2013 16th Euromicro Conference on Digital System Design

Router Designs for an Asynchronous Time-Division-Multiplexed Network-on-Chip

Evangelia Kasapaki, Jens Sparsø, Rasmus Bo Sørensen
DTU Compute
Technical University of Denmark
Email: evka@dtu.dk, jspa@dtu.dk, rboso@dtu.dk

Kees Goossens
Faculty of Electrical Engineering
Eindhoven University of Technology
Email: k.g.w.goossens@tue.nl

***Abstract*—In this paper we explore the design of an asynchronous router for a time-division-multiplexed (TDM) network-on-chip (NOC) that is being developed for a multi-processor platform for hard real-time systems.**

TDM inherently requires a common time reference, and existing TDM-based NOC designs are either synchronous or mesochronous, but both approaches have their limitations: a globally synchronous NOC is no longer feasible in today's sub-micron technologies and a mesochronous NOC requires special FIFO-based synchronizers in all input ports of all routers in order to accommodate for clock phase differences. This adds hardware complexity and increases area and power consumption.

We propose to use asynchronous routers in order to achieve a simpler, more robust and globally-asynchronous NOC, and this represents an unexplored point in the design space.

The paper presents a range of alternative router designs. All routers have been synthesized for a 65 nm CMOS technology, and the paper reports post-layout figures for area, speed and energy and compares the asynchronous designs with an existing mesochronous clocked router. The results show that an asynchronous router is 2 times smaller, marginally slower and with roughly the same energy consumption, while offering a robust solution to the clock distribution problem. The paper further explores "clock-gating" of the individual pipeline stages in the asynchronous routers, and shows that this can lead to significant power savings.

I. Introduction

Today the preferred interconnect used in general-purpose chip multi-processors (CMP) and in multi-processor systems-on-chip (MPSOC) is some form of packet switched network-on-chip (NOC) and "networks-on-chip" has been an active area of research for more than a decade. The choice of NOC for a given platform depends on many factors and requirements including: bandwidth requirements, types of traffic (streaming or address-space read/write transactions), provision of service guarantees, clocking and timing issues.

The context for the work presented in this paper, is a project in which we are developing a *general-purpose* multi-processor platform that is intended specifically for use in *hard real-time* systems [1].

A generic illustration of a NOC-based multi-processor platform is shown in Fig. 1(a). It consists of a set of so-called IP-cores communicating through a packet switched structure of routers and links. Typical IP-cores are processors with some amount of local memory, blocks of shared memory or IO-devices, and typical NOC topologies are planar structures like meshes and tori. Each IP-core is connected to the network through a network interface (NI) that translates (core-specific) read/write transactions into (NOC specific) packets. Moreover, the NIs typically implement clock domain crossings.

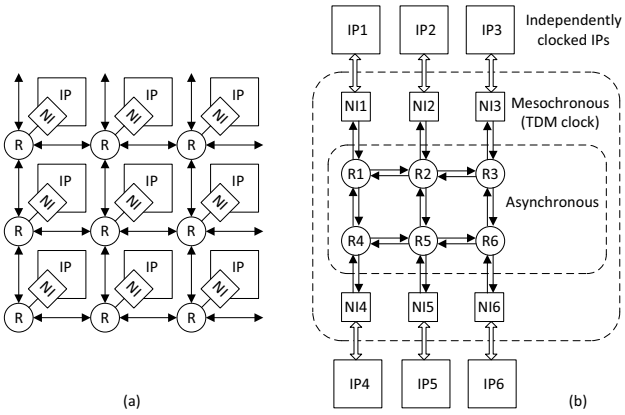

Figure 1. (a) Structural organization of a generic NOC-based multi-core platform. (b) Timing organization of a platform using the asynchronous time-division-multiplexed NOC explored in this paper.

Most published NOCs only support best-effort traffic and some published NOCs offer multiple priority-levels in order to support different qualities of service. Such NOCs are inadequate for hard real-time systems. In order to provide the time predictability that is needed to guarantee worst case execution time (WCET), a NOC for a hard real-time platform must provide some form of end-to-end circuits. This can be FPGA-style physical connections or virtual circuits implemented using time-division-multiplexing (e.g. Æthereal [2]) or using non-blocking routers combined with rate control [3] (e.g. Mango [4]).

Furthermore, in order to be practically viable in today's sub-micron semiconductor technologies, some form of globally-asynchronous locally-synchronous (GALS) organization of the hardware platform is a requirement [5]. Local islands of

`/13 $26.00 © 2013 IEEE
DOI 10.1109/DSD.2013.40

circuitry like IP-cores and NIs, Fig.1(a), can be implemented synchronously. But because the NOC spans the entire chip is is not possible to distribute a clock with zero skew to all routers. Solutions to this include the use of mesochronous routers or fully asynchronous routers.

A more detailed discussion of issues related to real-time systems and implementation of GALS-systems is provided in Section II. Based on this discussion we have decided for a statically scheduled TDM-based NOC, to deal with hard real-time requirements, using asynchronous routers, to face the clock distribution problem. The overall timing architecture of the platform is illustrated in Fig.1(b) – a GALS-style design using asynchronous routers, mesochronous NIs and (possibly) independently clocked IP-cores.

