Lecture 12:
Sequential Networks: Timing (contd), Standard Modules

CSE 140: Components and Design Techniques for Digital Systems

Diba Mirza
Dept. of Computer Science and Engineering
University of California, San Diego
If $t_{\text{setup}}$ is the setup time of a flip flop then a “setup time violation” is said to occur for the flip-flop if the _____1______ of the flip flop does not remain stable for at least $t_{\text{setup}}$ time _____2______the rising edge of the clock.

Q1 (1 min): Fill in the blanks above by choosing one of the combinations below

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>output</td>
<td>after</td>
</tr>
<tr>
<td>B.</td>
<td>output</td>
<td>before</td>
</tr>
<tr>
<td>C.</td>
<td>input</td>
<td>after</td>
</tr>
<tr>
<td>D.</td>
<td>input</td>
<td>before</td>
</tr>
</tbody>
</table>
Q2 (2 min): If you find a setup time violation in a sequential circuit, which of the following solutions would you propose?

A. Increase the minimum register to register delay by adding buffer gates
B. Decrease the minimum register to register delay by removing unnecessary gates
C. Increase the maximum delay by adding buffer gates
D. Increase the time period of the clock signal
E. None of the above
Q3 (2 min): If you find a hold time violation in a sequential circuit, which of the following solutions would you propose?

A. Increase the minimum register to register delay by adding buffer gates
B. Decrease the minimum register to register delay by removing unnecessary gates
C. Increase the maximum delay by adding buffer gates
D. Increase the time period of the clock signal
E. None of the above
Q4 (2 min): True or false: Fixing the hold time violation may cause a setup time violation, even if there was no setup time violation to begin with?

A. True
B. False
Q5 (4 min) Given the FSM circuit below, which of the following represent a valid transition sequence for $Q_1Q_0$?

A. 00 -> 01 -> 10 -> 11  ×
B. 00 -> 10 -> 01 -> 11
C. 01 -> 11 -> 01 -> 11
D. Both B and C
E. None of the above.
Clock Skew

- The clock doesn’t arrive at all registers at the same time
- Skew is the difference between two clock edges
- Examine the worst case to guarantee that the dynamic discipline is not violated for any register – many registers in a system!

\[(tpo + tpcq + tsmp) \leq \frac{1}{c}\]
\[(tcq + tccq) > thold\]
Setup time constraint

Consider a circuit where the setup constraint is satisfied for R2 when there is no clock skew.
Setup time constraint

Consider a circuit where the setup constraint is satisfied for R2 when there is no clock skew.

Is the setup constraint guaranteed to be satisfied if the rising edge of CLK2 arrives later than that of CLK1, i.e. CLK 2 is delayed (as shown in the figure)?

A. Yes  
B. No
Setup Time Constraint with Clock Skew

- In the worst case, the CLK2 is earlier than CLK1

\[
T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}
\]

\[
t_{pd} \leq T_c - (t_{pcq} + t_{setup} + t_{skew})
\]
Timing Analysis with clock skew

Timing Characteristics

- \( t_{ccq} = 30 \text{ ps} \)
- \( t_{pcq} = 50 \text{ ps} \)
- \( t_{setup} = 60 \text{ ps} \)
- \( t_{hold} = 70 \text{ ps} \)
- \( t_{pd} = 35 \text{ ps} \)
- \( t_{cd} = 25 \text{ ps} \)
- \( t_{skew} = 50 \text{ ps} \)

Setup time constraint:
What is the minimum allowable clock period given that the clocks are skewed by 50ps?

A. 215ps
B. 265ps
C. None of the above

\[ (t_{pd}(CL) + t_{pcq} + t_{setup} + t_{skew}) \leq T_c \]
\[ (105 + 50 + 60 + 50) \leq T_c \]
Timing Analysis with clock skew

Timing Characteristics

- $t_{ccq} = 30$ ps
- $t_{pcq} = 50$ ps
- $t_{setup} = 60$ ps
- $t_{hold} = 70$ ps
- $t_{pd} = 35$ ps
- $t_{cd} = 25$ ps
- $t_{skew} = 50$ ps

Setup time constraint:

- $T_c \geq 265$ ps
- $f_c = \frac{1}{T_c} = 3.77$ GHz

Without skew we got $f_c = 4.65$ GHz
Hold Time Constraint with Clock Skew

- In the worst case, CLK2 is later than CLK1

\[ t_{ccq} + t_{cd} > (t_{hold} + t_{skew}) \]

\[ t_{cd} > t_{hold} + t_{skew} - t_{ccq} \]
Hold Time Violation

Add buffers to the shortest paths:

