Two Efficient Synchronous ⇔ Asynchronous Converters Well-Suited for Network on Chip in GALS Architectures

A. Sheibanyrad and A. Greiner

The University of Pierre and Marie Curie
4, Place Jussieu, 75252 CEDEX 05, Paris, France
abbas.sheibanyrad@lip6.fr, alain.greiner@lip6.fr

Abstract. This paper presents two high-throughput, low-latency converters that can be used to convert synchronous communication protocol to asynchronous one and vice versa. These two hardware components have been designed to be used in Multi-Processor System on Chip respecting the GALS (Globally Asynchronous Locally Synchronous) paradigm and communicating by a fully asynchronous Network on Chip (NoC). The proposed architecture is rather generic, and allows the system designer to make various trade-off between latency and robustness, depending on the selected synchronizer. These converters have been physically implemented with the portable ALLIANCE CMOS standard cell library and the architecture has been evaluated by SPICE simulation for a 90nm CMOS fabrication process.

1 Introduction

NoCs (Network on Chip) are a new design paradigm for scalable, high throughput communication infrastructure, in Multi-Processor System on Chip (MP-SoC) with billions of transistors. The idea of NoC is dividing a chip into several independent subsystems (or clusters) connected together by a global communication architecture which spreads on the entire chip.

Because of physical issues in nanometer fabrication processes, it is not anymore possible to distribute a synchronous clock signal on the entire wide chip area. The Globally Asynchronous Locally Synchronous (GALS) addresses this difficulty. In this approach, each subsystem is a separated synchronous domain running with its own local clock signal.

Table 1. Timing Dependency Methods

Type	Δ Frequency	Δ Phase
Synchronous	0	0
Pseudochronous	0	Constant
Mesochronous	0	Undefined
Plesiochronous	ε	ε
Heterochronous	Rational	Undefined
Multi-synchronous	Undefined	Undefined
Asynchronous	-	-

J. Vounckx, N. Azemard, and P. Maurine (Eds.): PATMOS 2006, LNCS 4148, pp. 191 – 202, 2006.

Several solutions have been proposed to resolve the problems of clock boundaries between different clusters, and the risk of synchronization failures (metastability). The proposed solutions depend on the constraints that must be respected by the clock signals in different clusters: The mesochronous, plesiochronous, pseudochronous (or quasi synchronous) and heterochronous approaches correspond to various hypothesis regarding the phases and frequencies of clocks signals. Table 1 summarizes these conditions.

The Globally Pseudochronous Locally Synchronous scheme (GPLS) is proposed in [1]. In [2], [3] and [4] the authors have proposed plesiochronous solutions which rely on exact or nearly exact frequency and phase matching of the clocks. Mesochronous solutions are described in [4], [5] and [6]: it is argued that maintaining the same frequency in several clock domains is not too difficult. The main problem is the skew between clock phases. In heterochronous solutions ([4], [7], [8] and [9]) all clock signals can have different frequencies, but with fixed and integer ratios.

In MP-SoC design, a fundamental challenge is the capability of operating under totally independent timing assumptions for each subsystem. Such a multi-synchronous system contains several synchronous subsystems clocked with completely independent clocks.

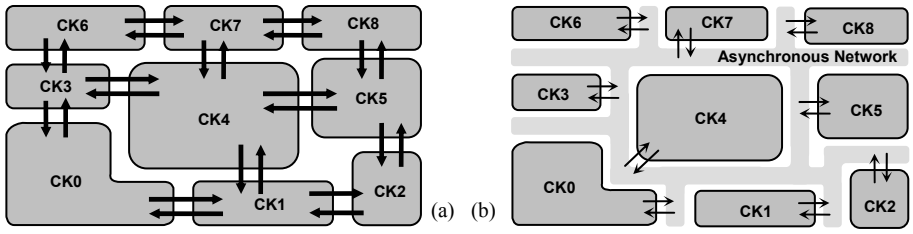

Fig. 1. Multi-Synchronous System in 2D Mesh Topology

In a two dimensional mesh respecting the multi-synchronous approach (Fig. 1a) we must solve the problem of communication between two neighbors clusters clocked with two fully asynchronous clocks. In [4], [5], [10] and [11] several authors have proposed different types of bi-synchronous FIFOs which can be used as a robust interface between neighbors. But this type of architecture implies one possibility of synchronization failure (metastability) at each clock boundary between neighbors.

