CDA 3103: Study Set 3

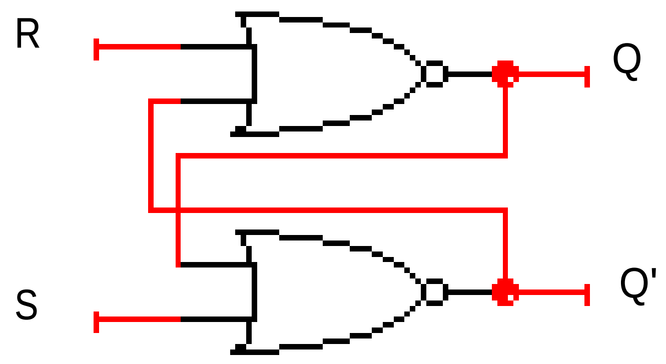
LATCHES, FLIP-FLOPS, REGISTERS, TIMING METHODOLOGY, SEQUENTIAL LOGIC DESIGN, SYNCHRONOUS SYSTEMS

Review: Sequential Logic

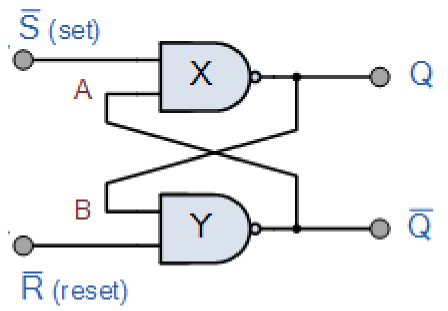
- Sequential Circuits are made up of the same pieces as combinational circuits: wires and gates. The primary difference is the sequential circuits have “States” implemented with a feedback loop. If you provide inputs 1, 1, and 0 to combinational circuit and receive as output 0; then the next time you provide the same circuit inputs 1, 1, and 0 you are guaranteed to receive the same output 0.
- In a sequential circuit, this is not guaranteed. Outputs may differ based on the state of the circuit and outputs may differ based on the previous results of the circuit.

Review: Set-Reset Latch

- The Set-Reset Latch is a sequential element that produces an output based on the current state of the latch. If the latch is in the Set state, the output is 1. If the latch is in the Reset state, the output is 0. If the latch is the Hold state, then the output is the same as the most recent previous state.
- The Set-Reset Latch is said to be "event driven" in that the state changes as soon as the inputs R and S change.



<- The diagram we use in class shows the latch built from cross-coupled NOR gates.
-> It can also be built of cross-coupled NAND gates as these two gates are universal. Note how the inputs need to modified to preserve the behavior of the latch.



Example: Set-Reset Latch

Given: In the following truth table, indicate which inputs correspond to which states for the SetReset latch. Suppose the latch is in the Set state and we want to move it to the Reset state, instead. Explain how this scenario might lead us to an unstable state. How can we recover from an unstable state?

Partial Credit 1: The Set-Reset Latch has three defined states and one unstable state. The state depends on the combination of inputs.

Solution 1:

Truth Table		
S	R	State
0	0	__hold__
0	1	__reset__
1	0	__set__
1	1	__unstable__

Example: Set-Reset Latch

Given: In the following truth table, indicate which inputs correspond to which states for the SetReset latch. Suppose the latch is in the Set state and we want to move it to the Reset state, instead. Explain how this scenario might lead us to an unstable state. How can we recover from an unstable state?

Partial Credit 2: The Set-Reset Latch is event driven which means the state changes as soon as the inputs change.

Solution 2: If the current state is Set, that means input S is 1 and input R is 0. If we want to switch to the Reset state, S must be changed to 0 and R must be change to 1. If S changes first the inputs to the Latch will be 1 and 1 which indicates an unstable state.

