Reducing NoC Energy Consumption Exploring Asynchronous End-to-end GALS Communication

Iaçanã Weber*, Leonardo de Oliveira†, Everton Carara†, Fernando G. Moraes*
*School of Tecnhology - PUCRS – Av. Ipiranga 6681, 90619-900, Porto Alegre, Brazil
iacana.weber@edu.pucrs.br, fernando.moraes@pucrs.br
† UFSM - Federal University of Santa Maria
leonardo@mail.ufsm.br, carara@ufsm.br

***Abstract*—Systems-on-Chip (SoCs) with a large number of cores adopt Networks-on-chip (NoCs) as the communication infrastructure due to its scalability. The complexity to distribute a skew-free synchronous clock signal over the entire chip increases in current fabrication technologies due to the process variability. Thus, designers may choose among fully asynchronous and Globally Asynchronous, Locally Synchronous (GALS) NoCs. This work proposes an intermediate solution. Each Intellectual Property (IP) core may have its clock domain, and the NoC supports both synchronous and asynchronous communication. The NoC has its own clock domain, with the synchronous communication used to establish end-to-end paths. Once the end-to-end path is established, the NoC connects the IPs asynchronously, as wires with repeaters. The message is transmitted using handshake protocol at the source IP frequency. The benefit of the proposal is the reduction in the communication energy consumption (up to 52%). The cost is a latency increase (16% to 30%) and area overhead (4%).**

***Index Terms*—NoC, GALS, asynchronous communication, clock domain crossing.**

I. Introduction

GALS design is a technique where each IP core has its own clock domain. This approach assumes that it is not feasible to distribute a global clock signal with acceptable skew, but it is still achievable within a limited region, named clock domain. The communication between two clock domains requires synchronizers among each domain to avoid metastability [1]. The literature presents examples of GALS NoCs [2], where each router is in the same clock domain as its IP. The problem with this approach is the need for successive synchronizations starting at the source router up to the target one, increasing the communication latency between IPs. Fully asynchronous NoCs may also be used. The drawback of asynchronous NoCs is the design complexity and the lack of Electronic Design Automation (EDA) tools.

Our *goal* is to present an end-to-end GALS approach to solve those problems, avoiding synchronizers in the paths, using a synchronous NoC with support for packet- and circuit-switching (PS and CS). The path establishment and release use PS, while messages are transmitted using CS, bypassing all routers' buffers. Thus, our NoC can directly connect two IPs, each one in its own clock domain, providing an asynchronous communication path, based on an asynchronous handshake protocol. The NoC operates at lower frequencies than system IPs decreasing the portion of energy spent in clock distribution.

Computation requirements of current applications, such as machine learning, computer vision, autonomous driving and big data, demand huge computational power, leading to a further increase in power densities and dark silicon issues. The constant increase in power density leads to thermal issues, that are known for generating reliability and aging problems in the chip. Experiments show that at 22 nm, 21% of the chip must be dark to meet power restrictions, and may increase to 50% for 8 nm technology [3].

The *motivation* for using synchronous NoC operating at lower frequencies than system IPs and supporting direct connections, comes from the following observations [4]: (*i*) the IPs' injection rate is low (less than 10%) and in bursts, favoring the asynchronous transmission; (*ii*) NoCs consume 20 to 30% of the SoCs energy [5][6], thus, by reducing the buffers' depth and the router frequency, the overall SoC consumption is also reduced due to the reduction on the switching activity; (*iii*) in nanoscale technologies, signals may traverse several hops using wires with repeaters.

The *original contribution* of this paper is the proposed method mixing synchronous and asynchronous communication for reducing the communication energy consumption.

This paper is organized as follows – Section II reviews methods presented in the literature that exploit the buffer bypass technique. Section III and Section IV detail the NoC communication protocol and the NI asynchronous circular FIFO that supports the asynchronous end-to-end communication. Section V presents the results in terms of area, energy, and latency. Section VI concludes this paper.

II. Related Work

The buffer bypass approach has been explored in different works [7][8], aiming performance gains and energy consumption reduction. Perez et al. [7] propose a mechanism that can be applied to wormhole and virtual-cut-through, each of them with different advantages. The method selects the best flow control for each packet situation, maximizing the bypass utilization. Their results show a reduction in latency and dynamic power up to 30% and 23%, respectively. Kodi et al.

