is called a virtual channel (VC) [76]. There are four cardinal input ports and output ports connected from and to +x, -x, +y and- y directions. The last pair of input/output ports is connected from and to the processing element (PE). The four VCs are sandwiched between the demultiplexer connected to the input port, and the multiplexer connected to the crossbar. Each input unit can communicate with router, virtual-channel allocator, and switch allocator, which are responsible for Routing Computation (RC), Virtual-Channel Allocation (VA), and Switch Allocation (SA), respectively. The crossbar is controlled by the switch allocator for correctly connecting input ports to output ports. [20]

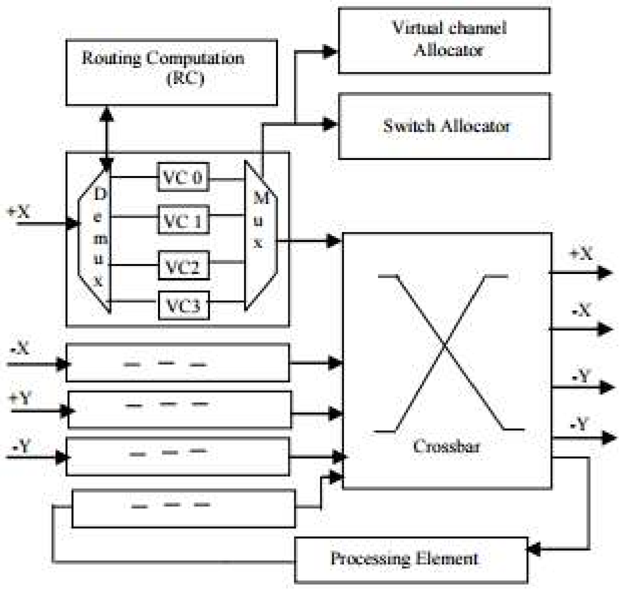


Figure 1.4 NoC Router[20]

3.2.4 The links:

Links are logical connections between two (or more) communicating elements [64] [76][6]. They allow point-to-point connections between IPs resources and RNI s network-resource interfaces, between RNI and routers and between routers. They are the ones that offer bandwidth and carry information between communicating elements.

A link can consist of several virtual channels, there are three types of links that are definedby the direction of the transmissions. They can be unidirectional in input, unidirectional inoutput or bidirectional (input / output).

In electronic component manufacturing technologies (CMOS: Complementary Metal Oxide Semiconductor), the channels are implanted as global signals of the circuit. Therefore, although reduced, the problems of delays and dispersion of Signals may also exist in NoCs. In order to solve these problems, the signals are commonly segmented by repeaters, usually buffers, which allow restore the level of the tension of the files. Since the queues have a significant length, repeaters can incorporate registers to pipeline the data transmission. So, these come at a rate that coincides with their use by the calculation elements connected to the network [14][3].

3.3 Flow control:

The control logic can be used for tasks such as error detection / correction or flow control. In order to ensure that the packets reaches its specified destination and without errors or loss it must control the flow. In the following, we will define the most affecting flow control in NoC.

3.3.1 The delay of the packets:

In order not to be forced to destroy conflicting packets, they can be delayed in queues. However, in the majority of NoCs, the surface consumed by a router comes from the place taken by these files. Therefore, a compromise must be found during the design of the router to minimize the area used by the queues without degrading the required performance.[1][3]

a. The size of queues:

The size includes the width of the queues in number of bits and their depth defining the number of words that can be stored. The first parameter that directly affects the size of the queues is the chosen switching mode [18][3]. Indeed, the files may contain several packets, a single packet or a part of the packet. Regarding the width of the queue, it can not be smaller than that of a flow control unit (flit). In addition, the size must be rigorously determined because it can affect the maximum clock frequency and the cost in terms of area and power consumption of the routers. Finally, an inappropriate size may imply congestions within the network with a reduction in bandwidth useful. Queues can be placed at different positions in the router: input, output, or virtual output [2][3].

b. The position of the queues:

We distinguish three type of queues, and they are defined as follows:

i. Input queues: in this first case, each input port of the router has a queue. Although this technique is the least expensive on the surface, it can induce saturation due to head-of-line blocking. This happens when data at the top of the queue can not access the associated output port, blocking other packets in the queue, even if their output is free.

ii. Output queues: When the queues are positioned at the output of the router, each output port has as many queues as there are ports of entry. This technique offers better performance than the previous one but the surface is necessarily more important.

iii. Queues in virtual output or virtual channels: The idea of queues in virtual output is to combine the advantages of both techniques preceding. Thus, each port of entry has several queues to distribute the packets according to their destination or priority. We are talking about virtual channels because for a single physical channel, there will be as many virtual channels as there are queues waiting [1] [52][3].

Although they involve an increase in surface area and latency, because of the memory and control, virtual channels have several advantages:

- they avoid deadlocks (blocked packets are stored in different channels virtual ones to let other packets pass);
- they optimize the use of links;
- they increase network performance (improved latency and bandwidth);
- they provide separate services (through separate traffic and priority levels different)[3]

3.3.2 The switching matrix:

The router's input and output ports are all connected to each other through the switching matrix [38]. It can be implemented by a crossbar or by cascading several stages of multiplexers because the matrix has the role of multiplexing the input data of the router to its output ports [1][3].

3.3.3 Arbitration:

When within a router, at least two packets wish to exit through the same port and this, at the same time, an arbitration must be done in order to choose the one that will come out first. Even though, there are different techniques for arbitrating conflicting packets, the most commonality is impartiality. Indeed, the arbitrator must be able to provide equitable access to output ports if the packets have the same priority [1][3].

The main arbitration policies that can be found in the NoCs are Access Time-Division Multiple Access (TDMA), reservation of Time-slot, Turnstile or Round-robin, and fixed priority [25][3].

i. By TDMA: it is a mode of multiplexing consisting of dividing the time into periods short called time-slots, and to be assigned to each element communicating one or more beaches. Thus, for the duration of a beach, the shared resource will be fully reserved for the element to which the time-slot is associated. However, the time range must be carefully aligned with the risk of introducing additional latencies [25][3].

ii. By time-slot reservation: the time-slot reservation policy is actually a TDMA type policy in which the reservation of the time range is repeated periodically.

iii. By tourniquet or round-robin: round-robin or round-robin type arbitration represents the most impartial arbitration policy. Indeed, at each cycle, it gives access to a packet from a different input.

iiii. By fixed priority: in the fixed priority technique, each packet has a priority fixed by the issuer before entering the network. This priority is used to route the packet from the source to the destination. [3]

3.4 NoC Main Characteristics:

Performance of NoC depends on various factors such as network topology, routing strategy and switching technique, and each one should be taken in account during the implementation of NoC.

3.4.1 Topology:

Topology generally refers to the way in which the IP Core, NI, Router and Physical Links are connected in the NoC. The topology of the NoC is defined according to the requirements of the application. For different applications different topologies are chosen. Different topologies have different properties, and are best suited for different applications. Some of the most commonly used topologies are described below. [90]

3.4.1.1 Mesh Topology

In mesh topology; all the nodes are connected in a grid pattern. Each node has 2 to 4 neighbors depending upon its position in the grid. The nodes at the edge have 2 neighbors but the nodes in the center have 4 neighbors. Adding of a new node in this topology is very easy. [90]

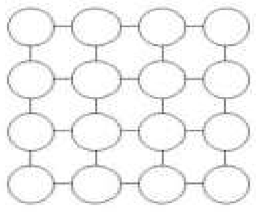


Figure1.5Mesh Topology

3.4.1.2 Torus Topology

The torus topology can be obtained by modifying the existing mesh architecture by adding direct connections to two end nodes in the same row or column. The torus topology may have long links. [90]

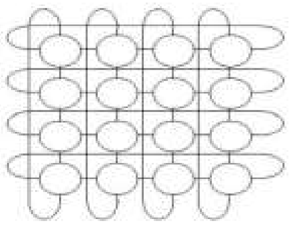


Figure1.6 Torus Topology[90]

3.4.1.3 Ring Topology

In ring topology, all the nodes are connected in a ring pattern. Regardless of the size of the network, every node has only two neighbors. The main advantage of this topology is that if faults occur they can be easily detected; but if any link breaks the entire network can be disrupted this is a major problem. [90]

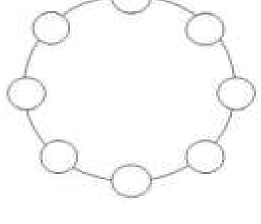


Figure1.7 Ring topology[90]

3.4.1.4 Star Topology

In star topology, all the nodes are connected to a single central node. All the nodes except the central node has only one neighbor, where as the central node has (N-1) neighbors. This topology is very simple, and the average hop distance is 2. However, the failure of the central node results in the failure of the entire system. [90]

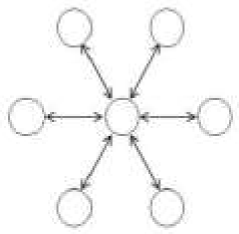


Figure1.8 Star Topology[90]

3.4.1.5 Binary Tree Topology:

In binary tree topology, the top node is called the root and the bottom node is called the leaves. Every node has two off springs. As the tree becomes big, it becomes complicated to configure.

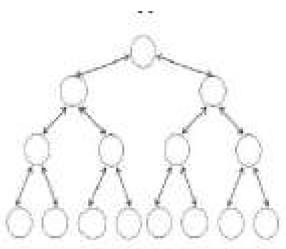


Figure1.9 Binary tree topology[90]

3.4.1.6 Fat Binary Tree Topology

In fat binary tree topology, only the leaves are the IP core. There are more number of links increase by the order of 2. This topology is best suited for applications which require high degree parallelism. [90]

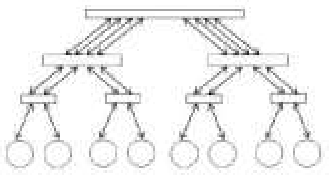


Figure1.10 Fat binary tree topology[90]

3.4.1.7 Butterfly topology:

In butterfly topology, there is only one path from source to destination node. There are long physical links which are a major drawback. [90]

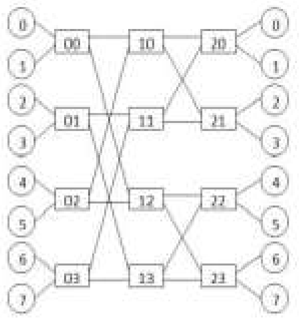


Figure1.11 Butterfly topology[90]

3.4.1.8 SPIN topology:

SPIN (Scalable Programmable Integrated Network)topology is similar to the butterfly architecture. In this topology, the router in each level consist of same number of parent port and child port. [90]

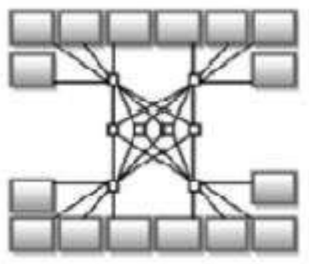


Figure1.12 Spin topology[90]

3.4.2 Switching Protocol:

3.4.2.1 Circuit Switching:

In circuit switching[44], the physical path from source to destination is reserved for the entire duration of data transmission. This is realized by injecting the header flit14 into the network. The header flit contains the destination address and some additional control information. It moves to ward 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. The reserved path may then be released by the destination or by the last bits of message itself.

This technique is advantageous when the messages are infrequent and long. In which, 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.[59]

3.4.2.2 Packet Switching:

This technique is also called Store-and-Forward (SAF) [44] technique, where the messages divided into fixed-length blocks, called packets. Unlike the circuit switching technique, which sets the path before sending a data, each packet is routed individually from source to destination. 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.

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 at the individual router scan become extensive if packet size becomes large and multiple packets must be buffered at a node.[59]

a) Store and Forward (SAF):

Store-and-forward (SAF) switching interprets the header information at each intermediate router and uses the information to determine the output link through which the packet is to be forwarded. A packet transfer is initiated only when the receiving router has sufficient buffer space for the entire packet, which requires the buffer size to be at least the size of a packet. A stalled packet occupies the local node itself [83].

Store-and-forward is the simplest routing mode where packets move in one piece, and entire packet has to be stored in the router’s memory before it can be forwarded to the next router. So, the buffer memory has to be as large as the largest packet in the network. The latency is the combined time of receiving a packet and sending it ahead. Sending cannot be started before the whole packet is received and stored in the router’s memory[105].

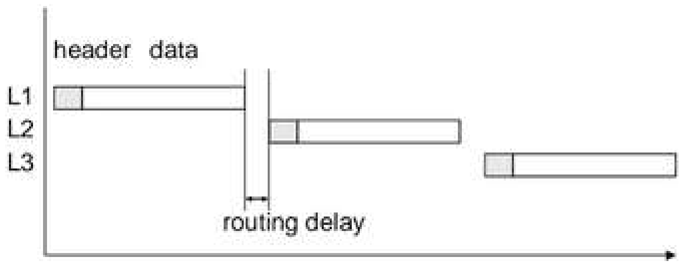


Figure1.13: store and forward.[105]

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 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) [44]. 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[2].

Virtual cut-through is an improved version of store-and-forward mode. A router can begin to send packet to the next router as soon as the next router gives a permission. Packet is stored in the router until the forwarding begins. Forwarding can be started before the whole packet is received and stored to router. The mode needs as much buffer memory as store and-forward mode, but latencies are lower.[105]

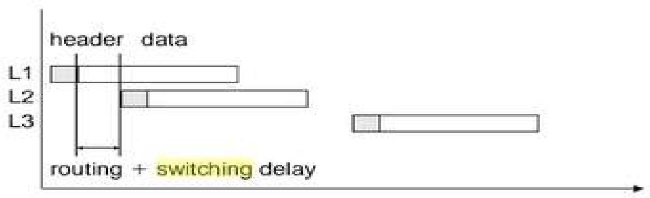


Figure1.14: virtual cut-through (VCT) switching.[105]

c) Wormhole Switching:

In wormhole switching [44] 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 flit level in 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 at 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.[44]

In wormhole routing packets are divided to small and equal sized flits (flow control digit or flow control unit). A first flit of a packet is routed similarly as packets in the virtual cut-through routing. After first flit the route is reserved to route the remaining flits of the packet. This route is called wormhole. Wormhole mode requires less memory than the two other modes because only one flit has to be stored at once. Also the latency is smaller and a risk of deadlock is larger. The risk can be reduced by multiplexing several virtual ports to one physical port, so the possibility of traffic congestion and blocking decreases. [26][105]

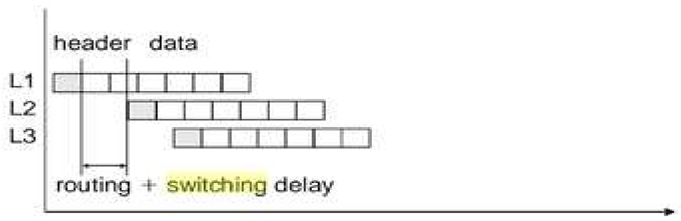


Figure1.15:Wormhole switching.[105]

3.4.3 Routing Protocol:

Before defining the routing protocol, two concepts should be defined; deadlock and live lock that have a huge impact on the routing protocol improvements.

i. 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 other 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.[59]

Routing is in deadlock when two packets are waiting each other to be routed forward. Both of the packets reserve some resources and both are waiting each other to release the resources. Routers do not release the resources before they get the new resources and so the routing is locked.[105]

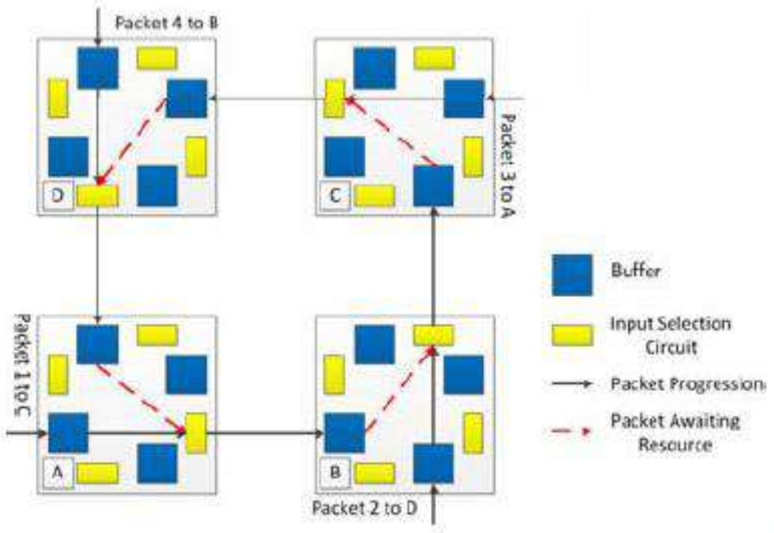


Figure1.16: Deadlock.[59]

ii. Live lock:

Live lock usually happens in adaptive routing schemes. It happens when a packet is 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 for an algorithm that can cause a live lock. 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. [59]

Live lock occurs when a packet keeps spinning around its destination without ever reaching it. This problem exists in non-minimal routing algorithms. Live lock should be cut out to guarantee packet’s throughput. There are a couple of resorts to avoid the live lock. Time to live (TTL) counter counts how long a packet has travelled in the network. When the counter reaches some predetermined value, the packet will be removed from the network. The another resort is to give packets a priority which is based on packet’s age. The oldest packet always finally get the highest priority and will be routed forward. [107][105]

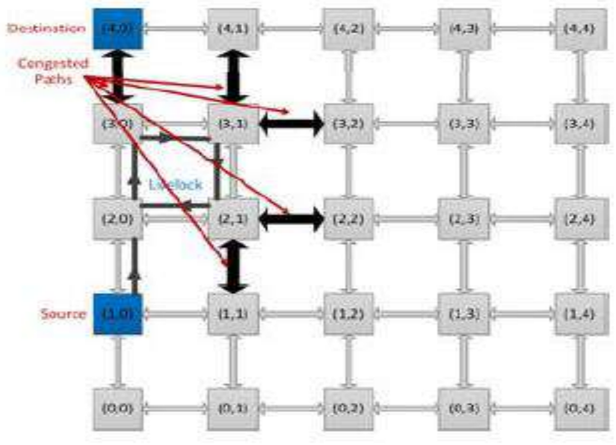


Figure1.17: Livelock.[59]

3.4.3.1 Deterministic Routing Algorithms

Deterministic routing algorithms route packets every time from a certain point A to a certain point B along a fixed path. Deterministic algorithms are used in both regular and irregular networks. In congestion free networks deterministic algorithms are reliable and have low latency. They suit well on real time systems because packets always reach the destination in correct order and so a reordering is not necessary. In the simplest case each router has a routing table that includes routes to all other routers in the network. When network structure changes, every router has to be updated.

a) Shortest Path Routing:

A shortest path routing is the simplest deterministic routing algorithm. Packets are always routed along the shortest possible path. A distance vector routing and a link state routing are shortest path routing algorithms.

i. Distance Vector Routing:

Each router has a routing table that contains information about neighbor routers and all recipients. Routers exchange routing table information with each other and this way keep their own tables up to date. Routers route packets by counting the shortes path on the grounds of their routing tables and then send packets forward. Distance vector routing is a simple method because each router does not have to know the structure of the whole network.

ii. Link State Routing:

Link state routing is a modification of distance vector routing. The basic idea is the same as in distance vector routing, but in link state routing each router shares its routing table with every other router in the network. Link state routing in Network on Chip systems is a little bit customized version of the traditional one. The routing tables covering the whole network are stored in router's memory already during the production stage. Routers use their routing table updating mechanisms only if there are remarkable changes in the network's structure or if some faults appear. [104,105]

b) Source Routing :

In a source routing a sender makes all decisions about a routing path of a packet. The whole route is stored in a header of packet before sending, and routers along the path do the routing just like the sender has determined it.

A vector routing works basically like the source routing. In the vector routing the routing path is represented as a chain of unit vectors. Each unit vector corresponds to one hop between two routers. Routing paths do not have to be the shortest possible.

Arbitration look ahead scheme (ALOAS) is a faster version of source routing. The information of routing path has been supplied to routers along the path before the packets are even sent. Route information moves along a special channel that is reserved only for this purpose. [105]

A contention-free routing is a algorithm based on routing tables and time division multiplexing (TDM). Each router has a routing table that involves correct output ports and time slots to every potential sender–receiver pairs. Contention free routing algorithm is used in Philips Ethereal NoC system and it is also called as a clockwork routing.[105]

c) Destination-tag Routing:

A destination-tag routing is a bit like an inversed version of the source routing. The sender stores the address of the receiver, also known as a destination-tag, to the header of the packet in the beginning of the routing. Every router makes a routing decisions independently on the grounds of the address of the receiver. The destination-tag routing is also know as a floating vector routing.[105]

d) Topology Adaptive Routing:

Deterministic routing algorithms can be improved by adding some adaptive features to them. A topology adaptive routing algorithm is slightly adaptive. The algorithm works like a basic deterministic algorithm but it has one feature which makes it suitable to dynamic networks. Systems administrator can update the routing tables of the routers if necessary. A corresponding algorithm is also know as an online oblivious routing. The cost and latency of the topology adaptive routing algorithm are near to costs and latencies of basic deterministic algorithms. A facility of topology adaptiveness is its suitability to irregular and dynamic networks.[105]

3.4.3.2 Adaptive Routing Algorithms:

Adaptive routing algorithms do not restrict a message to a single path when traveling 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 back track, releasing previously reserved channels, thus systematically searching for appropriate path. For deterministic algorithms backtracking technique is useless asthe message will go by the same path again. Adaptive routing algorithms increase routing flexibility but the hardware becomes more complex and slower.[59]

3.4.3.3 Minimal Adaptive Routing:

Minimal routing algorithms always use the shortest path in order to reach destination. While being adaptive they restrict the routing direction in some level. One of the examples for 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 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 sub network is used. If dx = sx then either sub-network can be used.

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 appropriate manner [21]. The packet that is sent to the destination should path 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 (nis the number of dimensions), due to the additional channel requirement.[59]

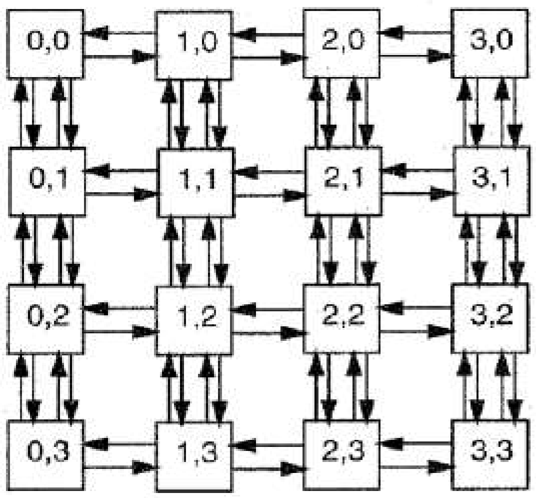


Figure1.18: Double Y-channel 2D mesh.[19]

3.4.3.4 Non-Minimal Adaptive Routing:

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

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. [106]. 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 it h sub-network consists of all the it h 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 creaches 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.[44]

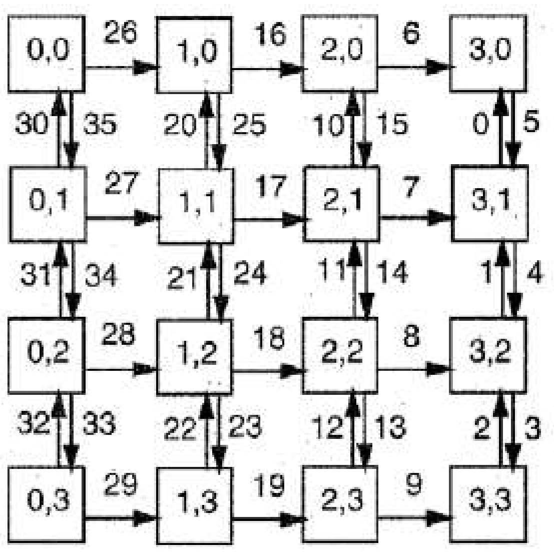


Figure 1.19: +X sub-network and labeling.[19]

3.4.3.5 Turn Model

This algorithm was proposed by Glass and Ni [15] 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 traveling through the network, switches from one dimension to another, in order to reach the destination. Switching from one dimension to another is called 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[59]. Turn model algorithms determine a turn or turns which are not allowed while routing packets through a network. Turn models are live lock-free.

a) West-first Routing:

A west-first routing algorithm prevents all turns to west. So the packets going to west must be first transmitted as far to west as necessary. Routing packets to west is not possible later.

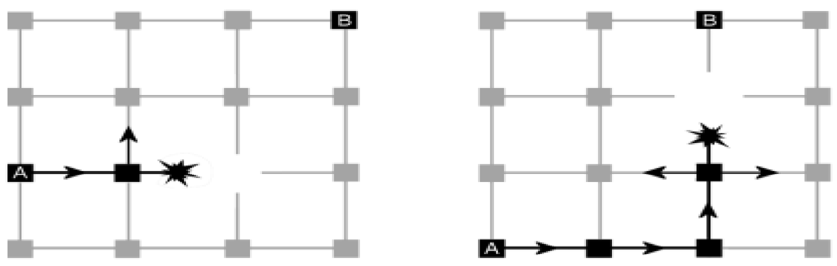


Figure 1.20:West-first Routing.[105]

b) North-last Routing:

Turns away from north are not possible in a north-last routing algorithm. Thus the packets which need to be routed to north, must be transferred there at last.

c) Negative-first Routing:

Negative-first routing algorithm allows all other turns except turns from positive direction to negative direction. Packet routings to negative directions must be done before anything else. [105]

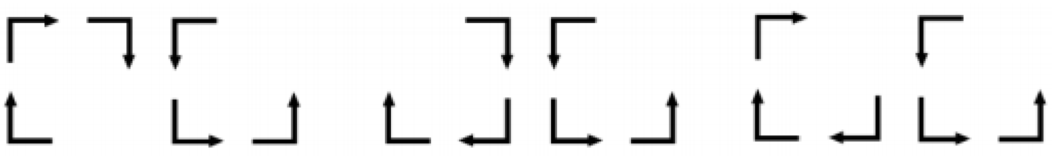


Figure1.21: Negative-first Routing.[105]

3.4.3.6 Randomized Routing Algorithms

DOR algorithm is considered to be one of the most popular deterministic routing algorithms due to its simplicity for implementation and good performance according to average packet delay and throughput metrics [96].

Valiant and Brebner [53] 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.

Another routing algorithm which uses randomization is Randomized, Oblivious, Multi-phase, Minimal (ROMM) [26] 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 same source and destination will not take the same path.

