2. Timing Design in Digital Systems

Dr. Paul D. Franzon

Outline

1. Timing design in Synchronous (clocked) Logic
   - Min/Max timing with flip-flops
   - Latch-based design
3. Timing Issues in CMOS circuits
4. Timing verification Flow
5. Techniques to Improve Performance
Course “Mantras”

- One clock, one edge, Flip-flops only
- Design BEFORE coding
- Behavior implies function
- Clearly separate control and datapath
Objectives

Module Objectives:

- Describe how a clock-synchronized design behaves at a clock-cycle to cycle level. Know how to draw a timing diagram for a design.
- Describe how clocks are distributed. Define clock skew and jitter.
- Describe the basic behavior of flip-flops and latches.
- Define setup and hold times.
- Derive the timing equations for flip-flop based designs.
- Describe the basic behavior of latches, including set-up and hold times.
- Derive the timing equations for latch based designs.
- Understand the application of cycle stealing in latches.
- Describe timing closure in the CAD tool flow.
- Understand the relationship between design and clock frequency
Motivation

- Synchronous design is one cornerstone of modern digital design
  - Events are synchronized by a master clock using flip-flops and latches

- The logic design strongly impacts the achievable clock frequency and thus IC performance
  - The synthesis tools help achieve a specific clock target but cannot fix a poorly structured design

- The designer needs to understand
  - How the design and the CAD tools achieve a clock frequency that satisfies set-up requirements
  - How the CAD tools try to fix hold requirements
    - Any why this is much harder when latches are used
References

Smith and Franzon,

- Section 11.2 (timing relationships)
- Chapter 13 (CAD flow)

Ciletti,

- Sections 3.1, 3.2.1 (flip-flops, latches)
- Section 11.2.1 (Static timing analysis)
**Mantra #1**

**One clock, one edge; Flip-flops only**

- For your design (at least for each module) use one clock source and only one edge of that clock
- Only use edge-triggered flip-flops

**Why?**
- Moving data between different clock domains requires careful timing design and synthesis “scripting”

**If you need multiple clocks in your design**
- Make them related by a powers of 2
  - E.g 50, 100 and 200 MHz
- Consider one clock per module
- Consider resynchronizing using flip-flops between clock domains

**Caveat**
- Tools and designers are getting better at using latches and multi-phase clocks. However, this requires some experience to get correct.
General Approach to Timing Design

- In general, all signals start and end in registers every clock period
- This is how the clock synchronizes logic events
Clock Level Timing

- Example (Revision):

```
<table>
<thead>
<tr>
<th>In[0]</th>
<th>In[1]</th>
<th>In[2]</th>
<th>In[3]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- Clock

```
Clock
```

```
<table>
<thead>
<tr>
<th>Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

```
In
```

```
<table>
<thead>
<tr>
<th>In</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
```

```
Out
```

```
<table>
<thead>
<tr>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
```

©2011 Dr. Paul D. Franzon, www.ece.ncsu.edu/erl/faculty/paulf.html
Critical Path

- Thus, the clock speed is determined by the slowest feasible path between registers in the design
  - Often referred to as "the critical path"

Critical path is longer with increased logic depth (# gates in series)
Synchronous Clock Distribution

- The goal of clock tree is for the clock to arrive at every leaf node at the same time:
- Usually designed after synthesis: Matched buffers; matched capacitance loads
- Common design method:
  - “H tree”
  - Clock tree done after design
  - Balance RC delays
  - Balance buffer delays
Clock skew and jitter

- Clock skew = systematic clock edge variation between sites
  - Mainly caused by delay variations introduced by manufacturing variations
  - Random variation
- Clock jitter = variation in clock edge timing between clock cycles
  - Mainly caused by noise

Equations below lump jitter and skew into a single "skew" number. Skew is larger anyway. Skew + jitter called "Uncertainty" in Synopsys.
Comments on Clock Skew

- ASIC design relies on automatic clock tree synthesis
  - Works to guarantee a global skew target
- Custom clock distribution can be used to add the following features to a clock:
  - Smaller skews
  - Local skews < Global skew
  - Multiple non-overlapping clock phases
  - Deliberate non-random skew at flip-flop/latch level
  - Future automatic clock tree synthesis tools might include features like this
Flip-Flop based design

**Edge triggered D-flip-flop**
Q becomes D after clock edge

*Set-up time:*
Data can not change no later than this point before the clock edge.

*Hold time:*
Data can not change during this time after the clock edge.

*t_clock-Q*
Delay on output (Q) changing from positive clock edge
**Preventing Set-Up Violations**

**Set-up violation:**
Logic is too slow for the correct logic value to arrive at the inputs to the register on the right before one set-up time before the clock edge.