We illustrate in Fig. 1b an alternative solution: The global interconnect has a fully asynchronous architecture. As an example we can denote the *MANGO* architecture presented in [26] which was one of the first asynchronous NOC. This type of NoC respects the GALS approach by providing synchronous ⇔ asynchronous interfaces to each local subsystem. In this case, the synchronization failure can only happen in the two synchronous ⇔ asynchronous converters located in the destination and source clusters. Providing a robust solution for those hardware interfaces is the main purpose of this paper. The *MANGO*'s designers have proposed in [27] an OCP Compliant Network Adapter (NA) interfacing local systems to NOC. The synchronization in NA has a minimized overhead.

The design of various architectures that can be used to interface an asynchronous domain with a synchronous one, are presented in [10], [14], [15], [16] and [17]. We present in this paper two converters architectures which can be used to convert a Four Phase Bundled Data asynchronous protocol to a synchronous FIFO protocol and vice versa. Using FIFOs to interface mixed timing systems couple two fundamental issues which need to be considered in designing such interfaces: flow control (high level issue) and synchronization (low level issue). This coupling reduces the need of hardware synchronizer to the handshake signals that are used for the flow control.

Some of the published solutions for interfacing asynchronous and synchronous domains are strongly dependent on synchronizer choice. In [15] and [16] the designs of various types of synchronizers using pausible clocking methods ([12]), are proposed and in [17] the authors have suggested to generate a stoppable clock for local systems.

In [14] a pipelined synchronizing FIFO is proposed. This FIFO requires that the producer produces data at a constant rate. The latency of this design is proportional to the number of FIFO stages and requires the use of a specific synchronizer.

In [10] various FIFOs are used to interface four possible combinations of independent mixed timing environments. These four FIFOs have the same basic architecture. The small differences in FIFO designs are simple adaptation to the consumer and the producer interface type. There is at least two weak points in this proposal: the architecture is dependent on a specific synchronizer (the cascaded Flip-Flops), and the use of a more conservative synchronizer (with latency larger than one clock cycle), can decrease the throughput to a value less than one data transfer per cycle. Furthermore, the authors of [10] didn't said anything about silicon area, but we believe that the architecture proposed in the present paper has a smaller foot-print.

We discuss the synchronizer issue in section 2. We present the general architecture of the synchronous ⇔ asynchronous converters in section 3. The detailed schematic is described in section 4. The system architecture is briefly illustrated in section 5. The hardware implementation is presented in section 6. The main conclusions are summarized in the last section.

2 Synchronizer: The Latency / Robustness Trade-Off

Transferring data between different timing domains requires safe synchronization. The main goal of the synchronization is the prevention of metastability (synchronization failure). The metastability can happens, for example, when an asynchronous signal A, connected to the input of a Flip-Flop controlled by a clock signal CK, is sampled when it has an intermediate value between VDD and VSS. It means when A doesn't respect Setup and Hold time to the sampling edge of CK; the Flip-Flop can enter a metastable state. As the duration of this metastable state is not predictable, the Flip-Flop output will have itself an asynchronous behavior and metastability can propagate in the synchronous part of the circuit, resulting generally in loss of functionality.

Some authors recommended stretching the clock signal (modifying the cycle time). In these methods, instead of synchronizing asynchronous inputs to the clock, the clock

is synchronized to the asynchronous inputs. The synchronizer must be able to detect that it will be in the metastable situation and it stretches the clock cycle of the local system until the probability of metastability is zero. For more than one asynchronous input, the clock must be stretched until all the synchronizers are sure that the metastable states don't occur. Consequently, as it is said in [7] and [18], these solutions are not well suited for high speed designs with IP cores having large clock buffer delays.

Some others suggested modifying the Flip-Flop design to avoid propagation of metastable state values ([19]). Although the output of such circuit has well defined values (VDD or VSS), and undesirable values are prevented to propagate, this does not solve the problem: the precise duration of the metastable state being not predictable. The transition of the Flip-Flop output signals is asynchronous compared with the clock signal of the next Flip-Flop...

The metastability in multi-synchronous systems can not be totally suppressed, but as it is explained in [21], the synchronization failure probability (typically expressed in terms of Mean Time Between Failures or MTBF) can be bounded to an acceptable value by a carefully designed synchronizer ([22] and [23]). The simplest and safest solution is to use several cascaded Flip-Flops. According to [21], with two cascaded Flip-Flops with 200 MHz clock frequency and 20 MHz input data rate, for the 0.18 μm technology MTBF can be estimated to about 10^{204} years. For three consecutive Flip-Flops in the same condition, MTBF will be 10^{420} years!

Increasing synchronization delay is a penalty for obtaining extra safety: When synchronization latency is not an issue, the MTBF can be improved by using conservative synchronizers. We believe that the synchronizer choice must be a design decision depending on the application requirements. The general architecture of our converter will support trade-off between latency and robustness.

