Issues on Timing and Clocking

**Combinational Logic**

- $X \rightarrow \text{FF} \rightarrow Q \rightarrow D \rightarrow Z$
- $\text{clock}$
- $\text{clock period}$
Latch and Flip-Flop

For correct operation of a synchronous circuit:

- The clock period must be longer than the delay of the longest path in the combinational logic.
- The width of the clock pulse must be long enough to allow the flip-flops to change state.
Flip-Flop Setup and Hold Times

- Flip-flop setup time $T_{su}$: the required time the data input signal value must be held stable prior to the arrival of clock pulse.

- Flip-flop hold time $T_{h}$: the required time the data input signal value must be held stable after the arrival of clock pulse.

Flip-flip Timing Parameters

Delays can be different for rising and falling data transitions
Example: For the circuit shown below, assume the delay through the register ($t_{pd}$) is 0.6 and the delay through each logic block is indicated inside the box. Assume that the registers, which are positive edge-triggered, have a set-up time $T_{su}$ of 0.4. What is the minimum clock period?

A Simple RC Model for Logic Gates
Interconnect Models

1. Ignoring interconnects
2. Lumped capacitance model
3. RC tree model
4. RLC tree model
5. Transmission line models (RC, LC, RLC)
6. RC / RLC Network model

Interconnect Models as a Capacitor

1. Ignoring Interconnects:
   \[ V_{out}(t) = V_{DD} (1 - e^{-t/T}) \]
   \[ T = \frac{R_d (C_{input} + C_1 + C_2 + C_3)}{C_{wire}} \]

2. Lumped Capacitance model:
   \[ V_{out}(t) = \frac{1}{2} V_{DD} \]
   \[ V_{out}(T) = 0.632 V_{DD} \]
Analysis of Simple RC Circuit

\[ R \cdot i(t) + v(t) = v_T(t) \]

\[ i(t) = \frac{d(C \cdot v(t))}{dt} = C \frac{dv(t)}{dt} \]

\[ \Rightarrow RC \frac{dv(t)}{dt} + v(t) = v_T(t) \]

First-order linear differential equation with constant coefficients

Zero-input response:
\[ v(t) = v_0 \left( 1 - e^{-\frac{t}{RC}} \right) \]

Step-input response:
\[ v(t) = v_0 u(t) + v_0 \left( 1 - e^{-\frac{t}{RC}} \right) u(t) \]

You can get the same result by Laplace Transform
Delays of Simple RC Circuit

\[ v(t) = v_0(1 - e^{-t/RC}) \quad \text{-- waveform under step input } v_0u(t) \]

\[ v(t) = 0.5v_0 \Rightarrow t = 0.7RC \]
- i.e., delay = 0.7RC \quad (50\% \text{ delay})

\[ v(t) = 0.1v_0 \Rightarrow t = 0.1RC \]
\[ v(t) = 0.9v_0 \Rightarrow t = 2.3RC \]
- i.e., rise time = 2.2RC

Rise time (Fall time): time for a waveform to rise from 10\% to 90\% (90\% to 10\%) of its steady state value

Interconnect Models as a Tree

3. RC tree model
4. RLC tree model
5. Transmission line models (RC, LC, RLC)

L-type
\[ \pi\text{-type} \]
\[ T\text{-type} \]

or transmission line

or transmission line
Determining Which Model to Use

Some Rule-of-Thumbs:

Need to consider C:
- if interconnect C is comparable to C of gates driven

Need to consider R:
- if interconnect R is comparable to R of driver

Need to consider L:
- if \( \omega L \) is comparable to R of interconnect

Model Interconnects as RC Trees

- Each wire maybe segmented into several edges
- Each edge E modeled as a \( \pi \)-type or L-type circuit
  - \( r_E = \) unit res. \( \times \) length(E)
  - \( c_E = \) unit cap. \( \times \) length(E)
Interconnect Delay: Putting All Models Together

- Delay is proportional to the loading capacitance $C_L$.
- Driving more gates result in longer rise/fall time.
- Longer interconnects (larger $R_{int}$ and $C_{in}$) also result in longer rise/fall time.

