ELEN0040 - Electronique numérique

Patricia ROUSSEAUX

Année académique 2014-2015

CHAPITRE 5

Sequential circuits design - Timing issues

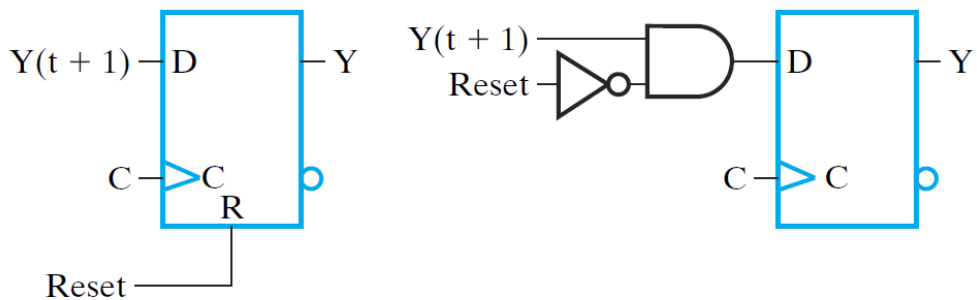
General procedure

Procedure for the design of clocked synchronous sequential circuits.

1. Specification : what the system should do
2. Formulation : obtain either a state diagram or a state machine model and derive the state table
3. State assignment : assign binary codes to the states defined in the state table
4. Outputs and flip-flop inputs equations : derive the outputs and flip-flop inputs equations
5. Optimization : optimize the above equations (use Karnaugh maps or other techniques)
6. Logic diagram : draw the logic diagram of the circuit using flip-flops and primitive gates for the combinational part
7. Technology mapping : choose the gate and flip-flop technology and transform the logic diagram accordingly
8. Verification : verify the correctness of the final design.

Formulation : finding a state diagram

- Key issue : the definition of the states
- Remember :
  - a state remembers meaningful properties of past input sequences that are essential to predict future output values
  - it is an abstraction of the history of the past applied inputs to the circuit, including power-up reset or system reset
  - a careful design avoids the creation of equivalent states which should be finally recognized and merged if present = state minimization procedure
- Usually, the system starts from an initial or reset state with no history
- At start-up, a reset master signal is provided to set the flip-flops of the circuit to a defined state. The reset can be synchronous or asynchronous



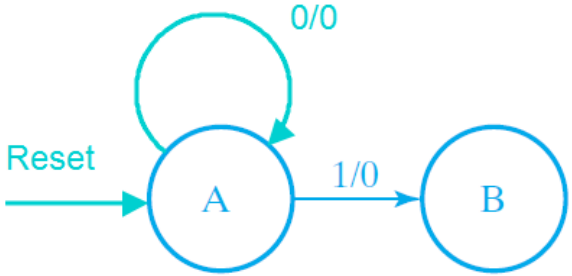
(a) Asynchronous Reset (b) Synchronous Reset

Example : Sequence recognizer

- Specification : design a system that recognizes a given input sequence occurrence
- For example : recognize the sequence "1101" in a continuous stream of input values $X(t)$, regardless of where it occurs.
- Set output $Z(t)$ to one when the sequence has been detected, the three previous values were "110" and the present value $X(t)$ is "1".
- Otherwise $Z(t) = 0$.
- Initially, $Z(0) = 0$.
- Note that the sequence "1101101" contains 1101 as both an initial subsequence 1101101 and a final subsequence 1101101, with the "1" in the middle in both
- The system should recognize the two appearances

Sequence recognizer (continued)

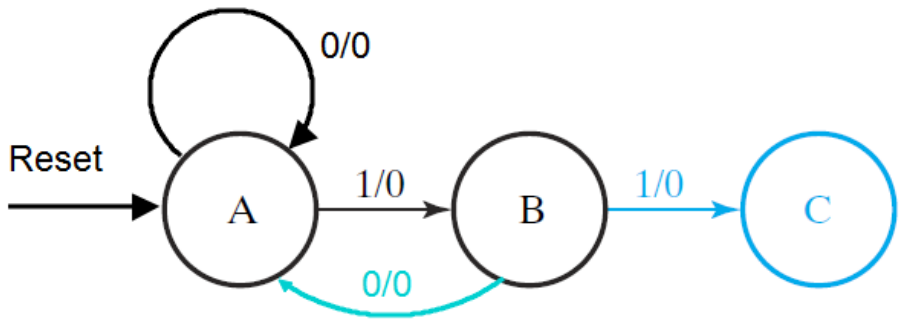
- Formulation : convert the problem specification into a state diagram
  - Since $Z(t)$, the current output, depends on $X(t)$, the current input, it is a Mealy model
  - Start from the reset state, arbitrarily named A, none of the initial portion of the sequence has occurred
  - Add a new state B if a "1" is recognized in the input ($X(t) = 1$), if $X(t) = 0$ the system remains in initial state A
  - The first portion of the state diagram is



Since the searched sequence has not yet been found, $Z(t) = 0$ in both cases

Sequence recognizer (continued)

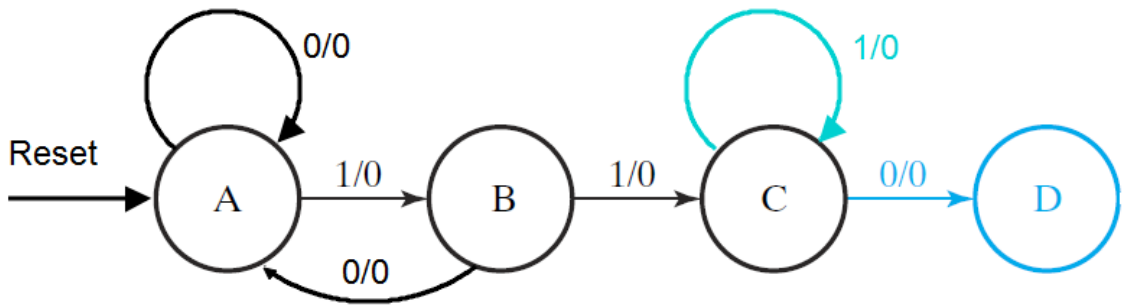
- In state B
  - the fact that a first "1" has been detected is remembered
  - if $X(t) = 1$, the sequence "11" has occurred, a new state C is added
  - state C remembers the sequence "11"
  - if $X(t) = 0$, it means that sequence "10" has appeared, which is not the initial subsequence of "1101", the system has to go back to initial state A, and reinitialize the search of "1101"
  - the corresponding transition arcs are added to the state diagram



  - In both cases, $Z(t) = 0$

Sequence recognizer (continued)