Example: Set-Reset Latch

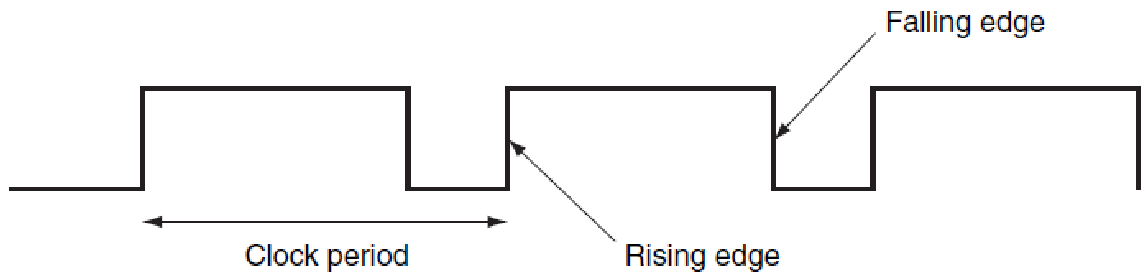
Given: In the following truth table, indicate which inputs correspond to which states for the SetReset latch. Suppose the latch is in the Set state and we want to move it to the Reset state, instead. Explain how this scenario might lead us to an unstable state. How can we recover from an unstable state?

Partial Credit 3: An unstable state occurs when the Latch receives 1 for both of it's inputs. If we try to hold an unstable state, the latch begins to oscillate between 1 and 0 for it's outputs even thought the inputs are unchanging.

Solution 3: To exit an unstable state, we need to supply either a Set input or a Reset input. This will force the latch into the corresponding state and the latch will stabilize.

Review: Clocks

- Clocks are regular periodic signals and the clock period is divided into two portions
  - When the clock is “high” (when the signal is asserted or 1) and when the clock is “low” (deasserted or 0)
  - The transition from low to high is called the rising edge or positive edge
  - The transition from high to low is called the falling edge or negative edge
  - One of these edges is deemed the “active” edge for the system, the other is the offset edge



- We use the clock signal in conjunction with certain gates to control when circuits receive their inputs. Circuits that receive inputs based on the active edge of the clock are said to be synchronous.

Review: Timing Methodology

- We primarily assume positive-edge triggered timing methodology, which means the rising edge or the positive edge is considered the active edge. All inputs and outputs must be stable for a certain period of time around the active edge. These signals are allowed to change and stabilize during the rest of the clock period.

Example: Clocked Set-Reset Latch

Given: Construct a Set-Reset Latch that is synchronous to prevent accidental occurrences of the unstable state.

Partial Credit 1: We can prevent the Set-Reset Latch from accidentally reaching an unstable state by controlling when the Set and Reset signals arrive. To do this, we need a clock and some additional gates. Let's start by building a truth table for the Set signal.

Solution 1: We only want to send S to the Latch when the clock is asserted. When clock is deasserted, we should send a signal that will not harm the Latch. 0 is a safe signal to send since it takes two 1's to enter an unstable state. When the clock is asserted, we want to send the value of S through.

This truth table is equivalent to S AND Clock.

Truth Table		
S	Clock	S
0	0	__0__
0	1	__0__
1	0	__0__
1	1	__1__

Example: Clocked Set-Reset Latch

Given: Construct a Set-Reset Latch that is synchronous to prevent accidental occurrences of the unstable state.

Partial Credit 2: We can also build a truth table for Reset.

Solution 2: The logic for Reset is the same as the logic for Set, so we end up with a similar truth table.

This truth table is equivalent to R AND Clock.

