System Timing Issues

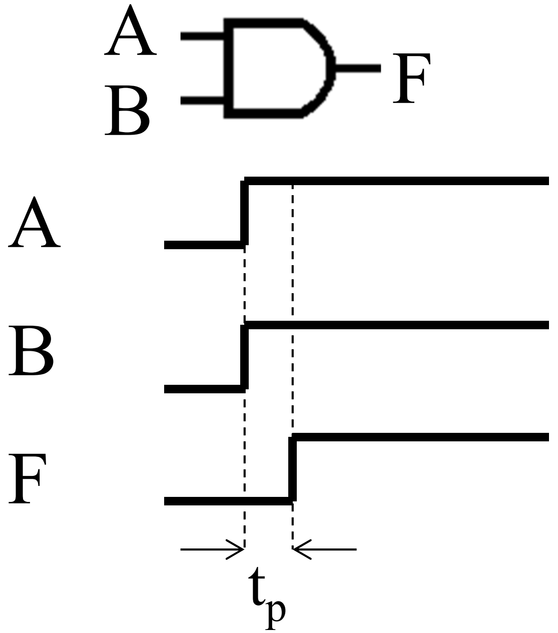
- Propagation Delay
- Hazards
- Synchronous Design
- System Timing

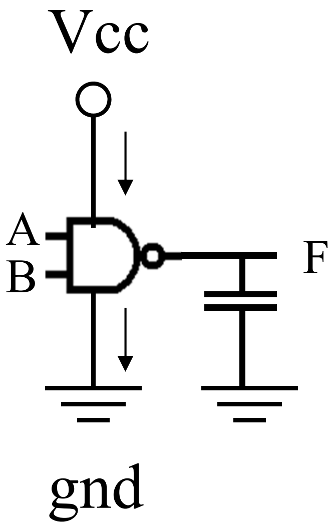
Gate Delay

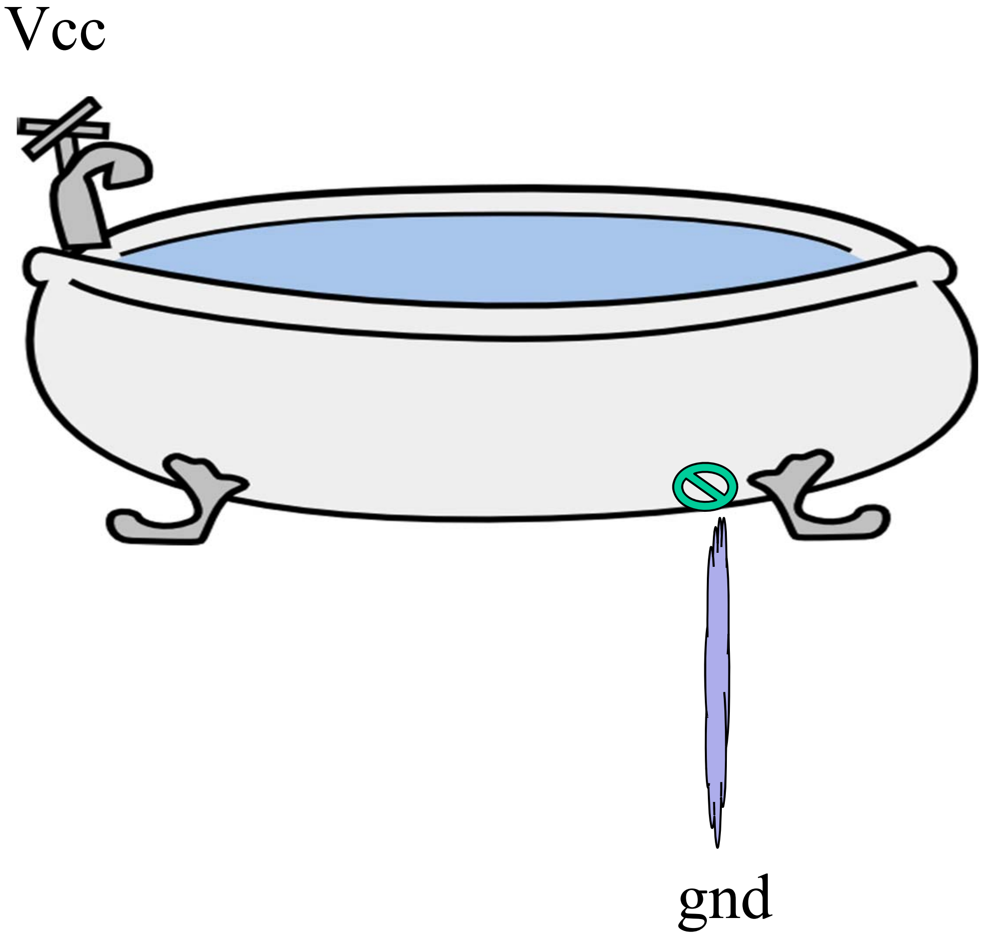
In the real world, gates have physical limitations dependant on the technology used to implement them.

