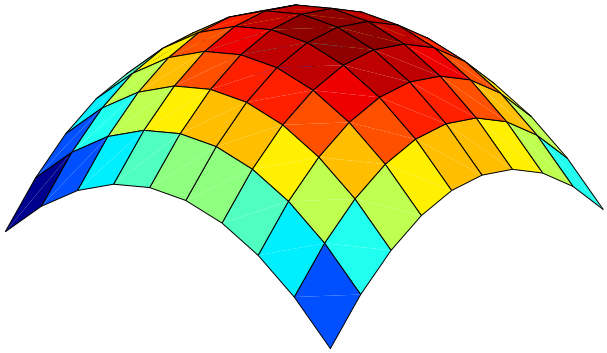
Linköping Studies in Science and Technology

Thesis No. 996

An on-chip network architecture for hard real time systems

Daniel Wiklund



LiU-TEK-LIC-2002:69
Department of Electrical Engineering
Linköpings universitet, SE-581 83 Linköping, Sweden
Linköping 2003

Linköping Studies in Science and Technology

Thesis No. 996

An on-chip network architecture for hard real time systems

Daniel Wiklund

INSTITUTE OF TECHNOLOGY
LINKÖPINGS UNIVERSITET

LiU-TEK-LIC-2002:69
Department of Electrical Engineering
Linköpings universitet, SE-581 83 Linköping, Sweden
Linköping 2003

ISBN 91-7373-577-9
ISSN 0280-7971

Abstract

With the ever increasing demands on processing power and communication on a single chip the industry is facing a huge obstacle in closing the gap between possible complexity and achieved complexity, the so called design gap. A possible path out of this is the increase (re-)use of intellectual property (IP) blocks from within the company or from other suppliers. We have identified the problem area in the on-chip communication between IP blocks where the time-division multiplex buses are quickly becoming saturated.

Another problem arising with the increased use of deep submicron manufacturing technologies is the relatively long delay of wires compared to the gates. This problem forces the synchronous part of a chip to either shrink or run at a slower speed. With the goals of keeping the clock rate and increasing the complexity the only feasible solution is to use smaller synchronous subsystems that communicate asynchronously. This approach is known as globally asynchronous but locally synchronous (GALS).

This thesis presents the work on a bus replacement for on-chip communication. The goal of this bus replacement is to achieve very high performance compared to the old solution while allowing for higher flexibility, GALS style implementation, and simpler verification of the system.

With this goal in mind we investigated the possible topologies for a switched on-chip network (OCN) and concluded that a 2-d mesh or torus is the most appropriate. To keep the latency low we decided on a pseudo-circuit switched network using the 2-d mesh. We have developed a novel approach for route setup in the circuit switched network called packet connected circuit (PCC) which allows very short latency both for routing and payload transfer while having a very low silicon cost.

A simulator for this network has been implemented together with behavioral models of the network components. Simulations have shown that the PCC concept is not very suitable for general purpose processing platforms but that it is very suitable for a hard real time system that uses some communication scheduling.

Preface

This thesis presents my research during the period from January 2000 through November 2002. The following three papers are included in the thesis:

- Daniel Wiklund and Dake Liu, “Switched interconnect for system-on-a-chip designs”, in *Proceedings of the IP2000 Europe conference*, Edinburgh, Scotland, 2000, pp 185-192
- Daniel Wiklund and Dake Liu, “Design of a System-on-Chip Switched Network and its Design Support”, in *Proceedings of the International conference on communications, circuits and systems (ICCCAS)*, Chengdu, China, 2002, pp 1279-1283
- Daniel Wiklund and Dake Liu, “SoCBUS: Switched network on chip for hard real time embedded systems”

The following publications related to the SoCBUS project are not included in the thesis:

- Daniel Wiklund, “Switched interconnect for embedded System-on-a-Chip signal processing designs”, in *Proceedings of the Swedish System-on-Chip Conference (SSoCC)*, Arild, Sweden, March 2001
- Daniel Wiklund, “Implementation of a behavioral simulator for on-chip switched networks”, in *Proceedings of the Swedish System-on-Chip Conference (SSoCC)*, Falkenberg, Sweden, March 2002
- Dake Liu, Daniel Wiklund, Erik Svensson, Olle Seger och Sumant Sathe, “SoCBUS: The solution of high communication bandwidth on chip and short TTM”, in *Proceedings of the Real-Time and Embedded Computing Conference*, Gothenburg, Sweden, September 2002

Acknowledgments

First of all I would like to thank my advisor, Professor Dake Liu. First of all for inviting me to become a Ph.D. student and second for the perpetual optimism and abundance of (mostly good) ideas.

I would also like to thank Professor Christer Svensson for all the support and all interesting discussions on my research as well as almost all possible other things.

Further I would like to thank the people involved in my project. Erik Svensson for his valuable work on the simulator implementation and Sumant Sathe for his work on wrappers and nodes.

There are a lot of people left to thank for molding the university into a much liked place to work. First of all, Professor Per Larsson-Edefors for being such a rascal and a gleam of light in an otherwise depraved world. And we also had some interesting *serious* discussions as well! Then the "inner circle" with Tomas Henriksson, Ulf Nordqvist, Mikael Olausson, Erik Tell, Stefan Andersson, Peter Caputa, Henrik Eriksson, and Daniel Eckerbert. And I especially would like to thank Kalle Folkesson for creating a sound environment of puns. It's a shame that some of you took to the road that leads to the west!

I would also like to thank all my other current and former coworkers at Electronic Devices and Computer Engineering for making the time here at Linköpings universitet a very enjoyable time.

Last but definitely not least I would like to thank my parents Erling and Anna Wiklund for supporting me and coping with me still living in their house!

This thesis work was sponsored by the Swedish Foundation for Strategic Research (SSF) through the Integrated Electronics Systems program (INTELECT).

Daniel Wiklund

Contents

Part I

Introduction

Chapter 1

Introduction

1.1 Background

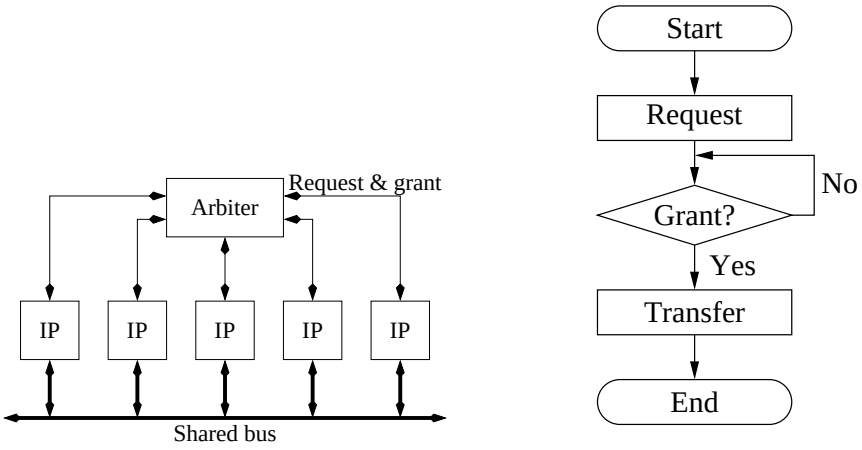
As the transistor feature size continue to decrease the level of integration has continued to rise. Still with no end to the scaling in sight the industry is faced with tremendous demands on the chip designs. There are several faces to this beast where the design productivity gap and the verification complexity together with the sheer cost of failure are the most pronounced.

The cost of failure, e.g. lost time-to-market (TTM) and extra labor cost for fixing the errors, is hard to tackle. The only way to avoid this is to make sure that the chip is correct on the first try.

With this background it is easy to understand that the only feasible way to success in the future will be the increased (re-)use of intellectual property (IP) blocks. The IP blocks can be of any type, from processor cores (e.g. ARM processor) via programmable and configurable functions to entirely hardwired ASIC functions. The IP blocks come in two flavors, hard and soft IP. The hard IPs consist of complete layouts which are foundry dependent while the soft IPs in principle are sets of RTL code that can be synthesized for any semi-custom cell library and the soft IPs are thus foundry independent.

With a roughly constant clock frequency in the IP blocks and the increasing impact of wire delay due to process scaling it has been predicted that the size of the IP blocks will stay roughly constant. A maximum of 250 k gates per block at 1 GHz should be feasible without insurmountable difficulties [1]. The way to increase the functionality of the chips with this limitation in mind is not to increase the complexity of each IP block but rather to increase the number of IP blocks on the chip. With this development it is clear that the on-chip inter-block communication will become significantly heavier over time.

As the design cost of complex integrated circuits will become ever higher in



(a) TDM bus topology

(b) TDM bus protocol

Figure 1.1: Network topologies

the foreseeable future there will be an increasing demand for flexibility in the chips allowing them to fit more and more applications without redesign of the hardware.

In the light of these predictions and current problems it makes sense to develop a novel integration procedure that will allow higher design productivity, increase the flexibility, and cut the time-to-market. This thesis presents a system for on-chip communication and presents the general method of usage together with some simulations on the system performance.

1.2 Traditional buses

The traditional bus for both on-chip and off-chip communication is the time-division multiplex (TDM) bus.

Typical examples of the traditional bus are the ISA and PCI buses used in personal computers [2]. The older ISA bus became the de facto standard of the industry after its introduction in the early 1980s and has been phased out from production in desktop PCs only in the last few years, but it is still going strong in the embedded PC market showing the incredible historical success of the ISA solution.

Besides the slow clock rate and narrow width used in the ISA bus, the major drawback is the fact that the bus can only have one master. All these draw-

backs were attacked by some of the biggest PC hardware manufacturers when they jointly developed the PCI bus.

The PCI bus work according to the TDM principle where a master has to ask an arbiting unit for access to ("ownership of") the bus. Then the bus is occupied for a number of clock cycles while the transfer is taking place. During this period no other can access the bus than the current sender. Multiple masters are possible in the PCI setup and they have to compete with each other to gain control of the bus. This is done through an arbiter unit that all masters must be connected to. This arbiter will select which master will get control of the bus when multiple requests are present at the same time. Another way to increase the maximum throughput of the PCI bus is to divide it further into bus domains that communicate through bus bridges. The bus bridges connect two (or more) PCI buses so that a master on one bus can access a slave on another bus. This will give the advantage that both buses can perform different transfers at the same time as long as the bridging function is not necessary for any transfer.

1.2.1 ARM AMBA bus

The on-chip counterpart for the PCI bus is the AMBA bus from ARM, Inc. [3], with their second version well established on the market and the third version coming. In the recent years the AMBA has become more or less of a de facto standard for on-chip communication tightly coupled to the success of the ARM processor series for all kinds of embedded system designs from simple laundry machine controllers to highly advanced mobile phone base-stations. The AMBA bus is divided into two bus layers. The fist is called the Advanced High-speed Bus[1] (AHB) and is used for demanding communication such as CPU to memory and CPU to co-processor communication. The second is known as the Advanced Peripheral Bus (APB) and is a lower speed bus that is mainly used for communication to the lower speed I/O peripherals, e.g. serial ports. The AHB/APB solution is very similar to the PCI solution for PCs. A typical system architecture using AMBA can be found in fig. 1.2. The given reason for dividing the bus into AHB and APB is that the simple peripherals, e.g. a UART, that do not need high speed communication should not have to implement the complex high speed bus interface required for the AHB. Another reason is that the AHB will have fewer interfaces connected to it and will thus be possible to run at a higher frequency.

The AHB is a multi master bus while the APB is a single master bus. The single master on the APB is the bus bridge and all transfers have to be initiated by this bridge. All transfers to and from the APB slaves must be initiated by the

[1] There is also a simplified version of the AHB that is called Advanced System Bus (ASB). This is used when some of the advanced features of the AHB, e.g. split transactions, are unnecessary.

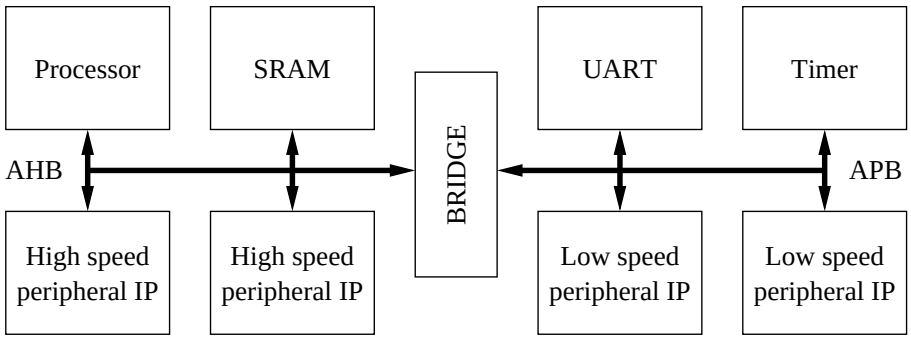


Figure 1.2: AMBA AHB/APB bridged bus example

bridge which in turn is initiated by a transfer on the AHB. In the case where an APB slave wishes to tell the processor that there is data to send this has to be done through other means such as interrupts.

One of the drawbacks with multi-master TDM buses in general is the arbiter needed to control access to the bus. The purpose of the arbiter is to decide which master has control of the bus at any given time. So when a master wishes to initiate a transfer on the bus the master signals a request to the arbiter. In a case where more than one master requests access to the bus at the same time the arbiter has to select one in accordance to some priority scheme. The selected master will then be signaled a bus access grant for the requested bus transfer.

The performance of the AMBA AHB is somewhat scalable since the widths of the data (and address) buses can be increased. The minimum recommended width is 32 bits giving a theoretical bandwidth of 4 bytes per cycle. The AHB supports pipelined bus transfers so if there are no transfers that require any wait states a figure quite close to this might be achieved. The APB on the other hand specifies a maximum width of the data bus of 32 bits and all transfers take two clock cycles. This together will yield a maximum throughput of 2 bytes per clock cycle.

The bridge between the AHB and the APB is very simple. The main job is to translate between the different bus signals and generate the appropriate control signals for the APB. If the APB is run at the same frequency as the AHB the penalty for accessing the APB is at least one wait state. If the APB runs at a lower frequency this has to be taken into account when calculating the penalty. Running the APB at half the frequency will yield a three wait states transfer penalty.

1.2.2 Problems with traditional buses

The traditional TDM bus is associated with some problems that do not affect the performance in a significant way when the number of connected ports are small

but that quickly become a problem when that number starts to rise. The first problem is the arbiter that has to connect separately to each of the connected ports that has master capabilities, i.e. the possibility to initiate a transfer. This arbiter will quickly become a bottleneck since the design has to be adaptable to the number of master ports. The second problem is the shared media that is used for transfers. There is a hard limit on the amount of data sent over the shared media in any time period and when the number of transfers increase the bandwidth available to each of them will decrease.

The only solution seen so far to tackle these problems is to divide the bus into several smaller bus domains that each have their own TDM bus and communicate with each other using bus bridges. While the use of bridges will allow for a higher compound bandwidth in the system it will also introduce a further disadvantage when communicating across domain borders. In order to guarantee the transfer the master will initiate a transfer that will be accepted by the bridge who will then wait for bus ownership on the destination bus before the transfer can take place. This situation leads to a high overhead for inter-domain transfers that will affect the performance on both the initiator and destination buses. The bus bridge solution is seldom feasible for more than three bus domains.