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. Ap-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 minimal path from source to destination.[59]

3.4.3.7 Comparison of Routing Algorithms:

Deterministic algorithms in comparison to adaptive ones achieve higher throughput behavior under uniform traffic pattern in 2D mesh with the same number of VCs and have lower latencies at higher throughputs. However, adaptive algorithms outperform them when used in 2D torus [21]. Deterministic algorithms suffer from channel underutilization 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.

DOR outperforms O1TURN under uniform traffic when the network size is small (4x4). It achieves lower latency and higher throughput behavior. 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[59].

4. Conclusion

Today, SoCs on-chip systems have become increasingly complex with the evolution of integrated circuit technology. The performance of these SoCs is highly dependent on the on-chip interconnection architecture and the communication protocol between the IPs cores. The amount of data exchanged in the systems is thus increased. In this context, the NoC paradigm has become the preferred solution for communication in complex SoCs. However, lack of methodologies and tools adapted experimentation and design assistance, the performance evaluation of these networks on a chip is a major challenge for the industrialization of these systems.[3]

The network on chip contains three are main types of components, namely routers / switches, resources also known as Intellectual Property (IP), and Resource Network Interfaces[10].The routing algorithm is an important factor in NoC designs because it establishes the path of the messages between cores. We can group routing algorithms, based on their path decision function, as deterministic or adaptive. [59]

CHAPTER 2

ENERGYEFFICIENCY AND DELAY MODEL

1. Introduction

As renewed attention has been given by policy makers to energy conservation issues, it has frequently been asserted that an energy-efficiency gap exists between actual and optimal energy use. In this chapter we will talk about energy-efficiency and delay model.

2. Synchronous data flow

2.1 Definition

Synchronous data flow (SDF) is a special case of data flow hardware and software methodology popular among computer scientists for parallel computation. Under the data flow paradigm, algorithms are described as directed graphs where the nodes represent computations (or functions) and the arcs represent data paths. [29]

SDF specification is specialization of the dataflow modeling paradigm. In SDF, parallelism is represented explicitly, which makes such specification convenient for HW/SW co synthesis.

Restriction imposed on the dataflow model refers to the fact that number of tokens produced or consumed by the specific actor needs to be predetermined and fixed. Therefore, the amount of data flowing through the system is predetermined and cannot be changed depending on the elapsed time or received token values. In a way, actor could be observed as a function defined in some programming language, where input tokens correspond to the arguments of a function, and output tokens to the return values of the function. This makes dataflow graph an efficient representation of the function call graph, e.g. JPEG encoder application due to its modular nature could be easily represented using SDF.

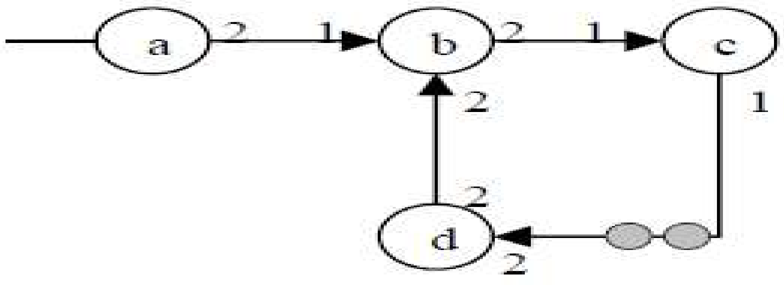


Figure2.1 Synchronous Data Flow example[34]

Such restricted behavior can be represented using balance equations, which enable to determine relative execution rates of particular actors. Balance equations represent a linear association over a specific arc between two actors, i.e. number of the tokens produced by the source actor multiplied by its execution rate has to be equal to the number of tokens required by the destination actor multiplied by its corresponding execution rate. If a system of such balance equations is solvable, given SDF could be statically scheduled, enabling for a system level performance estimation. Consequently, SDF is decidable, i.e. required buffer sizes and deadlock condition can be detected statically. In general, algorithms expressed as SDF graphs can always be converted into an implementation that is guaranteed to take finite-time to complete all tasks and use finite memory. However, choice of scheduling can still influence overall memory requirements. E.g. single-appearance schedule can lead to reduction in code size at the expense of buffer requirements [34].

2.2 Model structure

We worked on three types of graphs: random and Benchmarks multimedia applications (automotive / industrial task graph and multimedia system task graph):

The graph consists of tasks and lien: each task is defined by name, clock cycle, priority, token and task before, task before 2, task after, task after 2 where:

name: Name that defines a task.

Clock cycle: The time that Execute the task in the node, where 1cc = 2 second.

priority: Is the priority in the Execute task, in the graph the priority of the largest is at the top of the graph and lower in the bottom, where the execution of tasks begins as a priority.

token: It is a token that receives and sends tasks between them. So that the Execute task does not start until it arrives token.

task before and task before 2: these are tasks that have a higher priority than an existing task, and which are sent to them token.

task after, task after 2: these are tasks that have a lower priority than an existing task, and which they receive token.

Line is the way through which token.

The architecture consists of 5×5 processor (node) and link:

each node is defined by name, voltage where:

name: that defines a task.

voltage: is the voltage it takes for a node in execution tasks.

I have a node table where there are tasks that will be executed in node.

link a road through which the massage passes where each link has a bandwidth.

2.3 Scheduling policies over SDF

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 models of computation can be scheduled statically. Static scheduling imposes pre runtime deadlock detection and buffer overflow detection. This is a significant advantage in comparison to KPN model, where employed scheduling policy can affect the memory requirement properties.

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. [102]

Operate multimedia applications, for instance an MP3 encoder on streams of data. These applications can be described by an application task graph in which the tasks are periodically executed. The period is determined by the throughput requirements of the application. Whenever a task executes, it exchanges messages with other tasks via data) streams. Fig.2.2 shows an example of a simple application task graph consisting of two tasks t1 and t2.

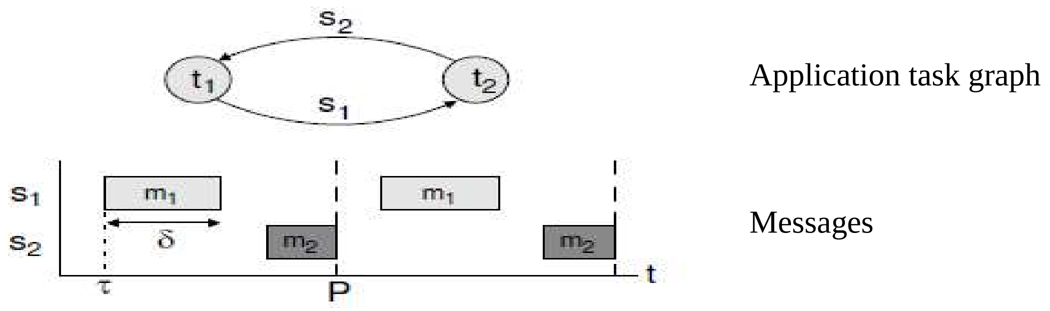


Figure2.2 Example application task graph[88]

Within each period P, task t1 sends a message m1 through stream s1 to task t2 and t2 sends a message m2 through s2 to t1. Note that the periodic execution of the tasks results in a periodic behavior of the communication between them, with potentially some data-dependent jitter. To meet the computational requirements of modern multimedia applications, multi-processor systems are used. The tasks, from an application graph, are mapped to the various processors in the system. Whenever multiple tasks are mapped to one processor, the execution order of these tasks is fixed through a schedule. These schedules and the timing constraints imposed on the application determine time bounds within which each task must be executed. Similarly, they determine time bounds within which messages must be communicated between the tasks [88].

2.3.1 Scheduling Strategies:

a. Overview

Given a set of messages M, a scheduling strategy must find a schedule entity e for each message m ϵ M and the set of scheduling entities E must form a feasible schedule function.

b. Greedy:

The greedy strategy explores a small part of the solutions pace. As a result, it has a small run-time. However, it may miss solutions or find non-optimal ones in terms of resource usage. The greedy strategy essentially tries to schedule the largest, most time-constrained messages first, via the shortest, least congested route that is available.

c. Rip up

The rip up strategy uses the greedy strategy described in the previous section to schedule all messages. This guarantees that all problems that are feasible for the greedy strategy are also solved in this strategy. Moreover, the same schedule function is found as soon as a conflict occurs.

d. Global knowledge

The rip up scheduler does not know a priori which unscheduled messages need to use links in the route it assigns to the message it is scheduling. It can only use local information to avoid congestion. The global knowledge strategy tries to estimate, before scheduling messages, the number of slots that are needed in each of the links. This gives the scheduling strategy global knowledge on the congestion of links. This knowledge is used to guide the route selection process when scheduling the messages.

e. Cost function

Cost function are used in the scheduling strategies to sort the messages M and routes R. The cost functions should minimize the chance of having a conflict when scheduling messages. They are constructed in such a way that the most resource constrained messages are handled first and that the resource usage is balanced over all links in the NoC. However, by doing so, they up-front exclude points from the solution-space. To circumvent this problem, randomly ordered sets M and R can be used as an alternative for the cost functions. [88]

3. Network on chip delay model

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:

$$NoC_{lat} = \sum_{i=1}^{T}\left(Ex_{ci}^{Pj} + Wt_{ci}^{pj}\right) + \sum_{j=1}^{C}\left((n-1) \times \left(wl_j + Com_j\right) + N \times (wl_j + R_j)\right)$$

Wt_{ci}^{pi} = The wait time of task ci for the processor p_i.

Wl_j = The wait time of packet j for the link .

Com_j =The communication time , or the transmission time of packet j.

R_j =The time it takes a router for transmission from the input port to the output port .

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:

$$Ex_{Ci}^{Pi} = \frac{W(i)}{f}$$

$W(i)$ = The workload of task i.

$f = The\ operating\ frequency.$

3.3 The communication delay

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

$$Com_j = \frac{N_{filt}}{bw}$$

N_{flit} =The number of flits or the size of the packet.

bw = the bandwidth of the network.

4. Network on chip power model

The classification of the optimization technique is done based on the type of dissipated power. Due to that, it is necessary to define the network on chip power model. The concept of energy, 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. The power model used to estimate the total power dissipated by a NoC (note that we consider the type of the transmitted flow as a multimedia application or a number of random tasks or benchmarks) is given by the formula:

$$E_{Noc} = N_{task} \times E_{Ci}^{Pj} + N_{channel} \times E_{ei,j}$$

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} , $E_{Ci}{}^{Pj}$ and E_t 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_t is defined as the number of cycles it takes to execute ci on processor P_j multiplied by the energy of one computational interval and calculated as:

$$E_t = r_j^i \times F_p \times E_{cip}^{Pj}$$

r_j is the execution time of the task ci on processor P , $f_{p}$is the operating frequency. $E_{Cip}{}^{Pj}$ is energy of one computational interval equals to (with σp = 0.2)

$$E_{cip}^{pi} = V_{\sigma d}^2 \left(1 + \sigma_p\right) \frac{a_p}{2} C_p$$

The energy consumed due to sending the message (ci,j):

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

$$E_{flit} = n_r + X_{Erouter} + (n_r - 1) \times E_{link}$$

$E_{ei,j}$ 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, $E_{Erouter}$, 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

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 as it is illustrated in Figure2.3

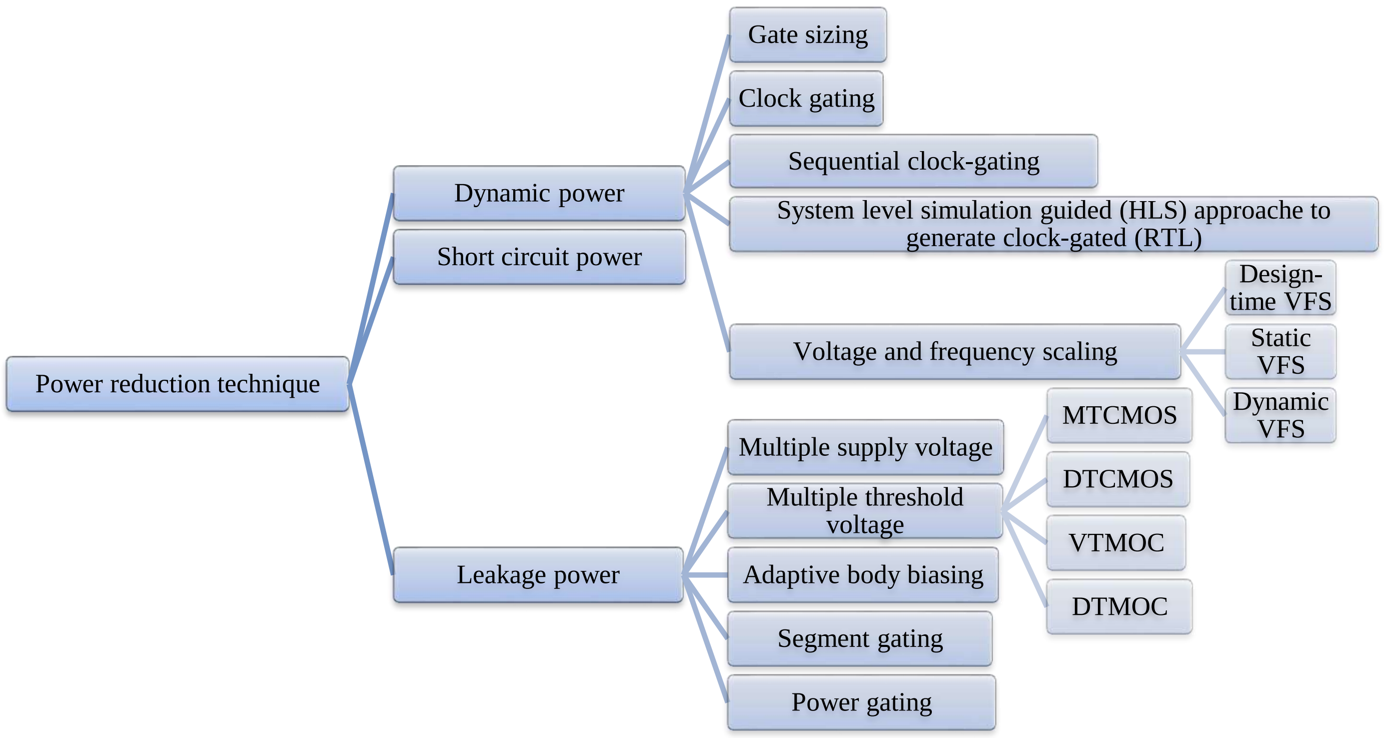


Figure 2.3 Power reduction techniques classification

5.1 Techniques for reducing dynamic power

The equation $P=\frac{1}{2}aC_l V^2 f$ represents the dynamic power of a circuit in which all the transistors switch once per clock cycle. Many researchers minimize or reduce the sys-tem 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 the output capacitance of the gate itself (due to parasitic.), the wire capacitance, the input capacitance of the gates in its fan out and 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 $AT(gj$,v) be the arrival time of the gate gi and $RT(gj$,v) be the required time of gate gi under a given delay constraint. The time slack of gate with respect to the input vector is given by:

$$\mathrm{slack}\ (\mathrm{gj}, \mathrm{v}) = \mathrm{RT}(\mathrm{gj}, \mathrm{v}) - \mathrm{AT}(\mathrm{gj}, \mathrm{v})$$

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

In [92], the author develops an algorithm to minimize total power in a dual-$V_{dd}$and dual-V_{th} design. First: an upsizing to create slack and maximize low V_{dd}assignments in a backward topological manner. Then, a forward topological fashion and both sizes and reassigns gates to high$V_{dd}$to enable significant static power savings through high V_{th} assignment. Simulation results show that power savings is about (50%) compared to the conventional scheme.

5.1.2 Clock gating

To reduce the clock power dissipation in synchronous circuits or in the adders, clock gating is the mostly used. Due to that, 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. As a solution, clock to an idle portion disabled by using an enable function, which is the input of the EN port of clock gate circuitry. Thus avoiding power dissipation due to unnecessary charging and discharging of the unused circuit

In [6], an algorithm for clock gating that 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.

In [79], the author presented a 4-bit synchronous counter, which use. the clock gating to improve the total dynamic power consumption. Simulations performed with and without using gated clock on Xilinx ISE design tool. As a result, the dynamic power of scheme with the gated clock is about 8mW whereas without the gated clock is about 9mW, and approximately 11% of save dynamic power.

In [62], the author aim to reduce the power consumption in modern processors and system-on-chip by presenting a novel Linear feedback shift register with look ahead clock gating technique where each flip-flop can be grouped so that they appears like sharing a common clock enabling signal. Simulation results show that power savings is about 15.03% compared to the conventional scheme and about 44.87% compared to data-driven clock gating.

5.1.3 Sequential clock-gating

In traditional clock-gating reduction technique, clock-toggles and inconsequential computation of the registers could occur. For that a sequential clock-gating [5] aim at eliminating that problem providing power saving control information across clock boundaries. Sequential clock gating is the process of extracting/propagating the enable conditions to the upstream/downstream sequential elements, so that additional registers can be clock gated. The results show up that the power saving is about 30% compared to the traditional (combinational) clock gating based power reduction techniques.

In [113], both verification and synthesis of clock-gated circuits were proposed for sequential circuits, based on the fact that sequential clock-gating synthesis only inserts sequential redundancies to target circuits. Experimental results showed that the pro-posed methodologies could effectively and efficiently verify the proposed clock gating conditions or to synthesize legal clock-gating conditions to reduce the frequencies of updating flip glops.

5.1.4 System level simulation guided (HLS) 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. 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, those methods 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.

In [7], the author extended CoDeL to insert clock-gating automatically at the be-havioral level to reduce the dynamic power dissipation in the resulting architecture while allowing hardware description at the algorithm level. A simulation based power analysis on the designed circuits shows that CoDeL’s clock gating performs better than Synopsys’ automated clock gating. CoDeL reduces the power dissipation by 83% on average, while Synopsys gives 81% savings.

In addition, the author in [14] proposed a novel system-level design methodology, which utilized a relative power reduction model' that can help in predicting the impact of clock-gating on each register quickly and accurately, by simulating the design at a cycle accurate transaction-level.

5.1.5 Voltage and frequency scaling

Dynamic power is proportional to the square of the operating voltage $P≅V2$, 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 lowered, and thereby a cubic power reduction is possible. An example [45] shows 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:

$$\tau = \mathrm{kCl}\frac{\mathrm{Vdd}}{(\mathrm{Vdd} - \mathrm{Vth})^2}$$

Where τ is the circuit delay, k is the gain factor, $Cl$is the load capacitance, Vdd is the supply voltage, and Vth 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 were 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 using 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 Vdd. However, smaller threshold voltages lead to smaller noise margins and increased leakage current. Furthermore, this cubic relationship holds only for a limited range of Vth scaling. The quadratic relationship between energy and Vdd deviates as Vdd was scaling down into the sub-threshold voltage level. In the sub-threshold region, while the dynamic power still reduces quadratic ally 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 efficient. In practice, extending the voltage range under half Vdd is effective, but extending this range to sub-threshold operations may not be beneficial. In the follow, we will discuss the main type of voltage and frequency scaling with some state of art of each class.

a. Design-time voltage and frequency scaling

One of the most common ways to reduce power consumption using voltage and frequency scaling technique is during design time where circuits are designed to exceed the performance requirements. Then, the supply voltage will reduced in purpose to meet the performances constraints of the system. This definition is 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 volt-ages have to be satisfied.

In [47], the author proposed real-time static voltage and frequency scaling (RT-SVFS) techniques that based on an optimal real-time scheduling algorithm for multi-processors. The techniques are theoretically optimal when the voltage and frequency can be controlled both uniformly and independently among processors. Simulation re-sults show that the independent RT-SVFS technique closely approaches the lower bound on energy consumption if the voltage and frequency can be controlled minutely.

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 [74], the author 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.

In [60], the author proposed a phase aware adaptive hardware selection technique, featuring dat prefetchers and dynamic voltage and frequency scaling. This technique exploits the fact that the memory bound code can be power optimized for achieving energy saving. Simulation results are while maintaining or exceeding the performance of a nun optimized system and it’s show that the power reduction is about 39% and energy reduction is about 37%.

In [11], the author applied a technique that reduces both dynamic and static power consumptions using adaptive design techniques applied locally for each synchronous NoC units while proposing a fully power-aware locally-synchronous and globally-asynchronous NoC circuit for the implementation of DVFS mechanism. The circuit is arranged around an asynchronous network-on-chip providing scalable communication and a 17 Gb/s throughput while automatically reducing its power consumption by activity detection.

In [111], 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%.

In [41], the author 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.

In [108], the author 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 with-out 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 volt-age-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.

In [114], the author 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 proportional to rise time and fall time on gates that is consumed by each transition, which increases with rise and fall times of input and decreases for larger output load capacitance. Therefore, reduction requires that gate output transition should not be slower than the input transition because faster gates can consume more short-circuit power, for that scaling down of supply voltage with respect to threshold voltages reduces short-circuit power.

An important point to note is that if the supply is lowered to below the sum of the thresholds of the transistors $Vdd<VTn+|VTp|$, the short circuit currents can be eliminated because both devices

will never be on at the same time for any input voltage value. Refer to [103], the effects of scaling down of supply voltage results about 1-16% of short-circuit power at 0.7 micron, 4-37% at 0.35 micron and 12-60% at 0.17 micron.

In [68], the author demonstrate the dependency of short-circuit power on threshold voltage while proposing a technique that allows significant reduction of short-circuit power without the need for process shift. The simulation results show good correlation with the analytical estimation at cell level, while demonstrating an average short-circuit power saving of 36%.

In [98], the author proposed a fine-grained power gating technique for an asynchronous-logic pipeline stage using locally controlled gating transistors. As a result dual gating achieves the best wasted power reduction of 86% for short-circuit power and 99% for leakage power at 10 Mbps input rate.

5.3 Techniques for reducing leakage power

At circuit level, the threshold voltage can be reduced by increasing the potential of the channel for the same gate-source voltage. As the channel potential is the result of the gate, source, drain and bulk/body (back-gate) potential, playing with the latter three can effectively alter the threshold voltage.

Apart from the gate, the terminal that has the largest effect on the channel potential is the bulk, this is particularly true for transistors with a relatively long channel and/or a thicker gate oxide. It is for this reason that you will find quite some examples of circuits that employ bulk/body biasing to lower the threshold voltage of circuits in the open literature. Increasing the bulk-source voltage (assuming an NMOS device) effectively reduces the needed gate-source voltage to accomplish the same drain current. Be aware though that increasing the bulk-source voltage will forward bias the bulk-source PN junction and thus may require additional current. In the follow some of techniques to reduce leakage power are discussed.

5.3.1 Multiple supply voltage

In [18], the author presented a dynamic programming approach for assigning voltage levels to the modules in non-pipelining and functionally pipelined data-paths. One can reduce the average power consumption by using a single lowered supply voltage. With only a single lower supply voltage, if the computation time constraint is violated, then one has to use pipelining or parallelism on whole or part of the circuit to recover performance. Experimental results show that using four supply voltage levels on a number of standard benchmarks, an average energy saving of 53% (with a computation time constraint of 1.5 times the critical path delay) can be obtained compared to using one fixed supply voltage level.

In [63], the author presented a study of different power metrics for varying micro-architectural configurations and a promising scheme to reduce the energy requirements of superscalar, out-of-order processors. The variation of energy (or energy delay product) per committed instruction with different architectural settings provides an insight into finding energy optimal settings for a given application. Such a configuration could be found in a run-time environment either for a given application, or on a finer grain, or a given computational kernel belonging to an application. The multiple voltage supply scheme is a simple and efficient way of reducing the energy requirements by up to 27% (for the given set of voltage values) of a processor, with no performance overhead. Since a completely synchronous scheme requires the existence of a global clock with high power overhead, to achieve more significant savings the multiple voltage supply solution could be applied in a globally asynchronous, locally synchronous architecture where the overhead of communication among different modules is minimized. Nonetheless, the paradigm of running slower stages at a lower voltage could be employed in a run-time environment that is able to adjust the voltage and clock frequency dynamically, on a fine grain, to fit the application needs.

In [19], the author proposed a technique where the power switches possess the feature of flexible programming after chip manufacturing. This novel technique prove by the use of a video decoder test chip, which shows 55% and 61% power reductions com-pared to conventional single-$Vdd$and low-voltage designs, respectively. This power-aware performance adjusting mechanism shows great power reduction with a good power-performance management mechanism.

In [42], 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-Vdd) applied to the circuit on non-critical paths, while the original voltage (high-Vdd) 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 typically designed using high-Vddgates at first. If the propagation delay of a circuit path is less than the required clock period, the gates in the path gave low-Vdd. In an experimental setting, the dual Vdd system applied on a media processor chip providing MPEG2 decoding and real-time MPEG1 encoding. By setting high-Vddat 3.3 V and low-Vddat 1.9 V, system power reduction of 47% in one of the modules and 69% in the clock distribution obtained.

5.3.2 Multiple threshold voltage

To reduce the leakage power on a single chip, multiple threshold CMOS technologies are used while providing both high and low threshold transistors. The high threshold (high-Vth) transistors suppress the sub threshold leakage current, while low threshold (low-Vth) transistors provide high performance. Based on the multiple threshold technologies, several multiple threshold circuit design techniques have been developed:

a. Multi-Threshold Voltage CMOS (MTCMOS)

MTCMOS is a most common technique. In this technique, the leakage power is reduced by inserting high threshold devices in series to low-Vth circuitry.

In [86], 1-V power supply high-speed low-power digital circuit technology with 0.5-/spl mu/m multi threshold-voltage CMOS (MTCMOS) is proposed. This technology features both low-threshold voltage and high-threshold voltage MOSFET's in a single LSI. The low-threshold voltage MOSFET's enhance speed performance at a low supply voltage of 1 V or less, while the high-threshold voltage MOSFET's suppress the stand by leakage current during the sleep period. This technology has brought about logic gate characteristics of a1.7-ns propagation delay time and 0.3-/spl mu/W/MHz/gate power dissipation with a standard load. In addition, an MTCMOS standard cell library has been developed so that conventional CAD tools can be used to lay out low-voltage LSI's. To demonstrate MTCMOS's effectiveness, a PLL LSI based on standard cells was designed as a carrying vehicle. 18-MHz operation at 1 V was achieved using a 0.5-/spl mu/m CMOS process.

b. Dual Threshold CMOS (DTCMOS)

DTCMOS technique suggests the assignment of higher threshold voltages to some transistors in noncritical paths to reduce the leakage current and lower threshold voltages to transistors in the critical paths to maintain the performances.

