Implementation of an Asynchronous Bundled-Data Router for a GALS NoC in the Context of a VSoC

Patrick Russell[1], Jens Döge[1], Christoph Hoppe[1], Thomas B. Preußer[2], Peter Reichel[1] and Peter Schneider[1]
[1]Fraunhofer Institute for Integrated Circuits IIS, Divison Engineering of Adaptive Systems EAS
[2]Technische Universität Dresden, Institute of Computer Engineering
Dresden, Germany
E-mail: patrick.russell@eas.iis.fraunhofer.de

***Abstract*—Designs of asynchronous networks-on-chip are of growing interest because a complete asynchronous implementation can solve the synchronization problems of large networks. However, asynchronous circuits suffer from the lack of proper design flows because their functionality often relies on timing constraints, which are not extensively supported by common CAD synthesis tools. This paper proposes the design and implementation of an asynchronous router architecture suitable for a network-on-chip in the context of a Vision-System-on-Chip. The developed design flow for the synthesis of asynchronous bundled-data pipelines is based on common synthesis tools and, therefore, enables high compatibility with synchronous designs and a low barrier to entry.**

***Index Terms*—network-on-chip, low power, low latency, GALS, asynchronous circuits, vision-system-on-chip, CAD, synthesis**

I. Introduction

Networks-on-Chip (NoCs) are considered to be a promising solution for high-throughput communication between different processing elements (PEs) within a System-on-Chip (SoC) [3] [13]. However, synchronization of large NoCs in nanoscale technologies using a single global clock signal can be extremely difficult and inefficient in terms of performance and power consumption. As a result, the methodology of globally asynchronous and locally synchronous (GALS) designs has evolved [9] [16].

Asynchronous circuits operate with local handshaking signals (e.g. request and acknowledge) rather than a global clock for the communication between components and pipeline stages. They can be classified by their handshaking protocol (2-phase or 4-phase) and their data encoding (e.g. bundled-data, dual-rail or 1-of-N) [10] [11]. Depending on their implementation, different timing assumptions are necessary to ensure the correct functionality of the circuit. Although asynchronous circuits show great advantages at low-power, high-performance applications, they are rarely used in industrial environments. A fundamental problem is their limited support from common CAD synthesis tools and hence their low compatibility with synchronous designs.

In this paper, we propose the implementation of an asynchronous bundled-data router for GALS NoCs. The architecture is designed for low latency and very high energy efficiency, which is mandatory for the specific requirements of a NoC in the context of a Vision-System-on-Chip (VSoC). A VSoC, such as the one proposed by Döge et al. [4], features integrated signal processing of image data besides the actual image sensor. The necessary functional units are controlled by an integrated control unit. A VSoC is typically a very large ASIC. A large area of the chip is occupied by the sensor matrix and is not available for the routing of global communication wires, complicating the synchronization problem even further.

We propose a design flow for synthesizing asynchronous bundled-data circuits, which is based on common CAD synthesis tools and therefore enables a high compatibility with synchronous designs.

A comparison to a synchronous GALS router implementation is given, and we will show the measurement results of a physically implemented network on a 180 nm chip.

II. Related Work

Many implementations of asynchronous routers were proposed in recent years, most of them utilizing the 4-phase protocol because of its easier implementation.

The ANoC, proposed by Beigné et al. [1], utilizes the 4-phase protocol and 1-of-4 data encoding, as well as wormhole and source routing. A complete SystemC model of the router has been used for its test and verification. Bjerregaard et al. [2] proposed the MANGO router, which utilizes the 4-phase protocol and bundled-data encoding. The router comprises a best-effort and a guaranteed-service router. Ghiribaldi et al. [5] were the first to propose a router utilizing the 2-phase protocol with bundled-data encoding. The architecture is based on the MOUSETRAP implementation of asynchronous pipelines proposed by Singh and Nowick [14].

