6.004 Fall 2020 Tutorial Problems
L10 – Sequential Circuits

Before looking at the problems, let's look at some important vocabulary for this section.

Combinational circuit: A circuit made up of a combination of gates like AND, OR, NAND, NOR, multiplexers, etc. Combinational circuits can have *n* inputs and *m* outputs, and have no cycles (feedback) or state elements.

Circuits with feedback: Circuits with feedback loops (i.e., cycles) that can hold state. An example of such circuit is a D Latch.

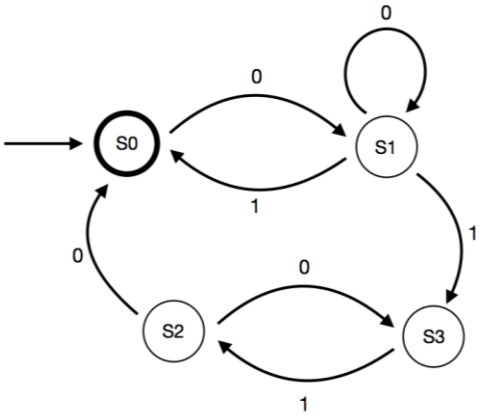
D Latch: A D latch circuit's output depends on a clock. If the clock is low, the input passes to output. If the clock is high, the latch holds its output. For the D latch, the latch is asynchronous and the outputs can change as soon as the inputs do.

D Flip Flop: A circuit with two stable states that can store one bit of state information. The output changes state by signals applied to one or more control inputs. A flip-flop is edge-triggered: it only changes state when the clock signal goes from low to high (or alternatively, from high to low). A D flip-flop is usually built from two D latches.

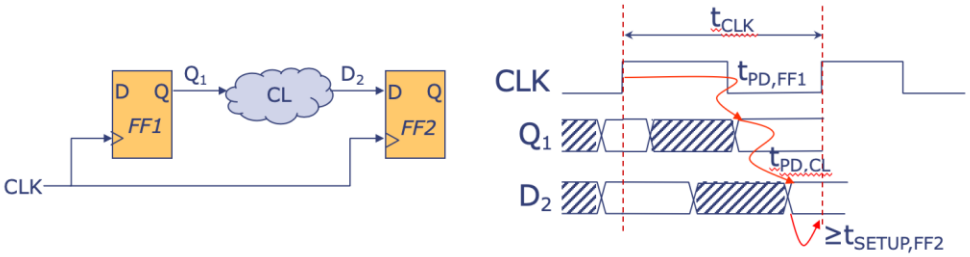
The diagram below shows a flip-flop enhanced with a *write-enable* signal (useful when we don't always want to update the flip-flop's value). Because D flip-flops hold an unknown circuit when first-powered up, D flip-flops also often include a *reset* circuit to set their initial value (not shown).

Finite State Machine: An abstraction for synchronous sequential circuits (where all state is kept in flip-flops that are driven by the same clock signal) that results from discretizing time into *cycles*. An FSM has inputs, outputs, and K possible states (encoded in flip-flops). At a given cycle, an FSM is in a particular state, and computes both its outputs and its next state (its state for the current cycle) based on its inputs and its current state.

An FSM behavior is precisely described with a *state-transition diagram* that shows the states (using circles), the possible transitions from each state to other states given its inputs (using arrows), and the initial state (circles in bold). An example state-transition diagram is shown below:

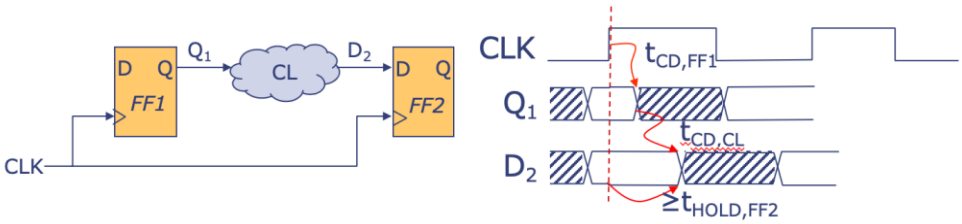


Sequential Circuit Timing: Sequential circuits must satisfy the setup time and hold time of each of the registers (i.e., flip-flops) in order to function correctly. To satisfy the setup times of all registers, the clock period of a sequential circuit must be at least as long as the maximum register to register propagation delay plus the setup time of the downstream register. To satisfy the hold times of all registers, the contamination delay along each register to register path must be at least as long as the hold time of the downstream register in that path.



