Multicycle Processor Design in Verilog

John Chargo and Matt Falat

Department of Computer Engineering

Iowa State University

Ames, IA 50012

Email: jchargo@iastate.edu, mfalat@iastate.edu

Keywords. multicycle processor design, central process unit, CPU, instruction set, Verilog

Abstract. The purpose of this project is to design a multicycle central processing unit. (CPU) The processor will be able to handle fifteen different instructions, including R-type, I-type, and J type. This exercise is done to enhance our understanding of processor design, instruction sets, and Verilog code.

Purpose of the Machine

This machine is designed to be able execute a variety of instructions in a multicycle implementation. The multicycle implementation breaks instructions down into multiple steps. Each step is designed to take one clock cycle. It allows each functional block to be used more then once per instruction if they are used on different clock cycles. This implementation has several key advantages over a single cycle implementation. First, it can share modules, allowing the use of fewer hardware components. Instead of multiple arithmetic logic units (ALU's) the multicycle implementation uses only one. Only one memory is used for the data and the instructions also. Breaking complex instructions into steps also allows us to significantly increase the clock cycle because we no longer have to base the clock on the instruction that takes the longest to execute. The multicycle implementation also uses several registers to temporarily hold the output of the previous clock cycle. These include an Instruction register, Memory data register, ALUOut register, etc.

The multicycle machine breaks simple instructions down into a series of steps. These steps typically are the:

- 1. Instruction fetch step
- 2. Instruction decode and Register fetch step
- 3. Execution, memory address computation, or branch completion step
- 4. Memory access or R-type instruction completion step
- 5. Memory read completion step

During the instruction fetch step the multicycle processor fetches instructions from the memory and computes the address of the next instruction, by incrementing the program counter (PC).

During the second step, the Instruction decode and register fetch step, we decode the instruction to figure out what type it is: memory access, R-type, I-type, branch. The format of these individual instructions will be discussed later.

The third step, the Execution, memory address computation, or branch completion step functions in different ways depending on what type of instruction the processor is executing. For a memory access instruction the ALU computes the memory address. An R-type instruction uses this third step to perform the actual arithmetic. This third step is

the last step for branch and jump instructions. It is the step where the next PC address is computed and stored.

The fourth step only takes place in load word, store word, R-type, and I-type instructions. This step is when the load and store word instructions access the memory and use an arithmetic-logical instruction to write its result. Values are either loaded from memory and stored into the memory data register, or loaded from a register and stored back into the memory. This fourth step is the last step for R-type and I-type instructions. For R and I type instructions this is the step where the result from the ALU computation is stored back into the destination register.

Only load instructions need the fifth step to finish up. This is the memory read completion step. In a load instruction the value of the memory data register is stored back into the register file.

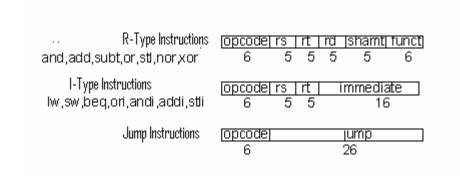
These different steps are all controlled and orchestrated by the "brain" of the multicycle cpu. This "brain" is the controller. The controller is a finite state machine that works with the Opcode to walk the rest of the components through all the different steps, or states. The controller controls when each register is allowed to write and controls which operation the ALU is performing.

Instruction Set

The multicycle CPU can handle a total of 15 instructions. Nine of these were also available on the single-cycle CPU implementation. Of these, there are five R-type instructions: add, subtract, and, or, and set less than. There are three I-Type instructions: load word, store word, and branch on equal. The jump instruction is also supported. Six additional instructions were added to our multicycle design. These are: nor, xor, or immediate, and immediate, add immediate, and store less than immediate. Nor and xor were added as R-type instructions. The additional R-type instructions were implemented by including by augmenting the ALU. They have the same op-code as other R-type instructions, but the function code dictates the operation that is being performed. The four additional I-Type instructions were implemented by wiring the opcode to the control module. This is necessary because there is no function code in the instruction. Code had to be added to the control code to read these specific opcodes. A pair of states were also

added to the state machine to handle these immediate instructions.

Instruction Format



Design Methodology

The most appropriate way to approach any project as large as a multicycle processor design is to break it up into simple steps. To design this project we:

- 1. Read project requirements.
- 2. Discussed what we knew, and what we needed to know.
- 3. Decide on the implementation we wanted to work with.
- 4. Discuss the modifications of the implementation we want to implement
- 5. Clearly define the goals of the project.
- 6. Develop a timetable and schedule to work with.
- 7. Break the large CPU into individual component modules that can be coded separately and assign these modules to the team.
- 8. Build individual modules.
- 9. Perform simple debugging of individual modules.
- 10. Integrate all these individual modules into one CPU.
- 11. Test the functionality of the CPU.
- 12. Debug, debug, and more debugging.
- 13. Discuss and add more features.
- 14. Document the project and the design process.

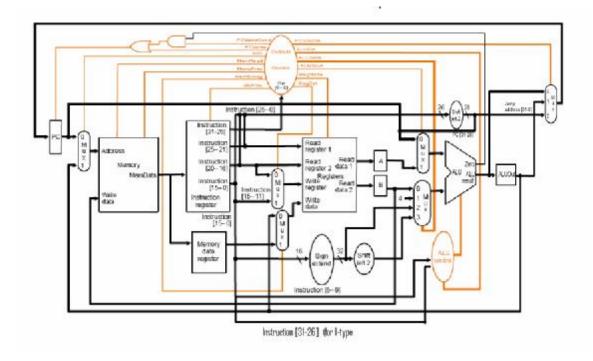
Our initial meeting was the most useful. As a team we decided on implementing a multicycle processor because it looked fun, challenging, and educational. Also when discussing what additions we wanted, the team decided that making a more robust ALU

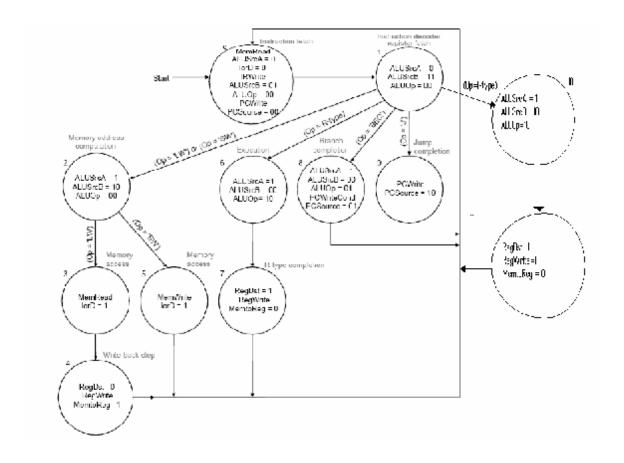
would be advantageous to the CPU. Adding support for NOR and XOR would greatly help the processor. Also, it was decided that support was for immediate instructions is incredibly useful when programming in assembly languages, so those had to be added. A modified schematic was sketched so the individual modules could be made.

The biggest challenge of coding the individual modules was figuring out how each module needed to function: what inputs and outputs were needed, and what triggered the changes. This challenge manifested itself all the way into the debugging stage of the whole CPU. The majority of the time of this project was spent debugging the code we had already written. This is understandable considering the complexity of the design.

Design

The design of this project can be broken down into two diagrams. One is a block diagram of our design. The second is the finite state machine that controls the CPU. The block diagram shows all of our individual modules as well as the wire interconnections between them. The finite state machine is a more conceptual schematic that shows the different states, their outputs, and the connections between them.





