A Mesochronous Network-on-Chip for an FPGA

Stephan Kubisch, Enrico Heinrich, and Dirk Timmermann

University of Rostock, Department of CS and EE
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany
{stephan.kubisch;enrico.heinrich;dirk.timmermann}@uni-rostock.de
http://www.networks-on-chip.com

Abstract. Networks-on-Chip (NoCs) are a new paradigm for the design of integrated systems. NoCs address design complexity and physical issues in ASIC and FPGA designs. In this paper, we present a scalable, mesochronous NoC architecture for an FPGA. A special hybrid switching scheme has been developed for globally asynchronous locally synchronous (GALS) operation of the whole NoC-based system. The NoC was designed with respect to simplicity and a small hardware footprint.

1 Introduction

The increasing design-productivity-gap demands for new approaches for the design of VLSI systems since current design flows cannot exploit the complexity of todays Systems-on-Chip (SoCs). Thus, Networks-on-Chip (NoCs) have been proposed [1,2] as new and modern hardware concept for on-chip interconnection and communication. NoCs enforce a modular design concept, reusability, and mitigate parasitic effects by splitting up global wires. NoCs have mainly been used in ASIC designs addressing physical issues. But due to increasing design complexity, growing FPGA device sizes, and the wide-spread use of field programmable devices, NoCs are also suitable in FPGA designs as addressed in this paper.

The communication within a NoC-based system is based on messages transmitted between functional units—so-called IP cores. While the message-based communication is already asynchronous in itself, the regularity of NoC infrastructures also allows for a globally asynchronous and locally synchronous (GALS) design style on the clock level. In [3] and [4], asynchronous NoCs for CMOS technologies are presented. But in contrast to ASICs, current FPGAs already incorporate a broad set of clocking resources like multiple clock networks, clock drivers, and phase locked loops (PLLs). Current structures of FPGA clock networks [5] are similar to the regular structure of NoCs. Thus, these features can be exploited for NoC-based GALS designs in FPGAs.

In this paper, we present a mesochronous NoC infrastructure for a Xilinx Virtex-4 FPGA as well as a hybrid switching mechanism for mesochronous operation of the NoC. Both are enhancements of a synchronous NoC version [6]. A mesochronous design operates with the same clock frequency but the clocks of adjacent clock regions need not to be phase-aligned. The main goal

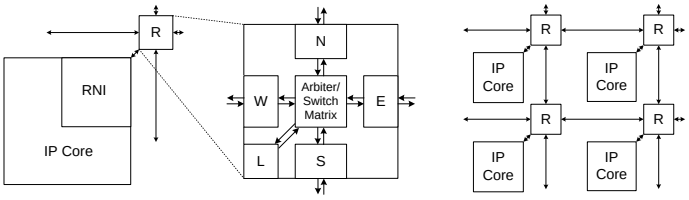


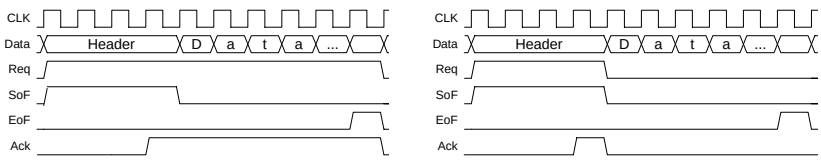
Fig. 1. Structure of a NoC router in a 2D mesh topology

is that the NoC's maximum operation frequency and therewith the maximum link bandwidth—contrary to the synchronous NoC version—depend only on the complexity of a single NoC router and remain stable even with increasing NoC size. The NoC is used as logical overlay on the FPGA's physical routing fabric since hard-wired NoC resources, as proposed in [7], are not yet available in current FPGAs. Instead, valuable FPGA resources must be used for the NoC part of the system. Simplicity and a moderate hardware footprint for the NoC itself are thus mandatory to keep the limited FPGA resources for the application.

2 The Network-on-Chip

An NoC basically consists of routers and independent links between the routers. Unlike buses, NoCs allow for concurrent transmission of data between modular IP cores. By means of separation of computation and communication, designers benefit from a divide-and-conquer approach when partitioning their applications. Figure 1 shows a NoC's basal elements: routers (R) and connected IP cores. Each router connects to up to 4 neighboring routers. In this case, a 2D mesh topology is realized. An IP core can be, e.g., external I/O, memory, or a processing unit. The resource-network-interface (RNI) provides bridging functionality from a router to an IP core and adapts to the IP core's specific interface. A router consists of five ports; one for each routing direction (N, S, W, E) and one port for access to the RNI (L). Each port contains a buffer and the routing logic. The arbitration and switching functionality connects inputs and outputs by configuring a switch matrix. Each channel between two routers consists of two independent unidirectional links indicated with arrows.

