Developing Mesochronous Synchronizers to Enable 3D NoCs

Igor Loi, Federico Angiolini, Luca Benini
DEIS, University of Bologna, Bologna, Italy.
{iloi, fangiolini, lbenini}@deis.unibo.it

ABSTRACT

The NETWORK-ON-CHIP (NOC) interconnection paradigm has been gaining momentum thanks to its flexibility, scalability and suitability to deep submicron technology processes. The next challenge is to use NoCs as the backbones of the upcoming generation of 3D chips, assembled by stacking multiple silicon layers. Multiple technical issues have to be tackled in this respect. One of the foremost is the unsuitability of a purely synchronous design style, as it is not straightforward to impose a strict bound on the clock skew among multiple clock trees across different layers. In this paper, we present a scheme to handle mesochronous communication in 3D NoCs and analyze (i) the circuit design, (ii) the timing properties, (iii) the requirements to support flow control across mesochronous links, (iv) the implementation cost of such a scheme after placement and routing.

1. INTRODUCTION

Over the years, advances in silicon technology have enabled the integration of larger and larger amounts of processing elements and memories into chips, increasing processing performance. This trend is still desirable, but new technological hurdles may prevent designers from being able to sustain the current pace of miniaturization. Simultaneously, there has been a strong push towards the mixing of functional blocks which may require a variety of processing steps, such as plain CMOS, DRAM, MEMS, passive and active analog circuitry, optoelectronic elements, chemical sensors, actuators, *etc.*. Unfortunately, each extra manufacturing step increases costs and decreases yield, imposing a limit on the heterogeneity of each silicon die. Vertically stacking multiple layers of silicon is an answer to both concerns: it represents a sustainable way of continuously adding functionality by the integration of more computing blocks, while pursuing design objectives such as ease of layer manufacturing, small package sizes, minimum footprints and modularity.

In planar implementations, interconnects are becoming a limiting factor to achieve design closure. This is due to several issues, such as the growing ratio of wire delay *vs.* logic delay, signal integrity concerns and stringent bandwidth requirements. At the system level, the key challenge is configuring, optimizing and verifying the communication architecture across many degrees of freedom in terms of topology, architecture and interface protocols. The NETWORK-ON-CHIP (NOC) paradigm, which brings packet-switching networking concepts to the on-die level, has been proposed [7, 5] to systematically tackle these challenges. NoCs are a structured, predictable and scalable approach to the problem, centered around wire segmentation and point-to-point signaling.

3D manufacturing and NoCs represent clearly synergistic techniques. The structured nature of NoCs ideally encapsulates the design properties and requirements (such as heterogeneous wiring resources and architecture, large degree of parallelism, high routing and topological complexity) of three-dimensional integration of two-dimensional building blocks. However, the implementation of 3D NoCs is not completely straightforward, as testified by the research papers on the topic that have been appearing recently [27, 8, 12, 9, 19].

One of the most obvious issues is the handling of clock domains. A completely synchronous approach to NoC design is feasible, even at frequencies around 1 GHz, in 2D chips [1]. On the other hand, minimizing the clock skew of a clock tree in a complex 3D structure becomes at least a challenging task. Two of the possible ways to implement a 3D clock tree are (i) the design of a separate clock tree per each 2D layer, and the insertion of a single vertical structure to which to attach all the tree roots, (ii) the design of a single clock tree in one of the layers, and the deployment of many vertical vias at the terminal nodes of this tree, thus distributing the clock to all the layers. According to [21, 22], solution (ii) would be better in terms of power and skew, but unfortunately at the price of an impractical number of vertical vias, severely impacting the cost and the modularity of the design (for instance, this solution does not readily apply to the very desirable scenario where 3D chips are assembled by stacking layers provided by different vendors and possibly built with completely different processes). On the other hand, solution (i) incurs major skew issues, which demand additional synchronization every time data is exchanged among layers. Since unfortunately there is no clear solution to the problem of skew-free and modular clock distribution in 3D chips, the need for clock synchronization at the inter-layer boundaries is well motivated.

Several possible solutions to this issue are available. Totally asynchronous NoCs are, unfortunately, very complex to design, validate and implement. Generic dual-clock FIFOs could be deployed; however, unless they are developed in full custom fashion (which implies design effort and portability drawbacks), their high implementation cost suggests using them only where absolutely needed. For example, instead of using them inside of the NoC topology for mere synchronization among clusters of routers, it may be wiser to instantiate them only at the edges of the NoC, *e.g.* at the interface of a core which is able to perform frequency scaling. To achieve functionality and flexibility at the minimum cost, mesochronous schemes are probably the most effective. This paper focuses on the implementation of mesochronous adapters for 3D NoCs, with emphasis on circuit design, timing properties, flow control support, and implementation cost. It is worth remarking that nothing prevents the proposed scheme from also being deployed in traditional planar designs.