Testing Methodology

The testbench used to test this design isn't any more complex then the rest of the design. The testbench can be seen at the end of appendix A. The testbench controls and alternates the clock signal. We programmed a stop instruction into it, so the test wouldn't run infinitely. The actual inner workings of our test setup is actually imbedded well within the processor design. These inner workings include the initial data values that are to be loaded into the registers and the instructions that the CPU is to perform. This initial values and instructions were design to test many of the different operations the processor can execute. They can be seen encoded as initial conditions in the data memory module. The data memory module is included in appendix A. The instructions the team decided to show in this report are the instructions for computing a simple Fibonacci number. For those not familiar with Fibonacci numbers, a summary can be viewed at: http://csl.ee.iastate.edu/~cpre305/Labs%20for%20Fall%202005/Fibonacci number com putation.pdf

The Fibonacci number program requires that the data portion of the main memory be initialized with 8, 1, and -1, and a series of instructions be loaded into the instruction memory portion of the main memory.

Conclusion

The multicycle implementation of a CPU is a great improvement over a single-cycle implementation. It allows for the use of fewer design modules and a faster clock speed by breaking instructions into up to five different steps. These steps are controlled by a finite state machine controller. Our design shows the implementation of a multicycle CPU capable of handling fifteen different instructions. These instructions are in a variety of categories: R type, I type, and j-type. Each of these categories has a different instruction format. A multi-stepped design methodology was implemented to break this big project into many smaller steps. This project shows the wide variety of things to consider and components of a multicycle processor implementation.

Lessons Learned

This project enhanced the team's knowledge of instruction sets, processor design, instruction types, and Verilog code. The project has also given the team more practice in engineering problem solving. The project has shown and demonstrated to the team a design methodology that can be used for solving complex problems. There were a huge variety of lessons learned. The team learned different things every step of the way. From decision making and goal setting all the way up to debugging and documenting.

