FSM Assignment Jakob Lutch

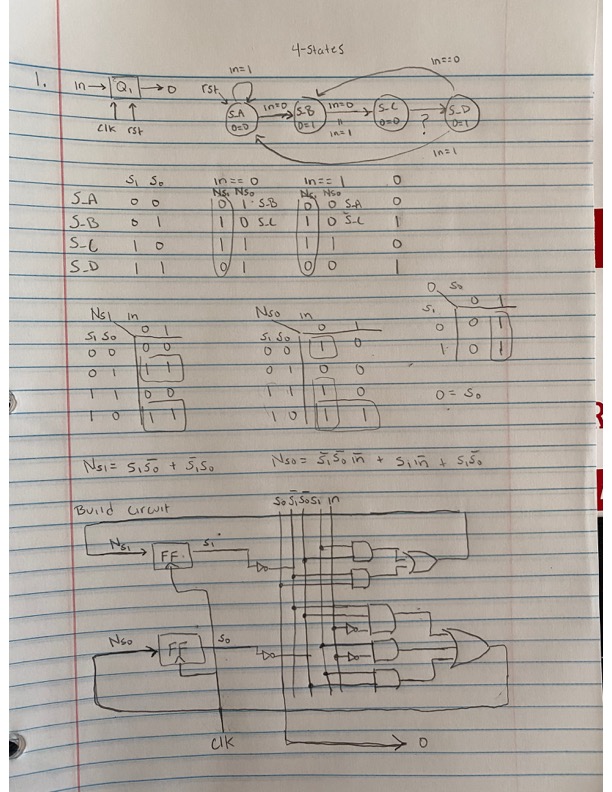
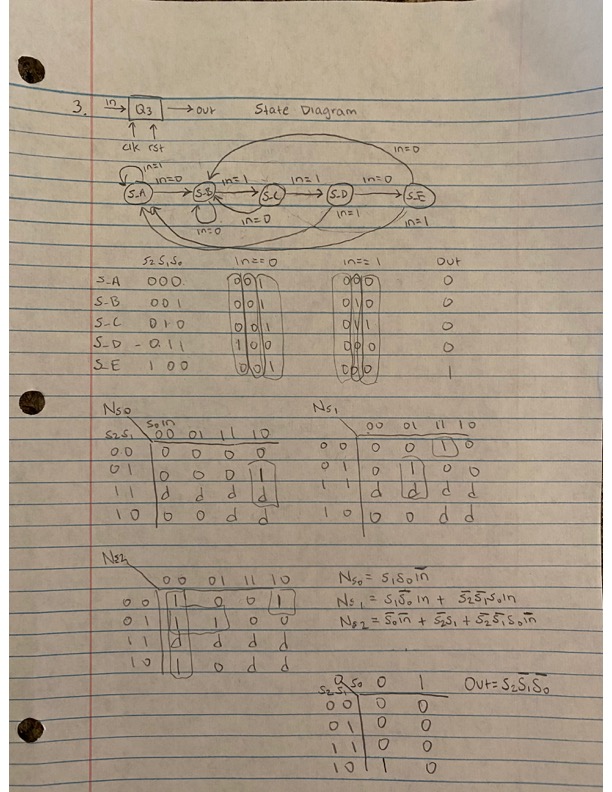
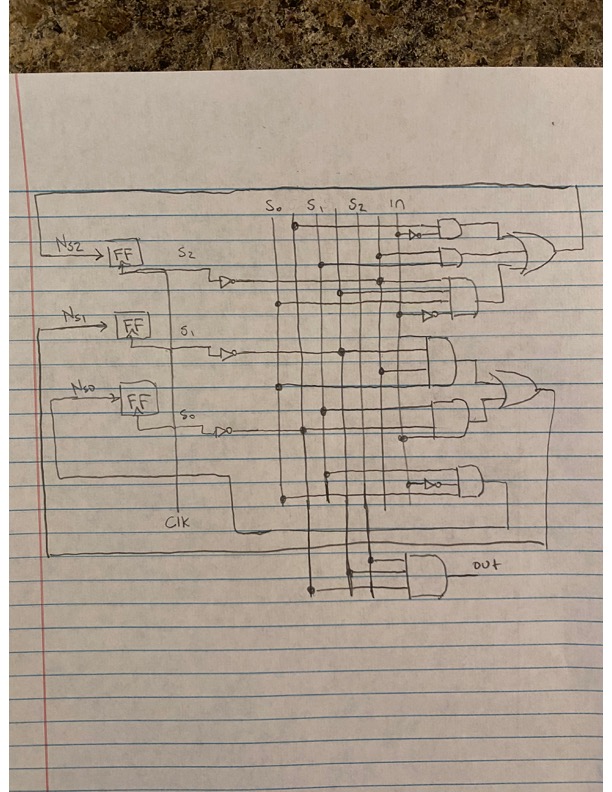


Fig1. Displays question 1 for the assignment.