Constraint to prevent this:

\[
t_{clock} \geq t_{clock-Q-max} + t_{logic-max} + t_{set-up} + t_{skew}
\]

- The amount of time required to turn ‘>’ into ‘=’ is referred to as **timing slack**.
Preventing hold violations

Hold violations occur when race-through is possible

Constraint to prevent hold violations:

\[ t_{\text{hold}} + t_{\text{skew}} \leq t_{\text{clock-Q-min}} + t_{\text{logic-min}} \]

- Sometimes have to insert additional logic to prevent hold violations

©2011 Dr. Paul D. Franzon, www.ece.ncsu.edu/erl/faculty/paulf.html
**What can happen with a timing violation?**

D changes outside setup and hold ➔ tclock-Q is correct

\[
\begin{align*}
Ck & \quad \downarrow \\
D & \quad t_{\text{setup}} \quad \leftrightarrow \quad t_{\text{hold}} \\
Q & \quad t_{\text{clock-Q}} \uparrow
\end{align*}
\]

D changes during setup and hold ➔ tclock-Q longer than specified, or Q does not transition correctly

\[
\begin{align*}
Ck & \quad \downarrow \\
D & \quad t_{\text{setup}} \quad \leftrightarrow \quad t_{\text{hold}} \\
Q & \quad t_{\text{clock-Q}} \uparrow
\end{align*}
\]

\[
\begin{align*}
Ck & \quad \downarrow \\
D & \quad t_{\text{setup}} \quad \leftrightarrow \quad t_{\text{hold}} \\
Q & \quad t_{\text{clock-Q}} \uparrow
\end{align*}
\]
What can happen with a timing violation?

D changes after during setup and hold

\[ Ck \]

\[ t_{\text{SetUp}} \quad t_{\text{hold}} \]

Q changes in WRONG clock cycle
- Racethrough
**Latch Based Design**

**D-latch**
- Q follows D while clock is high ("transparent")
- Value on D when clock goes low is stored on Q

**Set-up and hold times:**
D can not change close to the falling (‘latching’) clock edge.

- **t_clock-Q**
  Delay from clock going high to Q changing

- **t_D-Q**
  Delay from D changing to Q changing while clock high (assumed = t_\text{ck-Q} here)
**Latch Timing Constraints**

- Set-up constraints under nominal design same as for flip-flop

\[ t_{clock} \geq t_{clock-Q-max} + t_{logic-max} + t_{set-up} + t_{skew} \]

- “Transparency” of latch can be used to improve flexibility of timing
  e.g.: If critical path is in logic block 1:

![Diagram showing latch timing](image-url)
**Latch timing constraints WITH cycle stealing**

To prevent set-up violations:

\[
 t_{\text{clock}} + t_{\text{clock-high-max}} \geq t_{\text{clock-Q-max}} + t_{\text{logic-max}} + t_{\text{set-up}} + t_{\text{skew}}
\]

**Notes:**
- The percentage of time the clock is high is referred to as the *duty-cycle*
- If part of the following clock-high time is used to allow this logic to be slower, then the logic-block connected to Q2 must be proportionally faster
- Using the clock-high time like this is called *cycle-stealing*
Latch Set Up Violations

Notes:

- The percentage of time the clock is high is referred to as the duty-cycle.

- If part of the following clock-high time is used to allow this logic to be slower, then the logic-block connected to Q2 must be proportionally faster.

- Using the clock-high time like this is called cycle-stealing.

- Normally cycle stealing is not enabled.
Latches … Cycle Stealing

- Can use up to a total of $t_{\text{clock\_high}}$ within a pipeline structure, to help in timing closure
  - Example:

$$t_1, t_2 > t_{\text{clock}} \ (	ext{cycle stealing})$$

What can $t_3$ be?
...Latch timing constraints

To prevent hold violations:

\[ t_{\text{clock-high-max}} + t_{\text{hold}} + t_{\text{skew}} \leq t_{\text{clock-Q-min}} + t_{\text{logic-min}} \]

**Note:**
- Hold violations are harder to prevent in latch-based designs.
Revision – So far

- What is a set-up violation?
  
- How is a set-up violation fixed?

- What is a hold-violation?
  
- How is a hold-violation fixed?

- Why are edge-triggered flip-flops preferred over latches?
Example:

\[
\begin{array}{c}
\text{D-flip-flop} \\
\end{array}
\]
**CMOS Drive Strength**

Revision: CMOS transistors operating in the linear region:

\[ I_{ds} = \beta ((V_{gs} - V_t)V_{ds} - V_{ds}^2 / 2 \]

where \( \beta = (\mu \varepsilon / t_{ox})(W / L) \)

where \( W \) is the transistor width, and \( L \) is the channel length

i.e. To a first approximation,

\[ I_{ds} \approx V_{ds} / R_{on} \]

\[ R_{on} \approx 1 / \beta (V_{GS} - V_T) \]

\[ \tau = R_{on} C_{load} \]

Thus, delay in CMOS circuits depends largely on \( W / L \) of the drive transistor and the capacitance of the load it is driving.