Clock Tree

- If the # of flip-flops driven by the clock line is large, the clock rise time (also called slew rate) will be unacceptably long.
- Solution: Using clock power-up tree (adding buffers into the clock tree)
Solution: Adding Buffers and Limiting Number of Their Fanouts

- Limiting the number of fanouts of each buffer to N
- Create a buffer tree to drive all flip-flops while satisfying the constraint of fanout count
- The load seen by the clock source is significantly reduced
- The same idea can be used to reduce the delay of logic signals which drive a large number of gates and are on timing-critical paths

An Example

- Assume 64 flip-flops to be driven by a single clock source
- Buffer delay (with zero load): 0.2 ns
- Interconnect delay:
  - 0.2 ns with a single fanout (either to a flip-flop or to a buffer)
  - Addition 0.1 ns delay for each additional fanout

\[
\text{Total delay from clock source to clock ports of flip-flops: 6.9 ns} \\
0.2\text{ns} + 0.2\text{ns} + (0.2\text{ns} + 63 \times 0.1\text{ns}) = 6.9\text{ns}
\]
Example – Clock Tree

- Assume each buffer has four fanouts

![Clock Tree Diagram]

- Total delay from clock source to clock ports of flip-flops: 2.3ns
  
  \[
  0.2\text{ns} \{0^{\text{th}}\text{-level wire delay}\} + 0.2\text{ns} \{1^{\text{st}}\text{-level buffer delay}\} +
  \]
  
  \[
  (0.2\text{ns} + 3 \times 0.1\text{ns}) \{1^{\text{st}}\text{-level wire delay}\} + 0.2\text{ns} \{2^{\text{nd}}\text{-level buffer delay}\} +
  \]
  
  \[
  (0.2\text{ns} + 3 \times 0.1\text{ns}) \{2^{\text{nd}}\text{-level wire delay}\} + 0.2\text{ns} \{3^{\text{rd}}\text{-level buffer delay}\} +
  \]
  
  \[
  (0.2\text{ns} + 3 \times 0.1\text{ns}) \{3^{\text{rd}}\text{-level wire delay}\}
  \]

Techniques for Improving Speed

1. Keep the logic gate depth shallow between flip-flops.
2. Avoid circuit designs that have highly loaded gates in the critical path.
   - A gate delay will increase as the capacitive load is increased on the output of the gate.
   - The primary sources of load capacitance are routing capacitance and the input capacitance of the driven gates.
3. Duplicate logic to reduce fanouts (similar idea to clock tree buffering).
4. Avoid long interconnects
5. Gate sizing
Gate Sizing: Making The Driving Gate Larger (or Smaller)

- Larger the driving gate => Greater the driving current (so sharper $V_{in}$), but larger $C_{input}$ too (which slows down the signal coming into G1)

![Diagram of gate sizing]

Static Timing Analysis 101

- Technology rules, Circuit Library
- Timing specs
- Design netlist
- Clock definitions
- Inputs
- Outputs

(courtesy P. Joshi, IBM)
Static Timing Analysis

Netlist with delay for each gate

Arrival times

slack = required time - arrival time
Clock Non-idealities

- **Clock skew**
  - Spatial variation in temporally equivalent clock edges; deterministic + random, $t_{SK}$

- **Clock jitter**
  - Temporal variations in consecutive edges of the clock signal; modulation + random noise
  - Cycle-to-cycle (short-term) $t_{JS}$
  - Long term $t_{JL}$

- **Variation of the pulse width**
  - Important for level sensitive clocking
Clock Skew

- The delays from the clock source to the clock inputs of different flip-flops are different.

