Chapter 1

Review of Logic Design Fundamentals

J.J. Shann

1

Chapter Overview

2

Exercises in Textbook

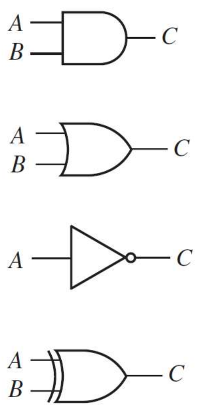
Sections	Exercises
§1-1	1.5
§1-2	1.4
§1-3	1.1, 1.2
§1-5	1.3, 1.6, 1.9, 1.10
§1-6	1.7, 1.8
§1-7	1.29~1.32
§1-8	1.11, 1.12, 1.28
§1-9	1.20
§1-10	1.13~1.19, 1.21~1.23, 1.25~1.27
§1-11	1.24

1-1

Combination Logic

Basic Gates

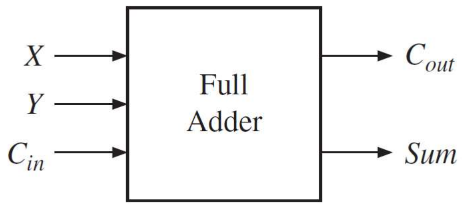
- AND gate:
  C = A AND B = A • B = AB
- OR gate:
  C = A OR B = A + B
- NOT gate or Inverter:
  C = NOT A = A′
- XOR gate:
  C = A XOR B = AB′ + A′B = A⊕B



J.J. Shann 1-5

5

Example: Full Adder



(a) Full adder module

X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

(b) Truth table

J.J. Shann 1-6

6

Canonical Forms

- Derive directly from a Boolean function's truth table
  - Generally are not the simplest form (can be minimized)
- Two canonical forms:
  - ***Sum of*** **minterms**
  - ***Product of maxterms***

Sum of Minterms

X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- Sum of minterms:
  - E.g.: $C_{out} = X'YC_{in} + XY'C_{in} + XYC_{in}' + XYC_{in}$
- Minterm expansions can be written in m-notation or decimal notation:
  - E.g.: $Sum = m_1 + m_2 + m_4 + m_7 = \sum m(1,2,4,7)$

    $C_{out} = m_3 + m_5 + m_6 + m_7 = \sum m(3,5,6,7)$

Product of Maxterms

- Product of maxterms
  - E.g.: $C_{out} = (X + Y + C_{in}) \bullet (X + Y + C_{in}') \bullet (X + Y' + C_{in}) \bullet (X' + Y + C_{in})$

X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- Maxterm expansion in M-notation or decimal notation:
  - E.g.: $C_{out} = M_0 \bullet M_1 \bullet M_2 \bullet M_4 = \Pi M(0, 1, 2, 4)$

J.J. Shann 1-9

9

Standard Forms

- Two standard forms:
  - ***Sum of Products*** (***AND terms***)
  - ***Product of Sums*** (***OR terms***)

J.J. Shann 1-10

10

1-2

Boolean Algebra and Algebraic Simplification

J.J. Shann

▪11

Laws and Theorems of Boolean Algebra (1/4)

- Operations with 0 and 1:

  $X + 0 = X \qquad X \cdot 1 = X$

  $X + 1 = 1 \qquad X \cdot 0 = 0$

- Idempotent laws

  $X + X = X \qquad X \cdot X = X$

- Involution law:

  $(X')' = X$

- Laws of complementarity:

  $X + X' = 1 \qquad X \cdot X' = 0$

J.J. Shann 1-12

▪12

Laws and Theorems of Boolean Algebra (2/4)

- Commutative laws:

  $X + Y = Y + X \qquad X\,Y = Y\,X$

- Associative laws:

  $(X + Y) + Z = X + (Y + Z) = X + Y + Z$

  $(XY)Z = X(YZ) = XYZ$

- Distributive laws:

  $X(Y + Z) = XY + XZ$

  $X + YZ = (X + Y)(X + Z)$

J.J. Shann 1-13

Laws and Theorems of Boolean Algebra (3/4)

- Simplification theorems:

  $X\,Y + X\,Y' = X \qquad (X + Y)(X + Y') = X$

  $X + XY = X \qquad X(X + Y) = X$

  $(X + Y')\,Y = XY \qquad XY' + Y = X + Y$

- DeMorgan's laws:

  $(X + Y + Z + \ldots)' = X'Y'Z' \ldots$

  $(X\,Y\,Z \ldots)' = X' + Y' + Z' + \ldots$

  $[f(X_1, X_2, \ldots X_n, 0, 1, +, \bullet)]'$

  $= f(X_1', X_2', \ldots X_n', 1, 0, \bullet, +)$

* Add parenthesis to ensure proper order of operations.

J.J. Shann 1-14

DeMorgan's Law

$(X + Y + Z + \ldots)' = X'Y'Z' \ldots$
$(X\,Y\,Z \ldots)' = X' + Y' + Z' + \ldots$

- DeMorgan's Law:
  - Complement all the terms in the expression:
    - Replace each variable by its complement.
    - Switch 1 with 0 and 0 with 1.
    - Change all ANDs to ORs and all ORs to ANDs.
  - Add parenthesis to ensure proper order of operations.
    - If AND is performed before OR in *F*, then parenthesis may be required to ensure that OR is performed before AND in F'.

J.J. Shann 1-15

Laws and Theorems of Boolean Algebra (4/4)

- Duality:

  $(X + Y + Z + \ldots)^D = X\,Y\,Z \ldots$
  $(X\,Y\,Z\ldots)^D = X + Y + Z + \ldots$
  $[\,f\,(X_1, X_2, \ldots X_n, 0, 1, +, \bullet\,)\,]^D$
  $= f\,(X_1, X_2, \ldots X_n, 1, 0, \bullet, +\,)$
- Theorem for multiplying out and factoring:

  $(X + Y)\,(X' + Z) = X\,Z + X'\,Y$
  $XY + X'Z = (X + Z)\,(X' + Y)$
- Consensus theorem:

  $XY + YZ + X'Z = XY + X'Z$
  $(X + Y)(Y + Z)(X' + Z) = (X + Y)(X' + Z)$

J.J. Shann 1-16

Algebraic Simplification

- Four ways of simplifying a logic expression:
  - *Combining terms*: **XY + XY′ = X**
  - *Eliminating (redundant) terms*:
    - i. **X + XY = X**
    - ii. **XY + X′Z + YZ = XY +X′Z** (consensus theorem)
  - *Eliminating (redundant) literals*:
    - **X + X′Y = X + Y**
  - *Adding redundant terms*:
    - i. **adding XX′**
    - ii. **multiplying by (X + X′)**
    - iii. applying consensus theorem **XY +X′Z = XY + X′Z + YZ**

J.J. Shann 1-17

▪17

1-3 ★

Karnaugh Maps

J.J. Shann

▪18

Karnaugh Maps

- (K-maps) provide a convenient way to simplify logic functions of three to four variables.
- For a five-variable function, two four-variable Karnaugh maps can be used to simplify the function.

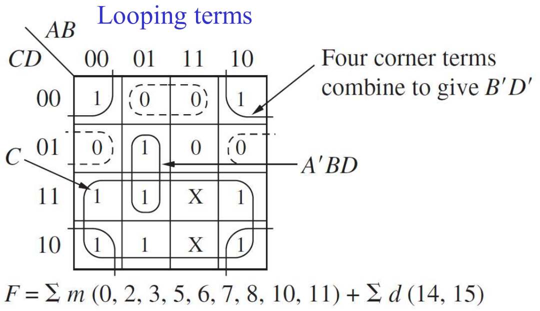
Example: 4-Variable K-map

- 4 variable K-map: $F(A, B, C, D)$

Location of minterms

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

Looping terms



CD \ AB	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	1	1	X	1
10	1	1	X	1

$$F = \Sigma\, m\,(0, 2, 3, 5, 6, 7, 8, 10, 11) + \Sigma\, d\,(14, 15)$$
$$= C + B'D' + A'BD$$

- Procedure to obtain a minimum sum of products from a Karnaugh map:
  1. Choose a minterm (a 1) that has not yet been covered.
  2. Find all 1s and Xs adjacent to that minterm. ⇒ Prime implicant
  3. If a single term covers the minterm and all the adjacent 1s and Xs, then that term is an essential prime implicant, so select that term.
  4. Repeat steps 1, 2, and 3 until all essential prime implicants have been chosen.
  5. Find a minimum set of prime implicants that cover the remaining 1s on the map.

J.J. Shann 1-21

K-Map with Map-Entered Variables

- For a function has more than 5 variables, Karnaugh map using ***map-entered variables*** can be used for simplification.
  - ***Map-entered variable***: A variable P_i is placed in square m_j of a map of function F

    ⇒ $F = 1$ when $P_i = 1$ and the variables are chosen so that $m_j = 1$

J.J. Shann 1-22

Example: 6-Input Function

- Consider a truth table w/ six input variables and one output variable:
  - Only certain rows of the truth table have been specified: *partial truth table*
  - The input combinations not specified in the truth table result in an output of 0.

A	B	C	D	E	F	G
0	0	0	0	X	X	1
0	0	0	1	X	X	X
0	0	1	0	X	X	1
0	0	1	1	X	X	1
0	1	0	1	1	X	1
0	1	1	1	1	X	1
1	0	0	1	X	1	1
1	0	1	0	X	X	X
1	0	1	1	X	X	1
1	1	0	1	X	X	X
1	1	1	1	X	X	1

$$G(A, B, C, D, E, F) = m_0 + m_2 + m_3 + Em_5 + Em_7 + Fm_9 + m_{11} + m_{15} + d_1 + d_{10} + d_{13}$$

<Ans.>

$$G(A, B, C, D, E, F) = m_0 + m_2 + m_3 + Em_5 + Em_7 + Fm_9 + m_{11} + m_{15} + d_1 + d_{10} + d_{13}$$

- Since E and F are the input variables w/ the greatest # of don’t cares (X), a Karnaugh map can be formed w/ A, B, C, D, and the remaining two variables can be entered inside.

A	B	C	D	E	F	G
0	0	0	0	X	X	1
0	0	0	1	X	X	X
0	0	1	0	X	X	1
0	0	1	1	X	X	1
0	1	0	1	1	X	1
0	1	1	1	1	X	1
1	0	0	1	X	1	1
1	0	1	0	X	X	X
1	0	1	1	X	X	1
1	1	0	1	X	X	X
1	1	1	1	X	X	1

CD \ *AB*	00	01	11	10
00	1			
01	X	*E*	X	*F*
11	1	*E*	1	1
10	1			X

G

Simplification Using Map-Entered Variables

- General method of simplifying a K-map w/ map-entered variables P_i s :
  - Given a map with variables $P_1, P_2, \ldots$ entered into some of the squares, the (minimum) sum-of-products form of F:

$$F = MS_0 + P_1 MS_1 + P_2 MS_2 + \ldots$$

    - MS_0 : the minimum sum obtained by setting $P_i = 0$ $(i = 1, 2, \ldots)$
    - MS_i : the minimum sum obtained by setting $P_i = 1$, $P_j = 0$, and replacing all 1s on the map with don't cares. $(i = 1, 2, \ldots)$ $(j \neq i)$

J.J. Shann 1-26

26

- The resulting expression for F will always be a correct representation of F.
- This expression will be a minimum sum provided that the values of the map-entered variables can be assigned independently.
- On the other hand, the expression will not generally be a minimum sum if the variables are not independent (e.g., if $P_1 = P_2'$).

J.J. Shann 1-27

27

Example

- Simplify the following function, G, by using K-map with map-entered variables:

G(A, B, C, D, E, F)

$$= m_0 + m_2 + m_3 + m_{11} + m_{15} + Em_5 + Em_7 + Fm_9 + d_1 + d_{10} + d_{13}$$

CD \ AB	00	01	11	10
00	1			
01	X	*E*	X	*F*
11	1	*E*	1	1
10	1			X

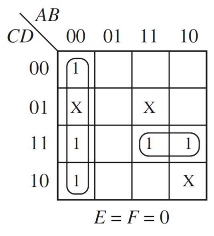
G

<Ans.>

CD \ AB	00	01	11	10
00	1			
01	X	*E*	X	*F*
11	1	*E*	1	1
10	1			X

G

* Write the minimum expression directly from the original map w/o first plotting individual maps for each of the minimum sums.



