CS/EE 3700
Spring 2019
Myers

Final Exam

- Fill in your name:
- This exam is closed book and open notes.
- Notes must be things you created, and it may include copies of your assignments and labs.
- One exception is the midterm/final review slides, which are also permitted.
- The exam is 120 minutes and worth 200 points.
- Show all your work.

Question	Score
1(a)	
1(b)	
1(c)	
1(d)	
1(e)	
2(a)	
2(b)	
2(c)	
2(d)	
2(e)	
Total	

Mean = 156.4
Standard Deviation = 30.3

1. **Digital System Design**

In this problem, you are going to be designing a circuit to compute the greatest common divisor (GCD) of two numbers. In the initial state, it should load A and B into 16-bit registers and wait until it receives a start signal s. Once it sees s go high, it should execute the algorithm shown below. Once this algorithm completes, it should assert the *done* signal high and wait until s goes low. Note that the outputs of the A and B registers (i.e., Aout and Bout, respectively) will be outputs that the user of the GCD calculator can use to observe the result. Finally, once it sees s go low, it should return to the initial state to wait for the next request for computation.

```
while (A ≠ B) do
  if (A > B) then
    A = A - B
  else
    B = B - A
  end if
end while
```

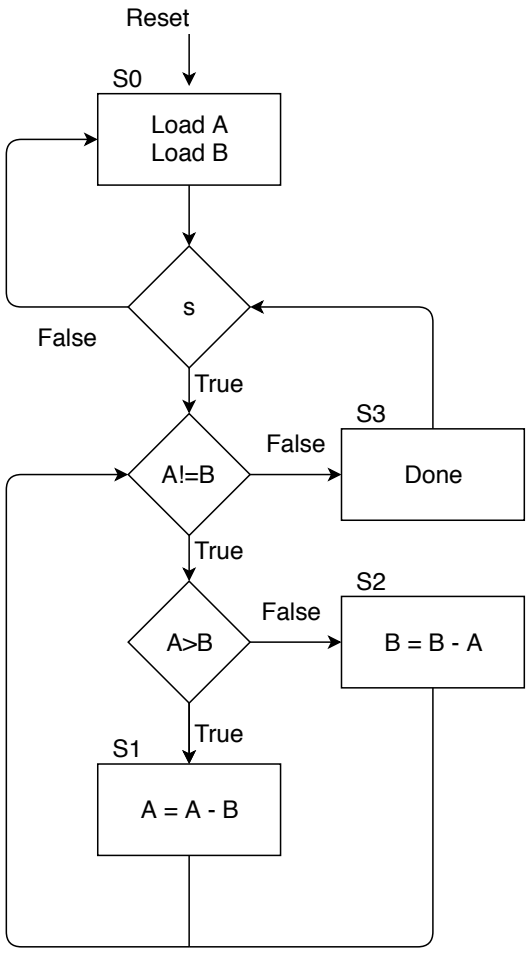
(a) **ASM Charts** (20 points)

Draw a high-level ASM chart for the GCD calculator described above.

