

Reconfigurable Computing I: Motivation and Concepts

Lecture 6

February 6, 2004

Scribe: Kapil Sachdeva

Overview

This lecture introduces reconfigurable computing and the field-programmable gate array (FPGA) by discussing the density advantage of configurable computing and some of the trade offs involved in designing and FPGA. Some important conclusions were that wiring and routing area dominate total area, three and four input lookup tables are ideal for area efficiency, five and six input lookup tables are ideal for speed efficiency, and FPGAs achieve a high computational density by utilizing shallow instruction memories at low fabrication costs.

Reconfigurable Computing

In configurable computing, a set of gates is configured to implement some specified behavior. The configuration is usually defined by configuration bits that define the way interconnects between gates behave. Configurable computing allows one to experiment with circuits tailored to perform well for specific applications. This is because the fabrication time is more reasonable than for conventional processors. Configurable computing has proven to be an economical method in RSA decryption, DNA sequence matching, signal processing, chip emulation, and cryptography. Often times, a task that takes hundreds of cycles on a conventional microprocessor only takes one or two cycles on a reconfigurable computing device.

FPGAs

Among the more popular types of reconfigurable computing are programmable logic arrays (PLAs), programmable logic devices (PLDs), and field-programmable gate arrays. The most popular type of configurable computing device, the FPGA is a customizable array of combinational logic blocks (CLBs) connected by a programmable interconnect. FPGAs were first used as “glue logic” to connect large integrated circuits. For example, they were used in generating global control signals and for data formatting. With the aid of computer-aided design (CAD) tools, FPGAs reduced system complexity and cost while improving performance. The interconnection can be programmed after fabrication. In an FPGA, tasks are performed by composing operations spatially, as opposed to linking them temporally in traditional processors. The majority of FPGA implementations use lookup tables (LUTs) as the combinational logic elements. The interconnect accounts for most of the area in a FPGA cell.

The most basic CLB consists of a LUT, a latch, one or more multiplexors, and a tri-state buffer to drive the output (Figure 1). The multiplexors are used to select, for example, the latched output or the combinational LUT output. The register or latch is used to optionally store the output of the LUT. The LUT implements a set of combinational logic functions. An n -input LUT can be used as to implement any n -input logic function and can be thought of as a $2^n \times 1$ memory. The inputs are used to select one of the memory locations or latches.

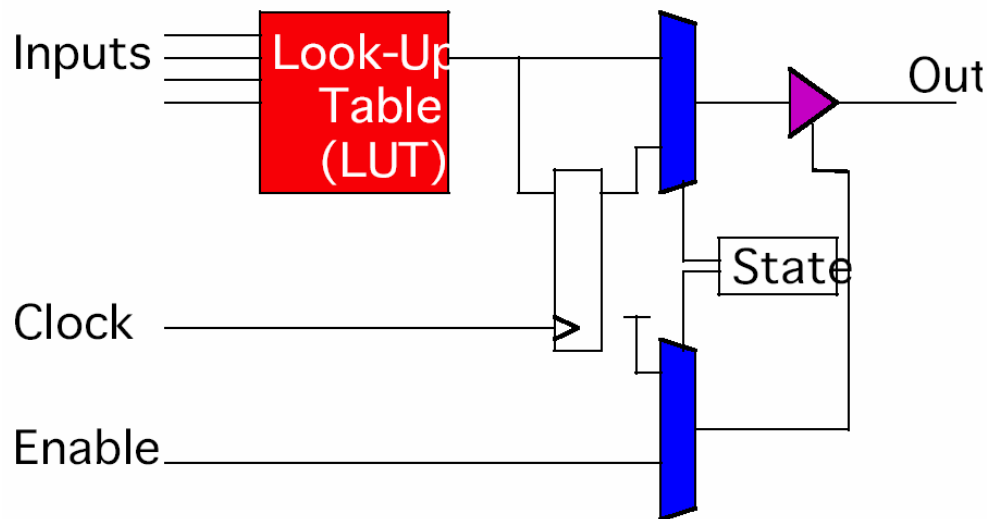


Figure 1. A basic combinational logic block.¹

The LUT itself is usually a static RAM lookup table. The interconnect is implemented using pass transistors and multiplexors. Connections from the logic block to the channel are made via MUXs controlled by static RAM. Static RAM bits control connections made by transistors between horizontal and vertical channels. Sometimes antifuses are used to implement the logic block and the interconnect.