- To meet FF2's setup time,

$$t_{CLK} \geq t_{PD,FF1} + t_{PD,CL} + t_{SETUP,FF2}$$



- To meet FF2's hold-time constraint

$$t_{CD,FF1} + t_{CD,CL} \geq t_{HOLD,FF2}$$

Note: A small subset of essential problems are marked with a red star (★). We especially encourage you to try these out before recitation.

Problem 1. ★

Write the truth tables for both a D latch and a D flip-flop. (Note: Q* is the next state of Q)

D latch Truth Table

C	D	Q	Q*
0	0	X	0
0	1	X	1
1	X	0	0
1	X	1	1

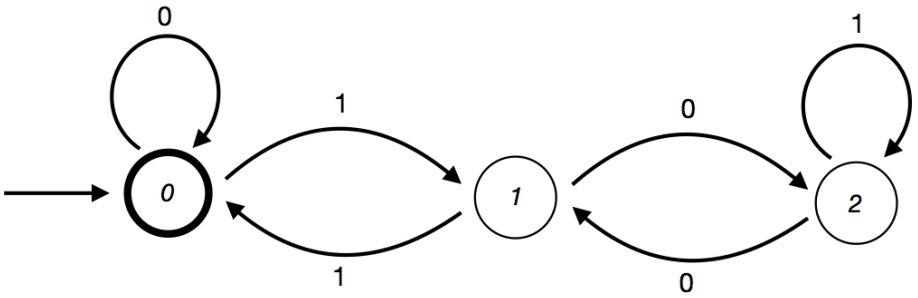
D flip-flop Truth Table

EN	D	Q	Q*
0	X	0	0
0	X	1	1
1	0	X	0
1	1	X	1

Problem 2. ★

Consider a "divisible-by-3" FSM that accepts a binary number entered one bit at a time, most significant bit first. The FSM has a one-bit output that indicates if the number entered so far is divisible by 3.

If the value of the number entered so far is N, then after the digit b is entered, the value of the new number N' is 2N + b. This leads to the following state-transition diagram where the states are labeled with the value of N mod 3.



(A) Construct a truth table for the FSM logic. Inputs include the state bits (i.e. *00*, *01*, or *11*) and the next (least significant) bit of the number; outputs include the next state bits and the output.

