ELC 2137 Lab 09: ALU with Input Register

Justin Woods

April 1, 2020

Summary

The lab involved using combinational logic and sequential logic. By using sequential logic, it was possible to store values from previous inputs, as opposed to combinational logic when the output only depends on the present input. First, a D Register was created and tested using a parameter defining the bits input and output. Next an ALU was created(the combinational part of the lab), so that different arithmetic functions could be performed on the values to be input. Lastly, these modules were combined to create a Top Level ALU with a Register input.

Results

Time (ns):	0-5	5-10	10-15	15-20	20-25	25-30	30-35	35-40	40-45	45-50	50 55
D (hex)	0	0	A	A	3	3	0	0	0→6	6	6
clk	0	1	0	1	0	1	0	1	0	1	0
en	0	0	1	1	1→0	0→1	1→0	0	0→1	1	1
rst	0	0→1	0	0	0	0	0	0	0	0	0
Q(hex)	X	0→0	0	A	A	A	A	A	A	6	6

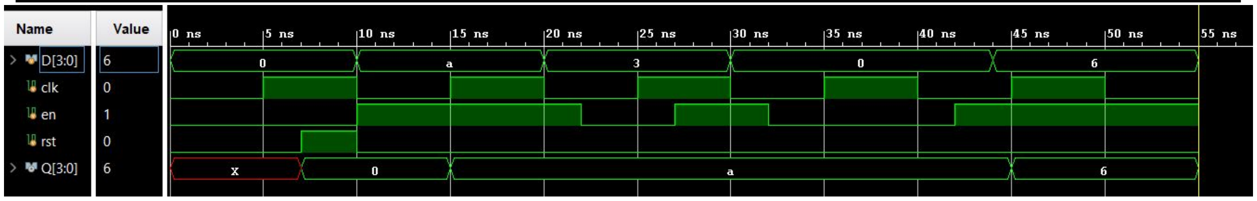


Figure 1: Register Simulation Waveform and ERT

Time (ns):	0-10	10-20	20-30	30-40	40-50	50-60
in0	4	14	1	1	1	8
in1	10	5	0	0	1	0
op	0	1	2	3	4	5
out	E	9	0	1	0	8

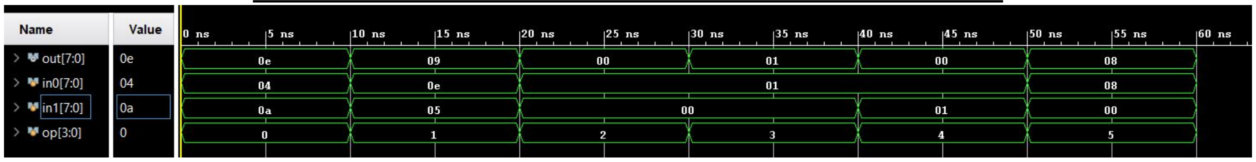
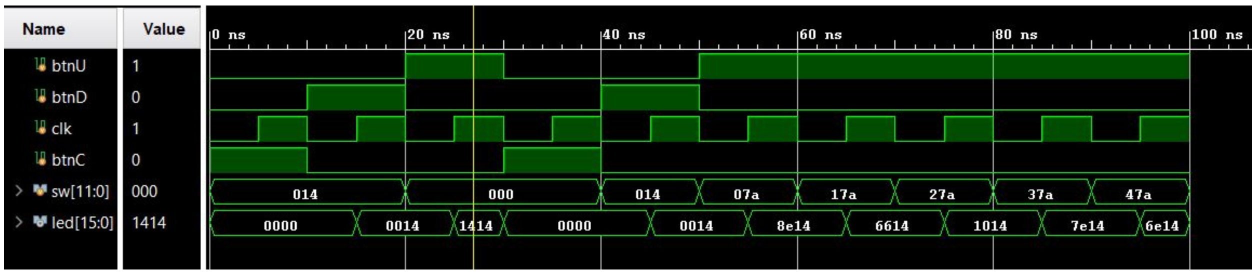


Figure 2: ALU Simulation Waveform and ERT



Name
Value
btnU 1
btnD 0
clk 1
btnC 0
sw[11:0] 000
led[15:0] 1414
0 ns 20 ns 40 ns 60 ns 80 ns 100 ns
014 000 014 07a 17a 27a 37a 47a
0000 0014 1414 0000 0014 8e14 6614 1014 7e14 6e14

Figure 3: Top Level Waveform

Code

Listing 1: Register Source Code

```
module register #(parameter N=1)
   (
   input clk, rst, en, input [N-1:0] D,
   output reg [N-1:0] Q
   );

   always @(posedge clk, posedge rst)
   begin
      if (rst==1)
         Q <= 0 ;
      else if (en==1)
         Q <= D ;
   end
endmodule
```

Listing 2: Register Test

```
module registerTEST();

   reg [3:0] D;
   reg clk, en, rst;
   wire [3:0] Q;

   register #(.N(4)) r(.D(D), .clk(clk),
      .en(en), .rst(rst),
      .Q(Q) );

   always begin
      clk = ~clk; #5;
   end

   // this block only runs once
   initial begin
      clk=0; en=0; rst=0; D=4'h0; #7;
      rst = 1; #3;
      // reset
      D = 4'hA; en = 1; rst = 0; #10;
      D = 4'h3;   #2;
      en = 0;     #5;
      en = 1;     #3;
      D = 4'h0;   #2;
      en = 0;     #10;
      en = 1;     #2;
      D = 4'h6;   #11;
      $finish;
   end
endmodule
```

Listing 3: ALU Source Code

```
module alu #(parameter N=8)
   (
   output reg[N-1:0] out,
   input [N-1:0] in0,
   input [N-1:0] in1,
   input [3:0] op
   );

   // Local parameters
   parameter ADD =0;
   parameter SUB =1;
   parameter AND =2;
   parameter OR  =3;
   parameter XOR =4;

   always @*
   begin
      case(op)
         ADD: out = in0 + in1;
         SUB: out = in0 - in1;
         AND: out = in0 & in1;
         OR: out = in0 | in1;
         XOR: out = in0 ^ in1;
         default: out = in0;
      endcase
   end
endmodule
```

Listing 4: ALU Test

```
module aluTEST();

   wire[7:0] out;
   reg [7:0] in0, in1;
   reg [3:0] op;

   alu #(.N(8)) a(.in0(in0), .in1(in1), .op(op),
      .out(out) );

   initial begin
      op = 0; in0 = 4; in1 = 10; #10
      op = 1; in0 = 14; in1 = 5; #10
      op = 2; in0 = 1; in1 = 0; #10
      op = 3; in0 = 1; in1 = 0; #10
      op = 4; in0 = 1; in1 = 1; #10
      op = 5; in0 = 8; in1 = 0; #10
      $finish;
   end
endmodule
```

Listing 5: Top Level Source Code

```
module top_lab9(
   input btnU, btnD, clk, btnC,
   input [15:0]sw,
   output [15:0] led
   );

   wire [7:0]reg1out, a1out, reg2out;

   register #(.N(8)) reg1 (.D(sw[7:0]), .clk(clk),
      .en(btnD), .rst(btnC),
      .Q(reg1out) );

   alu #(.N(8)) a1 (.in1(reg1out), .in0(sw[7:0]), .op(sw[11:8]),
      .out(a1out) );

   register #(.N(8)) reg2 (.D(a1out), .clk(clk),
      .en(btnU), .rst(btnC),
      .Q(reg2out) );

   assign led[7:0] = reg1out;
   assign led[15:8] = reg2out;

endmodule
```