Multi NoC Performance Driven Exploration on FPGA SoC for Industrial Applications

Daksha Sharma, Pritam Majumder, Mikhail Galeev, Johanan Agarwal
Intel
2501 NE Century Blvd
Hillsboro, OR 97124 USA

Abstract

Field Programmable Gate Arrays (FPGAs) are pervasive everywhere, even more so in industrial and embedded domains. It provides support for diverse workloads with stringent latency requirements. In order to provide flexibility of programmable logic with power, performance and cost savings of hard IP, a Hard Processor System (HPS) along with FPGA fabric delivers reduced board space, system power and total cost of ownership savings by eliminating the requirement of a discrete embedded processor. This paper covers system level performance modeling of an HPS platform with multiple NoCs (Network-on-chip) , peripherals and memory hierarchy using Synopsys Platform Architect tool. As a case study, performance of Time Sensitive Networking (TSN) endpoints along with USB endpoints is thoroughly evaluated using concurrent traffic analysis. Our performance optimization results show 55% average latency reduction compared to baseline architecture. Early design space exploration allowed architectural optimizations impacting NoC silicon changes and sign off ≈ 8 months ahead of RTL (Register transfer level) define here verification start.

Key words

Hard Processor System (HPS), Industrial Applications, Network-on-Chip (NoC), Time Sensitive Networking (TSN)

I. Introduction

Industrial applications are one of the major volume drivers for FPGAs [1], [2]. FPGAs can uniquely solve problems involving I/O flexibility, low latency processing, safety and security, programmable logic for customer workload offload, etc. FPGA solutions [3] provide the connectivity for Industry 4.0 [4] factories to be smart, agile, and increasingly autonomous while maintaining robustness and reliability. FPGA based designs allow industrial equipment manufacturers to quickly add new features, get new products to market, and deliver best-in-class performance with seamless deterministic interoperability. All industrial applications need some sort of real time visibility and performance data, with remote access and asset tracking.

A key component enabling industrial workloads on FPGAs is *Hard Processor Subsystem (HPS)*. It brings heterogeneous compute to the FPGA with hardened SoC complex of compute and high-speed I/O peripherals alongside soft FPGA core fabric to program embedded customer use cases. General purpose Operating Systems (OS) like Linux runs on HPS Application Processors to support various applications like Ethernet with TSN, USB3, FPGA based accelerator application through I/O coherent interface to SDRAM, etc. The key benefits of HPS in FPGA SoC are:

- It can support diverse field bus protocols, in an ever evolving industrial segment.
- As it is an integrated SoC, it incurs low latency to fabric that is essential for time sensitive workloads.
- It has modular and scalable architecture to support application parallelism, and available programmable FPGA fabric for implementing custom accelerators.
- It is power efficient which is a prerequisite for edge applications.

Multi-NoC heterogenous SoC's have complex data paths which makes it hard to identify design bottlenecks and users can observe a drop in system performance post integration. Our key contribution was to develop a performance modeling framework, to analyze the system, isolate the source of the problem and suggest potential design changes with tradeoffs to be made. This is particularly challenging because of the various IPs one has to deal with and the different available bus protocols and parameters that the

system/platform has to support. This framework would be particularly useful in HPS systems that exist in most state-of-the-art FPGA families. And to test the efficiency of our platform, we used it to evaluate HPS system on an Intel FPGA.

- A framework for system level modeling, comprising of third party NoCs, memory controller and other peripherals.
- Plug and play setup to exercise varied customer workloads and swap components.
- It has allowed to create continuum in performance modeling that is portable across current and future FPGA architectures involving the embedded HPS subsystems.
- The performance model is 2-3x faster than the normal simulation model (example OVM aka Open Verification Methodology define OVM basedverification environment) which helps architectural exploration of SoC level NoC design and configuration.
- To demonstrate that Software knobs like Quality of Service (QoS), Hardware NoC knobs like outstanding credits should be set correctly in order to achieve optimal performance within acceptable power bracket on dataflows of interest. Our performance results show 55% average latency reduction compared to baseline architecture.
- Left shift analysis as part of product life cycle to intervene early in design cycle to avoid late changes in SoC design. This enables faster execution and time to production.
- This model will also be used for post-silicon correlation analysis and evaluation efforts.

The rest of the paper is organized as follows. Section II provides brief overview and background of the proposed work. Section III describes HPS Architecture and different components. Section IV goes over the performance modeling methodology and framework used. Section V presents experimental evaluations. Sections VI is overall discussion and Section VII concludes the paper summarizing the key contributions.