There are also some very good properties of the TDM bus where the first and foremost is the simplicity. The knowledge of TDM buses are widespread and since most bus implementations are similar to each other leading to very short learning time for new users. This simplicity also allow for a high observability and simple verification of the bus. Other good properties include that a TDM bus has a fixed cycle cost and very low latency. There is very little or no need for buffers at the bus interface and there are no buffers at all in the bus.

1.3 References

[1] C. Svensson, “Electrical interconnects revitalized,” in *unpublished manuscript, http://www.ek.isy.liu.se/~christer/CSPapers.htm*, 2001.

[2] PCI Special Interest Group, “PCI 2.3 specification,” *http://www.pcisig.com/specifications*, 2002.

[3] ARM, Inc., “AMBA 2.0 specification,” *http://www.arm.com*.

Chapter 2

On-chip networks

2.1 On-chip networks

With a move from traditional TDM busses to on-chip networks (OCNs) the communication infrastructure of the chip can allow significant - up to hundredfold or more - increase in the available bandwidth. This is possible since an OCN uses point-to-point links between the switches in the network. The primary reason for the increase in bandwidth is the multiple resources that are available for transfers in the OCN compared to the TDM bus. This multitude of resources allows for several concurrent transmissions to take place at the same time thus giving significantly higher bandwidth.

In general an on-chip network is a like any other network. The biggest differences are that the on-chip network reside in a much more controlled environment and that the demands on bandwidth and latency are higher. A generalized on-chip network, see fig. 2.1, is a means to connect different IP blocks together. The simplest “network” is just an instance of the TDM bus but the network could just as well be a completely connected crossbar switch that can connect any IP block to every other IP block.

2.2 On-chip vs. general purpose networks

It is very easy to compare the on-chip network to a general purpose network for computer communication such as the Internet. There are some important differences that must be taken into account when comparing the two classes of networks. The most important is that the on-chip network is a well known and static network whereas the general purpose network can change over time as users plug in their computers, routers go down, etc. The on-chip network in general also must have much higher performance when considering real-time transmissions.

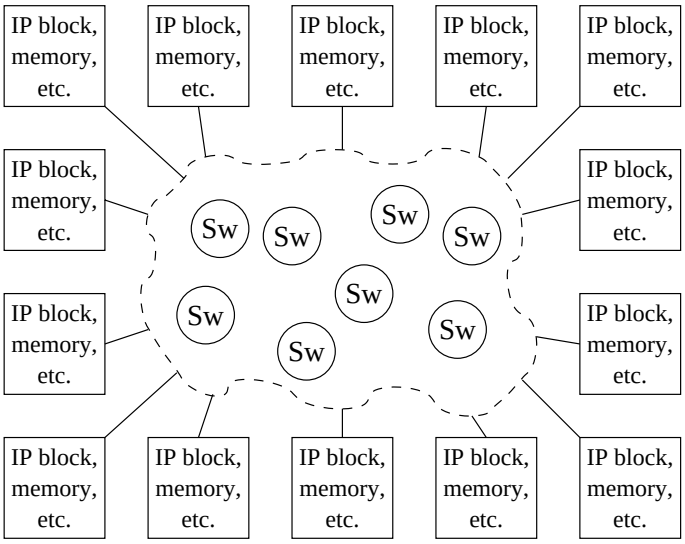


Figure 2.1: A generalized on-chip network

2.2.1 The OSI model

It is common to use the seven layer OSI reference model, see fig. 2.2 to describe the network protocols and divide them into layers, from the physical to the application layer. This division into several layers is suitable for both the understanding of networking concepts and for describing the protocol requirements in general purpose networks. The OSI model is also used by some research groups for on-chip networks to divide the design into protocol layers [1]. But in the case of on-chip networks it is more doubtful if the division into the seven layers is really useful. The main reasons is that the OCN is fixed and virtually fault-free under certain physical design rules. This means that once a network has been laid down in silicon it will not change on the lower layers, i.e. transport layer down to physical layer. The upper layers will - as in a general purpose network - be dependent on the application and endpoint hardware/processing and can not be decided by the network implementation. The seven layers can still be used to explain the implementation but should not decide the block partition or implementation of the physical system.

2.3 Network topologies

There are a multitude of topologies that can be used for an OCN. The concept of an OCN is very similar to that in a network for parallel computing so that is an

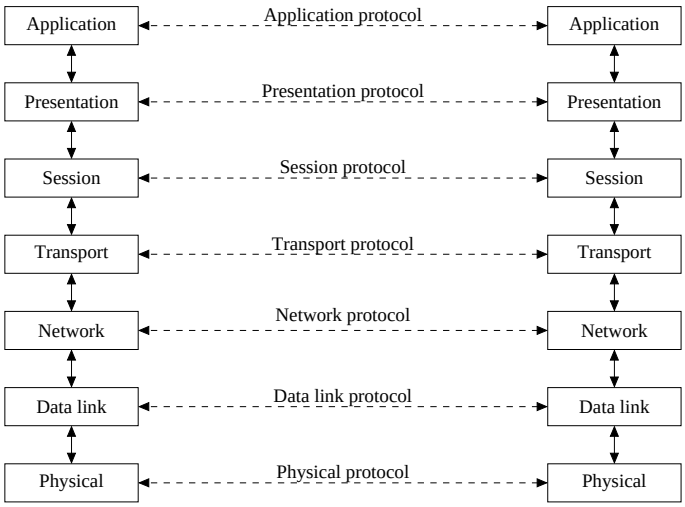


Figure 2.2: The OSI 7-layer model

obvious area for inspiration. Most textbooks in the area have a comprehensive description of the different topologies and their pros and cons. The books by Hwang and Xu [2] and Culler et al. [3] have been used as references for this work.

There are several basic structures, e.g. trees, meshes, tori, and cubes. The tree structures most commonly seen are *k*-ary trees and fat trees.

The *k*-ary tree is very simple since all nodes (except the root node) have k down connections and one up connection, see fig 2.3(a). This simplifies wiring but the big drawback is the bottleneck near the root where all traffic crossing that boundary need to go through a single node, thus giving a bisection bandwidth[1] of only one link capacity.

The fat tree, see fig 2.3(b), avoids the drawback of the *k*-ary tree by creating multiple trees so that the capacity at every level is the same as at the leaves. This gives the maximum bisection bandwidth that can be achieved in a tree since all levels have the same capacity. The latency is also low compared to the *k*-ary tree. The main drawbacks with the fat tree are the complexity of wiring and the problem with fan-in and fan-out due to the many ports. A strongly related topology is the butterfly. The butterfly uses very much the same connection style as the fat tree but have a lower fan-in and fan-out. Each switched is connected upwards to two switches and downwards to two switches. This solves some of the problems compared to a fat tree but increases the problem with wiring complexity and area

[1]Bisection bandwidth is the bandwidth across the boundary that divides the network in two equal halves.

consumption since the height of the butterfly tree will be higher.

The array-style networks can have very different properties dependent on dimension, size, and degree of the nodes[2]. Starting from one dimension networks we get the linear array or ring, see fig. 2.3(c), dependent on whether the ends are connected together or not. Compared to the TDM bus the linear array has the added possibility to run several transmissions on smaller sections of the array at the same time but the performance will be similar to that of a TDM bus if many transmissions are long distance (i.e. from end to end). The bisection bandwidth is on link for a linear array and two links for a ring.

If the dimension is increased by one we get a 2-d mesh , see fig. 2.3(e), or a 2-d torus, see fig. 2.3(f). The mesh and torus have the added advantage over the linear structures that there are multiple links leading in the same direction and thus there are more than one way to reach from one node to another in the network. This allows for a increase by the square root of the number of nodes (assuming a square network), in the bisection bandwidth compared to the linear array. The same increase in bisection bandwidth can be found in the torus compared to the ring. The topology of the 2-d mesh and torus networks can be exploited further if the traffic patterns are fairly well known so that IP blocks that communicate frequently are located close to each other. This optimization will give an even higher performance for the 2-d network after synthesis.

The array- and mesh-style networks are just special cases of the so called *k*-ary *n*-cube . The *k* is the number of nodes in each dimension and *n* is the dimension. A binary 3-cube is the most basic form of a *k*-ary *n*-cube, see fig. 2.3(d). The cubes continue the increase in bisection bandwidth and may have up to the bandwidth of k^n links dependent on the values of *k* and *n*.

2.4 Packet and circuit switching

The basic switching style in a network can be divided into two areas. The first (and oldest!) is the circuit switching technique that are used very much in old-fashion telephone networks. The circuit switching will require that a path is open from the source to the destination before any transmission can take place. The good thing with circuit switching is that when a circuit has been established the source have perfect knowledge of the available bandwidth and very good knowledge of the transmission latency. The drawbacks are the fact that the circuit has to be established before any transmission can take place and that the resources occupied by the circuit is unavailable to other transmissions. Further drawbacks are that

[2]The node (or switch) degree denotes the number of ports on the node, e.g. a 4th degree node has got four ports.

the circuit switched network need some central resource controller and that the flexibility is lower than for packet switching.

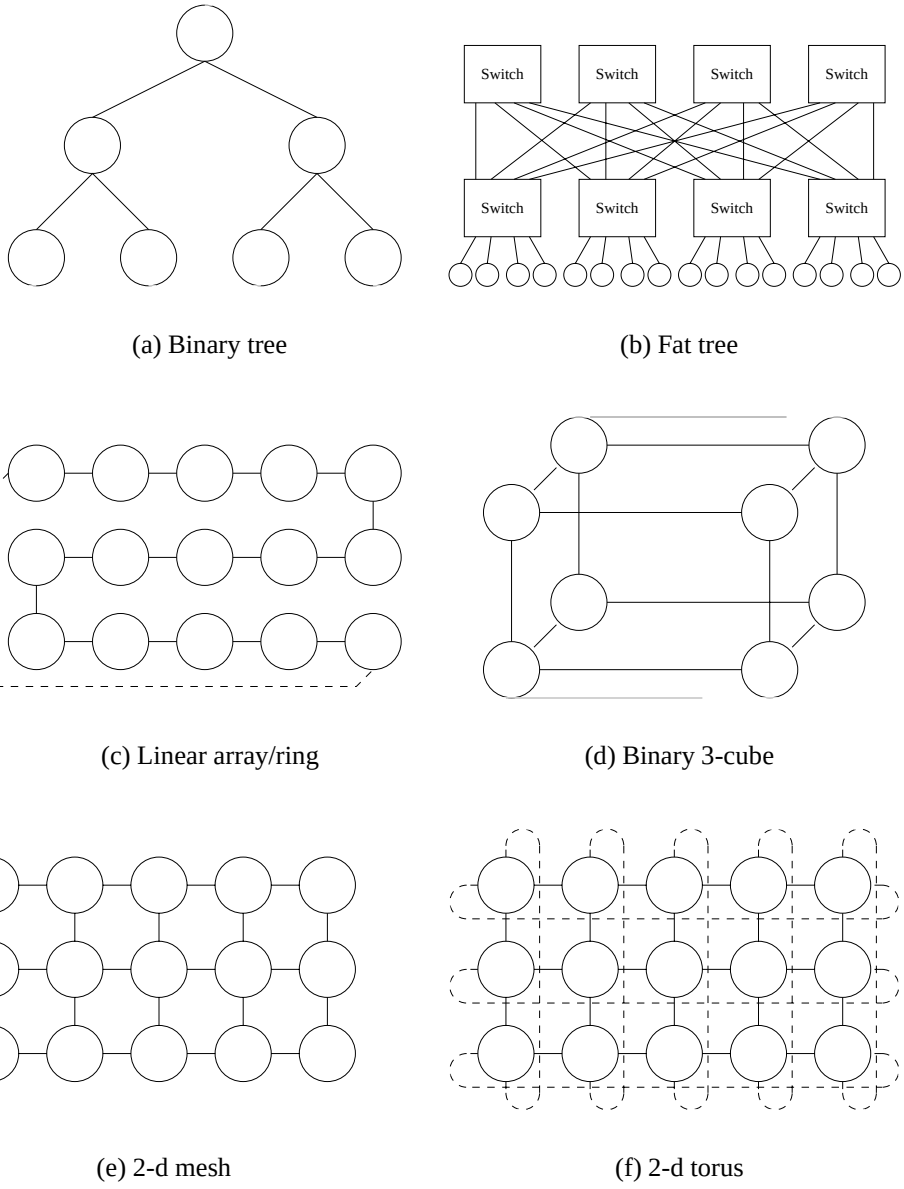
The second technique is the packet switching. In packet switching the data that is to be sent will be assembled into a packet by the source. The source will then send the packet to the network that forwards the packet to the destination. The biggest advantage for packet switching compared to circuit switching is that less resources are used exclusively for the same transmission and there is thus a possibility to time-share the resources more aggressively. Another good thing is that a packet switched network do not need a central resource controller. The drawbacks are that each switch will be more complex, that buffers are needed to avoid deadlocks, and that the data latency is generally longer.

2.5 References

[1] Iikka Saastamoinen, David Sigüenza-Tortosa, and Jari Nurmi, "Interconnect IP node for future system-on-chip designs," in *IEEE int'l workshop on Electronic design, Test, and Applications*, 2002.

[2] K. Hwang and Z. Zu, *Scalable parallel computing*. McGraw-Hill, 1998.

[3] D. E. Culler, J. Pal Singh, and A. Gupta, *Parallel computer architecture, A hardware/software approach*. Morgan Kaufmann Publishers Inc., 1999.



(a) Binary tree

(b) Fat tree

(c) Linear array/ring

(d) Binary 3-cube

(e) 2-d mesh

(f) 2-d torus

Figure 2.3: Network topologies

Part II

SoCBUS

Chapter 3

SoCBUS overview

3.1 SoCBUS introduction

This chapter introduces the SoCBUS solution developed by me and my colleagues. The results introduced in this chapter are the lions share of my research contribution so far. The following chapter deals with a more thorough investigation of the routing principles and algorithms used. Then the methodologies and tools used are described in the next chapter followed by a short OCN survey. The next three chapters contain three conference papers, two that have been presented and one that is currently under review, that show parts of the research in more detail.

The main interest of our switched interconnect research project, known as SoCBUS, is to achieve a well-behaved switched network style interconnect for on-chip communication in hard real time systems. There are several secondary goals where the most challenging is to try to use the on-chip network as a means to cut integration verification time when using several IP blocks on a single chip.

The SoCBUS project has resulted in a network using a two-dimensional mesh topology with switches in every mesh crossing. The mesh topology was chosen after a thorough investigation into many different possible topologies because of the simplicity in use and implementation, both on register transfer level (RTL) as well as physical level. In principle there is nothing that restricts us to the mesh topology and very small adjustments would be enough to use a two-dimensional torus topology. The only real difference would be that the basis for routing decisions changes and thus other routing tables would be necessary.

From a high level perspective the network using the standard two-dimensional mesh topology will consist of a number of tiles, see fig. 3.1. Each tile is made up of one switch connected to one IP block through a wrapper. The four ports of the tile are connected to the neighboring tiles.