The 3 states are encoded as {S1,S0} = 00, 01 and 10 respectively

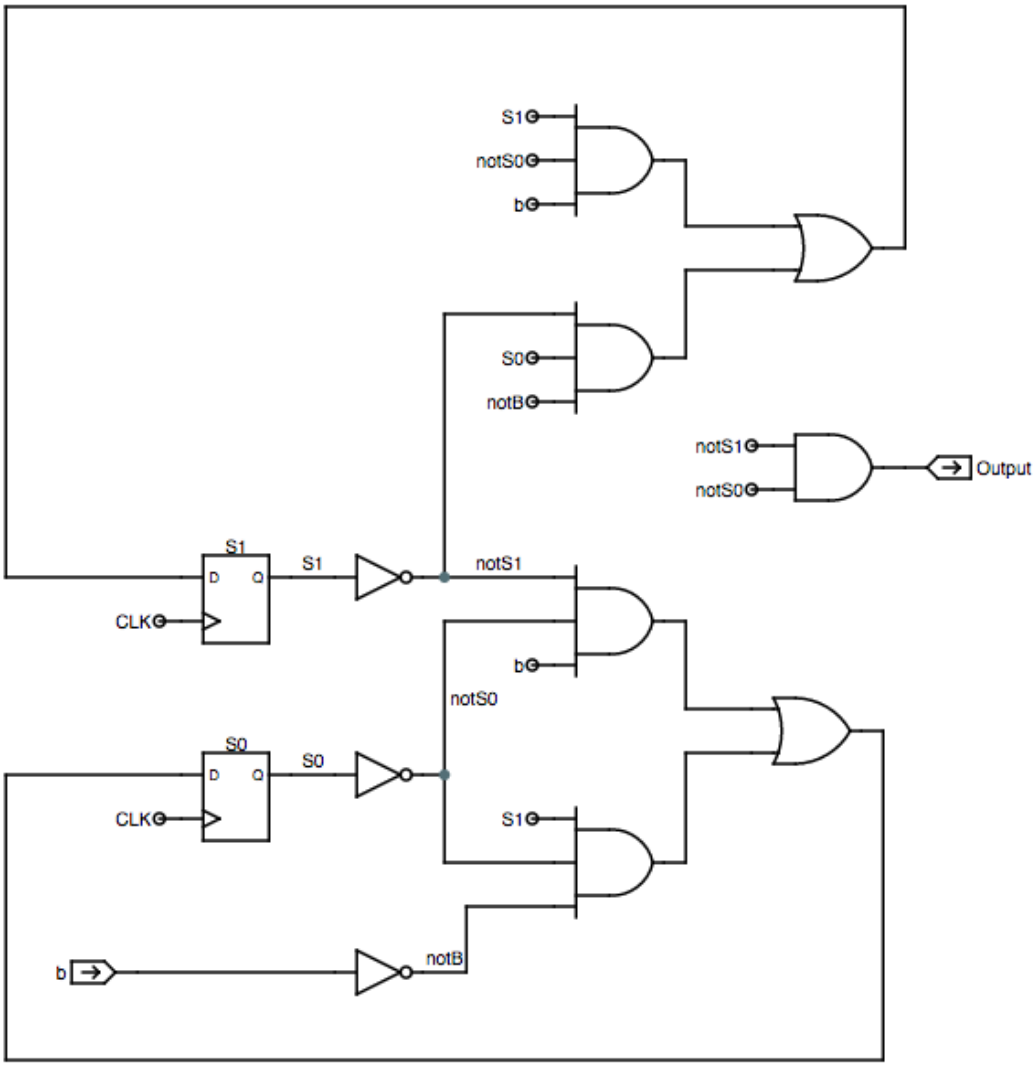
$S1^t$	$S0^t$	b	$S1^{t+1}$	$S0^{t+1}$	output
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	1	0	0

(B) Based on the truth table, implement the FSM using D flip-flops.

$S0^{t+1} = \sim S1^t \sim S0^t\, b \;+\; S1^t \sim S0^t \sim b$

$S1^{t+1} = \sim S1^t\; S0^t \sim b \;+\; S1^t \sim S0^t\, b$

$\text{Output} = \sim S1^t\; \sim S0^t$


S1
notS0
b
S0
notB
notS1
notS0
Output
S1
D
Q
S1
notS1
CLK
b
notS0
S0
D
Q
S0
CLK
S1
b
notB

Problem 3.

In this problem, we construct a sequential circuit to compute the N^{th} Fibonacci number denoted by F_N. The following recurrence relation defines the Fibonacci sequence.

$$F_0 = 0, F_1 = 1, F_N = F_{N-1} + F_{N-2} \; \forall \; N >= 2$$

There are two registers **x** and **y** that store the Fibonacci values for two consecutive integers. In addition, a counter register **i** is initialized to N-1 and decremented each cycle. The computation stops when register **i** goes down to 0 and the result (F_N) is available in register **x**.

(A) What are the initial values for registers **x** and **y**?

The initial values are y = 0, x = 1 respectively.

(B) Derive the next state computation equations for the three registers.

$i^{t+1} = i^t - 1$

$y^{t+1} = x^t$

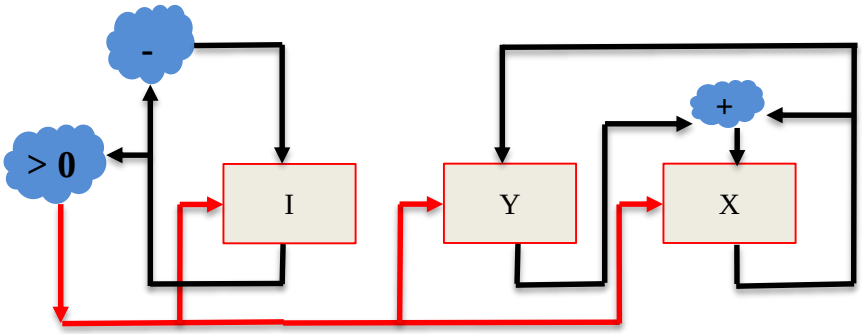
$x^{t+1} = x^t + y^t$

(C) Derive the logic for the enable signal that determines when the registers are updated using the next state logic. Note that all three registers are controlled by a single enable signal.

