TALLINNA UNIVERSITY OF TECHNOLOGY
Information technology department
Computer science institute

Jaanus Alnek 107361

Evaluation of the impact of the number of TSVs in 3D NoCs

Master Thesis

Advisor:
Gert Jervan
Professor / Ph. D

Tallinn 2013

I hereby declare that this master thesis, my original investigation and achievement, submitted for the master's degree at Tallinn University of Technology, has not been submitted for any degree or examination. I have made the presented thesis myself and solely with the aid of the means permitted by the examination regulations of the Tallinn University of Technology. The literature used is indicated in the used literature.

(Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.)

_____

Jaanus Alnek

List of used abbrevations and terms

IC	Integrated Circuit
IP	Intellectual Property
SoC	System on Chip
NoC	Network on Chip
VLSI	Very Large-Scale Integration
FEOL	Front-End-Of-Line
BEOL	Back-End-Of-Line
TSV	Through Silicon Via
FPGA	Field Programmable Gate Array
TAM	Traffic Assessment Method
SIM	Simulated Annealing

Abstract

Technological evolution has made great progress in circuit design for several past decades. However in the recent years it has become apparent that we are reaching the pinnacle of our current methods of technological progress scaling down dimensions as we have hit the physical limitation barrier. Many new research directions have branched off looking for new ways to continue the legacy of Gordon Moore, to keep technological advancement from slowing down. In the recent years research into 3D stacking technology has shown great results over current technology and is believed to be the successor of the current generation of integrated chip design.

New technology requires new evaluation tools for development and efficiency assessment, therefore the objective is create a target system simulation and define a method to find tradeoffs points between systems configurations using different number of vertical interconnections. Research into evaluation methods and tools is necessary to design and improve new technologies. In this paper a method will be proposed to evaluate TSV impact in a 3D NoC platform.

Annotatsioon

(Magistri töö teema nimetus – TSVde arvu mõju hindamine 3D kiipvõrkudes)

Tehnoloogia evolutsioon on viimaste aastakümnete jooksul teinud suuri edusamme ja ei ole veel pidama jäänud. Viimastel aastatel on selgeks saanud tõsiasi, et oleme jõudnud tehnoloogia mõõtmete vähendamise füüsilistele piiridele praeguse tehnoloogia mõõtmetes. Palju uusi uurimis suundi on alustatud, otsides uusi meetmeid Gordon Moore-i ennustuste jätkamiseks, et tehnoloogia areng ei aeglustuks.3D kiip-virnastus tehnoloogia on näidanud paremaid tulemusi hetkese tehnoloogia suhtes ja seda peetakse uueks kiipsüsteemide tehnoloogia järglaseks. Ühendades kiipvõrgud 3D tehnoloogiaga on võimalik luua väga erinevate otstarvetega platvorme mis on pindalalt väiksemad ja tarbivad seetõttu vähem energiat. Uus tehnoloogia ei tule ilma uute probleemideta nii disaini kui tootmis protsessides.

Uued tehnoloogiad vajavad uusi arendusmeetmeid ja hindamisvahendeid. Ülesandeks on koostada uue platvormi simulatsioon koostamine ja selle abil leida meetod hindamaks kompromisse platvormi konfiguratsioonide vahel, mis sisladavad erinevaid TSV-de arve. Selles töös pööratakse peamiselt tähelepanu kiipvõrkudele ja 3D kiip-virnastus tehnoloogiale, tutvustades nende kujunemislugu ja kirjeldatakse nende ehitust ja tootmist viise. Töö viimases peatükis pakutakse välja uus meetod, leidmaks vertikaal ühenduste arvu mõju 3D kiipvõrkude platvormidel, mis kasutavad TSV tehnoloogiat.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 67 leheküljel, 4 peatükki, 16 joonist ja 4 tabelit.

Contents

Contents of tables

Contents of Figures

1. Introduction

Intel's co-founder Gordon Moore published a paper in 1965 [1], in which he noted that the number of transistor on a square inch (2.54cm) doubled every year after the invention of Integrated Circuits (ICs) and predicted this trend to continue for a minimum of 10 subsequent years, which was later called Moore's law. Moore's initial prediction was later corrected and would state that the number of transistors would double every 18months as the initial boost from the technological revolution had calmed down. Recent technological advances have begun to deviate from this ideal scaling theory - standard technological solutions have been hampered by severe growth limitations resulting from the physical limitations of materials used, and are therefore no longer sufficient to keep up with the ever-growing performance requirements that are expected from the industry. The main cause is difficulty of voltage scaling [2]. Temperature difference on energy transition between different metals does not scale down, making it hard to lower threshold voltage of a MOS transistor without increasing sub-threshold leakage, noise and heat. Without threshold voltage scaling, power and performance has become a tradeoff. Two methods have been used to increase a chip's performance – the first method changes the circuit and system architecture from power consumption point of view, the other method changes integration structure to lower the wiring length and pin capacitance. Many reasons, such as increased application complexity, physical limitations and computational requirements, paved the road to system that used more than a single processing unit. Multiple processing units on a single chip, later designated multi-core processors, are processors that house multiple processing units on the same silicon wafer with shared memory and peripherals.
The objective of the thesis is to propose a new method to evaluate the impact of vertical interconnections in upcoming 3D stacked Network-on-Chip systems (NoC).

A method will be proposed that is able to assess the impact of vertical interconnections in 3D NoC systems. The new method will have comparatively short runtime that scales a linear way towards systems with larger search space. For comparison purposes, Simulated Annealing will also be implemented as a secondary algorithm.

1.1. Overview

SoC and NoC technologies

System on Chip (SoC) is an IC that incorporates all electronic components required to implement an entire system on a single chip. Using this method often integrates several types of signals such as analog, digital, mixed-signals and even radio frequency functions. Typical application example of this technology is an IC embedded into other systems, where chip area is in most cases quite limited and power consumption should be as low as possible. The increase in complexity of computer systems has also increased the number of challenges we meet in the design and manufacture of technology. Several drawbacks still exist, using SoC technology such as processing core placement and count, wire arbitration, global wiring related delays and so on. The first SoC evolved from a wrist computer prototype [2], 4000 bonding wires connecting 44 circuits, which was simply too unreliable due to the fragmented timekeeping circuitry. The solution was later redesigned onto a single chip, however due to the high power consumption of a display using light emitting diodes no advances were made until Liquid Crystal Displays (LCD) were invented in 1973.The first true SoC solution appeared in 1974 inside the form of a digital watch invented by Peter Stoll in which he integrated an LCD driver interface with timing functions onto a single CMOS chip. In the 1990s many Application Specific Integrated Circuit vendors started embedding microcontrollers and digital signal processing units into single system-level chips which gave way to a wave of new application areas such as hand held products, data communications and peripheral products.

Communication plays a crucial role in the design and performance of SoC systems and will become more important when more processing units are implemented. Traditionally Integrated Chips(IC) use dedicated point-to-point links for each signal, however the given method however does not scale well for very large circuits on account of physical aspects such as increasing chip area occupied by wiring, wire length and resulting complications in signal propagation. Bus architecture have become inefficient when it comes to systems with large number of Intellectual Property (IP) modules and intense parallel communication, as

it may not meet the performance requirements needed by large applications. Network on Chip (NoC) has been deemed a good solution to simplify and optimize such designs as they continue to grow with each new processor generation. NoC technologies were designed to lower overall system complexity and increase performance by redesigning component and wiring layout along with communication methods.

Introduction to 3D stacking technology

During the last two decades three-dimensional large scale integration (3D LSI) technology has become a viable alternative to traditional circuit design. The promise of 3-D IC technology lies in the numerous benefits it can potentially provide to increase performance and response speed of electronic devices over traditional 2-D ICs designs such as the integration of different signals, analog digital and even radio frequencies, or a method to overcome current design tradeoffs due to physical limitations between flexibility system performance physical dimension and cost. 3D technology principle divides a planar chip into blocks, where each block can be stacked on top of the previous one. By allowing chips to grow in the vertical dimension instead of requiring larger die area, higher packing density and smaller footprint can be achieved. Generally 3D-IC contain multiple layers of active devices that extensively utilize the vertical dimension by utilizing shorter wires as interconnections between layers to connect components and are expected to address interconnect delay related problems. This can be exploited to build faster SoC circuits by choosing the optimal placement between layers, therefore optimizing load in blocks with different performance requirements. Since the process of implementing vertical interconnections is more challenging compared to the planar ones, the physical characteristics of these interconnections can become a burden for achieving the required performance. Various types of vertical interconnections have been proposed so far [3] such as TSV, wire bonding, metal bumps and contactless bonds. For the purpose of system efficiency, TSV technology has been the preferred technology for new design patterns.

System dependability and reliability

System reliability and dependability become more important as the technology is scaled down due to increase in component break-down possibilities resulting from physical problems. Systems with high dependability issues are often required to continue operation after an active component or wire fails. With a built in functionality that updates all of its components about the changes in the system, it is possible to warrant at least partial capability of the system in case a component fails. The trait to maintain partial functionality when a component failure occurs is called *graceful degradation*. Graceful degradation is often considered equivalent to fault tolerance. Systems with *fault tolerant* designed have backup components that take over the work of a failed component, *graceful degradation* however is a method of effective *fault management*, where the system will detect, isolate and resolve failure problems. Examples include application areas such as real-time systems in critical areas where even a smallest breakdown can mean loss of life, systems that can employ adaptive technological advantages bypassing failed components by rerouting communication and rescheduling applications to avoid complete system failures until repairs can be made. Backup components can also degrade over time and might be unusable when need arises.

1.2. Objective

The objective of this thesis is to provide an assessment methodology for IC design by creating a method to evaluate the impact of the number of TSV links in upcoming 3D technology implementations. Assessment in a large search space of possible configurations of such a platform can become cumbersome if algorithms with a long runtime that consume too much time are used. Quicker analysis methods are necessary to lower development cost and duration or assess system functionality after a failure has occurred. This thesis will propose a simple heuristic method to help assess the effectiveness of TSV based three dimensional NoC with non-uniform layout configurations. This is done by quickly assessing the necessity of each TSV by comparing application execution time to TSV count. For comparison purposes a simulated annealing algorithm was used. Comparison results have shown that the simulated annealing algorithm provides evaluation values with lower estimated application runtime time, but takes several times longer to execute to

assess the search space. For quick difference comparison purposes the new assessment method is well suited.

1.3. Organization

This thesis is divided into 4 chapters. Chapter 1 provides a short overview of the thesis, making introductions into SoC and NoC technologies and how they evolved towards three dimensional technologies, the target technology line for this thesis. Additionally a short introduction into system dependability can be found. Chapter 2 will explain the background of the target system and technology and the design flow needed to optimize the performance. Chapter 3 presents a new method to assess TSV impact in 3D NoC layouts, a comparison algorithm and test results comparing the outcome of the algorithms. Chapter 4 contains the summary of the background and test results.

2. Background

The aim of this chapter is to explain the basics of the target platform by giving a short overview of SoC and NoC structure and capabilities in detail and provide an overview of their structure. Chapter 2.1 will explain the basics of a NoC - point out properties characteristics and building blocks, and benefits of using them instead of ad-hoc global wiring or regular bus architectures. Chapter 2.2 will provide a broader overview of 3D stacking technology, while Chapter 2.3 take a closer look into TSV technology and its design.

MPSoC systems can be divided into homogenous and heterogeneous systems according to the IP variety they employ. Homogeneous systems integrate only one type of general-purpose processors for the sole purpose of increasing system parallelism. General implementations include PCs and general purpose handheld devices. However the SoC ability to be globally asynchronous and locally synchronous known as GALS, allows the usage of a wide variety of IPs, creating heterogeneous systems [18]. Heterogeneous systems are more challenging and costly to design as task specific IPs have to be mapped

onto proper positions. IP mapping is a step in design methodology when the system has two or more different IPs with specific purposes as opposed to general-use processing units. It is possible to minimize the execution of specific applications by strategically placing task specific IPs in favorable positions in the system during the design phase. During the task mapping process on heterogeneous systems, if specific task execution time on every type of IP is known beforehand, a complex scheduling algorithm is needed to optimize application execution. Heterogeneous system performance becomes poor when the application does not fit the ideal form or type due to longer routing delays or possible task execution time increase resulting from task execution on non-task specific IP. Heterogeneous structures are generally used in embedded systems.

2.1. The NoC Paradigm

To circumvent the communication bottlenecks in MPSoC architecture bus and network infrastructures are used. Embedded systems, which are often dedicated to specific tasks, compared to general purpose computers such as Personal Computers (PC). A dedicated processor unit is cheaper to manufacture than general purpose processor as it requires fewer logical components.

Similar to modern telecommunication networks a NoC consists on multiple point-to-point communication links interconnected by switches, allowing packets to be sent from one module to any other in the system using digital packet switching over multiplexed links. It is possible to use any regular network topology but by far the most common design is the square mesh topology.

Mapping algorithms for MPSoC can be divided between two processes – hardware (IPs and interconnections) placement algorithms for heterogeneous hardware layout design, and task mapping to optimize software concurrency which will be explained in *chapter 3*. For heterogeneous MPSoC systems dedicated to a specific task, both application task mapping and IP placement are often optimized simultaneously to for better execution times and system performance.

2.1.1. NoC architecture composition

Generally a network consists of multiple terminals interconnected via switches and wiring between them. Network on Chip solutions in MPSoC are represented with three main components such as **links, routers and network interfaces**. *Figure.1* shows an example layout of NoC implementation using a mesh topology.

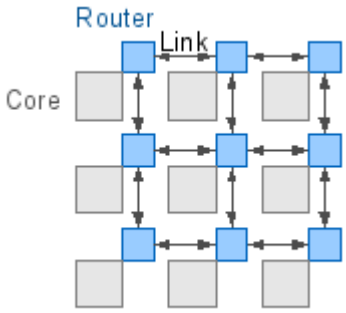


Figure 1: Simple NoC with standard resource elements

Links in NoC systems consist of channels – physical channels, represented by a group of wires, and optional virtual channels, represented by a set of additional buffers built into routers. A virtual channel is created by temporarily storing the message in the extra memory buffer of a router, guided according to a set of protocols. The width of the channel, the number of parallel wires, may vary depending on the bandwidth requirements and avalible chip area. The number of wires in a unidirectional channel is usually constant throughout the system and is known as the "channel bitwidth". A channel can be unidirectional or bidirectional depending on the design requirements and limitations. Generally links in networks architectures have two physical unidirectional channels for opposite directions, creating a fully-duplexed link, allowing communication to flow both ways without collisions. A link in NoC architecture can in most cases only be used in a single direction at the same time for noise mitigation in an extremely constrained area. The use of interwoven bidirectional channel structure however, using alternating clock cycles and using the other channel as noise shielding, enables the use of the in link both ways simultaneously [2]. Networks using point-to-point links can be viewed as a set of

interconnected switches, each connected to zero or more nodes. Direct networks are router based and correspond to cases where every router is connected to a single node and can be divided into mesh, torus and hypercube subdivisions. Indirect networks are switch based and can be separated into crossbar networks and multistage interconnection networks. Network topologies are explained in-depth in *chapter 2.1.2*.