II. Background

A cheaper, smarter, and more adaptable industrial automation systems are already transforming manufacturing in a host of different ways. Process platforms, such as a robot arm equipped with a weld gun, power supply, and control electronics, can be standardized, applied, and reused in multiple applications, simplifying programming, maintenance, and product support. Advances in computing power, software, and networking have made assembling, installing, and maintaining robots faster and less costly than before. For example, while sensors and actuators once had to be individually connected to robot controllers with dedicated wiring through terminal racks, connectors, and junction boxes, they now use plug-and-play technologies in which components can be connected using simpler network wiring [5]. Industrial Ethernet is used to connect devices such as *Programmable Logic Controllers (PLCs)*, local and distributed I/O, servo controllers and drives on the plant floor and in industrial facilities. For visualization and monitoring, industrial Ethernet connects PC-based and embedded *Human Machine Interfaces (HMIs)* to controllers, and to each other. It's also used to connect HMIs to the Internet, and to upper-level servers running historian, quality, manufacturing and other enterprise applications.

These industrial Ethernet systems rely on time-synchronized (and timely) communication among sensing, computing, and actuating devices. A control loop, commonly referred in context of industrial automation applications, consists of a source of sensing data, compute/ control element and an actuator as a sink for control data, as shown in Fig. 1. Latency of the overall control cycle includes communication latency and latency within devices for source and sink of data. Latency requirements can vary from microseconds to milliseconds range [6], depending on the application. *Time Sensitive Networking (TSN)* [7] has emerged as an enabling networking technology to achieve precise time synchronization and timeliness in a network. At its fundamental level, TSN is an enhanced version of Ethernet that incorporates the capability to transmit precise time information between different locations, ensuring the delivery of time-sensitive packets on time and without disruption from lower priority traffic. TSN provides deterministic data delivery with bounded latency without loss due to congestion or errors. All devices participating in the TSN network are synchronized to a global time and are aware of a network schedule that dictates when prioritized messages will be forwarded from each switch. The transfer of time is achieved through the implementation of *IEEE 1588 Precision Time Protocol (PTP)*. An example use case for TSN would be in an industrial factory floor. There may be requirements to time synchronize the entire factory to the same sense of time to allow for appropriate synchronization of activities. It often presents the need for priority traffic for low-latency control loops, for instance to send safety shutdown messages. The timed release of messages ensures that delays in the network can be deterministically predicted and managed. This allows for the convergence of critical traffic and non-critical traffic on the same network.

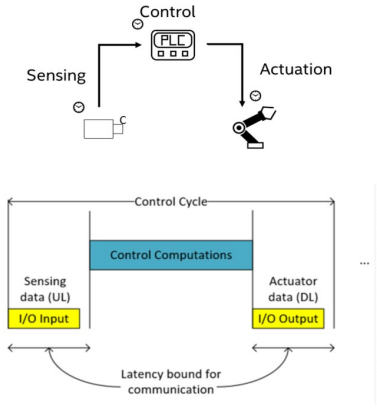

Fig. 1. Latency components within a control loop observed in industrial applications.

A total latency number constitutes not only a network latency but also latency within a device itself. Hence, to minimize the overall latency number, the latencies associated with internal data processing shall be optimized as well. Fig. 2 shows an example of an HPS device, which is sourcing camera sensor data along with Ethernet TSN packets. HPS processes both the raw sensor data from MIPI/USB3 and latency-critical data to/from TSN *Media Access Controller (MAC)*. Data access for software application has varied latency requirements depending on the type of peripheral endpoint. TSN MAC has low latency requirement whereas an image access on MIPI/USB3 port is less time sensitive.

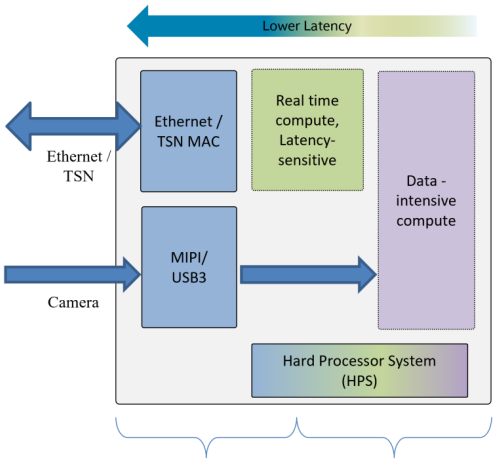

Fig. 2. HPS Latency Comparison highlighting variation in latency requirements based on endpoint.