978-1-7281-9625-1/20/$31.00

[8] present a combination of two techniques, adaptive channel buffers, and router pipeline bypassing, aiming to reduce the energy consumption and improve performance. The technique presented a power reduction of 62% over the baseline and improved performance by more than 10%.

Jain et al. [9] propose a multi-synchronous NoC (each router operating at the IP frequency), which avoids the synchronization overhead at the intermediate nodes via an Asynchronous Bypass Channel (ABC) around these nodes. Limitations of this approach include: (*i*) when a packet needs to make a direction turn (e.g., east to north), it must be latched, increasing the overall latency; (*ii*) since there is no connection establishment (our approach), routers employ bi-synchronous FIFOs to store packets in the event of congestion, also increasing the transmission latency.

Krishna et al. [10][11] propose a low-swing link circuit using clockless repeaters. They replace conventional links with SMART (Single-cycle Multi-hop Asynchronous Repeated Traversal) links at design time. However, it is required prior knowledge of the application communication to configure the paths that will use the SMART links. Also, if packet collision occurs, it is necessary to store packets in buffers. Pérez et al. [12] propose SMART++, witch reduces the cost and improve efficiency of the original approach. Daya et al. [13] propose the SCEPTER NoC, a bufferless version of SMART. SCEPTER uses deflection routing when a flit is blocked due to the congestion in the links.

Stensgaard et al. [14] present the Reconfigurable NoC (ReNoC), with the goal to enable reconfiguration of the network topology. ReNoC contains a conventional NoC router, wrapped by a topology switch. According to the implemented logical topology, some routers are bypassed by the topology switches. The reconfiguration is static, and a logical topology configuration must ensure that the delay of the longest logical link does not exceed a clock period.

Lines [15] presents Nexus, an asynchronous crossbar that connects synchronous IPs through asynchronous channels. Each IP can have an independent clock frequency. Clock domain converters connect the asynchronous crossbar to each synchronous IP. The drawback of crossbar interconnections is the quadratic area growth, sacrificing scalability.

From one side the bypass approach is adopted in NoC designs, but on the other side, buffers or deflection routing are used to deal with congestion. We assume a different approach by mixing switching modes. At runtime the NoC establishes connections, using the routing algorithm to determine the path, with no need of prior knowledge of the application. In case of congestion, the routers' buffers only need to store the header flit, resulting in shallow buffers. Once the path is established, the NoC clock domain no longer impacts the communication, since it becomes a dedicated link between the source and target IPs. Due to the asynchronous handshake protocol employed in the end-to-end communication, the NoC frequency does not limit the number of hops in a path, which makes the approach scalable. Our proposal also addresses the clock domain crossing drawback that still is a concern in the GALS paradigm [16][17].

III. NoC Design

The NoC, named Arke [18], adopts a 2D-mesh topology. Figure 1 presents the router main blocks and interfaces. Each port has an *input buffer* for flit storage during PS communication. The router has a control logic (*switch control*) that implements the routing algorithm and arbitrates new connections between input and output ports, and a *crossbar* to connect an input port to an output port. Each transmitted packet has a header flit with the communication type (synchronous or asynchronous) and the packet destination address. After the header, it follows the payload with the last flit signaled by the control signal *eop* (end-of-packet). Once transmitted the last flit, the router releases the internal connection.

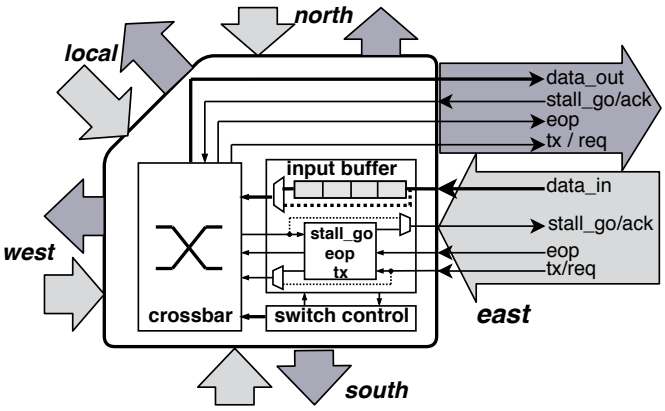

Fig. 1. Arke router. Dotted signals are the bypass paths, used for asynchronous communications. All ports have the same structure presented for the East port.

