Asynchronous design of Networks-on-Chip

Sparsø, Jens

Published in:
Proceedings of the 25th IEEE NORCHIP Conference 2007

Link to article, DOI:
10.1109/NORCHP.2007.4481080

Publication date:
2007

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

Link back to DTU Orbit

Citation (APA):
Sparsø, J. (2007). Asynchronous design of Networks-on-Chip. In Proceedings of the 25th IEEE NORCHIP Conference 2007 IEEE. DOI: 10.1109/NORCHP.2007.4481080

Asynchronous Design of Networks-on-Chip

Jens Sparsø
Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads, Building 322, DK-2800 Kgs. Lyngby, Denmark
E-mail: jsp@imm.dtu.dk

Abstract— **The Network-on-chip concept has evolved as a solution to a broad range of problems related to the design of complex systems-on-chip (SoC) with tenths or hundreds of (heterogeneous) IP-cores. The paper introduces the NoC concept, identifies a range of possible timing organizations (globally-synchronous, mesochronous, globally-asynchronous locally-synchronous and fully asynchronous), discusses the circuitry needed to implement these timing methodologies, and provides some implementation details for a couple of asynchronous NoCs designed at the Technical University of Denmark (DTU). The paper is written to support an invited talk at the NORCHIP'2007 conference.**

I. Introduction

The Network-on-chip concept [1], [2], [3], [4], [5], [6] has evolved as a solution to a large and diverse set of challenges ranging from synchronization and long distance communication in deep submicron technologies to system-level design methodologies necessary to quickly and safely design and program complex systems-on-chip (SoC) with tenths or hundreds of (heterogeneous) IP-cores, such as processors, memories, I/O-units etc.

This paper addresses timing issues related to the design of such NoC based SoCs, i.e., clocking, synchronization and use of asynchronous circuit techniques.

The paper is organized as follows: Section II introduces some additional NoC fundamentals. Section III identifies possible timing organizations of NoC based SoCs and section IV discusses the circuitry involved in implementing these timing organizations. This leads to conclusions which support asynchronous implementation of NoC routers and links. Section V reviews the implementation of a couple of asynchronous NoCs designed at DTU [7], [8], [9], [10], [11], [12], [13], and finally section VI concludes the paper.

II. Networks-on-Chip

Networks-on-chip (NoCs) are built from communication links and routers which can be composed to form regular topologies such as mesh, torus, tree as well as application specific topologies which are often irregular. IP-blocks are connected to the network nodes using network adapters (NA) which provide standard communication sockets like OCP [14] to the IP-cores. The network itself is typically based on some form of packet switching and often it provides both connectionless best-effort (BE) communication and connection oriented guaranteed-service (GS) communication. Figure 1 shows a fragment of a SoC using a NoC with a 2D-mesh topology.

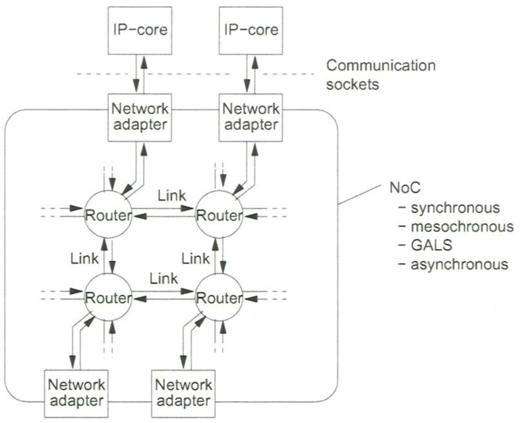

Fig. 1. A fragment of a SoC using a mesh-type Network-on-Chip.

Many proposed NoCs provide a traditional bus-like communication interface to the IP-cores; an interface which provide read and write transactions into a shared-memory address space. The NAs must therefore be able to transform a read or a write transaction into a packet which is sent across the network. Often NoCs use source routing to simplify the routers and a routing table in the source NA is used to convert the shared memory address into a route. The packet payload normally includes both address and data for the transaction. Transmission of one packet through a router may stall another packet and perhaps the only guarantee which can be made is that the packet eventually arrives at the destination NA. This is called best effort (BE) routing.

