CR-16 Chip Design

CS6710

Anh Luong, Jacob Sanders, and Willis Lutz

University of Utah

***Abstract*—The project is a simple embedded processor. It provides a user with the ability to interact with a program and to control periferals through a UART. A CR-16 CPU is constructed with a simplified instruction set, built in VGA, UART, and PS/2 interfaces. It was created with a speed of 29 MHz. The VGA will display 64 selected characters from the ASCII table and provide it in three colors.**

I. Introduction

The instruction set architecture (ISA) for our processor is based on the National Semiconductor CR-16 micro-controller. Our ISA is a subset of the CR-16 with simplified instruction encoding. The processor specification is based on RISC concepts and can be implemented as a two stage pipeline. The memory used is divided into memory mapping and I/O communication. The complete memory map will be explained in section III. A VGA controller will be implemented into the IC chip providing the an interface of ASCII characters. The VGA will be able to display three colors to the screen. In order for the user to interact with the processor a UART and PS/2 port will be used. The PS/2 will be connected to a keyboard, allowing the user to type characters onto the screen.

The creation of the processor will be designed in Verilog and then synthesized in Cadence. A library of transistor gates was created to provide a base for the Cadence tools to work with. The transistor gates that was created are: BUFFX1, BUFFX4, BUFFX8, D Flip Flop, D Flip Flop with clear enable, INVX1, INVX4, INVX8, MUX2X1, MUX2X2, MUX4X1, NAND2X1, NAND2X2, NAND3X1, NAND3X2, NOR2X1, NOR2X2, NOR3X1, TIEHI, TIELO, XNOR2X1, XNOR2X2, XOR2X1, and XOR2X2. A SRAM memory blocks was created by the school and was provided for the creation of the register file needed in the processor.

Once the base library is created, and op-

timization parameters are given, the Cadence software will take the description of the processor, created in Verilog, and produce a circuit.

II. Central Processing Unit

The simplified CR-16 processor is based on a word size of two bytes, and use eleven bits for addressing. Each address refers to a two byte word, where the high order three address bits are used to specify the section of the memory map, and the remaining bits specify the memory address within its corresponding section of the memory map. All instructions are single word. Most instructions refer to a sixteen entry register file. The operation code is contained in the highest nibble, destination register address follows the next nibble, the remaining byte is used either as an immediate data value for some instructions or split into a four bit operation code extension and a four bit source register address for other instructions. The instruction set includes ANDI, ORI, XORI, ADDI, SUBI, CMPI, MOVI, MULI, LUI, SUBUI, ADDUI, AND, OR, XOR, ADD, SUB, CMP, MOV, MUL, ADDU, SUBU, LOAD, STOR, JAL, Jcond, LSHI, LSH. The following condition codes are set depending on the output of the alu after conditional instructions: EQ, NE, CS, CC, HI, LS, GT, LE, FS, FC, LO, HS, LT, GE, and UC. The processor is a 3 stage processing unit consisting of fetching, decoding, and executing. Write back's can take a cycle due to the possibility of a data hazard.

III. Memory

Our CPU uses several different storage devices including:

- Read-only memory (ROM)
  - Character ROM to store characters provided to display on-screen through the VGA interface
  - 1 Mb (64Kx16), 45ns, Atmel EPROM for our program memory
- Static random-access memory (SRAM)
  - 16x16 SRAM cell designed by Dr. Erik Brunvand used as our register file
  - 4 Mb (256Kx16), low-power, 55ns, Alliance Memory SRAM chip for our memory mapped I/O

A. Character ROM

The character ROM was created in Verilog and then Synthesis was used to create the layout view. The ROM size was 128 x 32 bits and was used in an array. When information is requested from the ROM it will pick from 32 addresses and will send back a 128 bit value.

B. SRAM

The SRAM is a 4 megabyte 256Kx16 memory chip used for the processor to store data

in. This will be a IC chip purchased separately and installed next to the IC chip that is being fabricated.

C. Memory Mapped I/O

The memory mapped I/O was designed to use 65.5k lines of memory. This is broken down into four parts, the two MSB being used. The top part of the memory will be used for the I/O, the middle two is used for the program data, and the bottom portion of the memory is for PROM. The PROM is where the program will be stored.

