Microprocessors and Microsystems 56 (2018) 169–183

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Novel architectural space exploration environment for multi-FPGA based prototyping systems

Umer Farooq[*,a], Roselyne Chotin-Avot[b], Moazam Azeem[b], Maminionja Ravoson[b], Habib Mehrez[b]

[a] Electrical and Computer Engineering Department, Dhofar University, Salalah, Oman
[b] Sorbonne Universites, UPMC Univ, Paris 06, CNRS, LIP6, Paris, France

ARTICLE INFO

Keywords:
Multi FPGA-based prototyping
Exploration environment
Inter-FPGA routing
Debugging

ABSTRACT

Prototyping of complex digital systems using multi-FPGA platforms offers several key advantages over other prototyping techniques. These advantages include higher execution speed, lower cost, and real world testing experience. The quality of a prototyped design, however, is adversely affected by nonexistence of multi-FPGA exploration environments. This work presents a novel, generalized exploration environment for multi-FPGA platforms that gives end-to-end exploration experience. For experimentation purpose, ten large benchmarks are generated, synthesized, and partitioned using a combination of locally developed and commercial tools. FPGA board exploration is then performed through locally developed timing-driven inter-FPGA routing tool where five FPGA boards are used and for each board, four different inter-FPGA track combinations are explored. For experimentation, number of FPGAs on board are varied from two to six and impact of this variation is observed on the frequency of prototyped design. Experimental results show that FPGA boards with inter-FPGA tracks corresponding closely to cut net requirement of partitioned benchmarks give, on average, best frequency results. Moreover, FPGA boards having higher number of FPGAs give, on average, better frequency results as compared to boards having smaller number of FPGAs. Furthermore, a comparison between timing-driven and routability-driven inter-FPGA routing approaches shows that former approach requires, on average, 46% less execution time than the later while giving same frequency results. Finally, validation of proposed environment is also performed through in-circuit verification of sample benchmarks on a stack of FPGA boards.

1. Introduction

Modern day System-on-Chip (SoC) designs have enormous computation capability. Today, a medium sized SoC houses multi-million logic gates and it has the computation capability that dwarfs even the super computers of few years back [1]. This advancement in computation capability and performance, however, has come at the cost of complex and expensive design process of new digital systems. Ever decreasing product life cycle and faster time-to-market constraints further increase the design pressure and a fast, efficient process is required to ensure smooth design-to-silicon transition. Today, the design process of an Application Specific Integrated Circuit (ASIC) takes about two-three years to role out first prototype while requiring hundreds of thousands of dollars in investment [2,3]. Continuous miniaturization of processing technology metrics further leaves an adverse effect on an already weak reliability index of final rolled out design. This makes pre-silicon verification an important step in the design process of a digital system. As stated in [2], design verification step takes around 70% of total time and around 80% of total design cost. The reason for such a huge effort on verification is that it can eventually save the company from loss both in terms of money and reputation [4,5].

Several techniques are used for pre-silicon functional verification of digital systems among which simulation, emulation, and prototyping based verification methods are the most popular [6]. Each of these techniques has its pros and cons. For example, simulation based verification offers complete visibility of the design with quick set-up time and low price. Different market leaders offer simulation based platforms like Cadence Incisive [7], Mentor Graphics Modelsim [8], Synopsys VCS [9]. The execution speed of simulators is, however, limited (~ 1kHz) and run time of complex designs can span several weeks. Emulators, on the other hand, are faster as compared to simulators and offer execution speed in MHz range (1 ~ 2 MHz). They also offer profound visibility into the Register Transfer Level (RTL) description of the design with large logic capacity and complete debugging ability. Three emulator platforms offered by Electronic Design Automation (EDA) vendors are Cadence Palladium Emulator [10], Synopsys EVE Zebu Emulator [11], and Mentor Graphics Veloce Emulation systems [12]. Although emulators offer significant advantages over simulation, their

* Corresponding author.
E-mail address: ufarooq@du.edu.om (U. Farooq).

https://doi.org/10.1016/j.micpro.2017.12.006
Received 4 January 2017; Received in revised form 5 October 2017; Accepted 11 December 2017
Available online 12 December 2017

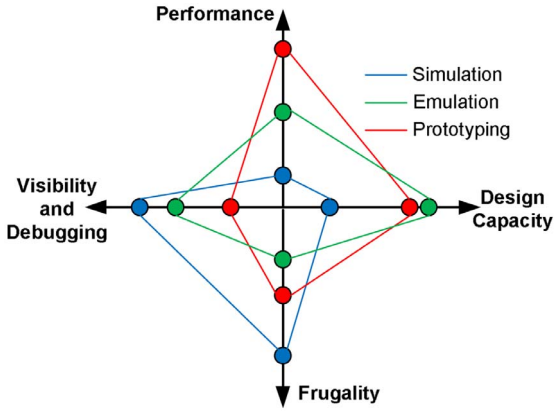

Fig. 1. Comparison between different verification techniques.

high price tags and prolonged initial set-up time make them unattractive for systems where price and time-to-market are principal constraints. Finally, we have Field Programmable Gate Array (FPGA) based prototyping platforms which execute the design at cycle-accurate, bit-accurate level and run it at significantly higher speed (~ 15 MHz). Multi FPGA-based platforms offer almost similar design capacity as emulators while offering better portability. Unique to FPGA-based prototyping is its real world testing experience with actual external interfaces. Although FPGA-based prototyping platforms are inexpensive as compared to emulators, they offer poor system visibility with comparable set-up time. A pictorial comparison between three verification techniques is shown in Fig. 1. It can be seen from this figure that advantages like frugality, high performance, and real world testing experience make FPGA-based prototyping most favored among three verification techniques.