In [110], the author use the dual-threshold technique to reduce leakage power by as-signing a high-threshold voltage to some transistors in noncritical paths, and using low-threshold transistors in critical path(s). In order to achieve the best leakage power sav-ing under target performance constraints, an algorithm is presented for selecting and assigning an optimal high-threshold voltage.

A general leakage current model which has been verified by HSPICE simulations is used to estimate leakage power. Results show that the dual-threshold technique is good for leakage power reduction during both standby and active modes. For some ISCAS benchmark circuits, the leakage power can be reduced by more than 80%. The total active power saving can be around 50% and 20% at low- and high-switching activities, respectively.

c. Variable Threshold CMOS (VTMOS)

Variable threshold CMOS technique is a body biasing technique. A self-substrate bias circuit is used to control the body bias that in turn varies the threshold voltage. In the active mode, a nearly zero body bias is used. In the standby mode, a deeper reverse body bias is applied to increase the threshold voltage to reduce the leakage current.

In [50], Circuit techniques for dynamically varying threshold voltage are introduced to reduce active power dissipation by 50% with negligible overhead in speed, standby power and chip area. No additional external power supply or additional step in process is required. A gate array with this scheme is fabricated in a 0.3μm CMOS technology whose performance is investigated. The gate array is best fit for multimedia portable applications that require low standby power dissipation and high performance.

d. Dynamic Threshold CMOS (DTMOS)

In DTMOS technique, the threshold voltage is altered dynamically according to the state of the circuit. In standby mode, the threshold voltage is increased to reduce leakage whereas in active mode threshold voltage is small to have high speed. Dynamic thresh-old CMOS can be achieved by connecting the gate and body together.

In [10], the author proposed a novel operation of a MOSFET that is suitable for ultra-low voltage (0.6 V and below) VLSI circuits. Experimental demonstration was carried out in a Silicon-On-Insulator (SOI) technology. In this device, the threshold voltage of the device is a function of its gate voltage, i.e., as the gate voltage increases the thresh-old voltage (Vth) drops resulting in a much higher current drive than standard MOSFET for low-power supply voltages. On the other hand, Vth is high at Vgs= 0, therefore the leakage current is low. An extensive experimental results was provided and two–dimensional (2-D) device and mixed-mode simulations to analyze this device and com-pare its performance with a standard MOSFET. These results verify excellent inverter dc characteristics down to Vth= 0:2 V, and good ring oscillator performance down to 0.3 V for Dynamic Threshold-Voltage MOSFET (DTMOS).

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 thresh-old 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 Vbp and the bias to the p-type well of an nMOS transistor is termed Vbn. The voltage between Vdd and Vb, or between GND and Vbn is termed Vbb. In the active mode, the transistors are made to operate at low- Vdd and low- Vth for high performance. The fluctuations in Vth are reduced by an adaptive system that constantly monitors the leakage current, and modulates Vbb to force the leakage current to be constant. In the idle state, leakage current is blocked by raising the effective threshold voltage Vth by applying substrate bias Vbb. 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 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 trans-form processor reported a small 5% area overhead. The substrate-bias current of Vbb control is less than 0.1% of the total current, a small power penalty.

In [93], A CMOS body-bias generating circuit has been designed for generating adaptive body-biases for MOSFETs in CMOS circuits for low voltage operation. The circuit compares the frequency of an internal ring oscillator with an external reference clock. When the reference clock is “high,” forward body-bias is generated. When the reference clock is “low,” a reverse body-bias is generated. The forward body bias is limited to no more than 0.4 V to avoid CMOS latch up. The reverse body bias is limited to 0.4 V and is very effective in suppressing the sub threshold current.

The frequency adaptive body-bias generator circuit has been implemented in standard 1.5 μm n-well CMOS technology and simulated using SPICE. Excellent agreement is obtained between the simulated output characteristics and the corresponding experimentally measured behavior. It is also demonstrated that up to 90% leakage current in CMOS circuits can be reduced by applying the adaptive bias generator to lower threshold voltage CMOS circuits. The design is simple and can be embedded in low power CMOS de-signs such as the physical nodes of wireless sensor networks.

In [49], the author proposed the use of Adaptive Body Bias (ABB) to counter temperature variations along with process variations. ABB can be used to pull back the chip to the nominal operational region. Process variations and temperature variations can cause both the frequency and the leakage of the chip to vary significantly from their expected values, thereby decreasing the yield. CAD perspective was presented for achieving process and temperature compensation using bidirectional ABB. Mathematical models are used to determine the exact amount of body bias required to optimize the delay and leakage, and an algorithmic flow that can be adopted for gig scale LSI systems is provided.

In [99], Bidirectional ABB is used to compensate for die-to-die parameter variations by applying an optimum pMOS and nMOS body bias voltage to each die which maximizes the die frequency subject to a power constraint. Measurements on a 150-nm CMOS test-chip which incorporates on-chip ABB, show that ABB reduces variation in die frequency by a factor of seven, while improving the die acceptance rate. An enhancement of this technique, that compensates for within-die parameter variations as well, increases the number of dies accepted in the highest frequency bin. ABB is there-fore shown to provide bin split improvement in the presence of increasing process parameter variations.

5.3.4 Power gating

Power Gating is an extremely effective scheme for reducing the leakage power of idle circuit blocks. The power (Vdd) 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.

In [69], a new low power gating scan cell for scan based designs has been proposed in order to reduce power consumption in the scan chain as well as the combinational part during shifting. The author modified the conventional scan cell and augmented it with state preserving and gating logic that enables an average power reduction in combinational logic during shift mode. The new scan cell mitigates the number of transitions during shift and capture cycles. Thus, it reduces the average power consumption inside the scan cell and as a result the scan chain during scan shifting with a low impact on peak power during the capture cycle. Furthermore, due to introducing a new shorter shift path, improvements are observed in terms of propagation delay and power consumption in the scan chain during shifting. This leads to higher feasible shift frequency whereby the shift frequency is limited by the maximum power budget and hence results in reducing the test application time. The post-layout spice simulation results show a 7.21% reduction in total power consumption, an average 12.25% reduction of shift power consumption, and a 50.7% improvement in the clock (CLK)-to-shift propagation delay over the conventional scan cell in Synopsys 32/28 nm standard CMOS technology

In [97], the author introduced a new power gating technique for the GasP family of asynchronous circuits to achieve power savings. Large amount of power utilization in digital circuits is due to leakage current, as sub threshold conduction, junction leakage, and tunneling leakage through gate oxide. As per result from experiment, it is found that power gating is the most effective method to reduce sub threshold leakage. PMOS, a NMOS transistor is used to provide virtual power supply to block which is known as virtual Vdd and Virtual GND. NMOS, and PMOS transistor is known as sleep transistors. The power control logic turns on the power in anticipation of the receiving signal. The power control logic turns off the power when the circuit block is idle because either it is empty or pipeline is obstructed. GasP circuit make possible power gating is used in each stage. A latch is used in this article for storing the data coming from previous stage. This latch is power efficient because it drives only when necessary. It preserve its output and permits power gating.

5.3.5 Segment gating

The author in [39] 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.

In [51], the author establish an optimal power-down scheme, which is used as an up-per bound to evaluate several static policies on synthetic traffic patterns. Also, an evaluation of dynamic utilization-aware power-down policies using traces from the PARSEC benchmark suite is done. The author shows that both static and dynamic policies can greatly reduce static energy at low injection rates with only minimal increases in dynamic energy and latency. The author aim to reduce static power consumption through a comprehensive approach that targets buffers, switches, arbitration units, and links. Simulation results shows that the saved leakage energy for up to 99% of total cycles.

6. Comparison between different power reduction techniques

Works	**Power reduction**	**Parameters to reduce**
Techniques for reducing dynamic power		
[92]	50%	V_{th}/V_{add}
[79]	11%	Clock/a
[62]	15.03%	Clock/a
[5]	30%	Clock/a
[7]	81%	Clock/a
[60]	39%	V_{th} / f
[111]	17%-30%	V_{th} / f
[108]	50%	V_{th} / f
Techniques for reducing short circuit power		
[103]	1%-16% of short circuit power 0.7 micron	Voltage
[68]	36%	Voltage
[98]	86%	Voltage
Techniques for reducing leakage power		
[18]	53%	Voltage
[63]	27%	Voltage
[19]	55%-61%	Voltage
[42]	47%	Voltage
[110]	50%-20%	Voltage
[50]	50%	Voltage
[69]	7.21%	Power consumption

Table 2.1: Comparison between different power reduction techniques

7. Conclusion

The computer architecture community has adopted the multi-processor design as an attempt to avoid the power wall and further scale the performance of computers. The multi-processor architecture is more scalable than the traditional monolithic design, because increasing performance through parallelism consumes less power than increasing clock frequency. However, many challenges still need to be overcome in order to take full advantage of this massive on-chip parallelism. In particular, understanding the impact of different design choices on power and energy consumption is of ultimate importance to the success of multi- and many-processor chips in the future.

This dissertation analyzed multiple aspects affecting the scalability of the multi-processor design with focus on energy and power consumption on the network on chip.

CHAPTER 3

HEURISTIC OPTIMIZATION ALGORITHMSAND TASK MIGRATION MECHANISMS BASED NETWORK ON CHIP

1. Introduction

A heuristic algorithm is an algorithm that using a strategy that does not examine all possible solutions to a problem. Heuristic algorithms make no attempt to find the perfect solution to the problem. Instead, heuristic algorithms look for a "good enough" solution in an acceptable amount of time. A heuristic algorithm is one that will provide a solution close to the optimal, but may or may not be optimal. The concept of heuristic solutions to problems normally solved via non-polynomial time algorithms has changed the way programmers regard NP and NP-Complete problems.[70]

A heuristic is an optimization algorithm that can quickly find a solution to a problem without providing a guarantee on the quality of the solutions. This type of algorithm is often used to solve NP-hard problems since it is not expected to reach the optimal solution in a reasonable time. The goal of optimization is not only to find a valid solution, but to find the best solution out of all possible ones, that is, one that maximizes (or minimizes, depending on the problem definition) the value of the objective function.[58]

In this chapter we will talk about different heuristic methods and mention the characteristic of heuristic methods, classification of heuristic optimization methods, mapping and scheduling techniques in NOC.

2 Preliminaries Of Heuristic Optimization Methods

The techniques of improvement vary depending on the outcome to be achieved and the results provided by these techniques. To achieve this, there are several strategies that can be followed to ensure the validity of the results reached:

2.1 Exhaustive search

Exhaustive search is the simplest of search algorithms is exhaustive search that tries all possible solutions from a predetermined set and subsequently picks the best one.

2.2 Local Search

Local Search is a version of exhaustive search that only focuses on a limited area of the search space. Local search can be organized in different ways. Popular hill-climbing techniques belong to this class. Such algorithms consistently replace the current solution with the best of its neighbours if it is better than the current.

2.3 Divide and Conquer

Divide and Conquer algorithms try to split a problem into smaller problems that are easier to solve. Solutions of the small problems must be combinable to a solution for the original one. This technique is effective but its use is limited because there is no a great number of problems that can be easily partitioned and combined in a such way.

2.4 Branch-and-Bound

Branch-and-Bound technique is a critical enumeration of the search space. It enumerates, but constantly tries to rule out parts of the search space that cannot contain the best solution.

2.5 Dynamic Programming

Dynamic Programming is an exhaustive search that avoids re-computation by storing the solutions of sub problems. The key point for using this technique is formulating the solution process as a recursion. Dynamic Programming is a recursive method for solving sequential decision problems(hereafter abbreviated as SDP).

2.6 Greedy Technique

Greedy Technique A popular method to construct successively space of solutions is greedy technique, that is based on the evident principle of taking the (local) best choice at each stage of the algorithm in order to find the global optimum of some objective function.[70]

3. Characteristics Of Heuristic Optimization Methods

- They do not guarantee finding an optimal solution.
- They can be applied to many problems (general) because they do not rely on rigorous mathematical characteristics of the problem.
- Heuristics are generally used on global optimization problems. As a result they sometimes accept “uphill” moves (for a minimization problem).[112]
- The construction strategy,
- The improvement strategy,
- The component strategy,
- The learning strategy.[23]

4. Classification Of Heuristic Optimization Methods

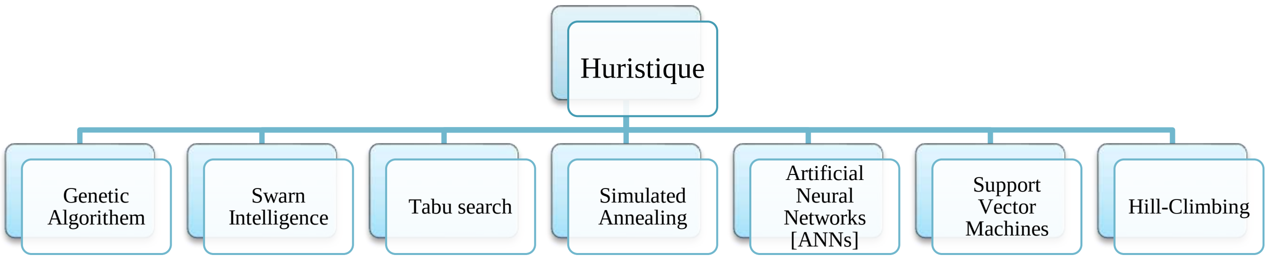


Figure 3.1: classification of heuristic optimization methods

4.1 Genetic Algorithms

4.1.1 Definition:

are subcategories of evolutionary algorithms. It is inspired by the theory of evolution, with features such as mutation, crossover, and selection. Parameter and system diagnosis, control systems, robot applications, image and voice recognition, engineering designs, planning, artificial intelligence applications, expert systems, function and combinatorial optimization problems such as network design problems, routing problems, scheduling problems, social and economic planning problems they are used in many different areas.

Genetic algorithms are a type of optimization algorithm, meaning they are used to find the optimal solution(s) to a given computational problem that maximizes or minimizes a particular function. Genetic algorithms represent one branch of the field of study called evolutionary computation [48], in that they imitate the biological processes of reproduction and natural selection to solve for the test' solutions [35].

4.1.2 Working of Genetic Algorithm:

GA starts its working from a set of solutions rather than a single solution. The set of solutions is called population. The initial population is generated randomly. Each solution of the problem is adequately represented by encoding a string (Chromosome) of bits or characters (Genes). Every chromosome has a fitness value associated with it.

The collection of chromosomes with their corresponding fitness values is called population. The population at a particular instance is called generation.

Fitness function is one of the major decisive parameters of „Genetic Algorithm". It defines the objective of the problem to be optimized. A pair of chromosomes based on their fitness values is used to reproduce off springs.

The genetic properties of both the chromosomes are intermixed to generate better offspring, such a mechanism is called crossover. After crossover operation, the genetic characteristics of the generated offspring are further modified. Mutation is a procedure to modify the characteristics of the generated offspring to make it more effective. The algorithm terminates when the required condition is fulfilled .

4.1.3 Characteristics Of Genetic Algorithm:

- Genetic Algorithm is based on the concept of natural selection and natural genetics.
- These are easy to understand and implement.
- These are extensively used in optimization problems.
- These work on population of points rather than an individual point.
- These use probabilistic transition rule instead of deterministic rules.
- Genetic Algorithms can effectively deal with large number of variables.
- These algorithms do not require any derivative information.
- These are more effective for complex problems than simple problems.
- These provide solution quickly as compared to other traditional optimization techniques [61]

4.1.4 GA Operators :

A genetic algorithm consists of three principal operations:

4.1.4.1 The selection operation:

Selection is one of the important processes of „GA‟. It is used to select an individual on the basis of its fitness value. The individual chromosomes that have higher fitness values are most probable candidates to be selected for reproduction. The individuals having low values will get little chance of their selection. In simple words the probability of a chromosome to be selected for reproduction is proportional to its fitness value.[61]

a. Type Of Selection :

- ❖ Tournament selection
- ❖ Roulette wheel selection
- ❖ Proportionate selection
- ❖ Rank selection
- ❖ Steady state selection[24]

4.1.4.2 The crossover operation:

The crossover operation generates the off spring from two chosen individuals in the population, by exchanging some bits of the two individuals. The offspring will then inherit some characteristics of their parents.[95]

It is a technique used to combine the individual chromosomes which produces a new chromosome. The offspring generated by crossover is supposed to have the features of both of the parents chromosomes. The objective of crossover operation is to find better solution from good solution. It is also called recombination operator. It selects a pair of individual chromosomes (Parents) for mating. First of all, a location for crossover is selected, and then bits or characters of the selected parents following marked location are swapped. Crossover operation is used to avoid duplication of parents used in recombination. The off springs generated by crossover operation may have either higher fitness or in some cases lower fitness than their parents. However a high fitness offspring is desirable from crossover operation. Crossover operator can be implemented by using several techniques.[61]

a. Type Of Crossover:

- ❖ Single-point crossover
- ❖ Two-point crossover
- ❖ Uniform crossover
- ❖ Flat crossover[81]

4.1.4.3 The Mutation Operation:

The mutation operation generates the off spring by randomly changing one or several bits of individuals the offspring may then possesses different characteristics from their ascendants. Mutation can then avoid local search in the searching space and increase the probability of finding the global optimum.[61]

a. Type Of Mutation:

- ❖ Insert Mutation
- ❖ Inversion Mutation
- ❖ Scramble Mutation
- ❖ Swap Mutation
- ❖ Flip Mutation
- ❖ Interchanging Mutation
- ❖ Uniform Mutation
- ❖ Creep Mutation[72]

4.2 Swarm Intelligence :

was introduced in 1989. It is an artificial intelligence technique, based on the study of collective behavior in decentralized, self-organized, systems. Two of the most successful types of this approach are Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). In ACO artificial ants build solutions by moving on the problem graph and changing it in such a way that future ants can build better solutions. PSO deals with problems in which a best solution can be represented as a point or surface in an n-dimensional space. The main advantage of swarm intelligence [32] techniques is that they are impressively resistant to the local optima problem.[32]

4.2.1 The swarm algorithms:

The swarm algorithms are search algorithms that are inspired by the movements of the swarms in nature. A lot of individuals interact with each other to solve a certain problem. There are a lot of algorithms. Famous among them are Particle Swarm Optimization, Acritical Bee Colony, Ant Colony, Firefly Algorithm.[58]

4.3 Tabu Search:

Tabu search is prohibited again in the next steps to prevent repetitive movement during the steps leading up. Thus, regional research is conducted to investigate solutions to achieve the best solution. [58]

Tabu Search (TS) is an optimization method designed to help a search negotiate difficult regions (i.e. to escape from local minima or to cross-infeasible regions of the search space) by imposing restrictions. It was originally developed as a method for solving combinatorial optimization problems (these are problems where the control variables are some form of ordered list

Tabu search is a “higher level” heuristic procedure for solving optimization problems, designed to guide other methods (or their component processes) to escape the trap of local optimality. Tabu search has obtained optimal and near optimal solutions to a wide variety of classical and practical problems in applications ranging from scheduling to telecommunications and from character recognition to neural networks. It uses flexible structures memory (to permit search information to be exploited more thoroughly than by rigid memory systems or memory less systems), conditions for strategically constraining and freeing the search process (embodied in tabu restrictions and aspiration criteria), and memory functions of varying time spans for intensifying and diversifying the search (reinforcing attributes historically found good and driving the search into new regions). Tabu search can be integrated with branch-and-bound and cutting plane procedures, and it has the ability to start with a simple implementation that Can be upgraded over time to incorporate more advanced or specialized element.[33]

Tabu search [34] 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.

4.4 Simulated annealing (SA):

Simulated annealing is a generic probabilistic meta-algorithm for the global optimization problem, namely locating a good approximation to the global minimum of a given function in a large search space. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities). For certain problems, simulated annealing may be more effective than exhaustive enumeration provided that the goal is merely to find an acceptably good solution in a fixed amount of time, rather than the best possible solution.

4.4.1 The SA algorithm:

In the Simulated Annealing algorithm, an objective function to be minimized is defined. Here it will be the total path length through a set of points. The distance between each pair of points is equivalent to the "energy" of a molecule. Then, "temperature" is the average of these lengths. Starting from an initial point, the algorithm swaps a pair of points and the total "energy" of the path is calculated.[32]

4.4.2 The simulated annealing algorithm:

can give a reasonable approximation for a function with a large search space. The place decides between probabilistic survival or reversion to another situation. This will direct the system to low energy. Electronic circuit design is widely used in the solution of problems such as image processing, road finding problems, travel problems.[58]

4.5 Neural Networks :

Neural Networks are inspired by biological neuron systems. They consist of units, called neurons, and interconnections between them. After special training on some given data set Neural Networks can make predictions for cases that are not in the training set. In practice Neural Networks do not always work well because they suffer greatly from problems of under fitting and over fitting [46]. These problems correlate with the accuracy of prediction. If a network is not complex enough it may simplify the laws, which the data obey. From the other point of view, if a network is too complex it can take into account the noise that usually assists at the training data set while inferring the laws. The quality of prediction after training is deteriorated in both cases. The problem of **premature convergence** is also critical for Neural Networks.[32]

4.6 Artificial neural networks(ANNs):

Artificial neural networks are very functional models for pattern recognition and machine learning, which categorize new patterns from acquired training data. It was inspired by the neuron function in the animals' brains. Many areas such as speech analysis, image processing, etc. are used.[58]

4.7 Support Vector Machines (SVMs):

Support vector machines extend the ideas of Neural Networks. They successfully overcome premature convergence since convex objective function is used, therefore, only one optimum exists. Classical divide and conquer technique gives elegant solution for separable problems. In connection with SVMs, that provide effective classification, it becomes an extremely powerful instrument.

A Support Vector Machine (SVM) performs classification by constructing an N dimensional hyper plane that optimally separates the data into two categories. SVM models are closely related to neural networks. In fact, a SVM model using a sigmoid kernel function is equivalent to a two-layer, perception neural network.**[**32**]**

Supporter Vector Machine is a model that grasps patterns and analyzes data by using training data with artificial intelligence. These algorithms are used for regression analysis and classification purposes. Using the sample data, the algorithm will sort the new samples into groups. These SVMs are a subset of the artificial intelligence that systems learn from knowledge in machine learning, and require training data before analyzing new examples.[58]

4.8 Hill-Climbing Algorithm:

Hill-climbing algorithm is effective, but it has a significant drawback called **premature convergence(**When a genetic algorithms population converges to something, which is not the solution, we wanted.**)**. Since it is greedy, it always finds the nearest local optima of low quality. The goal of modern heuristics is to overcome this disadvantage. [32]

5. Mapping And Scheduling Techniques In Noc

Mapping and scheduling are known to be computationally hard problems. A large range of exact and approximate optimization algorithms have been proposed by different groups for solving these problems. The methods include Branch-and–Bound (BB), constructive and transformative heuristics such as List Scheduling (LS), Genetic Algorithms (GA) and various types of Mathematical Programming algorithms. [36]

5.1 Key Factors On Task Mapping:

Due to its criticity, some key factors must be considered in the stage of mapping of tasks onto a NoC system. Such key factors are described below.

5.1.1 Target Architecture:

The target architecture is related to whether nodes on the NoC system are heterogeneous or homogeneous. Heterogeneity is the most common case, because this factor may improve system performance in 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)[57].

5.1.2 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 on 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, or even the entire system [55][57].

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 System C [30][57]. TLM has been used successfully for synthesizing high speed MPSoC systems [101][57], and for modeling the communications infrastructure of a NoC [80][57].

5.1.3 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 for the

Objective Functions:

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.

5.1.4 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).

5.1.5 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 connections distribution, which are generated by means of mathematical functions [54][31][77,78][56][37][73][31][16][87][27][43][83][101][91][57]. Irregular Topologies are often the mixture of two or more regular forms, which leads to hybrid, hierarchical or totally irregular topologies.

5.1.6 Optimization Algorithms:

As already mentioned, the mapping stage relies on an optimization process, which searches along a 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 in the communications nature [37]. For instance, off–line (static) optimization forces to having predictable communication assessments, whilst dynamic algorithms allow a more flexible communication scheme. A subset of static algorithms encompasses the so called exact approaches, which are based on mathematical modeling of the optimization problem. Integer Linear Programming (ILP), Non Integer Linear Programming, and Mixed Integer Linear Programming, are well–known instances of exact algorithms, but their drawback relies on their poor convergence performances as the problem size increases [73][57].

On the other hand, search–based techniques are divided in heuristic and deterministic algorithms. Deterministic algorithms are devoted to search along the whole solution space, whereas heuristic algorithms use the previous experience in order to improve the searching process. Among heuristic algorithms there are some approaches which work with evaluative techniques (transformative) and some others which produce partial solutions in an iterative fashion until a good–enough solution has been reached (constructive). Dynamic algorithms are all based on heuristics. They must be quick enough to deliver a reasonably good solution in run time, without sacrificing task mapping quality.

5.1.7 Tools

Some software tools are available for supporting the task mapping stage in NoC design environments. Among such tools the following can be mentioned below.

• SUNMAP selects the best topology according to application constraints (power, bandwidth, communication delay) and generates the nodes allocation for the target application and architecture. The process involves three steps. Firstly, routing algorithm and allocation objectives must be selected. In second place, the best topology is chosen and thirdly, a model of the system is provided through SystemC descriptions [85][57].

• SMAP is a mapping and simulation tool developed in the Matlab environment, which provides several optimization choices for the solution space exploration. Some of these options are Genetic Algorithms, random, and spiral. The communication among nodes can be simulated with both deterministic routing algorithms (such as XY) or adaptive algorithms (such as west-first or backtracking). Some figures of merit assessments, such as power consumption or execution time, are provided by the tool[87][57].

• HeMPS is a custom platform for design and simulation of NoC-based MPSoCs. This tool is based on the Hermes network, a Noc with a two dimensional mesh topology, a XY routing algorithm, and a wormhole commutation mode. Nodes in Hermes may be a MIPS processor, a RAM memory module, a DNA module, or a NI module. First design stage in HeMPS implies identifying the application specifications and constraints. After that, some hardware platform parameters (such as the size of the network, packet size, memory size, etc.) must be settled and the partitioning algorithm is able to start. The last stage implies the task mapping of the application on the selected platform. Both static and dynamic mapping is supported. The designer is able to integrate hardware and software components to perform a simulation and validation of the whole system. The final stage generates a description of the platform by means of a Hardware Description Language (HDL) [27][57].

• OPNEC is an open code platform for designing and simulating NoC systems. It supports a variety of 2D and 3D NoC architectures and several topologies (mesh, torus, ring, bus). It is also capable of working with both static XY routing algorithms and adaptive approaches, and supports several kinds of processor and memory modules. Several optimization objectives might be used, such as energy consumption and communication delay. Energy assessments are achieved by means of RTL models and are aimed to provide estimations of the whole network system [57].