Two main decisions form the basis for our NOC. Firstly, we decided for a TDM-based NOC because it avoids arbitration, flow-control and buffering thereby reducing the network of routers and links to the bare minimum – a pipelined structure of multiplexers, wire segments and registers. Secondly, we decided for an asynchronous router implementation because it avoids the FIFO-synchronizers used in mesochronous designs. Asynchronous circuits are normally larger than their asynchronous counterparts, but in this case the asynchronous router is smaller than the corresponding clocked mesochronous router.

The use of asynchronous routers in a TDM-based NOC represents a point in the design space that has not been explored before, and the contribution of this paper is a thorough exploration of a range of such router designs. Additionally, tha paper contributes with the application of the idea of gating in asynchronous designs and studying the effect of it. The designs have been implemented in 65 nm CMOS standard cell technology and the paper reports post-layout results on area, speed and energy consumption. The paper achieves asynchronous designs that are smaller in area than alternative clocked solutions.

The paper is organized as follows. In Section II we discuss related work and provide some necessary background on NOCs for real-time systems and implementation of GALS systems. Section III presents the overall architecture of our NOC-based multi-processor platform. Section IV describes the design of the baseline synchronous router design (aelite [6]), and a mesochronous implementation of the same. Section V presents a number of asynchronous router designs: (i) a straightforward four-phase bundled-data design, (ii) a two-phase bundled-data design, (iii) a 2-phase bundled-data that implements "clock gating" to avoid switching activity in unused router ports, and (iv) a 2-phase bundled-data design with "clock gating" that uses delay-insensitive signaling on the links. Section VII explains the implementation and EDA-tool flow. Section VII provides results for the four designs, and finally Section VIII concludes the paper.

II. Background and Related Work

This section reviews related work in the area of NOCs for real-time systems and GALS timing organization

A. Real-Time NOCs

A NOC for a real-time platform must allow guarantees on bandwidth and latency to be made for individual point-to-point transactions. This calls for a solution that provides some form of end-to-end connection; physical or virtual.

Examples of NOCs offering physical connections is the NOC used in the 4S-platform [7] that offers initialization-time FPGA-style configurable connections and SoCBUS [8] that implements a dial-up mechanism. In both cases connections, when established, own resources exclusively, so real-time guarantees are easily provided. The downside is a (potential) low utilization of resources. Furthermore, in SoCBUS a dial-up attempt cannot be guaranteed to succeed.

The use of virtual channels allows sharing of resources. Virtual channels can be implemented in two fundamentally different ways. One approach is to use TDM where the resources (routers and links) are used in a time-multiplexed fashion according to a static schedule. Examples are Æthereal [9], aelite [10], Nostrum [11] and TTNoC [12]. The other approach is to use non-blocking routers with rate control [3]. An example of the latter is the asynchronous MANGO NOC [4]. In MANGO several connections may share a link but each connection has a private (virtual-channel) buffer in every router along the connection. Additional hardware complexity is caused by the larger crossbar that follows from the use of virtual channel buffers and from the arbitration and flow control that is needed in every output port. It is interesting to observe that a typical MANGO router is 10 times larger than a aelite-router (the smallest of the TDM-based routers) [2], [13]. Based on this observation, we decided to explore the design of an asynchronous aelite-style TDM-based router.

B. GALS Timing Organization

Distributing a clock signal across a chip and achieving timing closure throughout the whole chip is becoming increasingly difficult due to parameter uncertainty and variability [5]. A GALS-style architecture reduces these problems by dividing the design into several independent clock domains and by implementing some form of asynchronous communication among these. A NOC-based multi-core platform naturally supports a timing organization where the IP-cores and the NIs constitute separate clock domains, and where the packet-switched structure of routers and links, which spans the entire chip, is implemented using techniques that are more robust to timing uncertainties.

A mesochronous NOC represents a first step away from a globally-synchronous design towards a more robust timing architecture. There are numerous solutions in the field of mesochronous communication [14, Sec. 10.3.1]. Mesochronous considers same clock frequency, but a phase difference is allowed. The choice also depends on the precise meaning of mesochronous: is the clock phase difference assumed to be constant or is it allowed to drift within some bounded interval. (Our design allows the latter).

The preferred solution in most of the published work on mesochronous NOCs use small FIFOs. The bi-synchronous

FIFO presented in [15] has been used in DSPIN [16] and aelite [2]. It consists of 5 clocked registers and some control logic. Other works on aelite report using a more area-efficient full-custom FIFO presented [17]. This design is an asynchronous pipeline using latches instead of flip-flops, and in [10, Ch. 8] its area is reported to be half of the area of the bi-synchronous FIFO. A later paper [18] presents a FIFO that is optimized specifically for mesochronous systems. The NOC described in [19] merges this new FIFO into the virtual-channel buffers of the router. As TDM-based router does not have such buffers this is not an option and the use of mesochronous FIFOs is pure overhead. For the aelite NOC the mesochronous FIFOs are reported to more than double the area of a router [10, Ch. 8].

A more straightforward solution to achieve robustness would be an entirely asynchronous implementation of the routers and links. A representative set of asynchronous NOCs are CHAIN [20], MANGO [4] and ANOC [21]. The only asynchronous NOC that provides hard real-time guarantees is MANGO, but as stated above its hardware cost is considerable compared to the simple TDM-based aelite router. And as previously mentioned, this is what motivated our work on exploring the asynchronous TDM-based router designs. Initial studies of an asynchronous version of an aelite style router was reported in [22], but this work only considered the router and it was not continued.