Present-day ASIC designs are quite large and logic requirement of even a moderately complex ASIC design is multi-million logic gates. Although contemporary FPGAs are efficient in terms of speed, their capacity is outstripped by the requirements of ASIC design due to their generalized and configurable nature [13]. Today, FPGAs are estimated to be many times larger than their ASIC counterparts and for complex ASIC designs, this area gap calls for multi-FPGA based prototyping platforms. Prototyping of an ASIC design using multiple FPGAs is a challenging task as it follows a complex back-end flow where a single design is partitioned into multiple parts which are then implemented on different FPGAs. These FPGAs are normally mounted on a single or multiple boards and they are connected to each other through two-point or multi-point physical tracks. The number of FPGAs in a multi-FPGA prototyping platform depends upon the complexity of SoC/ASIC design to be prototyped and it may vary from a few FPGAs [14] on a single board to several dozen FPGAs on different FPGA boards [15]. When we consider back-end flow of multi-FPGA prototyping, the quality of final prototyped design is severely affected by the caliber of tools used. Back-end flow normally starts with the RTL description of a design under test. The design is first synthesized and then it is partitioned. In multi-FPGA prototyping, partitioning is a step that divides the large design under consideration into multiple parts. The number of parts is equal to the number of FPGAs on target multi-FPGA board and size of each part is kept in such a way that it does not exceed the logic capacity of target FPGA architecture. Principle optimization objective of any partitioner is to divide the design in such a way that resultant partitions have minimum possible inter-partition (i.e. inter-FPGA) communication. Signals connected to different FPGAs on board are termed as cut-nets of the design and their count is inversely related to the performance of prototyped design.

Inter-FPGA routing is a step that follows partitioning of the design under consideration. In this step, cut-nets of the partitioned design are routed on the tracks of FPGA board in a Time Division Multiplexed (TDM) manner [16]. In this technique, many cut-nets are combined through a multiplexer which is connected to one of the I/Os of source FPGA. The I/O of source FPGA is connected to an inter-FPGA routing track which routes these cut-nets to the respective I/O of destination FPGA where these cut-nets are demultiplexed. Multiplexer and demultiplexer use a fast clock to send the cut-nets over inter-FPGA tracks. Normally, all the multiplexed data should be sent in one system clock cycle whose length is a multiple of fast clock cycle. Maximum number of cut-nets passing through a multiplexer is termed as multiplexing ratio and it has an inverse relation with the execution speed of prototyped design. While performing inter-FPGA routing, cut-nets are either routed directly (point-point) from source to destination FPGA or an intermediate FPGA might be used as a hop if direct path between source and destination FPGA does not exist. Addition of hops in the routing path further deteriorates the execution speed of prototyped design. So, the objective of inter-FPGA routing process is to optimize the multiplexing ratio while keeping number of hops to a minimum. This is a challenging problem because the number of I/Os in newer generations of FPGAs has not increased at the same rate as their logic capacity and this trend has resulted in the scarcity of I/O resources of FPGAs. This trend can be understood with the help of Table 1 where I/Os and logic gates per I/O metrics for Altera's Stratix and Xilinx's Virtex family are given. It can be seen from this table that newer generations of FPGAs in both families have same or fewer I/Os while their logic capacity is increased enormously. For example, almost 3000 logic gates are trying to pass through a single I/O in case of Xilinx Virtex-4 and this number has increased to almost 20,000 in case of Virtex-7. Similar trend holds for Altera's Stratix family as well. This trend shows that FPGA logic to I/O ratio is worsening generation after generation and it calls for high quality inter-FPGA routing tool that can improve the execution speed of final prototyped design. Once inter-FPGA routing is done, partitions of the design are placed and routed on the target FPGA and prototyping process culminates with in-circuit verification of the design. Further details on the multi-FPGA prototyping flow are given in the following sections of the paper.

It is evident from the discussion presented above that prototyping of SoC/ASIC designs using multi-FPGA platforms requires expertise both at hardware and software level. From hardware perspective, there are companies which sell hardware platforms only like Dinigroup [17]. There are also some solutions which cover parts of back-end flow for multi-FPGA prototyping like Synopsys Protocompiler [18] that accompanies its HAPS platform [19], Auspy and Wasga partitioning tools by Mentor Graphics [20,21]. However, the problem with aforementioned solutions is that either they provide only partial solutions [20,21] or they are too constrained [18] and can only be used for a specific hardware platform. In this work, we propose a novel back-end flow for architecture space exploration of digital systems based on multi-FPGA prototyping. The proposed flow is generic in nature and starting from benchmark generation to in-circuit verification, it gives end-to-end user experience. The flow starts with large, complex benchmark generation through locally developed benchmark generator. The benchmarks are then synthesized and partitioned using generic commercial tools. Later, inter-FPGA routing is performed on

Table 1
FPGA logic capacity to I/O ratio for different FPGAs.

FPGA Name	No of I/Os	Gates per I/O
Virtex 4	960	3000
Virtex 5	1200	2900
Virtex 6	1200	8000
Virtex 7	1200	20000
Stratix 2	1170	2000
Stratix 3	1120	4000
Stratix 4	1120	9500
Stratix 5	840	16000