# FPGA Design - the Making of an Intel 8086 Microprocessor with Modern Technology

***Abstract*—The Intel 8086 microprocessor was first introduced in 1978. Since then the semiconductor industry has changed vastly from the old chip manufacturing techniques of the time. Today we can fit thousands of Intel 8086 microprocessors in the same size package with use of modern semiconductor techniques such as the ability to design with 22nm feature size and better yield from improved wafer quality. This paper examines how we can still learn from ancient technology but with a new more modern twist. By utilizing field programmable gate arrays, we can easily implement the same technology from the past and learn about architectures that are still in use today.**

## I. Introduction

It was in the mid 1970s when Intel announced their latest project, the Intel 8086 - a 16-bit microprocessor capable of supporting up to a revolutionary 1 mega*byte* of address space and 64 kilo*bytes* of I/O. Gone were the days of simple computing in only 8-bits of freedom, this was the 70's and 16-bits was here to take over. Along with the increases in accessible memory and larger size ALU computations, Intel introduced a new type of architecture and instruction set known as x86, this new method of computing revolved around the use of registers that stored input/output data which could then have computations performed on them. This improvement has since paved the way for future computing by setting a standard on how to receive data and how data would be processed in a regular clock cycle. The 8086 supported 80 assembly instructions which gave software developers of the time more way to write better code that performed better with the new hardware.

The field-programmable gate array (FPGA) has been around since the 1980's, its purpose was to be able to easily create custom hardware without the need to buy large quantities of logic chips and instead use one chip that could be customized after manufacturing to act as the hardware needed at the time, essentially the perfect prototyping device. The FPGA accomplishes this by using "logic blocks" which is typically a circuit consisting of multiplexers and low level logic gates that can be configured in such a way as to create custom complex logic such as adders or even be used for more simple XOR and NAND gates a basic structure of a logic block can be found in Figure 1. The blocks are most often configured as a matrix with interconnects for inputs, outputs and configuration paths in between, latest improvements in silicon technologies have allowed companies to greatly increase the number of logic blocks on a chip into the hundreds of thousands and beyond.

## II. Implementation Requirements

Through extensive research of various FPGA manufacturers and the different styles of development boards offered from a few different companies, it was determined that an FPGA able to hold more than 9,000 logical elements or LEs would be required to successfully implement the Intel 8086 microprocessor. As an added necessity, it was important to select a development board that would be able to handle the project inputs and outputs such as PS2 keyboard and VGA output in order to spend more time on the study and not building miscellaneous external hardware. From this research, it was determined that the Altera DE0 development board sufficiently met the project needs with over 15,000 LE's, a VGA port, a PS2 port, buttons, LEDs, switches, and USB interface. Another important addition to the board is it's SecureDigital (SD) memory card slot which would allow for external flash memory storage of an operating system.

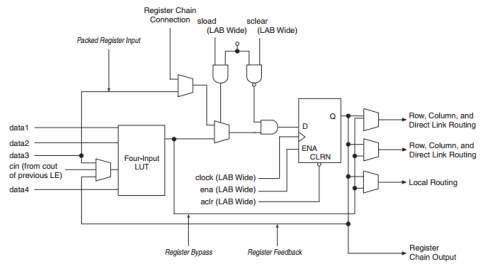


Fig. 1. Cyclone III Device Family LEs in Normal Mode [1]

Other components for the project include; an LCD monitor to visualize the system status, a PS2 keyboard to interact with the operating system, a computer with Linux to compile the processor and write the operating system to the memory card.

Since the study is sponsored by the Altera University Program, an Altera DE0 development board has been provided as well as an Altera DE0-Nano development boards at no cost. This makes the projects overall required budget $0 since the devices come with the necessary software to program the FPGA and all of the other software packages used are open source.

## III. Procedure

The Zet Processor is more than just verilog code for an Intel 8086 microprocessor, it also contains the necessary BIOS and drivers for treating the SD memory card as a mounted hard disc. The preliminary steps for installing the processor first include compiling the BIOS by running it through the Open Watcom compiler which takes the BIOS code written in C and compiles it into x86 Assembly optimized for 16-bit instructions [2]. We then take the BIOS assembly and convert it into hexadecimal by means of a simple file conversion script, this is now our ROM for placing onto the FPGA. The BIOS then gets placed onto the FPGA through running the

DE0_Control_Panel, an example program that comes with the FPGA software, that allows for files to be placed directly onto the on-board flash memory. This is important because when the processor boots, it will automatically look at address 0x00000 for instructions on where to next proceed such as how to mount the operating system and ultimately boot it.