CD \ AB	00	01	11	10
00	1			
01	X		X	
11	1		1	1
10	1			X

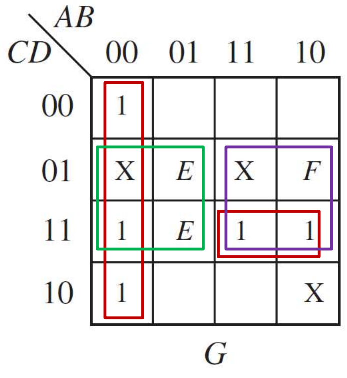
E = *F* = 0

MS_0 = A′B′ + ACD MS_1 = A′D MS_2 = AD

⇒ G = A′B′ + ACD + EA′D + FAD

<Ans.>

* Write the minimum expression directly from the original map w/o first plotting individual maps for each of the minimum sums.



$MS_0 = A'B' + ACD$ $MS_1 = A'D$ $MS_2 = AD$

$\Rightarrow G = A'B' + ACD + EA'D + FAD$

Exercise of §1-3

For the following function, find the minimum sum of products using 4-variable maps with map-entered variables. In the following function, m_i represents a minterm of variables A, B, C, and D.

$$Z(A, B, C, D, E, F, G)$$
$$= \Sigma m(2, 5, 6, 9) + \Sigma d(1, 3, 4, 13, 14) + E(m_{11} + m_{12}) + F(m_{10}) + G(m_0)$$

1-4

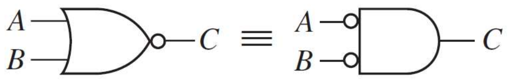
Designing with NAND and NOR Gates

J.J. Shann

■32

Designing with NOR Gates

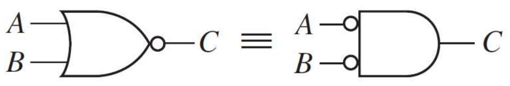
- Design a circuit of NOR gates:



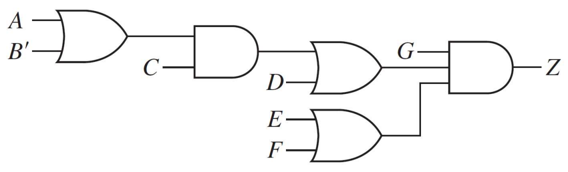
  - Get a product-of-sums (*PoS*) representation of the function. (2-level OR-AND ckt)
  - Find a circuit of OR and AND gates that has an AND gate at the output.
  - If AND gate output doesn't drive an AND gate input and an OR gate output doesn't connect to an OR gate input, conversion is done by replacing all gates with NOR gates and complementing inputs if needed.

J.J. Shann 1-33

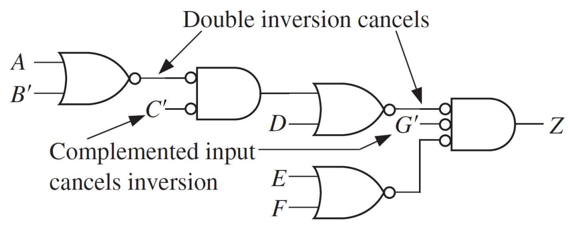
■33



Example: Conversion to NOR gates



(a) AND-OR circuit

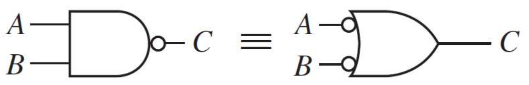


(b) Equivalent NOR-gate circuit

J.J. Shann 1-34

Designing with NAND Gates

- Conversion to a circuit of NAND gates:



  - Starting point is sum-of-products (*SoP*). (2-level AND-OR ckt)
  - Output gate of the AND-OR circuit should be an OR gate.

J.J. Shann 1-35

1-5 ★

Hazards in Combinational Circuits

J.J. Shann

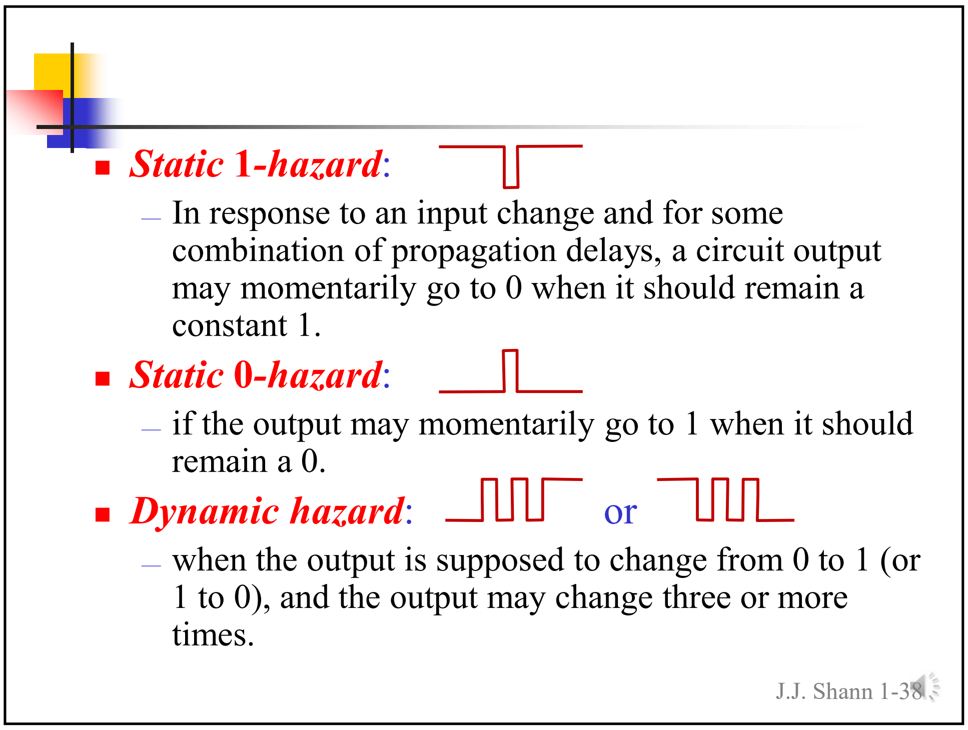
▪36

Hazards in Combinational Circuits

- When the input to a combinational circuit changes, unwanted ***switching transients*** may appear in the output.
  - These transients occur when different *paths* from input to output have different propagation delays.
- Three types of hazards in combinational ckts:
  - ***Static 1-hazard***
  - ***Static 0-hazard***
  - ***Dynamic hazard***

J.J. Shann 1-37

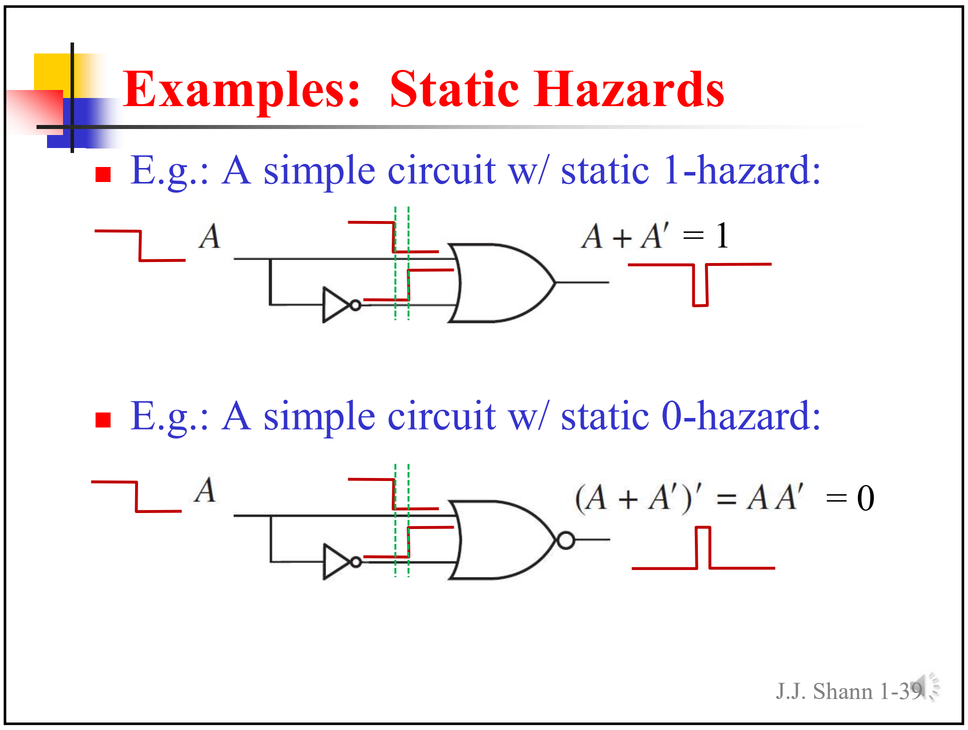
▪37



- ***Static 1-hazard***:
  - In response to an input change and for some combination of propagation delays, a circuit output may momentarily go to 0 when it should remain a constant 1.
- ***Static 0-hazard***:
  - if the output may momentarily go to 1 when it should remain a 0.
- ***Dynamic hazard***: or
  - when the output is supposed to change from 0 to 1 (or 1 to 0), and the output may change three or more times.

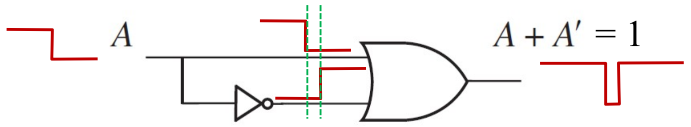
J.J. Shann 1-38

38

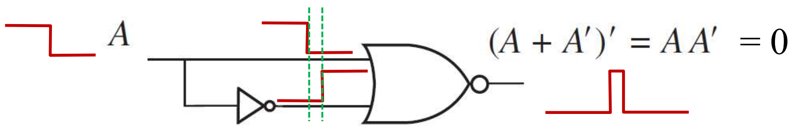


Examples: Static Hazards

- E.g.: A simple circuit w/ static 1-hazard:



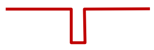
- E.g.: A simple circuit w/ static 0-hazard:



J.J. Shann 1-39

39

Static 1-Hazard

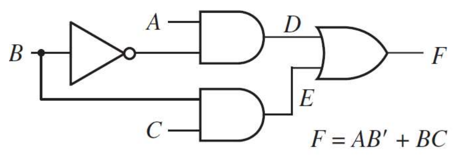
- Static 1-hazard: 
  - occurs in a sum-of-product implementation when two minterms differing by only one input variable are not covered by the same product term.
  - E.g.: (next page)

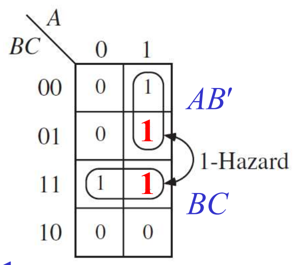
J.J. Shann 1-40

■40

Example: Static 1-Hazard

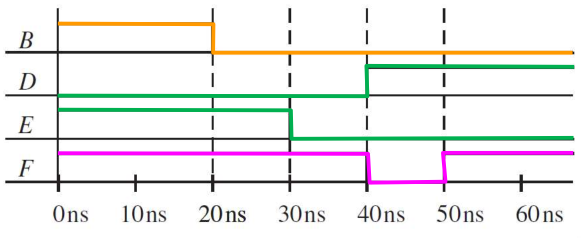
- Circuit with 1-hazard:





- Timing chart: A = 1, C = 1

* Assumption:
propagation delay of each gate = 10 ns



J.J. Shann 1-41

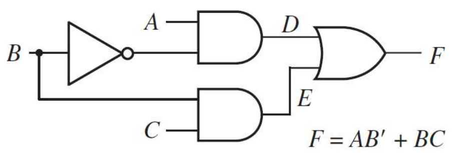
■41

Identification of Static Hazards

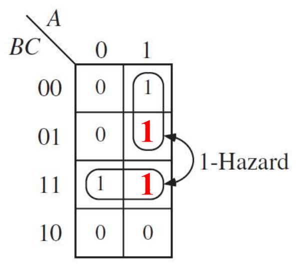
- Identifying static hazards in a given circuit:
  - ***Static-1 hazards***:
    - Write an expression for the output in terms of the inputs exactly as it is implemented in the circuit & manipulate it to a ***sum-of-products*** form, treating x_i and x_i' as independent variables.
    - An Karnaugh map can be constructed, and all implicants corresponding to each term can be circled.
    - If any pair of adjacent 1's is not covered by a single term, a static 1-hazard can occur.
  - ***Static-0 hazards***:
    - Can be identified by writing a ***product-of-sums*** expression for the circuit.