8 points - States

8 points - Decisions

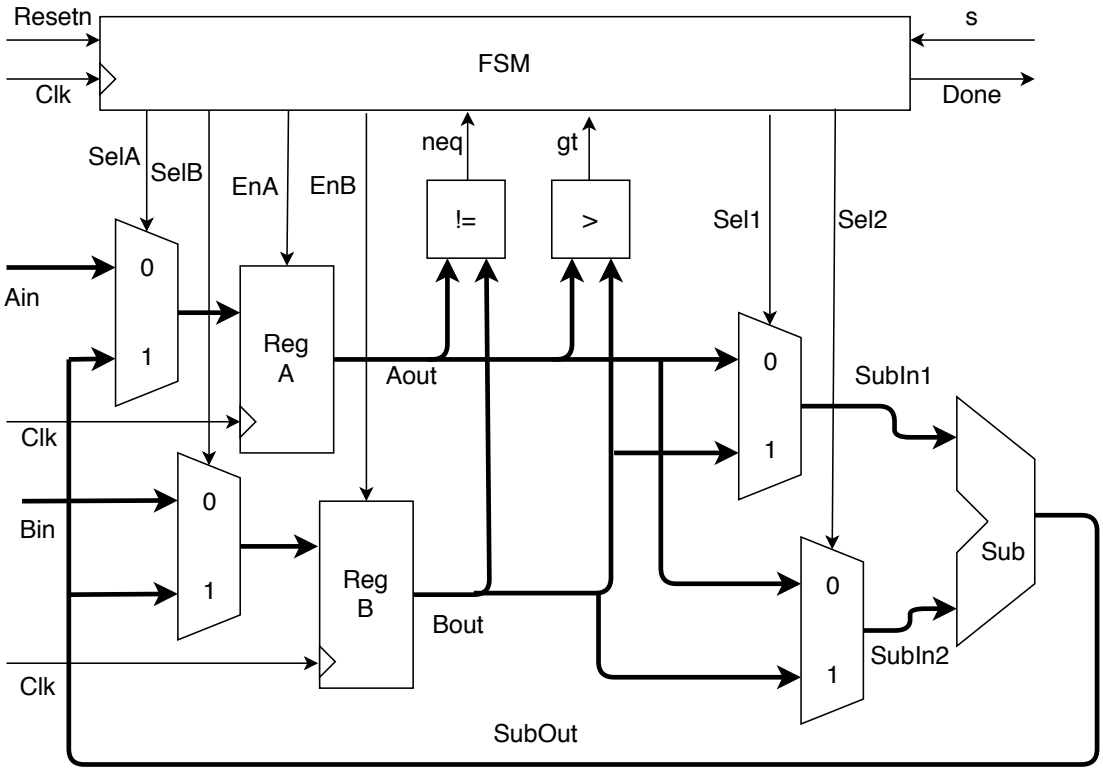
4 points - Other Details



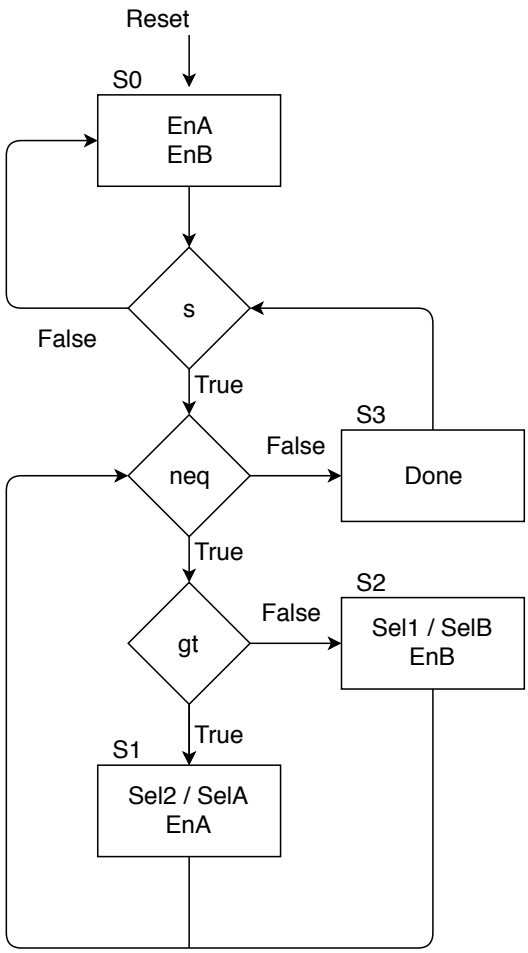
(b) **Datapath Design** (20 points)

Draw a block diagram for the datapath for the GCD calculator. Other than your FSM, your block diagram should include only basic blocks such as registers, multiplexors, arithmetic circuits, comparators, etc.

4 points - Register A
4 points - Register B
4 points - Arithmetic
4 points - Comparators
4 points - FSM



(c) **FSM Design** (20 points)
Draw a ASM chart for the FSM controller for the GCD calculator.
4 points - Correct entry and exit points
4 points - Correct syntax and transitions
6 points - Correct states and flow
6 points - Correct signals in states



(d) **Behavioral Verilog** (30 points)
Write Verilog code for the FSM controller. Be careful that your code does not infer latches in the combinational portion.

```
module FSM(Clk, Resetn, s, neq, gt, EnA, EnB, SelA, SelB, Sel1, Sel2, Done)

  input Clock, Resetn, s, neq, gt; // 3 points declarations
  output reg EnA, EnB, SelA, SelB, Sel1, Sel2, Done;
  reg [1,0] PS, NS;

  parameter [1:0] S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11; // 2 points

  always@(posedge Clock, negedge Resetn) // 5 points
    if (Resetn) PS <= S0; else PS <= NS;

  always@(*) // 15 points + 5 for no inferred latches
  begin
    EnA = 0; EnB = 0; SelA = 0; SelB = 0; Sel1 = 0; Sel2 = 0; Done = 0;
    case(PS)
    S0: begin
      if (s && neq && gt) NS=S1;
      else if (s && neq) NS=S2;
      else if (s) NS=S3;
      else NS=S0;
      EnA = 1; EnB = 1;
    end
    S1: begin
      if (neq && gt) NS=S1;
      else if (neq) NS=S2;
      else NS=S3;
      SelA = 1; Sel2 = 1; EnA = 1;
    end
    S2: begin
      if (neq && gt) NS=S1;
      else if (neq) NS=S2;
      else NS=S3;
      SelB = 1; Sel1 = 1; EnB = 1;
    end
    default: begin
      if (s && neq && gt) NS=S1;
      else if (s && neq) NS=S2;
      else if (s) NS=S3;
      else NS=S0;
      Done = 1;
    end
    endcase
  end

endmodule
```