The antifuse is used to perform a range of combinational and sequential functions. EPROM technology has also been used to implement the logic block. The architecture is similar to a PLD. The interconnect is a two-level hierarchy, the first level connecting logic blocks and the second level connecting groups of logic blocks. Other architectures have been proposed, including one that uses a NAND, a latch, and a MUX as the logic block.

An FPGA is configured by first creating a design in a hardware description language like Verilog or VHDL or using a schematic editor. That design is then partitioned, placed, and routed. A simulator is then used to verify the design. Finally, the design is loaded onto the FPGA device

There are a number of parameters used in designing an FPGA. The number of inputs to the lookup table is a trade off between the number of CLBs used and the size of each CLB. One must also choose how to implement the combinational logic: Should a LUT or a programmable and-or-invert or something else be used? Should there be a latch in the CLB? Should one include additional functionality, such as carry chains?

The Density Advantage

FPGAs have a higher throughput despite sometimes requiring tens of cycles of latency to calculate a result. However, since it performs the computation with many processing elements in parallel, it achieves higher throughput. Unlike conventional processors, the fundamental units of an FPGA only hold a single instruction. Thus, FPGAs have lower instruction overhead than processors and can exploit more parallelism in a cycle. Furthermore, FPGAs can control their operations at the bit level, making efficient use of their computational power. FPGAs have an order of magnitude more computational power per unit area than processors by forgoing the deep instruction memories found in processors and allowing fine-grained control at the bit level. Figure 2 compares the computational densities of a wide range of processors and FPGA implementations.

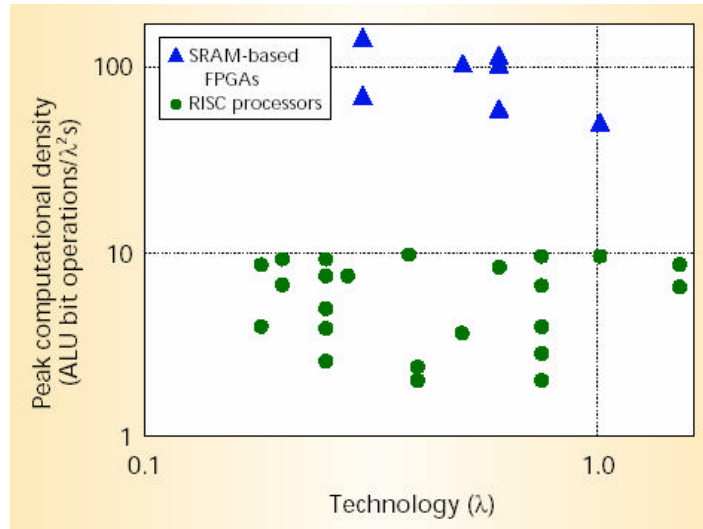


Figure 2. A comparison of FPGA and processor computational densities.²

Processors sacrifice some computational density in order to tightly pack larger computation onto the die. They use their SIMD control to help offset the cost of deep instruction memories. The FPGA pays little for its bit-level granularity. Often times, hardwired functional units provide superior performance density than reconfigurable devices or specialized processors. To fully take advantage of this increased performance, though, the hardwired unit must be used frequently while not being too general. Though custom units increase performance density, they often conflict with the needs of an application. Configurable computing devices can provide modest performance gains while still remaining flexible enough to match the needs of the application.

The Effect of Logic Block Functionality on Area

Logic block functionality is the number of logic blocks required to implement an unspecified set of circuits. If a logic block has little functionality interconnection area rises. On the other hand, if a logic block has too much functionality it may be underutilized. Two trade offs in designing a logic block are the number of inputs to use to a lookup table and whether the design should include a latch. Experiments indicate that, regardless of whether the block has a latch, the best number of inputs to use is between three and four. In addition, a latch should be used because it reduces chip area. It was also found that interconnection area dominates active area; hence, the trade off between routing area and logic block functionality is important. Also, as logic block functionality decreases, the number of logic blocks necessary increases. However, the number of pins on the block increases with functionality. The lower density and speed of FPGAs stems from the higher resistance and capacitance of the large interconnect switches. Inserting more logic into a block improves area efficiency and FPGA speed by reducing wiring and lowering the number of connections. It was found that routing area increases in proportion to the number of pins on a logic block. Thus, for an increase in logic block functionality to be beneficial, the number of logic blocks must be reduced enough to compensate for the increase in routing area.

