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
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!
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

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

\[ t_{pd} \leq T_c - (t_{pcq} + t_{\text{setup}} + t_{\text{skew}}) \]
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. 265 ps
C. None of the above

t_{pd} = 3 \times 35 \text{ ps} = 105 \text{ ps}
t_{cd} = 25 \text{ ps}
Timing Analysis with clock skew

Timing Characteristics

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

$T_c \geq 265 \text{ ps}$

$f_c = \frac{1}{T_c} = 3.77 \text{ GHz}$

Without skew we got $f_c = 4.65 \text{ GHz}$
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:

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

Hold time constraint:

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

\[ (30 + 50) \text{ ps} > (70 \text{ ps} + 50) \text{ ps} \]?

Timing Characteristics

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

$E$ (enable)

$I_0$ → 0
$I_1$ → 1
$I_2$ → 2

$0$  $1$  $2$  $3$  $4$  $5$  $6$  $7$

$y_0$  $y_1$  $y_2$  $y_3$  $y_4$  $y_5$  $y_6$  $y_7$

$n$ to $2^n$ decoder function:

$y_i = 1$ if $E = 1 \& (I_2, I_1, I_0) = i$

$y_i = 0$ otherwise

$n$ inputs

$n = 3$

$2^n$ outputs

$2^3 = 8$
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 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0</td>
<td>0 1</td>
<td>0 1</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 1</td>
<td>1 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>1 1</td>
<td>1 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
</tbody>
</table>
Decoder: Logic Diagram (Inside a decoder)

\[ y_i = m_i \cdot En \]

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

\[
\begin{array}{c|cccc}
A_1 & A_0 & Y_3 & Y_2 & Y_1 & Y_0 \\
\hline
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 \\
\end{array}
\]

\[ y_3 \]

\[ y_7 = A_1A_0En \]
1. Decoder: Definition

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

\[
Y = AB + \overline{AB} = A \oplus B
\]
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) \( f_3(a,b,c) = \Sigma m(0,5,6) \)
Decoder Application: universal set \{\text{Decoder, OR}\}

Example:
Implement the following functions with a 2:4 decoder and OR gates.

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

How many 2:4 decoders are required to implement the above function?
A. One
B. Two
C. Three
D. Four
Decoder Application: universal set \{\text{Decoder, OR}\}

Example:
Implement the following functions with a 1:2 decoder and OR gates.

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

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.
Tree of 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
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 \( I_i = 1 \) \& \( En = 1 \)

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

\[ A = 1 \text{ if } En = 1 \text{ and one } i \text{ s.t. } I_i = 1 \]

\[ A = 0 \text{ otherwise.} \]
Encoder: Logic Diagram

I_1 \quad I_3 \quad I_5 \quad I_7

I_2 \quad I_3 \quad I_6 \quad I_7

I_4 \quad I_5 \quad I_6 \quad I_7

I_0 \quad I_1 \quad I_6 \quad I_7

A
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
Selects between one of $N$ inputs to connect to the output.
Log$_2N$-bit select input – control input
PI Q: What is the output of the following MUX for the given inputs and $\text{En}=1$, $\text{S}=1$?

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

2:1 Mux

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

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

\[ Y = AB \]

<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>

A diagram of a multiplexer is shown with inputs A and B and output Y.
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 \{\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>
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>

\[ D(c) = \begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases} \]

\[ D_0(c) = 1 \]
\[ D_1(c) = 0 \]
\[ D_2(c) = 0 \]
\[ D_3(c) = c \]
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](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>

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

<table>
<thead>
<tr>
<th>( b )</th>
<th>( c = 0 )</th>
<th>( c = 1 )</th>
<th>( l_1(0) = 0 )</th>
<th>( l_1(c) = c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

\[
D_1(b, c) = \begin{array}{cc}
0 & 0 \\
0 & 1 \\
\end{array}
\]

\[
D_0(b, c) = \begin{array}{cc}
1 & - \\
1 & 0 \\
\end{array}
\]

\[
\begin{array}{c}
0 \\
1 \\
\end{array}
\]

\[
\begin{array}{c}
0 \\
1 \\
\end{array}
\]

\[
\begin{array}{c}
b' \\
0 \\
1 \\
\end{array}
\]

\[
\begin{array}{c}
a \\
\text{En} \\
\end{array}
\]

\[
\begin{array}{c}
y \\
\text{b} \\
\text{b'} \\
\end{array}
\]

\[
\begin{array}{c}
\text{0} \\
\text{1} \\
\end{array}
\]
Example 3: Given \( f(a,b,c) = \Sigma m(0,1,7) + \Sigma d(2) \), implement with 2-input Muxes.

<table>
<thead>
<tr>
<th>( b )</th>
<th>( c = 0 )</th>
<th>( c = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
4. Demultiplexers

[Diagram showing a demultiplexer with input labeled as 'En' and control input labeled as 'Control Input']
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

- **Shift**
  - $S_0$: O or 1 shift
  - $S_1$: O or 2 shift
  - $S_2$: O or 4 shift

- **Input** $x$

- **Output** $y$

- Sample inputs and outputs for different shifts:
  - 010101
  - 01010101
  - 0101010101

45