A simple clockless Network-on-Chip for a commercial audio DSP chip

Stensgaard, Mikkel Bystrup; Bjerregaard, Tobias; Sparsø, Jens; Pedersen, Johnny Halkjær

Published in:
Euromicro Conference on Digital System Design: Architectures, Methods and Tools

Link to article, DOI:
10.1109/DSD.2006.17

Publication date:
2006

Document Version
Publisher's PDF, also known as Version of record

A simple clockless Network-on-Chip for a commercial audio DSP chip

Mikkel B. Stensgaard†, Tobias Bjerregaard†, Jens Sparsø†, and Johnny Halkjær Pedersen‡
†Informatics and Mathematical Modelling (IMM) ‡Oticon A/S
Technical University of Denmark (DTU) www.oticon.com
mail: {mibs,tob,jsp}@imm.dtu.dk

Abstract

We design a very small, packet-switched, clockless Network-on-Chip (NoC) as a replacement for the existing crossbar-based communication infrastructure in a commercial audio DSP chip. Both solutions are laid out in a 0.18 μm process, and compared in terms of area, power consumption and routing complexity. Even though the NoC turns out to be larger and more power consuming than the existing crossbar implementation, it still accounts for less than 1% of the total chip area and power consumption, and is justified by a long list of advantages: The NoC is modular, scalable, and in contrast to the existing crossbar, it allows all blocks to communicate. The total wire length is decreased by 22% which eases the layout process and makes the design less prone to routing congestion. Not least, the communicating blocks are decoupled by means of the NoC, providing a Globally-Asynchronous, Locally-Synchronous (GALS) system where independent clocking of the individual blocks is enabled. This study shows that NoCs are feasible even for small systems.

1 Introduction

Current integrated circuits contain millions of transistors. In order to utilize the available chip area, it is becoming common design practice to adopt the modular System-on-Chip (SoC) design style. In SoC, a number of IP blocks such as Microprocessors, Memories, and Digital Signal Processors are embedded and connected by a communication infrastructure. As technology advances into the deep submicron domain, billions of transistors become available, and the number of IP blocks is foreseen to increase dramatically. This increases the demand for a scalable, plug-and-play, communication infrastructure which can provide the necessary bandwidth and support parallel communication in different parts of the chip. Shared busses are infeasible in this context and while hierarchical busses solve some of the problems at hand, they are still insufficient. A proposed solution is a segmented, shared, communication structure, denoted Network-on-Chip (NoC) [1][10].

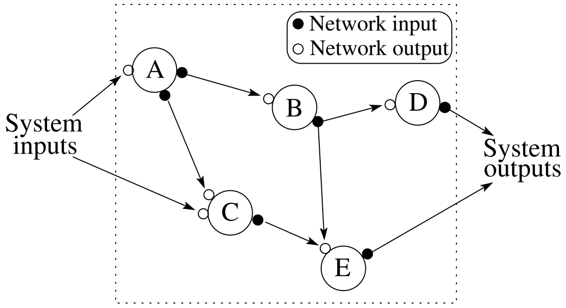


Figure 1. The network sets up a dataflow between the different audio processing blocks. Blocks can have more than one input/output.

Most NoC research has focused on functionality and speed of different implementations and topologies. This leads to fairly large and feature rich NoCs, providing large amount of bandwidth and advanced Quality-of-Service (QoS) features. Examples of such NoCs are Nostrum [11], Mango [4], XPipes [3] and Æthereal [7]. While a lot of research has been done, to our knowledge there has not been many demonstrations of NoCs in real applications.

In this paper we design a small, packet-switched, source-routed, clockless NoC for a commercial audio DSP chip. The existing communication infrastructure, which is implemented as a partially connected crossbar, is replaced and compared with the NoC. This is a unique opportunity to demonstrate a NoC in a real application.

