Fundamentals of Digital Circuits

Lecture 69, v01

A description of basic techniques useful in developing Simulink models

Purpose

- When developing state-based simulation models in Stateflow, it is often necessary to first construct a representation of the “environment” that the system operates within
- Such an environment would present the signals that describe the conditions existing in the environment and that are used as inputs to the state-based simulation
- The simulation would then provide responses back to the environment that represent the system’s influence on the environment
- This lecture covers a minimal amount of information about the subject of Digital Circuitry that you, as a Systems Modeler, would need to construct a digital representation of the simulation environment in Simulink
- It is not expected that you would become an expert in modeling of digital circuits with this information, but that you would have sufficient information to construct the digital circuitry elements of a basic model of a simulation environment

Digital Modeling Techniques Presented

- As a systems modeler, it is up to you to understand the environment being modeled in Simulink, and to chose the Simulink modeling elements and techniques that can best represent the environment in a Simulink model
- There almost always exists multiple ways to represent a given problem (an environment) in Simulink
  - There is no single modeling solution
  - Developing a model requires knowledge and experience on the part of the modeler to determine what are the appropriate modeling elements and techniques to use
  - The elements and techniques shown here are just a starting point
  - A modeler would build up an inventory of elements and techniques over time that could be used when constructing a model
  - As the modeler develops experience, new elements and techniques can be added to such an inventory

Objectives

- In this lecture, we will examine 3 types of digital circuits with the following definitions
  - Latches – These hold their output value until the input is changed (level-triggered)
  - Enabled (or gated) latches – The timing of output changes are controlled by an additional input signal (Enable signal)
  - Flip flops – The timing of output changes are controlled by a clock signal (edge-triggered)
    - Rising-Edge Triggered
    - Falling-Edge Triggered
- The goal of this lecture is to
  - Understand what digital circuits are
  - Understand how to use the digital circuit models available in Simulink
- In addition, a few other related subjects are briefly mentioned
  - Digital ciruits with additional Preset (PRE) and Clear (CLR) inputs
  - Pulse-Triggered (Master-Slave) circuits
  - Designing circuits based on a truth table specification using Karnough Mapping

Index of Topics

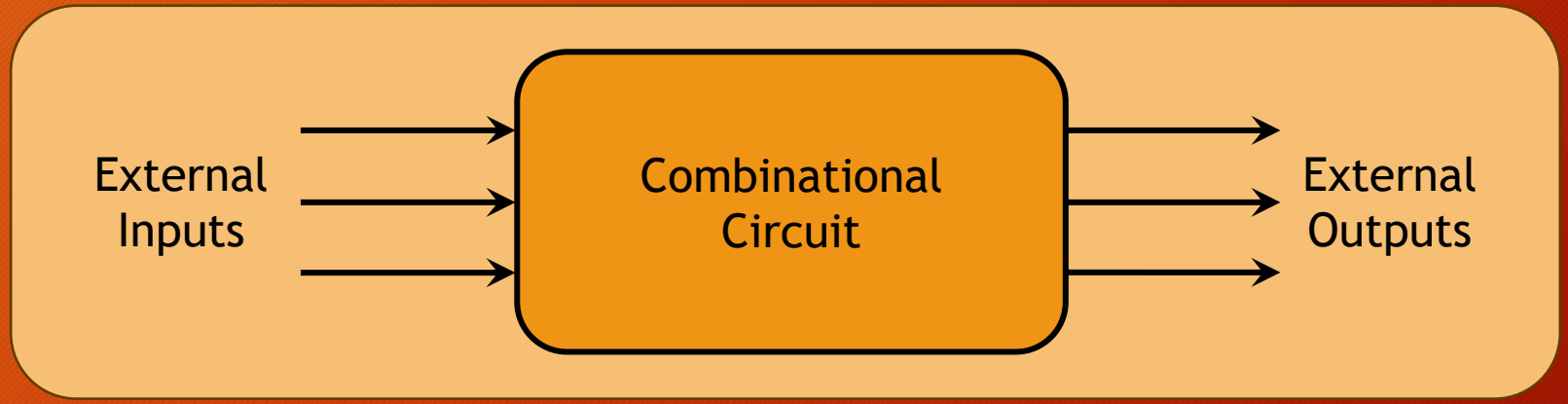
To see applicable Simulink Blocks
Go directly here

Combinational Circuits

Return to Index

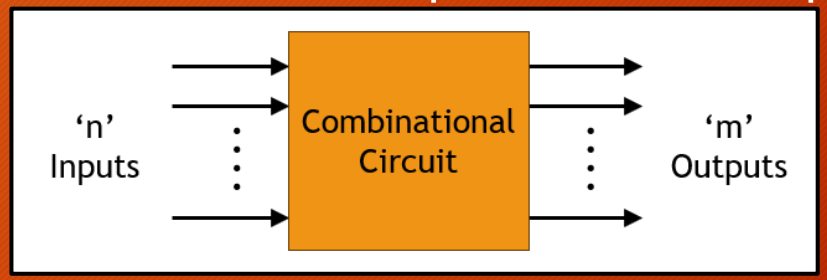
Combinational Circuit Description

- Combinational circuits are time independent circuits which do not depend upon previous inputs to generate an output
- The external outputs depend only on
  - The present external inputs
  - The logical rules of the combinational circuit
- The outputs are the result of processing of the inputs by the internal logic of the combinational circuit



Combinational Circuits Description (continued)

- A combinational circuit consists of an arrangement of logic gates
  - AND gate
  - OR gate
  - XOR gate
  - NOT gate
  - NAND gate
  - NOR gate
  - NXOR gate
- These circuits operate with binary (1/0 or T/F) input and output values
- The outputs of a combinational circuit depend on the combination of present (current) inputs
  - The typical combinational circuit has ‘n’ inputs and ‘m’ outputs



  - Each combination of inputs will affect the outputs
- The output of a combinational circuit at any instant of time, depends only on the levels present at input terminals
  - Combinational circuits do not use any memory
  - The previous state of the inputs does not have any effect on the present state of the circuit

Design of Combinational Circuits

- Design procedure of Combinational circuits
  - Find the required number of inputs and outputs from a given specification
  - Formulate a Truth table
    - If there are ‘n’ inputs, then there will be 2^n possible combinations
  - For every combination of inputs, find the output values
  - Find the Boolean expressions for each output
    - Each output expression will be expressed as a function of all n inputs
  - Implement the above Boolean expressions corresponding to each output by using some combination of Logic gates

Truth Table for 2 Inputs

Inputs		Outputs			
X	Y	1	2	…	m
T	T	T	F	T	F
T	F	F	T	T	T
F	T	T	T	F	F
F	F	F	F	T	T

$2^2 = 4$

Truth Table for 3 Inputs

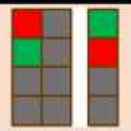
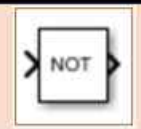
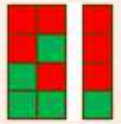
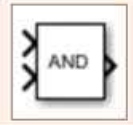
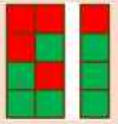
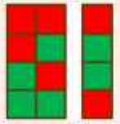
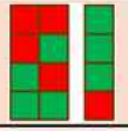
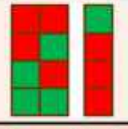
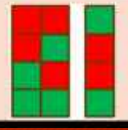
Inputs			Outputs			
X	Y	Z	1	2	…	m
T	T	T	T	F	T	F
T	F	T	F	T	T	T
F	T	T	T	T	F	F
F	F	T	F	F	T	T
T	T	F	F	T	F	T
T	F	F	T	F	T	T
F	T	F	T	F	F	T
F	F	F	T	T	T	F

$2^3 = 8$

EXAMPLE

These are bogus examples
The Boolean expressions that produce the outputs according to the various input combinations are not shown

Common Logic Gates

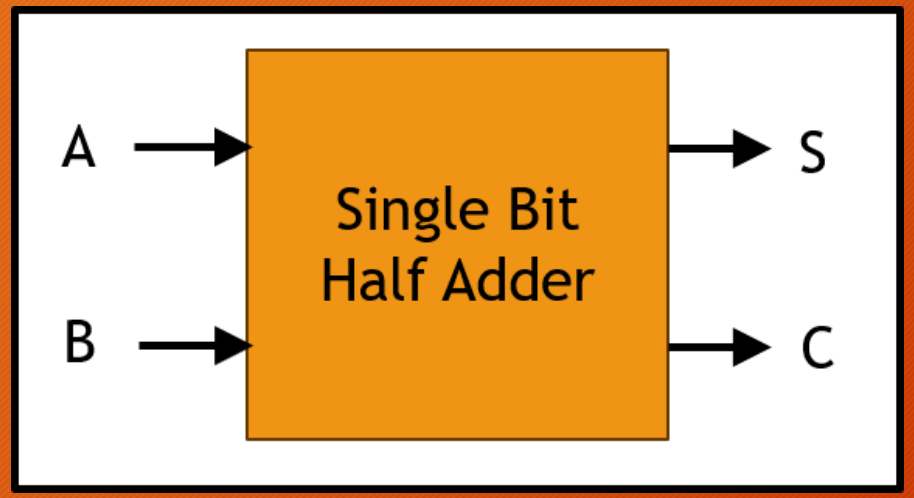
Gate Type	Formal Name	Meaning	Math Symbol	Truth Table (IN A B / OUT)	Simulink Classic Icon	Simulink Rectangular Icon
NOT	Negation	The opposite truth value	~A			
AND	Conjunction	Both inputs must be true for the output to be true	(A∧B)			
OR	Disjunction	Either of the inputs can be true for the output to be true	(A∨B)			
XOR	Exclusion	Only one of the inputs must be true for the output to be true	(A⊕B)			
NAND	Negation of Conjunction	Negation of the result of an AND operation	~(A∧B)			
NOR	Negation of Disjunction	Negation of the result of an OR operation	~(A∨B)			
NXOR	Negation of Exclusion	Negation of the result of an XOR operation	~(A⊕B)			

LEGEND: = TRUE (1) = FALSE (0) = N/A

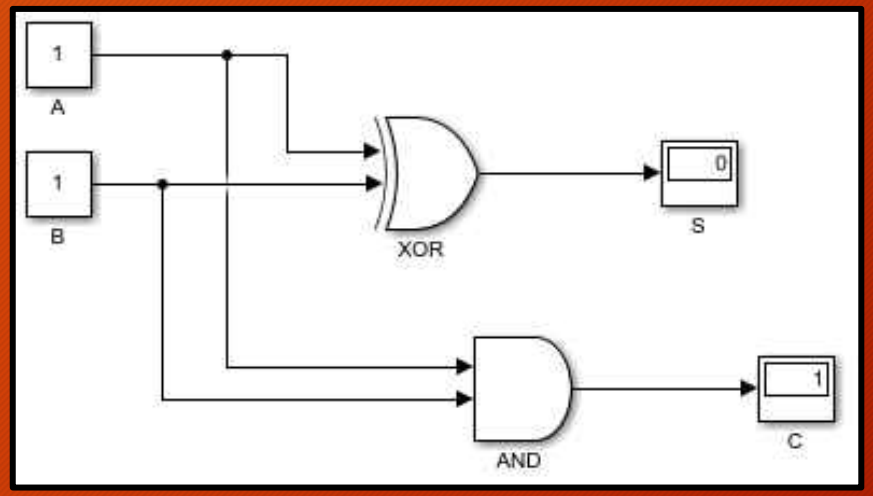
Example Combinational Circuit - Half Adder

- The Half Adder is a combinational logic circuit designed to add two single bit binary numbers A and B
  - This circuit has two inputs: The first addend (A), and the second addend (B)
  - This circuit has two outputs: Sum (S) and Carry (C)

Circuit Schematic



Simulink Model



Truth Table

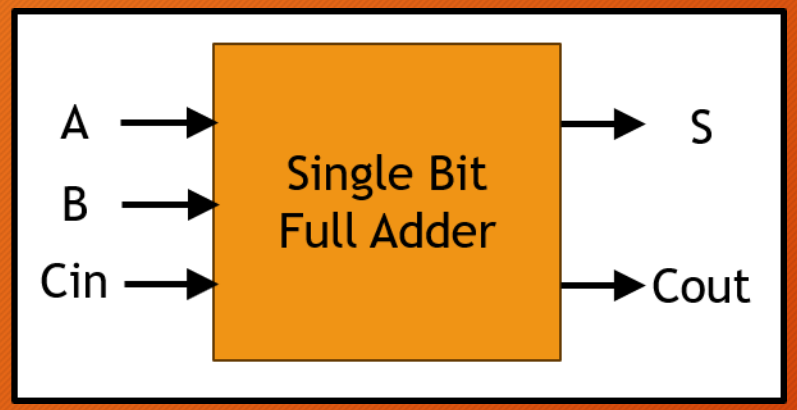
Inputs		Outputs	
A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

- This circuit is not designed to be stacked one on top of another, since the carry bit is not accepted as an input

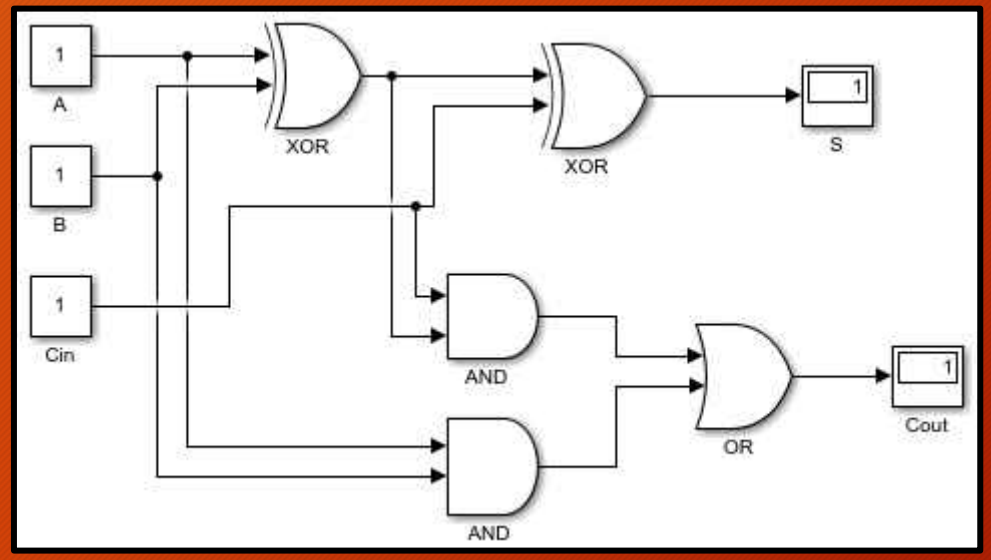
Example Combinational Circuit - Full Adder

- The Full Adder was developed to overcome the drawback of the Half Adder circuit in which the Half Adder cannot accept the carry bit from a previous operation
  - The Full Adder is a three input and two output combinational circuit
  - The Full Adder can add two one-bit numbers A and B, as well as the carry value from a previous adder operation

Circuit Schematic



Simulink Model



Truth Table

Inputs			Outputs	
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- The Full Adder will accept the carry bit from a previous operation