When the input to a gate changes, there is a time delay before the output responds to the input change. This is known as the **propagation delay** and is a result of the finite time that it takes to charge and discharge the nodes in the transistor circuit that forms the gate.

The propagation delay time (t_p) can be obtained from the input and output waveforms and is usually measured in nanoseconds or picoseconds.

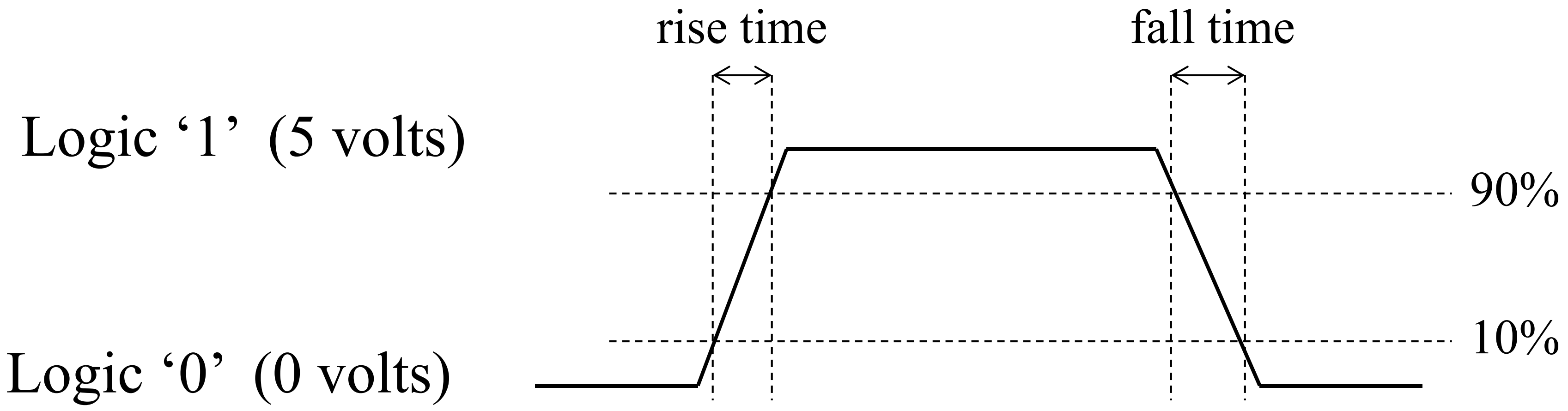



Vcc
A
B
F
gnd


Vcc
gnd

Rise Time and Fall Time

Signal values do not change instantly. The time taken for a signal to switch from 10% to 90% of its nominal value defines its **rise time**. Similarly, the time taken to switch from 90% to 10% of its nominal value defines its **fall time**.

