A Low-Swing Crossbar and Link Generator for Low-Power Networks-on-Chip

Chia-Hsin Owen Chen[1], Sunghyun Park[2], Tushar Krishna[1], Li-Shiuan Peh[1]
Dept. of Electrical Engineering and Computer Science, Massachusettes Institute of Technology, Cambridge, MA 02139
[1]{owenhsin, tushar, peh}@csail.mit.edu, [2]pshking@mit.edu

***Abstract*—Networks-on-Chip (NoCs) are emerging as the answer to non-scalable buses for connecting multiple cores in Chip Multi Processors (CMPs), and multiple IP blocks in Multi Processor Systems-on-Chip (MPSoCs). These networks require an extremely low-power datapath to ensure sustained scalability, and higher performance/watt. Crossbars and links form the core of a network datapath, and integrating low-swing links within these will reduce power significantly. Low-swing links however require significant custom circuit design effort to deliver good power efficiency and high bit rate, in the face of noise. As a result, low-swing links have not been able to make it to mainstream chips which rely on crossbar and link generators from RTL. In this paper, we present a datapath generator that creates automated layouts for crossbars with noise-robust low-swing links within them. To the best of our knowledge, this is the first crossbar generator that (1) creates layouts, instead of generating just synthesizable RTL; and (2) integrates noise-robust low-swing links in an automated manner. We demonstrate our generated datapath in a fully-synthesized NoC router, and observe 50% power reduction on datapath.**

I. Introduction

Continued transistor scaling has enabled more compute and storage units to be added on the same chip. However, power limitations have forced designers to go parallel and to realize sustained throughputs with simpler computing blocks connected together. In the processor domain, the power limitations have resulted in the emergence of CMPs, while in the embedded domain, MPSoCs have started becoming popular. These trends put the interconnection fabric into limelight to enable fast and low-power communication between these processing units. On-chip buses are not scalable beyond a few cores, since they are limited by wire-delay and bandwidth [1]. There has been a trend towards using NoCs to manage wires more efficiently. For some systems, this network might comprise only a crossbar [2], while for others, an interconnection of packet-switched routers is used [3], [4] with each router comprised of buffers, arbiters, and a crossbar to enable sharing of links. In both kinds of systems, a crossbar is the fundamental building block that connects input ports to output ports.

A 1-bit $N \times M$ crossbar consists of $N \times M$ interconnected wires that are controlled by switches and enable any port to connect to any other port. The outputs of a crossbar connect to links that then connect to an IP block or a router. The crossbar and links thus together form the datapath of a NoC. This datapath has been found to dominate the NoC power consumption. Fabricated chips from academia, such as MIT RAW [5] and UT TRIPS [6], use RTL synthesis to generate the datapath, and the ratio of datapath power consumption and the total on-chip network power consumption are reported to be 69% and 64%, respectively. Intel TERAFLOPs [4] uses a custom-designed double-pumped crossbar with a location based channel driver to reduce the channel area and peak channel driver current [7] and is thus able to reduce datapath power to 32% of the total on-chip network power. Other circuit techniques that have been proposed to reduce this power consumption involve dividing the crossbar wires into multiple segments and partially activating selected segments [8], [9] based on the input and output ports. These circuit techniques present only the capacitance between the input and output port, and disable/reduce other capacitances. They are thus successful in reducing wasteful power consumption. However, they still require complete charging/discharging of the long wires from the input port to the output port and the core-core links, which are significant power consumers.

Low-swing signaling techniques can help mitigate the wire power consumption. The energy benefits of low-swing signaling have been demonstrated on-chip from 10mm equalized global wires [10], through 1-2mm core-to-core links [11], to less than 1mm within crossbars [12]–[14]. However, such low-swing signaling circuits, which can be viewed as analog circuits, require full custom design, resulting in substantial design time overhead. Circuit designers have to manually design schematic/netlists, optimize logic gates for each timing path, and size individual transistors. Moreover, layout engineers have to manually place all the transistors and route their nets with careful consideration of circuit symmetry and noise coupling. This custom design process leads to high development cost, long and uncertain verification timescales, and poor interface to other parts of a many-core chip, which are mostly RTL-based.

In the past, designers faced similar challenges while integrating low-power memory circuits with the VLSI CAD flow, with their sense amplifiers, self-timed circuits and dynamic circuits. Memory compilers, which are now commonplace, have solved the problem and enabled these sophisticated analog circuits to be automatically generated, subject to variable constraints specified by the users. This paper proposes to similarly automate and generate low-swing signaling circuits as part of the datapath (crossbar and links) of a NoC, thereby integrating such circuits within the CAD flow of many-core chips, enabling their broad adoption.

978-1-4577-1400-9/11/$26.00 ©2011 IEEE

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 15,2020 at 05:20:53 UTC from IEEE Xplore. Restrictions apply.