Routers in NoC are elements composed of buffers, input/output channels and an implementation of any networking protocol(s). The router implementation complexity impacts the cost of design and validation, as well as the area and power consumption. A router acts by checking the destination address of the packets received and re-reoutes the packets according to a network protocol implemented in the system on local router level or global routing paths assigned by the source node.

Network interfaces are network translation devices built into processing cores. Their main purpose is tag packets with destination addresses. Since most NoCs are message-passing by nature, an adapter is needed. It should be observed that in realistic NoC architectures, network interfaces play a significant role in determining the overall chip area requirements. Network interfaces allow the separation between computation and communication platform, which in turn allows the reuse of both, core and communication infrastructure independent of each other. Network interfaces can also be implemented on block level to better accommodate the integration of different signals throughout the entire chip.

2.1.2. NoC architecture layouts

A NoC can be characterized by the structure of the router connections. Many regular network topologies can be implemented in NoC technology using different component designs and protocols for a wide variety of purposes. In direct-network topologies nodes are composed of a router and the associated processing unit(s), and connected to a fixed number of neighboring nodes. Messages between two nodes go through one or more intermediate nodes. In these topologies only the routers and links are involved in

communication between nodes, using a routing algorithm implemented by routers. In this arrangement, nodes are distributed in an n-dimensional space where packets can move in only 1 dimension at a time. *Figure 2* shows a several examples of the most common direct topologies.

- Ring topology is the simplest network topology that comes with a high cost, the average "hop count". A hop count refers to the number of intermediary network nodes used to reach the destination. Ring topology can often be compared to BUS designs due to the simplistic structure, large channel count and channel bitwidth as the the links of a ring topology are unidirectional. Communication flow is unidirectional and can only move in circular motion to reach other nodes. Absence of routing path diversity can lead to performance bottlenecks under heavy loads and is a major obstacle for fault tolerance. Regular ring topology is not recommended for larger networks. Hierarchical ring topology divides large systems into small clusters and connects them with opposite directional ring flow.
- Mesh topology nodes create a semi-permeable barrier similar to a mesh, in the shape of interlinked squares. By far the most implemented NoC topology, due to its structure flexibility and low complexity. The link structure is bidirectional and allows for dynamic communication flow to avoid stalls resulting in
- Torus topology is an evolutionary step in mesh topology with a set of links connecting opposite side nodes of the same dimension to form shorter paths. Torus layout requires several additional metal layers to implement the extra link set.

A generalization of the cube to dimensions greater than three is called a hypercube, n-cube or measure polytope. Hypercube topology is based on the fourth dimension principle, creating the illusion of a tesseract. Just as the surface of the cube consists of 6 square faces, the hyper-surface of the tesseract consists of 8 cubical cells.

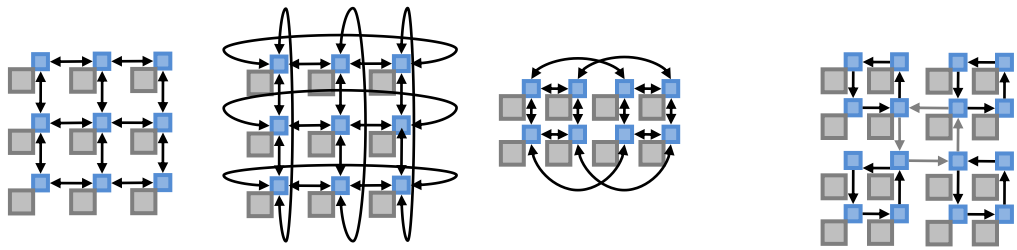


Figure 2: (a) Mesh, (b) Torus, (c) Hypercube, (d) Hierarchical ring

In indirect topologies some routers are not directly connected to any processing units and are only meant to propagate the communication to nodes further away.

- Fat-tree topology does not have a specific form. The basic principle connects the network links into large centralized channels creating bundles that look like large tree trunks. By judiciously choosing the fatness of links, the network can be tailored to efficiently use any bandwidth made available by packaging and communications technology. In contrast, other communications networks, such as hypercube and mesh topologies, have communication requirements that follow a specified mathematical law, and therefore cannot be tailored to specific packaging technologies.
- The crossbar's high degree of connectivity allows for a large number of simultaneous connections to keep data moving through the network. Unfortunately, this connectivity comes at a high cost. Crossbars utilize a large number of switches and a large number of wires, which translates into high power consumption, large size, and low operational frequencies. The key feature of the crossbar is that it is a strictly non-blocking network; any free input port can be connected to any free output port without changing existing input/output pairs. However, non-blocking design may be too cumbersome for system constraints.
- Multi-stage topology is a regular NoC, where routers are identical and organized stages. Input and output stages are connected to the functional units in one side and to the internal nodes in another side.

2.1.3. Routing rules

A routing protocol specifies how routers communicate with each other, propagating information to processing units enabling the selection of different paths between any two nodes on a network if more than one is available. Routing algorithms determine the specifics leading to a choice of a route. Depending on the network structure, routing can be blocking or non-blocking, if it can manage all requests that are issued during operation. Communication in a NoC system is usually carried out using an implemented form of a handshake protocol, similar to how communication in regular computer networks. Handshake protocols are built on the principle that no communication will be carried out until the destination node is notified about an incoming transfer and the reply has been received in the form of an acknowledgement message. Routing can be divided into two main categories, *adaptive* routing and *oblivious* routing. When a routing path is blocked by another communication activity, adaptive routing considers other paths to reach its destination following a set of rules. Oblivious routing routes packets without any information about traffic and conditions of a network, ignoring the possibility of situational routing paths. Oblivious routing algorithms never end up in a *dead-lock* situation. A dead-lock is a situation where the communication activity is closed in a loop between routers, this situation occurs when a path is blocked and the communication flow is redirected dynamically to avoid stalls. Both of the packets reserve some resources and both are waiting each other to release the resources. Below are a few examples of oblivious routing algorithms that can be integrated into most path finding algorithms.

Dimension order routing algorithms route packets along dimension in a specific order, once a packet has traveled to the distance in the specific dimension, it cannot be routed along that dimension again. XY routing, a dimension order routing algorithm, suits well on a network using mesh or torus topology. Addresses of the routers are geometrical coordinates in two-dimensional space. Turn model algorithms can be applied to prevent dead-lock situations in dynamic routing environments where real-time constraints are critical. Turn model algorithms determine one or more directional turns which are not allowed during routing [21].

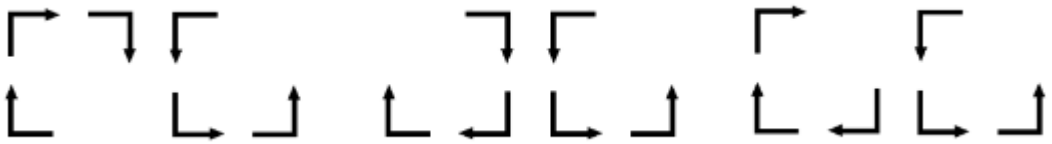


Figure 3: Turn model routing examples, (a) west-first, (b)north-last, (c) negative first

Shortest path routing algorithms are the simplest deterministic routing algorithm, where packets are always routed along the shortest possible path. Examples of shortest path routing algorithms are distance vector routing and a link state routing. In Distance Vector Routing, each router has a routing table that contains information about neighboring 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 shortest path on the grounds of their routing tables and then send packets forward. Distance vector routing is a simple and cost efficient method as each router does not have to know the structure of the whole network. If nodes are always updated on the layout, a source routing method can be applied where 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 carry out the sender's instructions.

2.2. Three-dimensional IC architecture and manufacture

Taking full advantage of expanding in three dimensions instead of two requires sophisticated design techniques and new computer aided design tools for higher manufacturing precision. There are still very few standards for TSV-based 3D-IC design, manufacturing, and packaging. In addition, there are many integration options being explored such as via-last, via-first, via-middle, interposers and direct bonding. Currently there are three 3D technology design methods: monolithic 3D, where all of the electronic components and interconnects are built on-top of a single silicon wafer layer by layer and then diced into fully functional 3D chips, wafer-level stacking, where components are built on two or more wafers that are thinned stacked bonded and diced into chips, die on wafer stacking, where multiple dies are aligned and bonded onto each other and then onto a

carrier wafer or a silicon interposer layer. A cross-section example of 3D stacking can be seen in *Figure 3*.

- A silicon interposer is a thick silicon layer containing only TSVs that connect multiple chips that are stacked separately on top of the same interposer layer, often referred to as 2,5D, a method that allows the reuse of pre-built dies with similar footprints. A silicon interposer minimizes the TSV area penalty and allows the use of a large variety of different types of TSV, including optical TSVs for high speed data transfer and polymer clad TSVs for heat collection [4]. Silicon interposers are already used in FPGA board manufacturing process [7], where integration of different signals and reuse of pre-built chips is an important requirement.
- Stacking approach includes Wafer-to-wafer, die-to-wafer and die-to-die stacking methods. Wafer-on-wafer method stacks entire silicon wafers, with a single layer of active devices, on-top of each other. In this method, the vertical interconnections are etched through the entire wafer and all metal layers. Wafer-on-wafer bonding can reduce yields, since if any 1 of N chips in a 3D IC are defective, the entire 3D IC will be defective. Die-on-wafer stacking method utilizes pre-made components which are built on one or several semiconductor wafers. All but one wafer are diced into chips, then aligned and bonded onto die sites of a wafer that will remain un-diced for the duration of the stacking process. Using die-on-die stacking method, electronic components are built on multiple die, which are then aligned and bonded. Thinning and TSV creation may be done before or after bonding. One advantage of die-on-die is that each component die can be tested first, so that one dysfunctional die does not ruin an entire stack.
- Monolithic approach is an upcoming technology that aims to replace TSV based 3D technologies when it comes to multiple layers of active components. The manufacturing process involves a sequential device process where the frontend device layer construction is repeated on a single wafer to build several layers of active devices. The main design problem for this method has always been heat dissipation, however recent research has shown that using polymer clad TSVs to direct heat directly into the heat-sink is possible. Many advances may have taken

place in the recent years, but has yet to prove reliability and usability from mass production point of view.

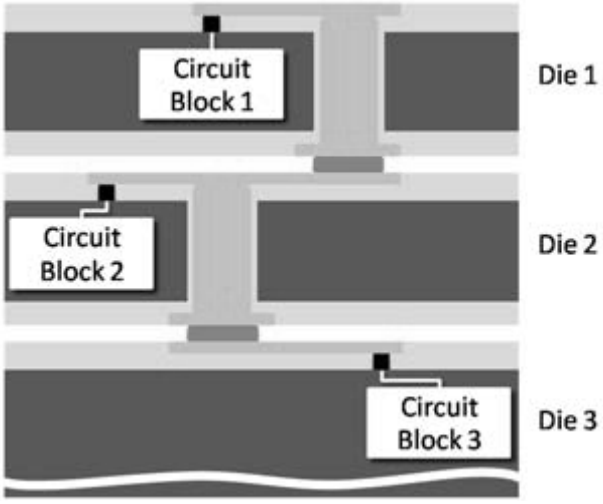


Figure 4: 3D architecture

2.3.Through Silicon Vias

Through Silicion Via (TSV) is one of the interconnection technologies for 3D IC, enabling the use of multiple layers by bonding them together and able to relay communication, power and even heat depending on its design. TSVs are vertical interconnections that pass through the entire silicon substrate which no active area can overlap with [13].At the 45 nm technology node, the area footprint of a 10μm x 10μm TSV is comparable to that of about 50 gates. Additional area around a TSV is also reserved in the form of a "keep-out-zone" representing the signal saturation distance it takes to prevent noise related problems. Although TSV reduces interconnection length between cells, when placed on top of each other, they increase wire length on planar scale since they occupy significant silicon area spreading out the placement. Furthermore routing becomes more difficult on planar scale, especially for TSV last methodology as the TSVs go through all metal layers becoming possible obstacles. Excessive or ill-placed TSV not only increase die area but have also negative impact on coupling. Depending on VIA first or VIA last methodology, may even interfere with not only the device but also the metal layer.

Interconnection placement is mainly used in three-dimensional IC design in system without a NoC

2.3.1. Design and manufacture of through silicon vias

The goal of TSV design is to minimize the size and maximize the TSV pitch without exceeding the maximum resistance permitted by an application. This combination results in a TSV whose low capacitance and resistance leads to a power-efficient design that meets the system's performance requirements. No metal wire lines can be routed over the TSV zone except the highest and lowest metal layer deposits that are used for routing and redistributing the TSV's own signal for the same reason. The integration of TSVs can be divided into three groups depending on the phase they are etched into the design. Via-first TSVs are manufactured before metallization, thus occupy the device layer and result in placement obstacles. Via-middle are manufactured after Front-End-Of-Line process but before Back-End-Of-Line. Via-last TSVs are manufactured after metallization and pass through the chip. Thus, they occupy both the device and metal layers, resulting in placement and routing obstacles. Typically TSVs manufactured using via-first and via-middle methods are smaller, denser and with larger aspect ratios than via-last method. An example of via-middle manufacturing process can be seen in *Figure. 4*. While the usage of TSVs is generally expected to reduce wire length, this depends on the number of TSVs and their characteristics.

TSV architecture typically consists of a cylinder with a uniform circular cross-section of a conducting material surrounded by an insulator which is intended to prevent voltage leaks and lower parasitic capacitance [3]. The final characteristics of a TSV depend on the geometrical parameters (height, diameter, pitch, and oxide thickness) and electrical parameters (metal conductivity, oxide permittivity and silicon resistivity). Several issues such as fabrication technology, heat removal, reliability, application technology and many others, have to be resolved simultaneously and have been preventing proper implementation of this method in the past. Metal filled TSVs with very high density can be achieved by depositing a thin titanium-nitrate film that acts as a seed layer and diffusion

barrier for the tungsten deposits using metal-organic chemical vapor deposition. To release stress, the tungsten is partially etched back. This process is better best suited for small TSVs as larger than 5µm are preferably filled with quicker and cheaper process like electrodepositing copper. Stress monitoring shows that the maximum stress and strain with tungsten filling is observed not in the bulk region of the TSV but in the upper section between the metal layer and tungsten filler, while copper-filling experiences more stress in the TSV [12]. The possibility of manufacturing TSVs with other properties such as coaxial and optical TSVs for data transfers, polymer clad electrical TSVs for power distribution or even fluidic TSVs for coolant material routing [13] [14].