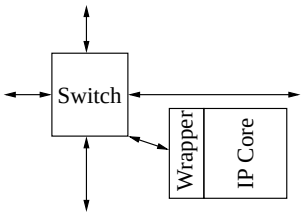


Figure 3.1: Network connected processing tile

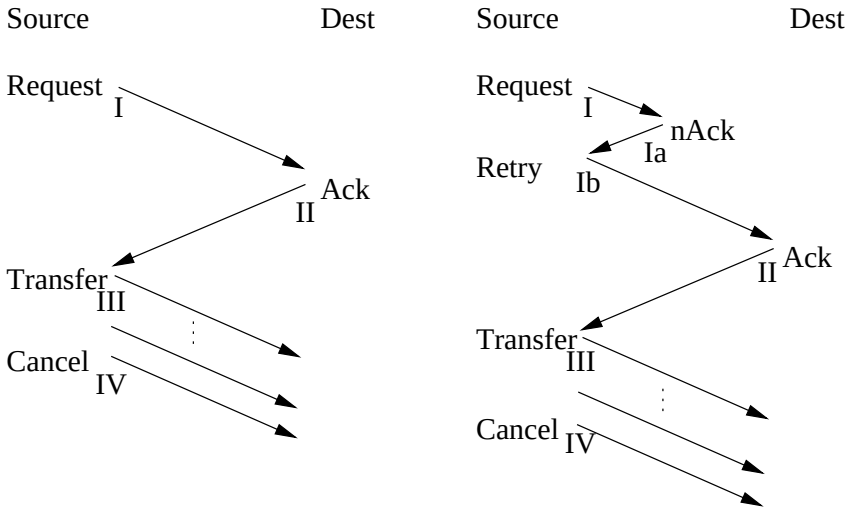
3.2 Packet connected circuit

The network uses a novel style circuit switching where a request packet traverses the network finding its way to the destination. While doing this the packet trail is locked and used as a circuit connection for the payload transfer. If a route can not be established, i.e. the packet can not find a free exit from a switch leading closer to the destination, the attempt is canceled and the request has to be retried. We call this novel approach for packet connected circuit or PCC for short. There are several advantages with PCC. Two clear advantages are that there is no need for buffers in the switches and that the PCC scheme is inherently free from deadlocks since there is no situation where a routing request will wait for another routing request to finish. Also the payload transfer latency can be kept at the minimum of one bus clock cycle per switch as soon as the route is established.

The drawback is the fact that a PCC routing failure will result in a completely new routing attempt. This drawback would be significant for a general purpose computing platform interconnect where the designer has got very little control of the communication scheduling. Another drawback is that the resource will be locked as soon as a routing request has passed by and will not be unlocked until the routing request has failed or the transfer is finished.

In a hard real time system we generally have to have some communication scheduling in order to achieve the appropriate levels of bandwidth, latency, etc. With a perfectly scheduled communication pattern we can get almost to the theoretical bandwidth limit with roughly 90 % bandwidth efficiency for a 10x10 network where the mean transferred data size is 100-200 words per transfer. When some unscheduled traffic is present the bandwidth usage of the network must be decreased to allow for the uncertainties introduced by that traffic.

The PCC scheme uses a basic four-phase route setup, see fig. 3.2(a). (I) The first phase starts with the source sending a request packet to its local switch. The switch then decides on a route that will lead closer to the destination, locks the corresponding output, and passes on the request. This is then repeated for every switch that the route passes through until the request has reached the destination.



(a) First-try successful (b) Second-try successful

Figure 3.2: Two successful circuit setups

(II) The second phase is then started with the destination returning a positive acknowledgment to the last switch that in turn passes the acknowledgment on to the previous switch and so on. (III) When the acknowledgment has reached the source the third phase is started. During this phase the source will send the payload data sequentially through the switches to the destination at high speed. (IV) When the data stream ends the fourth phase is performed consisting of a cancel request that passes through the network and while doing that unlocks the resources.

If any switch along the path is unable to fulfill the request it will return a negative acknowledgment. The negative acknowledgment will traverse the chain of switches to the source. As a switch sees the negative acknowledgment it will unlock the output that has been locked for this particular route. If this situation occurs the source will have to retry the routing request until a positive acknowledgment arrives, see fig. 3.2(b).

More information, some simulations, and further analysis of the PCC scheme can be found in chapter 4 and in paper 3, see chapter 9.

3.3 Switch implementation

The switch node for SoCBUS is a fairly regular structure, see fig. 3.3, with a simple state machine for each input that takes care of state tracking in the transmissions. The input finite state machines (FSMs) communicate with each other through the arbiting and locking block that handles request collision arbiting and output port locks. The arbiter and lock block controls the crossbar that connects the inputs to the outputs. The simplicity of the switch is because most of the switching functionality is handled at the inputs and none of it is handled at the outputs. Another contribution to the simplicity is the fact that all ports are equal. In principle it is possible to change the number of ports on the switch to an arbitrary number and connect the switches according to a arbitrary topology. Of course there is a practical limit on the number of ports since the complexity of both the crossbar and the arbiter will be too high to achieve high performance.

When a request is received at one of the port the corresponding input FSM reacts accordingly by decoding the destination address into possible outputs from the switch. An internal request to lock one of the possible outputs is then sent to the arbiting and locking unit that selects a suitable output from these and locks this. When the lock is set the crossbar connection is made from input to output and the request can be passed on to the next switch along the route.

The input FSM will then wait for an acknowledgment from the receiving end. If the acknowledgment received is negative this means that the network was unable to fulfill the routing request and thus the resources are freed. If the acknowledgment is positive the input FSM will keep the route open until a route cancel request is received from the source.

3.3.1 Routing

The routing principle described in chapter 2 is implemented in the switch as a static knowledge table . This table holds the information for the switching decisions. Each entry in the table shows the possible outputs that will take the route closer to the destination. Each entry thus has five bits of information corresponding to the five outputs of a switch. The states of the five bits simply show which outputs will lead closer to the destination. For a network of 16x16 switches this static table will be 1280 bits in total.

The static knowledge in the table is combined with the dynamic state of output locks to select the appropriate output for a routing request. The lock status show the currently unused outputs that can be claimed for the current routing request. If multiple outputs are available the switch will decide which to take according to a

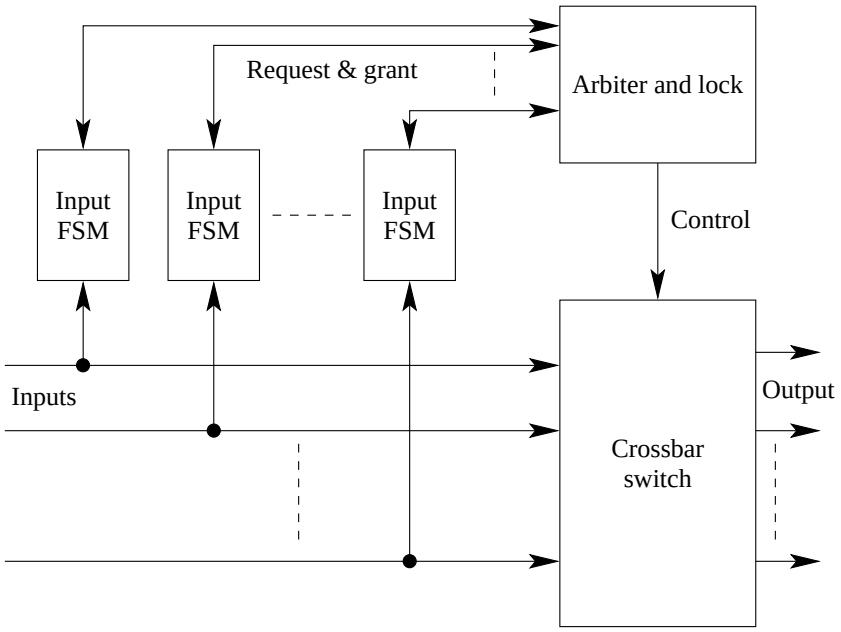


Figure 3.3: Generalized switch implementation

dynamic priority[1]. This dynamic priority typically follows a round robin scheme so that the average load will be roughly equal between the output ports which would equalize the load in the network compared to a static priority approach.

When two (or more) routing requests want to use a single output at the same time the switch must arbitrate between them. This is done in very much the same way as the output selection described in the previous section with a round robin scheme so to lower the likelihood of starvation[2].

3.4 Wrappers

The interface between the network and the IP blocks is intended to be implemented through wrappers that handle the format conversion, necessary buffering, asynchronous clock domain bridging, and network signaling. The wrappers can either connect directly to an IP block or connect to a local bus with several IP blocks as suggested by Wielage and Goossens [1]. Thus different wrappers can have very differing formats on the IP port and must be able to cope with a huge

[1]It would be very simple to implement a static priority if that is desirable in some applications.

[2]Starvation has not been addressed as a problem so far in the project. Further investigations into this will be conducted in the future research.

range of possible IP port clock rates. Most effort in the research so far has been on the network design and implementation and the wrappers are still mainly a part of the future research in the project.

3.5 Physical implementation

Since deep submicron chips will have increased delay in the wires compared to the logic the path towards the globally asynchronous but locally synchronous (GALS) design methodology seems like the only way to go. This adds yet another problem for the on-chip network to solve in the bridging between different local clock domains in the IP blocks. The relatively long delays for wires also make the on-chip network in itself impossible to implement in a fully synchronous way.

In the light of these fact the physical implementation of the SoCBUS network is proposed to use a mesochronous[3] clocking scheme for the signaling internal to the network. This will allow for a fairly straight-forward implementation where there is no need for FIFOs and other circuitry needed in an asynchronous bridging within the network. It is enough to have simple retiming circuits on the edges of each switch to synchronize the incoming signals with the local clock phase. The asynchronous bridging is then moved to the wrappers where the probability of the IP block and the network running at the same clock frequency is fairly low.

The mesochronous clocking scheme has some very interesting consequences such that the wire lengths between switches can be very different between different switch pairs. Since the signal retiming is done per link it does not matter if the delay of the wires is several clock cycles.

3.6 References

[1] Paul Wielage and Kees Goossens, “Networks on silicon: Blessing or nightmare?,” in *Euromicro Symposium On Digital System Design (DSD 2002)*, (Dortmund, Germany), Sept. 2002. Keynote speech.

[3] Mesochronous means same clock frequency but unknown phase.

Chapter 4

Routing analysis

4.1 Introduction

With the basic PCC scheme introduced in chapter 3 we still need some algorithm to select which output to take from a switch in order to get to the destination. A thorough investigation into the published material on routing algorithms was conducted and the conclusion from this was to keep the algorithm as simple as possible. The reason for this was that there were no shown advantages in the papers that were worth the extra complexity. We selected a very simple minimum path length routing where every output link taken will lead closer to the destination. Since the network is a rectangular mesh it is trivial to realize that there is at most two and at least one output that will lead closer to the destination. The first situation is when the destination is not in the same row or column as the current switch. If the destination is somewhere up to the right both the up and right outputs will take the request an equal distance closer to the destination. The other situation is when the destination is in the same row, the same column, or connected directly to the current switch.

The contents of this chapter is a rather mathematical analysis of the routing algorithm used for the SoCBUS project simulations so far. The main focus here is the routing for a two-dimensional mesh but some results for a two-dimensional torus are also presented. All results that are not for the 2-d mesh will be marked explicitly.

4.2 Basic principles

4.2.1 Addressing

The first thing is to define the addressing scheme of the network. The addressing is part of the basis that the following calculations stem from. The switches are identified by their coordinates in the mesh so that the upper left corner is considered to be $(0, 0)$. This addressing is used solely for the discussions in this chapter.

Definition 1 *Each node in the network has an address tuple (x, y) that is the distance (along the x- and y-axis, respectively) from the upper left corner of the network as seen in a picture. The upper left corner has the address $(0, 0)$.*

It should be noted that we do not use the physical grid coordinates above to identify the IP blocks. Instead an arbitrary mapping of addresses to coordinates is used for addressing the IP blocks and these are the actual addresses used when performing routing.

Definition 2 *Each wrapper (or IP block) has an associated address A. The addresses are found using an arbitrary (one on one) mapping between these A_i and the associated switch nodes (i.e. where the wrappers are connected) addresses (x_i, y_i).*

4.2.2 Path length

A central issue in the analysis of resource usage for the network is the path (route) lengths for every possible route. The path length is defined as the Manhattan distance between the switches or IP blocks (or wrapper). This is calculated by summing the number of *switch node inputs* consumed by the route. In the distance between wrappers we must also take into account the switch input consumed by the wrapper to switch interface.

Lemma 1 *The path length (using Manhattan distance) between two* nodes (x_i, y_i) *and (x_j, y_j) is the sum of absolute differences between the two node addresses.*

$$l_n = |x_i - x_j| + |y_i - y_j| \tag{4.1}$$

Lemma 2 *The path length (using Manhattan distance) between two* wrappers A_i *and A_j is the sum of absolute differences between the two associated node addresses plus one.*

$$l_w = |x_i - x_j| + |y_i - y_j| + 1 \tag{4.2}$$

With the path lengths as a background it is now fairly straight-forward to calculate the mean distance between all IP block pairs in the network. This mean distance is a direct measure of the mean resource consumption for random traffic transfers in the network.

Theorem 1 *The mean path length between wrappers in a network of size X, Y is the sum of the mean distances along each dimension divided by the total number of nodes.*

$$l_{wm} = \frac{\sum_{i=1}^{X}\sum_{j=1}^{X}|i-j| + \sum_{i=1}^{Y}\sum_{j=1}^{Y}|i-j|}{XY} + 1 \tag{4.3}$$

Lemma 3 *For a square network of size X, X the mean distance equation in theorem 1 is simplified.*

$$l_{wmeq} = 2\frac{\sum_{i=1}^{X}\sum_{j=1}^{X}|i-j|}{X^2} + 1 \tag{4.4}$$

It is very interesting to compare the square network case for a 2-d mesh and for a 2-d torus. The torus will have a shorter mean distance because of the wrap-around connections at the edges.

Lemma 4 *For a square torus network of size X, X the mean distance is based on the minimum distances in both directions.*

$$l_{wmeq} = 2\frac{\sum_{i=1}^{X}\sum_{j=1}^{X} min\left(|i-j|, X-|i-j|\right)}{X^2} + 1 \tag{4.5}$$

4.3 Static analysis

All examples and calculations in this section will be based on a 10 x 10 network but everything is applicable to any size.

A 10 x 10 network has been statically analyzed to find the routing density for the network. The routing density show how many of the total amount of the possible routes that go through every node. This density is a measure on the distribution of the routing across the network and should preferably be as flat as possible.

Assuming minimum path length routing and only one turn per route[1] the distribution of fig. 4.1 is found. The actual figures used in the graph is shown in table 4.1[2]. The graph is (as expected) perfectly symmetrical around the center.

[1] May be slightly stupid, see section 4.4.1.

[2] Since the data is symmetrical along the two centers only the upper left quadrant of the table is shown.

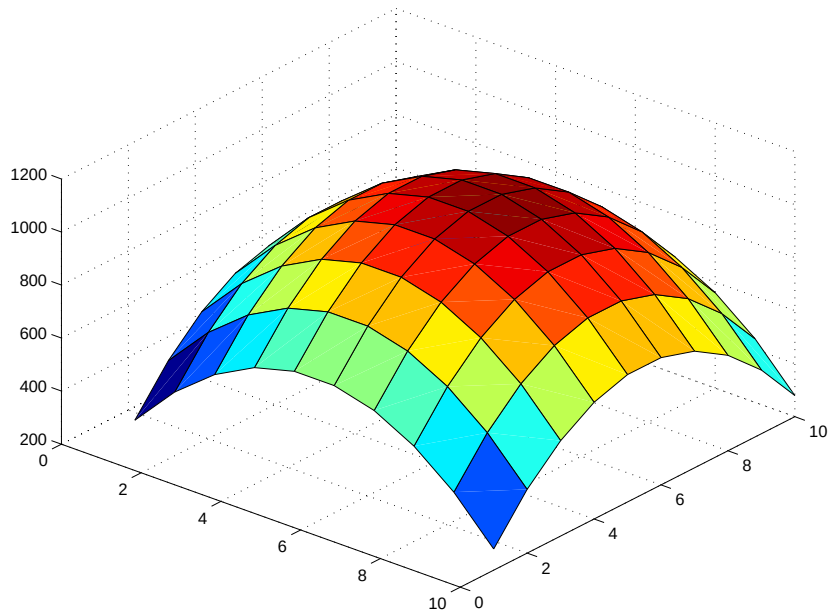


Figure 4.1: Number of routes consuming an input port of switch (10x10)

The number of possible routes in the network is

$$r_{poss} = X^2 Y^2 \tag{4.6}$$

since there are XY nodes and for each node there is XY possible destinations (including itself).

