ECE 211_21: Digital Circuits I

Practice Set Solutions

Notes:

- Apart from these questions, please do go through all the homework questions, short test questions and practice set questions provided for the two short tests.
- Final Exam is cumulative, so the questions asked in the final will be amongst any of the topics covered in the class.

Questions:

Signed Numbers and Binary Arithmetic

1 (a) Convert the following six-bit two's complement number to decimal:

Answer:

110101 = <u>-11</u>

```
 001010
      1
 ------
   1011
```

(b) The state of a 12-bit register is 010110010111. What is the content stored in the register if it is represented as a binary coded decimal number.

Answer: Given binary value stored in the register is

0101 | 1001 | 0111 = $(597)_{BCD}$

(c) Perform the binary subtraction of $(229)_{10} - (46)_{10}$.

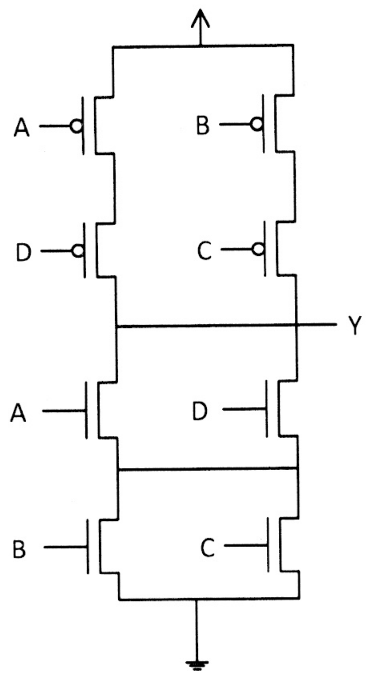
The entire procedure of the subtraction has to be shown clearly.

Answer:

B		001111100
X	229	11100101
Y	-46	00101110
X-Y	183	10110111

2. CMOS Logic

Write a Boolean equation that represents the logic function implemented by the CMOS logic gate shown below.



$\overline{\mathbf{Y = (A + D) \bullet (B + C)}}$

3. Combinational Logic Design

Implement the following Boolean function with an 8 × 1 multiplexer, a 2-to-4-line decoder and two 2- input OR gates. Note that the complemented inputs are not available. [15 pts.]

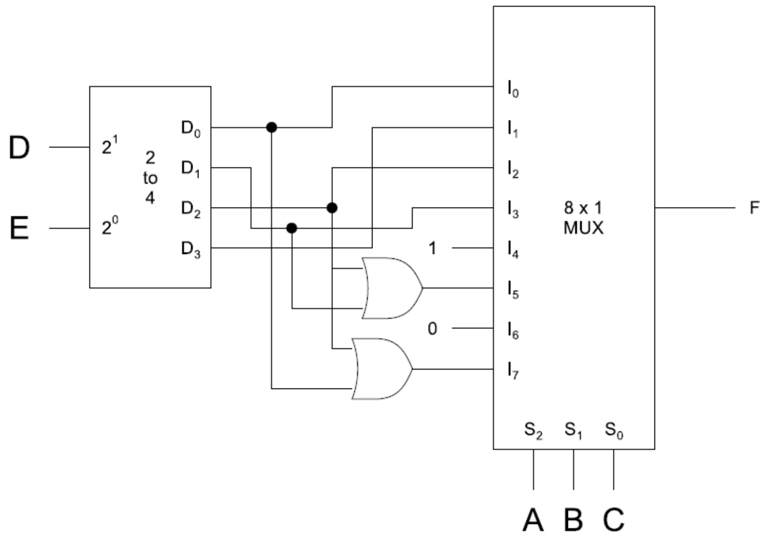
$$F(A, B, C, D, E) = \sum(0,7,10,13,16,17,18,19,21,22,28,30)$$

(a) Derive the multiplexer inputs I_0 to I_7 for function F in terms of D and E. [7pts]

S_2	S_1	S_0	Inputs		Outputs	
A	B	C	D	E	F	
0	0	0	0	0	1	$I_0 = D'E'$
0	0	0	0	1	0	
0	0	0	1	0	0	
0	0	0	1	1	0	
0	0	1	0	0	0	$I_1 = DE$

0	0	1	0	1	0	
0	0	1	1	0	0	
0	0	1	1	1	1	
0	1	0	0	0	0	I_2 =DE'
0	1	0	0	1	0	
0	1	0	1	0	1	
0	1	0	1	1	0	
0	1	1	0	0	0	I_3=D'E
0	1	1	0	1	1	
0	1	1	1	0	0	
0	1	1	1	1	0	
1	0	0	0	0	1	I_4=1
1	0	0	0	1	1	
1	0	0	1	0	1	
1	0	0	1	1	1	
1	0	1	0	0	0	I_5= D'E + DE'
1	0	1	0	1	1	
1	0	1	1	0	1	
1	0	1	1	1	0	
1	1	0	0	0	0	I_6= 0
1	1	0	0	1	0	
1	1	0	1	0	0	
1	1	0	1	1	0	
1	1	1	0	0	1	I_7 = D'E' + DE'
1	1	1	0	1	0	
1	1	1	1	0	1	
1	1	1	1	1	0	

(b) Draw the logic schematic based on the given hardware specifications. You need not draw the internal schematic of the decoder and multiplexer.