To ensure correct functionality of a bundled-data architecture, several relative timing constraints (RTCs) between data and control signals have to be satisfied. These RTCs have to be considered by the synthesis and must be supplied to the corresponding tools. Several design methodologies were proposed for the synthesis of asynchronous bundled-data circuits using common synthesis tools, mostly using very specific design styles resulting in a very low compatibility with synchronous designs [8] [15]. Ghiribaldi et al. [5] describe an iterative synthesis flow based on common CAD synthesis tools, which expects a description in common hardware description languages. Gibiluka et al. [7] proposed a framework

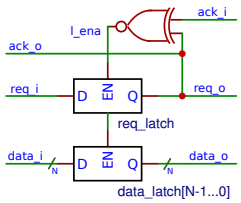

Figure 1. MOUSETRAP pipeline stage implementation

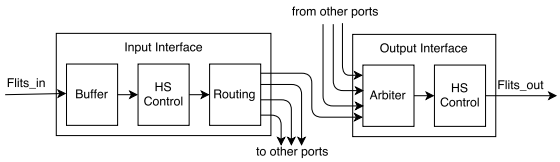

Figure 2. Structure of the router (adapted from [12] and based on [6])

with a similar approach, which utilizes a XML description of the necessary RTCs and enables the automated fulfillment of these constraints during synthesis.

However, in most publications, no detailed information, such as circuit description level or the utilized synthesis process, is given about the procedure of the synthesis flow.

III. Architecture

The proposed and implemented 5-port router is based on the router switch presented by Ghiribaldi et al. [5], which provides a basic architecture and an optimization of the performance of a single router. This meets the requirements of a network in the specific context of a VSoC. The implemented router utilizes a 2-phase bundled-data architecture for asynchronous pipelining based on MOUSETRAP. Consequently, the router has great potential for high performance, low area overhead and low power consumption [10] [11]. The register of a MOUSETRAP pipeline stage uses simple D-latches to store data. The small stage control logic consists solely of a D-latch to store the incoming request and a single XNOR gate to control the pipeline register's enable signal and the request latch [14] (Figure 1).

Each port contains an input and an output interface and provides distributed, highly modular routing. The structure is presented in Figure 2. The architecture uses wormhole and source routing as well as a single bit to determine the end of a packet (EOP) along with an adjustable flit (flow control unit) size of N bits.

A. Input Interface

The input interface of every port routes incoming packets to the output interface of the addressed output port by decoding the address in the routing information of the header flit and rotating the routing information by two bits (source routing). Other routing algorithms can be easily implemented as well. The top-level view and microarchitecture are presented in Figure 3. The handshaking stage control logic (HS_CTRL) operates based on the principle of a mousetrap pipeline and controls the register consisting of latches and a flip-flop to store the EOP bit. The routing logic decodes the routing information and places a request (*arb_req*) to the arbiter of the

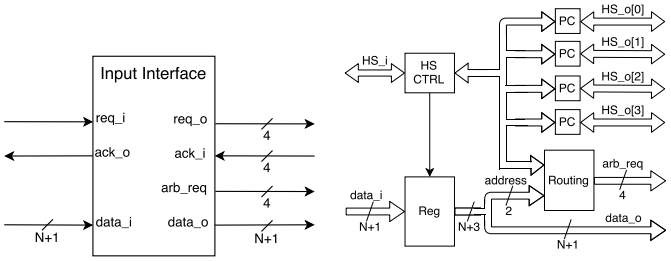

Figure 3. Top-level view and microarchitecture of an input interface (adapted from [12]).

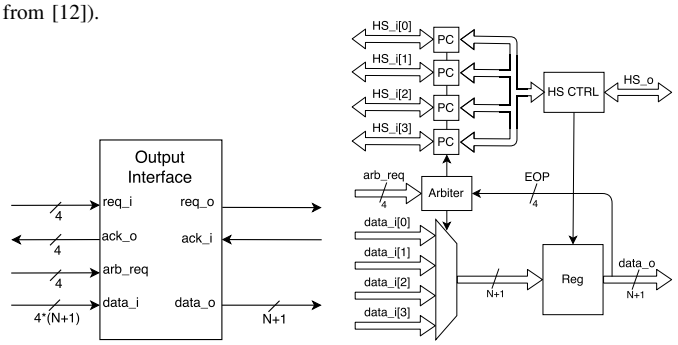

Figure 4. Top-level view and microarchitecture of an output interface (adapted from [12]).

addressed output interface. Phase converters (PC) adjust the handshaking signals' (HS) phases between the input interface and the four connected output interfaces. This is necessary because the phases do not necessarily match when the communication alternates between multiple output interfaces.