Finally we mention that a GALS-style multi-processor platform may use bi-synchronous FIFOs with flow-control [15], [17] in the interfaces between the IP-cores and the NIs and in the interfaces between the NIs and the routers. This will allow the use of independently clocked IP-cores.

III. Overall Platform Architecture

In order to provide some background and context for the router designs that are presented in the following, this section provides a brief introduction to time division multiplexing as used in our NOC and to the architecture of the multi-core platform in which the asynchronous TDM-based NOC is being used.

Time division multiplexing (TDM) allows several end-to-end communication circuits to share the same physical resources (here the links and routers of the NOC). Time is partitioned into fixed-duration time-slots, and packets/flits are injected into the NOC according to a predetermined periodic static schedule. The bandwidth and latency of a virtual circuit depends on the number of slots it is assigned within a schedule period. The choice of a TDM-based NOC is outside the scope of of this paper. For more insight on this issue the interested reader is referred to [2].

The NOC uses packet switching and wormhole routing and the TDM-schedule ensures that a packet/flit can traverse the NOC without colliding with any other packet/flits. The NOC presented in this paper uses source routing, and it is thus similar to the aelite NOC [6]. The combination of TDM-scheduling and source-routing leads to the simplest possible router implementation – a pipelined multiplexer that

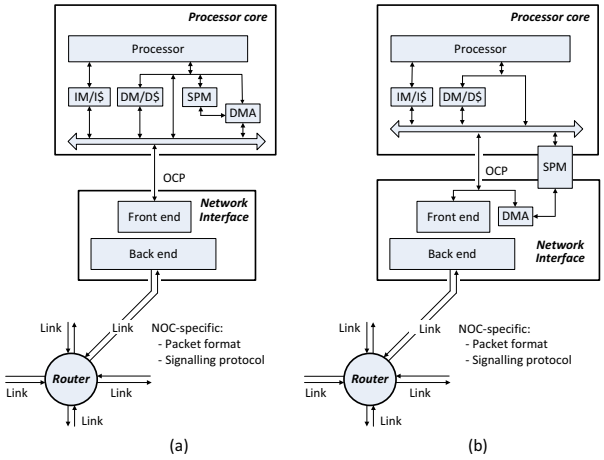

Figure 2. (a) Block-diagram showing the conventional architecture of processor node. The figure expands the view of Fig. 1. (b) Block-diagram using the new area-efficient NI-design presented in [24].

switches packets/flits from input ports to output ports – and this flow-based operation is a good fit to an asynchronous implementation. With this router design, all the intelligence (including routing-tables and schedule-tables) are in the NIs and a NI is a synchronous circuit that is synthesized from an RTL-description. The router itself is a simple and shallow pipeline without buffers, arbitration and flow-control – features that are typically found in routers and which are responsible for most of the area of typical routers [13].

As shown in Fig. 2(a) a processor node for our platform comprises the processor itself, instruction and data memories or caches, a local private scratchpad memory (SPM) and one or more DMA-controllers. This architecture is similar to the CompSoC platform [23]. The architecture supports message passing implemented by DMA-driven transfer of blocks of data from the local SPM to SPMs in remote nodes. The SPM in a node is dual-ported in order to allow the processor and the DMA-controller to access it without interfering with each other. A separate tree-shaped NOC (not shown) provide access to a shared block of memory and this will serve the instruction and data memories or caches. The entire platform has one global address space and there is no hardware support for cache coherency.

The network interface (NI) consists of a front-end that is responsible for the interfacing to the processor node (that use a read/write style protocol like OCP), and a back back end that is responsible for the interfacing to the routers (that use a packet-based protocol). During a slot in the TDM-schedule the NI can transmit a packet/flit into the NOC and receive a packet/flit from the NOC. For completeness we mention that the actual implementation of our platform use a novel area-efficient NI micro-architecture shown in figure 2(b) [24].

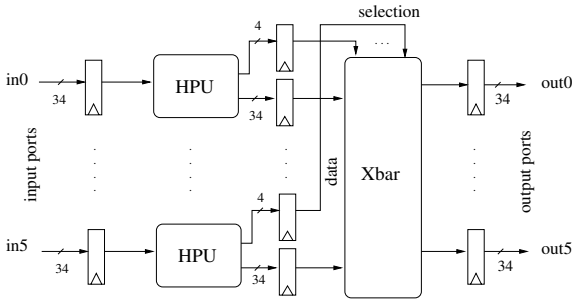

Figure 3. Block diagram showing the micro-architecture of a synchronous TDM router. The design is a 3-stage pipeline: (i) link traversal, (ii) header parsing unit (HPU) and (iii) traversal of the crossbar switch (Xbar)

IV. The Clocked Router Designs

This section presents our re-implementation of the synchronous and the mesochronous aelite routers [6]. It serves as the starting point for the asynchronous designs and since all routers will be implemented in the same technology it enables a fair apples-to-apples comparison of the designs.