As the required bandwidth is low, and the application has very low power consumption, a feature rich NoC is infeasible. Instead, we focus on simplicity and design the NoC as small as possible. Previous simple NoCs include FLEET-zero [5] and CHAIN [2], but our design is even simpler than these. The NoC is constructed as a non-pipelined, binary tree and employs a clockless, 4-phase, bundled data protocol to minimize the power consumption. It includes net-

Proceedings of the 9th EUROMICRO Conference on Digital System Design (DSD'06)
0-7695-2609-8/06 $20.00 © 2006 IEEE

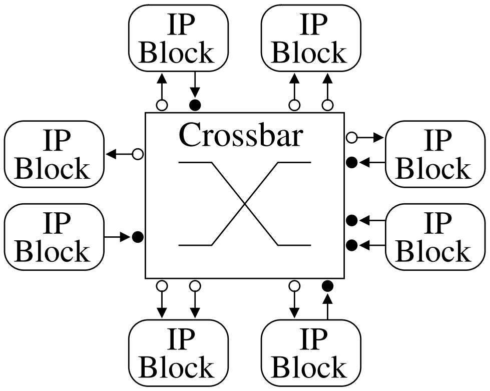


Figure 2. The system consists of a number of audio processing blocks communicating through a crossbar.

work adapters to interface with the communicating blocks, handle multicast and perform synchronization. The clockless modules are implemented using standard cells, and the entire design flow is using standard synchronous design tools. Both the existing crossbar and the NoC are laid out in order to estimate the power consumption, area and the total length of wire which affects the routing complexity.

Section 2 introduces the application and its current crossbar implementation. The NoC design is described in section 3 and section 4 comments on the implementation. The results are presented and discussed in section 5 and 6, respectively.

2 Application and original implementation

The application is a configurable DSP for audio applications developed at *Oticon A/S*. It consists of a number of audio processing blocks, communicating by means of a network. Figure 1 illustrates how static connections are set up between the different IP blocks to control the flow of data through the DSP. The IP blocks include microphone inputs, headphone outputs, audio processing blocks and an interface to a microprocessor. The number of IP blocks and their specific functionality is confidential and is not important in the context of this paper. Thus, in all figures, the IP blocks will be treated as generic IP blocks.

From a practical system point of view, figure 2 illustrates how the IP blocks are connected to a crossbar which is used to communicate data between the blocks. A block can have multiple input and output ports, each port consisting of 18 data bits and a flow control bit.

The DSP is globally synchronous. A *main clock* running at 1-10 MHz is used to clock all blocks in the system. In addition, a *sample clock* is used to control the injection of data into the system, at e.g. the microphone inputs. The sample clock thereby indirectly determines how often data is communicated in the network as well. It runs 96 times lower than the *main clock*, approximately 10-100KHz.

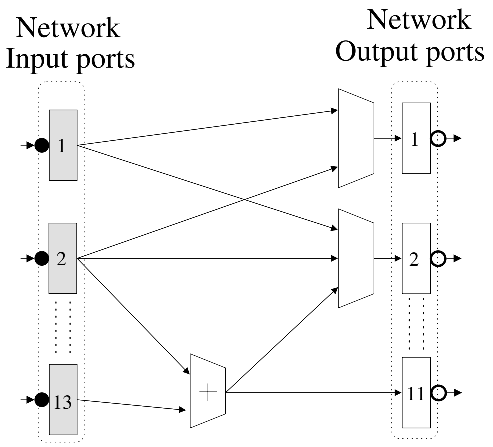


Figure 3. The network is currently implemented as a subset of a fully connected crossbar using multiplexers at the output ports of the network.

A specific configuration of the network is denoted a use-case, as the one illustrated in figure 1. Each channel in the figure represents a single data transmission each sample period. The specific use-case shown in figure 1 involves 7 unicasts and 1 multicast, which is a typical amount of communication for the DSP. Hence, the communication is very sparse and the needed bandwidth is low.

The network contains 13 inputs and 11 outputs, and must support all possible use-cases in which the DSP is to operate. It is currently implemented as a partially connected crossbar using multiplexers at the network output ports. This is illustrated in figure 3. Three of the multiplexers contain adder functionality, and are denoted *MuxAdders*. Besides operating as normal multiplexers, they are able to wait for several inputs and add these before forwarding the result. This can for example be used to overlay ringtones. The *MuxAdders* are connected to a network output port, but are also used as input to other multiplexers as illustrated in figure 3. The crossbar is statically configured by controlling the multiplexers which contain between 2 to 5 inputs. The normal multiplexers implement a fully combinatorial network, while the *MuxAdders* are clocked due to their functionality.