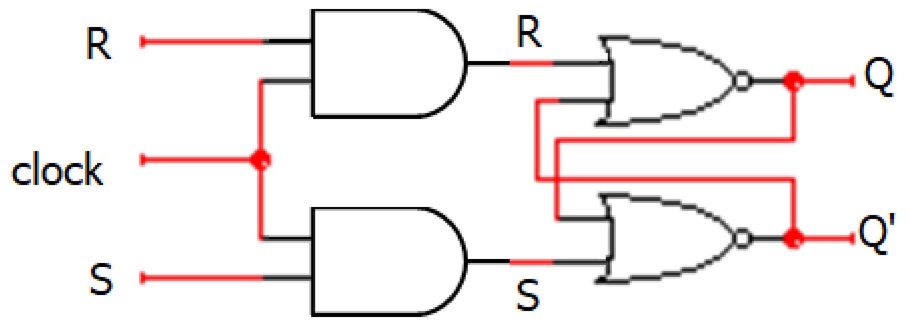
Truth Table		
R	Clock	R
0	0	__0__
0	1	__0__
1	0	__0__
1	1	__1__

Example: Clocked Set-Reset Latch

Given: Construct a Set-Reset Latch that is synchronous to prevent accidental occurrences of the unstable state.

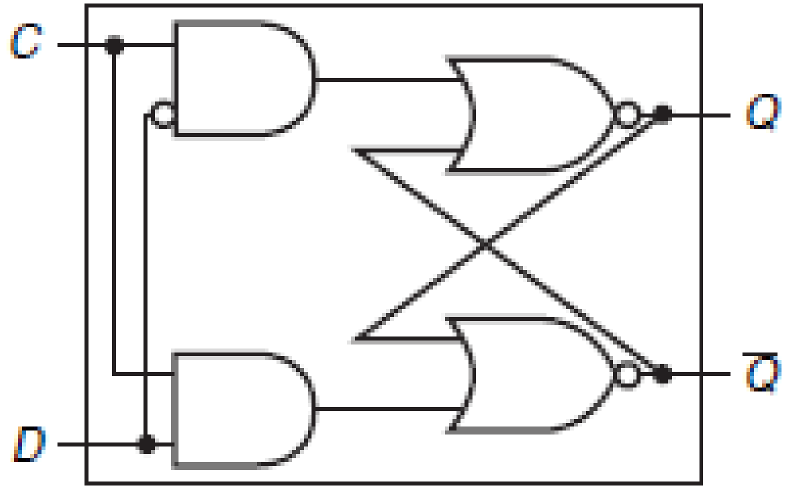
Partial Credit 3: This tells us the circuit can be controlled with the clock signal and two AND gates.

Solution 3: This is one implementation of a clocked Set-Reset Latch. Other implementations include only NOR gates and only NAND gates.



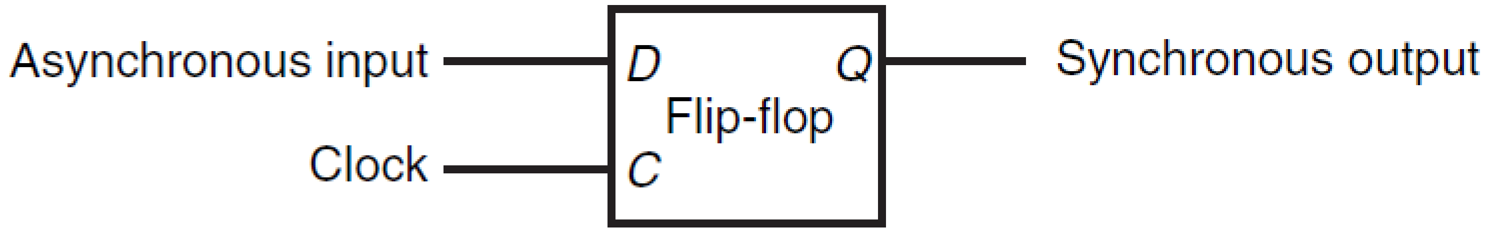
Review: Flip-Flops

- The Data Flip-Flop or D Flip-Flop is a memory element that holds a single bit. This bit may be updated only on the active clock edge. It is build from a clocked latch, where the clock and it's additional gates are arranged in a way that prevents the unstable state from being sent to the latch. The result is that the output Q becomes equal to D whenever the clock is asserted.