Since crossbars and links are such an essential component of on-chip networks, there have been efforts in the past to automate their generation. Sredojevic and Stojanovic [15] presented a framework for design-space exploration of equalized links, and a tool that generates an optimized transistor schematic. However, they rely on custom-design for the actual layout. ARM AMBA [16], STMicroelectronics STBus [17], Sonics MicroNetworks [18], and IBM CoreConnect [19] are examples of on-chip bus generators; DX-Gt [20] is a crossbar generator; and ×pipes [21] is a network interface, switch and link generator. These tools are aimed at application specific network-on-chip (NoC) component generation, but they all stop at the synthesizable HDL level, i.e. they generate RTL, and then rely on synthesis and place-and-route tools to generate the final design. This is not the most efficient way to design crossbars, as we show later in Section IV, highlighting that a synthesized crossbar design consumes significantly more power than a custom low-swing crossbar.

Contribution. In this work we present a NoC datapath generator, which is the first to integrate low-swing links in an automated manner. It is also the first to generate a noise-robust layout at the same time, embedded within the synthesis flow of a NoC router. Our tool takes a low-swing driver as input and ensures (1) a crosstalk noise-robust routing, (2) supply noise-robust differential signaling, and (3) crosstalk-controlled full-shielded links, in the generated datapath. To the best of our knowledge, our tool provides the following important contributions to the low-power NoC community:

1) It is the first automated generation of noise-robust low-swing links within the crossbar, and between routers.
2) It is the first automated layout generation of a crossbar for a user specified number of ports, channel-width, and target frequency.
3) It is the first demonstration of a generated low-swing crossbar and link within a fully-synthesized NoC router.
4) Our automatically generated low-swing crossbar achieves an energy savings of 50%, at the same targeted frequency of the synthesized crossbar, at 3-4 times the area overhead. Relative to the entire router, the larger footprint of the crossbar is manageable, at just 30% of the overall router area.

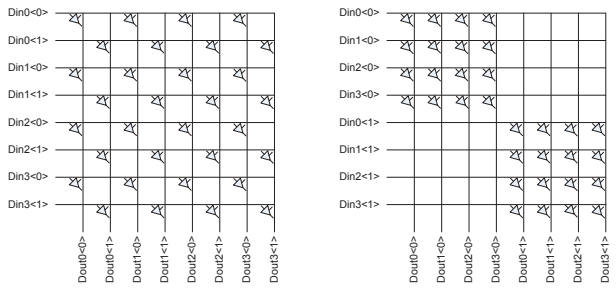
The rest of the paper is organized as follows. Section II presents some background on crossbars and low-swing link design. Section III explains our low-swing crossbar and link generator. Section IV provides some evaluation results for some datapaths generated using our tool, and Section V concludes the paper.

II. Background

In this section we present background on crossbars, low-swing links, and the limitations of the current synthesis flow.

A. Crossbar

A $N \times M$ crossbar connects N inputs to M outputs with no intermediate stages, where any inputs can send data to any non-busy outputs. Figure 1 shows the schematic of a 2-bit



(a) Port-sliced organization (b) Bit-sliced organization

Fig. 1. 2-bit 4×4 crossbar schematic

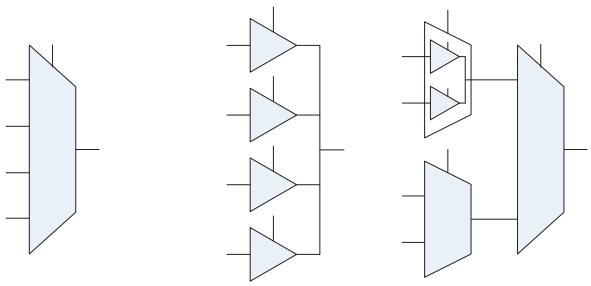


Fig. 2. Logical 4:1 multiplexer (a) and two realizations (b)(c)

4×4 crossbar. In effect, a 1-bit $N \times M$ crossbar consists of M $N:1$ multiplexers, one for each output. The $N:1$ multiplexer can be realized as one logic gate or cascaded smaller $N':1$ multiplexers, where $N' < N$, as shown in Figure 2. A custom-circuit designer often favors the former implementation due to the layout regularity, as it enables various optimization techniques. However, this implementation suffers from the fact that the intrinsic delay of the multiplexer grows with N. Synthesis tools usually use the latter approach that cascades smaller multiplexers to implement a $N:1$ multiplexer with arbitrary N. By using this approach, the intrinsic delay grows with $\log N$ instead of N. However, it may lead to higher power consumption since more multiplexers are used.

Two gate organizations are possible for many-bit crossbars, as shown in Figure 1. One organization, port-slicing, groups all the bits of a port close to each other. The other organization, bit-slicing, groups all the gates of a bit together. The former approach eases routing (since all bits for an input/output port are grouped together), and minimizes the span of the control wires that operate the multiplexers for each input port. However, using the former approach leaves lot of blank spots that increases area, and folding the crossbar over itself to reduce area is non-trivial. The latter approach, on the other hand, minimizes the distance between the gates that contribute to the same output bit. This design is easier to optimize for area by placing all the bit-cells together and eliminating blank spaces, but requires more complicated routing to first spread out and then group all bits from a port.