Appendix A: Verilog Code and Test Bench

```
//Multicycle CPU
module cpu(cycle, PC, inst, alu_out, mem_out, clock,
       regdst, aluop, alusrc, branch, memread,
       memwrite, regwrite, memtoreg, zero);
       // input/output
       input clock;
       output[31:0] cycle, PC, inst, alu_out, mem_out;
       output regdst, alusrc, branch, memread, memwrite, regwrite, memtoreg;
       output[1:0] aluop;
       output zero;
       // for debug
       reg[31:0] cycle=32'd0;
       always @ (posedge clock) begin
               cycle = cycle + 1;
       end
       // control variables
        wire [1:0] ALUOp, ALUSrcB, PCSource;
       wire PCWriteCond, PCWrite, lorD, MemRead, MemWrite, MemtoReg, RWrite,
               ALUSrcA, RegWrite, RegDst;
       // data path wires
       wire[31:0] pcmuxout, MemData, IROut, MDROut, MDRMux, SignExtendOut,
               SES4Out, RD1, RD2, ARegOut, BRegOut, AMuxOut, BMuxOut, ALUResult,
               ALURegOut, nextpc, pcshift2out;
       reg[31:0] PC=128;
       wire [4:0] instructionmuxout;
       //stuff to update PC
       wire PCWriteEnable;
       wire tempPCvar;
       //pcMux
       assign pcmuxout = lorD ? ALURegOut : PC;
       //memory
        DataMemory DataMemory1(clock, pcmuxout, BRegOut, MemWrite, MemRead,
               MemData):
       //Instruction register
       IR IR1(IRWrite, MemData, IROut, clock);
       //Memory Data Register
       MDR MDR1(MemData, MDROut,clock);
```

```
//Controller
       controller Control1(clock, Reset, IROut[31:26], PCWriteCond, PCWrite, lorD,
              MemRead, MemWrite, MemtoReg, IRWrite, PCSource, ALUOp, ALUSrcB,
              ALUSrcA, RegWrite, RegDst);
       //instruction decoding
       assign instructionmuxout = RegDst ? IROut[15:11] : IROut[20:16];
       assign MDRMux = MemtoReg ? MDROut : ALURegOut;
       //Registers
       RegFile
              registers(IROut[25:21],IROut[20:16],instructionmuxout,RegWrite,MDRMux,RD
              1,RD2,clock);
       ABregister A(RD1, ARegOut ,clock);
       ABregister B(RD2, BRegOut ,clock);
       // sign extend and shift 2;
       SignExtend SE1(IROut[15:0], SignExtendOut);
       assign SES4Out = SignExtendOut*4;
       //muxes for ALU input
       assign AMuxOut= ALUSrcA ? ARegOut : PC;
       Four2One Bmux(ALUSrcB, BRegOut, 32'd4, SignExtendOut, SES4Out, BMuxOut);
       //ALU controller and ALU
       wire [3:0] Operation;
       ALUControl ALUControl1(IROut[31:26], ALUOp[1], ALUOp[0], IROut[5:0],
              Operation);
       MainALU alu1(AMuxOut, BMuxOut, Operation, ALUResult, zero);
       ALUOut aluout1(ALUResult, ALURegOut ,clock);
       //jump stuff
       wire [31:0] jumpaddress;
       wire [27:0] jumpaddresstemp;
       assign jumpaddresstemp = IROut[25:0] *4;
       assign jumpaddress= {PC[31:28],jumpaddresstemp[27:0]};
       Four2One muxfourtwoone1(PCSource, ALUResult, ALURegOut, jumpaddress,
              jumpaddress, nextpc);
       //updating pc
       assign tempPCvar=zero&PCWriteCond;
       assign PCWriteEnable=tempPCvar|PCWrite;
       always@ (posedge clock) begin
       if(PCWriteEnable==1'b1)begin
              PC <= nextpc;
       end
       end
endmodule
```

```
// Register File module
module ABregister(DataIn, DataOut ,CLK);
       input [31:0] DataIn;
       input CLK;
       output [31:0] DataOut;
       reg [31:0] DataOut;
       always @(posedge CLK) begin
               assign DataOut = DataIn;
       end
 endmodule
// Register File module
module ALUOut(DataIn, DataOut ,CLK);
       input [31:0] DataIn;
       input CLK;
       output [31:0] DataOut;
       reg [31:0] DataOut;
       always @(posedge CLK) begin
               assign DataOut = DataIn;
        end
 endmodule
```

```
// ALU Control Module, multicycle CPU
module ALUControl(Opcode, ALUOp1, ALUOp0, Funct, Operation);
                       input ALUOp1, ALUOp0;
                     input [5:0] Funct, Opcode;
                     output [3:0] Operation;
                     reg [3:0] Operation;
                     always @ (ALUOp1, ALUOp0, Funct) begin
                                          if((ALUOp1==1'b0)&(ALUOp0==1'b0)) begin
                                                               Operation \leq 3'b010;
                                          end
                                          if(ALUOp0==1'b1) begin
                                                               Operation <= 4'b0110;
                                          end
                                          if((ALUOp1==1'b1)&(Funct[0]==1'b0)&(Funct[1]==1'b0)&(Funct[2]==1'b0)&(
                                                               Funct[3]==1'b0) begin
                                                               Operation <= 4'b0010;
                                          end
                                          if((ALUOp1==1'b1)&(Funct[0]==1'b0)&(Funct[1]==1'b1)&(Funct[2]==1'b0)&(
                                                               Funct[3]==1'b0) begin
                                                               Operation <= 4'b0110;
                                           end
                                          if((ALUOp1==1'b1)&(Funct[0]==1'b0)&(Funct[1]==1'b0)&(Funct[2]==1'b1)&(
                                                               Funct[3]==1'b0) begin
                                                               Operation \leq 4'b0000;
                                          end
                                          if((ALUOp1==1'b1)&(Funct[0]==1'b1)&(Funct[1]==1'b0)&(Funct[2]==1'b1)&(
                                                               Funct[3]==1'b0)) begin
                                                                Operation <= 4'b0001;
                                            end
                                          if((ALUOp1==1'b1)&(Funct[0]==1'b0)&(Funct[1]==1'b1)&(Funct[2]==1'b0)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Funct[2]==1'b1)&(Fu
                                                               Funct[3]==1'b1) begin
                                                               Operation <= 4'b0111;
                                          end
                                          if((ALUOp1==1'b1)&(Funct[0]==1'b1)&(Funct[1]==1'b1)&(Funct[2]==1'b1)&(
                                                               Funct[3]==1'b0) begin
                                                               Operation <= 4'b1100;
                                            end
```

```
if((ALUOp1==1'b1)\&(Funct[0]==1'b0)\&(Funct[1]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Funct[2]==1'b1)\&(Fu
                                                                                                                                        Funct[3]==1'b0)) begin
                                                                                                                                         Operation <= 4'b0011;
                                                                                          end
                                                                                          if(Opcode==6'b001000) begin //for add immediate
                                                                                                                                        Operation <= 4'b0010;
                                                                                          end
                                                                                          if(Opcode==6'b001101) begin //for or immediate
                                                                                                                                         Operation <= 4'b0001;
                                                                                          end
                                                                                          if(Opcode==6'b001100) begin // for andi
                                                                                                                                         Operation <= 4'b0000;
                                                                                          end
                                                                                            if(Opcode==6'b001010) begin // for slti
                                                                                                                                         Operation <= 4'b0111;
                                                                                          end
                                            end
endmodule
```

```
// Main ALU Module
module MainALU(DataA, DataB, Operation, ALUResult, Zero);
       input [31:0] DataA, DataB;
       input [3:0] Operation;
       output [31:0] ALUResult;
       output Zero;
       reg [31:0] ALUResult;
       reg Zero;
       always @ (Operation, DataA, DataB) begin
               if(Operation== 4'b0010) begin
                       ALUResult <= DataA+DataB;
               end
               if(Operation== 4'b0110) begin
                      ALUResult <= DataA-DataB;
               end
               if(Operation== 4'b0000) begin
                      ALUResult <= DataA&DataB;
               end
               if(Operation== 4'b0001) begin
                      ALUResult <= DataA|DataB;
               end
               if(Operation== 4'b0111) begin
                      if(DataB>=DataA) ALUResult <= 0;</pre>
                      if(DataB<DataA) ALUResult <= 1;</pre>
               end
               if(Operation== 4'b1100) begin
                      ALUResult \le \sim (DataA|DataB);
               end
               if(Operation== 4'b0011) begin
                      ALUResult <= DataA^DataB;
               end
               if(DataA==DataB) begin
                      Zero<=1'b1;
               end
               if(\sim(DataA == DataB)) begin
                      Zero<=1'b0;
               end
       end
endmodule
```

```
module controller (Clk, Reset, Op, PCWriteCond, PCWrite, lorD, MemRead, MemWrite,
       MemtoReg, IRWrite, PCSource, ALUOp, ALUSrcB, ALUSrcA, RegWrite, RegDst);
       input [5:0] Op;
       input Clk, Reset;
       output [1:0] ALUOp, ALUSrcB, PCSource;
       output PCWriteCond, PCWrite, lorD, MemRead, MemWrite, MemtoReg, IRWrite,
              ALUSrcA, RegWrite, RegDst;
       reg [1:0] ALUOp, ALUSrcB, PCSource;
       reg PCWriteCond, PCWrite, lorD, MemRead, MemWrite, MemtoReg, IRWrite,
              ALUSrcA, RegWrite, RegDst;
       reg [4:0] state =0, nextstate;
       parameter S0=0;
       parameter S1=1;
       parameter S2=2;
       parameter S3=3;
       parameter S4=4;
       parameter S5=5;
       parameter S6=6;
       parameter S7=7;
       parameter S8=8;
       parameter S9=9;
       parameter S10=10;
       parameter S11=11;
       always@(posedge Clk) begin
              state=nextstate;
       end
       always @(state, Op) begin
       case(state)
              S0: begin
                     MemRead=1'b1;
                      ALUSrcA=1'b0;
                     lorD = 1'b0;
                     IRWrite=1'b1;
                      ALUSrcB=2'b01;
                      ALUOp= 2'b00;
                     PCWrite=1'b1;
                     PCSource=2'b00;
                      nextstate=S1;
                     RegWrite = 1'b0;
                     MemWrite=1'b0;
                     PCWriteCond= 1'b0;
                      MemtoReg=1'b0;
              end
              S1: begin
                     MemRead=1'b0;
```

```
ALUSrcA=1'b0;
                             ALUSrcB=2'b11;
                             PCWrite=1'b0;
                             ALUOp= 2'b00;
                             if(Op==6'b100011) begin //if op code is lw or sw
                                                         nextstate=S2;
                             end
                             if(Op==6'b101011) begin //if op code is lw or sw
                                                         nextstate=S2;
                             end
                             if(Op==6'b000000) begin // if R type instruction
                                                         nextstate=S6;
                             end
                             if(Op==6'b000100) begin //if beq instruction
                                                         nextstate=S8;
                             end
                             if(Op==6'b000010) begin //if jump instruction
                                                         nextstate=S9;
                             end
                             if((Op==6'b001100)|(Op==6'b001101)|(Op==6'b001110)|(Op==6'b001110)|(Op==6'b001110)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b001100)|(Op==6'b0
                                                           11)) begin //if I type
                                                            nextstate=S10;
                             end
end
S2: begin
                             ALUSrcA = 1'b1;
                             ALUSrcB= 2'b10;
                             ALUOp = 2'b00;
                             if(Op==6'b100011) begin //if lw
                                                         nextstate=S3;
                             end
                             if(Op==6'b101011) begin // if SW instruction
                                                         nextstate=S5;
                             end
end
S3: begin
                             MemRead=1'b1;
                            lorD = 1'b1;
                            nextstate=S4;
```

IRWrite=1'b0;

```
end
S4: begin
       RegDst = 1'b0;
       RegWrite = 1'b1;
       MemtoReg= 1'b1;
       nextstate=S0;
       MemRead=1'b0;
end
S5: begin
       MemWrite=1'b1;
       lorD= 1'b1;
       nextstate=S0;
end
S6: begin
       ALUSrcA= 1'b1;
       ALUSrcB= 2'b00;
       ALUOp = 2'b10;
       nextstate = S7;
end
S7: begin
       RegDst= 1'b1;
       RegWrite = 1'b1;
       MemtoReg = 1'b0;
       nextstate= S0;
end
S8: begin
       ALUSrcA= 1'b1;
       ALUSrcB= 2'b00;
       ALUOp=2'b01;
       PCWriteCond= 1'b1;
       PCSource = 2'b01;
       nextstate= S0;
end
S9: begin
```

