Data-Driven Design for Adaptive Multi-Die SoC

Ankita Roy
Silicon Domain Architect
Intel Corporation
Bangalore, India

Vikrant Kapila
Principal Engineer/Director
Intel Corporation
San Jose, United States

Abstract- **The Altera (Intel) Agilex™ adaptive System-on-Chips (SoCs) range integrates pre-optimized hardware blocks (CPU, DSP, NoC, memory, I/O) with flexible, reconfigurable fabrics, enabling customizable hardware execution architectures. This combination achieves high performance and efficiency, allowing post-production customization and swift workload adaptation. Optimizing performance and power efficiency for heterogeneous SoCs requires addressing intricate architectural challenges, particularly in multi-die configurations. This involves disaggregating IPs based on workload demands, communication needs, and power budgets. A prominent example is the definition of a scalable Network-on-Chip (NoC) topology, necessitating efficient allocation and management of diverse resources across multiple dies. We propose a data-driven approach using Synopsys Platform Architect® for early Design Space Exploration (DSE). This enables: i) Scalable architecture strategies: Optimizing resource allocation across multiple dies based on diverse application requirements. (ii) Advanced workload modeling: Accurately predicting performance within complex multi-die architectures. (iii) Cross-die communication trade-offs: Balancing power consumption and performance through efficient resource allocation and communication pathways. This work unlocks the true potential of adaptive SoCs, empowering developers to design efficient, high-performance, and workload-optimized multi-die architectures for next-generation applications. Our methodology and analysis strategy mainly assist in architectural exploration based on dynamic workload simulation which is more realistic as compared to static analysis done by architects usually through spreadsheets. This approach (i) reduces design iterations with early performance prediction and resource optimization accelerating time-to-market significantly. (ii) is highly scalable and applicable to any multi-die SoC DSE.**

I. INTRODUCTION

Intel® Altera® possesses a distinctive competitive edge in delivering FPGA device innovation through a unique approach that employs tiles or chiplets to integrate various process technologies, capabilities, and suppliers into a single device. This innovative heterogeneous architecture leverages an extensive library of FPGA chiplets designed for processing, storing, and moving data. Additionally, it includes custom or specialized chiplets to facilitate rapid FPGA silicon development and introduction. This concept, termed “Any-to-Any Heterogeneous Integration,” allows any value-added function, including custom ASICs, to be integrated into the FPGA. Currently, several programs are underway that integrate custom ASICs into the FPGA.

The Network on Chip (NoC) plays a crucial role in multi-die systems, managing heterogeneous System on Chips (SoCs) and ensuring efficient utilization of system resources. In FPGA-based devices, which must support a diverse array of applications, the NoC architecture needs to be highly scalable. There are two primary use cases for multi-die systems: (i) disaggregating a large monolithic SoC into smaller chiplets due to the reticle limits of semiconductor manufacturing equipment or yield issues with very large dies, making it more economical to fabricate multiple smaller dies—this is applicable to large data center CPU and AI accelerator SoCs; and (ii) integrating discrete chips into a single package that were previously assembled on a Printed Circuit Board (PCB). Die-to-die communication in a multi-die system offers significantly lower power consumption and better throughput compared to chip-to-chip communication over a PCB, with notable applications in the networking domain, such as co-packaged optics.

For both disaggregation and integration, the common advantages include quicker assembly of products with reduced risk using existing dies, enabling rapid creation of new product variants [5]. Figure 1 illustrates the evolution from single-die to multi-die designs using reusable building blocks, which expedite time to market for SoC families while optimizing R&D investments. In Figure 1, the red boundary surrounding the SoC and vertical columns represent the high-speed NoC topology. The NoC connects to the Secure Device Manager (SDM), I/O, memory, PCIe, and Ethernet chiplets at the boundary and to the blue configurable fabric at the center of the die.

Through simulating thousands of traffic flows across dies and various NoC interfaces, we have precisely defined microarchitectural specifications such as buffer sizes and quality of service. The Task Graph-based traffic generation flow accurately replicates real-world application use cases, allowing early estimation of architectural constraints in the design cycle. Moreover, converting silicon-captured traces into SystemC-based task graphs has enabled the correlation of customer-experienced performance bottlenecks and accelerated the resolution of architectural challenges.

Complexity of Multi-Die Designs: Multi-die designs are significantly more complex than monolithic SoCs. They incorporate multiple Networks on Chip (NoCs) and various peripherals with unique functionalities, such as memory controllers, transceivers, processor cores, crypto engines, and Secure Device Managers (SDM). Efficient distribution of components across these NoCs is crucial. This complexity necessitates a scalable and flexible Design Space Exploration (DSE) framework capable of integrating multiple custom and third-party modules for comprehensive architectural analysis [4,5]. Our methodology addresses this by leveraging a unified platform that utilizes SystemC-based performance models [13], developed both in-house and by external IP vendors, to perform DSE on the entire SoC architecture. Additionally, it enables the exploration of various hardware/software configurations through a series of simulations, minimizing the need for frequent codebase changes, and allows for the comparison of Key Performance Indicators (KPIs) through effective visualization.

This paper focuses on the performance exploration of NoC across various data paths, highlighting its scalability and efficiency in handling diverse applications in multi-die systems. This challenge is particularly unique for heterogeneous FPGA-based solutions, as users building custom workloads [6], such as AI, networking, and embedded applications, have specific requirements for performance and the placement of logic in the FPGA fabric. They need a fast performance DSE solution to conduct what-if analyses regarding the placement of ASIC chiplets around the periphery of the FPGA fabric. If workload components that interact frequently are placed across different dies, they may consume excessive power and deliver low performance due to increased distance and latency.