In addition to a crossbar, links and receivers form a datapath.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 15,2020 at 05:20:53 UTC from IEEE Xplore. Restrictions apply.

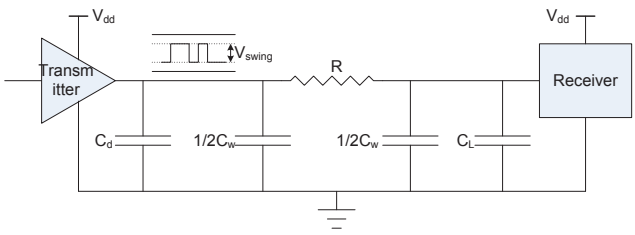


Fig. 3. Simplified datapath

Different design decisions for these components would result in trade-offs in area, power and delay. From the perspective of sending a signal, a datapath can be simplified to three components connected together: a transmitter, a wire, and a receiver, as shown in Figure 3. The corresponding delay and energy consumption can be formulated as follows:

$$Energy = (C_d + C_w + C_L)V_{DD}V_{swing} \quad (1)$$

$$Delay = ((C_d + C_w + C_L)V_{swing}/I_{av}) \quad (2)$$

where C_d is the output capacitance of the transmitter, C_w is the wire capacitance, C_L is the input capacitance of the receiver. V_{DD} is the power supply of the circuit, and V_{swing} is the voltage swing on these capacitors. I_{av} is the average (dis)charge current. In general, lowering the capacitance, reducing the voltage swing, and increasing the (dis)charging current can help in reducing energy consumption and delay.

A transmitter with larger sized transistors would have larger (dis)charging current which would decrease the delay. But it has larger footprint and C_d. Greater wire spacing lowers the coupling capacitance between wires but it takes larger metal area. Increasing wire width could reduce the wire resistance but it also increases capacitance and metal area.

B. Low-swing signaling

Current on-chip network architectures require both long interconnects for the connection of processor cores, and small wire spacing for higher bandwidth. This trend has significantly increased wire capacitance and resistance. Unfortunately, physical properties of the on-chip interconnects are not scaling well with transistor sizes. To reduce the delay and power consumption caused by these RC-dominant wires, low-swing circuit techniques [22] are now in the spotlight of on-chip networks.

In low-swing interconnects, the propagation delay decreases linearly with the signal swing, under the condition that the charge/discharge current is not affected by the reduction in voltage swing. Furthermore, the lower signal swing carries the major benefit of reducing dynamic power consumption, which can be substantial when the interconnect load capacitance is large. The low-swing signaling schemes, however, give rise to noise concerns.

Some of the noise concerns in low-swing designs can be mitigated by sending data differentially, which helps eliminate common-mode interference. However, this takes up two wires which doubles the capacitance and area. Adding shielding wires also helps reduce crosstalk and could potentially lower voltage-swing, but it also adds coupling capacitance and area. Increasing the sensitivity of the receiver helps lower voltage-swing on the wires, but it often needs a larger sized transistor or more sophisticated receiver design that has larger footprint and capacitance.

Thus low-swing links offer tremendous advantages in terms of power, and latency, but require careful design to ensure robust performance.

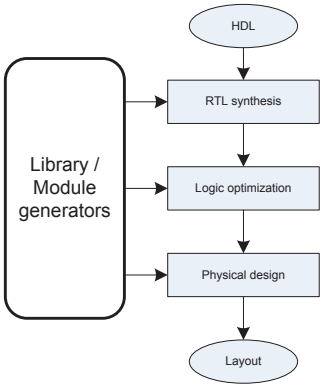


Fig. 4. Standard synthesis flow

C. Limitations to current synthesis flow

Given a hardware description of a crossbar, the existing synthesis flow, like the one shown in Figure 4, with a standard cell library could synthesize and realize a crossbar circuit. Unfortunately, the existing synthesis flow and standard cell libraries are designed for full voltage-swing digital circuits. New features in certain CAD tools enable low power designs by supporting multiple power domains and power shutdown techniques. However, none of them support analysis and layout for low voltage swing operations. Moreover, place-and-route tools are often too general and cannot take full advantage of the regularity of a crossbar and fail to generate an area-efficient layout. Therefore, a system designer needs to custom-design a low-swing crossbar, which is time-consuming and error-prone.

III. Datapath Generator

In this section we present our crossbar and link generator for low-swing datapaths. The low-swing property is enabled by replacing the cross-points of a crossbar with low-swing transmitters, and adding receivers at the end of the links to convert low-swing signals back to full-swing signals. The data links that connect transmitters and receivers are equipped with shielding wires to improve signal integrity. As shown in Table I, our proposed datapath generator takes architectural parameters (e.g. the number of inputs and outputs, data width per port, link length), user layout preferences (e.g. port locations, link width and spacing), and technology files (e.g. standard cell library, targeted metal layers, TX and RX cells), and generates a crossbar and link layout that meets specified

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 15,2020 at 05:20:53 UTC from IEEE Xplore. Restrictions apply.

