Design of a System-on-Chip Switched Network and its Design Support *

Daniel Wiklund†, *Dake Liu*

Dept. of Electrical Engineering
Linköping University
S-581 83 Linköping, Sweden

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.

Keywords —System-on-chip, network, switch, simulation, design flow

I. 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 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 1b, that do not suffer from those problems.

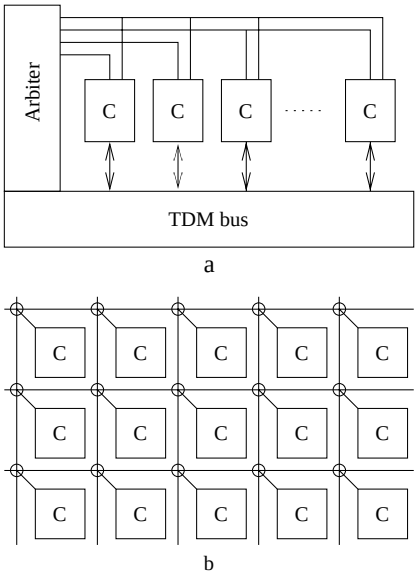

Fig. 1. Comparison of traditional TDM bus and a switched interconnect bus replacement

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.

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 struc-

*This work is supported by the Integrated Electronics (INTELECT) program of the Swedish Foundation for Strategic Research (SSF). Daniel Wiklund is a Ph.D. student at the division of Computer Engineering and Dake Liu is the chairman professor of the division of Computer Engineering. Phone +46 13 28 8965, fax +46 13 13 9282.
†Corresponding author, email: danwi@isy.liu.se

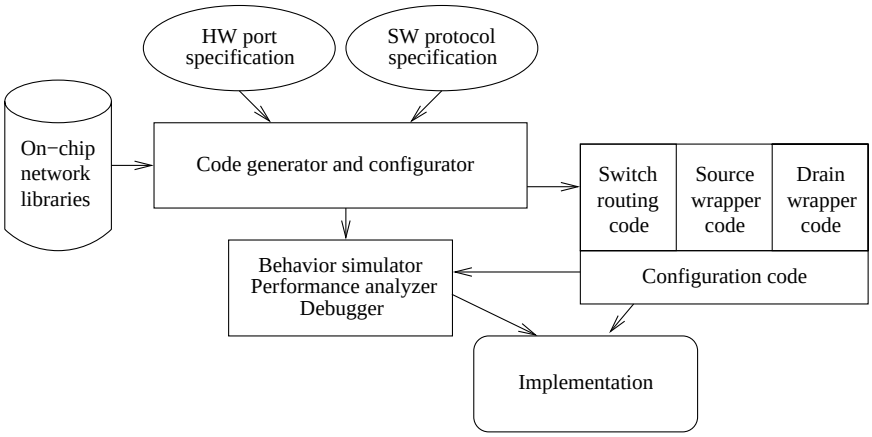

Fig. 2. Design flow for a on-chip network design

ture. The platform should be easy to use and ease the burden of verification during the system-on-chip integration phase.

II. 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 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 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.

III. 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 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.

IV. Network components

The network can be seen as a two-dimensional matrix of tiles shown in fig 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

Table 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	Bidirectional	Bidirectional	Bidirectional	Unidirectional	Bidirectional
Bisection BW	2	$\sqrt{N}$	1	N	N
Wiring complexity	Low	Low	Low	High	High
Max wire length	Low	Low	Medium	High	High
Routing complexity	Low	Medium	Low	Medium	High

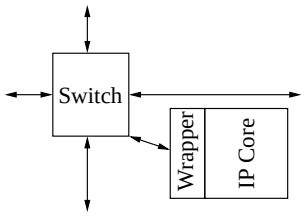

Fig. 3. Network connected processing tile

Table 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

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.

V. 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 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.

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.

VI. 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.

VII. 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 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.

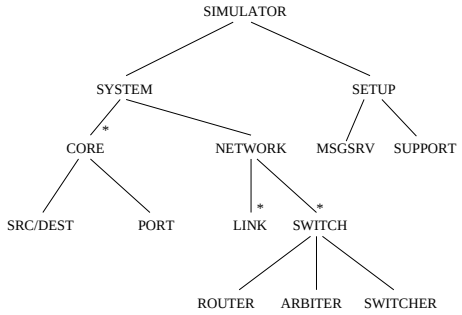

Fig. 4. Object structure of the behavioral simulator network model

Table 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

VIII. 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 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.

IX. Behavioral simulator output

Taking one simulation result as an example of output from the simulator, we considered the case where

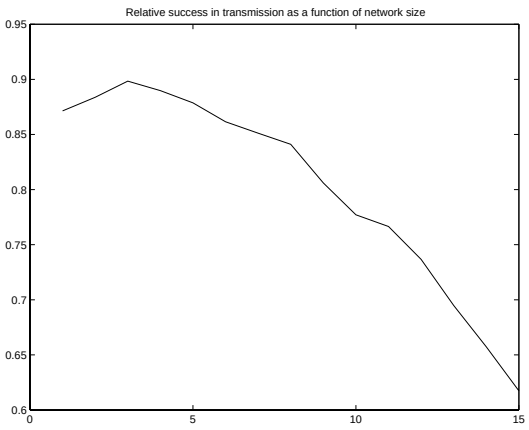

Fig. 5. Example of simulator results

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 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.

X. 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.

XI. 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.

XII. References

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

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

[3] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc of the DAC*, 2001.

[4] M. Sgroi et al., "Addressing the system-on-chip interconnect woes through communication-based design," in *Proc of the DAC*, 2001.

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

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

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