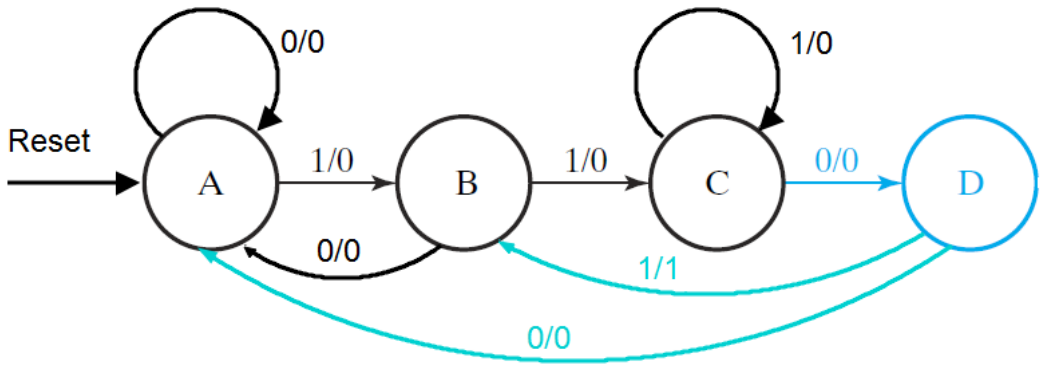
- In state C
  - the fact that the initial sequence "11" has been detected is remembered
  - if $X(t) = 0$, the sequence "110" has occurred, a new state D is added
  - state D remembers the sequence "110"
  - if $X(t) = 1$, it means that sequence "111" has appeared, which is not the initial sequence of "1101"
  - but the system has to remain in state C since the last two inputs are 1 in that case
  - State C means : two or more 1 have occurred as last inputs, which is the initial sequence of the sequence searched
  - The corresponding transition arcs are added to the state diagram



  - In both cases, $Z(t) = 0$

Sequence recognizer (continued)

- In state D
  - the fact that the initial sequence "110" has been detected is remembered
  - if $X(t) = 1$, the searched sequence "1101" is recognized, output is set to 1
  - to which state does the system switch ?
  - not to the initial reset state A but to state B since the last input "1" can also be the first "1" of a new "1101" sequence
  - if $X(t) = 0$, two successive 0 have occurred, the system switches to initial reset state A and $Z(t) = 0$
  - The final state diagram is :



Sequence recognizer : states meanings

To summarize, the states have the following abstract meanings :

A : no proper initial sequence has occurred

B : the initial sequence “1” has occurred

C : the initial sequence “11” has occurred

D : the initial sequence “110” has occurred

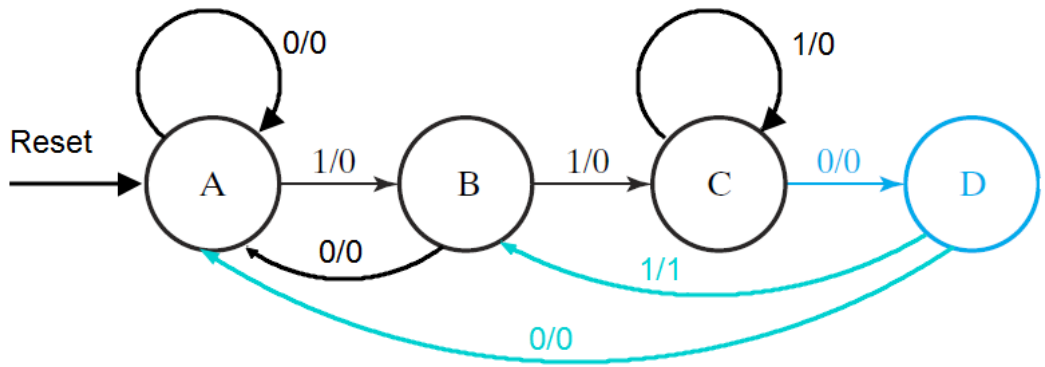
the 1/1 on the arc from D to B means that the last 1 has occurred and thus, the sequence 1101 is recognized

The procedure can be easily extended to longer sequences.

Sequence recognizer : the state table

The state table is derived from the state diagram.

Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1



State Assignment

- Each if the m states must be assigned a unique binary code
- The minimum number of bits required is n such that

$$n \geq \lceil \log_2 m \rceil$$

- State assignment with the minimum number of bits provides circuits with the minimum number of flip-flops
- However, this does not always provide the circuit with the minimum cost ; cost of the combinational circuit has also to be taken into account
- For n bits, there are $2^n!$ possible assignments : examples
  - $n = 1$ bit and 2 states : 2 possible assignments :

$$A = 0, B = 1 \quad \text{or} \quad A = 1, B = 0$$

  - $n = 2$ bits and 4 states : $4! = 24$ possible assignments :

$$A = 00, B = 01, C = 10, D = 11 \quad \text{or}$$

$$A = 00, B = 01, C = 11, D = 10 \quad \text{or} \ \cdots$$

Sequence recognizer : state assignment 1

- Assign binary codes to states in "counting" order

$$A = 00, B = 01, C = 10, D = 11$$

- The state table becomes

Present State	Next State		Output Z	
$Y_1 Y_0$	$X = 0$	$X = 1$	$X = 0$	$X = 1$
00	00	01	0	0
01	00	10	0	0
10	11	10	0	0
11	00	01	0	1

Output and Flip-Flop inputs equations

- Optimize equations using Karnaugh maps
- Flip-flop input equations : state variables Y_1, Y_0, input X

$Y_1(t+1)$

Y_1 \ Y_0 \ X	0	1
00	0	0
01	0	1
11	0	0
10	1	1

$Y_0(t+1)$

Y_1 \ Y_0 \ X	0	1
00	0	1
01	0	0
11	0	1
10	1	0

$Z(t)$

Y_1 \ Y_0 \ X	0	1
00	0	0
01	0	0
11	0	1
10	0	0

$$Y_1(t+1) = Y_1\bar{Y}_0 + \bar{Y}_1 Y_0 X$$

$$Y_0(t+1) = Y_1\bar{Y}_0\bar{X} + \bar{Y}_1\bar{Y}_0 X + Y_1 Y_0 X$$

$$Z(t) = Y_1 Y_0 X$$

Sequence recognizer : state assignment 2

- Assign binary codes according to Gray code order

$$A = 00, B = 01, C = 11, D = 10$$

- The state table becomes

Present State	Next State		Output Z	
$Y_1 Y_0$	$X = 0$	$X = 1$	$X = 0$	$X = 1$
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

Output and Flip-Flop inputs equations

- Optimize equations using Karnaugh maps
- Flip-flop input equations : state variables Y_1, Y_0, input X

