UNIVERSITY OF NEVADA LAS VEGAS. DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING LABORATORIES.

Class:	**CpE300L**	Semester:	**Spring 2021**

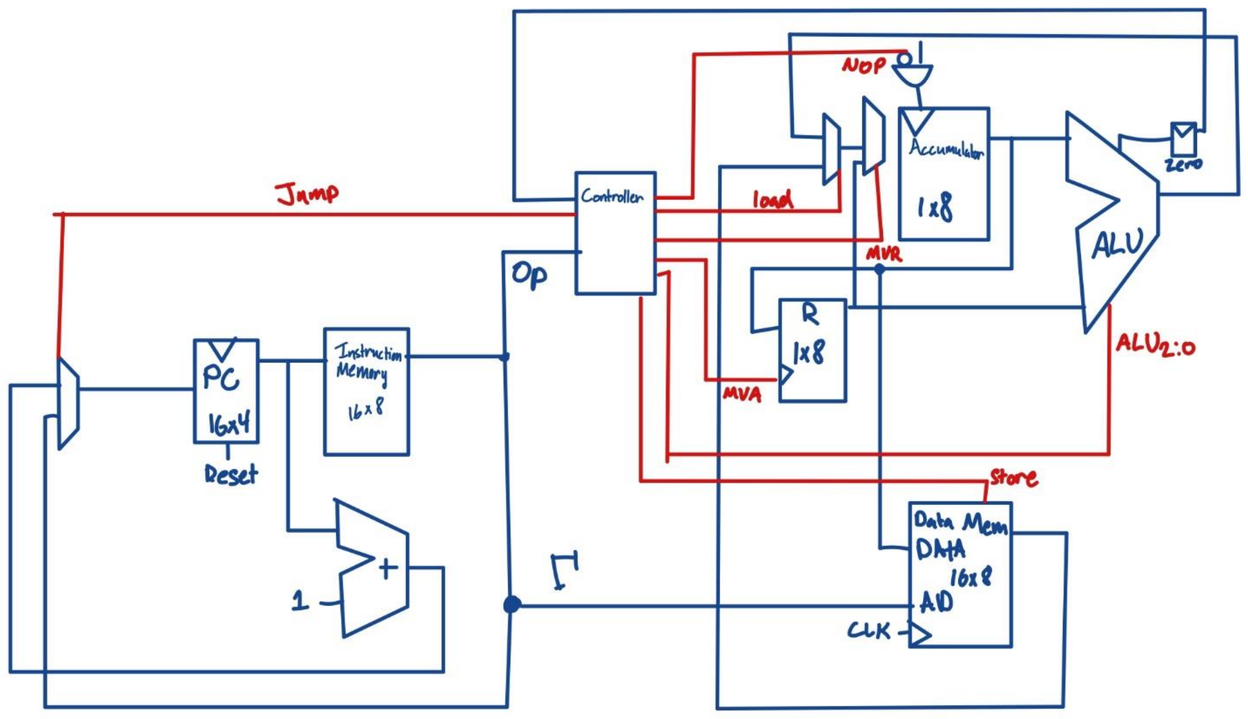
Points		Document author:	**Carlos Georges**
		Author's email:	**Georgc4@unlv.nevada.edu**
		Document topic:	**Final Project**

Instructor's comments:

1. Introduction / Theory of Operation

This report will detail the design and operation of the SMP8, a very simple 8-bit microprocessor designed with similar principles as the MIPS single-cycle processor. The processor consists of a combinational control unit with a small datapath. The datapath includes several multiplexers, instruction and data caches, 2 general registers, an ALU and other supporting components. The processor supports 16 instructions including a NOP instruction. The processor is not very powerful but it was designed for simplicity and not efficiency. We wrote testbenches for 2 test codes and implemented them on the DE2 FPGA board as well as in simulation (VCS).

2. Results of Experiments



SMP8 block diagram

#	Instruction	NOP	LOAD	STORE	MVA	MVR	JUMP	ALU
		8	7	6	5	4	3	2:0
1	NOP	1	0	0	0	0	0	100
2	LDAC	0	1	0	0	0	0	100
3	STAC	1	0	1	0	0	0	100
4	MVAC	0	0	0	1	0	0	100
5	MOVR	0	0	0	0	1	0	100
6	JUMP	1	0	0	0	0	1	000
7	JMPZ	1	0	0	0	0	Z	000
8	JPNZ	1	0	0	0	0	!Z	000
9	ADD	0	0	0	0	0	0	000
10	SUB	0	0	0	0	0	0	001
11	INAC	0	0	0	0	0	0	010
12	CLAC	0	0	0	0	0	0	011
13	AND	0	0	0	0	0	0	100
14	OR	0	0	0	0	0	0	101
15	XOR	0	0	0	0	0	0	110
16	NOT	0	0	0	0	0	0	111