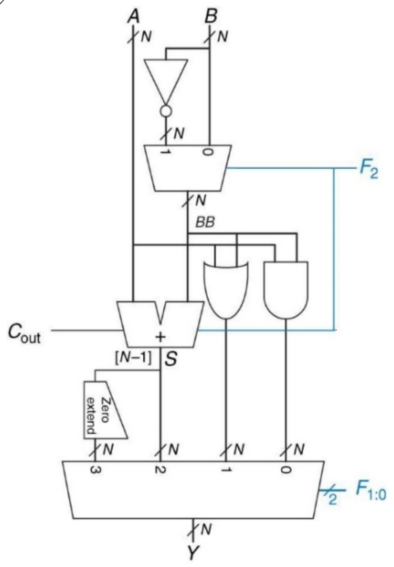
(c) Describing Combinational Logic Circuit in SystemVerilog

Write a SystemVerilog module to describe the functionality of the logic schematic designed in the question 3 (a).

```
module Q3 (input logic D,E,

           input logic [2:0] s,

           output logic F);

         always_comb

            begin

               case (s) // the internal wires are ignored and the code is

                          developed based on logic//

                       3'd0: F = !D & !E;

                       3'd1: F = D & E;

                       3'd2: F = D &!E;

                       3'd3: F = !D & E;

                       3'd4: F = 1'b1;

                       3'd5: F = D ^ E; // F = D &!E + !D & E

                       3'd6: F = 1’b0;

                       3'd7: F = !E;

             endcase

      end

endmodule
```

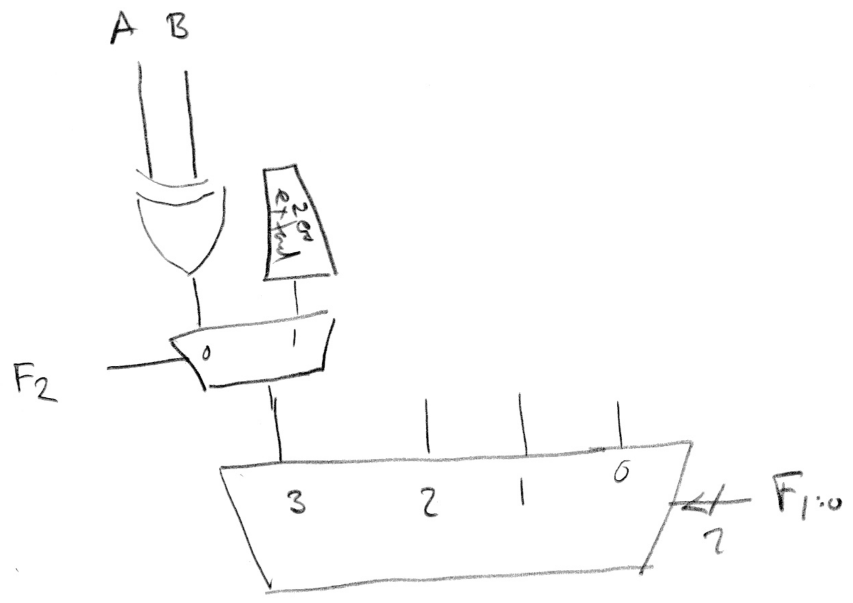
4. Combinational Design

The diagram below shows the ALU designed in class along with the original function table. Modify or redraw the necessary part of the ALU diagram so that the currently unused function code F2:0 = 011 will perform A XOR B. *Minimize the amount of hardware that you add.*

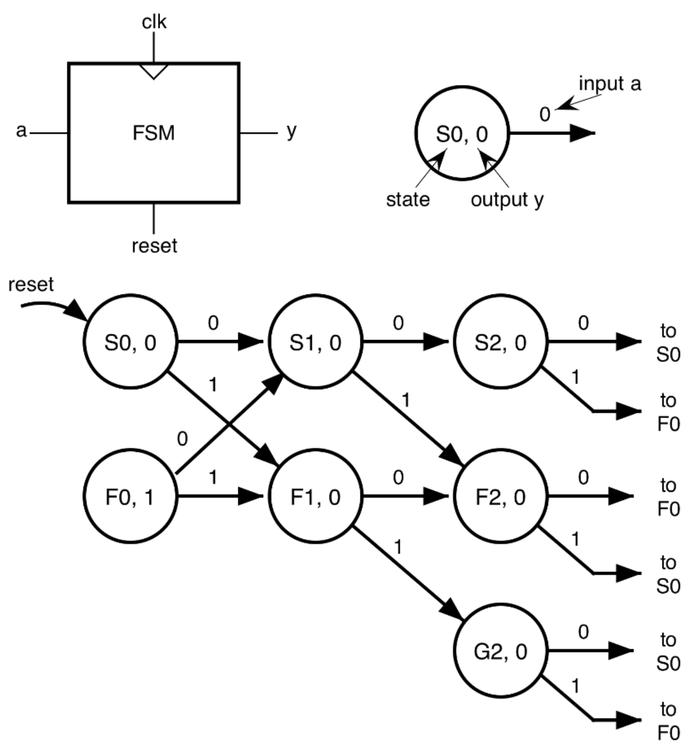


$F_{2:0}$	Function
000	A & B
001	A | B
010	A + B
011	*unused* $\Rightarrow A \oplus B$
100	A & ~B
101	A | ~B
110	A – B
111	SLT

Modify logic that feeds input 3 of the multiplexer.



5. **FSM Design:**
The FSM shown below looks at a 3-bit sequences of values on input **a** during successive clock cycles and generates an output **y** at the end of the sequence



(a) Show the response of the FSM to input **a** over successive clock cycles (i.e., state and output value) by completing the following table:

a	0	0	0	1	0	0	0	1	1	0	0	1	1	1	1	1
state	S0	S1	S2	S0	F1	F2	F0	S1	F2	S0	S1	S2	F0	F1	G2	F0
y	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1

(b) Briefly describe what this FSM does:

Solution:

It outputs a 1 when the 3-bit sequence has odd parity. (i.e. the number of 1's is odd)

Describing FSMs in Verilog

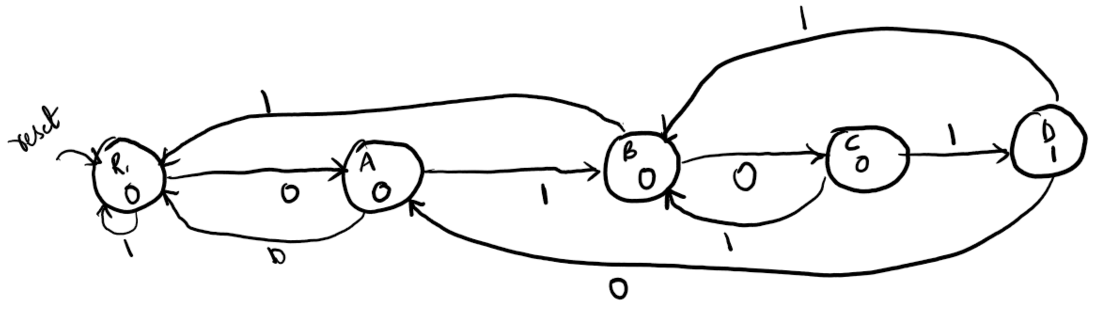
6. Write a System-Verilog module to implement the state transition diagram shown in the previous problem.

