Diving Deeper into the Z80

An adventure of discovery of how the Z80 functions using logic simulation

Brian L. Little

Table of Contents

Introduction

The first computer I had the Timex Sinclair 1000, the US variant of the ZX81, which had a Z80 CPU. I quickly found out that I could bypass the limitations of Basic by using assembly language. The only limits now were the hardware. Since I learned 6502 assembly in college I quickly picked up on the Z80 assembly. I spent many hours pushing the limits of the Timex Sinclair 1000.

A few years later when I moved onto the PC I was hired as an electrical engineer for a company that made its own Z80 based computer used in quality control for the plastic molding industry. Working with the code rekindled my interest in the Z80. By that time my Timex Sinclair 1000 was long gone but the PC was more than capable of emulating it so I could once again enjoy coding Z80 assembly on the Timex Sinclair 1000.

Around 2010 I was introduced to FPGAs (Field Programmable Gate Arrays). The ability to construct hardware using Hardware Defined Language was intriguing. When I found out that you could also use logic symbols and simulate 74 series TTL chips it made me flash back to the days when I found the article for the Elf 1802 computer. I couldn't afford the 1802 on my lawn mowing profits of my neighbors but my room was littered with logic diagrams for expansion of the long sought after Elf II. I had to get a FPGA development board.

My first project was adapting a ZX81 written in VHDL to my new board. I felt triumphant when I finally got the flashing K in the corner of the screen and was able to play Mazogs on it. After that the next step was to actually design a cpu from scratch, either the 1802 or the Z80. That is where things ended. I didn't have much time to devote to it so from time to time I dabbled with my FPGA over the years, not accomplishing much. I even got another FPGA board with more bells and whistles to motivate me, which didn't work.

Over the last few years through the internet I found sites that had high resolution pictures of the Z80 substrate. Sites that reversed engineered the substrate into schematics and used them to make a Z80 on the FPGA. I also found a net level Z80 simulator and a digital simulator that could analyze a circuit and export hdl code. This sparked my interest again.

One thing I found lacking is more detailed explanation of the logic inside the Z80. So I am writing this to document what I found in my adventure.

So now armed with these resources we are ready to start our adventure by taking the deep dive into the Z80.

The Adventure Begins

The purpose of this book is to take us on an adventure through the inner workings of the Zilog Z80 computer processing unit (cpu) henceforth will be mentioned as the Z80. We will analyze the logic that makes up the Z80. All code related to this guide can be found on GitHub at

Diving Deeper into the Z80

Luckily for us there have been ardent adventurers that have blazed the trail before us. We give great thanks for this as this makes our travels easier. These adventurers are as follows;

- *Visual6502.*

  - Z80 Die shots

  - This site is hosted by a group that has decapped the Z80 and made high resolution shots of the dies which others have used for reverse engineering of the logic. It might be worth a look to see the other cpus and peripherals they have made die shots for.

- *Ken Shirriff's blog*

  - Z80 Registers

  - Z80 16 Bit increment/decrement

  - Z80 ALU

  - Two interesting gates in the Z80

  - Ken reverse engineers critical parts of the Z80 dies. He explains in detail the logic of the parts listed above. He has also reverse engineered various other cpus and shares their highlights as well.

- *Baltazar Studios*

  - Baltazar Studio

  - Goran Devic hosts this site and he has many articles where he has reversed engineer the Z80. The site mentions his Github site that has the A-Z80 a FPGA soft core made up of the logic he reversed engineered from the die. He has broken down the logic into the schematics that we use in this book. He also has coded a Z80 simulator, the Z80 Explorer, that is a netlist simulator. You can watch the die in action and drill into the logic of any part of the die.

Now any adventurer is only as good as his skills and to go on this adventure you should have a basic understanding of digital logic, reading schematics and optionally FPGAs.

We must equip ourselves for the adventure ahead so these items must be added to or gear bag.

- Digital

  - [Digital on GitHub](#)

  - This will be our go to tool to delve into the mysteries of the Z80. You can find the code for the logic diagrams shown in this guide on my GitHub page. This tool will let you simulate the logic flow in the logic diagram, generate truth tables, run test scripts, generate Verilog code with test bench script and VHDL code with test bench script. Their are other options as well for deeper understanding of the logic diagrams.

- Z80 Explorer

  - [Z80 Explorer on GitHub](#)

  - This is an optional item but well worth having if you want to dive into the depths of the Z80. This is a netlist simulator that will run assembly and highlight the areas of the die that is being used. You can also get a netlist diagram of the logic for any selected section.

Our itinerary for our journey will take us first through the fundamental parts of the Z80. The Arithmetic Logic Unit (ALU), Registers, Programmable Logic Array (PLA) and Control Logic sections will be covered in their own sections. We will then construct a simple Z80 based computer in Digital and make a simple operating system. For the truly adventurous we will take the generated Verilog and VHDL code to program a DE0-Nano and DE10-Lite FPGA development board.

The ALU (Arithmetic Logic Unit)

ALU Slice

Registers

PLA (Programmable Logic Array)

Control

Putting it all together