Control word table

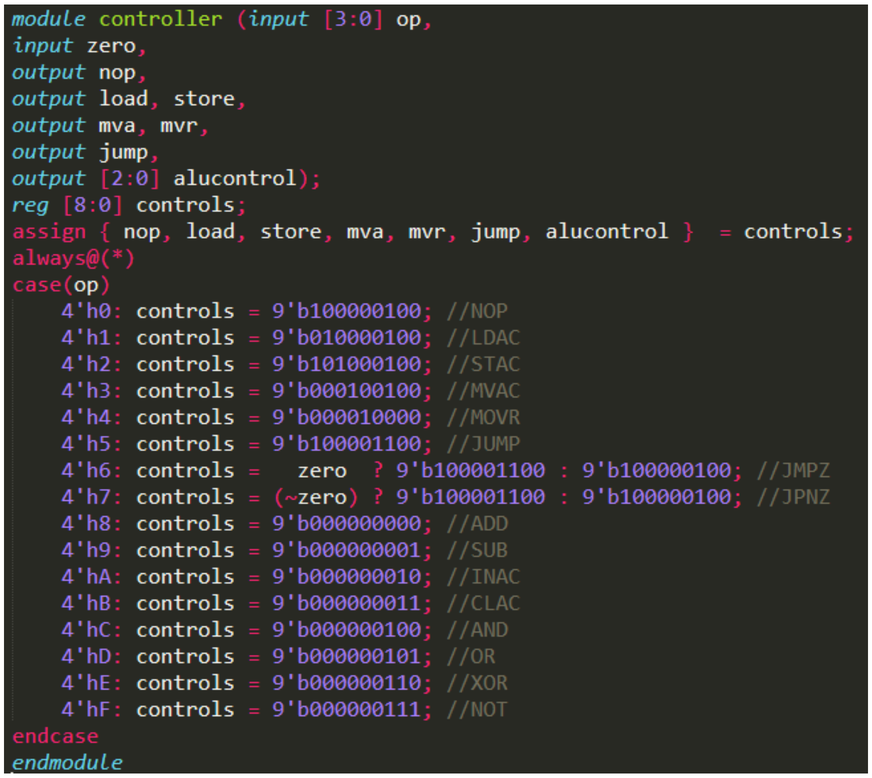
```
10      B0
A0      A0
74      30
50      F0
A0      E0
30      6A
80      24
22      00
00      00
```

Machine code for test 1 (left) and test 2 (right)

Design Codes:

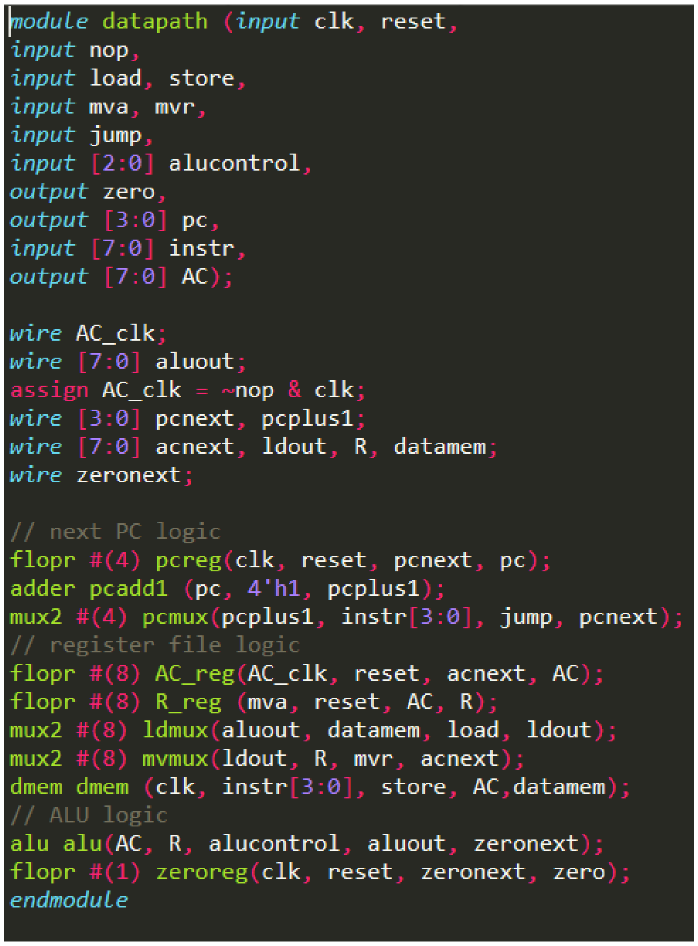
```
module smp8 (input clk, reset,
output [3:0] pc,
input [7:0] instr,
output [7:0] AC);
wire zero, nop, load, store, mva, mvr, jump;
wire [2:0] alucontrol;
controller c(instr[7:4], zero, nop, load, store, mva, mvr, jump, alucontrol);
datapath dp(clk, reset, nop, load, store, mva, mvr, jump, alucontrol, zero, pc, instr, AC);
endmodule
```