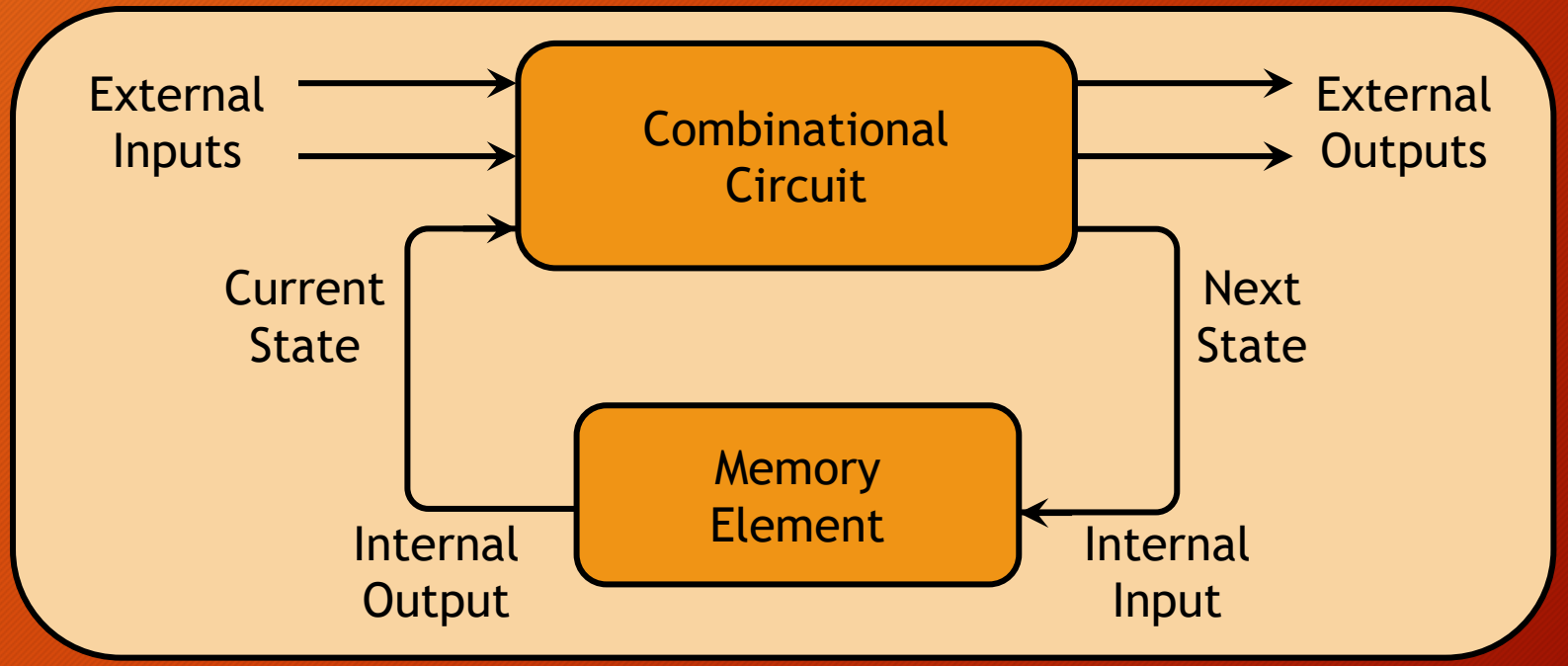
These are just two out of countless Combinational Circuit types

Sequential Circuits

Return to Index

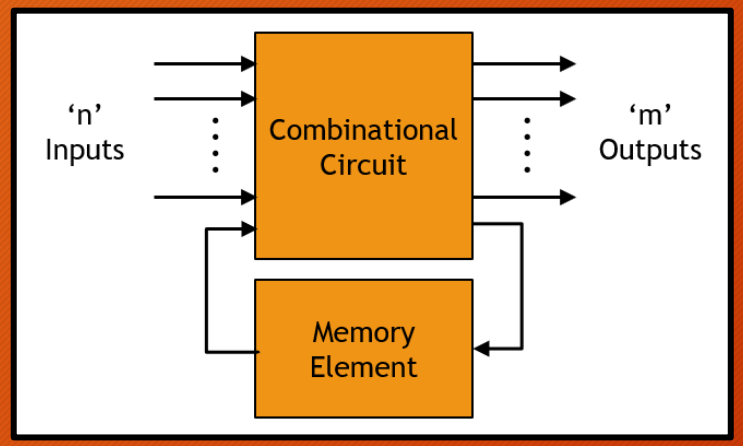
Sequential Circuit

- Consists of a Combinational Circuit in coordination with a Memory Element
- Outputs depend on both the present external inputs and the current state
  - The present inputs are those that are currently present on the external input lines
  - Whereas, the present state
    - Has already been determined in the previous cycle from the past inputs
    - Is stored in the memory element



Sequential Circuits

- This sequential circuit contains a set of inputs and outputs
  - The outputs of a sequential circuit depend not only on the combination of present inputs but also on the previous outputs
  - Previous output is the same as the present (current) state prior to execution of the circuit
  - To have access to the previous output, sequential circuits contain memory storage elements
  - While not common, some sequential circuits may not contain combinational circuits, but only memory elements
- The outputs of a sequential circuit depend on the combination of present (current) inputs
  - The typical sequential circuit has ‘n’ input variables and ‘m’ outputs along with memory capacity



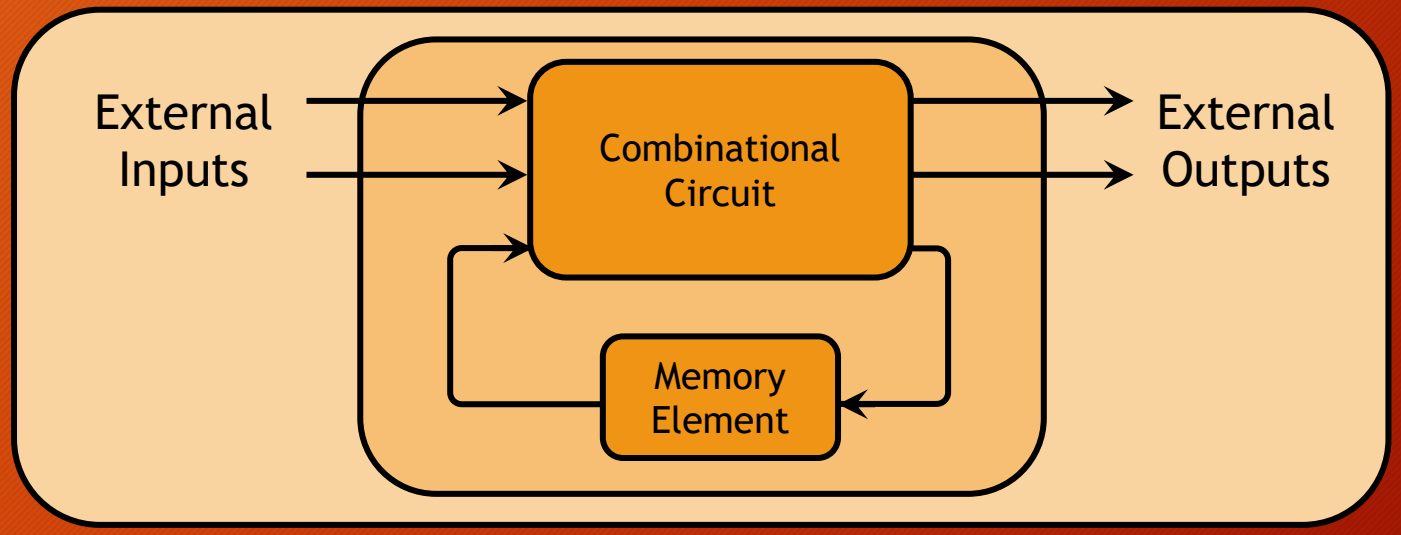
Combinational Circuits	Sequential Circuits
Outputs depend only on present inputs	Outputs depend on both present inputs and present state
Feedback path is not present	Feedback path is present
Memory elements are not required	Memory elements are required
Clock signal is not required	Clock signal is required
Easy to design	Difficult to design

Asynchronous Sequential Circuits

Return to Index

Asynchronous Sequential Circuits

- If some or all the outputs of a sequential circuit do not change output with respect to active transition of clock signal, then that sequential circuit is called as Asynchronous sequential circuit
- This means, all the outputs of asynchronous sequential circuits do not change output at the same time
- Therefore, most of the outputs of asynchronous sequential circuits will not be synchronous with either only positive edges or only negative edges of a clock signal



Asynchronous Sequential Circuit

Latches

- Latches are asynchronous sequential circuits that store a single bit of information and hold its value until it is updated by new input signals
  - They are used as temporary storage elements to store binary information
  - They operate based on signal levels (rather than signal transitions)
  - They are sensitive to the input voltage applied and do not depend on the clock pulse
  - They are useful in the design of asynchronous (time-independent) sequential circuits
- Latches have two stable output states
  - These output states are sensitive to the input voltage (level) applied and do not depend on a clock pulse
- The two types of latches we will examine here are:
  - Set-Reset (S-R) latch – Has an indeterminate output state for certain inputs
  - Data (D) latch – Corrects the indeterminate output state of the S-R Latch
  - J-K Latch – Also can toggle the output, irrespective of the previous output state
  - Toggle (T) Latch – Similar performace as D Latch

S-R Latch

Return to Index

S-R Latch

- S-R latches are the simplest form of latches and are implemented using two inputs:
  - S = Set (AKA Preset or PRE)
  - R = Reset (AKA Clear or CLR)
- In this truth table, Qn-1 is the output at the previous time step
- When both S and R are 0, the latch stays in the previous state
  - (Qn is Qn-1)
- When S is 1 and R is 0, the latch goes to the set state (Qn is 1)
- When R is 1 and S is 0, the latch goes to the reset state (Qn is 0)
- NOTE: Avoid the state where S and R are both 1
  - In this state, both Q and !Q are 0
  - This state is undefined because !Q (0) is not the complement of Q (0)
- Static 0 hazards can set/reset the S-R Latch
  - A glitch on the S input can accidentally set the latch
  - A glitch on the R input can accidentally reset the latch

Truth Table

Inputs		Outputs	
S	R	Q_n	$!Q_n$
0	0	Q_{n-1}	$!Q_{n-1}$
0	1	0	1
1	0	1	0
1	1	UND	UND

A static hazard is a condition where a single input variable change (a glitch) produces a momentary output change where no output change should occur

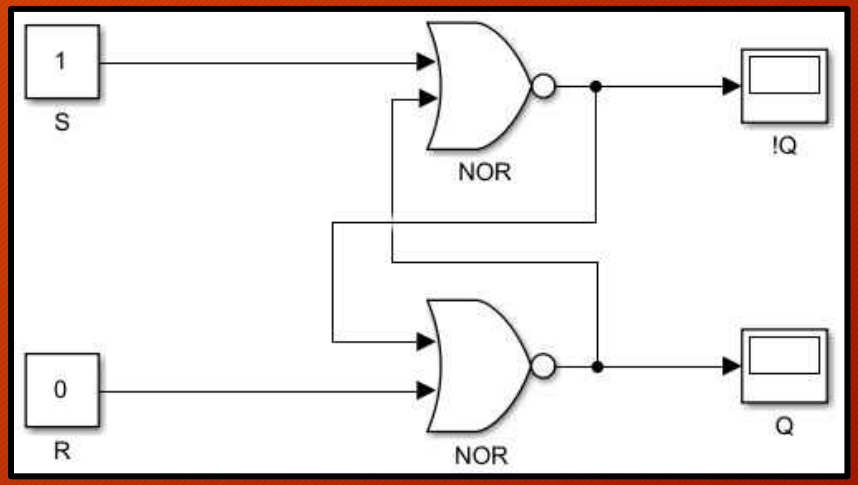
Glitches can occur due to delays in one path of two or more parallel paths causing the unexpected glitch to appear momentarily

Static hazards are classified as either static 1 hazards or static 0 hazards

Set-Reset (S-R) Latch (NOR Version)

- This S-R Latch is constructed using NOR gates
- NOTE: Avoid the state where R and S are both 1
  - In this state, both Q and !Q are 0
  - This state is undefined because !Q (0) is not the complement of Q (0)
  - If this state is important to the design, consider using the J-K Flip-Flop block instead

S-R Latch Schematic
(NOR Version)



This schematic is for illustrative purposes only
It will not actually execute in Simulink

Truth Table

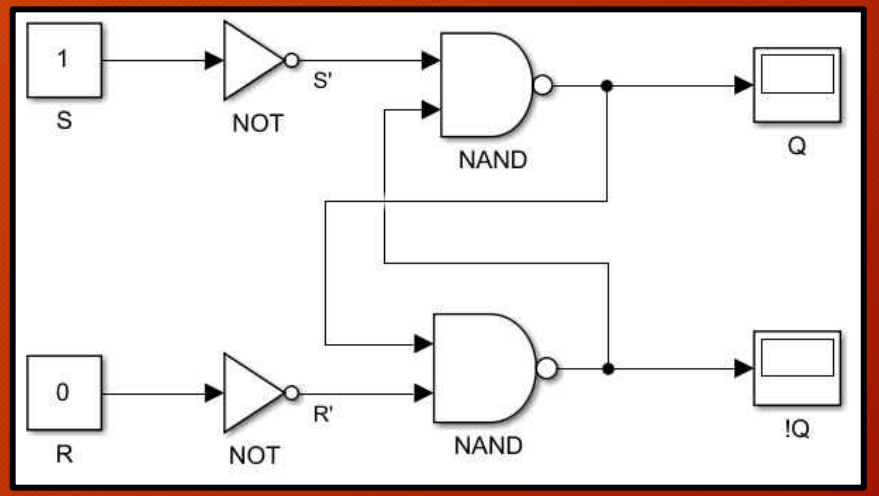
Inputs		Outputs	
S	R	Q_n	$!Q_n$
0	0	Q_{n-1}	$!Q_{n-1}$
0	1	0	1
1	0	1	0
1	1	UND	UND

- In this truth table, Qn-1 is the output at the previous time step
- When S is 1 and R is 0, the latch goes to the set state (Qn is 1)
- When R is 1 and S is 0, the latch goes to the reset state (Qn is 0)
- When both S and R are 0, the latch stays in the previous state (Qn is Qn-1)

Set-Reset (S-R) Latch (NAND Version)

- This S-R Latch is constructed using NAND gates
- The inputs S and R are negated (S' and R')in order to provide the correct inputs to the NAND gates that will result in the correct results
- NOTE: Avoid the state where S and R are both 1
  - In this state, both Q and !Q are 0
  - This state is undefined because !Q (0) is not the complement of Q (0)
  - If this state is important to the design, consider using the J-K Flip-Flop block instead

S-R Latch Schematic
(NAND Version)



- This schematic is for illustrative purposes only
- It will not actually execute in Simulink

Truth Table

Inputs		Inputs'		Outputs	
S	R	S'	R'	Q_n	$!Q_n$
0	0	1	1	Q_{n-1}	$!Q_{n-1}$
0	1	1	0	0	1
1	0	0	1	1	0
1	1	0	0	UND	UND

- In this truth table, Qn-1 is the output at the previous time step
- When S is 1 and R is 0, the latch goes to the set state (Qn is 1)
- When R is 1 and S is 0, the latch goes to the reset state (Qn is 0)
- When both S and R are 0, the latch stays in the previous state (Qn is Qn-1)

D Latch

Return to Index

Data (D) Latch

- In the S-R Latch there is a restricted input condition
  - The S and R inputs should not both be the same value
  - One or the other should be high to represent either S (set) or R (reset)
- There is no need to be concerned about the states where S and R are either both 0, or both 1
  - It is impossible to reach these input conditions
  - The single input D is split into two inputs R and S
  - An inverter is connected to the R input to provide S
- The D Latch solves the static 0 hazard of the S-R Latch
- This D Latch is constructed using NAND gates
- The inputs S and R are negated (S' and R') in order to provide the correct inputs to the NAND gates that will result in the correct results

Truth Table

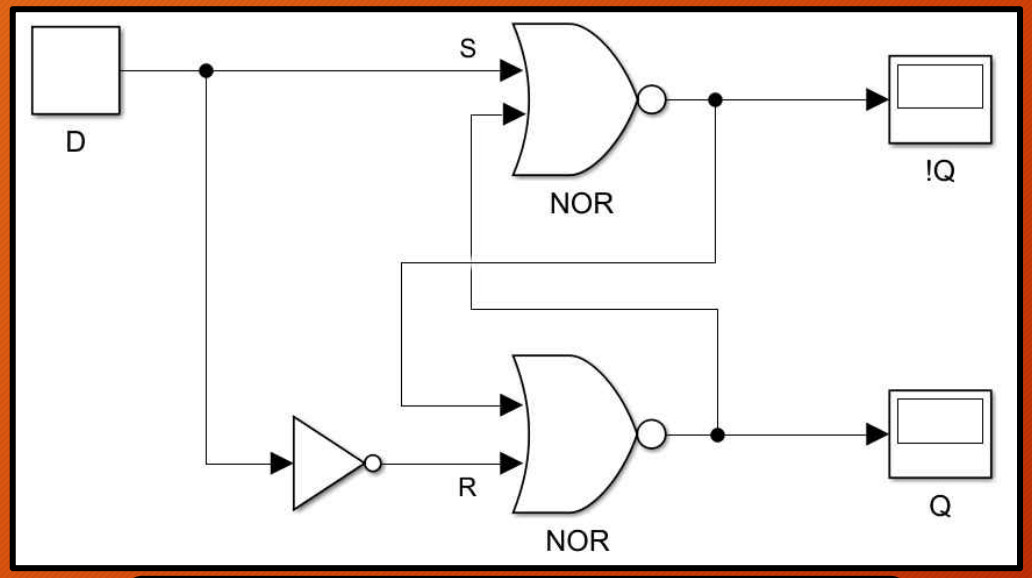
Inputs			Outputs	
D	S	R	Q	!Q
0	0	1	0	1
1	1	0	1	0