The synchronous communication mode employs wormhole PS, using the stall-and-go flow control protocol. This method uses a buffer in the receiver side and a return signal (*stall_go*) to inform if there is available buffer space. Valid data on *data_in* bus is signalized by the transmitter to the receiver through the *tx* signal.

The asynchronous communication employs a CS connection with three phases: (*i*) establishment, (*ii*) transmission, and (*iii*) release.

The establishment phase starts with the source IP injecting the header into the local router, which stores it waiting for the routing process and crossbar configuration. Next, the input buffer and control signals are set to *bypass state* (dotted signals in Figure 1). Once bypassed the input buffer and control signals, the source IP becomes connected directly to the next router in the path. This process is repeated up to the target IP. At the end of this phase, the source and target IPs are connected, bypassing all input buffers in the path.

During the transmission phase, the IPs communicate through a 2-phase Non-Return-to-Zero bundle-data protocol. Signals *tx* and *stall_go* assume the role of *request* and *acknowledgment*, respectively. In order to transmit data, the source IP first makes a flit available (*data_in*), followed by a transition in the request signal (*tx/req*). The target IP read

the flit and answers with a transition in the *stall_go/ack* signal. After receiving the acknowledgment, the source IP may send a new flit.

The connection release phase starts with the last packet flit. As can be seen in Figure 1, the *eop* signal is not bypassed, which means that it is synchronized with the NoC clock domain and transmitted synchronously, in the same way as the header. Upon receiving the *eop* signal, the router starts the process of disconnecting the path. The disconnection follows the same process of the first phase, one router at a time, and to avoid losing the last flit, it is stored at each disconnected router until it reaches the target IP.

Hazards are not a concern in synchronous design because data and control signals are sampled only after stabilization and clock synchronization. On the other hand, they are harmful in asynchronous designs since a single signal transition can change the state of the circuit.

Attention was given to avoid hazards during the asynchronous protocol. The post logical synthesis simulation with annotated delay revealed that the table controlling the crossbar connections does not update all code bits at the same time. As a consequence, during the table updating, hazards were observed in *req* and *ack* signals, leading to unwanted requests and acknowledgments. The solution was the adoption of one-hot encoding to guarantee the stability of *req* and *ack* signals during the crossbar switching transitions. Also, hazards were identified during the establishment phase, being generated by the final router in the path as depicted in the Figure 2.

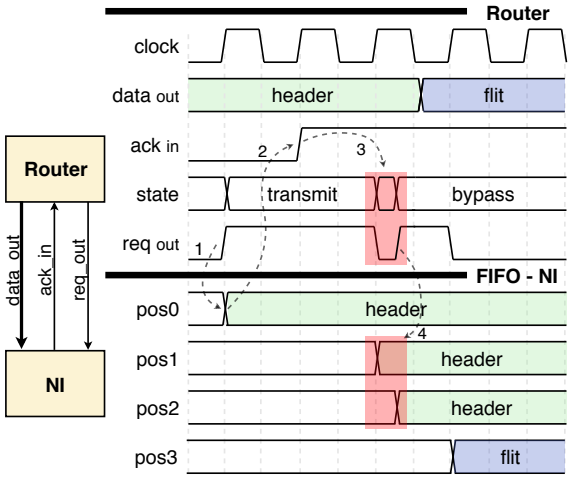

Fig. 2. Hazard demonstration in the NoC output.

The router is directly connected to the asynchronous circular FIFO (inside the NI, presented in the next section), which in turn connects to the target IP. Once the router delivers the header to the NI (event 1 in Figure 2), an acknowledgment is sent confirming the flit reception (2) and the last router changes to the state of bypass (3) starting the end-to-end communication. At this point, hazards occurs in the req_{out} signal, leading to unwanted writing in the asynchronous circular FIFO (4). To solve this problem, we changed the regular multiplexer responsible for the req_{out} signal by an anti-hazard multiplexer and also set the states changing (that controls the multiplexer) so that each change only switches one bit of state code, avoiding static, dynamic and functional hazards.

IV. Network Interface (NI)

Instead of a traditional bi-synchronous FIFO, the NI employs an asynchronous circular FIFO which allows IPs to bypass the NoC clock domain once the end-to-end connection is established. The communication throughput is bounded by the slower communicating IP since the NoC buffers in the path are bypassed. Our proposal needs buffering only at the receiver's side, contrasting with the bi-synchronous FIFO that requires buffering in both sender and receiver sides, in order to enable cross-domain clock between IP→NoC (sender) and NoC→IP (receiver). Figure 3 presents the NI architecture.