$Y_1(t+1)$

Y_1 \ Y_0 \ X	0	1
00	0	0
01	0	1
11	1	1
10	0	0

$Y_0(t+1)$

Y_1 \ Y_0 \ X	0	1
00	0	1
01	0	1
11	0	1
10	0	1

$Z(t)$

Y_1 \ Y_0 \ X	0	1
00	0	0
01	0	0
11	0	0
10	0	1

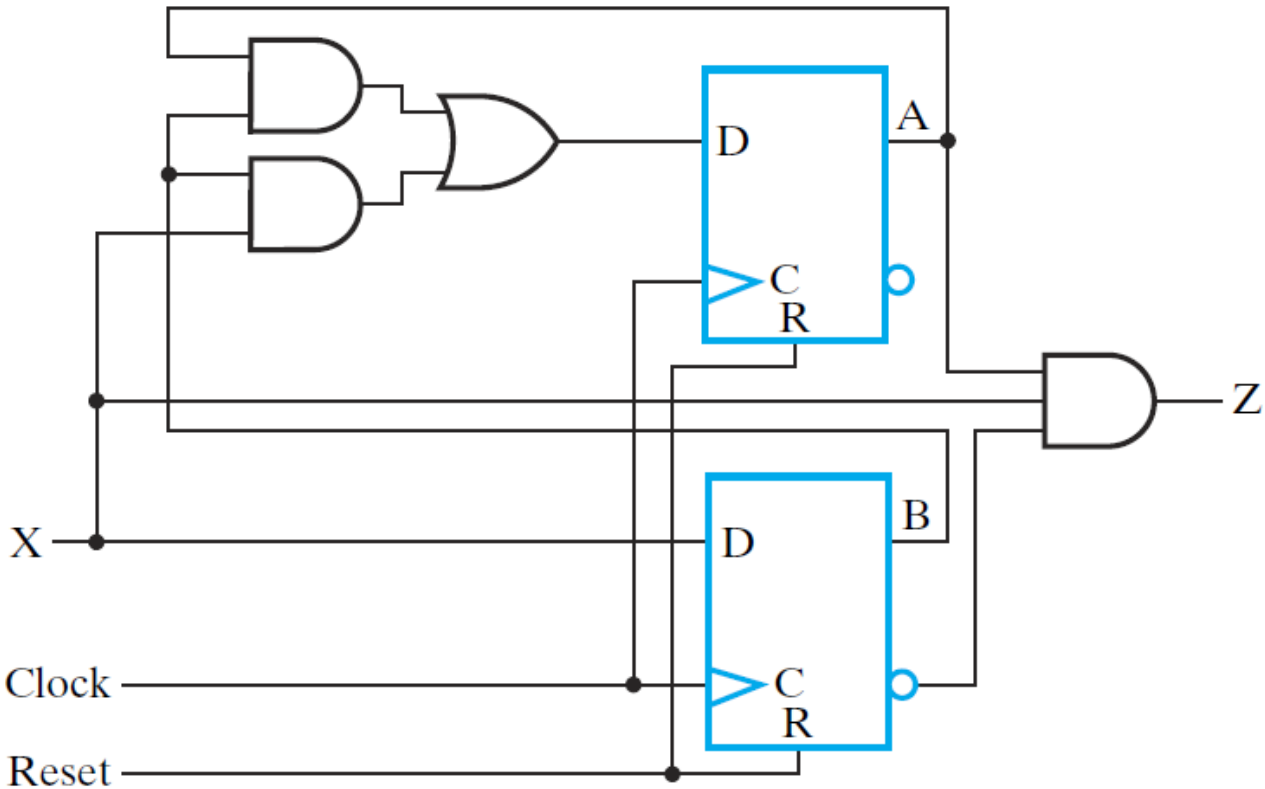
$$Y_1(t+1) = Y_1 Y_0 + Y_0 X$$

$$Y_0(t+1) = X$$

$$Z(t) = Y_1 \bar{Y}_0 X$$

- The cost is lower with this state assignment !

Logic diagram using primitive gates

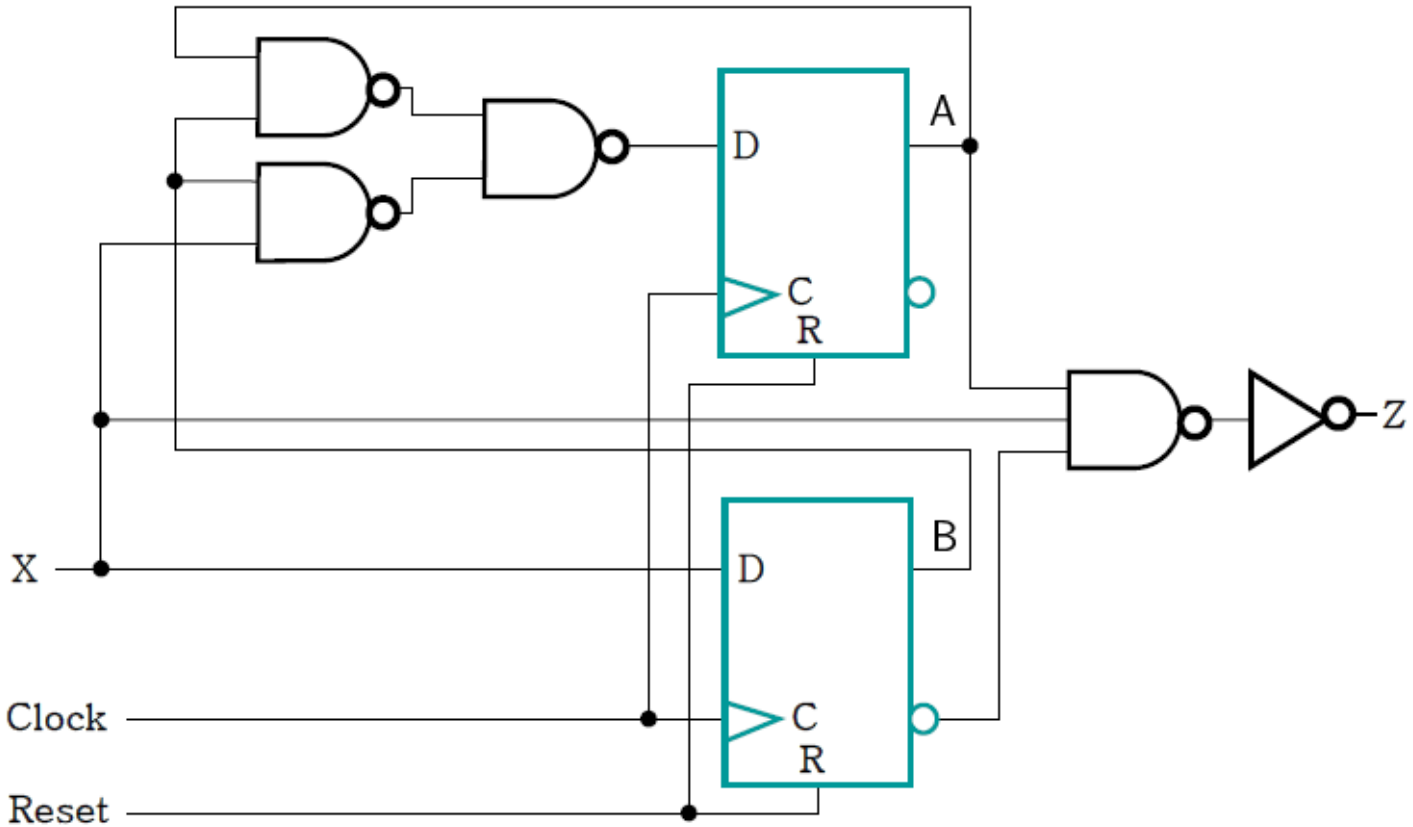


$$A = Y_1 \qquad B = Y_0$$

Technology mapping

Realize the circuit with

- D flip-flops with asynchronous reset
- NAND gates with up to 4 inputs
- inverters



$$A = Y_1 \qquad B = Y_0$$

Sequence recognizer : state assignment 3

- Assign binary codes according to "One-Hot" assignment

$$A = 0001, B = 0010, C = 0100, D = 1000$$

- There is only one "1" in each code word
- $n > \lceil \log_2 m \rceil$, there exists "unused" states
- can lead to a simpler combinational circuit but at the price of additional flip-flops
- The state table becomes

Present State	Next State		Output Z	
$Y_3\,Y_2\,Y_1\,Y_0$	$X = 0$	$X = 1$	$X = 0$	$X = 1$
1000	1000	0100	0	0
0100	1000	0010	0	0
0010	0001	0010	0	0
0001	1000	0100	0	1