Advantages of this implementation are that multicast is implicitly supported because an input is routed to all feasible outputs, and that the multiplexers are very small. The disadvantages are that the crossbar scales poorly, that there is a large amount of wires which consume power and complicates routing, and that only a subset of the blocks can communicate. This also means that the crossbar is application specific.

3 Details on the NoC design

The following subsections describe the details of the NoC design. As the NoC is meant as a direct replacement of the existing crossbar, the interface between the IP blocks and the network is maintained. This also allows verification of the NoC by inserting it into the original application.

Bandwidth requirement for this application is very low, as the individual blocks injects data into the network at most once every 96 *main clock* cycle. Additionally, latency in the order of μs is acceptable, relaxing the requirements even more. The NoC is therefore designed with simplicity in mind, employing a simple topology, as few queuing-buffers as possible, and a simple multicast implementation. For a more comprehensive discussion of the design choices and implementation we refer to [13].

3.1 Separation of communication and computation

In order to design a NoC which is useable in other applications, communication and computation must be properly separated. This is done by removing the *MuxAdders* from the network and instead implementing them as ordinary synchronous blocks. This also implies that the number of ports in the network is increased. Figure 4 illustrates our *MuxAdder* implementation which contains one input and one output port. It accumulates a number of packets before sending the result as a new packet into the network. All three *MuxAdders* adds an additional network input and output ports to the network. We are able to remove two network output ports, as two if the IP blocks only receive data from the *MuxAdders*. The separation thereby increases the number of network ports to 16 inputs and 12 outputs.

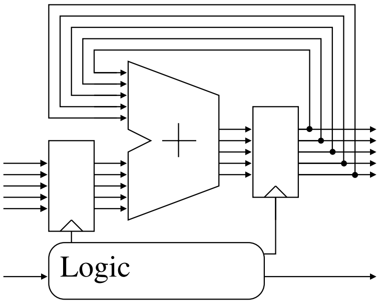


Figure 4. The *MuxAdders* are implemented as synchronous blocks which accumulate the received packets.

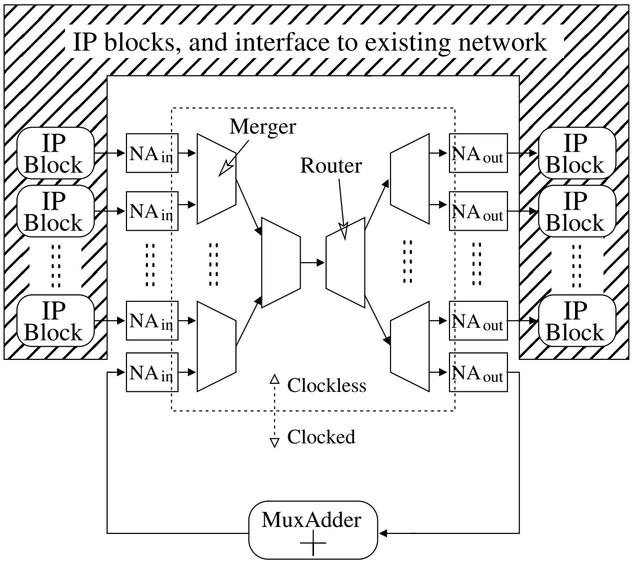


Figure 5. Overview of the NoC which consists of router and merger modules, besides network adapters to interface with the blocks and handle multicast.

3.2 Overview of architecture

Figure 5 shows an overview of the NoC architecture. The hatched area represents the IP blocks and the non-hatched area the network. The interface between the IP blocks and the network is that same as in the original application.