Calculating the mean path length for a square network of size (10x10) is done using eqn. 4.4.

$$l_{wmeq}\,(10) = 2\frac{330}{100} + 1 = 7.6 \tag{4.7}$$

Table 4.1: Number of routes that use an input port of the switch for a 10x10 network.

	0	1	2	3	4
0	280	440	560	640	680
1	440	600	720	800	840
2	560	720	840	920	960
3	640	800	920	1000	1040
4	680	840	960	1040	1080

Lemma 5 *Considering eqn. 4.6 and theorem 1 the total average inport usage in a network with totally random traffic is*

$$r_{inport} = r_{poss} l_{wm} \tag{4.8}$$

Lemma 5 shows that the total number of used input ports for all possible routes is 76000 which is consistent with the simulation results shown in fig. 4.1.

As is apparent from fig. 4.1 the most congested area is the center with roughly 11% of all routing[3] going through (at least) one of the four center nodes while only less than 3% going through the corner nodes. The conclusion that can be drawn from this result is that communication between IP blocks that are far from each other in the network preferably should be avoided. This is also apparent from the network traffic simulations conducted in paper 3, see section 9.7.

Since the torus will have shorter mean distance it is interesting to compare the inport usage for the torus with the mesh. The inport usage for the torus is calculated using eqns. 4.5 and 4.8.

$$l_{wmeq\_Torus}(10) = 6.0 \tag{4.9}$$

$$r_{inport\_Torus}(10) = 10000 \cdot 6.0 = 60000 \tag{4.10}$$

This gives the situation where the average route usage per switch for a complete 2-d torus will be a flat surface at the level of 600. This is almost half of the peak value in the mesh. This flat route usage is a desirable advantage and the use of a torus should be investigated further.

4.4 Possible improvements

4.4.1 Avoiding blocking

Some simple simulations with one-step misrouting allowed in case of congestion have been conducted. One-step misrouting means that the routing may turn along a non-profitable (i.e. not leading closer to the destination) link to avoid the congestion situation if the destination is in the same row or column as the current switch. The drawback is that it might only move one step along the new column or row before returning to the current one. The risk with this is that the route will look like the trail of a snake, see fig. 4.2. This one-step misrouting was shown in the simulations to have an adverse impact on performance.

Since the basic problem here is that when the routing request has entered the row (or column) of the destination there is no way to reroute around a blocking

[3]All routing here means the total switch inport usage for the default routes between every sender/receiver pair.

route and still maintaining the minimality. Thus a simple improvement is to postpone entering the row (or column) of the destination for as long as possible just to make sure there is more than one minimal path to the destination. This feature have not been simulated yet.

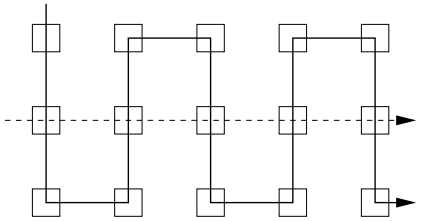


Figure 4.2: Possible bad situation with misrouting

4.4.2 Flattening the density

One important improvement is to try to flatten the routing density by "smearing" the peak over a larger area. This is a non-trivial problem because the routing decision might need to be dependent not only on the destination but also on the source.

The interesting thing here is that the total volume under the graph in figure 4.1 has to be constant (according to lemma 5) so the only possible improvement is to make the graph flatter.

4.4.3 Traffic optimization

The most important optimization for network efficiency is to allocate the network ports to IP blocks in such a way that the mean traffic path lengths for the finished system is lowered. This can be achieved by doing the IP block-to-network switch connection allocation based on the traffic knowledge retrieved in system simulations. Since a system-on-chip implementation of a communication system will have quite well-known and well understood traffic patterns this knowledge can be used in the step of synthesizing a SoCBUS instance for the specific chip to get this important advantage of traffic localization. One of the important goals of the SoCBUS project is to enable such knowledge based network synthesis.

4.5 Conclusion

The calculations and simulations presented in this chapter have shown that the current routing algorithm is minimal concerning switch-input occupation when blocking does not occur. The most important thing left for investigation is to find an optimal (or near optimal) route distribution across the network. It would also be interesting to see the impact on performance of the non-minimality because of blocking in more detail. Some more investigation into the value of using a torus is interesting as well.

Chapter 5

Research methodology and achievements

5.1 Introduction

This chapter is a brief summary of the research flow in the project so far and an outlook towards the future. Some methods and tools that have been used in the research are also described. The final part of this chapter summarizes my achievements so far that are related to the SoCBUS project.

5.2 Prestudy

An early prestudy of the demands, design space, and design goals for the SoCBUS project were conducted in my thesis work for the M.Sc. degree [1]. This prestudy focused on the basics of on-chip networks and the demands on these. The prestudy was continued after the publication of the thesis with further investigations into the specifics of on-chip networks.

5.3 Behavioral simulator

As a means of evaluation we developed a behavioral simulator model for the network components and a simulator for this model was developed[2, 3]. The behavioral simulator is cycle true[1] and is implemented using object oriented programming in C++. The main objectives when implementing the simulator were to make a tool available for design space exploration and to be able to find bottlenecks and prospective problems.

[1]The simulator also supports bit true simulations if the models are designed accordingly.

The simulator have been used to simulate the different design choices taken so far. This include the topology choice, routing principles, routing tables, and switch implementation details.

5.4 Hardware

Currently a register transfer level model of the switch is being developed according to the descriptions in this thesis. This model will be used for further simulations as well as synthesis experiments for gate count, area, and power estimations.

A couple of simple wrappers for synchronous serial ports are also being developed for the SoCBUS project partly to be used for testing the SoCBUS network in simulations and hardware implementations. The next step in hardware development is the creation of a more complex high speed wrapper for a standardized bus interface, e.g. the AMBA AHB interface.

5.5 Synthesis

The ultimate goal of the research project is to create a bunch of libraries that can be instantiated in a design and synthesized together with IP blocks and memories to create a complete chip. This synthesis should allow for traffic pattern knowledge based optimizations to attain a certain transfer locality advantage with lower average network resource usage compared to the non-optimized case.

5.6 Achievements

My main achievements so far in this project is the specification and simulation of the network together with the development of the PCC routing scheme and a behavioral simulator.

Inspiration for a project like this can be found everywhere since the design choices for an on-chip network are vast and the literature that are related to topology, routing, etc. for parallel computers and other networks are plentiful. From this origin the prestudy was successfully completed with much knowledge gained in the special purpose networks field.

This knowledge was used to specify the mesh network topology and the basic circuit switching requirements to attain a high performance, low latency on-chip network. The components used in this network have been specified together with an interconnection link design based on low-voltage mesochronous signaling. The low voltage signaling style with limitation analysis and physical design

is researched in another group at our department and in close cooperation with us [4].

The routing was thoroughly investigated and a novel packet based setup circuit switching scheme was developed by me and my advisor, Prof. Dake Liu. This novel scheme, packet connected circuit (PCC), is based on a routing request packet traversal of the network to connect the circuit from sender to receiver.

Based on the SoCBUS specification and the PCC routing scheme I developed a network simulator and behavioral models that later have been used for simulating both random traffic patterns as well as special traffic patterns from case studies in communications.

5.7 References

[1] D. Wiklund, *Prestudy of a bus for system-on-chip*. LiTH-IFM-Ex-848, Dept of physics and measurement technology, Linköpings universitet, Sweden, 1999.

[2] D. Wiklund, "Implementation of a behavioral simulator for on-chip switched networks," in *Proceedings of the Swedish System-on-Chip Conference (SSoCC)*, March 2002.

[3] Daniel Wiklund and Dake Liu, "Design of a system-on-chip switched network and its design support," in *Proc of the Int'l conference on communications, circuits and systems (ICCCAS)*, 2002.

[4] Peter Caputa and Christer Svensson, "Low power, low latency global interconnect," in *Proc of the 15th int'l ASIC/SoC conference*, 2002.

Chapter 6

On-chip network survey

6.1 Introduction

The on-chip network research has only been around for a couple of years with some of the oldest publications from 2000. In these years many research projects have been started and most are still at the proposal stage with very little published quantitative data as evidence for the appropriateness of each approach. This chapter gives a brief overview of some of the more well-known projects from both universities and companies.

6.2 Survey

6.2.1 Université Pierre et Marie Curie

One of the first suggestions for an on-chip network is from Université Pierre et Marie Curie in Paris, France [1]. A packet based fat tree connected network was suggested for the implementation of an on-chip network. The performance results are good but not very suitable for systems with real-time constrains on the communication because of the very varying latencies. In their experiments they found that a small fraction (less than 0.1 %) of the transmissions would have a latency of 150 cycles or more while the average would be somewhere around 20 cycles. The OCN also has got the clear disadvantage of both big area (mostly due to buffers) and complex wiring.

6.2.2 Stanford

Prof. Dally et al. at Stanford proposed a two-dimensional torus network, see fig. 2.3(f), for on-chip purposes [2]. Their proposal uses datagram (send-and-

forget packets) based transport with virtual channels and local buffers for avoiding deadlocks. The total memory cost for one switch is then roughly 40 kilobits. They have estimated the silicon overhead to be around 6.6 % including buffers, wires, and switches. The routing can either be pre-scheduled to meet hard real time constrains or dynamic for general-purpose traffic.

6.2.3 Tampere Institute of Technology

Saastamoinen et al. at the Tampere Institute of Technology in Finland have proposed an OCN that uses SCI (scalable coherent interface) style asynchronous links between switches [3]. The topology constrains or the silicon overhead of their proposal is not clear from their publications. The interfaces between network and IP blocks are handled by the use of wrappers.

6.2.4 Sonics, Inc

Sonics, Inc. is a system-on-chip integration company that have a product suite for simplified integration of multiple IP blocks in one chip. The center of their methodology is the SiliconBackplane packet based connection network [4]. The SiliconBackplane is a linear array (see fig. 2.3(c)) with a token passing scheme to allow the use of the linear interconnect. The interconnect also uses a kind of wrappers, called MicroNetwork Agents, to connect between the IP blocks and the network.

The success so far for Sonics is probably mostly because of the complete integration flow for IP block reuse and not because of an extremely high performance of the interconnect. The SiliconBackplane interconnect is just a central part to make the flow possible.

6.2.5 Philips Research

Philips research has a OCN project called the ÆTHEREAL NOS [5]. This project is a hybrid packet switched and time division circuit switched network inspired by the asynchronous transfer mode (ATM) general purpose network with guarantees on quality-of-service. The network both supplies guaranteed throughput and latency through the time division circuit switch as well as best-effort packet switching without packet loss.

6.2.6 Other industry projects

There are several projects in the industry but there have not been any details undisclosed at the time of writing. The facts that the on-chip network field is very new,

that there is no established way of doing OCNs, and that there are no products released yet make the companies very restrictive in undisclosing any detail.

6.3 References

[1] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proc of the design and test in Europe (DATE) conference*, 2000.

[2] William J. Dally and Brian Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc of the design automation conference (DAC)*, 2001.

[3] Iikka Saastamoinen, David Sigüenza-Tortosa, and Jari Nurmi, "Interconnect IP node for future system-on-chip designs," in *IEEE int'l workshop on Electronic design, Test, and Applications*, 2002.

[4] D. Wingard, "Fully-pipelined fixed-latency communications system with a real time dynamic bandwidth allocation." US Patent 5948089, 1999.

[5] K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage, "Networks on silicon: Combining best-effort and guaranteed services," in *Proceedings of the design automation and test conference*, Mar. 2002.

Part III

Papers

Chapter 7

Paper 1

Switched interconnect for system-on-a-chip designs

Daniel Wiklund and Dake Liu

Department of Physics and Measurement Technology

Linköping university,

SE-581 83 Linköping, Sweden

{danwi,dake}@isy.liu.se

Proceedings of the IP2000 conference

Edinburgh, Scotland, October 23-24, 2000

Switched interconnect for system-on-a-chip designs

Daniel Wiklund and Dake Liu
Department of Physics and Measurement Technology
Linköping university,
SE-581 83 Linköping, Sweden

Abstract

With the increased use of IP cores in chip designs, an increasing amount of time is spent on design and verification of glue logic. To solve this problem together with the bottleneck problem of arbitration based buses, a novel approach in system-on-a-chip interconnect has been investigated. The approach is based on a switched interconnect structure, with small crossbar switches connected in a mesh for intercore communications with low latency in system-on-chip solutions. The interfaces between the interconnect network and the cores are handled by configurable wrappers that adapt the port parameters from core to network format. The core functionality of the interconnect network can be fully verified with a fairly low work effort even when configurable, so the main problem for cutting verification time is the quite complex wrappers. The concept is to make the wrappers highly configurable yet needing short verification time in an application by making a fairly complete verification of the wrappers for all configurations. How this can be achieved is under investigation. The approach described in this paper is mainly aimed for use in communication equipment where high bandwidth and low latency is essential.

7.1 Introduction

With the recent advances in communication systems, the classical way of building electronics is becoming infeasible due to the inherent problems with speed, power consumption, physical size, and so forth. These problems have forced the industry to use higher and higher levels of integration to keep up with competition and customer demands. Nowadays, many products are implemented using only a few or even a single chip. The trend towards higher integration is associated with new problems such as design complexity, higher economic risk, heavy verification work, as well as physical problems like power and clock distribution. Since many chip designs use intellectual property cores, the main problem beside verification of the IP cores is the complexity of glue logic and interconnects. Something has to be done to cut the design and verification complexity of the glue logic and in-

terconnects in order to allow even larger designs to be accomplished successfully on time. The trend towards more and more battery powered designs where power efficiency is a main objective implies that the clock rates in the system should not be unnecessarily high. This together with the problem of clock distribution will lead to designs that may use several clock domains. Thus a good way to bridge between clock domains is needed. The current bus solutions for system-on-a-chip designs are almost all based on a shared synchronous media and uses arbitration to allow several initiators to use the bus. This is the classical way of implementing interconnects that has been used for many years. This bus type is rapidly becoming a major bottleneck for system performance in system-on-a-chip solutions as an increasing amount of data is transferred between different parts of a chip. Especially, it is impossible to use a classical bus for central services in communication, such as a radio base station or a gateway for voice over IP. In order to increase the performance of interconnections, buses based on arbitration must be exchanged for novel architectures that allow higher compound bandwidth as well as greater simultaneous connectivity. The concept of networks in system-on-chip design has a lot of similarities with networks for parallel computers, although many requirements differ between the two areas. The single largest difference is that while computers emphasize on compound bandwidth, latencies are often the most critical in communications. There is also a big difference in that the jobs are better known at the design stage in applications in the communications field than for computers. The routing algorithms for parallel computer networks also require high tolerance against dynamic faults such as broken cables and nodes. This is a fairly small problem in on-chip networks since the system is less susceptible to failures.