HINT: separate the next state and output logic to make the description simpler.

Sequence Detectors

7. Design a Moore Circuit to recognize a pattern consisting of an arbitrary number alternating 0's and 1's followed by two 0's, e.g., "0100", "010100", "01010100", etc. When the pattern is recognized, the circuit should output a "1" for one clock cycle.

(a) Draw a state transition diagram of this circuit in the space provided below. Include a reset state R that indicates where the circuit begins operation.



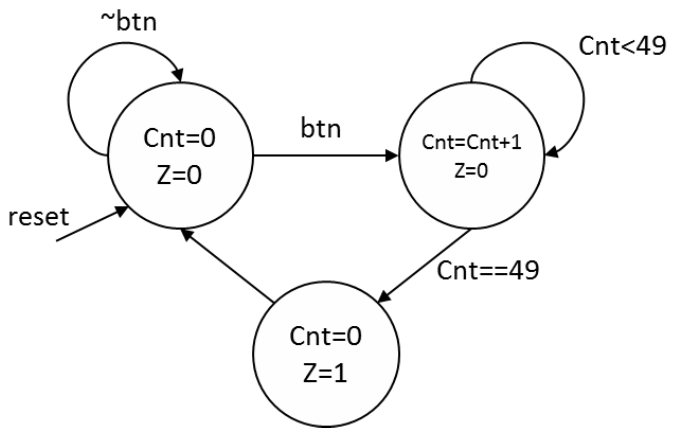
(b) Complete the table below to show how your circuit will react to the input assuming that each column of the table corresponds to one clock period.

a	1	0	1	0	1	0	0	0	0	1	1	0	1	0	0	1
state	R	R	A	B	C	B	C	D	A	R	R	R	A	B	C	D
y	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

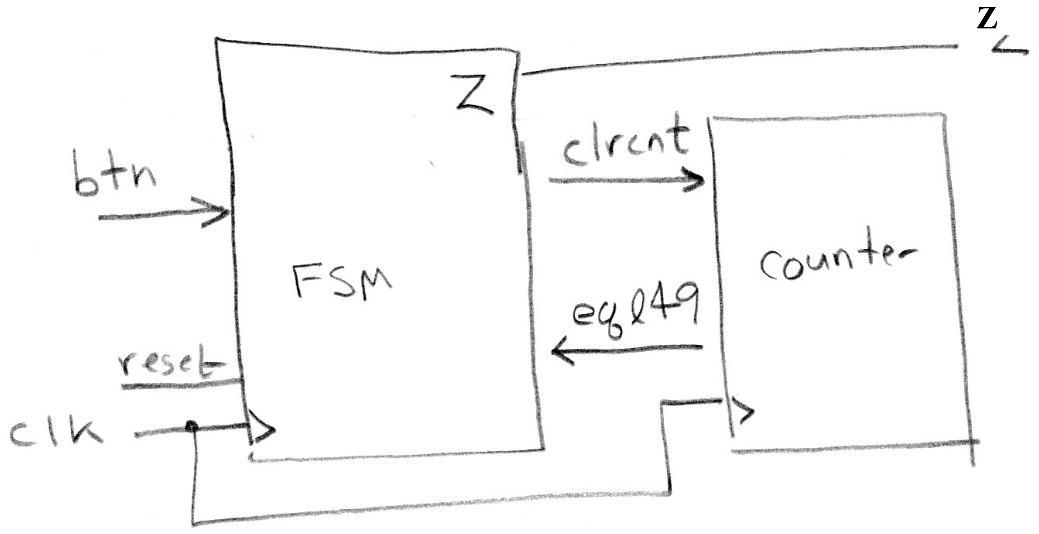
(c) Describing FSM's in SystemVerilog
Implement your FSM from the problem in SystemVerilog in the space below.

You can write this, it is similar to any FSM

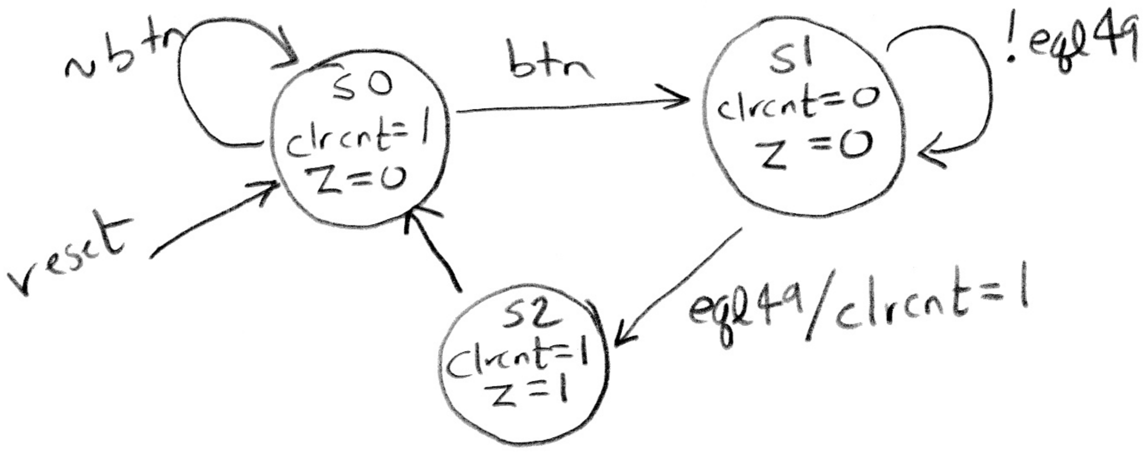
8. The abstract (high-level) state machine shown below describes a system that outputs a one for a single cycle 50 clock cycles after an input button is pressed.