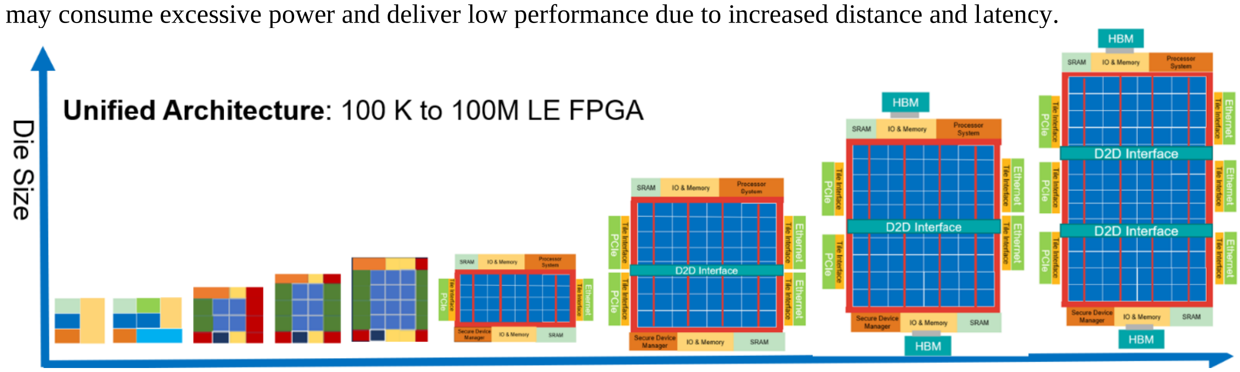


Figure 1. Scalable Architectural Evolution from Single to Multi-Die

II. ARCHITECTURAL CHALLENGES OF MULTI DIE SOC

A. IP selection, Location, Connectivity

To efficiently architect a multi-die device, the initial step involves selecting IP blocks and positioning them on the SoC according to customer and project requirements. Following IP selection, it is crucial to assess the impact of various IP configurations and connectivity on system performance, area, cost, and other design aspects. For instance, early performance analysis based on the location of initiators and targets can estimate the required buffer sizes, outstanding and predict system KPIs for targeted user workloads [6]. Additionally, the connectivity between multiple initiators and targets on an SoC directly affects the number of wires and the overall cost.

B. Protocol & Configuration of Die-to-Die interfaces

When multiple IPs are connected to one or more interconnects, it is essential to identify the communication protocol for each interface to achieve a well-balanced tradeoff between system and IP-level power, performance, and area (PPA). The choice of protocol determines the number of communication channels and wires. Additionally, interface widths and signals are closely tied to the selected protocol. Evaluating the impact of various configurations on the NoC is also crucial, as it significantly affects power, performance, and area, which are essential for SoC architecture. For example, selecting the appropriate QoS mechanisms and configurations is vital to meet customers' performance requirements.

C. HW/SW Partitioning

Another essential step in developing SoC architecture is hardware/software partitioning [2], which should be conducted early in the design cycle to avoid expensive hardware changes later. This HW/SW tradeoff is frequently neglected in the traditional hardware-first approach, adversely affecting time to market, design quality, and system performance. In the FPGA market, this process also involves deciding whether a specific function should be implemented as an ASIC chiplet or left as a soft logic RTL implementation mapped to the FPGA fabric. This decision provides customers with the flexibility to evolve their implementation, combining the adaptability of software with the performance of hardware.

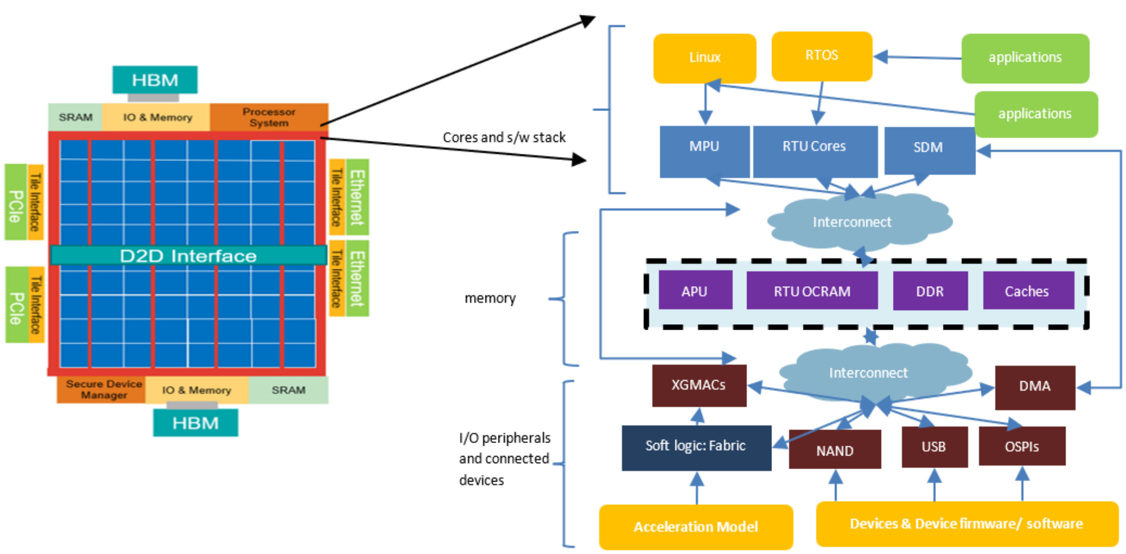


Figure 2. DSE with HW-SW Co-design

D. *Routing, Place & Route algorithm*

Quartus® dynamically performs place and route for customer RTL mapped to the FPGA fabric to optimize resource utilization and achieve maximum frequency (Fmax) for FPGA designs [14]. With the recent introduction of heterogeneous chiplets across the FPGA boundary connected via NoC, it is crucial to consider the placement of customer RTL in relation to peripheral functions, such as Crypto, DDR, and Processor, as NoC latency and throughput depend on the distance. RTL-based simulations are too slow for this purpose, so a faster simulation approach is used. We have developed an ML XGBoost library-based NoC performance predictor model, trained with millions of SystemC simulations. This model enables place and route algorithms to estimate the performance of specific customer use-cases and RTL placement on the FPGA within 3 seconds.

