Chapter 2. Technologies

2.1. Full Custom

A full custom integrated circuit (IC) is constructed entirely by its designer [13][18]. Each transistor, connection, and wire is laid out by hand. This allows a designer more control in optimizing a circuit's critical parts. A human designer in general can create a faster and smaller circuit than a computer-aided design (CAD) tool can since a designer can create a specific size and placement for each transistor while a CAD tool must use a fixed library of parts. However there are also disadvantages to full custom circuits, including a long design time, especially for circuits with complex logic. In addition, when dealing with the details of connecting thousands of transistors, more effort is generally required to ensure error-free operation. Thus, full custom design is often used for small or regular parts of a circuit where area or timing considerations are especially critical. For example, a memory design has a basic cell that is repeated many times in a RAM IC. A smaller cell will result in a greater memory capacity on the IC, so size is critical. A single cell is simple enough that the advantage of small size outweighs the disadvantage of a longer design time. Control logic for selecting which particular bits to read and write can be created with a faster method, since size is not as important here.

Once a design has been completed and tested, it is sent to a foundry to be fabricated. In general, an IC is created by imprinting a circuit into a wafer of silicon in layers, each of which takes hours to complete. The first layers define a circuit's transistors, followed by the transistor-to-routing contacts, and finally two or more layers of metal wires to connect the transistors.

Each layer requires masks to be designed, which adds to the time and cost involved in the process. In general, metal layers require one mask, contacts require one mask, and transistor layers require at least four masks for CMOS circuits.

Since every part of a full custom circuit is uniquely specified for each circuit, each mask and each layer used in fabricating it must be built after the design has been submitted to the foundry. The process will take a relatively large amount of time since, after receiving a design, many layers must be fabricated. This can take as long as sixteen weeks. The generation of masks and customized layers is a non-recurring engineering (NRE) cost of the fabrication process that can be amortized over the number of chips produced. So for low volumes of production, this is an expensive technology, but in high volumes full custom costs are reasonable. Because of this technology's efficient use of silicon area, full custom is often the cheapest option per chip for a combination of large designs and high volumes of production.

2.2. Standard Cell

When creating a circuit with standard cells, a designer generally first writes a description with a hardware description language (HDL) such as Verilog or VHDL. Then this description is input into a computer program that compiles it into a physical layout [18]. This is accomplished by maintaining a library of *cells* that implement standard functions such as NAND gates and D flip-flops. The program translates the description into a series of these standard functions, then replaces each function with a cell from its library. An advantage to this method is that it has a shorter design time than full custom circuits. Writing a behavioral description takes much less time than drawing an actual circuit. In addition, a computer can then translate this description into an implementation in hours, instead of taking weeks or months to implement it by hand.

However, the cells that are used by the program are standard, which means they were designed to work for a general case. Therefore the computer cannot always make refinements that would improve the speed or size of a particular circuit. A flip-flop, for example, comes in many forms. There are J-K, D, T, and S-R flip-flops. Some designs need them to hold their previous values, some need a reset signal, and some use a two-phase clock. Some need to drive larger loads than others. It would be difficult to create a library that contains every permutation of the flip-flop, so cells are made robust enough to work in diverse designs. A designer is limited to parts included in the library that will work, but may not be exactly what is desired. Logic added to enhance a simple cell's functionality or unwanted logic in a complex cell will make a design bigger than necessary, and a restriction on available gate sizes can increase the time of the critical path.

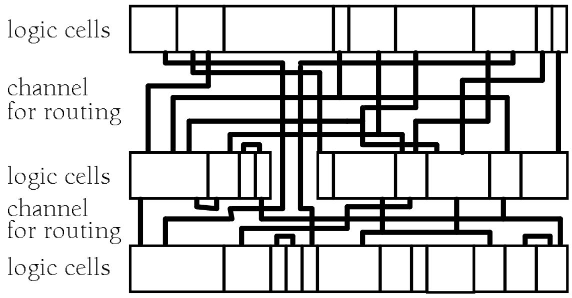

Figure 1. A typical standard cell structure.

