Novel Infrastructure Design for multi-FPGA Prototyping system

Vandana Singh
Intel Corporation
Banagalore, India

Vani V
Intel Corporation
Bangalore, India

Ranjith Kulai
Intel Corporation
Banagalore, India

Gregory Matthew James
Intel Corporation
Bangalore,India

Abstract*—In a multi-FPGA based prototyping, an SoC is partitioned into multiple modules. Each FPGA contains a partitioned module in addition to FPGA specific infrastructure components. The functions of these components are to generate clocks, to multiplex interconnect signals and to interface with peripheral components. The handling of a multi FPGA system in the engineering process is challenging task. It requires comprehensive knowledge of system architecture, specification of FPGA device used and hardware definition languages. In this paper, we propose a 'plug and play' framework which aids in seamless stitching of partitioned module with FPGA infrastructure. An FPGA has limited resources for logic implementation and routing. These infrastructure components are overhead to implementation of the desired partitioned module. This paper discusses a novel infrastructure design which is scalable, robust and reduces turnaround time. Additionally, this paper also presents a scheme for pin aware placement TDM components that can address routing congestion.

Keywords— prototyping; multi-FPGA; multiplexing; Timing closure; pin placement; partitioning

I. Introduction

There are three verification methods in design cycle: Logic Simulation, Emulation and FPGA (Field Programmable Gate Array) prototyping. Logic simulation, although providing the best visibility and debugging capabilities, is extremely time consuming and often unreliable. It is best suited for small blocks. Emulation offers reduced run-time than logic simulation. It has quick turnaround times and good control and high visibility. The availability of emulator machine per user is limited since it is shared across several projects. The high cost limits number of emulator machines. Emulator is good option for critical code debug but expensive for software development. The multi-FPGA-based prototyping is cheaper in cost, but lacks in control and visibility [1]. It also has much longer turnaround times. A robust 'plug and play' FPGA infrastructure for prototyping can significantly reduce the turnaround time.

Every next SoC (System-on-Chips) is getting more complex than previous: multiple cores, big memories, huge IP blocks, as well as numerous gated clocks. The mapping of a complex SoC into prototyping platform requires weeks or sometimes several months. The SoC subsystems are developed incrementally in phase by geographically diverse teams. Also, there is a high expectation to get the "right-first-time" chip with no re-spins. Therefore, the objective of prototyping is to build an FPGA platform, containing a mature SoC design (ideally the tape-out version) which could be undertaken by software development teams. The software development begins much earlier than first silicon tape-out. The shrinking project time frames pose real challenge for prototyping. The only way to meet the two contrary requirements of design maturity and software development is to have an extremely fast FPGA mapping flow [2].

There are two choices for prototyping. Firstly, to build customized multimillion-gate prototype using off-the-shelf FPGAs. Secondly, to buy or rent predesigned prototyping systems. We are using an in-house, custom developed multi-FPGA platform for prototyping the large SoC or Cluster of IPs. This paper is issued from the experience of prototyping latest highly complex Intel SoCs.

II. Prototyping Process and Platform

A. *Steps of prototyping*

The sequence of steps is shown in fig. 1.

1) The first step of prototyping is partitioning of SoC design into multiple FPGAs [3]. Sometimes, a sub block of an SoC can be larger than an FPGA in which case it is split into smaller modules. Conversely, one FPGA can accommodate multiple SoC sub modules also. Finally, based on the logic resources available and number of interconnect signals, each FPGA gets one paritioned module.

2) The partitioning of design creates thousands of interconnect signals between FPGAs. The inter-FPGA signals are in order of hundreds and thousands whereas, the routes etched on the board between FPGAs are limited. Hence, interconnect signals are time division multiplexed (TDM) and then sent over FPGA route [4].

