Hard vs. Soft: The Central Question of Pre-Fabricated Silicon

Jonathan Rose

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering

University of Toronto

jayar@eecg.utoronto.ca

Abstract

It is possible to foresee the day when prefabricated, programmable devices such as Field-Programmable Gate Arrays (FPGAs) are used as the dominant silicon implementation medium. This paper will explore the forces that already drive in that direction and the architecture, CAD and circuit enhancement opportunities that may also help to make it happen. We will focus on the central question in FPGA architecture: what hard, *dedicated* circuit structures should be included on the FPGA? These structures are contrasted with the regular *soft* fabric, which can always be used to implement logic functions, but with less efficiency and performance. We will discuss the trade-offs involved, and the requirements for CAD tools and algorithms needed to support these hard structures. An interesting specific case that will be addressed is whether processors should be implemented in hard or soft form. Finally we will look at an alternative: enhancing the capability of the soft fabric itself.

1. Introduction

Over the past decade, Field-Programmable Gate Arrays have become a widely used implementation medium for the creation of digital circuits. Their native advantages of instant fabrication and low-cost/risk silicon have lead to the vast majority of all design starts being done using FPGAs. Even so, they still represent only about 5% of the total market for digital silicon, by dollar volume. Several trends in the technology and economics of the alternative - fully fabricated integrated circuits - have lead many to believe that the market share of pre-fabricated, programmable silicon will dramatically increase:

1. As integrated circuit process scaling continues to produce smaller transistors and wires, the number of difficulties that have arisen in the manufacture and design of ICs has increased dramatically. The most recent example of this is the high leakage current in transistors fabricated in the 90nm process node [3], which threatens to make the creation of custom ASICs untenable. Other examples include the ever-increasing complexity of Optical Proximity Correction (OPC) required in the sub 100-nm regimes. By contrast, an FPGA user does not have to worry about any of these issues - the FPGA vendor is required to deal with the issues once, for all users of the silicon, alleviating the design headache for the user.
2. The market risk of inventory. An ASIC user is required to predict, long in advance, how many devices his market will require. If he chooses too many, he pays the steep cost of inventory. If he chooses too few then he may miss important market opportunities or lose the ability to acquire market share. When an FPGA is used, the vendor shoulders the inventory risk, which is shared across a much larger number of customers.

In addition to these advantages, the creators of prefabricated, programmable silicon have several interesting avenues to improve the performance, density and power consumption of the devices. Key among these is the selection of appropriate hard, dedicated circuit structures for inclusion with the soft fabric. In the following section we will define the context for the decision to include (or not) such structures. The subsequent section discusses a specific and important example of the issue - processors. Finally, we look at an alternative avenue to hard structures - improving the capability of the soft fabric itself.

2. Heterogeneity in FPGAs

The central question of FPGA architecture has always been to decide what specific, dedicated structures should be included in addition to the basic soft logic fabric. In this section we will provide

some basic definitions of homogeneity and heterogeneity to permit the discussion of when and how to employ hard, heterogeneous FPGA architectures.

2.1 Definitions

It is instructive to give a clear definition of what constitutes the soft logic fabric, to distinguish it from hard, dedicated structures. The *soft logic fabric* of an FPGA consists of an array of combinational logic elements - each consisting of a logic function implemented as a set of gates, or most typically, a lookup table (LUT) that is connected through a programmable routing fabric. We will call any other circuitry that the designer employs in the device a *hard dedicated circuit structure,* and define it as follows: a hard dedicated circuit structure is a circuit structure that allows the implementation of a logic function *that could also be implemented in the soft logic fabric*. Note that with this definition, a dedicated flip-flop inside a logic block is considered a hard circuit structure, as it should be - it is possible to build flip-flops from an interconnection of programmable LUTs or gates. Certainly dedicated flip-flops are exceedingly common [11][12], with only much older FPGAs being built without them [2]. Similarly, modern commercial FPGAs contain dedicated logic within each block to support the arithmetic carry and sum functions [11][12] as well as some memory functions [12].