2. RELATED WORK

A number of technologies for 3D chip manufacturing have been explored in recent years. In this paper we focus on wafer stacking approaches, as one of the most promising avenues for the implementation of high-performance yet inexpensive (multiple 3D chips can be processed in a single pass) three-dimensional ICs. Wafer stacking relies on Through-Silicon Vias (TSVs) [31] for vertical connectivity, guaranteeing low parasitics (*i.e.* low latency and power) and, if needed, extremely high densities of vertical wires (*i.e.* high bandwidth). Tezzaron Semiconductor Corporation [26] and IBM Technologies [33] are active players in this field.

NoCs have been suggested as a scalable communication fab-

978-3-9810801-3-1/DATE08 © 2008 EDAA

ric [7, 5]. CAD tools for NoC instantiation and optimization can be found for example in [23, 30]. The synthesis flow of NoCs has been explored by several groups, including full custom [15] (with actual test chips), FPGA targets [35, 34] and plain logic synthesis [2].

Some research is being undertaken on 3D NoCs. For example, in [27, 8], the authors focus on topologies (meshes, stacked meshes, *etc.*) and on performance metrics. In [12], the authors propose a dimension decomposition scheme to optimize the cost of 3D NoC switches, and present some area and frequency figures derived from a physical implementation. Post-silicon nano-scale 3D interconnections have also been recently investigated [9], but large scale availability of these technologies in the near future is uncertain.

A large body of research exists on asynchronous NoC design styles. For example, the CHAIN network [3] is completely based on clockless circuit design techniques. Other asynchronous NoC libraries include MANGO [6] and NEXUS [18]. ANOC [4] is based on a Quasi-Delay-Insensitive circuit design. Specific network building blocks are presented for example in [29] and asynchronous link design is tackled in [25].

The main goals of asynchronous NoCs have traditionally been lower power consumption than synchronous alternatives, increased tolerance to delay variability, and reduced electromagnetic emissions [11]. Despite all the research efforts, however, the actual physical implementations of asynchronous NoCs [32, 28] are few and limited in complexity (few millions of gates, $0.13\mu m$ technology). This is often attributed to the current lack of fully mature synthesis toolchains, simulation environments and testing infrastructures, hindering industrial implementations. Suitable component libraries are also very difficult to build and characterize.

GLOBALLY ASYNCHRONOUS LOCALLY SYNCHRONOUS (GALS) approaches do not disrupt as much the existing design flows. GALS systems [13, 14, 16] attach together a number of synchronous building blocks, and provide asynchronous facilities for the inter-block communication. While some of the tool maturity issues mentioned above still hold, the encapsulation of mixed-clock concerns within well-defined boundaries, which can be validated separately, provides a more conservative, and possibly more promising, solution to the interconnection issue. Several ways to synchronize clock domains at the boundaries exist, such as interleaving pipeline registers, using dual-clock FIFOs, adding programmable delays [17], deploying synchronous-to-asynchronous wrappers [13]. Although some of these solutions (for instance, dual-clock FIFOs) are very flexible, allowing for arbitrary clock frequencies in the sender and receiver domain, they all have one or more drawbacks, ranging from robustness to implementation complexity, from high latency to large area overhead. Some solutions have instead been specifically tuned only for the relatively simpler problem of mesochronous signaling, and have therefore been focused on low complexity and ease of implementation in existing tool flows. Two recent papers [10, 20] both suggest to implement the boundary interface with a source-synchronous design style, and propose some form of ping-pong buffering to counter timing and metastability concerns. We improve on these papers by studying such synchronizers inside of a NoC layout for a 3D chip, and considering full duplex communication with flow control, as discussed later.

3. REFERENCE NOC ARCHITECTURE

To make our study realistic, we integrate it on the ×pipes [2] NoC library. This NoC infrastructure is best suited for several reasons. First, it provides facilities for arbitrary switch connectivity, allowing the designer to easily deploy topologies for any kind of 3D arrangement of computing cores. Second, it already leverages a semi-automated design flow [24] spanning from RTL description to layout-level verification; this makes it possible for us to explore and validate the design down to the placement and routing steps. Third, it provides an interesting case study due to its configurability in terms of flow control. The ×pipes switches come in two radically different variants, conceived to best match two flow control protocols (see Figure 1). The first is ACK/NACK, a retransmission-based protocol featuring increased error resilience. The second is STALL/GO, a simple variant of credit-based flow control allowing

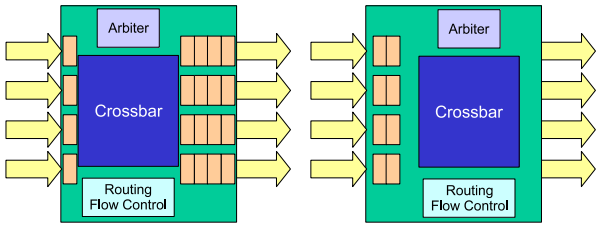


Figure 1: Block diagram of two switches, (a) with ACK/NACK (inputs and outputs are registered), (b) STALL/GO (only inputs are registered).

for pipelined links to be transparently deployed. In the ACK/NACK case, output buffers need to be inserted within switches, since any transmitted packet should be stored for potential retransmission. This implies a hardware cost, but it also means that NoC links are enclosed between two clocked buffers at the sending and receiving ends. Hence, a whole clock period is available for signal propagation along the wires of the inter-switch links. In any case, the link length and the switch logic are decoupled by the output buffer.