The NoC consists of a clockless switching network that handles the routing of packets, and network adapters to interface with the clocked blocks. At the input port, the NA_{in} network adapter encapsulates the data in a *packet* that also contains the route to the destination block. The switching network then routes the packet to the destination port where the NA_{out} network adapter strips the data from the packet, and interfaces with the connected block. A packet consists of a single flow control digit (flit). This is because each packet is small, and serialization and de-serialization would contribute a considerable overhead in both area and power consumption.

A binary tree topology is chosen as it takes up the least amount of area, and as the topology does not contain any cycles, deadlocks can not occur. This avoids the need for pipeline-buffers inside the switching network in order to avoid deadlocks, thereby simplifying the NoC considerably. The tree is constructed using two types of modules: An arbitrating *merge* which merges two inputs into one output and a *router* which routes a single input to one of two possible outputs. Routing information is contained in the packets, represented as a single bit for each routing decision. A routing bit is consumed when a packet passes a router. The network contains 12 output ports which means that a maximum of 4 routing decisions are needed. Each packet contains 25 bits: 18 data bits, 4 routing bits and 1 control bit used in the *MuxAdders*.

This topology does not exploit locality, as all packets have to pass through the root of the tree. For larger applications it might be advantageous to make the tree bidirectional instead.

3.3 Clockless design

The NoC employs 4-phase, bundled data, clockless circuits [12]. A clockless NoC supports flow-control inherently, and is able to provide large amounts of bandwidth without pipelining and/or a fast and power consuming clock. This is because the NoC can handle several packets within the time that corresponds to a single clock period of the 1 MHz clock. A clockless implementation also means that the design becomes Globally-Asynchronous, Locally-Synchronous (GALS), thereby enabling the blocks to run at different clock frequencies. Besides easing global timing closure, this enables power savings using dynamic voltage scaling and/or lower frequencies in some blocks.

In order to minimize the area and power consumption, the actual switching network does not contain any pipeline-buffers. Queuing-buffers are inserted at the edge of the NoC in the network adapters as described in section 3.5. This means that the NA$_{in}$ network adapters handshake directly with the NA$_{out}$ network adapters and that packets are arbitrated in the merge modules.

3.4 Multicast

In contrast to the existing crossbar, multicast is not implicitly supported in a packet-switched network. We have chosen a solution where multiple packets are generated in the NA$_{in}$ network adapter at the input to the network as shown in figure 6. The multicast functionality is implemented in the asynchronous domain to decouple it from the connected IP block. The data is the same in all packets, while the route is specific for each packet. The route is generated by the multicast module, which also handles the handshaking. Each NA$_{in}$ network adapter is able to handle two multicasts, but this number could be increased at design time depending on the application requirements.

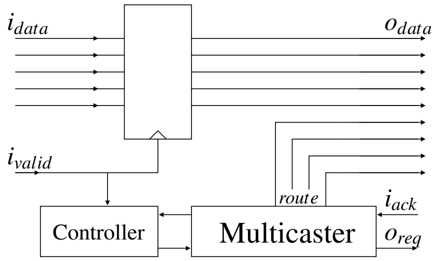


Figure 6. The NA$_{in}$ network adapter handles multicast by generating multiple packets with different routes.

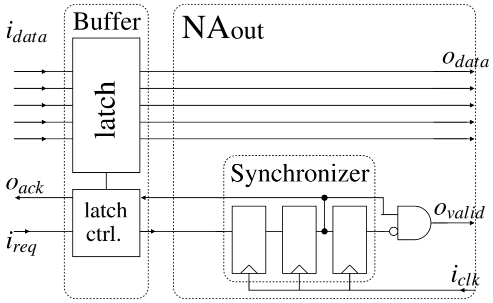
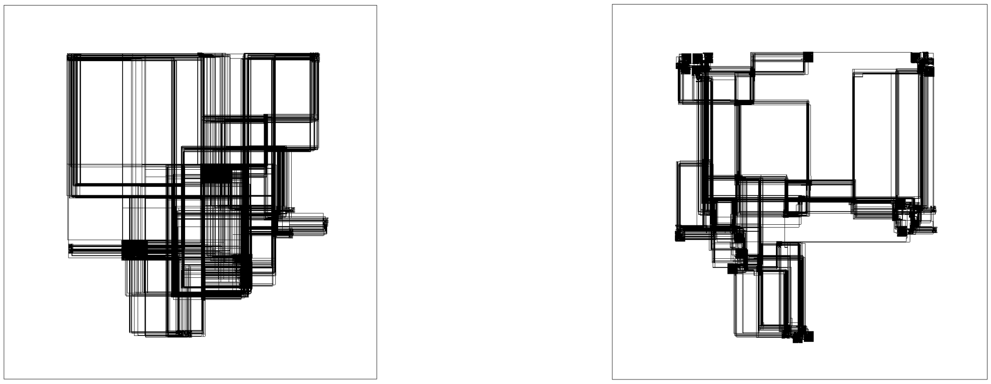


