ECE 2300
Digital Logic & Computer Organization
Spring 2017

Timing Analysis
Binary Number Representation
Announcements

• Today’s instructor office hour cancelled

• HW4 to be released tonight

• Sign up for decoder games
  – 2 designers + 2 challengers per game
    • Game designers will meet with instructor 24hrs before class
  – Bidding starts around 9:00pm this evening on Piazza
    • First come first serve
Synchronous Circuits

- The changes in the state of the memory elements are synchronized by a clock signal
  - All flip-flops (FFs) are synchronized to capture the inputs “simultaneously” on the clock tick

- Must ensure the output of the combinational logic has settled before the next clock tick
Review: Glitches in Synchronous Circuits

\[ S \cdot Y \]
\[ S' \cdot X \]
\[ F \]

CLOCK
Stable FF Situation

- CLOCK
- D: stable
- Q: stable

$t_{	ext{clk}}$, $t_{	ext{setup}}$, $t_{	ext{hold}}$, $t_{	ext{ffpd}}$
What if This Happens?

D2 input still transitioning

May capture neither HIGH nor LOW
Metastable State

- Q stuck in the undefined region between 0 and 1
- Eventually moves to a stable state, but may take a while (metastable resolution time)
But What About This Situation?

Wrong value captured
Avoiding Timing Failure

• Possible causes of metastability and wrong value capture
  – Clock pulse that is too narrow
  – Input changes too soon before a clock edge
  – Input changes too soon after a clock edge

• Avoid by meeting setup time, hold time, and minimum clock pulse width specifications
Sequential Circuit Timing Analysis

- **Timing analysis** involves calculating the time delays between all FF pairs within the circuit.

- To determine the maximum operating frequency and ensure that setup time requirements are met:
  - The clock cannot be too fast.

- To ensure that hold time requirements are met:
  - The minimum propagation delay of the combinational logic (contamination delay) cannot be too small.
  - *Independent of clock frequency*.
Important Timing Parameters

CLOCK

combinational logic

CLOCK

$\begin{align*}
&\text{CLOCK} \\
t_H &\quad t_{\text{clk}} \\
&\quad t_L \\
&\text{flip-flop outputs} \\
t_{\text{ffpd}} \\
&\text{combinational outputs} \\
t_{\text{comb}} \\
&\text{flip-flop inputs} \\
t_{\text{setup}} &\quad t_{\text{hold}}
\end{align*}$
Setup Time Constraint

- **FF input must be stable** \( t_{\text{setup}} \) **before the triggering clock transition**
Determining Clock Cycle Time

\[ t_{\text{ffpd(max)}} + t_{\text{comb(max)}} + t_{\text{setup}} \leq t_{\text{clk}} \]

Every circuit path between every pair of FFs must satisfy the above equation to run the circuit at a frequency of \( 1/t_{\text{clk}} \)

- The longest timing path (worst case) determines the maximum clock frequency
  - Worst case temperature and voltage
  - Worst case manufacturing variations
Example: Setup Time Calculations

<table>
<thead>
<tr>
<th></th>
<th>Prop Delay (ns)</th>
<th>Setup Time (ns)</th>
<th>Hold Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>max</td>
<td></td>
</tr>
<tr>
<td>FF</td>
<td>1</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Comb</td>
<td>3</td>
<td>9</td>
<td>-</td>
</tr>
</tbody>
</table>

- How fast can we run this circuit?
Example: Setup Time Calculations

\[ t_{\text{clk}} \geq t_{\text{ffpd(max)}} + t_{\text{comb(max)}} + t_{\text{setup}} = 7 + 9 + 3 = 19\text{ns} \]
Hold Time Constraint

- FF input must stay stable $t_{\text{hold}}$ after the triggering clock transition
  - Otherwise, the receiving flip-flop may be contaminated with an unexpected value
- Need to consider minimum propagation delays (contamination delays) for hold time calculations
  $$t_{\text{ffpd(min)}} + t_{\text{comb(min)}} \geq t_{\text{hold}}$$
Example: Hold Time Constraint

Hold time windows ($t_{\text{hold}}$)
D2 must be held stable for FF2

$$t_{\text{ffpd}}(\text{min}) + t_{\text{comb}}(\text{min}) = t_{\text{ffpd}}(\text{min}) + 0 \geq t_{\text{hold}}$$
Example: Hold Time Calculations

<table>
<thead>
<tr>
<th></th>
<th>Prop Delay (ns)</th>
<th>Setup Time (ns)</th>
<th>Hold Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>max</td>
<td></td>
</tr>
<tr>
<td>FF</td>
<td>1</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Comb</td>
<td>3</td>
<td>9</td>
<td>-</td>
</tr>
</tbody>
</table>