- The D Latch is like an S-R Latch in which the inputs are forced to be inverted copies of each other
- S is equal to D and R is equal to D'

Data (D) Latch

- S (Set) has the same value as D (Data)
- R (Reset) is ALWAYS the inverted state of S
- This latch avoids the issues of incompatible values for S and R in the S-R Latch

D Latch Model



- This schematic is for illustrative purposes only
- It will not actually execute in Simulink

Truth Table

Inputs			Outputs	
D	S	R	Q	!Q
0	0	1	0	1
1	1	0	1	0

J-K Latch

Return to Index

J-K Latch

- The J-K latch has two inputs J and K
  - The J-K latch is similar to the S-R latch
    - But it eliminates the undefined state of S-R latch
  - Instead, when both J and K = 1, then the output states simply toggle from their previous states

Truth Table

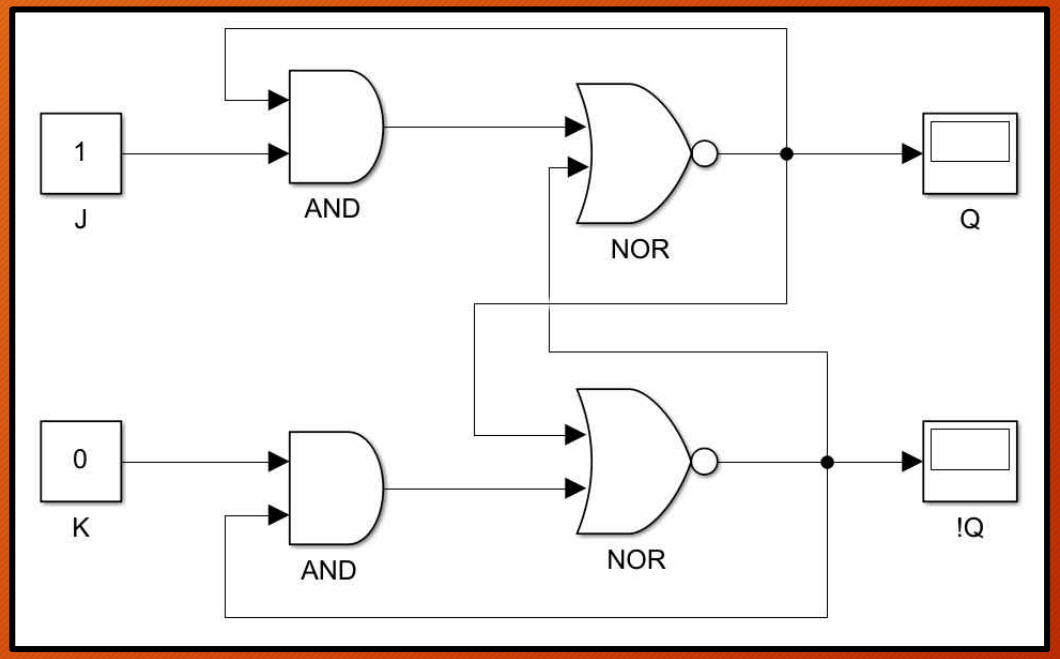
Inputs		Outputs	
J	K	Q_n	$!Q_n$
0	0	Q_{n-1}	$!Q_{n-1}$
0	1	0	1
1	0	1	0
1	1	$!Q_{n-1}$	Q_n

Simulink has no native model for a J-K Latch

J-K Latch Circuit

- J = K = 0 simply keeps the last output active
- J = K = 1 simply toggles the last output

J-K Latch Model



- This schematic is for illustrative purposes only
- It will not actually execute in Simulink

Truth Table

Inputs		Outputs	
J	K	Q_n	$!Q_n$
0	0	Q_{n-1}	$!Q_{n-1}$
0	1	0	1
1	0	1	0
1	1	$!Q_{n-1}$	Q_n

T Latch

Return to Index

T (Toggle) Latch

- The inputs of the J-K Latch are shorted to create the T Latch
- The T Latch output switches on and off when the input is set to 1 or high
- When the input of T is 0 then the output will retain the same (no change)
- The circuit diagram and truth table of the T latch is shown below

Truth Table

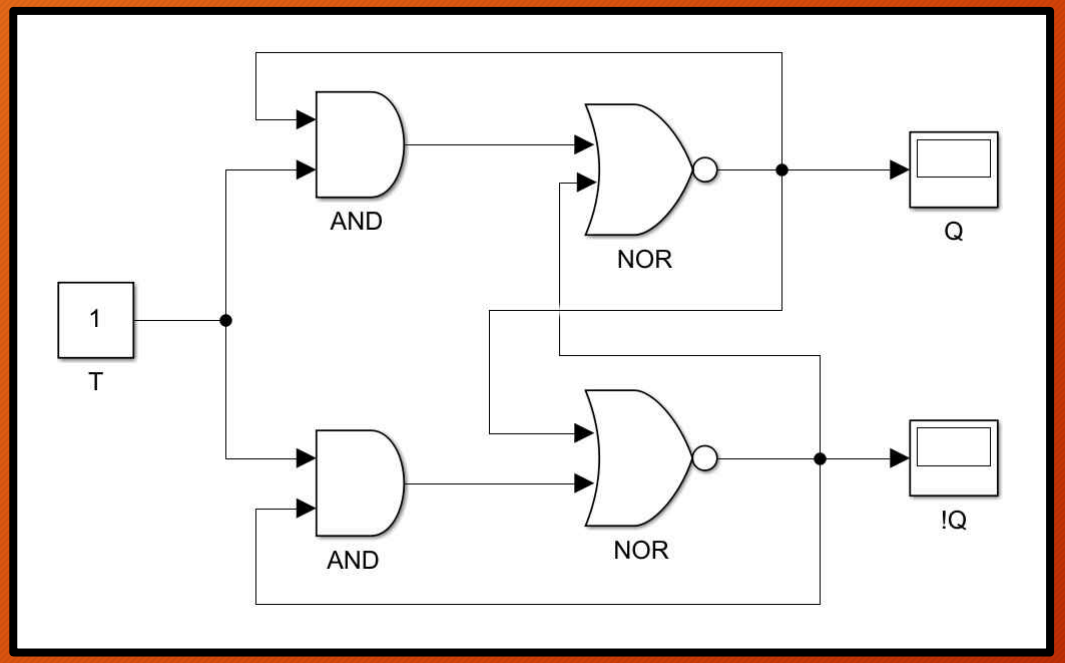
Inputs	Outputs	
T	Q_n	$!Q_n$
0	0	1
1	1	0

Simulink has no native model for a T Latch

T Latch Circuit

- The T Latch simply toggles the two outputs
  - Each output is the compliment of the other at all times

T Latch Model



- This schematic is for illustrative purposes only
- It will not actually execute in Simulink

Truth Table

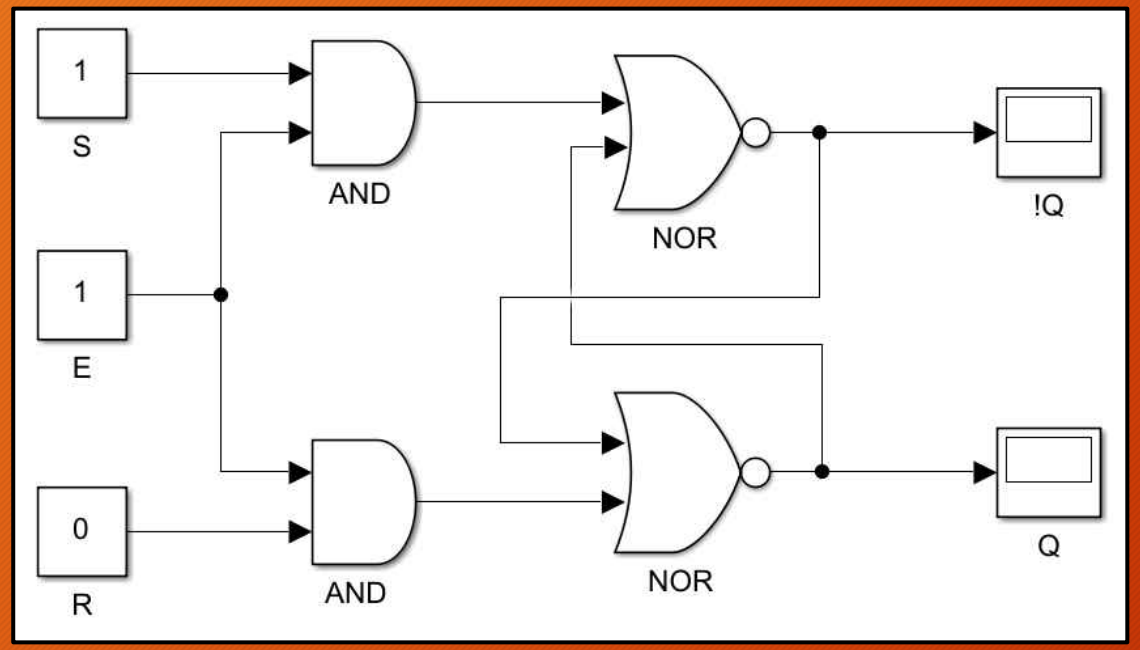
Inputs	Outputs	
T	Q_n	$!Q_n$
0	0	1
1	1	0

Enabled Versions of Latches

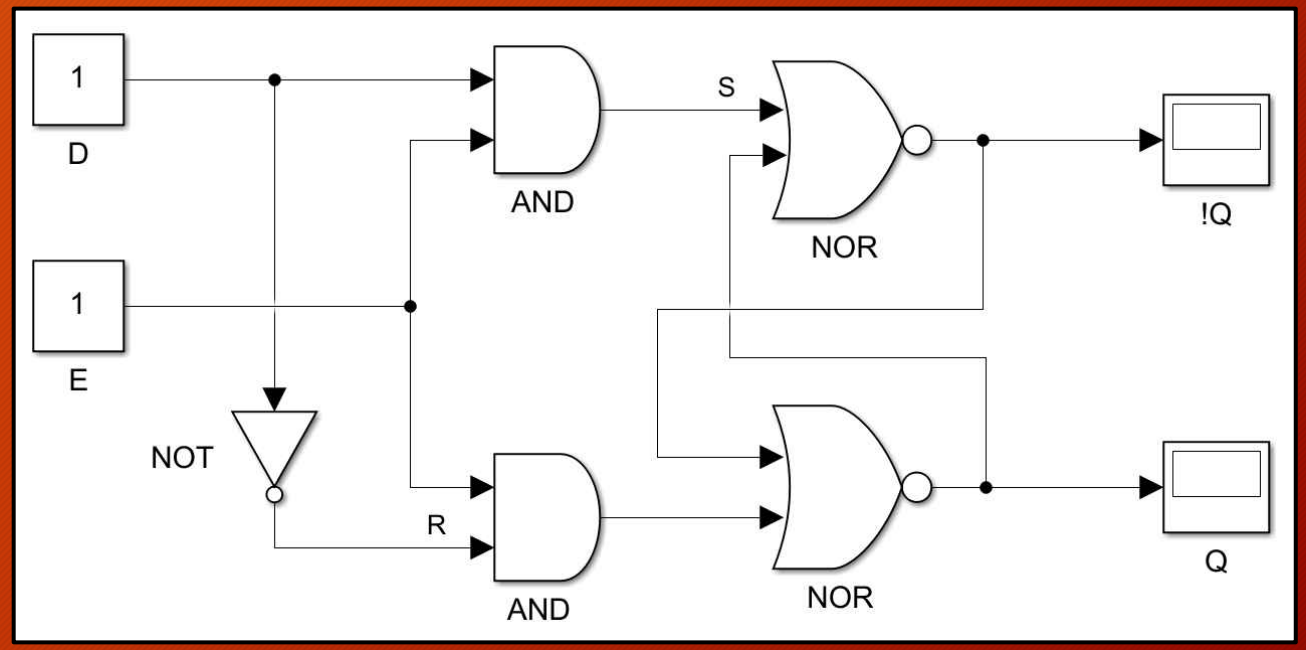
Return to Index

Enabled Versions of Latches

- Each of the latches described above have “enabled” versions which allow a control signal to enable/disable the circuit
- This is done simply by adding AND gates that turn on/off the circuitry controlled by the new “enable” signal



Enabled S-R Latch

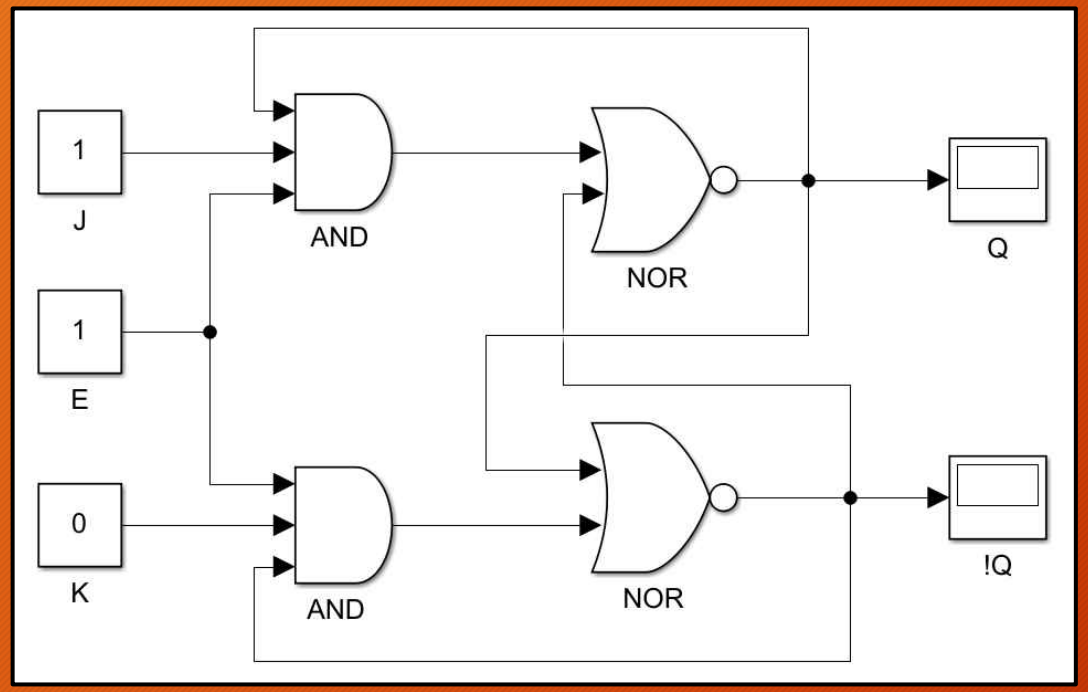


Enabled D Latch

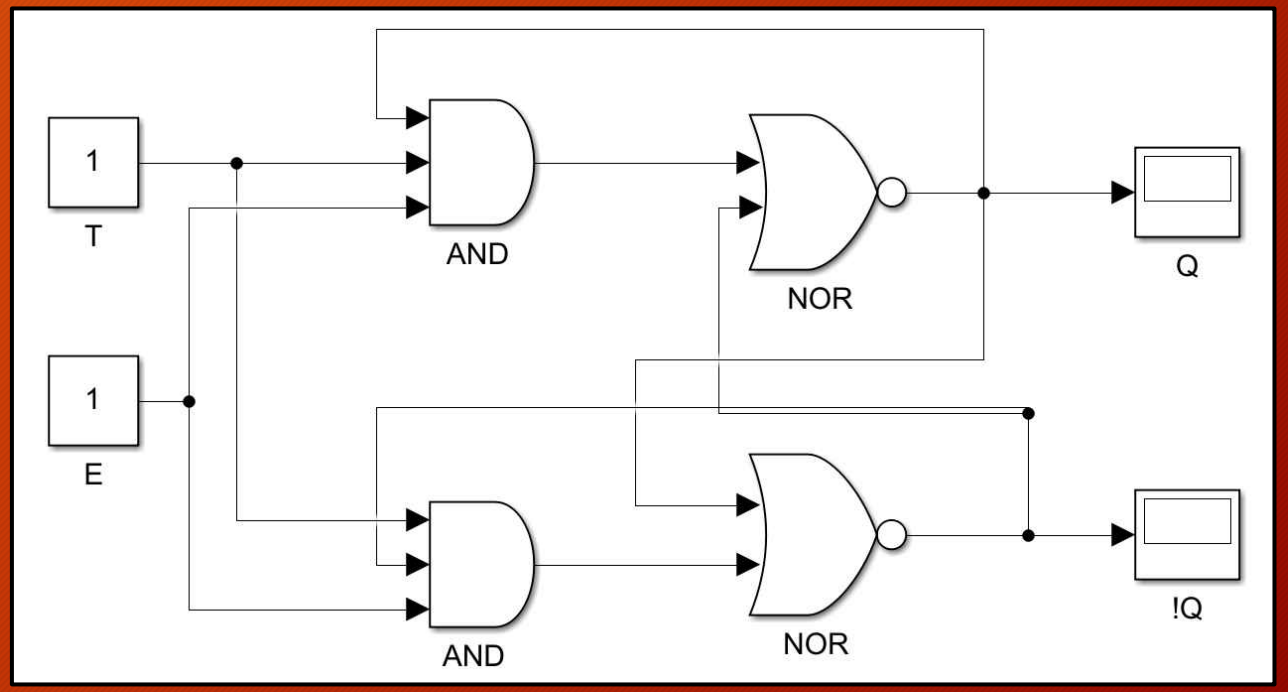
- These schematics are for illustrative purposes only
- They will not actually execute in Simulink