The I/O portion of the memory will use its two MSB to break it up into four parts. The I/O is broken up into two screen address, PS2, and UART. Because the screen address is using two portions of the I/O block, this allows it to aquire one more bit of data to use. The screen address will use 8192 lines of data.

The three MSB of the UART memory will be broken up into 8 portions. After all the bits are divided up for mapping there will be eight bits left for the UARt to use. The eight portions of the UART that are divided is, 001 data to transmit, 010 is transmitting data, 011 transmit data is ready, 100 is data received, 101 grab data, 110 is receiving , 111 receive error, and 000 is never used.

IV. Interfacing

A. VGA Interface

The VGA is an interface that we have implemented into the design. It is structured around a screen resolution of 640 x 480 pixels. The graphics will consist of characters only and will be displayed in three colors. The 64 characters we used will consist of capital letters, numbers, and alpha-numeric.

Communication to the VGA will be done through A 16 bit address associated with the memory I/O. The memory I/O will use 1536 lines of a 16 bit address. The 16 bits will hold two character values and an RGB value for each character value. The RGB value is a two bit value instead of the normal three. The red bit of the RGB will be set to zero all the time. This was reduce to allow the screen address to hold two character values at the same time. The character value stored in the screen address is a six bit value that will tell you what character it is, based on a matrix.

Once the VGA module has collected the information about the block on the screen, it will collect the needed glyph information from an internal ROM. This ROM is a 128 x 32 bit block of memory that will hold the glyph information in an array. The VGA module will decide what row that the screen is printing and will print the desired glyph data.

V. PS/2 Keyboard Interface

Another feature we implemented was a PS/2 keyboard input port. This enables the user to process PS/2 scan-codes, which are then interpreted and converted to ASCII code. The corresponding characters stored in the character ROM can be displayed on a monitor through the VGA interface. The PS/2 keyboard works as follows: whenever a key is pressed, that key's make code is output on the data out line. This code has nothing to do with the ASCII character code, so the proper translation is needed. When the key is released, a break code is then output on the data out line. The break code is only needed for confirming that a key has been released because most keys (all the keys which were implemented) use the same break code. The following keys have been implemented into our design, meaning, the following keys can be detected, processed, and stored in off-chip SRAM: all upper and lower case letters, numbers 0-9 and their corresponding characters, all numpad numbers and characters, the special characters and left/right shift.

VI. Process timing

As the components of the processor were created a timing of each component was recorded. The list bellow shows the components and their timing associated with them. What is observed is that the Multiplier is the component that requires the most amount of time to complete. If this would be used in the processor then the final speed would be 13 MHz. If the multiplier is not included in the processor the max speed would be 29 MHz.

- Processor 14.67 ns
- Multiplier 40.22 ns
- PS/2 9.66 ns
- VGA 14.75 ns
- UART 14.66 ns
- SRAM 55 ns read cycle

VII. Conclusion

The characteristics of the processor were described in Verilog and tested on an FPGA. Designing the processor took a significant amount of time to construct and test. The most difficult part of this project was the processor design. Our initial proposal was to design an ARM Thumb, however once we began studying the ISA, we realized it was quite difficult and had to settle for the CR-16 ISA. We were also unable to implement interrupts, which reduces the usefulness of our processor. When we initially documented our pinout, we thought we had plenty of pins to fabricate a processor with UART, PS/2, and VGA. Unfortunately the VGA required more pins than expected, and we were unable to add it to our final layout. The overall system schematics and layouts for each

component have been implemented, however, we were unable to complete the final chip assembly. Our plan is to finish the chip assembly before the designs are sent off for fabrication. Overall, we have learned a lot about processor design and have a better understanding of how the Cadence tools work.

ACKNOWLEDGEMENT

The character ROM generator and VGA controller was used from by Dr. Erik Brunvand's power point slides.

The 16x16 SRAM cell designed by Dr. Erik Brunvand used as our register file and came from the memCellsF09.

The VGA controller was originally written by Harvey Wilson, but we modified it to fit our needs.