Homework 5

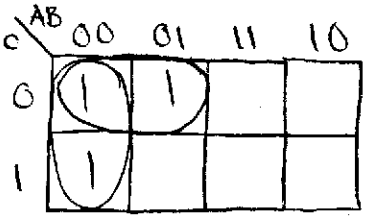
Rhea Mae Edwards
Student ID #
982-389-303

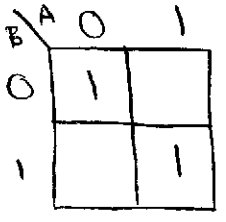
1) Combinational Logic in Verilog

4.1 Simplified Schematic of a Circuit

System Verilog

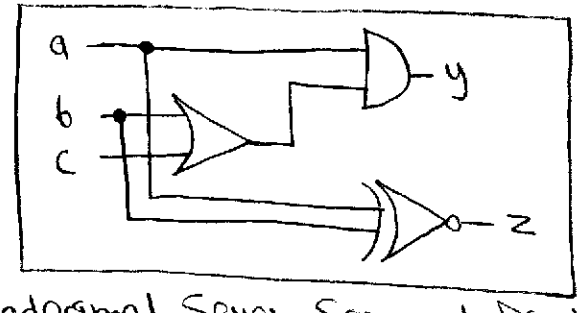
```
module exercise1 (input logic a, b, c,
                  output logic y, z);

  assign y = a & b & c | a & b & ~c | a & ~b & c;
  assign z = a & b | ~a & ~b;

endmodule
```





XOR Gate

$AC + AB$
$A(C + B)$



4.6 HDL Module Hexadecimal Seven-Segment Display Decoder

Should handle digits A, B, C, D, E, and F as well as 0 - 9

System Verilog

```
module display_decoder (input logic [3:0] data,
                        output logic [6:0] segments);

  always_comb  // always for combinational logic
    case (data)
      4'h0: segments = 7'b111_1110;   // abc_defg
      4'h1: segments = 7'b011_0000;
      4'h2: segments = 7'b110_1101;
      4'h3: segments = 7'b111_1001;
      4'h4: segments = 7'b011_0011;
      4'h5: segments = 7'b101_1011;
      4'h6: segments = 7'b101_1111;
```

```
        4'h7: segments = 7'b 111_0000;
        4'h8: segments = 7'b 111_1111;
        4'h9: segments = 7'b 111_0011;
        4'ha: segments = 7'b 111_0111;
        4'hb: segments = 7'b 001_1111;
        4'hc: segments = 7'b 000_1101;
        4'hd: segments = 7'b 011_1101;
        4'he: segments = 7'b 100_1111;
        4'hf: segments = 7'b 100_0111;
    endcase
endmodule
```

2) Sequential Logic in Verilog

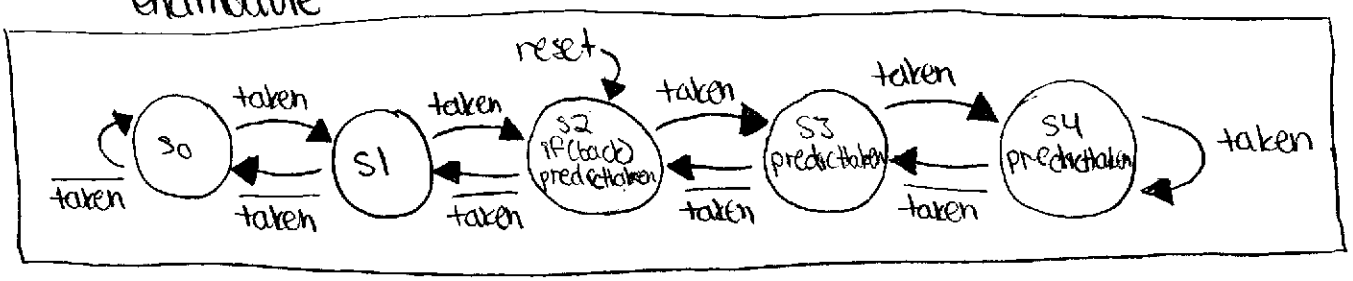
4.25 State Transition Diagram of a Described FSM

SystemVerilog

```
module fsm1 (input logic clk, reset,
             input logic taken, back,
             output logic predicttaken);

  logic [4:0] state, nextstate;

  parameter S0 = 5'b 00001;
  parameter S1 = 5'b 00010;
  parameter S2 = 5'b 00100;
  parameter S3 = 5'b 01000;
  parameter S4 = 5'b 10000;

  always_ff @(posedge clk, posedge reset)
    if (reset) state <= S2;
    else       state <= nextstate;

  always_comb
    case (state)
      S0: if (taken)  nextstate = S1;
          else        nextstate = S0;
      S1: if (taken)  nextstate = S2;
          else        nextstate = S0;
      S2: if (taken)  nextstate = S3;
          else        nextstate = S1;
      S3: if (taken)  nextstate = S4;
          else        nextstate = S2;
      S4: if (taken)  nextstate = S4;
          else        nextstate = S3;
      default:        nextstate = S2;
    endcase
```

```
assign predicttaken = (state == S4) |
                      (state == S3) |
                      (state == S2 && back);
```

endmodule



3) Interview Questions

4.2 Blocking vs Nonblocking in SystemVerilog

HDLs support blocking and nonblocking assignments in an always / process statement. A group of blocking assignments are evaluated in the order they appear in the code, just as one would expect in a standard programming language. A group of nonblocking assignments are evaluated concurrently; all of the statements are evaluated before any of the left hand sides are updated.

SystemVerilog

always statement
- = blocking assignment
- <= nonblocking assignment