Enabled Versions of Latches (continued)

- Each of the latches described above have “enabled” versions which allow a control signal to enable/disable the circuit
- This is done simply by adding a third input port to the existing AND gates so that they turn on/off the circuitry controlled by the new “enable” signal



Enabled J-K Latch



Enabled T Latch

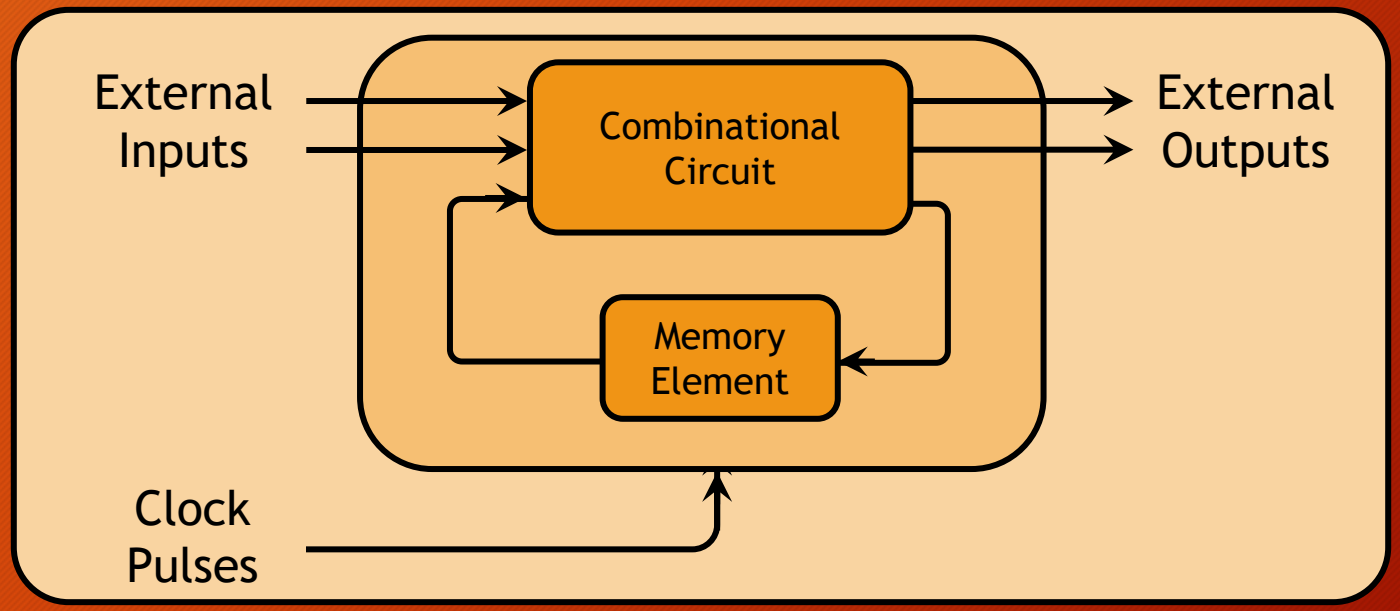
- These schematics are for illustrative purposes only
- They will not actually execute in Simulink

Synchronous Sequential Circuits

Return to Index

Synchronous Sequential Circuits

- If all the outputs of a sequential circuit change output with respect to the same active transitions of a clock signal, then that sequential circuit is called a Synchronous sequential circuit
- This means, all the outputs of synchronous sequential circuits change output at the same time
- Therefore, the outputs of synchronous sequential circuits will be synchronous with either only positive edges or only negative edges of a clock signal



Synchronous Sequential Circuit

Latches and Flip-Flops

- Both are Sequential Circuits with memory devices
  - They store the value of the sampled data
- The difference is in the way they are triggered to sample and store data
  - Latch
    - The output responds to a change in the sampled data (input) immediately
    - Thus, the memory element is asynchronous (not time based)
    - “Gated” latches will only perform the sampling (reading) operation if “enabled” (by the gate)
  - Flip-Flop
    - Reads the sampled data only when triggered by a clock signal (with pulses having a regular frequency)
    - The sampled data (input) can change often, but will only be read at the rate of the clock signal
    - Thus, the memory element is synchronous (time based)

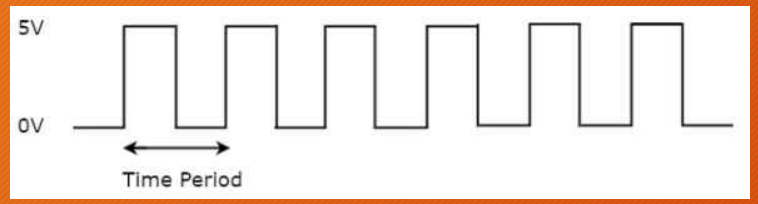
- There is disagreement in the literature about what distinguishes a latch from a flip flop
- This lecture makes the distinction that if the enable signal is not a regularly repeating pulse, then the circuit is a latch
- If the enable signal signal is a regularly repeating pulse, then the circuit is a flip flop

Clock Signal Triggering

- The outputs of latches are constantly affected by their inputs as long as the enable signal is asserted
  - When they are enabled, the output of latches changes immediately when their inputs change
- The output of flip-flops instead change by a clock signal trigger
- Flip-flops employ two different types of triggering methods to start a signal transition from one output state to another
  - Level triggering
  - Edge triggering
  - Both are used to regulate the timing of events in the system
- After the output state change is triggered by the clock, the flip-flop output remains constant until the next clock trigger, even if the input changes

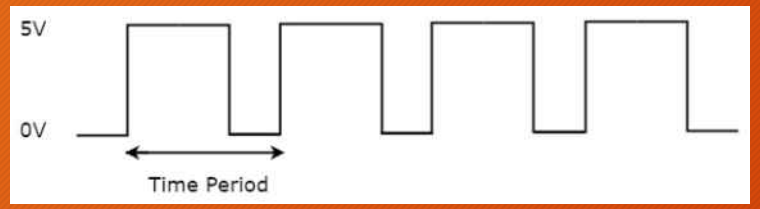
Clock Signal

- The clock signal is a periodically repeating signal whose ON time and OFF time need not be the same
  - A clock signal can be represented as a square wave, in which both its ON and OFF times are same



- This signal stays at logic High (5V) for some time and stays at logic Low (0V) for equal amount of time
- This pattern repeats with some time period
- In this case, the time period will be equal to either twice the ON time or twice the OFF time

  - Alternatively, the clock signal can be represented as train of pulses, whose ON time and OFF time are not same



- This signal stays at logic High (5V) for some time and stays at logic Low (0V) for some other time
- This pattern repeats with some time period
- In this case, the time period will be equal to sum of ON time and OFF time

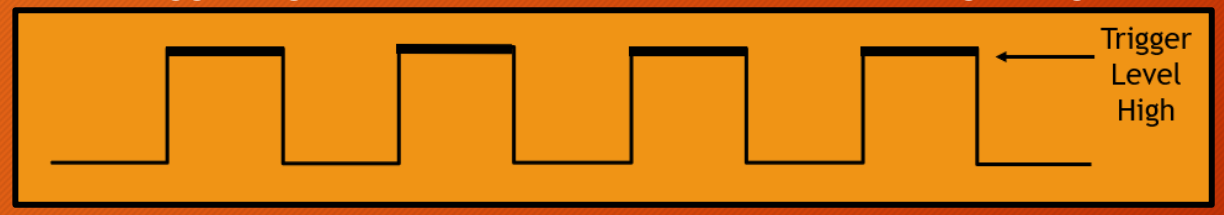
- The reciprocal of the time period of clock signal is known as the frequency of the clock signal
- All sequential circuits are operated with a clock signal
- The frequency at which the sequential circuits will operated is determined by the circuit's clock signal frequency

Level Triggering

- Level triggering is based on the detection of a particular signal level
- It is applied when detecting the input signal level at a specific moment rather than a change in state is required
- Level triggering is frequently utilized in applications where continuous monitoring of an input signal is necessary
- The output signal is activated when the input signal reaches a predetermined level
- Depending on the circuit's design, this level may be high or low

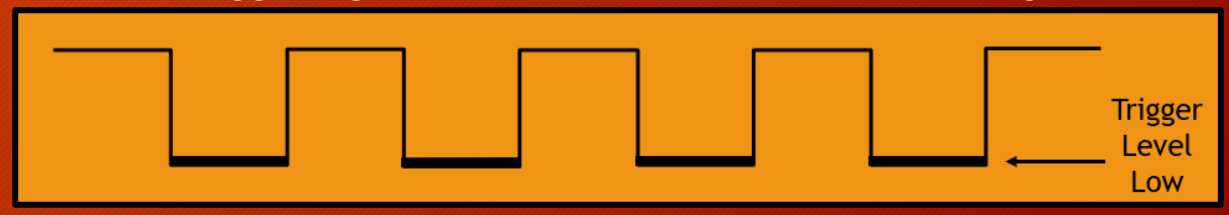
Positive level triggering

Triggering ocurrs when the clock reaches Logic High



Negative level triggering

Triggering ocurrs when the clock reaches Logic Low

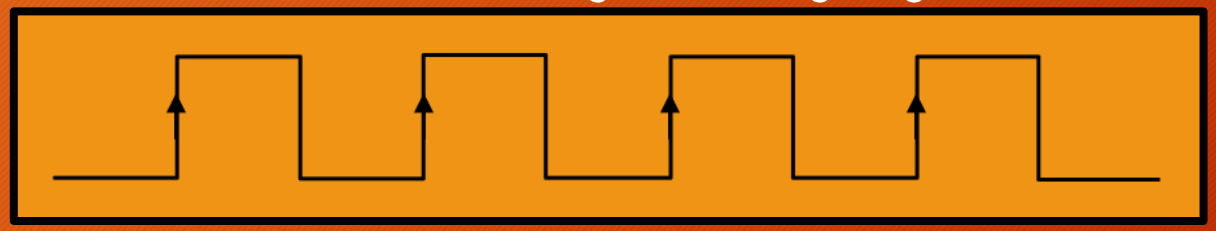


The logic circuit designer selects the logic circuit components that have the triggering type that is needed

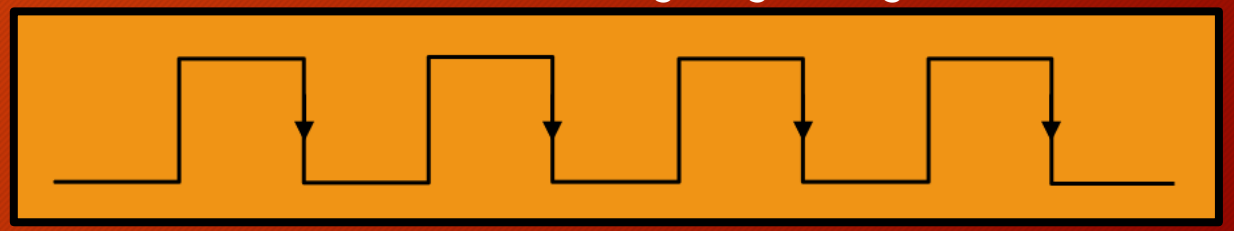
Edge Triggering

- This is a triggering mechanism that is based on the recognition of a change in the signal level from low-to-high or high-to-low
- Edge triggering is frequently employed in situations when exact timing is necessary
- In edge triggering, the input signal's sharp edge triggers the signal change
- The trigger edge is this abrupt edge, and depending on how the circuit is constructed, it may be rising or falling
- The circuit reacts by switching the output signal to the opposing state when it detects the trigger edge

Positive edge triggering
Also called rising edge triggering
Triggering ocurrs when the clock signal transitions from Logic Low to Logic High



Negative edge triggering
Also called falling edge triggering
Triggering ocurrs when the clock signal transitions from Logic High to Logic Low



The logic circuit designer selects the logic circuit components that have the triggering type that is needed

Examples of Types of Triggering

Asynchronous

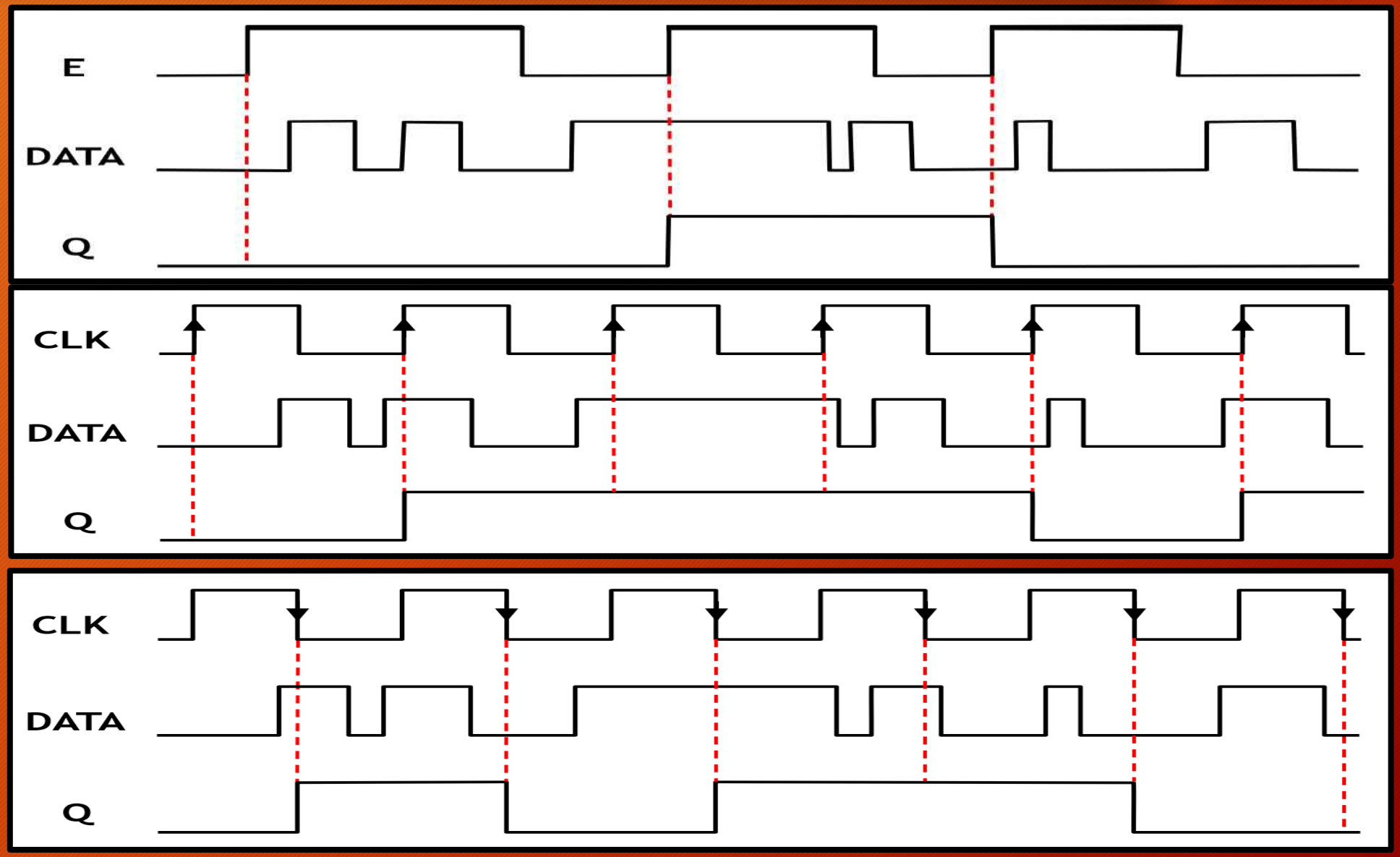
Level Triggering
Data is sensed when enable signal level goes high

