Timing Consideration

- Circuit timing is a very important consideration in the design of any electronic systems
- The following timing issues are considered:
  For Flip-flops:
  – Set-up time
  – Hold time
  – Propagation delay
  For Combinational circuits:
  – Contamination delay
  – Propagation delay
  For Sequential circuits: (this lecture)
  – Clock frequency / Clock cycle time

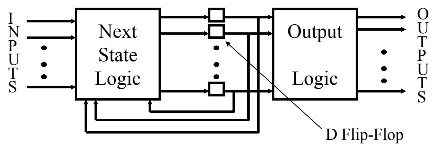
Speed of Sequential circuit and Clock frequency

- Clock frequency
  – is the number of rising clock edges (clock ticks) in a fixed period of time
  – determines the speed of a sequential circuit
- Clock cycle time (or clock period) is the time between two rising clock edges
- If circuit runs at clock frequency of f, corresponding clock cycle time is
  – T = 1/f, or
  – f = 1/T
- A frequency of 1 MHz gives a clock period of 1 micro second
- A frequency of 500 MHz gives a clock period of 2 nano second
- A frequency of 2 GHz gives a clock period of 0.5 nano second
- A frequency of 1 GHz gives a clock period of 1 nano second

(1 micro second = 1e-6 second, 1 nano second = 1e-9 second)

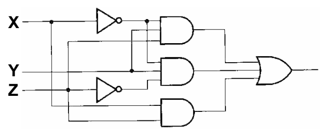
Timing in a Sequential Circuit (State machine)

- From a rising clock edge, we should allow enough time for:
  – D FFs to generate stable output for the state
  – next state logic to generate the next state
  – D FFs to set up after the next state is available
- Then we can have the next rising clock edge
- Thus, D Flip-Flop propagation delay + Next state logic propagation delay + D FF set-up time sets a lower bound to the clock cycle time



Review: Timing Issues of Combinational Circuits

- Contamination delay:
  – Minimum delay before any output starts to change once input changes
- Propagation delay:
  – Maximum delay after which all outputs are stable once input changes



- Contamination delay = 2
- Propagation delay = 3

(Assume that delay of all gates = 1)

Propagation delay for next-state logic

- The propagation delay for next-state logic is also called the compute time
- Consider a four states system

$$\begin{tikzcd} S0 \arrow[r] & S1 \arrow[r] & S2 \arrow[r] & S3 \arrow[lll, bend left] \end{tikzcd}$$

- State transition table and implementation level state transition table are given below

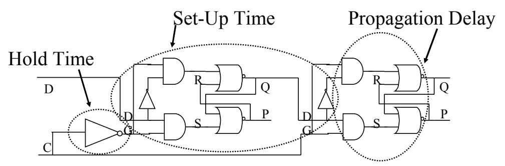
Current State	Next State
S0	S1
S1	S2
S2	S3
S3	S0

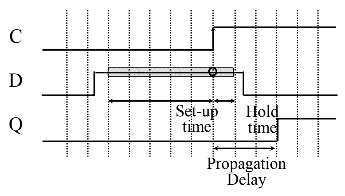
Current X Y	Next X Y
0 0	0 1
0 1	1 0
1 0	1 1
1 1	0 0

- Using the logic expressions below, combination logic for next state takes up to two gate delay (if both X and X' are available)

$$X := X'Y+XY'$$
$$Y := X'Y'+XY' = Y'$$

Review: Timing Issues of FFs



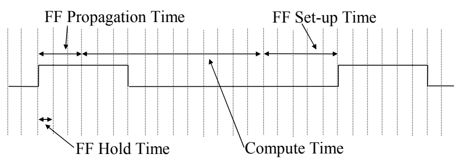


For this design:
- Set-up time = 5
- Hold time = 1
- Prop. delay = 3

(Assume that delay of all gates = 1)