Many proposed NoCs use wormhole routing where a packet is sent as a sequence of data-words called flits. Often the links use some form of multiplexing at the flit level in order to allow simultaneous transmission of several packets across a physical link. In this way a link may provide several virtual channels between two neighboring router nodes. Using a sequence of such virtual channels it is possible to create an end-to-end virtual circuit connecting a source NA to a sink NA effectively connecting two IP-cores. In this way it may be possible to provide hard latency and/or bandwidth guarantees. This is called guaranteed service (GS) routing.

A router consists of some form of crossbar switch and a number input buffers and/or output buffers—one buffer per port or per virtual channel per port. The switch connects input buffers/ports to output buffers/ports on a per flit basis, and flits belonging to different packets may be handled concurrently provided they do not compete for the same output port/buffer.

Consider as an example a CPU IP-core performing a read from memory IP-core. This transaction results in two packet transmissions: a read-request from the CPU to the memory and a subsequent read-response from the memory to the CPU. In principle the two packets can be transmitted using BE or GS routing or any combination thereof. For more background on NoCs the reader is referred to [3], [4], [5], [6].

The area and power figures for NoC router designs which have been published so far have been on the high side, and many 2nd generation designs have aimed at reducing area and power. A latency of 1-2 clock cycles through a router is often listed as a key design goal.

III. Timing organizations of NoC based SoCs

For the circuit level implementation of a NoC-based SoC a number of timing methodologies are possible:

Several published NoC-based SoCs assume a *globally synchronous operation of the entire SoC*. This is somewhat unrealistic. Large SoCs containing many different IP-blocks typically involve a number of *independent clock domains*, and techniques for passing signals between different clock domains are now standard design practice [15, Chapter 10]. Other proposed NoC-designs implement the *entire NoC as a single clocked module*, arguing that the regularity of the NoC allows clock-skew problems to be handled at modest effort. Synchronization is thus only required on the interfaces between the NoC and the (independently) clocked IP-cores. Many globally-synchronous NoC-implementations have been published, including [16], [17], [18].

As the NoC is a chip-wide structure, it is becoming increasingly difficult to implement as a simple synchronous module. The International Technology Roadmap for Semiconductors [19] predicts a shift towards schemes which can tolerate timing uncertainty. This includes *mesochronous operation* which tolerate a phase difference between the clocks in the communicating blocks [15, Chapter 10], [20], [21] and *globally-asynchronous locally-synchronous (GALS) operation* [22], [23] where clock domains are confined to the individual IP-blocks and individual routers in the NoC, and where communication between these is performed using asynchronous request-acknowledge based communication protocols [24]. In both the mesochronous and the GALS-approach some form of synchronization is needed when signals enter a clock-domain (IP-block, NA or router).

An extension of the GALS approach is to implement the *entire NoC as an asynchronous circuit* [24], [7], [25]. In this way no clock domain extends across the entire chip, and synchronization is only required when data enters the clock-domain of an IP-core. Other advantages of an asynchronous NoC is that its dynamic power consumption when idle is zero, and that its implementation can be made insensitive to gate and wire delays, which according to the International Technology Roadmap for Semiconductors are becoming increasingly difficult to estimate and model. The latter may result in faster circuits and hence higher bandwidth.

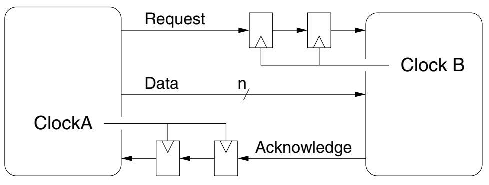

Fig. 2. Transfer of data between two clock domains requires a protocol and synchronization of the control signals–here Request and Acknowledge.

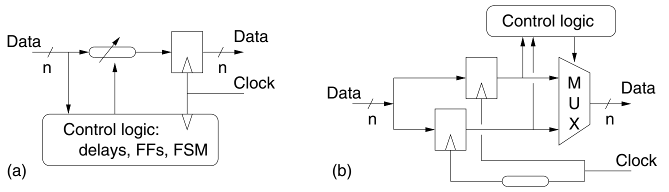

Fig. 3. Mesochronous synchronizers delay input data or select one of two clock phases for sampling the input data. More details may be found in [15].