Synchronous

Rising-Edge Triggering
Data is sensed when clock is rising

Synchronous

Falling-Edge Triggering
Data is sensed when clock is falling



Flip Flops

Return to Index

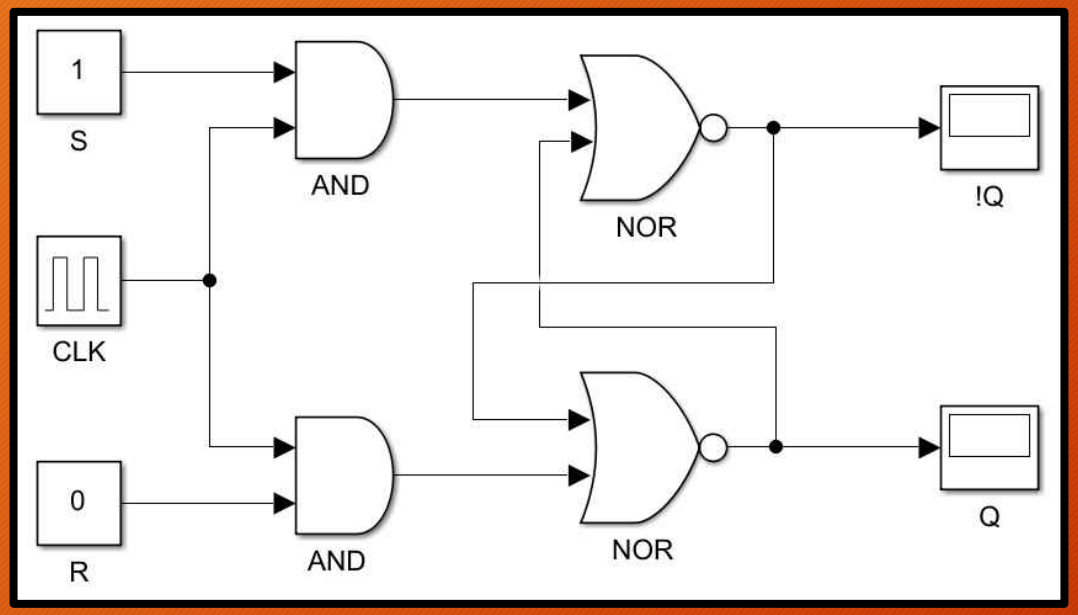
Flip-Flops

- Flip flops are latches controlled by a clock signal
- The name flip-flop comes from the circuit's ability to "flip and flop" between two stable states
  - The two stable states can be used to store one bit of binary data (either 0 or 1)
  - Flip-flops are the fundamental building blocks of all memory devices
  - Flip-flops switch between the two stable states based on a triggering event (a clock signal)
  - Flip-flops do not change output state with a change in the input signal until a clock signal edge is detected
  - By latching (holding) a value and changing it when triggered by a clock signal, flip-flops can sample (read) and store data at a set rate
- Types of Flip-Flops
  - Set-Reset (S-R) flip-flop
  - J-K flip-flop
  - Data (D) flip-flop
  - Toggle (T) flip-flop

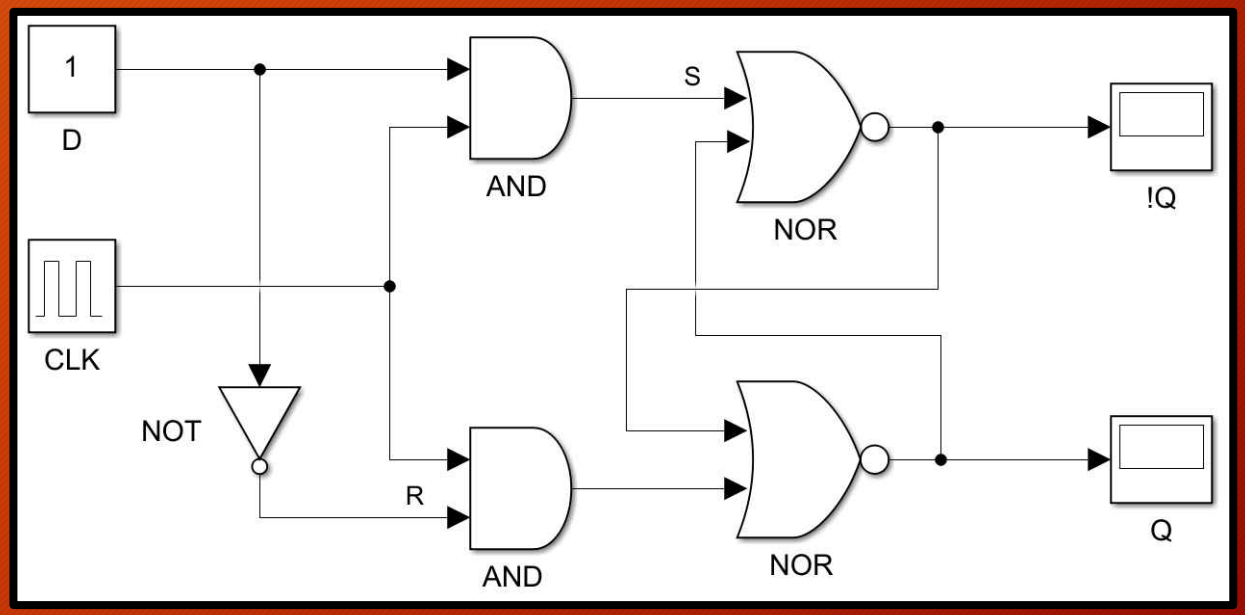
Schematics of Basic Flip Flop Types

- Basic Flip Flops are simply gated latches with a clock driving the gate

S-R Flip Flop



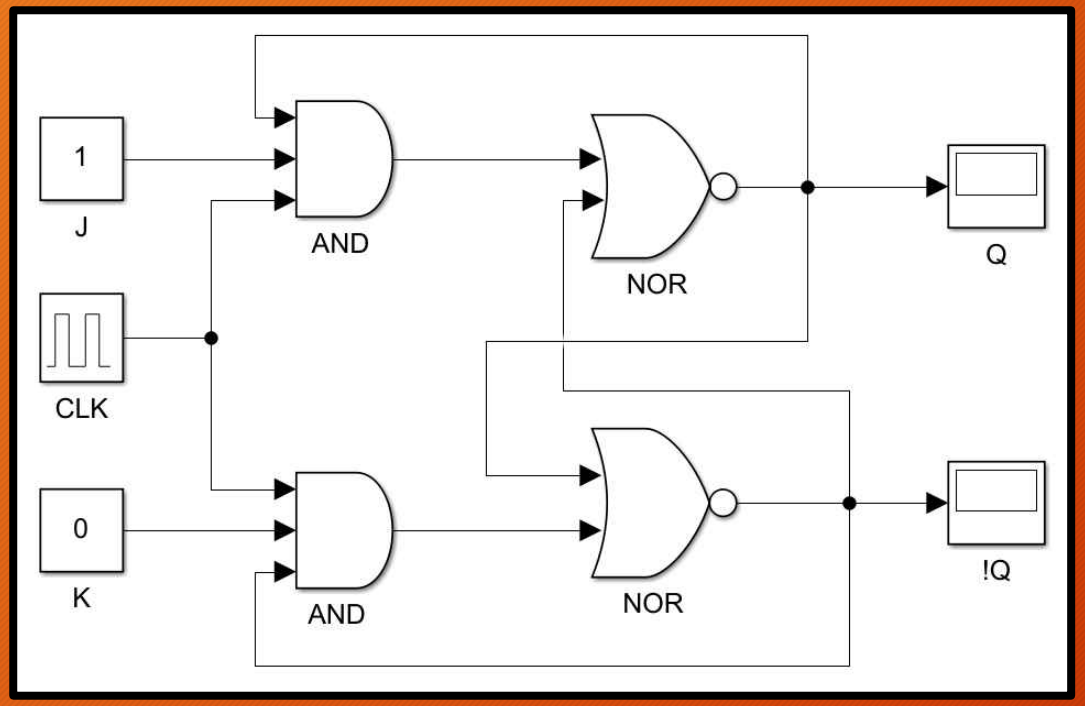
D Flip Flop



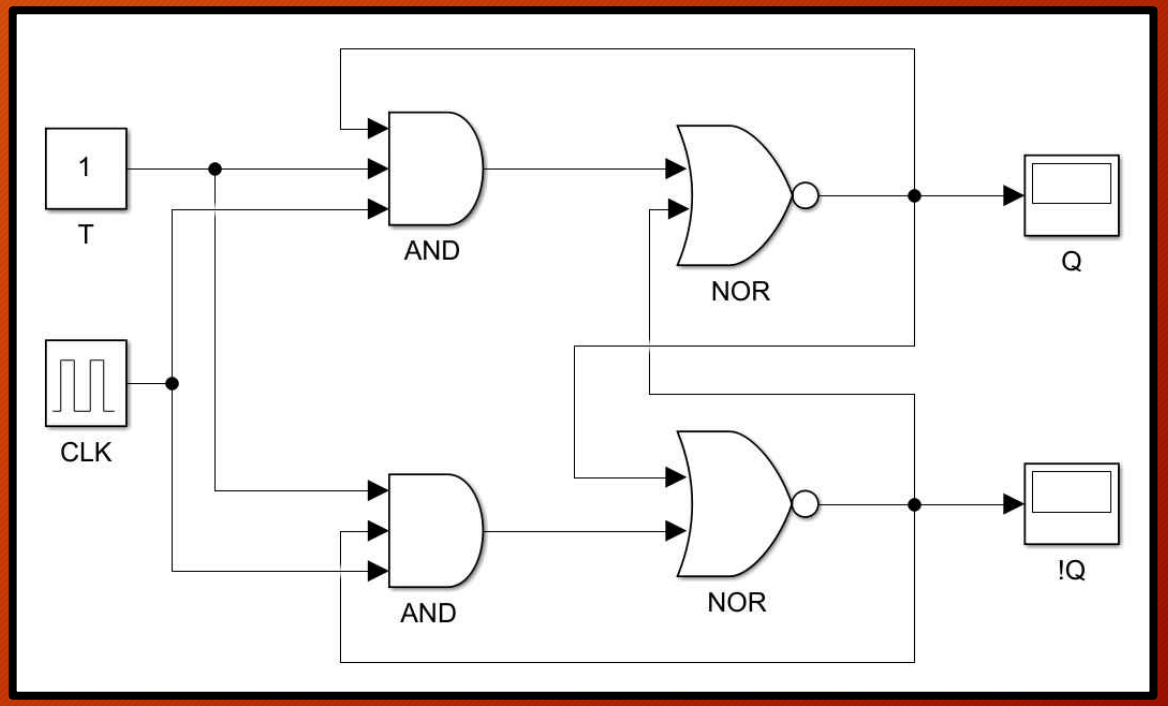
- These schematics are for illustrative purposes only
- They will not actually execute in Simulink

Schematics of Basic Flip Flop Types (continued)

J-K Flip Flop



T Flip Flop



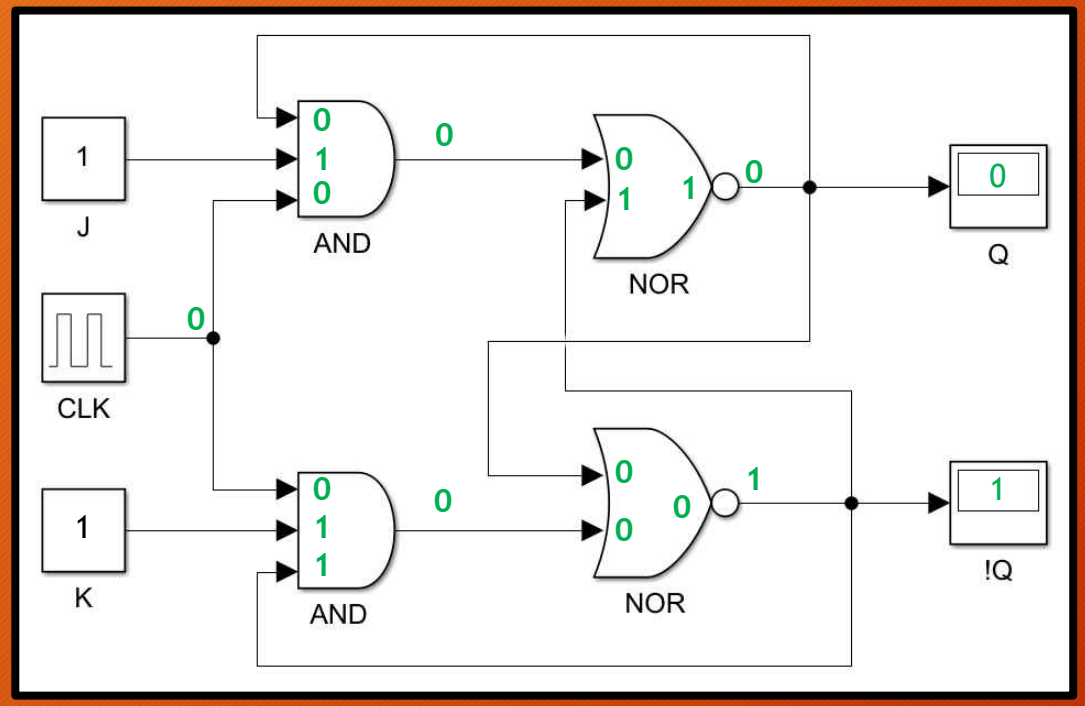
- These schematics are for illustrative purposes only
- They will not actually execute in Simulink

Under certain circumstances, both the J-K and the T Flip Flops can experience what is called a “race around condition”

J-K Level-Triggering Flip Flop Race Around Condition

Assumptions

- Flip Flop is Level Triggered
- J = K = 1
- Q is currently 0 (!Q is 1)
- Clock is currently low (0)
- Clock is about to transition to high (1)

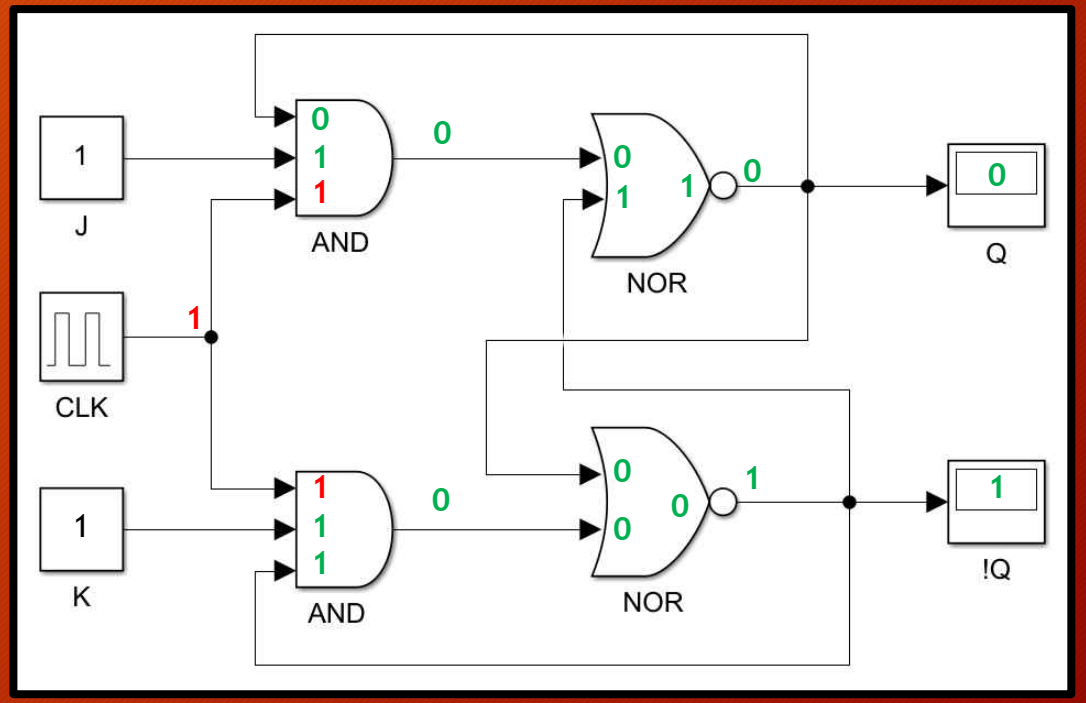


This circuit is stable

Assumptions

- J = K = 1
- Clock transitions to high (1)

A potential error condition exists when the clock pulse is long compared to the time it takes to process the clock signal at level high
If the feedback is provided while the clock is still high, the output will toggle continuously until the clock level goes low (not good!)