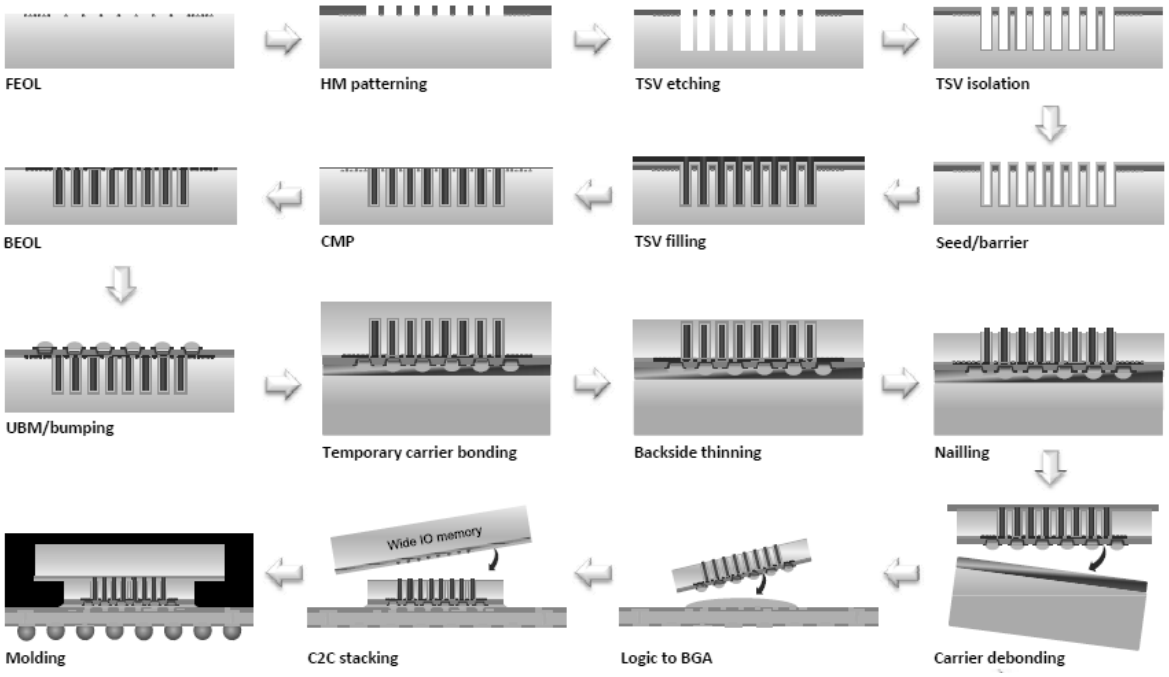


Figure 5: Via-middle manufacturing process

TSVs in 3D design are connections to other layers stacked on top of each other by either soldered micro-bumps or thermo compressed bump-less bond pads. Lack of bonding strength due to bad bond pad distribution during die stacking and vibration testing can lead to delamination and cracks in the substrates. Bonding failures can be lowered by changing bond pad density, a technique that has already been accounted for in the physical design phase of bond pads. The most straightforward bonding method assumes that, for every

TSV, there is at least one bond pad to attach to the next die in the stack. Addition of further bond pads would require more TSVs to be inserted into the design; however, as suggested by [3], maximum TSV amount (NTSV) on a chip is limited due to area penalty. None the less, using this approach creates the requirement of a minimum number of bond pads. This leads to the assumption, that interconnections between dies are solely limited by TSVs, not bond pads. The need to decrease NFP and increase NTSV in 3D integration starts contradicting the fabrication process as long as NFP > NTSV requirement is set. A proposed solution would be to create redundant or dummy bonding pads what leads to complex problems in operations that involve two or more dies. According to [6], backside bond pad routing is a difficult process after wafer thinning, resulting in a larger routing pitch. Vertical alignment of bond pads to corresponding TSVs can be done to reduce costs by removing the need for bond pad routing. Test results have shown a highest success ratio of 93% for proper bonding using double bond pads, where the dummy bond pads have no net connection. In a more recent approach to building 3D chips, 2D chips are stacked and either bump- or adhesively bonded to a base wafer. In this design vertical connections are achieved within but not through the chips. The sizes of the chips can be different which permits the integration of chips from different sources and different technologies, but the alignment of the pads on the base wafer and the chips to be attached must be compatible. To minimize the TSV pitch, etching method is required that maximizes the aspect ratio of the depth to the width of the TSV cavity. In addition the etching process must be done in a way that avoids erosion which can lead to an increase size of the cavity at the surface, increasing the area required by the TSV. A masking agent that is not affected by the etching process can be used to maintain the dimensional integrity of the mask but removal can be a challenge [15].

2.3.2. Design challenges using stacking and vertical interconnection technologies

Several design challenges have prevented the efficient use of TSVs in IC manufacturing process, including noise and heat mitigation, area penalty, bonding strength and stacking accuracy. Substrate noise in 2D ICs is well-studied but noise caused by the use of TSVs in

3D chip has not been studied thoroughly. Establishment of more effective design guidelines are necessary to better understand how critical circuit design parameters, such as signal slew rate, and TSV-to-TSV/device spacing impact signals. Placing substrates on dies with grounded backside planes have been used in certain 2D packaging but have yet to be redesigned for 3D purposes. The experiments in [14] pointed out that body voltage for the substrate without grounded backside was extremely high, reaching almost transition voltage levels due to coupling and having no charge collection ways. Voltage transition in the TSV affects the substrates, causing change in body voltage of nearby devices. Although with a small time window, such voltage changes affects both analog and digital devices, the latter being more susceptible to the timing of the peak voltage change. Experimentations with slew rate have shown that peak noise always occurred at constant distances from the and further analysis showed that body voltage is independent of voltage transition time in the TSV. Thickness of the TSV sidewall has shown similar results as peak locality did not change, but showed considerable noise reduction in both configurations. TSV height however affects peak body voltage locality, amplitude and affected substrate area for both devices and other TSVs.

The major manufacturing yield limiters for a 3D IC technology are die and wafer bonding defects from stacking, TSV shorts due to stack misalignment, and changes of device parameters due to 3D processing. Thermal management is one of the important issues of 3D IC integration. Effective thermal management methodologies and solutions are needed for widespread use of 3D IC integration. The heat dispersal/source environment of a 2D IC is the cooling material, but the same environment of a die within a 3D IC may be another die that also generates heat. The thermal analysis is an important and proper heat collection TSV layout modification from thermal and stress distribution point of view can enhance the circuit reliability. Therefore, Thermal management in 3D ICs is critical for maintaining required reliability, performance, and power dissipation target.

The addition of a third dimension would require more advanced planning tools to account for the new dimension [3] [18]. 3D IC physical design has attracted an increasing amount of attention and has generated a significant amount of research work on the floor planning,

placement and routing for 3D ICs. However, all these tools have been developed by different groups, using different formats to represent the design data, creating barriers for researchers who need to use the existing design tools to conduct further studies. Considering the above, IC design is in need of new standardization to accommodate 3D IC design.

2.4. Design flow

To optimize the performance of a technology a proper design flow must be followed. Systems with multiple processors for example have increased performance compared to single processor systems but include many new complications related to design complexity, proper resource usage and communication between them. The performance of a platform does not multiply when adding more processor as complications with application concurrency and increase in data transmissions leads to a drop in overall system performance, creating bottleneck situations for communication and memory systems. To increase the performance gain, several design and management methods have been devised. Chapter 3.1 gives a short overview of task mapping and Chapter 3.2 a short overview of task scheduling respectively.

2.4.1. Mapping algorithms

Task mapping entails the designation of an application's tasks to different processing units in multi-processor systems. Application performance can be increased by mapping tasks in a way that would minimize the task execution time and communication delay between them. The main weakness of dynamic mapping is the incomplete data available of the task graph, since the task being mapped considers only the communication with its recipient task [17]. On the other hand, static algorithms consider all tasks and resources together, allowing for better mapping exploration using more complex algorithms. Simulated annealing algorithm, explained in detail in *chapter 4.3*, is often used for task mapping purposes in MPSoC systems. Most static task mapping algorithms are based on critical path

principle, first mapping tasks that are designated in the critical path sequence. Commonly used with all forms of projects, including construction, aerospace and defense, software development, research projects and engineering among others. Although the original critical path algorithm is no longer used, the term is still generally applied to all approaches used to analyze a logical diagrams and graphs. A schedule generated using critical path techniques is often not processed precisely, as estimations are used to calculate values, if mistakes are made, the results of the analysis may change. For example, the bottom-up algorithm is based on the critical-path algorithm. Instead of starting with the first task, the bottom up algorithm maps tasks start from the bottom of the graph and work their way up.

2.4.2. Scheduling

Scheduling in general is a process of deciding how to commit resources between a variety of possible tasks. In computing, scheduling is a concept by which processes and data flows are given access to system resources such as processor time or communication bandwidth. Initially on a single processing unit, scheduling consisted of sharing the processor resource between several processes running on the system using a variety of different methods to fill empty processor cycles or switch between currently running processes based on task priority. A scheduler, a protocol responsible for choosing processes to execute, works by selecting tasks based on priority values influenced by scheduler type.

Scheduling methods can be divided into two sub-categories. The first category is non-preemptive scheduling, where a process once scheduled will be executed until completion. The first category is preemptive scheduling, a method where a process can run for a predetermined amount of time before a preemptive check allows another process to run. Dynamic scheduling that falls under this category implies the possibility of executing tasks as soon as they can be executed, making it possible for new instructions to be carried out when a stall occurs as long as they do not produce application structural hazards or dependencies. Due to the contextual objective of the thesis and for the purpose of lowering complexity of the scheduling phase described in detail in *chapter 4*, dynamic scheduling will be ignored.

Non-preemptive scheduling is mostly used in static schedules, which are optimized by the compiler. When the application is stalled, no further instructions are given until the situation is resolved by hardware. Real-time schedules are dynamic by nature and are mostly enforced by hardware. Most static scheduling algorithms are based on the *list scheduling technique* [1]. The basic idea of list scheduling is to make a scheduling list, a sequence of tasks yet to be executed, by assigning priorities to them and repeating the simple process of checking each entry of the list sequentially if it can be executed starting with the highest priority tasks. The priority of a task can be assigned using any number of evaluation parameters defined by the algorithm or the user.

3. TSV impact assessment on 3D NoC platforms

The highest priority objective in chip design is the balance between performance and cost; in consideration, the need to assess new design aspects properly becomes apparent as TSVs require chip area several times larger than a regular interconnection. With the integration of each new TSV, the chip area will be increased, resulting in relative distance increase for all interconnection. Additionally the possibility exists that chips with few dysfunctional TSVs can be reconfigured to provide at least partial system performance and capabilities from the ideal form. A method is needed to reassess the chip performance and cost values. In consideration of the above, a method is needed to assess the efficiency of an entire system for possible impacts the TSVs might have. For comparison purposes a modified simulated annealing algorithm will also be implemented on the testing platform.

3.1. Architecture platform

MPSoC systems generally use a tile-based multiprocessor template often found in parallel computer architecture related literature where each tile contains one or more processor cores and local memories. Tiles (nodes) can be either homogeneous or heterogeneous from the system point of view – in a homogeneous MPSoC system all nodes contain identical processor cores, while heterogeneous MPSoC system tiles are chosen from a variety of

different processor cores available. The tile architecture and processor core differences have no impact from communication point of view.

Definition 1. (Architecture) *The network architecture* $N(P, L_i)$ *is a three dimensional mesh composed of* $m \times n \times k$ $(X\ Y\ Z)$ *number of homogenous tiles, where each node* $p \in P$ *represents individual tiles and* $l_k = (p_i, p_j) \in L$ *represents a link between nodes* p_i *and* p_j *. Vertical dimension links* $L_i \subseteq L$ *connecting horizontal layouts can be arbitrarily mapped and can create non-uniform paths. Vertical links, in X and Y dimensions, differ from horizontal links, in Z dimension, in length and bandwidth.*

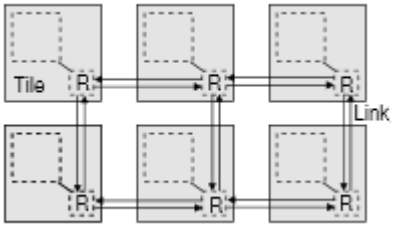


Figure 6: NoC-based MPSoC architecture example

Communication between tiles involves sending data over a sequence of links from the source to the destination tile. This sequence of links through the architecture layout is called a route and is defined formally as follows.

Definition 2. (Route) *A route* $r_{i,j}$ *, between tile* p_i *and* p_j *where* $p_i \neq p_j$ *, is a sequence of links* $l_m, l_{m+1}, ..., l_{n-1}, l_n$ *in the network architecture. Src and dst are the respective source and destination tile operators of the route or a link. For a route* $r_{i,j}$ *we assume that:*

- *The source of the first link* $src(l_m)$ *is equal to the source tile of the route* $src(r_{i,j})$ *and the destination of the last link* $dst(l_n)$ *is the destination tile* $dst(r_{i,j})$ *.*
- *For any two consecutive links* l_k, l_{k+1} *in a sequence,* $dst(l_k) = src(l_{k+1})$ *.*
- *There is no cycle in a route, i.e. for any two links* $l_k \neq l_l$ *in the sequence holds* $dst(l_k) \neq dst(l_{k+1})$ *.*

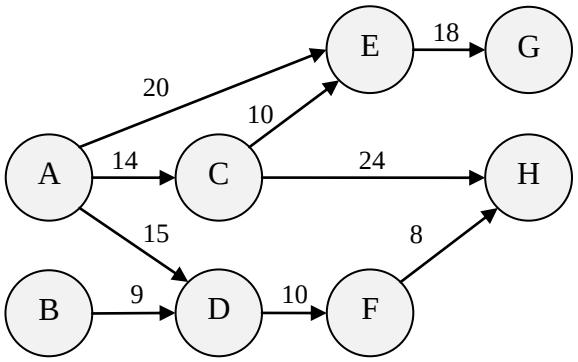
- *When* $src(r_{i,j}) = dst(r_{i,j}) \ni r_i = r_j$ *will result in an empty route* $r_{i,j} = \varnothing$.