In contrast, in STALL/GO, low switching latency and reduced buffer cost are the main goals. The ×pipes STALL/GO switches therefore adopt a lean architecture, where only switch inputs are buffered. In other words, the switch logic and the link propagation time (up to the following switch or to the first link pipeline stage) contribute to the same timing path, which can become the bottleneck for the system.

4. ARCHITECTURE OF A MESOCHRONOUS SYNCHRONIZER FOR 3D NOCS

In this paper, we leverage the baseline architecture proposed in [10]. This choice features substantial pros, including minimal complexity, ease of implementation in traditional design flows, and ability to function even during chip testing (which is typically performed at a lower frequency than the target operating one). It is important to notice, however, that the reference paper is aimed especially at handling mesochronous communication over very long and slow links (where it provides variation tolerance and high performance as additional benefits), but does not focus on short-range mesochronous synchronization, such as the one likely to be happening across 3D NoC vertical links. Therefore, it does not provide a sufficiently in-depth discussion about two issues that are crucial for any such implementation:

- Timing margins, which are key to assessing circuit robustness and to the tuning of the low-level details of the design, are not studied in enough depth in a real NoC test case, therefore preventing the related optimizations.
- Support for bidirectional communication, *i.e.* for flow control, is lacking. Mesochronous signaling is useless if proper backwards flow control cannot be issued.

Exploring and quantifying the tradeoffs required by these features is clearly key to assessing the viability of the overall approach.

4.1 Circuit Description

The proposed scheme is based on a synchronization circuit at the receiving end of a mesochronous link (see Figure 2 for a slightly simplified depiction) [10]. The circuit receives as its inputs a bundle of NoC wires representing a regular NoC link, carrying data and/or flow control commands, and a copy of the clock signal of the sender. Since the latter wire experiences the same propagation delay as the data and flow control wires, it can be used as a strobe signal for them.

The circuit is composed of a *front-end* and a *back-end*. The front-end is driven by the incoming clock signal, and strobes the incoming data and flow control wires onto a set of parallel latches in a

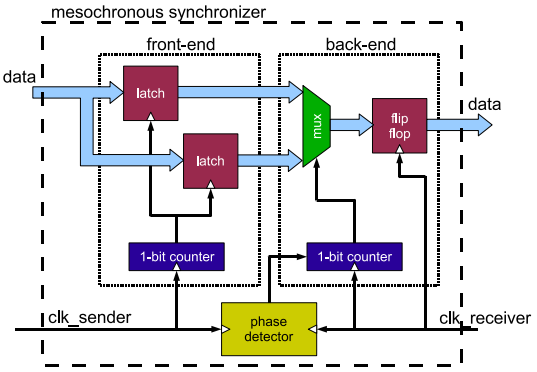


Figure 2: Proposed mesochronous synchronizer circuit.

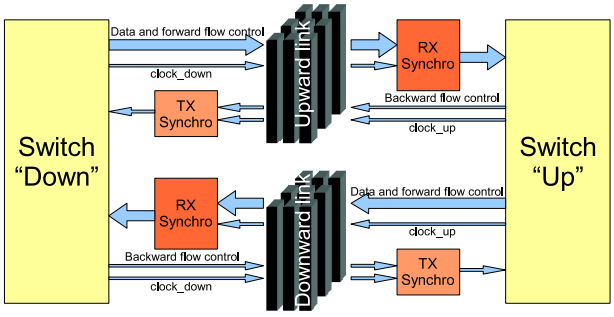


Figure 3: Proposed scheme for two-way synchronization across two layers.

rotating fashion, based on a counter. The back-end of the circuit leverages the local clock, and samples data from one of the latches in the front-end thanks to multiplexing logic which is also based on a counter. The rationale is to temporarily store incoming information in one of the front-end latches, using the incoming clock wire to avoid any timing problem related to the clock phase offset. Once the information stored in the latch is stable, it can be read by the target clock domain and sampled by a regular flip-flop. The counters in the front-end and back-end are initialized upon reset, after observing the actual clock skew among the sender and receiver with a phase detector [10], so as to establish a proper offset. This is to guarantee that information can safely settle in the front-end latches before being sampled on the target domain clock. The phase detector only operates upon the system reset, but given the mesochronous nature of the link, its findings hold equally well during normal operation; the advantage is that power consumption in normal mode is negligible.

With respect to the baseline scheme [10], we apply several changes, tuning the architecture to the problem at hand. One clear feature of the 3D NoC scenario, for example, is that vertical inter-switch links are typically short and feature extremely small propagation delays, in the range of tens of picoseconds [19]. Therefore, there is typically no need for the synchronizer to support multi-cycle propagation delays. As a result, one of the most notable architectural changes is the presence of only two latches, thus also dramatically simplifying the structure of the front-end and back-end counters to 1-bit elements. This change, which allows for large area savings, is allowed by the timing properties discussed in the following. Shall the need arise, more latches could still be deployed in case of a mesochronous link spanning over a very long distance, and requiring multiple clock cycles for signal propagation.