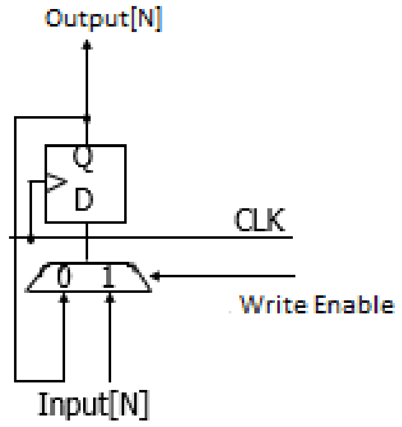
Review: Synchronizers

- Some inputs within computing systems are, by necessity, asynchronous. To maintain a synchronous system, we need to pass these inputs through a synchronizers. The clock signal associated with Flip-Flops controls when these inputs are admitted to the rest of the system.



Review: Registers

- Flip-Flops may be combined in many different ways to create sequential hardware. We can create pattern recognizers, counters, shifters, and registers. The data register we will be using on our data path is comprised of 32 flip-flops joined with similar controls and logic.



Example: Grey-Code Counter

Given: Construct a two-bit Grey-Code counter using D flip-flops and associated logic gates.

Partial Credit 1: Grey Code is unique counting system in which only one bit can change from one bit to the next. If we have two bits to work with our number sequence will be 00, 01, 11, 10. Our counter should cycle through this sequence.

Solution 1: A two-bit counter will require two flip-flops and some associated logic. A truth table will help us see the relationship between the current number of the sequence and the next number to be produced.

Truth Table

Current Bit1	Current Bit0	Next Bit1	Next Bit0
0	0	0	1
0	1	1	1
1	0	0	0
1	1	1	0

Example: Grey-Code Counter

Given: Construct a two-bit Grey-Code counter using D flip-flops and associated logic gates.

Partial Credit 2: Now we can construct the equations for each bit.

Solution 2: Next Bit1 represents the more significant of the two bits. Next Bit1 = Current Bit0. The more significant of the two bits should be equal to the less significant of the two bits from the previous number in the sequence.

Truth Table			
Current Bit1	**Current Bit0**	**Next Bit1**	**Next Bit0**
0	0	0	1
0	1	1	1
1	0	0	0
1	1	1	0

Example: Grey-Code Counter

Given: Construct a two-bit Grey-Code counter using D flip-flops and associated logic gates.

Partial Credit 3: Now we can construct the equations for each bit.

Solution 3: Next Bit0 represents the less significant of the two bits. Next Bit0 = Current Bit1'. The less significant of the two bits should be equal to the negation of the more significant of the two bits from the previous number in the sequence.

Truth Table

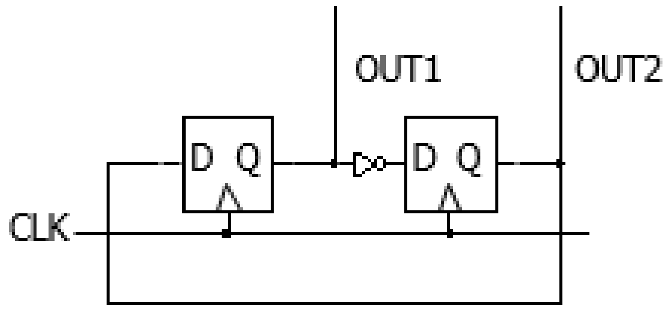
Current Bit1	Current Bit0	Next Bit1	Next Bit0
0	0	0	1
0	1	1	1
1	0	0	0
1	1	1	0

Example: Grey-Code Counter

Given: Construct a two-bit Grey-Code counter using D flip-flops and associated logic gates.

Partial Credit 4: Now we can construct the diagram for the counter:
Next Bit1 = Current Bit0.
Next Bit0 = Current Bit1'.

Solution 4:


OUT1
OUT2
D Q
D Q
CLK