  The length of a route $r_{i,j}$ *is equal to the number of links in its sequence, denoted by* $|r_{i,j}|$. $l_m \in r_{i,j}$ *denotes that link* l_m *is part of the link sequence of route* $r_{i,j}$.

Links and processor tiles can generally be used by multiple processes simultaneously due to conventional pipeline methods built into the elements themselves, however for the purpose of lowering test complexity all system elements can be used by only one activity at a time. All nodes will only be able to process a single task at any given time. All links in a route $r_{i,j}$ sequence will be reserved and blocked to other activities until a communication transfer has finished. When the source and destination nodes of an inter-task communication data transfer are the same, communication time and cost between the tasks is equal to 0.

3.2. Problem formulation

Definition 3. (Application entity) *An application is represented by* $G(V,E)$ *and is an task precedence graph consisting of two subsets: a subset of nodes* $v_i \in V$ *with data volumes denoted as* $vw(v_i)$ *and a subset of edges with precedence relation between two nodes* v_i *and* v_j, *denoted as* $e_{i,j} \in E$, *and is associated with the weight of the edge* $ew(e_{i,j})$ *indicating the communication volume between* v_i *and* v_j.



node	units
A	20
B	12
C	17
D	22
E	16
F	10
G	8
H	14

Figure 7 – An example of a precedence graph with edge weights and execution times

Definition 4. (Mapping) *Given a set of precedence constricted tasks* $V \subset G$ *and assigning them to a set of processors* $P \subset N$ *for execution, forms a mapping* $M(V,P)$ *which dictates on what processor resource each task is carried out on. A mapping does not include communication resource assignments and cannot therefore dictate in which order the tasks are executed.*

A mapping assigns tasks to be executed by specific nodes. If no constraints are set for node structure and type, both homogeneous and heterogeneous mappings are possible. On a homogeneous system, if the assumption that the system is heterogeneous and the application's tasks are already partitioned and mapped optimally on the target system is made, creates a situation where the system can be perceived to be either homogenous or heterogeneous. Random and heuristic mapping methods

Definition 5. (Schedule) *A Schedule, a 3-tuple* $S(N,G,M)$*, is an execution order and communication routing schema, where N is the architecture description, G is the application graph and M is a mapping of said graph onto the architecture platform. Nodes and Edges in set G cannot be scheduled until all of itheir predecessors are scheduled.*

A schedule provides a time frame, within which a communication event can be sent along the route r. All links along the route's sequence are reserved for the entire duration of the event.

Definition 6. (Schedule function) *A scheduling function* $S : E \to C$

- *the route starts from the source tile:* $i = src(r)$,
- *the route ends at the destination tile:* $j = src(r)$,
- *the communication does not start before the earliest moment in time at which the data is available:* $t_{start} > t_{ready}$,
-

The total communication cost can be formulated as the sum of all communication transmissions for every *communication process between tasks, while the communication cost for a single transfer is equal to 0 if two consecutive tasks are mapped onto the same processor.

$$T_{comm} = \sum_{\forall (i,j)} r_{i,j} \quad (1)$$

The problem can be stated as the following:

Given:

1. Application task graph, $G(V, E)$
2. Network topology, $N(P, L)$
3. Mapping $M(V, P)$

Determine:

A *pareto efficiency set*** for TSV number and execution time.

Such that:

*note that link speeds may vary according to the interconnection type that is used.

***The pareto efficiency set provides an overview about the impact of adding/removing TSVs in the NoC.*

- A single schedule provides an area penalty value based on the number of TSVs in the network topology
- The placement of inter-layer communication links is different for every schedule.
- Total number TSVs and graph execution time for each schedule form a pareto efficiency set.

3.2.1. Traffic assessment method

Given a 3 dimensional NoC layout with $m \times n \times k$ number of tiles, defines the max number of vertical interconnections to be equal to $i = m \times n \times (k-1)$. The possibility of a vertical interconnection in the system is binary, either existent or non-existent, giving us a layout combinatorial search space given by *equation.1*. The last layout for every vertical layer is discarded as unfeasible due to lack of vertical interconnections between layers, making inter-layer communication impossible.

$$\sum_{j=0}^{i} \frac{i!}{(i-j)! \times j!} = 2^i - (k-1) \quad (2)$$

Example: if number of tiles in system is equal to $8 = 2 \times 2 \times 2$, the max number of vertical interconnections would be equal to $4 = 2 \times 2 \times 1$ and the search space consists 15 of possible vertical interconnection layouts (3).

$$15 = \frac{4!}{(4-0)! \times 0!} + \frac{4!}{(4-1)! \times 1!} + \frac{4!}{(4-2)! \times 2!} + \frac{4!}{(4-3)! \times 3!} + \frac{4!}{(4-4)! \times 4!} \quad (3)$$

Optimal number of tested solutions becomes critical for high tile count, due to the $\mathbf{2^n-1}$ complexity of the search space. When all interconnections are given a priority value the number of solutions that need to be tested becomes equal to **n-1**.

The general idea of this method is to quickly and efficiently find 3D NoC layout assessment when TSV layout is non-uniform. This is done by removing a TSV from a fully homogenous mesh layout, therefore changing the routing paths and impacting the execution time of an application. Time needed for communication events inside an application is influenced by the routing distance. This process is repeated until there are no vertical interconnections remaining. The result will be a pair of vertical interconnection number and execution time for each point where a change to the network layout occurred.
Probabilistic and random load-balancing mapping methods have an even distribution chance for the entire system, but are less efficient when inter-task communication is present. The amount of inter-task communication generates traffic that can be used to assess interconnection suitability, therefore the task mapping will be carried out randomly to simplify the testing process of the algorithm and increase the accuracy of test data.

The system is initialized before the algorithm is executed, this include the construction of the network structure and processing the graph data. The initial scheduling is done with an unmodified fully homogenous system to ascertain the maximum values and evaluate the initial state. All vertical interconnections are checked and order by traffic load into a working set and the interconnection with the lowest value is set to inactive. The next process is an iterative loop that creates a new schedule based on the new layout and recalculates the execution time. The process finishes when no interconnections in the set remain. A diagram of the entire process is depicted on *Figure 8.*

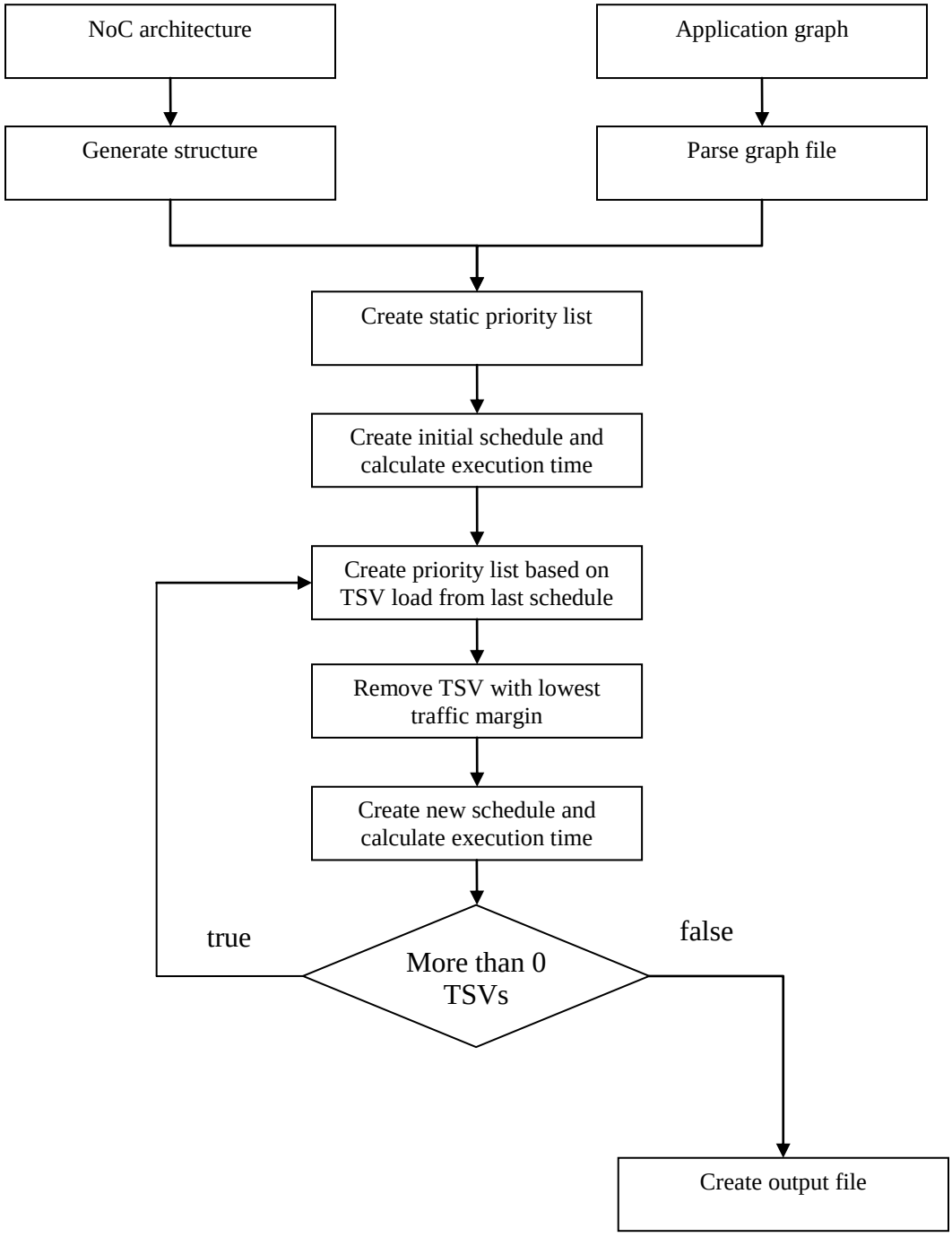


Figure 8: Traffic priority algorithm diagram

To determine the effectiveness of different assessment parameters a series of small tests were carried out. The first test on the platform described in *chapter 4.2.2* is carried out with the traffic based heuristic algorithm to see what type of communication evaluation is possible. Tests were carried out using two slightly different traffic assessment parameters, such as activity count (AC) and total time spent on activities (TS). AC counts the routes that were mapped through the specified vertical interconnection during the scheduling process, while TS calculates the total time the vertical interconnection had been in use during the execution of these tasks. The first test application APP1 consist of 41 tasks with execution length of 5-15ms and 40 communication events ranging from sizes of 20-30 packets. The second application APP2 consists of 59 tasks with execution length of 5-15ms and 40 communication events ranging from 20-30 packets. An example of the results of the tests can be seen in *table.1*. In *Table 1*, *column 2* and *5* show test results using AC as priority assessor variable, while *column 3* and *6* show TS as assessor variable. *Table 1*, *Column 4* and *7* show the execution time difference between the two methods. The difference between AC and TS methods can be seen on *Table 1, row 7* to *9* with only TSV count goes below 4.

Table 1: comparison of A1 and A2. Test series 1

TSV count	exec AC (app1) [ms]	Exec TS (app1) [ms]	delta |TS-AC| [ms]	exec AC (app2) [ms]	Exec TS (app2) [ms]	delta |TS-AC| [ms]
9	357	357	0	1307	1307	0
8	394	394	0	1318	1318	0
7	378	378	0	1350	1350	0
6	378	378	0	1350	1350	0
5	404	404	0	1424	1424	0
4	406	406	0	1415	1415	0
3	442	418	24	1498	1630	132
2	427	428	1	1511	1694	183
1	571	613	42	2416	2443	27

The results show little difference between the methods but show that AC method is slightly more efficient when using the same mapping and scheduling methods. AC method will be therefore used for all other test sets in this thesis.

3.2.2. Test system description

The test platform will imitate a 3D mesh NoC layout by implementing a data structure that simulates node and link positioning that is generated at the execution of the test application. The test system consists of 4 individual components:

- Customizable network structure implemented as a three dimensional cube structure where each node is an abstract element defined before compile time. The necessary elements for a layout are defined on compile time prior to execution. The nodes can be turned on or off by the user or the algorithm.
- A graph parser is built into the system that accepts TGFF [10] file format and generates a graph structure based on the graph description found. It is possible to generate random precedence task graphs of any size, suited for the purpose of these tests, using the TGFF platform.
- An implementation of a path routing algorithm that can find a path given a 3D data structure based on shortest path principle. The implemented A* algorithm is capable of assessing relative distance using the *Manhattan distance* principle [7] instead of the *Euclidian distance* due to the nature of the network structure.
- A scheduler, the "execution unit", checks the task graph iteratively and checks the status of tasks, the node element responsible for the task will schedule the task if the required data has been received. The scheduler built into each network node and link will mark down the process and check its current schedule if the proposed time slot is open. If the time slot is occupied or overlapping, the next task will be scheduled after the currently last process is scheduled to end. Links used in communication processes will use the start time of the entire process for all links in the sequence, blocking the entire route for the duration of the transfer.

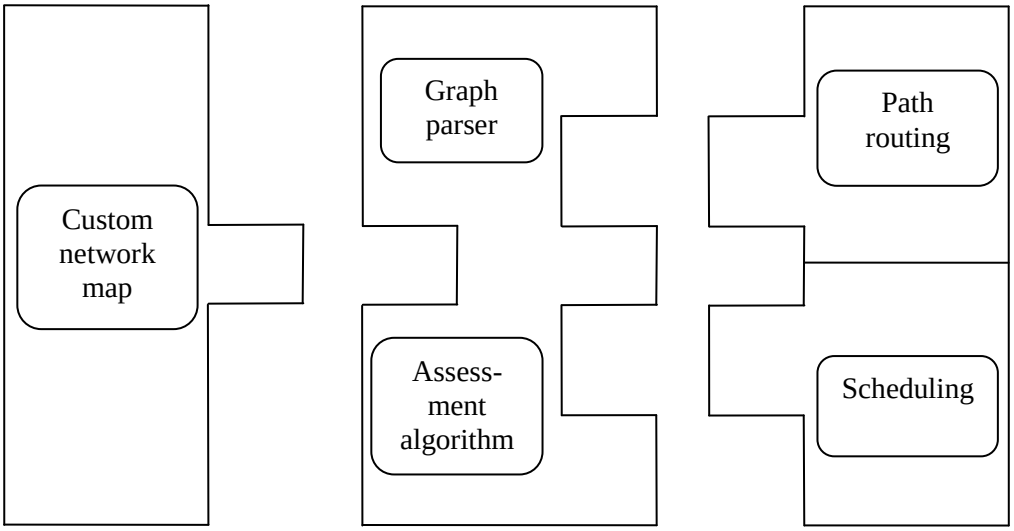


Figure 9: Test system layout

Test system will use a generated task graph file and platform dimensions as variable inputs. The platform dimensions are currently defined at compile time and have to be changed manually, for the duration of the tests for this thesis the platform dimensions will be 3x3x2.