Figure 3 summarizes the intended configuration for a system with two layers and two vertical links, one going upwards, one going downwards. For each such link, one main synchronizer ("RX Synchronizer") must be deployed to adjust the incoming information to the new clock signal. Since few flow control wires are travelling backwards, a smaller "TX Synchronizer" is also needed to handle them. A single block, grouping the TX and RX synchronizers by the same side of the vertical link, could be designed; this would optimize the resource usage, for instance allowing to share the phase detector. Separate synchronizers on the other hand allow for the instantiation of unidirectional (*e.g.* upwards only) links without unneeded control logic.

4.2 Timing Margins of the Proposed Circuit

In order not to incur metastability and not to lose data within the mesochronous synchronizer, timing constraints must be met at two points in the circuit: (i) the front-end must latch incoming data safely, (ii) the back-end must sample incoming data when it is stable.

To fulfill condition (i), the latches must become transparent at the right point in time. The ideal control signal `latch_enable` to do so would be perfectly aligned with the strobe clock `clk_sender`, upon which `data` is designed to be sampled. Unfortunately, such an ideal condition is impossible to reproduce. First, `clk_sender` must be conditioned by local signals in the mesochronous synchronizer (namely, the output `count` of the front-end counter), which introduces a delay t_{cond}. Second, `clk_sender` and `data` may not be perfectly in sync any more if the vertical link among the sending switch and the mesochronous synchronizer is not ideal, *e.g.* if the wires/vertical vias carrying `clk_sender` are slower than those carrying `data` by $t_{routingskew}$. This means that `latch_enable` has a worst-case offset, with respect to the ideal edge on which `data` should be sampled, of $t_{cond} + t_{routingskew}$; this is an advance if $t_{routingskew}$ is negative and larger than t_{cond} (`clk_sender` wires much faster than `data` wires), and a delay otherwise.

On the other hand, the good news is that `data` is not supposed to be switching extremely close in time to the clock edges of `clk_sender`. Even if `data` were to be the direct output of registers in the sending switch, it would still take the propagation time of a flip flop before any transition could be noticed. In practice, it is likely that output buffers in the sending switch may also have some additional logic downstream of such registers, such as multiplexers to select the output of one of multiple buffer locations. Similarly, the `data` propagation delay must be designed to allow for at least a flip-flop setup time before the following clock edge, and probably a bit more to account for a bit of extra logic at the receiving buffer, such as multiplexers again. In general, the minimum transition delay of `data` after the previous clock edge of `clk_sender` can be called $t_{data_{min}}$, and the maximum can be called $t_{data_{max}}$.

In order to generate as robust a circuit as possible, we propose the circuit of Figure 4 to generate `latch_enable`; example waveforms are in Figure 5. This circuit is an improvement with respect to [10]. Since two latches are enough to implement the front-end (see below), the counter is 1-bit, and therefore a single flip-flop, while the logic to check the counter output against a fixed value becomes a single XOR. The circuit evaluates the counter output `count` on the positive edges of `clk_sender`, but only asserts `latch_enable` when `clk_sender` goes low, *i.e.* half a clock cycle later. This shortens the critical path among `clk_sender` edges and `latch_enable` edges, *i.e.* t_{cond}, to the delay of a single NOR gate, irrespective of the delay of the counter and comparison logic - as long as these fit within a clock semiperiod, which is trivial. With this arrangement, the latches in the front-end are only transparent for one clock semiperiod every two clock periods. The conditions for correct functionality can then be summarized as:

$$t_{cond} + t_{routingskew} + t_{latchhold} < t_{data_{min}} \quad (1)$$

$$t_{clk} + t_{cond} + t_{routingskew} > t_{data_{max}} + t_{latchsetup} \quad (2)$$

$$t_{counter} + t_{comp} < \frac{t_{clk}}{2} \quad (3)$$

Equation 1 expresses the fact that `latch_enable` should come early enough not to let the following piece of `data` slip in the front-end latch by mistake. Equation 2 ensures that `latch_enable` comes late enough to actually let `data` settle down before latch-

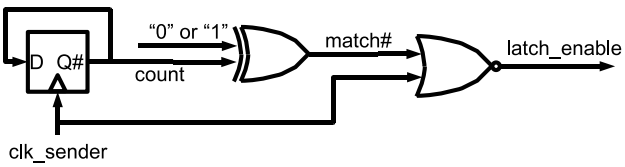


Figure 4: Circuit to generate the `latch_enable` control wire.

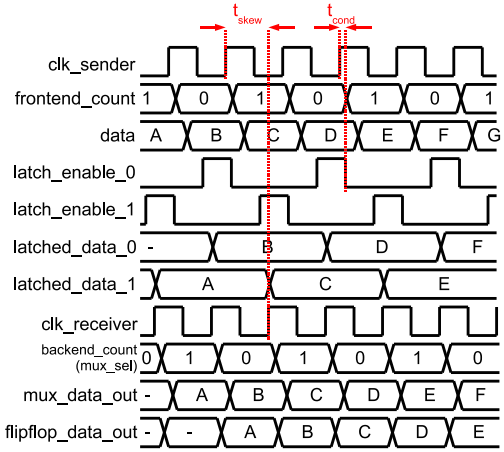


Figure 5: Example of the waveforms in the proposed synchronizer.

ing it in the front-end. Finally, Equation 3 ensures that the critical path for the generation of `latch_enable` is indeed determined by the edges of `clk_sender` plus a NOR delay. Experimental results validating that these equations are actually holding will be presented in Section 5.2.