$i^t > 0$

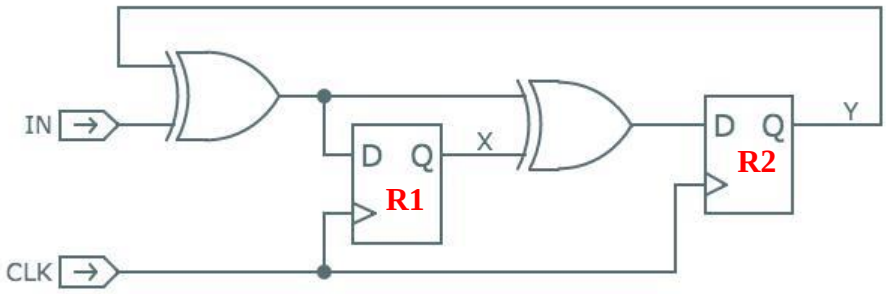
This ensures that computation stops when counter i becomes 0.

(D) Implement the sequential circuit using the next state and enable logic derived above.



Problem 4. ★

Consider the following sequential logic circuit. It consists of one input IN, a 2-bit register that stores the current state, and some combinational logic that determines the state (next value to load into the register) based on the current state and the input IN.



(A) Using the timing specifications shown below for the XOR and DREG components, determine the shortest clock period, t_{CLK}, that will allow the circuit to operate correctly or write NONE if no choice for t_{CLK} will allow the circuit to operate correctly and briefly explain why.

Component	*tCD*	*tPD*	*tSETUP*	*tHOLD*
XOR2	0.15ns	2.1ns	–	–
DREG	0.1ns	1.6ns	0.4ns	0.2ns

Minimum value for t_{CLK} (ns): _6.2_ or explain why none exists

Longest path is from R2 -> R2 through two XOR gates.

$t_{CLK} >= t_{PD,REG} + 2 * t_{PD,XOR2} + t_{SETUP,REG} = 1.6 + 2 * 2.1 + 0.4 = 6.2$

(B) One of the engineers on the team suggests using a new, faster XOR2 gate with $t_{CD} = 0.05$ns and $t_{PD} = 0.7$ns. Determine a new minimum value for t_{CLK} or write NONE and explain why no such value exists.

Minimum value for t_{CLK} (ns): __ NONE ____
or explain why none exists

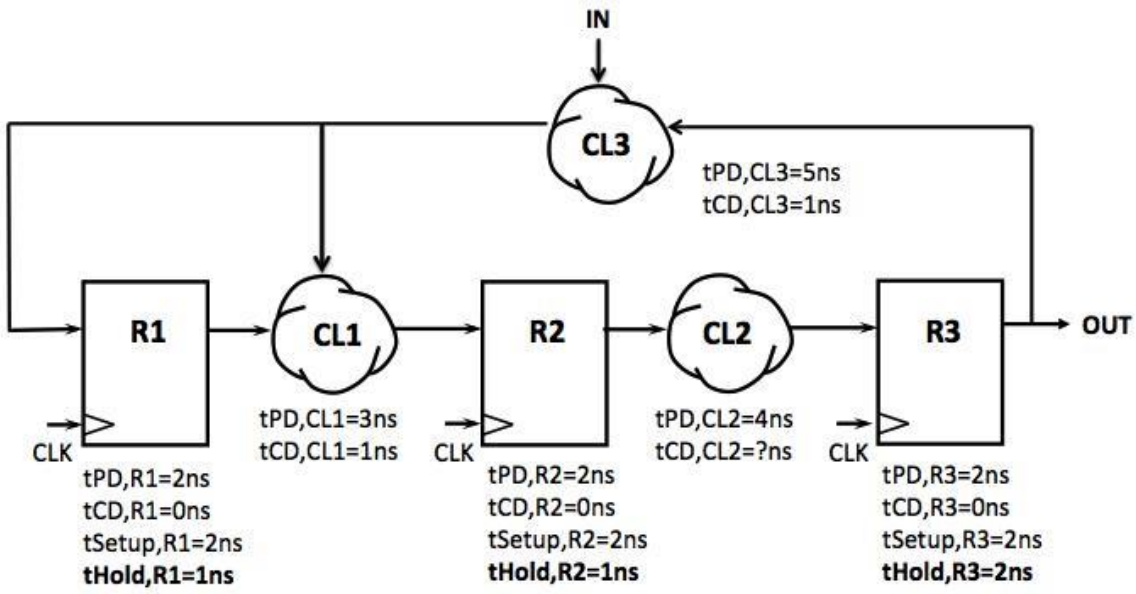
Hold time constraint with old delays.
$t_{CD,REG} + t_{CD,XOR2} = 0.1 + 0.15 = 0.25 >= t_{HOLD,REG} = 0.2$

Hold time constraint does not hold with new delays.