S10: begin

end

PCWrite= 1'b1; PCSource= 2'b10; nextstate= S0;

```
ALUSrcA= 1'b1;
                      ALUSrcB= 2'b10;
                      ALUOp = 2'b10;
                      nextstate = S11;
              end
              S11: begin
                      RegDst= 1'b1;
                      RegWrite = 1'b1;
                      MemtoReg = 1'b0;
                      nextstate= S0;
              end
              endcase
       end
 endmodule
// Register File module
module IR(IRWrite, DataIn, DataOut, CLK);
       input [31:0] DataIn;
       input CLK, IRWrite;
       output [31:0] DataOut;
       reg [31:0] DataOut;
       always @(posedge CLK) begin
              if(IRWrite==1'b1) begin
                      DataOut <= DataIn;
              end
       end
endmodule
// Register File module
module MDR(DataIn, DataOut ,CLK);
       input [31:0] DataIn;
       input CLK;
       output [31:0] DataOut;
       reg [31:0] DataOut;
       always @(posedge CLK) begin
               assign DataOut = DataIn;
       end
endmodule
```

```
// Data Memory module
module DataMemory(Clk, Address, WriteData, MemWrite, MemRead, MemData);
       input [31:0] Address, WriteData;
       input MemRead, MemWrite, Clk;
       output [31:0] MemData;
       wire [31:0] MemData;
       reg [31:0] RegFile [512:0];
       always @ (posedge Clk) begin
               if(MemWrite==1'b1)begin
                      RegFile[Address] <= WriteData;
               end
       end
       assign MemData = (MemRead==1'b1)? RegFile[Address] : 0;
        initial begin //load in data and instructions of program
               RegFile[0] \le 32'd8;
               RegFile[1] <= 32'd1;
               RegFile[2] <= 32'd1;
               RegFile[128] <= 32'h8c030000;
               RegFile[132] <= 32'h8c040001;
               RegFile[136] <= 32'h8c050002;
               RegFile[140] <=32'h8c010002;
               RegFile[144] <=32'h10600004;
               RegFile[148] <=32'h00852020;
               RegFile[152] <=32'h00852822;
               RegFile[156] <=32'h00611820;
               RegFile[160] <=32'h1000fffb;
               RegFile[164] <=32'hac040006;
       end
endmodule
```

```
// Register File module
module RegFile(RA,RB,W,WE,WD,RDA,RDB,CLK);
       input [4:0] RA, RB, W;
       input [31:0] WD;
       input WE, CLK;
       output [31:0] RDA, RDB;
       reg [31:0] RegFile [31:0];
       reg [5:0] i;
        always @(posedge CLK)
               if(WE) RegFile[W] <= WD;</pre>
        assign RDA = RegFile[RA];
        assign RDB = RegFile[RB];
        initial begin
               for (i = 0; i < 32; i = i + 1)
                       RegFile[i] = 32'd0;
        end
endmodule
// Sign Extender
module SignExtend(In, Out);
       input [15:0] In;
       output [31:0] Out;
       assign Out = \{\{16\{In[15]\}\},In[15:0]\};
endmodule
// 32-bit four to one mux
module Four2One(Control, A, B, C, D, Out);
       input [1:0] Control;
       input [31:0] A, B, C, D;
       output [31:0] Out;
       reg [31:0] temp1, temp2, Out;
        always @ (Control, A, B, C, D) begin
               assign temp1 = Control[0] ? B : A;
               assign temp2 = Control[0] ? D : C;
               assign Out = Control[1] ? temp2 : temp1;
       end
```

endmodule

```
//John Chargo
// Multicycle CPU Testbench
module testbench;
       reg clock=0;
       wire[31:0] cycle, pc, inst, alu_out, mem_out;
       wire regdst, alusrc, branch, memread, memwrite, regwrite, memtoreg;
       wire[1:0] aluop;
        wire zero;
       cpu cpu1(cycle, pc, inst, alu_out, mem_out, clock,
               regdst, aluop, alusrc, branch, memread,
               memwrite, regwrite, memtoreg, zero);
       always begin
               #20 clock<=~clock;
       end
       initial begin
               #2500 $stop;
       end
       initial begin
               $monitor(":At Time: %d; Clk: %b; ",
                               $time, clock);
        end
```

endmodule

Appendix B: Simulation Results

- # Compile of AB Register.v was successful.
- # Compile of ALUControl.v was successful.
- # Compile of ALUOp.v was successful.
- # Compile of ALUOut Register.v was successful.
- # Compile of controller.v was successful.
- # Compile of four to one mux.v was successful.
- # Compile of Instruction Register.v was successful.
- # Compile of MainALU.v was successful.
- # Compile of Memory data register.v was successful.
- # Compile of Memory.v was successful.
- # Compile of Multicycle CPU.v was successful.
- # Compile of Registers.v was successful.
- # Compile of SignExtend.v was successful.
- # Compile of testbench.v was successful.
- # 14 compiles, 0 failed with no errors.
- vsim work.testbench
- # vsim work.testbench
- # Loading C:\Modeltech_6.0c\win32/libswiftpli.dll
- # Loading work.testbench
- # Loading work.cpu
- # Loading work.DataMemory
- # Loading work.IR
- # Loading work.MDR
- # Loading work.controller
- # Loading work.RegFile
- # Loading work. ABregister
- # Loading work.SignExtend
- # Loading work.Four2One
- # Loading work.ALUControl
- # Loading work.MainALU
- # Loading work.ALUOut
- add wave -r /*
- run –all

