Timing Analysis, Decoders, Multiplexers for CSE 140-S

Chester Holtz

Based on SP21 slides

March 30, 2022
Overview

- Timing analysis
- Decoder
- Multiplexer
Definitions

- Setup time $t_{\text{setup}}$ Time before the clock data that must be stable (i.e. not change)
  - Setup time violation occurs if the input signal $D$ does not settle to a stable value at least $t_{\text{setup}}$ before the clock edge.
- Hold time $t_{\text{hold}}$ Time after the clock edge that data must be stable
  - Hold time violation occurs if the input signal $D$ does not remain unchanged at least $t_{\text{hold}}$ after the clock edge.
Sequential Networks & Timing Analysis

Definitions

- Setup time $t_{\text{setup}}$ Time before the clock data that must be stable (i.e. not change)
  - Setup time violation depends on maximum delay through the combinational logic
- Hold time $t_{\text{hold}}$ Time after the clock edge that data must be stable
  - Hold time violation depends on the minimum delay through the combinational logic
Sequential Networks & Timing Analysis

Timing constraints

- Skew constraint
  \[ t_{\text{skew}} = \text{CLK1} - \text{CLK2} \]

- Setup time constraint
  \[ T_C \geq t_{\text{pcq}} + t_{\text{pd}} + t_{\text{setup}} - t_{\text{skew}} \]

- Hold time constraint
  \[ t_{\text{ccq}} + t_{\text{cd}} > t_{\text{hold}} + t_{\text{skew}} \]

Circuit Properties

<table>
<thead>
<tr>
<th>Circuit Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flip-Flops</strong></td>
</tr>
<tr>
<td>( t_{\text{setup}} )</td>
</tr>
<tr>
<td>( t_{\text{hold}} )</td>
</tr>
<tr>
<td>( t_{\text{skew}} )</td>
</tr>
<tr>
<td>( t_{\text{pcq}} )</td>
</tr>
<tr>
<td>( t_{\text{ccq}} )</td>
</tr>
</tbody>
</table>
Sequential Networks & Timing Analysis

Timing constraints

- **Skew constraint**
  \[ t_{\text{skew}} = \text{CLK1} - \text{CLK2} \]

- **Setup time constraint**
  \[ T_C \geq t_{pcq} + t_{pd} + t_{\text{setup}} - t_{\text{skew}} \]

- **Hold time constraint**
  \[ t_{ccq} + t_{cd} > t_{\text{hold}} + t_{\text{skew}} \]

---

<table>
<thead>
<tr>
<th>Circuit Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flip-Flops</strong></td>
</tr>
<tr>
<td>( t_{\text{setup}} )</td>
</tr>
<tr>
<td>( t_{\text{hold}} )</td>
</tr>
<tr>
<td>( t_{skew} )</td>
</tr>
<tr>
<td>( t_{pcq} )</td>
</tr>
<tr>
<td>( t_{ccq} )</td>
</tr>
</tbody>
</table>
Sequential Networks & Timing Analysis

Timing constraints

- **Skew constraint**
  \[ t_{\text{skew}} = \text{CLK1} - \text{CLK2} \]

- **Setup time constraint**
  \[ T_C \geq t_{\text{pcq}} + t_{\text{pd}} + t_{\text{setup}} - t_{\text{skew}} \]

- **Hold time constraint**
  \[ t_{\text{ccq}} + t_{\text{cd}} > t_{\text{hold}} + t_{\text{skew}} \]

### Examples given a circuit:

- What is the maximum clock frequency?
- Clock skew tolerance s.t. hold time constraint satisfied?

<table>
<thead>
<tr>
<th>Circuit Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flip-Flops</strong></td>
</tr>
<tr>
<td>( t_{\text{setup}} )</td>
</tr>
<tr>
<td>( t_{\text{hold}} )</td>
</tr>
<tr>
<td>( t_{\text{skew}} )</td>
</tr>
<tr>
<td>( t_{\text{pcq}} )</td>
</tr>
<tr>
<td>( t_{\text{ccq}} )</td>
</tr>
</tbody>
</table>
Sequential Network & Timing Analysis Example

Setup time constraint
\[ T_C \geq t_{pcq} + t_{pd} + t_{setup} - t_{skew} \]

Hold time constraint
\[ t_{ccq} + t_{cd} > t_{hold} + t_{skew} \]

Circuit Properties

<table>
<thead>
<tr>
<th>Flip-Flops</th>
<th>Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{setup} = 60\text{ps} )</td>
<td>( t_{pd} = 100\text{ps} ) (prop. delay)</td>
</tr>
<tr>
<td>( t_{hold} = 20\text{ps} )</td>
<td>( t_{cd} = 55\text{ps} ) (contam. delay)</td>
</tr>
<tr>
<td>( t_{pcq} = 70\text{ps} )</td>
<td></td>
</tr>
<tr>
<td>( t_{ccq} = 50 )</td>
<td></td>
</tr>
</tbody>
</table>