5.2 Related Work Of Mapping And Scheduling Techniques In Noc:

Amit Kumar Singh, Wu Jigang, Alok Prakash, ThambipillaiSrikanthan (Senior Member, IEEE) working for packing strategy and two run-time mapping heuristics based on it, to map the applications efficiently onto an 8 × 8 NoC-based heterogeneous MPSoC.First heuristic tries to map the tasks of an application in close proximity, reducing the communication overhead (communication time) between the communicating tasks. The second heuristic considers traffic in addition to the proximity of tasks while mapping, resulting in more uniformly distributed channel load. The hardware resources (Reconfigurable Logic) present in the platform support two tasks in parallel, resulting in further reduction of communication overhead.[9]

Amit Kumar Singh ,ThambipillaiSrikanthan, Akash Kumar, Wu JigangThey rose describes a new mapping strategy where placement for a task is found by looking at previously mapped tasks onto a processing element (PE) in the multi-tasking MPSoC platform. they have relied on this mapping strategy to propose four run-time mapping heuristics. A simulation platform has been extended to support the mapping of more than one task on each PE, which can be either a CPU or a reconfigurable hardware (RH) block. they show that the ideal static mapping solution can lead to performance improvement. However, the ideal static mapping has been shown to improve the overall performance only when all the applications and their workloads are known prior to the mapping process. However, this does not cater for realistic scenarios in which the run-time characteristics call for dynamic mapping strategies. all the proposed heuristics have been evaluated using an 8*8 NoC-based MPSoC platform. they clearly demonstrate that the newly presented heuristics can consistently provide for notable reduction in the communication overhead. The potential of mapping adjacent communicating tasks and those of the same application on to the same PE whenever possible has contributed to the overall reduction in the communication overhead. they have investigated different scenarios depending on performance metrics of interest and they show that improvement in the total execution time can be up to 90% when the packet execution time is increased.[8]

Dawei Li, Jie Wu they address the energy-efficient contention aware application mapping and scheduling problem on NoC-based MPSoCs.they present a model where processors' voltage scaling and NoC links' frequency tuning can be combined together to reduce the overall system energy consumption. A two-step approach is adopted. First, the application mapping problem, which aims to find the mapping that minimizes the communication energy, is formulated as a quadratic integer programming problem, and solved by a relaxation-based iterative rounding scheme; they also provide a light-weight algorithm as an alternative, which reduces the overall algorithm complexity significantly and achieves comparable energy saving levels. The second problem, the application scheduling problem, is solved by a developed genetic algorithm, combined with a scheduling algorithm based on the ETF strategy. Finally, a mapping and scheduling for the application is derived, which significantly reduces the overall system energy consumption, verifying that jointly utilizing dynamic voltage scaling on processors and frequency tuning on NoC links provides great potential for overall energy reduction in MPSoCs.[22]

Ou He, Sheqin Dong, Wooyoung Jang, JinianBian David Z. Pan they have a unified flow combining task scheduling and core mapping named UNISM is proposed to support regular mesh, irregular mesh and custom NOC, using MILP. To enable this MILP modeling on irregular and custom NOC, a novel graph model called Labeled Graph is developed to calculate the communication latency and energy. Moreover, this model does not introduce any new variables,whichmakes our unifiedmodel as easy as a single scheduling algorithm in terms of the number of variables in MILP. Then, we accelerate our UNISM using NOC localization.[75]

Heng Yu, Yajun Ha, and Bharadwaj Veeravallithey proposed an NoC-targeted algorithm which determines transmission routing and scheduling in the process of task mapping and scheduling. We use three real life applications to evaluate our algorithm and the results appear significantly better than contemporary NoC-targeted mapping algorithms. However, this work achieves predictable performance gain with the prerequisite that the communication should exhibits regular access patterns (hence data transmission time per instance does not have large variation). In the future, they are planning to investigate the scheduling algorithm to deal with irregular access patterns and cache influences.[108]

Raina and Muthukumar (2009) have presented a heuristic and traffic aware mapping and scheduling algorithm for hard NoCs. This algorithm acts static and NoC is assumed to be homogeneous, so, in this algorithm, the cost of any PEs is not considered in the scheduling and mapping. Goals of this algorithm are reduce traffic and NoC delay. Rajaee et al (2010) have presented an energy aware and heuristic scheduling and mapping algorithm. This algorithm is suitable for scheduling and mapping tasks to real-time applications on heterogeneous and hard NoC. This algorithm simultaneously considers the costs of processing of PEs and communication in chip. This algorithm can reduce energy consumption and traffic of NoC. Jueping et al. (2010) proposed a new energy and communication aware architecture for NoC with a Shared Memory (SM) on each node to reduce communication delay between PEs. In this architecture PE to PE communication are divide into two steps: PE to SM and SM to PE. With using Simulated Annealing (SA) algorithm authors can reduce power consumption and transferring cycles. Jang and Pan (2010) proposed a new algorithm for NoC which called Architecture-Aware Analytic Mapping (A3MAP). This algorithm proposed for homogeneous NoC with regular mesh topology, heterogeneous NoC with irregular mesh topology and custom NoC architecture. Authors use of two algorithms for development of itself algorithm: relaxation algorithm and genetic algorithm. Performance of this algorithm with genetic algorithm show that can be better than relaxation algorithm to reduce traffic in homogeneous NoC with regular Mesh, heterogeneous NoC with irregular mesh topology and custom NoC architecture.[66]

Hu and Marculescu (2005-a) proposed a new Energy-Aware Scheduling (EAS) algorithm. This algorithm presented for statically scheduling of specific transactions and computing tasks in heterogeneous NoCs. This algorithm automatically distributes application tasks onto PEs and schedules under real-time constraints. This algorithm instead of considering the performance criteria as the objective optimization, try reducing energy consumption under hard performance limitations. In this algorithm communication scheduling and computing tasks scheduling can done parallel. Mehran et al (2007) proposed a new core mapping algorithm for NoC that called Spiral. To develop this algorithm used of heuristic method and presented for two dimension mesh topology. Authors used of XY routing algorithm to implement and compared this algorithm with genetic algorithm and random mapping algorithm. This algorithm reduced energy consumption than genetic and random algorithm. Authors show that this algorithm is fast and can use for fast dynamic core mapping. Chou and Marculescu (2008) first reviewed effect of different type of contention in the network on the application mapping and then proposed a contention-aware mapping technique for reduce contention and energy consumption in the NoC. Authors implement this technique with using XY routing algorithm in 2-D mesh topology. Authors show that with using of this technique packet latency can be reduced. Hu and Marculescu (2005-b) proposed a new energy and performance aware mapping algorithm. This algorithm presented for regular NoC. For develop this algorithm authors use of deterministic routing which is deadlock free. Authors show that this algorithm can be reduce communication energy than ad hoc implementation and can be use for different regular NoC topologies.[66]

Bandwidth can be an effective role on energy consumption in the NoC architecture. To reduce bandwidth requirement of NoC, Wang et al (2011) proposed a new mapping technique to task scheduling of an application. In this technique use of Ant Colony Optimization technique to task mapping. Authors show that this technique can reduce bandwidth requirement and cost of NoC implementation. Wu et al (2011) has proposed a new Energy-aware Mapping Algorithm which called GA-MMAS. This algorithm used from Genetic Algorithm (GA) and MAX-MIN Ant System Algorithm (MMAS) to reduce energy consumption of NoC. Authors for this algorithm first improve MMAS, and then with combination of MMAS and GA, make compensation to lack of pheromones in the initial stage of MMAS. Increase the accuracy of the optimal solution, which will reduce energy consumption.[66]

6. Task Migration Mechanisms

6.1 Definition of task migration:

Task migration 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[84]. 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.

6.2 Task Migration Mechanisms:

the task migration is initiated by a process monitoring the load statistics. The migration is started after the migration monitor has decided on a migration ,the main mechanisms that exists in literature such as:

6.2.1 Checkpoints:

A migration checkpoint is a physical point in the program where a task migration is possible. The point is often provided manually by the programmer and should be put in a adequate location. The location of this point must not violate any logic or misuse any variables if the task is migrated. At this migration point the program will, with some mechanism, check if the currently running task has got a migration request. This request is made by the migration monitor that runs either on the local core or on some distant core, as explained earlier in the chapter.

The Condor project [65][91] deals with transparent checkpoints, that do not have to be set by the programmer. The programmer inserts checkpoints by including the condor library, and the software sets the checkpoints automatically. The Condor project assumes a UNIX base from which certain system calls are made in order to build the checkpoints. The Condor project does not support migration of tasks that communicate via signals, sockets, pipes or files.

The authors in [94][91] carried out experiments related to minimizing overhead and maximizing performance. The system was an asymmetric multi-core platform with a shared memory. The migration mechanism used a master-slave approach (see next section) and re-creation way of moving tasks on the target core. The first experiment determined the overhead of the checkpoint mechanism.

The next experiment in [94] was used to determine how long time is needed for certain tasks to receive a certain amount of CPU time. The experiment used two CPU cores and different sets of loaded tasks.

6.2.2 debug registers:

Task migration using debug registers [100][91] is a poll-free strategy that uses hardware generated interrupts when a task has got a migration request. This approach is only supported if a platform has programmable debug registers. Debug registers are special registers used for example in real-time debugging for setting break points in the code.

When a resource decides to migrate a task it simply activates the IAC registers, which will now generate an interrupt when the PC reaches the value of IAC. Activating the IAC registers means turning on the compare match interrupts for IAC. The interrupt routine can later call the task migration handler and provide the information regarding the relevant checkpoint to the handler.

Task migration with debug registers, according to the code 3.2, uses no overhead checking the migration table like the manual checkpoint approach did. The reaction time t is still present though, because of the time from where the IAC are activated until the PC reaches its value. The Hardware platform must however support these debug registers in order to provide this functionality. The debug registers are often limited to a rather small amount and therefore few migration points are available for the tasks.[91]

6.2.3 incoming migrations:

Once a migrating task has been transferred to the target core, the receiver must have a mechanism capable of loading the task into the new CPU core. The two cores must handle both communication and data transfer in order to achieve a successful migration. A problem arises when a busy core is going to handle an incoming task without altering both its own and the transmitter’s performance or response time.[91]

6.2.4 accepting task migrations by polling:

Polling is a method in which the target core regularly checks a certain location for information. The location storing the information could be shared or local. The response time of accepting a task by polling is dependent on the polling interval. Due to this response time, the incoming tasks could be more than one — and should therefore be stored in a buffer. The buffer could be either located in a shared memory or in the local memory.

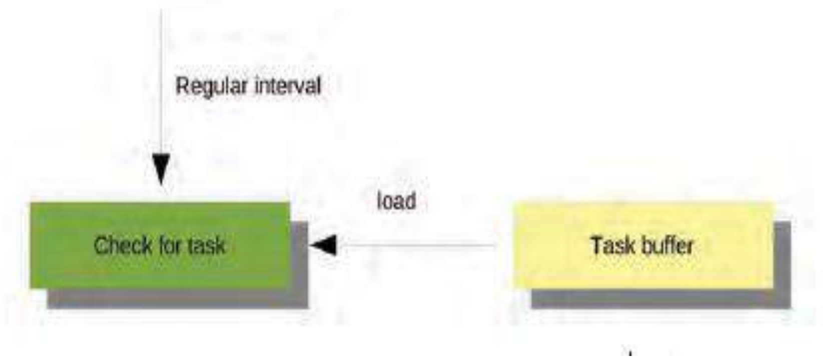


Figure3.2 Polling mechanism for incoming tasks[91]

The response time of accepting a task by polling is dependent on the polling interval. Due to this response time, the incoming tasks could be more than one — and should therefore be stored in a buffer. The buffer could be either located in a shared memory or in the local memory. The model shown in figure 3.1 places all the incoming tasks in the task list of the operating system until the buffer is empty. In the most cases there are no tasks in the input buffer, so the mechanism that checks the buffer must be made very light. Minimal overhead should be considered in the aforementioned default case.[91]

6.2.5 accepting task migrations by interrupts:

An interrupt means that the CPU stops its current task and jumps to a selected interrupt routine when an event occurs. Interrupts can be used in order to eliminate the reaction time present in a regular polling strategy. The interrupt can be triggered by an external pin, incoming data transfer or some other interrupt based mechanism.

As shown in figure 3.2, the incoming task triggers an interrupt on the target core in order to signal that the core must handle this request. In this case the incoming migrated task is only one to the amount, because each task triggers exactly one interrupt. The task is stored right away in the target task list (or possibly rejected) by the handler. Even by using interrupts, the core-to-core communication is important. The communication in this case must ensure that the receiver core is not in a critical state or sleep state; thus not have any interrupts enabled, which would lead to no response from the target core at all.[91]

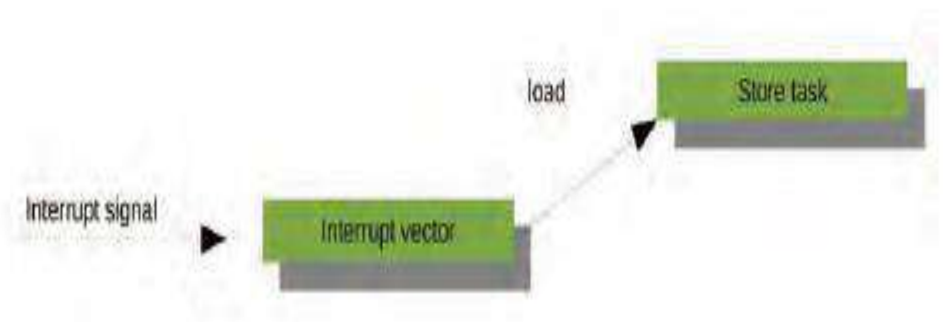


Figure3.3 Interrupt mechanism for incoming tasks[91]

7. Conclusion

The optimization algorithms contribute to finding the optimal and best solutions, despite the different methods of work and their uses. Therefore, one must choose one of them by comparing their characteristics with the elements of the problem to be addressed.

Despite the arithmetic difficulty represented by the planning and scheduling, the use of optimization algorithms to a great degree of satisfaction from the researchers, has been easy in their work significantly, the best possible to transfer and migration of tasks through NOC we have mentioned some of them in the chapter.

For the migration and migration of tasks at the level of NOC different reasons and yet it does not move randomly, but there are mechanisms to be followed first determine the destination of the transition and compare with other possible destinations

CHAPTER 4

CONTRIBUTIONS, TESTS AND EXPERIMENTS

1. Introduction

Network-on-Chip (NoC) is an emerging paradigm for communications within large systems implemented on a single silicon chip.An NoC is constructed from N×N processors and links interconnected by routers, such that messages can be relayed from any source to any destination over several links, by making routing decisions at the routers. Where when the execution ends task on processor (node), the token is sent to a task of lower priority, this increases the power and time of execution.

Our objective is to minimize the energy and time, in this chapter we will detail the proposed solution while taking as case of study the mapping of two multi-applications task graph (the multimedia system (MMS) and the automotive industrial) and random task graph into 5x5 NoC based mesh

2. Contribution in task migration

The idea in solving the task migration problem depends on the characteristics of both, the target architecture and the application that will be run on it.

2.1 Target architecture

that the models we adapt is a NOC 2D-mesh N*N , which can symbolize him G(P,L)with P represent a Processors in NOC and L represent Link carrier of messages between processors,

where all processing elements (Pes) have distinguished by each other in terms of VOLTAG, Messages are transferred from the wizard to the wizard, so before moving them, make sure that the wizard is not busy. If the processor is busy, the wizard searches for another port to pass the message, so the port is next to it and if not, it can wait until the end of the processor

Mapping has to make sure that only one task can run on a resource at a time, since the tasks are inhabited in the form of graph, there is a means that are in line with them so that task can't be executed only after the token ,this token are moving from the node that contain the executive task to node that the cargo will carry.

2.2 Application

In the application we adopted we are working to distributed all tasks on the various nodes to be implemented in a manner as well as priority and distinctive property, examplein Figure 4.1: the nods represent the router and the bows represent the direct physical links between the elements of the architecture in the platform, The nodes of this graph represent the tasks and the line represent the communications.

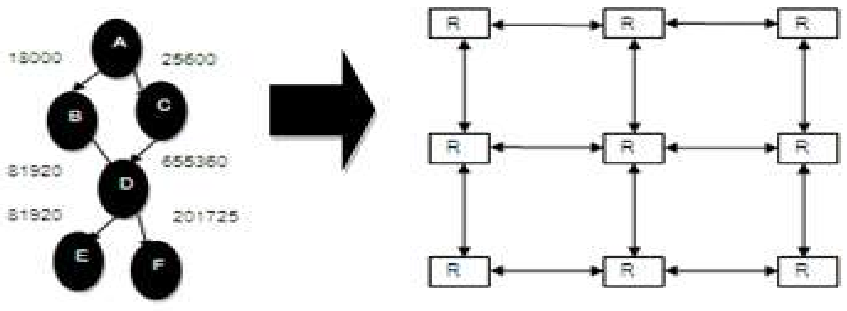


Figure 4.1: example to random task graph

2.3 Task model

The application we have selected is synchronous dataflow graph (SDF),which can symbolize him G(T, l) with T represent a task and l represent line between the tasks .

Each task is characterized by its name and clock_cycle representthe time it needs to execute. Priority represent It also has the execution property before another task. The initial level is given based on the dependencies between the tasks. the token represents the number of commands it needs to execute. The task before it represents the task it sends to the task. Task After that, which is waiting for the order of implementation of them, as we can find a task waiting for the order of implementation of two tasks before it to implement and send the order of implementation of two tasks after.

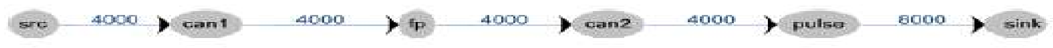
2.4 Benchmarks' multimedia applications (Case of study)

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.

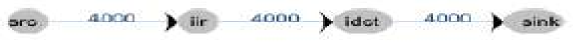
a. Automotive /industrial application

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

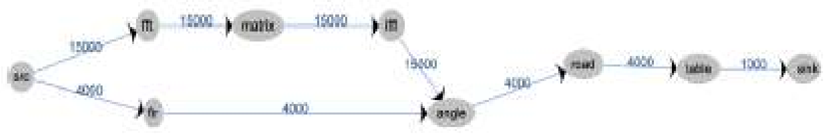
CTG 0 models an embedded automotive system that is composed of two controller area network (CAN) interfaces, a basic integer and floating point module and an actuator driven by pulse width modulation (PWM) signal.



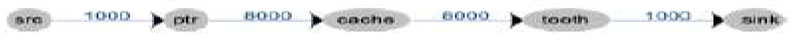
CTG 1 describes an embedded automotive system that is composed of an infinite impulse response (IIR) filter and an inverse discrete cosine transform (iDCT) module.



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.



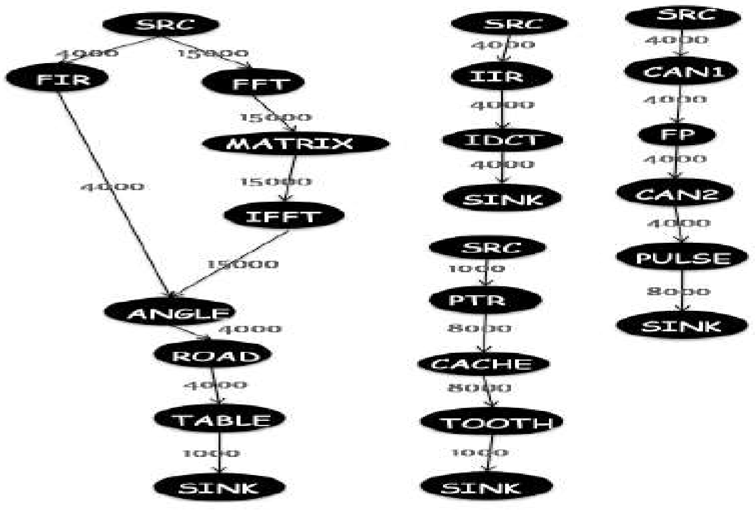


Figure 4.2: Automotive/industrial task graph.

b). Multimedia system (MMS) application

A generic Multimedia System (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.

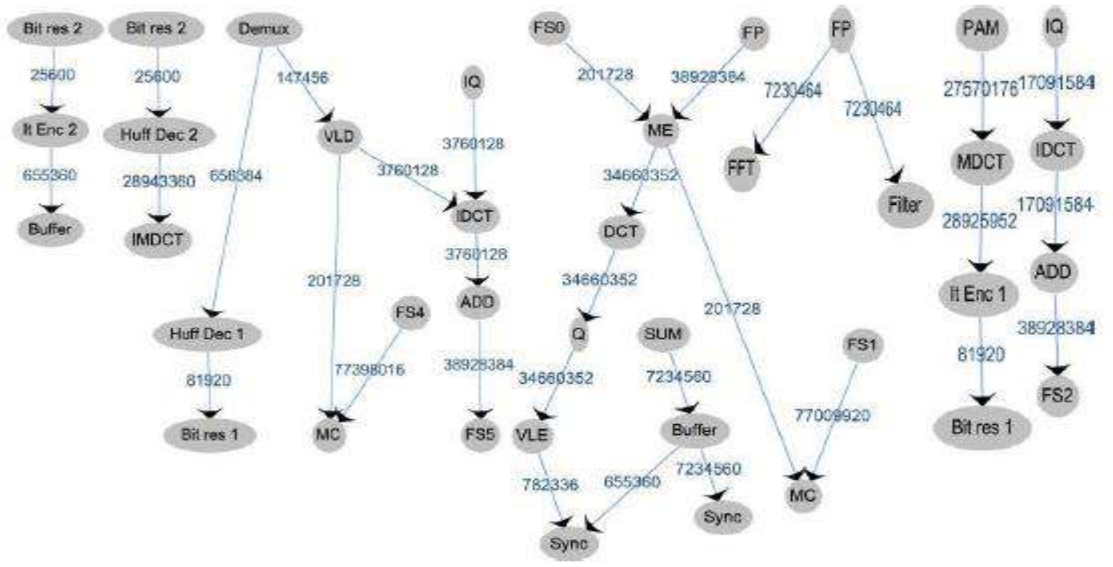


Figure 4.3: Multimedia system task graph.

2.5 Scheduling algorithm

In our program, each core has its own priority ordered queue to schedule tasks, so for a set of tasks that are allocated to the same node, a scheduling algorithm for single processor can be applied to determine the task with the highest priority in each time.

ALGORITHM 1.1 SCHEDULING_ALGORITHM

```
 Given: nbr_task: number of task ;
          N: number of node in NoC;
  Find: priority ordered queue for each node ;
FOR i ← 0 to nbr_task DO
Nbr_random : random number for random distribution of tasks on node;
FOR j ← 0 to N DO
IF (Nbr_random = j) then scheduling task i on node j ;
END IF
END FOR
END.
```

2.6 Routing algorithms

Performs the tasks in the chip as priority, where the completion of a task with a higher priority than a task current, token is sent to a lower priority task for execute. This requires Routing to guide token in the links to choose the best route from sender to receiver.

ALGORITHM 2.1 ROUTING_ALGORITHM

```
 Given: source, destination;
  Find: count;
WHILE(source != destination) DO
Search Coordinates source and destination;
IF(($x_{after\_source}$!= $x_{destination}$) AND (after_source = free))THEN source = after_source; x++;
ELSE IF(after_source =full) THEN wait;
       END IF
ELSE IF(($Y_{after\_source}$ != $Y_{destination}$)AND(after_source =free))THEN
     source =after_source;  y++;

ELSE IF(after_source =full) THEN wait;
         END IF
ELSE return count = x+y;
END IF
END WHILE
END.
```

2.7 Contribution in task migration

Task migration (TM) has been widely used in the distributed systems domain, but as a drawback, the migration time and energy overhead may be long in point that it will increase network congestion and degrade the system performance, we aim to optimize those problems ,For this you must be migration the tasks to only neighbor, to avoid these problems. We must take into account the values of fitness function before migration the task from one processor to another, because we want to optimize time and energy, not reverse.

ALGORITHM 4.1. TASK_MIGRATION_ALGORITHM

```
Given: chromosome
FOR i ← 0 to chromosome-length DO
Chromosome division on the number of processors;
Distribution 0 and 1 in matrix, the line represents task and The columns represent
processor (node);
FOR j ← 0 to number_line DO
FOR k ← 0 to number_columns DO
IF (matrix[j][k]==1 ) THEN fitness1=calculate fitness function $task_j$ on $processor_k$;
IF (matrix[j][k+1]==1) THEN fitness2=calculate fitness function $task_j$ on $processor_{k+1}$;
END FOR
IF(fitness1 <fitness2) THEN we leave a $task_j$ in $processor_k$ ;
ELSE migration $task_j$ in $processor_{k+1}$;
END IF
END FOR
END.
```

2.7.1 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.

2.7.2 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.

2.7.3 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.

2.7.4 Program code:

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

2.7.5 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, which can only beused if the tasks are executing on the same core. Otherwise the interface of communicationmust change, so that the tasks communicate over some kind of inter-core communication. Thetask calls between cores must be handled either in a way that the tasks pass values betweencores, 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.

2.7.6 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 tomigrate. For security reasons tasks that are selected for migration should preferably not shareglobal variables with other tasks, since the communication using global variables could bebroken or slowed down if one of the tasks is moved to another core.

3. The use of genetic algorithms with task migration

3.1 Problem description

After the task is distributed randomly on the processor, it is implemented on the characteristic of it so that a task cannot be executed until after it is executed by executing it. This comes in the form of a token. This token moves through the processor (from the processor that contains the executed task On the task waiting for the implementation of the implementation), and this movement consumes time and energy in addition to the time to wait for the implementation order, because of this problem, we thought to transfer these tasks so that the more tasks are closer to each other whenever we save energy and time, but can not transfer these tasks Random Wan A must determine the best positioning for these tasks Fast any genetic algorithm to achieve this.

Given: $G(P, L)$, $G(T, l)$

Find: minimized time of execution and energy consumption.