Output and Flip-Flop inputs equations

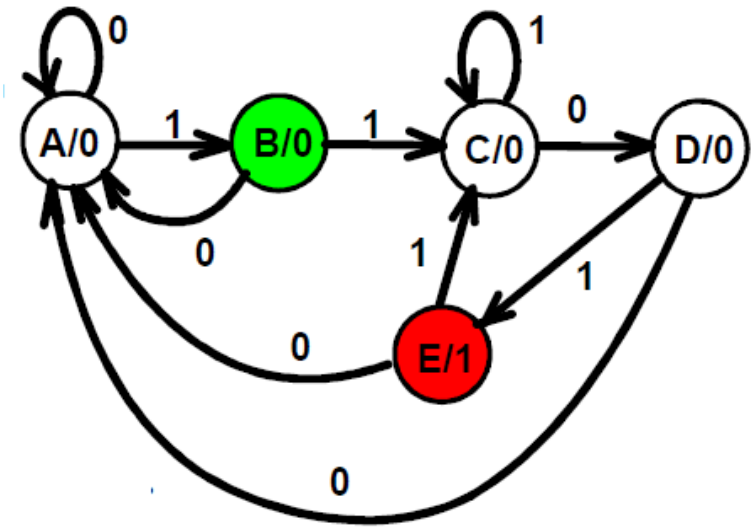
- The unused states are "don't care". This gives simple equations.

$$
\begin{aligned}
Y_3(t+1) &= \bar{X}(Y_3 + Y_2 + Y_0) = \bar{X}\bar{Y}_1 \\
Y_2(t+1) &= X(Y_3 + Y_0) \\
Y_1(t+1) &= X(Y_2 + Y_1) \\
Y_0(t+1) &= \bar{X}Y_1 \\
Z &= XY_0
\end{aligned}
$$

- 4 flip-flops are needed

Sequence recognizer : Moore model

- In Moore models, the output is associated to the state
- State D gives rise to two different output values depending on the input X
- To build a Moore model, we need to add a state "E" with output value "1" that acknowledges the recognition of the searched sequence
- This new state is "similar" to B but with different output values
- Generally, the Moore model of a system has more states than the Mealy model
- The Moore model of the sequence recognizer has the following state diagram

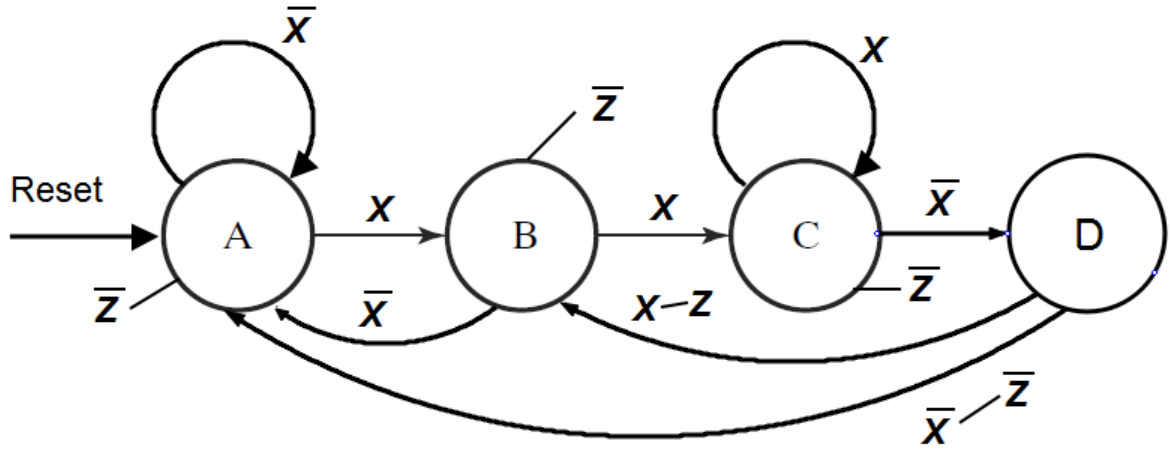


State machine model

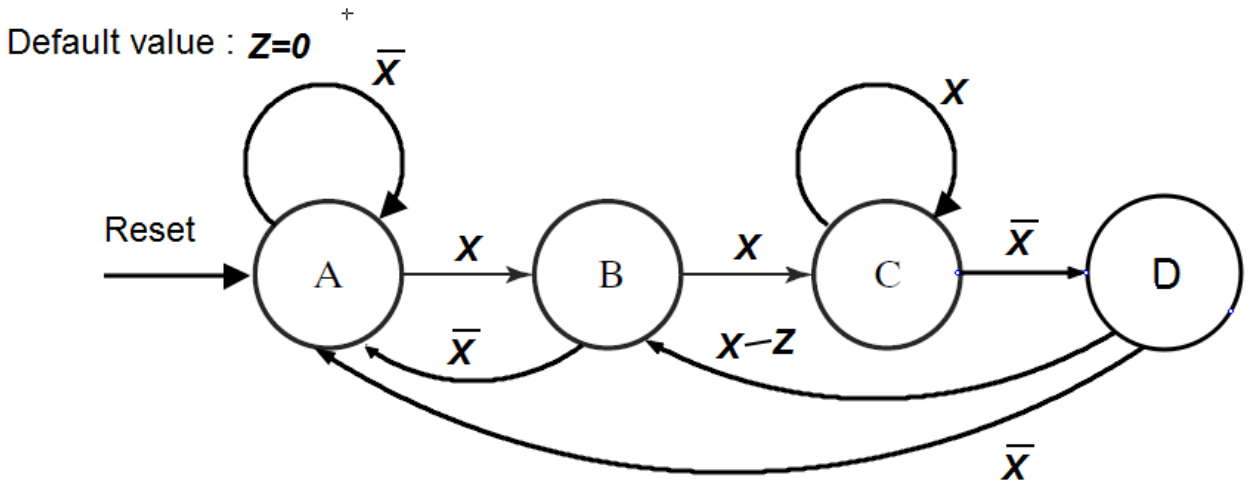
- State diagrams and state table requires
  - enumeration of all input combinations for each state in defining next state
  - enumeration of all input combinations for each state in defining Mealy outputs
  - enumeration of all applicable output combinations for each state (Moore) and for each input combination-state pair (Mealy)
- For systems with larger numbers of inputs and states, these representations become intractable
- A more elaborate representation uses Finite State Machine model
  - The Finite State Machine is a mathematical model used for sequential circuits
  - Roughly speaking, the enumerations are replaced by Boolean expressions of transition conditions and output conditions to provide a State Machine Diagram

Example of State Machine Diagram

The state diagram of the sequence recognizer becomes



Often, default values are defined for outputs to simplify the diagram :