(a) Draw the high-level organization to implement the functionality described by the state diagram. The FSM can be represented as a single block. Make sure to label ports for different elements and indicate the widths of lines that hold more than a single bit.



(b) Redraw the state transition diagram to be a true finite state machine that uses the concrete inputs and outputs shown in your Part (a) organization.

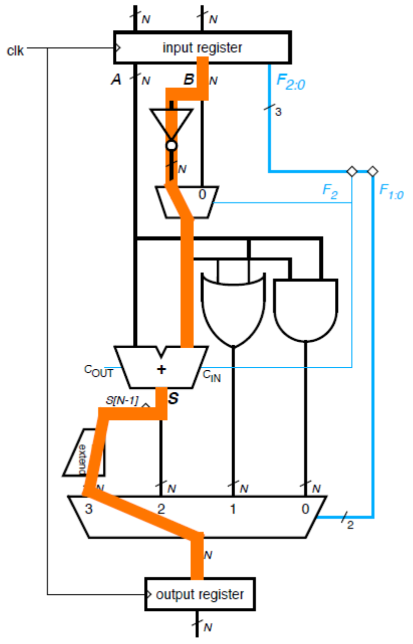


Sequential Circuit Timing

9.

In the diagram below the ALU similar to the one in Lab 6 is implemented on an integrated circuit as part of a microprocessor design with a bitwidth N=32. The ALU connected to input and output registers as shown in the diagram below. In this integrated circuit the logic gates and flip-flops have the following timing characteristics:

Registers:	
Clock-Q Propagation Delay	t_{pcq} = 90 ps
Setup time	t_{setup} = 20 ps
Hold time	t_{hold} = 10 ps
Combinational Logic Gates[1]:	
Propagation Delay	t_{pd} = 100 ps
Contamination Delay	t_{cd} = 60 ps
2-1 Multiplexer:	
Propagation Delay	t_{pd} = 200 ps
4-1 Multiplexer:	
Propagation Delay	t_{pd} = 300 ps
32-bit Adder (N=32)	
Propagation Delay Cin-S	t_{pdcs} = 3000ps
Propagation Delay Cin-Cout	t_{pcico} = 3200ps
Propagation Delay A/B-S	t_{pas} = 3000ps
Propagation Delay A/B-Cout	t_{pacp} = 3100ps



(a) Calculate the minimum clock period and maximum clock frequency at which this circuit can operate.

Solution:

Look for the longest path from a register output to a register input (highlighted in diagram above):

tpcl = tpinv + tpmux2 + tpas + tpmux4

= 100ps + 200ps + 3000ps + 300ps = 3600ps

Tc > tpcq + tpcl + tsetup

Tc > 90ps + 3600ps + 20ps

Tc > 3710ps f > 269.54 MHz

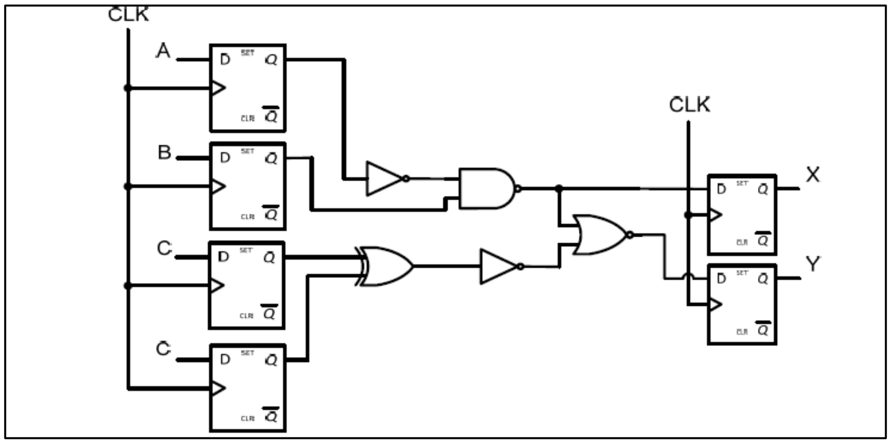
(b) Information about contamination delay for this circuit is only available for the simple logic gates. Alyssa P. Hacker claims that this is enough information to prove that this circuit cannot have a hold violation. Briefly explain whether Alyssa is correct and why.

The hold time constraint is:

tccq + tccl > thold

We don't have enough information about the contamination delay of the combinational logic in the ALU. However, the contamination delay of a single logic gate is 100ps which is greater than the hold time of 60ps. It is reasonable to assume that more complex logic blocks like adders have higher contamination delays than a single gate; therefore there is no way that the contamination delay can be greater than the hold time so Alyssa is correct.

10. A sequential circuit is shown in the following figure:



The timing parameters for the following circuit are

clock-to-q propagation delay, $t_{pcq} = 15ps$
clock-to-q contamination delay, $t_{ccq} = 10ps$

set-up time of the D-flip flop, $t_s = 15ps$
hold-up time of the D-flip flop, $t_h = 10ps$

Gate	**T_{pd} (ps)**	**T_{cd} (ps)**
NAND	15	10
NOR	25	15
XOR	35	25
NOT	10	5

(a) What is the maximum clock frequency for reliable operation of the given circuit, assuming there is no clock skew.

Identifying the longest combinational logic path in the given circuit, firstly to calculate propagation delay:

$t_{pd} = t_{pd\_XOR} + t_{pd\_NOT} + t_{pd\_NOR}$

$= 35ps + 10ps + 25ps = 70$ pos

$T_c \geq 15ps + 70ps + 15$ ps

$\geq 100ps$

$F_{clk = I/Tc} = 1GHz$

(b) How much clock skew can the circuit tolerate before it experiences a hold time violation?

