**Design and Applications for Embedded Networks-on-Chip on FPGAs**Mohamed S. Abdelfattah, Andrew Bitar and Vaughn Betz

Dear Reviewers, Associate Editor and Editor In-Chief,

Thank you very much for your helpful reviews. Many of the comments were insightful and have indeed helped greatly improve our submission. In this document, we respond to the reviewers comments in detail, and we point to changes that we made in the revised manuscript. In the following pages, we attach the reviewers’ comments (highlighted in gray) and our responses are inline.

Sincerely,

Mohamed Abdelfattah, Andrew Bitar and Vaughn Betz

University of Toronto

26 July 2016

***Reviewer: 1  
  
Public Comments (these will be made available to the author)  
Thanks for the very interesting paper. It was an informative read. I have just a few minor questions & comments:***

Thank you very much for your detailed review of our work. ***Figure 3:  
I think read\_enable and write\_enable labels on signals entering the aFIFO in Fig 3(b) need to be switched around. Also, should an "almost\_full" signal be added between the VC Buffers in Fig 3(b) and the write\_en logic?***

Thank you very much for catching that error; the read/write enables were indeed switched and I have now corrected them in Fig. 3.

Regarding an “almost\_full” signal, it is not needed because the NoC will never forward a flit to the FabricPort if there are no available buffer spaces in the VC buffers. The upstream “credits\_out” signal informs the upstream router of the number of available buffer spaces – every time a flit is read out of the VC buffers, a credit is sent upstream signalling that a buffer space has become available. ***At end of Sec 4.2: (after Condition 1) if packets are interleaved on alternate VCs, why can't four 1-flit packets be handled with 2 VCs?***

Combine-data mode is limited to combining packets equal to the number of VCs for the following reason: each VC has its own ready/valid set of signals. This means that if we have two VCs, there are only two ready signals going into the FabricPort output, and therefore we can only stall two data streams coming out of the FabricPort. If I combine 4 1-flit packets using two VCs, it could still be possible but without the ability to stall each data stream independently which would limit the use-cases for this feature and make it more complex to use. To clarify this in the manuscript, I have added the following sentence at the end of section 4.2:

***“Note that we are limited to the merging of 2 packets with 2 VCs because each packet type must have the ability to be independently stalled, and we can only stall an entire VC, not individual flits within a single VC.”  
  
Sec 4.4 (the part of the 2nd para beginning "To avoid this deadlock..." until Condition 4: The description is considerably more complex than Fig 3 and should therefore, in my opinion, be elaborated upon, perhaps with an additional figure?***

We clarified the text to better explain the deadlock-free FabricPort output. The idea is that we extend the VCs beyond the NoC reader by having a separate aFIFO and Demux for each VC, then a MUX at the output of the FabricPort will choose from which VC to read.

Figure 9 explains the logical organization of a deadlock-free FabricPort Output. Ideally, we would add another figure to better explain the circuitry, this but we are very restricted by size limits of the journal.

***Also, does this section not contradict Condition 3, as these packets are likely sharing the same connection? Also, the approach assumes the module knows which VC each message type arrives upon. How would it do so?***

You are correct in that we contradict Condition 3 because different message types may indeed share the same connection. We made Condition 3 more specific by restricting its applicability to packets that share both the connection and message type.

The module knows which VC each message type arrives on since this is determined statically before the application is programmed onto the FPGA, and VCs are not allowed to be switched mid-application.  ***Fig 9(b): Aren't the stall & release of the VC handled by the arbiter? How does the module control these?***

The module can control these by stalling or releasing the TDM, which propagates the stall back to the aFIFO, which propagates the stall signal back to the NoC reader and that will stall data coming from the NoC. In other words, we propagate backpressure through the FabricPort so that the whole data path is elastic and can respond to backpressure.