Integrated circuits

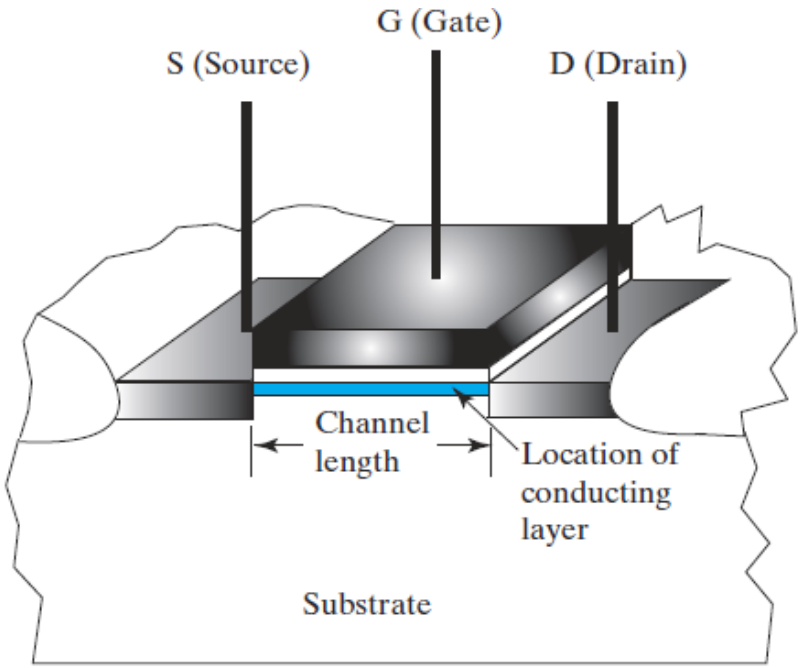
- Digital circuits are constructed with integrated circuits
- An integrated circuit or a "chip" is a semiconductor crystal (most often silicon) containing the electronic components, the gates and the memory storage elements
- Basic IC components may be interconnected through "pins"
- Different levels of integration :
  - SSI : fewer than 10 gates
  - MSI : 10s to 100s gates (functional blocks of chapter 3)
  - LSI : up to thousands of gates (small processors, small memories)
  - VLSI : 100s to 100s of millions of gates (complex μprocessors, complex structures)

Implementation technologies

- Digital circuits are also classified according to their implementation technology
- Each technology has its own electronic devices and circuit structures
- Each technology is characterized by its own electrical properties and parameters
- Principle technologies :
  - TTL (Transistor-Transistor logic) : use bipolar junction transistors (BJT) ; the oldest technology, lowest cost
  - CMOS (Complementary Metal–Oxide–semiconductor) : most used nowadays for high integration level : higher circuit density and lower power consumption
  - alternative technologies used for μelectronics : the variant of CMOS SOI technology (Silicon On Insulator), technologies based on gallium arsenide (GaAs) and silicon germanium (SiGe)

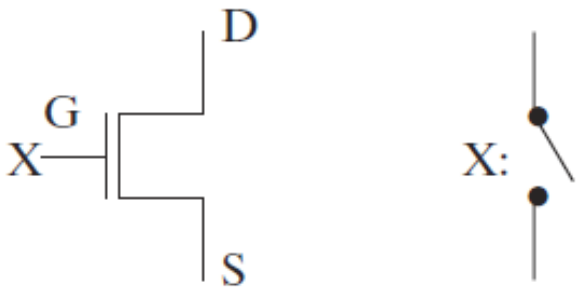
CMOS technology

The CMOS technology uses MOS transistors



- conduction through the channel controlled by the gate voltage
- the transistor acts as a "switch"
- if $V_G - V_S > \varepsilon$, conduction, "switch" closed
- if $V_G - V_S < \varepsilon$, no conduction, "switch" open

Switch model for logic implementation



- S connected to voltage 0
- D connected to high voltage
- G connected to input signal = logic variable X
- "switch" closed if $X = 1$

- logic functions are realized by circuits of switches that model transistors
- Logic value 1 is implemented if there exists a closed path through the circuit
- There exist two families of MOS transistors : n-channel and p-channel
- Both are used to implement ComplementaryMOS circuits

Technology parameters

Parameters that have to be taken into account in the circuit design procedure. They are linked to electronic considerations.

Fan-in : the number of inputs of a gate. The number of inputs may limit the speed of the circuit so that a maximum fan-in may be imposed in the design procedure.

Fan-out : the number of standard loads driven by a gate output cannot exceed some limit

Cost of a gate : often related to the area occupied by the gate which depends on the number of transistors, directly linked to the gate input cost

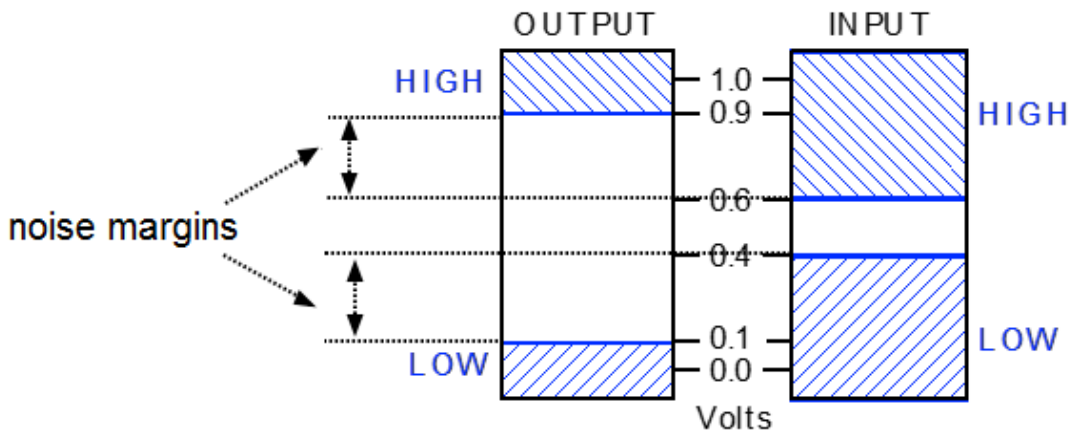
Propagation delay : the time required for a change in an input value to propagate in the circuit to force change in the output

Power dissipation : the power consumed by the gate and dissipated as heat (cooling requirements !)

Technology parameters (continued)

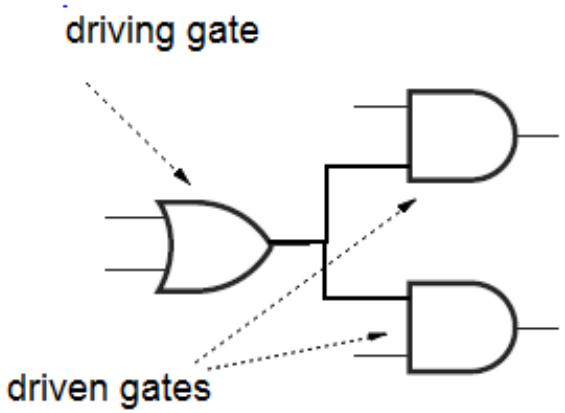
Logic levels : the voltage ranges for representation of 1 and 0 on the inputs and on the outputs of the gate

Noise margin : the maximum external noise voltage that can be superimposed on a normal input value and will not cause an undesirable change in the output



