Lecture 9: Sequential Circuit Design

Slides courtesy of Deming Chen

Slides based on the initial set from David Harris
Outline

- Sequencing
- Sequencing Element Design
- Max and Min-Delay
- Clock Skew
- Time Borrowing
- Two-Phase Clocking

- Readings: 10.1-10.3.4
Sequencing

- **Combinational logic**
  - output depends on current inputs

- **Sequential logic**
  - output depends on current and previous inputs
  - Requires separating previous, current, future
  - Called *state* or *tokens*
  - Ex: FSM, pipeline
Sequencing Cont.

- If tokens moved through pipeline at constant speed, no sequencing elements would be necessary.

- Ex: fiber-optic cable
  - Light pulses (tokens) are sent down cable
  - Next pulse sent before first reaches end of cable
  - No need for hardware to separate pulses
  - But dispersion sets min time between pulses

- This is called *wave pipelining* in circuits

- In most circuits, dispersion is high
  - Delay fast tokens so they don’t catch slow ones.
Sequencing Overhead

- Use flip-flops to delay fast tokens so they move through exactly one stage each cycle.
- Inevitably adds some delay to the slow tokens.
- Makes circuit slower than just the logic delay.
  - Called sequencing overhead.
- Some people call this clocking overhead.
  - But it applies to asynchronous circuits too.
  - Inevitable side effect of maintaining sequence.
Sequencing Elements

- **Latch**: Level sensitive
  - a.k.a. transparent latch, D latch
- **Flip-flop**: edge triggered
  - A.k.a. master-slave flip-flop, D flip-flop, D register
- **Timing Diagrams**
  - Transparent
  - Opaque
  - Edge-trigger
Latch Design

- Pass Transistor Latch

- **Pros**
  - + Tiny
  - + Low clock load

- **Cons**
  - - $V_t$ drop
  - - nonrestoring
  - - backdriving
  - - output noise sensitivity
  - - dynamic
  - - diffusion input

- Used in 1970's
Latch Design

- Transmission gate

\[\phi\]

\[D \rightarrow Q\]

\[\phi\]
Latch Design

- Inverting buffer
  +
  +
  + Fixes either
    •
    •
  -
Latch Design

- Tristate feedback

- Static latches are now essential because of leakage
Latch Design

- Buffered input

\[ \text{Buffered input} \]

\[ \phi \]

\[ Q \]

\[ D \]

\[ X \]

\[ \phi \]

\[ \phi \]
Latch Design

- Buffered output
  - +

- Widely used in standard cells
  - + Very robust (most important)
  - - Rather large
  - - Rather slow (1.5 – 2 FO4 delays)
  - - High clock loading
Latch Design

- Datapath latch
  - +
  - +
  - -
Flip-Flop Design

- Flip-flop is built as pair of back-to-back latches

![Flip-flop diagram]
Enable

- Enable: ignore clock when \( en = 0 \)
  - Mux: increase latch D-Q delay
  - Clock Gating: increase en setup time, skew
Reset

- Force output low when reset asserted
- Synchronous vs. asynchronous

Symbol

**Synchronous Reset**

**Asynchronous Reset**
Set / Reset

- Set forces output high when enabled

- Flip-flop with asynchronous set and reset

![Flip-flop circuit diagram with set and reset inputs]
Sequencing Methods

- Flip-flops
- 2-Phase Latches
- Pulsed Latches
# Timing Diagrams

Contamination and Propagation Delays

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{pd}$</td>
<td>Logic Prop. Delay</td>
</tr>
<tr>
<td>$t_{cd}$</td>
<td>Logic Cont. Delay</td>
</tr>
<tr>
<td>$t_{pcq}$</td>
<td>Latch/Flop Clk-&gt;Q Prop. Delay</td>
</tr>
<tr>
<td>$t_{ccq}$</td>
<td>Latch/Flop Clk-&gt;Q Cont. Delay</td>
</tr>
<tr>
<td>$t_{pdq}$</td>
<td>Latch D-&gt;Q Prop. Delay</td>
</tr>
<tr>
<td>$t_{cdq}$</td>
<td>Latch D-&gt;Q Cont. Delay</td>
</tr>
<tr>
<td>$t_{setup}$</td>
<td>Latch/Flop Setup Time</td>
</tr>
<tr>
<td>$t_{hold}$</td>
<td>Latch/Flop Hold Time</td>
</tr>
</tbody>
</table>
Max-Delay: Flip-Flops

\[ t_{pd} \leq T_c - (\text{sequencing overhead}) \]

Combinational Logic

\[ t_{pd} \leq T_c - (\text{sequencing overhead}) \]

\[ t_{pcq} + t_{pd} + t_{setup} \leq T_c \]
Max Delay: 2-Phase Latches

\[ t_{pd} = t_{pd1} + t_{pd2} \leq T_c - \left( \text{sequencing overhead} \right) \]

\[ t_{pd} = t_{pd1} + t_{pd2} \leq T_c - \left( \text{sequencing overhead} \right) \]

\[ \frac{1}{2} \left( t_{pd1} + t_{pd2} \right) \leq \frac{T_c}{2} - \left( \text{sequencing overhead} \right) \]
Max Delay: Pulsed Latches

\[ t_{pd} \leq T_c - \max\left(\text{sequencing overhead}\right) \]

(a) \( t_{pw} > t_{\text{setup}} \)

(b) \( t_{pw} < t_{\text{setup}} \)

Sequential Circuits

CMOS VLSI Design 4th Ed.
Min-Delay: Flip-Flops

\[ t_{cd} \geq \]

Sequential Circuits
CMOS VLSI Design 4th Ed.
Min-Delay: 2-Phase Latches

\[ t_{cd1}, t_{cd2} \geq \]

Hold time reduced by nonoverlap

Paradox: hold applies twice each cycle, vs. only once for flops.