Due to low latency requirements, SoC requires an efficient method to move data between CPU (central processing unit) and peripherals. TSN MAC, as other high-speed peripherals, utilizes DMA for data transfer to and from system memory. DMA buffer's data structure allocated in system memory serves as common storage place for CPU/ SW application and TSN MAC to access data. Descriptor's ring is used not only for pointing to the location of the data, but also to arbitrate ownership of this data between CPU/SW application and TSN MAC's DMA engine. Hence, data accessing latency may be split into two distinct components. The first component is the hardware related latency of moving data between TSN MAC and system memory. The second component is the CPU/SW application latency for accessing data in the same memory (as shown in Fig. 3).

The first component, related to hardware latency needs to be characterized before chip tape-out and it is a subject of discussion in this paper. Multiple TSN MACs as well as other peripherals like USB3 are sharing the same access to system memory. Hence, SoC architecture needs to consider data throughput during concurrent operation of multiple traffic originators which have high bandwidth requirement. The second latency component (software related) comprises of multiple factors, like workload specifics, Operating System (OS) parameters, system stack etc. Such latency may be optimized by software update and it is not included in the scope of this paper.

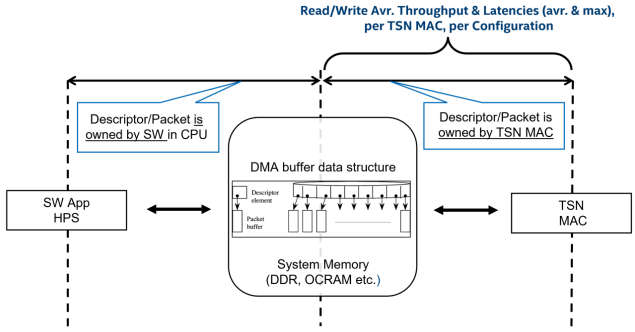

Fig. 3. System memory serves as common storage place for CPU/SW application and TSN MAC to access data.

III. HPS Architecture

This section aims to provide a high-level view of the Hard Processing System that facilitates the fabric to accelerate complex operations in the soft logic with the help of hard cores, easy access to memory and peripherals. We briefly describe the components and their interconnection networks. The key components of HPS Architecture exercised in TSN paths are shown in Fig 4. Peripheral NoC (PSS) provides high speed agents and low speed peripherals to send data over to fabric/ memory via cache coherent unit. Different initiators like TSN, USB, etc., use this NoC to send traffic. It has Quality of Service (QoS) built in to prioritize traffic based on performance requirements. The cache coherency unit acts as an interconnect among the processor, FPGA-to-HPS bridge, memory NoC and peripheral masters interfacing the system interconnect and supports weighted priority of memory accesses. The Multi-Port Front End NoC (MPFE NoC) connects the HPS to the hard memory controller adaptor, I/O and memory controllers, connected to memory models facilitating connections with DDR4, LPDDR4 or LPDDR5, etc. We describe each component in HPS and corresponding interfaces in more detail from the modeling perspective.

Microprocessor unit (MPU) consists of several cores, each of which is equipped with their private caches and a Dynamic Shared Unit (DSU) that facilitates the coherency among the cores, utilizing its Snoop Control Unit (SCU). DSU also has a cache, shared across the cores. The MPU is closely connected with the Application Processing System (APS), equipped with a cache coherent NoC (CCU), an interrupt support unit (GIC), a System Memory Management Unit (SMMU), an On-chip RAM (OCRAM) (APU details are not shown in Fig 4). The CCU is mostly responsible to manage the I/O coherency between the Fabric and peripheral

accesses with SDRAM. The GIC is a generic interrupt controller that handles interrupts from peripherals to the cores and between cores. It manages all the interrupts coming from the SoC and routes them to the proper intended core. It also handles all the core to core interrupts. SMMU translates an input address to an output address, by performing one or more translation table walks. OCRAM provides general purpose memory solution for the on-chip memory accesses.