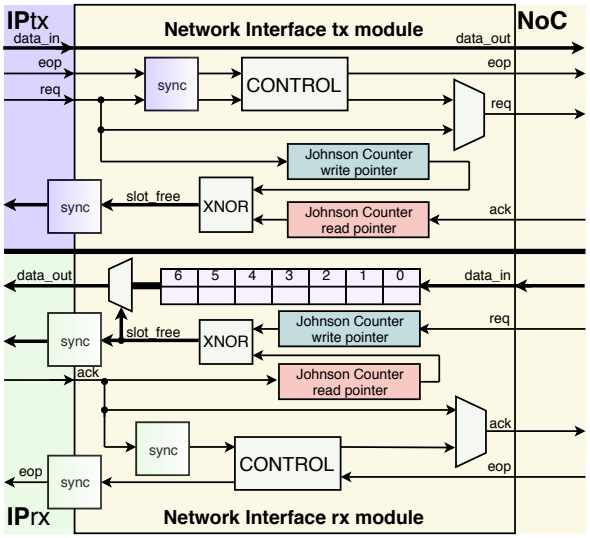

Fig. 3. Architecture of Network Interface transmitter and receiver modules.

The receiver NI side (Figure 3 bottom) employs a FIFO based on Mousetrap pipeline stages [19]. Data and its corresponding requests are latched while waiting for the IP reading. The latches implementing the FIFO slots are standard level-sensitive D-type. Johnson counters are used as read and write pointers. The write counter is triggered by transitions on the *req* signal generated remotely by the transmitter while transitions on the *ack* signal, generated locally by the receiver, triggers the read counter. An XNOR operation between the read and write pointers is used to indicate the storage status. Free slots ($slot\_free$ bus) are indicated by '1' (transparent latch) and occupied ones indicated by '0' (opaque latch). Based on the $slot\_free$ bus, a multiplexer selects which slot is the next one to be read by target IP.

During the transmission phase, the transmitter IP writes directly into the receiver's FIFO transitioning the *req* signal. To know the receiver's FIFO occupation and avoid overwrites, the transmitter module has the same aforementioned Johnson counters. In the transmitter module, the read counter is triggered remotely by the receiver IP through the *ack* signal and

the write counter is triggered locally through the *req* signal. The same process occurs at the receiver's end, where the write counter is triggered remotely by the transmitter IP through the *req* signal and the read counter is triggered locally through the *ack* signal.

The FIFO depth is equal to seven. This depth was chosen so that in a scenario with two IPs working at the same clock frequency but different phases, could communicate at maximum rate (one flit per cycle). The FIFO's main function is actually to maximize the communication throughput keeping an uninterrupted flow of flits. Since the FIFO's control signals (*req*, *ack*, *slots_free*) are asynchronous, they must be synchronized at the IPs clock domain. Such synchronization is based on the traditional pair of flip-flops. To cover the entire round trip latency caused by the synchronization, there must be seven positions in the FIFO, which consists in: one source IP writing cycle, two cycles for request synchronization at target IP *(slots_free)*, one target IP cycle for reading and another two cycles for acknowledgment synchronization at source IP *(slots_free)*. Thus, considering that both communicating IPs are on the same frequency, when source IP is writing in the seventh FIFO position, the acknowledgment for the first flit will be arriving. Hence, in the next clock cycle, the IP can write a new flit without interrupting the data transmission. Note that, the source IP does not need to wait for the acknowledgment from each flit before request the next one, since the Johnson counters keep the information about the target's FIFO occupation.

The presented asynchronous infrastructure covers only the data transmission phase. The NI performs the connection establishment and closure phases synchronously operating in the NoC clock domain.

V. Results

Results obtained in this Section comes from a gate-level simulation, with the netlist generated by logic synthesis. Such a procedure considers delays for asynchronous communications and switching activity.