Condition (ii) is easy to fulfill given a proper initialization of the counters at reset. It is a degree of freedom whether to have the back-end sample data from the upper latch at "even" clock edges and the lower latch at "odd" clock edges, or vice versa, based on the initial value imposed to the back-end counter during reset. Since the latches in the front-end are transparent one semiperiod every two periods, and opaque (frozen) for the remaining three semiperiods, it is always possible to choose a counter setup where the sampling clock edge in the back-end captures the output of the latches in a stable condition, even accounting for a large timing margin to neutralize jitter. Please note that this discussion also proves that no more than two latches in parallel are needed in the front-end, at least as long as the link propagation time remains shorter than a single clock period.

4.3 Adding Support for Backwards Flow Control

A key open issue to understanding whether the circuit can be used to implement a useful link for a 3D NoC is to check the overhead it mandates for a design with flow control. In fact, a unidirectional mesochronous link is relatively straightforward to design; once bidirectional communication must be taken into account, the implementation details and the related resource overhead become crucial. We see two properties that the system must feature to define a proper implementation of flow control over mesochronous links: (i) the system must never incur data loss or corruption, (ii) if the receiver is not busy for independent reasons (such as contention for the same switch output port), the system must be able to sustain a transfer bandwidth of one flit per clock cycle.

The solution to be applied depends on the flow control deployed in the platform, but is anyway based on the main observation that the maximum added time to convey flow control signals across a vertical link, and to resynchronize them, is in any case less than two clock cycles. Based on this information, the following solutions

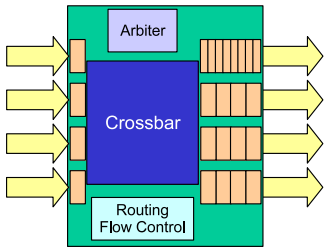


Figure 6: ACK/NACK modified switch block diagram. The port upstream of the vertical link has a deeper output buffer.

can be envisioned.

4.3.1 *Backwards Flow Control in ACK/NACK*

In the ACK/NACK flow control, in absence of flow control information heading back (either ACKs or NACKs), the sender "optimistically" pushes flits out. Since a copy of each flit must be stored locally, the maximum number of outstanding flits is as many as the output buffer can hold. When flow control information is eventually received, in case of NACKs, old flits are resent; if, on the other hand, it is an ACK which makes its way back to the sender, an old flit can be discarded from the output buffer, and a new one can be stored and sent.

Strictly speaking, the ACK/NACK flow control protocol does not require any corrective action to handle the timing changes introduced by a mesochronous synchronizer. The synchronizer merely delays the reception of flow control signals; this introduces no critical change in behaviour, and data safety is still guaranteed. However, changes need to be performed to support maximum bandwidth over the mesochronous link. The added latency of two clock cycles on each way (forwards and backwards) means that flits will reach their destination two cycles later, and ACKs will bounce back four cycles later than normal. To cope with this condition, output buffers in the sender need to be extended by four entries, *e.g.* from four (the minimum buffer depth to support maximum throughput in normal circumstances) to eight. This does not require any architectural change; a parameter adjustment in the output buffer is sufficient. A block diagram of the modified ACK/NACK switch is presented in Figure 6. The area cost related to this change will be presented in Section 5.3. No changes are required to the receiving switch, at the other side of the vertical link, unless a link in the opposite direction is also desired.

4.3.2 *Backwards Flow Control in STALL/GO*

In STALL/GO, flits are sent only as long as the STALL feedback wire is deasserted. This has two implications, which are the opposite of the ones for the ACK/NACK case. On the one hand, if STALLs are never injected by the receiver, the sender never receives them, and full transmission bandwidth can always be sustained; no circuit change is needed to meet this criterion. On the other hand, data safety is critical. STALLs are the only way the receiver can withhold the flow of flits from the sender in case they cannot be processed (such as in case of lost arbitration for a switch output port). If STALLs cannot reach the sender in time, namely within one clock cycle, flits leaving the sender while the receiver is busy simply get lost.