Figure 7. The NA$_{out}$ network adapter handles synchronization and generation of a *valid* signal for one clock cycle. The fully decoupled latch decouples the synchronization from the switching network.

3.5 Synchronization and decoupling

The NA$_{out}$ network adapter handles synchronization with the connected block. As shown in figure 7, this is implemented as a simple 2 flip-flop synchronizer [9], clocked by the block to which it is connected. The third flip-flop is used to produce a flow control signal that indicates the validity of data for a single clock cycle.

Packets must be synchronized before they can be acknowledged and the acknowledge signal is therefore delayed two clock cycles. If the synchronization was not decoupled from the switching network, the switching network would be locked during the synchronization. This is because the switching network is not pipelined and the NA$_{in}$ network adapter handshakes directly with the NA$_{out}$ network adapter. In order to decouple the synchronization from the switching network, a queuing-buffer using fully-decoupled latch controllers [8] is inserted. A fully-decoupled latch controller is a clockless storage element where the handshake on the input port is totally independent of the handshake on the output port. This allows the handshake with the switching network to complete, even though the synchronization takes 2 clock cycles. When a single-flit queuing-buffer is inserted, only a single packet can be received every 4 clock cycles without locking the switching network. Alternatively a larger queuing-buffer can inserted



(a) Original crossbar with 810 mm wire.

(b) NoC with 630 mm wire.

Figure 8. Layout of the two designs on a 2.8^2 mm^2 chip which shows the wire distribution.

to increase the number of packets waiting to be synchronized. In this application, only the three *MuxAdder* ports can receive more than one packet within 96 *main clock*. A three-flit queuing-buffer is inserted at these ports, while all other ports contain a single-flit queuing-buffer.

4 Implementation

Both networks are implemented using a 1.3V low leakage 0.18 μm standard cell library from STMicroelectronics. The original crossbar has been synthesized using Synopsys Design Analyzer, while the NoC has been designed by connecting a number of clockless modules. The clockless modules are designed by describing their behavior as a Signal Transition Graph (STG), and using Petrify [6] to extract boolean expressions for the circuit realizations. Basic clockless cells, such as the Muller C-elements and arbitration modules, are created by hand using the standard cell library. Both designs have been simulated, laid out, back-annotated and power-estimated using ModelSim, Cadence SoCEncounter, and Synopsys Design Analyzer.

The NoC is designed as a direct replacement for the original crossbar. This enables us to verify the functionality of the NoC using the testbenches created for the original DSP chip.

5 Results

In order to compare the NoC with the original crossbar, the two designs are laid out on a chip of 2.8 mm x 2.8 mm, corresponding to the size of the original chip. Only the networks are laid out and not the actual IP blocks. The input and output ports are placed such that they imitate a realistic placement of the processing blocks.

The ports are placed at exactly the same position in the two layouts, enabling a comparison independent of the actual port placement. The layouts include elements of uncertainty in terms of placement of multiplexers, in the original crossbar, and routers/mergers in the NoC. Even though this placement will affect the wire length, the comparison still illustrates the trend concerning wire length and power consumption. Also, as the actual blocks are not laid out, routing restrictions from these are not included. The NoC consists of short shared links, why it is easier to cope with routing restrictions in the NoC than in the original crossbar. Hence, this does not invalidate the comparison, but makes it more conservative in favor of the crossbar.