The shortest combinational path in the given circuit has to be considered for the contamination delay.

$t_{cd} = t_{cd\_NAND} = 10ps$

$t_{hold} < t_{ccq} + t_{cd} - t_{skew}$

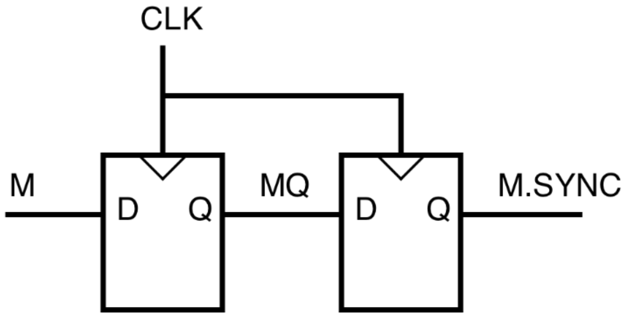
Plugging in all the given values, we get

$t_{skew} < 10ps$

11. Metastability

Suppose that a synchronizer is implemented in an FPGA with the following flip-flop timing characteristics:

Timing		Metastability
Clock-Q Contamination Delay	$t_{ccq} = 0.5$ ns	$\tau = 50$ ps
Clock-Q Propagation Delay	$t_{pcq} = 0.72$ ns	$T_o = 1$ ns
Setup time	$t_{setup} = 0.53$ ns	
Hold time	$t_{hold} = 0$ ns	



Assume that the synchronizer clock f_c = 250 MHz and the asynchronous input M changes at an average rate of N = 20 MHz. Calculate the MTBF of this circuit and include an assessment of whether it is sufficiently large to ensure reliable circuit operation.

Solution:

$$MTBF = \frac{T_c \cdot e^{\frac{T_c - t_{setup}}{\tau}}}{N \cdot T_0} \qquad T_c = \frac{1}{250MHz} = 4ns$$

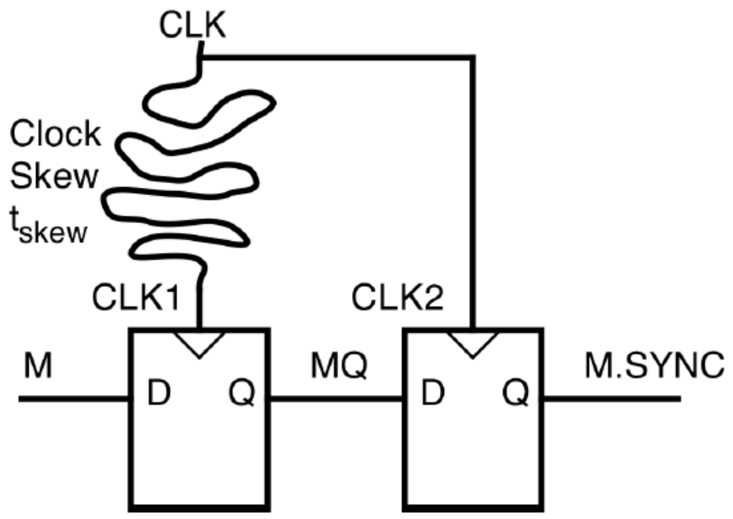
$$= \frac{(4\times10^{-9})\, e^{\frac{4ns - 0.53ns}{50ps}}}{20\times10^{6} \cdot 1\times10^{-9}} = 2.76\times10^{23}\ sec$$

$$= 8.75\times10^{15}\ years$$

This is a very long time and is sufficiently large to ensure reliable circuit operation

12. Metastability and Clock Skew

Suppose that the clock wiring connecting the two flip-flops is uneven resulting in a clock skew. Specifically, assume that in the worst case the rising edge of the clock signal for the first flip-flop clock signal arrives *later* than the rising edge for the second flip-flop by tskew=1ns. Calculate the MTBF of this circuit *accounting for skew* and include an assessment of whether it is sufficiently large to ensure reliable circuit operation.



Solution:

The clock skew reduces the amount of time that the synchronizer has to resolve metastability from $T_c - t_{setup}$ to $T_c - t_{setup} - t_{skew}$. So the resulting MTBF will be

$$MTBF = \frac{T_c \cdot e^{\frac{T_c - t_{setup} - t_{skew}}{\tau}}}{N \cdot T_0}$$

$$= 5.6 \times 10^{14} s$$

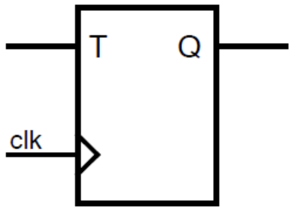
$$= 18.046 \times 10^{6} \text{ years}$$

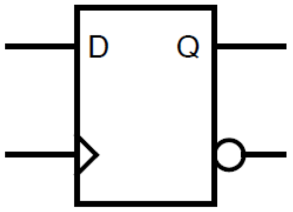
Every eighteen million years is still a very long time unless we manufacture this circuit in very high volume

13. Sequential Circuits

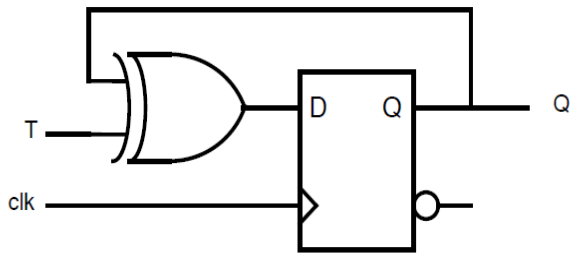
A *T flip-flop* (also called a *toggle*) flip-flop has a single input *T*. When *T*=0, the flip-flop output Q holds its current value. When *T*=1, the flip-flop inverts its Q value on each successive clock edge as shown in the table below (Q* is the next value of Q). Implement a T-flip-flop using a D flip-flop and combinational logic in the space provided below.