Top module



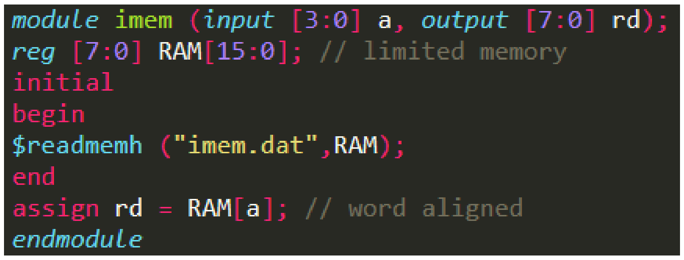
```verilog
module controller (input [3:0] op,
input zero,
output nop,
output load, store,
output mva, mvr,
output jump,
output [2:0] alucontrol);
reg [8:0] controls;
assign { nop, load, store, mva, mvr, jump, alucontrol }  = controls;
always@(*)
case(op)
    4'h0: controls = 9'b100000100; //NOP
    4'h1: controls = 9'b010000100; //LDAC
    4'h2: controls = 9'b101000100; //STAC
    4'h3: controls = 9'b000100100; //MVAC
    4'h4: controls = 9'b000010000; //MOVR
    4'h5: controls = 9'b100001100; //JUMP
    4'h6: controls =   zero  ? 9'b100001100 : 9'b100000100; //JMPZ
    4'h7: controls = (~zero) ? 9'b100001100 : 9'b100000100; //JPNZ
    4'h8: controls = 9'b000000000; //ADD
    4'h9: controls = 9'b000000001; //SUB
    4'hA: controls = 9'b000000010; //INAC
    4'hB: controls = 9'b000000011; //CLAC
    4'hC: controls = 9'b000000100; //AND
    4'hD: controls = 9'b000000101; //OR
    4'hE: controls = 9'b000000110; //XOR
    4'hF: controls = 9'b000000111; //NOT
endcase
endmodule
```

Control unit

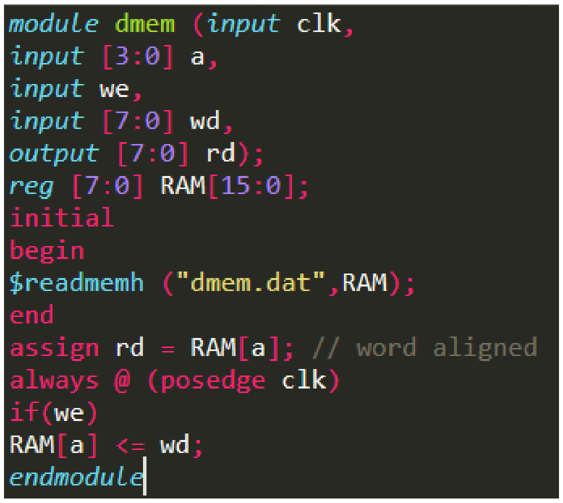


```verilog
module datapath (input clk, reset,
input nop,
input load, store,
input mva, mvr,
input jump,
input [2:0] alucontrol,
output zero,
output [3:0] pc,
input [7:0] instr,
output [7:0] AC);

wire AC_clk;
wire [7:0] aluout;
assign AC_clk = ~nop & clk;
wire [3:0] pcnext, pcplus1;
wire [7:0] acnext, ldout, R, datamem;
wire zeronext;

// next PC logic
flopr #(4) pcreg(clk, reset, pcnext, pc);
adder pcadd1 (pc, 4'h1, pcplus1);
mux2 #(4) pcmux(pcplus1, instr[3:0], jump, pcnext);
// register file logic
flopr #(8) AC_reg(AC_clk, reset, acnext, AC);
flopr #(8) R_reg (mva, reset, AC, R);
mux2 #(8) ldmux(aluout, datamem, load, ldout);
mux2 #(8) mvmux(ldout, R, mvr, acnext);
dmem dmem (clk, instr[3:0], store, AC,datamem);
// ALU logic
alu alu(AC, R, alucontrol, aluout, zeronext);
flopr #(1) zeroreg(clk, reset, zeronext, zero);
endmodule
```