IV. Synchronization circuits

A. Synchronizing asynchronous inputs

When communicating data from one clock domain to another, some kind of protocol is required along with circuitry to synchronize the signals implementing the protocol. Figure 2 shows a possible implementation for the most general situation where nothing is known about the clock signals in the two clock domains. If a dual-flip-flop-synchronizer does not provide a sufficiently low failure rate, additional flip-flops may be added. If the circuit in one domain is implemented using asynchronous techniques, then signals into this domain need not be synchronized. The overhead in terms of area and power is limited, but the overhead in terms of latency is considerable (i.e. one or more clock cycles for each protocol signal event which needs to be synchronized). The interface shown in figure 2 is thus capable of transferring one data word for every 2-4 clock cycles. By synchronizing blocks of data it is possible to amortize the latency across several data items, but area and power will increase.

B. Synchronizing mesochronous inputs

A synchronizer for a mesochronous input can do one of the following: delay the local clock such that its phase "fits" the timing of the input data, or delay the input data to "fit" the local clock. Both solutions require several variable delay elements. Two basic designs from [15] are shown in figure 3. Since a router node for a NoC communicate with several other modules (routers or NAs) the solution which adjusts the delay of the clock is not possible. As an alterative to using analogue delay elements we mention that by using multiplexers and registers clocked on the up-going edge of the clock and registers clocked on the down-going edge of the clock signal is possible to delay the data in increments of a half clock period. In any case the overhead in terms of latency is modest, but the overhead in terms of area and power is considerable. Furthermore some initialization is necessary to adjust the delays.

C. Discussion

The bottom line is that synchronization is expensive–for asynchronous inputs mostly in terms of latency, and for mesochronous inputs mostly in terms of power. As both are critical for a NoC implementation, this observation suggests that the most realistic solutions are a globally synchronous NoC or a NoC built using asynchronous circuitry. The next section addresses the latter.

V. Asynchronous NoCs designed at DTU

A. Introduction

In addition to the above mentioned advantage of avoiding synchronization overhead, asynchronous implementations of routers and links offer a number of additional benefits: (1) Since they make use of self-timed and data-driven control they always operate at the maximum speed possible, and they can be pipelined aggressively if this is needed. (2) The latency of an end-to-end connection is minimized, partly because of the fewer synchronizers and partly because the connection behaves like a fall-through FIFO, whereas a clocked NoC behaves like a clocked shift register. (3) An asynchronous NoC has zero dynamic power consumption in those parts of the NoC that are idle. In real-life applications where communication requirements fluctuate a lot, this leads to lower power consumption. (4) Flow-control, which needs to be explicitly implemented in clocked NoCs, is inherent in asynchronous implementations. The handshaking overhead associated with asynchronous design is therefore less significant in a NOC implementation, than it is in other applications.

Finally we note that an asynchronous router will need a to arbitrate among flits contending for the same output port or output buffer. This may cause metastability in the arbiter and hence added latency, but only when a conflict actually occurs, and in these situations the latency is limited to the time actually taken to recover from metastability. On average the added latency is therefore negligible, which is in contrast to the worst case latency of a clock domain synchronizer.

At DTU we have experimented with different asynchronous NoC applications. We have implemented an ultra-low power NoC for an audio DSP application, and a full-featured general-purpose NoC for large SoCs. We have used a data-flow design style [24, Chapter 3] using handshake components like latch, mux, demux, merge etc. and a mutex and a priority arbiter. Our experience is that this design style, supplemented by special speed-independent control circuits designed using Petrify [26], fits nicely with the type of functionality which is required to implement a NoC. In the following sections the two NoC designs are briefly reviewed.

B. The audio DSP NoC

The application we considered is an industrial audio DSP subsystem in which a number of filters and other processing blocks are connected into some static topology where data is streamed across communication channels connecting the blocks. Some restricted form of multicast is also used. In order to allow the same chip to be used in different applications

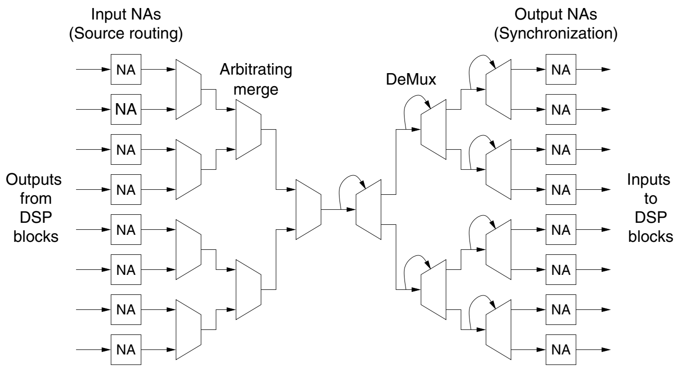

Fig. 4. Block diagram showing the fundamentals of the NoC for the audio DSP subsystem. More details may be found in [13].

and/or in order to be able to support several use cases in a given application, the original design is based on one big crossbar switch implemented in the following way: Every block broadcasts its output to the input of every other block which may need the result. Multiplexers on the inputs of the blocks then selects the proper input and the multiplexers are controlled by data stored in a set of control registers. This solution works fine for a moderate number of blocks but it does not scale well.