For a low hardware footprint, only necessary and simple mechanisms have been implemented in the NoC [6]. XY-routing is used. It is minimal, deterministic, and deadlock-free. Different versions of the NoC use either wormhole (WHS) or virtual-cut-through switching (VCTS). A message is thereby divided into 32-bit wide flow control digits (flits), which is the smallest information unit in the NoC. The header flit contains address information and defines the route in the NoC. The data flits follow the header flit. This way, the complete message is transmitted as a "worm" through the NoC and occupies the routers' ports and buffers along its path. A message can only be stopped at the header flit. The



(a) WHS (b) VCTS

Fig. 2. Switching mechanisms in the synchronous NoC versions

signaling for WHS and for VCTS is shown in Figure 2. Routing and arbitration require at least 4 clock cycles to configure the router. Start and end of a message are indicated with Start-of-Frame (SoF) and End-of-Frame (EoF). Valid data is indicated with a request signal (Req). For WHS (Figure 2(a)), each port contains a buffer for one flit. With a combinational acknowledge signal (Ack), one flit can be transmitted each clock cycle. With a registered Ack, the transmission of one flit would take at least 2 cycles, which would halve the maximum link bandwidth. If the header flit is blocked, the message waits like a worm in the NoC. That is why every flit needs to be acknowledged. For VCTS (Figure 2(a)), each port contains buffer space for one maximum transmission unit (MTU). If the header is blocked, previously occupied ports and buffers will not remain occupied since the whole message can be stored in the last port's buffer. Otherwise VCTS behaves like WHS except that only the header flit needs to be acknowledged. For detailed information on the synchronous NoC, we refer to [6].

3 Modifications for a Mesochronous NoC

GALS architectures circumvent and alleviate technology and scaling problems. Parasitic effects and long wire delays are reduced by splitting up global wires [8]. Multiple clock domains exist, each with an own clock. When designing GALS systems, the application's functionality is typically partitioned into independent and parallel tasks, which are implemented as concurrent blocks. Performance increases by decentralization and parallelization, modular blocks enforce reuse of existing IP cores, and each IP core can be driven with a matched clock frequency to either reduce idle times or to speed up bottleneck functions. In the proposed mesochronous NoC, each router uses an independent clock, which is derived from a global clock. Thus, the clock frequencies are the same but the clocks' phases may differ. Clock domain crossing (CDC) mechanisms are required to handle metastability and timing issues between adjoining clock regions. The following resources of the Virtex-4 FPGA's range of features have been used for the mesochronous NoC:

– Registers are used for simple CDC circuits between adjacent router ports [9]. Figure 3 shows the basic structure of this circuit for 1-bit signals between router R_0 and R_1. Two D-type flip-flops (FFs) are serially linked in each

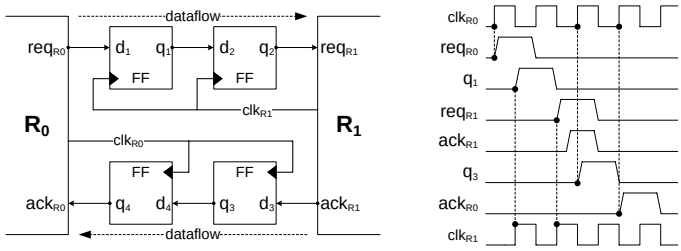


Fig. 3. CDC circuit using serially connected flip-flops and timing diagram for a simple req&ack-example

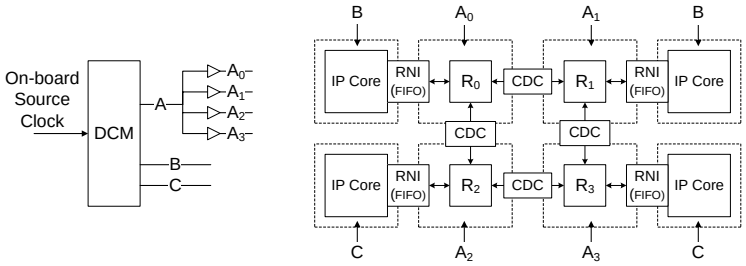


Fig. 4. Clock domain (dotted lines) in the asynchronous NoC-based system

direction. The FFs are clocked by the reading clock domain to synchronize incoming data signals. A simple request-acknowledge-protocol is demonstrated in the figure and the timing diagram. At least 3 clock cycles are required for R_0 to receive an acknowledge for a request from R_1 .