TABLE I
INPUTS TO PROPOSED DATAPATH GENERATOR

Type	Proposed datapath generator
Architectural parameters	Number of input ports (N) Number of output ports (M) Data width in bits (W) Link length (L)
User preferences	Input port location Output port location Link wire width and spacing
Technology related information	Standard cell library Metal layer information Transmitter and receiver design Second power supply level (if needed)
System design constraints	Target frequency, power, area

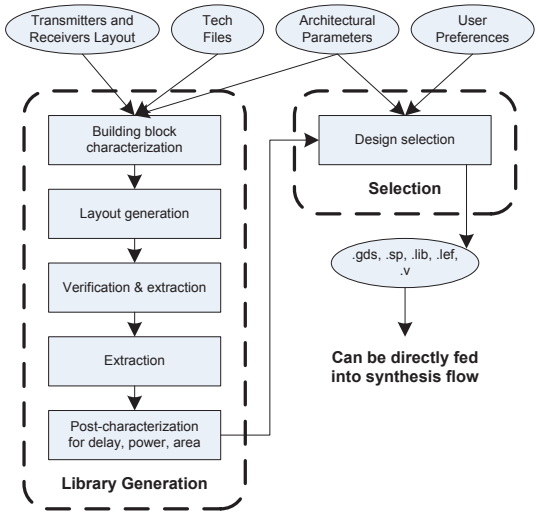
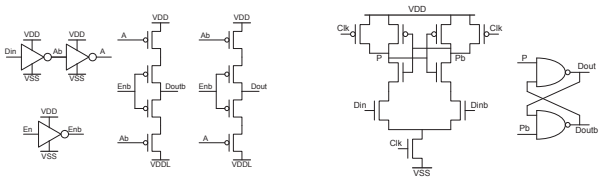


Fig. 5. Proposed Datapath Generator's Tool flow

user preferences and system design constraints: area, power, and delay. The output files of our proposed datapath generator are fed directly into a conventional synthesis tool flow, which is similar to how we use a memory compiler. Figure 5 shows the proposed datapath generation flow. The generation involves two phases, library generation and selection. In the library generation phase, the program takes a suite of custom-designed transmitters and receivers, architectural parameters that users are interested in, and technology files as inputs; Then, it pre-characterizes the custom circuits. Next, the tool generates the layout of all possible combinations and simulates them to get post-layout timing, power, and area. This forms the library of components for the selection phase. In the selection phase, the generator takes architectural parameters and user preferences as inputs to find the most suitable design from the results generated in the library generation phase, and outputs the files needed for the synthesis flow.

In the following subsections, we walk through a detailed example of generating a datapath with a 64-bit 6×6 crossbar, 1mm links, and receivers in a 45nm SOI HVT technology.



(a) Transmitter (b) Receiver

Fig. 6. Schematic of transmitter and receiver

TABLE II
PRE-CHARACTERIZATION RESULTS

	Transmitter	Receiver
Average current (μA)	2.6	11.0
Input cap (fF)	1.52 (select) 2.87 (data)	1.05 (clk) 0.4 (data)

A. Building block pre-characterization

We treat the 1-bit transmitters and receivers as atomic building blocks of the generator, thus giving users the flexibility of using different kinds of transmitter and receiver designs. Given the transmitter and receiver designs, the generator first performs pre-characterization using Spice-level simulators (we used Cadence UltraSim) to obtain average current and input capacitances. The average current is later used to determine the power wire width, while the input capacitances are used to determine the size of the buffers that drive these building blocks.

For example, Figure 6 depicts the schematic of a low-swing transmitter design and a receiver design we chose as inputs to the generator. The experiments in both this section and Section IV are performed using the IBM 45 SOI HVT technology, and the pre-characterization results are shown in Table II.

B. Layout generation

In this step, the generator tiles the transmitters and receivers to form the datapath, taking various aspects into consideration, such as building block restrictions, floorplanning, routing, and link design. This section details each of these aspects.

1) Building block restrictions: We applied constraints to the transmitters' and receivers' pin locations. The reason is twofold. First, the gates of the transistors for low-swing operations are more sensitive to coupling from full-swing wires. Therefore, some constraints on transmitters' and receivers' pin location are helpful to avoid routing low-layer full-swing signal wires over these transistors. Second, constraints on pin locations make the transmitter/receiver cells more easily tileable. Without loss of generality, we chose one specific pin layout, restricted as shown in Figure 7. The power and ground pins' locations are chosen to be the same as the corresponding pins in standard cells. All other pins are placed relative to the transmitter's core, which contains noise-sensitive transistors. For example, the Select pin is on the left of the core, the Data-in pin is at the bottom, and the Data-out pin is on the

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 15,2020 at 05:20:53 UTC from IEEE Xplore. Restrictions apply.