The key trade off to consider is the one between routing area and logic block functionality because routing area dominates total active area. Figure 3 shows the relationship between K, the number of inputs to an arbitrary combinational logic block, and total area for several bit areas (programming

technologies). It shows that, regardless of programming technology, the lowest area is achieved with K between three and four. The relationships between K and the active and routing area were derived to explain this result.

The Area Question

Total active area is the product of the number of logic blocks and the area of a single logic block. The number of logic blocks is inversely proportional to K ; that is, more inputs to a logic block beget more functionality with the block. The logic block area increases exponentially with K . When taken together, these two relationships result in 2, 3, and 4 input logic blocks giving the lowest total active area. See Figure 4. However, the active area pales in comparison to the routing area for values of K surrounding the minimum total-area K .

Routing area equals the product of the number of logic blocks and the routing area per block. The routing area is the space taken by the routing tracks on two of the four sides of the logic block. The routing area per block was observed to be an increasing function of K . The product of the curves representing the K -routing-area relationship and the K -number-of-blocks relationship gives a minimum when K is between three and four. See Figure 5.

So, while increasing the functionality of a logic block—increasing K —reduces the total number of logic blocks, it is offset by the increase in routing area due to the larger number of pins per block. These same experiments were run on logic blocks not containing a latch. Roughly the same final result was observed: The minimum total area occurs at K values between three and four.

The Latch Question

An important design trade off to consider is whether the CLB should contain a latch. Experiments indicate that logic blocks without a flip-flop require more blocks because flip-flops must be implemented using several logic-blocks. The number of blocks required to implement each test circuit increased between 1.4 and 2.3 times when the flip-flop was removed from the block. Still, the logic block size decreased between 2.1 and 2.5 times when the flip-flop was removed. Thus the active area decreases and the routing area increases. Thus, the total area increases. It is always better, then, to include a flip-flop in the CLB.

The Effect of Logic Block Functionality on Performance

When designing an FPGA, it is important to consider not only area trade offs but speed trade offs as well. Experiments indicate that K values of five and six produce the lowest total delay. The primary reason is that five and six input LUTs can implement common logic operations using the fewest levels of logic. Another reason is that their intrinsic combinational delay is relatively modest. Finer-grained combinational implementations, such as NAND gates and AND-OR gates, require many levels of hierarchy to implement common circuits and thus require a large routing delay. The multiplexor-based combinational logic block, however, does not suffer from these delays.

Figure 6 shows the total delay of the best types of logic blocks from each of the four classes—NAND, AND-OR, MUX, and LUT—verses the routing delay (D_R). NAND gates exhibit the worst delay. At low routing delays, the combinational delay dominates. Since MUXs are relatively fast, they have the lowest delay. At midrange and large routing delays, the number of logic levels should be minimized. Five and six input LUTs have the lowest average number of levels; thus, they have the smallest delays.

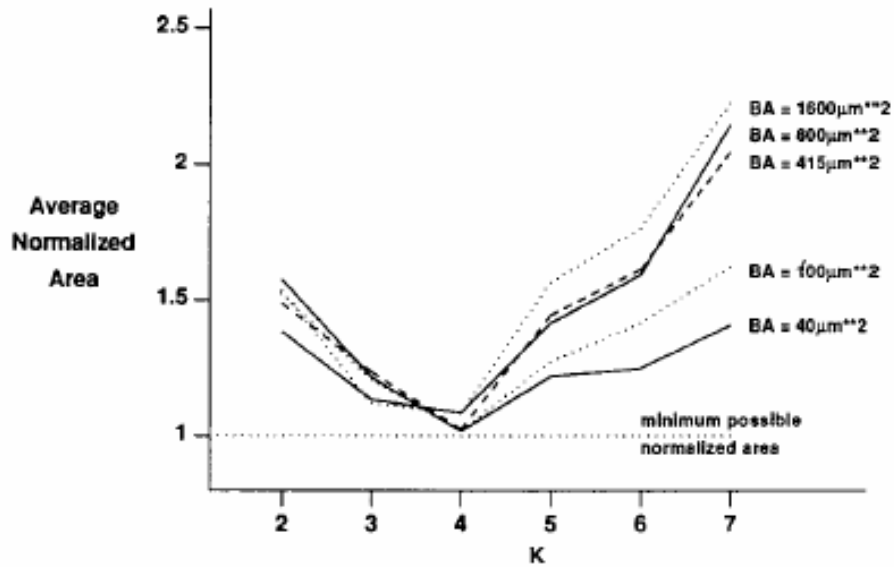


Figure 3. Average normalized area over all circuits versus K for circuits that contain a latch.³

