What is an FPGA?

4 min · Justin Rajewski | Suggest Changes

▶ Table of Contents

So what exactly is an FPGA? You may have heard the term thrown around, or maybe you have no idea what I'm talking about. Either way, FPGAs (**F**ield **P**rogrammable **G**ate **A**rrays) are amazing devices that now allow the average person to create their very own digital circuits. The cost has come down enough that you don't have to be a huge company to get your hands dirty.

You can think of an FPGA as a blank slate. By itself an FPGA does nothing. It is up to you (the designer) to create a configuration file, often called a bit file, for the FPGA. Once loaded the FPGA will behave like the digital circuit you designed!

One of the reasons FPGAs are so awesome is that unlike an ASIC (**A**pplication **S**pecific **I**ntegrated **C**ircuit) the circuit design is not set and you can configure an FPGA as many times as you like! Creating an ASIC also costs potentially millions of dollars and takes weeks or months to create. That's not exactly hobbyist friendly.

FPGA vs Microcontroller

When I first learned about FPGAs, all I really knew about before was microcontrollers. So first it is important to understand that they are *very* different devices. With a microcontroller, like an Arduino, the chip is already designed for you. You simply write some software, usually in C or C++, and compile it to a hex file that you load onto the microcontroller. The microcontroller stores the program in flash memory and will store it until it is erased or replaced. With microcontrollers you have control over the **software**.

FPGAs are different. *You* are the one designing the circuit. There is no processor to run software on, at least until you design one! You can configure an FPGA to be something as simple as an and gate, or something as complex as a multi-core processor. To create your design, you write some HDL (**H**ardware **D**escription **L**anguage). The two most popular HDLs are Verilog and VHDL. You then *synthesize* your HDL into a bit file which you can use to configure the FPGA. A slight downside to FPGAs is that they store their configuration in RAM, not flash, meaning that once they lose power they lose their configuration. They must be configured every time power is applied.

That is not as bad as it seems as there are flash chips you can use that will automatically configure the stored bit file on power up. There are also some development boards which don't require a programmer at all and will configure the FPGA at startup.

With FPGAs you have control over the **hardware**.

The Possibilities

With a typical microprocessor, you have dedicated pins for specific features. For example there will be only two pins on some microprocessors that are used as a serial port. If you want more than one serial port, or you want to use some other pins, your only solution besides getting a different chip is to use software to emulate a serial port. That works fine except you are wasting valuable processor time with the very basic task of sending out bits. If you want to emulate more than one port then you end up using all your processor time.

With an FPGA you are able to create the actual circuit, so it is up to you to decide what pins the serial port connects to. That also means you can create as many serial ports as you want. The only limitations you really have are the number of physical I/O pins and the size of the FPGA.

Just like microcontrollers that have a set amount of memory for your program, FPGAs can only emulate a circuit so large.

One of the very interesting things about FPGAs is that while you are designing the hardware, you can design the hardware to be a processor that you then can write software for! In fact, companies that design digital circuits, like Intel or nVidia, often use FPGAs to prototype their chips before creating them.

« PREV
Combinational Logic

NEXT »
How does an FPGA work?