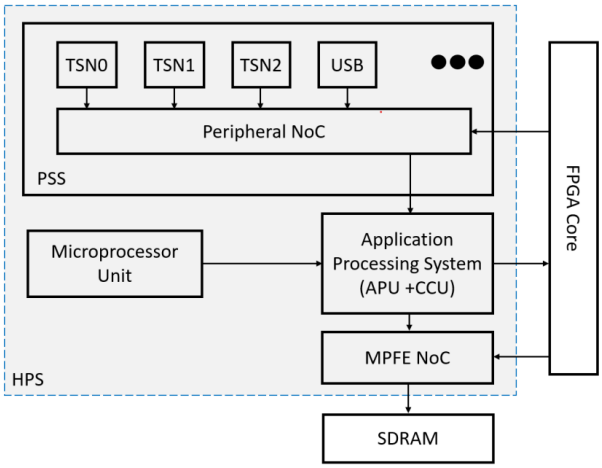

Fig. 4. HPS Architecture high level block diagram. APU: Application Processing Unit, CCU: Cache Coherent NoC Unit, MPFE: Memory-Ported Front End.

Peripheral Subsystem (PSS) is another integral part of the HPS, which consists of several peripherals. Listing all the PSS components here would be beyond the scope of our discussion; hence, we will only focus on the components relevant to our study. TSN MAC Controller supports 2.5G interface when used with Transceiver macro and 1G/100M/10M when used with HPS or Fabric I/O pads. USB3.1 Gen1 controller supports both Device and Host controller modes, USB3.1 and USB2.0 interfaces must both be configured as either device or host, mixing modes is not supported. Peripheral NoC acts as a glue-logic that connects the peripherals with the rest of the HPS via several ports.
Multiported Front End (MPFE) interfaces to the nearest IO96 on all devices. The MPFE provides the interface between the CCU and either one or two IO96 [8]. It supports the following modes of operation. (1) one 16-bit SDRAM channel, (2) one 32-bit SDRAM channel, (3) two 16-bit SDRAM channels utilizing a single IO96, and (4) two 16-bit or two 32-bit SDRAM channels utilizing a two IO96. In addition, it also supports (1) fabric bypass and (2) four 16-bit SDRAM channels utilizing two IO96. MPFE works as the pathway to SDRAM for both HPS components like peripherals, MPUs, etc., and non-HPS components like FPGA fabric.

IV. Performance Modeling

The simulation platform is created by integrating several component models that are developed using SystemC by several vendors (like Synopsys [9], Arteris [10], etc.,), compatible to use with Synopsys proprietary CAD tool "Platform Architect Ultra (PA Tool)". These models can be categorized as Transaction Level Models (TLM 2.0), where transaction level protocols (e.g. Advanced eXtensible Interface (AXI) [11], Coherent Hub Interface (CHI) [12], AXI Coherence extension protocol (ACE) [11], etc.) are implemented accurately in detail using finite state machines on each input and output ports of the module. The PA tool provides a framework for building large platforms, creating workloads, mapping workloads to the hardware and creating simulation scenarios. In addition, it also facilitates results analysis to a great extent. Please refer to Section V-A for more details.

A. Standalone Models

Standalone platforms aim to verify the functionality of a particular module and ensure the key performance indicators (KPIs) are met. In general, any module has one or many initiator ports that let the traffic enter in the module and one or many target ports that let the traffic go out from the module. We attach a combination of generic packet processor and traffic generator with each of the initiator ports. The module logic consumes the transactions in the traffic and sends out the required output traffic through target ports following its internal policy. We attach memory component on the receiver end to mimic the target behavior in the standalone platform.

B. Platform Model

The current HPS platform model focuses on Concurrent TSN Traffic analysis to study performance bottlenecks. The critical paths for TSN applications consist of three main NoCs as described earlier. In this path, the peripheral NoC connects to CCU over CCU IOM port, which then connects to MPFE NoC using the dual DMI ports. The MPFE NoC connects to SDRAM via the memory controller. Each of these NoCs and DDR controller are implemented in detail as cycle approximate TLM 2.0 models, correlated against RTL. MPU, APS and FPGA are represented using abstract delay models as they are not in the scope of our analysis for target use-cases. As part of the platform creation, synthetic workloads are being created as trace files mimicking the traffic through Ethernet ports, following their line rate requirements. In addition, USB traffic is also modeled which is exercised concurrently along with the TSN applications.