Datapath stitched together

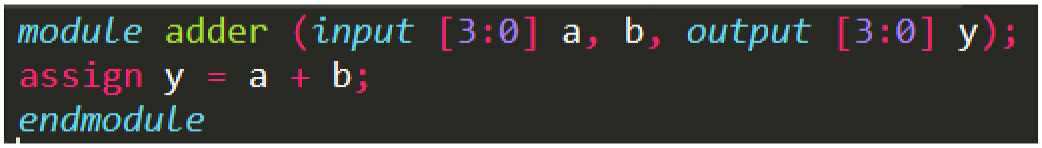


```verilog
module imem (input [3:0] a, output [7:0] rd);
reg [7:0] RAM[15:0]; // limited memory
initial
begin
$readmemh ("imem.dat",RAM);
end
assign rd = RAM[a]; // word aligned
endmodule
```

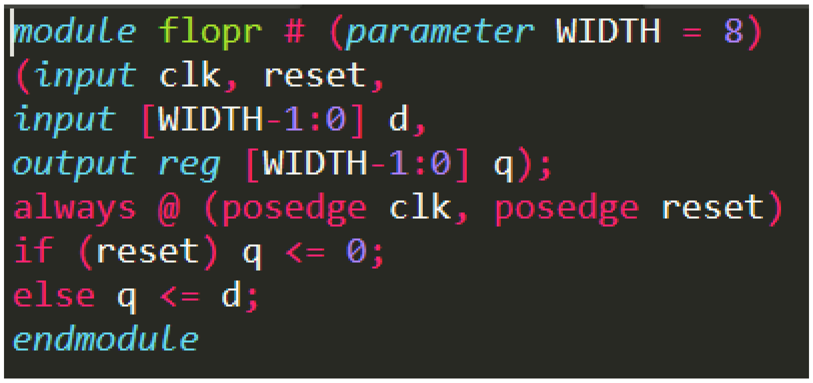


```verilog
module dmem (input clk,
input [3:0] a,
input we,
input [7:0] wd,
output [7:0] rd);
reg [7:0] RAM[15:0];
initial
begin
$readmemh ("dmem.dat",RAM);
end
assign rd = RAM[a]; // word aligned
always @ (posedge clk)
if(we)
RAM[a] <= wd;
endmodule
```

Cache memories

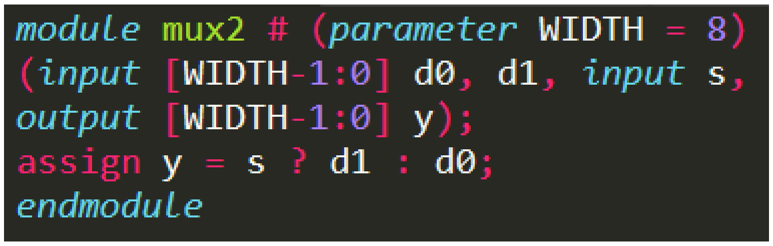


```verilog
module adder (input [3:0] a, b, output [3:0] y);
assign y = a + b;
endmodule
```