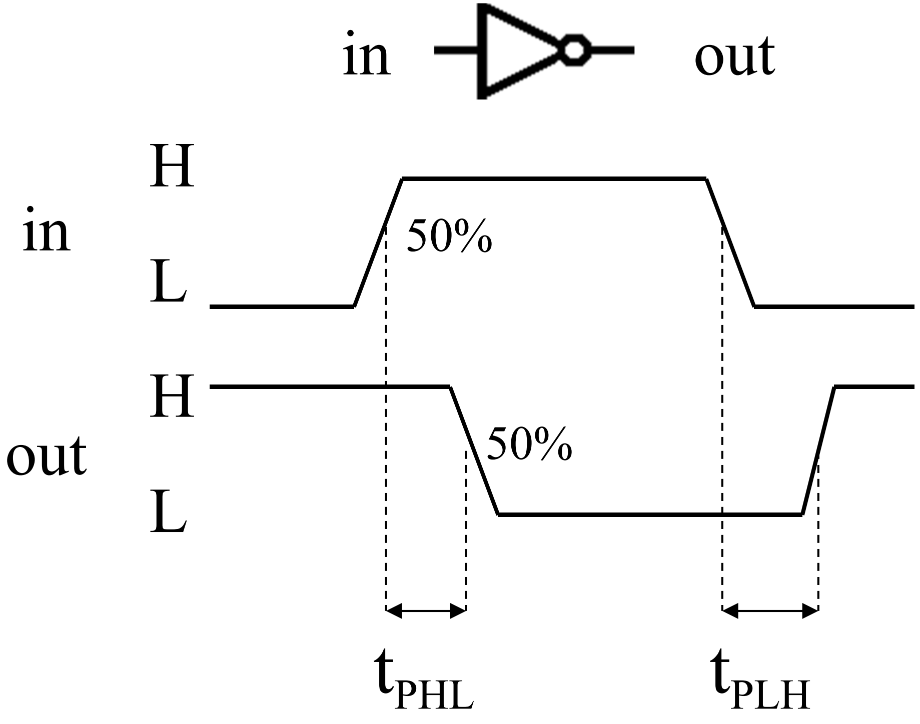


Low-to-High and High-to-Low transitions are not delayed equally.

Propagation Delay

t_{PHL} is the time for the output to reach 50% of its nominal value on the H-to-L transition after the input signal reached 50% of its nominal value.

t_{PLH} is the time for the output to reach 50% of its nominal value on the L-to-H transition after the input signal reached 50% of its nominal value.



$$\boxed{t_p = (t_{PHL} + t_{PLH})/2}$$

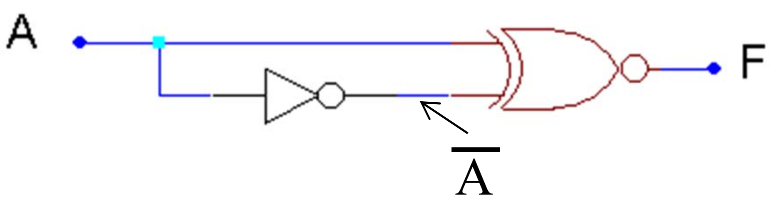
The propagation delay limits the switching speed or frequency at which a logic circuit can operate.

X	Y	F
0	0	1
0	1	0
1	0	0
1	1	1

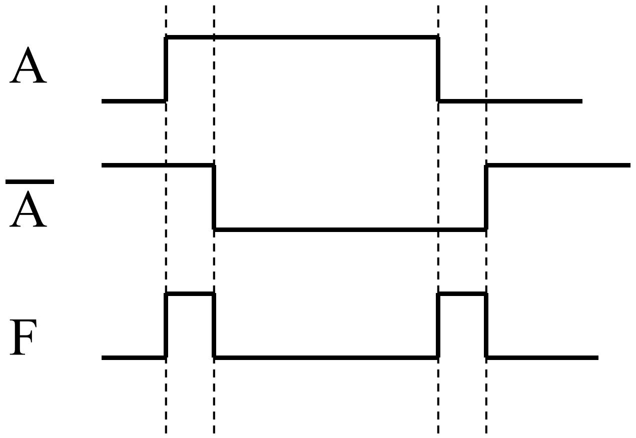
Hazards

The propagation delay in real logic circuits can lead to unwanted switching transients called hazards.

Two signals originating from the same source, may end up taking different paths through the circuit and end up racing each other. This **race condition** can be seen in the XNOR gate below.



The inverter will delay the signal $\overline{A}$ into the XNOR gate. This leads to the small pulses shown on the timing diagram.

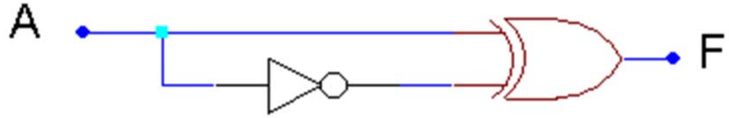


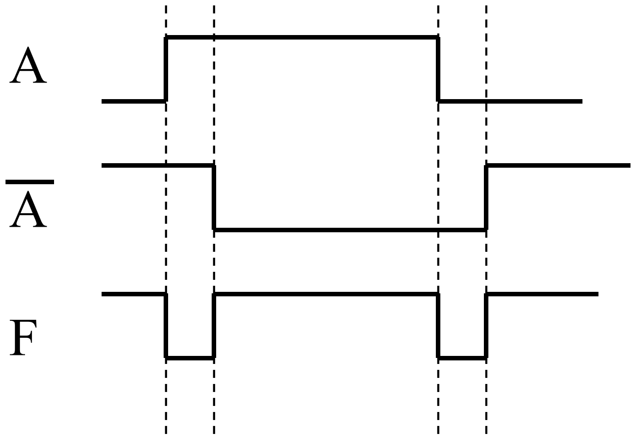
These short pulses are known as **spikes** or **glitches**.