The first experiment evaluates the *latency* behavior as a function of the packet size, varying the NoC frequency, and using asynchronous communication. Figure 4 presents the latency results considering one pair of communicating IPs operating at the same frequency. The dotted horizontal lines present the normalized latency w.r.t. the NoC using the synchronous protocol latency (*baseline*) - IPs and NoC at the same frequency. The solid curves present the normalized latency for communication using the asynchronous protocol with IPs operating at the baseline frequency, and NoC's frequency ranging from one (NoC1) to one fifth (NoC1/5) times the *baseline* frequency (e.g., NoC1/5 means: IPs at 1 GHz and NoC at 200 MHz).

The asynchronous communication latency has an asymptotic behavior w.r.t. to the packet size. For small packets, the impact on establishing/releasing CS paths is higher than for large packets. In particular, when the relationship between IPs and NoC frequency is equal to two (*NoC1/2*), the latency overhead

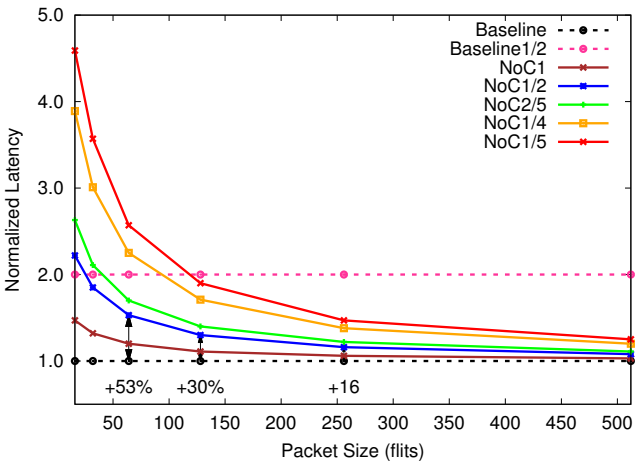

Fig. 4. Synchronous and asynchronous latency, varying the packet size and the NoC frequency w.r.t. the IP frequency.

is 30% and 16% for 128- and 256- flit packets (typical packet size for a cache refill packet). If IPs run at the NoC frequency, the asynchronous communication overhead is 6% for 256-flit packets. *Summarizing, IPs operating at the double of the nominal frequency (NoC1/2), with packets ranging from 64 to 256 flits, the latency impact is in the range of 16-53%.*

To highlight the advantage of the asynchronous communication, it is worth mentioning that the latency doubles (overhead equal to 100%) with the synchronous NoC running at half of the nominal frequency. For example, suppose a scenario where IPs and NoC run at 1 GHz. In this case, both baseline and Arke will operate using the synchronous protocol (the asynchronous protocol is available, but it will always present latency penalty w.r.t. the synchronous in this situation), with the same latency to transmit packets. Eventually, for example, due to a temperature violation, the system changes the NoC frequency to 500 MHz to reduce the energy consumption. In this situation, the baseline NoC latency penalty is 100%, as presented in the violet dotted line (Baseline1/2), doubling the amount of time required to deliver a packet. On the other hand, the system with Arke can use the asynchronous protocol that will provide better results, with the latency varying according to the packet size as presented by the blue line (NoC1/2).

The second experiment evaluates the NoC *energy* for a 4x4 NoC, with IPs sending packets to random targets with random sizes (varying from 16 to 512 flits). The NoC frequency is constant, while the IPs' frequency increases – Figure 5. We obtained the energy values using the switching activity generated after logic synthesis simulation (CADENCE tools).

The dotted horizontal line corresponds to the baseline synchronous NoC, with buffer depth equal to 4. Its consumed energy does not change with the increase of the IP's frequency. In this case, the consumed energy is constant (*Baseline*) due to the bi-synchronous FIFO's limitation that converts the flit injection generated at IP's frequency to the NoC frequency (constant in the experiment). Using asynchronous communication, besides four-slot buffers (*b4*), we also present

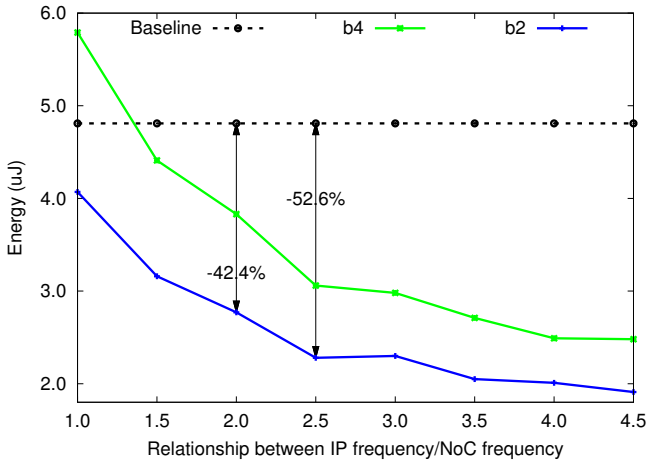

Fig. 5. Synchronous and asynchronous energy consumption.