Example: Eliminating of Static 1-Hazard

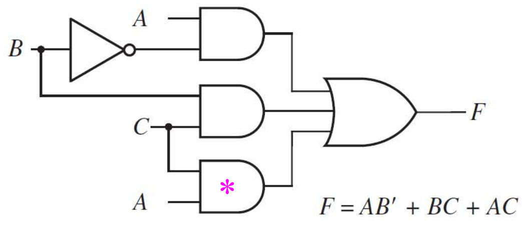
- Circuit with 1-hazard:



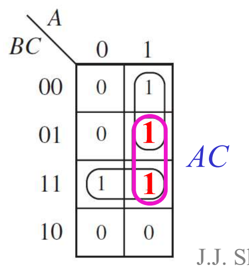
$F = AB' + BC$



<Ans.> Circuit w/ hazard removed:

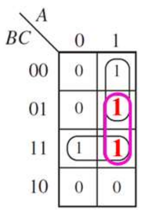


$F = AB' + BC + AC$



Design of Hazard-Free Combinational Circuits

- Hazard-free: no static and dynamic hazards
- Method 1: *minterms* (1's)
  - Find a ***sum-of-products*** (***SoP***) expression (F^t) for the output in which every pair of adjacent 1s is covered by a 1-term, i.e., an AND-term.
    - The sum of all prime implicants will always satisfy this condition.
    - A two-level AND-OR circuit based on this F^t will be free of 1-, 0-, and dynamic hazards.
  - If a different form of circuit is desired, manipulate F^t to the desired form by using simple *factoring*, *DeMorgan's law*, and so on. Treat each X_i and X_i' as independent variable to prevent introduction of hazards.



J.J. Shann 1-44

- Method 2: *maxterms* (0's)
  - Find a ***product-of-sums*** (***PoS***) expression (F^t) for the output in which every pair of adjacent 0s is covered by a 0-term, i.e., an OR-term.
    - A two-level OR-AND circuit based on this F^t will be free of 1-, 0-, and dynamic hazards.
  - If a different form of circuit is desired, manipulate F^t to the desired form by using simple factoring, DeMorgan's law, and so on. Treat each X_i and X_i' as independent variable to prevent introduction of hazards.

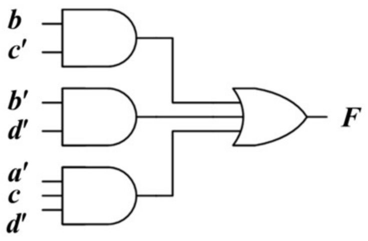
J.J. Shann 1-45

Exercise of §1-5

For the circuit given below,

(a) Find all of the ***static 1-hazards*** in the circuit.

(b) Indicate which changes are necessary to eliminate the hazards, and derive the revised equation of F.



J.J. Shann 1-46

46

1-6

Flip-Flops and Latches

J.J. Shann

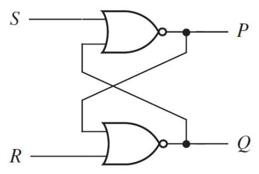
47

Flip-Flops and Latches

- Commonly-used storage devices for sequential circuits:
  - Flip-flops: for synchronous sequential circuits
  - Latches: for asynchronous sequential circuits

Latches

- Some types of latches:
  - S-R latch

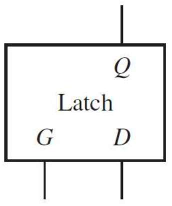


S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

Example: Transparent D Latch

- Transparent D latch:

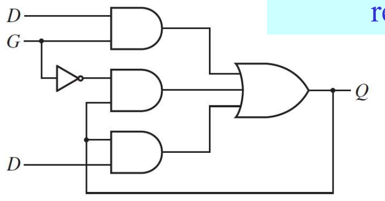
* G: gate signal



G	D	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$Q^+ = DG + G'Q + (DQ)$

* DQ: a redundant term for removing hazard.

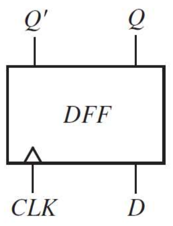


J.J. Shann 1-50

50

Flip-Flops (1/2)

- Types of flip-flops:
  - Delay (D) flip-flops, J-K flip-flops, Toggle (T) flip-flops
- Clocked D Flip-Flop: $Q^+ = D$



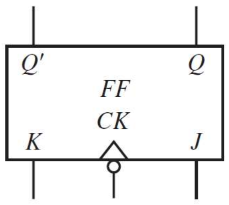
D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

J.J. Shann 1-51

51

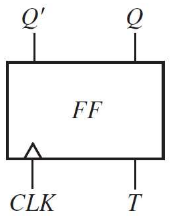
Flip-Flops (2/2)

- Clocked J-K Flip-Flop: $Q^+ = JQ' + K'Q$



J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

- Clocked T Flip-Flop: $Q^+ = Q \oplus T$



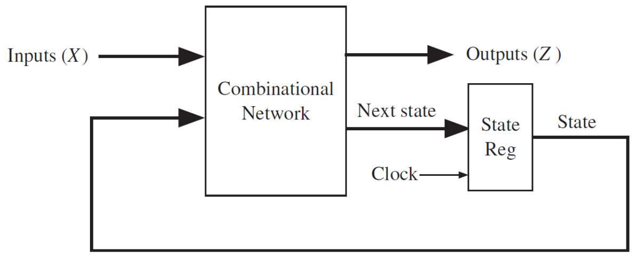
T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

1-7 & 1-8 & 1-9

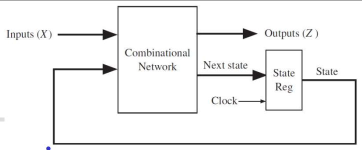
Design of (Synchronous) Sequential Circuit

Sequential Circuit

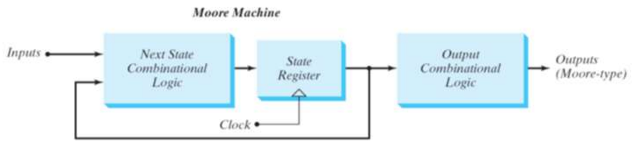
- Consists of:
  - combinational circuit: generates the outputs and the next state
  - a state register: holds the present state.
    - State register is usually comprised of D flip-flops.



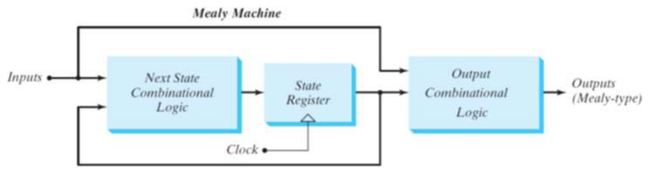
J.J. Shann 1-54



- Two types of sequential circuits:
  - ***Moore***: the outputs depend only on the present state



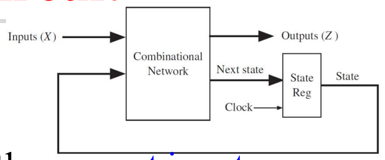
  - ***Mealy***: the outputs depend on both the present state and the present inputs



J.J. Shann 1-55

Mealy Sequential Circuit

- Mealy sequential circuit:
  - Outputs depend only on both the present state and the present inputs.



  - The normal sequence of events:
    1. The X inputs change to a new value.
    2. After a delay, the corresponding Z outputs and next state appear at the output of the combinational circuit.
    3. The next state is clocked into the state register and the state changes.
    4. The new state feeds back into the combinational ckt, and the process is repeated.

Moore Sequential Circuit

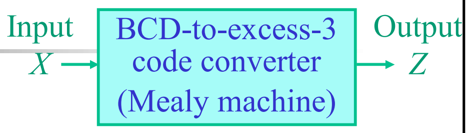
- Moore sequential circuit:
  - Outputs depend only on the present state.
  - The outputs are associated entirely to the state.
    - ⇒ No outputs occur during the transition.
    - ⇒ Cannot respond to an input until the active edge of the clock occurs; this is in contrast to a Mealy circuit.
  - Pros: easier to design and debug than Mealy machines
  - Cons: often contain more states than equivalent Mealy machines

Sequential Circuit Design

- Steps required to design a sequential circuit:
  1. (Specification) Determine the required relationship b/t the input and output sequences.
  2. (Formulation) Find a ***state graph*** and ***state table***.
  3. Reduce the table to a minimum number of states.
  4. If the reduced table has m states ($2^{n-1} < m \leq 2^n$) , n flip-flops are needed at least. Use either the ***encoded state assignment*** technique or the ***one-hot assignment*** technique.
  5. Form the ***transition table***.
  6. Plot next-state maps and input maps for each flip-flop and derive the ***flip-flop input equations***. Derive the ***output functions***.
  7. Realize the flip-flop input equations and the output equations using the available logic gates.
  8. Check design.

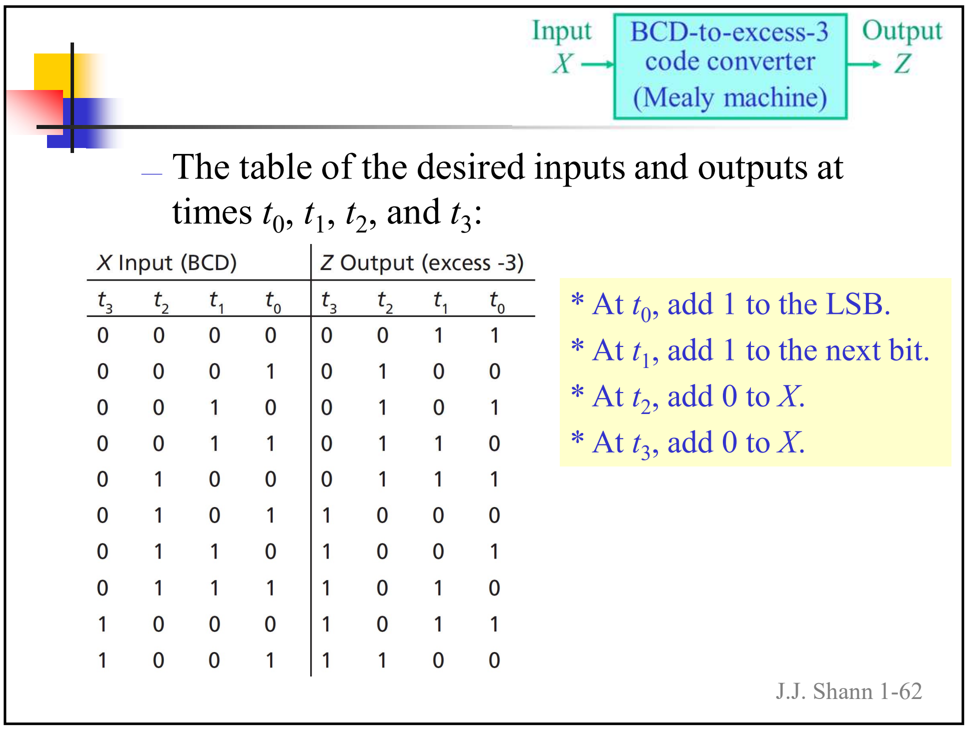
J.J. Shann 1-58

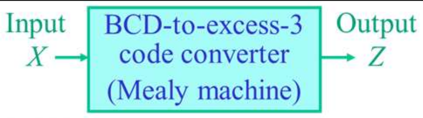
Example: BCD to Excess-3 Code Converter



- Problem description:
  - Design a ***Mealy-type*** serial code converter that converts an 8-4-2-1 binary-coded-decimal (BCD) digit to an excess-3-coded decimal digit.
  - The input (X) will arrive serially w/ the least significant bit first.
  - The outputs will be generated serially as well.
  - After receiving four inputs, the circuit should reset to its initial state, ready to receive another BCD digit.