Since the processor runs 16-bit x86 it is necessary to choose a compatible operating system. This leaves a few options including MS-DOS 6.22, FreeDOS 1.1, and Microsoft Windows 3.0. MS-DOS was chosen due to it's small file size and easy to work with command line interface, as an added bonus since the copyright for MS-DOS has since expired it is very easy and convenient to find the source code in various places on the Internet. To load the operating system, it was necessary to load the files exactly as described from the downloaded image. The process was accomplished by running `dd if=./msdos.img of=/dev/sdc` on a Linux computer, this instruction mounts the MS-DOS image byte-by-byte onto the SD memory card which is necessary for when the BIOS looks at a specific memory location to start the operating system.

In order to begin the process of loading the Zet Processor hardware onto the FPGA, it is first important to read the documentation and manuals for the Altera DE0 Development Board and accompanying Quartus II software manual.

The process for loading verilog onto the device is fairly straightforward, that is to compile the verilog code and debug any miscellaneous warnings and compilation errors and then to utilize the on-board programmer to load the code onto the device. When loading the code it is important to take note of the different ways in which the code can be loaded. If the code is loaded through the Joint Tag Action Group (JTAG) interface, it is important to note that this only temporarily loads the hardware and all progress will be lost after powering down the device. This feature is due to the fact that JTAG is made for testing and only loads values directly into the flip-flops and accompanying hardware but does not save this setup data to the flash memory. Since this method only sets the hardware, it can load the hardware almost instantaneously. The other method for loading hardware is known as Active Serial programming (AS), this method requires that the FPGA device is placed into programming mode which can be done by flipping a switch placed on the development board. AS places the FPGA configuration data into FLASH memory which is read into the device at power up.

## IV. Results

## V. Problems Faced & Troubleshooting

The number one challenge in completing this study was a lack of full documentation and user base for the Zet Processor. It became evident very early on in research that this project would require extensive digging through the source code and custom modifications to get it to work with the Cyclone III device. The last update for the project was over a year ago and their online forums provided little resource with many of the problems faced when trying to implement the processor. It was also difficult to determine which version of the processor was relevant to the documentation so it was assumed that most of what was found was obsolete or not correct.

| Flow Summary | |
|---|---|
| Flow Status | Successful - Fri Mar 22 23:52:26 2013 |
| Quartus II 64-Bit Version | 12.1 Build 243 01/31/2013 SP 1 SJ Web Edition |
| Revision Name | kotku |
| Top-level Entity Name | kotku |
| Family | Cyclone III |
| Device | EP3C16F484C6 |
| Timing Models | Final |
| Total logic elements | 8,434 / 15,408 ( 55 % ) |
| Total combinational functions | 7,858 / 15,408 ( 51 % ) |
| Dedicated logic registers | 2,952 / 15,408 ( 19 % ) |
| Total registers | 2976 |
| Total pins | 154 / 347 ( 44 % ) |
| Total virtual pins | 0 |
| Total memory bits | 74,547 / 516,096 ( 14 % ) |
| Embedded Multiplier 9-bit elements | 2 / 112 ( 2 % ) |
| Total PLLs | 1 / 4 ( 25 % ) |

Fig. 2. Compilation results of Zet Processor from Quartus II software.

Issues in implementing the processor involved problems with the BIOS not compiling due to Open Watcom not being correctly added to the system path, once this was resolved the BIOS compiled correctly. The installation document also does not make clear that it is necessary to convert the BIOS into hexadecimal, when this was done and added into the ROM the processor behaved correctly.

There were also some stability issues with the SD memory, the project site lists a way to mount MS-DOS to the SD card through a program known as Winimage but this failed multiple times and always left the processor hung up at different parts of the boot sequence. This was corrected by instead loading the image byte-by-byte in Linux as mentioned previously, which was not as simple as it sounds since there were also issues with the SD card not being formated properly. It was finally fixed by formating the card with all zeros and then the operating system began to work properly.