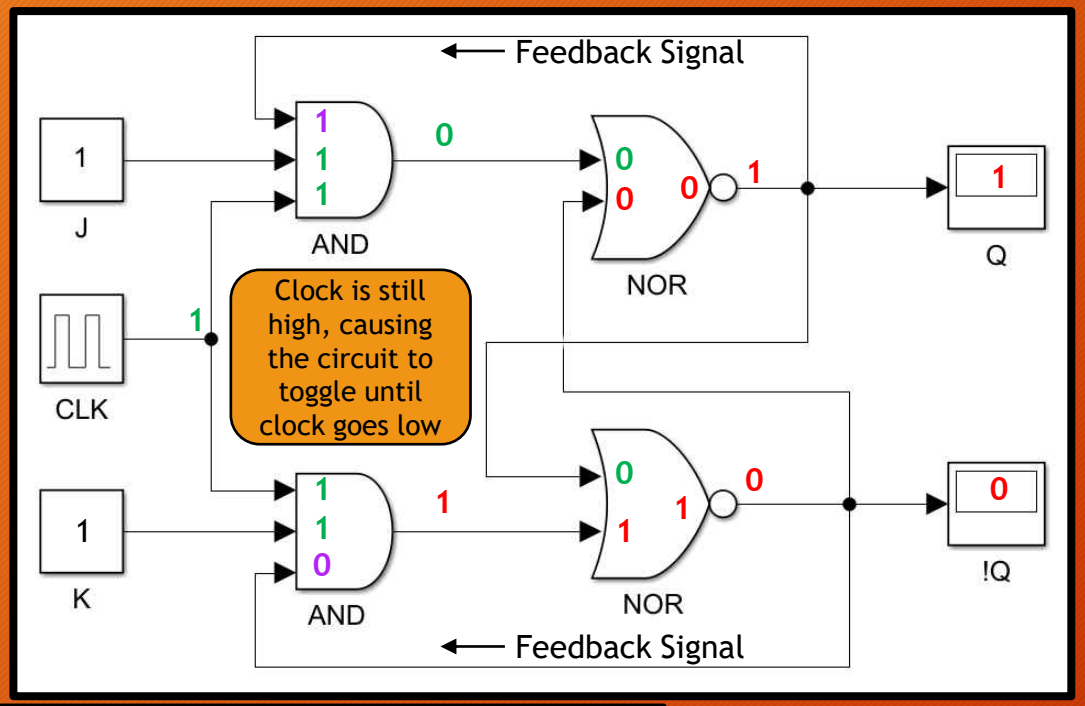


The race is on!

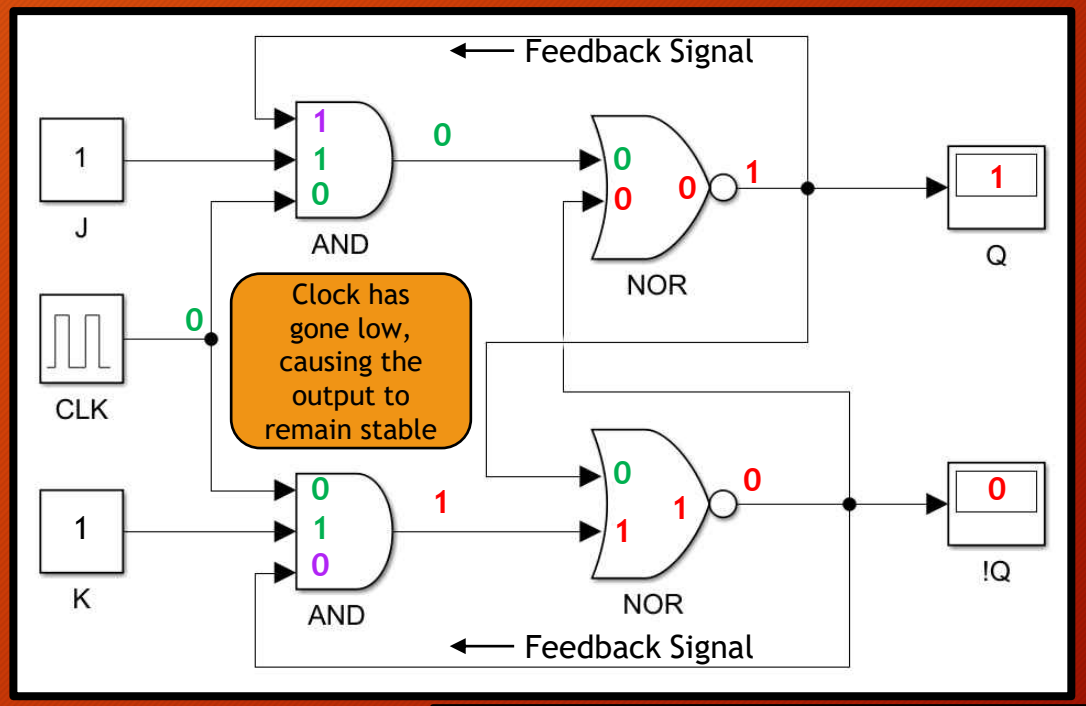
Red indicates propagation of processing of clock signal level high

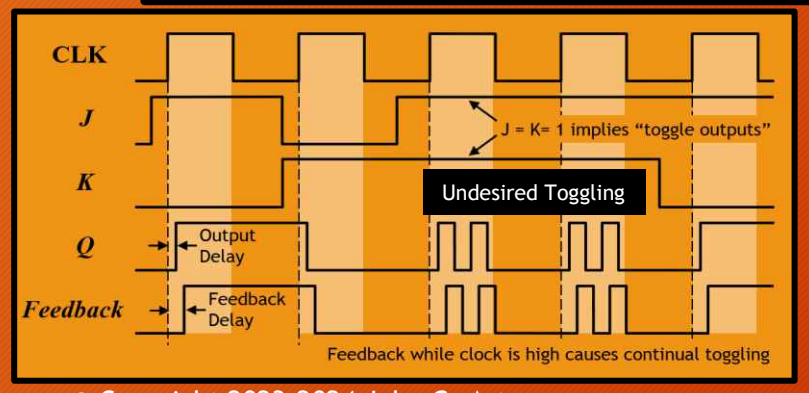
Examples of Race Around in J-K Flip Flop

Clock level high pulse duration is long compared to feedback processing time



Clock level high pulse duration is short compared to feedback processing time

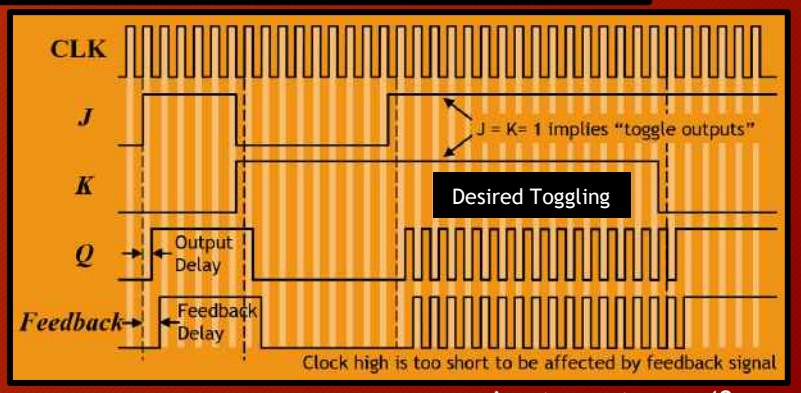




Red indicates propagation of processing of clock signal level high
Purple indicates arrival of feedback signal

Two methods for avoiding this condition:

- Use an edge-triggered J-K flip flop circuit instead of a level-triggered flip flop
- Use a pulse-triggered J-K Flip Flop (discussed further ahead) instead a level-triggered flip flop

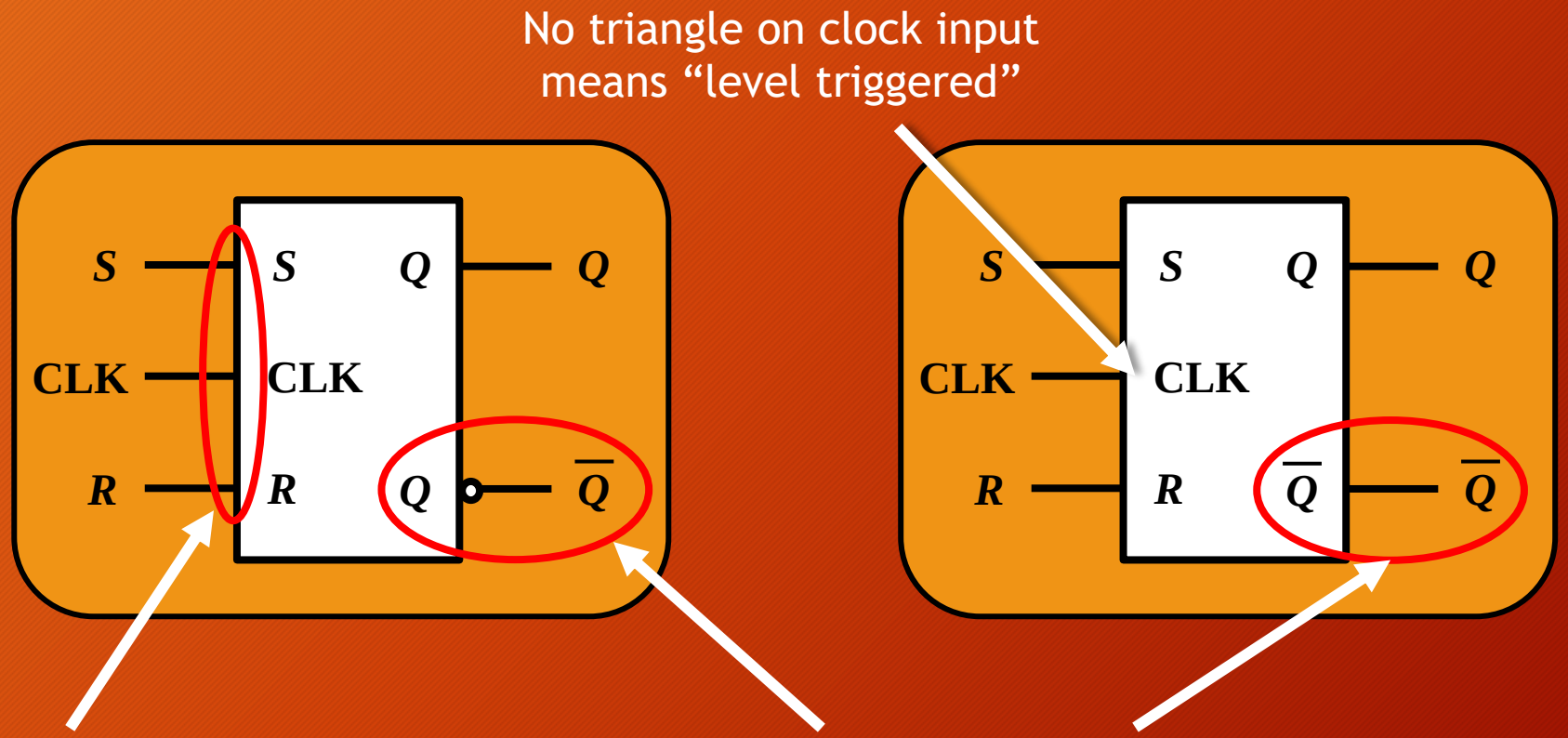


Symbolic Representations for Flip Flops

Return to Index

Symbolic Representations for Latches and Flip Flops

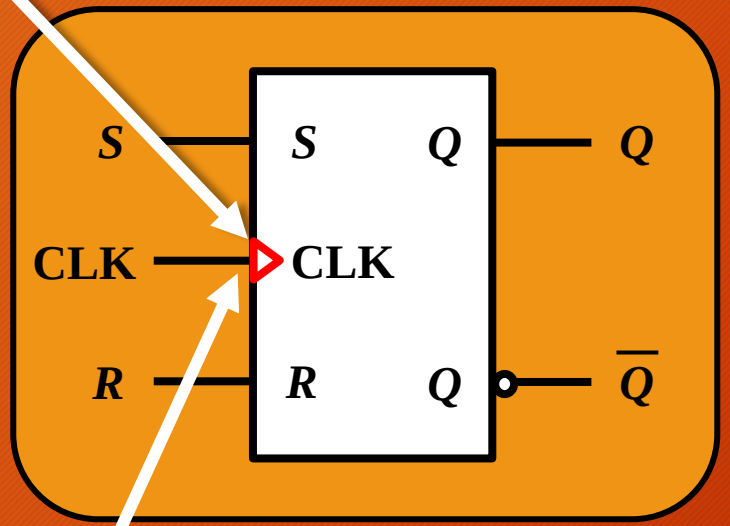
- Standard symbology is used to identify certain features of digital circuits
  - This example uses S-R Flip Flops to illustrate the symbology used



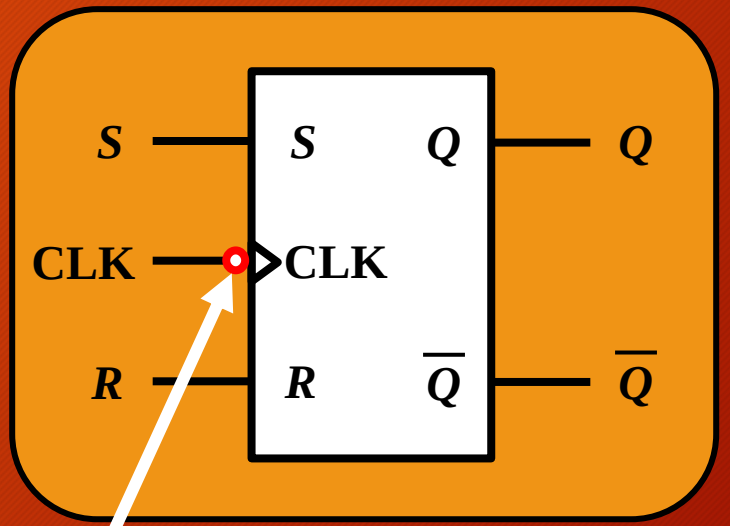
Symbolic Representations for Latches and Flip Flops (continued)

- Standard symbology is used to identify certain features of digital circuits
  - This example uses S-R Flip Flops to illustrate the symbology used