Static Hazards

A static hazard is one where there are two changes in the logic level of the signal when none is expected. The XNOR case showed a static-0 hazard because the signal was expected to stay at zero.

Replacing the XNOR with an XOR demonstrates a static-1 hazard.



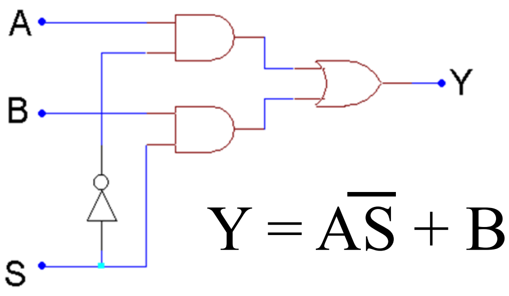


The logic level of the output changes twice when it should remain at 1.

The hazard occurs because the different paths taken by A lead to a race condition and attain their final values at different times.

Multiplexer Hazard

Consider the circuit for a 2-to-1 multiplexer.

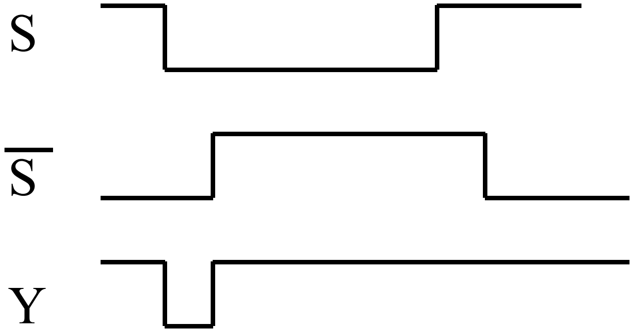


$Y = A\overline{S} + BS$

If a hazard occurs, it will be because of the race condition on the select line S. This will only lead to an error on the output for the case when A = B = 1.

All other cases for A and B will disable the race paths and errors will not propagate further.

Consider the case where A = B = 1. In this case $Y = \overline{S} + S$.

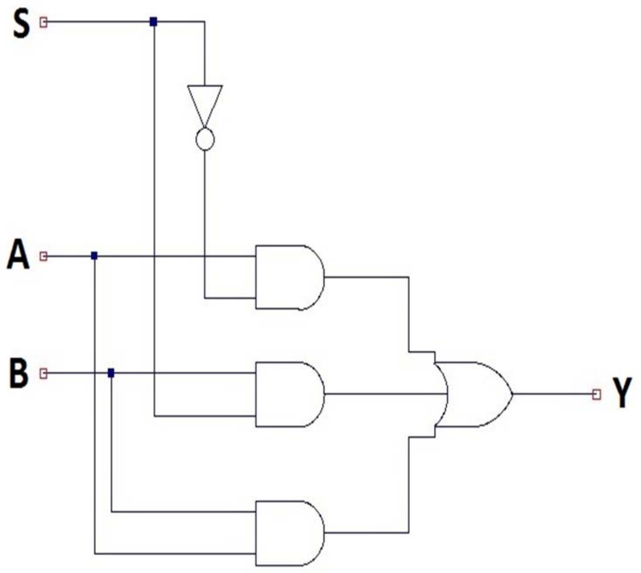


The error occurs because $\overline{S}$ is delayed.

There is a static-1 hazard on the output when A = B = 1 and S changes.

Hazard Removal

The static-1 hazard can be removed by adding an extra gate which will ‘hold’ the output at 1 for the case when A = B =1 and Y = $\overline{S}$ + S.



$$Y = A.B + A.\overline{S} + B.S = A.\overline{S} + B.S$$

For the condition A = B = 1, the expression for Y is now:

$$Y = 1 + \overline{S} + S = 1$$

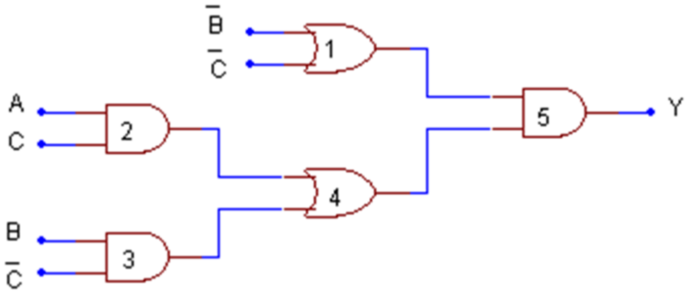
The extra AND gate which is used to form A.B is sometimes called a blanking or holding gate. It eliminates the hazard by holding the output at 1 for the case when Y = $\overline{S}$ + S.

