Foundations and Trends® in
Electronic Design Automation
2:2

FPGA Architecture: Survey and Challenges

Ian Kuon, Russell Tessier, and Jonathan Rose

now
the essence of knowledge

FPGA Architecture: Survey and Challenges

FPGA Architecture: Survey and Challenges

Ian Kuon

University of Toronto
Toronto, ON
Canada
ikuon@eecg.utoronto.ca

Russell Tessier

University of Massachusetts
Amherst, MA
USA
tessier@ecs.umass.edu

Jonathan Rose

University of Toronto
Toronto, ON
Canada
jayar@eecg.utoronto.ca

Boston – Delft

Foundations and Trends® in Electronic Design Automation

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
USA
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is I. Kuon, R. Tessier and J. Rose, FPGA Architecture: Survey and Challenges, Foundations and Trends® in Electronic Design Automation, vol 2, no 2, pp 135–253, 2007

ISBN: 978-1-60198-126-4

Foundations and Trends® in Electronic Design Automation
Volume 2 Issue 2, 2007

Editorial Board

Editorial Scope

Foundations and Trends® in Electronic Design Automation will publish survey and tutorial articles in the following topics:

- System Level Design
- Behavioral Synthesis
- Logic Design
- Verification
- Test
- Physical Design
- Circuit Level Design
- Reconfigurable Systems
- Analog Design

Information for Librarians
Foundations and Trends® in Electronic Design Automation, 2007, Volume 2, 4 issues. ISSN paper version 1551-3939. ISSN online version 1551-3947. Also available as a combined paper and online subscription.

Foundations and Trends® in
Electronic Design Automation
Vol. 2, No. 2 (2007) 135–253

DOI: 10.1561/1000000005

now
the essence of knowledge

FPGA Architecture: Survey and Challenges

Ian Kuon[1], Russell Tessier[2] and Jonathan Rose[1]

[1] *The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, {ikuon, jayar}@eecg.utoronto.ca*
[2] *Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, USA, tessier@ecs.umass.edu*

Abstract

Field-Programmable Gate Arrays (FPGAs) have become one of the key digital circuit implementation media over the last decade. A crucial part of their creation lies in their architecture, which governs the nature of their programmable logic functionality and their programmable interconnect. FPGA architecture has a dramatic effect on the quality of the final device's speed performance, area efficiency, and power consumption. This survey reviews the historical development of programmable logic devices, the fundamental programming technologies that the programmability is built on, and then describes the basic understandings gleaned from research on architectures. We include a survey of the key elements of modern commercial FPGA architecture, and look toward future trends in the field.

Contents

1

Introduction

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. They provide a number of compelling advantages over fixed-function Application Specific Integrated Circuit (ASIC) technologies such as standard cells [62]: ASICs typically take months to fabricate and cost hundreds of thousands to millions of dollars to obtain the first device; FPGAs are configured in less than a second (and can often be reconfigured if a mistake is made) and cost anywhere from a few dollars to a few thousand dollars.

The flexible nature of an FPGA comes at a significant cost in area, delay, and power consumption: an FPGA requires approximately 20 to 35 times more area than a standard cell ASIC, has a speed performance roughly 3 to 4 times slower than an ASIC and consumes roughly 10 times as much dynamic power [120]. These disadvantages arise largely from an FPGA's programmable routing fabric which trades area, speed, and power in return for "instant" fabrication.

Despite these disadvantages, FPGAs present a compelling alternative for digital system implementation based on their fast-turnaround and low volume cost. For small enterprises or small entities within large

corporations, FPGAs provide the only economical access to the scalability and performance provided by Moore's law. As Moore's law progresses, the ensuing difficulties brought about by state-of-the-art deep submicron processes make ASIC design more difficult and expensive. The investment required to produce a useful ASIC consists of several very large items in terms of time and money:

(1) State-of-the-art ASIC CAD tools for synthesis, placement, routing, extraction, simulation, timing analysis, and power analysis are extremely costly.
(2) The mask costs of a fully-fabricated device can be millions of dollars. This cost can be reduced if prototyping costs are shared among different, smaller ASICs, or if a "structured ASIC" approach, which requires fewer masks, is used.
(3) The loaded cost of an engineering team required to develop a large ASIC over multiple years is huge. (This cost would be related, but smaller for an FPGA design team.)

These high costs, and the need for a proportionally higher return on investment, drive most digital design starts toward FPGA implementation.