Lastly, once MS-DOS was finally operational after quite a few hours of debugging it was powered down for the evening and then turned back on in the morning, the result was an error stating that the SD card had become corrupt. This was odd due to the fact that it was operational not but a few hours beforehand. After power cycling the device a few times and with feelings of great defeat that the processor was still not functional, a last ditch effort of removing the SD card and blowing air into the slot and placing the card back into the slot was attempted. The device was then powered back on and worked flawlessly.

## VI. Conclusion & Trends in Industry

The topic of this study on FPGA design and x86 archeticure was of great imp


## Acknowledgment

The author would like to thank the Altera University Program [3] for providing development boards and necessary software for work on this research.

## APPENDIX

### A. Full specifications for Altera DE0 development board:

- FPGA
  - Cyclone III 3C16 FPGA
  - 15,408 LEs
  - 56 M9K Embedded Memory Blocks
  - 504K total RAM bits
  - 56 embedded multipliers
  - 4 PLLs
  - 346 user I/O pins
  - FineLine BGA 484-pin package
- Memory
  - SDRAM
    - One 8-Mbyte Single Data Rate Synchronous Dynamic RAM memory chip
  - Flash memory
    - 4-Mbyte NOR Flash memory
    - Support Byte (8-bits)/Word (16-bits) mode
  - SD card socket
    - Provides both SPI and SD 1-bit mode SD Card access
- Interface
  - Built-in USB Blaster circuit
    - On-board USB Blaster for programming
    - Using the Altera EPM240 CPLD
  - Altera Serial Configuration device
    - Altera EPCS4 serial EEPROM chip
  - Pushbutton switches
    - 3 pushbutton switches
  - Slide switches
    - 10 Slide switches
  - General User Interfaces
    - 10 Green color LEDs
    - 4 seven-segment displays
  - Clock inputs
    - 50-MHz oscillator
  - VGA output
    - Uses a 4-bit resistor-network DAC
    - With 15-pin high-density D-sub connector
    - Supports up to 1280x1024 at 60-Hz refresh rate
  - Serial ports
    - One RS-232 port (Without DB-9 serial connector)
    - One PS/2 port
  - Two 40-pin expansion headers
    - 72 Cyclone III I/O pins, as well as 8 power and ground lines, are brought out to two 40-pin expansion connectors
    - 40-pin header is designed to accept a standard 40-pin ribbon cable used for IDE hard drives

### B. Available x86 Instructions on the Zet Processor [?]:

**Data transfer instructions**
mov, push/pop, in/out, lahf/sahf, lds/lea/les, pushf/popf, xchg, xlat

**Arithmetic instructions**
aaa/aas, aam, aad, daa/das, cbw/cwd, inc, dec, add/adc, sub/sbb, mul/imul, div/idiv, neg, cmp

**Bitwise handling instructions**
and/or, not, rcl, rcr, rol, ror, sal/shl, sar, shr, test, xor

**Control transfer instructions**
call, ja/jnbe, jae/jnb/jnc, jb/jnae/jc, jbe/jna, jcxz, je/jz, jg/jnle, jge/jnl, jl/jnge, jle/jng, jne/jnz, jno, jnp/jpo, jns, jmp, jo, jp/jpe, js, loop, loope/loopz, loopne/loopnz, ret

**String handling instructions**
cmpsb/cmpsw, lodsb/lodswm, movsb/movsw, rep (pref), repe/repz (pref), repne/repnz (pref), scasb/scasw, stosb/stosw

**Interrupt instructions**
int, into, iret

**Microprocessor control instructions**
clc, cld, cli, cmc, hlt, nop, stc, std

## REFERENCES


[1] Altera Corporation. (2012, Aug.) Cyclone III device handbook. [Online]. Available: http://www.altera.com/literature/hb/cyc3/cyclone3_handbook.pdf

[2] Open Watcom. (2010, Jun.) Welcome to Open Watcom. [Online]. Available: http://www.openwatcom.org/index.php/Main_Page

[3] Altera Corporation. (2012, Aug.) Altera University Program - Learning Through Innovation. [Online]. Available: http://www.altera.com/education/univ/unv-index.html