(e) **Structural Verilog** (30 points)
Write structural Verilog that connects up your datapath blocks to your FSM controller. You do not need to write Verilog code for the datapath blocks. You may assume they exist with the input and outputs shown in your datapath block diagram.

```
module gcd(Clk, Resetn, s, Ain, Bin, Aout, Bout, Done) // 2 points

  input Clk, Resetn, s; // 2 points
  input [15:0] Ain, Bin;

  output [15:0] Aout, Bout; // 2 points
  output Done;

  wire neq, gt, EnA, EnB, SelA, SelB, Sel1, Sel2;
  wire [15,0] AinM, BinM, SubIn1, SubIn2, SubOut; // 4 points

  // 8 points for FSM instantiation
  myFSM FSM(.Clk(Clk), .Resetn(Resetn), .s(s), .neq(neq), .gt(gt), .EnA(EnA), .EnB(EnB),
    .SelA(SelA), .SelB(SelB), .Sel1(Sel1), .Sel2(Sel2), .Done(Done));

  // 10 points for datapath instantiations (with 2 points for explicit wiring)
  muxA mux(.sel(SelA), .in0(Ain), .in1(SubOut), .out(AinM));
  muxB mux(.sel(SelB), .in0(Bin), .in1(SubOut), .out(BinM));
  regA reg(.Clk(Clk), .En(EnA), .in(AinM), .out(Aout));
  regB reg(.Clk(Clk), .En(EnB), .in(BinM), .out(Bout));
  compN compNeq(.in0(Aout), .in1(Bout), .out(neq));
  compG compGT(.in0(Aout), .in1(Bout), .out(gt));
  muxS1 mux(.sel(Sel1), .in0(Aout), .in1(Bout), .out(SubIn1));
  muxS2 mux(.sel(Sel2), .in0(Aout), .in1(Bout), .out(SubIn2));
  sub1 sub(.in0(SubIn1), .in1(SubIn2), SubOut);

endmodule
```

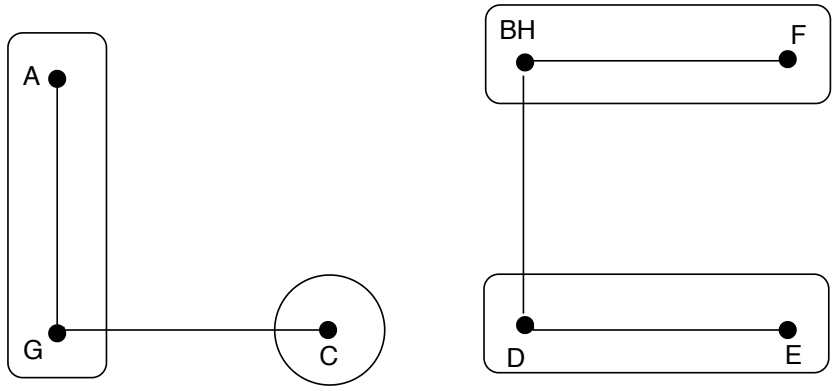
2. **Advanced Logic Design**
The questions in this problem relate to the flow table shown below.