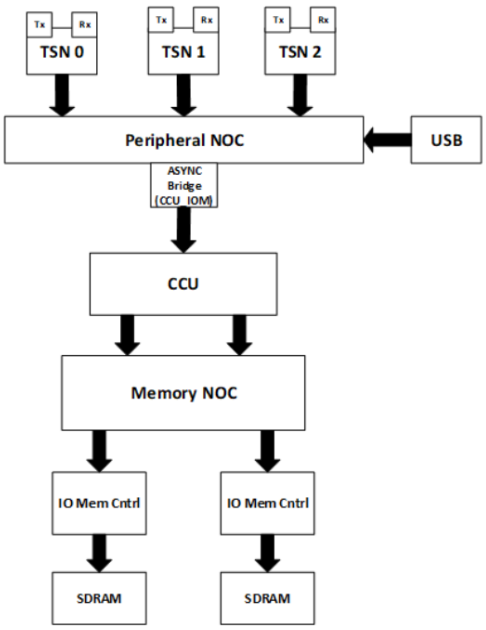

Fig. 5. Concurrent traffic analysis in HPS.

C. TSN Flow in HPS

In Fig 5 the TSN flow on HPS is depicted as a packet flow diagram. As described in Section IV-B, one of the critical customer usecases is for concurrent analysis. The transactions are translated into packets and injected into the PSS NoC through the initiator ports. The packets reach the CCU NoC through asynchronous bridge and CCU NoC sends the packets to the Memory NoC that is connected to the DDR memory. This is the path we analyze in Section V-B1, V-B2. We also do a comparative study on I/O-Coherent traffic going through PSS, using HPS-to-FPGA bridge and then re-enter CCU to reach memory. This analysis is presented in Section V-B3.

V. Evaluation

A. Simulation Setup

We use proprietary PA tool from Synopsys to build the platform using several IPs from different IP vendors. The IPs are ported in the PA platform by creating their parameterized instances. Since the IPs follow Transaction Level Modeling (TLM 2.0) principle, they all need to support a list of common transaction protocols, for instance, AXI, CHI, ACE, etc. The NoC IPs follow their own topology, routing mechanism, credit control, virtual channel, etc., independent of other NoCs and yet connect with each other through ports that support common transaction protocols and data widths. We have connected TLM 2.0 DDR controller models to simulate realistic memory access time.

Synthetic workloads are fed into the virtual processors, which are modeled with decoupled computation and packet processing element, as part of the PA framework. The workload creation is also supported in the PA framework as form of one or multiple task graphs based on the requirements. The tasks are the customizable instances in PA framework that takes several input parameters to generate accurate traffic pattern with required traffic attributes (ex. read buffer enable, write buffer enable, write allocate, etc.). In our study we create Internet Traffic Mix (IMIX) [13], [14] traffic using mix of 1520B, 570B, 64B size transactions in a 7:3:1 ratio, commonly used in industrial Ethernet applications. It is also capable to attach traffic generation engine or traces for customized traffic generations purposes. We have used synthetically generated transaction traces following real application access pattern for the TSN & USB applications. We also generated a set of traces to sweep across the transaction sizes to gather insights regarding performance sweet points with respect to transaction sizes in HPS. Each of the tasks are mapped onto the respective virtual processing elements and based on the designed task graph the traffic is generated to evaluate the platform. We create several scenarios for the simulation using combination of the workloads as Socket Transactional Language (STL) and collect results for further analysis with the help of post processing tool supported by PA simulation framework.

B. Performance Evaluation

We evaluate the TSN and USB endpoints performance for different Ethernet line rates on a combination of one to three Ethernet ports exercised simultaneously along with the USB port. The goal of this evaluation is to ensure that the latency requirements of the TSN application are satisfied while maintaining a steady line rate on each of the Ethernet ports. If TSN isn't able to maintain its latency within bounds, it can start dropping packets. This can be fatal. In an industry factory floor, it could mean transmission loss for functionality critical scenarios. In addition, we also intend to find an optimum packet size capable of providing low latency and high throughput exploiting the available shared resources in the system for a specific system configuration.

The PSS NoC and CCU NoC are orthogonal in behavior in terms of handling packets in HPS. One of them has lowest latency for large packets, while the other one breaks down packets into smaller size chunks while passing through. This behavior results in a contention point when transferring packets from one to the other. In order to resolve this, it is critical to ensure that all ports have correct settings for QoS, outstanding credits. From now on, default setup with NoCs will be referred to as Config-A. Config-B refers to enhanced PSS NoC with QoS settings enabled. Config-C refers to Config-B and CCU NoC with outstanding buffer changes.