And red Arch The And	10				
	ő		. 3		5
1 de de de stripe	12	12 112			×
_ display the description of the graph of the defect the colors of the		} ;	J		
344 Availand Action Control Co.	li	 	- v		
 Availand Appli/9804/0804d 	0				
 And Final Action 2010 Another above 10 m² 					\vdash
Also the activity of the file	2				+
Aschalch (co. Louis Sign	v				
± ♦ And and Accident Conf.	125	137 (1.7	, , , , , , , , , , , , , , , , , , ,	11.	1.0
3-9 Andred Address Co. 3-4 Andred Andres Co.	34553046	175 F4 (0) 08,000	31	Sign-en in	1 801.00
7 4 1-44. 3.470.);~~··)	>	. 01948991	7801200
1 🍎 Alectication and Albie		117		1133	2(
 Asstruction delign (with in 		71			
g-4 Andowhiko.173449. G-4 Andowhiko.1714	ľ	<u>n</u>			
The Analysis Analysis	ļ,	<u> </u>			
my 🧇 the Lorent to Mark Rept.		ν .			
production of the control of the con	12)			
ij ♦ Asche which door (d). <u>a</u> ♦ Asche which (d).	17	. 20 132	-	1122	×
g-4 Anion Action For . g-4 Anion Action For Du	1%	10. (0.	-		1-7
3-4 Archard Acceleration 3-4 Archard Acceleration	18 >	30		9	70
na Arabad Araban i and Arabad Araban union resist	18	rs fr.		3	174
April and April 550-505-144.					
 And set As the quarter 	0				
The Assistant Assistance		<u>:</u>			
* Asothe white the mondidate * Asothe white the mondidates		785480 785480			025000
→ Accept Accept Accept Addition	12	DA (1)	30		N
3-4 Am Lond Act, Int. dather College Audulo		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,			
 Archard Actual states and Markson Archard Scholzschein (1994) and 					
 The Line of the Matthews of Matthews (iii) Assert with for Matthews empty (10) 	n n				
g 🍫 Also the let Art. De Calabi-Arthyl J. Harry Gille.	2 FS 1F4	97H)4H0)	 3	190991	,
🔟 🥎 Wedža schikovita i Papalata	7 F5.3F4.	51F57(153)	3	[181*8a-2-1	1
 Architect Act, to Stock 5 Architect Act, to Stock 4 	V .				
The grant and the second of th		36.00			AL1.00
m 🧇 Asi 1-101 A. J. NOC 145-201	- resource	19864:0	>	. 01908961)
 Asstruktor Josef (141). 	ļ!				
As the short design 4.140 to 14 As the short due to 140 to 140.	ľ	70	*	1,41,464.1	!'
September 1921 For the	jo				
 Analysis Adalysis of the Analysis of the Analysis					+
m 🧇 for head for define or that figures is a first of the formal of the first of the formal of the formal of the formal of the first of the formal of the first	l ^o .	<u> </u>			
No.	200 n		1 m 1 m 1 m		
7.w	0 v	-17	77 W 177 127	4 90 70 87	SM 7:0
Table And Let Av. 17 av. (at 1975)	ļ.	2	ا	5	/
The state of the s	10				
 Asothe white the individual visit of the Asothe white the individual individual visit of the visit of the control of the control					+
🤣 Assido whites, Indo Are 1 years	0				
					1
Available Available of Verily or	0				
Another shifter the Total Community	ļ."				
 Aparticular of an include time. 	lu .				
 Accomplées, laus été le liegué m 	10				
 Architect Action of Chicago Action and Action of Chicago Action 	,			,	
 And and Actificated Angled And and Actificate Control And and Actificate Control 	0	. 2	9	1 2	<u> </u>
Additional Additional Control Control Additional Additional Control Control Additional Contr	0	2 2	DI 5	1 3	;
1 de de de seléction de marcone 11	0		D1 27		
1	> 0 1		50	3	; ;
6	> 0 1 > > >	3 ×	51 9	· 2	
The desire desired and the control of the desired and the de	0 1 2 2 3 4 9 0	3 × 0	51	· 2	
products which discrepand II and discrete which discrepand is and discrete which discrete which and discrete which discrete which and discrete which are which and builty and are all the	0 1 2 2 3 4 9 0 0	3 ×	51 72		
The desire desired and the control of the desired and the de	0	n 3 2 2 2 3	51.0 ————————————————————————————————————	1 0 5	
1	0 1 2 3 4 9 9	3 × 0	51 52	7	
The destroy of the control of the co	0 0 1 2 2 3 4 4 4 6 6 7 7 8 7 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9	25 T	51 22		
Control of the Control of Contr	0	35 - 27 - 27 - 27 - 27 - 27 - 27 - 27 - 2			
products with discourse of the discourse	0	22 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	5.0 		
products with discrepance of a set of the control o	> 0 0 1	25 20 20 20 20 20 20 20 20 20 20 20 20 20	55		
The destroy of the control of the co	0	10 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	57		
The destroy of the control of the co	0 0 1	25 20 20 20 20 20 20 20 20 20 20 20 20 20			
The destroy of the control of the co	> 0 0 1	10 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	27 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2		
The destroy of the control of the co	0	2	57 b		
The destroy of the control of the co	> 0	2			
The destroy of the control of the co	> 0 0 1	2			
The destination described in the destination of the destination of the described in the destination of the d	> 0	1	25 to 10 to		
The destroy of the control of the co	· > 0 · · > > > > 0 · > · > > > 0 · > · >	2			
destricted destricted in the control of the control	0 0 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2			
The destroy of the control of the co	· × 0 0 1 · × × × × 0 0 · × · × · × · × · × · × ·	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2			
The destroy of the control of the co	÷	1		7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	
The destroy of the control of the co	· > 0 · · > > > 0 0 · · > > > > 0 0 · > > > > > 0 0 · > > > > > 0 0 · > > > > > 0 0 · > > > > > 0 0 · > > > > > 1 0 0 0	18 20.			
Control of the Control of Control	4 5 100	1		7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	
The destroy of the control of the co	1.00 1.00 10	18		20 20 20 20 20 20 20 20 20 20 20 20 20 2	A
The destroy of the control of the co	1.05 1.05 10 10	10		3 2 2 2 2 3 3 3 3	,
The destination described in the destination of the destination of the described in the destination of the d	1.05 1.05 10 10	18		20 20 20 20 20 20 20 20 20 20 20 20 20 2	# # # # # # # # # # # # # # # # # # #
The destination of the control of th	- 1.01 10 10 0 2	18		20 20 20 20 20 20 20 20 20 20 20 20 20 2	#
The destroy of the control of the co	- 1.01 10 10 0 2	18		21 22 23 24 24 24 24 24 24 24 24 24 24 24 24 24	# # # # # # # # # # # # # # # # # # #