Another reason standard cell designs are more limited than full custom designs is standard cells have a fixed underlying structure. Cells are placed in rows on the chip. The channels between each row are used to route signals between the cells (Figure 1). This means that cells in the library need to have a fixed height, with inputs and outputs at the top or bottom edges of the cells. Instead of routing a signal from one cell directly to its neighbor, the routing is required to leave the cell from the top or bottom, then travel along the channel to the neighbor's inputs. This makes the routing more complicated, and also requires that the channels be wider than they would otherwise be. If local routing went directly between cells, space would not have to be left in the channels for it. Also, some space is lost because of the fixed height requirement. Extra height is needed to accommodate the internal routing of complicated structures and of cells with many inputs. Cells generally cannot fill this internal routing space with logic, so in simple cells this extra height is wasted.

The finished design is completely designer-specified, so again the fabrication process needs

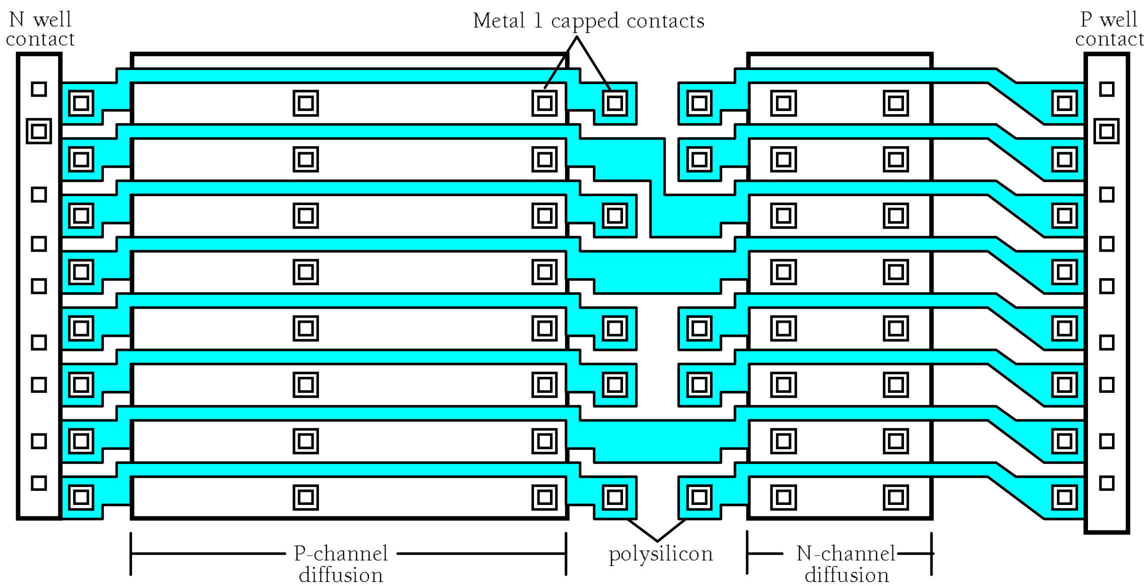

Figure 2. Typical gate array cell structure [2]. Crossing polysilicon over diffusion forms a gate and channel of a transistor. Power rails run vertically over the diffusions, so cells can be stacked vertically.

to begin from a blank silicon wafer, and every mask must be custom made for the design. This process saves on design time, but the fabrication time and costs are still large. For this reason, standard cell design is often used in conjunction with full custom design. Most of the circuit is created with standard cells, while the most critical pieces are designed by hand.

2.3. Gate Array and Sea-of-Gates

A gate array design uses a pre-defined pattern of transistors and transistor contacts [10]. This way, the transistor pattern can be fabricated before the design is created, and only the metallization step remains when the design is finished. A designer creates a particular circuit by connecting these pre-set contacts in unique ways. Gate array circuit elements are arranged in the same way as standard cells. Transistors and their contacts are grouped into cells (Figure 2), and these cells are laid out in rows. A typical die structure is shown in Figure 3. The pre-defined contacts prevent routing from overlapping any transistors without creating a connection, so transistor-free channels are left between rows of transistor patterns. Sea-of-gates designs [10], unlike gate arrays, do not have prefabricated transistor contacts. Because of this, channels are not necessary, and a pattern of transistors can cover an entire chip. A sea-of-gates design lays its routing over unused transistors. If the ratio of logic to routing in a design does not match the ratio in a gate array, the silicon will not be efficiently used. Sea-of-gates designs avoid this problem.

Gate arrays and sea-of gates designs are used to cut down on the fabrication time and cost.