1) TSN Line Rate Variation Study: Fig 6 depicts the latency improvement of overall TSN path with changes from Config-B and Config-C applied for a mixed traffic scenario as described earlier (a mix of 1520B, 570B, 64B size transactions in a 7:3:1 ratio). The scenarios on the X-axis are developed considering realistic application line-rate requirements and combination of standard Ethernet 1 Gbps and 2.5 Gbps line rate support. It is evident that the NoC QoS is effective that is resulting more than 40% performance

improvement over the baseline setup for the small and moderate traffic volume. The power and area analysis with and without QoS enabled is the part of our future work. We do not show the throughput here as throughput is almost steady because of constant line rate maintained by the TSN ports. Because of the constant line rate, the QoS policy could work effectively by not hoarding all the resources for low to moderate traffic volume. For the same reason when we have higher line rate for 3×TSN at 2.5G, the traffic volume increases and QoS is shown to be less effective as more resources are engaged. In addition, the overall average transaction time also increases along with the traffic volume. The QoS policy applied, moderates the traffic based on the requirements, assessed through dynamic traffic prioritization. For the bigger transactions we observe that network resources are becoming the bottleneck and hence we increase the outstanding buffer sizes and can sustain the required throughput with minimal latency increase. We suspect that the power consumption of the second optimization will be higher than the baseline. However, we leave quantitative power analysis for our future work.

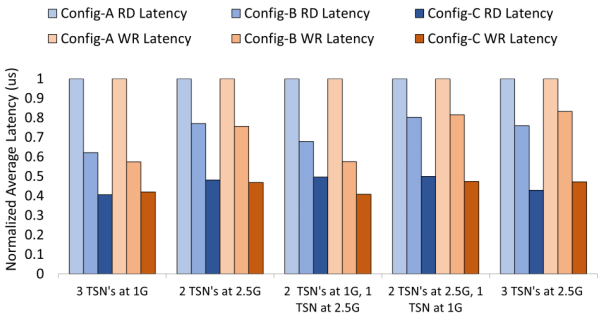

Fig. 6. Variation of normalized average latency (major axis) for 3 concurrent TSN endpoints at different line rates using IMIX traffic (first three bars in each category represent reads).

2) Sensitivity Analysis: Fig 7 depicts latency and throughput variations with the change in packet sizes. In these experiments the number of transactions remains the same, whereas increase in the packet size effectively increases the traffic volume. For low to moderate traffic volume the Config-B is beneficial at lowering latencies for packet sizes greater than 1024B as compared to the Config-A. For the larger packets (1024B, 1520B) this is not the case, however with enhancements to the CCU NoC in Config-C the latency improves significantly. We can see a similar trend in the throughput, as the line rate at larger packets can only be sustainedwith the Config-C enhancements. Overall, the optimized CCU NoC results in a much lower latency and increased throughput at larger packet sizes at the cost of higher power/area consumption within the acceptable limit.

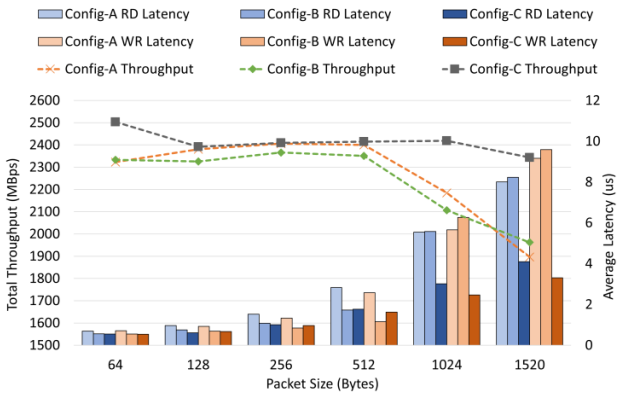

Fig. 7. Variation of total throughput (line, major axis) and average latency (first three bars in each category are read accesses, minor axis) for 3 concurrent TSN endpoints at different packet sizes. Traffic is moderated at 2.5G line rate.

It is apparent from Fig 8 that the throughput differences between the different configurations (A/B/C) are quite significant and that in optimized cases the throughput is well above the TSN minimum requirements. It may be that uncontrolled traffic hoards resources all at once and this issue is exacerbated at larger packet sizes resulting in a declining throughput for packet sizes more than 256B. The results show that the sweet spot for packet size which minimizes latency while maintaining an acceptably high throughput is 256B.