→ And red Art k	0					
To de lead April	jä	<u> </u>	,		111	2.
g 🥎 discript while	12	IX.			14	
_	ļ!	· .			-	2
G-4 And Land Act, Innuition of	j.	1 Î			-	
 Availand Act 1980 (4) (20 ad) 	0		+		\vdash	
 And Find Scills 2000. And Scills Advantage 			+		-	
🍑 Aserba ablee de Benke	į:					
 Aschalch Act, Louis Sign 	V.					
gety And Lord Actions (1997) gety And Lord Action (1997)	125 325530245	1 2		519H-105 (7)	l''	
(4.4) A ≤ 1.44 A < 1.37 m		ACTAIN:	<u> </u>		kt:at.	
m is delicated at the MOTO . 1.4 destruction and the		2	+	\$194.805/A	<u> </u>	
The state of the field which in	ľ	[#C)	10	144	
_a→ Acc±c at Acc. (±32,440).	,	-			 	
3-4 And Lad Add Art 1912		ian marka and a second				
The first of the description	ľ				$\overline{}$	
g 🤝 Gentle white Jan Degni.						
	12	IX C		. 3	ĮZ.	
#* And and And Installant	1%	17		(1 (1)	1-7	7
3-4 And Lord And Indus Fugilia		141 - 1		2.11 2.11 2.12	[17]	i i
rade Ara Leed Art Leed γ 3-4 Ara Leed Ara Leed action respect	18	PA !	++	240	1/4	<u>'</u>
4 Ara Lord Act 1/20/4025 abb.					,	
 And set Art to spEC = a 	n		+			
The American Actual Specials A Contract Configuration		2 1045280			03 0728	
 4 Aboths white the monthless 4 Aboths white the monthlesses 		10128			15-23	
🚁 🐤 Portbolich (co. In/iskal kmo//I./Addon)	12	ix j		(1)	177	
And State of No. 1 (1944) one of NASA (1) And Lind And Individual Control And		<u> </u>				
 And Land St. Hollands and J. Mark St. Mark And Land St. Hollands and J. Mark 	n					
 As the white the Laboratory LT As the white the Laboratory LTH as the 	n					
 j * Asche shitz de Calabant ylüden sele j * Asche shitz de Propiele 	7 85 184 7 85 3845		2	019000004 (19400000	2	
4 Anitation (Children)	0	l' 	<u> </u>	TIME CAN.	<u> </u>	
 Archard Act to Bio R= 05 						
교수 하시 64년 1년 (14년) 기 2 4년 14년 4. 교사학(14년)	- 9500MI	2010MC 2 C1	+	\$198.805/A	Delaste.	
4 Act when Joh Clark	ii	<u>'</u>	r e	-17-22-20-	ř –	
🔟 🥎 45: the school deal of 140: v 14		7 7	'n	161,58407	h	
→ Acido et Acido de Nuo Vellado → Acido et Acido de Nuo Vell → Acido et Acido de Nuo Vell → Acido et Acido et Acido et Acido → Acido et Ac	i.	"				
🍎 And Led Act (2004) - 3 Au	ľ					
rg 🧇 the fill wall for the Time of the Control	n	,				
properties at Annulus Theory and Theory No. 2	200 m	. • 	+		 	
7. w	0 V	1 Skr 1 me 1 1 am 1 1 am 1	90 170	The second of	27 P.0	IM PET
(a) A set (a) A set (b) a set (b	ı	· c		•	ŀ	52
 As 1 and A. J. C. Brown Colored As the shifted the plant in Plant 2 for 	10	,	+			
And the advice that in a first in teach of a first And the advices that in a first in a first in a	;		+		+ +	
🕹 As (\$5 white, layer the five of	v					
Available Activities of the Pecch Available Activities of the Available			7		1	
 And Lord Activities of West Williams Surface of Manager Street 	l,					
🌎 Asatte ab Arado Triano i AD-Mei						
 Asotherables de maior lest dives. Asotherables, leut des le legacità. 	l::					
* Archard Accident Conglet	ļ,					
→ Aralland Accol/State (1984)	0		+ I	D C	h	[
The Art and Art I of the Control of	<u> </u>	}	''	<u> </u>	''	<u>-</u>
de deste which destroyed H	,	:			7	
🔟 🧇 Alecte within delegation (+)		:				
Accide white, large conti-to Accident for the	0	14	,	; n	147	7
 Are Lored Act, In Lorent will be 	ő					
rg 🧽 the fill well for the first of \$5.00		,				
i j 🤝 Asate white Jacobs (1981)	l:					
👔 👙 Alecte white James 2004)				
👔 👙 Alecte white James 2004) 22 1				
 Description of the description Description of the description Description of the description 	, , ,	` <u> </u>				
1 🍫 decta white described	> > 0 >) 22 1				
productive about the money of the desire which the desire of the desire))))	<u>, </u>				
1 to Andrew State American John Medican of Medican Colored Grand and American Colored Grand American American Grand American American State American American State American American State American American John American American State American American John American American John American American John American American John American American John))))	<u> </u>				
productive about the money of the desire which the desire of the desire	0	<u>, </u>				
March with the continue March with the Conduct March with the Co	0	; ;				
Some and the control of the contr	0	; ;			\$:	
Solid Section Continues)))))))	; ;		: : : : : : :	\$:	
1 Section of the description of the section of the section of the section of the section of the section of the section of the section of the section of the section of the sec)))))) - - -	; ;		3	2	
Machine Articulum Communication of the Communication Control Communication Communic	, , , , , , , , , , , , , , , , , , ,	; ;		3	\$ 2	
1 Activities of the Americans Actived Section Section Active Section Section Active Section Active Section Section Active Se	· · · · · · · · · · · · · · · · · · ·	; ;		3	2	
Machine Articulum (Machine) Mach	> > 2	1		3	:	
Machine Machine Community Machine Machine Machine Machine Machi	, , , , , , , , , , , , , , , , , , ,			3 3		7
1 Activity of the American 2 Activity of the American 3 Activity of the Activity 3 Activity of the Activity 3 Activity of the Activity 4 Activity of the Activity of the Activity of the Activity 4 Activity of the Acti	2 102 4 5			3 3 3 2 3	:	7
1 Activities of the American Actived Section Section Active Section Active Section Section Active Se	2 2 105 4 3 107	1		3 3	1 1 2	7
1 decide of the demonstration	2 2 105 - - - - -			3 3 17 3 20	:	
1 decide of the demonstration	2 2 105 4 3 107	1		3 3 5 20 20		7
1 decide of the demonstration	2 2 105 - - - - -	1		3 3 17 3 20	E	
1 Activities of the Americans 2 Activities of the American 3 Activities of the American 4 Activities of the Activities of the American 4 Activities of the American 4 Activities of the Activiti	2 2 105 - - - - -	1		3 3 20 20 20		37 11 11 12 12 13 14 14 15 16 16 17 18 18 18 18 18 18 18 18 18 18 18 18 18
1 Method Will American 2 Method Well Level Duran 3 Method Well Level 4 Method Well Level 3 Method Well Level 4 M	5 5 2 2 125 4 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1			3 3 20 20 20 20	1	9 1 2 2
1 decide of the demonstration of the control of the	5 5 2 102 4 5 107 101 0 5		Vr. **A	3 3 20 20 10 20	1	