3 General Architecture of the Converters

The design of any synchronous ⇔ asynchronous protocol converter must satisfy two main requirements: minimizing the probability of metastability, and maximizing the throughput (data transfer rate). Fig. 2 illustrates how these two aims can be achieved.

The Four-Phase, Bundled Data, asynchronous protocol, is a sequence of REQ+, ACK+, REQ- and ACK- events, where REQ and ACK are the asynchronous flow control signals. Data is valid when REQ is in the positive state. The high level of ACK indicates that the request of data communication is accepted.

In the synchronous FIFO protocol, the producer and the consumer share the same clock signal, and the protocol uses two handshake signals: ROK (correspondingly WOK) and READ (correspondingly WRITE). The ROK signal (or not empty) is set by the producer at each cycle where there is a valid data to be transferred. The READ signal is set by the consumer at each cycle where the consumer wants to consume a data on the next clock edge. Both the ROK and READ signals are state signals that can be generated by Moore FSMs.

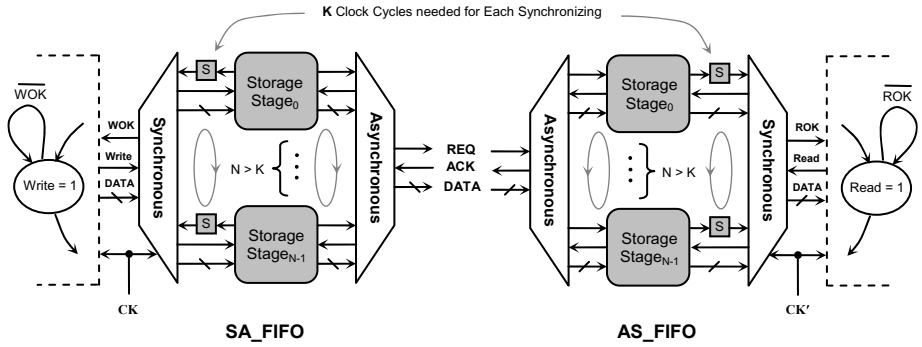

Fig. 2. Synchronizing FIFOs with Maximum Throughput

We call AS_FIFO, the asynchronous to synchronous converter, and SA_FIFO, the synchronous to asynchronous converter. The task of protocol converting is the responsibility of the storage stages of the FIFOs. The signals that have a risk of metastability (and must use a synchronizer) are the handshake signals transmitted from the asynchronous side to the synchronous side.

As it is said in previous section, the synchronizer design is a trade-off between robustness (i.e. low probability of metastability) and latency (measured as a number of cycles of the synchronous domain). If the synchronization cost is K clock cycles, the FIFO must have at least K+1 stages, if we want a throughput of one data transfer per cycle. In such pipelined design, the effect of synchronization latency is different in the two FIFO types. In the asynchronous to synchronous converter (AS_FIFO), it is visible only when the FIFO is empty. In the synchronous to asynchronous converter (SA_FIFO), it is visible when the FIFO is full. The latency between the arrival of data to an empty AS_FIFO and its availability on the output (typically named FIFO Latency) is about K clock cycles. For a full SA_FIFO, the latency between the consumption of a data and the information of the availability of an empty stage on the other side is about K clock cycles. For a data burst these latencies are just the initial latencies.

4 Detailed Architecture

The Fig. 3a and 4a show the internal architecture of the SA_FIFO and AS_FIFO converters, with a depth of 2 storage stages. Clearly, these two architectures can be generalized to n-stage FIFOs.

We present in Fig. 3b and Fig. 4b the FSM of the synchronous side controllers. These controllers are Mealy Finite State Machines. The state W_i means that the next WRITE event will be done to stage i. Similarly, the state R_i means that data will be read in stage i at the next READ event. Consequently, the WOK and ROK signals depend on both the FSM state and the asynchronous stage content (signals WOK_i or ROK_i). A synchronous Hazard free command is generated ($WRITE_i$ or $READ_i$) when there is a synchronous request (WRITE or READ signals) and the current asynchronous stage is ready to accept. ROK_i means that stage i is not empty. WOK_i

means that stage i is not full. The positive edge of $Write_i$ indicates to i^{th} asynchronous stage of SA_FIFO that the synchronous data must be written. The positive edge of $Read_i$ informs the i^{th} asynchronous stage of AS_FIFO that the stage must be freed. The positive edge of $wasRead_i$ means that the synchronous consumer has read data and the stage can change its value.