7.2 Bus system overview

The bus system considered consists of four major parts. These are the sender and receiver wrappers, the network and the switching nodes as found in the overview of the system in figure 7.1. The purpose of each of these parts will be explained later in this paper. Wherever it is needed, a core may have several wrappers connected, both senders and receivers, in order to allow the communication that is necessary for the core.

The dataflow in the system is that when a core needs to send data over the bus system, a route is first set up to the receiver. This route is then locked during the entire time that a transfer is in progress. After that, the actual payload will be delivered through the interconnection network and finally the route is cancelled and the resources that were in use will be available again. Since the system should have as low latency as possible, a circuit switched technique is used. Packet

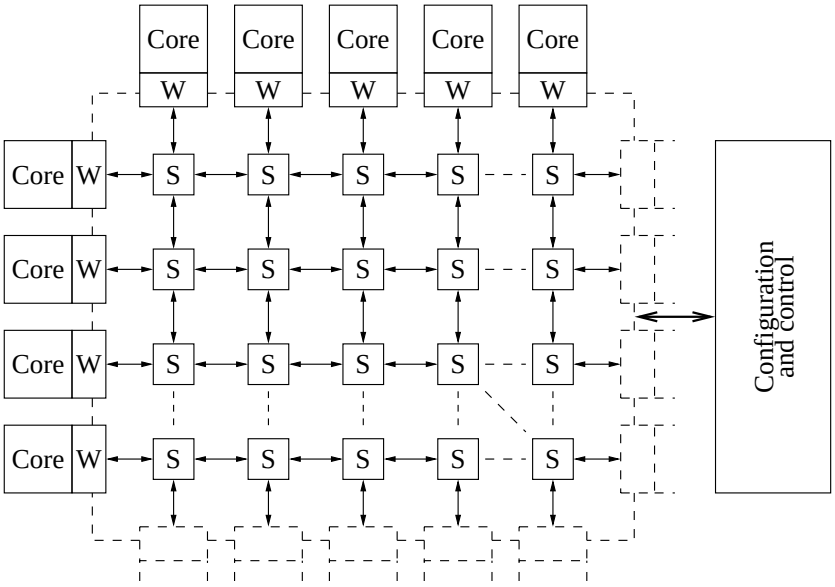


Figure 7.1: Interconnect system overview

switched systems tend to have too much overhead in packet header processing and buffering. A circuit switched system can also guarantee the latency and available bandwidth for one transmission since no other transmissions can use the route concurrently.

The tasks of the wrappers are to adapt the port and transfer parameters between the cores and the interconnect network. The main parameters that need to be handled are data width, data rate, clocking and synchronization. The configuration and control block performs general control tasks such as supervising the network, doing initial setup and run-time configuration. This block also monitors the network for faults.

7.3 Network architecture

There are several network topologies that are worth considering for the bus system. The most commonly seen in parallel computers are fat trees, meshes and tori [1]. A fat tree is the best choice from a connectivity point of view, but the fat trees are complex when seen from a wiring perspective and will thus consume more silicon area.

The network architecture that most closely resembles the layout of a chip is a mesh. The dimension of the mesh is limited to a fairly low number in order to fit on a chip without problems. Higher dimensions lead to better performance

but also lead to more complex switching nodes. The simplest solution here is to use a two-dimensional mesh with common 4-input / 4-output crossbar switches and links that allow duplex DMA connections. The links may be either true full duplex, but may just as well use time division duplex or some other technique to allow for full duplex at the DMA level.

For a small system (up to some 8 or 16 nodes), it is possible to use a fat tree as network topology. For larger systems, a 2-D mesh is more suitable since it has simple wiring while using a single switch type. This circumvents the problem with extensive and complicated wiring to achieve good connectivity since this problem does not occur in a low-dimensional mesh.

7.4 Switching node

The actual routing in the network is done in the switching nodes. The nodes consist of a crossbar switch, a routing controller and some buffers (if needed). The crossbar does the actual switching while the routing controller decides what way to route incoming data through the crossbar. The crossbar is able to route incoming traffic to any output except the port that leads back to the previous node. More than one route can go through the crossbar as long as each route uses input and output ports that are not occupied by another route.

Buffers in the switches should be avoided, since buffering inevitably will increase latencies in the network. Buffering is almost always needed in a packet switched network because the switches must buffer the incoming packet whenever there is congestion or if the routing decision takes time. In order to keep a low latency and high bandwidth in the network, it is better to use a circuit switching strategy that sets up an entire route and then sends the data as quickly as possible along the route. The route is then released immediately when the transfer has ended so that the resources become free as soon as possible.

7.5 Routing algorithms

Routing algorithms for interconnect networks come in many flavors. One major feature that distinguishes different algorithms is whether the algorithm uses deterministic or adaptive (dynamic) routing. Deterministic algorithms always use the same route between two ports, while a dynamic algorithm can use different routing each time. Other important issues in routing are deadlock and indefinite postponement. Since the algorithm should be robust considering these issues, the algorithm must be both fair and free from deadlocks.

Simple deterministic routing algorithms, such as the west-first algorithm are

easy to show if they are fair and deadlock free or not. The west-first algorithm (also known as X-Y routing) [1] has both of these qualities, but it is too simple to be usable in the system outlined in this paper. The major flaws are that it has no built-in fault tolerance and that it is very inflexible.

Dynamic algorithms generally have better performance under heavy load than the simpler deterministic ones. The trade-off here is that a dynamic algorithm calls for a more complicated and possibly slower routing controller in the switching nodes. Another drawback is that a dynamic algorithm is much more complicated than a deterministic algorithm when it comes to deadlock issues. The concept of proving deadlock freedom is much harder with a dynamic algorithm than for a deterministic one.

A suitable routing algorithm is a variation of the pipelined circuit-switching algorithm found in the work by Gaughan and Yalamanchili [2]. If a deterministic algorithm is to be used, the routing algorithm developed by Badr and Podar [3] is suitable. Since both of these algorithms are developed for parallel computers, some modifications will probably be needed to adapt them to the bus system. Both algorithms have some built-in fault tolerance that makes the system more robust against failures in the network.

7.6 Wrappers for network-core connections

In order to allow IP cores from different vendors and with different interfaces to coexist on the bus, the wrappers need to be very flexible. This flexibility is achieved by making the wrappers highly configurable. This in turn leads to big problems at the verification stage where the wrappers must be proven correct for every possible configuration. If this is not done, the wrappers and the bus system must be verified in every application it is used in.

The wrappers should take care of such tasks as synchronization, clocking and adoption of data parameters. All of these can be of very various types. Synchronization can for example be achieved by using one or more extra signals (e.g. strobe or mask), self clocked data or through an asynchronous handshaking protocol. Clocking can also be done in many ways, such as positive or negative slope, dual edge and self clocked data. All this lead to a very complicated wrapper that needs to be simplified at the design-time configuration in order to save die area and power. Since all the different configurations will lead to different simplifications at the ASIC design phase, it is very hard to prove that the system is correct in all cases. Verification of these kind of configurable systems is an area where much more research needs to be done. The current standardization effort by the VSI Alliance is interesting since it may simplify the wrappers considerably if most IP providers will follow the standards.

7.7 Configuration

Configuration of the bus system must be done at several levels. First, at the ASIC design stage where the ports of each wrapper are configured in accordance with the ports of the corresponding core. This may include such things as port widths, clocking and synchronization. The second level of configuration is done during the software design stage, where the system is directed towards a specific application with e.g. fixed routes. Finally, the system is configured while the system is running. This includes things like on-the-fly routing.

Many parts of the configuration could be done at more than one level. What to configure at each level is determined by trade-off between flexibility and complexity. The later the configuration is done the more flexible system you get, but at the expense of possibly larger silicon area and higher power consumption.

7.8 Related work

The concept of switched networks for on-chip solutions has surfaced quite recently. Because of this there is not very much work available on these kinds of systems. One of the few is the work by Guerrier and Greiner [4]. They have reached a high bandwidth, strong connectivity, and reasonable silicon area. The drawback is that they use a packet switched fat tree topology that has the inherent drawbacks of a quite high average latency and a probability to get prohibitively high latencies for some packets. The maximum usage is in practice also limited to approximately 40%-50%. This means that their solution is not very suitable for chips in communication systems.

7.9 Conclusions

For an efficient interconnect network to be designed, a lot of issues need to be considered. The most promising design consists of a 2-D mesh with crossbar switches that uses adaptive routing. This leads to simple physical routing of wires on the chip combined with good performance and fault tolerance. For smaller system, a fat tree can be considered instead of the 2-D mesh, but with considerable differences in the routing compared to the mesh.

In order to allow different IP cores to connect to the bus system, wrappers need to be designed that are highly configurable in respect of data parameters, clocking, synchronization and so forth. The verification of these wrappers is a major obstacle since they have to be fully verified in order to allow easy integration with the other parts of a chip.

In most respects, the configurability must be high for the bus system to be usable in as many applications as possible. While this work primarily emphasizes communications, computers can also be considered a candidate for this system.

7.10 References

[1] D. E. Culler, J. Pal Singh, and A. Gupta, *Parallel computer architecture, A hardware/software approach*. Morgan Kaufmann Publishers Inc., 1999.

[2] P. T. Gaughan and S. Yalamanchili, "Pipelined circuit switching: A fault-tolerant variant of wormhole routing," in *Proc. of the Fourth IEEE Symposium on Parallel and Distributed Processing*, 1992.

[3] Hussein G. Badr and Sunil Podar, "An optimal shortest-path routing policy for network computers with regular mesh-connected topologies," *IEEE transactions on computers*, 1989.

[4] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proc of the design and test in Europe (DATE) conference*, 2000.

Chapter 8

Paper 2

Design of a system-on-chip switched network and its design support

Daniel Wiklund and Dake Liu

Department of Electrical Engineering

Linköping university, SE-581 83 Linköping, Sweden

{danwi,dake}@ifm.liu.se

Proceedings of the International conference on communications, circuits, and systems (ICCCAS)

Chengdu, China, June 29-July 1, 2002

Design of a system-on-chip switched network and its design support

Daniel Wiklund and Dake Liu
Department of Electrical Engineering
Linköping university, SE-581 83 Linköping, Sweden
{danwi,dake}@isy.liu.se

Abstract

As the degree of integration increases, the on-chip communication is becoming a bottleneck. A solution to this problem is to use an on-chip switched interconnect network. Such a system-on-chip network was proposed in 2000 by the same authors. In this paper we present the system-on-chip network in detail together with the design flow support. The choice of topology for the network as well as some ways to use the network to overcome the future physical implementation issues of wire delay and to gain performance is also discussed. To aid the design choices of the network, a behavioral simulator has been created. The importance of the behavioral simulator is clearly shown from the design flow and the design and implementation of this simulator is discussed in detail.

8.1 Introduction

With the trend towards highly integrated system-on-chip designs with many on-chip processing resources like processors, DSPs, and ASICs, the on-chip communication is soon becoming a bottleneck. Classically this would be done with a traditional time-division multiplexed (TDM) bus as shown in fig 8.1a. This bus suffers from the clear bottleneck of the shared media used for the transmission. There is also a need for a bus-global arbiter in a multi-master setup with such a bus. This is nevertheless the topology used today by many companies in the system-on-chip arena, e.g. Sonics Inc. [1]. A better way of communication is to resemble the way networks for parallel computers are generally designed, see fig 8.1b, that do not suffer from those problems.

Networks for parallel computing are generally implemented with rather long latency and limited performance due to the constrains set on the system in the context of limited link bit width, requirements on fault tolerance, etc. To move the network from parallel computers into a communications system-on-chip requires different hardware/software protocols to attain much higher performance and much lower latency.

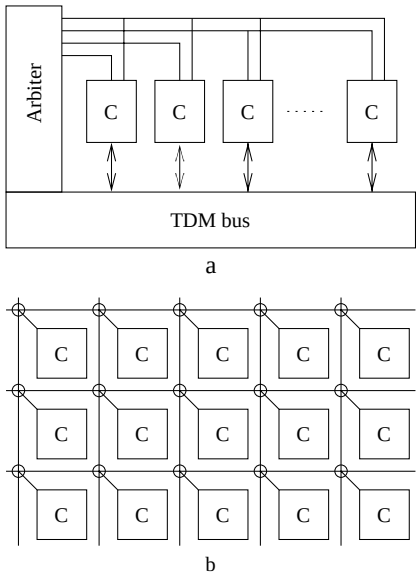


Figure 8.1: Comparison of traditional TDM bus and a switched interconnect bus replacement

We have previously proposed the use of a two-dimensional switched network for system-on-chip communication [2]. Later work by Dally and Towles [3] and Sgroi et al. [4] confirm the general trend towards on-chip networks.

The expected result of this research project is a general platform for system-on-chip integration using the two-dimensional network as the interconnect structure. The platform should be easy to use and ease the burden of verification during the system-on-chip integration phase.

8.2 A new system-on-chip integration flow

The intended design flow for system-on-chip integration using the switched on-chip network is depicted in fig 8.2. The user input is the specification and requirements needed to configure the on-chip network with core interface wrappers for the specific application. This is the hardware interface specification for the cores (e.g. processing and storage elements) that are to be connected in the system-on-chip design as well as the requirements on bandwidth, latency, etc. put on the network hardware to support the necessary transfers. The software protocol interface specification and requirements are also needed as user input in the design.

The configuration and network control code is then generated and synthesized based on the on-chip network library together with the user input data to create

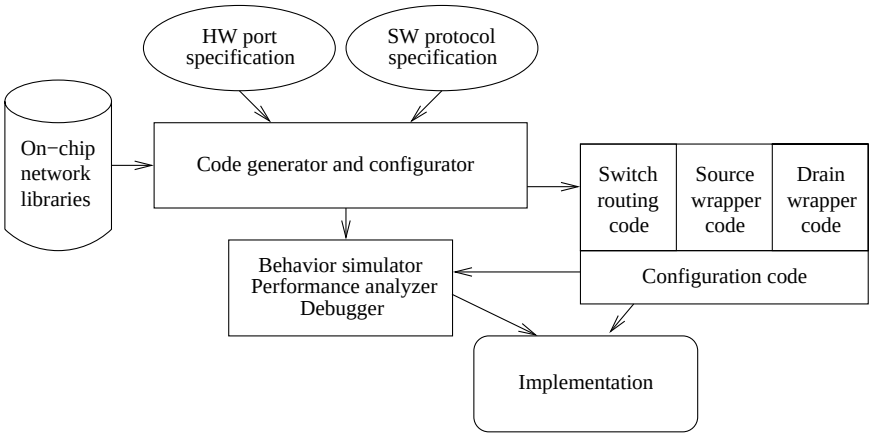


Figure 8.2: Design flow for a on-chip network design

Table 8.1: Theoretical performance of different network topologies assuming N cores along the edges.

	Ring	2-d Mesh	Binary tree	Benes Network	Fat Tree
No of nodes	N	$(N/4)^2$	$2N-1$	$NlogN$	$2^{N/4-1}$
Links	Bidir	Bidir	Bidir	Unidir	Bidir
Bisection BW	2	$\sqrt{N}$	1	N	N
Wiring compl	Low	Low	Low	High	High
Max wire length	Low	Low	Medium	High	High
Routing compl	Low	Medium	Low	Medium	High

both behavioral and structural descriptions of the interconnect network.