***Minor typos:  
Abstract, line 7: access-latency shouldn't be hyphenated  
Sec 1, 2nd para, 4th last line: hamper not hampers  
Sec 1, 3rd para, 11th line: becomes built INTO the FPGA  
Sec 4.5, 1st para, 2nd last line: design late in the design compilation, and thus improve  
Sec 5.1, 1st para, 5th line: like NOCs ARE error-prone and slow  
Sec 5.3.1 3rd last line: considerably  
Ref [2]: capitalize FPGA***

We really appreciate your detailed corrections to our manuscript; we have corrected these minor typos in the revised manuscript.

***Reviewer: 2  
  
Public Comments (these will be made available to the author)  
This paper presents the architecture of an embedded (hard IP) Network-on-Chip for FPGAs and evaluates the performance and resource efficiency based on selected case studies. All aspects (except the IO-links) have been already reported in previous conference and journal publications.  
  
The paper is easy to read, technically sound and covers an interesting architectural option for FPGAs. However, the very little amount of new material makes this manuscript a rather borderline paper. The only new aspect described in this submission is the IOLink interface that connects the NoC directly to the FPGA's IOs. This feature is evaluated using a direct DDR3 interface.  
  
I think the paper is only just eligible for a journal publication; it could be much stronger if the authors would provide a more general evaluation of their concept. I can see that the hard NoC is beneficial for selected applications. However, I would like to understand if the described architecture is flexible enough to justify its integration into a mainline FPGA device. Furthermore, how would the presence of a hard NoC influence the design style for FPGAs?  
  
A desirable feature for many applications are non-point-to-point communication patterns that can be implemented in the FPGA fabric. Is there a plan to enhance the hard NoC with broadcast and/or multicast support? I also miss a discussion about the suitability of the NoC approach for modern high-level programming models for FPGAs (i.e., OpenCL). Could the NoC (maybe with modifications) be used as a baseline communication infrastructure to connect processing engines and peripherals?***Thank you very much for your comments and review. We made sure to increase the amount of new material by adding a section about using embedded NoCs for compute acceleration such as that with OpenCL (See section 5.6). We explain how the NoC can be used to implement “communication infrastructure”, commonly called the accelerator “shell” while simplifying design using the NoC and making it more efficient.

Regarding broadcast: this is one of our items to investigate in the future. Like you pointed out and reviewer 3 pointed out as well, this would indeed be a very useful addition to the NoC and it comes with low overhead.

Regarding design style: we address latency-sensitive and latency-insensitive design implementation using the NoC in this work. However, we also address the CAD tools to use an embedded NoC in other work which has been accepted to the FPL 2016 conference. Our FPL paper complements this manuscript and explains how the FPGA design style need not be changed to suit the NoC. By using FPGA-design-compatible CAD tools, the NoC can be transparently used by the designer. The FPL paper also compares the NoC to general-purpose buses used by FPGA designers now, better addressing the general applicability of an embedded NoC to FPGA applications.

See: Mohamed S Abdelfattah and Vaughn Betz. LYNX: CAD for Embedded NoCs on FPGAs. In

International Conference on Field-Programmable Logic and Applications (FPL). IEEE, 2016 (Accepted)

***Reviewer: 3  
  
Public Comments (these will be made available to the author)  
The paper proposes a hard NoC to be embedded into an FPGA fabric. The authors use this to reduce wire utilization and to improve applications that need switching anyway (such as a network packet processing engine).  
  
The paper is mostly very well presented and the topic is very important as datapaths  
are currently rapidly growing for providing large throughput while at the same time FPGA capacity is growing which puts enormous pressure on the inter-module routing.***

Thank you for your detailed and insightful review. Your comments greatly helped in improving our manuscript. ***Your approach adds a certain level of uncertainty in the communication, which may eventually result in very difficult problems such as race conditions.  
So at the end, designing for FPGAs using your NoC approach might be actually not more productive than what we do today, or?***