A. The Aelite Router

The aelite router is a 3-stage pipeline as shown in Fig 3. The three pipeline stages are: (i) link traversal, (ii) header parsing unit (HPU) and (iii) traversal of the crossbar switch (Xbar). As already mentioned we use source routing of packets. A packet consists of one or more flits and a flit consists of 3 phits. The first flit in a packet consists of a header-phit and two data-phits. The router is agnostic to the number of flits in a packet but the current NI-design uses packets consisting of a single flit. A phit is 32 bits of data or header supplemented by 2 type-bits: a valid bit that indicates if a phit is present in the current clock cycle and a bit that identifies the header phit. The original aelite-design uses a different encoding of the type bits. A router typically has five ports, and this allows construction of mesh-type topologies.

The header phit contains the route and the local address in the destination SPM. Each input port of the router has a HPU that extracts two bits from the route that are used to select the destination output port of the router. At the same time the HPU shifts the header phit two positions to align the header for the next router along the path. The path through the crossbar is locked until the last phit of the packet has propagated through the crossbar.

The fact that the pipeline depth matches the number of phits in a flit allows the TDM-scheduling to work at the flit-level rather than the phit-level. This is an engineering compromise. The advantage is smaller schedule tables. The disadvantages are that two independent parameters, the pipeline depth and the number of phits in a flit, are linked. It also means that if links are to be pipelined, then 3 pipeline stages must be added.

B. The Mesochronous Aelite Router Design

A mesochronous router obtained by extending the clocked router described above with FIFOs on all input ports. One

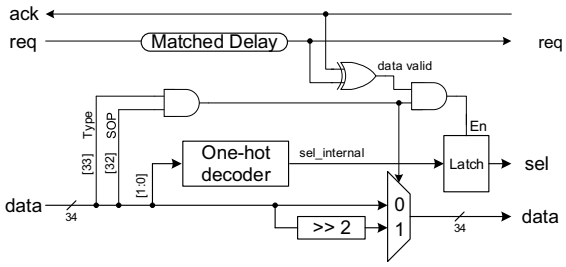

Figure 4. Diagram of the HPU implementation in the asynchronous designs.

of the published mesochronous aelite routers uses the bi-synchronous FIFOs from [15] and this is one of the designs that we have explored. The implementation of a FIFO uses standard cells and consists of 5 registers and token-ring based read and write pointers. This FIFO provides flow control signals and can sustain a throughput of one data-item per cycle for a clock skew of plus/minus one clock period. Other options for the mesochronous synchronization could be the full-custom FIFOs from [17] or the FIFO design from [18]. As flow control signals are not needed in a mesochronous system, the latter use only 3 register stages – one stage based on flip-flops and 2 stages based on latches. We have not implemented this FIFO but considering the standard cells involved we estimate its size to be roughly half of the bi-synchronous FIFO.

V. The Asynchronous Router Designs

This section presents four alternative asynchronous router designs.

A. Introductory remarks

Asynchronous circuits use handshaking among neighboring registers/latches to control the transfer of data. This handshaking can be 4-phase (return-to-zero) or 2-phase (non return-to-zero). In one full handshake cycle one token (*i.e.* phit) is forwarded from one latch stage to the next. In addition, different encodings of data are possible. This means that a range of asynchronous implementations exist. Readers who are not familiar with asynchronous design are referred to a textbook on the subject [25].

The designs that will be presented are a four-phase bundled-data design ("4ph-bd"), a two-phase bundled-data design ("2ph-bd"), a 2-phase bundled-data with a “gating” mechanism to avoid switching activity in unused router ports ("2-ph-bd-g"), and a 2-phase bundled-data design with “gating” that uses delay-insensitive, Level-Encoded Dual-Rail (LEDR) [26], on the links ("2ph-bd/LEDR-g").

The asynchronous designs presented below all use the same flit-format (3 phits per flit) and the same 3-stage pipeline. Due to the use of handshake-latches in the pipeline stages, it is not possible to store 3 consecutive tokens (*i.e.* phit) in a router. For this reason the asynchronous router designs require TDM-scheduling at the phit-level. There is no fundamental problems in this, and the larger schedule-table has little impact on the

size of the NI. A deeper discussion of flit-level vs. phit-level scheduling is beyond the scope of this paper.

B. A 4-phase Bundled-Data Design

The first design (4ph-bd) uses 4-phase bundled-data handshaking and it is a straightforward implementation corresponding to the clocked design, using handshake latches in the place of clocked registers. A handshake latch is composed of a regular enable latch and a simple 4-phase bundled-data latch controller [25]. Delay elements are needed in the request signal paths to match the propagation delays in the links, in the HPU and in the crossbar.

The header parsing unit (HPU) follows the same design as the Aelite router that is described in Section IV-A. The two least significant bits of the route field are extracted and recoded into a one-hot (1-of-4) encoded control signal for the crossbar. A diagram of the implementation of the HPU can be seen in Fig. 4.

The crossbar switches an input to the specified output. It is implemented using a handshake de-multiplexer for each input port and a merge for each output port. With the one-hot encoded select-signal from the HPU stage, each output of a de-multiplexer can be implemented as an array of and-gates. With this implementation the crossbar has only two levels of logic resulting in a fast implementation. The control path contains a join that synchronizes all request signals from the 5 input ports, a join that synchronizes all the acknowledge signals from the 5 output ports and a matched delay for the propagation delay of the crossbar. Each join is implemented using one C-element, resulting in 2 C-elements for the synchronization of ports.