results considering two-slot buffers (*b2*) since the buffer depth does not affect the latency due to the bypass. The NoC energy consumption decreases with the reduction in the buffer depth due to the reduction in the gate count. There is also a reduction in energy consumption for packet transmissions because buffers are bypassed, so there is no switching activity related to the temporary flit storage. For a 2-slot buffer, the energy reduces by 42.4% and 52.6% considering IPs operating 2.0 and 2.5 faster than Arke. Note that the 42.4% on energy reduction corresponds to a latency overhead around 16% (Figure 4). Such latency overhead is for 128 and 256 flit packets, and Figure 5 considers a broader packet size range (16 to 512 flits).

Table I presents the router and NI *area* results for logic synthesis using an IBM 180 nm standard cells library. The asynchronous NI (27,448 μm^2) presented an area reduction of 42% when compared to the bi-synchronous NI (47,860 μm^2). This result is expected due to the fact that the bi-synchronous NI requires two buffers: one to cross the clock domain from source IP to NoC and one to cross the clock domain from NoC to target IP. In the other hand the proposed NI requires only one buffer at the target IP. The area overhead in the router, for the same buffer depth, reaches 28.2% due to the addition of circuitry to avoid hazards, logic elements to control the new communication technique and the asynchronous bypass channel. The pair router-NI presents an area overhead of 4% for a buffer depth equal to 4, but when using a buffer depth equal to 2 the pair area is reduced by 2.2%. The comparison using the pair router-NI is valid due to the fact that each router typically has a NI associated. Also it is important to highlight that the proposed router supports both communication protocols (synchronous and asynchronous).

TABLE I
ROUTER AND NI AREA (μm^2), IBM 180 nm. *b2*/*b4* STANDS FOR INPUT BUFFER DEPTH EQUAL TO 2 AND 4, RESPECTIVELY. EG: EQUIVALENTE GATE (NAND2 AREA).

Module	Sync. – baseline	Async. – Arke
NI	47,860 (3,179 EG)	27,448 (1,823 EG)
Router - *b4*	89,603 (5,953 EG)	114,836 (7,629 EG)
Router - *b2*	64,839 (4,307 EG)	81,964 (5,445 EG)

ACKNOWLEDGEMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. Fernando Gehm Moraes supported by FAPERGS (17/2551-0001196-1 and 18/2551-0000501-0) and CNPq (302531/2016-5).

VI. CONCLUSION

This paper proposed the Arke NoC, with support to synchronous (PS) and asynchronous (CS) communication modes. Results demonstrated important energy savings (50%) with moderate latency overhead for medium packets (128-256 flits). Even if there are in the literature works with direct links between IPs, the proposed approach is, in the Authors' knowledge, the first work using a hybrid switching mode to leverage energy gains. The proposal also benefits from the synchronous design approach, enabling to synthesize the NoC and NI with standard EDA tools. Future works include physical design implementation, exploration of power and clock gating during asynchronous transmission and keep the connection established for a set of packets, instead of creating it for each packet.

REFERENCES

[1] R. Ginosar, "Metastability and Synchronizers: A Tutorial," *IEEE Design & Test of Computers*, vol. 28, no. 5, pp. 23–35, 2011. [Online]. Available: https://doi.org/10.1109/MDT.2011.113

[2] E. Beigné and P. Vivet, "Design of On-chip and Off-chip Interfaces for a GALS NoC Architecture," in *ASYNC*, 2006, pp. 172–183. [Online]. Available: https://doi.org/10.1109/ASYNC.2006.16

[3] H. Esmaeilzadeh, E. R. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark Silicon and the End of Multicore Scaling," *IEEE Micro*, vol. 32, no. 3, pp. 122–134, 2012. [Online]. Available: https://doi.org/10.1109/MM.2012.17

[4] V. Y. Raparti and S. Pasricha, "RAPID: Memory-Aware NoC for Latency Optimized GPGPU Architectures," *IEEE Trans. Multi-Scale Computing Systems*, vol. 4, no. 4, pp. 874–887, 2018. [Online]. Available: https://doi.org/10.1109/TMSCS.2018.2871094

[5] H. Cheng, J. Zhan, J. Zhao, Y. Xie, J. Sampson, and M. J. Irwin, "Core vs. uncore: the heart of darkness," in *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*. ACM, 2015, pp. 121:1–121:6. [Online]. Available: https://doi.org/10.1145/2744769.2747916

[6] M. Moulika, M. Vinodhini, and N. Murty, "Data flipping coding technique to reduce NOC link power," in *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE, 2017, pp. 1–6.