But a flop is made of two latches!
Min-Delay: Pulsed Latches

$t_{cd} \geq$

Hold time increased by pulse width
Time Borrowing

- In a flop-based system:
  - Data launches on one rising edge
  - Must setup before next rising edge
  - If it arrives late, system fails
  - If it arrives early, time is wasted
  - Flops have hard edges

- In a latch-based system
  - Data can pass through latch while transparent
  - Long cycle of logic can borrow time into next
  - As long as each loop completes in one cycle
Time Borrowing Example

(a) Latch → Combinational Logic → Latch
   Borrowing time across half-cycle boundary

(b) Latch → Combinational Logic → Latch
   Borrowing time across pipeline stage boundary

Loops may borrow time internally but must complete within the cycle
How Much Borrowing?

2-Phase Latches

\[ t_{\text{borrow}} \leq \frac{T_c}{2} - (t_{\text{setup}} + t_{\text{nonoverlap}}) \]

Pulsed Latches

\[ t_{\text{borrow}} \leq t_{\text{pw}} - t_{\text{setup}} \]
Clock Skew

- We have assumed zero clock skew
- Clocks really have uncertainty in arrival time
  - Decreases maximum propagation delay
  - Increases minimum contamination delay
  - Decreases time borrowing
Skew: Flip-Flops

\[
t_{pd} \leq T_c - (t_{pcq} + t_{\text{setup}} + t_{\text{skew}})
\]

sequencing overhead

\[
t_{cd} \geq t_{\text{hold}} - t_{ccq} + t_{\text{skew}}
\]
Skew: Latches

2-Phase Latches

\[ t_{pd} \leq T_c - \left( 2t_{pdq} \right) \]

\[ t_{cd1}, t_{cd2} \geq t_{hold} - t_{ccq} - t_{nonoverlap} + t_{skew} \]

\[ t_{borrow} \leq \frac{T_c}{2} - \left( t_{setup} + t_{nonoverlap} + t_{skew} \right) \]

Pulsed Latches

\[ t_{pd} \leq T_c - \max \left( t_{pdq}, t_{pcq} + t_{setup} - t_{pw} + t_{skew} \right) \]

\[ t_{cd} \geq t_{hold} + t_{pw} - t_{ccq} + t_{skew} \]

\[ t_{borrow} \leq t_{pw} - \left( t_{setup} + t_{skew} \right) \]
Two-Phase Clocking

- If setup times are violated, reduce clock speed
- If hold times are violated, chip fails at any speed
- In this class, working chips are most important
- An easy way to guarantee hold times is to use 2-phase latches with big nonoverlap times
- Call these clocks $\phi_1$, $\phi_2$ (ph1, ph2)
Safe Flip-Flop

- Past years used flip-flop with nonoverlapping clocks
  - Slow – nonoverlap adds to setup time
  - But no hold times
- In industry, use a better timing analyzer
  - Add buffers to slow signals if hold time is at risk
Adaptive Sequencing

- Designers include timing margin
  - Voltage
  - Temperature
  - Process variation
  - Data dependency
  - Tool inaccuracies

- Alternative: run faster and check for near failures
  - Idea introduced as “Razor”
    - Increase frequency until at the verge of error
    - Can reduce cycle time by \(~30\%)
Summary

- **Flip-Flops:**
  - Very easy to use, supported by all tools

- **2-Phase Transparent Latches:**
  - Lots of skew tolerance and time borrowing

- **Pulsed Latches:**
  - Fast, some skew tol & borrow, hold time risk

<table>
<thead>
<tr>
<th></th>
<th>Sequencing overhead ((T_c - t_{pd}))</th>
<th>Minimum logic delay (t_{cd})</th>
<th>Time borrowing (t_{borrow})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flip-Flops</td>
<td>(t_{peq} + t_{setup} + t_{skew})</td>
<td>(t_{hold} - t_{eq} + t_{skew})</td>
<td>0</td>
</tr>
<tr>
<td>Two-Phase Transparent Latches</td>
<td>(2t_{pdq})</td>
<td>(t_{hold} - t_{eq} - t_{nonoverlap} + t_{skew}) in each half-cycle</td>
<td>(\frac{T_c}{2} - (t_{setup} + t_{nonoverlap} + t_{skew}))</td>
</tr>
<tr>
<td>Pulsed Latches</td>
<td>(\text{max}(t_{pdq}, t_{peq} + t_{setup} - t_{pev} + t_{skew}))</td>
<td>(t_{hold} - t_{eq} + t_{pev} + t_{skew})</td>
<td>(t_{pev} - (t_{setup} + t_{skew}))</td>
</tr>
</tbody>
</table>
Next Lecture

- Wires
  - Readings: 6.1-6.2.2; 6.3.1-6.3.3; 6.4.1-6.4.2