Chapter 1
Introduction

1.1 Background

Field Programmable Gate Arrays (FPGAs) are pre-fabricated silicon devices that can be electrically programmed to become almost any kind of digital circuit or system. First modern era FPGA was introduced almost two and a half decades ago. That FPGA contained very small number of logic blocks and I/Os. Since then, FPGAs have witnessed an enormous expansion both in terms of capacity and market. They have come a long way from the devices that were once considered only as glue logic to the devices that can now implement complete applications. FPGAs are now widely used for implementing digital circuits in a wide variety of markets including telecommunications, automotive systems and consumer electronics.

FPGAs consist of an array of blocks of potentially different types, including general purpose logic blocks and specific purpose hard blocks like memory and multiplier blocks. Among these blocks, general purpose logic blocks are programmable and along with specific purpose hard blocks they are surrounded by a programmable routing fabric that allows these blocks to be programmably interconnected. The array of blocks along with routing fabric is surrounded by programmable input/output blocks that connect the chip to the outside world. The "programmable" term in FPGA indicates that virtually any hardware function can be programmed into it after its fabrication. This customization is realized with the help of programming technology, which is a method that changes the behavior of the chip in the field after its fabrication.

FPGAs rely on an underlying programming technology that is used to control the programmable switches that give FPGAs their programmability. There are a number of programming technologies and their differences have a significant impact on programmable logic architecture. Earlier programmable logic devices used very small fuses as their programming technology [55]. However, later this programmable technology was replaced by now widely used static memory based programming technology. Most commercial vendors [76, 126] use Static Random Access Memory (SRAM) based programming technology because of its easy re-programmability and

U. Farooq et al., *Tree-Based Heterogeneous FPGA Architectures*,
DOI: 10.1007/978-1-4614-3594-5_1,

the use of standard CMOS process technology. Although, some other programming technologies like flash [2] and anti-fuse [23] are both smaller in area and are non-volatile, the use of standard CMOS manufacturing process makes the SRAM-based programming technology dominating. As a result SRAM-based FPGAs can use the latest CMOS technology and therefore benefit from increased integration, higher speed and lower dynamic power consumption of new process with smaller geometry. Primarily, in an FPGA, SRAM cells are used to program multiplexors that steer the interconnect of FPGAs. Further, they are also used to store the data in general purpose logic blocks also termed as Configurable Logic Blocks (CLBs) that are typically used in SRAM based FPGAs to implement logic functions.

The flexibility and reprogrammability of FPGAs leads to lower Non-Recurring Engineering (NRE) cost and faster time to market than more customized approaches such as Application Specific Integrated Circuit (ASIC) design. The pre-fabricated and programmable nature of FPGAs provides digital circuit designers access to the benefits of latest process technology. In case of custom design, however, significant time and money must be spent on ever-increasing complex issues associated with design and fabrication using latest custom VLSI process technology. On contrary, FPGA-based design cycle time and NRE cost is much lower than full-custom or standard-cell based ASIC layouts.

FPGAs pay for these advantages, however, with some significant disadvantages. Compared with the non-programmable devices, FPGAs have higher area, lower performance and higher power consumption. The large area gap affects the costs, and also limits the size of the designs that can be implemented on FPGAs. The loss in performance also drives up costs as more parallelism and hence greater area may be needed to achieve a performance target. Also it simply may not be possible to achieve the desired performance on an FPGA. Similarly, higher power consumption often limits FPGAs from markets requiring high efficiency in terms of power consumption. Together, this area, performance and power gap limits the applicability of FPGAs when area, speed and/or power requirements of an application are not met. Authors in [60] have reported that FPGAs are 20–35 times larger, 3–4 times slower and 7–14 times more power consuming than ASICs. As a result of this large area, performance and power gap between FPGAs and ASICs, FPGAs become unsuitable for some applications. To address this limitation, a range of alternatives to FPGAs exist.

The primary alternative to an FPGA is an ASIC that has speed, power and area advantages over an FPGA. However, compared to FPGAs, ASICs have certain disadvantages in the form of higher non-recurring engineering (NRE) cost, longer manufacturing time and increasingly complicated design process. While an ASIC implementation offers area, performance and power gains, the difficulties associated to their design process have led to the development of devices that lie in between FPGAs and ASICs. These devices are termed as Structured-ASICs. Structured-ASICs can cut the NRE cost of ASICs by more than 90% while speeding up significantly their time to market [125]. Structured-ASICs contain array of optimized elements which implement a desired functionality by making changes to few upper mask layers. The density and performance of a Structured-ASIC is directly related to the number of