1. If there is no clock skew, what is the maximum operating frequency?

2. How much clock skew can the circuit tolerate before hold-time violation?
Sequential Network & Timing Analysis Example

**Setup time constraint**
\[ T_C \geq t_{pcq} + t_{pd} + t_{setup} - t_{skew} \]

**Hold time constraint**
\[ t_{ccq} + t_{cd} > t_{hold} + t_{skew} \]

### FA’10 Midterm2 Q.4

<table>
<thead>
<tr>
<th>Circuit Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flip-Flops</strong></td>
<td>Gates</td>
</tr>
<tr>
<td>( t_{setup} ) = 60,ps</td>
<td>( t_{pd} ) = 100,ps (prop. delay)</td>
</tr>
<tr>
<td>( t_{hold} ) = 20,ps</td>
<td>( t_{cd} ) = 55,ps (contam. delay)</td>
</tr>
<tr>
<td>( t_{pcq} ) = 70,ps</td>
<td>( t_{ccq} ) = 50</td>
</tr>
<tr>
<td>( t_{pd} ) = 100,ps (prop. delay)</td>
<td></td>
</tr>
</tbody>
</table>

1. If there is no clock skew, what is the maximum operating frequency?

\[
T_C \geq t_{pcq} + t_{pd} + t_{setup} - t_{skew} \\
\geq 70 + 3 \cdot 100 + 60 = 430 \, \text{ps}
\]

\[
F = \frac{1}{T_C} = 2.33 \, \text{GHz}
\]
Sequential Network & Timing Analysis Example

**Setup time constraint**

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

**Hold time constraint**

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

---

FA’10 Midterm2 Q.4

2. How much clock skew can the circuit tolerate before *hold-time violation*?

\[
T_{ccq} + T_{cd} \geq T_{hold} + T_{skew}
\]

\[
T_{skew} \leq 50 + 1 \cdot 55 - 20
\]

\[
\leq 85 \text{ ps}
\]
Mealy and Moore Machines

- **Mealy machine:** general
  - $y_i(t) = f_i(X(t), S(t))$

- **Moore machine:** output is independent of current input
  - $y_i(t) = f_i(S(t))$
  - $S_i(t + 1) = g_i(X(t), S(t))$
Mealy and Moore Machines

- Mealy machine: general
  - $y_i(t) = f_i(X(t), S(t))$

- Moore machine: output is independent of current input
  - $y_i(t) = f_i(S(t))$
  - $S_i(t + 1) = g_i(X(t), S(t))$
Mealy and Moore Machines

- Mealy machine: general
  - $y_i(t) = f_i(X(t), S(t))$
- Moore machine: output is independent of current input
  - $y_i(t) = f_i(S(t))$
  - $S_i(t+1) = g_i(X(t), S(t))$

Which is which?
Conversion from Mealy to Moore

Algorithm

1. Identify distinct \((NS, y)\) pair
2. Replace each distinct \((NS, y)\) pair with distinct new states
3. Insert rows of present state=new states
4. Append each present state with its output \(y\).
Conversion from Mealy to Moore

Algorithm

1. Identify distinct \((N_S, y)\) pair
2. Replace each distinct \((N_S, y)\) pair with distinct new states
3. Insert rows of present state=new states
4. Append each present state with its output \(y\).

A pattern recognizer produces output \(y = 1\) when \(x(t - 4, t) = 01010\)

\[
\begin{array}{c|c|c}
\text{Input} & x = 0 & x = 1 \\
\hline
S_0 & \text{PS} & S_1 \\
S_1 & S_2 & S_3 \\
S_2 & S_3 & S_4 \\
S_3 & S_4 & \text{PS} \\
S_4 & \text{PS} & S_0 \\
\end{array}
\]
Conversion from Mealy to Moore

Algorithm

1. Identify distinct \((NS, y)\) pair
2. Replace each distinct \((NS, y)\) pair with distinct new states
3. Insert rows of present state=new states
4. Append each present state with its output \(y\).

A pattern recognizer produces output \(y = 1\) when \(x(t - 4, t) = 01010\)

<table>
<thead>
<tr>
<th>Input</th>
<th>(x = 0)</th>
<th>(x = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S_0)</td>
<td>((S_1, 0))</td>
<td>((S_0, 0))</td>
</tr>
<tr>
<td>(S_1)</td>
<td>((S_1, 0))</td>
<td>((S_2, 0))</td>
</tr>
<tr>
<td>(PS) (S_2)</td>
<td>((S_3, 0))</td>
<td>((S_0, 0))</td>
</tr>
<tr>
<td>(S_3)</td>
<td>((S_1, 0))</td>
<td>((S_4, 0))</td>
</tr>
<tr>
<td>(S_4)</td>
<td>((S_3, 1))</td>
<td>((S_0, 0))</td>
</tr>
</tbody>
</table>
Conversion from Mealy to Moore

Algorithm
1. Identify distinct \((NS, y)\) pair
2. Replace each distinct \((NS, y)\) pair with distinct new states
3. Insert rows of present state=new states
4. Append each present state with its output \(y\).