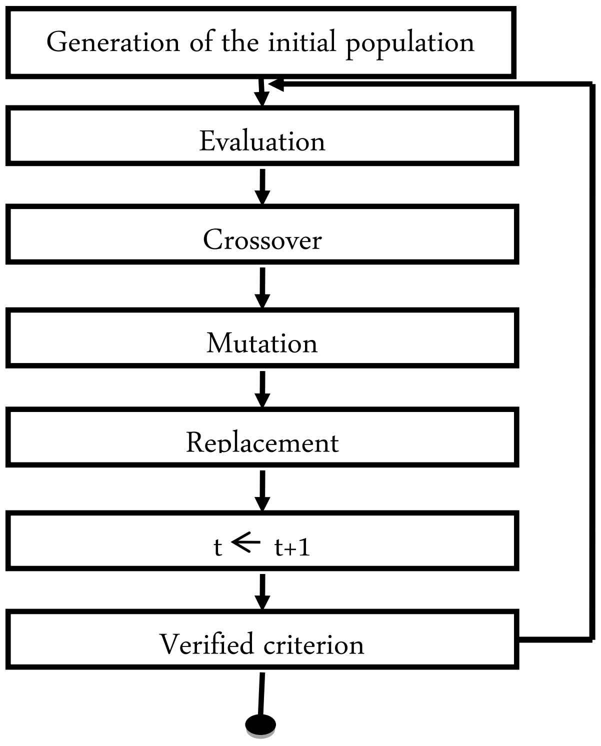
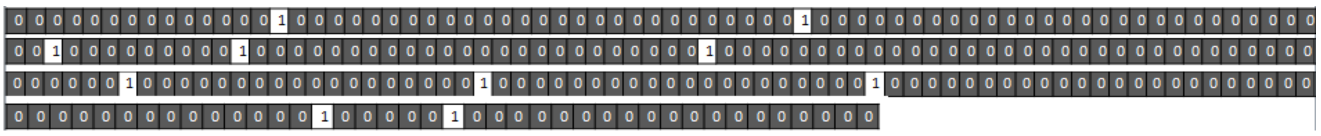


Figure 4.4:Classical genetic algorithm structure.

1) Problem Representation: The representation problem is the energy consumption and the time during implementation

2) Population: We create a set of chromosomes so that the length of each chromosome (number of treatments * number of tasks) contains one chromosome on suit number 0et 1

The chromosome is created by dividing the tasks on the processors so that they are marked with 1 if the task is in the processor and 0 if not, considering that it is not possible for a task to be in two processors at the same time.



```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 4.5: example of create chromosome

3) Evaluation: After we create the chromosomes we evaluate each chromosome alone by calculating the execution time, energy consumption and calculation fitness of them.

4) Crossover: This process is done by random selection of chromosomes, then we determine the center of each of them. Then we change the second half of the chromosome, so that the second

half of the second chromosome is placed with the first half of the first chromosome and the second half of the first chromosome. We put it with the first half of chromosome II,Example in figure 4.6

Before Crossover:

Chromosome2 :

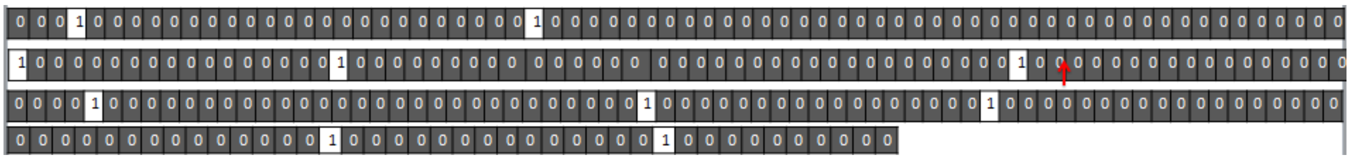


Figure 4.6: chromosome 2 before crossover

Chromosome16:

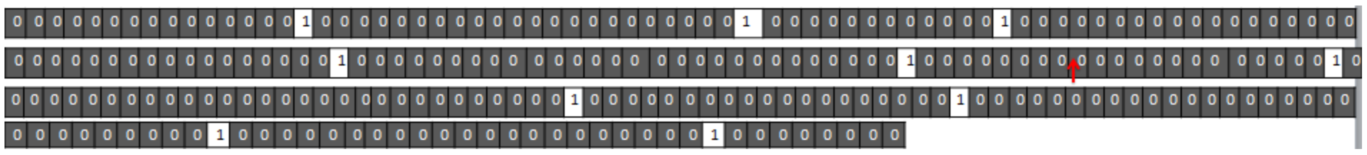


Figure 4.7: chromosome 16 before crossover

After Crossover:

Chromosome2 :

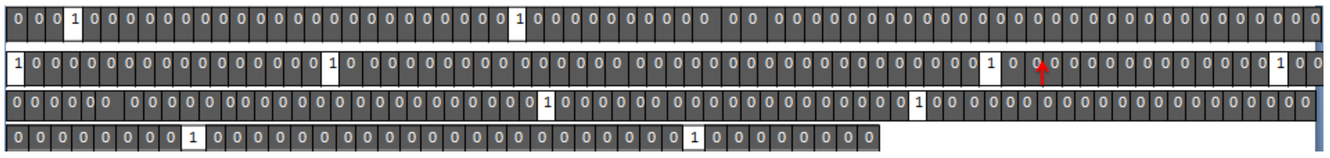


Figure 4.8:chromosome 2 after crossover

Chromosome16 :

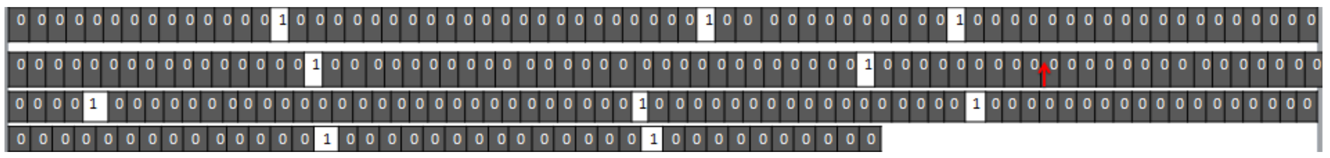


Figure 4.9: chromosome 16 after crossover

5) Mutation: After the crossover process, we select a random number enclosed between the middle of the chromosome and its end. We change the value of the location of this number so that if it is 1, replace it with 0, and if its value is 0, replace it with 1,exemple in figure 4.10

Chromosome 2:

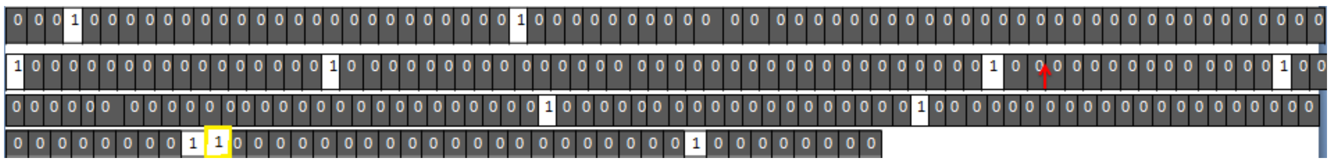
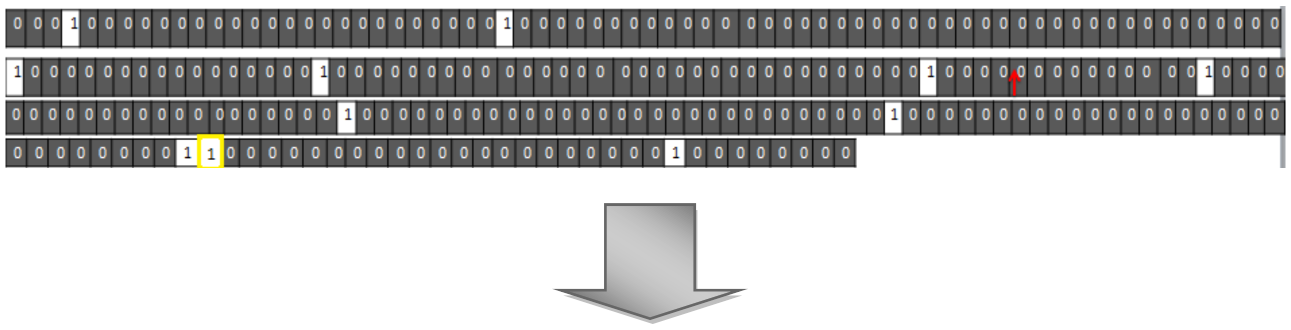


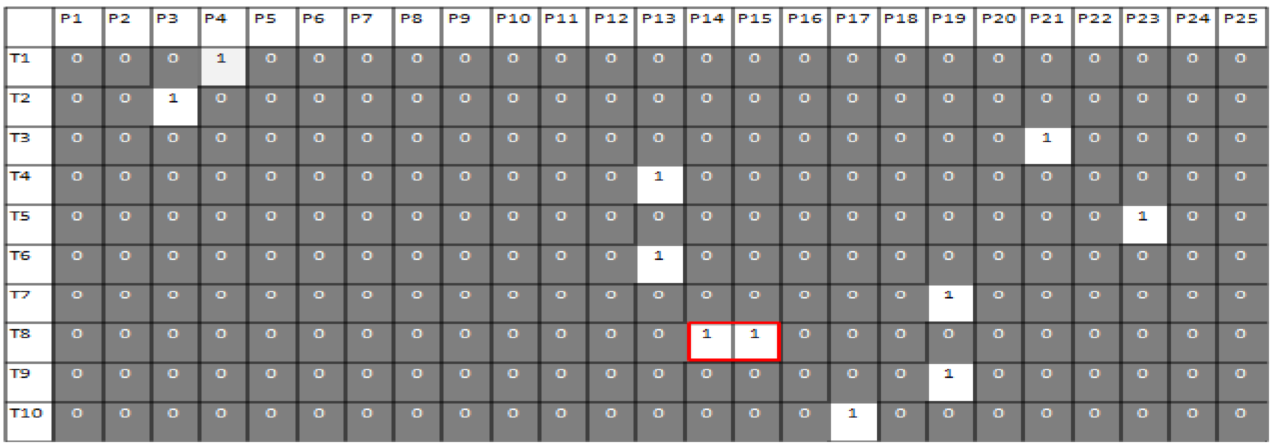
Figure 4.10: chromosome 2 after mutation

6) Replacement: Here we make a comparison between the chromosome of the Father and the Son and so through the calculation of the fitness of each of them. If the fitness of the Father is the best of the Son, we shall descend to the Son of the Father in the Father, in the population Son, and if the fitness of the Father is better than the Son, we shall descend the Father's chromosome in the Father's population.

7) Verified criterion: Here we put the values of the chromosome in a table to see if it is possible to transfer tasks between the processors provided that the processor next to the processor in which the task in which we redistribute tasks on the processors and re-implementation and calculate the time of implementation and energy consumption, example in figure 4.11

Chromosome2 :





	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25
T1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
T4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
T5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
T6	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
T7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
T8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
T9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
T10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 4.11: Replacement and verified criterion

4. The use of genetic algorithms and tabu search with task migration

Use the genetic algorithm to migrate tasks to reduce time and energy during execution next to the research table. This table contributes to the good direction of task migration by placing chromosomes that negatively affect the process of reducing tasks and energy to avoid dependence on them.

The use of the genetic algorithm next to the research table contributes greatly to the migration of tasks, where the genetic algorithm puts the possibility of moving through the processors in the form of chromosomes, and then systematically examine them to select possible solutions that facilitate the process of migration without resorting to random migration, It is placed at the level of the research table. If the mission wishes to migrate, it will be used to avoid wasting time and effort in moving through different treatments.

Algorithm TS and GA :