This code can then be used in the behavioral simulator to ensure correctness and to generate a number of different benchmarking informations to make sure that the specification is fulfilled. After behavioral verification the structural description is used as a part in the integration of the system-on-chip design to achieve the final implementation.

8.3 Theoretical performance of network topologies

In order to determine the most appropriate topology for the system a thorough investigation of the advantages and drawbacks of a number of common topologies were done during the prestudy phase.

A short summary of the theoretical performance for some of these network

topologies can be found in table 8.1. The ring topology is very easy to implement but is subject to the same fundamental limitations as a normal bus since the only available resource for transmission at a node will be occupied whenever a transmission wants to pass that node. On the other hand, the more powerful topologies like fat trees are very complex when it comes to wiring.

During the prestudy phase the different topologies were thoroughly studied from a theoretical perspective resulting in the conclusion that the two-dimensional mesh is most suitable for on-chip networks. The main advantages of the two-dimensional mesh is the good performance to resource ratio, the ease of routing, and that the topology is very easily mapped onto a chip.

8.4 Network components

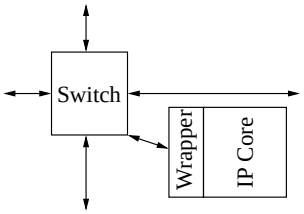


Figure 8.3: Network connected processing tile

The network can be seen as a two-dimensional matrix of tiles shown in fig 8.3. These tiles are made up of three components except for the IP core. These are the switch nodes, the source wrappers, and the drain wrappers. In addition to these there is a simple control circuit with global responsibility.

The purpose of the source and drain wrappers is to isolate the port formats of the custom cores used in the system from the network format to make it possible to ease the task of verification of the whole system-on-chip design. This is done by using configurable wrappers at the ports between the network and the cores. The wrappers can be thoroughly verified for the different possible configurations together with the rest of the network system so that only the interfaces have to be verified at the system integration phase, thus alleviating the verification task.

The switch nodes are responsible for routing the data transport streams between source and drain ports. The choice of implementation of the switch nodes mainly affect the connectivity and routing latency in the network.

8.5 Configuration and control layering

In order to make a usable system, the network components have to be configured according to the chip design requirements. Additional configuration and control tasks have to be carried out during runtime. These tasks of configuration and control can be divided into different groups that reflect the level of physical or virtual links that are affected by the configuration and layering is shown in table 8.2. The layering can also be seen as a protocol hierarchy in the implemented system where lower levels are implemented as hardware fixed protocols and upper levels are implemented as software controlled protocols.

Table 8.2: Configuration and control layering

Data transport	Data package
	Data link protocol
Link setup	Physical configuration
	Addressing and routing
	Contention control
	Other (FEC, buffering, etc.)
Circuit	Network control
	Switching nodes
	Wrappers

The data transport layer includes the data link protocol for control of source-drain transfers and handles transmission-specific configuration information such as packet sizes. The link setup layer is at the wrapper-switch or switch-switch level and controls contention control and physical configurations such as clocking and framing of data. The final layer is the circuit layer that handles the low-level configuration of the network parts.

8.6 Physical implementation

The physical implementation of network components can be realized in a number of ways. The realization should preferably help in overcoming some fundamental problems that are becoming more and more pronounced in modern processes. One obvious problem is that with the increasing size of chips and shrinking feature size that are available, the globally synchronous way of building electronics is rapidly becoming infeasible due to wire delays. One way of alleviating this problem is to design the chips using smaller cliques of synchronous logic (up to 250k gates

[5]) that communicate using asynchronous or mesochronous (i.e. same clock but unknown phase) links using a masked clock from the data source.

The fact that the complexity in the network components is fairly low and most functions can be heavily pipelined implies that the clock-rate of the network can be relatively high. Since the network is distributed by its nature the delays in the wires can be very long, up to several cycles, compared to the clock period of the network. This suggests that the network should be implemented in a mesochronous fashion using simple retiming circuits [6].

By making the core wrappers the bridge between the clock domains used in the cores and the network this "globally asynchronous" principle can be implemented with a minimum of verification effort compared to the principles used today.

Also by using a very high clock rate (compared to the cores) in the network it is possible to show a very low apparent latency despite a higher number of pipeline stages in the network.

8.7 Behavioral simulator

It is clear from the design flow that the importance of the behavioral simulator is very high. This simulator can not only be used as a tool in the customer implementation flow but also as a platform for early cost evaluation of performance and complexity of different implementations and schemes, e.g. for routing and switching.

The input to the behavioral simulator is the generated code for switches with the routing code and the code for source and drain wrappers. The code containing the configuration (network size etc.) is also input to the simulator. Also an user supplied description of the transmission traffic patterns are needed for the best simulation results.

The simulator runs the complete network responsible for both control setup and data transport for a period of time and checks the data to ensure functional correctness. The simulator also collects performance measurements such as momentarily and average available bandwidth, maximum and average latency, and contention measures.

The output contents of the behavioral simulator, see table 8.3, is divided into three main areas to evaluate the correctness, performance of the network, and cost to implement and verify the network. The configuration and verification cost is estimated from the amount of change in the network system needed for the current configuration. During development of the network components this can be used for evaluation of design choices that affect different costs and benchmarks to allow for trade-offs between different design issues such as increased cost of switches compared to increased connectivity and data throughput.

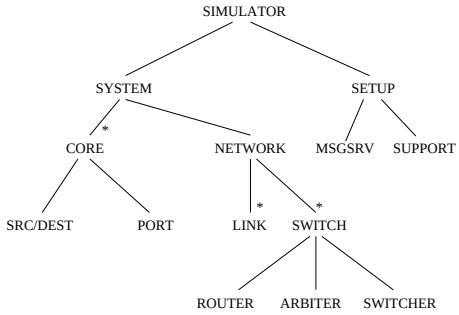


Figure 8.4: Object structure of the behavioral simulator network model

Table 8.3: Output generated by the behavioral simulator

Benchmark	Connectivity eval
	Data throughput eval
	Latency evaluation
Functionality	Correctness
Cost	Circuit selection cost
	Protocol selection cost
	Control subsystem cost
	Configuration cost
	Verification cost

8.8 Behavioral simulator implementation

The behavioral simulator is developed using object oriented programming in C++. The simulator is implemented using a message passing structure to resemble the structure of the real network system. The simulator is event-driven and both bit and cycle true in its execution.

The object structure of the simulator can be found in figure 8.4. The leaf nodes in the system branch corresponds to the actual network components except for the switch that is subdivided into its subtask components where the arbiter is for local arbiting between the ports of a switch. The setup branch is the message passing support for the simulator. This program structure is very suitable for the simulation of an essentially message-passing hardware like the on-chip network presented in this paper.

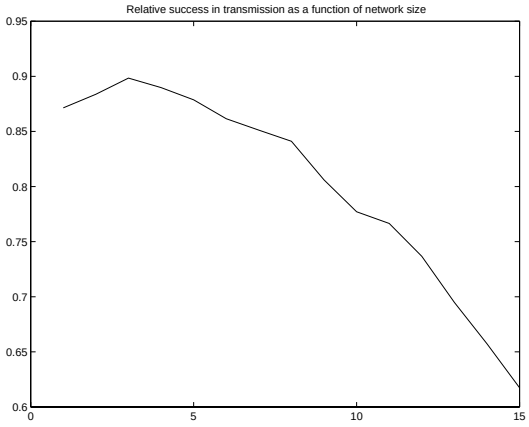


Figure 8.5: Example of simulator results

8.9 Behavioral simulator output

Taking one simulation result as an example of output from the simulator, we considered the case where cores are only connected to the switches along the edges, the switches use the deterministic dimension-order routing algorithm, and the “packets” use only a single cycle for transmission between switches. As the network in this example does not use any kind of retry mechanism all blocked packets will be dropped. A plot of the relative number of successful transmissions compared to all initiated transmissions with a mean usage (i.e. the probability that a core wishes to send) of 20% can be found in figure 8.5. The network size is the number of switches along each dimension. Thus the number of cores is equal to the size times four (i.e. because of four edges) and the number of switches is equal to the size squared. In this example we did not consider resending dropped packets and the possibility of deadlocks was eliminated through the algorithm choice.

Although this is a relatively simple implementation of the switch nodes and will not be relevant in a real system, it is interesting too see that it is very clear from the simulations that the relative number of successfully received transmissions decreases as the size of the network increases and thus the number of senders increases. This is basically due to the congestion that occurs near the center of the network when using dimension-order routing near the saturation limit.

8.10 Future work

The next step is to use the network-on-chip behavioral simulator to evaluate the appropriateness of different implementations of the parts that make up the network system.

Work is currently taking place to fund a demonstrator system for the Socware[1] program based on the switched network-on-chip that is described in this paper. The demonstrator will focus on routing heavy communications in real-time applications.

We will also evaluate different schemes of connecting the cores to the network as well as implement and evaluate different routing algorithms for the network.

8.11 Conclusions

The concept of an on-chip switched network as a platform for system-on-chip integration is discussed and an analysis of network topologies prompts for the use of a two-dimensional network. An appropriate design flow is introduced and the importance of a behavioral simulator in this flow is clearly shown.

The simulator design and implementation is discussed as an platform for evaluation of the design choices and cost trade-offs in the network parts, such as routing algorithms and switch connectivity constrains.

The physical implementation is discussed and it is suggested to use a higher internal clock rate in the network system to gain even more performance compared to traditional buses.

8.12 References

[1] D. Wingard, “Fully-pipelined fixed-latency communications system with a real time dynamic bandwidth allocation.” US Patent 5948089, 1999.

[2] Daniel Wiklund and Dake Liu, “Switched interconnect for system-on-a-chip designs,” in *Proc of the IP2000 Europe conference*, 2000.

[3] William J. Dally and Brian Towles, “Route packets, not wires: On-chip interconnection networks,” in *Proc of the design automation conference (DAC)*, 2001.

[4] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli, “Addressing the system-on-chip interconnect woes

[1] Socware is a national Swedish government research and education program in electronics.

through communication-based design," in *Proc of the design automation conference (DAC)*, 2001.

[5] C. Svensson, "Electrical interconnects revitalized," in *unpublished manuscript, http://www.ek.isy.liu.se/˜christer/CSPapers.htm*, 2001.

[6] Fenghao Mu and Christer Svensson, "Self-tested self-synchronization circuit for mesochronous clocking," *IEEE Transactions on Circuits and Systems*, vol. 48, no. 2, pp. 129–140, 2001.

Chapter 9

Paper 3

SoCBUS: Switched network on chip for hard real time embedded systems

Daniel Wiklund and Dake Liu

Department of Electrical Engineering

Linköping university

SE-581 83 Linköping, Sweden

{danwi,dake}@isy.liu.se

This manuscript has been submitted and is currently under review.

SoCBUS: Switched network on chip for hard real time embedded systems

Daniel Wiklund and Dake Liu
Department of Electrical Engineering
Linköping university
SE-581 83 Linköping, Sweden
{danwi,dake}@isy.liu.se

Abstract

With the current trend in integration of more complex systems on chip there is a need for better communication infrastructure on chip that will increase the available bandwidth and simplify the interface verification. We have previously proposed a circuit switched two-dimensional mesh network known as SoCBUS that increases performance and lowers the cost of verification. In this paper, the SoCBUS is explained together with the working principles of the transaction handling. We also introduce the concept of packet connected circuit, PCC, where a packet is switched through the network locking the circuit as it goes. PCC is deadlock free and does not impose any unnecessary restrictions on the system while being simple and efficient in implementation. SoCBUS uses this PCC scheme to set up routes through the network. We introduce a possible application, a telephone to voice-over-IP gateway, and use this to show that the SoCBUS have very good properties in bandwidth, latency, and complexity when used in a hard real time system with scheduling of the traffic. The simulations analysis of the SoCBUS in the application show that a certain SoCBUS setup can handle 48000 channels of voice data including buffer swapping in a single chip. We also show that the SoCBUS is not suitable for general purpose computing platforms that exhibit random traffic patterns but that the SoCBUS show acceptable performance when the traffic is mainly local.

9.1 Introduction

System-on-chip (SoC) designs pose many challenges on the designers. Design of a complex SoC with limited time and resources is a major hassle in the industry already today. With further technology progress and the ever increasing demands on integration there are a number of problems that quickly become big issues, e.g. design time and verification.

The major direction out of some of these problems is to make use of (both external or internal) intellectual property (IP) blocks, i.e. complete subsystems - like a microcontroller or a DSP - that can be integrated on chip together with other IP and custom logic to make up the complete system-on-chip product.

As the number of IPs on a chip increases there is an increasing demand of high performance communication between the IPs. The current solution when implementing SoCs with multiple processors, ASICs, memories, etc. is to use a time-division multiplex (TDM) bus, e.g. the AMBA bus from ARM, Inc [1]. The fundamental limits in this kind of bus is the arbiter used for a multi-master setup and the shared media.

Another problem when designing complex chips using IP blocks is the verification of the interfaces between the IPs, the communication structure, and the custom "glue" logic generally used to connect them. To alleviate the problems of verification and performance we have proposed a two dimensional mesh network, called the Linköping SoCBUS [2].

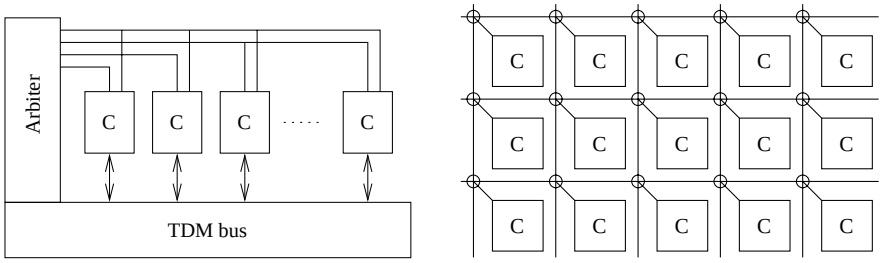
This paper is divided into three main parts. Sections 9.2 through 9.4 discusses general issues concerning SoC networks as a concept as well as the proposed SoCBUS with network architecture and transaction handling. Section 9.5 introduces an example of a possible hard real-time application, the PSTN/VoIP gateway. Sections 9.6 and 9.7 briefly introduces the simulator and discusses the stimuli and simulation results for our SoC network using random pattern traffic and the PSTN/VoIP gateway example. Finally the paper is closed with a description of some future work and the conclusions in sections 9.8 and 9.9.

9.2 The SoCBUS as a bus replacement on chip network

Since the shared TDM bus is becoming a bottleneck many researchers and companies have realized the need of a better way of handling the on-chip communications. With the widespread knowledge in general purpose computer networks at hand, researchers saw a great possibility of using networks not only for off-chip communication but also for on-chip communication. fig. 9.1(a) shows a traditional TDM bus setup and fig. 9.1(b) show the switched network counterpart. It is obvious even from a quick glance that the possible performance is significantly higher for the network based solution.

Several research groups around the world are currently involved in research on on-chip networks. A common feature among almost all of these research projects is that they use packet switching [3, 4] following the seven layer OSI model [5].