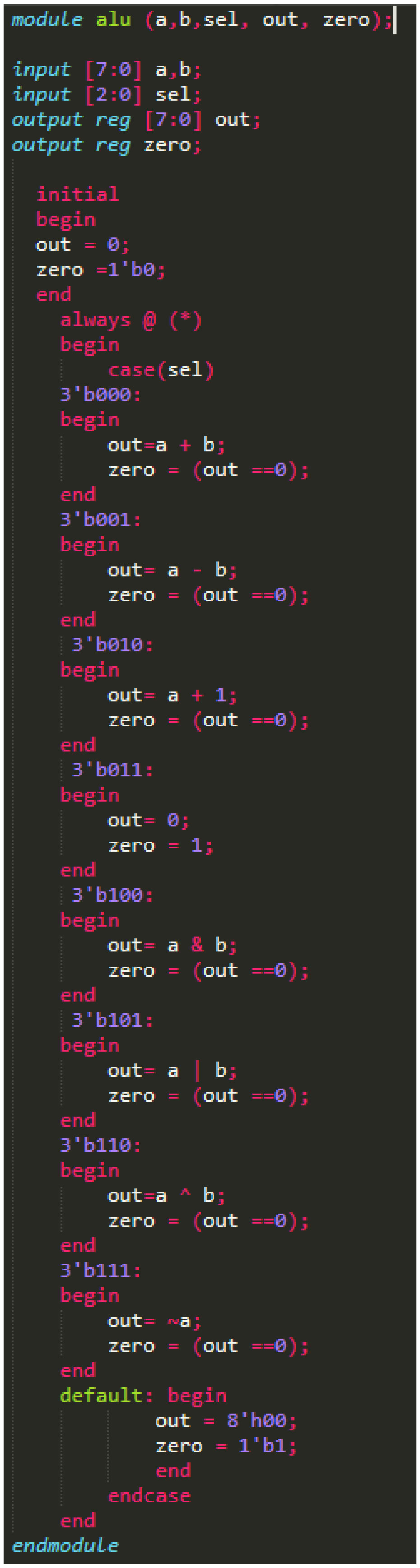


```verilog
module flopr # (parameter WIDTH = 8)
(input clk, reset,
input [WIDTH-1:0] d,
output reg [WIDTH-1:0] q);
always @ (posedge clk, posedge reset)
if (reset) q <= 0;
else q <= d;
endmodule
```

Supporting modules

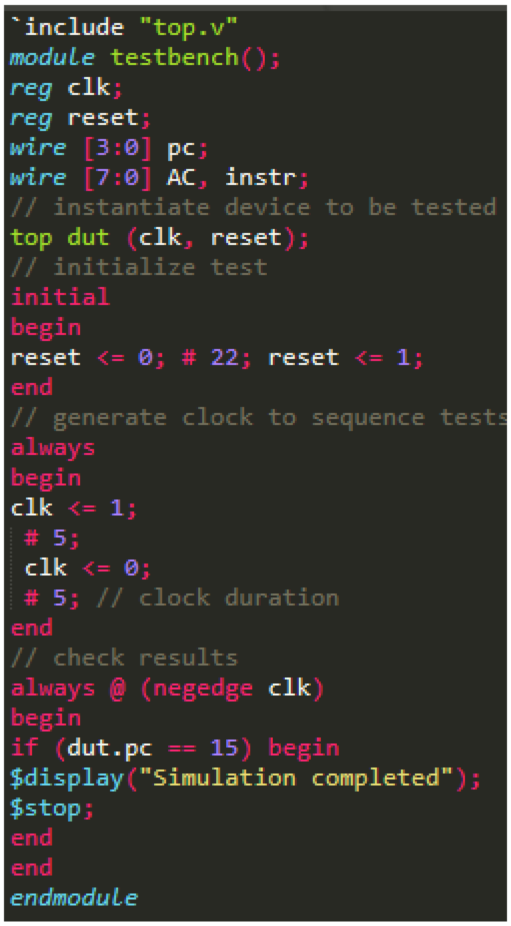


```verilog
module mux2 # (parameter WIDTH = 8)
(input [WIDTH-1:0] d0, d1, input s,
output [WIDTH-1:0] y);
assign y = s ? d1 : d0;
endmodule
```



```
module alu (a,b,sel, out, zero);

input [7:0] a,b;
input [2:0] sel;
output reg [7:0] out;
output reg zero;

  initial
  begin
  out = 0;
  zero =1'b0;
  end
    always @ (*)
    begin
        case(sel)
    3'b000:
    begin
        out=a + b;
        zero = (out ==0);
    end
    3'b001:
    begin
        out= a - b;
        zero = (out ==0);
    end
     3'b010:
    begin
        out= a + 1;
        zero = (out ==0);
    end
     3'b011:
    begin
        out= 0;
        zero = 1;
    end
     3'b100:
    begin
        out= a & b;
        zero = (out ==0);
    end
     3'b101:
    begin
        out= a | b;
        zero = (out ==0);
    end
    3'b110:
    begin
        out=a ^ b;
        zero = (out ==0);
    end
    3'b111:
    begin
        out= ~a;
        zero = (out ==0);
    end
    default: begin
            out = 8'h00;
            zero = 1'b1;
            end
        endcase
    end
endmodule
```