We avoid race conditions for dependent messages by enforcing design rules that ensure that dependent messages arrive on different VCs. This is one example of how we make the NoC compatible with FPGA design styles such that it can be used instead of a bus-based interconnect without making design more difficult or complex. ***I haven't understood the clock infrastructure. If I got it right, each of the 16 interfaces can have its own clock, but then you want to derive the 4x clock locally which means that the NoC backplane is not using a global clock (what you call intermediate clock)?  
Or if you have the a global intermediate clock, does this mean that all user modules have to meet 400MHz, which may also be a nightmare on a 1M LUT FPGA?  
Please go through the document and remove all statements that can cause confusion on how the clock infrastructure works. What are adjustable clock speeds, what is global and what is local, where are clock domain crossings?***

Fig. 2 clarifies the design-facing clocks F\_fabric and F\_noc. The NoC clock is fixed at 1.2 GHz and does not change. However, each interface clock can be different. This is why we have two-stage clock-crossing at the FabricPorts (see Fig. 3). First, the Module Frequency F\_fabric is multiplied by 4 to become the F\_int (intermediate clock), then an aFIFO crosses to the NoC clock domain F\_noc. The NoC operates at a fixed global clock while each module can have an independent local clock F\_fabric. The intermediate clock F\_int is derived locally within each FabricPort as it is only used with the de/mux and the aFIFO. ***With respect to your description, I don't understand Fig. 4. Why is the red 2 flit not sent together with red 0 and 1 but instead nested with flits from the other VC? You are really not sending packets atomically?***

An NoC that uses VC flit-based flow control is inherently allowed to interleave flits of different packets if they are on different VCs. This is why we make sure to sort them back into packets within the FabricPort. ***Figure 6; your approach does not show an improvement in latency over traditional soft buses. This is kind of expected, but how does it looks like under more realistic load levels?***

In this paper, we focus more on application case studies; however, we explore different traffic and interconnection patterns in a follow-up conference paper that compares the performance of NoCs and general-purpose buses. We show that embedded NoCs can handle all the features of transaction communication on buses while improving on latency and throughput in most cases.

See: Mohamed S Abdelfattah and Vaughn Betz. LYNX: CAD for Embedded NoCs on FPGAs. In

International Conference on Field-Programmable Logic and Applications (FPL). IEEE, 2016 (Accepted)

***Section 4.2  
If you use 150 wires per signal direction, this may not fit well the otherwise more homogeneous fabric layout. Does this make timing closure and tools more complicated?***

Our previous work [TRETS] quantified the metal area overhead of a typical NoC to 3% of two intermediate metal layers. This is not a large amount and is not likely to make the layout of the existing homogenous FPGA wiring more difficult. Additionally, we make sure that the router layout and aspect ratio to fit within the FPGA fabric thus reducing its impact on the homogeneous FPGA logic.

Additionally, current FPGAs already contain block RAM modules that are up to 27x as big as an FPGA logic cluster, and multiplier modules that are about 11x as big as logic clusters [TRETS]. These relatively irregular components are commonplace in FPGAs nowadays and we view NoC routers in the same light.

[TRETS] Mohamed S Abdelfattah and Vaughn Betz. Networks-on-Chip for FPGAs: Hard, Soft or Mixed?

ACM Transactions on Reconfigurable Technology and Systems (TRETS), 7(3):22, 2014 (Invited) ***Dimension ordered routing intends to high congestion.  
You should provide at least some details about the routing algorithm and how you deal with congestion.***

To fulfill our design conditions, we must use deterministic routing functions such as DOR. All paths must be determined statically to make sure that our ordering and deadlock avoidance constraints are fulfilled. Furthermore, our embedded NoC is overprovisioned to the requirements of applications that are likely using it; therefore, we believe that deterministic routing functions are sufficient.

***Also interesting for a several applications could be multicast routing, which should be cheap to implement.***

This is one of our items to investigate in the future. Like you pointed out and reviewer 2 pointed out as well, this would indeed be a very useful addition to the NoC and it comes with low overhead.