It is important to note that in every handshake cycle the crossbar consumes one token (*i.e.* phit) from every input port and it produce one token (*i.e.* phit) on all output ports – the crossbar is a strongly indicating function block [25]. This is necessary in order to implement time division multiplexing at the global level. The difference from a clocked design is that routers are not synchronized and that the inherent FIFO-behavior of asynchronous pipelines can accommodate timing fluctuations.

C. A 2-phase Bundled-Data Design using LEDR Links

The rate at which phits can traverse the router depends on the worst-case time it takes to perform a handshake cycle transferring data from one handshake latch to the next. Changing from a 4-phase protocol to a 2-phase protocol can ideally double the speed and (depending on the implementation) also reduce the energy consumption. Furthermore, to make the design robust against tolerances and fluctuations in gate and wire delays and to allow plug-and-play composition of routers and links, it may be desirable to use delay-insensitive signaling on the links.

A good compromise between robustness and area-efficiency is to make the links delay-insensitive and to implement the router using bundled-data techniques. In order to avoid complex decoding circuitry in the routers, in the we use LEDR links. A block diagram of this router design (called 2ph-bd/LEDR-g) is shown in Fig. 5 and explained below.

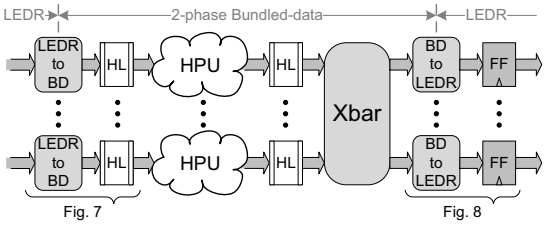

Figure 5. Block diagram of the 2-phase bundled-data design using LEDR links.

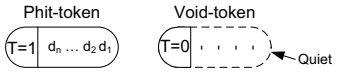

Figure 6. The two types of tokens for communication on the links.

As mentioned in the previous section the crossbar switch in the router relies on the supply of tokens from all inputs of a router, to step into the next time slot. If no token arrives at the input of a router, no progress will happen. When using 2-phase LEDR links one wire per bit switches for every token sent, even if no data is transmitted.

This overhead can be reduced by not switching the 33 data wires in cycles where phits are not transmitted. This can be implemented at the port level by introducing two types of tokens: *Phit-tokens* that carry a header phit or a data phit, and *void-tokens* that carry an empty synchronization token. Fig. 6 illustrates the two types of tokens. The distinction between the two types of tokens is already present in the form of the valid bit explained in section IV-A.

The LEDR to bundled-data converter that is used in the input ports of the router, see Fig. 5, is implemented as shown in Fig. 7. The converter detects a token and its type in the

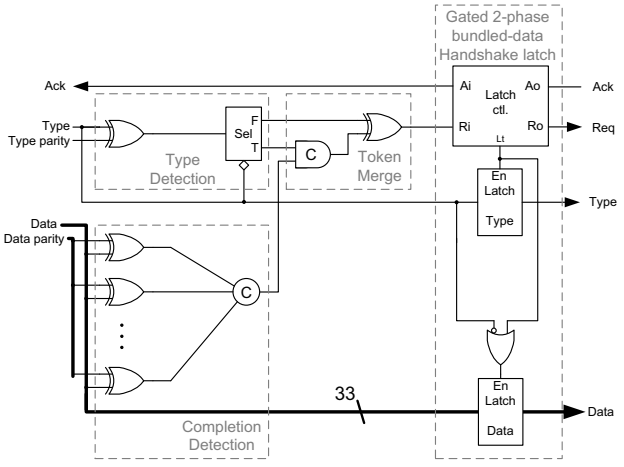

Figure 7. Circuit implementation of the LEDR to 2-phase bundled-data converter – the first pipeline stage of the router.

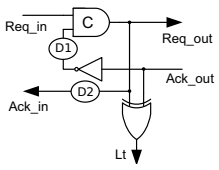

Figure 8. Circuit implementation of 2-phase latch controller.

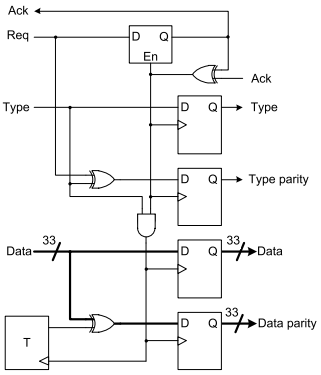

Figure 9. Circuit diagram of Bundled-data to LEDR converter [27].

Type Detection circuit. The select-element transitions the True port if the token is a phit-token or the False port if the token is a void-token. If the token is a phit-token the Token Merge circuit synchronizes with the Completion Detection through the C-element. The output of the Token Merge serves as the request signal to the latch controller. The Type bit is stored in the type latch. If the token is a phit-token the Data Latch is enabled.

The latch controller used in the first two pipeline stages of the router is the reduced complexity 2-phase micro-pipeline latch controller from [28, Fig. 4] shown in Fig. 8. This design avoids using a complex toggle element.