Dynamic Hazards

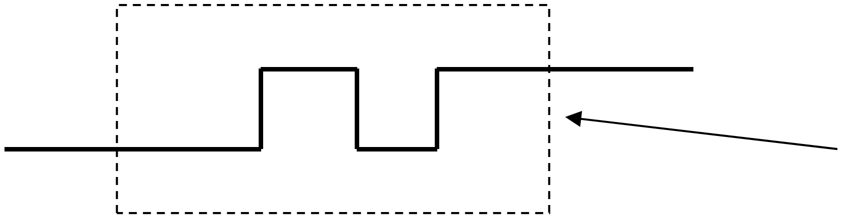
When a combinational logic circuit switches between levels, it is possible for a signal to change state and then transiently go back to its original state before making the final transition to the expected state. This is known as a dynamic hazard.

Consider the function

$$Y = (\overline{B} + \overline{C}).(A.C + B.\overline{C})$$



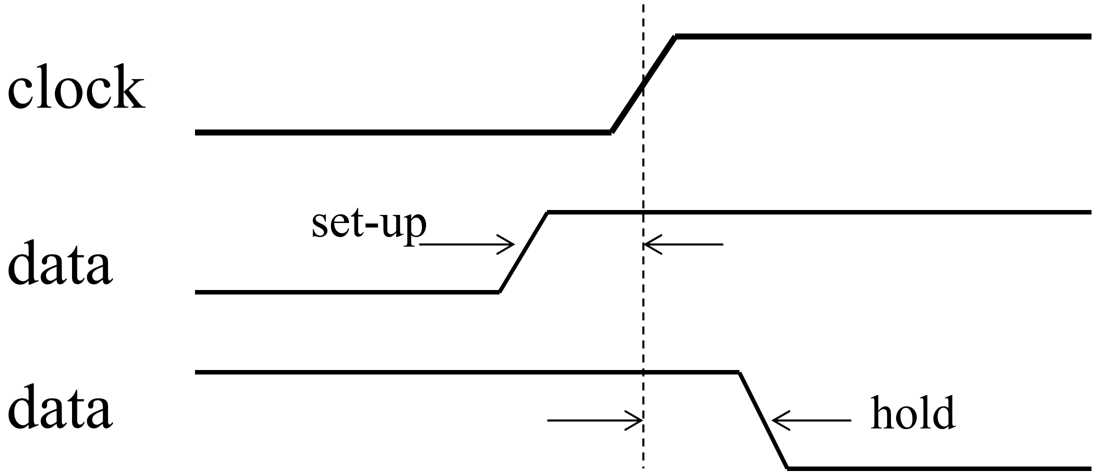
C has three paths through the logic circuit. There is a possibility of race conditions in three paths which may lead to a dynamic hazard.



dynamic hazard

Set-up and Hold time

The synchronous inputs to a flip-flop (e.g. D) must meet certain requirements in order for the levels to be reliably clocked in.



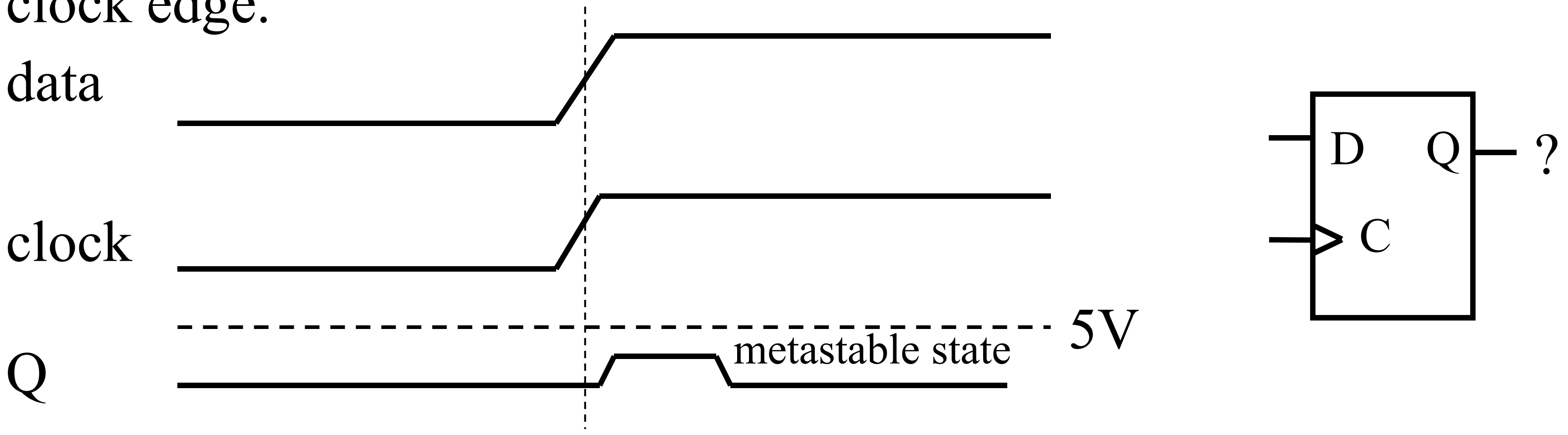
If the data changes during the set-up or hold time, correct operation cannot be guaranteed.