The two essential technologies which distinguish FPGAs are architecture and the computer-aided design (CAD) tools that a user must employ to create FPGA designs. The goal of this survey is to examine the existing state of the art in FPGA architecture and to project future trends; a companion paper on CAD for FPGAs appeared in a previous edition of this journal [54].

The survey is organized as follows: we first give a brief overview of programmable logic to provide a context for the subsequent sections which review the history of programmable logic, and the underlying programming technologies. The following sections define the terminology of FPGA architecture, and then describe the foundations and trends in logic block architecture and routing architecture including a discussion of power management techniques and related circuit design issues. A brief overview of the input/output structures and architectural questions is then presented followed by an explicit comparison between FPGAs and competing ASIC standard cell technology. Finally,

the survey concludes with a review of some of the design challenges facing FPGAs and a look at emerging architectures for FPGAs.

1.1 Overview

FPGAs, as illustrated in Figure 1.1, consist of an array of programmable logic blocks of potentially different types, including general logic, memory and multiplier blocks, surrounded by a programmable routing fabric that allows blocks to be programmably interconnected. The array is surrounded by programmable input/output blocks, labeled I/O in the figure, that connect the chip to the outside world.

The "programmable" term in FPGA indicates an ability to program a function into the chip after silicon fabrication is complete. This customization is made possible by the programming technology, which is a method that can cause a change in the behavior of the pre-fabricated chip after fabrication, in the "field," where system users create designs.

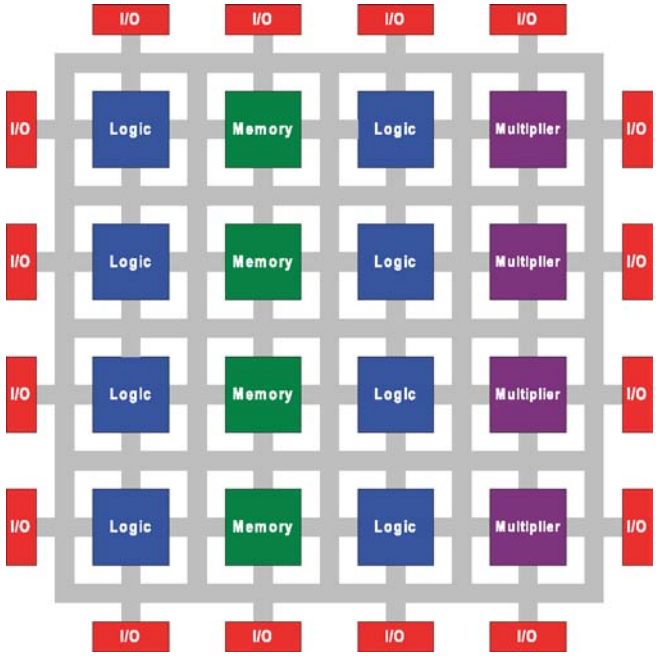

Fig. 1.1 Basic FPGA structure.

The first programmable logic devices used very small fuses as the programming technology. These devices are described briefly in the following section on the history of programmable logic. Section 3 goes into more detail on the three principal programming technologies in use today in modern FPGAs.

2

Early History of Programmable Logic

The origins of the contemporary Field-Programmable Gate Array are tied to the development of the integrated circuit in the early 1960s. Early programmable devices employed architectural regularity and functional flexibility. Cellular arrays [150] typically consisted of a two-dimensional array of simple logic cells with fixed, point-to-point communication. These first arrays, such as the Maitra cascade [143], contained logic cells which could be programmed via metalization during manufacturing to implement a range of two-input logic functions. By the mid-1960s, field-programmability, the ability to change the logic function of a chip after the fabrication process, was achieved via the introduction of "cutpoint" cellular arrays [150]. Although the connections between the elements of the array were fixed, the functionality of each logic cell in the array could be determined by setting programmable fuses. These fuses could be programmed in the field through the use of programming currents or photo-conductive exposure [150]. As a result, field-customization allowed for simplified array manufacturing and wider applicability.

In the 1970s, a series of read-only memory (ROM)-based programmable devices were introduced and provided a new way to

implement logic functions. Although mask-programmable ROMs and fuse-programmable ROMs (PROMs) with N address inputs can implement any N-input logic function, area efficiency quickly becomes an issue for all but small values of N due to the exponential dependence of area on N. The first programmable logic arrays (PLAs) improved on this with two-level AND–OR logic planes (each plane in a wired-AND or wired-OR structure along with inverters can build any AND or OR logic term) that closely match the structure of common logic functions and are significantly more area-efficient. An example PLA is shown in Figure 2.1(a).