T	Q	Q*
0	0	0
0	1	1
1	0	1
1	1	0

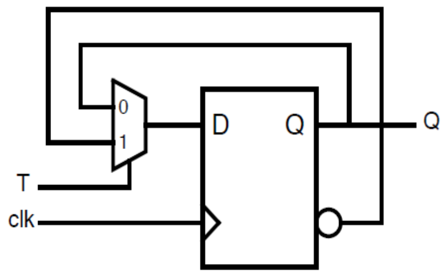




Solution



Alternative 1



Alternative 2

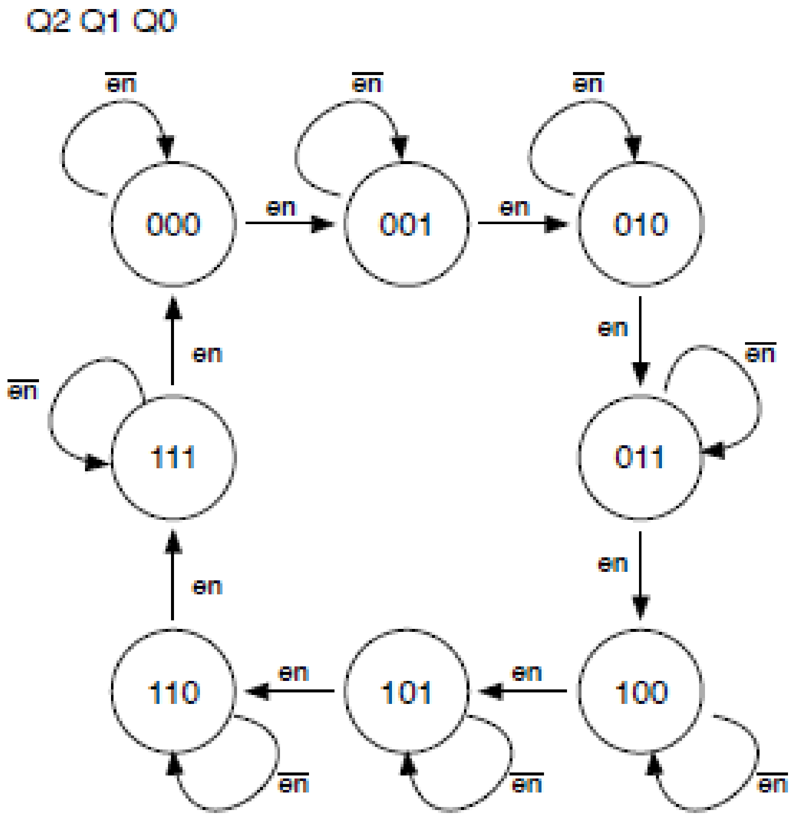
14. Flip-Flops in SystemVerilog

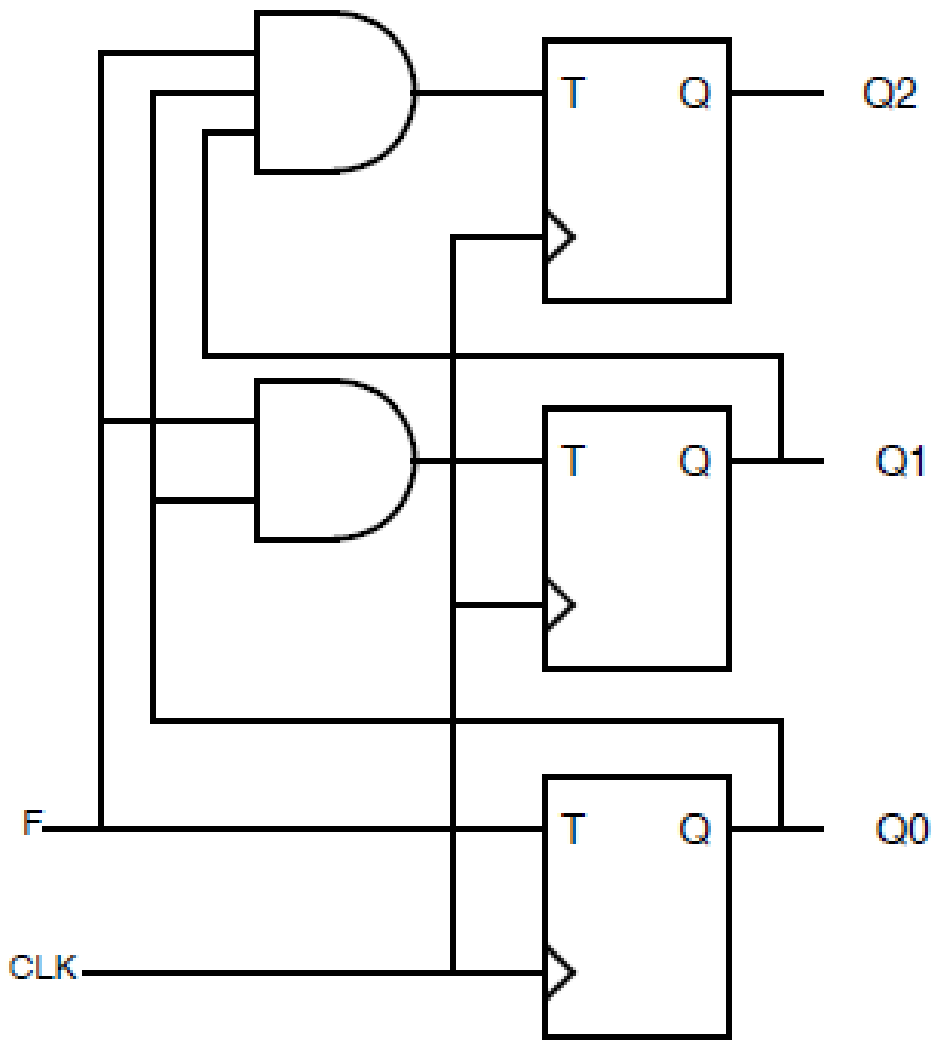
Describe the function of the T flip-flop in SystemVerilog. Include a *synchronous* reset input:

```
module tff (input logic clk, rst, t, output logic q);

  always_ff @(posedge clk)
    if (rst) q <= 0;
    else if (t) q <= ~q; // alternative: else q <= q ^ t;

endmodule // tff
```