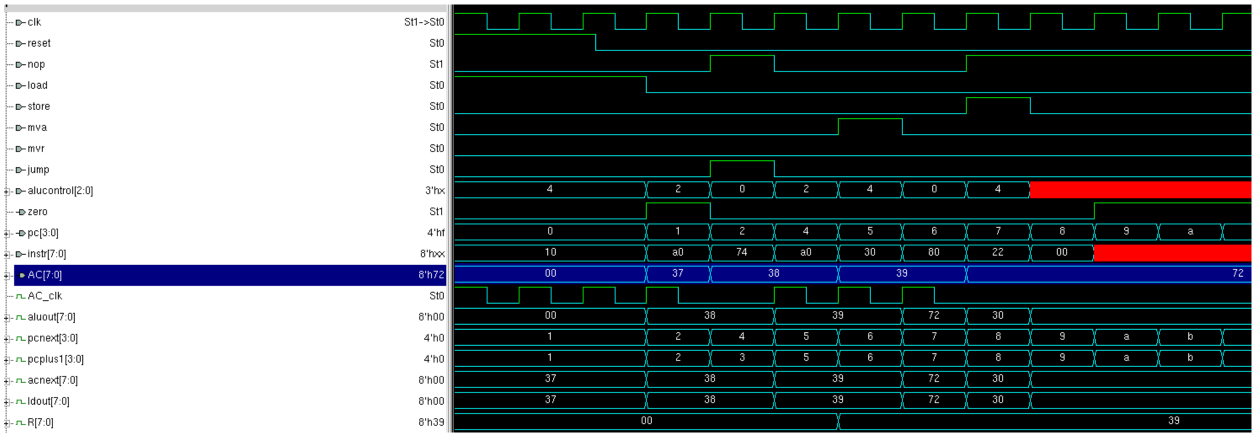
ALU



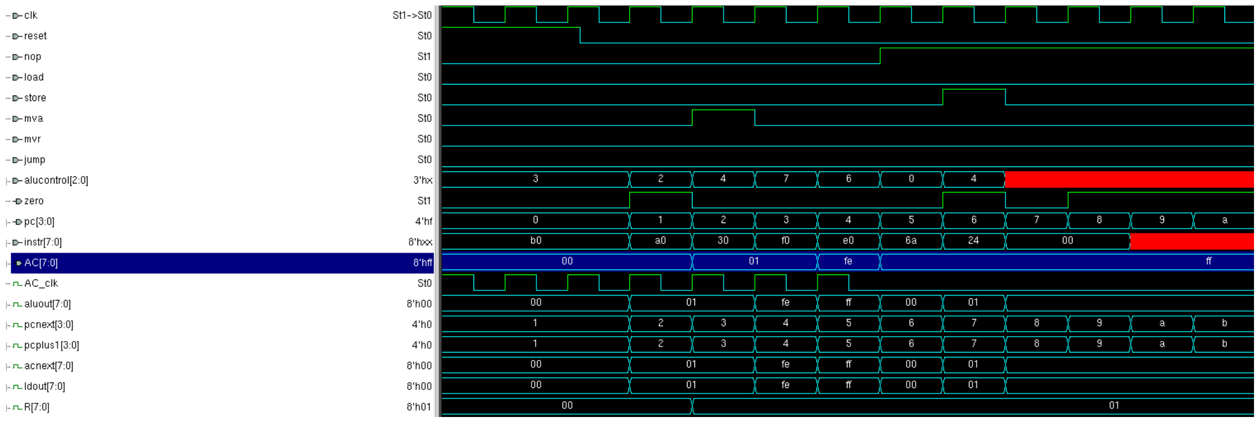
```
`include "top.v"
module testbench();
reg clk;
reg reset;
wire [3:0] pc;
wire [7:0] AC, instr;
// instantiate device to be tested
top dut (clk, reset);
// initialize test
initial
begin
reset <= 0; # 22; reset <= 1;
end
// generate clock to sequence tests
always
begin
clk <= 1;
 # 5;
 clk <= 0;
 # 5; // clock duration
end
// check results
always @ (negedge clk)
begin
if (dut.pc == 15) begin
$display("Simulation completed");
$stop;
end
end
endmodule
```

Testbench, values were verified in waveform

Waveforms:



Accumulator ends at 0x72 = 114 = 55 + 1 + 1 + (57) CORRECT!