3.2.3. Traffic assessment method

The objective of this method is to assess the effectiveness of adding/removing TSVs in a 3D NoC systems in a shorter time span compared to more exhaustive search algorithms while providing solutions close to global minimum. The method is iterative – by adding or removing a horizontal interconnection on every iteration from the structure, after the current state has been evaluated, allows for the viability assessment of each TSV using communication traffic as the evaluation variable. For this method to work, on the given platform, several other algorithms/methods have to be implemented that are explained in detail below.

3.2.4. Communication routing algorithm

When choosing the routing algorithm there are several path-finding algorithms besides the regular NoC routing rule types, as the scheduling method is non-dynamic and the mapping is done prior to execution. Although there are no hard deadlines implemented, speed and efficiency are required so that tests can be carried out with systems of all sizes. As the search space is three dimensional and direct path might not be available considering the non-uniform layout making traversal paths non-constant throughout the testing phase, regular NoC routing algorithms like XY or turn models are not admissible. Due to the lack of 3D specific routing algorithms a time efficient shortest path search algorithm will be used and no routing path restrictions will be applied as oblivious routing will be used. The choice between the following three different routing algorithms will be considered.

- A* (also known as A-star) algorithm is a general use path finding algorithm and is widely used for its simplicity and application possibilities. Given a method to find a node's neighbors and a relative distance assessment function, it is possible to traverse and search any type and shape of system for the shortest path by generating a search tree using the node's neighbors as branches. A* is a greedy algorithm that chooses nodes that seem closest to the destination node, using heuristic methods to ascertain the relative distance to the target node and the distance already traveled. The heuristic method h(x) uses calculation methods such as Manhattan distance or Euler's distance based on the search space and requirements of the problem. The complexity of the A* algorithm depends on the heuristic method used.
- Dijkstra's shortest path algorithm is a simple graph traversal algorithm that always chooses the shortest path among all possible paths. The search space has to be deconstructed into a graph. In directed acyclic graphs it is possible to find shortest paths from a given starting vertex in linear time, by processing the vertices in a topological order, and calculating the path length for each vertex to be the minimum length obtained via any of its incoming edges. The algorithm is less effective compared to A* for large search spaces due to the complexity of O ($|E|\log_2|E|$), where E is the set of edges in the graph. Dijkstra's algorithm can be viewed as a special case of A* where the heuristic $h(x) = 0$.

- Bellman–Ford algorithm is weighted graph shortest distance search algorithm that can use graphs with negative edge weights. Bellman–Ford algorithm, similar to Dijkstra's algorithm, is based on the relaxation principle that refines the approximation of the correct distance gradually with more accurate values until eventually reaching optimum solution. The distributed form of the algorithm has already been used in some CISCO networking routers that use distance vector routing methods. Computation speed is slower compared to the other two shorte1st path algorithms and does not scale well for larger systems.

Given the fact that distance to neighboring nodes is always constant in a matrix and due to the parameterized system boundaries, the choice was made to use A-star over other shortest path algorithms.

3.2.5. Static priority list scheduling

Static scheduling is a list scheduling method that similar to critical path method uses longest paths to evaluate priorities in the list. Differently to a critical path algorithm however, static scheduling method calculates priority values by starting from the exit nodes of the graph instead of the starting node. Static value for a node is based on its execution time and the static level values of child nodes - by adding the highest static value from its child nodes to its own execution time creates a priority assessment value. This method was originally described in Hu's algorithm, but has been used in several other scheduling and mapping algorithms since [17] [19]. The original algorithm has been modified to accommodate task graphs without clear level indentations to fit the test platform.

The scheduling process starts by assigning static levels to all tasks and sorts the task list using descending static level values as its comparison parameter. After the initialization process, the scheduler checks each task individually in the given order if it can be executed or not - this is done by checking if all necessary data has arrived prior to its execution. Once a task has been deemed ready to be executed on the core it was assigned to, the secondary function of the scheduler will check if the given processing unit is free at that point in time. The scheduler will delay the execution of the task if the processing unit is

currently busy and will mark the task to be executed after the current task has been carried out. When the task execution has finished, all outgoing communication will be checked and carried out starting from the communication with the highest volume.

3.3. Simulated annealing

Simulated Annealing is a general probabilistic non-greedy algorithm for approximating the global optimum in a wide array of optimization problems [17]. The general idea behind simulated annealing comes from physics where each molecule tries to achieve a zero charge value by accepting or rejecting electrons. In condensed matter physics, annealing is known as a thermal process for obtaining low energy states of a solid matter using a heat bath. The general annealing process is described by the following two steps: the first process increases the temperature of the heat bath to a maximum value at which the material melts, followed by a slow careful process to decrease the temperature of the heat bath until the particles arrange themselves in the ground state of the solid. Hurried or interrupted cooling process would normally lead to a sub-optimal energy distribution between the molecules weakening the bonds which may result in a brittle state.

Simulated annealing accepts changes into higher cost states with a probability that decreases over time but always accepts changes that lower the cost in the system. The acceptance of cost states gives the algorithm the means of escaping local minima and the possibility of achieving the global minimum. The algorithm cannot guarantee the global minimum solution at all times and despite being slower than heuristic methods is often used for NP-complete problems with large search spaces to avoid more exhaustive methods. A general form of a simulated annealing algorithm can be found below.

3.3.1. General simulated annealing algorithm

In this chapter we introduce a very basic Simulated Annealing algorithm and explain how the algorithm works. The following pseudo code contains all aspects of Simulated Annealing but needs to be adjusted to a specific problem.

1. $S \leftarrow S_0$ (Set initial state)
2. $C \leftarrow Cost(S_0)$ (Cost of initial state)
3. $C_{best} \leftarrow C$ (Set initial cost)
4. $T \leftarrow T_0$ (Initial temperature)
5. For $i \leftarrow 0$ to ∞ {
6. $T \leftarrow Cooling(T0,i)$ (Calculate temperature)
7. $S_{new} \leftarrow Move(S)$ (Perform move operation)
8. $C_{new} \leftarrow Cost(S_{new})$ (Calculate cost of new state)
9. $\Delta C \leftarrow C_{new} - C$ (Evaluate new state)
10. if ($\Delta C < 0$ OR $(0,1) < Acceptance(T, \Delta C)$){
11. if ($C_{new} < C_{best}$){
12. $S_{best} \leftarrow S_{new}$ (Chosen as current best)
13. $C_{best} \leftarrow C_{new}$
14. }
15. $S \leftarrow S_{new}$ (Move is accepted)
16. $C \leftarrow C_{new}$
17. }
18. if (End condition met){ (End condition)
19. break loop
20. }
21. }

The algorithm starts by assigning an initial state and parameters (1-4) that can be user defined or random, the initial state defines the starting position and influences the run time of the algorithm greatly. The initial state is evaluated and assigned as the current best solution (2-3). The iterative loop that follows (5) generates neighboring states and evaluates them by changing only a single detail in the current state (7-8). The new state can be accepted as the current state or rejected, depending on the evaluation function (10). The evaluation function always accepts moves into better states but also accepts moves into worse state using an acceptance function (10). The chance of accepting a new state depends on the "temperature value" that decreases over time (6), ultimately lowering the chance of accepting moves into worse states the longer the algorithm loop is executed. If the current best state is evaluated as worse than the new state it will be replaced with the new state (11-13). The length of the execution is depends on the end condition(s) defined by the user (18) and are often specifically tailored for the problem being solved. Steps 6 to 17 will be repeated until end condition in step (18) is met.

End conditions can include number of consecutive rejected moves, a final temperature or a final accepted cost which the algorithm has to achieve before it can be terminated. Since the objective is not targeted at real-time operations, final temperature and final acceptable cost

value would normally be priority choices, however consecutive rejects is by far the most popular choice. The functions and conditions used in the simulated annealing algorithm are subject to change based on the objective of task the algorithm is meant to perform. The right choice of the function procedures and variables is necessary for optimal efficiency of the algorithm yet difficult to determine.

A *problem oriented tailored* simulated annealing algorithm is needed to fit the specifics of the current problem. As there is more than one base state, adding/removing a TSV creates a new base state for each number of TSVs, a second iterative loop has to be generated to assess all of the possible layout solutions, effectively search through most of the search space with less time than exhaustive tests.

3.3.2. Revised simulated annealing algorithm

The revision to the simulated annealing algorithm is carried out to using a two stage process. The main purpose for the outer loop is to generate layouts with different number of vertical interconnections, while the main loop generates slight differences in the base layout provided within the iterations of the first stage. Using the two stage algorithm makes it possible to go through all viable combinations in the search space with less time compared to an exhaustive search, however reaching global minimum is not always guaranteed.

```
S ← S0 ;                                    //Set initial state
C ← Cost ( S0 );                            //Cost of initial state
Sbest ← S0;                                 //Set best state
Cbest ← C;                                  //Set best cost
T ← T0;                                     //Initial temperature
For i ← 0 to ∞ {
S ← Move1 (Sbest);                          //perform move operation 1
C ← Cost ( S );                             //calculate cost of next
                                            Iteration state cost
      For k ← 0 to ∞ {
            T  ← Cooling (T0,k);            //Calculate temperature
            Snew ← Move2 ( S );             //Perform move operation 2
            Cnew ← Cost ( Snew ) ;          //Calculate cost of new state
            ΔC ← Cnew – C;                  //Evaluate new state

            if ( ΔC < 0 OR (0,1) < Acceptance (T, ΔC)){
                  if ( Cnew < Cbest ){
                        Sbest ← Snew;       //Chosen as current best
                        Cbest ← Cnew;
                  }
```

```
                    S ← S_new;                  //Move is accepted
                    C ← C_new;
              }
              if (0,5 > T){                     //Inner loop end condition min
                    break loop;                 temperature value reached
              }

        }
        If(TSV_num < 1){                        //Outer loop end condition no
              break loop;                       TSV remain
        }
}
```

During the execution of this algorithm a number of random layout mappings will be generated for each number of TSVs to test out a larger portion of the search space and to follow the trend of the traffic based assessment algorithm. To optimize the search algorithm several parameters and functions have to be manually defined.

- **Cooling function –** The objective of the cooling function is to lower the simulated annealing temperature slowly. Given a simple cooling function (x), where $\alpha \neq 0$ is a cooling coefficient multiplied with the iteration temperature T_i, results in a limit value (x2) where the temperature T would endlessly come closer to 0 but would never reach it.

$$T_{i+1} = \alpha * T_i \quad (x)$$

$$\lim_{T \to 0} \alpha * T_i = 0 \quad (x2)$$

  The most common value for α coefficient in simulated annealing has been 0.95. The temperature related parameters are very important for the success of the algorithm and should therefore be tailored for the given problem. Specific testing on the platform with change in alpha coefficient has shown that lowering its value decreases the runtime of the algorithm but changes the exploration depth of the search algorithm slightly. Increasing its value increases the algorithm runtime significantly but shows no improvement. The algorithm currently outputs the boundaries of the maximum and minimum values the simulated annealing algorithm has registered during the runtime process; however they may not necessarily be the respective global minimum and maximum of the entire search space.

Tabel 1: Alpha coefficient tests

	Application runtime estimate(ms)								
TSVs	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9
9	926	926	926	926	926	926	926	926	926
8	926	926	926	926	926	926	926	926	926
7	926	926	953	953	926	926	953	926	951
6	888	926	888	926	888	888	888	888	888
5	881	881	881	881	881	888	881	881	881
4	900	900	900	913	900	900	910	900	900
3	933	942	933	933	933	933	933	933	933
2	1084	1084	1084	1084	1084	1084	1084	1084	1084
1	1387	1287	1287	1287	1287	1287	1287	1287	1287
Alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Beta	100	100	100	100	100	100	100	100	100
Time (sec)	0.295	0.312	0.321	0.398	0.455	0.56	0.638	0.97	1.806

Table 1 shows a small example of tests carried out using different alpha values, where the alpha value was increased with each test from 1 to 9. As simulated annealing is a probabilistic algorithm, result values may vary, but show little change in end result values but higher alpha values give more consistent and stable results. End results may also seem to be influenced by possible incompatibility with scheduling and path finding subroutines.

On closer inspection of the given problem to solve, to warrant more accurate search patterns the initial temperature should dynamically change throughout the algorithm. The reason for this lies in the fact that the number of possible TSV layouts changes every time one of them is removed and can easily be calculated by multiplying the number of remaining and the number of removed TSVs. Therefore the initial temperature will also be calculated using the following equation, where β is the user definable multiplication constant to keep the temperature from lowering too fast.

$$T_{initial} = \beta * TSV_{open} * TSV_{closed}$$

Several tests show that for this particular problem, increasing initial temperature changes, using a coefficient variable, has no impact besides increasing execution time. An example of the test can be seen in *table 2*. Beta coefficient value should therefore be as low as possible, but increased according to the application and platform complexity.