Clock Skew and Jitter

- Both skew and jitter affect the effective cycle time.
- Only skew affects the race margin.
Clock Uncertainties – Sources of Skew and Jitter

- **Power Supply**
- **Interconnect**
- **Capacitive Load**
- **Temperature**
- **Coupling to Adjacent Lines**

**Sources of clock uncertainty**

### The clock skew problem: the race problem

- **Correct operation:** \( D_1 \rightarrow Q_1, \ D_2 \rightarrow Q_2 \)
- **Due to clock skew:** \( D_1 \rightarrow Q_2 \) (error!!)

### Minimizing clock skew:

Distribute the clock signal in such a way that the interconnections from the clock source to the FF’s clock inputs are of equal length.
Clock Distribution

H-Tree Network

Observe: Only Relative Skew is Important

Clock Network with Distributed Buffering

Reduces absolute delay, and makes Power-Down easier
Sensitive to variations in Buffer Delay
Positive and Negative Skew

(a) Positive skew

(b) Negative skew

Positive Skew

Launching edge arrives before the receiving edge
Negative Skew

Receiving edge arrives before the launching edge

“Useful” Clock Skew

- Clock skew is not always bad!!

Example:

- Assume the FF propagation delay (clock to Q delay) $t_{c\rightarrow q}=0.6$, the FF setup time $t_{su}=0.4$, the FF hold time $t_{hd}=0.5$
- If $t_0'-t_0 = 0$ (no clock skew),
  - minimum clock period = 11
If $t_0' - t_0 = \delta = 1$

- For proper operation, the time between positive edges at registers A and B must be greater than or equal to 11
  - clock period + clock skew $\geq 11$
  - minimum clock period = 10

On the other hand, the clock skew cannot exceed 3.1 ns.
Otherwise the data latched into register A may propagate through the short path and reach the data input of register B before the rising edge of the clock pulse of the same cycle reaching $\theta'$. 
Effect of Negative Clock Skew

- If $t_0 - t_0' = 1$

  - For proper operation, the time between positive edges at registers A and B must be greater than or equal to 11.
  - If we define: $t_0' - t_0 = \delta$, $\delta$ would be negative for negative clock skew.
  - For this example, $\delta = -1$; $T + \delta \geq 11$, thus, the minimum clock period $T_{\text{min}} = 12$.

Effect of Negative Clock Skew

- Register B will never get the chance to latch the wrong data no matter how large the negative clock skew is.
- Race problem is not a concern for negative clock skew.
**Summary**

- Clock Period: T
- Longest delay from Reg. A to Reg. B: LD \( A \rightarrow B \)
- Shortest delay from Reg. A to Reg. B: SD \( A \rightarrow B \)
- FF propagation delay, setup time, hold time: \( t_{\text{c} \rightarrow \text{q}}, t_{\text{su}}, t_{\text{hold}} \)
  
  1. \( T + \delta \geq (t_{\text{c} \rightarrow \text{q}} + \text{LD } A \rightarrow B + t_{\text{su}}) \)
  2. \( \delta \leq (t_{\text{c} \rightarrow \text{q}} + \text{SD } A \rightarrow B - t_{\text{hold}}) \)

  Where \( \delta = (t_{\theta}' - t_{\theta}) \)

- Requirement (1) have to be satisfied for datapath between any pair of registers where \( \delta \) would be either positive or negative
- Requirement (2) have to be satisfied for datapath between any pair of registers with a positive clock skew

---

**Example:** Assume \( t_{\text{c} \rightarrow \text{q}} \) is 0.6, \( t_{\text{su}} \) is 0.4 and \( t_{\text{hold}} \) is 0.5.

(a) Determine the minimum clock period assuming a positive clock skew: \( \delta = (t_{\theta}' - t_{\theta}) = 1 \).
(b) Repeat part(a), factoring in a positive clock skew: \( \delta = 3 \).
(c) Repeat part(a), factoring in a negative clock skew: \( \delta = -2 \).
(d) Derive the maximum positive clock skew (i.e. \( t_{\theta}' > t_{\theta} \)) that can be tolerated before the circuit fails.
(e) Derive the maximum negative clock skew (i.e. \( t_{\theta}' < t_{\theta} \)) that can be tolerated before the circuit fails.
Impact of Jitter

Be Very Careful About Gated Clock

An undesirable glitch or spike will result & cause an additional trigger of the clock.
An alternative design:

- This design causes less timing problem but consumes more power.