* h-1-15- k	۸.						
And red Art is And red Artis And red Artis	li,	in the	20.		s		
plop dische stiller	12	140			<u></u>		
→ As to show the 1 =	10						
g-4 Arabo shAro, haddada g-4 Arabo shAro, hada sa	li.	.			2	<u>'</u>	
 Availand Act 1990 William 	ó						
 No.1 and No. (1/27) No. 				- I - I			
 Assite ablantions? Assite ablantion forte 	;		_				
Ascha shifted to US of	lo lo						
🚁 🍫 Archand Accidence (1997)	122	1.5	[14]			7	
g-4 And and And Albertain g-4 And and And In State	34559046	1. 2	\$48V	(B-1)		5	
東 4 가지 5년 5년 1년왕(6) 구 4 4년 1년 4년 4년 1년왕(6)	,-m-	AtaMa A D	f reg-	.91.13	av .uc	r ₁	
1 🍨 dische ablän des 4 kirp		: "	2144		52 II		
🔟 💠 Asate which the Epsilowers in		`					
g-Aprido at Action 20040. G-April and Action 1	ľ.	P					
3-4 An Land An. 1/512	ľ,						
ry 🧽 that he distributed Supple		,					
products which defined to	<u>}</u>	<u> </u>					
ig ♦ Aside et Artillore (d). a ♦ Aside et Artillore (d).	12	₽	1102		<u>. </u>		
#* And and And Installant	1%	,	2111				
3-4 And and Analysis Fuglia		l	2111		s9 5		
The Art with transfer	18	-	2111		<u> </u>		
340 And Land Act, In notation respect to And Land Act, 1980-4003 (ab).		ľ					
🧇 Arati set Arii teli gr#Clum	n						
m 🧇 Aside ad Aside Tre wise		:					
As the obligation of a money determined to the obligation of		8.53			<u> </u>		
_i ♦ Plante white the monthleament _i=0 Points white, to likely know IVA date in	12	14. 2	2115		// //		
3-4 And and Action water a state of the state.	>	, 					
Belling the factor of the state of the state of the state of the factor							
The Lorent Struck State of Address 12	n o						
 Asother show the County on 1917 Asother show the College on 1917 Hard selection 	0 2 85 184) :1	- 40:	2912 8			
n 🍁 Akaba sehkat, In Probablis	7 Ph. JP4.	1 3 		7917 E			
 Avalored Act, 1987 (Act.) 	V						
 Archard Action 3 of K+0 or \$4.0 Archard Action 1 of Action 	l	ALLEN.					
7 2 44 1-44 4. 3. 470 1420	· 2523612	/ 3	5196	291.13	av .uc	5	
 Activities des Calle 	п						
→ 45/16 about 1/24 4 140-9 14		7 7		-पा- र -		:	
Acres of Acc. Into Acc. (3)	l.	"					
And Land Activities of Shoul	ľ						
mg 🤣 from and South Times and Times	n	,					
productive about the first and the first	1	<u>د ب</u>	 				
No.		577 527	20 m 500	W 98	<u>in in iim</u>	1000 1100	7/7
T. W. Radio Architecture (al Ref	0.0			1 6	. 5.		
mark the London A. A. C. and Philosophics	ļ.	;					
🤣 Aberta abiker de hidroid Aberra Sorri	lu .						
About with for the indicate of the form of	2					_	
👲 Asido white, Ivac Vicinia							
 Acide et Rectau de Princip Acide de Carte de Cart	0						
Acade white the deficiency Acade at Acade at Control	 v o						
As the white, the party of the death of the the West of the Land As, the ball of the the the land As, the land of the the the land As the land of the land the land As the land of the land of the death white defends of the land	V 0 0						
 Assistant Acoustic Control of C	0 0 0						
As the white, the party of the death of the the West of the Land As, the ball of the the the land As, the land of the the the land As the land of the land the land As the land of the land of the death white defends of the land	0 0 0						
A design of Man, Note Andrews A design of Assistant Assistant Assistant A design of Assistant Assistant A sign of Assist	0 0 0 	, DI	1 01		-	-	
 A status of Neuroland Control Control A status of Neuroland Control	0 0 0 	, 51 2	1 31		-		
A state of Act, the Periods A state of Act, the Periods A state of Act, the Act of Act, the Act of Act, t	0 0 0 0 1	, DI	1 21		- 26	99	
A state of Action of Acti	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	- DI	. 3		20	200	
 As place of Note that a profession of the control of	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	, DI	1 31		- - - - - -	0 2	
A state of New York Control of State of State of New York Control of State of New York Control of State of New York Control of	0 0 1 0 1 0 1 7 7	/ DI	. 3		D D		
A state of New York Control of State of State of New York Control of State of New York Control of State of New York Control of	0 0 1 0 1 0 1 0 1	, 51 , 5 , 5	. 3				
A state of Men. In a definition A state of Men. In a state of Men. In a definition A state of Men. In a state of Men. In a definition A state of Men. In a sta	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	, DI	. 3		5 2 2 3	9	
A status of the file of the file of the status of the status of the file of the status of the statu	00.	7 D1 J J J J J J J J J J J J J J J J J J	. 3		0 0 0		
A state of the following of the control of the cont	000000000000000000000000000000000000000) ; ;	. 3		D D		
A state of the following of the control of the cont	00.110.200) ; ;	. 3		D D		
A state of the first of the second of the se	00.110000000000000000000000000000000000) ; ;	. 3		D 2		
A state of the close of the control of the close of the c	00.110>01.2>>00.2) ; ;	. 3		20 20 20 20 20 20 20 20 20 20 20 20 20 2		
A status of the following of the control of the con	00.110.501) ; ;	. 3		D D		
A status of Act, the above the act of Act, and Act, the above the act of Act, and Act, the act of Act, and and Act, and and act, and and act, and and act, and act, and act,	00.000000000000000000000000000000000000) ; ;	- <u> </u>				
A status of the final point of the control of the c	00.110>01.2>>>00.2.30>>>0.2.30) ; ;	- 200		D D		
A state of the close of the control of the close of the c	00.000000000000000000000000000000000000) ; ;	- 2III				
A state of the close of the control of the close of the c	00.110>01.2>>00.2.3) ; ;	- 200				
A state of the close of the control of the close of the c	00.110.00.2.2.20.2.2.2.2.2.2.2.2.2.2.2.2) ; ;					
A status of the Local Performance of the Local	00.110.501.5.500.5.505.5.17.4.500.5.505.5.17.4.500.5.505.5.17.4.500.5.505.5.17.4.500.5.505.5.17.4.500.5.505.5.17.4.500.5.505.5.17.4.500.5.505.5.17.4.500.5.505.5.17.4.500.5.505.5.17.4.500.5.505.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.5.505.505.505.505.5.505.5.50) ; ;					
A status of the close of the control of the close of the	00.110>01.2>>>00.>.>0>>.12.4>00.2>						
A status of the Local Performance of the Local	00.000000000000000000000000000000000000) ; ;	- 2111 - 2111 - 2				
A status of the Local Performance of the Local	00. III U > 0		- 2111 - 2111 - 2		7 7 7		
A status of the close of the control of the close of the	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0						
A status of the close of the control of the close of the	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		- 200 - 200		7 7 7		
As the above the control of the cont	i.or		- 2111 - 2111 - 22 - 22 - 27 - 211 - 214 - 214		3		
A status of the Local Performance of the Local	t		- 200 - 200		y		
A status of the characteristics of the charac	i.or		11 22 2 2 31 31 32 32 32 32 32 32 32 32 32 32 32 32 32		3		
A status of the Land Anthony of the Anthony of the Land Anthony of	4 1.07 101 101 0 5 0		2011 2011 211 22 22 20 20 20 20 20 20 20 20 20 20 20		3		
A status of the Local Performance of Annual	i.or		11 22 2 2 31 31 32 32 32 32 32 32 32 32 32 32 32 32 32		3		
A status of the large of the con- declared declared to the control of the control	2 107 10 101 101 0 2 0		211 211 22 24 25 27 212 212 212 212 212 212 212 212 212		3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3		
A status of the Land Anthony of the Anthony of the Land Anthony of	4 1.07 101 101 0 5 0		2011 2011 211 22 22 20 20 20 20 20 20 20 20 20 20 20		3 3 5 6 7 8 8 9 9		