To cope with this situation, we extend regular input buffers by two entries (from two, which is the minimum to provide full bandwidth, to four) and change their control logic. Instead of raising the STALL wire when the buffer is actually full, we raise it when two locations are still available. This approach is conservative; for example, a 4-deep STALL/GO buffer could in principle operate forever and at full bandwidth with three or four of its locations full, provided that, at each clock cycle, a flit can be extracted to make room for a new incoming one. However, if the same buffer were to be this full and were to experience further downstream congestion, there would simply be no way to notify the sender in time and to store the flits in flight anywhere. Thus, we choose instead to raise STALLs in advance, so that, by the time the sender is notified of the congestion, at most two flits are in flight, and they can still

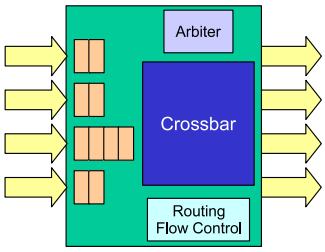
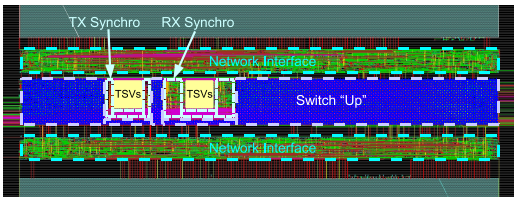
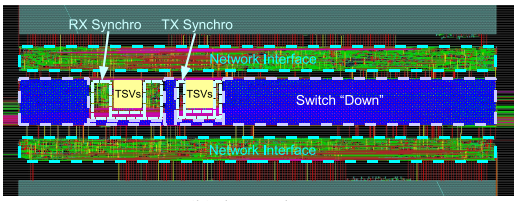


Figure 7: STALL/GO modified switch block diagram. The port downstream of the vertical link has a deeper input buffer and modified control logic.



(a) upper layer



(b) lower layer

Figure 8: Layout of a 3D chip stack with a mesochronous NoC link. Bundles of 7x7 vias are laid among the layers, supporting 32-bit links plus flow control connections, and leaving some spare vias.

be stored. A block diagram of the modified STALL/GO switch is presented in Figure 7. The area cost related to this change will be presented in Section 5.3. No changes are required to the sending switch, at the other side of the vertical link, unless a link in the opposite direction is also desired.

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

5.1 Example Layout of Mesochronous Link Implementation

We synthesize the proposed circuit with the UMC $0.13\mu m$ technology library and insert it into a 3D chip stack floorplan, then perform routing. Figure 8 summarizes the result. It is possible to see the layout of the upper and lower layers, each featuring a switch and two NIs. Two obstructions model the vertical vias (one "Up" link and one "Down") interconnecting the layers. The RX and TX synchronizers are wrapped around the via bases, and are swapped among the layers. This layout is found to be very area-efficient.

5.2 Timing Properties of the Mesochronous Synchronizer Front-End

In Section 4.2, a set of conditions to be fulfilled for proper operation have been presented. Our experiments on a post-routing netlist show the following:

- Equation 1 is easily fulfilled. Thanks to our optimized design (Section 4.2), we measure t_{cond} values of about $60ps$. The propagation time skew among different wires of a NoC link is very low, typically yielding a $t_{routingskew}$ below $20ps$. The typical latch hold time $t_{latchhold}$ is roughly $60ps$. On the other hand, for our NoC, we measure a $t_{data_{min}}$ of about $370ps$, irrespective of whether the flow control is ACK/NACK or STALL/GO. The constraint is therefore fulfilled. (Please note that $t_{data_{max}}$, however, is dependent on the chosen flow control due to the reasons explained in Section 3, and can be of up to $900ps$, imposing an operating frequency of $1GHz$ at most).

- Equation 2 poses no issue. This is because the condition $t_{clk} > t_{data_{max}} + t_{latchsetup}$ is automatically met by any fully synchronous circuit. On the other hand, the term $t_{cond} + t_{routingskew}$, which appears because of the mesochronous synchronizer logic, never becomes negative in any of our test layouts. In other words, the propagation time difference among `data` and `clk_sender` is normally negligible, and in no case `clk_sender` is so much faster than `data` so as to more than offset t_{cond} (also see the bullet above). Therefore Equation 2 is always verified in our tests. Please note that, even in case of a violation of this condition, the circuit could still be made to work safely by slightly increasing t_{clk}, *i.e.* slightly decreasing the operating frequency.

- Equation 3 is fulfilled by a very large margin. The typical clock period of our reference NoC is larger than $1ns$ in $0.13\mu m$ technology, yielding a semiperiod of at least $500ps$. Given the simplicity of the counter and comparator logic for a front-end with just two latches, we observe $t_{counter} + t_{comp}$ times of less than $200ps$, well within the desired range.

Given these results, the proposed architecture proves robust under all circumstances.