[7] I. Perez, E. Vallejo, and R. Beivide, "Efficient Router Bypass via Hybrid Flow Control," in *NoCArc@MICRO*, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/NOCARC.2018.8541147

[8] A. K. Kodi, A. Louri, and J. M. Wang, "Design of energy-efficient channel buffers with router bypassing for network-on-chips (NoCs)," in *ISQED*, 2009, pp. 826–832. [Online]. Available: https://doi.org/10.1109/ISQED.2009.4810399

[9] T. N. K. Jain, M. Ramakrishna, P. V. Gratz, A. Sprintson, and G. Choi, "Asynchronous Bypass Channels for Multi-Synchronous NoCs: A Router Microarchitecture, Topology, and Routing Algorithm," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 30, no. 11, pp. 1663–1676, 2011. [Online]. Available: https://doi.org/10.1109/TCAD.2011.2161190

[10] T. Krishna, C. O. Chen, S. Park, W. Kwon, S. Subramanian, A. P. Chandrakasan, and L. Peh, "Single-Cycle Multihop Asynchronous Repeated Traversal: A SMART Future for Reconfigurable On-Chip Networks," *IEEE Computer*, vol. 46, no. 10, pp. 48–55, 2013. [Online]. Available: https://doi.org/10.1109/MC.2013.260

[11] H. Kwon and T. Krishna, "OpenSMART: Single-cycle multi-hop NoC generator in BSV and Chisel," in *ISPASS*, 2017, pp. 195–204. [Online]. Available: https://doi.org/10.1109/ISPASS.2017.7975291

[12] I. Perez, E. Vallejo, and R. Beivide, "SMART++: reducing cost and improving efficiency of multi-hop bypass in NoC routers," in *NOCS*, 2019, pp. 5:1–5:8. [Online]. Available: https://doi.org/10.1145/3313231.3352364

[13] B. K. Daya, L. Peh, and A. P. Chandrakasan, "Towards High-Performance Bufferless NoCs with SCEPTER," *Computer Architecture Letters*, vol. 15, no. 1, pp. 62–65, 2016. [Online]. Available: https://doi.org/10.1109/LCA.2015.2428699

[14] M. B. Stensgaard and J. Sparsø, "ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology," in *NOCS*, 2008, pp. 55–64. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/NOCS.2008.13

[15] A. Lines, "Asynchronous Interconnect for Synchronous SoC Design," *IEEE Micro*, vol. 24, no. 1, pp. 32–41, 2004. [Online]. Available: https://doi.org/10.1109/MM.2004.1268991

[16] M. Paschou, A. Psarras, C. Nicopoulos, and G. Dimitrakopoulos, "CrossOver: Clock domain crossing under virtual-channel flow control," in *DATE*, 2016, pp. 1183–1188. [Online]. Available: http://ieeexplore.ieee.org/document/7459491/

[17] J. Ax, N. Kucza, M. Vohrmann, T. Jungeblut, M. Porrmann, and U. Rückert, "Comparing Synchronous, Mesochronous and Asynchronous NoCs for GALS Based MPSoCs," in *MCSoC*, 2017, pp. 45–51. [Online]. Available: https://doi.org/10.1109/MCSoC.2017.19

[18] I. I. Weber, F. G. Moraes, L. L. de Oliveira, and E. A. Carara, "Exploring Asynchronous End-to-End Communication Through a Synchronous NoC," in *SBCCI*, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/SBCCI.2018.8533228

[19] M. Singh and S. M. Nowick, "MOUSETRAP: High-Speed Transition-Signaling Asynchronous Pipelines," *IEEE Trans. VLSI Syst*, vol. 15, no. 6, pp. 684–698, 2007. [Online]. Available: https://doi.org/10.1109/TVLSI.2007.898732