***You can consider deadlock situations that require more than 2 VCs, So you need another condition that the application designer has to stick to this limit.***

This is true, thank you for your comment. We extended Condition 4 to include the following sentence:

“The number of dependencies must be less than or equal the number of VCs.” ***Latency-sensitive design,  
Condition 5 and 6 are basically a restrictive version of traffic shaping.  
You can probably relax these conditions such that a single NoC link doesn't get overloaded. However that requires knowledge of the topology, the exact mapping, and the used routing algorithm.***

Relaxing these conditions might not necessarily guarantee the cycle-by-cycle relationship of data in a latency-sensitive system; therefore, we opt to enforce more conservative constraints that ensure cycle-precise latency behaviour. ***Simulation  
Does the Booksim simulation supports uncertainties?  
For example the latency and even the order over multiple VCs can change and may have impact on the operation of a module.***

Booksim simulates the exact movement of packets in an NoC with a cycle-by-cycle accuracy. The simulation is a deterministic one since we use round-robin arbiters in our routers – there is no uncertainty in the movement of packets on the NoC with our chosen parameters.

***Fig. 11 is a bit small.  
Section 5.2.1 Memory Interface Components could be written much more concise, if space is needed for the final submission.***

Thank you for this suggestion, we shortened the explanation of the memory interface to make room for the revisions and the addition of Section 5.6. ***If you write so much about memories, you could have mentioned how you would connect a module through a fabric NoC port to the memory.***

Connecting through the FabricPort would be in the same way we connect a memory interface to a soft bus on an FPGA. The memory data would have to be down-converted to a slower speed and larger width, and then the FabricPort would up-convert it again to transport it through the NoC. Adding a discussion of that alternative would be, in our opinion, distracting to the reader, especially that we do not recommend using that option. Additionally, the reader can figure out the performance of such a connection from the data available in Table 2. For these reasons, we choose to keep the comparison clear by comparing our preferred way of connecting to the memory interface (through IOLink) to the current way it is being done (with soft bus and down-conversion). We hope that this is satisfactory.

***Considering Condition 5 and 6, the NoC seems pretty much to be globally occupied when using it for interfacing memory, right?  
In this case, the NoC seems to be too expensive.***

Condition 5 and 6 are for latency-sensitive systems. Interfacing to memory does not fall under that category since memory transfers inherently have variable latency. Additionally, in terms of bandwidth, the NoC is not globally occupied. Each of the NoC Links can transport the entire bandwidth of memory which means the NoC has much aggregate bandwidth left over to transport other data throughout the FPGA – a memory interface would occupy approximately one tenth of the NoC’s aggregate bandwidth. ***The JPG case study seems to be sensible. However, there may be a minor flaw in the sense that you would firstly parallelize within each block, e.g. such that you could handle, for example, 8 pixels at once rather than 8 pixels of 8 blocks in 8 different processing pipelines (because you have less data reordering). This way, you will also have larger monolithic modules rather than several small modules which is probably even the better test case for your NoC.***

Thank you for your suggestion. Indeed, coarser-grained monolithic blocks would be a better test case for the NoC. However, it was easier to parallelize the JPG application by doing multiple streams in parallel. This made it easier to investigate different sizes of the application and investigate varying system sizes. Additionally, we try not to favour the NoC in our comparisons to draw conservative conclusions.  ***The heat map shown in Fig. 14 hides the NoC wire utilization. While the original version looks inferior to the NoC version, you are using now wires that are hidden in the NoC, but that cost resources to manufacture. Also, you will probably have quite an over-provisioning of resources for the NoC considering your JPEG example. Can you make a clear point that investing into a NoC instead of investing into long distance wires is the better solution (considering all VLSI aspects)?***

We agree that more wires are being used, but the dedicated NoC wires are not part of the FPGA programmable interconnect. The heat map is intended to quantify the effect of using an NoC on the programmable wiring resources of the FPGA. By reducing routing stress, it becomes easier to place and route the design and this indirectly improves both the design frequency, and possibly the compilation runtime.