B. Output Interface

The output interface of every port arbitrates between multiple incoming packets from the four connected input interfaces. The interface consists of a pipeline stage (HS_CTRL and corresponding register (Reg)), further phase converters (PC) and an arbiter. The top-level view and microarchitecture are presented in Figure 4. The arbiter selects one of the requests (*arb_req*) trying to access the output port at a time. The architecture of the arbiter is based on the one presented by Ghiribaldi et al. [5] and consists of RS-Flip-Flops, a 4-input mutual exclusion component (mutex) and several AND-gates that ensure a fast reset time. After forwarding an EOP bit, the arbiter is reset and prepared for the next arbitration. The phase converters and the multiplexer, which are controlled by the arbiter, ensure that only the winner's handshaking and data signals can reach the asynchronous pipeline stage.

C. Timing Constraints

Several RTCs have to be satisfed to ensure correct functionality of the bundled-data architecture:

- The first constraint affects the delay time within the routing logic. To prevent glitches, the address decoder logic has to be switched before the request of the header flit sets the corresponding *arb_req* signal.
- A second constraint occurs after arbitration. The minimum delay time of the request signal through the phase converters in the output interface controlled by the arbiter must be equal to or greater than the maximum delay time

of the corresponding data through the 4-input multiplexer, which is also controlled by the arbiter. As a result, it is guaranteed that the data of the winning request reaches the register of the output interface pipeline stage on time and can be stored safely.
- A third RTC is found between the delays of the request signal from the input interface through the two phase converters to the pipeline stage of the output interfaces and the corresponding data signals through the multiplexer to the register of the same pipeline stage.
- Further RTCs affect the matching delays between the data and request signals at the input and output ports.

IV. Design Flow

For the synthesis of asynchronous bundled-data pipelines, a design flow based on common CAD synthesis tools has been developed. The starting point is an RTL description in common hardware description languages like Verilog or VHDL. Therefore, the design flow provides high compatibility with synchronous designs and a low barrier of entry. For synthesis and place and route, the flow utilizes the *Synopsys Design Compiler* and the *Cadence SoC Encounter* tools.

A. Hardware Description

To ensure correct operation, not only the logical functionality, but also the dynamic behavior of an asynchronous circuit is important [5]. It is mandatory to prevent the used synthesis tools from applying harmful logic optimizations within the control circuits of the asynchronous pipelines. The gate level implementations of these circuits have to match the original design exactly in order to avoid unwanted glitches of the handshaking signals. However, adjusting driver strength or inserting additional buffers during synthesis are desired optimizations.

To achieve a correct mapping from design to implementation, a few specific cells inside the control path of the handshaking signals have to be directly instantiated as standard cells on gate level. After elaboration of the remaining circuit, described on register-transfer level, the logic optimization of these directly instantiated standard cells is prevented by using the synthesis tool's `set_size_only` directive, which prevents logic optimization but allows for adjusting driver strengths.

B. Timing Loops

By definition, asynchronous circuits contain combinational loops inside the handshaking control logic. In order to enable the synthesis tool to correctly analyze the circuits' timing properties, these timing loops have to be ignored. Thus, the designer of the asynchronous pipeline has to detect and disable them by using the `set_disable_timing` directive.