The duration of time data (D) needs to be present before the clock edge arrives is called the set-up time (t_s).

The duration of time data (D) must remain stable after the clock edge is called the hold time (t_h).

Metastability

What happens if the set-up time for a flip-flop is violated ? Consider the case below where data is changing at the same time as the active clock edge.



It is possible for the output to go to a level between the valid low and high levels, that is to a metastable state. Eventually it will snap to either high or low, but errors may have occurred in this time.

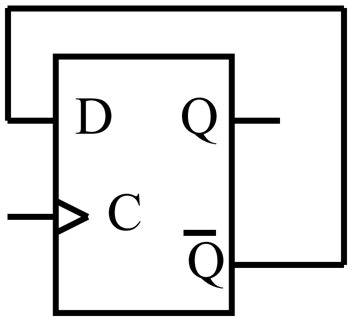
Metastability must be considered carefully when designing asynchronous interfaces.

Maximum Clock Frequency

The maximum clock frequency (f_{max}) is the highest rate at which the flip-flop can be operated reliably. Above this frequency it is unable to respond quickly enough.

The propagation delay of a flip-flop (t_p) is the delay from the active clock edge to data appearing at the output Q.

What is the toggle rate of the flip-flop shown.



t_s = 2ns
t_h = 0ns
t_p = 3ns

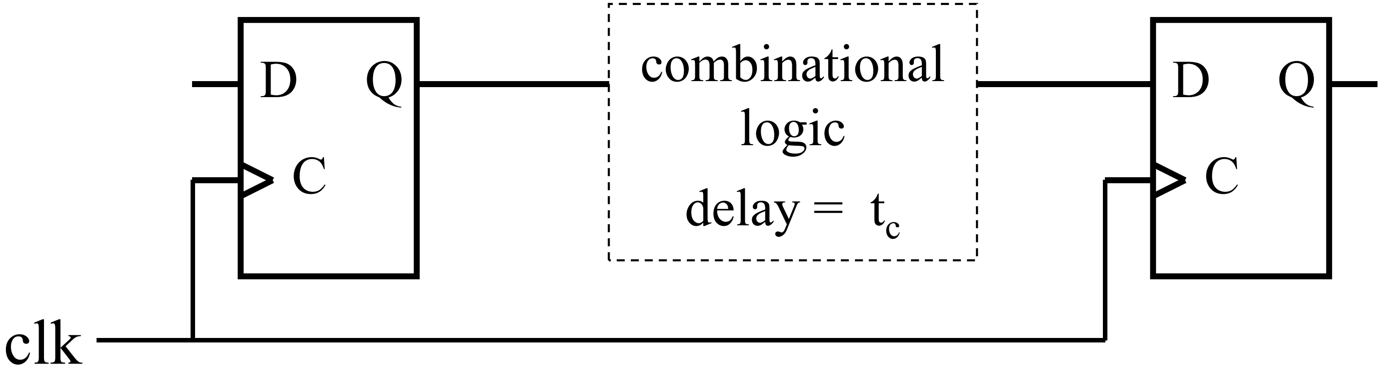
$$f_{max} = 1 / (t_s + t_p)$$

$$= 1 / (2\text{ns} + 3\text{ns})$$

$$= 200 \text{ MHz}$$

Synchronous Design

For a circuit which includes combinational logic also, the propagation delays must be added to find the maximum speed of operation.



t_s = 2ns t_c = 15ns

t_h = 0ns

t_p = 3ns

$$f_{max} = 1 / (t_s + t_p + t_c)$$
$$= 1 / (2\text{ns} + 3\text{ns} + 15\text{ns})$$
$$= 50 \text{ MHz}$$