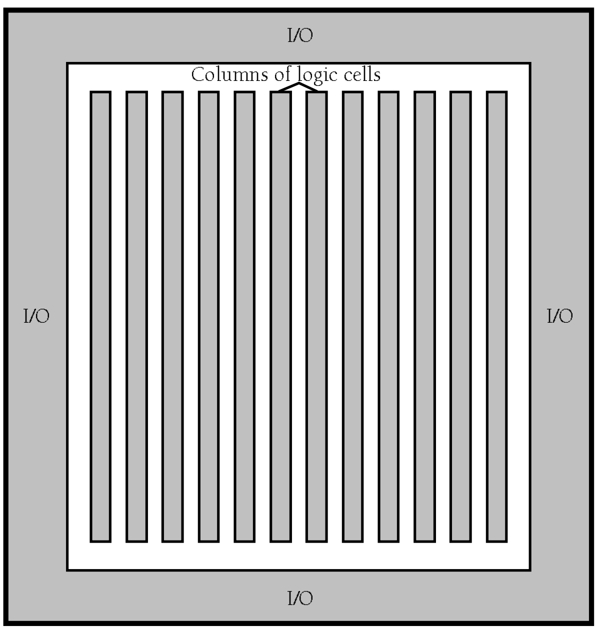

Figure 3. Typical gate array die pattern. Space between columns is for routing. For more details see [2].

A pattern of transistors is predetermined in this type of design, so the cost of building the corresponding masks can be amortized over all gate array or sea-of-gates designs. Thus all designers using the same type of gate array share the NRE costs of the bulk of processing. The amortized cost allows a smaller number of IC's to be produced at a reasonable price per chip. Also, foundries can process this transistor pattern onto a chip and stockpile these partially complete circuits. Then the foundries have a head start on the fabrication when they receive a design. A sea-of-gates design only needs masks for the contacts and the metal wires. A gate array only needs the mask for the wiring. This significantly cuts down the fabrication time for a design. If speed and size are not critical, a gate array can be used to shorten the time between the finished design and the working product.

Unfortunately, designs using gate arrays or sea-of-gates tend to use more silicon and have worse performance than either standard cell or full custom designs. Transistors must be placed on the chips before their function is known, so limited optimization of size or placement can occur. Designs can be placed to try to minimize the routing, but the transistors are fixed in some structure such as in columns and cannot be moved to help this minimization. For example if more p-type transistors are needed than n-type transistors in one area of a design, n-type transistors cannot be replaced by p-type transistors. Extra n-type transistors in the area are just wasted. This limits the maximum speed and minimum size of the final design. In this technology designs are often created with automatic placement and routing tools in the same way as standard cell designs.

2.4. FPGA

Extending the gate array idea further, a field programmable gate array (FPGA) is completely fabricated before any design has been created. This way, when a designer needs to implement a circuit, there is no waiting for a foundry. A design can be implemented on an FPGA almost instantly by just loading a configuration.

In general this is enabled by tying circuit elements to SRAM that is set when configuring the FPGA. Paths and logic in an FPGA are controlled by this memory. Logic is divided into groups of small lookup tables (LUTs) distributed throughout the chip. Outputs of these LUTs are determined by the SRAM. Flip-flops are also arranged throughout an FPGA, generally near a lookup table's output. Whether or not they are used is controlled by the SRAM configuration. Connections among an FPGA's routing resources are made with transistors that are turned on or off also by appropriately setting an SRAM bit.

An n-input LUT is commonly implemented as a 2^nx1 multiplexer, since a multiplexer can implement any n-input function where n is the number of selection bits. Each combination of