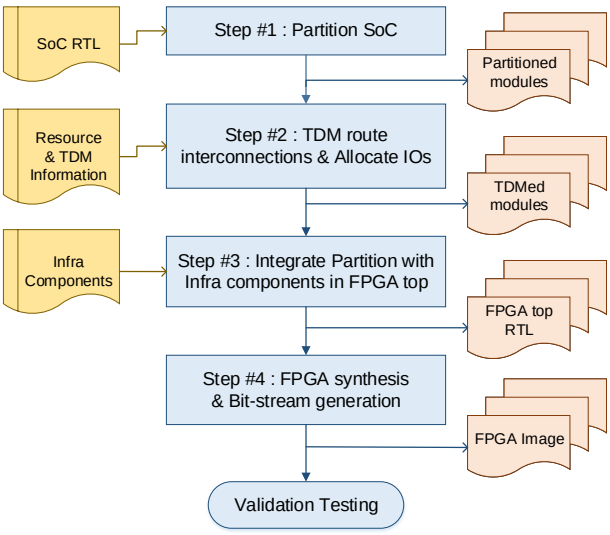


Fig. 1. Steps of prototyping

The number of interconnect signals that can be reliably multiplexed by one route is called Multiplex Ratio. The details about inter FPGA routes and their multiplexing ratios collated in a file called 'Time Division Multiplexing (TDM) information' file. Based on TDM information, each interconnect signal is allocated one time slot in one assigned route. Each route is associated with two Input-Output (IO) pins, one on sending and other on receiving FPGA [5]. The IO links used for TDM are both general purpose IO pins and High Speed Serial Interface (HSSI) channels. The output of second step is a vast database of TDMed modules with FPGA IOs assignement.

3) In third stept he TDMed module is plugged into a FPGA infrastructure and FPGA top RTL is generated. Henceforth, infrasturcture will be referred as infra. The infra provides clock, connection to TDM components and interface to other peripherals. This step is discussed further in sub section D of section II.

4) In the fourth step, the FPGA RTL top files are synthesized into bitstreams. The Static timing Analysis (STA) for each FPGA build is cleaned before the board testing. This step is a recursive and sometimes leads to re-partition the design.

B. *Multi Slot Platform*

The in-house prototyping system is a multi-FPGA, multi slot chassis as shown in figure 2. The chassis is built on an industry standard Advanced Telecommunications Computing Architecture. It has a custom backplane with fixed routes for signals and clock. A Peripheral Component Interconnect Express (PCIe) based host card provides the interface to user for configuration, control and status monitoring. A PCIe network provides connectivity to all FPGAs in the chassis.

One slot supports one blade/board. The words blade and board are used interchangeably. Stratix10 devices and several peripheral devices are placed on each blade. Peripheral module includes DDR4 memory, Serial Peripheral Interface (SPI) Flash, Serial gigabit media-independent interface (SGMII) links etc. The FPGAs are connected by traces etched on the board and over backplane. Additionally, there are