C. Enforcing Relative Timing Constraints (RTCs)

Common synthesis tools do not support RTCs. However, they can be enforced by an iterative process similar to the approaches proposed by Ghiribaldi et al. [5] and Gibiluka et al. [7]. To ensure compatibility with the Synopsys design constraints format (SDC), the proposed flow enables the possibility to define paths between pins and/or ports inside the design and RTCs between different paths and groups of paths. The associated directives `set_path` and `set_rtc` can be used along with other design constraints in a single constraints file.

In a first initial synthesis run, only the maximum delays of every path in the pipeline are communicated to the tool through the `set_max_delay` directive. The specified delays are a measurement for the performance of the pipeline, similar to the clock frequency of synchronous designs. If the desired performance cannot be reached, the constraints are adjusted automatically and the initial synthesis is repeated.

After a successful synthesis, in a second synthesis run, the minimum delays necessary to fulfill the RTCs of the design are defined by the `set_min_delay` directive. These constraints are determined from the maximum delays achieved in the initial synthesis run. The second synthesis run tries to achieve the indicated minimum delays by inserting delay cells (buffers and inverters) by use of the compile instruction `compile -only_hold_time`. If the minimum constraints cannot be satisfied, the maximum delays are adjusted automatically and the second synthesis is repeated.

D. Analysis Condition and RTC Verification

The analysis condition determines which operating conditions (process corner, supply voltage and temperature) are used during the timing analysis of the synthesis tool. The following analysis conditions can be set for the synthesis:

single - The tool only considers a single operating condition during the analysis of maximum delays (setup) and minimum delays (hold). Possible process, voltage or temperature (PVT) variations are not taken into account by the timing analysis.

best-case/worst-case (bc/wc) - In bc/wc analysis, setup violations and maximum-delay constraints are checked for a given worst-case (worst-speed) operating condition, while hold time violations and minimum-delay constraints are checked for a given best-case (best-speed) operating condition. This analysis type considers the global ("Die-to-Die") PVT variations for setup and hold violations by assuming that the two given operating conditions are the worst cases with respect to their analyzed constraints. The constraints set with the `set_max_delay` and `set_min_delay` directives are checked for two different operating conditions.

on-chip variation (OCV) - In OCV analysis, the maximum and minimum delays of the setup and hold times are checked for two different operating conditions. The maximum delays are checked for a given worst-case (worst-speed) and the minimum delays are checked for a given best-case (best-speed) operating condition. This analysis tries to consider local (on-chip) PVT variations for setup and hold violations by making very pessimistic assumptions. There is no difference between bc/wc and OCV analysis for constraints set with `set_max_delay` and `set_min_delay` directives. An alternative solution is the use of the `set_timing_derate`

directive, which sets a percentage variation of the delays specified by the operating conditions along with using the bc/wc analysis.

RTCs of asynchronous circuits are independent of global PVT variations. Therefore, the single analysis condition is suitable for the synthesis of asynchronous bundled-data circuits.

While the absolute values of the delay times may vary, the RTCs have to be fulfilled for every operating condition. To guarantee correct functionality under all operating conditions, the design flow analyzes and verifies the RTCs for best-case and worst-case operating conditions during synthesis.

The `set_timing_derate` directive can be used along with the single analysis condition to ensure correct functionality in deep sub-micron technologies.

V. Experimental Results

To evaluate the implementation, we first compared the asynchronous router architecture with a synchronous implementation of a GALS NoC router. Secondly, we physically implemented a complete network on a test chip. Both evaluations used a 180 nm CMOS technology with a nominal supply voltage of 1.8 V.

The following parameters were used for the evaluation:

Area - The area of the router $A_{\texttt{router}}$ in mm² and the area $A_{\texttt{nand2}}$ in an equivalent amount of NAND2 gates.

Latency - The latency indicates the time in nanoseconds needed for a header flit to cross the router from one input port to the addressed output port, if the flit is not blocked inside the router. It contains the time for a necessary arbitration.