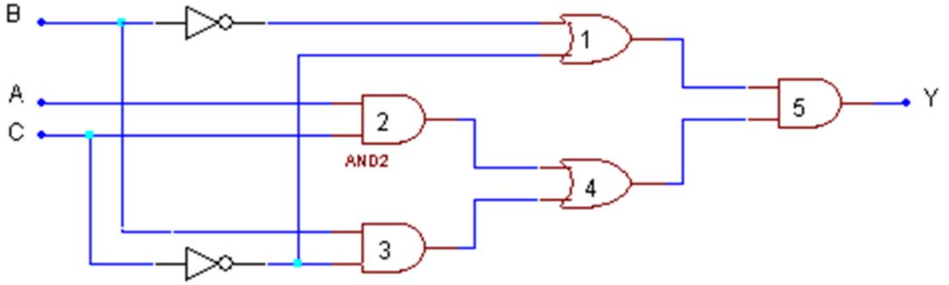
By leaving sufficient time between active clock edges for all transient activity to settle, problems with glitches can be avoided.

Critical Path

The maximum time required for the transient activity to settle will depend on the slowest path through the combinational circuitry.

This path is known as the **critical path** and it defines the maximum frequency at which the circuit can operate.

What is the maximum frequency of operation for the circuit shown?

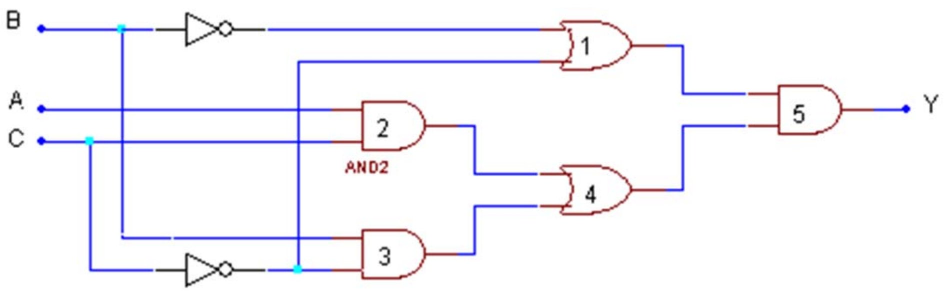


inverter delay = 2ns

and delay = 2.5ns

or delay = 3ns

Critical Path



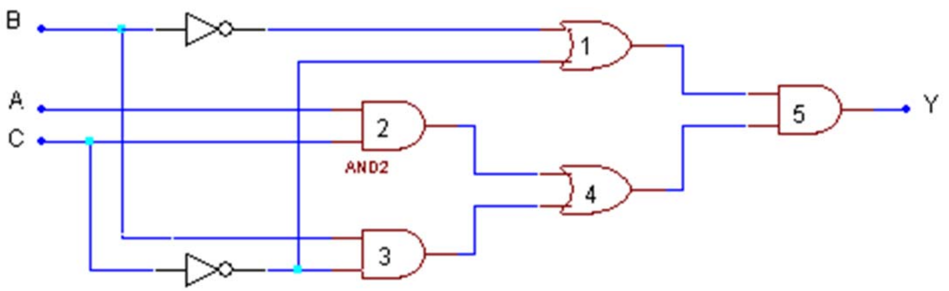
inverter delay = 2ns

and delay = 2.5ns

or delay = 3ns

Path 1, 5 = 2 + 3 + 2.5 = 7.5 ns

Critical Path



inverter delay = 2ns

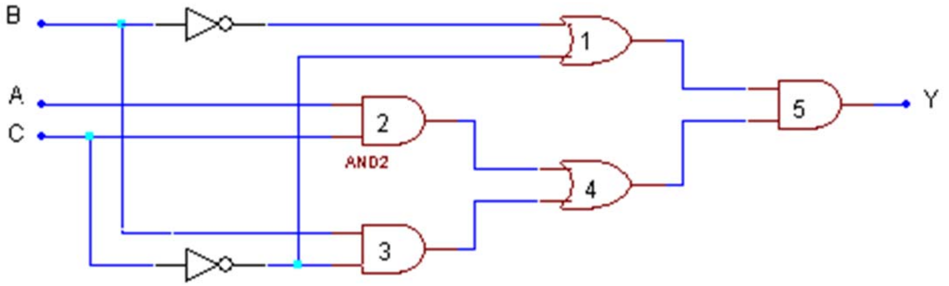
and delay = 2.5ns

or delay = 3ns

Path 1, 5 = 2 + 3 + 2.5 = 7.5 ns

Path 2, 4, 5 = 2.5 + 3 + 2.5 = 8 ns

Critical Path



inverter delay = 2ns

and delay = 2.5ns

or delay = 3ns

Path 1, 5 = 2 + 3 + 2.5 = 7.5 ns

Path 2, 4, 5 = 2.5 + 3 + 2.5 = 8 ns

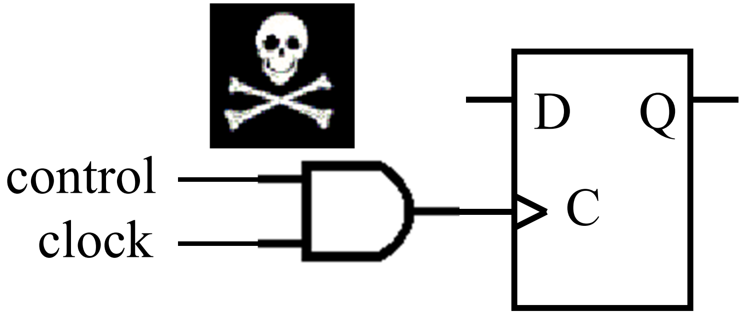
Path inv , 3, 4, 5 = 2 + 2.5 + 3 + 2.5 = 10 ns

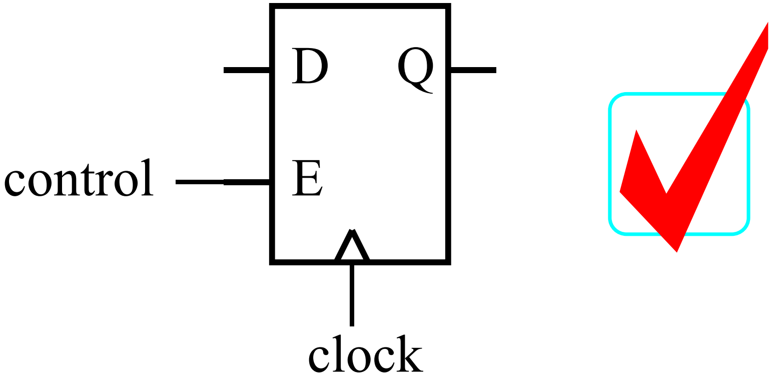
Critical path = 10 ns