Present State	Next state (w_2w_1) 00	01	10	11	Output z
A	–	F	G	Ⓐ	0
B	Ⓑ	F	D	–	1
C	B	Ⓒ	–	A	0
D	H	–	Ⓓ	E	1
E	–	C	D	Ⓔ	1
F	H	Ⓕ	–	A	1
G	B	–	Ⓖ	A	0
H	Ⓗ	F	D	–	1

(a) **State Minimization** (20 points)
Circle the stable states in the flow table above. Using the partitioning procedure and a merger diagram minimize the flow table above. Note that you must show all your steps to get full credit. Be sure to provide the minimized state table.
2 points for circling stable states correctly
4 points for partitioning

$$\begin{aligned} P_1 &= (A)(BH)(C)(G)(D)(E)(F) \\ P_2 &= P_1 \end{aligned}$$

10 points for the merger diagram



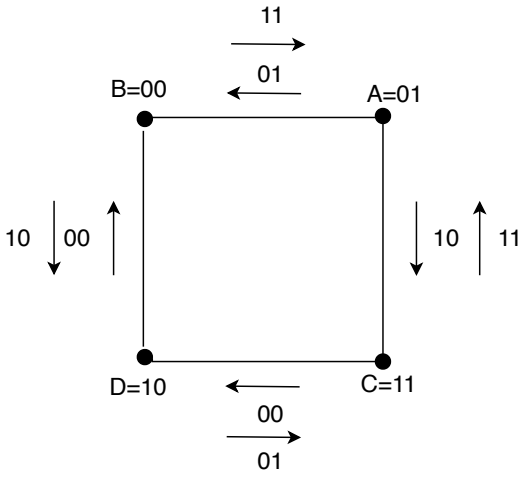
4 points for providing the minimized state table

Present State	Next state (w_2w_1) 00	01	10	11	Output z
A	B	B	Ⓐ	Ⓐ	0
B	Ⓑ	Ⓑ	D	A	1
C	B	Ⓒ	–	A	0
D	B	C	Ⓓ	Ⓓ	1

(b) **Critical Race Free State Assignment** (10 points)
A modified version of the minimized flow table from the last problem is shown below. Assuming a state assignment of $A = 01$, $B = 00$, $C = 11$, and $D = 10$, draw a transition diagram and briefly explain why this diagram proves that it is a critical race free state assignment.

Present State	Next state (w_2w_1) 00	01	10	11	Output z
A	–	B	C	A	0
B	B	B	D	A	1
C	D	C	C	A	0
D	B	C	D	D	1

8 points for the transition diagram



No diagonals, so no races (2 points)

(c) **Quine-McCluskey Logic Minimization** (30 points)
In this problem, consider the excitation table shown below. Use the Quine-McCluskey method to find the minimum two-level SOP implementation for **the next state variables** Y_2, NOT the output. HINT: use a Karnaugh map to check your answer. Be sure to provide the Boolean logic equation.

Present State y_2y_1	Next state (w_2w_1) 00 Y_2Y_1	01 Y_2Y_1	10 Y_2Y_1	11 Y_2Y_1	Output z
00	–	01	11	00	0
01	01	01	10	00	1
11	10	11	11	00	0
10	01	11	10	10	1

$$Y_2 = y_2\bar{y}_1w_1 + y_2y_1\bar{w}_2 + w_2\bar{w}_1$$

List 1

2	0010	√
6	0110	√
9	1001	√
10	1010	√
12	1100	√
11	1011	√
13	1101	√
14	1110	√

List 2

2,6	0x10	√
2,10	x010	√
6,14	x110	√
9,11	10x1	
9,13	1x01	
10,11	101x	
10,14	1x10	√
12,13	110x	
12,14	11x0	

List 3

2,6,10,14	xx10

Next is the covering table.

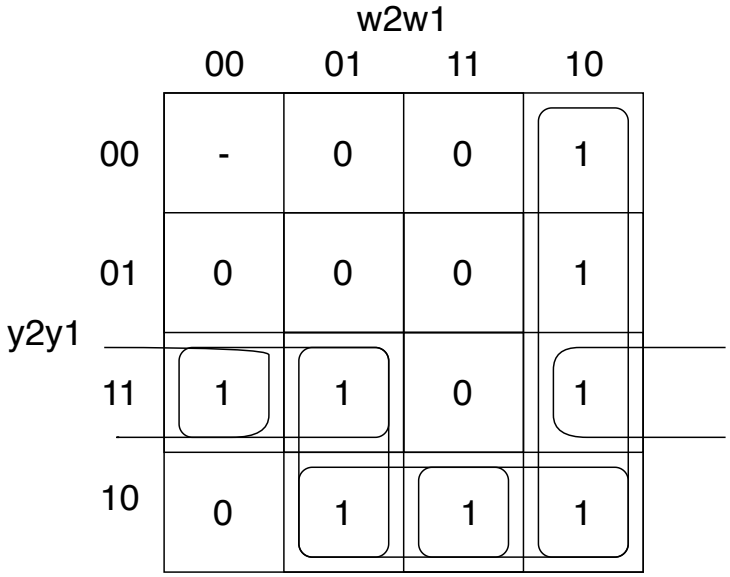
Prime Implicant			minterms 2	6	9	10	11	12	13	14
p1	(9,11)	10x1			√		√			
p2	(9,13)	1x01			√				√	
p3	(10,11)	101x				√	√			
p4	(12,13)	110x						√	√	
p5	(12,14)	11x0						√		√
p6	(2,6,10,14)	xx10	√	√		√				√

Only row p6 is essential.