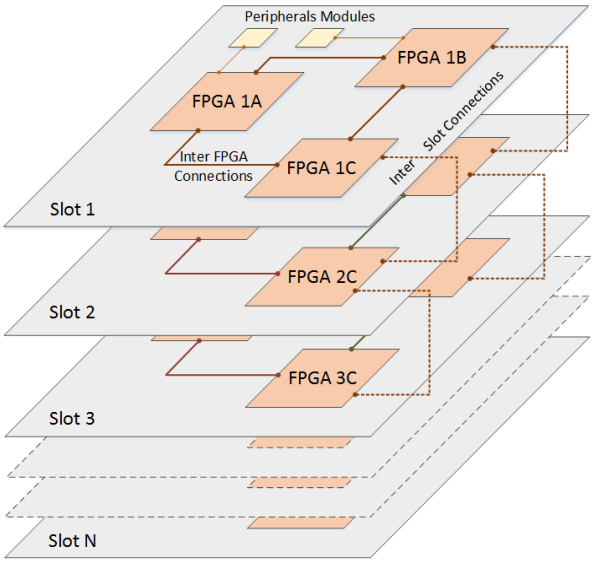


Fig. 2. Multi-slot chassis

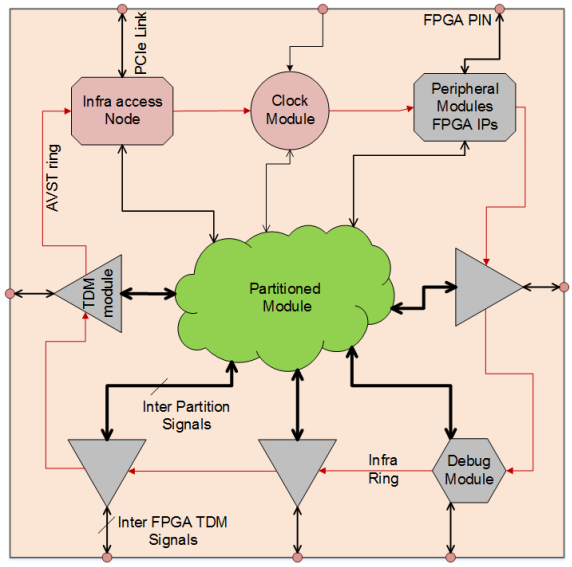


Fig. 3. Infra components of FPGA top RTL

flexible cables which are connected via mini-SAS. Thereby, making it highly connected multi-FPGA platform.

C. *SoC to FPGA translation*

While the SoCs are clocked by specialized clock circuitry which includes PLLs, together with a clocking network. A partitioned design spanning multiple FPGAs is clocked locally. From the original design, the clocking circuitry is removed and replaced with much slower frequency clocks in all the partitions. Upon power up, all these are synchronized before SoC reset is de-asserted. A clock synchronization scheme is employed to make sure that clocks in all the partitioned modules are edge aligned within acceptable skew limits. Thus, the partitioned modules virtually get the clock edge at the same time in all FPGAs.

If the SoC has Memory Interfaces, then the PHY layer is replaced by FPGA's DDR4 PHY. Similar is the case with Ethernet, SPI flash, UART, JTAG interfaces etc. Additionally, there are some custom designed debug components to capture the partitioned module's signals and present data to the user.

The interconnect signals between partitioned modules are time division multiplexed and transferred across FPGAs by TDM components. Based on the resource file of the chassis, a router algorithm allocates IOs for group of interconnect signal to be multiplexed. The process is automated and iterated for any change in partition modules. Every FPGA gets different number of TDM components with their configurations of number of interconnect signals, pin assignment, clock domain et cetera. The information is saved in large database as explained in section II.A.2.

D. *FPGA Top RTL Design Process*

The clocking scheme, various peripheral interfaces, TDM modules and debug components are part of infra [6]. As shown in figure 3, the infra envelops the partitioned module and completes a FPGA top RTL. These components and partitioned modules are assembled together by an automatic *method* developed by Industry standard packaging and integration tools. The method ensures generation of an RTL correctly connected and configured components. As shown is figure 4, all the three FPGAs have different infra. Clock module and infra access node is common to all. FPGA 1B is least congested. FPGA 1C is highly congested. The