Cycle Time / Throughput - The cycle time $t_{\texttt{cyc}}$ in nanoseconds of an asynchronous pipeline indicates the time needed by a pipeline stage to acknowledge a transfer request. The throughput $T_{\texttt{pipe}}$ in Gigaflits per second of the router pipeline is given by the inverse of the cycle time and indicates the number of flits, which can cross the pipeline per second. Because of the architecture's highly modular structure, the router can handle up to five parallel connections. Thus, the total throughput $T_{\texttt{router}}$ in Gigaflits per second of a single router is given by:

$$T_{\texttt{router}} = 5 \times \frac{1}{t_{\texttt{cyc}}}$$

Area Efficiency - The area efficiency $AE_{\texttt{router}}$ indicates the maximum achieved throughput divided by the area needed:

$$AE_{\texttt{router}} = \frac{T_{\texttt{router}}}{A_{\texttt{router}}}$$

A. Simulative Comparison

To compare the implemented asynchronous architecture with a second GALS NoC Router, we implemented a multi-synchronous architecture, which is characterized by a separate clock domain for each router in the network. Such an implementation respects the GALS methodology and allows for the operation of parts of the network with independent clocks. Communication between two neighboring routers takes place via a source-synchronous interface.

Parameter	asynchronous		synchronous
	typical	worst-case	
$A_{\texttt{router}}$ in mm²	0,0284		0,1451
$A_{\texttt{nand2}}$ in NAND2 gates	3230		16525
Latency in ns	7,27	12,06	31,35
$t_{\texttt{cyc}}$ in ns	4,76	7,69	5,7
$T_{\texttt{router}}$ in Gflits/s	1,05	0,65	0,875
$AE_{\texttt{router}}$ in Gflits/s per mm²	36,97	22,89	6,03

Table I
Performance parameters (adapted from [12])

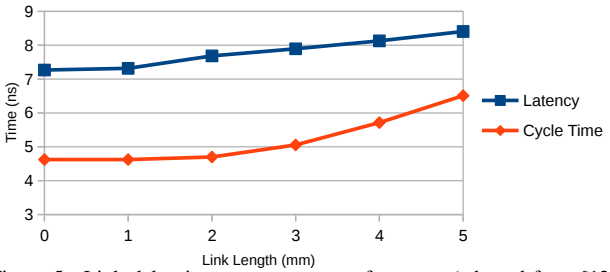

Figure 5. Link delay impact on router performance (adapted from [12]).

Both architectures were implemented with a flit size of 16 bits, an additional EOP bit and the minimum amount of flit buffer memory at each input port needed to ensure maximum throughput when receiving continuous packets (a single flit buffer needed for the asynchronous implementation and 16 flit buffer memory needed for the synchronous counterpart because of the synchronization between neighboring routers [12]). For evaluation purposes, both routers received packets from packet generators, which had to be routed to an addressed receiver.

The simulation took place under typical (1.8 V, 25 °C and typical process corner) and, for the asynchronous router, under worst-case (1.62 V, 85 °C and worst-speed process corner) operating conditions as well, because the performance of asynchronous circuits is directly affected by the operating conditions.

1) Performance: Table I summarizes the simulation results. The multi-synchronous implementation achieved a maximum clock frequency of 175 MHz.

The asynchronous architecture requires 80% less area and is three times as area-efficient as its synchronous counterpart – even under worst-case operating condition. However, it must be noted that a majority of the area occupied by the synchronous architecture is due to the amount of buffer memory needed to synchronize neighboring routers.

Under worst-case operating conditions, the asynchronous implementation delivers 60% lower latency compared to the synchronous one. The total latency of 5.5 clock periods of the multi-synchronous implementation encompasses the needed synchronization (average 2.5 clock periods), the arbitration, the output stage and an additional pipeline stage.

2) Link Delay: In the performance analysis, we have assumed ideal links between the routers of the network. However, link delay directly affects the performance of the asynchronous architecture. To determine this influence, we have measured the performance of the asynchronous router with links of several millimeters in length. The results are

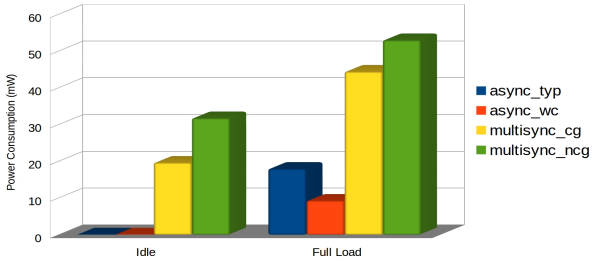

Figure 6. Power consumption of one router in idle state and under full utilization (adapted from [12]).