J.J. Shann 1-61



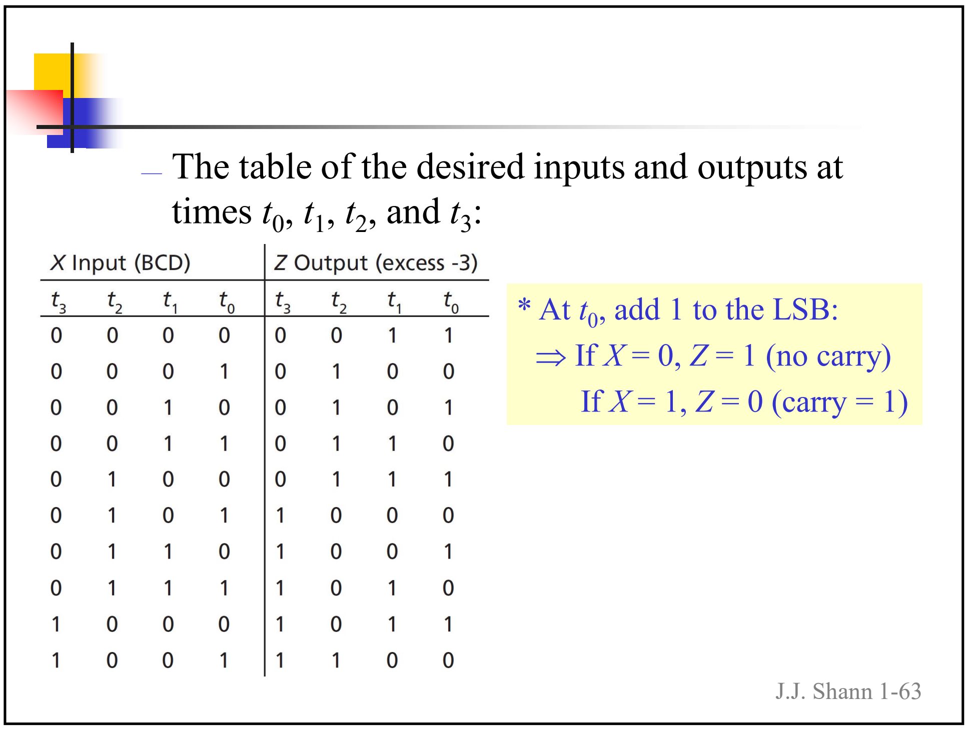


– The table of the desired inputs and outputs at times t_0, t_1, t_2, and t_3:

X Input (BCD)				*Z* Output (excess -3)			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

* At t_0, add 1 to the LSB.
* At t_1, add 1 to the next bit.
* At t_2, add 0 to X.
* At t_3, add 0 to X.

▪62

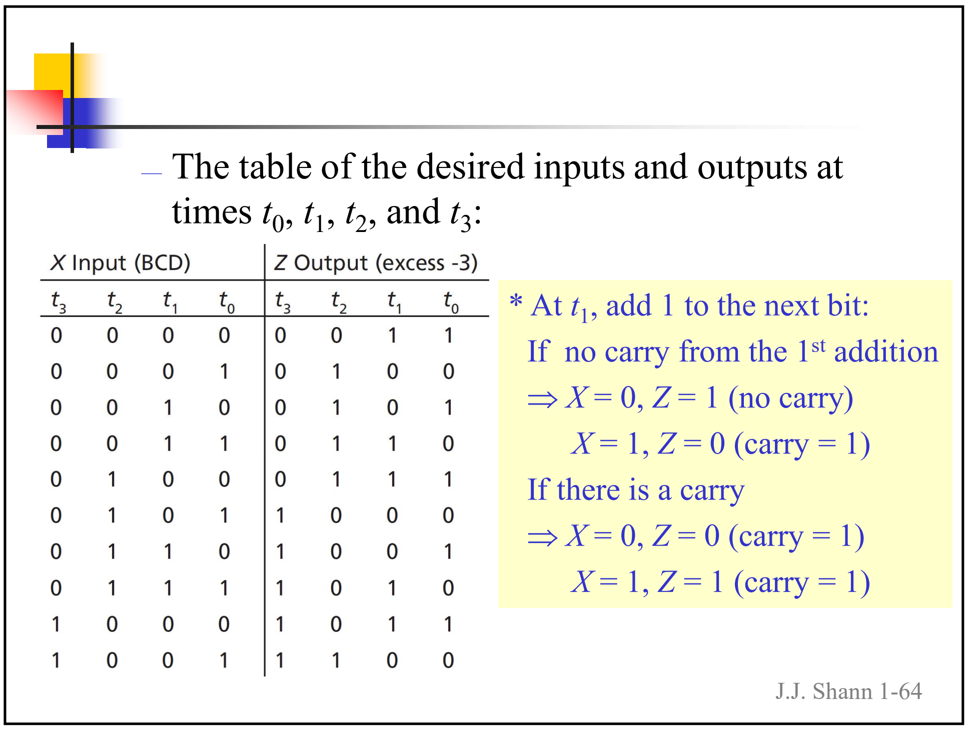


– The table of the desired inputs and outputs at times t_0, t_1, t_2, and t_3:

X Input (BCD)				*Z* Output (excess -3)			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

* At t_0, add 1 to the LSB:
$\Rightarrow$ If $X = 0$, $Z = 1$ (no carry)
If $X = 1$, $Z = 0$ (carry = 1)

▪63



– The table of the desired inputs and outputs at times t_0, t_1, t_2, and t_3:

X Input (BCD)				*Z* Output (excess -3)			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

* At t_1, add 1 to the next bit:

If no carry from the 1st addition

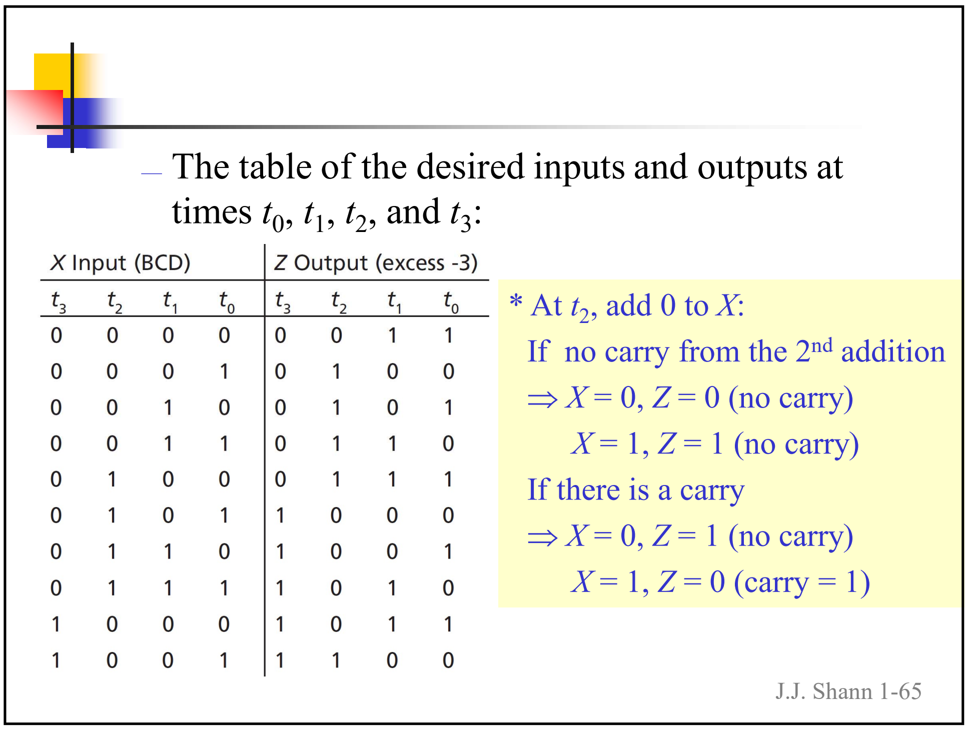
$\Rightarrow X = 0, Z = 1$ (no carry)

$X = 1, Z = 0$ (carry = 1)

If there is a carry

$\Rightarrow X = 0, Z = 0$ (carry = 1)

$X = 1, Z = 1$ (carry = 1)



– The table of the desired inputs and outputs at times t_0, t_1, t_2, and t_3:

X Input (BCD)				*Z* Output (excess -3)			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

* At t_2, add 0 to *X*:

If no carry from the 2nd addition

$\Rightarrow X = 0, Z = 0$ (no carry)

$X = 1, Z = 1$ (no carry)

If there is a carry

$\Rightarrow X = 0, Z = 1$ (no carry)

$X = 1, Z = 0$ (carry = 1)

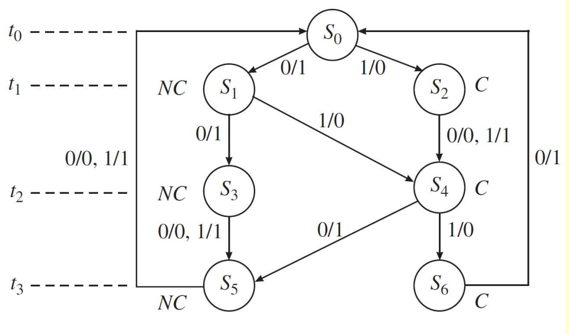
– The table of the desired inputs and outputs at times t_0, t_1, t_2, and t_3:

X Input (BCD)				*Z* Output (excess -3)			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

* At t_3, add 0 to X:
If no carry from the 3rd addition
$\Rightarrow X = 0, Z = 0$ (reset)
$X = 1, Z = 1$ (reset)
If there is a carry
$\Rightarrow X = 0, Z = 1$ (reset)
($X = 1$ will not occur)

<Ans.> Mealy

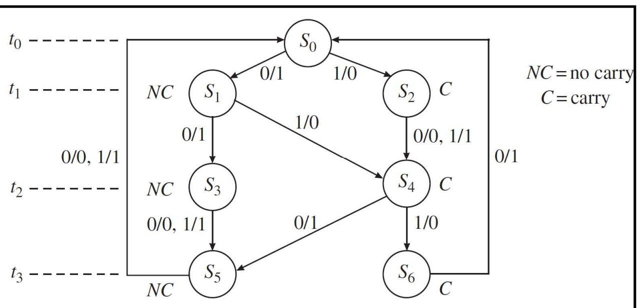
- State graph:



NC = no carry
C = carry

* Add 1 to the LSB (t_0)
$\Rightarrow$ If X = 0, Z = 1 (no carry)
If X = 1, Z = 0 (carry = 1)
* Add 1 to the next bit: (t_1)
If no carry from the 1st addition
$\Rightarrow$ X = 0, Z = 1 (no carry)
X = 1, Z = 0 (carry = 1)
If there is a carry from the 1st addition,
$\Rightarrow$ X = 0, Z = 0 (carry = 1)
X = 1, Z = 1 (carry = 1)
* At t_2, add 0 to X.
If no carry from the 2nd addition
$\Rightarrow X = 0, Z = 0$ (no carry)
$X = 1, Z = 1$ (no carry)
If there is a carry from the 2nd addition
$\Rightarrow X = 0, Z = 1$ (no carry)
$X = 1, Z = 0$ (carry = 1)
* At t_3, add 0 to X.
If no carry from the 3rd addition
$\Rightarrow X = 0, Z = 0$ (reset)
$X = 1, Z = 1$ (reset)
If there is a carry from the 3rd addition
$\Rightarrow X = 0, Z = 1$ (reset)
($X = 1$ will not occur in this case)

State table:



PS	NS X=0	NS X=1	Z X=0	Z X=1
S_0	S_1	S_2	1	0
S_1	S_3	S_4	1	0
S_2	S_4	S_4	0	1
S_3	S_5	S_5	0	1
S_4	S_5	S_6	1	0
S_5	S_0	S_0	0	1
S_6	S_0	–	1	–

J.J. Shann 1-68

68

State assignment:

PS	NS X=0	NS X=1	Z X=0	Z X=1
S_0	S_1	S_2	1	0
S_1	S_3	S_4	1	0
S_2	S_4	S_4	0	1
S_3	S_5	S_5	0	1
S_4	S_5	S_6	1	0
S_5	S_0	S_0	0	1
S_6	S_0	–	1	–

Q_2Q_3 \ Q_1	0	1
00	S_0	S_1
01		S_2
11	S_5	S_3
10	S_6	S_4

(a) Assignment map

$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$ X=0	$Q_1^+Q_2^+Q_3^+$ X=1	Z X=0	Z X=1
000	100	101	1	0
100	111	110	1	0
101	110	110	0	1
111	011	011	0	1
110	011	010	1	0
011	000	000	0	1
010	000	xxx	1	x
001	xxx	xxx	x	x