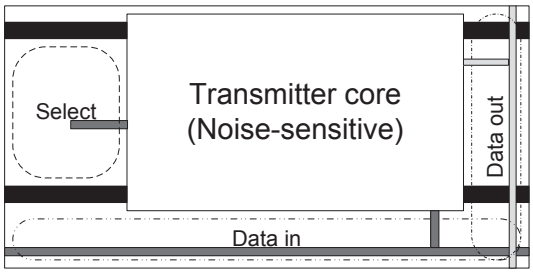


Fig. 7. Transmitter abstract layout

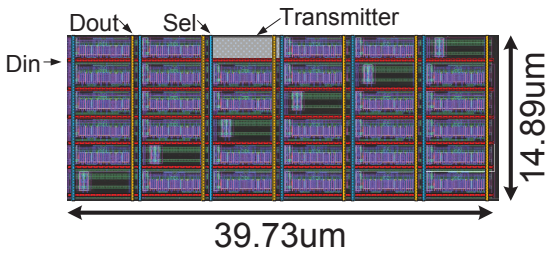


Fig. 8. Example single-bit crossbar layout with 6 inputs and 6 outputs

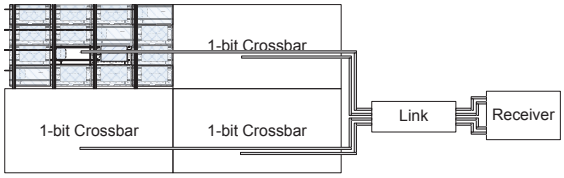


Fig. 9. 4-bit crossbar abstract layout with 1 port connecting to the link

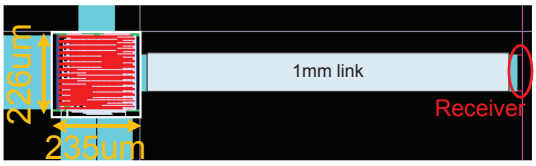


Fig. 10. Example 6x6 64-bit datapath layout with one link shown

right. Similar constraints are also applied to the receiver cell design.

2) Floorplanning: To achieve higher transmitter cell area density, we chose the bit-sliced organization, which was shown earlier in Figure 1(b). The tool first generates a 1-bit $N \times M$ crossbar as shown in Figure 8. The transmitters are placed at the cross-points of input horizontal wires and output vertical wires. The tool then places W 1-bit crossbars in a 2-dimensional array to form a W-bit $N \times M$ crossbar, as shown in Figure 9. The number of 1-bit crossbars on each side is calculated to square the crossbar layout area so as to minimize the average length of the wires each bit needs to traverse. Receivers are placed so that the routing area from the links to the receiver inputs is minimal.

Although a port-sliced organization is also effective, it requires a more sophisticated wire routing algorithm to achieve the same cell area density as a bit-sliced organization. A naive approach, as shown in Figure 1(a), would result in low-transistor density and a W^2 bit-to-area relationship, instead of W which can be readily achieved by using the bit-sliced organization.

3) Routing: For each 1-bit crossbar, the number of metal layers needed to route the power and signals is kept minimal , to maximize the number of available metal layers for output wire routing. No wiring is allowed above noise-sensitive transistors in lower metal layers. While this increases the total crossbar area, it lowers the wiring complexity for Data-out wires from each 1-bit crossbar to crossbar outputs. Since we employed the bit-sliced organization, the Data-out wires are distributed across the entire crossbar. Two metal layers are used to route the Data-out wires to the edge of the crossbar: one is used for outputs in horizontal direction, while the other is used for the vertical direction. Since the same metal layer is used to route all wires in a particular direction, the crossbar area is limited by the wire pitch if the transmitter's cell area is small. Otherwise, it is limited by transmitter cell area. As shown in Figure 9, Data-out wires coming out from the edge of the 1-bit crossbar array are routed to the inputs of links. We carefully designed the routing algorithm so that it takes minimal wiring area to connect the outputs of a crossbar to the links.

A structured layout of the power distribution network is applied. A power ring that surrounds the whole crossbar, one that surrounds the whole receiver block, and power stripes, are all automatically generated. The widths of the power wires are calculated based on the average current so that the current density is less than 1 mA/um to avoid electromigration. Using the results from the pre-characterization, we used both 0.8μm-wide and 0.7μm-wide power wire for crossbar and receiver respectively.

4) Link Design: Link parameters such as link wire length, width, and spacing are specified as the inputs of the generator. Since the links are running at low-swing, they are more vulnerable to noise. We thus add shielding wires to improve the noise immunity. We chose the shielding wire organization that is shown in Figure 11, where a shielding wire is placed on the same layer as link between two different bits and two shielding wires are placed right below the differential wires. This is chosen as it minimizes low-swing noise from other links and full-swing logic from lower metal layers.