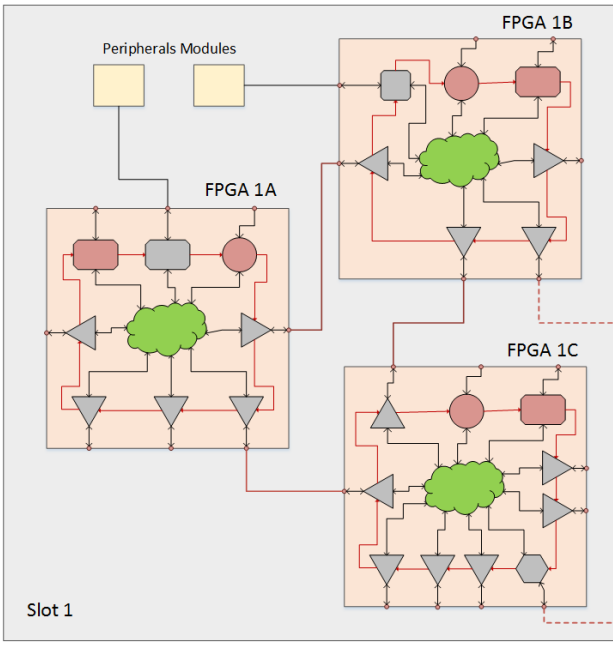


Fig. 4. Infra implementation in 3 FPGAs

incremental changes are rapidly and reliably integrated into the *method*.

III. NOVEL DESIGN METHODOLOGY

A. Infra Design

Infra design is a portable, configurable and highly flexible network of components. From partition to bring-up on the board is execute in a plug-and-play fashion.

During the testing on board, there is an Infra-health check done before reset to SoC partitions is lifted. First and foremost, it is checked whether FPGA PLLs are locked, and clocks are synchronized across FPGAs. In the TDM modules, there could be data corruption due to poor signal integrity on the board. Also, there could be misalignment in TDM data frame. Each infra component has a "Control and Status Register (CSR)" space which monitors and control its behavior during health check.

B. Infra-Ring

1) Structure and Protocol

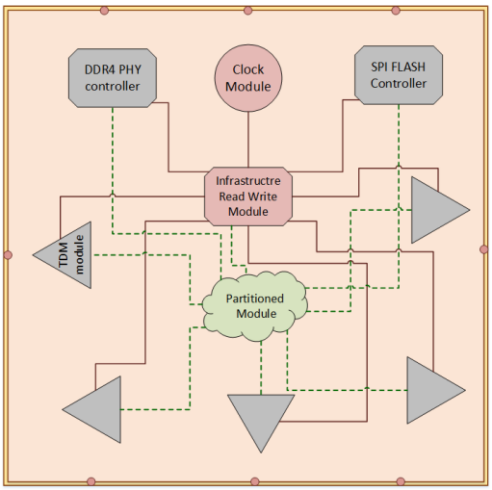


Fig. 5. Infra components connected in star fashion

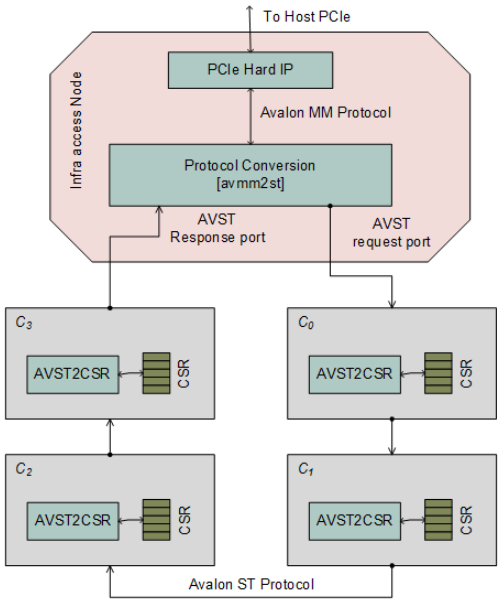


Fig. 6. Infra ring structure

The communication from the host PCIe card to each CSR space of infra component is performed via Infra access node. Within an FPGA, communication bus of infra component is connected to PCIe host via Infra access node. There are two topologies for this connection- Star and Ring. As shown in figure 5, star topology offers minimum latency at the cost of dedicated communication bus for each component. Ring topology offers communication pathway (bus) at the cost of high latency [7].

Typically, there are 200-300 infra components to connect. These components are connected in ring topology because it offers better routing implementation and high latency is acceptable for control access and status check. As shown in figure 6, Avalon Memory-mapped (AVMM) interface is converted to Avalon Streaming interface (AVST) in infra access node. AVST ports are connected to each component to form a ring [8].