A pattern recognizer produces output \(y = 1\) when 
\[x(t - 4, t) = 01010\]

<table>
<thead>
<tr>
<th>(S_0)</th>
<th>(S_1)</th>
<th>(S_2)</th>
<th>(S_3)</th>
<th>(S_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((S_1, 0))</td>
<td>((S_2, 0))</td>
<td>((S_3, 0))</td>
<td>((S_4, 0))</td>
<td>((S_3, 1))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input (x)</th>
<th>(x = 0)</th>
<th>(x = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S_0)</td>
<td>((S_0, 0))</td>
<td>((S_0, 0))</td>
</tr>
<tr>
<td>(S_1)</td>
<td>((S_2, 0))</td>
<td>((S_2, 0))</td>
</tr>
<tr>
<td>(S_2)</td>
<td>((S_0, 0))</td>
<td>((S_0, 0))</td>
</tr>
<tr>
<td>(S_3)</td>
<td>((S_4, 0))</td>
<td>((S_4, 0))</td>
</tr>
<tr>
<td>(S_4)</td>
<td>((S_0, 0))</td>
<td>((S_0, 0))</td>
</tr>
</tbody>
</table>
**Conversion from Mealy to Moore**

**Algorithm**

1. Identify distinct \((NS, y)\) pair
2. Replace each distinct \((NS, y)\) pair with distinct new states
3. Insert rows of present state=new states
4. Append each present state with its output \(y\).

A pattern recognizer produces output \(y = 1\) when \(x(t - 4, t) = 01010\)

\[
\begin{array}{c|c|c}
\text{Input} & \text{Out} & x=0 & x=1 & y \\
\hline
S_0 & S_1 & S_0 & 0 \\
S_1 & S_1 & S_2 & 0 \\
S_2 & S_3 & S_0 & 0 \\
S_3 & S_1 & S_4 \\
S_4 & S_5 & S_0 \\
S_5 & \\
\end{array}
\]
Conversion from Mealy to Moore

Algorithm

1. Identify distinct \((NS, y)\) pair
2. Replace each distinct \((NS, y)\) pair with distinct new states
3. Insert rows of present state=new states
4. Append each present state with its output \(y\).

A pattern recognizer produces output \(y = 1\) when
\[x(t - 4, t) = 01010\]
Decoders

Definitions

- Digital module, $N$ inputs, $2^N$ outputs
- One-hot outputs: only one output HIGH (1) at most
Decoders

Definitions

- Digital module, $N$ inputs, $2^N$ outputs
- One-hot outputs: only one output HIGH (1) at most

Can be used to implement Boolean functions

- Decoder produces minterms when $E=1$.
- Can use an OR gate to collect the minterms to cover the On-set.
- For the Don’t Care-Set, we can just ignore the terms
Decoders

Example

\[ f(a, b, c, d) = \sum m(12, 15) \]

Implement \( f \) using a 4–16 decoder and OR gates
Decoders

Example

\[ f(a, b, c, d) = \sum m(12, 15) \]

Implement \( f \) using a 4 - 16 decoder and OR gates
Decoders

Example

\[ f(a, b, c, d) = \sum m(12, 15) \]

Implement \( f \) using 3–8 decoders and OR gates
Decoders

Example

\[ f(a, b, c, d) = \sum m(12, 15) \]

Implement \( f \) using 3–8 decoders and OR gates
Multiplexers

Definitions

▶ Selects one of \( N \) inputs to connect to the output
▶ \( \log_2(N) \)-bit select input – control input
Example

\[ f(a, b, c) = \sum m(0, 3, 5, 7) + \sum d(6) \]

implement the function using a minimal network of 2:1 multiplexers.
Multiplexers

Example

\[ f(a, b, c) = \sum_{m}^{}_{(0, 3, 5, 7)} + \sum_{d}^{}_{6} \]

<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>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</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>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Multiplexers

Example

\[ f(a, b, c) = \sum m(0, 3, 5, 7) + \sum d(6) \]

implement the function using a minimal network of 2:1 multiplexers.

<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>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</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>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bc</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>a=0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>a=1</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

I0 = d0(bc)
I1 = d1(bc)
### Example

$$f(a, b, c) = \sum m(0, 3, 5, 7) + \sum d(6)$$

implement the function using a minimal network of 2:1 multiplexers.

<table>
<thead>
<tr>
<th>bc</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>a=0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>a=1</td>
<td>0</td>
<td>1</td>
<td>×</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \begin{align*}
I_0 &= d_0(bc) \\
I_1 &= d_1(bc)
\end{align*} \]

<table>
<thead>
<tr>
<th>c\b</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c\b</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Multiplexers

Example

\[ f(a, b, c) = \sum m(0, 3, 5, 7) + \sum d(6) \]

implement the function using a minimal network of 2:1 multiplexers.

<table>
<thead>
<tr>
<th>c</th>
<th>b</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>b=0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>b=1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Multiplexers

Example

\[ f(a, b, c) = \sum m(0, 3, 5, 7) + \sum d(6) \]

implement the function using a minimal network of 2:1 multiplexers.
Good Luck!