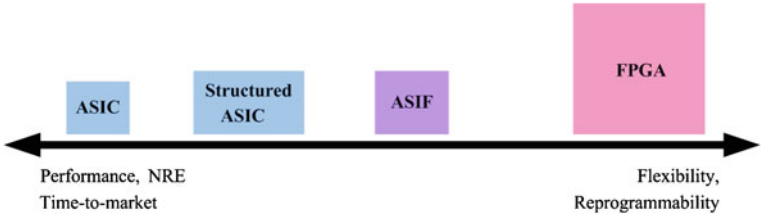

Fig. 1.1 Comparison of different platforms used for implementing digital applications

mask layers that are available for customization. Structured-ASICs are explored or manufactured by several companies [41, 91, 103, 125]. Although Structured-ASICs give lower NRE cost as compared to the standard cell ASICs, their overall efficiency is not as good as that of ASICs and additional cost of an ASIC implementation is not always prohibitive.

FPGA vendors have also started giving provision to migrate FPGA based application to Structured-ASIC. In this regard, Altera has proposed a clean migration methodology [98] that ensures equivalence verification between FPGA and its Structured-ASIC (known as HardCopy [56]). However, migration of an FPGA based application to HardCopy can execute only a single circuit. An Application Specific Inflexible FPGA (ASIF) [93], on the other hand, comprises of optimized logic and routing resources like Structured-ASIC but retains enough flexibility to implement a set of pre-determined applications that operate at mutually exclusive times. Contrary to Structured-ASIC which is basically a modified form of ASIC and which is capable of implementing only one application, an ASIF is a modified form of an FPGA and it can implement a set of application for whom it is designed. However, unlike FPGAs that are generalized in nature, an ASIF contains more customized logic and routing resources and it has only enough flexibility that is required to implement a pre-determined set of applications. Figure 1.1 presents a rough comparison of different platforms that can be used for implementing digital applications.

1.2 Book Motivation and Contributions

In general, the overall efficiency of an FPGA is in inverse relation with its flexibility i.e. improvement in one aspect causes a deterioration in the other and vice versa. The main theme of this work is to remedy the drawbacks that are associated with FPGAs with/without compromising their advantages. For this purpose, we explore and optimize a relatively new and unexplored tree-based (hierarchical) architecture along with an established and well investigated mesh-based (island-style) architecture. Although the two architectures comprise of similar logic and routing resources, it is the arrangement of these resources that converts them into altogether different architectures exhibiting different area, performance and power results. In a

tree-based architecture, logic resources are arranged in clusters and these clusters are connected to each other recursively to form a hierarchical structure. On the other hand, logic resources in a mesh-based architecture are arranged in an island-style and these resources are connected to each other using uniform routing resources that surround them. In order to explore the two architectures, a new exploration environment for tree-based architecture and an optimized, enhanced environment for mesh-based architecture are used. The two environments are based on a mixture of generalized and specifically developed tools for mapping different applications on the two architectures.

While exploring and optimizing two architectures, our main emphasis is on the area optimization; performance and power optimization are not performed in this work. Area improvement generally implies smaller architectures which result in an improvement both in performance and power consumption. In order to improve the area of the two FPGA architectures, following two broad techniques are employed:

1. Improve the utilization of logic resources of the architecture.
2. Improve the utilization of routing resources of the architecture.

Classic FPGA architectures used only a single type of block that provided the basic logic capability for the implementation of almost any kind of application. Although the use of logic blocks makes FPGAs a good alternative for almost any kind of application, it requires a large amount of logic and routing resources. Now a days, a lot of DSP and arithmetic intensive applications use memories, adders and multiply operations. When these applications are mapped on FPGAs, considerable amount of logic and routing resources can be saved in FPGAs by mapping such operations directly on the specific hard-blocks that are embedded in the architecture along with logic blocks. By embedding hard-blocks directly into the architecture, the overall size of the architecture can be reduced which eventually results in improved area and performance results. The types and quantities of hard-blocks in an FPGA can be decided from the application domain for which an FPGA is required. In this work, we embed hard-blocks in the two architectures under consideration to reduce the overall architecture size and hence improve the utilization of logic resources of the architecture.

The area of an FPGA can be further decreased by optimizing the routing network of an FPGA for a given set of application circuits. By optimization, here, we mean that routing network has a reduced flexibility and it can implement only a pre-determined set of applications. Such a reduced FPGA is called here as an Application Specific Inflexible FPGA (ASIF). An ASIF can be either used in the same scenario as that of Structured-ASIC where a product is initially designed and tested on an FPGA and later it is migrated to an ASIF for high volume production. However, the second and major application of ASIF can be a product that performs different tasks at different times. Such a product may comprise of a video application, a multi-standard radio application, or any set of DSP functionalities required at different times. For example, in the case of a camera different encoders and decoders are required for video and image processing. Further, various compression techniques can be used both for images (e.g. JPEG and PNG etc) and video recording (e.g. MPEG-4 and

H.264). So different digital circuits can be designed and tested on an FPGA and later, for high volume production, the FPGA can be reduced to an ASIF for the given application circuits. So in this work we improve the area of the FPGA architectures by first efficiently incorporating hard-blocks in them and then optimizing their routing networks for a particular set of applications.

