How does an FPGA work?

6 min · Justin Rajewski | Suggest Changes

This tutorial will cover how an FPGA can implement so many different logic circuits simply by being reprogrammed.

When explaining what an FPGA is to someone, I commonly tell them that there are basically a bunch of logic gates in the FPGA that can be connected however you want to create your circuit. While this is a good way to explain it to someone with little prior knowledge on digital design, it's not actually how it works. An FPGA has three main elements, **L**ook-**U**p **T**ables (**LUT**), flip-flops, and the routing matrix, that all work together to create a very flexible device.

Look-Up Tables

Look-up tables are how your logic actually gets implemented. A LUT consists of some number of inputs and one output. What makes a LUT powerful is that you can **program** what the output should be for every single possible input.

A LUT consists of a block of RAM that is indexed by the LUT's inputs. The output of the LUT is whatever value is in the indexed location in it's RAM.

As an example let's look at a 2-input LUT.

Since the RAM in the LUT can be set to anything, a 2 input LUT can become any logic gate!

For example, if we want to implement an AND gate the contents of the RAM would look like this.

Address (In[1:0])	Value (Out)
00	0
01	0
10	0
11	1

It's important to understand that the column **Value (Out)** could be set to anything! It doesn't have to model a single logic gate. If the value for address 01 was a 1, then the LUT would still perform as expected but the equivalent logic circuit would require more than one gate to implement. This is why the metric of equivalent gate count for an FPGA is a confusing and poor metric! It is tricky to specify how complicated of a circuit you can implement in a given FPGA because of all the variables.

LUTs in the Mojo

In the Spartan 6 used by the Mojo, each LUT is a 6-input LUT. However, it isn't a true 6-input LUT but rather two 5-input LUTs connected together by a multiplexer (MUX).

The reason the LUT is designed this way is because it can either be used as a single 6-input LUT, or two 5-input LUTs. The only restriction is that both 5-input LUTs must share the same inputs. When it is configured as two 5-input LUTs, **In[5]** is set to 0.

Flip-flops and Slices

Each LUT's output can be optionally connected to a flip-flop. Groups of LUTs and flip-flops are called **slices**. In the Spartan 6 used by the Mojo, a slice has 4 LUT6 and 8 flip-flops. These flip-flops are typically configurable allowing the type of reset (asynchronous vs synchronous) and the reset level (high vs low) to be specified. Some of the flip-flops can actually be configured as latches instead of flip-flops, although latches typically aren't good practice to use as they can lead to timing problems.

The FPGA used by the Mojo has 1,430 slices in it for a total of 5,720 LUTs!

All slices are not created equal, however. In the Mojo's FPGA there are three types of slices, **SLICEX**, **SLICEL**, and **SLICEM**. The SLICEX is the most basic type of slice and just consists of the four LUT6's and the eight flip-flops. A SLICEL is the same as a SLICEX except it contains extra hardware for a ripple-carry chain used in arithmetic circuits. These help speed up circuits like addition and multiplication. A SLICEM is the same as a SLICEL except the LUTs can be used as 64bits of RAM or a shift register up to 32bits long. The break down of the slices is roughly SLICEX 50%, SLICEL 25% and SLICEM 25%.

The Routing Matrix

The next size block in the FPGA is the **C**omplex **L**ogic **B**lock (**CLB**) and each CLB consists of two slices. Each CLB connects to a **switch matrix** that is responsible for connecting the CLB to the rest of the FPGA. The switch matrix can connect the inputs and outputs of the CLB to the **general routing matrix** or to each other. That way the output from one LUT can feed into the input of another LUT without having to travel far.

The routing resources in an FPGA are pretty complicated, but they are essentially a bunch of multiplexers and wires that are used to define what CLBs and other FPGA resources are connected to each other. These connections are again defined in RAM which is why the FPGA must be reconfigured every time the power is cycled (in the Mojo this is taken care of by the AVR).

There are also special routing resources available on the FPGA. The most notable are the clock routing resources. When you have a clock being used in your design, it is very important that the clock signal be distributed as evenly as possible throughout the FPGA so all the flip-flops will flip at roughly the same time. If you were to try and use the general routing resources for this, the clock signal would have large propagation delays from traveling through all the multiplexers. To solve this problem there are global and local routing resources dedicated to clocks. These are basically wires that connect through the entire chip (for global) or sections of the chip (for local) with very little propagation delay. Only inputs from certain pins on the FPGA

are allowed to drive a signal on the global clock routing resources. These inputs are labeled GCLK in the Mojo schematic.

All of these resources are used in an FPGA to make them very flexible slow things down. This is why you will never be able to clock an FPGA at speeds comparable to a dedicated chip. An ASIC design can reach speeds faster than 4GHz, while an FPGA is very fast if it's running at 450MHz. This is also why FPGAs consume considerably more power than their ASIC counterparts. An FPGA will require an order of magnitude more power to run than an 8bit microcontroller, like an AVR. However, they still have the huge advantage of being low-cost for small runs (or hobbyists) and re-configurable virtually unlimited times.

More Info

If you want to know more details about the FPGA used in the Mojo, this document has everything.

« PREV
What is an FPGA?

NEXT »
Encodings