It is appropriate to distinguish between two kinds of heterogeneity: the first kind is exemplified by the flip-flop and dedicated carry logic which appears along side the combinational logic in the soft logic fabric, in every logic block, and is arrayed across the device in the basic logic tile. This type of heterogeneity will be termed *soft fabric heterogeneity*. It can be distinguished from another type of heterogeneity in which there exist other tiles that are completely different from the basic logic tile. An example of this is the block memory such as the multi-bit block RAMs that appear in modern FPGAs (ranging in size from 2K bits to 64K bytes) such as that found in the Altera Flex 10K, 20K Stratix and Stratix II, series [11], and the Xilinx Virtex, Virtex II, II Pro and Spartan II and III series [12]. These blocks of RAM typically appear in vertical columns that separate columns of the basic tile array. A second example of this kind of heterogeneity is the Multiply-Accumulate (MAC) block that appears in the Stratix and Stratix II FPGAs [6][11]. This kind of heterogeneity will be termed *tile-based heterogeneity* to reflect the fact that a different tile is part of the array. An example of an FPGA that has heterogeneous tiles is shown in Figure 1. It shows the Altera Stratix II FPGA with separate columns for small and medium-sized RAM blocks, as well as the DSP/MAC block.

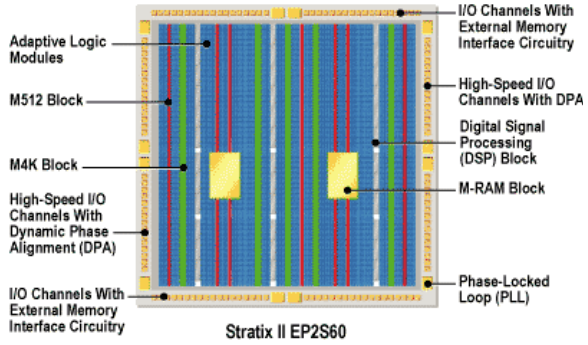

Figure 1 - A Heterogeneous FPGA - Stratix II [11]

2.2 Hard vs. Soft Structures

The benefit of either type of heterogeneous circuit structure is that, if an application circuit can make use of it, the structure is smaller, fast and consumes less power than the equivalent structure built out of the programmable fabric. However, if the structure isn't used by an application, then it is wasted, and the user would likely have benefited from the same area employed simply as the regular soft fabric. The key issue here is whether a need for the hard structure appears often enough in the set of target applications of the FPGA, and, in the case where the architect seeks enhanced speed, if that structure appears on the critical path of designs when implemented as part of the soft fabric.

The naïve user who fancies themselves as an architect of an FPGA will often, when faced with a design challenge, immediately express the wish that his/her specific application should exist as a hard structure on the programmable device. This will almost certainly solve the power, speed or area issues that the designer faces, and provide a significant advantage to that one application.

However, the only way to determine if a hard structure is truly useful in the FPGA context is to empirically measure the net benefit of the structure across a set of benchmark applications that are representative of the target market. For a generic FPGA this target market is essentially all applications. The typical way this empirical measurement is done [1] [6] is to build a CAD system that can target the new architectural structure and to synthesize the benchmark circuits into an FPGA that employs that structure, and compare it against an FPGA that does not use it. Doing so correctly measures the full benefit in the context of

complete synthesis, placement, routing, timing analysis and power analysis.

For example, [8] showed that there was a compelling empirical case (in terms of logic density) for including flip-flops on FPGAs.