15. Counters

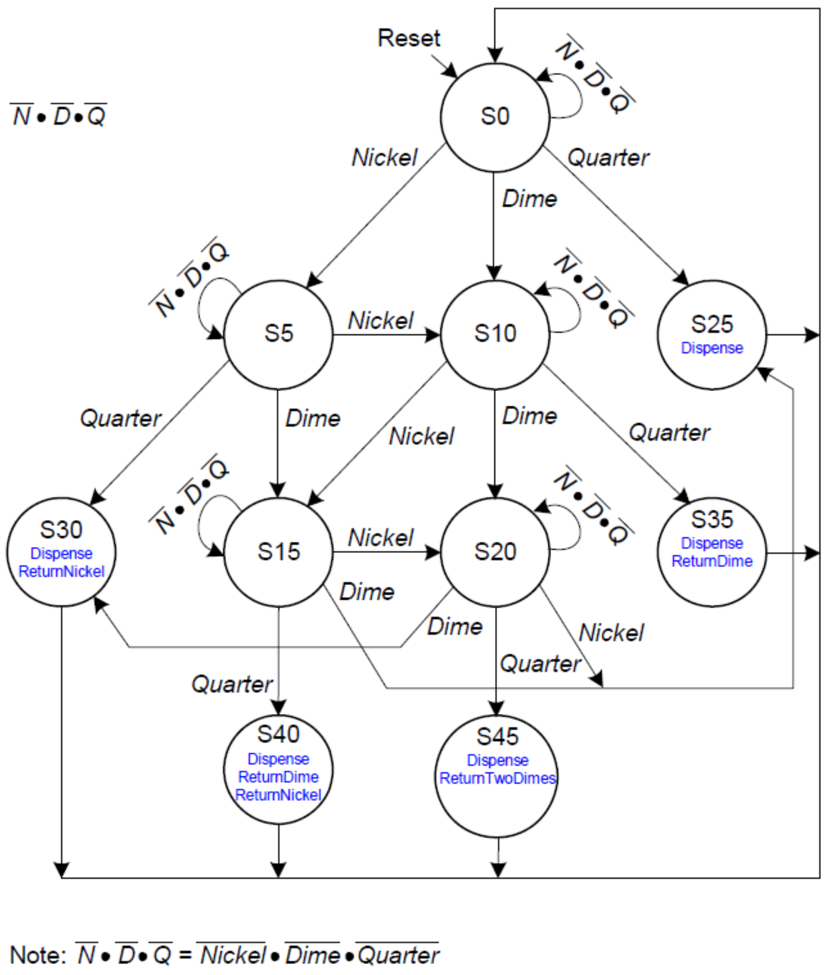
T flip-flops are often used to construct counter circuits. Show how to connect three T flip-flops with logic gates to create a 3-bit binary counter with enable (state transition diagram shown below).




T
Q
Q2
T
Q
Q1
F
T
Q
Q0
CLK

16. Finite State Machine Design

You have been enlisted to design a soda machine dispenser for yourdepartment lounge. Sodas are partially subsidized by the student chapter of theIEEE, so they cost only 25 cents. The machine accepts nickels, dimes, and quarters.When enough coins have been inserted, it dispenses the soda and returns anynecessary change. Design an FSM controller for the soda machine. The FSM inputsareNickel,Dime, andQuarter, indicating which coin was inserted. Assume thatexactly one coin is inserted on each cycle. The outputs areDispense,ReturnNickel,ReturnDime, andReturnTwoDimes. When the FSM reaches 25 cents, it assertsDispenseand the necessaryReturnoutputs required to deliver the appropriate change. Then it should be ready to start accepting coins for another soda.



$\overline{N} \bullet \overline{D} \bullet \overline{Q}$
Reset
S0
S5
S10
S15
S20
S25 Dispense
S30 Dispense ReturnNickel
S35 Dispense ReturnDime
S40 Dispense ReturnDime ReturnNickel
S45 Dispense ReturnTwoDimes
Nickel
Dime
Quarter
Note: $\overline{N} \bullet \overline{D} \bullet \overline{Q} = \overline{Nickel} \bullet \overline{Dime} \bullet \overline{Quarter}$

FIGURE 3.2 State transition diagram for soda machine dispense of Exercise 3.23

state	encoding $s_{9:0}$
S0	0000000001
S5	0000000010
S10	0000000100
S25	0000001000
S30	0000010000
S15	0000100000
S20	0001000000
S35	0010000000
S40	0100000000
S45	1000000000

FIGURE 3.3 State Encodings for Exercise 3.26

current state *s*	inputs			next state *s'*
	nickel	*dime*	*quarter*	
S0	0	0	0	S0
S0	0	0	1	S25
S0	0	1	0	S10
S0	1	0	0	S5
S5	0	0	0	S5
S5	0	0	1	S30
S5	0	1	0	S15
S5	1	0	0	S10
S10	0	0	0	S10

TABLE 3.11 State transition table for Exercise 3.26

current state *s*	inputs			next state *s'*
	nickel	*dime*	*quarter*	
S10	0	0	1	S35
S10	0	1	0	S20
S10	1	0	0	S15
S25	X	X	X	S0
S30	X	X	X	S0
S15	0	0	0	S15
S15	0	0	1	S40
S15	0	1	0	S25
S15	1	0	0	S20
S20	0	0	0	S20
S20	0	0	1	S45
S20	0	1	0	S30
S20	1	0	0	S25
S35	X	X	X	S0
S40	X	X	X	S0
S45	X	X	X	S0

TABLE 3.11 State transition table for Exercise 3.26

current state *s*	inputs			next state *s'*
	nickel	*dime*	*quarter*	
0000000001	0	0	0	0000000001
0000000001	0	0	1	0000001000
0000000001	0	1	0	0000000100
0000000001	1	0	0	0000000010