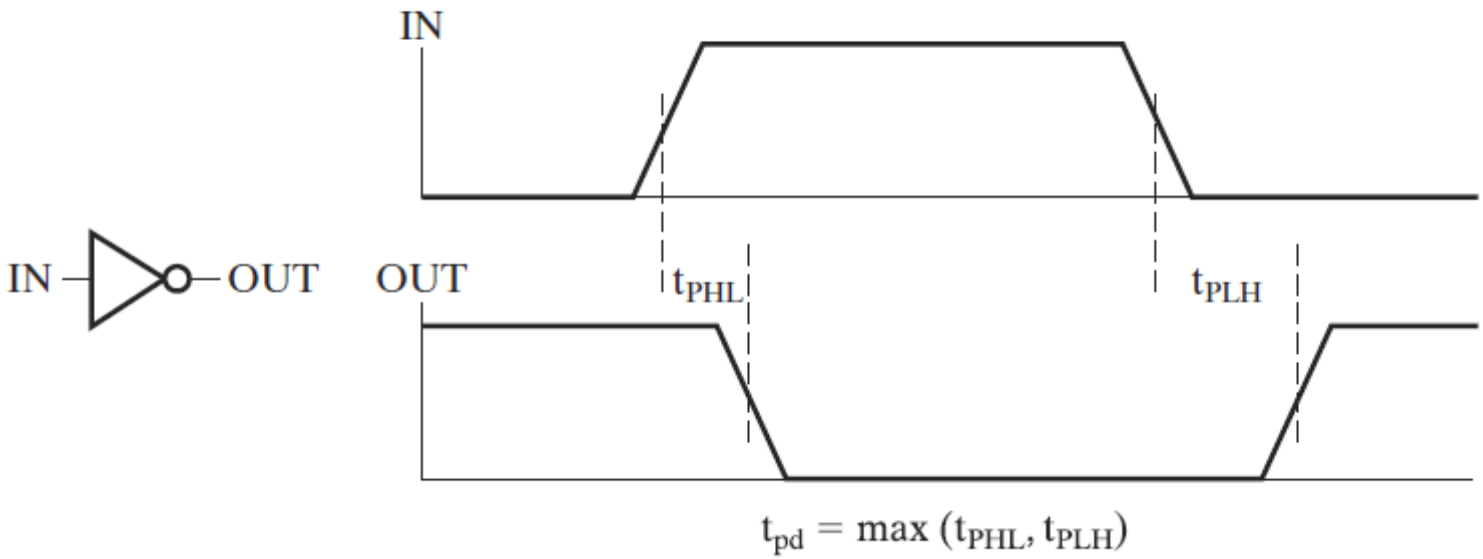
Fan-out

- Each input on a given gate provides a load on the output of the driving gate
- This load is expressed in terms of a standard load
- The fan-out is obtained by adding the contributions of all driven gates
- The determination of the load is function of the technology
  - for CMOS technology : load=capacitance
  - the capacitance has an effect on the time required for the output of the driven gate to change from LOW to HIGH or HIGH to LOW voltage levels = transition time
  - the maximum fan-out that can be driven by the gate is thus fixed by the maximum transition time



Gate propagation delay

- The time elapsed between a change in an input and the corresponding change in the output
- H-to-L and L-to-H may have different propagation delays
- In practice, changes in input and output do no occur instantaneously

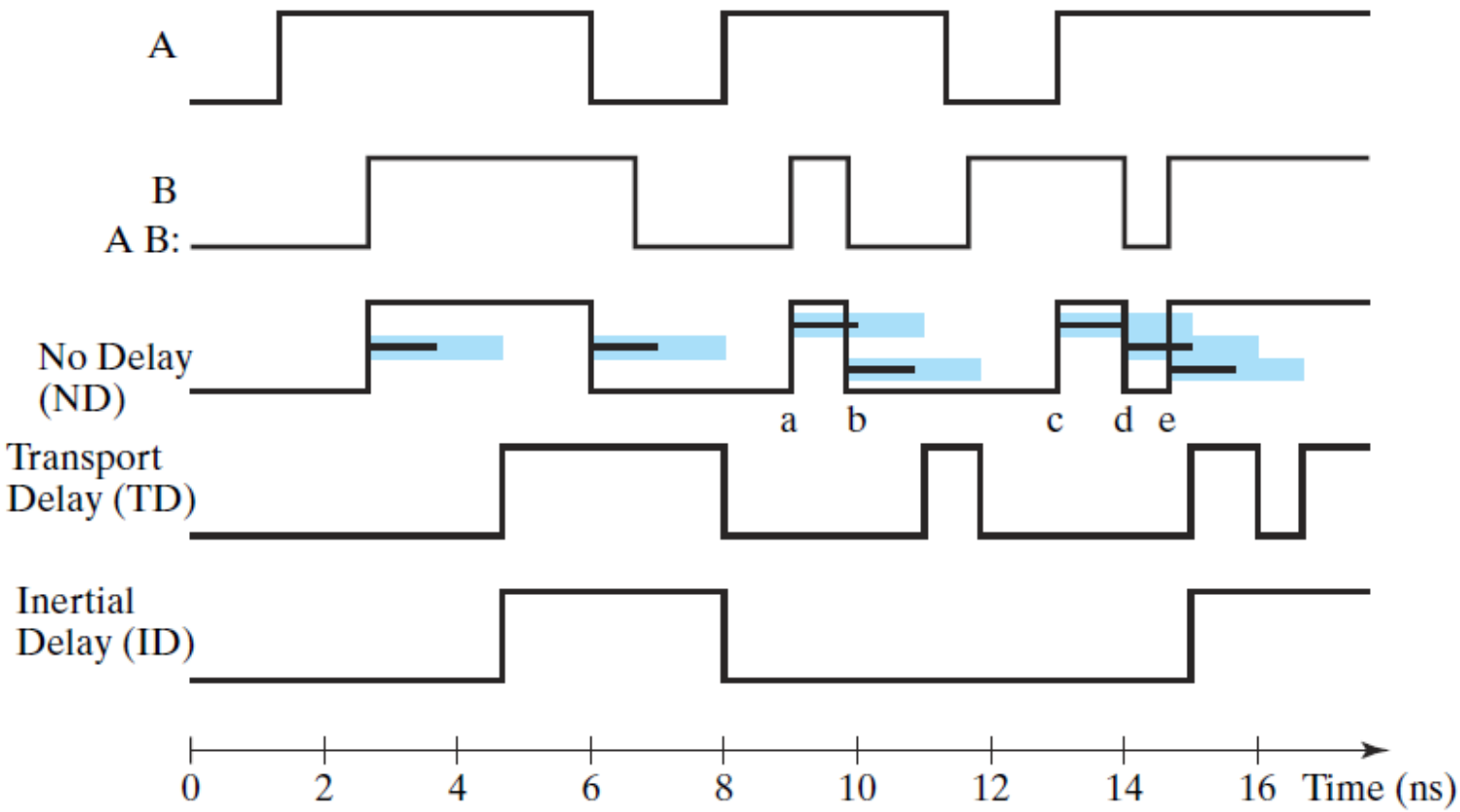


- Delay is usually measured at 50% of the H and L voltage levels

Delay in simulation

- Two different models are used for modeling gates in simulation :
  - transport delay : the change in the output occurs after a specified propagation delay, different for each type of gate
  - Inertial delay : if the input changes cause the output to change twice in a interval less than the rejection time, the output changes do not occur
  - Narrow "pulses" are rejected