- A hard-wired Digital Clock Manager (DCM) block, which provides features for clock deskew, frequency synthesis, and phase shifting, generates clocks with arbitrary frequencies out of an on-board source clock. These clocks are used for the NoC infrastructure and the IP cores as shown in Figure 4.
- For each router, a regional clock buffer and a regional clock network are used. These clocks are all derived from the same clock generated in the DCM as sketched in Figure 4. But they are not phase-aligned. The available number of clock networks of the FPGA limits the number of clock regions.
- Hard-wired first-in-first-out buffers (FIFOs) with independent read and write ports are used in the RNIs to support *different* clock frequencies for the IP cores. The NoC itself is mesochronous but the whole NoC-based system operates asynchronously. Figure 4 shows the clock domains of an example 2x2 NoC system. The clock domains are indicated with dotted lines. The DCM generates 3 different clocks (A–C). The routers (R_0–R_3) are driven with regional clocks (A_0–A_3) derived from clock A. CDC circuits are used between the routers. The IP cores are driven with clock B and C.

3.1 The Hybrid Switching Mechanism

The switching schemes WHS and VCTS (Figure 2) are not feasible any more when using the CDC circuit shown in Figure 3. This is because the FFs generate a delay of multiple clock cycles in both directions. Thus, a special hybrid switching mechanism (HSM) was developed. The transmission of a message with HSM is divided into two phases—*setup phase* and *fast-transmit phase*. HSM thereby bases on the following assumptions:

1. XY-routing or a similar *static* routing algorithm is used. This way, the remaining number of hops a flit still needs to take to the destination is known in each router. Thus, HSM is directly coupled with the routing algorithm.
2. To save resources and to keep a moderate hardware footprint, only the RNIs contain large FIFO buffers to buffer at least one complete MTU. Whereas each router port provides a small buffer for only one flit.

Setup phase: The setup phase lasts until the header flit arrives at the destination. The name already expresses the main function of that phase—to allocate a path to the destination. Each flit is acknowledged during the setup phase because, as explained in Section 2, the header flit can be blocked on its route to the destination. Therefore, each router port maintains a counter, which counts the flits already transmitted for the current message. When a header flit arrives at a router port, this port's counter is set to N. N is the number of hops the header flit still has to take from the current router to the destination router. Because of two-dimensional coordinate-based addresses and static XY-routing, this is the simple arithmetic function (1). Thereby, the maximum of N depends on the NoC dimensions (2).

Fast-transmit phase: During forwarding of the data flits, the ports' counters are decreased with every flit. When a counters reaches 0, this router port switches to fast-transmit. Now, data is just forwarded without being acknowledged since the header has reached the destination and the message cannot be blocked any more. The FIFO buffer of the destination RNI offers sufficient space to store the whole message. A toggle-signal (VDT) is carried along each link. In the RNI, it indicates new and valid data flits during this phase. In the best case, one flit can be transmitted each clock cycle. That is the point of time when all routers on the occupied path through the NoC have switched into fast-transmit mode. The last data flit is signalized with EoF to complete the current transmission and to deallocate the occupied router ports.

$$N = |x_{router} - x_{dest}| + |y_{router} - y_{dest}| \tag{1}$$

$$N_{max} = (x_{dim,max} - 1) + (y_{dim,max} - 1) \tag{2}$$

As an example for HSM, the complete transmission of a 7-flit message between the routers R_0 and R_1 is illustrated in Figure 5. For a better understanding, the same indices are used as in Figure 3. The upper part indicates signals of R_0's clock domain, the middle part shows the intermediate signals of the CDC-circuits, and the lower part depicts the signals in R_1's clock domain. Important steps during the transmission are highlighted with numbers and explained in the following:

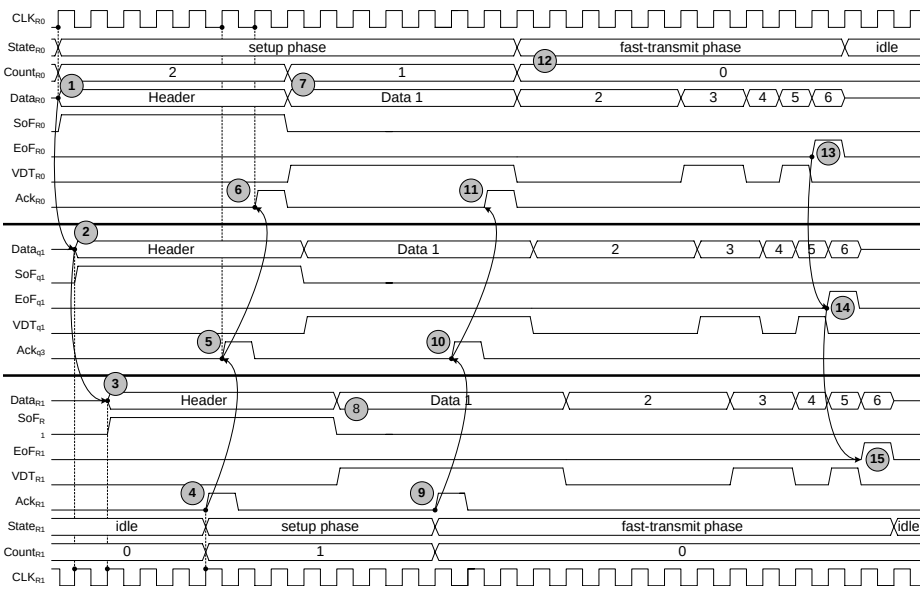


Fig. 5. Example transmission of a 7-flit message using the hybrid switching mechanism (compare indices with Figure 3)

Steps 1-3: R_0 itself has just processed a new header flit and forwards this header flit ($Data_{R_0}$) towards R_1. R_0 switches into setup phase and sets counter $Count_{R_0}$ to $N = 2$. The header flit is synchronized into R_1's clock domain.

Steps 4-6: After completing its internal routing and arbitration procedures, R_1 also switches to setup-phase and sets its counter $Count_{R_1}$ to $N = 1$. R_1 sets Ack_{R_1}, which is clocked into R_0's clock domain. During the setup-phase, a flit is delayed with at least 7 clock cycles needed for routing & arbitration, setting the acknowledge signal, and synchronization delay.

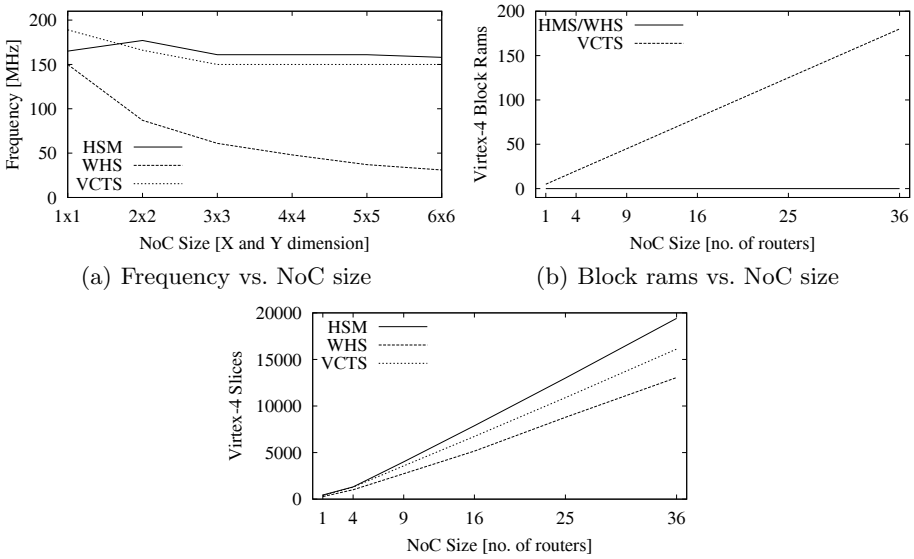
Step 7: R_0 receives the acknowledge (Ack_{R_0}). Then, the first data flit is assigned towards R_1, SoF_{R_0} is unset, and $Count_{R_0}$ is decremented.

Step 8: The data flit arrives at R_1 ($Data_{R_1}$). Now, both routers are in the setup-phase and wait for the acknowledge from the next router behind R_1.

Steps 9-11: R_1 receives the acknowledge and forwards it to R_0 using Ack_{R_1}, which is again clocked into R_0's clock domain. $Count_{R_1}$ is also decremented. Because $Count_{R_1}$ reaches zero, R_1 switches to fast-transmit mode. Data is now just forwarded without being acknowledged. The destination RNI uses the VDT signal, which is carried along the complete path, to identify new and valid flits.

Step 12: R_0 assigns the second data flit to $Data_{R_0}$. R_0 also goes into fast-transmit mode because $Count_{R_0}$ reaches zero.

Steps 13-15: The EoF flag indicates the last flit of the message and the end of this transmission. The router ports are deallocated. R_0 and R_1 switch into idle states and wait for a new header flit.