Timing Constraints for a Sequential Circuit

- Clock cycle time >= FF Prop delay + Compute time + FF set-up time
- Clock low time >= FF set-up time
- Clock high time >= FF Prop delay
- Contamination time of next state circuit >= FF hold time

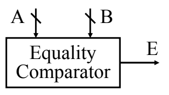


Some Relationships in Sequential Circuit timing

- Let
  - T be the clock period
  - t_{pd} be the propagation time of next-state logic circuit
  - t_{cd} be the contamination time of next-state logic circuit
  - t_{rd} be the propagation time of FF (register) circuit
  - t_{st} be the set-up time of FF (register) circuit
  - t_{ht} be the hold time of FF (register) circuit
- What is the minimum clock period? $T = t_{pd} + t_{rd} + t_{st}$
- By how long must any change in external inputs precede the next clock edge? $>= t_{pd} + t_{st}$
- How long after the clock edge must the external inputs be held valid? $>= t_{ht} - t_{cd}$
- What is the smallest time after the clock edge that outputs of state machine can be expected to be valid?
  t_{rd}+ propagation delay of output logic

n-bit Equality Comparator

- First, we build a circut to compare two n-bit numbers for equality

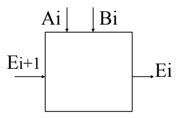


If A=B, E=1
If A!=B, E=0

- We want to build the n-bit circuit using 1-bit building blocks
- Start with MSB and compare one bit at a time
  - If the two bits are different, then the numbers are different. This becomes the solution.
  - Else, i.e, if the two bits are equal, then compare the next bit.

Basic building block for n-bit Equality Comparator

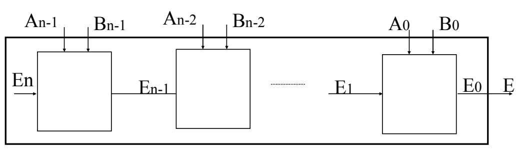
- 1-bit comparator forms the building block for comparing two n-bit numbers.
- E is the cascading signal
- E_{i+1} is the Cascading input, E_i is the cascading output
- $E_{i+1} = 0$ implies that the two numbers are not equal so far
- $E_{i+1} = 1$ implies that the two numbers are equal so far
- If $E_{i+1} = 0$, then $E_i = 0$
- else if $E_{i+1} = 1$, then E_i = Ai NXOR Bi
- In other words, $E_i = E_{i+1}$. (Ai NXOR Bi)



Ai	Bi	E_{i+1}	E_i
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Building an n-bit Equality Comparator

- Using the building block, we can build an n-bit comparator circuit
- The final result is, if $E_0 = 1$, then A = B
  and if $E_0 = 0$, then A != B
- What is the value of input E_n ?

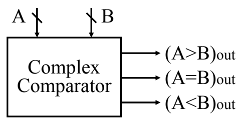


Another View of the n-bit Equality Comparator

- $E_{n-1} = E_n$. (An-1 NXOR Bn-1) = (An-1 NXOR Bn-1)
- $E_{n-2} = E_{n-1}$. (An-2 NXOR Bn-2)
- .....
- $E_1 = E_2$. (A1 NXOR B1)
- $E_0 = E_1$. (A0 NXOR B0)
- Therefore,
  - E_0 = (An-1 NXOR Bn-1) . (An-2 NXOR Bn-2)
    .... (A1 NXOR B1) . (A0 NXOR B0)
- We don't need to build a chain. We can use n NXOR gates to compare the n bit pairs first. Then we can use a n-bit AND gate to combine the n results together.

n-bit Complex Comparator

- Next, we compare the two numbers for A > B, A = B, or A < B

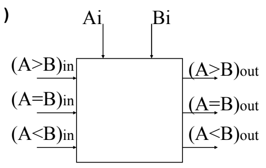


If A>B, $(A>B)_{out}=1$, $(A=B)_{out}=0$, $(A<B)_{out}=0$
If A=B, $(A>B)_{out}=0$, $(A=B)_{out}=1$, $(A<B)_{out}=0$
If A<B, $(A>B)_{out}=0$, $(A=B)_{out}=0$, $(A<B)_{out}=1$

