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 bit 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

It was determined that an FPGA capable of at least 256 bytes of internal ROM and more than 9,000 logical elements or LE' s. As an added convenience, it was important to select a board that would be able to handle my project inputs and outputs such as PS2 keyboard and VGA output in order to spend more time on the study and not building miscellaneous hardware. From this research, it was determined that the Altera DE0 development board sufficiently 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 SD card slot which would be used to store the MS-DOS files.

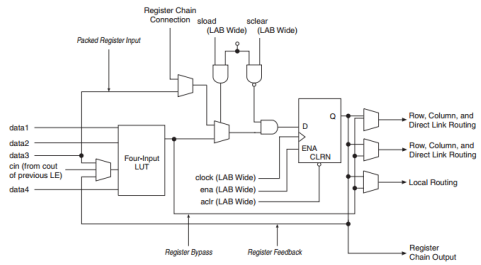

Fig. 1. Cyclone III Device Family LEs in Normal Mode [1]

This study has been sponsored by the Altera University Program in which they have provided an Altera DE0 development board as well as an Altera DE0-Nano development boards at no cost. This makes the projects overall required budget $0 since the sponsorship includes the relevant packages for creating hardware on the chip.

III. Procedure

In order to begin the process of loading hardware onto the FPGA, it is first important to read the necessary 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 miscilanious 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) input, 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 meant for testing and only loads values directly into the flipflops 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

VI. Conclusion & Trends in Industry

Acknowledgment

The author would like to thank the Altera University Program [2] 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] ——. (2012, Aug.) Altera University Program - Learning Through Innovation. [Online]. Available: http://www.altera.com/education/univ/unv-index.html