2) Addition in FPGA top RTL

The infra components are assigned with a unique sequential *channel id number*, C_n based on the order in which they are logged in "TDMed module database". These modules are then connected in order of channel id number.

C. Pin Aware placement

1) STA closure

The place and route of large partitioned module with huge interconnections is an exhausting task for machine and repetitive for user if fitter tool fails. There is an increasingly serious timing closure problem when using high-performance, high-complexity FPGAs implemented at the 14 nm technology node [9]. A multiplex ratio of 2000 on a HSSI channel is difficult to route because of routing congestion as well as hard to close STA timing.

2) FPGA pin Lock

The IO pins in FPGA have fixed location marked by banks and tiles. Thus, the placement of TDM component is always close to IO pin. The AVST-ring bus gets routed in crisscross manner with overlapping routing, as shown in figure 7. If the AVST ring is routed in non-overlapping

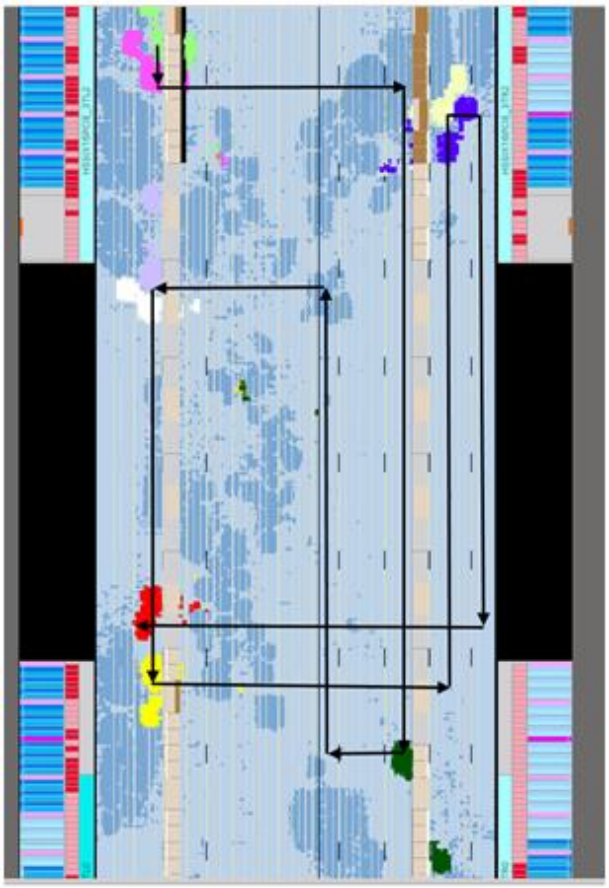


Fig. 7. Infra ring structure

manner it could free up the routing resource for partition interconnect signals.

3) Implementation

A non-overlapping AVST can be generated by manipulating the channel id number. First, spatial coordinates of the FPGA pin associated with each infra block is identified [10]. A *directed graph* from infra access node traversing each "spatial coordinate" is generated such that there is no overlap with another edge. The channel id numbers of the components are rearranged according to the respective node number derived from the *directed graph.*

IV. Experiments and Results

A. Results

The following results shown here are based on the mapping of an Intel SoC. Table 1 shows a snippet of heavy resource utilization in one Stratix10 FPGA.

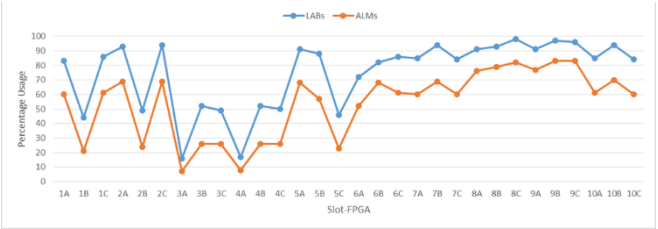
From the Table 1, it is evident that it is possible to TDM signals on ~50% IOs in a FPGA which has 99% logic utilization. The results clearly depict that a well-defined infra eases the placement and fitting of a resource extensive design. The number of IO pins and HSSI depends on the TDM modules instantiated. The LABs and ALMs are majorly consumed by partitioned module. As shown in Graph 1, most of the FPGAs are more than 90% utilized. Graph 2 shows the ability of the infra to TDM interconnect signals even at high logic utilization.