Typically the wire length is set based on the distance between the crossbar and the components this crossbar is connected to. Different choices of wire width and spacing would affect the timing and energy consumption of transmitting a signal. For example, one could reduce the delay by doubling the wire pitch but it requires larger wiring area. Table III shows this trade-offs between link area and link performance, where the wire width is normalized to the minimum wire width and the wire spacing is normalized to the minimum spacing. The

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 15,2020 at 05:20:53 UTC from IEEE Xplore. Restrictions apply.

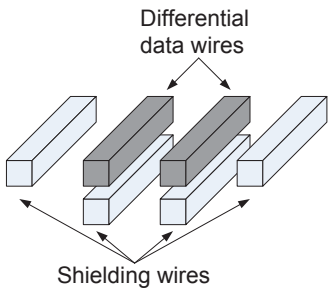


Fig. 11. Selected wire shielding topology

TABLE III
PERFORMANCE OF 1MM^2 LINK OF TWO ORGANIZATIONS

Wire width	Wire spacing	Delay (ps)	Energy/bit (fJ)	Link area (mm^2)
1	2	70.0	35.0	0.093
2	4	33.7	30.5	0.176

performance was simulated by transmitting a full-swing signal on the link.

A layout of the example datapath generated is shown in Figure 10.

C. Verification and Extraction

We use Calibre from Mentor Graphics to check if the generated circuit obeys the design rules, and to perform layout versus schematic (LVS) verification. A schematic netlist is generated for LVS. In order to get a more accurate delay of the circuit, RC extraction is done for the post-characterization of the generated design.

D. Post-characterization and selection

Post-characterization is performed to determine the actual frequency, power, and area the crossbar can achieve. The selection step chooses the suitable datapath design based on the results from the post-characterization step, and outputs the files needed for the standard synthesis flow.

The Table IV shows the simulation results for the walk-through examples. At the selection step, for example, if the criteria is to achieve high frequency and have little constraint on the area, the design with doubled link pitch is returned.

E. Discussion

The proposed generator enables the layout generation of low-swing datapaths for a given set of architectural parameters and design constraints, and outputs files for integration with the standard synthesis flow. The generator removes additional design effort necessary for a fully custom design by automatically and systematically tiling transceivers and routing wires. When compared to synthesizing a full-swing datapath using commercial tools, our generator adds SPICE simulation overheads to characterize the properties of the generated datapaths such as delay and power during the library generation step.

To use the generator with a different technology, the user needs to provide the technology related information as shown in Table I, and to provide the layout of the transmitters and receivers designed for that specific process.

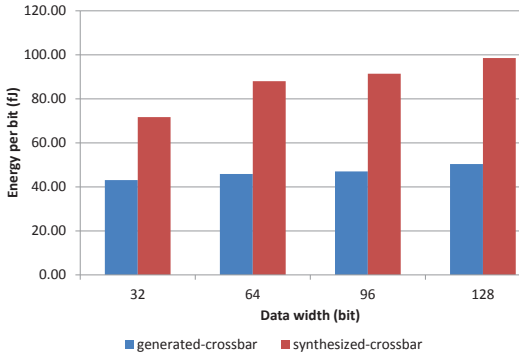


Fig. 12. Energy per bit sent of 6-port datapaths with different data width

IV. EVALUATION

In this section, we first evaluate the crossbars generated by our proposed tool, against the synthesized crossbars. We then present a case study of a 5-port NoC virtual channel router that is integrated through the standard synthesis flow with the low-swing datapath generated by our tool.

In all our experiments, we used Cadence Ultrasim to evaluate the performance and power consumption of the RC extracted netlists.

A. Generated vs. synthesized datapath

Using the transmitter and the receiver design we describe in Section III, we generated low-swing datapaths across a range of architectural parameters and compared the simulation results with datapaths generated by standard CAD tools using only standard cells. We will refer to the crossbar/datapaths generated by our tool as *generated* crossbars/datapaths, and those generated by standard CAD tools using standard cells as *synthesized* crossbars/datapaths. Evaluating generated datapaths with different transmitter and receiver designs can be done but is equivalent to evaluating the effectiveness of different low-swing techniques, which is beyond the scope of this work. In our experiments, we assumed a link length of 1mm and specified a delay constraint of 0.6ns from the input of the crossbar to the output of the link for synthesized datapaths so that the datapath can be run at 1.5GHz.