Figure 8 shows the wire distribution in the two layouts, and is included to illustrate the different layouts and to emphasize that the NoC is constructed of short shared links. It is seen from the layouts, that routing is more complex in the original crossbar. The total length of wire in the NoC is decreased by 22%, compared to the original version.

5.1 Area

Table 1 shows the different area contributions in the original crossbar. It consists of a number of small multiplexers and a driver contribution. The driver contribution is an estimate of the size of the drivers needed to drive the long wires, based on the capacitance extracted from the layout. Many of the wires have capacitances of 0.5 pF, requiring drivers of considerable size. In comparison, the largest capacitance in the NoC is 0.1 pF due to shorter, shared links.

Table 2 shows the area contributions in the NoC, of which 39% is in the switching network and 61% in the network adapters and multicast implementation. It is noticeable that approximately 40% of the area is to be found in the one-flit queuing-buffer in the network adapters. This, and the fact that the NoC takes up less than 1% of the total chip area, emphasizes how small the NoC is! Still, it is 3.7 times larger than the original crossbar, but scales lin-

Module	**# of instances**	**Area/instance**	**Total area**	**Percent**	**Power Consumption**	**Percent**
Multiplexers					0.04	3 %
Mux (2 inputs)	2	544	1088	5 %		
Mux (4 inputs)	7	1077	7539	35 %		
Mux (5 inputs)	6	1323	7938	37 %		
Drivers			5000	23 %	0.1	8 %
Links (wires)					1.15	89 %
Total			21565 μm^2	100 %	1.29 μW	100 %

Table 1. Cell area and power consumption of the original crossbar.

Module	**# of instances**	**Area/instance**	**Total Area**	**Percent**	**Power Consumption**	**Percent**
NA_{in}				*(36%)*		*(11%)*
Control logic	16	98	1568	2 %	0.06	1 %
Multicaster	16	795	12720	16 %	0.22	5 %
Buffer	16	885	14160	18 %	0.23	5 %
NA_{out}				*(25%)*		*(41%)*
Control logic	12	45	540	1 %	0.06	1 %
Buffer	18	987	17766	22 %	0.54	12 %
Synchronizer	12	147	1769	2 %	1.25	28 %
Switching network				*(39%)*		*(48%)*
Merger	15	1065	15975	20 %	0.61	14 %
Router	11	1343	14773	19 %	0.61	14 %
Links (wires)					0.91	20 %
Total			79271 μm^2	100 %	4.49 μW	100 %

Table 2. Cell area and power consumption of the NoC.

early with the number of ports. Besides, there is a long list of additional advantages as discussed in section 6.

5.2 Power consumption

Power consumption is estimated using a specific configuration which represents a typical amount of communication. It involves 5 unicasts and 2 multicasts for each *sample clock* cycle, corresponding to 93750 packets/sec as the system is running at 1 MHz. The injected data is random numbers and the power consumption is estimated at the best-case performance corner, that is 1.6V at -40C°. This corner is chosen because it gives the worst-case power consumption.

Table 1 summarizes the power consumption in the original crossbar. As the multiplexers are very small and consume a small amount of power, 97% of the power consumption is to be found in the links and drivers.

The power consumption in the NoC is summarized in table 2. 47% of the power is consumed in the switching network, and no less than 28% in the synchronization. These numbers are discussed in the following section.

6 Discussion

First, we discuss the obtained results and make some general comments. The subsequent subsections go into more detail with a number of important observations with regard to scalability, synchronization, bandwidth and locality.

6.1 General discussion

The NoC is 3.7 times larger than the existing crossbar and uses 3.5 times more power. These figures, however, have to be put into the context that the existing crossbar is very small and almost resembles direct links between the communicating blocks. The NoC still accounts for less than 1% of the total chip area and power consumption. The NoC has a large number of advantages over the existing solution:

- Flexible communication: The NoC enables communication between all blocks. This decrease the design time as no communicating subset have to be specified. In addition, it avoids redesign of the network if the communicating subset is changed.