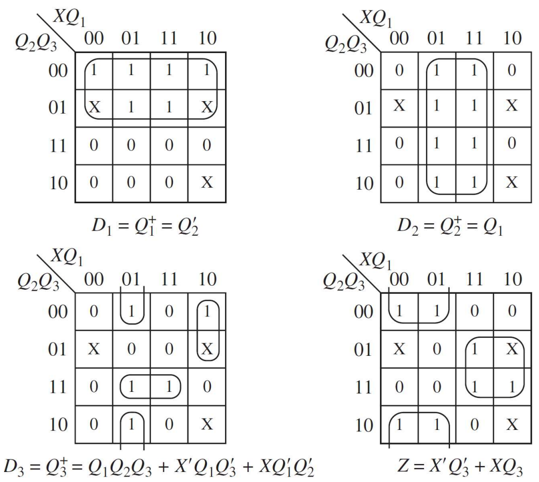
(b) Transition table

J.J. Shann 1-69

69

$Q_1Q_2Q_3$	$Q_1^+ Q_2^+ Q_3^+$ $X=0$	$X=1$	Z $X=0$	$X=1$
000	100	101	1	0
100	111	110	1	0
101	110	110	0	1
111	011	011	0	1
110	011	010	1	0
011	000	000	0	1
010	000	xxx	1	x
001	xxx	xxx	x	x

Nest state and output equations:



$D_1 = Q_1^+ = Q_2'$

$D_2 = Q_2^+ = Q_1$

$D_3 = Q_3^+ = Q_1Q_2Q_3 + X'Q_1Q_3' + XQ_1'Q_2'$

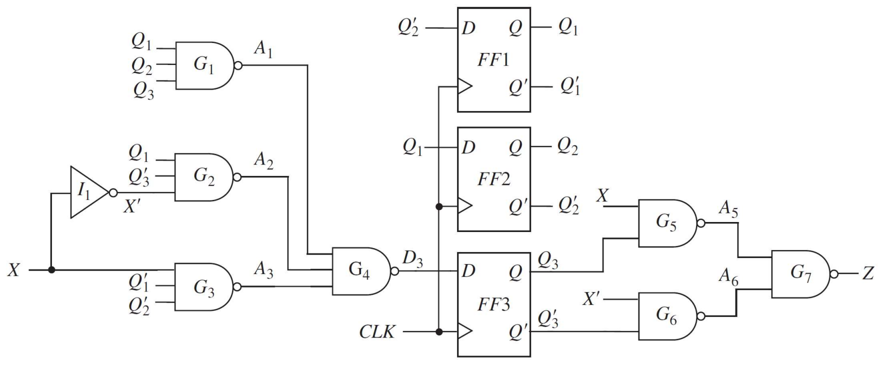
$Z = X'Q_3' + XQ_3$

* D flip-flops

J.J. Shann 1-70

Realization:

$D_1 = Q_2'$
$D_2 = Q_1$
$D_3 = Q_1Q_2Q_3 + X'Q_1Q_3' + XQ_1'Q_2'$
$Z = X'Q_3' + XQ_3$



J.J. Shann 1-71

1-10 ★

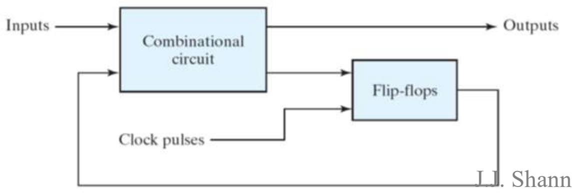
Sequential Circuit Timing

J.J. Shann

72

Analysis of a Digital Circuit

- Analysis of a digital ckt:
  - analyze the function of the ckt
  - analyze the performance/timing of the ckt
    - For comb. ckt: input-to-output delay
    - For sync seq ckt: the timing conditions for proper operation (e.g.: the min allowable clock period at which the ckt can operate ⇒ the max clock frequency, f_{max} , of the ckt)

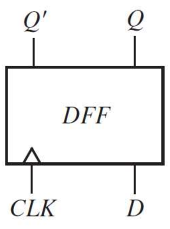


J.J. Shann 1-73

73

A. Timing Parameters of Flip-Flops

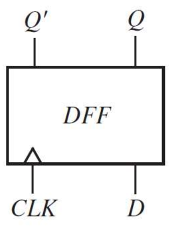
- **Propagation delay**: Clock-to-Q delay
  - small amount of time that elapses from the time the clock changes to the time the Q output changes.
- **Setup time** (t_{su}):
  - amount of time the D input is stable ***before*** the active edge of the clock.
- **Hold time** (t_h):
  - amount of time the D input is stable ***after*** the active edge of the clock.

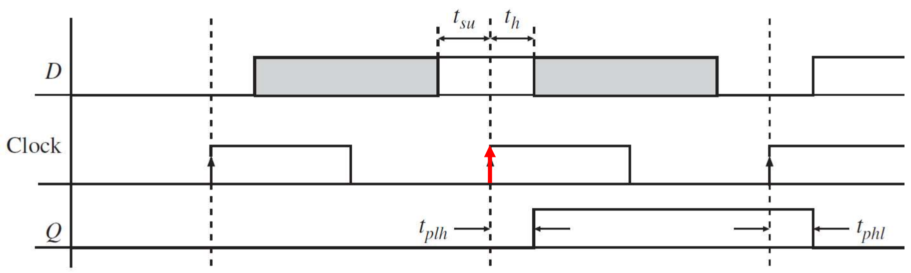


J.J. Shann 1-74

74

- Setup and hold times for D flip-flop:





J.J. Shann 1-75

75

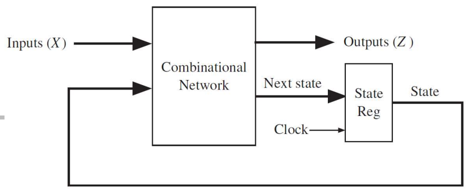
B. Timing Conditions for Proper Operation

- ***Maximum clock frequency*** for a sequential circuit:
  - ***Clock period*** must be long enough.
  - ***Propagation delays*** and ***setup*** and ***hold times*** create complications in timing.
- **Static timing analysis** (STA):
  - is a method of validating the timing performance of a design by checking all possible paths for timing violations under ***worst-case*** conditions

J.J. Shann 1-76

76

Timing Paths



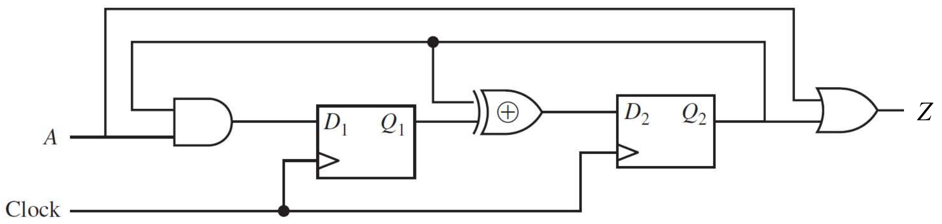
- 4 types of timing paths in a synchronous digital system:
  - i. ***Primary input to register paths***: *input to flip-flop*
  - ii. ***Primary Input to primary output paths***: *input to output* (no flip-flop)
  - iii. ***Register to register paths***: *flip-flop to flip-flop*
  - iv. ***Register to primary output paths***: *flip-flop to output*

J.J. Shann 1-77

77

Example

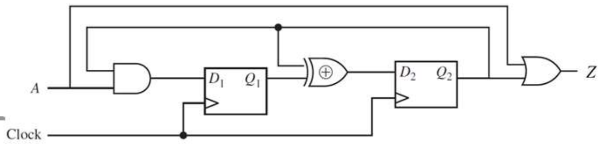
- Identify the static timing paths in the following circuit



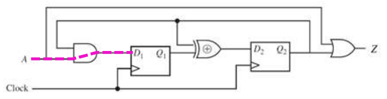
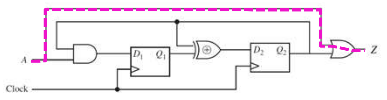
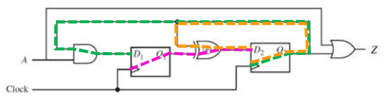
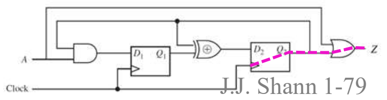
J.J. Shann 1-78

▪78

<Ans.>



- Static timing paths:
  - input to flip-flop:
    1. From A to D_1 via AND
  - input to output:
    2. From A to Z via OR
  - flip-flop to flip-flop:
    3. From D_1 to D_2 via XOR
    4. From D_2 to D_1 via AND
    5. From D_2 to D_2 via XOR
  - Flip-flop to output:
    6. From D_2 to Z via OR

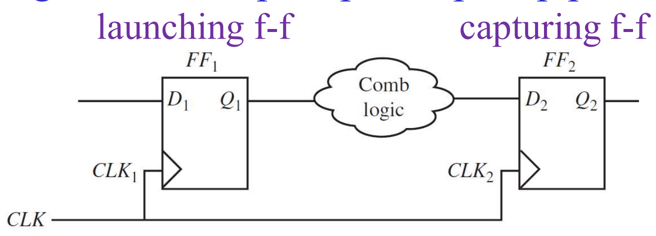





J.J. Shann 1-79

▪79

(a) Timing Rules for *Flip-Flop* to *Flip-Flop* Paths

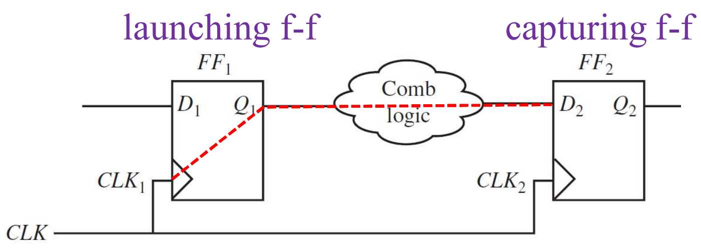
- Timing rules for flip-flop to flip-flop paths:



1. ***Setup time rule*** for flip-flop to flip-flop path: Clock period should be long enough to satisfy flip-flop setup time.
2. ***Hold-time rule*** for flip-flop to flip-flop path: Minimum circuit delays should be long enough to satisfy flip-flop hold time.

Rule #1: *Setup time rule* for *flip-flop* to *flip-flop* path

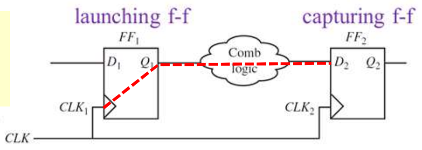
- ***Clock period*** should be long enough to satisfy flip-flop setup time: $t_{ck} \geq t_{pmax} + t_{cmax} + t_{su}$



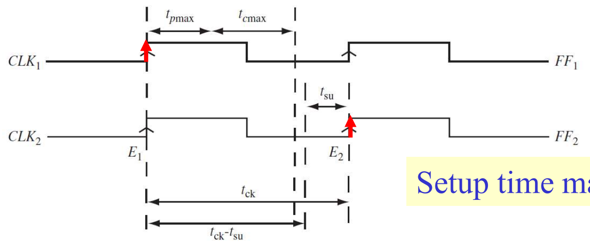
- Setup time margin: $t_{ck} - (t_{pmax} + t_{cmax} + t_{su})$
- Correction of a setup time violation: changing the clock frequency of the circuit

Setup time margin:

$t_{ck} - (t_{pmax} + t_{cmax} + t_{su})$

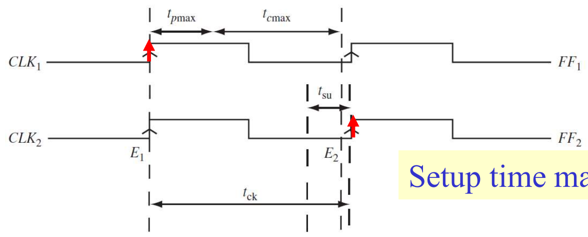


- Timing diagram when setup time is met:



Setup time margin ≥ 0

- Timing diagram when setup time is violated:



Setup time margin < 0

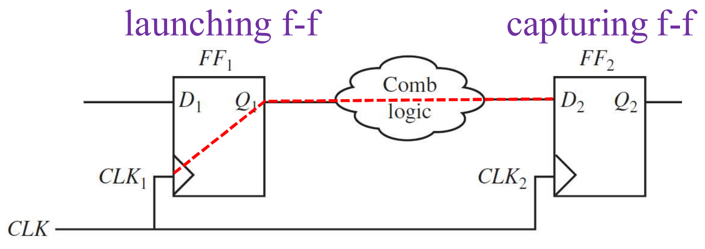
J.J. Shann 1-82

82

Rule #2: *Hold time rule* for *flip-flop* to *flip-flop* path

- ***Minimum circuit delays*** should be long enough to satisfy flip-flop hold time:

$t_{pmin} + t_{cmin} \geq t_h$



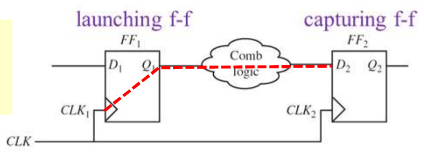
– Correction of a hold time violation: redesign the circuit to increase the combinational delays

J.J. Shann 1-83

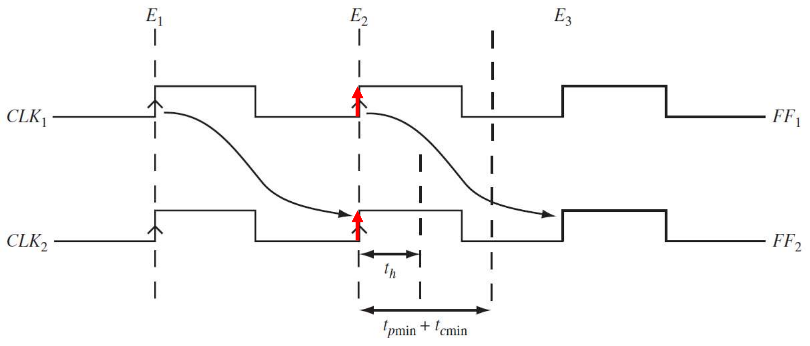
83

Hold time rule for f-f to f-f path:

$t_{pmin} + t_{cmin} \geq t_h$



- Timing diagrams illustrating hold time in flip-flop path:

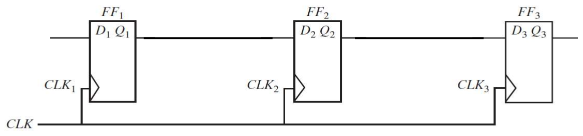


J.J. Shann 1-84

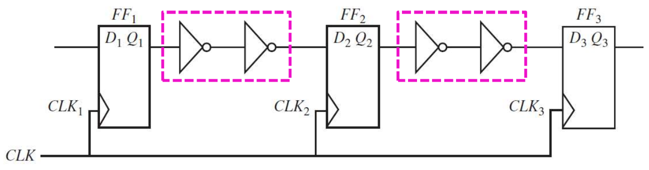
▪84

Example: Shift Register

- Designing shift registers by chaining together flip-flops:



  - Combinational ckt delay is zero.
    $\Rightarrow$ It is very difficult to meet hold-time constraints
  - For meeting hold-time constraints:

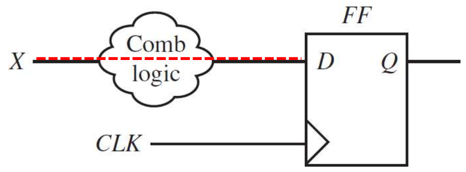


J.J. Shann 1-85

▪85

(b) Timing Rules for *Input* to *Flip-Flop* Paths

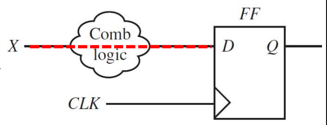
- Timing rules for input to flip-flop paths:

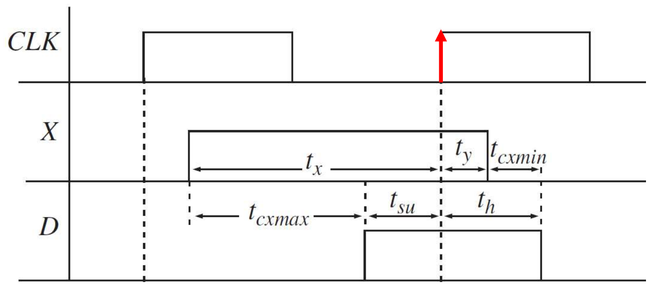


3. ***Setup time rule*** for input to flip-flop path: External input changes to the circuit should satisfy flip-flop setup time.
4. ***Hold-time rule*** for input to flip-flop path: External input changes to the circuit should satisfy flip-flop hold times.

Rule #3: *Setup time rule* for *input* to *flip-flop* path

- External input changes to the circuit should satisfy flip-flop setup time: $t_x \geq t_{cxmax} + t_{su}$
  - t_x : X changes at time t_x ***before*** the active edge of the clock

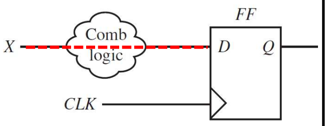


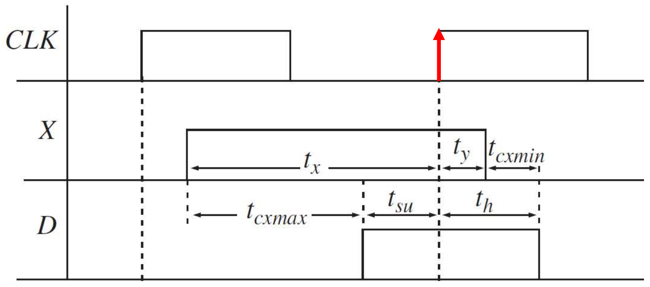


Rule #4: *Hold time rule* for *input* to *flip-flop* path

- External input changes to the satisfy flip-flop hold times :
  - t_y : X changes at time t_y ***after*** the active edge of the clock

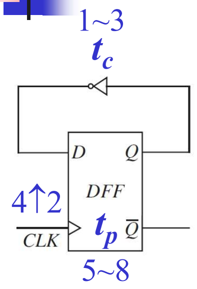
$$t_y + t_{cxmin} \geq t_h$$
$$\Rightarrow t_y \geq t_h - t_{cxmin}$$

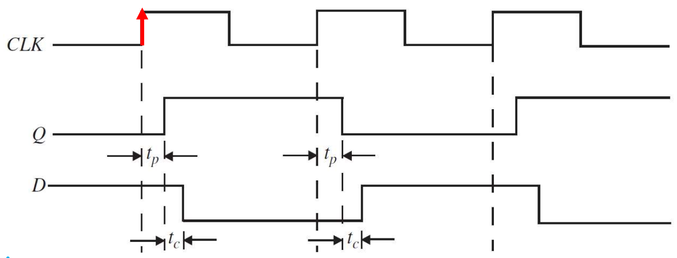




J.J. Shann 1-88

Example: Simple Frequency Divider (p.36)





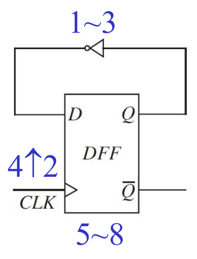
* No external input!

- Assumptions:
  - Inverter: $t_{invmin} = 1$ ns; $t_{invmax} = 3$ ns
  - Flip-flop: $t_{pmin} = 5$ ns; $t_{pmax} = 8$ ns;
    $t_{su} = 4$ ns; $t_h = 2$ ns

J.J. Shann 1-89

<Ans>

Inverter: $t_{invmin} = 1$ ns, $t_{invmax} = 3$ ns
Flip-flop: $t_{pmin} = 5$ ns, $t_{pmax} = 8$ ns,
$t_{su} = 4$ ns, $t_h = 2$ ns



* No external input

- The minimum clock period t_{ckmin}:

Rule#1 $t_{ckmin} = t_{pmax} + t_{cmax} + t_{su}$

$= 8 + 3 + 4 = 15$ ns

- The maximum clock frequency f_{max}:

$f_{max} = 1/t_{ckmin} = 66.67$ MHz

- Hold time requirement:
  - for filp-flip to flip-flop path: *Rule 2*

Rule#2 $t_{pmin} + t_{cmin} \geq t_h \Rightarrow 5 + 1 \geq 2$ (✓)

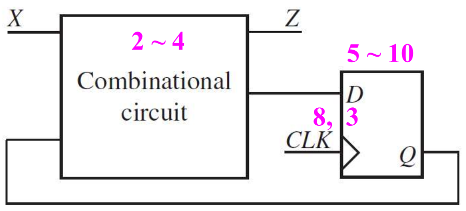
  - for input to flip-flop path: *Rule 4*

no external input

J.J. Shann 1-90

■90

Example (p.37)



Q	Q^+ (0)	Q^+ (1)	Z (0)	Z (1)
0	0	1	0	1
1	0	0	1	0

- Assumptions:
  - Delay of the combinational circuit: 2~4 ns
  - Flip-flop: t_p : 5~10 ns; $t_{su} = 8$ ns; $t_h = 3$ ns
- Check the timing rules for flip-flop to flip-flop paths and determine the safe region for input changes.

J.J. Shann 1-91

■91

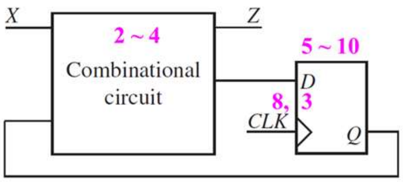
<Ans>

Delay of the combinational circuit: 2~4 ns
Flip-flop: t_p : 5~10 ns; t_{su} = 8 ns; t_h = 3 ns

- Setup time requirement for f-f to f-f path:

Rule#1 $t_{ck} \geq t_{pmax} + t_{cmax} + t_{su}$

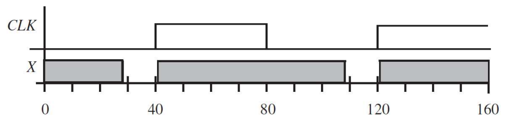
$t_{ck} \geq 10 + 4 + 8 = 22$ ns



- Hold time requirement for f-f to f-f path:

Rule#2 $t_{pmin} + t_{cmin} \geq t_h$

$5 + 2 = 7 \geq 3$ (✓)



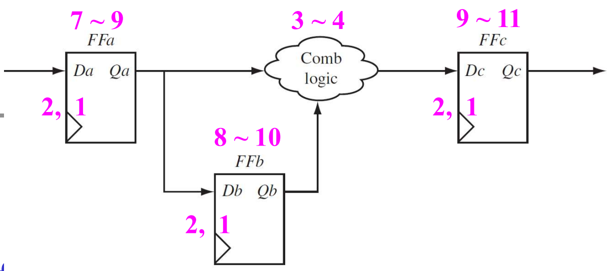
- Safe regions for changes in input X:
  - Rule 3: $t_x \geq t_{cxmax} + t_{su} \Rightarrow t_x \geq 4 + 8 = 12$
  - Rule 4: $t_y \geq t_h - t_{cxmin} \Rightarrow t_y \geq 3 - 2 = 1$

J.J. Shann 1-92

92

Example

- (p.38)

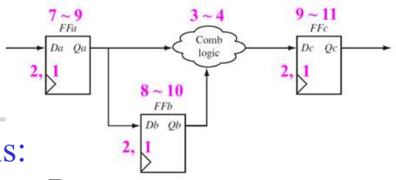


- Assumptions of the min/max delays:
  - CLK-to-Q for flip-flop A: 7 ns/9 ns
  - CLK-to-Q for flip-flop B: 8 ns/10 ns
  - CLK-to-Q for flip-flop C: 9 ns/11 ns
  - Setup time for flip-flops: 2 ns
  - Hold time for flip-flops: 1 ns
  - Combinational logic: 3 ns/4 ns
- Compute the max delays for all timing paths in this circuit and determine the maximum clock frequency allowed in this circuit.