Example

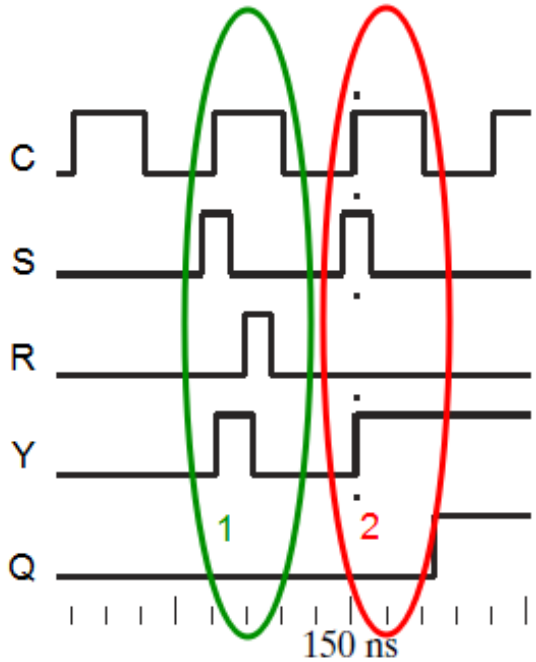


Propagation delay = 2 ns Rejection time = 1 ns

Circuit delay

- When interconnected with other gates in a digital circuit, the gate delay is affected by the fan-out
- The gate delay is made of two terms :
  - a constant delay, inherent to the gate and linked to its physical properties
  - a variable delay proportional to the fan-out of the gate, and thus dependent on the circuit topology
- Cost and speed are contradictory requirements :
  - speed is limited by the fan-out
  - to limit the fan-out, gates driving many inputs can be replaced by multiple gates in parallel
  - this implies a cost increase
- A compromise has to be found depending on the particular application

Master-Slave flip-flop and the 1's catching problem



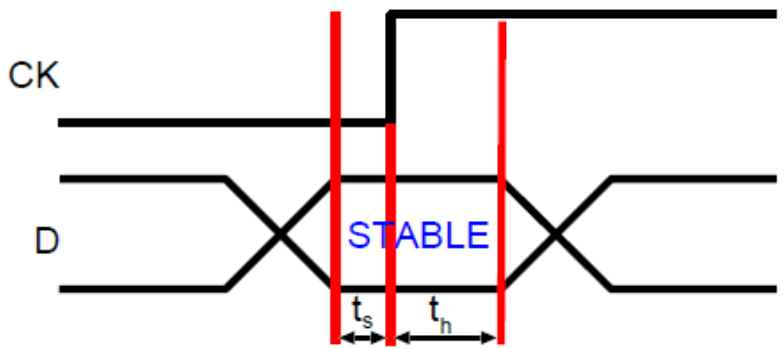
- Expected behavior : $Q(t+1)$ after falling edge of clock is determined from :
  - $Q(t)$ existing during the preceding clock period, i.e. just before clock goes high
  - the values of the inputs just before the clock goes down in the present clock period
- Suppose $Q = 0$, if
  1. S goes to 1 then back to 0 and then R goes to 1 and back to 0 while C still at 1
     - Y, output of the master latch, follows and goes to 1 and then to 0 after $R = 1$
     - finally 0 is passed to slave and $Q = 0$
  2. S goes to 1 then back to 0 and then R remains 0
     - Y is set to 1 and does not change anymore
     - 1 is passed to slave and $Q = 1$

- Case 1 is OK
- Case 2 does not provide the expected output since Q was zero before the clock pulse and S and R are both zero just before the clock goes to 0

Flip-Flop timing parameters

Three timing parameters are associated to the operation of pulse-triggered (master-slave) or edge-triggered flip-flops

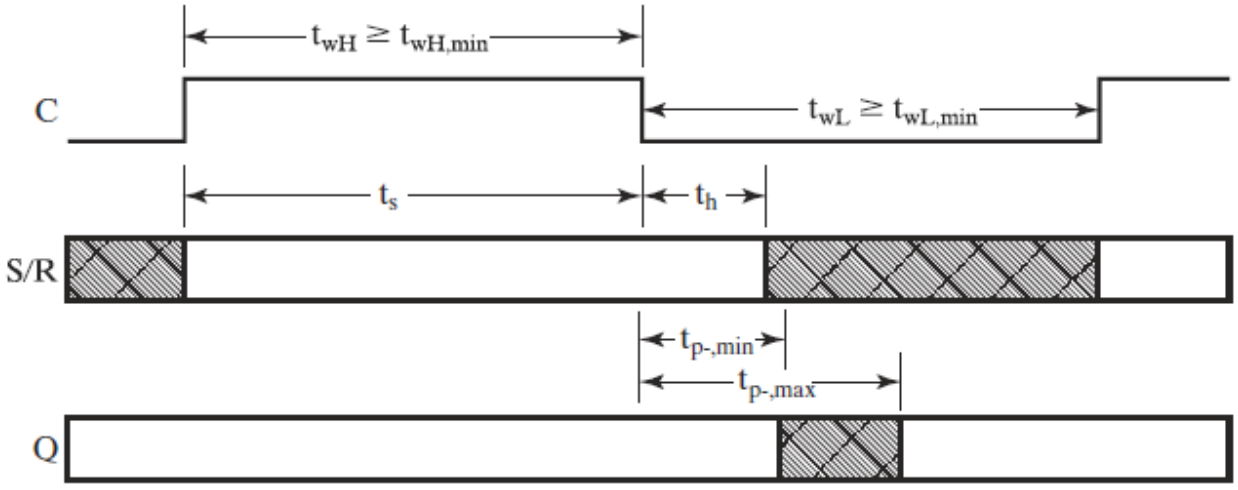
- Setup time t_s : the minimum time during which the inputs (S,R or D) must not change before the occurrence of the clock transition that triggers the output change
- Hold time t_h : the minimum time during which the inputs (S,R or D) must not change after the clock transition



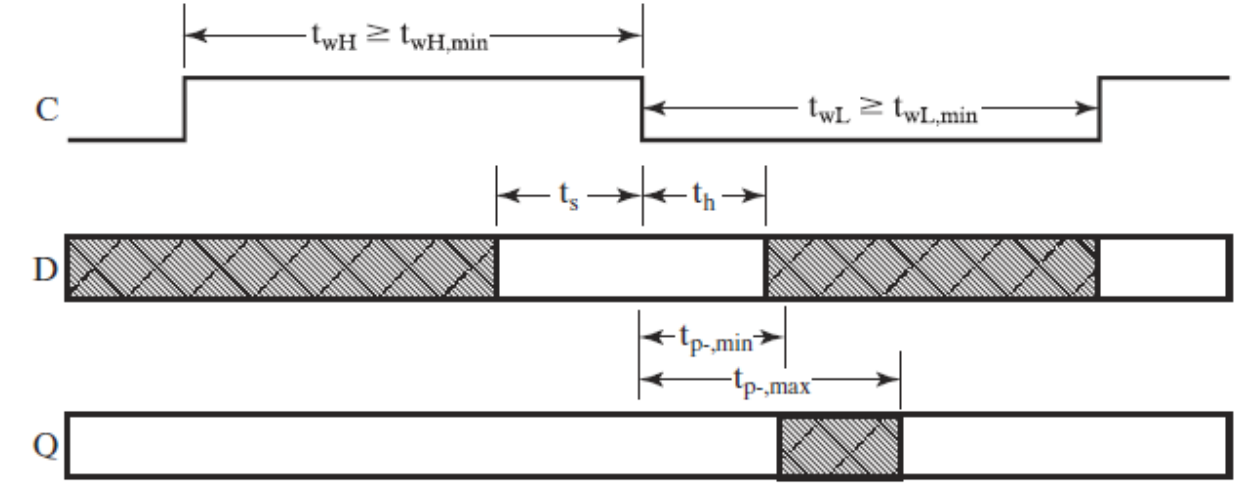
Flip-Flop timing parameters (continued)

- In practice
  - for pulse-triggered flips-flops : t_s is equal to the width of the triggering pulse (remember the 1's catching problem)
  - for edge-triggered flip-flops : t_s is smaller than width of the triggering pulse
  - edge triggering tends to provide faster designs since the flip-flop input can change later
  - t_h is often neglected
- The propagation delay t_{pd} is the interval between the triggering clock edge and the stabilization of the output to the new value
  - The minimum propagation delay should be longer than the hold time !