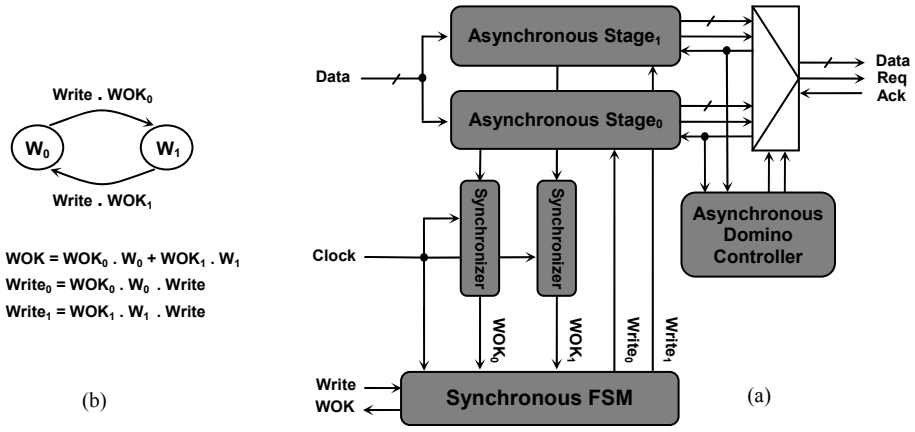

Fig. 3. Synchronous to Asynchronous Converter (SA_FIFO)

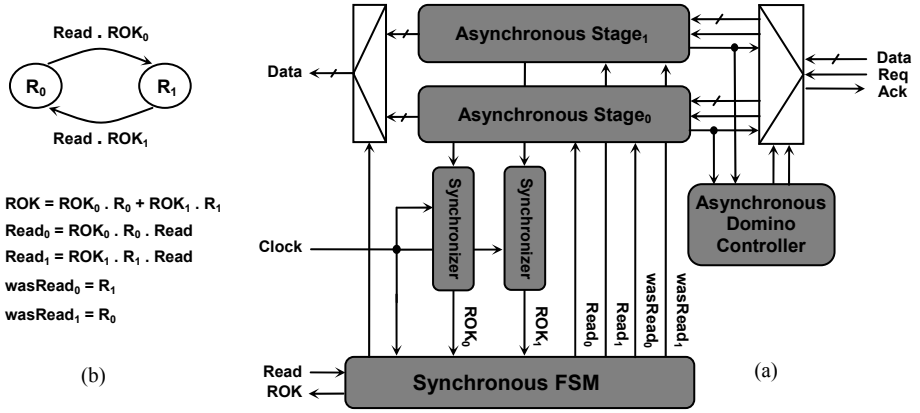

Fig. 4. Asynchronous to Synchronous Converter (AS_FIFO)

The asynchronous side of the design includes an asynchronous controller and an asynchronous Multiplexer in SA_FIFO. It includes an asynchronous controller and an asynchronous Demultiplexer in AS_FIFO. The design of the asynchronous Multiplexer and Demultiplexer using four phase bundled data protocol ([13]) are respectively shown in Fig. 5a and Fig. 5b. These circuits need to do a handshake with their controller module controlling the Select signals (S_i). This handshaking brings out with the sequence of S_i+, Ack_i+, S_i- and Acki- . After Ack_i- indicating the end of the

current four phase sequence, the controller can select another set to multiplexing or demultiplexing.

The asynchronous controller used in AS_FIFO and SA_FIFO is named Domino Controller. It is an asynchronous One-Hot counter providing required handshake protocol to the asynchronous multiplexer and demultiplexer. As an instance, the block diagram of a 3-bit Domino Controller is illustrated in Fig. 5c. Each cell of i has 2 outputs S_i and A_i (i^{th} bit of the counter) and 4 inputs Ack_{i-1}, Ack_i, Ack_{i+1} and A_{i-1}. The one bit is moved from cell to cell in a ring topology. At the initial state, A_2 and S_0 are 1 and the other outputs are 0. The High value of S_0 means that the first asynchronous event will be performed in $stage_0$.

The functionality of Domino Controller could be understood by looking the cell STG (Signal Transition Graph) demonstrated in Fig. 5d. The synthesized circuit of the STG is presented in Fig. 5e. Ack_i+ means S_i+ has done and is seen. So, the one bit that is in the previous cell (i-1) can be transferred to current cell. The handshake protocol continues by S_i- when the one transferring is ended.

