

FPGA Design Principles (a tutorial)

Dwight D. Hill
AT&T Bell Labs

Ewald Detjens
Exemplar

Abstract

This tutorial describes current Field Programmable Gate Array (FPGA) technology from three directions: programming technology, functional block capabilities, and routing architectures. It then sketches FPGA CAD issues: logic synthesis for FPGA's, placement and routing, and issues in the design of new FPGA's.

1 Technology Taxonomy

FPGA's constitute a radical new direction in microelectronics. This is because they significantly reduce the overhead involved in designing and producing new logic circuits. Today's FPGA's can be viewed as an evolution of PAL's where size is increased by an order of magnitude, or a refinement of mask-programmed gate arrays, where the reprogramming time and cost are drastically reduced. To the CAD professional, FPGA's represent a new challenge: what is the best way to exploit this new technology?

Users choose a part based on external metrics such as speed, density, cost and CAD effectiveness for their particular application. Suppliers try to optimize these across many applications by making technology and architectural tradeoffs. The result is a wide spectrum of features: anti-fuse versus reprogrammable configuration, block-structured versus channel-structured routing, and lookup table (LUT) versus multiplexor versus sum-of-products logic.

Generally speaking, the technology and architecture of the routing fabric is the most important factor in determining the effectiveness of an FPGA for a particular application. FPGA routing enables (almost) arbitrary connection among logic blocks, but at the cost of tying up more area and incurring more delay than present in a mask programmed part. Likewise, the logic architectures of FPGA's are larger and slower than mask defined gates, since their functionality must be programmable. But in comparison to the routing fabric, their delays tend to be more predictable and less of a limiting factor.

2 Basic Technology Issues

Anti-fuse parts, typified by the Actel Act-1 and Act2 architectures [2], use special processing steps to fabricate insulating layers between the routing wires. These anti-fuses have relatively low resistance when turned on and are relatively compact. Unfortunately, they require high voltages to activate, and large isolation transistors are usually required to protect logic from the programming voltages. Their channelled architecture is relatively friendly to CAD tools and has advantages for predicting routing delays, since almost all designs that can be fit onto the chip can be routed within a limit of 2 or 3 anti-fuses in series. However, the lower resistance of the anti-fuses comes at the cost of reconfigurability.

One alternative to anti-fuses is SRAM-based programmable interconnection points (PIP'stm). These consist of an SRAM cell controlling one or more FET's. When the SRAM is set, the FET is turned on, and the application signal can propagate from one

node to another. Reconfigurable parts are attractive for many reasons. Most obviously, they can be used over and over again on the lab bench to debug and/or refine a design, (which is important since many projects die before even the first copy is produced). Also, even after the design is working, many low volume products choose to include reconfigurable FPGA's in their production versions. This is because features can be added, errors corrected, and interfaces modernized with little or no cost in the field. Such changes can be implemented by shipping a new ROM, floppy, or even by modem, depending on the application.

Reconfigurable parts are typified the Xilinx 3000 and 4000 series [13]. These architectures consist of a set of Configurable Logic Blocks (CLB'stm) connected to routing wires, which are in turn connected to other routing wires by "switch boxes." Typical parts would have from 60 to 320 CLB's, and an equal number of switch boxes. A two-dimensional array of CLB's and switch-box blocks define the routing architecture. Because the routing is SRAM based, each PIP is several times larger than an anti-fuse. It is therefore essential to get the most routing functionality out of each PIP. This is done by increasing the complexity of the routing fabric, and by tolerating a greater number of PIP's in series. (On these chips, "maze routers" are often used, since they can handle arbitrary node-to-node interconnections.) The result is a system that can have high speed for some application nets, but not all. A complex, timing-directed CAD system is needed to make best use of these architectures.

Reconfigurable parts can also be based on EPROM technology, as typified by the Lattice ispLSI parts [8]. In some ways, these parts resemble a collection of PAL's connected with a pool of on-chip routing. In a PAL, the logic functionality is intermingled with the routing in an array of minterms.

3 CAD Issues

Today's system designers make wide use of Programmable Logic Devices (PLD's) and mask-programmed application specific integrated circuits (MP-ASIC's), a category which includes gate-arrays, standard-cell and full-custom parts. FPGA architectures differ from PAL's and MP-ASIC's by varying degrees. The more differences there are, the more nontraditional optimization techniques are necessary, whether performed by hand or by a program.

3.1 Design Capture and Environment

PLD's are typically designed with a low-level logic specification language. Placement, routing, and testing are simplified and hidden from the user. By contrast, MP-ASIC's are usually specified with schematics, and recently, via hardware description languages (HDL's) such as VHDL. Today's larger FPGA's tend to have a design flow closer to that of MP-ASIC's rather than PLD's. For example, place-and-route is an important and sometimes painful step, not guaranteed to complete. Also, since on-chip delays can vary by several orders of magnitude, back annotation of routing delays may be needed, just as in an MP-ASIC. But there are two

ways that FPGA CAD tends to differ from ASIC CAD: the architecture of the parts tends to eliminate the need for test vectors, and the majority of users today still use PC environments, rather than UNIX™ workstations.

3.2 Synthesis issues

Synthesis begins with logic optimization, which is the process of restructuring logic to minimize delay or area. Traditionally, PAL's and PLD's had sum-of-products architectures that required two-level minimization. Programs like MINI and Espresso were written to address this problem.

But most MP-ASIC's and FPGA's support multiple levels of logic. Multi-level minimization involves decomposing complex logic functions and finding common subfunctions [10], [12]. MP-ASIC's benefit from algebraic techniques, where logic functions are treated simply as algebraic formulas that can be decomposed by factoring. The MIS program from UC Berkeley uses this type of technique to minimize the number of logic literals that appear in the logic. These "literal count" metrics are well suited to mask programmed CMOS parts, but are less effective for FPGA's.