E. Interconnect and Memory dimensioning

One of the primary architecture challenges in designing SoCs is the interconnect and memory dimensioning. Achieving an optimal balance between interconnect bandwidth, memory capacity, latency, and power consumption is crucial for meeting performance requirements and system efficiency. This challenge involves several key considerations:

1) **Interconnect Topology Selection**: Choosing the right interconnect topology (e.g., mesh, ring, or tree) is essential for ensuring efficient communication between various IP blocks and cores. The topology must be scalable and capable of handling the anticipated data traffic without becoming a bottleneck.
2) **Bandwidth Allocation**: Allocating sufficient bandwidth to critical data paths while avoiding over-provisioning is vital. This involves analyzing data traffic patterns and performance requirements to ensure that high-priority tasks receive the necessary bandwidth without impacting other operations.
3) **Latency Management**: Minimizing latency is crucial for applications requiring real-time performance. This requires careful planning of data paths, optimizing routing algorithms, and potentially implementing Quality of Service (QoS) mechanisms to prioritize latency-sensitive traffic.
4) **Memory Hierarchy Design**: Designing an efficient memory hierarchy that balances capacity, speed, and power consumption is a significant challenge. This includes decisions on the size and placement of caches, the type of memory used (e.g., SRAM, DRAM), and the integration of non-volatile memory if necessary.
5) **Power Consumption**: Interconnects and memory are major contributors to overall power consumption. Therefore, implementing power-efficient designs, such as dynamic voltage and frequency scaling (DVFS) and power gating, is essential to manage power budgets and extend battery life in portable devices. In addition, mechanism like Fence and Drain need to be enabled to provide isolation from peripherals communicating with NoC.
6) **Scalability**: The interconnect and memory architecture must be scalable to accommodate future expansions and enhancements. This includes supporting additional cores, increased memory capacity, and new IP blocks without requiring significant redesigns.

F. System-level PnP Tradeoffs

Optimizing system operation requires a careful balance between performance and resource utilization. Efficient system modeling demands an even distribution of workloads across both hardware and software resources. Overusing a limited set of resources while leaving others idle results in a suboptimal design. In multi-die systems, it becomes even more critical to effectively segregate and distribute hardware and software resources. Strategically allocating workloads to maximize the benefits of both hardware and software components across multiple dies is crucial. For instance, in FPGAs where DDR/HBM channels are positioned at the north and south peripheries of the chip, it is essential to allocate customer workloads such that the upper section interacts with the north channels and the lower section with the south channels, rather than using a single monolithic engine. The concept of splitting workloads into microservices and scheduling these microservices close to the underlying hardware resources presents a challenging yet effective technique for achieving optimal performance and utilization from the system.

There are additional architectural challenges e.g Selection of technologies for each chiplet and selection of packaging technology which are not discussed in this paper in interest of space.

III. FPGA BUILDING BLOCKS AND MODELING METHODOLOGY

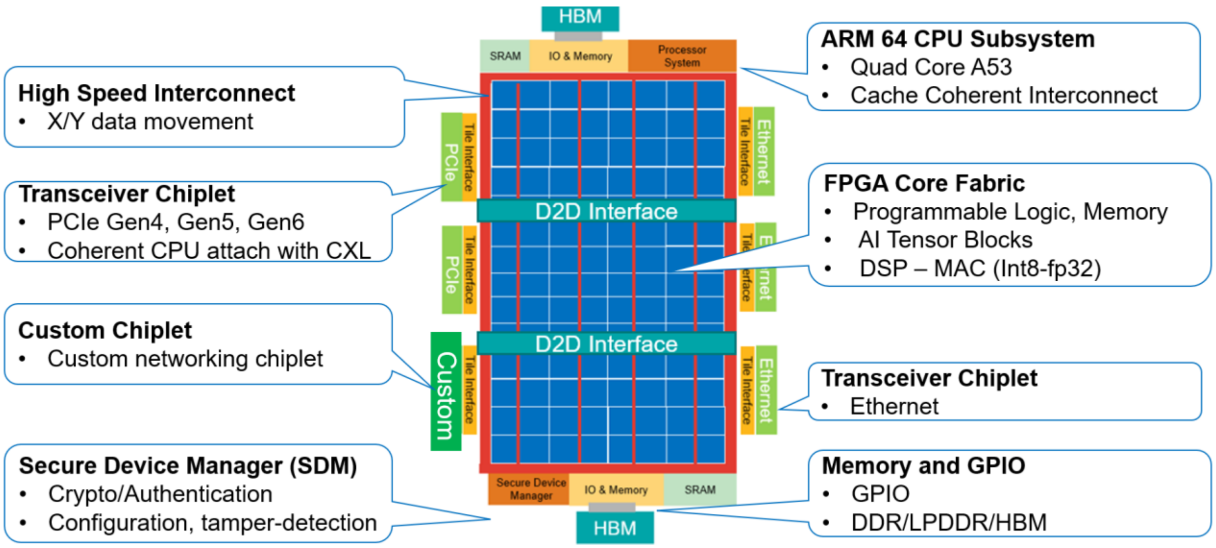


Figure 3. Multi Die FPGA Building Blocks

Figure 3 calls out heterogeneous building blocks of a multi die FPGA device as follows-

NoC: The high-speed interconnect surrounds the device's boundary, enabling both horizontal and vertical data movement. This interconnect facilitates communication between the blue configurable logic and various peripherals across multiple dies. The MNoC includes horizontal (HNoC) and vertical (VNoC) data paths for full crossbar communication. Our proposed methodology employs a SystemC TLM2.0-based model, providing user-configurable parameters that closely mirror the actual hardware.