Timing Characteristics

\[ t_{ccq} = 30 \text{ ps} \]
\[ t_{pcq} = 50 \text{ ps} \]
\[ t_{setup} = 60 \text{ ps} \]
\[ t_{hold} = 70 \text{ ps} \]
\[ t_{pd} = 35 \text{ ps} \]
\[ t_{cd} = 25 \text{ ps} \]
\[ t_{skew} = 50 \text{ ps} \]

\[ t_{pd} = 3 \times 35 \text{ ps} = 105 \text{ ps} \]
\[ t_{cd} = 2 \times 25 \text{ ps} = 50 \text{ ps} \]

Hold time constraint:

\[ t_{ccq} + t_{cd} > t_{hold} + t_{skew} ? \]
\[ (30 + 50) \text{ ps} > (70 \text{ ps} + 50) \text{ ps} ? \]
STANDARD MODULES
Interconnect: Decoder, Encoder, Mux, DeMux

Decoder: Decode the address to assert the addressed device
Mux: Select the inputs according to the index addressed by the control signals
Part III. Standard Modules

Interconnect Modules:
1. Decoder, 2. Encoder
3. Multiplexer, 4. Demultiplexer
Decoder Definition: A digital module that converts a binary address to the assertion of the addressed device

**n to \(2^n\) decoder function:**

\[ y_i = 1 \text{ if } E = 1 \& (I_2, I_1, I_0) = i \]

\[ y_i = 0 \text{ otherwise} \]

**n inputs**

\( n = 3 \)

**2^n outputs**

\( 2^3 = 8 \)

3 to \(2^3\) decoder

3 to 8 decoder
1. Decoder: Definition

- \( N \) inputs, \( 2^N \) outputs
- One-hot outputs: only one output HIGH at any point in time

<table>
<thead>
<tr>
<th>( A_1 )</th>
<th>( A_0 )</th>
<th>( Y_3 )</th>
<th>( Y_2 )</th>
<th>( Y_1 )</th>
<th>( Y_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( \overline{1} )</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

![Decoder Diagram]
Decoder: Logic Diagram (Inside a decoder)

\[ y_i = m_i \cdot En \]

\[ y_0 = 1 \text{ if } (A_1, A_0) = (0,0) \text{ & } En = 1 \]

\[ y_7 = A_1 A_0 En \]
PI Q: What is the output $Y_{3:0}$ of the 2:4 decoder for $(A_1, A_0) = (1,0)$?

A. $(1, 1, 0, 0 )$
B. $(1, 0, 1, 1)$
C. $(0, 0, 1, 0)$
D. $(0, 1, 0, 0)$
Implementing functions using Decoders

- OR minterms

\[ Y = AB + \overline{AB} \]

\[ = A \oplus B \]

\[
\begin{array}{c|c|c|c|c}
A & B & \text{2:4 Decoder} & & \text{Minterm} \\
\hline
0 & 0 & 00 & & AB \\
0 & 1 & 01 & & \overline{AB} \\
1 & 0 & 10 & & \overline{A}B \\
1 & 1 & 11 & & \overline{A}\overline{B} \\
\end{array}
\]

\[ f(A,B) = \sum m(0,3) \]
Decoder Application: universal set \{\text{Decoder, OR}\}

Example:
Implement the following functions with a 3-input decoder and OR gates.

i) \( f_1(a,b,c) = \Sigma m(1,2,4) \)

ii) \( f_2(a,b,c) = \Sigma m(2,3), \)

iii) \( f_3(a,b,c) = \Sigma m(0,5,6) \)
Example:
Implement the following functions with a 2:4 decoder and OR gates.

i) \( f_1(a,b,c) = \Sigma m(1,2,4) \)

How many 2:4 decoders are required to implement the above function?
A. One
B. Two
C. Three
D. Four
Example:
Implement the following functions with a 1:2 decoder and OR gates.

i) \( f_1(a,b,c) = \Sigma m(1,2,4) \)

How many 1:2 decoders are required to implement the above function?

A. Three  
B. Four  
C. Six  
D. Seven
Tree of Decoders

Implement a $4-2^4$ decoder with $3-2^3$ decoders.
Implement a 6-2^6 decoder with 3-2^3 decoders.
PI Q: A four variable switching function $f(a,b,c,d)$ can be implemented using which of the following?

A. 1:2 decoders and OR gates
B. 2:4 decoders and OR gates
C. 3:8 decoders and OR gates
D. None of the above
E. All of the above
Interconnect: Decoder, Encoder, Mux, DeMux

- **Decoder**: Decode the address to assert the addressed device
- **Mux**: Select the inputs according to the index addressed by the control signals

Diagram:
- Processors: P1, P2, Pk
- Memory Bank
- Arbiter
- Decoder: Decode the address to assert the addressed device
- Mux: Select the inputs according to the index addressed by the control signals
2. Encoder

• Definition (What is it?)
• Logic Diagram (How is it realized?)
2. Encoder: Definition

At most one $I_i = 1$.

$(y_{n-1}, \ldots, y_0) = i$ if $I_i = 1$ & $E_n = 1$

$(y_{n-1}, \ldots, y_0) = 0$ otherwise.