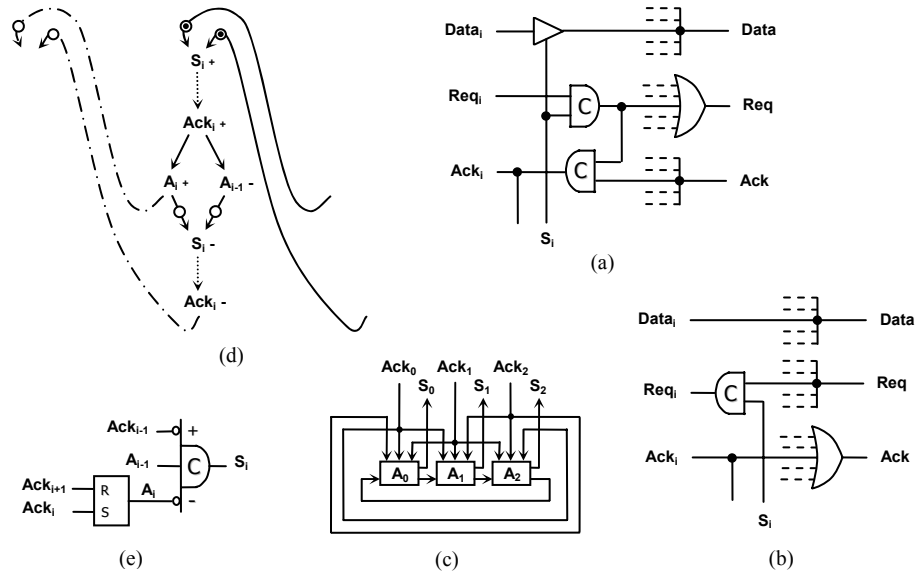

Fig. 5. Asynchronous Side Components

As we said before, the pipelined stages in AS_FIFO and SA_FIFO have two main functionalities: storing data and converting communication protocol. As it is demonstrated in Fig. 6a and Fig. 7a illustrating the schematics of the SA_FIFO and AS_FIFO stage circuits, data storage is done by the latches sampling on high value of WOK_i and of L. The transition to 0 of WOK_i means that this stage contains valid data and no more writing is permitted. So data sampling must be ended at this time. When the value of L on rising edge of $wasRead_i$ intending the content of the stage was read, is changed to 1 a new data can be written.

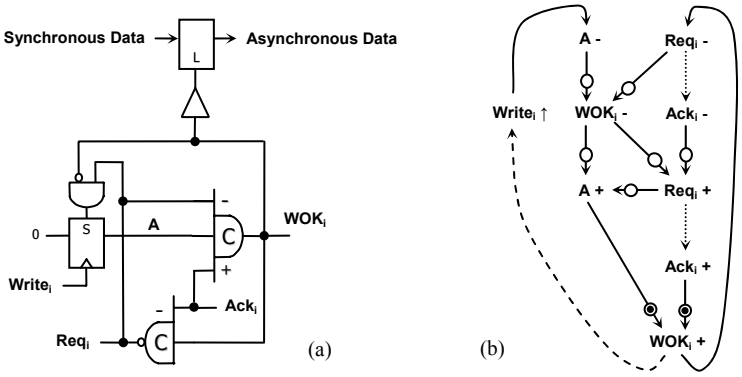

Fig. 6. Asynchronous Storage Stage of SA_FIFO

The operation of the SA_FIFO and AS_FIFO storage stages are analyzed as two STG in Fig. 6b and Fig. 7b. The dotted lines are the asynchronous side transitions and the dashed lines are that of the synchronous side. According to the synchronous protocol base, the synchronous side transitions should be considered on the edges. Regarding to two STG, on rising edge of $Write_i$, $Read_i$ and of $wasRead_i$ respectively, A, ROK_i and C must go to the low position. In the synthesized circuits three D Flip-Flops which have a constant value of 0 as input data, generate A, ROK_i and C. These Flip-Flops will asynchronously be set when their S input (Set) signal is 1.