Fabric and M20K: The blue region contains programmable customer logic and internal embedded memory blocks. An in-house model is utilized to access these M20K memory blocks. We also develop a SystemC TLM2.0-based performance model for this purpose.

Hard Processor Subsystem (HPS): The HPS comprises of a cache coherent unit (CCU), an L3 interconnect, and multi-core Arm Cortex processors. It communicates with the programmable fabric through the high-speed NoC (MNoC) mentioned earlier. The L3 interconnect and CCU are modeled using third-party components. Our methodology uses a SystemC TLM2.0-based performance model for the L3 interconnect and a verilated model of the CCU.

Memory Controller: The MNoC directly interfaces with various memory controllers, including DDR, LPDDR, and HBM. For DDR and LPDDR, we use third-party SystemC TLM2.0-based models, while HBM controller performance is correlated using in-house RTL.

Transceiver and Networking Chiplet: Heterogeneous transceiver tiles are connected to the VNoC portion of the MNoC. These transceiver chiplets include Ethernet, PCIe, and CXL. We employ a combination of in-house and third-party SystemC TLM2.0-based models for the transceivers.

Architecting an optimized, cost effective and area efficient D2D FPGA device requires evaluation of PPA on each IP block mentioned above as well as system level analysis is also essential. The challenges outlined in previous section

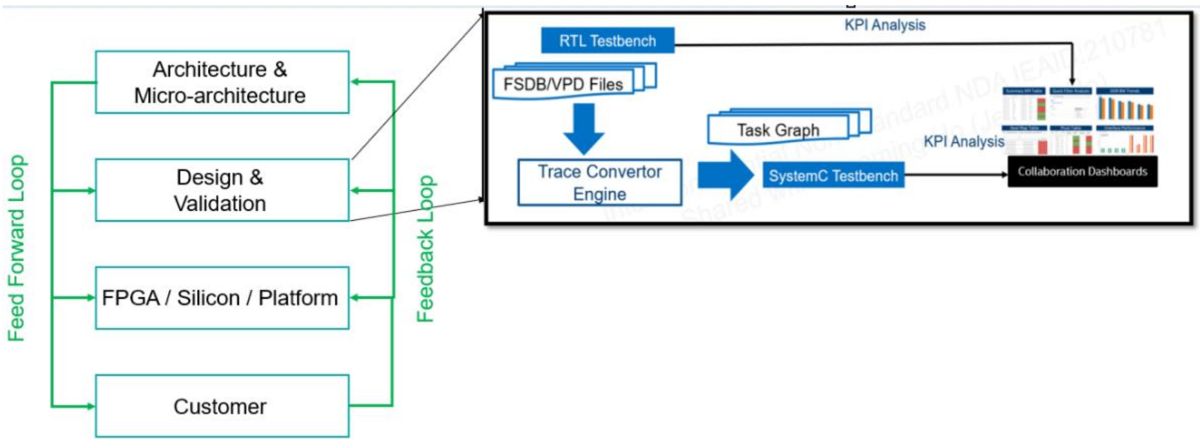


Figure 4. End-to-End DSE Flow

clearly necessitate a scalable framework that facilitates rapid early design space exploration at a higher abstraction level while remaining accurate enough to reflect real-world applications [15]. To predict the performance and behavior of the SoC including interconnect and memory subsystems under various workloads, precise simulation and modeling tools are essential. These tools help identify potential bottlenecks and optimize the design before physical implementation.

Tackling these challenges demands a holistic approach that blends advanced simulation methods, machine learning models for performance forecasting, and tight collaboration between hardware and software teams to ensure that the final design meets all performance, power, and cost goals. In our proposed methodology, we leverage the Synopsys Platform Architect Tool [13], particularly for its robust availability of both generic and third-party IPs. Each IP/Chiplet in the FPGA platform is modeled using SystemC TLM 2.0 FT modeling, equipped with performance instrumentation and root cause analysis capabilities. It offers scalability from monolithic to multiple versions of D2D device with heterogeneous peripherals as depicted in Figure 3. Parameter based hardware configurability, customizable connectivity among IPs and re-usability of chiplets in the form of hardware encapsulations as well as workload models make it convenient to perform DSE on various D2D devices with less turnaround time without need of codebase change.

IV. RESULTS

This scalable approach helps closely monitor key performance indicators (KPIs) for complex SoC architectures [9,10]. We sweep through application configurations using traffic generators to find out sweet spot for performance in existing hardware before proposing alterations. In case of not achieving favorable performance through application mitigation, hardware configurations are refined and same set of workloads is simulated on modified hardware resources to find performance/area/cost tradeoff. As shown in TABLE I, a minimum of 1568 number of scenarios are simulated for a use case consisting of two active traffic paths from one initiator to two targets. For single traffic path from an initiator to a target, a minimum 784 number of scenarios are simulated. In our ongoing exploration on full-fledged NoC topology consisting of hundreds of initiators and targets, the total number of simulated scenarios has turned out to be close to 10000. Such enormous amount of data plays a tremendous role in identifying minute architectural challenges and sensitivity to various types of stimuli. In later part of the paper, few synthetic use cases are listed which are simulated for any crossbar NoC to start with.