Flip-Flop timing parameters (continued)

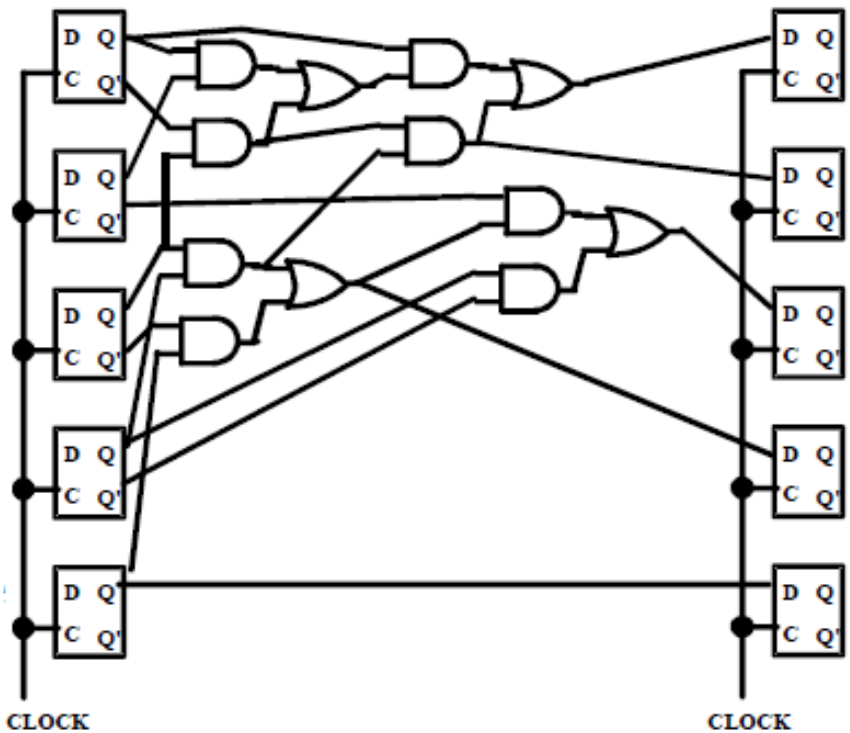


(a) Pulse-triggered (positive pulse)



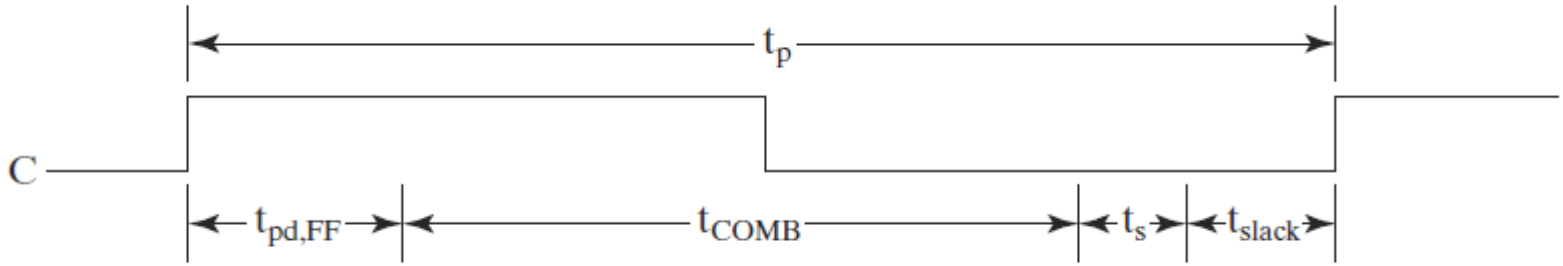
Sequential circuit timing

- The design of a sequential circuit must include timing specifications for a proper operation of the system
- They depend on the type of flip-flops used and concern :
  - flip-flops setup times and hold times
  - the minimum clock width or maximum clock frequency
  - the propagation delays



- Let us consider a system comprising 2 stages of flip-flops connected by a combinational circuit (shift register)
- if the clock period is too short, data changes will not propagate through the circuit to flip-flop inputs before the setup time interval begins

Decomposition of the clock period



(a) Edge-triggered (positive edge)

- t_p is the clock period
- $t_{pd,FF}$ is the propagation delay of the flip-flop
- $t_{pd,COMB}$ is the total propagation delay from a flip-flop output to a flip-flop input through the combinational circuit
- For proper operation : $t_{slack} \geq 0$
- Minimum clock period :

$$t_p \geq \max(t_{pd,FF} + t_{pd,COMB} + t_s)$$

  the max is taken over all possible paths from flip-flops outputs to flip-flops inputs
- Or, for a given clock period, maximum propagation delay of the combinational circuit

Example

Compute the maximum delay allowed for the combinational circuit for

- an edge-triggered flip-flop
- a master slave flip-flop
- Timing parameters
  - clock frequency = 250 MHz $\Rightarrow t_p = 4$ ns
  - $t_{pd,FF} = 1$ ns
  - $t_s = 0.3$ ns for the edge-triggered flip-flop
  - $t_s = t_{wH} = 1$ ns for the master slave flip-flop
  - clock frequency = 250 MHz $\Rightarrow t_p = 4$ ns
  - average gate delay (one gate of the combinational circuit) : t_g =0.3 ns
- edge-triggered flip-flop

  $4 \geq 1 + t_{pd,COMB} + 0.3 \Rightarrow t_{pd,COMB} \leq 2.7$ ns $\Rightarrow$ 9 gates max on a path
- master slave flip-flop

  $4 \geq 1 + t_{pd,COMB} + 1 \Rightarrow t_{pd,COMB} \leq 2$ ns $\Rightarrow$ 6 gates max on a path

Terms of Use

- All (or portions) of this material © 2008 by Pearson Education, Inc.
- Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.
- These materials or adaptations thereof are not to be sold or otherwise offered for consideration.
- This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.

Références

- *Logic and Computer Design Fundamentals, 4/E*, M. Morris Mano Charles Kime , Course material
  http ://writphotec.com/mano4/
- *Cours d'électronique numérique*, Aurélie Gensbittel, Bertrand Granado, Université Pierre et Marie Curie
  http ://bertrand.granado.free.fr/Licence/ue201/coursbeameranime.pdf
- *Lecture notes, Course CSE370 - Introduction to Digital Design*, Spring 2006, University of Washington,
  https ://courses.cs.washington.edu/courses/cse370/06sp/pdfs/