For example, LUT's have the same size and delay no matter what function they implement. The best fit for them is usually complex functions with a constant number of inputs (typically 4 or 5). Similarly, in Actel's logic modules, multiplexors are the dominant element. Optimization based on Binary Decision Diagrams (BDD's) or "If-Then-Else" graphs is appropriate.

Ideally, one would like to synthesize logic directly into the available hardware primitives. However, this is not yet generally done. Rather, most systems use a generic optimization package followed by a technology binding stage. Technology mapping is the process of binding logic to specific cells or structures on the target device. For an ASIC, this means selecting cells from a library. For LUT based FPGA's, this means collapsing AND/OR/INV gates into as few LUT's as possible [4]. For a multiplexor architecture it means graph matching, combined with hardware specific facilities such as bridging inputs or tying them to power and ground [3] [9] [6]. Two other FPGA specific problems are *packing* which is putting more than one LUT and/or flip-flop in each CLB, and *partitioning* which is dividing a large circuit across several FPGA's.

3.3 Integration and Performance

CAD systems strive to extract the highest possible performance from the available hardware. Unfortunately, this is difficult in logic synthesis for FPGA's because most of the delay is in the routing, and the routing cannot usually be characterized until after synthesis is complete. (One area that CAD can influence is state assignment. For MP-ASIC's, the best state encoding is usually a minimum-length encoding. For FPGA's with large register resources one-hot encoding is often better.)

At the present time, human intervention in logic assignment, placement and routing is sometimes needed to achieve the full potential of the silicon, especially in the SRAM parts. Future systems will probably have to integrate the physical design into the logic synthesis. This may mean that the physical place-and-route software will be integrated with the optimize-map-pack software, to achieve a total solution that can intelligently trade off gate-counts against routing delays.

4 Evaluation of New Architectures

Another area of CAD support for FPGA's is the evaluation of new FPGA architectures. This is of greatest interest to the FPGA's makers, but users may find the discussions interesting. Some of the work has concentrated on finding "optimal" values of architecture parameters such as the number of routing tracks [1] [11], or the type of lookup tables [7]. One limitation of such studies is that real FPGA architectures are usually very complex, and some information is lost if the study is based on a simplified model, such a routing fabric without "long-lines".

Other CAD research has considered the question of software structures to describe and manipulate FPGA primitives so that complex architectures can be accurately modeled [5]. But such studies tend to simplify the CAD algorithms involved, choosing more general but less effective techniques. And all paper studies lose accuracy from the absence of accurate electrical information, such as routing parasitics. In the end, it is difficult to predict the overall effectiveness of an architecture without actually laying it out and optimizing the CAD system around the measured values.

5 Closing Thoughts

At the present time, FPGA sales are increasing dramatically, as is the number of FPGA companies. In this environment each vendor tries to provide some distinguishing features that improve performance and reduce cost. But users may want a portable environment that allows them to migrate freely among vendors, and between FPGA's and mask programmed parts. This schism is partially resolved when the design can be processed automatically via synthesis from a common HDL, perhaps even using a common intermediate form such as the Library of Parameterized Macros (LPM). But other issues, such as investments in fuse-programmers, software, and engineer training, tend to lock users into one supplier. In many ways, the flurry of FPGA activity is similar to the melee that took place during the early days of microprocessors; its resolution will strongly influence the direction of microelectronics and of CAD in the 1990's.

References

- [1] S. Brown, J. Rose, and Z. Vranesic. A detailed router for field-programmable gate arrays. Proc ICCAD, pages 382–385, Nov. 1990.
- [2] A. ElGamal, J. Green, J. Reyneri, E. Rogoyski, K. El-Ayat, and A. Mohsen. An architecture for electrically configurable gate arrays. IEEE J. Solid State Circuits, 24(2):394–398, apr 1989.
- [3] S. Ercolani and G. D. Micheli. Technology mapping for electrically programmable gate arrays. 28th ACM/IEEE Design Automation Conference, pages 234–239, 1991.
- [4] R. Francis, J. Rose, and K. Chung. Chortle: A technology mapping program for lookup table-based field programmable gate arrays. Proc. Design Automation Conference, pages 613–691, 1990.
- [5] D. Hill. A cad system for the design of field programmable gate arrays. Design Automation Conference, 1991.
- [6] K. Karplus. Amap: a technology mapper for selector-based field-programmable gate arrays. 28th ACM/IEEE Design Automation Conference, pages 244–247, 1991.
- [7] J. L. Kouloheris and A. ElGamal. Fpga area versus cell granularity - lookup tables and pla cells. in the book FPGA's by Abindon Books, 1991.
- [8] Lattice Corporation. pLSI and ispLSI Data Book and Handbook. Lattice, 1992.
- [9] F. Mailhot and G. D. Micheli. Technology mapping with boolean matching. European Design Automation Conference, Glasgow, Scotland, pages 212–216, March 1990.
- [10] R. Murgai, Y. Nishizaki, N. Shenoy, R. Brayton, and A. Sangiovanni-Vincentelli. Logic synthesis for programmable gate arrays. Proc. Design Automation Conference, pages 620–625, 1990.
- [11] J. Rose and S. Brown. Flexibility of interconnection structures in field-programmable gate arrays. IEEE Journal of Solid State Circuits, 26(3):277–282, March 1990.
- [12] G. Saucier, P. Sicard, and L. Bouchet. Multi-level synthesis on pals. EDAC, pages 542–546, 1990.
- [13] Xilinx Corporation. Xilinx Programmable Gate Array User's Guide. Xilinx, 1988.