1. Verilog code for question 1
2. module q1(
3. input clk, rst, in,
4. output reg o
5. );
6. reg [1:0] s;
7. reg [1:0] Ns;
8. parameter s\_A = 2'b00,
9. s\_B = 2'b01,
10. s\_C = 2'b10,
11. s\_D = 2'b11;
12. always@(posedge clk or negedge rst)
13. if(rst == 1'b0)
14. s <= s\_A;
15. else
16. s <= Ns;
18. always@(\*)
19. case(s)
20. s\_A: if(in == 1'b0)
21. Ns = s\_B;
22. else
23. Ns = s\_A;
24. s\_B: if(in == 1'b0)
25. Ns = s\_C;
26. else
27. Ns = s\_C;
28. s\_C: if(in == 1'b0)
29. Ns = s\_D;
30. else
31. Ns = s\_D;
32. s\_D: if(in == 1'b0)
33. Ns = s\_B;
34. else
35. Ns = s\_A;
36. default: Ns = s\_A;
37. endcase
39. always@(\*)
40. if(s == s\_B | s == s\_D)
41. o = 1'b1;
42. else
43. o = 1'b0;
45. Endmodule

3. Question 3





module q3(

input clk, rst, in,

output reg o

);

reg [2:0] s;

reg [2:0] Ns;

parameter s\_A = 3'b000,

s\_B = 3'b001,

s\_C = 3'b010,

s\_D = 3'b011,

s\_E = 3'b100;

always@(posedge clk or negedge rst)

if(rst == 1'b0)

s <= s\_A;

else

s <= Ns;

always@(\*)

case(s)

s\_A: if(in == 1'b0)

Ns = s\_B;

else

Ns = s\_A;

s\_B: if(in == 1'b0)

Ns = s\_B;

else

Ns = s\_C;

s\_C: if(in == 1'b0)

Ns = s\_B;

else

Ns = s\_D;

s\_D: if(in == 1'b0)

Ns = s\_E;

else

Ns = s\_A;

s\_E: if(in == 1'b0)

Ns = s\_B;

else

Ns = s\_A;

default: Ns = s\_A;

endcase

always@(\*)

if(s == s\_E)

o = 1'b1;

else

o = 1'b0;

endmodule

\*Verilog code for question 3

5. State Diagram

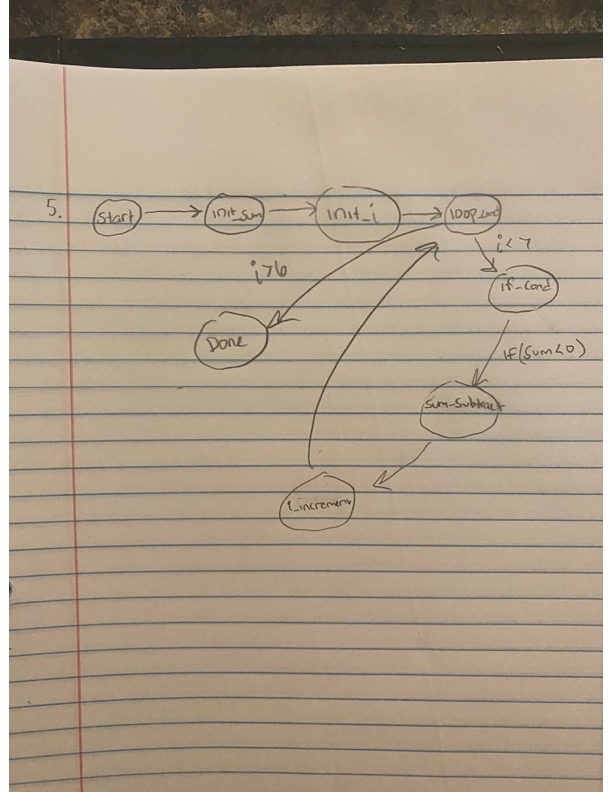


Fig. General state diagram for 5.

module q1(

input clk,

input rst,

input [7:0] input\_1,

input [7:0] input\_2,

output reg [7:0] sum,

output reg done

);

reg [3:0] s;

reg [3:0] Ns;

parameter START = 4'd0,

init\_sum = 4'd1,

init\_i = 4'd2,

loop\_cond = 4'd3,

if\_cond = 4'd4,

sum\_subtract = 4'd5,

i\_increment = 4'd6,

DONE = 4'd7,

ERROR = 4'hF;

reg [7:0] loop\_thresh;

reg [7:0] i;

always@(posedge clk or negedge rst)

if(rst == 1'b0)

s <= START;

else

s <= Ns;

always@(\*)

begin

case(s)

START:

begin

Ns = init\_sum;

end

init\_sum:

begin

Ns = init\_i;

end

init\_i:

begin

Ns = loop\_cond;

end

loop\_cond:

begin

if(i < loop\_thresh)

Ns = if\_cond;

else

Ns = DONE;

end

if\_cond:

begin

if(sum < 1'b0)

Ns = sum\_subtract;

end

sum\_subtract:

begin

Ns = i\_increment;

end

i\_increment:

begin

Ns = loop\_cond;

end

DONE:

begin

Ns = DONE;

end

default: Ns = ERROR;

endcase

end

always@(posedge clk or negedge rst)

begin

if(rst == 1'b0)

begin

i <= 8'd0;

loop\_thresh <= 8'd0;

sum <= 8'd0;

done <= 1'b0;

end

case(s)

init\_sum:

begin

sum <= 8'd0;

done <= 1'b0;

end

init\_i:

begin

i <= 1'b0;

loop\_thresh <= 4'b0111;

end

sum\_subtract:

begin

sum <= sum-i;

end

i\_increment:

begin

i <= i + 1'b1;

end

DONE:

begin

done <= 1'b1;

end

default: Ns = ERROR;

endcase

end

endmodule

Verilog Implementation for 5