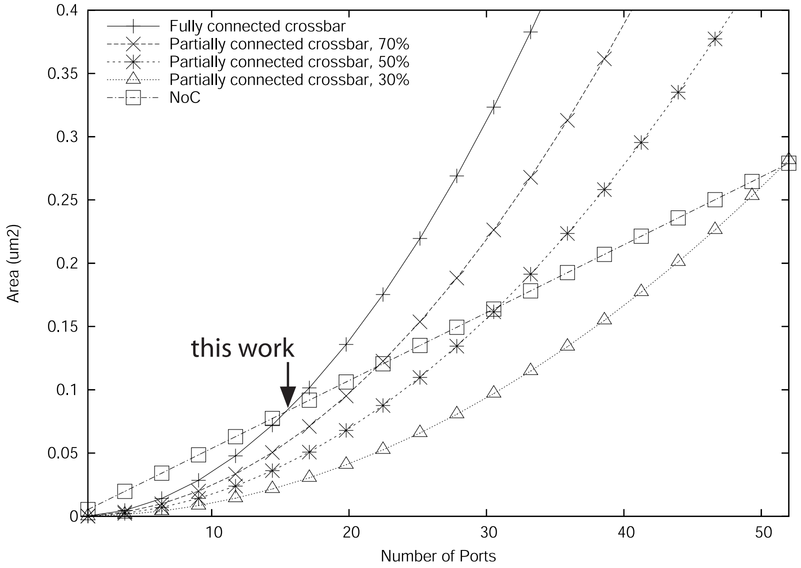


Figure 9. Area estimates as a function of the number of ports.

- Decreased wire length: The total length of wires is decreased by 22%, easing the layout process. In addition, routing congestion is less likely as the NoC uses shared links.

- Scalability: If the number of blocks is increased, the size of the NoC increases linearly while the power consumption increases with the depth of the tree, $O(log(n_i) + log(n_o))$ where n_i and n_o is the number of input and output ports.

- GALS: An important feature of the NoC is that it decouples the IP blocks, enabling them to run at different frequencies. This eases timing closure, and enables power savings using dynamic voltage scaling and/or lower clock frequencies in some blocks.

6.2 Scalability

In figure 9, the cell area of the NoC, a fully connected crossbar, and 3 partially connected crossbars, is estimated as a function of the number of ports. It is assumed that the number of input and output ports is the same, that the size of a multiplexer is linearly dependent of its number of inputs, and that the chip size is constant. The original crossbar is 30% connected, hence the number of ports should be larger than 50 before it becomes the same size as the NoC. If the crossbar was fully connected it would take up the same area as the NoC for the current number of ports!

Other aspects to consider are routing complexity and power consumption. For example, a crossbar can cause routing congestion, especially for a larger number of ports and/or as the crossbar approach full connectivity. Concerning power consumption, it is noticeable that links already consume more power in the original crossbar than in the NoC. This is despite the increased switching activity in both gates and wires in the NoC, and that the number of wires has been increased from 19 to 25. For a larger crossbar, the power consumption approaches the NoCs.

6.3 Synchronization

The synchronization takes 28% of the power consumption. This is mainly because the 1 MHz clock used in all blocks is high relative to the rate of communication. There is room for improvement in this context and this number could be decreased significantly. One option is to gate the clock for the NA$_{out}$ network adapters that do not receive any data in a given configuration. For example, the configuration used to estimate the power consumption, uses only 7 of the 12 output ports. Another option is to clock the synchronization registers at a lower clock frequency. This could be done without any overhead if such a clock already exists inside the connected block.

6.4 Bandwidth and locality

Even though the NoC is designed without any pipeline buffers inside the switching network, it still provides enough bandwidth for this application. The bandwidth could be increased by inserting buffers throughout the switching network. As an example, a single-flit queuing-buffer in the root of the tree, will double the bandwidth.

0-7695-2609-8/06 $20.00 © 2006 IEEE

In the current implementation all links are unidirectional, and all packets have to travel through the root of the tree. By employing 3x3 routers, a bidirectional tree could be build where locality of communication could be exploited. This would increase the size of the NoC, but could decrease the power consumption depending on the number of blocks, the size of the links and the type of communication in the specific application.