The net benefit must be large enough to justify the development of both the hardware and software for the architecture. One could use the following equation: consider the addition of some hard dedicated structure, **H**, to an FPGA. Through the empirical method (in which you've gathered a set of benchmarks that represent a sensible target market) the average gain in speed of those benchmarks in an FPGA using structure **H** over those that don't is S (expressed as a ratio). Similarly the gain in area efficiency (amount of logic implemented per unit area) is A, and power reduction is P. Secondly, assume that you can sensibly determine the relative weight of goodness of Speed, Area and Power, with coefficients **s**, **a** and **p** such that **s**, **a** and **p** sum to 1. Then the structure **H** is useful when the value of the following equation exceeds some threshold of goodness:

$$G = \mathbf{s}S + \mathbf{a}A + \mathbf{p}P \qquad (1)$$

That threshold is likely some ways above 1, to justify the extra investment required to support the structure.

2.3 A Little Bit Soft

So far our notion of heterogeneity is fairly brittle, in which there is a very specific function implemented in the new structure. It may well be beneficial to strike a compromise between a too-specific hard structure and a fully programmable (but expensive) soft fabric. If the central question of FPGA architecture is to determine what hard dedicated structures to include on an FPGA, then the *golden rule* is to build structures that are always useful, even if that use is less than perfectly efficient. The more useful a hard structure is, across a wider range of applications, then the greater its net benefit - provided the cost of the extra functionality is not excessive.

A hard structure is made more useful by choosing something that is inherently flexible, or by adding a certain amount of programmability. The Memory blocks that have been employed in FPGAs are good example of this - the same set of bits are typically configurable across a range of memory depths and widths [9][10][11][12]. Similarly, the DSP/MAC blocks in Stratix [6] and Stratix II have a range of configurable multipliers available.

This kind of configurability represents a spectrum between fully general soft fabric (that is flexible but slow and large) and very specific functions that can't be modified even slightly. An extreme version of this "softness" is to allow the complete, programmable replacement of the hard structure (through the use of multiplexers) with the regular soft fabric. This is possibly a good idea since the majority of the area is often taken in the routing, not in the logic.

2.4 CAD Considerations

To be able to both make the architectural decisions described above and to usefully employ a hard dedciated structure on an FPGA, a complete CAD flow is required that can make effective use of the structure. Almost the entire flow is typically impacted by a hard structure. The front-end HDL elaborator may be required to recognize HDL code for combinational and sequential structures that will map into the hard structure. Either the front-end or the synthesis step has to account for the typically limited number of hard structures available on the device - when there are an insufficient number of hard structures, this step must select which user functions should be implemented in the hard logic, and which in the soft fabric.

The placement step is faced with restrictions implicit in the case of tile-based heterogeneity, and the routing must be cognizant of different logic pinouts.

3. Processors - Is Soft Better?

One of the most important structures in digital circuits is the instruction set processor, as it is a central component of almost every digital system. Several FPGA vendors have proposed and implemented hard on-chip processors [13][11][12]. More recently, vendors have supplied the complete infrastructure to support soft processors, such as the Nios soft processor from Altera and the Microblaze processor from Xilinx. Soft processors are built from the soft logic fabric, as well as any useful heterogeneous structures such as flip-flops, memory and multipliers. Hard processors have clear performance and density advantages over soft processors - a modern desktop processor runs above 3GHz clock speed, whereas a soft processor runs in the territory of 100MHz, a factor of 30 difference. Some of this difference, however, can be made up through the use of specialized instructions built in the soft fabric. Tensilica, for example, builds hard processors with such specialized instructions on ASICs, and claims that such instructions can result in speedups ranging from 12% to a factor of 11.3 times [7]. If the latter is possible in soft processors, then it is possible that the deficit between fully hard

and soft processors can almost completely be recovered, still in the context of configurability.

Soft processors also offer two other advantages - they can have a configurable architecture, allowing a trade-off between performance and area by changing the architecture. Also, a programmable quantity of processors can be instantiated as needed, each tuned to required area and performance specifications.

