FPGA DESIGN IN SOFTWARE ENVIRONMENT

Ruzieva Sokhibjamol Nuralievna
soxibjamolruziyeva@gmail.com
Sotvoldiyeva Sabrina Burxon qizi
sotvoldiyevasabrina@gmail.com
Khalikova Madina Shukhratovna
xoliqovamadina78@gmail.com

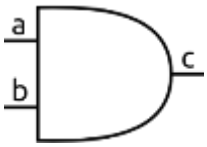
Annotation: *FPGA means "Field Programmable Gate Array" and is a huge array of doors that can be programmed and restored at anytime, anywhere. Many users still do not understand what FPGA is. "Set of large doors" - {textend} a simplified description of the model. Some FPGAs have built-in hard blocks: memory controllers, high-speed communication interfaces, and PCIe endpoints. Inside the FPGA there are many doors that can be freely connected to each other. The principle of operation is more or less similar to connecting individual microcircuits of logical elements. FPGAs are produced by the world's leading companies Xilinx, Altera and Microsemi. When software engineers and hardware engineers talk about FPGA, often the two worlds collide, each with different thoughts, skills and expectations. Until recently, FPGA programming was basically a hardware discipline performed by specially trained people. FPGA design's design productivity was much lower, and over time, the hardware engineer came up with several turn bits, the software engineer could implement a new large-scale cloud service program running in a large data center. Note that we were " very low!"Like the widespread success of the processor, with the development of expressive languages and related compilers, this is now happening for FPGA. For decades, FPGA programming has been implemented at very low abstraction levels (comparable to assembly language), but in recent years, several compilers have emerged that bring the FPGA algorithm to a much higher quality than human handwork can do. Currently, the software and hardware worlds are strongly united. Software is parallel, which has a long tradition of hardware design and vice versa, gets hardware designed from more and more expressive languages.*

Keywords: *FPGA, software environment, digital circuit, logic gates, multiplexer, design, electronics, alternative*

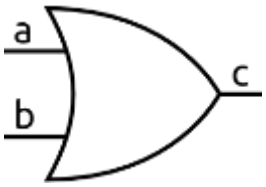
INTRODUCTION

Although we are talking about the use of FPGA to create digital circuits, circuits are usually not drawn to design their architecture. If we draw a scheme, the size and complexity of the schemes that the FPGA can contain will be very inconvenient. Instead, we can describe the action of the circuit we want, and the instruments use this description to create a circuit that matches this behavior. It's like programming in a way, because we're just entering text. However, when we create hardware, the fundamental implementation is significantly different. The disadvantage of FPGA is that they can only create digital circuits. Some newer FPGAs have built-in a/D converters, but they also convert the analog input signal directly to a digital one. In electronics, circuits that convert continuous voltage

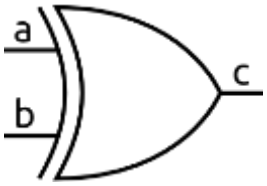
values to discrete and zero are called digital. For higher-order architecture, the actual voltages and limits are not as important, but within FPGA, often 0 V is defined as 0, while 1.2 V is defined as 1. If the actual voltage, say, is 0.8 v., which is close to 1.2 V to count as 1, and everything else works the same. The circuit circuits of digital contacts turn the voltage to an extreme level, which makes them incredibly resistant to noise and other real-world effects. Also, the concept of a digital device allows us to model complex behavior in a circuit without working on a low-level architecture. We can work in an ideal world. All the subtleties are hidden in the architecture of ordinary building blocks that we use. These building blocks are logical doors. There are several logic gates, the most common of which are AND, OR, choir, and NOT. Each of them receives digital signals at the input, performs its own logical function and outputs a digital value. The AND gate takes two input values and only outputs 1 if the first and second input values are 1. If at least one of the input values is 0, the output is 0. AND gate icon looks like this:



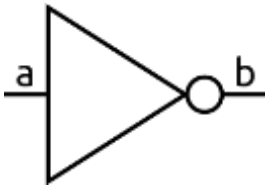
The OR gate takes two input values and outputs 1 if the first or second value is 1. If both input values are 0, the output value is 0. Here is the sign of the OR gate:



A chorus Gateway is similar to an or door, but it only releases 1 when the first or second input value is 1, but not when both are 1. It can also be called output 1 when the input values are different. X in XOR means exclusive. Here is the valve icon:

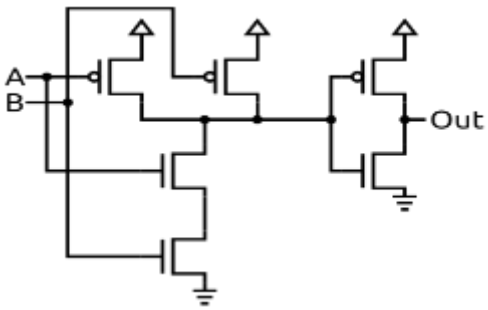


Not Gate is the simplest. It has only one input and simply outputs the opposite value. That is, 1 becomes 0 and 0 becomes 1.



There are variations of these main doors called NAND, NOR and XNOR. These are simple versions of standard doors, but with reverse outputs.

The AND gate, like all other logic gates, can be made from transistors. The figure below shows how the AND gate can be implemented. This scheme uses NMOS and PMOS MOSFETs. This type of architecture is called CMOS (additive metal-oxide semiconductor, CMOS, additive metal-oxide-semiconductor structure), which is used in most modern circuits.

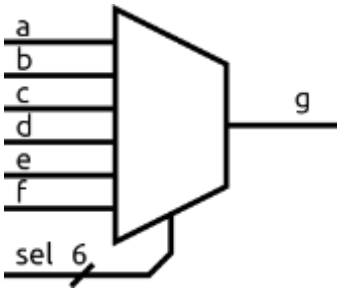


Note that the scheme shown above is actually NAND gate followed by not Gate. This is because CMOS circuits change the output value.

Now that we have basic building blocks from transistors to logic gates, we can build them on something more useful. Only with the help of logical doors can any digital Circuit be described. However, there are many high-level functions that are frequently used and have their own characters, such as in binary mathematics (adjuncts, multipliers, etc.

We consider one of the main building blocks of the FPGA, the Multiplexor.

The multiplexer selects a single input value from the set based on the selected input value. Here is its symbol:



/ On the flood line means that its width is 6bit. The number of inputs can vary, but the multiplexer always has only one output.

When creating your digital scheme, you have two main options. First, you can create it yourself from discrete logic. It takes a lot of time, more costs, and if something needs to be changed, then the flexibility will be much lower.

A second, more realistic alternative is to design the circuit directly in silicone. So you create a chain very quickly and efficiently, but with zero flexibility and for a ton of money. The large initial cost of special chips is associated with tools and preparation. However, the additional cost per chip would be less than for individual FPGAs. However, if you do not make tens of thousands of chips, it will still be more expensive. But even with large parties, sometimes it makes no sense to lock your architecture in Silicon. When using FPGA, you can change it at any time and at no additional cost. It should be borne in mind that FPGA is only one tool. The hammer is great for driving nails, but terrible when driving screws. And it will also be very uncomfortable to hit the nail with a screwdriver. It can be difficult to create your own schemes, and sometimes it is worth wondering if there is a better solution. There are many capable processors with tons of external devices that can handle most of your tasks. Performing certain tasks, such as receiving and sending data over WiFi, can be daunting with FPGA, but can easily be done with a few dollars of microcontroller, such as ESP8266.

REFERENCES:

1. The Design Warrior's Guide to FPGAs Clive "Max" Maxfield British Library Cataloguing-in-Publication Data
2. Dirk Koch ● Frank Hannig ● Daniel Ziener Editors FPGAs for Software Programmers Springer International Publishing Switzerland 2016
3. Cabrera, A.M., Young, A.R., Vetter, J.S.: Design and analysis of cxl performance models for tightly-coupled heterogeneous computing. In: Proceedings of the 1st International Workshop on Extreme Heterogeneity Solutions, ExHET '22. Association for Computing Machinery, New York, NY, USA (2022). DOI 10.1145/3529336.3530817. URL https://doi.org/10.1145/3529336.3530817
4. Chacko, J., Sahin, C., Nguyen, D., Pfeil, D., Kandasamy, N., Dandekar, K.: Fpga-based latency-insensitive ofdm pipeline for wireless research. In: 2014 IEEE high performance extreme computing conference (HPEC), pp. 1–6. IEEE (2014)
5. Introduction to FPGA Design with Vivado High-Level Synthesis UG998 (v1.0) July 2, 2013