The major contributions of this book are as follows:

1.2.1 Exploration Environment for Heterogeneous Tree-Based FPGA Architectures

This work presents a new exploration environment for tree-based heterogeneous FPGA architecture. This environment is generalized and flexible in nature and can be used to explore different architectural topologies with a varying range of logic blocks and hard-blocks. Further, this work also presents an exploration environment for mesh-based heterogeneous FPGA architectures. The environments of two architectures are used to explore and evaluate a number of techniques for both architectures.

The exploration and evaluation of two architectures start with respective architecture definition where separate architecture description mechanisms are used to select different architecture parameters for the two architectures under consideration. Once the architectures are defined, separate software CAD flows are then used to map application circuits on the two architectures. Each software flow uses appropriate techniques to optimize respective architecture. Although, the main objective of the book is not to establish the supremacy of one architecture over the other, however, a detailed comparison between mesh-based and tree-based architectures is presented using 21 heterogeneous benchmarks. Comparison results reveal that tree-based heterogeneous FPGA architecture gives better overall results than mesh-based heterogeneous FPGA architecture.

1.2.2 Exploration of Tree-Based ASIF Architecture

An Application Specific Inflexible FPGA (ASIF) is a modified form of FPGA with reduced flexibility that can implement a set of application circuits which will operate at mutually exclusive times. These circuits are efficiently placed and routed on an FPGA to minimize total routing switches required by the architecture. Existing placement and routing algorithms are modified to efficiently place and route circuits on the architecture. Later, all unused routing switches are removed from the FPGA to generate an ASIF.

In this work a new tree-based homogeneous ASIF is presented. Exploration of tree-based ASIF is performed using a set of 16 benchmarks and experimental results

have shown that tree-based ASIF is significantly smaller than an equivalent tree-based FPGA which is required to map any of these circuits. Further a comparison between mesh-based and tree-based ASIFs shows that tree-based ASIF gives better area results when compared to an equivalent mesh-based ASIF. The concept of ASIF is also extended to heterogeneous architectures where a comparison between tree-based heterogeneous ASIF with an equivalent tree-based heterogeneous FPGA is presented. Further, the comparison between mesh-based and tree-based ASIFs is also presented. The VHDL models of homogeneous and heterogeneous ASIFs are also generated using specifically developed VHDL model generator. Layout of the VHDL model is later performed using Cadence Encounter with 130 nm 6-metal layer CMOS process of ST Microelectronics.

1.3 Book Organization

The organization of this manuscript is as follows. Chapter 2 gives a detailed overview about the basic FPGA architecture and their associated design flow. Later in the chapter some current trends in the reconfigurable computing and in the FPGAs are presented also. Chapter 3 presents a detailed overview of the basic exploration environments of homogeneous mesh-based and tree-based architectures that are used in this work. This chapter also presents some new comparison results of the two architectures. Chapter 4 presents new exploration environment of tree-based heterogeneous FPGA architecture. An exploration environment for mesh-based heterogeneous FPGA architecture is also presented in this chapter. The two architectures are explored using 21 benchmarks which, based on their communication trends, are further divided into three distinct sets. Different techniques are explored using the exploration environments of two architectures and results obtained through experimentation are used for comparison between two architectures. Chapter 5 presents a new tree-based ASIF where four ASIF generation techniques are explored for a set of 16 MCNC [108] benchmarks and a comparison between tree-based ASIF and an equivalent tree-based FPGA is also presented. Later, for tree-based architecture, the effect of lookup table and arity size is explored for the most efficient technique among the four explored techniques. Further a detailed comparison between mesh-based and tree-based ASIFs is performed and finally a quality analysis of tree-based ASIF and a quality comparison between mesh-based and tree-based ASIFs is performed. Chapter 6 presents the extension of tree-based homogeneous ASIF to heterogeneous domain. Four ASIF generation techniques are explored for tree-based heterogeneous ASIF using 17 benchmarks and a comparison between tree-based ASIF and equivalent tree-based FPGA is also presented. Later experiments are performed to determine the effect of LUT and arity size on tree-based heterogeneous ASIF. After that, a comparison between mesh-based and tree-based ASIFs is performed and then a quality analysis of tree-based heterogeneous ASIF and a quality comparison between heterogeneous mesh-based and tree-based ASIFs is performed. Chapter 7 concludes this work and presents some future work.