(a) Frequency vs. NoC size

(b) Block rams vs. NoC size

(c) Slices vs. NoC size

Fig. 6. Synthesis results for three different NoC versions

The performance of HSM is comparable with the synchronous schemes WHS and VCTS (see [6]). The difference is in the setup phase, where at least 3 additional clock cycles are needed per flit due to synchronization of the backpressure signal *Ack* using the CDC circuit. But during fast-transmit, 1 flit is transmitted per clock cycle. Thus, HSM is nearly as fast as WHS/VCTS for longer messages where the fast-transmit phase lasts much longer compared to the setup phase.

4 Synthesis Results

NoCs with different sizes have been synthesized for a Xilinx Virtex-4 FX100 FPGA. The RNIs are not included since they are the same in each NoC version Figure 6 shows diagrams for various synthesis parameters. Figure 6(a) displays the frequency in dependence of the NoC size (given in X&Y dimensions). Figure 6(b) and 6(c) show the number of used block rams (BR) and slices in dependence of the number of routers.

The VCTS version shows a stable operation frequency around 150 MHz for different NoC sizes. This is due to the use of FIFO primitives with independently clocked read and write ports, which are used for clock domain crossing. But the drawback of the VCTS version is its large hardware footprint. Since one FIFO is required in each router port and each router has 5 ports, a simple 4x4 NoC for example already consumes 80 BRs. But on-chip memory is a valuable resource. In contrast, the WHS version does not need any FIFO buffers at all

and has the lowest hardware footprint regarding the slices. But the frequency decreases with increasing NoC size. It does not scale. However, HSM combines the advantages of both NoC versions. The maximum frequency scales with increasing NoC dimensions. With 160 MHz, it is even slightly higher than for VCTS. Besides, the NoC infrastrucutre for HSM does *not* require any BRs. But due to the CDC circuits and additional flow control logic in the router ports, the number of slices is higher for HSM. This is an acceptable compromise since the latest FPGAs provide abundant slices and registers and will provide even more in the future.

5 Conclusion

The paper presented a mesochronous NoC for an FPGA as alternative architectural approach compared to conventional pipelined systems and synchronous buses. The FPGAs inherent clocking facilities have therefore been exploited to realize this flavor of GALS operation. A special hybrid switching scheme has been presented, which is tightly coupled with the chosen XY-routing algorithm. The mesochronous NoC's performance characteristics, especially its maximum operation frequency, are independent of the size of the NoC. Besides, IP cores can be driven with completely different clock frequencies, which results in a fully asynchronous NoC-based system. To save valuable FPGA resources for the functional part of an FPGA design, the NoC has a reasonable hardware footprint since only simple mechanism are used within the NoC infrastructure.

References

1. Dally, W.J., Towles, B.: Route Packets, not Wires: on-chip Interconnection Networks. In: Proc. of the 38th DAC, Las Vegas, NV, USA (2001)
2. Benini, L., de Micheli, G.: Networks on Chips: A New SoC Paradigm. IEEE Computer **35**(1) (2002) 70–78
3. Bjerregaard, T., Stensgaard, M.B., Sparsø, J.: A Scalable, Timing-safe, Network-on-chip Architecture with an Integrated Clock Distribution Method. In: Proc. of the Conf. on Design, Automation and Test in Europe, Nice, France (2007)
4. Wiklund, D.: Mesochronous Clocking and Communication in on-chip Networks. In: Proc. of the Swedish System-on-Chip Conf., Eskilstuna, Sweden (2003)
5. Lamoureux, J., Wilton, S.J.E.: FPGA Clock Network Architecture: Flexibility vs. Area and Power. In: Proc. of the 14th Symp. on FPGAs, Monterey, CA, USA (2006)
6. Kubisch, S., Cornelius, C., Hecht, R., Timmermann, D.: Mapping a pipelined Data Path onto a Network-on-Chip. In: Proc. of the 2nd Symp. on Industrial Embedded Systems, Lisbon, Portugal (2007)
7. Hecht, R., Kubisch, S., Herrholtz, A., Timmermann, D.: Dynamic Reconfiguration with hardwired Networks-on-Chip on future FPGAs. In: Proc. of the 15th Conf. on Field Programmable Logic, Tampere, Finland (2005)
8. Jia, X., Vemuri, R.: Using GALS Architecture to reduce the Impact of long Wire Delay on FPGA Performance. In: Proc. of the 2005 Asia and South Pacific Design Automation Conf., Shanghai, China (2005)
9. Stein, M.: Crossing the Abyss: asynchronous Signals in a synchronous World. EDN Magazine **July** (2003) 59–69