$t_{CD,REG} + t_{CD,XOR2} = 0.1 + 0.05 = 0.15$ not $>= t_{HOLD,REG} = 0.2$

Problem 5.

Consider the following sequential logic circuit. It consists of three D registers, three different pieces of combinational logic (CL1, CL2, and CL3), one input IN, and one output OUT. The propagation delay, contamination delay, and setup time of the registers are all the same and are specified below each register. **The hold time for the registers is NOT the same** and is specified in bold below each register. The timing specification for each combinational logic block is shown below that logic.



(A) What is the smallest value for the t_{CD} of CL2 that will allow all the registers in the circuit to operate correctly?

Smallest value for t_{CD} of CL2 (ns): ____2_____

Check hold time constraint between R2 and R3

$t_{CD,R2} + t_{CD,CL2} >= t_{HOLD,R3}$ => $t_{CD,CL2} >= t_{HOLD,R3} - t_{CD,R2} = 2 - 0 = 2$

(B) What is the smallest value for the period of CLK (i.e., t_{CLK}) that will allow all the registers in the circuit to operate correctly?

$t_{PD} + t_{SETUP}$ on longest path R3 -> R2 **Smallest value for t_{CLK} (ns): 12**

$t_{PD,R3} + t_{PD,CL3} + t_{PD,CL1} + t_{SETUP,R2} = 2 + 5 + 3 + 2 = 12$

(C) What are the propagation delay and contamination delay of the output, OUT, of this circuit relative to the rising edge of the clock?

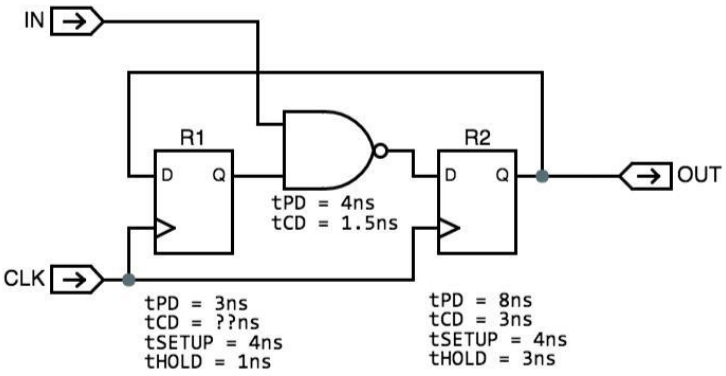
The delay measured from rising edge of the clock to OUT goes through R3

t_{PD} for OUT (ns): $t_{PD,R3} = 2$

t_{CD} for OUT (ns): $t_{CD,R3} = 0$

Problem 6. ★

Consider the following sequential logic circuit. The timing specifications are shown below each component. Note that the two registers do NOT have the same specifications.



(A) What is the smallest value for the period of CLK (i.e., tCLK) that will allow both registers in the circuit to operate correctly?

Smallest value for tCLK (ns): 12

For R1->R2 $t_{CLK} >= t_{PD,R1} + t_{PD,NAND2} + t_{SETUP,R2} = 3 + 4 + 4 = 11$
For R2->R1 $t_{CLK} >= t_{PD,R2} + t_{SETUP,R1} = 8 + 4 = 12$

(B) What is the smallest value for the tCD of R1 that will allow both registers in the circuit to operate correctly?