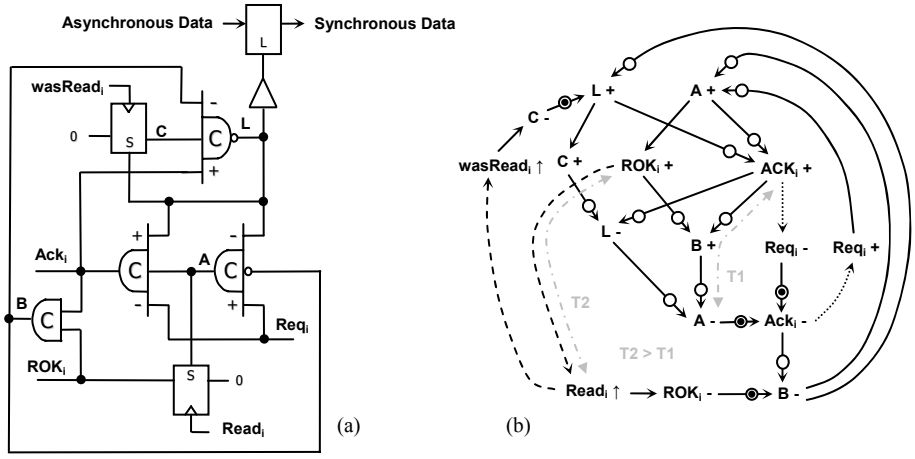

Fig. 7. Asynchronous Storage Stage of AS_FIFO

The synthesized circuit of the AS_FIFO stage shown in Fig. 7a has a time constraint: before rising edge of $Read_i$ where ROK_i- must be done, the value of A should be returned to 0; because, while A (as a set signal of Flip-Flop) has high value, ROK_i (as an output signal of the Flip-Flop) is hold at 1. The transition of ROK_i+ causes $Read_i$ to rise. Regarding to the AS_FIFO architecture (Fig. 4) the time between

ROK_i+ and rising edge of $Read_i$ (T2) is more than K clock cycles where K is the synchronizer latency. In the other side, A- happens after Ack_i+ occurring simultaneous with ROK_i+, by propagation delay of two gates (T1). Evidently a two gate propagation delay is less than the latency of a robust synchronizer. The latency of a two cascaded Flip-Flops is one clock cycle. But really it is true that if a designer uses a miraculous synchronizer (!) which has very low latency, this time constraint express a bother of functionality for the design.

5 System Architecture

As mentioned in the previous sections, the goal of this paper is to define a new design to robustly interface an asynchronous network to the synchronous subsystems on a chip. In the architecture presented in Fig. 8, SA_FIFO and AS_FIFO are instantiated between Network Wrapper and Asynchronous NOC. The Network Wrapper translates the local interconnect protocol (such as VCI or OCP) to the network protocol. The Network Interface Controller (NIC) is composed of one AS_FIFO, one SA_FIFO and one Network Wrapper. In fact, NIC provide local interconnect protocol compliant packets at the synchronous side and the asynchronous network compatible packets at the asynchronous ports. Required by Multisynchronous GALS approach, each subsystem may have its synchronous clock domain dependent neither on the frequency nor on the phase.

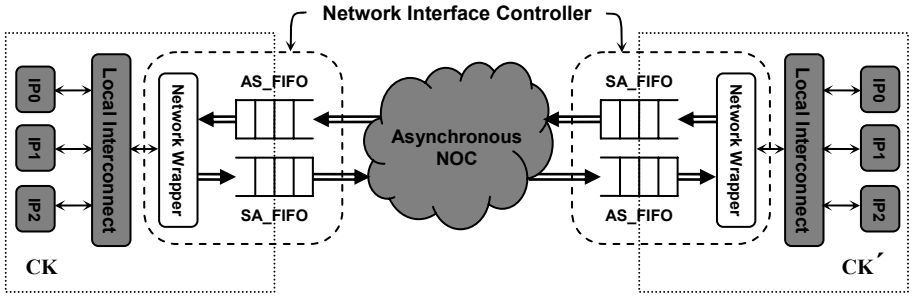

Fig. 8. AS_FIFO and SA_FIFO used in a Network Interface Controller

6 Implementation

We developed a generic converter generator, using the *Stratus* hardware description language of the *Coriolis* platform ([25]). This generator creates both a netlist of standard cells and a physical layout. The two parameters are the number of storage stages (depth of FIFO) and the number of data bits. In this implementation the synchronizer uses two cascaded Flip-Flops. As a standard cell library, we used the portable *ALLIANCE* CMOS standard cell library ([24]). The physical layout of the 32-bit converters with depth of 8, 3 and 2 stages are presented in Fig. 9. The silicon area of the 2-stage SA_FIFO is $12.15 \times 216\ \mu m^2$ for a 90 nm fabrication process.