TABLE I
SIMULATED TRAFFIC CHARACTERISTICS

	Parameter	Value	No. of scenarios
(1)	Packet Size	64B, 128B, 256B, 512B, 1KB, 2KB, 4KB	7
(2)	AXI ID	Fixed Random	2
(3)	Access Pattern	100% Read 100% Write 70% Read-30% Write 50% Read-50% Write	4
(4)	Address Pattern (For multi target use case)	Sequential Random	2
(5)	Target Pattern	Sequential Random	2
(6)	QoS Priority	0-7	7
	Total	(1)x(2)x(3)x(4)x(5)x(6)x(7)	1568

TABLE II lists the primary and optional KPIs measured for benchmarking of NoC, as well as other IPs.

TABLE II
KPIS MEASURED FOR PERFORMANCE BENCHMARKING

KPI	Measured Metric
Latency	Min Max Average
Throughput	Average (over specific simulation duration)
NoC Buffer Utilization (For Debug in case of performance bottlenecks)	Max Average (over specific simulation duration)
Memory Efficiency	Average (over specific simulation duration)

Following are summaries of few of the experiments performed-

A. Impact of positions of hardware resources across the interconnect on latency, utilization and throughput

Physical positions of resources across NoC shoreline impact effective utilization. As distance increases between initiators and targets across NoC, average roundtrip latency increases gradually. If buffer size is not sufficient in initiators, it becomes a challenge to sustain maximum throughput for longer duration. As shown in Figure 5, we simulated nearest to farthest path across the NoC and a gradual increase in latency was observed. After a certain distance loss in throughput was observed as latency increased beyond a threshold. When we increased reorder buffer size and number of max outstanding transactions, we could sustain throughput till the farthest target. This experiment was performed for traffic characteristics listed in previous table. As packet size increases, increase in effective utilization serves as a savior of throughput. In this experiment, we didn't only validate various traffic configurations, but we experimented with memory controller configuration as well. While DDR5 uses single channel (only red), LPDDR5 communicates through two channels (both blue and red). On the other hand, operating frequency of DDR5 and LPDDR5 are slightly different as well. As a result, between two memory controller modes, number of active targets and active data paths inside NoC are different. This experiment helped in 1) positioning latency sensitive cores across NoC efficiently. Also, it helped understanding how much additional area in terms of buffer size is required to sustain optimum throughput which was a key factor in deciding performance and area tradeoff. Figure 6 shows latency vs distance curve, where each point on the curve- wl_1x1_<position of initiator><position of target> represents latency and for each set, AXI packet size is varied from 64B to 4KB.

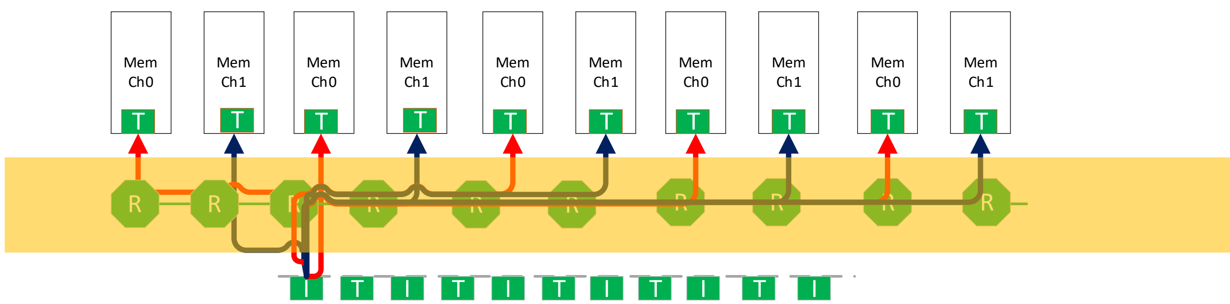
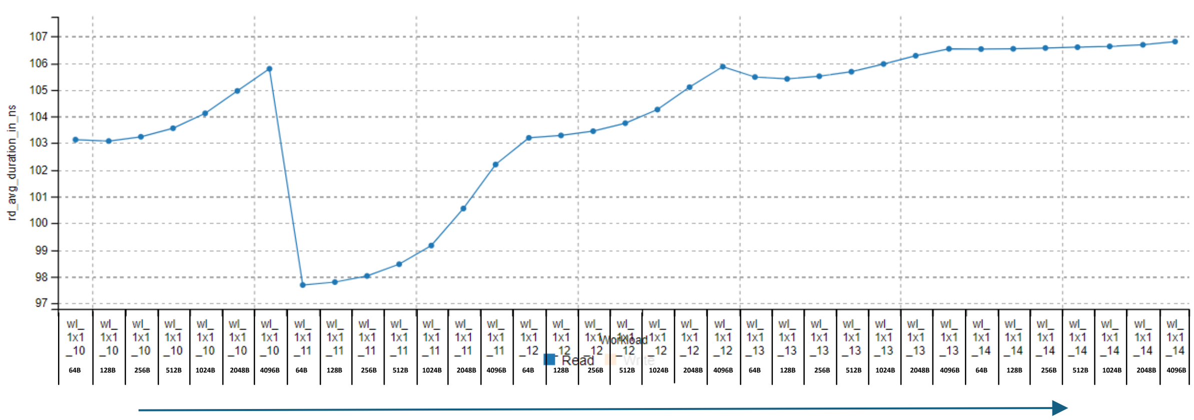