The aim of the NoC design was to reduce the total wiring in order to save power, and to provide a solution which scales better with the number of blocks. The fundamental ideas of the implementation are shown in figure 4. It consists of some very simple network adapters, a tree structure of arbitrating merge components and a tree structure of controlled switches (i.e., demultiplexers). The network use source routing and a packet header contain the control signals used when a packet travels through the demultiplexer tree. These routes are (pre)programmed into the source NAs using the same type of configuration registers as used in the original design. The actual implementation has a few added features including the option to add data from several sources.

The asynchronous NoC was implemented using standard cells in 0.130 micron technology and the audio DSP subsystem was simulated with crossbar replaced by the NoC. Since the NoC is asynchronous and since the DSP subsystem is clocked using the same global clock, the design can be viewed as a globally-synchronous locally-asynchronous design. Despite the potential bottleneck where the roots of the two trees are connected the NoC supports more than 10 data transfers in one clock period, without any need for generating this higher frequency clock. More details are available in [13]. Should more bandwidth be needed, a different NoC topology offering a higher bisection bandwidth may easily be implemented using the same merge and switch components.

C. The MANGO NoC

MANGO is an acronym: Message-passing, Asynchronous, Network-on-chip, providing Guaranteed services, over OCP-interfaces. The MANGO NoC was developed to support the

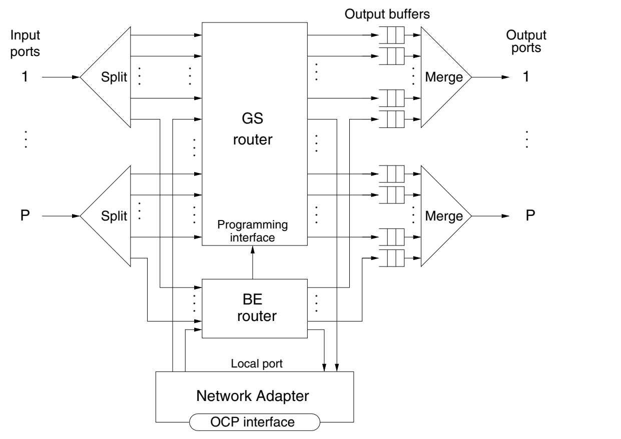

Fig. 5. Block Diagram of a MANGO router. For a 2D-mesh topology P=4.

communication requirements of larger SoCs. It provides both connection-based GS-communication as well as connection-less BE-communication, a routing node includes both a GS-router and a BE-router, figure 5. The links are multiplexed in order to provide a number of virtual channels between neighboring nodes [8], [10]. A virtual channel connects to either the GS router or the BE router. The routers use output buffers only and flow control is established from one output to the relevant output buffer in the next router. In this way the flow control spans the link and the switch in the next router. The design is significantly more complex than the audio DSP NoC, and the details are beyond the scope of this paper. The interested reader is referred to [7], [8], [9], [10], [11], [12], [13]. A prototype router for a 2D-mesh topology (c.f., figure 1) has been designed and implemented using standard cells in a 0.130 micron technology. The combined GS/BE router has 5 ports, each with 8 virtual channels per link and it uses flits with 32 bit data. Its area is 0.188 mm^2 and its speed is 795 M flits/sec per link (typical) and 515 M flits/sec per link (worst case).

VI. Conclusion

The paper introduced the NoC-concept and the possible timing organizations for a NoC based SoC: globally-synchronous, mesochronous, GALS and asynchronous. Following this the paper presented the synchronizer circuits which are required when implementing SoCs based on these timing organizations. Based on this the paper argued for implementing the NoC using asynchronous circuit techniques, and the paper presented two asynchronous NoCs designed for two different applications. The paper has been written to support an invited talk at the NORCHIP'2007 conference.