- Again, we want to build the n-bit circuit using 1-bit building blocks
- Similarly, we start with MSB and compare one bit at a time

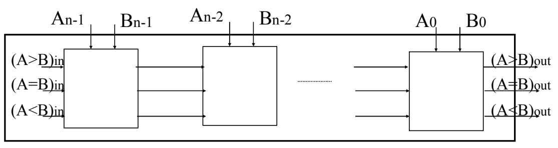
1-bit Complex Comparator

- The basic building block for the purpose is as shown
  - The primary inputs are Ai and Bi
  - Cascading inputs are $(A>B)_{in}$, $(A=B)_{in}$, and $(A<B)_{in}$
  - Cascading outputs are $(A>B)_{out}$, $(A=B)_{out}$, and $(A<B)_{out}$
- The equations can be directly written as
- $(A>B)_{out} = (A>B)_{in} + (A=B)_{in} . (A_i.B_i')$
- $(A<B)_{out} = (A<B)_{in} + (A=B)_{in} . (A_i'.B_i)$
- $(A=B)_{out} = (A=B)_{in} . (A_i B_i + A_i'.B_i')$



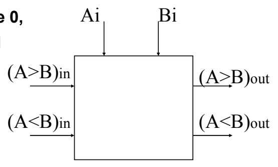
Building an n-bit Complex Comparator

- Using the building block, we can build an n-bit comparator circuit
- The final result:
  - If A>B, $(A>B)_{out}=1$, $(A=B)_{out}=0$, $(A<B)_{out}=0$
  - If A=B, $(A>B)_{out}=0$, $(A=B)_{out}=1$, $(A<B)_{out}=0$
  - If A<B, $(A>B)_{out}=0$, $(A=B)_{out}=0$, $(A<B)_{out}=1$
- What are the values of the inputs $(A>B)_{in}$, $(A=B)_{in}$, $(A<B)_{in}$?



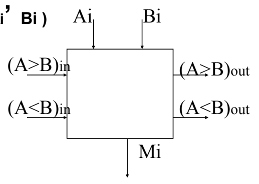
1-bit Simplified complex comparator

- The 1-bit complex comparator block is modified to have fewer cascading signals
- It has two primary inputs, two cascading inputs, and two cascading outputs
- The equations for the outputs are
- $(A>B)_{out} = (A>B)_{in} + (A<B)_{in}' . (A_i . B_i')$
- $(A<B)_{out} = (A<B)_{in} + (A>B)_{in}' . (A_i'. B_i)$
- If both $(A>B)_{out}$ and $(A<B)_{out}$ are 0, then the two numbers are equal so far



1-bit maximizer building block

- The 1-bit maximizer block is similar to the simplified complex comparator
- Primary inputs: Ai, Bi — Cascading inputs: $(A>B)_{in}$, $(A<B)_{in}$
  Primary output: Mi — Cascading outputs: $(A>B)_{out}$, $(A<B)_{out}$
- Mi is either Ai or Bi depending on whether A or B is greater (not just depending on the current bits Ai and Bi)
- The equations for the outputs are
  - $(A>B)_{out} = (A>B)_{in} + (A<B)_{in}' . (A_i \; B_i')$
  - $(A<B)_{out} = (A<B)_{in} + (A>B)_{in}' . (A_i' \; B_i)$
  - $M_i = (A>B)_{in} . A_i + (A<B)_{in} . B_i + (A>B)_{in}' . (A<B)_{in}' . (A_i + B_i)$


Ai
Bi
$(A>B)_{in}$
$(A<B)_{in}$
$(A>B)_{out}$
$(A<B)_{out}$
Mi