5.3 Silicon Cost of Proposed Synchronizer and Related Flow Control Adjustments

Figure 9 summarizes the area overhead for the implementation of the proposed mesochronous synchronizer. The numbers refer to a post-routing circuit model. The "Synchronous" baseline comprises the area of two 32-bit 5x5 switches, with the minimum buffering required for sustaining maximum throughput under no congestion, and that of the vertical obstruction required for a unidirectional vertical link (counted twice: once per chip layer). The "Mesochronous" figures add the area overhead for supporting mesochronous clocking over such a link, namely, the buffer depth increase in one of the switches and the two TX and RX Synchronizers. It is possible to notice that the synchronizers themselves feature minimal overhead, thanks to the drastic simplification in logic allowed by the implementation of 2-latch front-ends. The RX Synchronizer is about five times larger than the TX Synchronizer, since it must handle many more wires. The largest area overhead for mesochronous clocking support is within the switches themselves, and, as expected, is mostly accounted for in the sequential area budget. ACK/NACK incurs a much larger penalty than STALL/GO, since four extra buffers have to be deployed instead of just two. Overall, the global area overhead is about 13% of the baseline configuration for STALL/GO and about 40% of the baseline configuration for ACK/NACK. Especially in the STALL/GO scenario, the area cost is minimum and the implementation seems to be clearly affordable.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have shown a detailed implementation of a mesochronous synchronizer for usage in a three-dimensional chip with a NoC backbone. In this context, since completely synchronous designs are hard or impossible to achieve, such a device is key to correct functionality. Starting from a baseline circuit scheme, we have customized it, verified its timing properties, added flow control facilities on top of the basic circuit, and assessed the area overhead of the whole. Key advantages of the baseline circuit have been kept, such as simplicity and ability to operate correctly even during chip test at a low frequency. The experimental results show that the proposed scheme is robust and that its area cost is

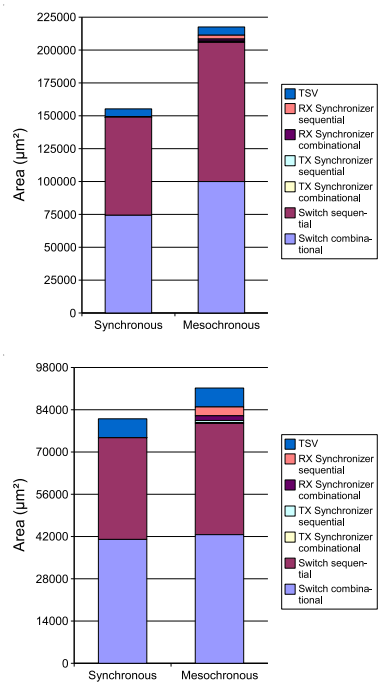


Figure 9: Area cost to implement mesochronous synchronization, (a) with ACK/NACK, (b) with STALL/GO.

minimal, proving the viability of this architecture for 3D chip implementations based on NoCs.

Future work to be performed includes more detailed comparisons against alternative schemes and against full synchronizers, such as dual clock FIFOs. Power overhead does not seem to be a concern given the simplicity of the circuit and the short length of the vertical vias to be traversed, but its quantification is nonetheless scheduled as future work.

Acknowledgments

This work is supported by a grant from Semiconductor Research Corporation (SRC project number 1188) and a grant by STMicroelectronics for DEIS.

7. REFERENCES

[1] F. Angiolini, P. Meloni, S. Carta, L. Raffo, and L. Benini. A layout-aware analysis of networks-on-chip and traditional interconnects for MPSoCs. *IEEE Transactions on Computer-Aided Design*, 26(3):421–434, March 2007.

[2] F. Angiolini, P. Meloni, S. Carta, L. Raffo, and L. Benini. A layout-aware analysis of networks-on-chip and traditional interconnects for mpsocs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(3):421–434, March 2007.

[3] J. Bainbridge and S. Furber. Chain: a delay-insensitive chip area interconnect. *IEEE Micro*, 22(5):16–23, Sep/Oct 2002.

[4] E. Beigne, F. Clermidy, P. Vivetand A. Clouard, 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, 2005.

[5] Luca Benini and Giovanni De Micheli. Networks on chip: a new SoC paradigm. *IEEE Computer*, 35(1):70–78, January 2002.

[6] Tobias Bjerregaard and Jens Sparsø. Scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip. In *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 34–43, 2005.

[7] William J. Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference*, pages 684–689, June 2001.

[8] Brett Feero and Partha Pratim Pande. Performance evaluation for three-dimensional networks-on-chip. In *Proceedings of the IEEE Annual Symposium on VLSI (ISVLSI)*, pages 305–310. IEEE Computer Society, 2007.

[9] S. Fujita, K. Nomura, K. Abe, and T.H. Lee. 3d on-chip networking technology based on post-silicon devices for future networks-on-chip. In *Nano-Networks and Workshops*, pages 1–5, September 2006.

[10] Maged Ghoneima, Yehea Ismail, Muhammad Khellah, and Vivek De. Variation-tolerant and low-power source-synchronous multicycle on-chip interconnection scheme. *VLSI Design*, 2007, 2007.

[11] Eckhard Grass, Frank Winkler, Milos Krstic, Alexandra Julius, Christian Stahl, and Maxim Piz. Enhanced gals techniques for datapath applications. In *PATMOS*, pages 581–590, 2005.

[12] Jongman Kim, Chrysostomos Nicopoulos, Dongkook Park, Reetuparna Das, Yuan Xie, N. Vijaykrishnan, Mazin S. Yousif, and Chita R. Das. A novel dimensionally-decomposed router for on-chip communication in 3d architectures. In *Proceedings of the 34th International Symposium on Computer Architecture (ISCA)*, 2007.

[13] M. Krstic, E. Grass, C. Stahl, and M. Piz. System integration by request-driven gals design. *IEE Proceedings on Computers and Digital Techniques*, 153(5):362–372, September 2006.

[14] D. Lattard and et al. A telecom baseband circuit-based on an asynchronous network-on-chip. In *Proc. of the International Solid State Circuits Conference, ISSCC2007*, pages 258–259, 2007.