Some timing assumptions need to be followed for the controller to work safely. The first assumption is that the latch needs to be transparent sufficient time for the data to be latched before the incoming acknowledge from the next stage changes the phase to opaque. The second timing assumption is that the latch needs to be opaque sufficient time before outgoing acknowledge is sent back to the left controller and a new handshake is initiated. Delays D1 and D2 are introduced to fulfill these assumptions, respectively.

The last pipeline stage in the router contains a bundled-data to LEDR converter, see Fig. 9. It is based on the circuit from [27] modified to handle phit-tokens and void-tokens. If the incoming token is a phit-token all four registers are clocked, otherwise only the two Type registers are clocked. The toggle flip-flop contains the phase of the token. The phase of the type signals and the data signals will not always be equal.

D. A 2-phase Bundled-Data Design (with gating)

The results in section VII show that the 2ph-bd/LEDR-g design is expensive in terms of area and energy consumption. The reason is the encoding and decoding of the LEDR code used on the links. For this reason we have implemented a third design, 2ph-bd-g, that uses 2-phase bundled-data on the links as well. The design uses the "gated 2-phase bundled-data handshake latch" shown in the right hand side of Fig. 7. In router ports propagating void-tokens this design avoids enabling/clocking the 33 latches that hold the data-part of a phit and the type bit indicating the header. Only the latch holding the valid bit that is used to distinguish between phit-tokens and void-tokens is enabled. This is similar to clock-gating, and saves power related to driving the 33 enable inputs.

E. A simple 2-phase Bundled-Data Design (without gating)

We anticipated that the 2-phase designs would be faster than the 4-phase design; perhaps not double but at least substantially faster. The results in section VII show that the opposite is the case. A closer look reveals that this has to do with re-shaping of the gated enable-pulse. This causes violation of setup and hold time requirements of the latches, and to fix this it is necessary to increase the delay of the delay elements (see Fig. 8). This negatively affects the handshake cycle time. For this reason we have implemented a fourth design, the 2ph-bd design, which is a straightforward 2-phase bundled-data design without gating that is similar to the 4ph-bd design.

VI. Implementation and Design flow

This section addresses the design flow and elaborates on some of the challenges faced when implementing asynchronous circuits using conventional EDA-tools.

All the above designs were described in VHDL and simulated using ModelSim. They were implemented using a 65nm CMOS standard cell library from STMicroelectronics with HVT and SVT cells. Synthesis was performed using Design Compiler from Synopsys. To have a more precise evaluation, taking into account the effects of placement and routing, actual layouts were produced using Cadence SOC Encounter. All four designs were tested through simulation after every step of the design flow (post-synthesis and post-layout). Power consumption was estimated using Synopsys PrimeTime.

Regarding the use of conventional design tools for the asynchronous routers, careful handling was needed for constraining the designs for synthesis. Local timing constraints were applied in order to optimize the combinational logic, and place delay elements of specific values. Manual handling of combinational loops was also done. The designs were synthesized with an aim for optimizing for speed.

Concerning the delay elements used in the asynchronous designs a trial-synthesis and simulation step was done to find the appropriate delay value needed in each case. A delay element matching the HPU combinational delay and one matching the crossbar combinational delay are needed in all asynchronous router designs. After determining the HPU and crossbar combinational delay, a safe margin of 20% was added to cover for delay fluctuations. The delays were implemented as a series of buffers and inverters of the same technology.

This was done by directing the synthesis tool to assign specific delay or delay ranges on a path. In addition to these, some timing assumptions need to be met in the 2-phase controller as described in Section IV. Due to wire-load effects that take place and affect the timing after the layout of the asynchronous routers, some adjustments needed to be made on the delay elements. An optimization process of adjusting the delay elements, performing synthesis and layout was repeated in order to achieve correct timing behavior (timing closure) of the asynchronous circuit without excessive delays compromising the performance.

VII. Results

This section presents results on cell area, frequency and energy consumption for the different router designs. The results are summarized in Table I and discussed below.

For the simulations the different router designs were connected to an environment consisting of independent ideal producers and consumers on the input and output ports. The inputs were driven with streams of 3-phit packets and with a link utilization of 70 %, meaning that 30 % of the tokens are void-tokens.

The frequency reported for the asynchronous designs is the frequency of a handshake. As seen in Table I the frequency of all the designs is comparable. Among the asynchronous routers the 2ph-bd design shows the best frequency. This was expected due to the 2-phase handshake protocol and because it is the simplest of the 2-phase designs.

The area shown in Table I is the cell area of the designs reported after synthesis. The area of the asynchronous bundled-data designs (4ph-bd, 2ph-bd and 2ph-bd-g) are all quite similar (7401-7594 um^2) and slightly smaller than the synchronous router. This is as expected because all these designs implement the same combinational circuitry and use either clocked flip-flops or handshake controlled latches (that are slightly smaller than clocked flip-flops). The area is almost doubled in the 2ph-bd/LEDR-g design. This is also as expected and it is due to the use of LEDR encoding, *i.e.* two wires per bit, and from the more complex completion detection circuitry.

The area of the mesochronous router is the sum of the area of the synchronous router and the area of the FIFOs added to the input ports. As mentioned in Section IV-B the area of the optimized FIFO design [17], [18] is an estimate.