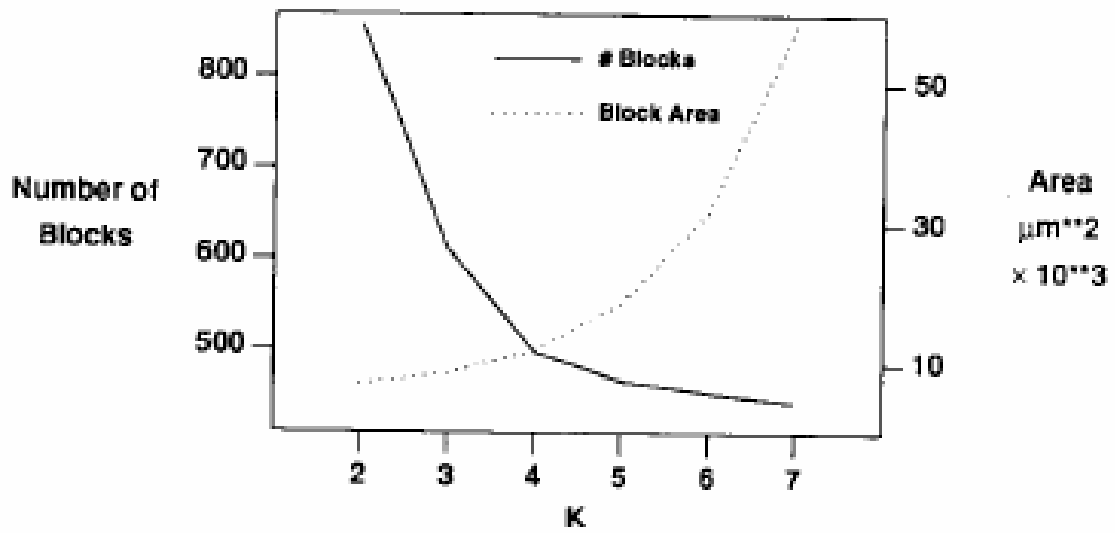


Figure 4. Number of blocks and active area versus K. The minimum of the product of the two curves occurs between K values of 3 and 4.⁴

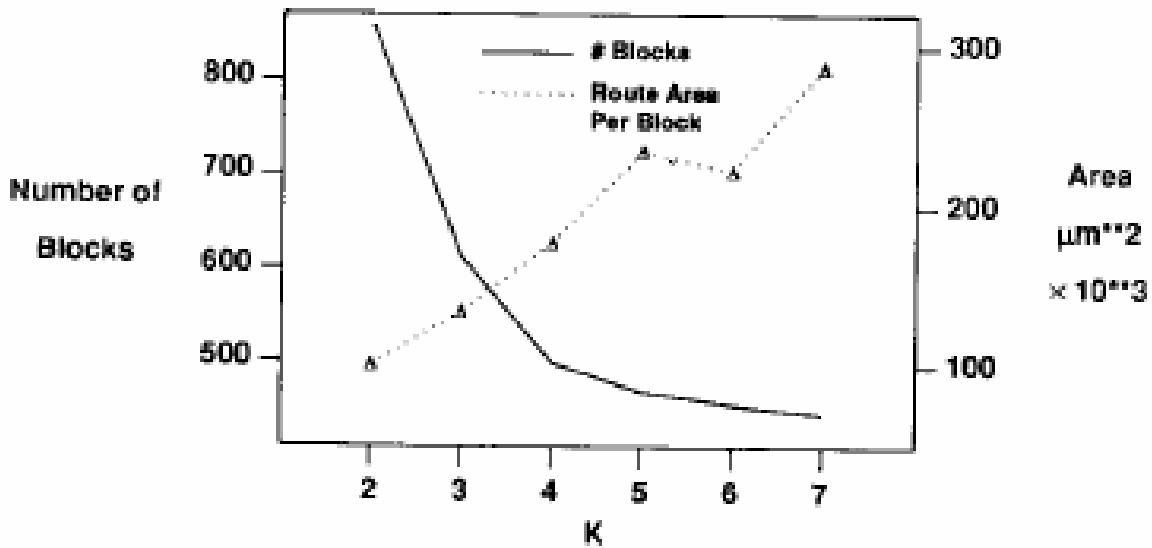


Figure 5. Number of blocks and routing area versus K. The minimum of the product of the two curves occurs between K values of 3 and 4.⁵