[15] Kangmin Lee, Se-Joong Lee, Sung-Eun Kim, Hye-Mi Choi, Donghyun Kim, Sunyoung Kim, Min-Wuk Lee, and Hoi-Jun Yoo. A 51mW 1.6GHz on-chip network for low-power heterogeneous SoC platform. In *Digest of Technical Papers of the 2004 IEEE International Solid-State Circuits Conference (ISSC)*, pages 152–518. IEEE Computer Society, 2004.

[16] Kangmin Lee, Se-Joong Lee, and Hoi-Jun Yoo. Low-power network-on-chip for high-performance soc design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(2):148–160, February 2006.

[17] Se-Joong Lee. Cost-optimization and chip implementation of on-chip network. Technical report, KAIST, 2005.

[18] A. Lines. Asynchronous interconnect for synchronous soc design. *IEEE Micro*, 24(1):32–41, Jan-Feb 2004.

[19] Igor Loi, Federico Angiolini, and Luca Benini. Supporting vertical links for 3d networks-on-chip: Toward an automated design and analysis flow. In *Proceedings of the Nano-Net Conference 2007*, 2007.

[20] D. Mangano, R. Locatelli, A. Scandurra, C. Pistritto, M. Coppola, L. Fanucci, F. Vitullo, and D. Zandri. Skew insensitive physical links for network on chip. In *1st International Conference on Nano-Networks (NanoNet)*, pages 1–5, 2006.

[21] Christopher A. Mineo. Clock tree insertion and verification for 3D integrated circuits. Technical report, North Carolina State University at Raleigh, 2005.

[22] Mosin Mondal, Andrew J. Ricketts, Sami Kirolos, Tamer Ragheb, Greg Link, N. Vijaykrishnan, and Yehia Massoud. Thermally robust clocking schemes for 3D integrated circuits. In *Proceedings of the 2007 Design, Automation and Test in Europe conference (DATE)*, pages 1206–1211, New York, NY, USA, 2007. ACM Press.

[23] Srinivasan Murali, Martijn Coenen, Andrei Radulescu, Kees Goossens, and Giovanni De Micheli. Mapping and configuration methods for multi-use-case networks on chips. In *Proceedings of the 2006 conference on Asia South Pacific design automation (ASP-DAC)*, pages 146–151, New York, NY, USA, 2006. ACM Press.

[24] Srinivasan Murali and et al. Designing message-dependent deadlock free networks on chips for application-specific systems on chips. In *VLSI-SoC*, pages 158–163, 2006.

[25] E. Nigussie, J. Plosila, and J. Isoaho. Delay-insensitive on-chip communication link using low-swing simultaneous bidirectional signaling. In *Proceedings of the 2006 Emerging VLSI Technologies and Architectures (ISVLSI06)*, page 217, 2006.

[26] Robert S. Patti. Three-dimensional integrated circuits and the future of system-on-chip designs. *Proceedings of the IEEE*, 94(6), June 2006.

[27] Vasilis F. Pavlidis and Eby G. Friedman. 3-D topologies for networks-on-chip. In *Proceedings of the IEEE SoC Conference (SOCC)*, pages 285–288. IEEE Computer Society, 2006.

[28] L. A. Plana, W. J. Bainbridge, and S. B. Furber. The design and test of a smartcard chip using a CHAIN self-timed network-on-chip. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, page 274, 2004.

[29] Dobkin Rostislav, Victoria Vishnyakov, Eyal Friedman, and Ran Ginosar. An asynchronous router for multiple service levels networks on chip. In *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 44–53, 2005.

[30] K. Srinivasan and K.S. Chatha. A methodology for layout aware design and optimization of custom network-on-chip architectures. In *Proceedings of the 7th International Symposium on Quality Electronic Design (ISQED)*. IEEE Computer Society, 2006.

[31] S.Spiesshoefer and et al. Z-axis interconnects using fine pitch, nanoscale through-silicon vias: Process development. In *Electronic Components and Technology Conference*, 2004.

[32] Mikkel B. Stensgaard, Tobias Bjerregaard, Jens Sparso, and Johnny Halkjaer Pedersen. A simple clockless network-on-chip for a commercial audio DSP chip. In *Proceedings of the 9th EUROMICRO Conference on Digital System Design*, pages 641–648, 2006.

[33] A. W. Topol, Jr. D. C. La Tulipe, L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, and M. Ieong. Three-dimensional integrated circuits. *IBM Journal of Research and Development*, 50(4/5):491–506, July/September 2006.

[34] Pascal T. Wolkotte, Philip K.F. Holzenspies, and Gerard J.M. Smit. Fast, accurate and detailed noc simulations. In *Proceedings of the First International Symposium on Networks-on-Chip (NOCS)*. IEEE Computer Society, 2007.

[35] Cesar Albenes Zeferino and Altamiro Amadeu Susin. SoCIN: A parametric and scalable network-on-chip. In *Proceedings of the 16th Symposium on Integrated Circuits and Systems Design (SBCCI03)*, pages 34–43, 2003.