Energy per bit. We simulated the datapaths (crossbar and link) at 1.5GHz and report the results for varying data widths and varying number of ports in Figure 12 and Figure 13, respectively. As shown in Figure 12, for both crossbars, as the data width increases, the energy per bit sent also increases because an increase in the data width leads to an increase in the area of the crossbar. This increase results in longer distance (on average) for a bit to travel from an input port to an output port. Longer distance translates to higher energy consumption. The energy per bit sent also increases with the number of ports, because a bit needs to drive more transmitters. Overall, our simulations showed that a generated datapath, as in our

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 15,2020 at 05:20:53 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
EXAMPLE GENERATED DATAPATHS

Link wire width	Link wire spacing	Max freq (GHz)	Crossbar area (mm^2)	Energy/bit (fJ)
1	2	2.5	0.053	46.4
2	4	2.7	0.084	48.3

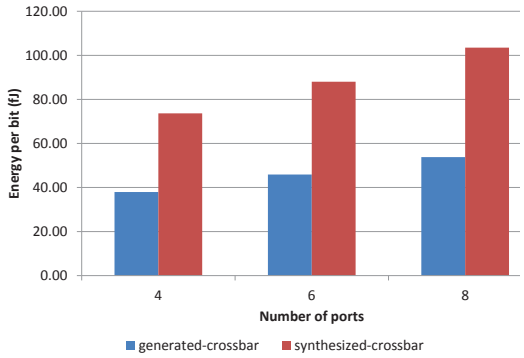


Fig. 13. Energy per bit sent of 64-bit datapaths with different number of ports

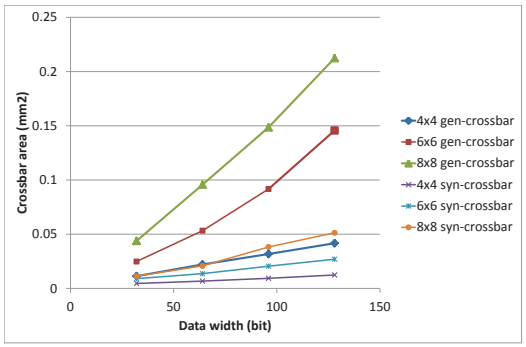


Fig. 14. Crossbar area with various architectural parameters

design, results in 50% energy savings (on average per bit sent) compared to a synthesized datapath.

Area. Figure 14 shows the area of the generated vs. synthesized crossbars. Due to the bit-sliced organization and larger transmitter size, the generated crossbar area is dominated by the transmitter area. Using this organization results in its crossbar area growing linearly with the data width and quadratically with the number of ports, as captured in Figure 14. On the other hand, as Figure 14 indicates, a synthesized crossbar has a smaller area footprint because the transmitter design we are simulating is differential, and our wire routing is conservative to achieve high immunity to noise. Both of these factors result in increased area footprint, a trade-off for better latency and lower power of low-swing datapath. In addition, the design of low-swing transceivers for on-chip is in its infancy. Future designs may be more area-efficient, and continue to be pluggable into our generator toolchain.

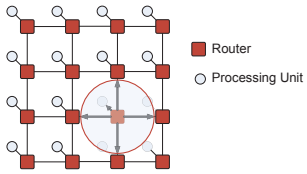


Fig. 15. Five-port router in a mesh network

TABLE V
ROUTER SPECIFICATIONS

# of input ports	5
# of output ports	5
Data width	64
# of buffers per port	16 (1k bits)
Flow control	Wormhole with VC
Buffer management	On/Off
Working frequency	1 GHz

B. Case Study

We synthesized a typical NoC router of a mesh topology integrated with a low-swing datapath using the files generated by our tool. The router is a 3-stage pipelined input buffered virtual channel (VC) router with five inputs and five outputs [23], and with a 64-bit data path. As shown in Figure 15, one input and one output port are connected to the local processing unit, while the remaining ports are connected to neighboring routers. We assumed that the local processing unit resides next to the router, the distance between routers is 1mm, and the target working frequency is 1GHz. Table V shows the detailed router specifications.

We used the same synthesis flow shown in Figure 4 to realize the router design from RTL to layout. Figure 16 shows the final layout of the router with the generated datapath. The black region in the figure is assumed to be occupied by processing units. The low-swing crossbar occupies about 30% of the total router area. The delay of the low-swing datapath is 630ps. The power consumed in the generated datapath is 18% of the total power consumed by the router[1]. The power consumption was obtained from UltraSim simulations by feeding a traffic trace through all the ports of the router. The traffic trace was generated from RTL simulations of a 4x4 NoC; every node injects one message every cycle destined to a random node. The final synthesized router with the generated low-swing crossbar and links consists of 286,079 transistors.

[1]It should be pointed out that this is a textbook NoC router. With a bypassing NoC router, such as that in [14], the NoC power will be largely that of the datapath, since most packets need not be buffered and can go straight from the input port through the crossbar to the output port and link.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 15,2020 at 05:20:53 UTC from IEEE Xplore. Restrictions apply.