TABLE I. Fitter Resource Usage Summary

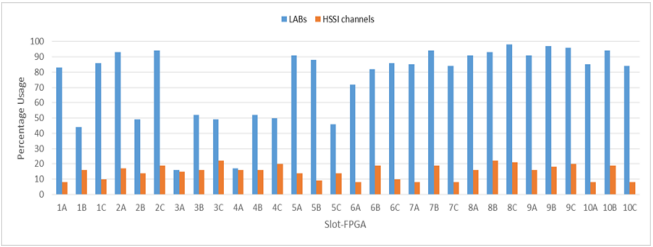
Resource	***Usage***	***%***
Total LABs: partially or completely used	92,207 / 93,312	99
Logic utilization (ALMs needed / total ALMs on device)	756,656 / 933,120	81
Total I/O pins	591 / 1,152	51
Total HSSI channels	19 / 96	20

V. Conclusion and Future Work

In this paper, we have discussed prototyping steps for an in-house prototyping platform. In the prototyping world, module partitioning, signal TDM and signal routing have been extensively researched. We have focused on the latter stage of FPGA top RTL generation. A flexible and robust infrastructure speeds up stitching of partitioned module with its components. This integration is done in a "plug and play" approach. This approach is employed in prototyping multiple SoC and yielded consistent results.



Graph 1. LABs and corresponding ALMs utilization



Graph 2. Comparison of LABs utilization vs IO pins used

We are motivated to improve the system's set of features. The foremost is the Graphical User Interface for analyzing the results. This enhancement will save lots of time in the development cycle.

VI. Acknowledgment

This work is developed by FPGA Prototyping team in Intel Corporation. We are very thankful to the whole technical team for their support and valuable suggestions.

VII. References

[1] Brian Bailey, "Do we need a new FPGA structure for prototyping?", Designlines, Automotive Designline, EETimes, 2011 [Online]

[2] H. Krupnova, "Mapping multi-million gate SoCs on FPGAs: industrial methodology and experience", Proceedings Design, Automation and Test in Europe Conference and Exhibition, 2004

[3] A Doug, A Lesea, R Richter, "FPGA-based prototyping methodology manual", Synopsys San Jose, 2011

[4] Zhaoxiang Zong, "Pin Multiplexing Optimization in FPGA Prototyping System", 4th International Conference on Systems and Informatics (ICSAI), 2017

[5] Mariem Turki, Zied Marrakchi, Habib Mehrez, Mohamed Abid, “Signal multiplexing approach to improve inter-FPGA bandwidth of prototyping platform”, Springer Science+Business Media New York, 2015

[6] C. Bieser and K.D. Mueller-Glaser. “Rapid Prototyping Design Acceleration Using a Novel Merging Methodology for Partial Configuration Streams of Xilinx Virtex-II FPGAs”, In 17th IEEE Int. Workshop on Rapid System Prototyping, pages 193-199, Los Alamitos, CA, USA, 2006.

[7] M. Saldana, L. Shannon, and P. Chow, “The routability of multiprocessor network topologies in fpgas,” in Proceedings of the 2006 international workshop on System-level interconnect prediction. ACM, 2006,

[8] Intel corporation, Avalon Interface Specifications, Available: https://www.intel.com.

[9] Angela Sutton and Jeff Garrison, “How to achieve timing-closure in high-end FPGAs”, Designlines, Programmable Logic, EETimes, [Online]

[10] Pritha Banerjee, “FAST I/O PAD PLACEMENT IN FPGAs”, Indian statistical institute, https://www.isical.ac.in/