Figure 5. Distance Sweep on Horizontal NoC



Positions of initiator and target, AXI packet size

Figure 6. Impact of horizontal distance on Latency

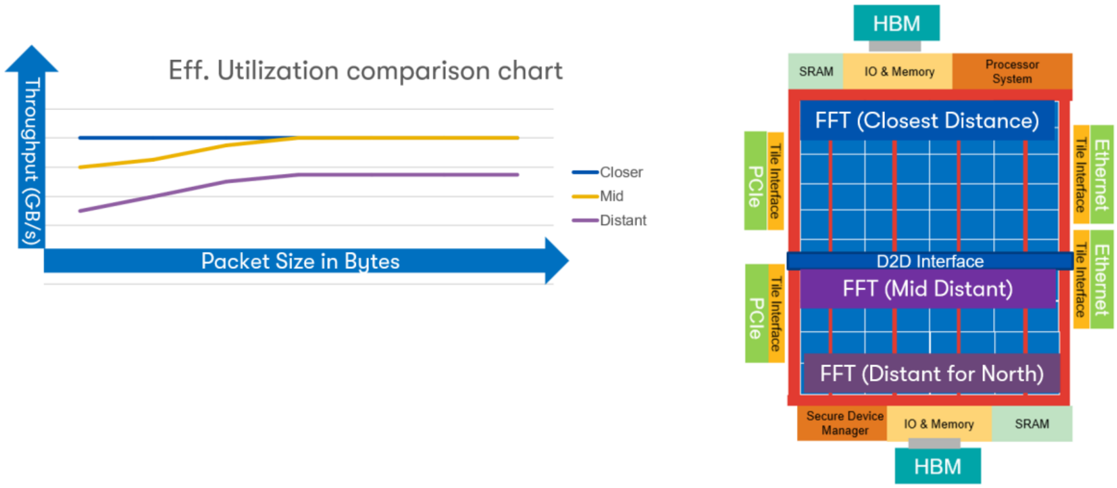


Figure 7. Impact of distance of Die-to-Die Interface on Latency

Figure 7 represents effective utilization of buffers and memory controller on shoreline as distance increases from top to bottom on a multi die SoC with increasing packet sizes. In such scenario,

1) One way to increase throughput and effective utilization is to increase max outstanding transaction capacity and buffer size of initiator and targets. It is done only if the application requires the specific initiator to achieve more bandwidth which it couldn't, due to buffer size constraint.

2) However, if the initiator is already able to achieve required bandwidth, another way to utilize the resources is to position other peripherals in such a way that necessary concurrent operations can be performed enabling more clients across the NoC.

B. Impact of memory access pattern on throughput, utilization and latency-

Figure 8 shows impact of access pattern in memory controller on latency as well as effective utilization with and without NoC for various packet sizes. Although, it is highly preferrable to have interconnects in Die to Die device due to requirement of any to any connectivity, a detailed analysis on this experiment helped us analyze-

1) Loss of utilization due to increase in latency when memory access pattern is random.

2) Impact of NoC on utilization and latency of overall path as well as memory controller.

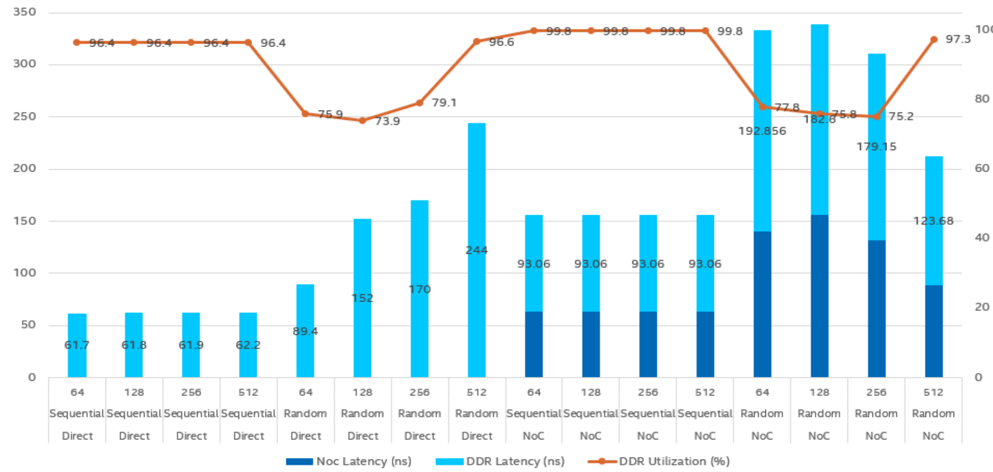


Figure 8. Impact of DDR Addressing Pattern on KPIs

C. Impact of congestion on throughput, utilization and latency-

As shown in Figure 9, in this experiment, multiple initiators and targets are active concurrently and each initiator sends out traffic to more than one target simultaneously thus creating congestion on the routers inside NoC. The targeted bandwidth distribution from each initiator is equal towards each memory controller, that is 25% of total memory bandwidth. With traffic configurations such as different AXI ID variation, we tend to stress reorder buffer. On the other hand, with increase in AXI packet size, max outstanding transaction budget is consumed faster which has vivid impact on performance.

The goal of this experiment is to-

1) Verify if current reorder buffer, number of supported max outstanding transactions can sustain throughput.
2) Validate if current QoS specifications are enough to sustain throughput within latency budget.
3) Identify if there is a congestion and for which workload configurations the impact is not sustainable for customer requirements.
4) If any latency or throughput sensitive client is suffering due to congestion and it's way out.