Triangle on clock input means "edge triggered"



No bubble on clock input means positive edge-triggered input



Bubble on clock input means negative edge-triggered input

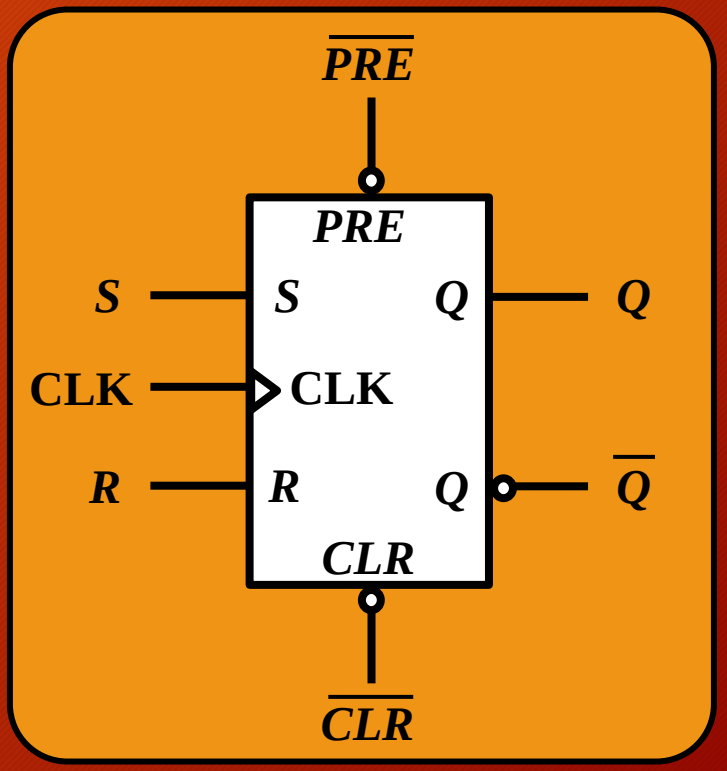
Digital Circuits with Additional Preset (PRE) and Clear (CLR) Inputs

Return to Index

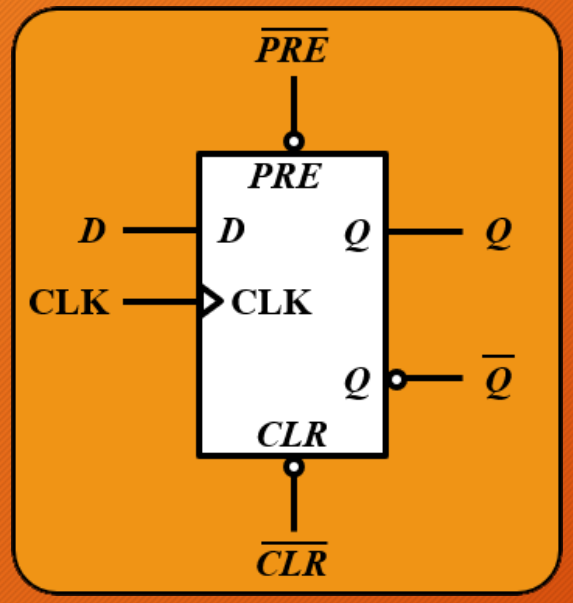
Asynchronous Inputs for Flip Flops

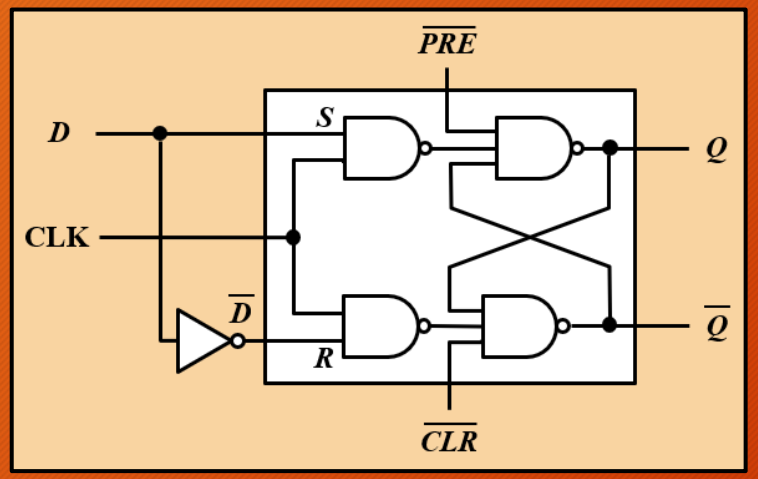
- The Preset (PRE) and Clear (CLR) inputs are asynchronous inputs
  - These asynchronous inputs disregard the clock
  - They Preset the flip-flop (also called asynchronously set)
  - Or Clear the flip-flop (also called asynchronously reset)
- When CLR is high and PRE is low, the flip flop is asynchronously reset (Q=1, Q=0), regardless of the CLK, S,and R inputs
- These active-low inputs are therefore normally pulled high to make them inactive
- The ability to apply asynchronous sets and resets is often used to clear entire registers that consist of an array of flip flops

S-R Flip Flop
With PRE and CLR
Asynchronous Inputs

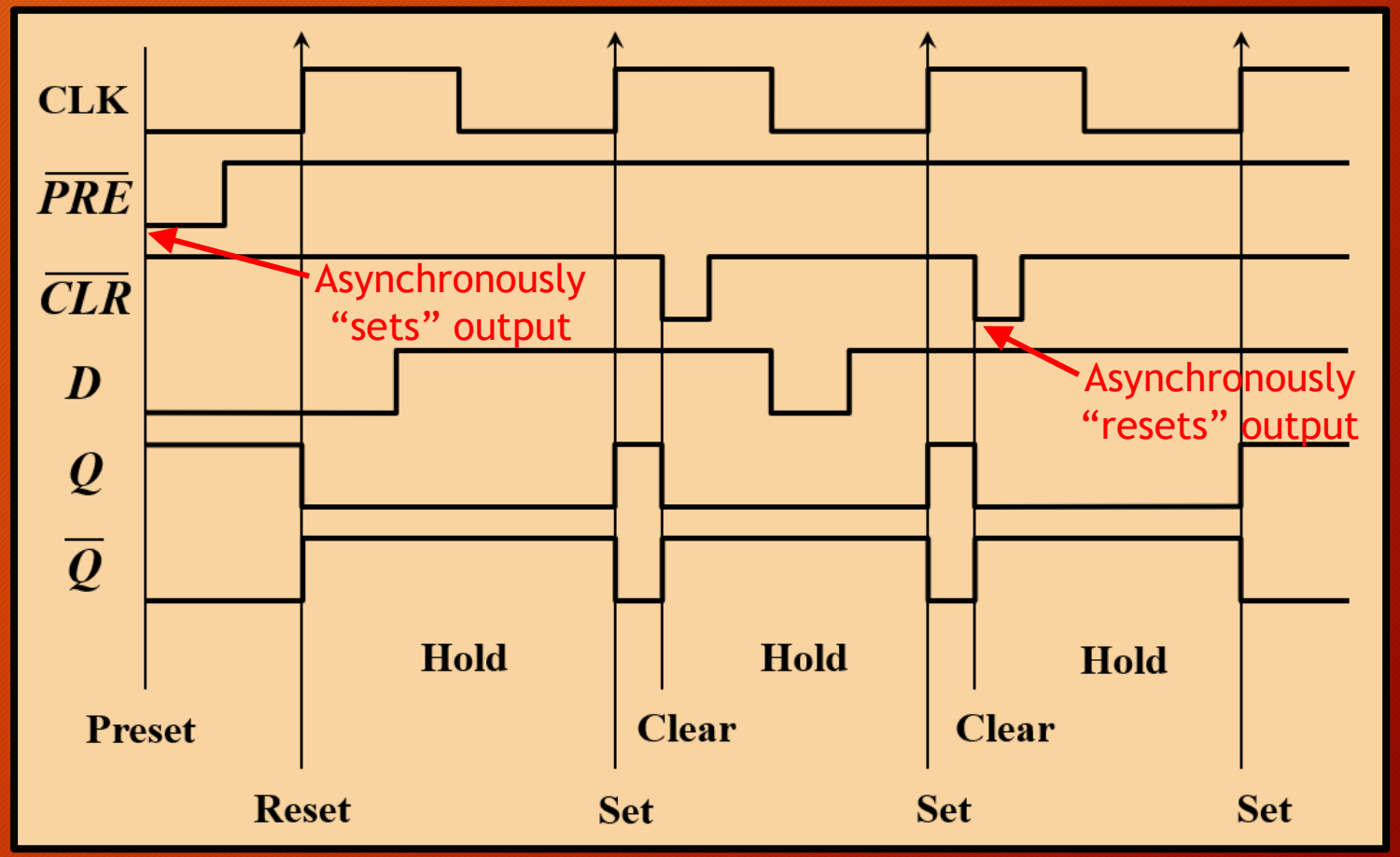


Example Usage of PRE and CLR



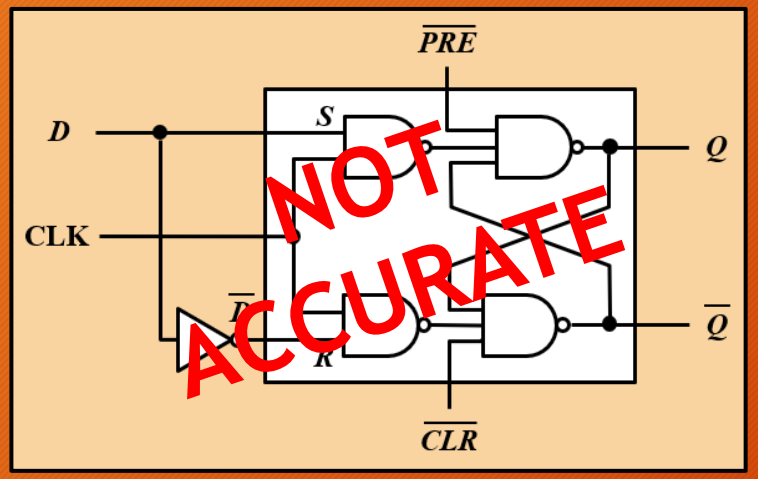


Signal Timeline for D Flip Flop with PRE and CLR Asynchronous Inputs



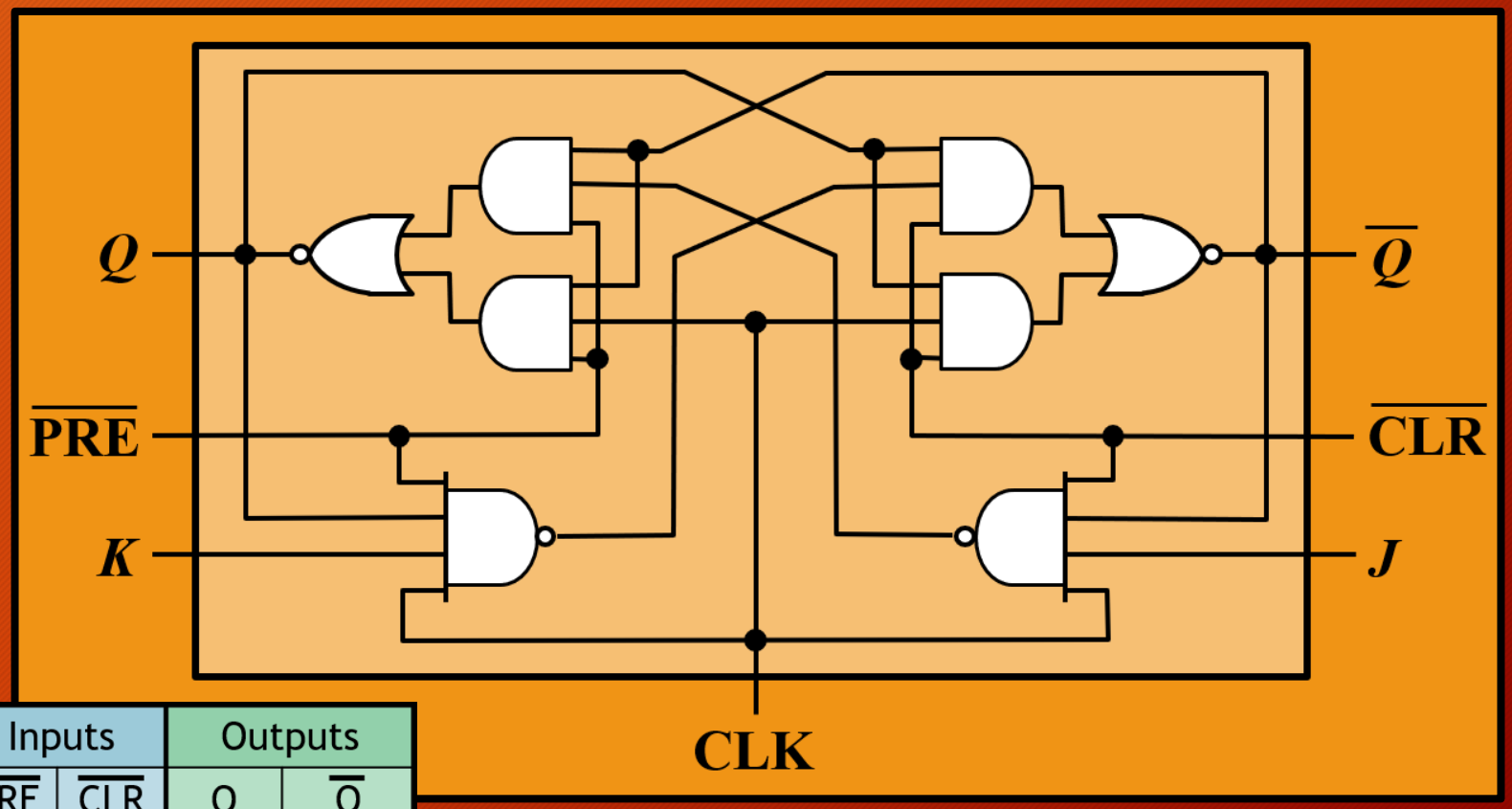
A More Accurate Representation of PRE and CLR

- The previous depiction of the logic circuit for PRE and CLR is not actually correct
- In order to have overall control of the circuit to set and reset the output irrespective of the inputs and clock, a more intricate logical arrangement is required



The circuit to the right is a more accurate representation of a J-K Flip Flop with PRE and CLR controls

It is left as an exercise to verify the following truth table that illustrates the desired level of control for PRE and CLR



Inputs		Outputs	
$\overline{PRE}$	$\overline{CLR}$	Q	$\overline{Q}$
0	1	1	0
1	0	0	1

