The Worst FPGA Ever Built

Designing my own FPGA using Discrete Electronics on Circuit Boards

Simon Burkhardt

June, 2023

abstract

Acknowledgements

I would like to express my deepest appreciation to my professor and supervisor Michael Pichler for his invaluable incentive to understand the topic and for sparking my joy in FPGA engineering. Additionally, this endeavor would not have been possible without the generous support from the Innosuisse Agency, who partially financed this research.

Special thanks goes to AnaPico AG for their funding and their valuable technical expertise. I am also grateful to my colleagues at FHNW for their expertise and moral support that impacted and inspired me.

Contents

1 Introduction **1**

1.1 Diving into FPGA 2

1.2 Ben Eaters 8 Bit CPU 3

1.3 The End Goal of this Project 4

2 A Brief History of FPGAs **5**

3 FPGA Design Concepts & Practices **6**

3.1 Configurable Logic Block 7

3.1.1 Look Up Table (LUT) 8

3.1.2 Fast Carry Chain 9

4 References **11**

1 Introduction

1.1 Diving into FPGA

1.2 Ben Eaters 8 Bit CPU

1.3 The End Goal of this Project

With the expected cost being way beyond 1'000 USD all while expecting a performance not even close to 20 year old microcontrollers, it is hard to justify this project. I admit that designing such a machinery will not deliver any reasonable performance let alone create any innovation. Yet, it will teach me (and hopefully you) about the inner workings of an FPGA and showcase all the difficulties that designers of modern FPGA chips face.

What I aim to achieve is to build an FPGA with enough complexity to run a few relevant circuits. Thereby teaching myself the basics of FPGA architecture while maintaining a functional, reconfigurable hardware. The entire FPGA should be able to run basic circuits such as a 4-bit counter or a 4-bit BCD to 7-segment decoder. Once the project grows further, more specialized logic can be added including block RAM or even DSP slices.

Since I do not plan to open another Pandoras box of synthesis and routing software algorithms, the bitstream is required to be manually managed resulting in a size constraint of around <4kB.

2 A Brief History of FPGAs

3 FPGA Design Concepts & Practices

3.1 Configurable Logic Block

3.1.1 Look Up Table (LUT)

The LUT is the implementation of a truth-table in hardware. Any thinkable logic function with A inputs and O outputs can be realized using a $O \cdot 2^A$ LUT.

Modern implementations of LUTs use between 4 and 6 inputs while having a maximum of 2 outputs. This means that an output variable can depend on up to 6 input variables. If an application requires more input variables, multiple LUT primitives need to be cascaded. Another approach would be to offer LUT primitives with a higher number of inputs (e.g. an 8:1 LUT). This comes at the cost of higher area consumption which is wasted for simpler functions using less than 8 inputs. Research in FPGA chip design suggests that 4 to 6 inputs is a good balance. Xilinx LUTs for example can either be configured as a 6:1 LUT or a 5:2 LUT.

The output of a LUT can be registered with a D-type flip-flop stage. Or the output can be routed to another LUT for additional functionality. The output register therefore is a configuration option in the bitstream.

3.1.2 Fast Carry Chain

LUTs only offer a low input signal count and can rarely add longer than 2x2 bit numbers. This is hardly practical which is why longer additions require multiple LUT primitives. The way an addition works is by starting at the LSB position. If both numbers are a "1" bit, a carry bit is added to the next higher bit. Each additional bit in the input vector length therefore translates to an additional LUT cascaded (in series). This increases the critical path significantly to the point where timing quickly becomes an issue in high speed FPGAs. To circumvent this problem, multiple versions of a (fast) carry chain were developed and are standard in modern CLB slices.

4 References