It is interesting to observe that the FIFOs make the mesochronous router 2-3 times larger than the synchronous router. It is even more interesting to observe that all the asynchronous bundled-data routers (4ph-bd, 2ph-bd and 2ph-bd-g) are 2-3 times smaller than the mesochronous router, and that even the 2ph-bd/LEDR-g design is 1.5-2 times smaller than the mesochronous router. These are key results of the paper.

The power consumption was measured using PrimeTime and the figures are the total of cell, wire and leakage power. Since power is dependent on speed we have calculated the energy per cycle, that is either clock cycle or handshake cycle, in order to better compare the energy efficiency of the designs.

Table I
Results for the router designs implementations.

	Cell Area	Post-synthesis		Post-layout	
		Freq.	Energy / cycle	Freq.	Energy / cycle
	um^2	MHz	pJ	MHz	pJ
Synchronous	8026	1111	1.73	885	1.40
Mesochronous					
FIFO [15]	24239	1111	4.32	724	7.87
FIFO [17], [18]	(16132)				
FIFOs					
from [15]	16213	1111	2.58	724	6.16
from [17], [18]	(8106)				
4ph-bd	7401	833	7.91	701	8.20
2ph-bd	7594	998	7.92	711	8.03
2ph-bd-g	7536	900		593	
Void 100 %			1.64		2.11
Void 30 %			6.51		7.23
Void 0 %			8.77		9.66
2ph-bd/LEDR-g	12578	862		645	
Void 100 %			3.82		3.80
Void 30 %			8.31		9.95
Void 0 %			10.02		12.00

As seen in Table I the energy consumption of the mesochronous and the asynchronous designs are relatively similar, in the order of 8-10 pJ per cycle. In order to explore the ability of the two gated designs (2ph-bd-g, 2ph-bd/LEDR-g) to avoid unnecessary switching, we simulated these two designs using input streams with 100 % phit-tokens and with 100 % void-tokens in addition to the 70 % / 30 % phit/void mix used in all cases. The results show that the 2ph-bd-g design saves considerable energy when propagating void-tokens, while it consumes the same energy as the mesochronous in the 70 % / 30 % phit/void case. 2ph-bd/LEDR-g design also saves considerable energy on void-tokens, but the additional complexity has some energy overhead. In practice, the percentage of void-tokens will be greater than phit-tokens allowing for great energy savings for the gated designs.

In conclusion, the above results show that all the asynchronous designs are more area-efficient than a mesochronous router. Additionally, different trade-offs are possible for speed, energy consumption per cycle and robustness. The 2ph-bd design offers the highest speed with 4ph-bd slightly slower but very close. Energy savings can be achieved in the 2ph-bd-g by gating off unnecessary switching by trading some of the speed. Finally robustness can be achieved in the 2ph-bd/LEDR-g design with a cost on area and a slightly higher energy consumption, that can be limited by gating.

VIII. Conclusion

TDM is a straightforward approach for providing virtual channels in NOCs for hard real-time systems and the routers for such a NOC have the smallest hardware complexity of any router we know. The paper extended previous work on synchronous and mesochronous TDM-based NOCs by exploring the use of asynchronous routers that allow a truly GALS-style implementation of a NOC-based multi-core platform.

The paper presented a new timing-architecture for a TDM-based NOC using mesochronous NIs and asynchronous routers

and links where the elasticity of the network of asynchronous routers and links covers for phase differences between the clocks used in the NIs. The paper presented a number of asynchronous router designs: a simple 4-phase bundled data design, a 2-phase bundled data design, a version of the 2-phase bundled data design using a mechanism that avoids unnecessary toggling of signals when a link is idle, and a more complex 2-phase bundled data design using level-encoded dual-rail links and the previous "clock gating" mechanism as well. The paper compared the area, speed and power of these designs with an existing router design for a mesochronous version of the aelite NOC;

Our results confirm previously published results and shows that the mesochronous FIFO-buffers account for 50-66,% of the total area of the mesochronous router. The asynchronous routers avoid these FIFOs and all asynchronous routers are smaller than the mesochronous router. The area of the 4ph-bd, the 2ph-bd and the 2ph-bd-g router is 33-50 % and the area of the 2ph-bd/LEDR-g router is 50-66 % of the mesochronous router. The speed of the mesochronous and the asynchronous designs are roughly similar; 600-750 MHz for post-layout designs. The energy consumed per phit is in the 8-10 pJ/phit range; largest for the complex 2ph-bd/LEDR-g design. However, a large gain is achieved in designs that gate off unnecessary switching activity. Overall, the contribution of this work is a solution to clock distribution problem while offering real-time guarantees with a range of router designs that apply various mechanisms and offer different trade-offs. Gains are on speed (4ph-bd, 2ph-bd), energy consumption per phit of data (2ph-bd-g), robustness(2ph-bd/LEDR-g), while keeping a very low area compared to alternative (mesochronous) designs.

ACKNOWLEDGMENTS

The authors would like to thank the T-CREST project partners for their support as well as Joachim Rodrigues and Oskar Andersson from the Department of Electrical and Information Technology of Lund Univeristy for their help with the EDA tools.

REFERENCES

[1] Time-predictable Multi-Core Architecture for Embedded Systems (T-CREST). [Online]. Available: www.t-crest.org

[2] K. Goossens and A. Hansson, "The aethereal network on chip after ten years: Goals, evolution, lessons, and future," in *Proc. ACM/IEEE Design Automation Conference (DAC)*, Jun. 2010, pp. 306 –311.