Table 2: Initial temperature tests

	Application runtime estimate(ms)								
TSVs	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9
9	1307	1307	1307	1307	1307	1307	1307	1307	1307
8	1218	1218	1218	1218	1218	1218	1218	1218	1218
7	1251	1226	1251	1251	1251	1251	1251	1251	1226
6	1195	1268	1195	1195	1195	1195	1195	1195	1195
5	1246	1276	1246	1246	1246	1246	1246	1246	1246
4	1293	1293	1293	1293	1293	1293	1293	1293	1293
3	1333	1333	1333	1333	1333	1333	1333	1333	1333
2	1511	1511	1511	1511	1511	1511	1511	1511	1511
1	2068	2068	2068	2068	2068	2068	2068	2068	2068
Alpha	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Beta	20	40	60	80	100	120	140	160	180
Time (sec)	0.647	0.731	0.769	0.807	0.852	0.892	0.894	0.952	0.955

- **Acceptance function –** The objective of the acceptance function is method that accepts a move into a worse state with a certain probability, which is the only means of finding a way out of a local minimum state to find the global minimum. The acceptance function is displayed below where ΔC is the cost difference between the current and the new state and T is the current temperature. The current acceptance function is based on Bolzmann probability principle.

$$\text{Acceptance}(\Delta C, T) = 1 / (1 + e^{(\Delta C/T)}))$$

- **Cost function –** The objective task of the thesis is to minimize the number of TSVs used relative to the total time it takes to execute the task graph on the system and

must therefore include these elements. A cost value per TSV could be inserted into the cost function to help minimize the total amount of TSVs.

$$T_{exec} = T_{task} + T_{comm}$$

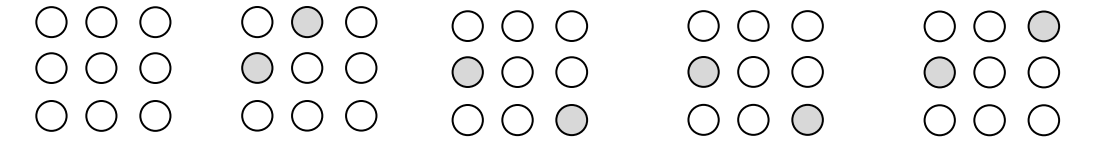
$$C = \alpha * T_{exec} + \beta * C_{area}$$

However the objective is to ascertain the effectiveness of removing/adding TSVs which will be carried out inside the inner loop of the new algorithm. Since cost effectiveness evaluation is carried on a broader spectrum of TSV placements, the amount of TSVs for each inner loop does not change and therefore does not influence the calculations at that stage. Alpha and Beta coefficients are there to lower or raise the importance of area cost resulting from adding/removing TSVs. The area cost is constant for the inner cycle where cost is calculated and is therefore ignored for the duration of this thesis to keep test system flexibility.

- **Move function 1&2 –** Move function one has the objective to deactivate TSVs each time it is called. When the first loop cycle creates the initial state for the loop nested inside it, it deactivates a TSV at random from the best state of the previous cycle. Move function two is responsible for changing to another neighboring state on every iteration of the first algorithms loop. This is done by turning an active TSV to inactive and vice versa for an inactive TSV, therefore changing the position of an active TSV link. The function creates two lists, one is filled with active TSVs and the other is filled with TSVs that have been deactivated. Upon execution of the move function a TSV is chosen from the first list and turned inactive followed by turning another TSV from the second list active, effectively moving the position of one of the TSVs.
- **End condition –** The end condition is another parameter that can be user defined. The end condition of rejected moves is a good solution for problems where algorithm runtime is of great importance and the search space is linear. Due to the added secondary objective of assessing the layout search space separately for each

number of TSVs, creating several sets of search spaces, each with different size. Due to the fact that the initial temperature is bound to the number of possible layouts for each search space set, a final temperature option becomes very lucrative. Several short test have shown that the final temperature should be relatively low to allow proper search space exploration and is set to 0.5 for the duration of this thesis.

The algorithm starts by initializing parameters – initial temperature, initial state and initial cost, the latter two are marked as part of the current best solution. When analyzing the initial state where the platform is fully uniform *Figure 10 (a)*, all TSVs are open (TSV count max) and therefore, during the initial state, only one possible layout exists. This layout is handled as an exception prior to the execution of the algorithm. The tasks from the test application graph are then mapped to selected processing units; the processing unit for each task is currently selected at random to lower test system complexity. To evaluate a network layout state, the test application is sent to the scheduler. The scheduler creates a static priority list out of the task graph and iteratively goes through the list until all tasks and communication events are scheduled. The end time of the last task is the execution time of the task graph.



(a) Initial layout Exec: 987 (b) 2nd Base layout Exec: 1081 (c) New layout Exec: 1058 (d) Best layout Exec: 1058 (e) Best layout(2) Exec: 1090

Figure 10: Example layouts

Each iteration in the first cycle disables a TSV on the current best layout at random (TSV count -1 on each iteration), evaluates the state and appoints it as the current state and best state for the upcoming annealing loop, along with the evaluated execution cost values. An example *Figure 10 (b)* shows a TSV layout after two TSVs have been removed. After this, the temperature for the annealing loop is calculated The new best state is the initial state for the second simulated annealing algorithm loop that changes the layout slightly, by rearranging a single TSV from one place to another on every iteration as can be seen on

Figure 10 (c). The initial temperature is calculated for each secondary loop based on the total number of possible layouts to improve search space exploration. After the new state is evaluated, a cost difference between the current state and new state is calculated by subtracting the current cost from the new cost. If the resulting delta value is negative or equal to zero, meaning the new state has a faster execution time, then the new state is accepted right away *Figure 10 (d).* If the new state has a higher execution time, then the state can still be moved into based on a probability comparison of a random number between 0 and 1 and the acceptance function return value that also is between 0 and 1. When a move is accepted into a worse state *Figure 10 (e)*, it is possible to find states that would otherwise be ignored. If the accepted state's execution value is also lower than the current best state with the same TSV count, then the current best state is replaced with the new state. If the current temperature has gone below the user specified minimum, the secondary loop ends and the primary loop moves into the next iteration. The primary loop exits once the number of TSVs in the layout has gone below 1, where further executions are impossible if tasks of the test application are mapped onto several different layers.

3.4. Experimental results

Using the algorithm presented in Chapter 2.3 a series of tests were carried out on a testing platform programmed in C++. The experiments were carried out on a NoC simulation model with a 3D mesh topology. In order to evaluate the performance of the interconnection system in controllable conditions, several precedence graphs with and their respective communication volumes, were generated with by TGFF task graph generator [11] for testing purposes. To test the capability of the platform using the test applications several sets of tests need to be run to assess the effectiveness of the traffic based algorithm compared to the simulated annealing algorithm. Due to complications resulting from multiple layers of TSVs on the test platform, only two layers of cores will be implemented at any given test. The simulation model does not consider buffer latency. The Simulated annealing algorithms' abbreviation in henceforth SIM and traffic assessment algorithms' abbreviation is TAM. Several tests were carried out using the random task mapping generator and the traffic assessment method.

The first test set objective is to provide a comparison between several random task mappings, showing the application execution time differences. An example of the results can be seen in *Figure 11* using a platform with 9 cores and 59 tasks.

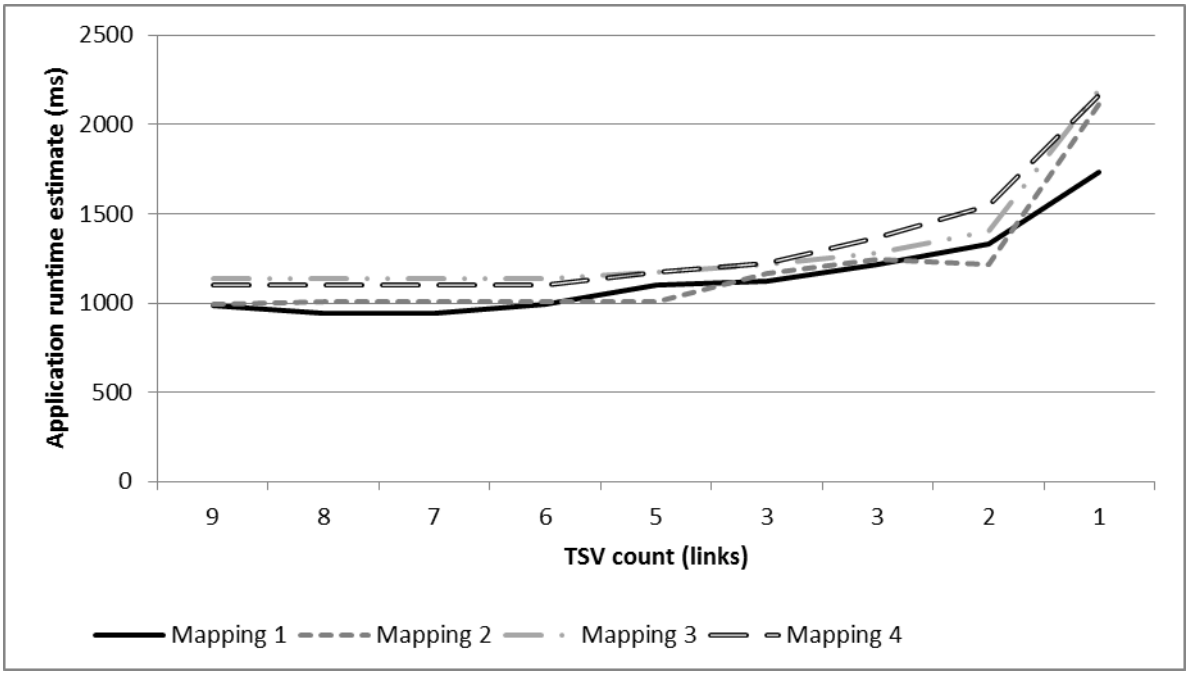


Figure 11: Traffic assessment algorithm output example

The task differences seem to influence the results slightly for TAM algorithm when choosing the TSVs to remove, which leads to the notion that mapping and scheduling method need to be in sync to find optimal execution times.

Secondary tests were carried out using the modified simulated annealing algorithm implementation for comparison purposes to evaluate the effectiveness of the traffic based method. To ascertain accurate data, both methods were used in succession using the same random task mapping designations for each test cycle. Only the best (*SIM min*) and worst (*SIM max*) execution times were added to the output. The time it takes to execute the algorithm is presented in parenthesis behind the algorithm's name on the figures. The results in *Figure 12* include the maximum execution time values explored during the runtime of the simulated annealing algorithm. The area created with the upper and lower

bound values of the simulated annealing algorithm gives a good perspective of the search space being explored and should give a better understanding on the effectiveness of the TAM algorithm.

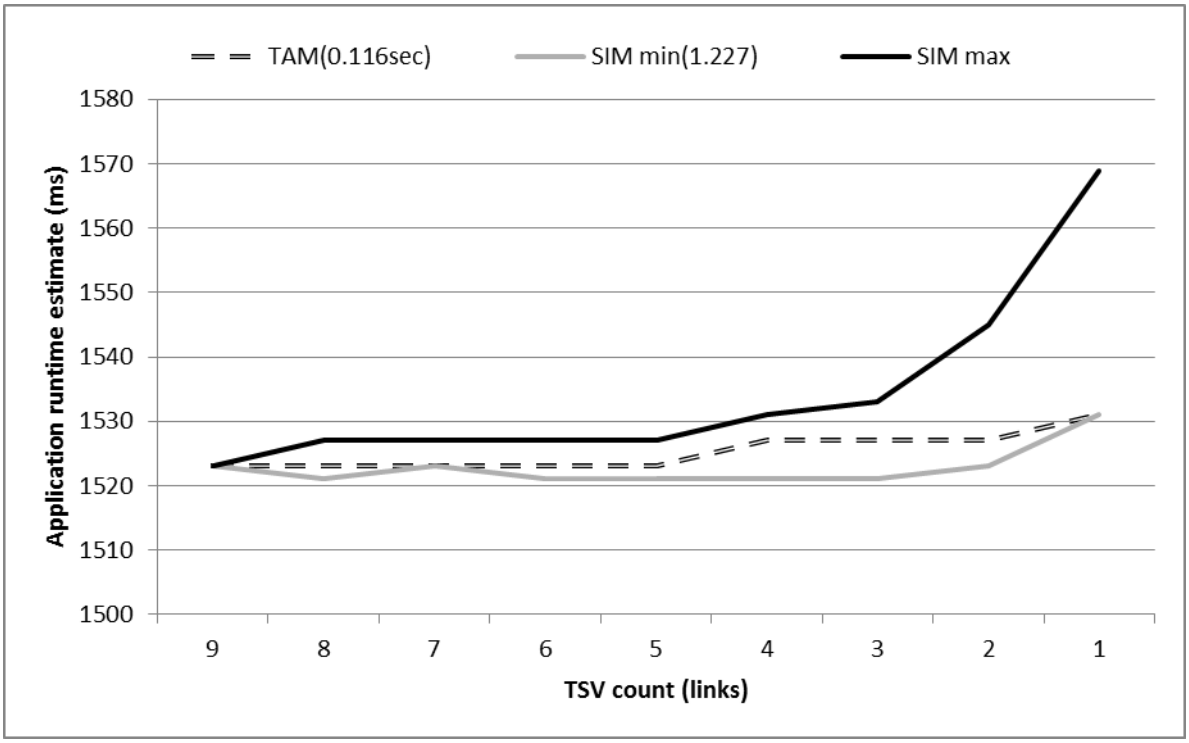


Figure 12: TAM and SIM comparison

APP2 was used with a 3x3x2 network platform. The results are displayed in *Figure. 12*. The results for test series 2 show that the TAM method does not perform as well as the SIM algorithm in terms of minimal execution time. The changes from higher execution time to lower execution time after removing a TSV are probably caused by the rescheduling and path mapping implementations and incompatibility between scheduling and mapping.

Figure 13 shows results for the execution of the same application on two platforms with different core counts. The results for the 18core platform (9TSVs) tests are slightly moved to the left for better comparison of execution times represented by a secondary axis line.

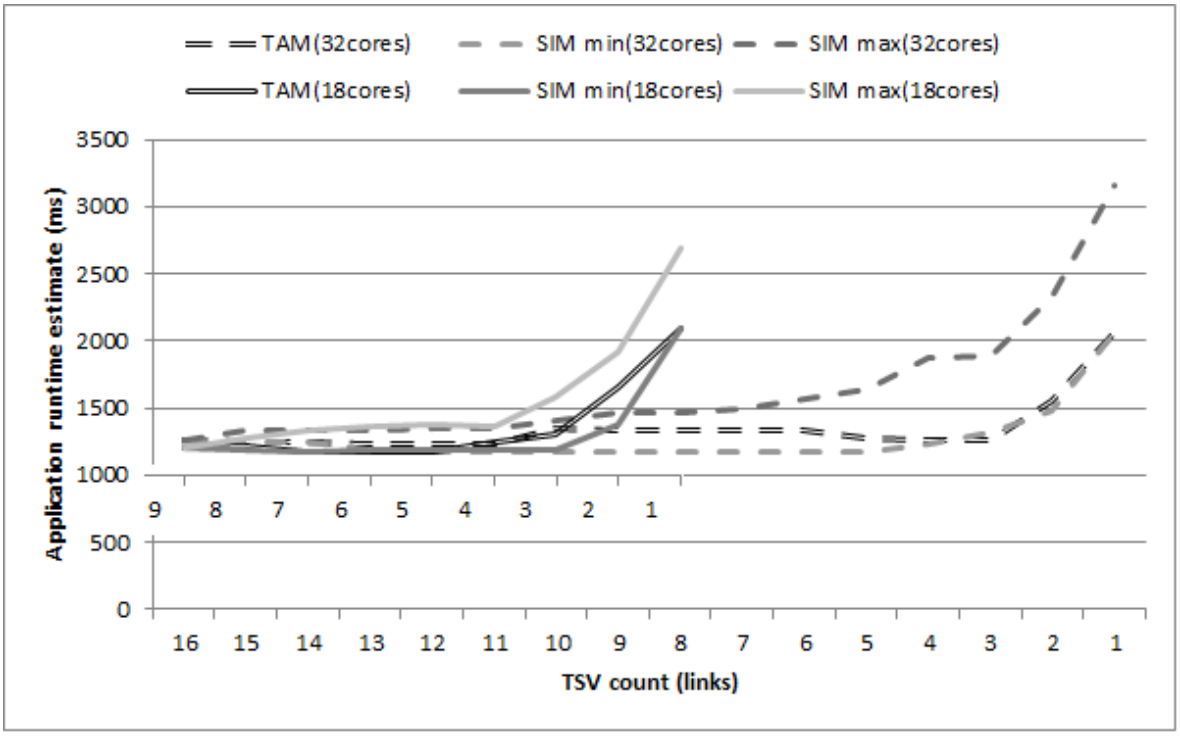


Figure 13: Traffic assessment and simulated annealing.