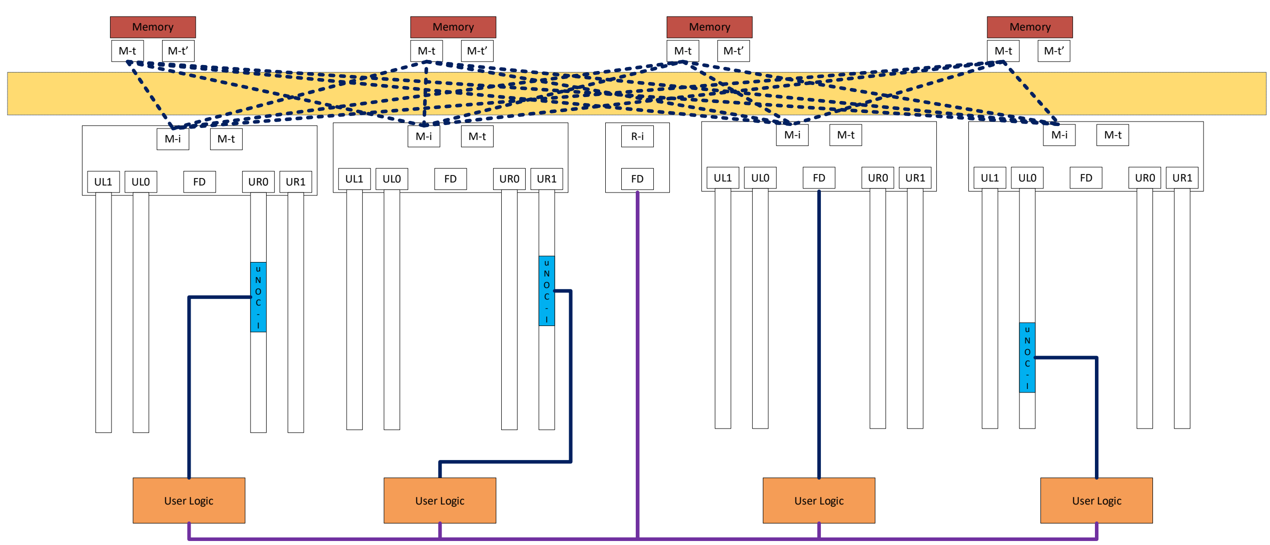


Figure 9. Multi Initiator Multi Target Inter Group Switching

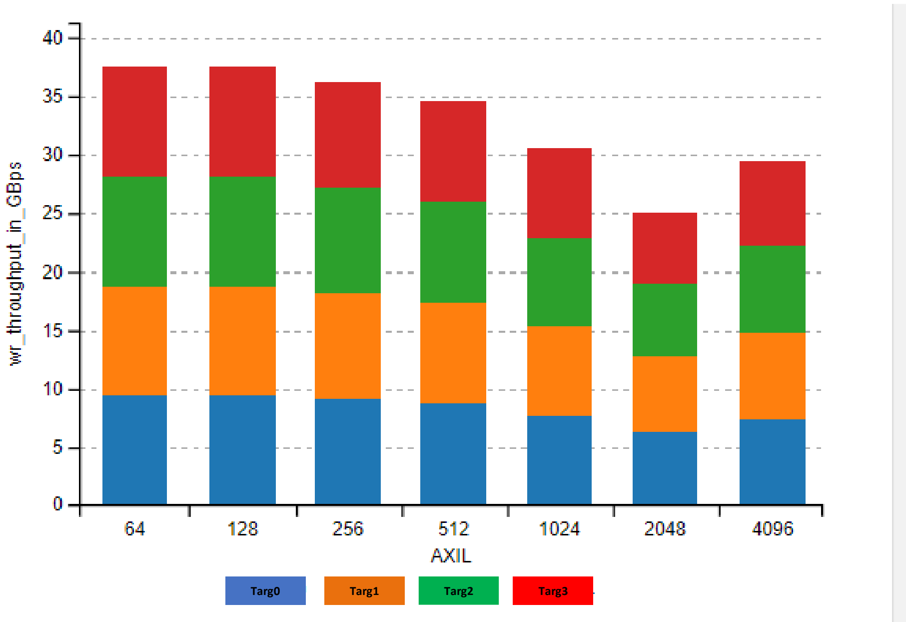


Figure 10. Achieved throughput distribution curve on initiators for 4x4 traffic mapping