```
ALGORITHM 5.1 TS and GA
Population = Create new population;
FOR i ← 0 to 100 DO
Crossover(population) ;
Mutation (population);
FOR i ← 0 to 20 DO
fitness = Fitness_algorithem(population.chromosome);
IF(fitness >max ) THEN Add in tabo search;
END FOR
Selection (chromosome );
IF(chromosome existing in tabo search ) THEN remove it;
END FOR
END.
```

5. Experiments and case of studies

In our study, we used 3 graph model so that the first graph contains a random task so that the clock cycle is defined from 2 to 8 and the characteristics between 7 and 10.The higher the priority, the greater the priority in implementation. The token is limited between 1 and 4, and the second contains a MMS task so that the clock cycle is defined from 2 to 9 and the priority between 5 and 10.The higher the priority, the greater the priority in implementation. The token is limited between 1 and 3, and the last one contains a auto-indust task so that the clock cycle is defined from 1 to 7 and the priority between 3 and 10.The higher the priority, the greater the priority in implementation. The token is limited between 1 and 4. Graph contains a set of line linking each task to the other task.

The number of processors is 25 processors so that the voltage is limited to 1 to 4 volts. The bandwidth value is equal to 269873419 bits at every 5 second. Each link consumes 0.15 volts when transferring the lattice between the processors, so that the massage is limited between 1500 and 77398016 bits. When we use the genetic algorithm, we create a Pobelacio containing 20 chromosomes, so that the length of each chromosome equals the number of processors multiplied by the number of tasks.

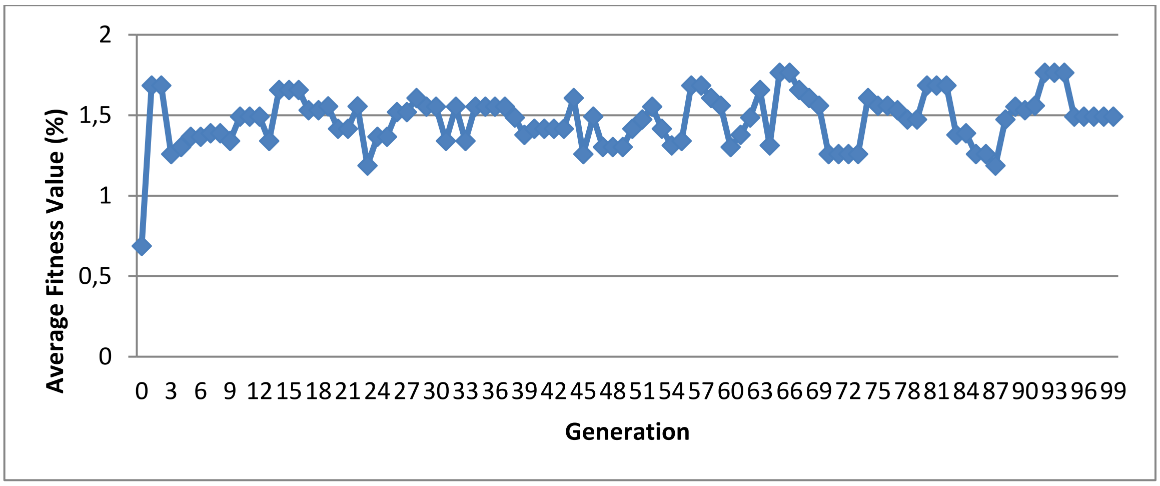


Figure 4.12:The average fitness value of each generation using GA in MMS.

In figure 4.12, we note that there is a difference between the values of fitness using a genetic algorithm in MMS. Note that there is a stability in the values of fitness in some fetal organs, because the value of average fitness in those areas is the same and if there is difference does not cause difference in the values of the curve and the sudden rise generation 2 indicates that the maximum value of the average values of attendance in this generation and the existence of a gap at the beginning of natural formation is normal because the tasks were not yet migrated and the effect was negative.

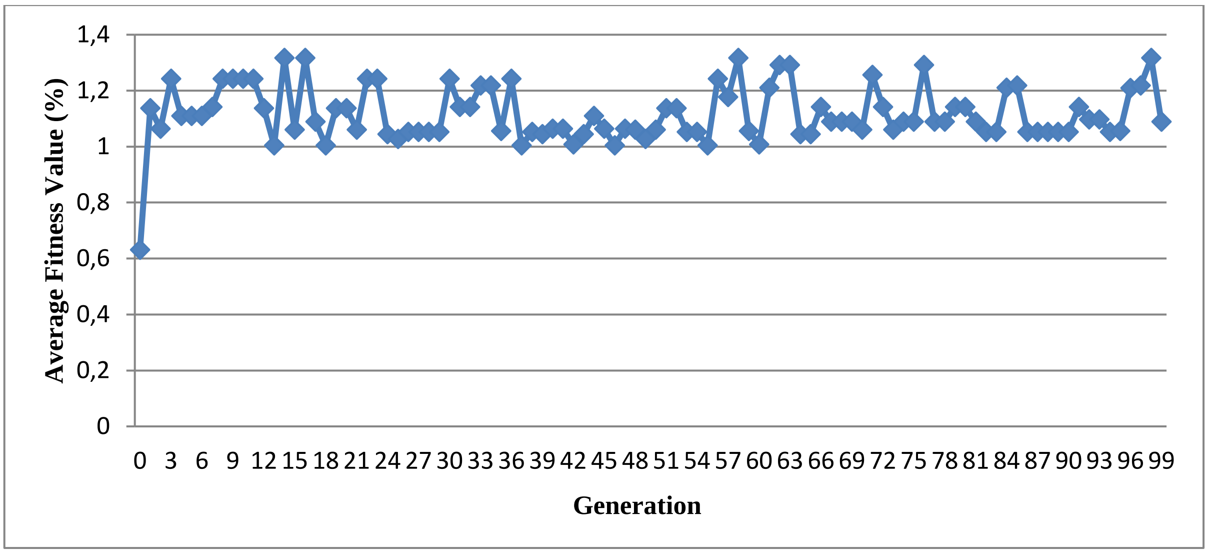


Figure 4.13:The average fitness value of each generation using GA in auto-indust.

In Figure 4.13, we observe convergence and stability in fitness values using GA in auto-indust. This stability is due to the selection of the same chromosome in the genes, a sudden increase in the value of the enzyme in generation 3 See for migration of all functions.

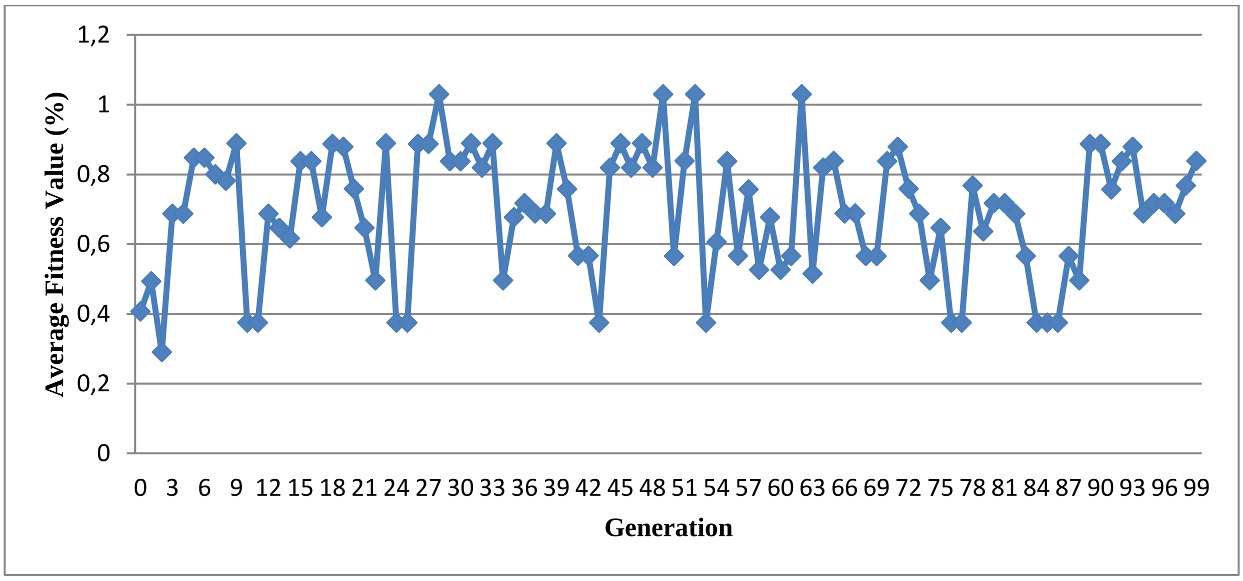


Figure 4.14: The average fitness value of each generation using GA in random task.

In Figure 4.14, we observe variation in the values The average fitnessusing GA in random task, where we observe a rapid rise in some fitness values and show that the values were better and there is a gap in some due to the negative values effect and stability in some of the same chromosome selection in it.

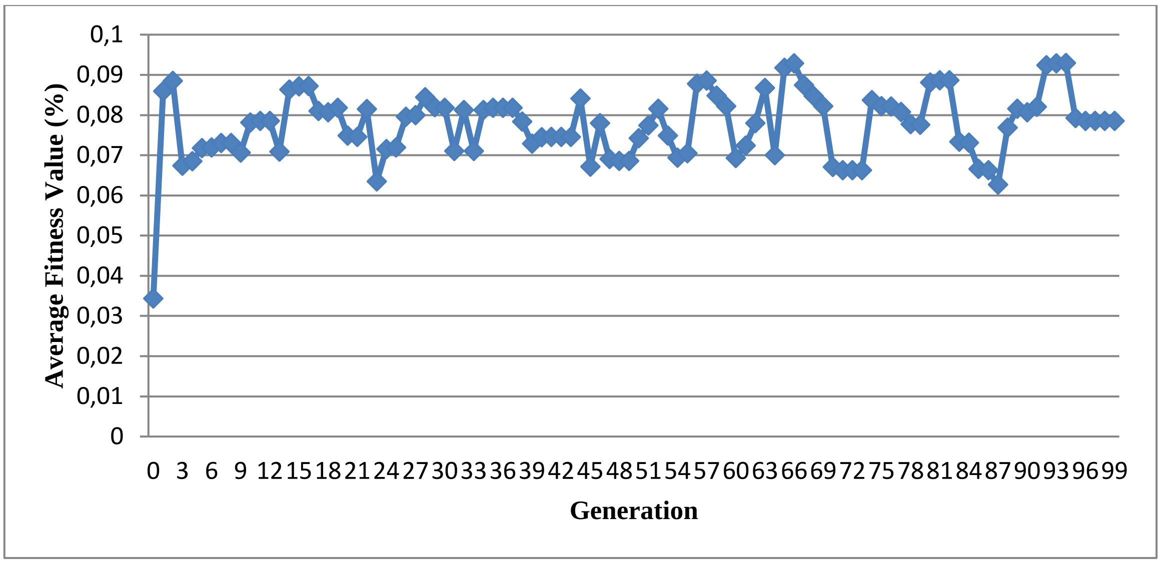


Figure 4.15:The average fitness value of each generation using GAand TS in MMS.

In Figure 4.15, we note the value of constant contrast at the generation end with a gap at the beginning of the generation. It is normal at the beginning of generation. The gap in the middle of this type is due to the negative effect of the specific chromosome

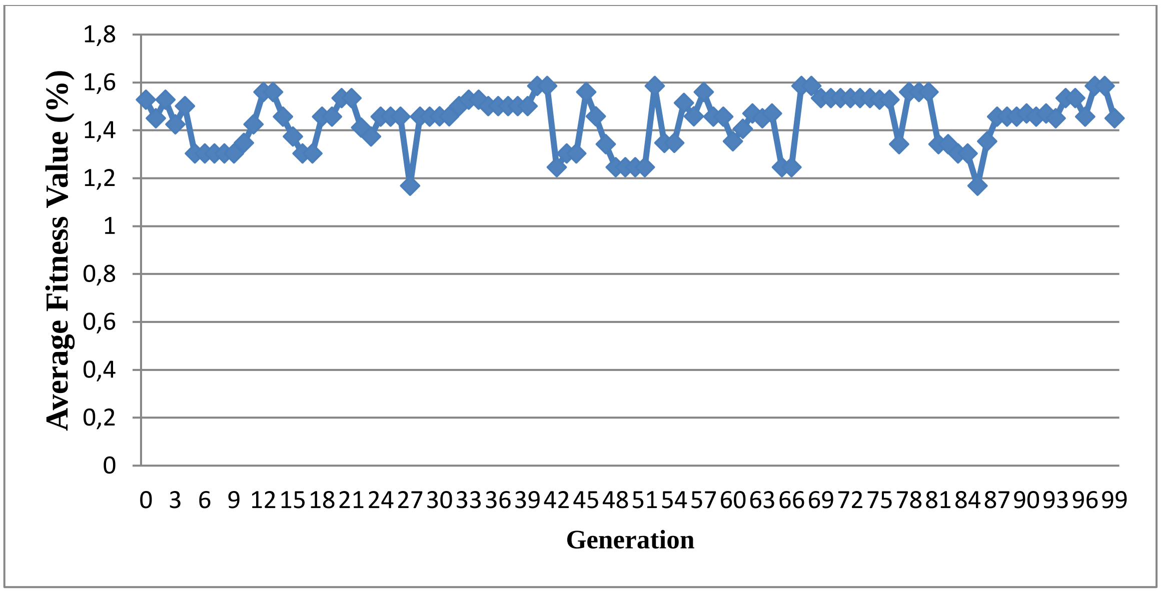


Figure 4.16:The average fitness value of each generation using GA and TS in auto-indust.

In Figure 4-16, the high fitness values at the beginning of the generation, to reduce the negative effect of the chromosome specified in this type of gene suddenly, to gradually return to height after removing the chromosome effect, the stability curve shows the same chromosome selection.

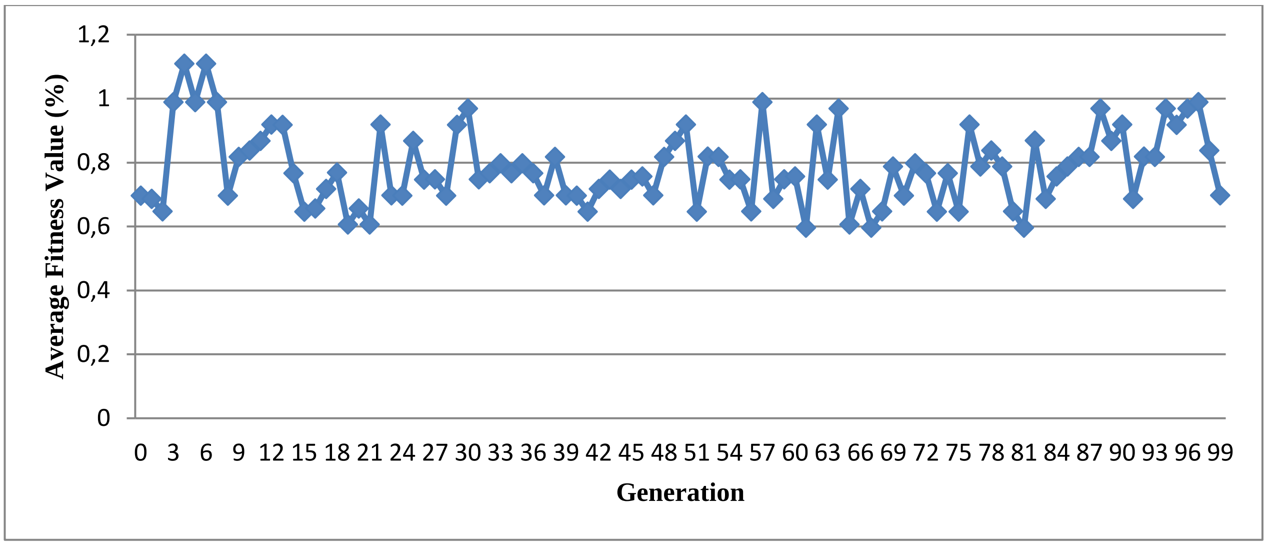


Figure 4.17:The average fitness value of each generation using GA and TS in random task.

Figure 4.17, we observe the variation in fitness values, which increases the real appearance of the positive effect of these chromosomes and gradually decreases without consistency in the fitness values of different generations.

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.

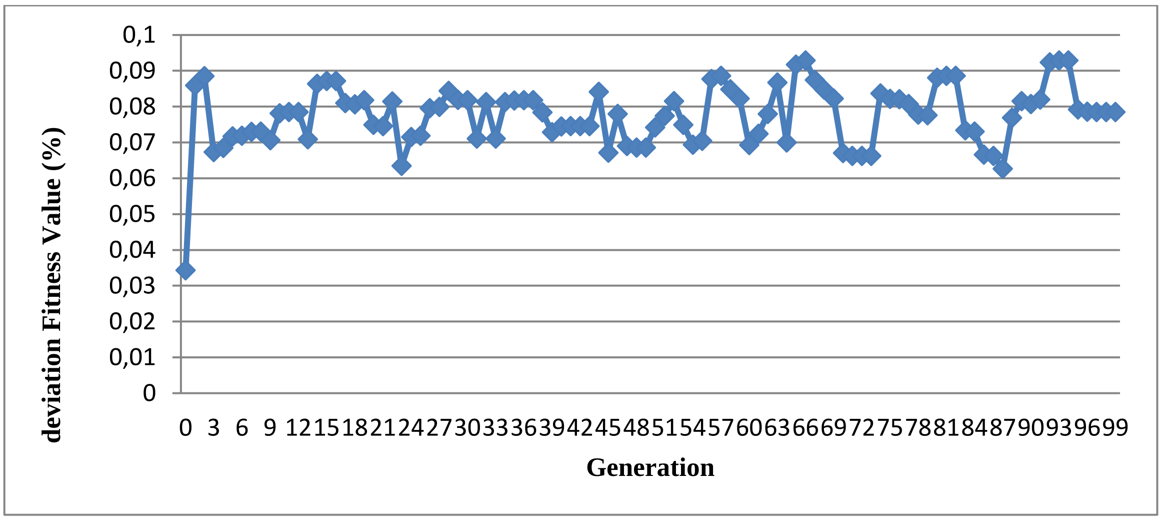


Figure 4.18:The deviation of each generation using GA in MMS.

In Figure 4.18, we note the value of constant contrast at the generation end with a gap at the beginning of the generation. It is normal at the beginning of generation. The gap in the middle of this type is due to the negative effect of the specific chromosome.

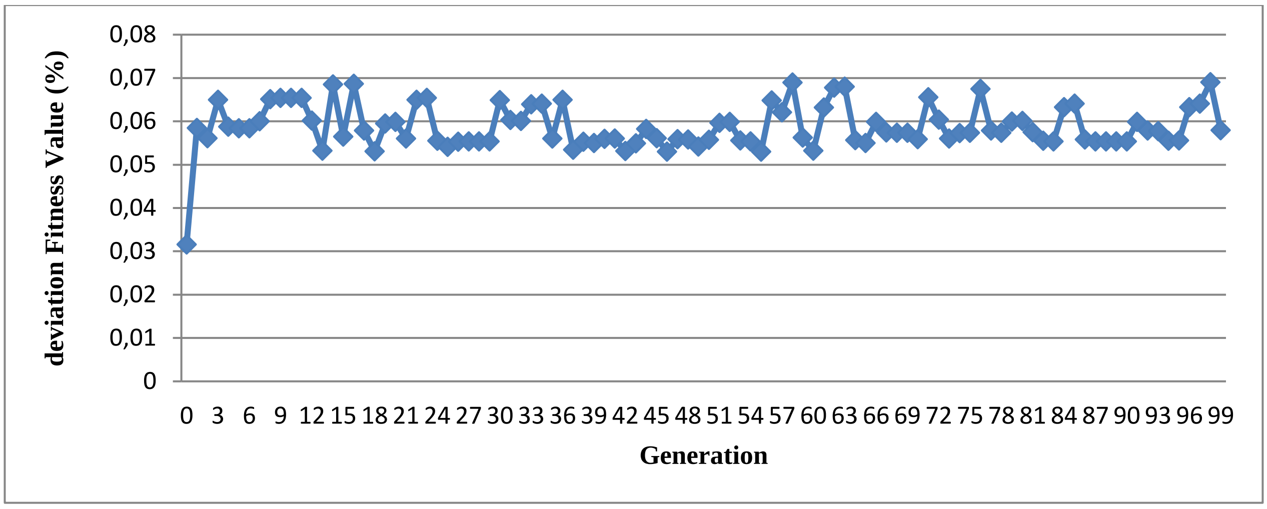


Figure 4.19:The deviation of each generation using GA and TS in MMS.

In Figure 4.19, we observe convergence and stability in fitness values using GA and TS in MMS. This stability is due to the selection of the same chromosome in the genes, a sudden increase in the value of the enzyme in generation 3 See for migration of all functions.

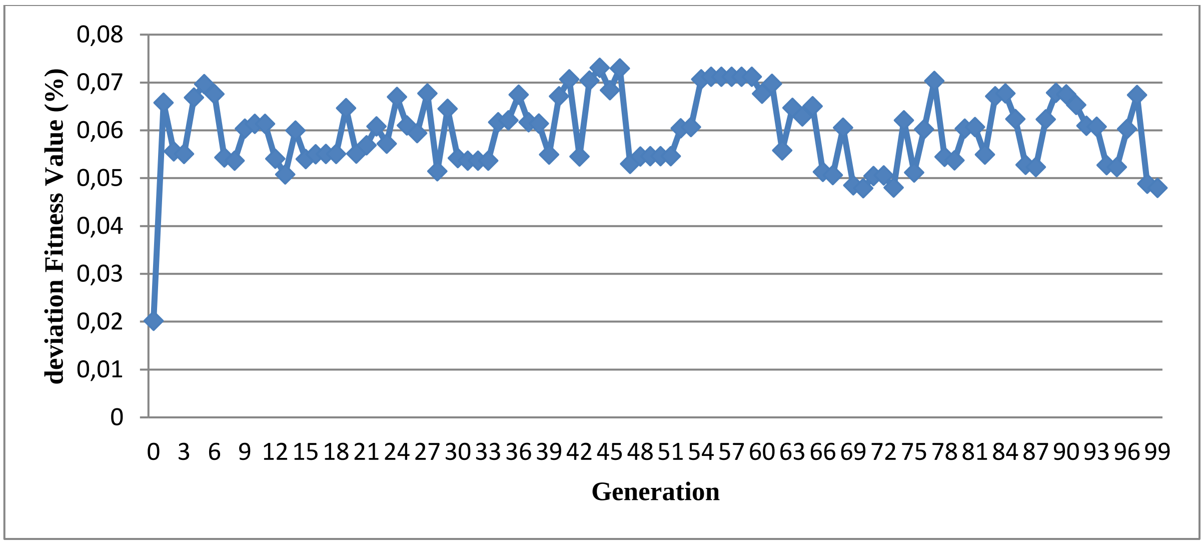


Figure 4.20:The deviation of each generation using GA in auto-indust.

Figure 4.20, we observe the variation in values, so that when the effect is positive the chromosome rises, the gap is when the chromosome effect is negative and stable when the chromosome is determined sequentially.

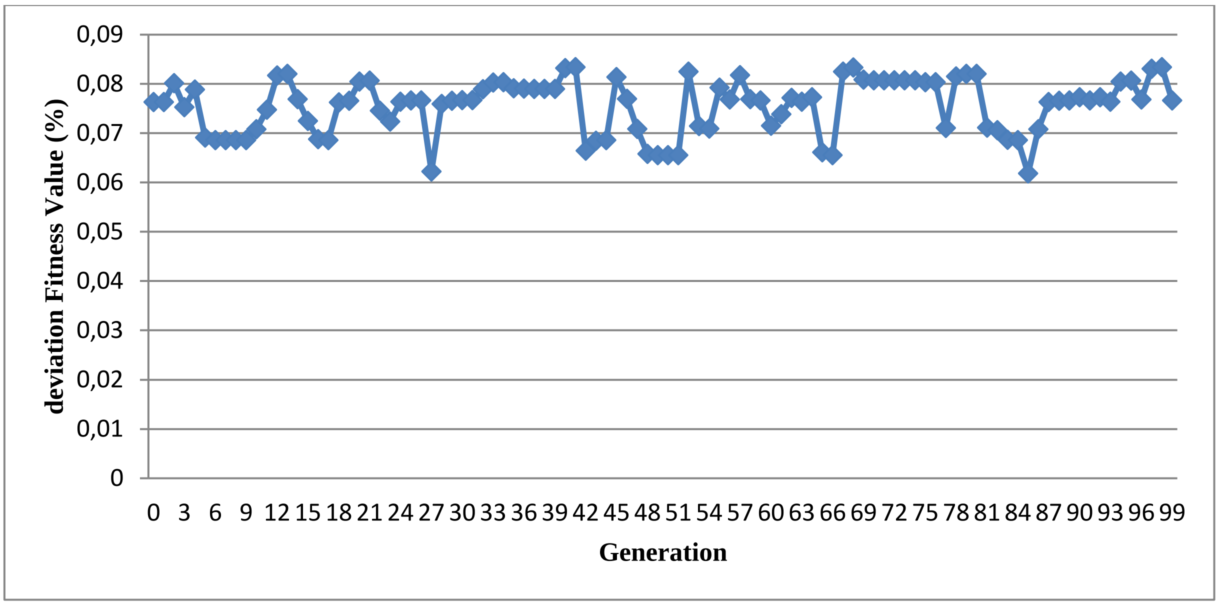


Figure 4.21:The deviation of each generation using GA and TS in auto-indust.

In Figure 4-21, the high fitness values at the beginning of the generation, to reduce the negative effect of the chromosome specified in this type of gene suddenly, to gradually return to height after removing the chromosome effect, the stability curve shows the same chromosome selection.

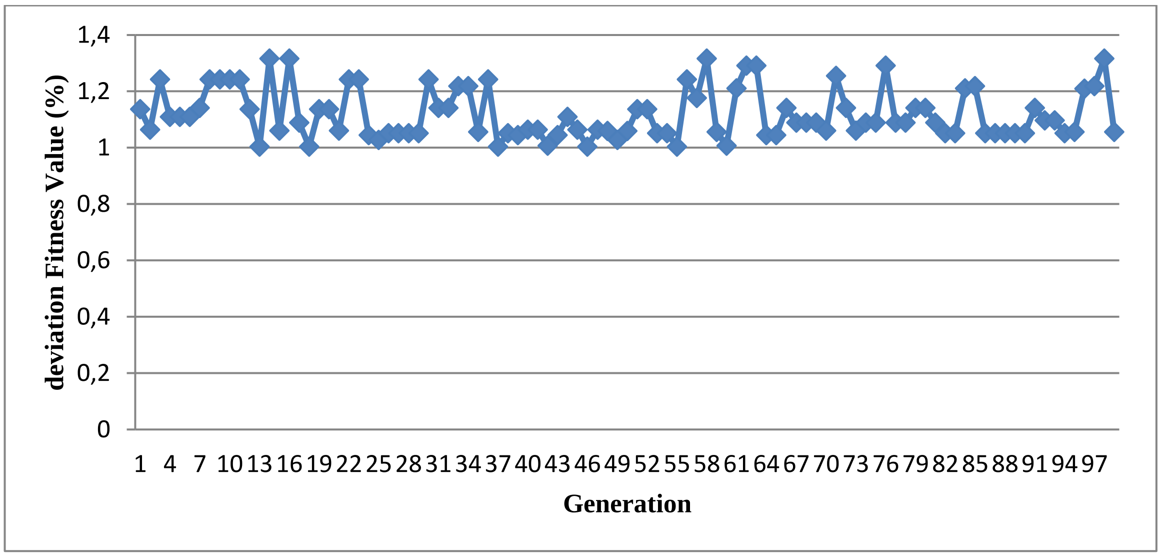


Figure 4.22:The deviation of each generation using GA in random task.

In figure 4.22 we observe a rapid rise in some generation due to the effective effect of the chromosome, and a rapid decrease is seen for the negative effect of the chromosome selector.

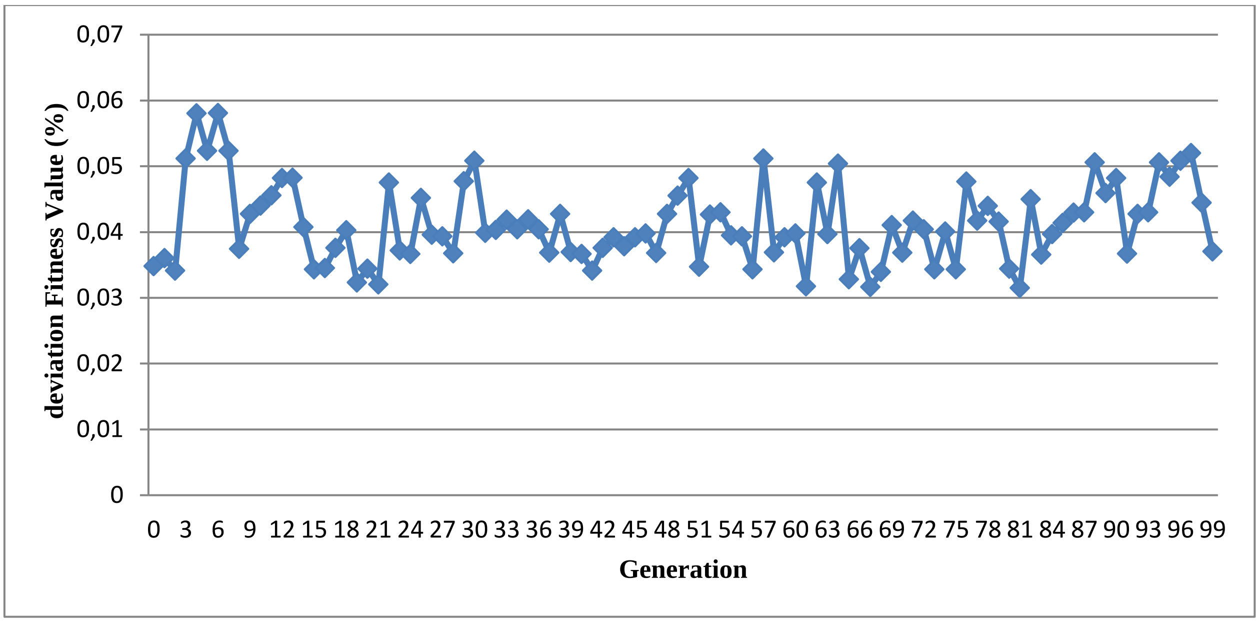


Figure 4.23:The deviation of each generation using GA and TS in random task.

In figure 4.23, we observe the variation in fitness values, which increases the real appearance of the positive effect of these chromosomes and gradually decreases without consistency in the fitness values of different generations.

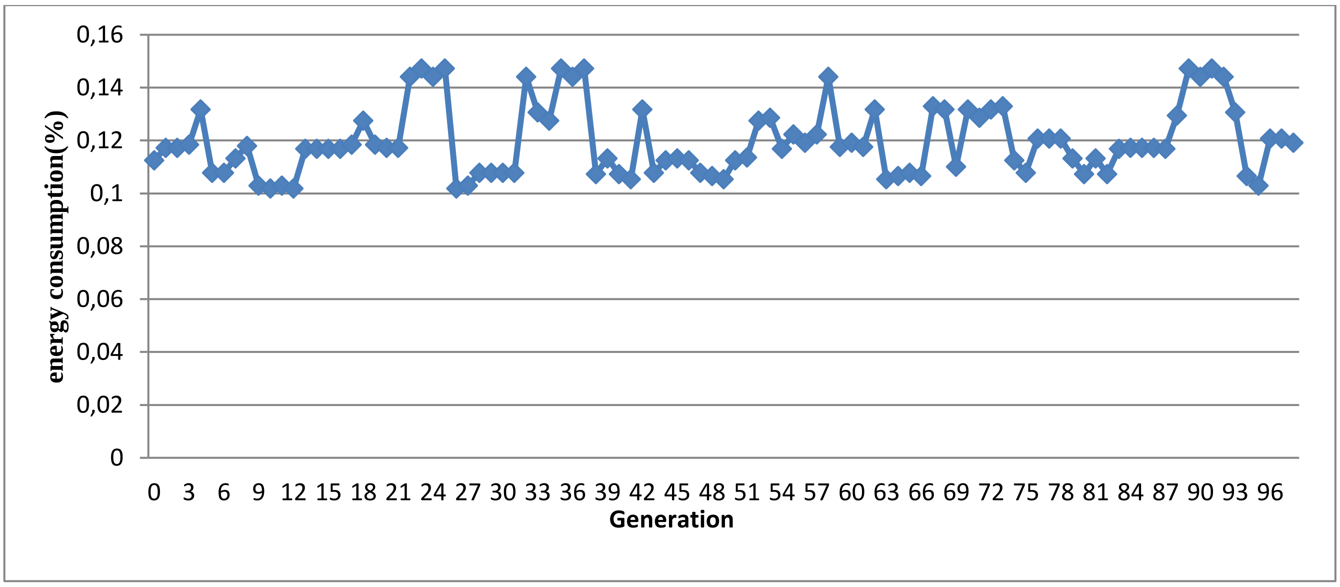


Figure 4.24:The energy consumption of executing MMS on 5x5 NoC with using task migration in GA and TS.

Figure 4.24, we observe a variation in proportions, where we observe the height of the value in some of the generation and decrease in some of them, where it reaches 0.1 in many geniuses and reaches the upper limit in the generation 26,34,3, 88

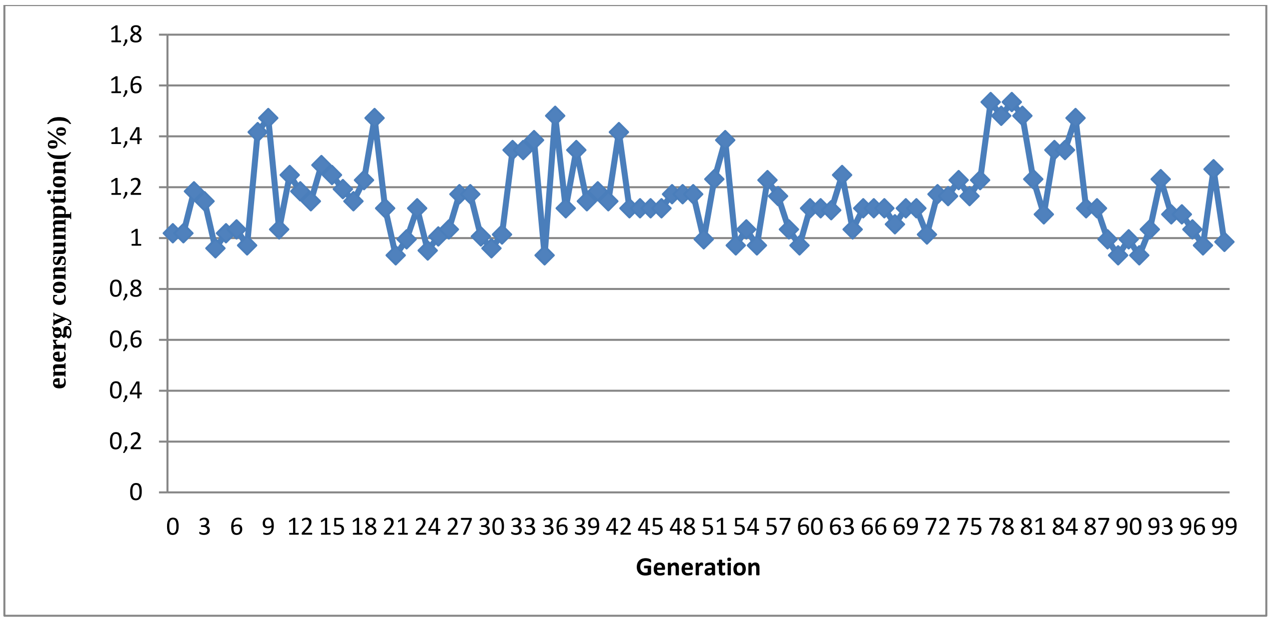


Figure 4.25:The energy consumption of executing MMS on 5x5 NoC with using task migration in GA

In figure 4.25, we note a disparity in energy consumption, with considerable energy consumption in the generation 76 which reached 1.59.

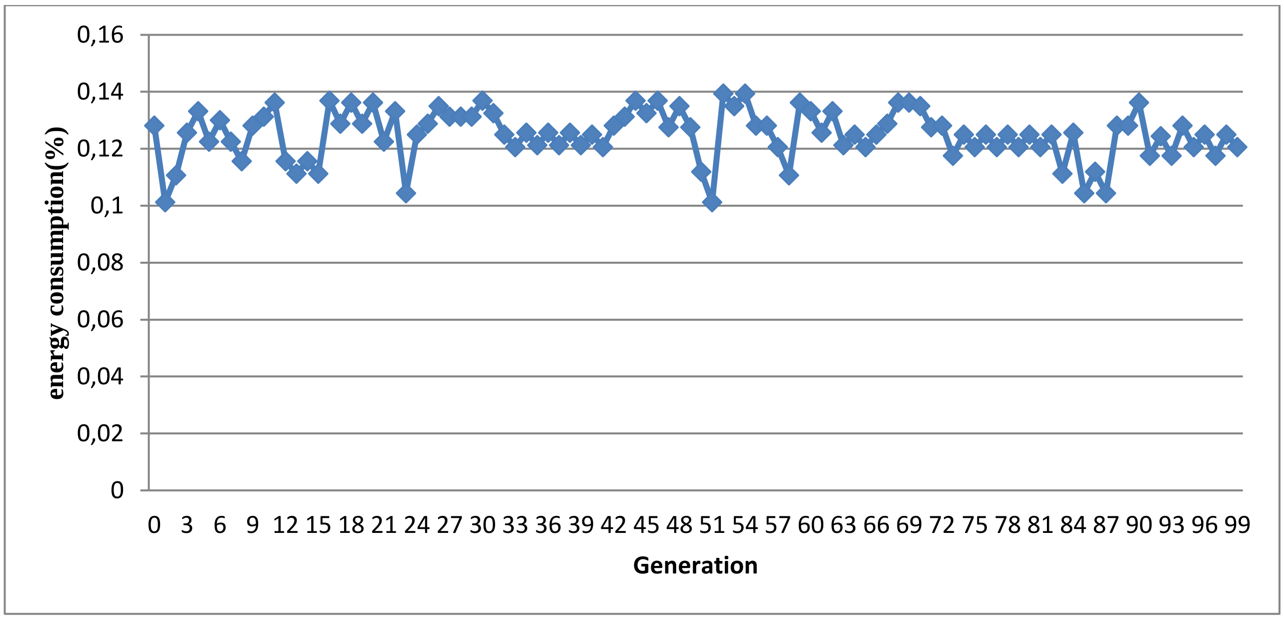


Figure 4.26:The energy consumption of executing auto-indust on 5x5 NoC with using task migration in GA and TS.

In figure 4.26, we note that the energy consumption does not exceed 0.14, to settle in some genotypes in 0.12 and the lowest value is 0.1. The rapid rise of the curve shows that energy consumption peaked at 51 generation.

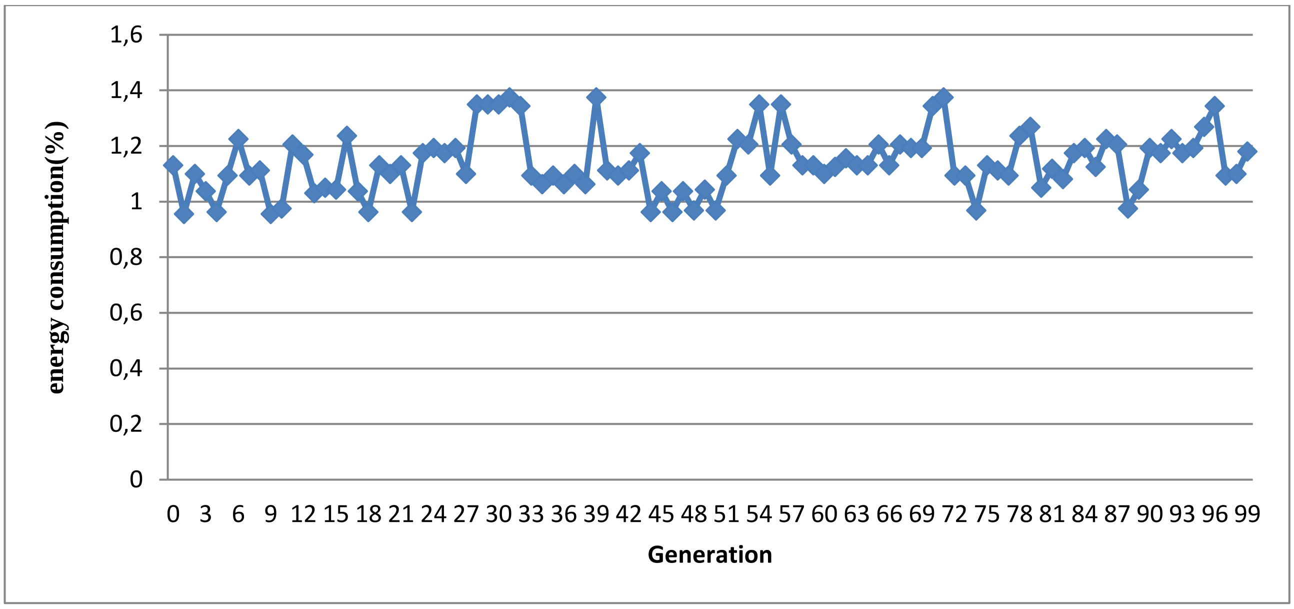


Figure 4.27 :The energy consumption of executing auto-indust on 5x5 NoC with using task migration in GA.

In figure 4.27, we observe a variation in energy consumption values, with power consumption reaching 1.4 in generations 32 38 70 and a minimum of 0.95 in some generations.

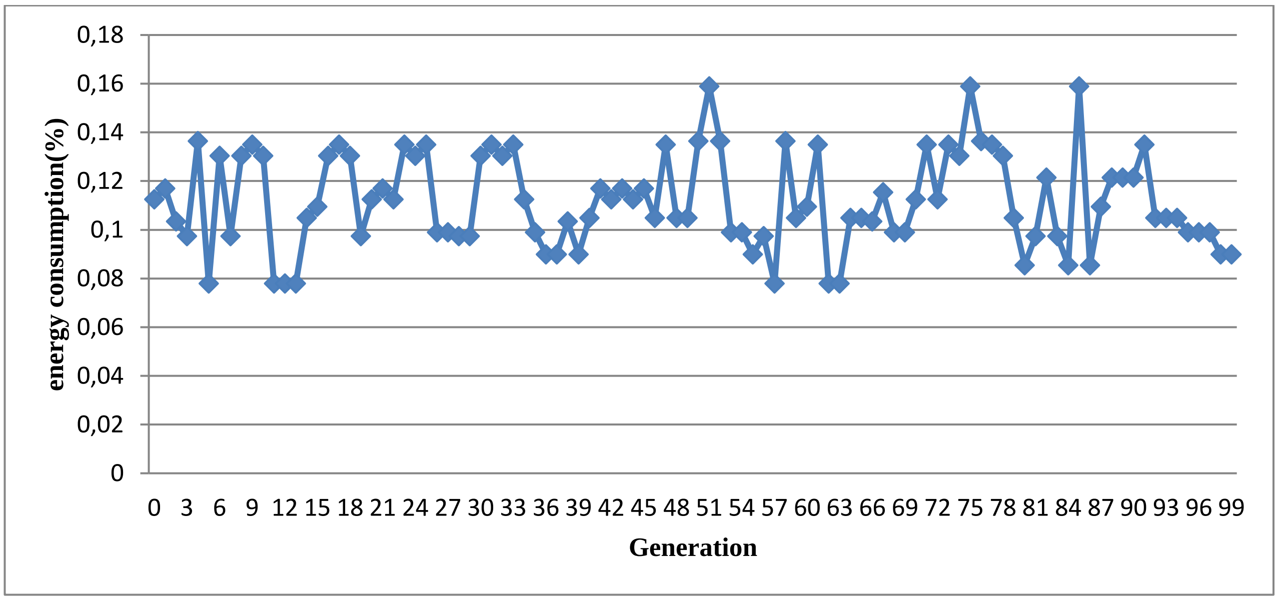


Figure 4.28 : The energy consumption of executing random task on 5x5 NoC with using task migration in GA and TS.

In figure 4.28, we observe variation in ratios where power consumption reached a peak of 0.16 in generation 75 and 84 to reach at least 0.08 in many generation.

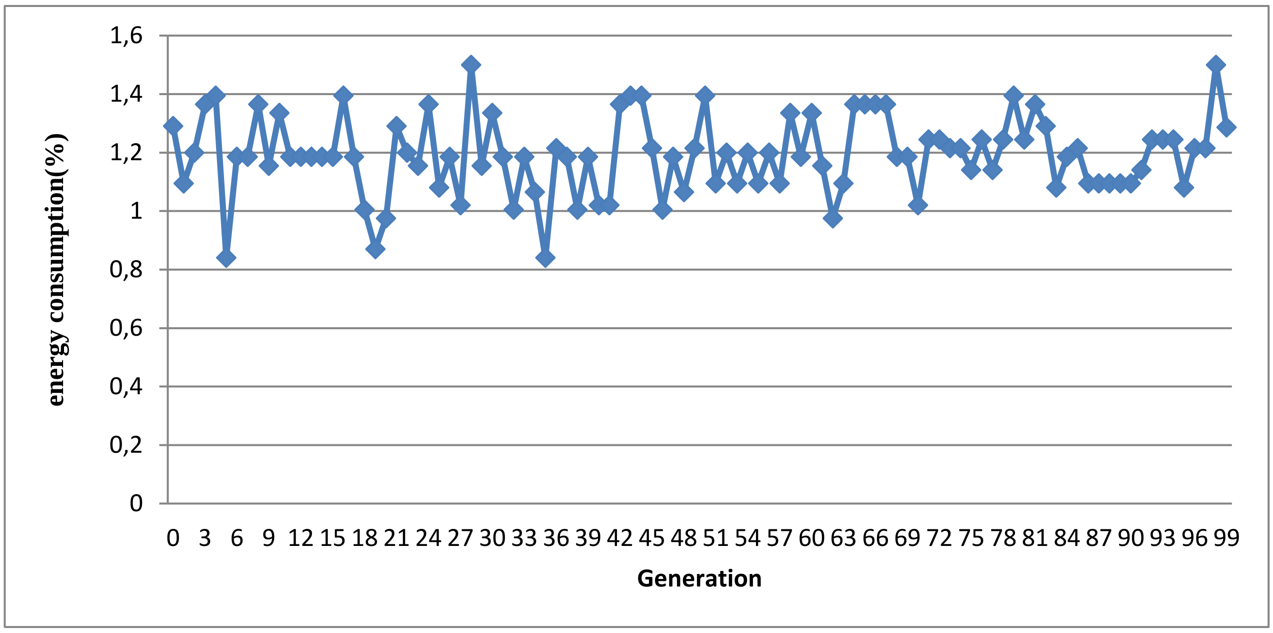


Figure 4.29 : The energy consumption of executing random task on 5x5 NoC with using task migration in GA .

In figure 4.29, we observed a difference in values of 0.8 in generation 5 35 61 and reached 1.55 in Generations 27 98 and stabilized at 1.5 from 82 to 92

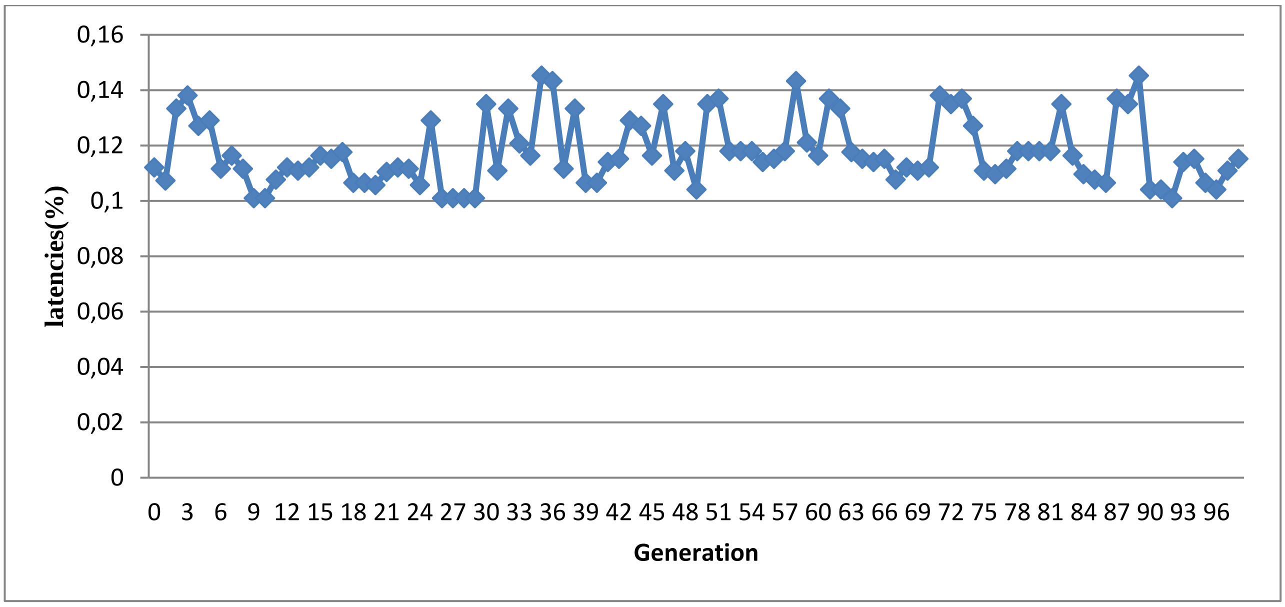


Figure 4.30: The latencies of executing MMS on 5x5 NoC with using task migration in GA.

In figure 4.30 we observe a difference in values where time consumption reaches, 0.15% at the maximum in Generation 34, 36 and 87 ,and 0.1% at a minimum to be stable from generation 24 to generation 30.

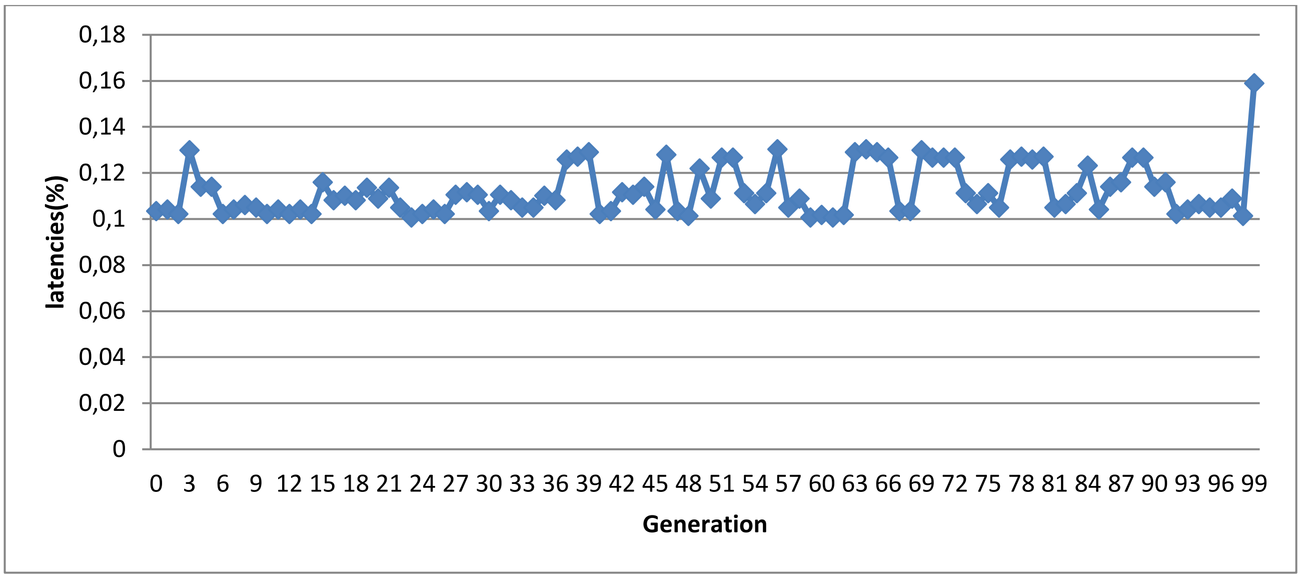


Figure 4.31: The latencies of executing MMS on 5x5 NoC with using task migration in GA and TS.

In figure 4.31, we notice that the consumption of the time has reached at the end of the generation to 0.16% as the maximum to be stable at the value of 0.1% in the different generation as the lowest value in energy consumption.

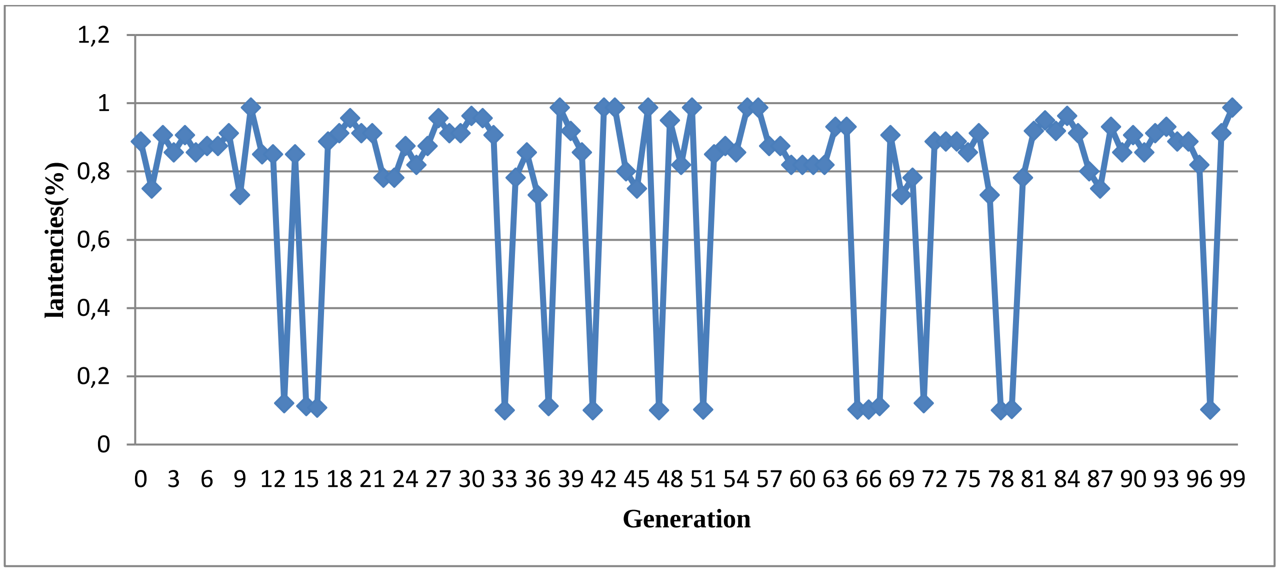


Figure 4.32: The latencies of executing auto-indust on 5x5 NoC with using task migration in GA and TS.

In figure 4.32, we observe that the consumption of time has reached 1 in most of the generation and reached to less than 0.2% in some and its stability at 0.8% of the generation 57 to 63.

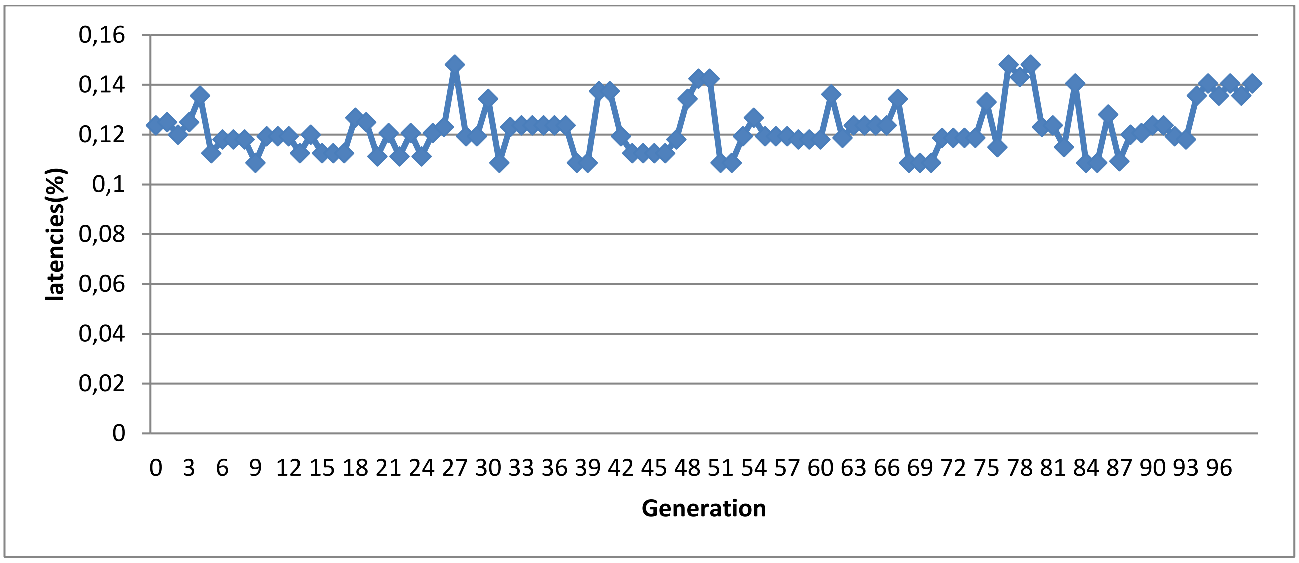


Figure 4.33: The latencies of executing auto-indust on 5x5 NoC with using task migration in GA .

In figure 4.33 ,we observe that the consumption of time across the Generation is less than 0.15% and the stability of its consumption in some of them in 0.11% with a difference in the rest of them.

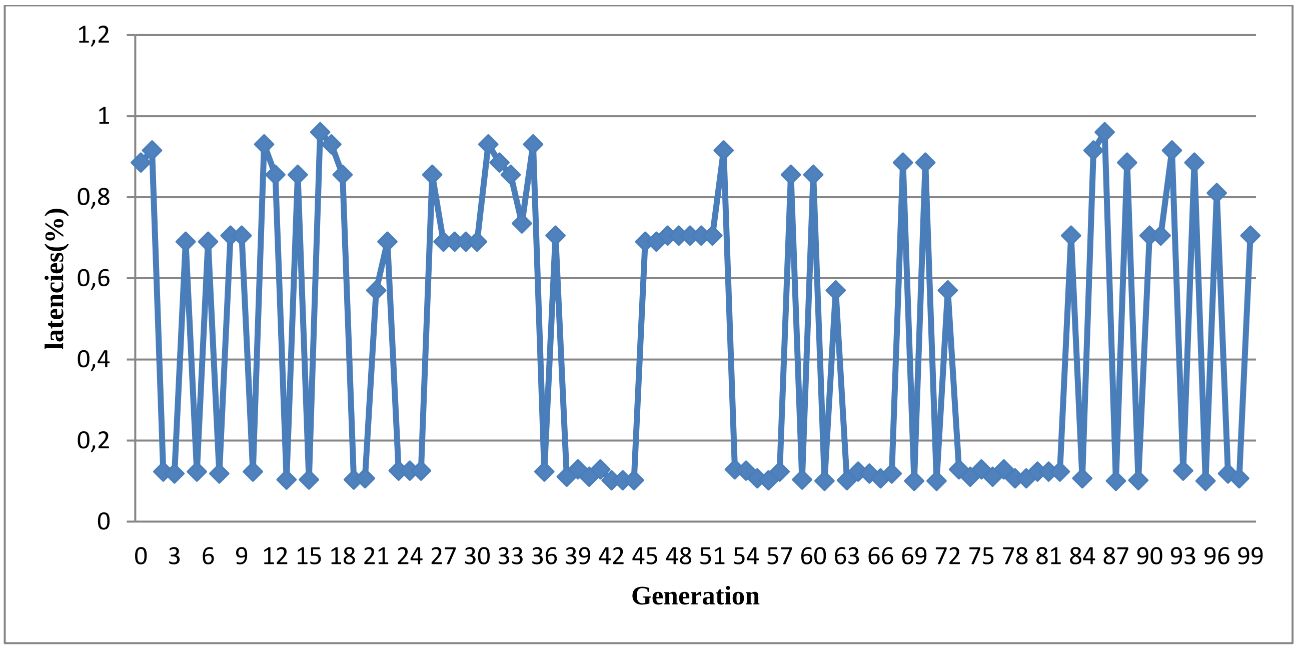


Figure 4.34: The latencies of executing random task on 5x5 NoC with using task migration in GA.

In figure 4.34, we observe a variation in the time consumption values constant across the Generation to a minimum of 0.1% and up to 0.99 %in some of them.

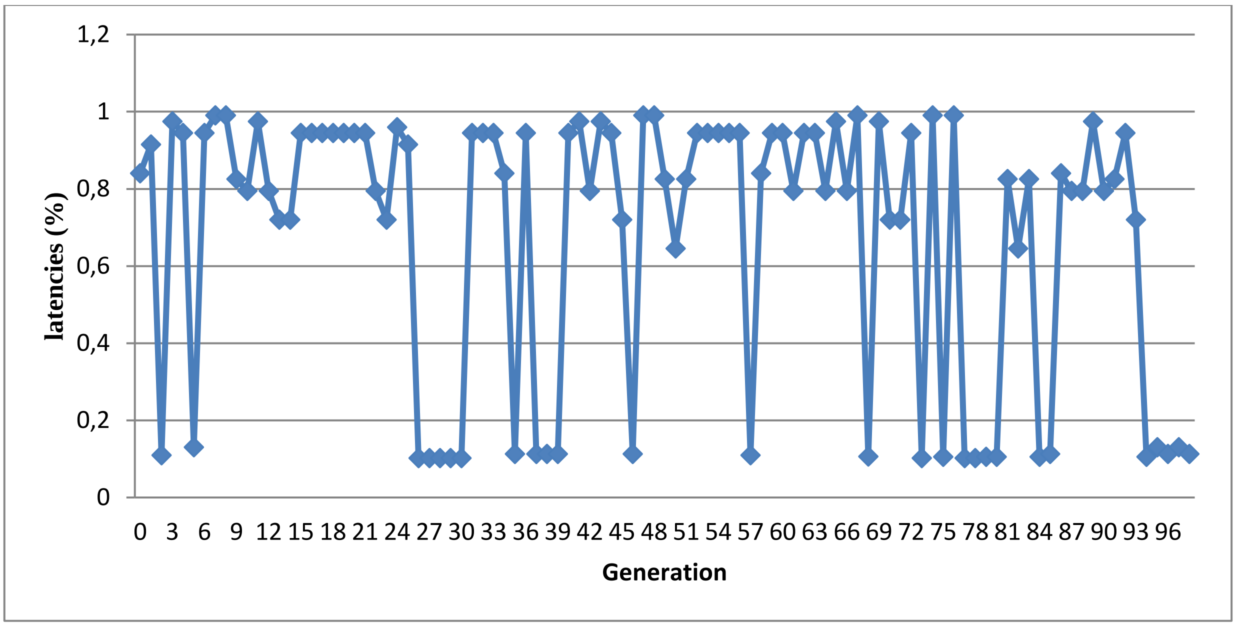


Figure 4.35: The latencies of executing random task on 5x5 NoC with using task migration in GA and TS.

In figure 4.35 ,we observe a variance in the values so that we observe that the consumption of time across the different generation was as large as 1% and the existence of an economy in consumption to reach 0.2%.

Through the curves, we noticed a difference in the values of MMS functionality, automatic manufacturing process and random task.

The average fitness value for each generation using GA in MMS and the automotive industry ranges between 0.035% and 1.8% compared to the average fitness value per generation using GA and TS in MMS and the automotive and randomization task, ranging from 0.035% to 0.095% .

The generation of GA in MMS, automatic dust, random task, the deviation of each generation using GA and TS in MMS, automatic dust and random task varies from 0.035 to 0.1 at each generation deviation using GA and the deviation for each generation using GA and TS between 0% and 0.085%

Also note the arrival of the value 0 at the deviation of each generation using GA in a random task in the generation number 76,Note the similarity of some curves

The results of overall energy consumption and latencies on GA andGA with tabu search are shown in the following table Table4.1 where where the energy consumption of MMS and auto-indust using GA is Confined between 30.23% and 31.99% , and using GA and TS Confined between 35.02 and 38.40%.The Latency minimization in MMS arrived to 20.66 and auto-indust arrived to 18.87 using GA, and arrived to 30.66 using GA and TS in MMS ,and 32.70 in Auto-indust.

	Energy consumption		**Latency**	
	MMS2	**Auto-indust**	**MMS2**	**Auto-indust**
GA	30.23%	31.99%	20.66%	18.87%
GA and TS	35.02%	38.40%	30.06%	32.70%

Table 4.1: Comparison between GA and TS**,** GA in executing mms2 and auto-indust.

6. Conclusion

In this chapter, we addressed the problem of power consumption and time through NOC processors using both GA and TS , to conclude that optimization algorithms such as GA and TS are one of the best solutions to solve any problem and that the good choice of these algorithms contributed significantly to reducing power consumption of processors and saving time during Perform tasks when you exchange these tasks between them.

GENERAL CONCLUSION

Our thesis is to solve optimize migration problems by improving crossover and potential for a boom using Tab Search. One of the problems we face in migration tasks is that the load time of the migration time is too long, resulting in increased network congestion and reduced system performance resulting in increased consumption power. So leave it next to it to save execution time. We directed this process, so that these tasks can not move randomly.

In the first phase of the improvement we used genetic algorithms to improve the performance of NoC such as energy consumption under the timing constraints, so that these algorithms are several stages of crossover and mutation, the most important is the selection phase and experimentation, which have a role in the choice of task location at the level of processors and then take The decision not to transfer, in the sense that the decision to migrate the current task or not. After applying a genetic algorithm to multimedia applications, we observed an improvement in energy consumption and latency in MMS2 by 30.23%, 20.66% sequentially, and auto-indust at 31.99% and 18.87% sequentially.

We have applied tabu research on both chromosomes; the stage of selection or the population stage (the creation of the first population), and each chromosome will be tested. If this chromosome increases energy consumption and latency (spoils improved fitness function values), it will be placed in the search table to avoid it in each future generation. This means that whenever we choose a chromosome before the experiment, we go to the search table to make sure it is not in the search table, and we get the following test results in MMS2. Energy consumption reached 35.02%, Latency 30.06% and Auto-indust improved. Energy consumption and Latency reached 38.40%, 32.70% on the sequential.

References

1. A. Leroy: Optimizing the on-chip communication architecture of low power systems on-chip in deep sub-micron technology, Université Libre de Bruxelles, 2006.
2. A. Mello, L. Tedesco, N. Calazans et F. Moraes: Virtual Channels in Networks on Chip, Implementation and Evaluation on Hermes NoC, chez in Proc. the Symposium on Integrated Circuits and Systems Design, 2005.
3. Abderrahim Chariete : Approches d'optimisation et de personnalisation des r ´eseaux sur puce (NoC : Networks on Chip), Thèse de Doctorat Universit ´e de Technologie de Belfort-Montb´ eliard,Sp´ ecialit ´e : Informatique, Soutenue le 11 mars 2014.
4. Agarwal, N. Dimopoulos, N,: High-level fsmd design and automated clock gating with codel, J. of Electrical and Computer Engineering, Canadian,(2008), 31–38.
5. Ahuja, S. Shukla, S. K,: MCBCG: Model Checking Based Sequential Clock-Gating, IEEE International workshop on High Level Design Validation and Test,(2009), 20-25.
6. Ahuja, S. Zhang, W. Lakshminarayana, A. Shukla, S.K,: A Methodology for Power Aware High-level Synthesis of Co-Processors from Software Algorithms, proceedings of International VLSI design Conference, India, (2010),282–287.
7. Ahuja, S. Zhang, W. Shukla, S. K,: System level simulation guided approach to improve the efficacy of clock-gating, IEEE International workshop on High Level Design Validation and Test, (2010).
8. Amit Kumar Singh , Thambipillai Srikanthan , Akash Kumar , Wu Jigang: Communication-aware heuristics for run-time task mapping on NoC-based MPSoC platforms, Journal of Systems Architecture 56 (2010) 242–255.
9. Amit Kumar Singh, Wu Jigang, Alok Prakash, Thambipillai Srikanthan: Efficient Heuristics for Minimizing Communication Overhead in NoC-based Heterogeneous MPSoC Platforms, Rapid System Prototyping 2009 IEEE/IFIP International Symposium on Rapid System Prototyping.
10. Assaderaghi, F. Sinitsky, D. Parke, S. Bokor,J. Ping, K. Ko, Hu, C,: Dynamic Threshold-Voltage MOSFET (DTMOS) for Ultra-Low Voltage VLSI, IEEE TRANSACTIONS ON ELECTRON DEVICES, VOL. 44, NO. 3, (1997).
11. Beigne, E. Clermidey, F. Lhermet, H. Miermont, S. Thonnart, Y. Tran, X. Valentin, A. Varreau, D. Vivet, P. Popon, X. Lebreton,H,: An asynchronous power aware and adaptive NOC based circuit, IEEE Journal of solid-state Circuits, Vol. 44, NO. 4, (2009), 1167 – 1177.
12. Bhavana Prakash Shrivastava, Kavita Khare: Areaand PowerEfficient Router Design for Network on Chip, International Journal of Scientific and Research Publications, Volume 3,Issue 5, May 2013 ISSN 2250-3153.
13. Brique Rose,Small Rose: System Temp-réel Embarqués.
14. Bruce, K.B., Cardelli, L., Pierce, B.C.: Comparing Object Encodings. In: Abadi, M., Ito, T.(eds.): Theoretical Aspects of Computer Software. Lecture Notes in Computer Science, Vol.1281 Springer-Verlag, Berlin Heidelberg New York (1997) 415–438.
15. 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.
16. C. Jueping, H. Gang, W. Shaoli, Y. Lei, L. Zan, H. Yue.: OPNEC-sim: An Efficient Simulation Tool for Network-on-Chip Communication and Energy Performance Analysis,10th IEEE International Conference on Solid- State and Integrated Circuit Technology (ICSICT), pp. 1892-1894, 1-4 Nov. 2010.