Smallest value for tCD of R1 (ns): 1.5

For R1->R2 $t_{CD,R1} + t_{CD,NAND2} >= t_{HOLD,R2} \Rightarrow t_{CD,R1} >= 3 - 1.5 = 1.5$

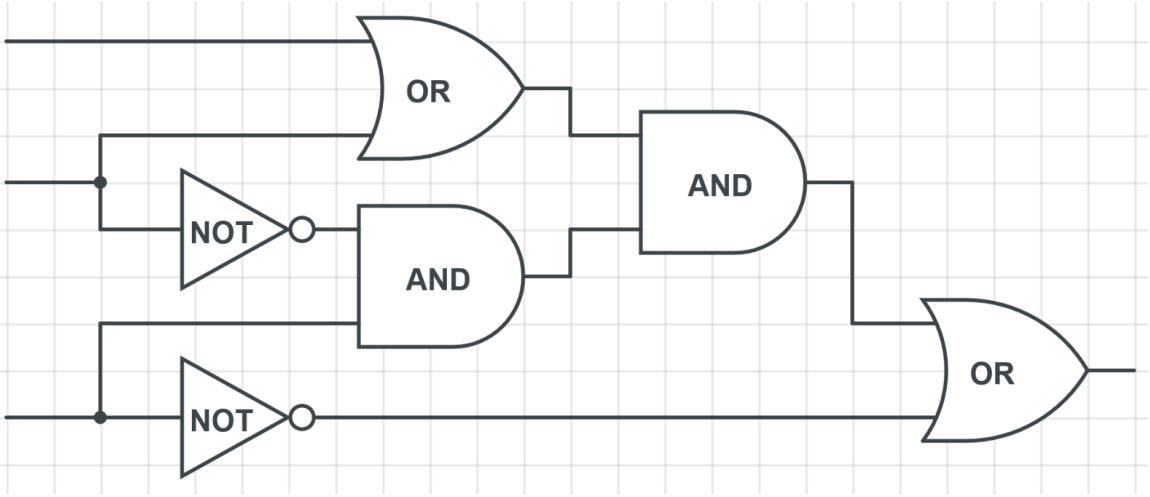
(C) Suppose two of these sequential circuits were connected in series, with the OUT signal of the first circuit connected to the IN signal of the second circuit. The same CLK signal is used for both circuits. Now what is the smallest value for the period of CLK (i.e., tCLK) that will allow both registers in the circuit to operate correctly?

Smallest value for tCLK (ns): 16

For R2 -> R2_COPY $t_{CLK} >= t_{PD,R2} + t_{PD,NAND2} + t_{SETUP,R2\_COPY} = 8 + 4 + 4 = 16$

Problem 7.

Consider the circuit shown below:



(A) Find the propagation delay (t_{PD}) and the contamination delay (t_{CD}) of the circuit, using the t_{PD} and t_{CD} information for the gate components shown in the following table.

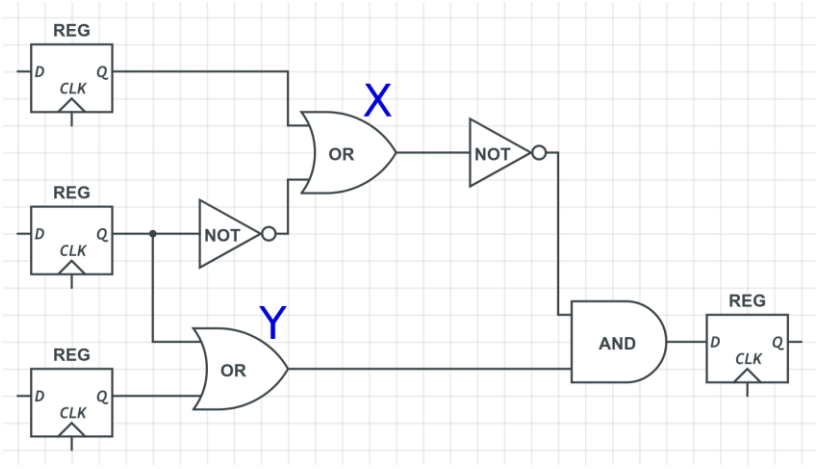
Component	t_{PD}	t_{CD}
NOT	100ps	10ps
AND	300ps	30ps
OR	350ps	35ps

t_{PD}: ____1050_____ps

t_{CD}:_____45______ps

$t_{PD} = t_{PD\text{-}NOT} + t_{PD\text{-}AND} + t_{PD\text{-}AND} + t_{PD\text{-}OR} = 1050ps$
$t_{CD} = t_{CD\text{-}NOT} + t_{CD\text{-}OR} = 45ps$

Now consider the sequential circuit below, which includes standard D flip flops, in addition to basic logic gates. All registers share a common clock. The table below shows the timing specification of each component.