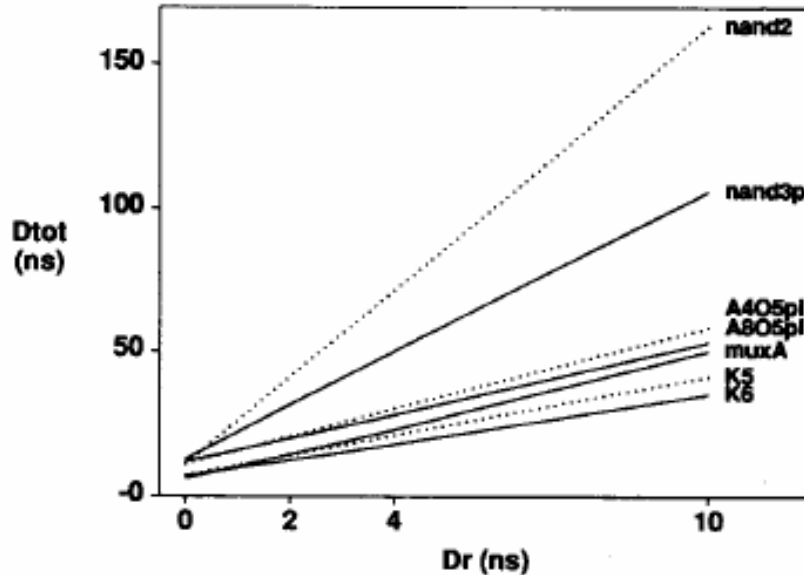


Figure 6. Total delay versus routing delay for the best blocks from each of the four categories—NAND, AND-OR, MUX, and LUT. K5 and K6 correspond to 5 and 6 input LUTs, respectively; nand2 and nand3pi correspond 2-input gate and a 3-input gate with programmable inversion; A405pi and A805pi correspond to five-product term 4-input and 8-input AND-OR blocks; and muxA corresponds to a logic block from [1].⁶

Conclusions

The trade offs analyzed above have shown that three and four input LUTs minimize total area and five and six input LUTs maximize speed. It has also been shown that wire delay and routing area dominate total area. These experiments, however, were performed over a decade ago and may very well be invalid today. Nonetheless, FPGAs provide run-time programmability as well as custom-circuit-level performance. They do not require nearly as much instruction memory and are easier to design than conventional processors. FPGAs achieve higher computational density by forgoing deep instruction memories. They can be used to exploit parallelism on data values and can use variable bit widths. Thus, they have been successfully used in an array of applications, such as cryptography and signal processing.

Readings

1. “The Density Advantage of Configurable Computing” by Andre DeHon. Appeared in the April 2000 issue of IEEE Computer.
2. “Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency” by Jonathan Rose, Robert J. Francis, David Lewis, and Paul Chow. Appeared in the IEEE Journal of Solid-State Circuits, October 1990.
3. “The Effect of Logic Block Architecture on FPGA Performance”, by Satwant Singh, Jonathan Rose, Paul Chow, and David Lewis. Appeared in IEEE Journal of Solid-State Circuits, March 1992.

References

- [1] A. El Gamal *et al.*, “An architecture for electrically configurable gate arrays,” *IEEE J. Solid-State Circuits*, vol. 24, no. 2, pp. 394-398, Apr. 1989.

¹ Figure Credit: “Lecture 6: Reconfigurable Computing I: Motivation and Concepts,” <http://www.crhc.uiuc.edu/ece497nc/>.

² Figure Credit: “The Density Advantage of Configurable Computing” by Andre DeHon.

³ Figure Credit: “Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency” by Jonathan Rose, *et al.*

⁴ Figure Credit: “Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency” by Jonathan Rose, *et al.*

⁵ Figure Credit: “Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency” by Jonathan Rose, *et al.*

⁶ Figure Credit: “The Effect of Logic Block Architecture on FPGA Performance” by Satwant Singh, *et al.*