17. C. L. Liu and J. W. Layland,: Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, Journal of the ACM 20(1), pp. 40-61 (1973).
18. Chang, J., Pedram, M,: Energy Minimization Using Multiple Supply Voltages, No. 94, ISLPED Monterey CA USA, (1996).
19. Cheng, C,: Using a Voltage Domain Programmable Technique for Low-Power Management Cell-Based Design, J. Low Power Electron. Appl. (2011), 1, 303-326.
20. D. GaoMing, Z. DuoLi, Y. YongSheng, M. Liang, G. LuoFeng, S. YuKung et G.MingLun, : FPGA prototype design of Network on Chips, chez Anti-counterfeiting, Security and Identification, 2nd International Conference on, 2008.
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. Dawei Li, Jie" Energy-efficient contention-aware application mapping and scheduling on NoC-based MPSoCs:,J. Parallel Distrib. Comput. 96 (2016) 1–11.
23. Dr. Mahmood B. Ridha: Heuristic Methods As A Tool to Support Decision Making in Production Planning.
24. Dr. Rajib Kumar: Introduction To Genetic Algorithms, Bhattach arjya Professor Department of Civil Engineering IIT Guwahati, 24 April 2015.
25. E. Rijpkema, K. Goossen, A. Radulescu, J. Dielissen, P. van Meerbergen, J. Wielage et E. Waterlander: Trade Offs in the Design of a Router with both Guaranteed and Best- Effort Services for Networks on Chip, Computers and Digital Techniques, IEE Proceedings, vol.150, n° 15, pp. 294-302, 2003.
26. E. Rijpkema, K. Goossens, P. Wielage: A Router Architecture for Networks on Silicon, Proceedings of Progress 2001, 2ndWorkshop on Embedded Systems.
27. E.A. Carara, R.P. de Oliveira, N.L.V. Calazans, F.G. Moraes. HeMPS-A framework for NoC-Based MPSOC generation, IEEE International Symposium on Circuits and Systems,(ISCAS), pp. 1345-1348, 24-27 May 2009.
28. Eduard Fernandez-Alonso, David Castells-Rufas, Jaume Joven2 and Jordi Carrabina: Survey of NoC and Programming Models Proposals for MPSoC, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 3, March 2012,ISSN (Online): 1694-0814.
29. EDWARD A. LEE :Synchronoud Data flow, PROCEEDINGS of the IEEE, vol75, NO.9,septemper 1987.
30. F. G. Assia, Transaction Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems.Springer 2005.
31. F. Moraes, N. Clazans, A. Mello, L. Moller y L. Ost. "HERMES: an infraestructure for low area overhead packet-switching Networks on Chip," Elsevier, INTEGRATION, the VLSIjournal 38, pp.69-93, 2004.
32. F.G. Assia, Transaction Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems. Springer 2005.
33. FRED GLOVER: A:Tutorial, center for applid Artificial Intellegence Boulder,Colorado 80309-0419.
34. 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.
35. Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Reading: Addison-Wesley.
36. Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Reading: Addison-Wesley.
37. Guerre A, Ventroux N, David R, Merigot A: Hierarchical Network-on-Chip for Embedded Many-Core Architectures,Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS). pp. 189-196. 3-6 May 2010.

38. H. Moussa: Architectures des réseaux sur puce pour décodeurs canal multiprocesseurs, l'école nationale supérieure des télécommunications de Bretagne, Bretagne, 2009.
39. Hale, C. Boris, G. Keckler, W,: Segment Gating for Static Energy Reduction in Networks-On-Chip, (2009).
40. Mohammad Behrouzian Nejad, Ebrahim Behrouzian Nejad, Aref Sayahi, Sayed Mohsen Hashemi, Javad Chaharlang:Mapping and Scheduling Techniques for Network-on-Chip Architecture, International Journal of Basic Sciences & Applied Research.Vol.2 (7), 686-690, 2013.
41. Herbert, S. Garg, S. Marculescu, D,: Exploiting process variability in voltage/frequency con-trol, IEEE Transactions on VLSI systems, Vol. 20, No. 8, (2012), 1392-1404.
42. 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).
43. J. Duato, S. Yalamanchili, and L. Ni. Interconnection Networks: An Engineering Aproach.Morgan Kaufmann/Elsevier, 2003.
44. J. Duato, S. Yalamanchili, L. Ni. Interconnection Networks: an Engineering Approach. s.l.: Morgan Kauffman, 2002.
45. 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), (2007), 360 – 363.
46. JORGE MAGALHÃES-MENDES: A Comparative Study of Crossover Operators for Genetic Algorithms to Solve the Job Shop Scheduling Problem, Department of Civil Engineering, CIDEM ,School of Engineering – Polytechnic of Porto Rua Dr. António Bernardino de Almeida, 431 – 4200-072 Porto PORTUGAL.
47. K. Funaoka, S. Kato, and N. Yamasaki, "Energy-Efficient Optimal Real-Time Scheduling on Multiprocessors," in Proc. of the 11th IEEE International Symposium on Object/Component/Service-Oriented RealTime Distributed Computing, (2008).
48. Kinnear, K. E. (1994).A Perspective on the Work in this Book. In K. E. Kinnear (Ed.), Advances in Genetic Programming (pp. 3-17). Cambridge: MIT Press.
49. Kumar, V. Kim, S. Sapatnekar S,: Mathematically Assisted Adaptive Body Bias (ABB) for Temperature Compensation in Gigascale LSI Systems, Asia and South Pacific Conference on Design Automation, (2006).
50. Kuroda, T. Fujita, T. Nagamatu,T. Yoshioka, S. Sei, T. Matsuo, K. Hamura, Y. Mori, T.Murota, M. Kakumu, M. Sakurai, T. :High-speed low-power 0.3µm CMOS gate array with variable threshold voltage (VT) scheme, Proceedings IEEE Custom Integrated Circuits Con-ference, San Diego, CA, USA, (1996), 53-56.
51. Kyle C. Hale, Boris Grot, Stephen W. Keckler, : Segment Gating for Static Energy Reduction inNetworks-On-Chip, NoCArc New York, USA (2009).
52. L. Bononi et N. Concer: Simulation and analysis of network on ship architectures: Ring, Spidergon, and 2D Mesh, chez Proc. Automation and Test in Europe, 2006.
53. 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.
54. M. A. Siala, S. B. Saoud: A survey on existing MPSOCs architectures, International Journal of Computer Applications (0975 – 8887), vol. 19 (3), pp. 28-41, Abr2011.
55. M. Grange, A. Y. Weldezion, D. Pamunuwa, R. Weerasekera, Lu. Zhonghai, A.Jantsch, D. Shippen: Physical mapping and performance study of a multi-clock 3-Dimensional Network-on-Chip mesh," IEEE International Conference on 3D System Integration, pp. 1-7, 28-30 Sept. 2009.
56. M. R. Garey and D. S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness. New York, NY, USA: W. H. Free- man & Co., 1990.

57. M. Sacanamboy, F. Bolaños, and R. Nieto: A Primer for Mapping Techniques on NoC Systems.
58. Maad M. Mijwel: Heuristic Algorithms, Article · May 2015.
59. Maksat Atagoziyev: Routing Algorithms For On Chip Networks, In Partial Fulfillment Of The Requirements For The Degree Of Master Of Science In Electrical And Electronics Engineering, December 2007.
60. Malkowski, K. Raghavan, P. Kandemir, M. Irwin, M. J, : Phase-aware adaptive hardware selection for power-efficient scientific computation, International Symposium on Low Power Electronics and Design (ISPLED'07), (2007),403-406.
61. Manik Sharm: Role and Working of Genetic Algorithm in Computer Science, Article in International Journal of Computer Applications & Information Technology · December 2012.
62. Manjith, R. Muthukumari, C,: Dynamic Power Reduction in Sequential Circuits Using Look Ahead Clock Gating Technique, J of Elect and Com Eng Vol.9, No.2,(2015).
63. Marculescu, D,: Power Efficient Processors Using Multiple Supply Voltages, Workshop on Compilers and Operating Systems for Low Power, (2000).
64. Med Aymen SIALA, Slim BEN SAOUD: A Survey on Existing MPSOCs Architectures, International Journal of Computer Applications (0975 – 8887), Volume 19– No.3, April 2011.
65. Michael Litzkow and Marvin Solomon. Supporting checkpointing and process migration outside the unix kernel. In Usenix Winter Conference, pages 154–162.ACM Press/Addison-Wesley Publishing Co, 1992.
66. Mohammad Behrouzian Nejad, Ebrahim Behrouzian Nejad, Aref Sayahi, Sayed Mohsen Hashemi, Javad Chaharlang:Mapping and Scheduling Techniques for Network-on-Chip Architecture, International Journal of Basic Sciences & Applied Research.Vol.2 (7), 686-690, 2013.
67. Monire Norouzi: DEVELOPING TRENDS OF SYSTEM ON A CHIP AND EMBEDDED SYSTEM , An Open Access, Online International Journal Available 2014 Vol. 4 (S3), pp.715-719/Norouzi.
68. Morgenshtein, A,: Short-Circuit Power Reduction by Using High-Threshold Transistors, J. Low Power Electron. Appl, (2012), 69-78.
69. Naeini, M. Dass, S, Ooi, C:, The Design and Implementation of a Low-Power Gating Scan Element in 32/28 nm CMOS Technology, Journal of Low Power Electronics and Applications, Vol. 7, Iss. 7, (2017).
70. Narendhar Maaroju: CHOOSING THE BEST HEURISTIC FOR A NP-PROBLEM, Thesis submitted in partial fulfillment of the requirements for the award of Degree of Master of Engineering in Computer Science and Engineering, Thapar University, Patiala, JUNE 2009.
71. Neil C. Audsley: Deadline-Monotonic Scheduling, Department of Computer Science, University of York, Heslington,York. Y01 5DD, September 1990.
72. Nitasha Soni, Dr Tapas Kumar : Study of Various Mutation Operators in Genetic Algorithms, Lingaya's University, Faridabad, Nitasha Soni et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 4519-4521.
73. O. He, S. Dong, W. Jang, J. Bian, and D. Z. Pan: UNISM: Unified Scheduling and Mapping for General Networks on Chip,IEEE Trans. VLSI Syst., vol. 20, no. 8, pp. 1496–1509,2012.
74. Oklobdzija, Vojin G,: The Computer Engineering, Int standard book n.13, (2011).
75. Ou He, Sheqin Dong, Wooyoung Jang, JinianBian David Z. Pan: UNISM: Unified Scheduling and Mapping for General Networks on Chip, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 20, NO. 8, AUGUST 2012.
76. 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, Mar. 2000, pp. 250–256.
77. P. K. Sahu, S. Chattopadhyay: Survey on application mapping strategies for Network on Chip design,Journal of Systems Architecture, vol. 59, Issue 1, pp. 60-76, Jan 2013.
78. P. Mesidis. Mapping of Real-time Applications on Network-on-Chip based MPSOCS. Tesis de Maestría Universidad de York.2011
79. Pooja, S. Lakshay, S. Archana, K,: Clock Gating For Dynamic Power Reduction In Synchronous Circuits, int J of Sci Res Eng& Tec (2014) 3-4.
80. R. Lemaire, S. Thuries, F. Heiztmann, C. Helmstetter, P. Vivet, F. Clermidy: A flexible modeling environment for a NoC-based multicore architecture,” IEEE International, pp. 140-147, 9-10 Nov. 2012.
81. R.Hols mark et M.Hogder: Modeling and prototyping of Network On Chip, Sweden 2002.
82. Raj Kamal: Embedded System-Introduction, publes::McGraw_Hill Education 2008.
83. Ronny Pui-Hung Pau: A Configurable Router for Embedded Network-on-Chip Support in Field-Programmable Gate Arrays, A thesis submitted to the Department of Electrical and Computer Engineering in conformity with the requirements for the degree of Master of Science (Engineering).
84. S. Bertozzi, A. Acquaviva, D. Bertozzi, A. Poggiali, ―Supporting Task Migration in Multi-rocessor Systemsonchip: a Feasibility Study,‖ In: Design, Automation and Test in Europe Conference (DATE), pp. 15-20,Munich, Germany, on 2006.
85. S. Murali, G. De Micheli: SUNMAP: a tool for automatic topology selection and generation for NoCs, Design Automation Conference. pp. 914-919, 7-11 Jul 2004.
86. S. Mutoh ; T. Douseki ; Y. Matsuya ; T. Aoki ; S. Shigematsu ; J. Yamada, 1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS, IEEE Journal of Solid-State Circuits, Vol. 30 , Iss. 8, (1995). 847–854.
87. S. Saeidi, A. Khademzadeh, A. Mehran: SMAP: An intelligent mapping tool for Network on Chip,International Symposium on Signals, Circuits and Systems, vol. 1, pp. 1-4, 13-14 Jul 2007.
88. Sander Stuijk, TwanBasten, Marc Geilen, Amir Hossein Ghamarian and Bart Theelen,: Resource-efficient Routing and Scheduling of Time-constrainedNetwork-on-Chip Communication, Eindhoven University of Technology
89. Santanu Kundu,Santanu Chattopadhyay: Network-on-Chip The Next Generation of System-on-Chip Integration, CRC Press is an imprint of Taylor & Francis Group, an Informatique business, 2015 by Taylor & Francis Group, LLC.
90. Shuchi A. Parkhani1, Vaishali A. Tehre:A Survey of Different Topologies for Network-on-Chip Architecture, International journal of recent trends in engineering & researd.
91. Simon Holmbacka: TASK MIGRATION IN VIRTUALIZEDMULTI-CORE REAL-TIME SYSTEMS, Master of Science Thesis Supervisor: Johan Lilius, Department of Information Technologies Åbo Akademi University, September 2010 page 33-39.

92.Srivastava, A. Sylvester, D. Blaauw, D,: Power minimization using simultaneous gate sizing, dual-Vdd and dual-Vth assignment, the 41st ann Desi Aut Conf; San Diego, CA, USA,(2004), 783-787.
93.Srivastava, A. Zhang, C,:An adaptive body-bias generator for low voltage CMOS VLSI circuits, International Journal of Distributed Sensor Networks, Vol. 4, Iss. 2, (2008), 213-222.
94.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, EuropeanDesign and Automation Association, 2006.

95.Suvarna Patil, Manisha Bhende: Comparison and Analysis of Different Mutation Strategies to improve the Performance of Genetic Algorithm, Suvarna Patil et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5(3) , 2014, 4669-4673.

96.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.

97. Tiwari, R. Ranjan, R. Baig, M. Sravya, E,: Power gating technique for reducing leakage power in digital asynchronous GasP circuits, (2017), 23-24.

98. Tong Lin, Kwen-Siong Chong, Bah-HweeGwee, and J Chang. Finegrained power gating for leakage and short-circuit power reduction by using asynchronous-logic. IEEE ISCAS,(2009),3162 – 3165.

99. Tschanz, J. Kao, J. Narendra S. Nair, R, Member, Antoniadis, D. Chandrakasan, A. De, V,:Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter Varia-tions on Microprocessor Frequency and Leakage, IEEE Journal Of Solid-State Circuits, Vol.37, No. 11, (2002).

100.V. Nollet, P.Avasare, J-Y. Mignolet, and D. Verkest. Low cost task migration initiation in a heterogeneous mp-soc. In Proceedings of the conference on Design, Automation and Test in Europe, volume 1, pages 252–253. SIGDA: ACM Special Interest Group on Design Automation, IEEE Computer Society, 2005.

101.V. Zadrija, V. Sruk.: Mapping algorithms for MPSoC synthesis,Proceedings of the 33rd International Convention MIPRO, pp.624-629, 24-28 May 2010.

102.Valentina Zadrija,: Survey of Formal Models of Computation for Multi-Core Systems, University of Zagreb, Croatia, 03/31/2009.

103. Vemuru, S. R. Steinberg, N,: Short Circuit Power Dissipation Estimation for CMOS Logic Gates, IEEE Trans. on Circuits and Systems I, vol. 41, (1994), 762-765.

104.Ville Rantala, Teyolehtonen, Juha plopila: Network On Chip Routing Algorithms, TUCS Technical Report No 779, August 2006.

105.Ville Rantala,Teijo Lehtonen,Juha Plosila: Network on Chip Routing Algorithms, University of Turku, Department of Information Technology,Joukahaisenkatu 3-5 B, 20520 Turku,Finland. TUCS Technical Report No 779, August 2006

106.W. J. Dally and H. Aoki. Adaptive routing Using Virtual Channels. MIT Laboratory for Computer Science. 1990. Technical report.

107.W.J. Dally, B. Towles: Principles and Practices of Interconnection Networks. Morgan Kaufmann, 2004.

108.Wang, J. Qian, Y. Lu, J. Li, B. Zhu, M. Dou W:, Designing Voltage-Frequency Island Aware Power-Efficient NoC through Slack Optimization, Information Science and Applications(ICISA), (2014), 1 – 4

109. Wayne Wolf, Fellow, IEEE, Ahmed Amine Jerraya, and Grant Martin, Senior Member, IEEE : Multiprocessor System-on-Chip (MPSoC) Technology, OCTOBER 2008.

110.Wei, L. Chen, Z. Roy, K. Johnson, M, Ye,Y. K. De, V,: Design and optimization of dual-threshold circuits for low-voltage low-power applications, J IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 7, Iss. 1, (1999),16-24.

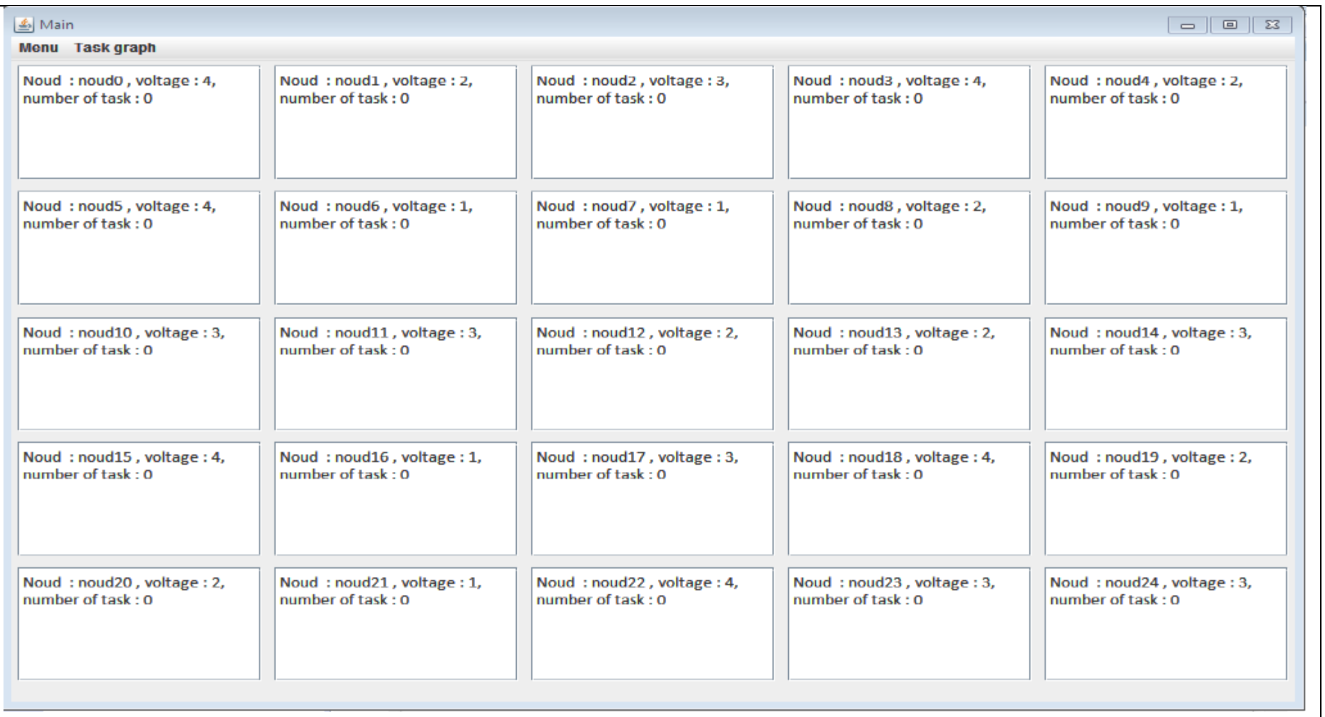
111. X. Wang, K. Ma, and Y. Wand, Adaptive power control with online model estimation for chip multiprocessors," IEEE Transactions on parallel and Distributed Systems, Vol. 22, No.10, (2011), 1681-1696.

112.Ying Wan ,Christine Shoemaker: Heuristic Methods for Optimization, August 24, 2011.

113.Yu-Yun, D,: Verification and Synthesis of Clock-Gated Circuits, California, Berkeley,(2017).

114.Zhan, J. Stoimenov, N. Ouyang, J. Thiele, L. Narayanan, V. Yuan, X,: 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 ,(2014), 1632 – 1643.



Noud : noud0 , voltage : 4, number of task : 0	Noud : noud1 , voltage : 2, number of task : 0	Noud : noud2 , voltage : 3, number of task : 0	Noud : noud3 , voltage : 4, number of task : 0	Noud : noud4 , voltage : 2, number of task : 0
Noud : noud5 , voltage : 4, number of task : 0	Noud : noud6 , voltage : 1, number of task : 0	Noud : noud7 , voltage : 1, number of task : 0	Noud : noud8 , voltage : 2, number of task : 0	Noud : noud9 , voltage : 1, number of task : 0
Noud : noud10 , voltage : 3, number of task : 0	Noud : noud11 , voltage : 3, number of task : 0	Noud : noud12 , voltage : 2, number of task : 0	Noud : noud13 , voltage : 2, number of task : 0	Noud : noud14 , voltage : 3, number of task : 0
Noud : noud15 , voltage : 4, number of task : 0	Noud : noud16 , voltage : 1, number of task : 0	Noud : noud17 , voltage : 3, number of task : 0	Noud : noud18 , voltage : 4, number of task : 0	Noud : noud19 , voltage : 2, number of task : 0
Noud : noud20 , voltage : 2, number of task : 0	Noud : noud21 , voltage : 1, number of task : 0	Noud : noud22 , voltage : 4, number of task : 0	Noud : noud23 , voltage : 3, number of task : 0	Noud : noud24 , voltage : 3, number of task : 0

Table A.1: Interface of application.