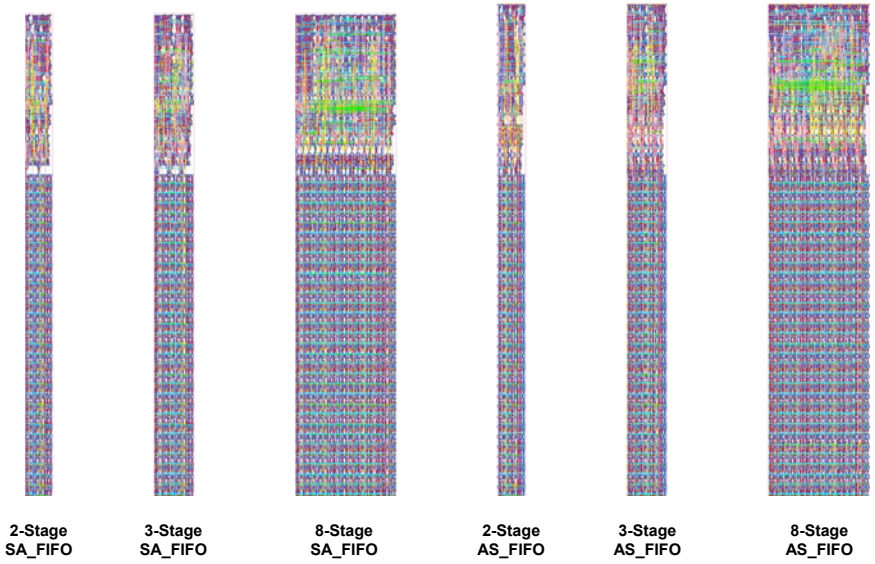

Fig. 9. Physical Layouts of Converters

From the physical layout, we extracted SPICE models of the converters, using *ALLIANCE* CAD Tools ([20]). The target fabrication process is the ST-Micro-electronics 90 nm LVT transistors in typical conditions. Electrical simulation under *Eldo* proved that the aim of maximum throughput of one event (data transfer) per cycle is attained, and these low-area FIFOs have low initial latencies. Due to the asynchronous event entrance time, the AS_FIFO has various latencies with a difference of about one clock cycle. The simulation results are presented in Table 2. In this Table, T is the clock cycle time.

Table 2. Simulation Results

Converter	Surface	Min Latency	Max Latency	Max Throughput
2-Stage SA_FIFO	2624 μm^2	177 pS		2.39 GEvents/S
3-Stage SA_FIFO	3791 μm^2	181 pS		2.36 GEvents/S
8-Stage SA_FIFO	9623 μm^2	192 pS		2.22 GEvents/S
2-Stage AS_FIFO	2679 μm^2	271 pS + T	271 pS + 2T	1.50 GEvents/S
3-Stage AS_FIFO	3870 μm^2	275 pS + T	275 pS + 2T	2.61 GEvents/S
8-Stage AS_FIFO	9823 μm^2	290 pS + T	290 pS + 2T	2.56 GEvents/S

The throughput value is related to the asynchronous handshake protocol. The low throughput value of 2-stage AS_FIFO compared with 3-stage and 8-stage AS_FIFOs, is because of existence of another constraint if maximum throughput of one word per cycle is required: In 2-stage AS_FIFO, Ack_i+ and $Req_{i+1}+$ must be happened in the same clock cycle. For 8-stage and 3-stage AS_FIFO, the time between these two transitions, respectively can be seven and two clock cycles.

Due to the inability of 2-stage AS_FIFO to reach the maximum throughput (comparing 1.5 GEvents/Sec with 2.61 of 3-stage AS_FIFO), in order to sustain the throughput, one could opt for 3-stage AS_FIFO. Its area (3870 μm^2) is not negligible, but it should not be forgotten that this component has another advantage: providing a storage place with a FIFO behavior. As we know, in order to obtain minimum overhead of data communication between two different timing domains, having a FIFO in the interface is not eliminable. So, we suppose that using an AS_FIFO or SA_FIFO with the storage stages of more than three may also be reasonable!

7 Conclusion

Two new converter architectures for interfacing asynchronous NoCs and synchronous subsystems in MP-SoCs have been presented. The designs can be used to convert asynchronous Four-Phase Bundled-Data protocol to synchronous FIFO protocol. The synchronizer used in the architectures can be arbitrarily chosen by the system designer, supporting various trade-off between latency and robustness. The Converters (FIFOs) can achieve the maximal throughput of one word per cycle, even if the selected synchronizer has a large latency. The designs have been physically implemented with the portable *ALLIANCE* CMOS standard cell library. The throughputs and latencies have been proved by SPICE simulation from the extracted layout.