Regarding metal wire overhead, our previous work [TRETS] quantified the metal area overhead of a typical NoC to 3% of two intermediate metal layers. This is not a large amount and is not likely to make the layout of the existing homogenous FPGA wiring more difficult. FPGAs currently use many intermediate metal layers (6 or more) and its programmable wiring resources are vastly overprovisioned to accommodate different designs. By reserving some of that wire area (less than 1% overall) for use with an NoC, we do not anticipate a noticeable impact on the programmable interconnect.

In our JPG experiments, we reserve router partitions on the FPGA fabric which effectively blocks the usage of the wires in these partitions, so it captures some of the effect of replacing wires in the programmable interconnect with wires of the NoC as well. The JPG design example showed that a design’s frequency can be made much more predictable with an NoC regardless of physical location constraints – we believe that this makes a strong case for using an NoC. While long-distance wires are also useful to FPGAs, they still suffer from uncertain performance. ***Regarding the Ethernet example, that seems a bit biased to show maximal area improvement and you omit that data has to be routed to the 16 ports first before you can actually route. That may result in that much congestion on the routing fabric that the logic for switching is a minor concern, or? Also you should say that the third capacity comes from the NoC architecture and that it will be less on larger NoCs.***

In our Ethernet case study, we implement the full design as shown in Figure 15. That means we implement the whole Ethernet switch including the MAC logic and input queue buffering before the switch fabric and output queue after it. The dominant source of area and delay is definitely the large switch itself, which we replace with our embedded NoC. ***The packet processor example seems to be biased as well and more experiments seem to be needed here. Your design uses very low resource utilization. This means that if you would increase the number of packet processing elements, your improvement would get much lower (at least on the area side).***

Increasing the number or size of the processing elements would increase both the size of NoC-PP and the conventional PP to which we compare. It is unclear whether the area improvement would eventually diminish.. To that end, we compared to the two designs we found in the literature to try and draw a complete picture of the comparison.

NoC-PP is also a new architecture for implementing packet processors that is quite different from previous approaches. Besides the area gains, we believe it has more flexibility that is especially suited to a reconfigurable FPGA-based packet processor. This is especially true if partial-reconfiguration were to be used in dynamically reprogramming the packet processor (however we haven’t investigated partial reconfiguration yet.) ***It is not always clear what has been actually implemented and what is conceptional.  
For example in Section 5.5 you talk about partial reconfiguration, have you used that?***

Thank you for pointing this out. We refined the text in Section 5.5 to better clarify what has been implemented. We did not use partial reconfiguration but we believe it could be made easier with an NoC. To avoid confusion, we moved discussions of partial reconfiguration to a new section (5.6) which discusses the future use of embedded NoCs within FPGA compute accelerators. ***With Stratix 10, Altera introduced pipeline flops in their routing which allows more precise retiming and in particular very deeply pipelined systems including buses between modules. That is a very competing solution to the problem described here and maybe even the better option. It is consequently absolutely necessary to discuss this related approach here.***

Thank you for this suggestion. We added a discussion of Altera’s pipelined routing in the introduction.

“New FPGAs now contain pipeline registers in their programmable interconnect which makes timing closure much easier [7]; however, designers must still redesign their system-level interconnect to suit each new application or FPGA device.”

While we believe that a pipelined interconnect will greatly help timing closure, it only solves half the problem that our NoC tackles. Other than aiding timing closure, an embedded NoC is a pre-built interconnect with buffering and switching – it can implement the system-level interconnection needs of FPGA applications at much lower cost and user-facing design complexity. ***There is one advantage for using a NoC that is not directly mentioned in the paper and that is that you can relax (eventually) global routing which allows for a better separation of individual modules which should improve performance.  
Currently, logic between two modules that are communicating with each other through wide buses often forces the placer to heavily nest the resources of the two modules in order to get the inter-module routing performance done.***

Thank you for pointing out this advantage. We agree and believe that the logical separation of IP within an application will be advantageous in many ways. We have added the following text to the introduction:

“Importantly, an NoC decouples the application’s computation and communication logic. This improves design modularity, relaxes placement and routing constraints, and enables the independent optimization and compilation of application modules, which is bound to improve performance.”