Prime Implicant			minterms 9	11	12	13
p1	(9,11)	10x1	√	√		
p2	(9,13)	1x01	√			√
p3	(10,11)	101x		√		
p4	(12,13)	110x			√	√
p5	(12,14)	11x0			√	

Last, p1 and p4 can cover all the midterms of 9,11,12,13. So, the solution set is $\{p1, p4, p6\}$.
1) 5 points for each of the 3 lists.
2) 10 points for the cover table
3) 5 points for a correct equation corresponding to their cover table.

(d) **Hazard-free Logic Minimization** (10 points)
Derive a cover of the next state function for Y_2 using the Karnaugh map below which is free of static 1-hazards. Be sure to provide the Boolean logic equation.



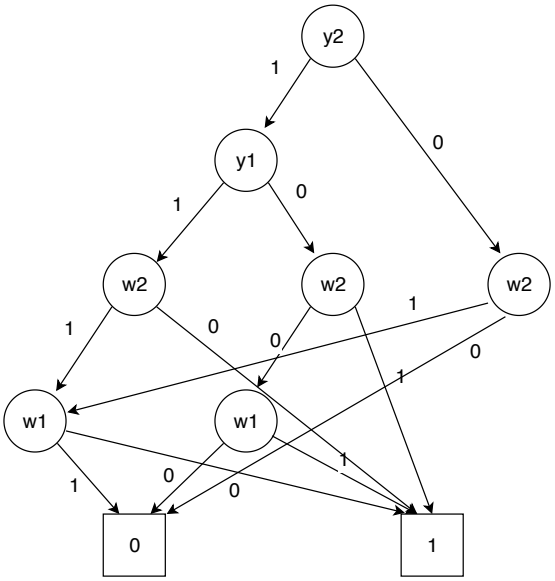
$$Y_2 \quad = \quad w_2\overline{w}_1 + y_2y_1\overline{w}_1 + w_2y_2\overline{y}_1 + y_2\overline{y}_1w_1 + \overline{w}_2w_1y_2 + y_2y_1\overline{w}_2$$

1.5 points for each cover
1 point for Y_2

(e) **BDDs** (10 points)
Using Shannon's expansion and the variable order y_2, y_1, w_2, w_1, derive a BDD for your answer to Problem 2(c).

5 points for the Diagram
5 points for Shannon's expansion

$$
\begin{aligned}
f &= y_2\bar{y}_1w_1 + y_2y_1\bar{w}_2 + w_2\bar{w}_1 \\
f_{\overline{y}_2} &= w_2\overline{w}_1 \\
f_{y_2} &= y_1\overline{w}_2 + \overline{y}_1w_1 + w_2\overline{w}_1 \\
f_{y_2\overline{y}_1} &= w_1 + w_2\overline{w}_1 \\
f_{y_2y_1} &= \overline{w}_2 + w_2\overline{w}_1 \\
f_{y_2\overline{y}_1\overline{w}_2} &= w_1 \\
f_{y_2\overline{y}_1w_2} &= 1 \\
f_{y_2y_1\overline{w}_2} &= 1 \\
f_{y_2y_1w_2} &= \overline{w}_1
\end{aligned}
$$



Extra Credit (10 points)
Repeat Problem 2(e) to derive a BDD for your answer to Problem 2(d).
5 points for the Diagram
5 points for Shannon's expansion

$$
\begin{aligned}
f &= w_2\overline{w}_1 + y_2y_1\overline{w}_1 + w_2y_2\overline{y}_1 + y_2\overline{y}_1w_1 + \overline{w}_2w_1y_2 + y_2y_1\overline{w}_2 \\
f_{y_2} &= w_2\overline{w}_1 + y_1\overline{w}_1 + w_2\overline{y}_1 + \overline{y}_1w_1 + \overline{w}_2w_1 + y_1\overline{w}_2 \\
f_{\overline{y}_2} &= w_2\overline{w}_1 \\
f_{y_2y_1} &= \overline{w}_1 + \overline{w}_2 \\
f_{y_2\overline{y}_1} &= w_2 + w_1
\end{aligned}
$$