7 Conclusion

We have designed a small, packet-switched, clockless NoC for a commercial audio DSP chip and compared it with the original crossbar-based implementation. The NoC gives a long list of advantages over the crossbar. It decreases the length of wires by approximately 22% as it utilizes shared instead of dedicated links. This eases the layout process and decreases the possibility of routing congestion. In contrast to the original crossbar, all blocks are able to communicate and the NoC decouples the communicating blocks, implementing a GALS system. This enables separate timing-closure of the IP blocks, and power savings using dynamic voltage scaling and/or lower clock frequencies in some blocks. The NoC is scalable, making it usable for other applications with a larger number of communicating IP-blocks. At last, the NoC is plug-and-play, requiring almost no design decisions thereby reducing the design time and avoiding the need for redesign of the NoC if late design changes are made.

Even though the NoC is designed as small as possible, it still uses 3.5 times more power and 3.7 more area than the existing crossbar. It must be emphasized that this is not large which can be seen by the fact that 40% of the area is to be found in the single-flit, queuing-buffers at the inputs and outputs of the NoC. The NoC still accounts for less than 1% of the total chip area and power consumption. If the crossbar was fully connected, the NoC and the crossbar would take up the same amount of area for the current number of ports.

This study shows that it is feasible to use NoCs for applications with limited bandwidth requirements. It also indicates that for small applications, a feature rich NoC [11, 4, 3, 7] takes up too much area and uses too much power. We believe there are a large number of current, and future, applications for which a simple NoC is a feasible solution.

Acknowledgment

The authors thank *Oticon A/S.* for providing information about the audio DSP application, and for the cooperation in the M.Sc. project from which this work originates.

References

[1] *Route Packets, Not Wires: On-Chip Interconnection Networks*. Proc. ACM/IEEE Design Automation Conference, June 2001.

[2] J. Bainbridge and S. Furber. CHAIN: A delay-insensitive chip area interconnect. *IEEE Micro*, 22:16–23, 2002.

[3] D. Bertozzi and L. Benini. Xpipes: a network-on-chip architecture for gigascale systems-on-chip. *Circuits and Systems Magazine, IEEE*, 4(2):1101–1107, 2004.

[4] T. Bjerregaard and J. Sparsø. A router architecture for connection-oriented service guarantees in the mango clockless network-on-chip. In *Proc. Design Automation and Test in Europe (DATE'05), ACM sigda, 2005*, pages 1226–1231, 2005.

[5] W. S. Coates, J. K. Lexau, I. W. Jones, S. M. Fairbanks, and I. E. Sutherland. FLEETzero: An asynchronous switch fabric chip experiment. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 173–182. IEEE Computer Society Press, Mar. 2001.

[6] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Transactions on Information and Systems*, E80-D(3):315–325, Mar. 1997.

[7] R. E., G. K., R. A., D. J., van Meerbergen J., W. P., and W. E. Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip. *Computers and Digital Techniques*, 150(5):294–302, 2003.

[8] S. B. Furber and P. Day. Four-phase micropipeline latch control circuits. *IEEE Transactions on VLSI Systems*, 4(2):247–253, June 1996.

[9] R. Ginosar. Fourteen ways to fool your synchronizer. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 89–96. IEEE Computer Society Press, May 2003.

[10] B. L. and D. M. G. Networks on chips: a new soc paradigm. *IEEE Transactions on Applied Superconductivity*, 35(1):70–78, Jan. 2002.

[11] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch. The nostrum backbone - a communication protocol stack for networks on chip. In *Proceedings of the IEEE International Conference on VLSI Design*, pages 693–696, 2004.

[12] J. Sparsø and S. Furber, editors. *Principles of Asynchronous Circuit Design: A Systems Perspective*. Kluwer Academic Publishers, 2001.

[13] M. B. Stensgaard. Design of an asynchronous communication network for an audio dsp chip. http://www2.imm.dtu.dk/pubdb/p.php?3974, Master thesis 2005.