[3] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 83, no. 10, pp. 1374–1396, 1995.

[4] T. Bjerregaard and J. Sparsø, "A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-chip," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. IEEE Computer Society Press, 2005, pp. 34–43.

[5] The International Technology Roadmap for Semiconductors, "ITRS 2011 Edition – Design," 2011. [Online]. Available: http://www.itrs.net/

[6] A. Hansson, M. Subburaman, and K. Goossens, "aelite: a flit-synchronous network on chip with composable and predictable services," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2009, pp. 250–255.

[7] P. T. Wolkotte, G. Smit, G. Rauwerda, and L. Smit, "An energy-efficient reconfigurable circuit-switched network-on-chip," in *Proc. 19th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2005*, April 2005.

[8] D. Wiklund and D. Liu, "SoCBUS: Switched network on chip for hard real time embedded systems," in *Proc. IEEE International Parallel and Distributed Processing Symposium, IPDPS 2003*. IEEE Computer Society, 2003, p. 78a.

[9] K. Goossens, J. Dielissen, and A. Rădulescu, "The Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, Sept-Oct 2005.

[10] A. Hansson and K. Goossens, *On-chip interconnect with aelite / Composable and predictable systems*. Springer, 2011, embedded systems.

[11] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip," in *Proc. Design, Automation and Test in Europe (DATE)*. IEEE Computer Society Press, Feb. 2004, pp. 890–895.

[12] C. Paukovits and H. Kopetz, "Concepts of switching in the time-triggered network-on-chip," *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 120–129, 2008.

[13] J. Sparsø, "Networks-on-chip for real-time multi-processor systems-on-chip," in *Proc. International Conference on Application of Concurrency to System Design (ACSD)*, Jun. 2012, pp. 1–5.

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

[15] I. M. Panades and A. Greiner, "Bi-synchronous fifo for synchronous circuit communication well suited for network-on-chip in gals architectures," in *International Symposium on Networks-on-Chip (NOCS)*, 2007, pp. 83–92.

[16] I. M. Panades, A. Greiner, and A. Sheibanyrad, "A low cost network-on-chip with guaranteed service well suited to the GALS approach," in *1st International Conference on Nano-Networks (Nano-Net)*, 2006, pp. 1–5.

[17] P. Wielage, J. Marinissen, M. Altheimer, and C. Wouters, "Design and DfT of a high-speed area-efficient embedded asynchronous FIFO," in *Proc. Design, Automation and Test in Europe (DATE)*, 2007, pp. 853–858.

[18] I. Loi, F. Angiolini, and L. Benini, "Developing mesochronous synchronizers to enable 3D NoCs," in *Proceedings of the conference on Design, automation and test in Europe (DATE)*, 2008, pp. 1414–1419.

[19] D. Ludovici, A. Strano, G. N. Gaydadjiev, and D. Bertozzi, "Mesochronous NoC technology for power-efficient GALS MPSoCs," in *Proc. International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip (INA-OCMC)*. ACM, 2011, pp. 27–30.

[20] J. Bainbridge and S. Furber, "Chain: A delay-insensitive chip area interconnect," *IEEE Micro*, vol. 22, no. 5, pp. 16–23, Sep./Oct. 2002.

[21] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An asynchronous NOC architecture providing low latency service and its multi-level design framework," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC).*, Mar. 2005, pp. 54 – 63.

[22] T. Felicijan, D. Dielissen, and K. Goossens, "Asynchronous TDMA Networks on Chip," Philips Research Eindhoven, Tech. Rep., Jan. 2004, technical Note 2004/00801.

[23] A. Hansson, K. Goossens, M. Bekooij, and J. Huisken, "CoMPSoC: A Template for Composable and Predictable Multi-Processor System on Chips," *ACM Transactions on Design Automation of Electronic Systems*, vol. 14, no. 1, 2009.

[24] J. Sparsø, E. Kasapaki, and M. Schoeberl, "An Area-efficient Network Interface for a TDM-based Network-on-Chip," in *Proc. Design Automation and Test in Europe (DATE)*, 2013, pp. 1044–1047.

[25] J. Sparsø, "Asynchronous circuit design – a tutorial," in *Principles of asynchronous circuit design – A systems perspective*, J. Sparsø and S. Furber, Eds. Kluwer Academic Publishers, 2001, ch. 1-8, pp. 1–152.

[26] M. Dean, T. Williams, and D. Dill, "Efficient self-timing with level-encoded 2-phase dual-rail (LEDR)," in *Advanced Research in VLSI: Proceedings of the 1991 UC Santa Cruz Conference*, C. H. Séquin, Ed. MIT Press, 1991, pp. 55–70.

[27] M. Imai and T. Yoneda, "Improving Dependability and Performance of Fully Asynchronous On-chip Networks," in *Proc. Intl. Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*. IEEE Computer Society Press, apr 2011, pp. 65–76.

[28] G. S. Taylor and G. M. Blair, "Reduced complexity two-phase micropipeline latch controller," *IEEE Journal of Solid State Circuits - Institute of Elect and Electr Engineers*, vol. 33, no. 10, pp. 1590–1593, 1998.