Max frequency of operation = 1/10ns = 100 MHz

System Clock

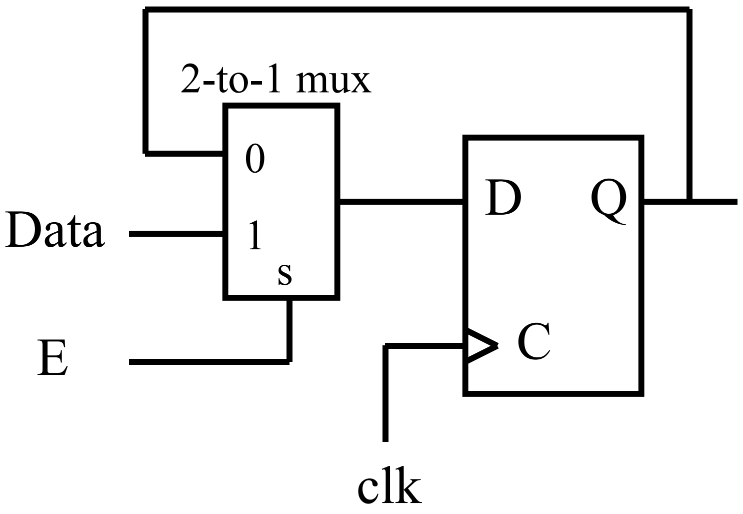
A system clock will generally be free running, but some control is required to decide which system components should be clocked at specific times. Flip-flops with ‘enable’ can be used to design these components. The system clock should not be gated.

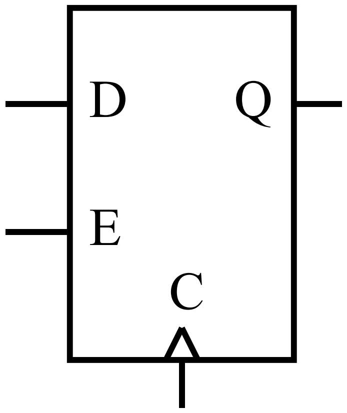




Registers can be built with a common enable line which is used to schedule events. The system clock line must be designed carefully to avoid clock skew. This is the delay between the same clock edge reaching different parts of the system.

Separate control signals are required to decide which specific clock edges have an effect on a particular flip-flop.





The circuit shown will allow loading of the data when the enable line (E) is high.

A register with Enable (E) could be formed by constructing it using flip-flops with enable, and connecting together the enable lines to a single enable input.

Summary

- The speed of operation of circuits is limited by propagation delays.
- Propagation delays in circuits can result in unwanted transient effects called spikes or glitches.
- Synchronous design techniques can be used to overcome problems with glitches.