$A = 1$ if $E_n = 1$ and one $i$ s.t. $I_i = 1$

$A = 0$ otherwise.
Encoder: Logic Diagram

I_1 \quad I_3 \quad I_5 \quad I_7

\text{En} \quad y_0

I_2 \quad I_3 \quad I_6 \quad I_7

\text{En} \quad y_1

I_4 \quad I_5 \quad I_6 \quad I_7

\text{En} \quad y_2

I_0 \quad I_1 \quad \vdots \quad I_6 \quad I_7

\text{En} \quad A

32
Multiplexer

• Definition
• Logic Diagram
• Application
Interconnect: Decoder, Encoder, Mux, DeMux

Decoder: Decode the address to assert the addressed device
Mux: Select the inputs according to the index addressed by the control signals
Multiplexer Definition: Example

Selects between one of \( N \) inputs to connect to the output.
\( \log_2 N \)-bit select input – control input
PI Q: What is the output of the following MUX for the given inputs and $En=1$, $S=1$?

A. 0
B. 1
C. Can’t say
Multiplexer (Mux): Example

2:1 Mux

\[
\begin{array}{c|c|c|c|c|c}
S & D_1 & D_0 & Y \\
\hline
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c}
S & Y \\
\hline
0 & D_0 \\
1 & D_1 \\
\end{array}
\]
Multiplexer Application

- Mux for a Boolean function with truth table as input

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ Y = AB \]
# Multiplexer: Application

\[ Y = AB \]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ Y = AB \]
Multiplexer Application: universal set \{\text{Mux}\}

Example 1: Given \( f(a,b,c) = \Sigma m(0,1,7) + \Sigma d(2) \), implement with an 8-input Mux.

<table>
<thead>
<tr>
<th>Id</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Multiplexer Application: universal set \{Mux\}

Example 1: Given \( f(a,b,c) = \Sigma_m (0,1,7) + \Sigma_d(2) \), implement with an 8-input Mux.

<table>
<thead>
<tr>
<th>Id</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Example 2: Given $f(a,b,c) = \Sigma m(0,1,7) + \Sigma d(2)$, implement with 4-input Muxes.

<table>
<thead>
<tr>
<th>Id</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$D(c)$

$D_0(c) = 1$

$D_1(c) = 0$

$D_2(c) = 0$

$D_3(c) = c$
Example 2: Given $f(a,b,c) = \Sigma m (0,1,7) + \Sigma d(2)$, implement with 4-input Muxes.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c = 0</th>
<th>c = 1</th>
<th>D(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$D_0(c) = 1$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>$D_1(c) = 0$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$D_2(c) = 0$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$D_3(c) = c$</td>
</tr>
</tbody>
</table>

![Diagram of a 4-input MUX implementation](image-url)
Example 3: Given \( f(a,b,c) = \Sigma m (0,1,7) + \Sigma d(2) \), implement with 2-input Muxes.

<table>
<thead>
<tr>
<th>Id</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Example 3: Given \( f(a,b,c) = \Sigma m(0,1,7) + \Sigma d(2) \), implement with 2-input Muxes.

\[
D_1(b,c) = \begin{cases} 
0 & \text{if } c = 0 \\
bc & \text{if } c = 1 
\end{cases}
\]

\[
D_0(b,c) = b'
\]

\[
\begin{array}{c|cc}
b & c = 0 & c = 1 \\
\hline
0 & 0 & 0 \\
1 & 0 & 1 \\
\end{array}
\]

\( l_1(0) = 0 \), \( l_1(c) = c \)

\[
\begin{array}{c|c}
c & 0 & 1 \\
\hline
1 & 0 & 0 \\
0 & 0 & 1 \\
\end{array}
\]

\( D_0(b,c) = b' \)

\( D_1(b,c) = bc \)
Example 3: Given \( f(a,b,c) = \Sigma m(0,1,7) + \Sigma d(2) \), implement with 2-input Muxes.

\[
\begin{array}{c|cc|c|c}
D_1(b,c) & c = 0 & c = 1 & l_1(0) = 0 & l_1(c) = c \\
\hline
b & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 \\
\end{array}
\]
4. Demultiplexers

Control Input

En

x
4. Demultiplexers

\[ y_i = x \text{ if } i = (S_{n-1}, \ldots, S_0) \& En = 1 \]
\[ y_i = 0 \text{ otherwise} \]
Shifter

Can be implemented with a mux

\[ y_i = x_{i-1} \text{ if } En = 1, s = 1, \text{ and } d = L \]
\[ = x_{i+1} \text{ if } En = 1, s = 1, \text{ and } d = R \]
\[ = x_i \text{ if } En = 1, s = 0 \]
\[ = 0 \text{ if } En = 0 \]
Barrel Shifter

- S0: O or 1 shift
- S1: O or 2 shift
- S2: O or 4 shift

\[ x \]
\[ y \]