These architectures evolved further with the realization that sufficient flexibility was provided by a programmable AND plane followed by a fixed OR plane, in the programmable array logic (PAL) devices that were introduced in 1977 by Monolithic Memories Incorporated (MMI) [39]. As shown in Figure 2.1(b), it is notable that these devices contained programmable combinational logic which fed fixed sequential logic in the form of D-type flip–flop macrocells.

With these devices, logic functions must be implemented using one or more levels of two-level logic structures. Device inputs and

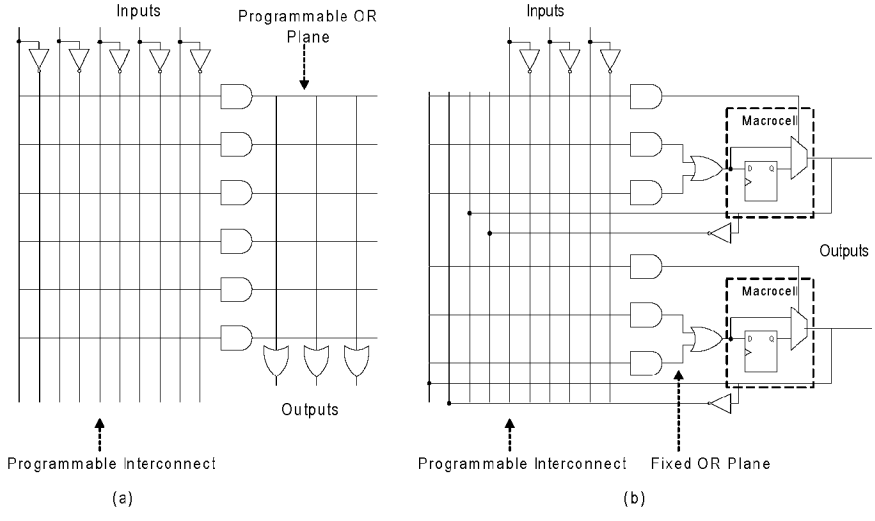

Fig. 2.1 PLA and PAL architectures.

intermediate combinational sums are fed into the array via a programmable interconnect that is typically a full cross-bar, leading to significant interconnect costs for this programmable architecture. For datapath and multi-level circuits, the area costs of two-level implementation quickly become prohibitive.

The first static memory-based FPGA (commonly called an SRAM-based FPGA) was proposed by Wahlstrom in 1967 [203]. This architecture allowed for both logic and interconnection configuration using a stream of configuration bits. Unlike its contemporary cellular array counterparts, both wide-input logic functions and storage elements could be implemented in each logic cell. Additionally, the programmable inter-cell connections could be easily changed (through memory-configurability) to enable the implementation of a variety of circuit topologies. Although static memory offers the most flexible approach to device programmability, it requires a significant increase in area per programmable switch compared to ROM implementations. It is likely this issue delayed the introduction of commercial static memory-based programmable devices until the mid-1980's, when the cost per transistor was sufficiently lowered.

The first modern-era FPGA was introduced by Xilinx in 1984 [49]. It contained the now classic array of Configurable Logic Blocks. From that first FPGA which contained 64 logic blocks and 58 inputs and outputs [49], FPGAs have grown enormously in complexity. Modern FPGAs now can contain approximately 330,000 equivalent logic blocks and around 1100 inputs and outputs [23, 231] in addition to a large number of more specialized blocks that have greatly expanded the capabilities of FPGAs. These massive increases in capabilities have been accompanied by significant architectural changes that will be described in the remainder of this survey.

3

Programming Technologies

Every FPGA relies 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 effect on programmable logic architecture. The approaches that have been used historically include EPROM [81], EEPROM [68, 174], flash [92], static memory [49], and anti-fuses [38, 93]. Of these approaches, only the flash, static memory and anti-fuse approaches are widely used in modern FPGAs. This survey focuses primarily on static memory-based FPGAs but, in this section, all these modern programming technologies will be reviewed to provide a more complete understanding of the advantages and disadvantages of static memory-based programming.

3.1 Static Memory Programming Technology

Static memory cells are the basis for SRAM programming technology which is widely used and can be found in devices from Xilinx [221, 224, 225, 227, 228, 229, 231], Lattice [124, 127], and Altera [18, 21, 22, 23, 24, 25]. In these devices, static memory cells, such as the