The results for *Figure 12* tests show that the traffic assessment algorithm, being a heuristic method, can be executed several times faster than the simulated annealing algorithm. The execution time for both layouts in *Figure 13* show similar test results to the simulated annealing algorithm for, but shows slightly larger best and worst case scenario discrepancies. Closer analysis suggests random mapping differences as the main cause, confirms the need for a better cooperation between mapping and scheduling algorithms. The conclusion for the tests show that without the need to find the global minimum execution times, it is possible to quickly and with a sufficient accuracy assess possible impact of adding or removing TSVs from a 3D NoC layout.

The test series in *table 3* was done with different core numbers per layer using the same test application. The objective of this test set is to see how the number of cores affects the results of the execution of the test application and observe the differences between TAM and SIM algorithms. Runtime in *Table 3, columns 2 and 5,* signify the application's estimated runtimes for the TSV count found in *column 3*. The time it takes to execute the SIM and TAM evaluation algorithms can be found in *columns 4 and 6* interlinked with core

count in *column 1*. For example for a platform of 8 cores the evaluation using SIM algorithm takes 0.141sec to execute while TAM evaluation algorithm only takes 0.027sec to evaluate the same search space. TAM execution (*Table3, column 4*) and SIM execution (*Table3, column 6*) shows that simulated annealing algorithm execution times grow at different speeds compared to each other, the higher the number of cores and TSVs grows.

Table 3: TAM and SIM comparison

cores	TAM runtime [ms]	TSVs [links]	TAM execution [s]	SIM runtime [ms]	SIM execution [s]
8(2x2x2)	970	4		970	
	1029	3		1029	
	1182	2		1139	
	1902	1	0.027	1646	0.141
18(3x3x2)	965	9		965	
	965	8		912	
	942	7		909	
	1023	6		909	
	990	5		909	
	990	4		912	
	1140	3		933	
	1307	2		1251	
	1561	1	0.064	1561	0.809
32(4x4x2)	1261	16		1261	
	1241	15		1241	
	1241	14		1241	
	1224	13		1189	
	1224	12		1176	
	1224	11		1176	
	1353	10		1176	
	1337	9		1176	
	1337	8		1176	
	1337	7		1176	
	1337	6		1176	
	1269	5		1176	
	1265	4		1233	
	1265	3		1319	
	1549	2		1483	
	2064	1	0.168	2064	2.873

While TAM execution time growth is linear, the execution time of the simulated annealing algorithm growth multiplier however seems to be exponential. The runtime of both algorithms increase each time core number is increased as the search space and minimal number of state evaluations increases. The fact becomes clear, that the larger the search space becomes, the larger the difference of the TAM and SIM algorithm's execution times grows. Considering the runtime of both algorithms and the results that can be seen in both *table 3* and *Figure 12* prove again that the simulated annealing algorithm lower bound hold slightly lower cost values but has the runtime of the algorithm times longer than TAM.

The test series, which example can be found in Figure 14, shows comparison results for TAM and SIM algorithms with a slight change in SIM algorithms regarding the removal of TSVs for each primary cycle. The simulated annealing algorithm was modified to remove TSV similar to the TAM algorithm instead of removing one at random.

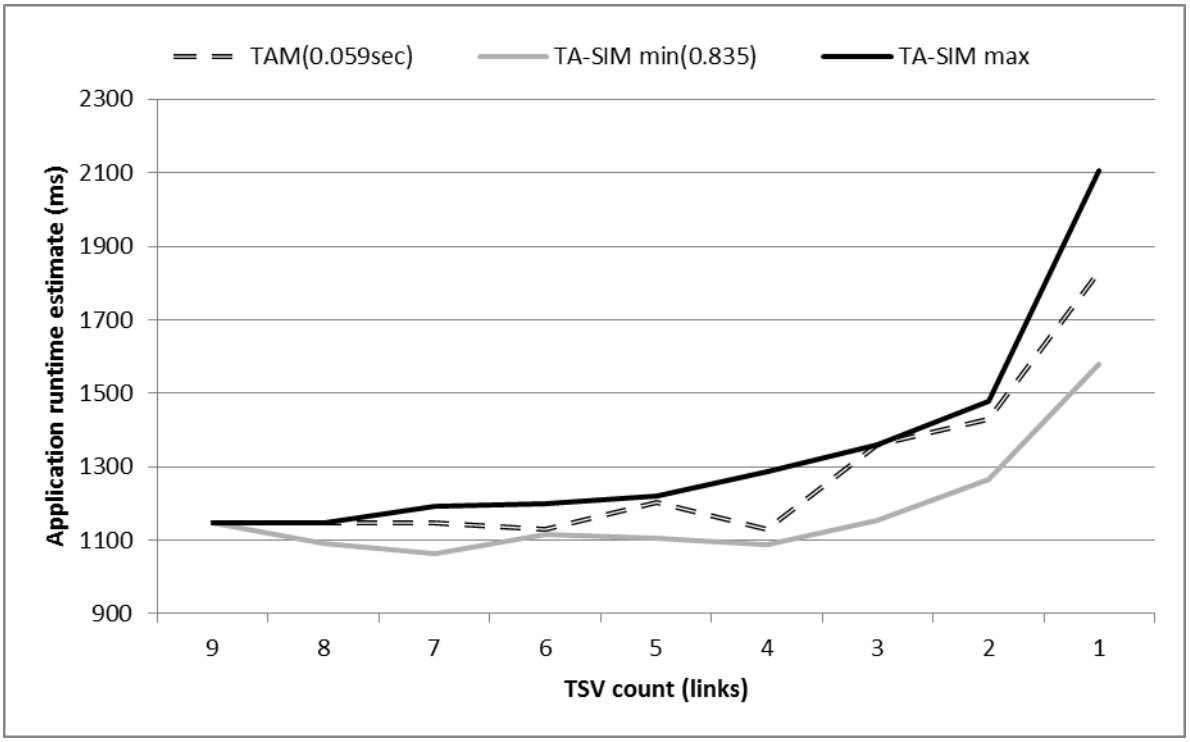


Figure 14: Traffic assessment and simulated annealing. Test series3

The results show a marginal decrease in execution times on both upper and lower bound and a slight increase of ~0.030sec in algorithm runtime compared to the original SIM algorithm.

The next test set was carried out using an application with task count several times higher than the number of cores to see how the application influences the results if the platform size remains low. Using a test application APP3 was used for test series 4, that has over 200 tasks ranging from 10 to 20 ms execution times and containing communication events with 90-120 packets. The platform has 2 layers and total IP count of 32.

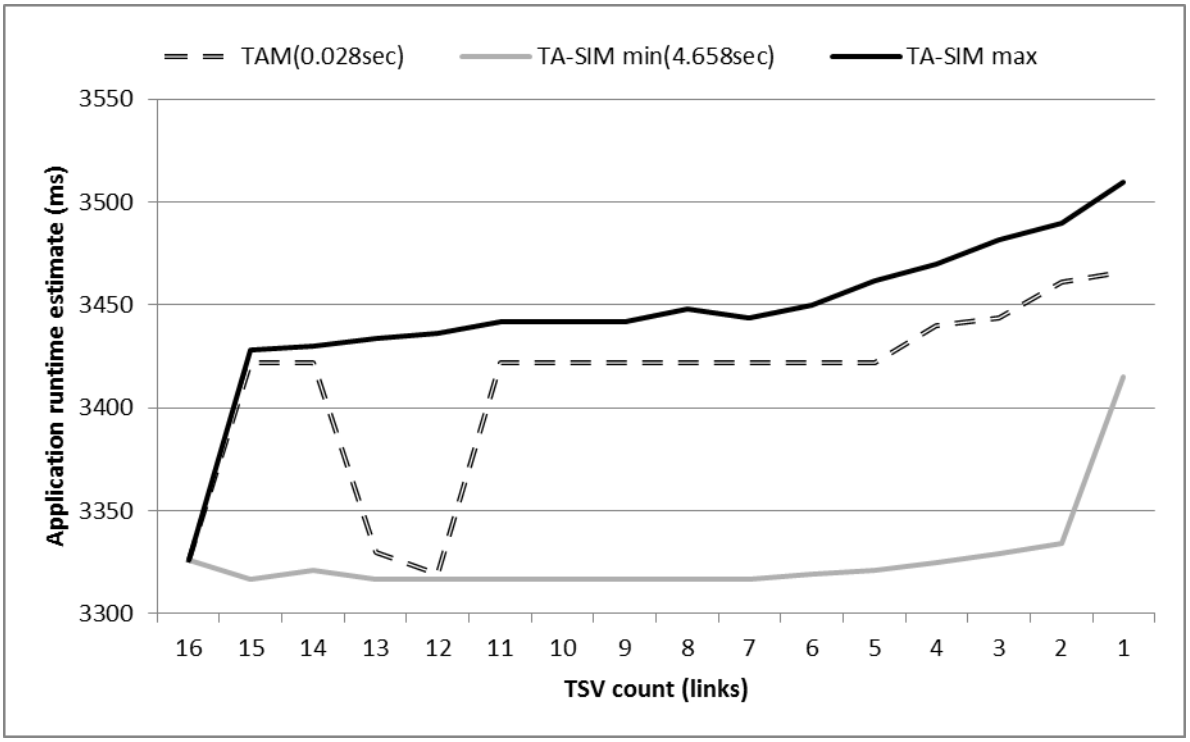


Figure 15: Traffic assessment and simulated annealing, test series 4

The difference between the algorithms has become more severe. *Figure 15* shows a significant spike in TAM algorithm's performance, but the overall performance shows the superiority of the simulated annealing algorithm. The spike in execution time shows the importance of choosing the right TSV to remove, but also shows that less time analyzing the search space gives sub-optimal results. When considering the *graceful degradation* principle where the choice of which TSV is removed cannot be predicted, the choice of

proper scheduling and communication mapping becomes even more critical. A non-virtual test environment would provide more accurate results, but cannot be constructed due to the lack design tools and manufacturer at this time.

The next set of tests was carried out to ascertain how the increase of core count influences the execution of APP3 that has high task count. *Table 4* shows the results from using APP3 in extensive tests over several platform sizes ranging from 8 cores to 50cores. Runtime in *Table 3, columns 2 and 5,* signify the application's estimated runtimes for the TSV count found in *column 3*. The time it takes to execute the SIM and TAM evaluation algorithms can be found in *columns 4 and 6* interlinked with core count in *column 1*. Comparing algorithm execution times between TAM (*Table4, column 4*) and SIM (*Table4, column 6*) shows that simulated annealing algorithm takes several times longer, the higher the number of cores and TSVs gets, the same conclusion that can be drawn from *Table 3*. *Columns 2 and 5* show that the smaller the platform is the longer the application takes to run as expected of the system.

Table 4: TAM and SIM comparison using APP3

cores	TAM runtime [ms]	TSV [links]	TAM execution [s]	SIM runtime [ms]	SIM execution [s]
8(2x2x2)	5429	4		5429	
	5441	3		5431	
	5445	2		5433	
	5447	1	0.039	5445	0.622
18(3x3x2)	3773	9		3773	
	3773	8		3674	
	3773	7		3674	
	3773	6		3672	
	3771	5		3672	
	3775	4		3678	
	3781	3		3601	
	3785	2		3605	
	3791	1	0.149	3640	3.57
32(4x4x2)	3427	16		3427	
	3427	15		3427	
	3427	14		3427	
	3427	13		3356	

	3427	12		3358	
	3427	11		3358	
	3427	10		3356	
	3427	9		3354	
	3431	8		3271	
	3431	7		3271	
	3362	6		3269	
	3362	5		3277	
	3362	4		3354	
	3368	3		3283	
	3372	2		3291	
	3495	1	0.309	3302	17.445
50(5x5x2)	3075	25		3075	
	3075	24		2839	
	3075	23		3075	
	3075	22		2839	
	3075	21		2806	
	2852	20		2852	
	2852	19		2835	
	2856	18		2794	
	2856	17		2852	
	3081	16		2831	
	3081	15		2796	
	3081	14		2799	
	3083	13		2836	
	3083	12		2843	
	3083	11		2802	
	3083	10		2839	
	3077	9		2811	
	3075	8		2840	
	3079	7		2836	
	3081	6		2814	
	3085	5		2798	
	3085	4		2813	
	3089	3		2851	
	3101	2		2881	
	3107	1	0.621	2906	57.879

The decrease in application execution time with higher core count is to be expected. The spikes in execution time however can only be observed when using a system with 50 cores.

This fact leads to the conclusion that complications arise from increasing complexity of the scheduling process and the fact, that test application graph branching leads to more than a single end task.

The following test is a replication of the execution time spikes to give a visual representation for the results in *Table 4*. The test used a 5x5x2 platform and APP3, the maximum task count and core count the test platform could handle. Erratic change in the minimal execution times found by the simulated annealing algorithm can be observed when the TSV count reached 23. The execution time of the application for TAM algorithm suddenly drops when the TSV count is between 20 and 17 TSVs.