Component	t_{PD}	t_{CD}	t_{SETUP}	t_{HOLD}
NOT	100ps	10ps	—	—
AND	300ps	30ps	—	—
OR	350ps	35ps	—	—
REG	1000ps	40ps	150ps	100ps

(B) What is the minimum clock period we can use for this circuit to function properly?

***Minimum* clock period for correct operation (ps): _____2000_____**

(C) We are looking to decrease the clock cycle required by this circuit and discover a selection of faster OR gates we can use instead. However, because they are very expensive, we can only replace ONE of the OR gates in our circuit with one of the faster OR gates to the right. Each OR gate has a label next to it (X or Y). Which OR gate should be replaced?

OR gate type	t_{PD}	t_{CD}
Model 1	300ps	30ps
Model 2	250ps	25ps
Model 3	150ps	15ps
Model 4	50ps	5ps

OR gate to be replaced (select X or Y as the gate to replace): (X) Y

Which gate should it be replaced with to minimize the minimum clock period required while ensuring that the circuit still functions properly? What is that minimum clock period?

OR gate to replace it with (write down the model number): Model __2_

After replacement, *minimum* clock period for correct operation (ps): _1900_

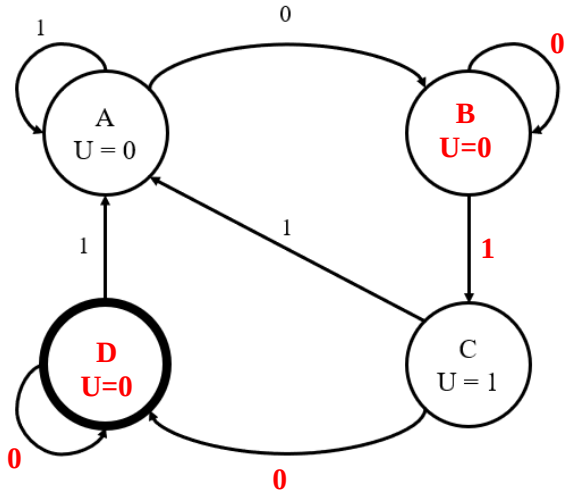
Problem 8.

Suppose Alan wants to read a stream of 6s and 0s and **find each separate instance** where he has a 6, followed by one or more 0s, and then followed by another 6. Each instance should have no shared numbers with each other. In the state machine Alan designs, it outputs 1 for **one cycle** after having read in a sequence of the form 6, followed by one or more 0s, and then followed by another 6. Otherwise, it outputs a 0. Additionally, the FSM **outputs a 1 the cycle immediately following the cycle where the last 6 input was processed.**

Alan encodes each input digit into a digital value in the following way: Let input = 0 be when reading a 0, and input = 1 be when reading a 6.

He expects his FSM to undergo the following behavior:

Cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12+**
Output	0	0	0	0	0	1	0	0	0	0	1	0
Input	6	0	0	0	6	6	6	6	0	6	0	-



State	Input	Next State	Output
A	0	B	0
A	1	**A**	
B	0	**B**	**0**
B	1	C	
C	0	**D**	1
C	1	A	
D	0	**D**	**0**
D	1	**A**	

Note that the input at cycle 5 will update the state at the beginning of the cycle 6, so the input at cycle 5 will have an effect at cycle 6. The – input in the table means that no valid input is given, and the state of FSM will not be updated. Also note that the output is only dependent on the current state, not the input.

(A) Using the provided partial information, complete both the truth table and the state-transition diagram. In the diagram, fill in the missing state labels with its corresponding output (U) and label each transition with either 0 or 1 to indicate which input value causes the transition. The state shown in bold is the initial state.

(B) Given the following inputs, determine what the next state will be (A, B, C, or D) after the last input is processed:

(i) 60066

Next state: _____A_____

(ii) 6606006

Next state: _____A_____

(iii) 60000006660000000006

Next state: _____C_____

(C) **Below is the truth table for a completely separate FSM from Parts (A & B).** S_1 and S_0 are the two bits of state associated with this FSM, and as such the FSM will be implemented using two registers. Based on the truth table, complete the partially implemented FSM circuit. **You may use NOT gates and 2-input AND, OR, and XOR gates** in your implementation. For full credit, your implementation will consist of 5 gates or fewer.

S_1^t	S_0^t	Input	S_1^{t+1}	S_0^{t+1}	Output
0	0	0	1	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	0	0	1