```
public static void distribution_tasks(Task table_task[],int nbr_task){
                                    Noud noud = null;
                                    String [] table_name_noud=new String [nbr_task];
                          for(int i=0; i<nbr_task ;i++){
                                   Task task =table_task[i];
                          int nbr_rondom=    min + (int)(Math.random() * ((max - min) + 1));
                          String name_noud="noud"+String.valueOf(nbr_rondom);
                          table_name_noud[i]=name_noud;
                            for(int j=0; j<25 ;j++){
                                      noud =architecteur.noud[j];
                                     String get_name =noud.get_name();
                                     if(get_name.equals(name_noud)){
                                                 int h=0;
                                                 for(int k=0; k<i ;k++){
                                                             if(get_name.equals(table_name_noud[k])){
                                                                         h=h+1;
                                                             }}
                                                 noud.task[h]=task;
                                     }}}}
```

Table A.2:code source of distribution_tasks.

```
public static void genetic_algorithem_tabusearch(Task[] task,Population population,Line[] line){
                              int min_genes=((task.length*25)/2), max_genes=(task.length*25);
                        int nbr_rondom = min_genes + (int)(Math.random() * ((max_genes - min_genes) + 1));
                                 creat_populations(task,task.length, population,line);
                                 i++;
                                 int index=0;
                                 float max=table_fitess[0];
                                 for(int j=0;j<20;j++){
                                            if(max<table_fitess[j] && table_fitess[j]!=0.0){
                                                          index=j;
                                            }}
                                for(int j=0;j<population.chromosome.length;j++){
                                              if(j==index){
                                    table_Search.add(population.chromosome[j]);
                                              }}
                                 list_enery.clear();
                                            list_time.clear();
                                for(i=1;i<100;i++){
                                 crosver(population,task);
                                 mutation(population,nbr_rondom);
                                 test(task,population,line);
                                selection_tabusearch(task,population,line);
                              }}
```

Table A.3:code source of genetic_algorithem_tabusearch.

Annex B: conferences papers :

1. Djalila Belkebir, Benazia Meriem and Kemassi Imane,: Power Reduction Techniques on Network on Chip, Conference on Electrical Engineering (CEE 2019) 22, 23 April 2019, EMP, Algeria.
2. Benazia Meriem and Kemassi Imane ,Djalila Belkebir,: Network on Chip Power Reduction Technique Based-physical level,Conference on Informatic and Appliad Mathematics 12,13 june 2019,Guelma,Algeria.