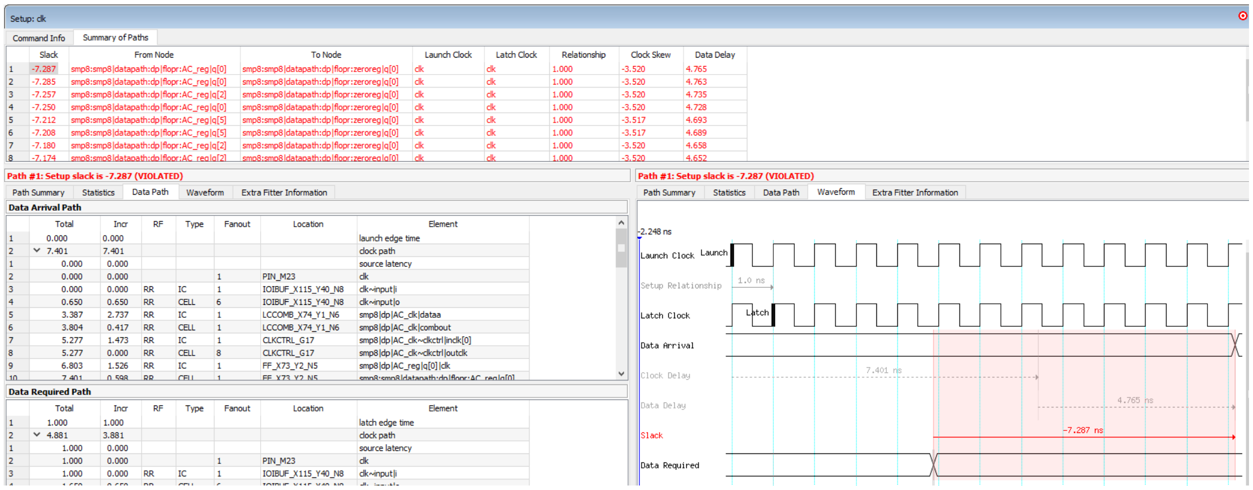


Accumulator ends at 0xFF = 255 = ~(0+1) ^ 1 CORRECT!

[Test Code 1 Video Demo]

[Test Code 2 Video Demo]

Timing:



Setup: clk

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-7.287	smp8:smp8|datapath:dp|flopr:AC_reg|q[0]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	1.000	-3.520	4.765
2	-7.285	smp8:smp8|datapath:dp|flopr:AC_reg|q[0]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	1.000	-3.520	4.763
3	-7.257	smp8:smp8|datapath:dp|flopr:AC_reg|q[2]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	1.000	-3.520	4.735
4	-7.250	smp8:smp8|datapath:dp|flopr:AC_reg|q[0]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	1.000	-3.520	4.728
5	-7.212	smp8:smp8|datapath:dp|flopr:AC_reg|q[5]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	1.000	-3.517	4.693
6	-7.208	smp8:smp8|datapath:dp|flopr:AC_reg|q[5]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	1.000	-3.517	4.689
7	-7.180	smp8:smp8|datapath:dp|flopr:AC_reg|q[2]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	1.000	-3.520	4.658
8	-7.174	smp8:smp8|datapath:dp|flopr:AC_reg|q[2]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	1.000	-3.520	4.652

Path #1: Setup slack is -7.287 (VIOLATED)

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Data Arrival Path

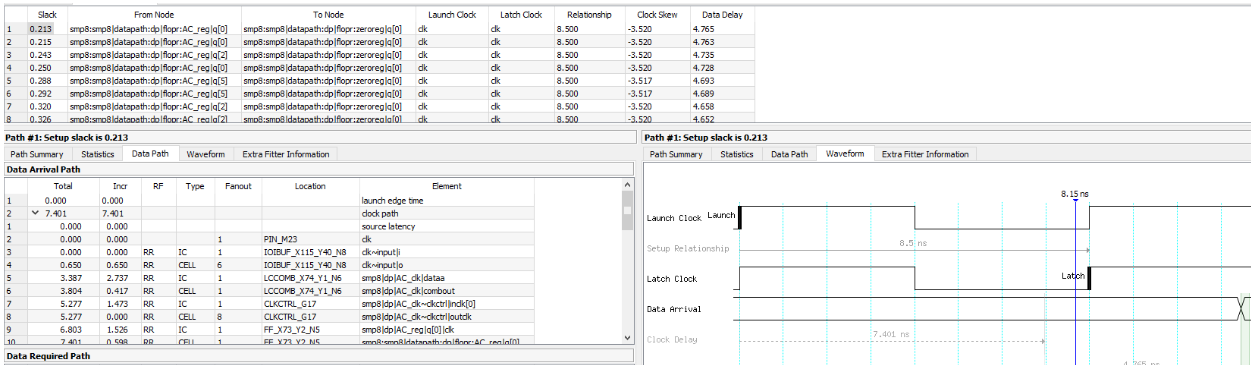
	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	7.401	7.401					clock path
1	0.000	0.000					source latency
2	0.000	0.000			1	PIN_M23	clk
3	0.000	0.000	RR	IC	1	IOIBUF_X115_Y40_N8	clk~input|i
4	0.650	0.650	RR	CELL	6	IOIBUF_X115_Y40_N8	clk~input|o
5	3.387	2.737	RR	IC	1	LCCOMB_X74_Y1_N6	smp8|dp|AC_clk|dataa
6	3.804	0.417	RR	CELL	1	LCCOMB_X74_Y1_N6	smp8|dp|AC_clk|combout
7	5.277	1.473	RR	IC	1	CLKCTRL_G17	smp8|dp|AC_clk~clkctrl|inclk[0]
8	5.277	0.000	RR	CELL	8	CLKCTRL_G17	smp8|dp|AC_clk~clkctrl|outclk
9	6.803	1.526	RR	IC	1	FF_X73_Y2_N5	smp8|dp|AC_reg|q[0]|clk
10	7.401	0.598	RR	CELL	1	FF_X73_Y2_N5	smp8:smp8|datapath:dp|flopr:AC_reg|q[0]