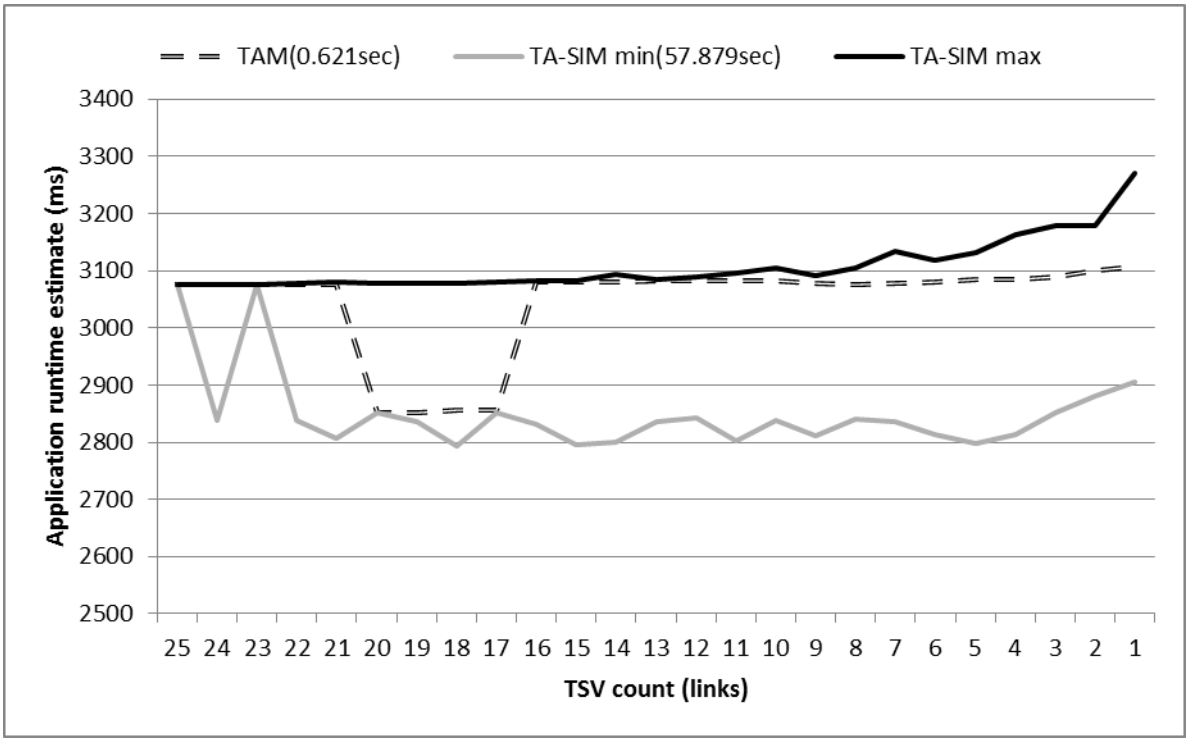


Figure 16: APP3 comparison using a 5x5x2 platform

Figure 15 uses APP3 but employs a 5x5x2 platform with 50cores. The cost spike can also be noted here that was present in *Figure 14*. This phenomenon was present at every test that used more than 100 tasks and might hint to a problem in the test system. However the results of the TAM algorithm still fall within simulated expected ranges defined by the SIM algorithm. Further testing in a real environment would be beneficial.

3.5. Conclusion

Unfortunately the platform size could not be increased beyond 5x5x2 and 200 tasks as memory leaks due to data structures prevented testing beyond the given parameters. Redesigning the test platform might yield testing capability with larger systems and applications. The test platform shows abnormal behavior when the number of TSVs is still relatively high, this phenomenon might be the caused by the inflexibility of the scheduling and communication path mapping algorithms. The path mapping algorithm used is designed to find an optimal path in a short time using relative position to the destination, however it might not always be the shortest path. The behavior does not seem to be related to the max number of TSV links in the system.

When comparing the execution times of SIM and TAM algorithms, it becomes clear that the linear scaling of the TAM execution time becomes a great advantage with higher TSV count where the search space is larger, if a general overview is needed. However SIM algorithm provides lower global minimum results compared to TAM with improving results the larger the search space becomes. Looking at the test results, depending on the aim and requirements of the assessment, both algorithms are be viable. The viability of TAM in real-time constrained dependable systems assessment can be observed if the objective is to ascertain if the system can maintain a minimal load requirement.

Several problems during testing indicate the need to choose better mapping and scheduling algorithms that have a high compatibility to each other. Further testing of the TAM algorithm would be advised using a more accurately represented test system. Further testing may be required if dynamic scheduling methods or adaptive routing methods are used as they would severely change the test results.

4. Summary

The technological advances in integrated circuit technology have started to slow down as we reach the physical limitations to technological scaling with our current manufacturing technologies. However the ever increasing need performance and functionality is pushing scientists to search for new ways to overstep our boundaries. Several research directions that have been taken, have already shown good test results, but have yet to be proven applicable in mainstream technology.

The current most promising solution to fulfill the requirements of our society is 3D chip stacking technology. The strongest positive influences of this technology can be summarized as the following. Due to wafer thinning, resulting vertical chip thickness is several times lower than initially estimated. Using this method potentially increases the chip's performance and power consumption used for signal propagation. The design can be built as separate parts and integrated later into a single chip. Silicon interposers enable the reusability of old designs with similar footprints, old research and development tools can still be used.

The opportunities 3D stacking technology provides do come with a set of challenges and limitations. Namely power consumption may be lower, but increase in power density creates problems in thermal management. Vertical interconnections increase the chip area by a small amount, resulting in a wire delay increase, which in turn leads to a limited number of interconnections for a design depending on the properties of the interconnections used. To mitigate the problematic aspects of using Through-Silicon-Vias in Network-on-Chip technology a method was devised to assess the viability of each TSV in a given design.

The proposed TSV impact assessment method can employ and adapt to a wide variety of scheduling and mapping algorithms on a network layout with any number and structure of nodes, given the assumption that the design's interconnection placement can change. However, the method assumes that the graph can be fully explored and contains

concurrency. Given a method to compare and assess systems with similar interconnection layouts quickly, allows for the improvement of reconfigurable and dependable 3D systems. The conclusion can be drawn that the Traffic Assessment method can be used to quickly reassess the capabilities of a 3D NoC system to ascertain if the system is capable to continue operation performing to a given minimal load. The viability of simulated annealing algorithm can still be observed if the system has low real-time constraints and reassessment of the system is not immediate.

Resümee

Tehnoloogilised saavutused integraalskeemide vallas on hakkanud vaikselt aeglustuma ja on jõudnud füüsiliste piiranguteni tingitud füüsikalistest limiitidest, mille tõttu ei saa elektroonika komponentide suurusi enam oluliselt vähendada. Sellest olenematta tõuseb nõudlus jõudluse ja funktsionaalsuse järele, mis on lükanud käima suurel hulgal uusi laialdasi uurimistöid erinevates valdkondades, otsides viise kuidas ületada tehnoloogia barjääri. Paljud uurimis suunad on andnud tulemusi , kuid pole veel suutnud ennast tõestada.

Hetkel parimaks tehnoloogiliseks järeltulijaks, täitaks ühiskonna vajadusi, on 3D kiipvirnastus tehnoloogia. Tema tugeivamad mõjualad saab kokkuvõtta lühidalt. Planaarne 2D kiipe saab tükeldada väiksemateks plokkideks ja virnastada üheks kiibiks, vähendades nende vahelisi ühenduste pikkusi ja signaali juhtimiseks vaja minevat energia tarvet. Tänu kiipide hõrendamisele, mis toob endaga kaasa on kiibi vertikaalse mõõte suurenemise, saavutuatud pindala võit suurem kui planaarsel asetusel. Tänu 2.5D tehnoloogia omadusele on võimalik kasutada vanu arendus tööriistu ja toomis vahendeid, langetades prototüübi väljaarendamise maksumust. Disainide osad saab luua erladi projektidena ja taaskasutada juba olemasolevaid disaine.

3D kiipvirnastus tehnoloogia omab samuti palju väljakutseid ja piiranguid. Nimelt väiksemat energiatarvet asendab energia tiheduse probleem mis omakorda püstitab temperatuuri probleeme.Vertikaal ühendused tõstavad omakorda vajaminevat kiibi pindala, mis tõstab viivitusi tulenevalt juhtmete pikenemisest. Juhtmete pikenemise tagajärgede vältimiseks on püstitatud limiit vertikaal ühenduste arvule, mida kokku kasutatakse nii energia jaotuseks, temperatuuri alandamiseks kui ka andmevoogude tarbeks. Selles töös koostati meetod TSV-de hindamiseks kiipvõrkudes, et aidata vähendada selle tehnoloogia negatiivseid tagajäregesid.

Koostatud TSVde hindamis meetod suudab kasutada palju erinevaid aja- ja ülesannete planeerimisalgoritme, ükskõik millise kiipvõrgu kuju ja protsessor tuumade arvu jaoks.

Arvesse tuleb võtta eeldust, et vertikaal ühenduste arv ja asukohad peaksid olema muudetavad ja meetmed nende teostamiseks olemas. Siinjuures peaks programmi graaf olema täies mahus teada ning omama paraleelselt töödeldavaid ülesandeid. Eeldades, et meil on meetod mille abil hinnata süsteemide sarnaste vertikaal ühenduste paiknemist ja effektiivsust kiiresti, lubab fukntsionaalsust hinnates uuesti konfigureeritavaid ja töökindlaid 3D süsteeme luua ja tõsta süsteemi töökidlust vigade esinemisel. Töö käigus jõuti järeldusele, et välja pakutud liiklus-põhine hindamismeetodit saab kasutada 3D kiipvõrk süsteemide jõudluse kiireks uuesti hindamiseks, et kinnitada süsteemi jätkusuutlikust vähemalt defineeritaval minimaalsel koormusel. *Simulated Annealing* algoritmi kasutamise võimalus eksisteerib süsteemidel, mis ei oma tugevaid reaalaja piiranguid, kus süsteemi jõudluse uuesti hindamine ei ole koheselt vajalik. Dünaamiliste paigutus meetodite kasutamisel tuleks liiklus-põhine hindamismeetod uuesti üle testida, kuna nende mõju süsteemile on tugev.

Used literature

1. Moore's Law [WWW] http://download.intel.com/museum/Moores_Law/Articles-Press_releases/Gordon_Moore_1965_Article.pdf (10.06.2013)
2. The Silicon Engine: A timeline of semi-conductors in computers [WWW] http://www.computerhistory.org/semiconductor/timeline/1974-digital-watch-is-first-system-on-chip-integrated-circuit-52.html (10.06.2013)
3. Khaled Salah Alaa EI Rouby Rani Ragai Yehea Ismail: 3D/TSV Enabling Technologies for SOC/NOC: Modeling and Design Challenges – Microelectronics (ICM), 2010 International Conference on, Pages: 268 - 271 [Online] IEEEXplore
4. Parekh, Thadesar, Bakir. Electrical, optical and fluidic through-silicon vias for silicon interposer applications: Electronic Components and Technology Conference (ECTC), 2011 IEEE 61st, Pg 1992-1998 [Online] IEEEXplore
5. D. N. Jayasimha, Bilal Zafar, Yatin Hoskote: On-Chip Interconnection Networks: Why They are Different and How to Compare Them [Online] http://glearning.tju.edu.cn/ pluginfile.php/74998/mod_resource/content/0/ODI_why-different.pdf
6. Ding-Ming Kwai, Chang-Tzu Lin: 3D stacked IC layout considering bond pad density and doubling for manufacturing yield improvement: Quality Electronic Design (ISQED), 2011 12th International Symposium on, pg.1-6 [Online] IEEEXplore (21.05.2012)
7. Kirk Saban: "Xilinx Stacked Silicon Interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency" [WWW] http://www.xilinx.com/support/documentation/white_papers/wp380_Stacked_Silicon_Interconnect_Technology.pdf (02.09.2012)
8. Manhattan distance principle [WWW] http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Clustering_Parameters/Manhattan_Distance_Metric.htm
9. A-star Shortest Path Algorithm (C++ recipe) [WWW] http://code.activestate.com/recipes/577457-a-star-shortest-path-algorithm/ (02.09.2012)
10. Designing and Building Parallel Programs: Chapter 2. Mapping (1995) [WWW] http://www.mcs.anl.gov/~itf/dbpp/text/node19.html (10.08.2012)
11. Task Graphs For Free [WWW] http://ziyang.eecs.umich.edu/~dickrp/tgff/ (10.05.2013)
12. Le Yu, Haigang Yang, Jia Zhang, Wei Wang. Performance Evaluation of Air-gap-Based Coaxial RF TSV for 3D NoC: 2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip [Online] IEEE (03.03.2012)
13. Dae Hyun Kim, Krit Athikulwongse, Sung Kyu Lim. Electrical, Optical and Fluidic Through-Silicon Vias for Silicon Interposer Applications: 2009 IEEE/ACM International Conference on Computer-Aided Design Digest of Technical Papers pg.674-680 [Online] IEEEXplore
14. Nauman H. Khan, Syed M. Alam, and Soha Hassoun. Through-Silicon Via (TSV)-induced Noise Characterization and Noise Mitigation using Coaxial TSVs: 3D System Integration, 3DIC 2009. IEEE International Conference pg.1-7 [Online] IEEEXplore

15. James Burns. Chapter 2. TSV-Based 3D Integration: Three Dimensional System Integration: IC Stacking 13 Process and Design, Springer Science+Business Media, LLC 2011 [Online] SpringerLink. LNCS
16. Introduction to A*: From Amit's Thoughts on Pathfinding [WWW] http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html (10.05.2013)
17. Heikki Orsila. Optimizing Algorithms for Task Graph Mapping on Multiprocessor System on Chip: Thesis for the degree of Doctor of Science in Technology. Tampere, Tampere University of Technology 2011
18. Ciprian Seiculescu, Srinivasan Murali, Luca Benini and Giovanni De Micheli. 3D Network on Chip Topology Synthesis: Designing Custom Topologies for Chip Stacks: 3D Integration for NoC-based SoC Architectures, Integrated Circuits and Systems, Chapter 9. Pg 194-211 [Online] SpringerLink. LNCS (21.05.2012)
19. Sih, G. C. and Lee, E. A., "A Compile-Time Scheduling Heuristics for Interconnection-Constrained Heterogeneous Processor Architectures", IEEE Transaction on Parallel and Distributed Systems, Vol. 4, No. 2, pp. 175-187, 1993 [Online] IEEExplore
20. É. Cota et al., Reliability, Availability and Serviceability of Networks-on-Chip: Springer Science+Business Media, LLC 2012, Chapter 2 Pg.1-21 [Online] SpringerLink. LLC
21. Ville Rantala, Teijo Lehtonen, Juha Plosila. Network on Chip Routing Algorithms: TUCS Technical Report 2006 [WWW] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.120.8910&rep=rep1&type=pdf