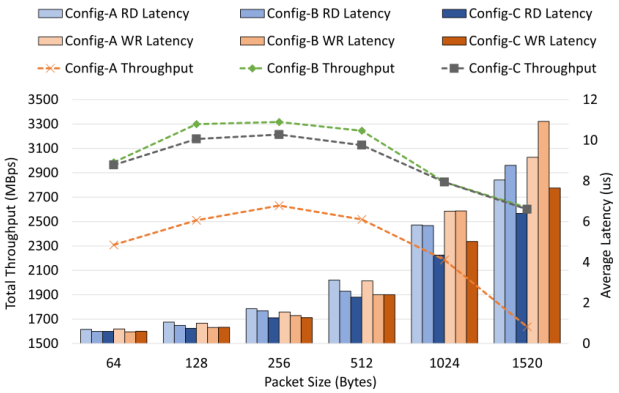

Fig. 8. Variation of total throughput (line, major axis) and average latency (first three bars in each category are read accesses, minor axis) for 3 concurrent TSN endpoints at different packet sizes. Traffic is not moderated (back-to-back without delay).

3) Coherent vs Non-Coherent Traffic Flow Study: Fig 9 compares two plausible data flows for concurrent traffic analysis of TSN & USB accessing SDRAM. In this setup, we exercise 3 TSN endpoints operating at peak line rate of 2.5Gbp/s, sending IMIX packets as described in Section V and USB traffic. The orange bar shows the more direct path to memory via F2SOC port on CCU NoC, whereas the blue bar shows the longer path via CCUIOM port on CCU NoC. We observe average latencies vary close to 50% and 40- 50%

for Reads and Writes respectively. We also observe that the average throughput on F2SOC path far exceeds the CCUIOM path. Thus, we can conclude that for end customer usecase where MPU cache access is not required, it would be preferable to use the HPS-to-FPGA bridge to access CCU NoC via F2SOC path. This would also relieve congestion in case of more endpoints arbitrating for critical resources on cache dataflow. Both these analyses were done on baseline architecture, which also shows that for I/O-Coherent traffic on F2SOC, we do not need to change outstanding credits thereby saving area on the chip.

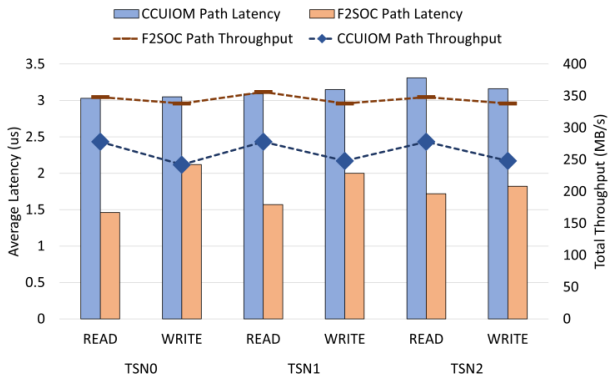

Fig. 9. Comparison of latency and throughput in concurrent traffic scenario with different target endpoints (CCUIOM vs. F2SOC), experimented separately for each path.

VI. Discussion

In this section we attempt to embark upon establishing correlation among the underlying hardware, usecase workload and observed performance in order to highlight how our study optimizes the production cycle time. Each of the hardware modules architecture and design are done keeping their individual Key Performance Indices (KPI) requirements in mind. Hence in ideal testing environment (standalone platform), where the response is instantaneous, provided the request is enqueued, the design usually passes all the regression tests for KPI requirements. Our quickly developed integrated platform, several months before the product RTL, allows to estimate the SoC performance with all the modules integrated in the desired way. The quick development is also possible as the models we use are highly configurable and reusable across products and accurate, providing results with ≈5% to ≈10% error margin. In this paper we consider several industrial use cases and detailed our study by developing a generic set of workloads after distilling the most critical common traits among those use cases. One of the most interesting features of our workload is the mixed nature of the workload, where the requests are made for 64B, 570B, and 1520B in a certain ratio, which demands agility of the underlying hardware. Transactions are sent in multiple bursts in the form of beats, which in turn are converted into packets and flits in the NoCs. In addition, port widths on the initiator and target may also be different, forcing the NoCs to carry out time intensive data width conversions.