References

1. Nilsson E., Öberg J., "*Reducing power and latency in 2-D mesh NoCs using globally pseudochronous locally synchronous clocking,*" 2nd IEEE/ACM/IFIP international Conference on Hardware/Software Codesign and System Synthesis (Stockholm, Sweden, September 08 - 10, 2004)
2. L.R. Dennison, W.J. Dally, D. Xanthopoulos, "*Low-latency plesiochronous data retiming,"* arvlsi, p. 304, 16th Conference on Advanced Research in VLSI (ARVLSI'95), 1995
3. W.K. Stewart, S.A. Ward, "*A Solution to a Special Case of the Synchronization Problem,"* IEEE Transactions on Computers, vol. 37, no. 1, pp. 123-125, Jan., 1988
4. Ajanta Chakraborty, Mark R. Greenstreet, "*Efficient Self-Timed Interfaces for Crossing Clock Domains,"* async, p. 78, 9th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'03), 2003
5. Yaron Semiat, Ran Ginosar, "*Timing Measurements of Synchronization Circuits"* async, p. 68, 9th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'03), 2003
6. Ran Ginosar, Rakefet Kol, "a*daptive Synchronization,"* iccd, p. 188, IEEE International Conference on Computer Design (ICCD'98), 1998
7. Joycee Mekie, Supratik Chakraborty, D.K. Sharma, Girish Venkataramani, P. S. Thiagarajan, "*Interface Design for Rationally Clocked GALS Systems,"* async, pp. 160-171, 12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'06), 2006

8. U. Frank, R. Ginosar, "*A Predictive Synchronizer for Periodic Clock Domains*," PATMOS 2004
9. L.F.G. Sarmenta, G.A. Pratt, S.A. Ward, "*Rational clocking [digital systems design],*" iccd, p. 271, IEEE International Conference on Computer Design (ICCD'95), 1995
10. S. M. Nowick, T. Chelcea, "*Robust Interfaces for Mixed-Timing Systems with Application to Latency-Insensitive Protocols,*" dac, pp. 21-26, 38th Conference on Design Automation (DAC'01), 2001
11. J. Jex, C. Dike, K. Self, "*Fully asynchronous interface with programmable metastability settling time synchronizer,*" US Patent 5 598 113, 1997
12. Kenneth Y. Yun, Ryan P. Donohue, "*Pausible Clocking: A First Step Toward Heterogeneous Systems,*" iccd, p. 118, IEEE International Conference on Computer Design (ICCD'96), 1996
13. Jens Sparsoe, Steve Furber, "*Principles of Asynchronous Circuit Design – A Systems Perspective,*" Kluwer Academic Publishers, 2001
14. Jakov N. Seizovic., "*Pipeline synchronization*," International Symposium on Advanced Research in Asynchronous Circuits and Systems, pages 87--96, November 1994
15. Simon Moore, George Taylor, Peter Robinson, Robert Mullins, "*Point to Point GALS Interconnect,*" async, p.69, 8th International Symposium on Asynchronous Circuits and Systems (ASYNC'02), 2002
16. David S. Bormann, Peter Y. K. Cheung, "*Asynchronous Wrapper for Heterogeneous Systems,*" iccd, p. 307, IEEE International Conference on Computer Design (ICCD'97), 1997
17. A. E. Sjogren, C. J. Myers, "*Interfacing Synchronous and Asynchronous Modules Within a High-Speed Pipeline,*" arvlsi, p.47, 17th Conference on Advanced Research in VLSI (ARVLSI '97), 1997
18. Rostislav (Reuven) Dobkin, Ran Ginosar, Christos P. Sotiriou, "*Data Synchronization Issues in GALS SoCs,*" async, pp. 170-180, 10th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'04), 2004
19. S. Ghahremani, "*Metastable Protected Latch,*" US Patent 6 072346, 2000
20. Greiner A., F. Pêcheux, "*ALLIANCE. A Complete Set of CAD Tools for Teaching VLSI Design,*" 3rd Eurochip Workshop on VLSI Design Training, pp. 230-37, Grenoble, France, 1992
21. Ran Ginosar, "*Fourteen Ways to Fool Your Synchronizer,*" async, p. 89, 9th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'03), 2003
22. C. Dike, e. Burton, "*Miller and Noise Effects in A synchronizing flip-flop,*" IEEE J. Solid-state circuits, 34(6), pp. 849-855, 1999
23. D.J. Kinniment, A. Bystrov, A.V. Yakovlev, "*Synchronization Circuit Performance,*" IEEE Journal of Solid-State Circuits, 37(2), p. 202-209, 2002
24. http://www-asim.lip6.fr/recherche/alliance/
25. http://www-asim.lip6.fr/recherche/coriolis/
26. Tobias Bjerregaard, Jens Sparsø, "*A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip," Proceedings of the Design, Automation and Test in Europe Conference*, IEEE, March 2005
27. Tobias Bjerregaard, Shankar Mahadevan, Rasmus Olsen, Jens Sparsø, "*An OCP Compliant Network Adapter for GALS-based SoC Design Using the MANGO Network-on-Chip,*" *Proceedings of the International Symposium on System-on-Chip*, IEEE, November 2005