References

[1] William Dally. Route packets, not wires: On-Chip interconnection networks. In *Proceedings of the 2001 Design Automation Conference (DAC-01)*, pages 684–689, New York, June 2001. ACM Press.

[2] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *Computer*, 35(1):70–78, January 2002.

[3] A. Jantsch and H. Tenhunen, editors. *Networks on Chip*. Kluwer Academic Publishers, 2003. ISBN 1-4020-7392-5.

[4] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Elsevier Science Publishers, 2003.

[5] G. De Micheli and L. Benini (Editors). *Networks on Chips: Technology and Tools*. Morgan Kaufmann Publishers, 2006.

[6] T. Bjerregaard and S. Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys*, 38(1):1–51, 2006.

[7] 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)*, pages 1226–1231. IEEE Computer Society Press, 2005.

[8] T. Bjerregaard and J. Sparsø. Virtual channel designs for guaranteeing bandwith in asynchronous network-on-chip. In *22nd Norchip Conference*, pages 269–272. IEEE, 2004.

[9] T. Bjerregaard and J. Sparsø. A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-chip. In *Proc. International Symposium on Asynchronous Circuits and Systems*, pages 34–43. IEEE Computer Society Press, 2005. (Best paper award).

[10] T. Bjerregaard and J. Sparsø. Implementation of Guaranteed Services in the MANGO Clockless Network-on-Chip. *IEE Proceedings: Computing and Digital Techniques*, 153(4):217–229, 2006.

[11] T. Bjerregaard, S. Mahadevan, R. G. Olsen, and J. Sparsø. An OCP compliant network adapter for GALS-based SoC design using the MANGO network-on-chip. In *Proceedings of the International Symposium on System-on-Chip (SoC'05)*, pages 171–174. IEEE, November 2005.

[12] T. Bjerregaard and J. Sparsø. Packetizing OCP Transactions in the MANGO Network-on-Chip. In *9th EUROMICRO Conference on Digital System Design (DSD'06)*. IEEE Computer Society Press, pp. 657-664 2006.

[13] M. B. Stensgaard, T. Bjerregaard, J. Sparsø, and J. H. Pedersen. A simple clockless Network-on-Chip for a commercial audio DSP chip. In *9th EUROMICRO Conference on Digital System Design (DSD'06)*. IEEE Computer Society Press, pp. 641-648 2006.

[14] Open Core Protocol (OCP) Specification, Release 2.0, 2003. http://www.ocpip.org.

[15] W. J. Dally and J. W. Poulton. *Digital Systems Engineering*. Cambridge University Press, 1998.

[16] M. Dallosso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini. Xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs. In *Proc. International Conf. Computer Design (ICCD)*, pages 536–539. IEEE Computer Society Press, 2003.

[17] Kees Goossens, John Dielissen, and Andrei Rădulescu. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design and Test of Computers*, 22(5):414–421, Sept-Oct 2005.

[18] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. The Nostrum backbone - a communication protocol stack for networks on chip. In *Proceedings of the VLSI Design Conference*, pages 693–696, Mumbai, India, January 2004.

[19] The International Technology Roadmap for Semiconductors. ITRS 2005 Edition. http://www.itrs.net/, 2005.

[20] B. Mesgarzadeh, C. Svensson, and A. Alvandpour. A new mesochronous clocking scheme for synchronization in SoC. In *Proc. Int'l. Symp. Circuits and Systems*, pages II.605–II.608, 2004.

[21] F. Mu and C. Svensson. Self-tested self-synchronization circuit for mesochronous clocking. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 48(2):129–140, 2001.

[22] Daniel M. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems*. PhD thesis, Stanford University, October 1984.

[23] Jens Muttersbach, Thomas Villiger, and Wolfgang Fichtner. Practical design of globally-asynchronous locally-synchronous systems. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 52–59, April 2000.

[24] J. Sparsø. Asynchronous circuit design – a tutorial. In J. Sparsø and S. Furber, editors, *Principles of asynchronous circuit design – A systems perspective*, chapter 1-8, pages 1–152. Kluwer Academic Publishers, 2001. 337 pages.

[25] R. Dobkin, V. Vishnyakov, E. Friedman, and R.Ginosar. An asynchronous router for multiple service levels networks on chip. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 44–53. IEEE Computer Society Press, 2005.

[26] Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, and Alexandre Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. In *XI Conference on Design of Integrated Circuits and Systems*, Barcelona, November 1996.