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	1.000	1.000					latch edge time
2	4.881	3.881					clock path
1	1.000	0.000					source latency
2	1.000	0.000			1	PIN_M23	clk
3	1.000	0.000	RR	IC	1	IOIBUF_X115_Y40_N8	clk~input|i

Path #1: Setup slack is -7.287 (VIOLATED)

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information



	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	0.213	smp8:smp8|datapath:dp|flopr:AC_reg|q[0]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	8.500	-3.520	4.765
2	0.215	smp8:smp8|datapath:dp|flopr:AC_reg|q[0]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	8.500	-3.520	4.763
3	0.243	smp8:smp8|datapath:dp|flopr:AC_reg|q[2]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	8.500	-3.520	4.735
4	0.250	smp8:smp8|datapath:dp|flopr:AC_reg|q[0]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	8.500	-3.520	4.728
5	0.288	smp8:smp8|datapath:dp|flopr:AC_reg|q[5]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	8.500	-3.517	4.693
6	0.292	smp8:smp8|datapath:dp|flopr:AC_reg|q[5]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	8.500	-3.517	4.689
7	0.320	smp8:smp8|datapath:dp|flopr:AC_reg|q[2]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	8.500	-3.520	4.658
8	0.326	smp8:smp8|datapath:dp|flopr:AC_reg|q[2]	smp8:smp8|datapath:dp|flopr:zeroreg|q[0]	clk	clk	8.500	-3.520	4.652

Path #1: Setup slack is 0.213

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	7.401	7.401					clock path
1	0.000	0.000					source latency
2	0.000	0.000			1	PIN_M23	clk
3	0.000	0.000	RR	IC	1	IOIBUF_X115_Y40_N8	clk~input|i
4	0.650	0.650	RR	CELL	6	IOIBUF_X115_Y40_N8	clk~input|o
5	3.387	2.737	RR	IC	1	LCCOMB_X74_Y1_N6	smp8|dp|AC_clk|dataa
6	3.804	0.417	RR	CELL	1	LCCOMB_X74_Y1_N6	smp8|dp|AC_clk|combout
7	5.277	1.473	RR	IC	1	CLKCTRL_G17	smp8|dp|AC_clk~clkctrl|inclk[0]
8	5.277	0.000	RR	CELL	8	CLKCTRL_G17	smp8|dp|AC_clk~clkctrl|outclk
9	6.803	1.526	RR	IC	1	FF_X73_Y2_N5	smp8|dp|AC_reg|q[0]|clk
10	7.401	0.598	RR	CELL	1	FF_X73_Y2_N5	smp8:smp8|datapath:dp|flopr:AC_reg|q[0]

Data Required Path

Path #1: Setup slack is 0.213

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Timing was fixed by changing clock to have **8.5 ns period**. Both test code 1 and 2 contain 9 instructions, each taking a single cycle, so 8.5ns*0 = **76.5ns execution time**

3. Conclusions & Summary

In conclusion, this project was an appropriate culmination of our studies of computer architecture. We were tasked with building an 8-bit processor in Verilog by first designing a block diagram that supports all the instructions specified, then implementing it in code and testing its operation on an FPGA development board and in simulation with waveforms. After putting a good effort into the block diagram, the code was the easy part. No changes were made to the block diagram after the drawing was finalized, meaning it worked exactly as designed which is something I am proud of. The project as a whole is worthy to go on my resume and I look forward to increasing my knowledge of computer architecture and possibly pursue a career in it.