- Hold time at FF2 met?
Clock Skew Complicates Matters Further

- Clock may not reach all flip-flops simultaneously

\[ t_{\text{skew}} \text{ (max)} : \text{Maximum clock skew} \]
\[ t_{\text{skew}} \text{ (min)} : \text{Minimum clock skew} \]
Cycle Time With Clock Skew

IN \rightarrow FF1 \rightarrow Q1 \rightarrow \text{Combinational logic} \rightarrow D2 \rightarrow FF2 \rightarrow \text{CLOCK}

- **IN**
- **CLK1** (delayed)
- **Q1** (delayed)
- **D2** (delayed)
- **CLK2**

\[ t_{\text{setup}} \quad t_{\text{skew}} \]
Negative Clock Skew

Sending FF receives clock later than receiving FF

\[ t_{ffpd(\text{max})} + t_{\text{comb}(\text{max})} + t_{\text{setup}} \leq t_{\text{clk}} - t_{\text{skew(\text{max})}} \]

Harmful skew for meeting setup time constraint
Positive Clock Skew

Receiving FF receives clock later than sending FF

\[ t_{ffpd(max)} + t_{comb(max)} + t_{setup} \leq t_{clk} + t_{skew(min)} \]

Beneficial skew for meeting setup time constraint
Hold Time With Positive Clock Skew

Receiving FF receives clock later than sending FF

\[ t_{\text{ffpd(min)}} + t_{\text{comb(min)}} \geq t_{\text{hold}} + t_{\text{skew(max)}} \]

Harmful skew for meeting hold time constraint
Hold Time With Negative Clock Skew

What if sending FF receives clock later than receiving FF?

\[ t_{ffpd\text{(min)}} + t_{comb\text{(min)}} \geq t_{hold} - t_{skew\text{(min)}} \]

Beneficial skew for meeting hold time constraint
Course Content

- Binary numbers and logic gates
- Boolean algebra and combinational logic
- Sequential logic and state machines
- Binary arithmetic
- Memories
- Instruction set architecture
- Processor organization
- Caches and virtual memory
- Input/output
Positional Number Representation

• **What does 1432.67 mean?**
  \[ 1432.67 = 1 \times 10^3 + 4 \times 10^2 + 3 \times 10^1 + 2 \times 10^0 + 6 \times 10^{-1} + 7 \times 10^{-2} \]
  – Base 10 positional representation
  – Uses digits 0, 1, 2, ..., 9

• **General base B positional representation**
  \[ a_n a_{n-1} \ldots a_2 a_1 a_0 = a_n B^n + a_{n-1} B^{n-1} + \ldots + a_2 B^2 + a_1 B^1 + a_0 B^0 \]
  – Uses digits 0, 1, 2, ..., B-1

• **Bases of interest to computer designers**
  – Base 2, Binary (digits 0,1)
  – Base 16, Hexadecimal (digits 0,1,...,9,A,B,...,E,F)
Hexadecimal Notation

- Often convenient to write binary (base-2) numbers as hexadecimal (base-16) numbers
  - Fewer digits: 4 bits per hex digit
  - Less error prone: easy to misread long string of 1’s and 0’s

<table>
<thead>
<tr>
<th>Binary</th>
<th>Hex</th>
<th>Decimal</th>
<th>Binary</th>
<th>Hex</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
<td>1001</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>2</td>
<td>1010</td>
<td>A</td>
<td>10</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>3</td>
<td>1011</td>
<td>B</td>
<td>11</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>4</td>
<td>1100</td>
<td>C</td>
<td>12</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>5</td>
<td>1101</td>
<td>D</td>
<td>13</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>6</td>
<td>1110</td>
<td>E</td>
<td>14</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
<td>1111</td>
<td>F</td>
<td>15</td>
</tr>
</tbody>
</table>
Converting from Binary to Hex

- Every group of four bits is a hex digit
  - Start grouping from right-hand side

\[ 0011 | 1010 | 1000 | 1111 | 0100 | 1101 | 0111 \]

This is not a new machine representation; just a compact way to write the number
Review: Binary Numbers

- Recall weighted positional notation for decimal numbers

\[
329 = 3 \times 10^2 + 2 \times 10^1 + 9 \times 10^0
\]

- Use similar weighted positional system for binary

\[
101 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0
\]
Unsigned Binary Integers

- An $n$-bit unsigned number represents $2^n$ integer values
  - From 0 to $2^n-1$

<table>
<thead>
<tr>
<th>$2^2$</th>
<th>$2^1$</th>
<th>$2^0$</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>
Unsigned Binary Fractions

• For the binary number $b_{p-1}b_{p-2}\ldots b_1b_0.b_{-1}b_{-2}\ldots b_{-n}$ the decimal number is

$$D = \sum_{i=-n}^{p-1} b_i \cdot 2^i$$

• Examples

$101.001_2 = ?$
Before Next Class

• H&H 5.1-5.2.3

Next Time

Binary Arithmetic