The seven layer OSI model was developed with general purpose networks in



(a) Traditional bus (b) Network bus

Figure 9.1: Comparison of traditional TDM bus and a switched interconnect bus replacement

Table 9.1: The seven OSI layers and their coverage in the SoCBUS system

7	Application layer	Not covered by SoCBUS
6	Presentation layer	Can be covered by wrappers
5	Session layer	Can be covered by wrappers
4	Transport layer	Covered by SoCBUS
3	Network layer	Covered by SoCBUS
2	Link layer	Covered by SoCBUS
1	Physical layer	Covered by SoCBUS

mind and is not necessarily applicable directly to all types of networks. One such network type where the layered approach of implementation is unsuitable is on-chip networks. The reason is that there is some fundamental differences between a general purpose network (e.g. Internet) and an on-chip network. The general purpose network is in principle unknown, i.e. nodes can be added and removed at any time, the topology is not fixed, and so on, while the on-chip network is well-known since it is fixed at the time of chip manufacturing.

With this additional knowledge of the network infrastructure it is possible to group several layers of the OSI model and therefore simplify the hardware and software while at the same time cut latency in the network. Our proposed solution has a broad coverage of the OSI-model, see table 9.1, with a minimum of complexity.

A network using circuit switching has low complexity switching nodes because their main function only is to connect an incoming link to an outgoing link. Deadlock avoidance is easily achieved since the circuit setup can either succeed

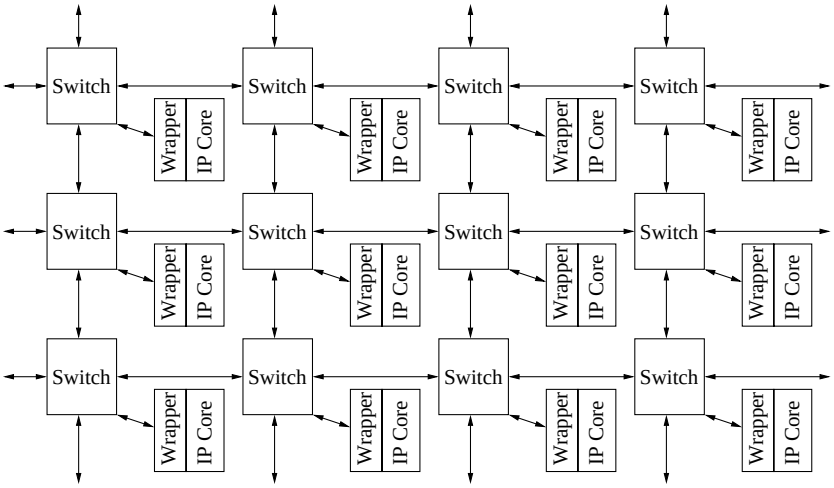


Figure 9.2: A 4x3 node switched network with wrappers and IP blocks

or fail but it can not stall somewhere in the process. Packet switching inarguably leads to more complex switching nodes since the node in order to avoid deadlocks either have to buffer every packet in its entirety before routing it to the next node, the node have to use several virtual channels [6], or the node has to restrict the possible paths [7].

Packet switching also suffers from latency problems where the packet delay through the network can be several hundred or even several thousands of cycles dependent on routing algorithms and switch implementations [8]. One example is where a fully buffering implementation of a packet switched network transfers 128 bytes of information between two diagonally opposite corners of a 8x8 mesh. The time spent in the network would then be more than $128 \cdot 15 = 1920$ cycles. Even for a wormhole routing network there is a possibility that the packets will be stalled for long times due to other traffic. This is because there is always a statistical distribution of packet delays in a packet switched network which also creates the possibility for packets to arrive out-of-order to the destination. Circuit switching has a clear advantage over packet switching since the data latency is only dependent on the distance and there is no dependency on other factors, e.g. other traffic in the network. All data is also guaranteed to arrive in the same order as sent. The only dependency on the traffic situation in a circuit switched network is when setting up a route.

9.3 SoCBUS architecture

A thorough investigation of different network topologies were conducted [9] and a two dimensional mesh was considered to be the most suitable for on-chip networks. This is also the common topology proposed by most other researchers [3, 4, 10]. The main reasons for selecting a the two dimensional mesh instead of other topologies such as hexagons, butterflies, or tree are that a two dimensional mesh have an acceptable wire cost, reasonably high bandwidth, and that it is easy to group IPs that communicate a lot so that they do not consume an unnecessary high amount of resources in the network.

The network, see fig. 9.2, uses five port switches that allocate four ports to connect to adjacent switches and one port to connect to the local IP block interface (port). The local IP port is connected through a wrapper to the local IP block. The wrappers handles IP and network port differences such as transaction handling, port width, endianness, etc. The wrappers also contain any necessary buffers and does the bridging between the IP block clock domain and the network clock domain.

The interfaces internal to SoCBUS all use the same physical format, see fig. 9.3. In total 11 wires are used in each direction where eight wires carry forward going data and routing request packets, one wire is used for forward control, and two wires are used for reverse control. The forward control handles framing of transmissions and clock information to allow for easy retiming of the transfers when using mesochronous clocking (i.e. same frequency but unknown phase) [9]. The reverse control carry the acknowledgments. The diagonal lines in the figure represents an identical interface to the local IP wrapper port.

The simplicity of the SoCBUS components allow for an implementation of switches and the network side of wrappers with very low logic depth, i.e. 6-8 gate delays. With this low logic depth it is possible to run the SoCBUS at 1.2 GHz in a 0.18 micron technology process. This is four times the maximum expected IP block clock frequency of 300 MHz in the same process. This difference in clock rates will further mask the latency since a four cycle latency in the network will look like an one cycle latency to a core.

Considering the high clock rate and the distributed nature of an on-chip network, the wire delays between the components become a serious problem if using traditional synchronous design methodology. In order to allow for wire delay and skew we propose the use of mesochronous (i.e. same frequency but unknown phase) clocking with signal retiming in the system. Using mesochronous clocking and retiming still requires the wires delays within a link to have reasonable skew but allows the design to use links that have differing delays without any problem. To further simplify the connection of network components we propose using optimized drivers and transmission-style wires [11, 12]. This gives many advantages,

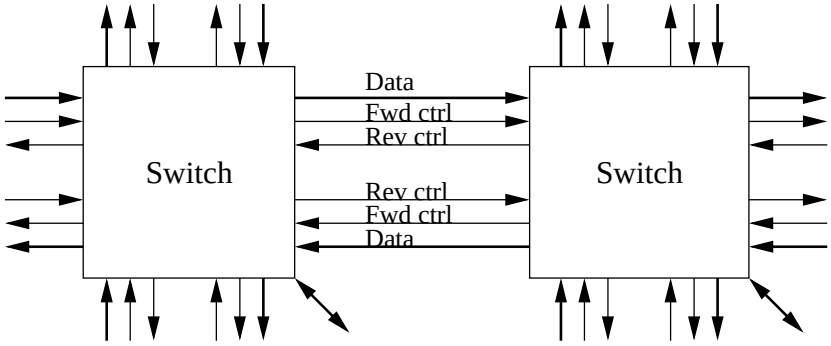


Figure 9.3: Interface between switching nodes and wrappers within the network

e.g. no repeaters are necessary, the system consumes a minimum of power, and there is no need of laying out the network in an ordered fashion on chip.

9.4 Network transaction handling

9.4.1 Route setup flow

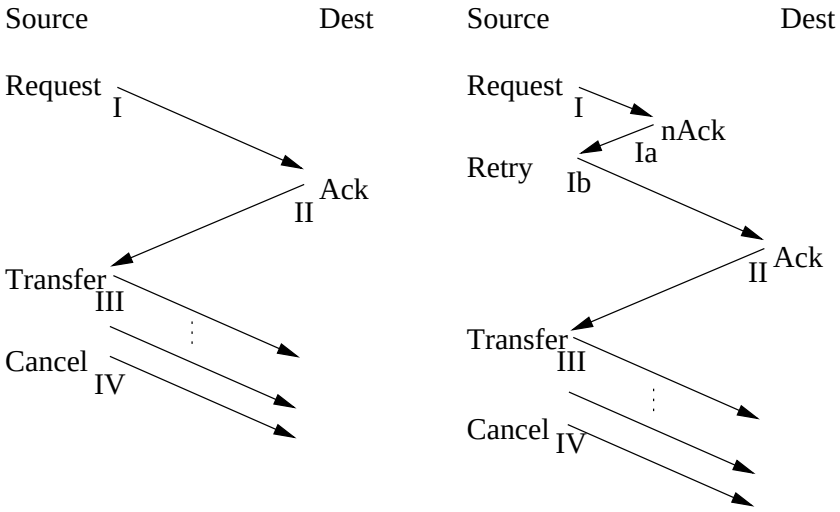
The network transactions consist of four to six phases dependent on whether the first try routing is successful or not. A successful transaction, see fig. 9.4(a), has four phases. (I) First a request is sent from the source to the network. As this request finds its way through the network the route is temporarily locked and can not be used by any other transactions. (II) The second phase starts when the request reaches its destination an acknowledge is sent back along the route and the locks are changed to permanent [1] locks. (III) When the acknowledge has returned to the source the third phase starts. This phase holds the actual transfer of data payload. (IV) Finally after the data has been transferred a cancel request is sent that releases all resources as it follows the route.

If a route is blocked in a node the routing request is canceled by (Ia) the blocking switch returning a negative acknowledge to the source, see fig. 9.4(b). (Ib) The source must then retry the route at a later stage which means that the additional two phases (nAck and retry) might need to iterate.

9.4.2 PCC: Packet connected circuit

We refer to the novel hybrid circuit switching with packet based setup introduced in the previous subsection as “packet connected circuit” or PCC for short. The

[1] Permanent refers to the current transaction time span only.



(a) No retry necessary (b) One retry used

Figure 9.4: Two successful circuit setups

PCC has very nice properties in several areas as follows:

- The PCC is deadlock free since no resources are locked while waiting (indefinitely) for other resources.
- The routing hardware in the switching nodes become very simple when no special cases, stalls, or virtual channels must be considered.
- Need for a minimum of buffering capable of holding just a request package is needed in the switching nodes.
- PCC gives the lowest possible latency of just one retiming latch pipeline at each switching node.
- There is no inherent limit on route selection algorithms in the PCC scheme.

9.4.3 Routing

After studying publications on routing algorithms and performing some simple experiments it was decided to go for a simple minimum path length routing. Each switch has the knowledge of the general direction to each destination, i.e. north,

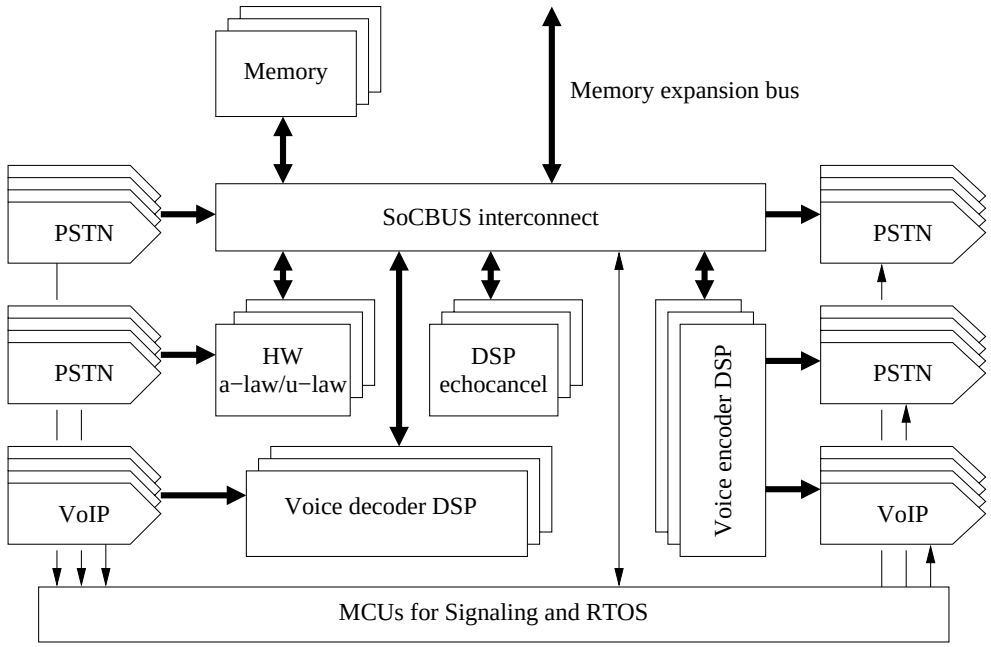


Figure 9.5: Data flow architecture of a PSTN/VoIP gateway chip

west, south, east, down, and combinations of these, e.g. north-west. Since the network do not change when the chip has been designed this knowledge is static and decided at the time of network layout design. The routing decisions are simply based on the destination address and the known direction. If there are more than one direction that leads to the destination one is selected according to a round robin scheduling. If the primary selection is occupied the second choice will be used instead. If there are no free outputs that lead towards the destination the routing will fail and the switch will return a negative acknowledgment to the source.

9.5 Application: The PSTN / VoIP gateway

As an example of a system where an on-chip switched network is suitable we have selected a gateway connecting between both the public switched telephone network (PSTN) and voice-over-IP (VoIP). The data flow architecture for such a gateway chip is shown in fig 9.5. The architecture supports PSTN to PSTN, PSTN to VoIP, VoIP to PSTN, and VoIP to VoIP connections. In addition to this it also supports echo canceling and A-law/u-law companding. The connections to the chip is typically T1/E1 (24/30 channels) or even up to OC-3 (capable of carrying

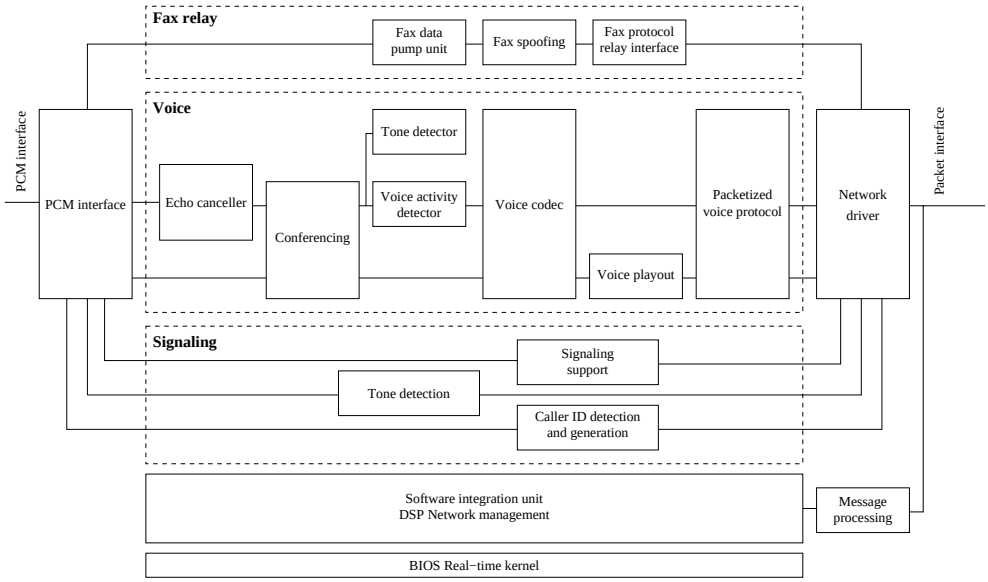


Figure 9.6: Software architecture for a PSTN/VoIP gateway

2048 channels) for PSTN and Ethernet for VoIP. This hardware architecture is used as a basis for the software architecture, see fig 9.6, and has been suggested by the industry [13].

We propose using the SoCBUS switched interconnect for dynamic connections of program loading, payload delivery, computing buffer passing, and control signaling in this architecture. A simplified version of the architecture has been used for simulations, see section 9.7.2, to show the suitability of SoCBUS as an interconnect structure for hard real-time systems.