J.J. Shann 1-93

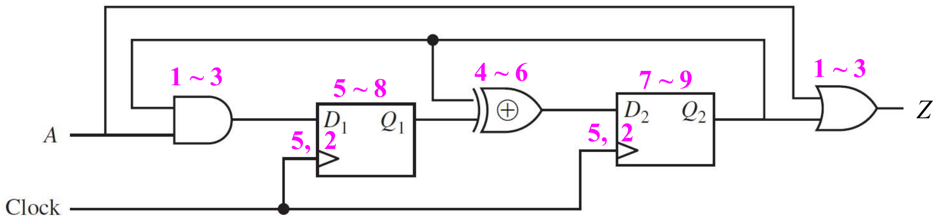
93

<Ans.>

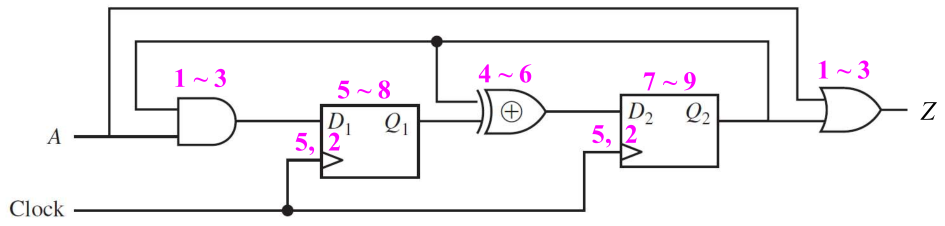


- Max delays for all timing paths:
  - Delay for path from flip-flop A to B:
    $t_{\text{clk-to-Q(A)}} + t_{\text{su(B)}} = 9 + 2 = 11$ ns
  - Delay for path from flip-flop A to C:
    $t_{\text{clk-to-Q(A)}} + t_{\text{comb}} + t_{\text{su(C)}} = 9 + 4 + 2 = 15$ ns
  - Delay for path from flip-flop B to C:
    $t_{\text{clk-to-Q(B)}} + t_{\text{comb}} + t_{\text{su(C)}} = 10 + 4 + 2 = 16$ ns
  - Delay for path from input to flip-flop A:
    $t_{\text{su(A)}} = 2$ ns
  - Delay for path from flip-flop C to output:
    $t_{\text{clk-to-Q(C)}} = 11$ ns
- Min clock period: $\Rightarrow$ 16 ns
- Max clock frequency:
  - 1/16 ns = 62.5 MHz

Exercise: Example (p.39)



- Assumptions of the min/max delays:
  - CLK-to-Q for flip-flop 1: 5 ns/8 ns
  - CLK-to-Q for flip-flop 2: 7 ns/9 ns
  - Setup time for flip-flops: 5 ns
  - Hold time for flip-flops: 2 ns
  - XOR gate: 4 ns/6 ns
  - AND/OR gate: 1 ns/3 ns



(a) What is the max clock frequency that this circuit can be safely clocked at?

(b) What is the latest time before the rising clock edge (t_x) that input A can safely change?

(c) What is the earliest time after the rising clock edge (t_y) that input A can safely change?

J.J. Shann 1-96

C. Times Conditions for Ckts with Clock Skew

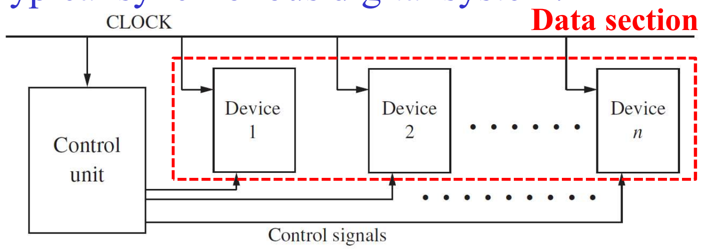
- Sync design vs. async design:
  - Sync circuits are more reliable than async circuits.
  - Sync design philosophy makes design and debugging easier as compared with async.
  - Sync designs consume more power than async designs ⇐ Power consumed in the *clock distribution network*.
  - Async designs can reduce power consumption, but it is very difficult to get timing issues under control.

  ⇒ Despite of their high power consumption, designers favor synchronous designs

J.J. Shann 1-98

Synchronous Digital System

- Typical synchronous digital system:

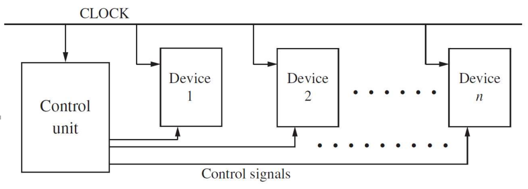


  - is built from several modules or devices
    - All the seq devices are ***synchronized*** w.r.t. the same clock.
  - ***Control section***: ***controller***
    - a seq machine that generates control signals to control the op of the data section
  - ***Data section***: architecture, ***data path***
    - May generate status signals that affect the control sequence

J.J. Shann 1-99

■99

Clock Skew



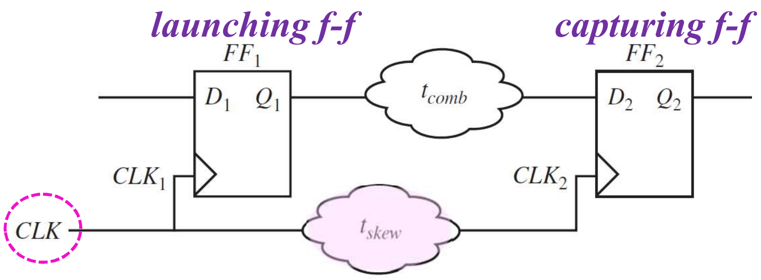
- Clock skew:
  - The absolute time difference in clock signal arrival b/t two points in the ***clock network***.
  - Causes:
    - The clock edge might arrive at two flip-flops at different times due to unequal wire delay.
    - Unequal amounts of combinational circuitry are used in the clock path to different devices
  - Categories:
    - ***Positive skew***
    - ***Negative shew***

J.J. Shann 1-100

■100

Positive Skew

- Positive skew:



  – The ***capturing flip-flop*** gets the clock delayed w/ reference to the ***launching flip-flop***.

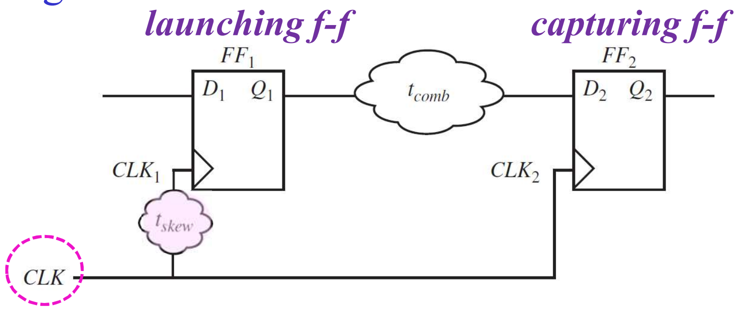
  * Positive skew is good for ***setup time***, but it is bad for ***hold time***.

J.J. Shann 1-101

■101

Negative Skew

- Negative skew:



  – The ***launching flip-flop*** gets the clock delayed w/ reference to the ***capturing flip-flop***.

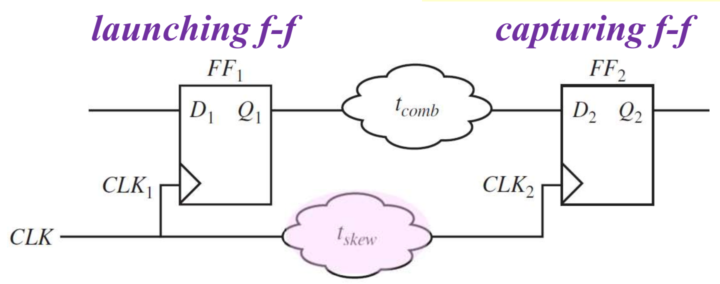
  * Negative skew is good for ***hold time***, but it is bad for ***setup time***.

J.J. Shann 1-102

■102

Timing Rules for Ckts with *Positive Skew*

* Positive skew is good for ***setup time***, but it is bad for ***hold time***.



- Timing rules for circuits with positive skew:

  5. modification of Rule#1 ($t_{ck} \geq t_{pmax} + t_{cmax} + t_{su}$)

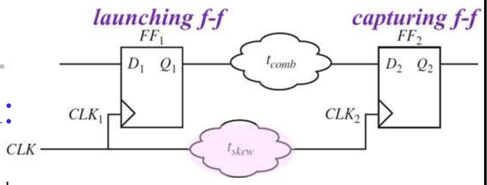
     $t_{ck} \geq t_{pmax} + t_{cmax} - t_{skew} + t_{su}$

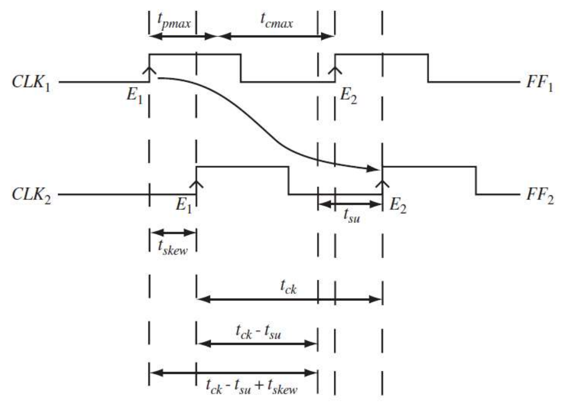
  6. modification of Rule#2 ($t_{pmin} + t_{cmin} \geq t_h$)

     $t_{pmin} + t_{cmin} \geq t_h + t_{skew}$

J.J. Shann 1-103

- Setup-time violation:

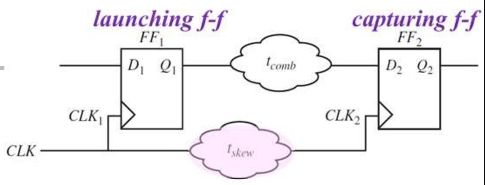


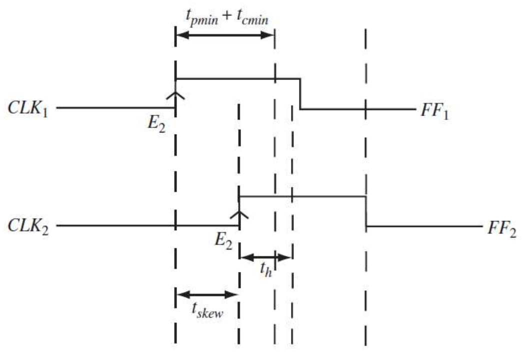


Condition $t_{pmax} + t_{cmax} \leq t_{ck} - t_{su} + t_{skew}$ not satisfied

J.J. Shann 1-104

- Hold-time violation:





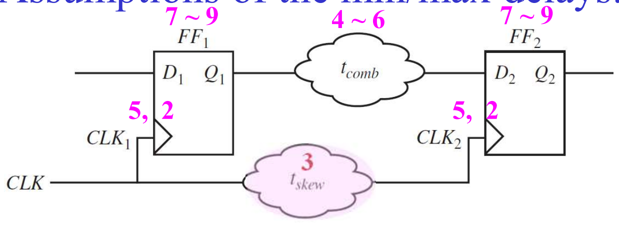
Condition $t_{pmin} + t_{cmin} \geq t_{skew} + t_{hold}$ not satisfied

J.J. Shann 1-105

105

Example: Positive skew (p.43)

- Assumptions of the min/max delays:



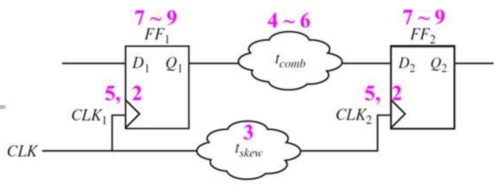
(a) If skew for the 2nd flip-flop is 3 ns, what is the max clock frequency? Compare it with the clock frequency if no skew is present.

(b) What is the biggest skew that the circuit can take while meeting the hold-time constraint for this circuit?