- Different drive cell sizes can be selected by synthesis tool (x1, x2, x3, etc.)
- That capacitance consists of:
  - Input gates of cells being driven, and Capacitance of wiring
Estimating and Improving Performance

- With a focus on timing:
- Topics:
  - Metrics: FO-4
  - Typical timing budgets
  - Pipelining and Parallelism
  - Logic style
Delay Metric

- Usual Metric for delay:
  - Fanout of 4 inverter delay: FO4

- Estimating FO4:
  - Typical \( \sim 360 \times \text{Leff} \) (ps)
  - Worst Case \( \sim 600 \times \text{Leff} \) (ps)
  - Leff = Effective gate length in \( \text{um} \) \( \sim 0.7 \times L_{\text{drawn}} \)
  - E.g. In a 0.18 \( \text{um} \) process, \( \text{Leff} = 0.126 \text{ um} \) and FO-4 \( \leq 75 \text{ ps} \)

- Exemplar delays:
  - Inverter = FO-4
  - 2-input NAND gate = 2 FO4
  - 1-bit adder = 10 FO4
  - 2-input Multiplexer = 4 FO4
  - Flip-flop \( t_{cp-Q} = 4 \) FO4
  - Flip-flop \( t_{su} / t_h = 2 \) FO4
  - Clock skew = 4 FO4
  - Clock jitter = 2 FO4
Examples of Improving Timing Performance

- Example 1: Benefits of Pipelining and Parallelism
- Example:

If \( t_{\text{comparator}} = 20 \) FO4, what is the clock period?
(Use values on previous page)

\[
T_{\text{cp}} = t_{\text{ck}} + t_{\text{logic}} + t_{\text{su}} + t_{\text{skew}} + t_{\text{jitter}}
\]

\[
T_{\text{cp}} = 4 + 20 \times 3 + 2 + 4 + 2 = 72 \text{ FO4}
\]
**Pipelining**

- Replace with:

  ![Diagram of pipelining](image)

  \[ T_{cp} = t_{ck} - Q + t_{logic} + t_{su} + t_{skew} + t_{jitter} \]
  \[ = 4 + 20 + 2 + 4 + 2 = 32 \text{ FO4} \]

- What is the delay improvement?

- What is the drawback?
Logic Level Parallelism

- Replace with:

- Clock Period = 52 FO-4
- No increase in area

To see how to code these three structures in Verilog, please refer to the end of the Verilog1 notes.
Retiming

- Impact of critical paths can often be reduced by retiming or rebalancing a design:

- Example:
  - Before:
    
    \[
    T_{cp} = 4 + 20 + 5 + 2 + 4 + 2 = 37 \text{ FO4}
    \]

  - After:
    
    \[
    T_{cp} = 4 + 20 + 2 + 4 + 2 = 32 \text{ FO4}
    \]

- Note: Clock level logic sequence has been changed
Timing in CAD Flow

- **During synthesis**
  - Tool calculates path delays under worst-case delay conditions
  - Determines critical path
  - Moves logic to a faster path if setup violation predicted
  - Some tools do a preliminary placement while doing logic synthesis so that wiring delay can be properly estimated

- **After synthesis:**
  - Perform Static Timing Analysis
  - Determine no setup violations exist under worst case conditions
  - Determine no hold violations exist under best case conditions

- **After place and route:**
  - Perform Timing Analysis
  - i.e. Run timing verification tools on netlist with actual delays
  - Back-annotate actual delays to netlist from later tools
Initial Delay Estimation Flow

- Primetime: Gate-level static timing analysis tool
  - Report timing for critical path
- Back-annotate wire parasitics for more accuracy
  - SPEF file from place and route tool
Sidebar

Not Examinable!

- What is asynchronous design?
- Can we use deliberate local clock skew in a design?
- Are you sure flip-flops are better, cycle stealing sounds useful?

Message: The CAD tools’ capabilities constrain your design flexibility.
Remember

- Methodology for purposes of ECE 520
  - If at all possible one-edge of one clock
  - If you need multiple clocks, they must have a common root and be related by factors of 2
    - E.g. Root clock: Tclock = 5 ns
    - This is BEST: Tclock only!!
    - This is OK:
      - Tclock, Tclock10, Tclock20
      - Tclock10 = 10 ns (Tclock*2)
      - Tclock20 = 20 ns (Tclock*4)
    - This is NOT OK
      - Tclock, Tclock15, Tclock17
      - Tclock15 = 15 ns (Tclock*3)
      - Tclock17 = 17 ns (??)
Summary

- What determines the maximum clock frequency?

- What is a hold violation?

- Why do we prefer flip-flop designs over using latches?

- What tool is used to check timing at design closure?
Summary

- What determines clock skew?

- Can the designer set the clock skew to be positive or negative?

- Does the logic design have a strong impact on achieving a desired clock speed?