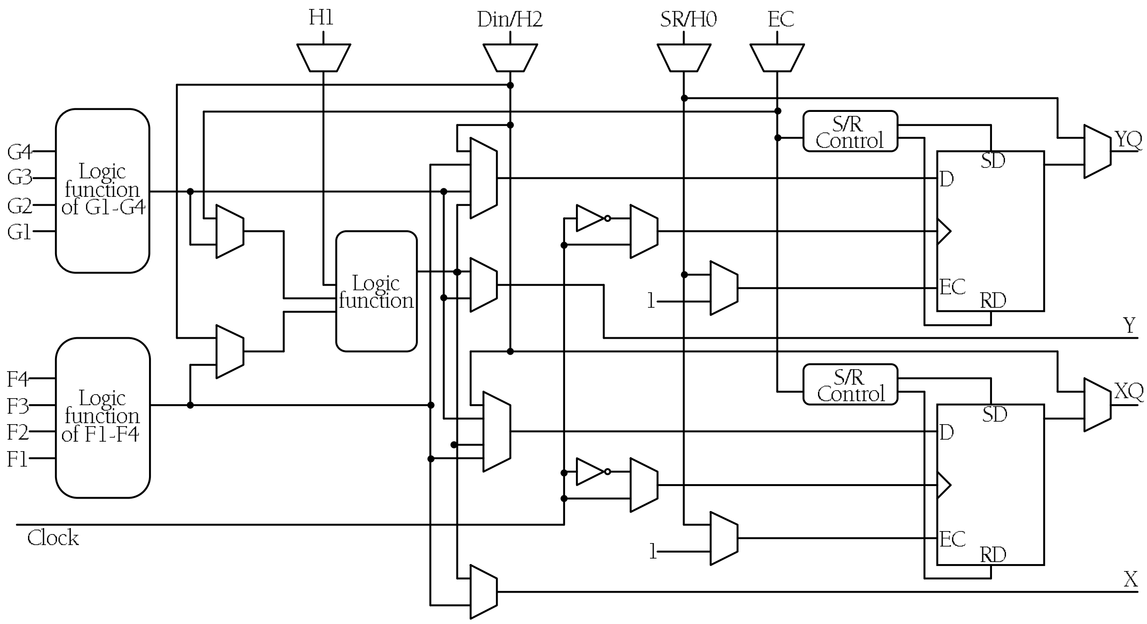

Figure 4. The Xilinx Combinational Logic Block (CLB) [15]. Multiplexers and logic functions are controlled by an SRAM configuration.

the n selection bits selects one of 2^n data inputs, and passes it to the output. To implement a function $f(n)$ on a multiplexer, the n selector bits are used as the function's inputs. Each multiplexer data input, D, is fixed at the value of $f(S)$, where S is the combination of selection bits that selects data input D. For example, to create an AND gate from a multiplexer, the multiplexer inputs corresponding to 00, 01, 10, and 11 are set to 0, 0, 0, and 1 respectively.

In an LUT, each multiplexer input is connected to an SRAM bit, so its value is fixed when the configuration to the FPGA is loaded. The n control lines of the multiplexer are a lookup table's inputs. So any n-input function can be placed in an FPGA by setting 2^n configuration bits. However, these gates are larger and slower than custom gates, which means that an FPGA cannot implement as dense a circuit as the other technologies. Also, 2^n configuration bits is a lot of memory for larger n's, so n is generally kept small.

In the Xilinx 4000 series FPGAs used in this study, n is 4. The outputs from groups of two LUTs are joined by a third multiplexer, so the Xilinx LUTs can be used in combination to create any 5-input function by using the same four inputs for each LUT, and a fifth one to control this extra multiplexer (see Figure 4) [15]. Some larger input functions can also be created by using different inputs into the two LUTs, but in general larger functions are broken into several groups four or five input functions and thus occupy several groups of LUTs.

The routing in most FPGAs is also controlled by the SRAM configuration. Most of the wiring of an FPGA either runs horizontally or vertically in a straight line, until another wire is encountered. At a junction there is a configurable connector that controls the signal flow. In the Xilinx FPGAs most junctions consist of four wires, and the connector involves six transistors that control the connections among any of the wires. Each transistor's gate is connected to the configurable SRAM. The interconnections of these transistors, shown in Figure 5, allows any combination of connections at the junction, including connections joining more than two wires. For example, if a transistor connecting the north and south wires is turned on, and one connecting the east wire to the west wire is on, signals will continue straight through the connector. Again, these configurable connectors take more space and are slower than permanent connections.

As soon as a design has been completed, it can be converted to an SRAM configuration, loaded into the FPGA, and used. There is no fabrication time involved, and the design time is comparable to the gate array and standard cell design times. Because they do not have the fabrication time and costs involved in custom and semi-custom circuits, FPGAs are often used for small, low volume circuits. For higher volumes the inefficient use of silicon can make FPGAs more expensive than other technologies. One significant aspect of FPGAs is their volatility, a side effect of having a design controlled by SRAM. The design is lost every time an FPGA is disconnected from its power source, and needs to be reloaded whenever the device is turned on. This volatility can be beneficial and allows FPGAs to perform tasks that the other technologies cannot match, but is a distinct disadvantage in some ASIC designs.

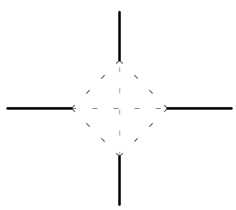

Figure 5. Programmable Interconnect [15]. Each dotted line represents a pass transistor controlled by SRAM.