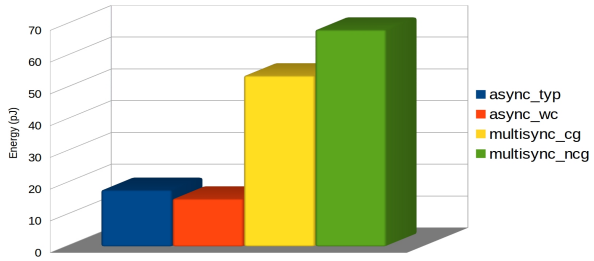

Figure 7. Average energy required to transfer a single flit through the router (adapted from [12]).

presented in Figure 5 and show a steady increase in latency. At approximately 2.5 mm, the critical cycle time of the router shifts from the inside of the router to the link between neighboring routers, thus reducing maximum throughput.

The influence of link delay has to be considered when implementing a complete network. If necessary, additional pipeline stages between neighboring routers can increase the throughput at the cost of latency.

3) Power Consumption: Power consumption of both implementations have been measured in idle state as well as under full utilization (five parallel connections). The multisynchronous architecture has been implemented with and without clock gating. The results are presented in figure 6 and 7.

In idle state, no packets are sent across the router. As expected [10], power consumption of the asynchronous implementation is negligibly low in this case (<1 µW), whereas the clock tree of the synchronous architecture still consumes a significant amount of power. Using clock gating allowed for reducing power consumption in idle state by 40%.

Under full utilization, the asynchronous architecture requires 60% less power than the clock gated synchronous implementation. The energy required to transfer a single flit through the router (figure 7) is reduced by 67% with respect to the clock gated architecture.

While power consumption of the asynchronous architecture is noticeably reduced under worst-case operating conditions, the energy required to transfer a single flit decreases only by a small amount, because the maximum throughput of the router decreases, too.

B. Measured Results

The proposed architecture of an asynchronous router has been prototyped as a complete network on a test chip in a 180 nm technology with a flit size of 8 bits and a single EOP bit. A synchronous design containing three packet generators and three receivers is connected to the network, which can be controlled via an SPI interface. The packet generators are able to send whole packets or just single flits across the network. Two request signals are connected to two bond pads to measure cycle time and latency of one selected router.

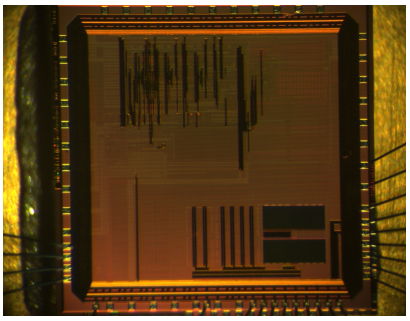

Figure 8. Chip photo of the test chip containing the measured NoC

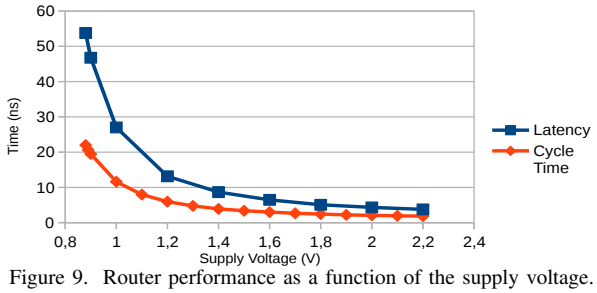

Figure 9. Router performance as a function of the supply voltage.