J.J. Shann 1-107

107

<Ans.>



(a) Max clock frequency:

i. with clock skew

$t_{clk} = t_{pmax} + t_{cmax} - t_{skew} + t_{su}$
$= 9\text{ ns} + 6\text{ ns} - 3\text{ ns} + 5\text{ ns}$
$= 17\text{ ns}$

$f_{max} = 1/17\text{ ns} = 58.82\text{ MHz}$

ii. without clock skew (p.39)

$t_{clk} = t_{pmax} + t_{cmax} + t_{su}$
$= 9\text{ ns} + 6\text{ ns} + 5\text{ ns}$
$= 20\text{ ns}$

$f_{max} = 1/20\text{ ns} = 50\text{ MHz}$

(b) The biggest skew:

$t_{pmin} + t_{cmin} \geq t_h + t_{skew} \Rightarrow 7\text{ ns} + 4\text{ ns} \geq 2\text{ ns} + t_{skew}$
$\Rightarrow t_{skew} \leq 9\text{ ns}$

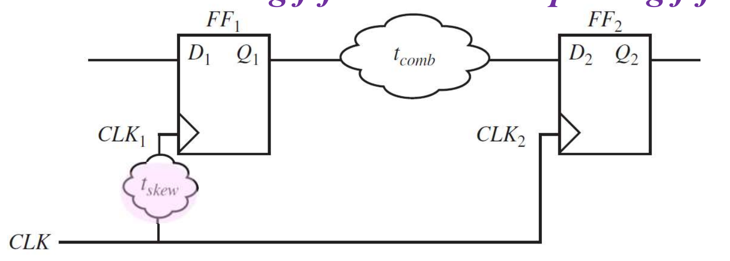
J.J. Shann 1-108

108

Timing Rules for Ckts with *Negative Skew*

* Negative skew is good for ***hold time***, but it is bad for ***setup time***.

launching f-f ***capturing f-f***



- Timing rules for circuits with negative skew:
  7. modification of Rule#5 ($t_{ck} \geq t_{pmax} + t_{cmax} - t_{skew} + t_{su}$)

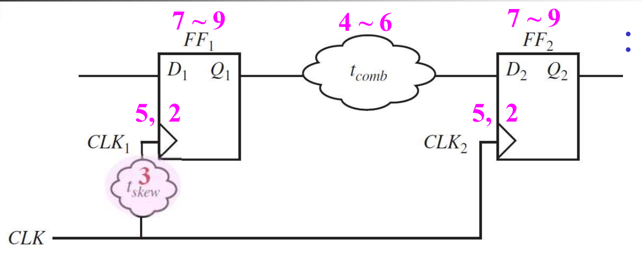
     $t_{ck} \geq t_{pmax} + t_{cmax} + t_{skew} + t_{su}$
  8. modification of Rule#6 ($t_{pmin} + t_{cmin} \geq t_h + t_{skew}$)

     $t_{pmin} + t_{cmin} \geq t_h - t_{skew}$

J.J. Shann 1-109

109

Example: Negative skew (p.43)

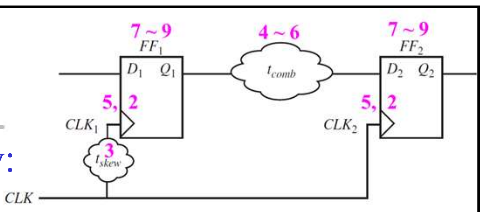


:

(a) If skew for the 1st flip-flop is 3 ns, what is the max clock frequency? Compare it with the clock frequency if no skew is present.

(b) What is the biggest skew that the circuit can take while meeting the hold-time constraint for this circuit?

<Ans.>



(a) Max clock frequency:

i. with clock skew

$$
\begin{aligned}
t_{ck} &= t_{pmax} + t_{cmax} + t_{skew} + t_{su} \\
&= 9 \text{ ns} + 6 \text{ ns} + 3 \text{ ns} + 5 \text{ ns} \\
&= 23 \text{ ns} \\
f_{max} &= 1/23 \text{ ns} = 43.47 \text{ MHz}
\end{aligned}
$$

ii. without clock skew (p.39)

$$f_{max} = 1/20 \text{ ns} = 50 \text{ MHz}$$

(b) The biggest skew:

$$t_{pmin} + t_{cmin} \geq t_h - t_{skew} \Rightarrow 7 \text{ ns} + 4 \text{ ns} + t_{skew} \geq 2 \text{ ns}$$

$\Rightarrow t_{skew} \geq -9$ ns

$\Rightarrow$ There will be no hold-time violation!

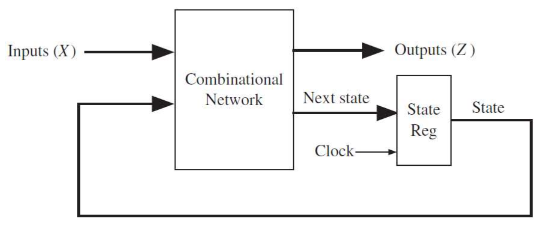
D. Glitches in Sequential Circuits

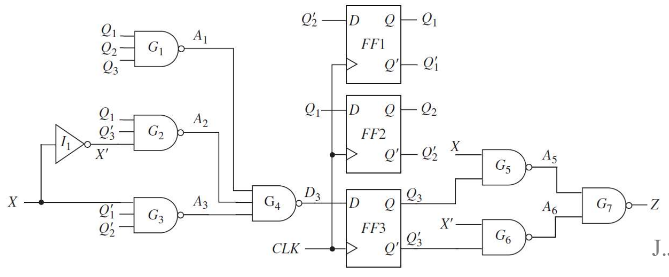
- *Glitch*: temporary false value appeared at the outputs and next states of a seq ckt
  - Cause: seq ckts often have ***external inputs*** that are ***asynchronous***

Example: a Mealy-Type Seq Ckt

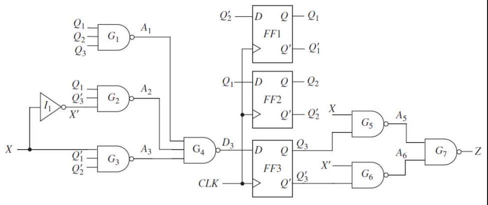
- BCD to excess-3 code converter

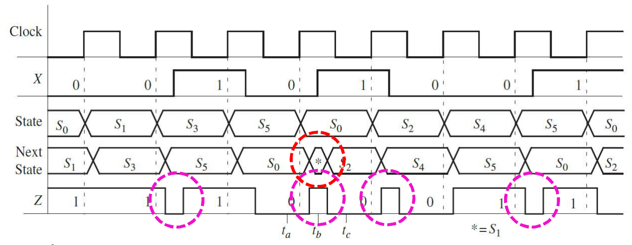
	NS		*Z*	
PS	$X=0$	$X=1$	$X=0$	$X=1$
S_0	S_1	S_2	1	0
S_1	S_3	S_4	1	0
S_2	S_4	S_4	0	1
S_3	S_5	S_5	0	1
S_4	S_5	S_6	1	0
S_5	S_0	S_0	0	1
S_6	S_0	–	1	–



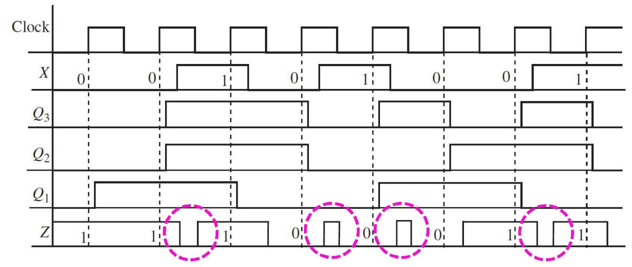


- Timing diagram:





	NS		Z	
PS	$X=0$	$X=1$	$X=0$	$X=1$
S_0	S_1	S_2	1	0
S_1	S_3	S_4	1	0
S_2	S_4	S_4	0	1
S_3	S_5	S_5	0	1
S_4	S_5	S_6	1	0
S_5	S_0	S_0	0	1
S_6	S_0	–	1	–



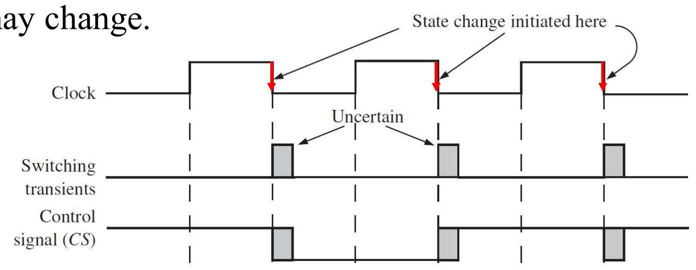
Q_2Q_3 \ Q_1	0	1
00	S_0	S_1
01		S_2
11	S_5	S_3
10	S_6	S_4

J.J. Shann 1-114

114

Glitches in Control Signals

- Problems of glitches in control signals:
  - After the triggering edge of the clock, there is a period of uncertainty during which control signals may change.



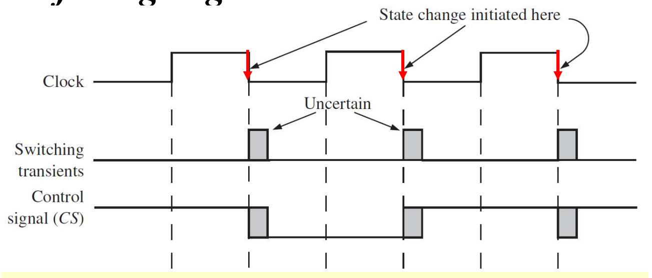
- Elimination of glitches in control signals:
  - ***Clock gating***
  - ***Clock enable***

J.J. Shann 1-115

115

Clock Gating

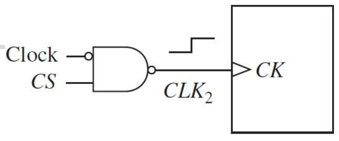
- Control signal gating: clock gating
  - For ***falling-edge*** device:

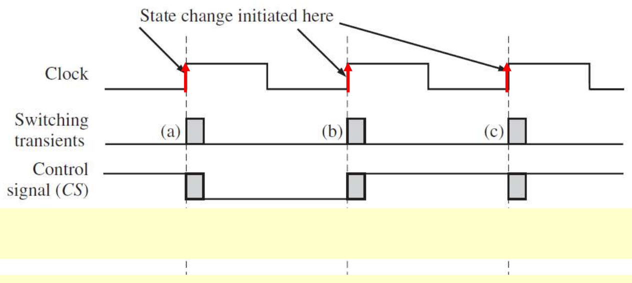


J.J. Shann 1-116

■116

  - For ***rising-edge*** device:





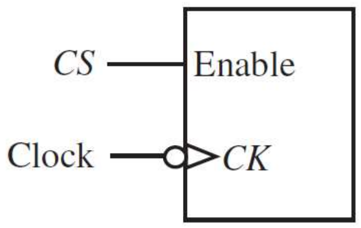
- Clock gating can lead to clock skew and additional timing problems . (×)

J.J. Shann 1-117

■117

Clock Enable

- Clock enable: (✓)
  - Use a clock enable (CE) input to synchronize.



1-11

Tristate Logic and Buses

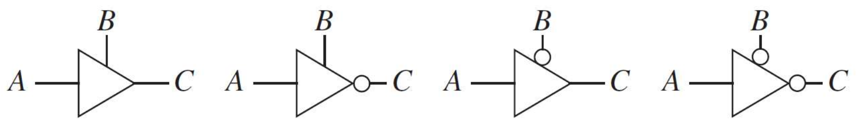
J.J. Shann

Tristate Logic and Busses

- Tristate buffers:
  - gates with a *high-impedance state* (*hi-Z*) in addition to high and low logic states.
  - The high-impedance state is equivalent to an open circuit.
- Use tristate buffers when connecting multiple gate outputs to the same wire or channel.
- Can be used to aid in data transfers b/t registers.
- Tristate buffers are either inverting or non-inverting.

Tristate Buffers

- 4 kinds of Tristate buffers:



B	*A*	*C*
0	0	Hi-Z
0	1	Hi-Z
1	0	0
1	1	1

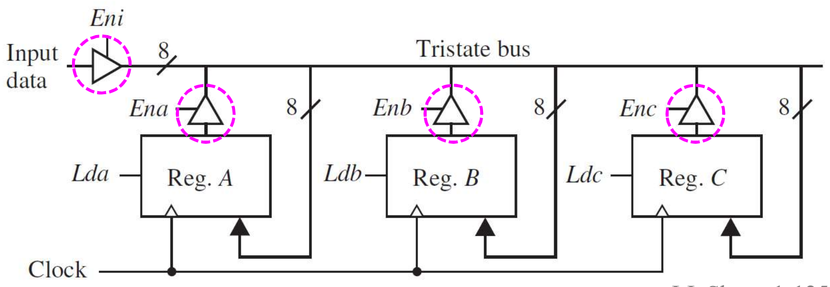
B	*A*	*C*
0	0	Hi-Z
0	1	Hi-Z
1	0	1
1	1	0

B	*A*	*C*
0	0	0
0	1	1
1	0	Hi-Z
1	1	Hi-Z

B	*A*	*C*
0	0	1
0	1	0
1	0	Hi-Z
1	1	Hi-Z

Example: Data Transfer Using Tristate Bus

- Only one group of buffers is enabled at a time.
  - *Enb* = *Ldc* = 1: The data in register B will be copied into register C when the active edge of the clock occurs.
  - *Eni* = *Lda* = *Ldb* = 1: The input data will be loaded in registers A and B when the registers are clocked.



J.J. Shann 1-125

Summary

- Review of important logic design topics:
  - Combinational logic
  - Sequential logic
  - Sequential circuit timing
  - Synchronous design
- For more details on any of the topics discussed in the chapter, refer to a standard logic design textbook.

J.J. Shann 1-126