Clock Gating

- Typically 40-50% of active power is in the IC clock trees
- Clock gating allows some of this power eliminated in active modes
- Root and branch clock gating can be significant

- Resumption of clocking is very fast
- So clock-gated modules can return to “run” mode without loss of services
Fine-grained Clock Gating

always @(posedge CLK) begin
    if (EN)
        Q <= D;
end

Clock Gating Synthesis

always @(posedge clk) begin
    if (sel == 1'b1)
        q0 <= d0;
    if (sel == 1'b0)
        q1 <= d1;
    en <= sel;
end
assign Out = en ? q0:q1;

Ref: Calypto
Sequential Clock Gating

Original RTL

Power Optimized RTL

Advanced Clock Gating (Example)
Improving Clock Gating Efficiency

Clock Gating Efficiency = average percentage of time each register is gated for a given testbench

<table>
<thead>
<tr>
<th>Block 1</th>
<th>Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>38.8%</td>
<td>29%</td>
</tr>
</tbody>
</table>

After identifying more gating opportunities

<table>
<thead>
<tr>
<th>Block 1</th>
<th>Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>54.9%</td>
<td>47%</td>
</tr>
</tbody>
</table>

Source: Calypto

Dealing with Asynchronous Inputs: Synchronizer

- Asynchronous signals incoming to a synchronous system must be synchronized with the rest of the system

SYNCHRONIZER

ASYNCHRONOUS INPUT

D Q

CLK

SYNCHRONIZED SIGNAL

CLK

D Q

METASTABLE STATE

POSSIBLE OUTPUTS
A multiple-stage synchronizer reduces the chance of synchronization failure.

A Timing Optimization Technique - Pipelining

- Pipelining: a technique to break up a timing-critical data path into a series of small data paths by placing registers between sections.
Another Timing Optimization Technique - Retiming

- **Retiming**: a technique to transform a given synchronous circuit into a faster circuit.

**An example: A digital correlator:**
The correlator takes a stream of bits $x_0, x_1, \ldots, x_k$ as input & compares it with a fixed-length pattern $a_0, a_1, \ldots, a_k$. After receiving each input $x_i$, the correlator produces as output the number of matches.

I.e.

$$y_i = \sum_{j=0}^{k} \sigma(x_{i-j}, a_j)$$

where $\sigma(x, y) = \begin{cases} 1 & \text{if } x = y; \\ 0 & \text{otherwise} \end{cases}$

**An implementation for $k=3$**

![Implementation Diagram]

- : register
- : adder
- : comparator

Here, $\sigma(P, Q) = \begin{cases} 0 & P \neq Q; \\ 1 & P = Q \end{cases}$
Suppose each adder has a propagation delay of 7 ns. & each comparator 3 ns.

\[ \Rightarrow \text{The longest propagation delay is 24 ns} \]

\[ \Rightarrow \text{The clock period must be } \geq 24 \text{ ns} . \]

**A better design:**

These two designs are functionally equivalent:

- all input signals to the box portion arrive one clock tick earlier.
- thus, the boxed portion performs the same sequence of computation as the first design, but one clock tick earlier.
- Since the output from the boxed portion is delayed one clock tick by the new register at B, the remainder of the circuit sees the same behavior as in the 1st design.

- The longest propagation delay is reduced to 17 nsec.
- The elements in the boxed portion “lead” by one clock tick.

**Retiming** - the technique of inserting & deleting registers to speed the design while preserve the function.
Retiming Transformation

Example:

Comparison: pipelining
Retiming

Edge label: # of registers

Retiming

Edge label: # of registers