For instance, the port width of the peripheral NoC that connects the CCU NoC is 64 bits wide, whereas the target port (DMI port) on the CCU NoC is 256 bit wide, which requires data beats on memory side to be combined after being split on the peripheral side. So, when a request of size 1520B comes either for read or write, around 190 beats (1520*B*/8*B*, 8 byte wide NoC port) are transferred, which overwhelms the network by occupying all its resources (like outstanding buffers, credits, etc.). This creates huge network contention completely disrupting pipelined data transfers. To address data contention in isolation each of the NoCs theoretically decide the number of switches on the path, number of buffers and required credits on each switch on that particular path.

However, we found that in device platform (where all the modules and NoCs are interconnected), the backpressure from the DDR memory exacerbates the delay in response, which in turn significantly worsens the network contention and ultimately halts transaction issued from the TSN ports. To address this issue the obvious intuitive solution is to provide more resources for each of the network. This however comes at the expense of an area and power increase, and we need to carefully analyze the tradeoff. Our workload study and performance estimation using the models helps the architects and designer, providing opportunity to re-calibrate the product architecture and design for target workloads at a very early stage of the product development cycle keeping the area and power envelop in the acceptable range. In future we plan to extend our study for identifying the particular regions or clock domains that are idle for a particular workload exercise, can be dynamically turned off to further optimize the power. We comprehensively study the performance impact of the outstanding buffers and credits keeping the area envelope in mind. Detailed area and power analysis is out of the scope for this paper.

Finding out the system bottlenecks at an early development cycle helps to generate a lot of revenue saving precious development cycles and capturing market early. This methodology can be generalized for any use-case and for a wide range of products by re-configuring the models and their interconnections, as the IP models are robust and reusable. So, our methodology not only helps to generate revenue in one production cycle, once in motion, it improves over generations of products across variety of product ranges amortizing the product budget itself.

VIII. Conclusion

In this work we propose a scalable modular performance model to characterize any end customer usecases on the desired platform of choice. We show that our model can be used to do quick design space exploration, it can support

diverse industrial field bus protocols and highlight various options for architecture optimization. It has allowed to create continuum in performance modeling that is portable across current and future FPGA architectures involving the embedded HPS subsystems. Our performance optimization results show 55% average latency reduction compared to baseline architecture. This model will also be used for post-silicon correlation analysis and evaluation efforts. We also plan to add power and area analysis in the future. The performance model is 2-3x faster than the normal simulation model (example OVM based verification environment) which helps architectural exploration of SoC level NoC design and configuration. The current work avoids high-cost late architectural changes in the design cycle and/or silicon re-spin.

IX. References

[1] AMD. [Online]. Available: {https://www.xilinx.com/applications/industrial.html}
[2] Intel. [Online]. Available: {https://www.intel.com/content/www/us/en/industrial-automation/programmable/applications/overview.html}
[3] Intel. [Online]. Available: {https://www.intel.com/content/www/us/en/docs/programmable/683567/23-1/introduction-to-the-hard-processor-system-51709.html}
[4] Wikipedia. [Online]. Available: {https://en.wikipedia.org/wiki/Fourth Industrial Revolution}
[5] McKinsey. [Online]. Available: {https://www.mckinsey.com/capabilities/operations/our-insights/automation-robotics-and-the-factory-of-the-future}
[6] Avnu. [Online]. Available: {https://avnu.org/wp-content/uploads/2014/05/2014-12-04-Machine-Control-Use-Case-v10_Final_Approved.pdf}
[7] TSNTaskGroup. [Online]. Available: {https://1.ieee802.org/tsn/}
[8] Intel. [Online]. Available: {https://www.intel.com/content/www/us/en/docs/programmable/683567/23-1/hps-block-diagram-falconmesa-stratix-10.html}
[9] Synopsys. [Online]. Available: {https://www.synopsys.com/}
[10] Arteris. [Online]. Available: {https://www.arteris.com/}
[11] ARM. [Online]. Available: {https://developer.arm.com/documentation/ihi0022/e/}
[12] ARM-CHI-Documentation. [Online]. Available: {https://developer.arm.com/documentation/ihi0050/latest/}
[13] N. Kero, T. Muller, T. Kernen, and M. Deniaud, "Analysis of precision ¨time protocol (ptp) locking time on non-ptp networks for generator locking over ip," *SMPTE Motion Imaging Journal*, vol. 123, no. 2, pp. 37–47, 2014.
[14] C. Popoviciu, E. Abegnoli, and P. Grossetete. [Online]. Available: {https://www.networkworld.com/article/813362/lan-wan-chapter-11-network-performance-considerations-coexistence-of-ipv4-and-ipv6.html}