TABLE 3.12 State transition table for Exercise 3.26

current state s	inputs			next state s'
	nickel	dime	quarter	
0000000010	0	0	0	0000000010
0000000010	0	0	1	0000010000
0000000010	0	1	0	0000100000
0000000010	1	0	0	0000000100
0000000100	0	0	0	0000000100
0000000100	0	0	1	0010000000
0000000100	0	1	0	0001000000
0000000100	1	0	0	0000100000
0000001000	X	X	X	0000000001
0000010000	X	X	X	0000000001
0000100000	0	0	0	0000100000
0000100000	0	0	1	0100000000
0000100000	0	1	0	0000001000
0000100000	1	0	0	0001000000
0001000000	0	0	0	0001000000
0001000000	0	0	1	1000000000
0001000000	0	1	0	0000010000
0001000000	1	0	0	0000001000
0010000000	X	X	X	0000000001
0100000000	X	X	X	0000000001
1000000000	X	X	X	0000000001

TABLE 3.12 State transition table for Exercise 3.26

$$S'_9 = S_6 Q$$

$$S'_8 = S_5 Q$$

$$S'_7 = S_2Q$$

$$S'_6 = S_2D + S_5N + S_6\overline{N}\,\overline{D}\,\overline{Q}$$

$$S'_5 = S_1D + S_2N + S_5NDQ$$

$$S'_4 = S_1Q + S_6D$$

$$S'_3 = S_0Q + S_5D + S_6N$$

$$S'_2 = S_0D + S_1N + S_2\overline{N}\,\overline{D}\,\overline{Q}$$

$$S'_1 = S_0N + S_1NDQ$$

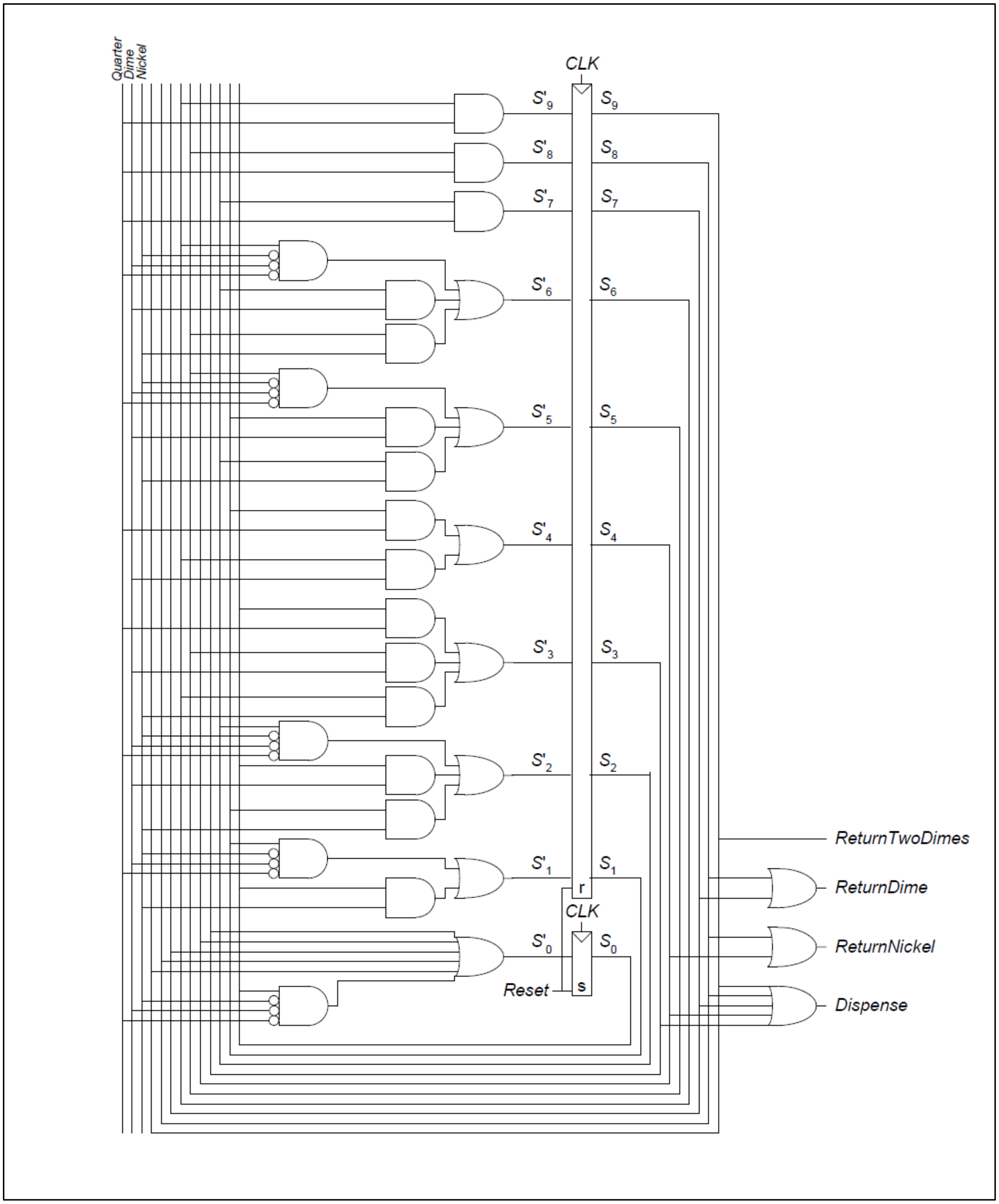
$$S'_0 = S_0\overline{N}\,\overline{D}\,\overline{Q} + S_3 + S_4 + S_7 + S_8 + S_9$$

$$Dispense = S_3 + S_4 + S_7 + S_8 + S_9$$

$$ReturnNickel = S_4 + S_8$$

$$ReturnDime = S_7 + S_8$$

$$ReturnTwoDimes = S_9$$



Logic Circuit Diagram