4. Making Soft Fabric better

An alternative to adding hard structures to an FPGA is to find ways to improve the performance of the soft logic fabric. If this can be done, it more easily makes all systems and applications faster, cheaper and lower power. A good example of this was recently done with the Altera Stratix II [11] FPGAs. Instead of using the now-common four-input LUT as the basic logic element this device employs a more complex 8-input combinational structure (illustrated in Figure 2) that is capable of implementing the following:

- Two four-input LUTs
- One 6-input LUT
- Two 6-input LUTs that share four inputs
- One five-input LUT and one 3-input LUT

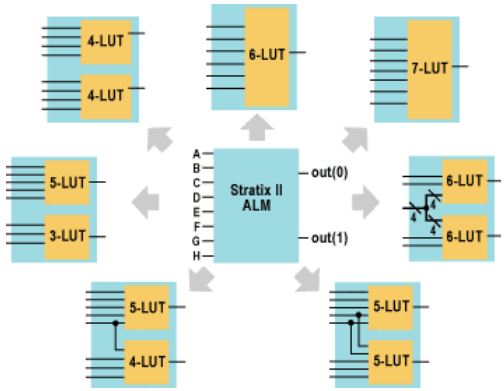

Figure 2 - New Soft Logic Function [11]

as well as several other combinations. One advantage of this structure is its ability to implement wider-input functions, reducing the depth of logic circuits and thereby increasing its speed. This speed advantage has been empirically measured [11]. A second advantage of this structure is that it can implement two 4-to-1 multiplexors that share the same data inputs, which is a common structure in switching fabrics. This results in a significant area density win, which can be applied to many circuits.

5. Conclusions

We have discussed various aspects of the central question in FPGA architecture - "what structures should be made into hard and specific circuits on the device?" While at first glance many structures seem to be a win, there are hard rules of architectural sensibility that limit how often such choices should be made. Good choices will hasten the day, however, that programmable silicon dominates the market for all digital silicon.

6. References

[1] V. Betz, J. Rose, and A. Marquardt, **Architecture and CAD for Deep-Submicron FPGAs**, Kluwer Academic Publishers, 1999.

[2] A. El Gamal, et. al, ‘‘An Architecture for Electrically Configurable Gate Arrays,’’ Proc. 1988 CICC, May 1988, pp. 15.4.1 - 15.4.4.

[3] R.S. Guindi and F.N. Najm, “Design techniques for gate-leakage reduction in CMOS circuits,” In IEEE International Symposium on Quality Electronic Design, pages 61–65, 2003.

[4] www.actel.com

[5] E. Ahmed and J. Rose, “The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density,” in FPGA 2000, ACM Symp. FPGAs, February 2000, pp. 3-12.

[6] D. Lewis, V. Betz, D. Jefferson, A. Lee, C. Lane, P. Leventis, S. Marquardt, C. McClintock, B. Pedersen, G. Powell, S. Reddy, C. Wysocki, R. Cliff, and J. Rose, “The Stratix Routing and Logic Architecture” in FPGA ‘03, ACM. Symposium on FPGAs, February 2003, pp. 15-20.

[7] D. Goodwin and D. Petkov, “Automatic Generation of Application Specific Processors,” Proceedings CASES ’03, November 2003, pp. 137-147.

[8] J.S. Rose, R.J. Francis, D. Lewis, and P. Chow, "Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency," IEEE JSSC, Vol. 25 No. 5, October 1990, pp. 1217-1225.

[9] T. Ngai, J. Rose, S. Wilton, “An SRAM-Programmable Field-Configurable Memory” in CICC ‘95, the IEEE Custom Integrated Circuits Conf., Santa Clara, CA, May 1995, pp. 499-502.

[10] S. Wilton, J. Rose, Z. Vranesic, “Architecture of Centralized Field-Configurable Memory,” 3rd ACM Int’l Symposium on Field-Programmable Gate Arrays, FPGA ‘95, Feb 1995, pp. 97-103.

[11] www.altera.com

[12] www.xilinx.com

[13] www.triscend.com