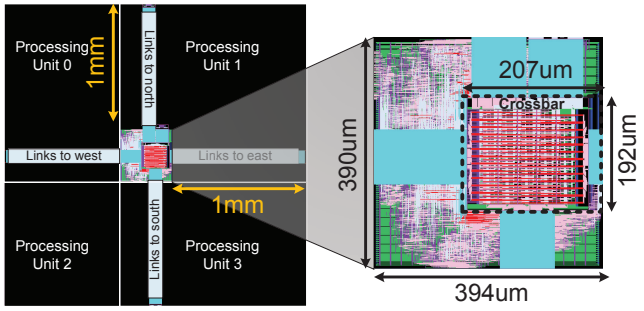


Fig. 16. Synthesized router with generated low-swing datapath

V. Conclusion

In this work, we present a low-swing NoC datapath generator that automatically creates layouts of crossbar and link circuits at low voltage swings, and enables the ready integration of such interconnects in the regular CAD flow of many-core chips. Our case study demonstrates our generated datapath embedded within the synthesis flow of a 5-port NoC mesh router, leading to 50% savings in energy-per-bit. While our case study leverages a specific low-swing transmitter and receiver circuit, our generator can work with any TX/RX building block and we will release it upon publication. We hope this will pave the way for low-swing signaling techniques to be incorporated within mainstream VLSI design, realizing low-power NoCs and enabling many-core chips.

Acknowledgment

The authors acknowledge the financial support of the Interconnect Focus Center, one of the six research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program, as well as NSF CPA-0811375.

References

[1] A. Pullini, F. Angiolini, S. Murali, D. A. G. D. Micheli, and L. Benini, "Bringing NoCs to 65nm," *IEEE Micro*, vol. 12, no. 5, pp. 75–85, September 2007.

[2] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA Workloads Using a Detailed GPU Simulator," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2009)*, April 2009, pp. 163–174.

[3] D. Wentzlaff *et al.*, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, 2007.

[4] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-ghz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, September 2007.

[5] M. B. Taylor, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, J. Kim, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "Evaluation of the Raw microprocessor: An exposed-wire-delay architecture for ILP and streams," in *Proc. Intl. Symp. Computer Architecture (ISCA)*, June 2004.

[6] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. R. Moore, "Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture," in *Proc. Intl. Symp. Computer Architecture (ISCA)*, June 2003.

[7] S. Vangal, N. Borkar, and A. Alvandpour, "A six-port 57gb/s double-pumped nonblocking router core," in *Symp. VLSI Circuits*, June 2005, pp. 268–269.

[8] H. Wang, L.-S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in *Proc. Intl. Symp. Microarchitecture (MICRO)*, 2003.

[9] K. Lee, S.-J. Lee, S.-E. Kim, H.-M. Choi, D. Kim, S. Kim, M.-W. Lee, and H.-J. Yoo, "A 51mw 1.6ghz on-chip network for low-power heterogeneous SoC platform," in *IEEE Intl. Solid-State Circuits Conference (ISSCC)*, February 2004.

[10] B. Kim and V. Stojanovic, "A 4gb/s/ch 356fj/b 10mm equalized on-chip interconnect with nonlinear charge-injecting transmit filter and transimpedance receiver in 90nm cmos," *IEEE International Solid-State Circuits Conference, Digest of Technical Papers.*, pp. 66–68, Feb. 2009.

[11] D. Schinkel, E. Mensink, E. Klumperink, A. van Tuijl, and B. Nauta, "Low-power, high-speed transceivers for network-on-chip communication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 1, pp. 12–21, January 2009.

[12] M. Sinha and W. Burleson, "Current-sensing for crossbars," in *IEEE Intl. ASIC/SOC Conference*, September 2001.

[13] P. Wijetunga, "High-performance crossbar design for system-on-chip," *System-on-Chip for Real-Time Applications, International Workshop on*, vol. 0, p. 138, 2003.

[14] T. Krishna, J. Postman, C. Edmonds, L.-S. Peh, and P.Chiang, "SWIFT: A SWing-reduced Interconnect For a Token-based Network-on-Chip in 90nm CMOS," in *Proc. Intl. Conference on Computer Design (ICCD)*, Oct. 2010.

[15] R. Sredojevic and V. Stojanovic, "Optimization-based framework for simultaneous circuit-and-system design-space exploration: A high-speed link example," *Computer-Aided Design, International Conference on*, vol. 0, pp. 314–321, 2008.

[16] "ARM AMBA," http://www.arm.com/products/system-ip/amba.

[17] "Stbus communication system: Concepts and definitions," http://www.st.com/stonline/books/pdf/docs/14178.pdf.

[18] D. Wingard, "Micronetwork-based integration for SoCs," in *Proc. Design Automation Conference (DAC)*, May 2001, pp. 673–677.

[19] "IBM CoreConnect," https://www-01.ibm.com/chips/techlib/techlib.nsf/productfamilies/CoreConnect_Bus_Architecture.

[20] M. A. Shalan, E. S. Shin, and V. J. M. III, "DX-GT: Memory management and crossbar switch generator for multiprocessor system-on-a-chip," in *Proc. Workshop on Synthesis and System Integration of MIxed Technologies*, April 2003, pp. 357–364.

[21] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, "xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs," in *Proc. Intl. Conference on Computer Design (ICCD)*, October 2003.

[22] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective, second edition*. Prentice Hall, 2003.

[23] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, 2004.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 15,2020 at 05:20:53 UTC from IEEE Xplore. Restrictions apply.