9.6 Simulation environment

In order to assess the performance of our proposed network we have developed a cycle true behavioral simulator for our SoC network and showed the usefulness both in the research as well as the customer design flow [9]. This behavioral simulator is event driven and implemented in C++.

The simulator uses models of the state machine in the switching nodes and the network end of the wrappers. A slightly simplified state chart for the specific implementation of the PCC scheme state machine for a switching node used in our simulations can be found in fig. 9.7. The switch starts up in the idle state (1) waiting for a request. When a route request packet is received the switch tries to find a route (2) that is available and leads towards the destination. If such a

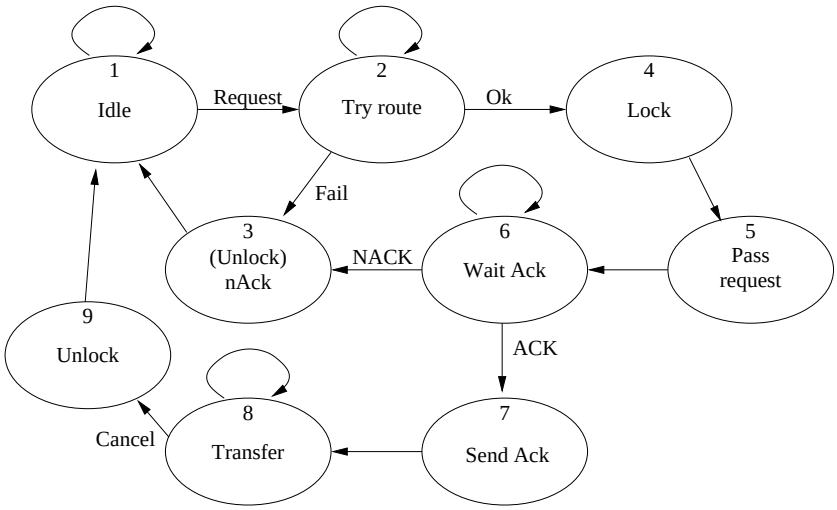


Figure 9.7: State diagram for the FSM implemented in the switching node

route is not found a negative acknowledgment is returned to the sender (3). If a route is found this is locked (4) and the switch passes the request packet on via the output (5). The switch then waits for an acknowledgment to be received from the output (6). If a negative acknowledgment is received the switch will release the lock and pass on the negative acknowledgment (3). If the result were a positive acknowledgment the switch passes the acknowledgment on to the source (7) and enters the transfer state (8). When the transfer is finished and the request cancel is received from the source the switch will release the lock (9) and return to idle. As can be seen in the figure, the switch FSM states are closely related to the transfer transaction phases.

9.6.1 Latency analysis

There are two primary types of latency in the network. One is related to the route setup time and one is related to the payload transfer. Both latencies are linearly dependent on the distance between source and destination. The route setup latency consists of first the request handling latency that is minimum of four bus cycles per switch for request buffering and route selection. The second part of the route setup latency is the acknowledgment latency that is one cycle per switch.

Since the network is circuit switched the data transfer latency is just one cycle per switch to allow for retiming.

Due to the high internal clock rate in the SoCBUS the latency will appear lower from the IP block perspective. With a SoCBUS clock of 1.2 GHz and the

typical IP block clock frequency of 300 MHz the apparent latency will be only a fourth. The route setup latency for every switch will appear as 1 cycle and the data transfer latency as 0.25 cycles.

9.7 Stimuli and simulations

9.7.1 Random pattern stimuli

Two sets of stimuli has been used in the simulations of the network. The first stimuli is random pattern communication with different mean usage levels. The stimuli simply consists of a set of transfers from a random source to a random destination at a random time with a random length where the random numbers have uniform distribution over specific intervals. This random communication patterns do not consider whether the destination port is occupied. Thus there is always a risk of the destination not being reachable at the time of transfer and can be rejected at the switch connecting to the destination port. The random pattern stimuli is mainly useful for modeling a general purpose computation platform.

A plot of the relative first-time blocking rate vs the mean usage for a 8x8 network can be found in fig 9.8. The mean usage here is defined as the ratio of used payload bandwidth at the sources divided by the maximum theoretical bandwidth of the sources (i.e. 64 GByte/s at 1 GHz). Three different cases have been simulated. The first case, see fig 9.8(a), uses a completely random transfer pattern. The second case, see fig 9.8(b), limits the distance of half of the transfers to four hops while the second half is still random using the maximum limit of 15 hops. Finally in the third case, see fig 9.8(c), all the transfers are using the short distance of four hops maximum. As can be seen in the graphs, the network is not suitable for totally random traffic but can be acceptable if the traffic is mainly local and only occasionally uses longer distances. The reason is that the longer distance transfers consume more resources in the network that may be needed by other transfers and thus the probability of blocking increases.

With the random pattern simulations showing bad performance compared to the theoretical limit we propose using a pre-runtime static scheduling [14] of most communications and allowing a certain slack to allow some intermittent unscheduled traffic.

9.7.2 PSTN/VoIP gateway

To simulate the PSTN/VoIP gateway introduced in section 9.5 we assume a model that has 7x7 switches running at 1 GHz where the memory swapping ports is connected to the topmost switches and the inputs and outputs are connected to the

leftmost nodes. The rest of the network is filled with 18 processors that perform a first layer of computation (e.g. echo canceling) and 18 processors that perform a second layer of computation (e.g. companding). The data flow in this simulation setup is that voice data arrives and is routed to the appropriate processor. The associated computing buffer is downloaded from the swap memory and the processing is done. When the computations have finished, the computing buffer is swapped to memory and the output from the processor is routed to the second layer processor. The second layer processor does basically the same thing and its output is sent to the correct output.

With a basic frame for a voice transfer of 4 ms that contains 32 bytes of data in conjunction with all swap buffers being 200 bytes each simulations have shown that the network capacity is up to 48000 voice channels with 0% blocking when consuming a total bandwidth of 12.8 GBytes per second if the network accesses are scheduled appropriately. The limit here is the port usage of sources and destinations that saturates before the network bandwidth is used up. Also this would require that a processor can sustain computing one voice channel in 1.5 us which is possible with the high end processors available today. The real limiting factor in such a gateway chip would most likely be the power consumption.

This limit of 48000 channels assume perfect scheduling of transfers which is not feasible in a real product. Even with significantly worse scheduling it would be possible to run a chip that handles 5000 channels together with a certain level (3-5%) of communications that are not scheduled. This is probably the real-life limit of a real gateway product based on issues such as risk of failure and economy.

9.8 Future work

The simulator is finished and will be used to get more simulations to prove the concept for a broader set of real-time applications. Further work is currently being done on the implementation of the SoCBUS and the goal is to build a demonstrator system that uses the SoCBUS concept. Currently the switching nodes are being implemented together with some example wrappers covering a few common interfaces. We further look into the verification of the system and the impact on system integration.

9.9 Conclusions

A two dimensional network for on-chip communications, the SoCBUS has been introduced. The operational principle of the SoCBUS using packet connected circuit, PCC, has been analyzed and explained in detail.

We have shown with simulations that the system is not suitable for general purpose computation platforms that exhibit random traffic patterns because of the high probability of route blocking when running without schedule. We also show that limiting the distance of communication in the random pattern case will allow a much higher bandwidth usage without saturation.

We further show that a pre-runtime static scheduling of communications approach is necessary for a hard real-time embedded system to be able to use a significant percentage of the theoretical maximum bandwidth. Using this approach we show the appropriateness for one particular application, a PSTN/VoIP gateway capable of handling up to 48000 voice channels simultaneously with perfect scheduling and that the SoCBUS will be capable of supporting a level of 5000 voice channels even with realistic scheduling and some unscheduled traffic.

9.10 References

[1] ARM, Inc., "AMBA 2.0 specification," *http://www.arm.com*.

[2] Daniel Wiklund and Dake Liu, "Switched interconnect for system-on-a-chip designs," in *Proc of the IP2000 Europe conference*, 2000.

[3] Iikka Saastamoinen, David Sigüenza-Tortosa, and Jari Nurmi, "Interconnect IP node for future system-on-chip designs," in *IEEE int'l workshop on Electronic design, Test, and Applications*, 2002.

[4] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli, "Addressing the system-on-chip interconnect woes through communication-based design," in *Proc of the design automation conference (DAC)*, 2001.

[5] "IEC/ISO 7498-1 information technology - open systems interconnection - basic reference model: The basic model," *http://www.iso.org*.

[6] Hussein G. Badr and Sunil Podar, "An optimal shortest-path routing policy for network computers with regular mesh-connected topologies," *IEEE transactions on computers*, 1989.

[7] Christopher J. Glass and Lionel M. Ni, "Fault-tolerant wormhole routing in meshes without virtual channels," *IEEE transactions on parallel and distributed systems*, vol. 7, no. 6, 1996.

[8] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proc of the design and test in Europe (DATE) conference*, 2000.

[9] Daniel Wiklund and Dake Liu, "Design of a system-on-chip switched network and its design support," in *Proc of the Int'l conference on communications, circuits and systems (ICCCAS)*, 2002.

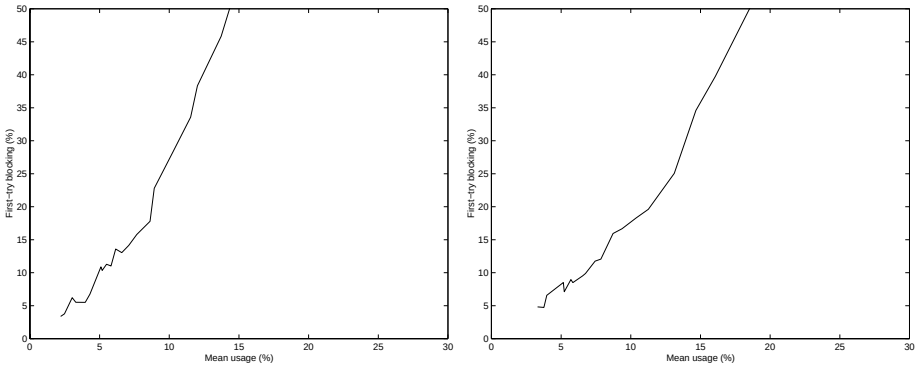
[10] William J. Dally and Brian Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc of the design automation conference (DAC)*, 2001.

[11] C. Svensson, "Optimum voltage swing on on-chip and off-chip interconnect," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 7, pp. 1108–1112, 2001.

[12] Peter Caputa and Christer Svensson, "Low power, low latency global interconnect," in *Proc of the 15th int'l ASIC/SoC conference*, 2002.

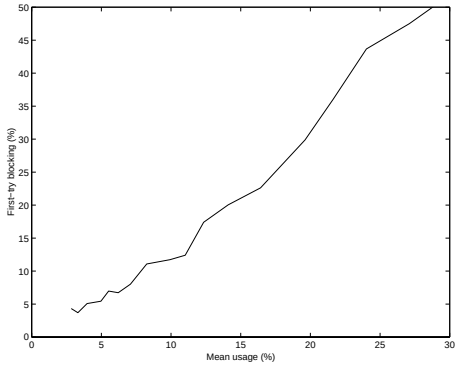
[13] William E. Witowsky and Dennis R. Gatens, "Carrier class, high density VoP," *Telogy Networks whitepaper, http://www.telogy.com*, 2002.

[14] J. Xu, "A comparison of priority scheduling and pre-run-time scheduling," in *Proc of the Embedded Systems Conference in China*, 2002.



(a) Mean distance only

(b) Short and mean distance



(c) Short distance only

Figure 9.8: Relative first-time blocking ratio vs mean used payload bandwidth using random pattern stimuli

Part IV

Appendix

Appendix A

Simulator overview

A.1 Introduction

This appendix is an introduction to the network simulator developed as a part of the research project. This include snippets of my paper to the Swedish System-on-Chip Conference in 2002 [1] which is not used as a part of this thesis. Much of the information in this appendix can also be found in paper 2, see chapter 8.

A.2 Simulator background

Because of the vast space of different design and implementation decisions involved in such a network, a behavioral simulator has a very high importance in the design flow for system integration using the switched system-on-chip network. The same behavioral simulator can also be used to evaluate and select the different design parameters, e.g. routing algorithms and switching schemes, while designing the actual network. The goal with the simulator is to have a general simulator for all simulation needs within the project, whether it may be network design decisions or final chip implementation evaluation.

The input to the behavioral simulator is the models for network components with a configuration setup including network size etc. A description of the traffic patterns in the specific setup is also necessary to get relevant simulation results.

The current simulator runs behavior level C++ models of the components so the simulator output will be on a behavioral level. The functionality of the behavior models must be verified against the RTL code for the components using some other means of simulation.

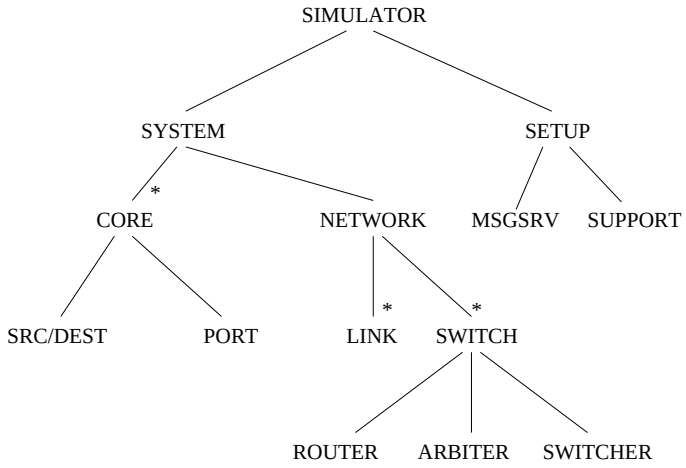


Figure A.1: Object structure of the behavioral simulator network model

A.3 Behavioral simulator implementation

The behavioral simulator is developed using object oriented programming in C++. The simulator is implemented using a message passing structure that resembles the structure of the real network system. The simulator is event-driven and both bit and cycle true in its execution.

The object structure of the simulator can be found in figure A.1. The leaf nodes in the system branch corresponds to the actual network components except for the switch that is subdivided into its subtask components where the arbiter is for local arbiting between the ports of a switch. The setup branch is the message passing support for the simulator. This program structure is very suitable for the simulation of an essentially message-passing hardware like the on-chip network presented in this paper.

The kernel is implemented using an event mechanism similar to those used in VHDL simulators using event lists and delta cycles to keep track of absolute time ordering of events. This implementation is very simple and efficient while allowing considerable flexibility in the simulation setup. It is also very general allowing the simulator kernel to be used in other simulators if needed.

A.4 Stimuli generation

The stimuli for the simulator can either be generated by a third party program, e.g. as a result of system simulations on process or behavioral level, or be generated in the stimuli generator accompanying the simulator. This stimuli generator can

create stimuli in accordance to some standard patterns such as completely random traffic, random traffic mixed with shorter mean distance traffic, etc.

A.5 Conclusions

A very simple yet powerful simulation kernel has been implemented using C++. This kernel has then been used as a basis to build a complete behavioral simulator for the evaluation of design and implementation decisions in the SoCBUS on-chip switched network. A stimuli generator has also been developed as a tool for the simulator.

A.6 References

[1] D. Wiklund, "Implementation of a behavioral simulator for on-chip switched networks," in *Proceedings of the Swedish System-on-Chip Conference (SSoCC)*, March 2002.

Appendix B

Index