1) Performance: The performance of a single router has been measured at different supply voltages. The results are presented in Figure 9. As expected, the supply voltage has a direct influence on the performance of the asynchronous circuit. As the supply voltage increases, both latency and cycle time decrease and the overall performance of the router increases.

2) Power Consumption: Power consumption of a single router has been measured at different supply voltages, as well as with different injection rates of flits at each input port.

The results show that the overall power consumption (Figure 10) has a larger slope than the energy required to transfer a single flit across the router (Figure 11), because the maximum throughput of the router increases with the supply voltage (see previous Section).

Furthermore, the results indicate a linear correlation between the injection rate of flits per port and the corresponding

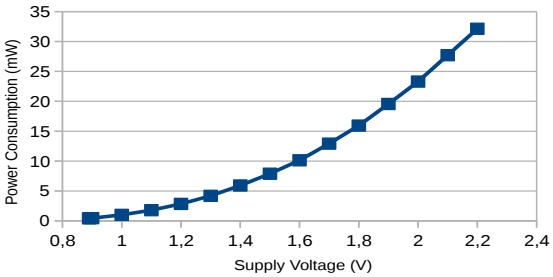

Figure 10. Power consumption of one router under full utilization and maximum injection rate of flits per port as a function of the supply voltage.

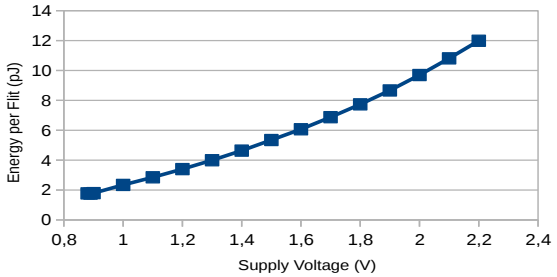

Figure 11. Average energy required to transfer a single flit through the router as a function of the supply voltage.

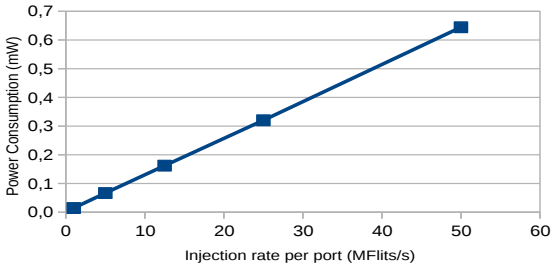

Figure 12. Power consumption of one router under full utilization as a function of the injection rate of flits per port (1.8 V supply voltage).

power consumption (Figure 12). Accordingly, the energy required to transfer a single flit across the network stays the same, independent of the router utilization (Figure 13). At very low injection rates the energy increases just a little bit, because of an increasing amount of static power consumption per flit.

VI. Conclusion

This paper proposes an implementation of an asynchronous router architecture for a GALS NoC. The proposed design flow is based on common CAD synthesis tools and enables the implementation of asynchronous bundled-data circuits with standard cells and a high compatibility with synchronous designs. The implemented router was evaluated in comparison to a synchronous implementation and was used to implement a complete network on a 180 nm test chip. The comparison shows significant resource reduction: 80% in area, 67% in energy per flit and 60% in latency. As expected, the asynchronous circuit shows negligible power consumption in idle state (<1 µW). Furthermore, the asynchronous architecture allows for the implementation with very little flit buffer memory at the input ports. Therefore, the implemented router is well suited for the requirements of a NoC in the context of a VSoC, where latency and energy efficiency are of paramount importance and the parallel utilization of the NoC is expected to be relatively low.

The measured results of the physically implemented network prove some very interesting properties of asynchronous circuits, such as resistance of the functionality against variations in power supply or the linear correlation between utilization and power consumption.

Acknowledgement

This work was supported by the German Federal Ministry of Education and Research (BMBF) within the Innovation Initiative "Entrepreneurial Regions", project consortium 3DSensation, project cSoC3D, grant number 03ZZ0427E. The authors of this paper are solely responsible for its content.