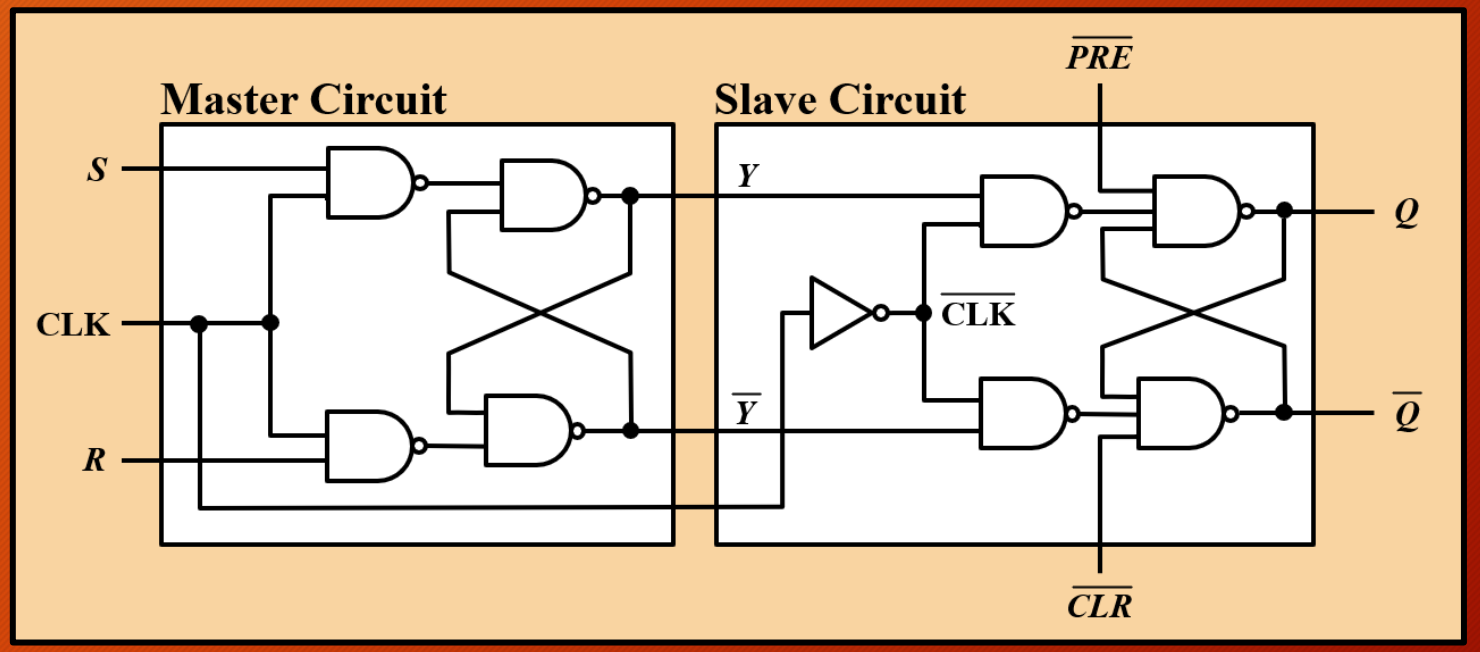
Pulse-Triggered Circuits

Return to Index

Pulse-Triggered SR Flip Flop

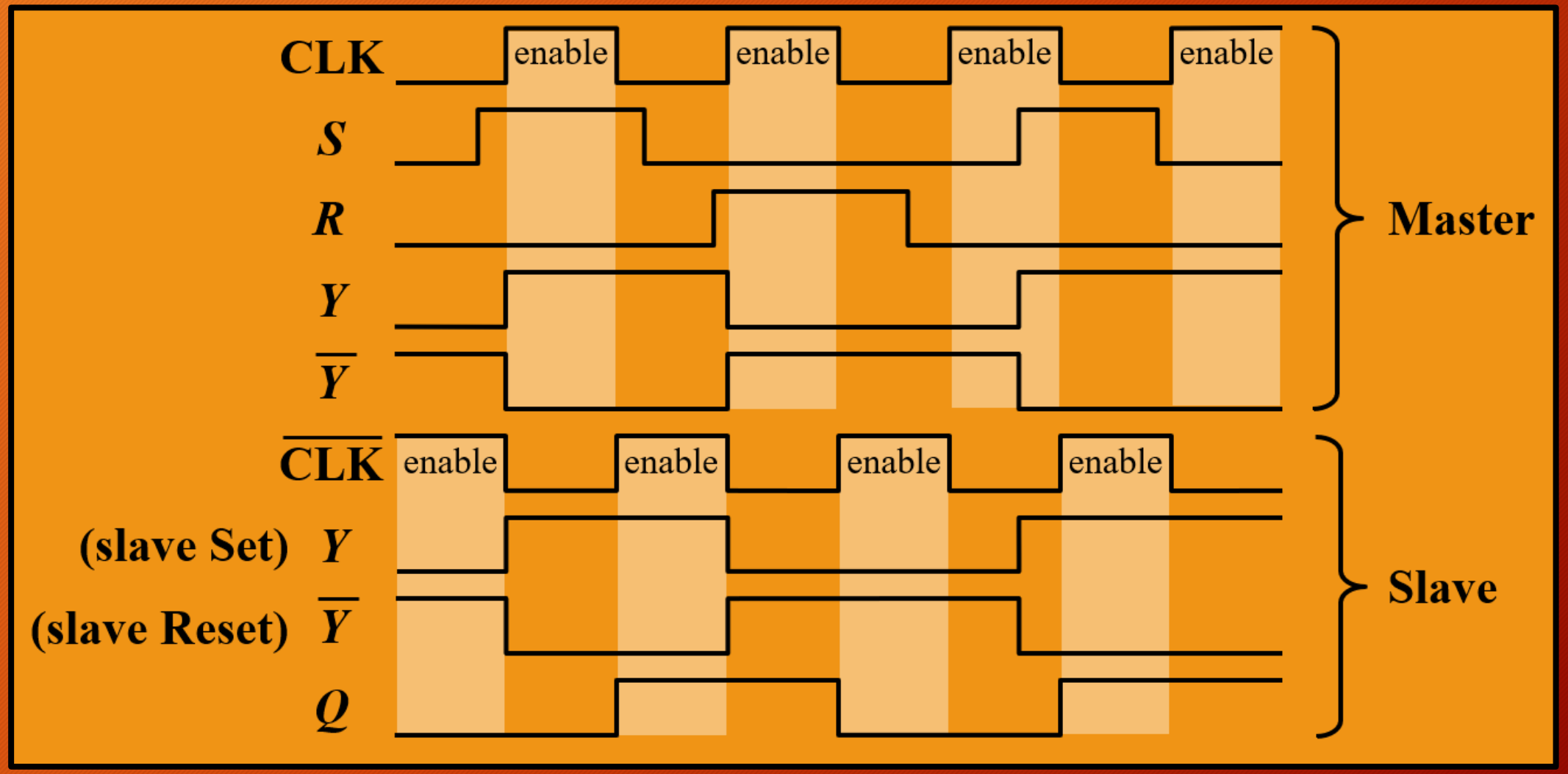
- A pulse-triggered SR flip flop is a level-clocked flip flop
  - However, for any change in output to occur, both the high and low levels of the clock must rise and fall
- Pulse-triggered flip flops are also called master-slave flip flop
  - The master circuit accepts the initial inputs when the positive clock edge arrives
  - Then triggers the slave circuit with its output when the negative clock edge arrives

Simplified Pulse-Triggered Cross-NAND SR Flip Flop



Example Usage of Pulse-Triggered SR Flip Flop

Signal Timeline for Pulse-Triggered SR Flip Flop



Designing Circuits Based on a Truth Table Specification using Karnough Mapping

Return to Index

Building a Logic Circuit Using Karnough Mapping

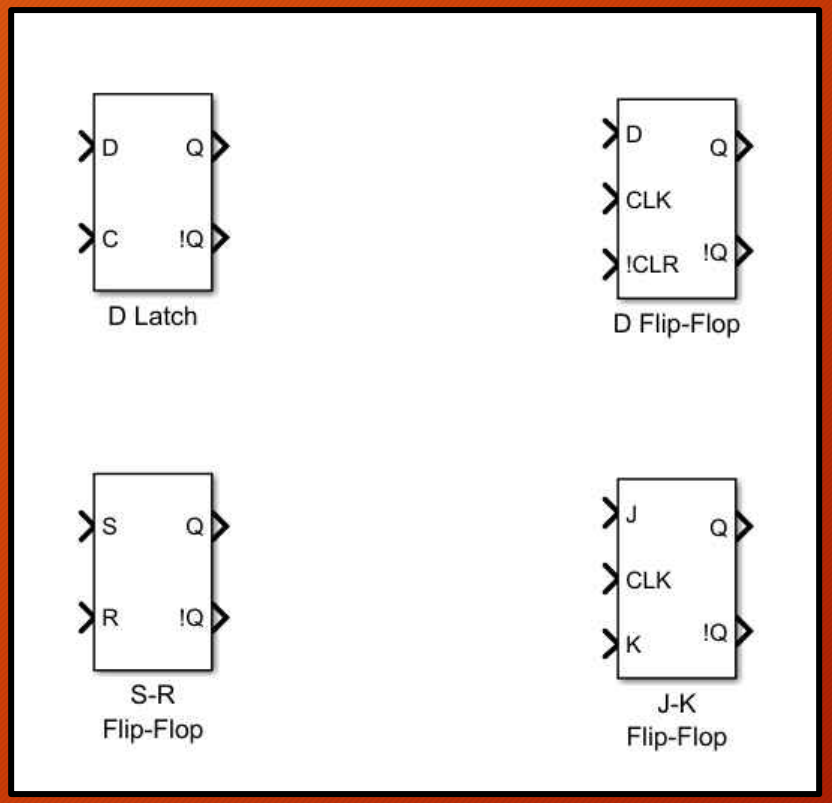
- Karnaugh Mapping is a technique used to redesign logic circuits designs to be simpler, and more cost effective
  - Since the mapping can result in fewer components than originally designed
- For this course, desiging a circuit in Simulink using logic components can be assisted by using Karnaugh mapping
  - If you can define a truth table that describes the available inputs and the desired outputs, then Karnaugh Mappong can identify a suitable logic circuit
  - The solution will produce the output described in the truth table for the given inputs
- The subject is beyond the purpose of this lecture material
  - But there are many good references that describe the technique
  - A few are identified here to get started:
    - https://www.allaboutcircuits.com/technical-articles/karnaugh-map-boolean-algebraic-simplification-technique/
    - https://www.allaboutcircuits.com/textbook/digital/chpt-8/logic-simplification-karnaugh-maps/
    - https://electricalacademia.com/digital-circuits/karnaugh-map-tutorial/
    - https://www.youtube.com/watch?v=RO5alU6PpSU
      - Final example in this video has an error identified in the video discussion

Summary of Latches and Flip Flops Provided in Simulink

Return to Index

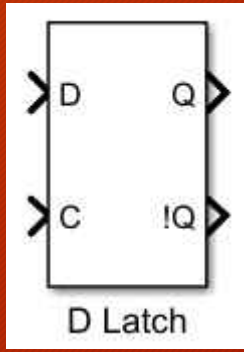
Sequential Circuits in Simulink

- Simulink offers only four representations of sequential circuits:
  - D Latch
  - S-R Latch
  - D Flip Flop
  - J-K Flip Flop



D Latch in Simulink

- The D Latch block models an enabled D Latch flip-flop
- Input
  - D — Data input signal
  - C — Chip enable input signal
    - The chip enable input signal (C) controls when the block executes
    - When C is greater than zero, the output Q is the same as the input D
- Output
  - Q — Output signal
  - !Q — Output signal
- The D Latch block treats a nonzero input as true (1)
- When the D Latch block is not enabled (C=0), the block remains in the previous state
- All inputs and outputs are required to have the same data type
  - Boolean data type recommended



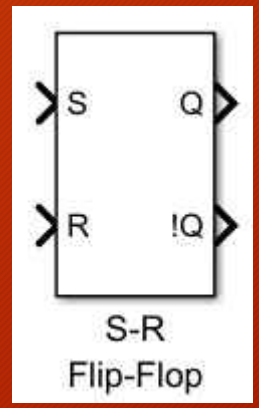
Truth Table

Inputs		Outputs	
C	D	Q_n	$!Q_n$
0	Any	Q_{n-1}	$!Q_{n-1}$
1	0	0	1
1	1	1	0

In this truth table, Qn-1 is the output at the previous time step

S-R Flip Flop

- The S-R Flip-Flop block models a simple Set-Reset latch constructed using NOR gates
- Input
  - S — Set input
  - R — Reset input
- Output
  - Q — Output signal Q
  - !Q — Output signal !Q
- Parameters
  - Initial condition (state of Q) is the initial value of output Q
- The S-R Flip-Flop block treats a nonzero input as true (1)
- Avoid the state where R and S are both 1
  - In this state, both Q and !Q are 0
  - This state is undefined because !Q is not the complement of Q
  - To handle this state, instead consider using the J-K Flip-Flop block
- All inputs and outputs are required to have the same data type
  - Boolean data type recommended



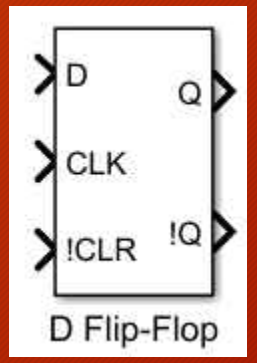
Truth Table

Inputs		Outputs	
S	R	Q_n	$!Q_n$
0	0	Q_{n-1}	$!Q_{n-1}$
0	1	0	1
1	0	1	0
1	1	0	0

In this truth table, Qn-1 is the output at the previous time step

D Flip Flop

- The D Flip-Flop block models a positive-edge-triggered enabled D flip-flop
- Input
  - D — Data input signal
  - CLK — Clock signal
  - !CLR — Enable input signal
- Output
  - Q — Output signal Q
  - !Q — Output signal !Q
- On the positive (rising) edge of the clock signal, if the block is enabled (!CLR is greater than zero), the output Q is the same as the input D
- The D Flip-Flop block treats a nonzero input as true (1)
- If the block is not enabled on the rising edge of the clock signal, Q is reset to zero
- When the clock signal is not rising, the block remains in the previous state
- All inputs and outputs are required to have the same data type
  - Boolean data type recommended

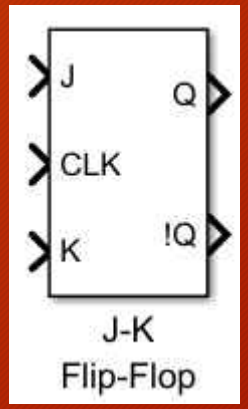


Truth Table

Inputs	Outputs	
D	Q	!Q
1	1	0
0	0	1

J-K Flip Flop

- The J-K Flip-Flop block models a negative-edge-triggered J-K flip-flop
- Inputs
  - J — Input signal J
  - CLK — Clock signal
  - K — Input signal K
- Outputs
  - Q — Output signal Q
  - !Q — Output signal !Q
- Parameters
  - Initial condition (state of Q) is the initial value of output Q
- When J is 1 and K is 0, the flip-flop goes to the set state (Qn is 1)
- When J is 0 and K is 1, the flip-flop goes to the reset state (Qn is 0)
- When both J and K are 0, the flip-flop stays in the previous state (Qn is Qn-1)
- When both J and K are 1, the flip-flop toggles (Qn is the complement of Qn-1)
- On the negative (falling) edge of the clock signal (CLK), the J-K Flip-Flop block outputs Q and its complement, !Q, according to the truth table
- The J-K Flip-Flop block treats a nonzero input as true (1)



Truth Table

Inputs		Outputs	
J	K	Q_n	$!Q_n$
0	0	Q_{n-1}	$!Q_{n-1}$
0	1	0	1
1	0	1	0
1	1	$!Q_{n-1}$	Q_{n-1}

In this truth table, Qn-1 is the output at the previous time step

- Digital Electronics Flip-flops and their Types
  - https://www.tutorialspoint.com/digital-electronics-flip-flops-and-their-types
- Flip-flop types, their Conversion and Applications
  - https://www.geeksforgeeks.org/flip-flop-types-their-conversion-and-applications/
- Latches in Digital Logic
  - https://www.geeksforgeeks.org/latches-in-digital-logic/?ref=lbp
- Truth Tables- Conjunction (and), Disjunction (or), Negation (not)
  - https://math.libretexts.org/Courses/Fullerton_College/Math_100%3A_Liberal_Arts_Math_(Claassen_and_Ikeda)/05%3A_Logic/5.02%3A_Truth_Tables-_Conjunction_(and)_Disjunction_(or)_Negation_(not)
- Understanding Digital Buffer, Gate, and Logic IC Circuits - Part 1
  - https://www.nutsvolts.com/magazine/article/understanding_digital_buffer_gate_and_ic_circuits_part_1
- Combinational Circuit
  - https://www.slideshare.net/satyaJoshi1/combinational-circuit
- JK Flip Flop: What is it?
  - https://www.electrical4u.com/jk-flip-flop/
- What is JK Flip-Flop ?
  - https://www.geeksforgeeks.org/what-is-jk-flip-flop/

- What is JK Flip-Flop ?
  - https://www.geeksforgeeks.org/what-is-jk-flip-flop/
- Logic gate diagram for JK latch?
  - https://electronics.stackexchange.com/questions/399286/logic-gate-diagram-for-jk-latch-not-flip-flop
- Latches and Flip-Flops
  - https://www.cs.ucr.edu/~ehwang/courses/cs120b/flipflops.pdf
- Latches In Digital Electronics
  - https://automationforum.co/latches-in-digital-electronics/
- Edge Triggering and Level Triggering
  - https://www.geeksforgeeks.org/edge-triggering-and-level-triggering/