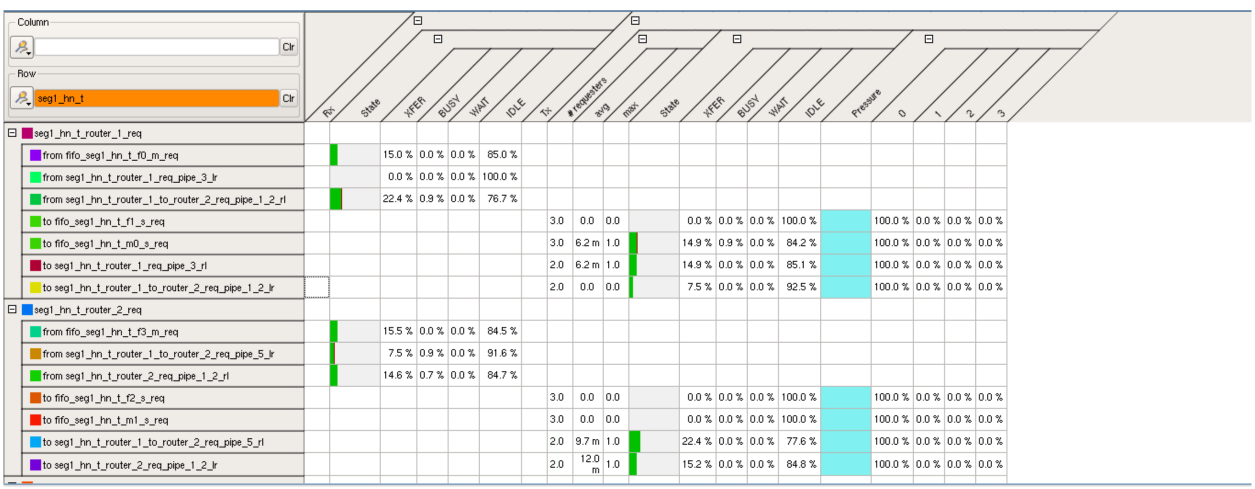


Figure 11. VCD trace to visualize health of NoC internal components

As shown Figure 10, each target contributes equally to achieved throughput on corresponding initiators, which is expected. However, we see that throughput reduces with increasing packet size, also we observed reverse behavior when compared 4KB to 2KB packet size. At this point, we visualize health of NoC's internal components to root-cause bottlenecks.

Figure 11 depicts visualization of internal components of NoC. Fill levels of reorder buffers, utilization of routers, measure of back-pressured traffic etc. act as significant indicators to identify cause of bottlenecks, high latency, low utilization and overall congestion. As shown in the picture, we can clearly understand how much of a router's bandwidth is consumed by each client (bridge, fifo, neighbouring router) connected to it and also, the amount of traffic in idle or busy state from each component. Dynamic QoS status over a simulation duration can also be visualized in terms of pressure levels inside the NoC.

As mentioned, above are some of the synthetic use cases that we simulate to benchmark the NoC. As FPGA offers flexibility to customers in terms of application mapping, it is vital to benchmark each IP and call out corner cases during product release. However, as mentioned earlier, we simulate customer specific use cases to sign off each IP and the device as well. We also convert post silicon traces into SystemC based task graphs to root-cause performance issues observed at later stage of product development. To maintain confidentiality of ongoing project, end-to-end use cases are not mentioned.

V. FUTURE WORK

This paper concentrated on well defined use case and data analysis flow for performance validation and silicon correlation. However, we're working on power modeling as well, which is another area of challenge in defining architecture of multi die SoC. We will be leveraging the same methodology for early power estimation and analyze impact of hardware placement, resource allocation etc. to identify end to end power, performance and area tradeoff.

VI. CONCLUSION

Multi-Die systems are becoming increasingly prevalent due to following advantages-

1) It offers efficient AI and compute disaggregation on large scale SoCs.

2) It helps partition system functionalities into different dies and components to create a sustainable and reusable design for multiple generations of devices.

However, these are only achievable if complex architectural challenges are identified by early DSE using highly configurable performance and workload models. On the other hand, it also enables different sets of customers to specify system use case flows that serves as an effective feedback loop between architects and customers. This methodology guides realization of critical architectural decisions using extensive what-if analysis. The end-to-end flow accelerates time-to-market of new devices with optimal power, performance and area (PPA) trade-off.

ACKNOWLEDGMENT

We would like to thank other members of the team Pritam Majumder, Ramasubramaniyan Venkatraman, Johanan Agarwal, Mohit Jain and our partners from Synopsys and Arteris.

REFERENCES

[1] R.K. Gupta, "CO-synthesis of Hardware and Software for Digital Embedded Systems", Kluwer Academic Publishers, 1995.
[2] W. Wolf, "Hardware-software co-design of embedded systems" in Proc. IEEE, Vol. 82, July 1994
[3] K. Keutzer, S. Malik, A.R. Newton, J.M. Rabaey, A. Sangiovanni-Vincentelli.System-level design: Orthogonalization of concerns and platform-based design. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 19(12):1523–1543, December 2000
[4] M. Jersak, R. Henia, R. Ernst. Context-Aware Performance Analysis for Efficient Embedded System Design. In "Proc. Int. Conf. on Design, Automation and Test in Europe(DATE)", 2004.
[5] Dr. Tim Kogel, Dr. Holger Keding, "Early Architecture Performance and Power Analysis of Multi-Die Systems"
[6] Vikrant Kapila, Dr. Tim Kogel, Dr. Holger Keding… "Case-Study: Generating a Workload Model of the Chrome Browser From Android Execution Traces for Early Analysis of Power and Performance Trade-Offs" http://events.dvcon.org/events/proceedings.aspx?id=241--1
[7] Ankita Roy, Vikrant Kapila, Ashish Gupta, Rahul Pal, "A Novel Network On Chip Architecture For FPGA Smart NIC", https://ieeexplore.ieee.org/document/10276404
[8] Kenneth Hoste, Lieven Eeckhout, "Microarchitecture-Independent Workload Characterization", IEEE Micro, Volume 27, Issue 3 , May-June 2007, https://ieeexplore.ieee.org/document/4292057

[9] http://semiengineering.com/software-modeling-goes-mainstream/
[10] http://semiengineering.com/software-modeling-and-kpi/
[11] https://developer.android.com/studio/command-line/adb.html
[12] http://events.dvcon.org/India/2016/proceedings/papers/10_2.pdf
[13] https://www.synopsys.com/verification/virtual-prototyping/platform-architect.html
[14] https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime.html
[15] Vikrant Kapila, Ingo Volkening, Anant Raj Gupta, "Accelerating Design Cycles with Software Workload Models and Cycle Accurate Hardware Models – A Home Gateway SoC Case Study", SNUG 2019