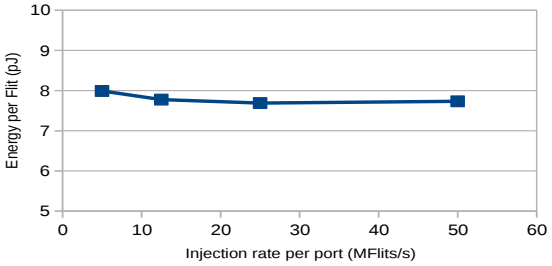

Figure 13. Average energy required to transfer a single flit through the router as a function of the injection rate of flits per port (1.8 V supply voltage).

References

[1] E. Beigné, F. Clermidy, P. Vivet, A. Clourad, and M. Renaudin. An Asynchronous NOC Architecture Providing Low Latency Service and Its Multi-Level Design Framework. In *11th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 54–63, March 2005.

[2] T. Bjerregaard and J. Sparsø. A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip. In *Design, Automation and Test in Europe*, pages 1226–1231 Vol. 2, March 2005.

[3] W. J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. In *Proceedings Design Automation Conference*, pages 684–689, June 2001.

[4] J. Döge, C. Hoppe, P. Reichel, and N. Peter. A 1 Megapixel HDR Image Sensor SoC with Highly Parallel Mixed-Signal Processing. In *International Image Sensor Workshop (IISW)*, Vaals, Netherlands, 2015.

[5] A. Ghiribaldi, D. Bertozzi, and S. M. Nowick. A Transition-Signaling Bundled Data NoC Switch Architecture for Cost-Effective GALS Multicore Systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 332–337, Grenoble, France, March 2013.

[6] M. Gibiluka. Design and Implementation of an Asynchronous NoC Router using a Transition-Signaling Bundled-Data Protocol, 2013. End of Term Work.

[7] M. Gibiluka, M. T. Moreira, N. Laert, and N. L. V. Calazans. A Bundled-Data Asynchronous Circuit Synthesis Flow Using a Commercial EDA Framework. In *Euromicro Conference on Digital System Design*, pages 79–86, August 2015.

[8] M. Iizuka, N. Hamada, H. Saito, R. Yamaguchi, and M. Yoshinaga. A Tool Set for the Design of Asynchronous Circuits with Bundled-data Implementation. In *29th IEEE International Conference on Computer Design (ICCD)*, pages 78–83, Amherst, USA, October 2011.

[9] A. J. Martin and M. Nystrom. Asynchronous Techniques for System-on-Chip Design. In *Proceedings of the IEEE (Vol. 94)*, pages 1089–1120, July 2006.

[10] S. M. Nowick and M. Singh. High-Performance Asynchronous Pipelines: An Overview. In *IEEE Design & Test of Computers (Vol. 28)*, pages 8–22, June 2011.

[11] S. M. Nowick and M. Singh. Asynchronous Design - Part 1: Overview and Recent Advances. In *IEEE Design & Test (Vol. 32)*, pages 5–18, March 2015.

[12] P. Russell. Entwurf und Implementierung eines asynchronen Netzwerkes für die schnelle Kommunikation auf grossen ASICs, October 2016. Diploma Thesis.

[13] A. Sheibanyrad, A. Greiner, and I. Miro-Panades. Multisynchronous and Fully Asynchronous NoCs for GALS Architectures. In *IEEE Design & Test of Computers (Vol. 25)*, pages 572–580, December 2008.

[14] M. Singh and S. M. Nowick. MOUSETRAP: High-Speed Transition-Signaling Asynchronous Pipelines. In *IEEE Transactions on Very Large Scal Integration (VLSI) Systems (Vol. 15)*, pages 684–698, June 2007.

[15] C. P. Sotiriou. Implementing Asynchronous Circuits using a Conventional EDA Tool-Flow. In *39th Design Automation Conference*, pages 415–418, New Orleans, USA, June 2002.

[16] P. Teehan, M. Greenstreet, and G. Lemieux. A Survey and Taxonomy of GALS Design styles. In *IEEE Design & Test of Computers (Vol. 24)*, pages 418–428, October 2007.