	0							س
♦ And helder h ¬ ♦ And helder	ñ	18 '9	<u> </u>	<u>.</u> .	= =	5:		
i j 🍫 decht shifte Li 🍫 decht shifte das i li e	12	14		INE		- X	-	
Andrew Acc. Industrial	ï	: 		ŞI	 	:		
3-4 Available Action Control of Available Action PSP Action	V.	'				j ,		
The Line of the Charles of the	ľ							
 Asate white utilities. 	ļo.							
 Apartie withfacture Technic Apartie withfacture Leville on 	0		_			_		
gen Anderd Action (vol.) gen Anderd Action Library	122	,	[H4	ÎH.		3		
事からかけられた。1987年 中央の大学の大学の1987年	347530445	kum.	07462 16	0 5		9	<u> </u>	
m 🥎 Asile (14. J. 1770).	,	ñ	2,402:18	λ		3	, ,	
 4 And extraction and their 4 And extraction and parameters 		-	INE	1184 20)	3	36 12	
_a→ At r±c at Acc. 1732 (40).	,	h —		- j-			D I	
3-4 And and Account 3-4 And and Account 202	>	Harman State of the State of th						
The first of the description		;						
production and an incident	>	,						
\$\forall Application of the control of the	7	<u> </u>	2144	114E 21=	8		150	
🚁 And and Academic Con-	1%	,	111.	2174	,	9	1 0	
3-4 Andred Action Fugles 3-4 Andred Actions	18	Ľ	(14 (14)	0 1 0-1	!: I	3	16 20	
340 Am Land Act, In neuroline respect	>			3				
 Availabel Accide College Like Availabel Availabel op PCL 								
m 🧇 Aside ad Aside Tye with	,	:			s	2		
 4 Section of Artificial Introduction 		36.72		22 (55)			22	
□ ◆ Alertha white the mondidective of □ ◆ Alertha white. It is followed that it is a province of the province o	12	AC - 20	<u> </u>	[11] [12] 0±10		y	5,	
3-0 Amiliand Application and Amilian Amilian	>	'						
 Anderst Andre Johnson (J.Mar Seal) Andre Johnson (J.Mar Seal) 	n							
pro-Artista of Paul Landal Control (Addition) Biol Artista of Artista (Addition of Artista)	ń							
 death white the Calabaran yill Hers (de- gree Asoba white). In The Main. 	7 85 184 7 85 384		27472516 27472516	y y		3	8	
 Availand Accele 877.233 	V							
 As Lord Act In Riv Review As Lord Act In Riv Conference 		aca.		Statute.			л.п.	
m 😼 Asile al Asido (1970) (45-20)	- PETTERT		2,40508	2		2	,	
 4 April 2007 (17) (14) (14) (14) 4 April 2007 (17) (14) (14) (14) 	"		T. State of the	у —		_,		
→ As r±b schikes, foue the finds	,			-			 	
And Led Address College And Led And And And And And And And And And An	٧							
may be head to define the life	n	,				2.		
pro-Arache ad Art Johnson and The C	1	•	 		' 	: , , , ,	 	
bus 1	00 n	20 20	275 007	rn en i	ir in	on on a	21 1 W	1
(a) A set (a) A set (b) B ∈ (a) B ∈ (b)	1		3	Si	b l		<u> </u>	
 As 1 and A. J. T. And Physics at A starte white the public flows in the 	l:	<u>' </u>			-	2		
🍑 Also the will for the individual to	; '				1			
 As above PRepulsive Delivers As above PRepulsive Delivers 	V			_				
Archard Act, Indian or World an Archard Act, Indian or World an	0							
- 🤣 North and North 20 at a 1997 of Free	٥							
 Assite at Assistant Affine A Figure Assite at Assistant and Code Code 	i.		_			=		
🍑 Pecitia schikou loud Arcii y legyli a	Ü		_					
 Architect Accordance of Architect Architect Accordance of Architect 	>	l. I.	2	5	l. I	0		
Tarak Araba ada da d	ĭ	<u>: </u>	<u> </u>	2	<u> </u>		<u> </u>	
m 🦻 Asi Fest A. B. H. H. W. P. T 🍤 Aside white Beeckeys III	ļ:	! 		2			' 	
🔟 🥎 Alecte within directory/-/	>			ŷ				
→ As a±o white. In tig (xm/+D)	>	,	511-	(I-1	,	3	ж.	
Aratical Araticity of some 5 Aratical Araticity of the	ŏ							
rg 🧇 the fill well for the first of \$5.00	:							
 p ** Another within the entire of CDI (*) p ** Another within the entire of the control of the contr	. ·	<i>)</i>						
🔟 🧇 Alecka schötz, has Abata e		,						_
	>							
* And and And InvAtil. 3-4 And and And InvAtil.	> V >							
 And and Act to All 1. And and Act to All 1. And and Act to All 1. 	> U >	;						
 Am Lord Act, 178 ACD; 	> U > 0	;						
 Assisted Act, 1880 in Bernard Act, 1880 in Date Bernard Act, 1880 in Date 	> 0 > 0 >	,		2			,	
 Assisted Assistantial Assisted Assistantial Assisted Assistantial Assisted Assistantial Assistantial Assistantial Assistantial 	> V > 0 > 1	, !		5	2	<u> </u>), is	
 Arabard Anal (SAR). 	> U > 0 > 1	,	7 2	5 31	2	;	, s	
4. Archard Act, 1980, It. 3. Archard Act, 1980, It. 3. Archard Act, 1980, It. 3. Archard Act, 1980, It. 4. Archard Act, 1980, It.	> U > C O >	, !	2	2 21 29		3	5.	
 And Lord Add (ACD) 	> U > \ 0 > \ 1 : > 0	, !	2	5 31	j.	;		
 And Lord And LORALITY And Lord And Lord Lorality And Lord And Lord Lord Lord 	> V	, !	2	2 11 10 ²	j.	3	5.	
 And Lord Add (ACD) 	> U > \ 0 > \ \ 1 - \ \ \ 0 \ 0 \ \ \ \ 0 \ 0 \ \ \ \ \ 0 \ 0 \ \ \ \ \ 0 \ 0 \ \ \ \ \ \ 0 \ 0 \ 0 \ \ \ \ \ \ 0	, !	2	2 21 29	j.	3	5.	
And Lord And LORAL TO COME to the And Lord And LORAL TO COME to the AN	> > 2	, , , , , , , , , , , , , , , , , , ,	2	2 31 37 37 37	j.	2 2	\$* D	
A STATE OF	> U > 0 0 > 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0	, , , , , , , , , , , , , , , , , , ,	2 2 3 3	2 31 37 37 37	j.	2 2	\$* D	
A shall and August 2000. A shall and August) 2 112 4	, , , , , , , , , , , , , , , , , , ,	2011	2 0 2 2 2 2 2 2 2 2 3 3 3	K N	3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	8. 10 15 10	
A STATE AND STATE A STATE AND A STATE A STATE A STATE AND A STATE A	2 102 2 103 107	, , , , , , , , , , , , , , , , , , ,	2 2 3 3	2 31 37 37 3 3 3 3 3 3 3	i	2 2	5 0	
A STATE OF	2 102 2 103 107	, , , , , , , , , , , , , , , , , , ,	2011	2 0 2 2 2 2 2 2 2 2 3 3 3	K N	3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	8. 10 15 10	
And Lord Age, 1970. And Lord Age, 1970. And Age	8 2 125 4 5	, , , , , , , , , , , , , , , , , , ,	20 20 20 20 20 20 20 20 20 20 20 20 20 2	200	K N	2 2 2 2 2 2 3 3 4 3 4 3 4 3 4 3 4 3 4 3	5°	
And Lord And DANCH And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And Lord And LORAL And And Lord And Lord And Loral And And Lord And Lord And Lord And And Lord And Lord And Lord And And And Lord And Lord And And And Lord And Lord And And And Lord And And Lord And And And And And And And And	> > 2 2 1725	, , , , , , , , , , , , , , , , , , ,	20 20 20 20 20 20 20 20 20 20 20 20 20 2	201	K	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	
And Lord Age, 1970. And Lord Age, 1970. And Age	2 102 2 103 107	, , , , , , , , , , , , , , , , , , ,	20 20 20 20 20 20 20 20 20 20 20 20 20 2	200	K N	2 2 2 2 2 2 3 3 4 3 4 3 4 3 4 3 4 3 4 3	5°	
And Lord And DANCH And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And LORAL And And Lord And Lord And LORAL And And Lord And Lord And Loral And And Lord And Lord And Lord And And Lord And Lord And Lord And And And Lord And Lord And And And Lord And Lord And And And Lord And And Lord And And And And And And And And	> > 2 2 1725	, , , , , , , , , , , , , , , , , , ,	20 20 20 20 20 20 20 20 20 20 20 20 20 2	2017 2017 2017 2017 2017 2017 2017 2017	K	3 3 3 3 3 3 3 3	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	

Appendix C: Mistakes often made in Verilog

Some problems arise as a result of a lack of familiarity with the Modelsim software and verilog code. Some of the more common errors are as follows: Often when working with a signal of multiple bits, the order of the bits is confused. Most significant bits can be treated as least significant bits and vice versa. Also, an incorrect variable is often used to trigger an event. A change in one input might be used to decide when to run a piece of code when another should be used. The use of 'begin' and 'end' statements is foreign to many of us, and misuse of these can lead to errors. Sometimes variables are declared incorrectly. Regs and wires are confused the most often. The biggest problem is a general lack of knowledge of the capabilities of the language. There are easier ways to do many of the things that we do in our programs, but we are not aware of them. Occasionally we are shown new techniques by the TA, but often these techniques would have been useful earlier.