Sequential Circuits

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
  - No $V_t$ drop
  - Requires inverted clock
Latch Design

- Inverting buffer
  - Restoring
  - No backdriving
  - Fixes either
    - Output noise sensitivity
    - Or diffusion input
  - Inverted output
Latch Design

- Tristate feedback
  
  + Static
  - Backdriving risk

- Static latches are now essential
Latch Design

- Buffered input
  - +
  - +

Latch Design

- Buffered input
  - + Fixes diffusion input
  - + Noninverting
Latch Design

- Buffered output
  - No backdriving

- 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
  - Smaller, faster
  - Unbuffered input
Metastability

- **Stable**: $A = B = 0$
- **Metastable**: $A = B = V_m$
- **Flip-Flop Design**: Flip-flop is built as pair of back-to-back latches

Flip-Flop Design

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

Diagram showing the metastability condition and flip-flop design.
Enable

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Multiplexer Design</th>
<th>Clock Gating Design</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Image" alt="Symbol Diagram" /></td>
<td><img src="Image" alt="Multiplexer Diagram" /></td>
<td><img src="Image" alt="Clock Gating Diagram" /></td>
</tr>
</tbody>
</table>

Reset

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Diagram 1</th>
<th>Diagram 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Image" alt="Symbol Diagram" /></td>
<td><img src="Image" alt="Diagram 1" /></td>
<td><img src="Image" alt="Diagram 2" /></td>
</tr>
</tbody>
</table>
Set / Reset

- Set forces output high when enabled
- Flip-flop with asynchronous set and reset

Sequencing Methods

- Flip-flops
- 2-Phase Latches
Timing Diagrams

Contamination and Propagation Delays

<table>
<thead>
<tr>
<th>$t_{pd}$</th>
<th>Logic Prop. Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{cd}$</td>
<td>Logic Cont. Delay</td>
</tr>
<tr>
<td>$t_{pcq}$</td>
<td>Latch/Flop Clk-$\rightarrow$Q Prop. Delay</td>
</tr>
<tr>
<td>$t_{ccq}$</td>
<td>Latch/Flop Clk-$\rightarrow$Q Cont. Delay</td>
</tr>
<tr>
<td>$t_{pdq}$</td>
<td>Latch D-$\rightarrow$Q Prop. Delay</td>
</tr>
<tr>
<td>$t_{cdq}$</td>
<td>Latch D-$\rightarrow$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

$t_{pd} < T_c - (t_{setup} + t_{pcq})$

sequencing overhead
Max Delay: 2-Phase Latches

\[ t_{pd} = t_{pd1} + t_{pd2} \]
\[ < T_c - 2t_{pdq} \]

\text{sequencing overhead}

Min-Delay: Flip-Flops

\[ t_{cd} \geq \]
Min-Delay: Flip-Flops

\[ t_{cd} \geq t_{hold} - t_{ccq} \]

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: 2-Phase Latches

\[ t_{cd1}, t_{cd2} \geq t_{\text{hold}} - t_{eq} - t_{\text{nonoverlap}} \]

Hold time reduced by nonoverlap

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

But a flop is made of two latches!

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
Borrowing time across pipeline stage boundary

(b) Latch → Combinational Logic → Latch

Loops may borrow time internally but must complete within the cycle

How Much Borrowing?

2-Phase Latches

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

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

Skew: Flip-Flops

\[ t_{pd} < T_c - (t_{setup} + t_{pcq} + t_{skew}) \]
sequencing overhead

\[ t_{ca} > t_{hold} - t_{ccq} + t_{skew} \]
Skew: 2-Phase Latches

\[ t_{pd} = t_{pd1} + t_{pd2} < T_c - 2t_{pdq} \]

sequencing overhead: not influenced by skew

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

\[ t_{borrow} < T_c / 2 - (t_{setup} + t_{nonoverlap} + t_{skew}) \]

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
  - No tools to analyze clock skew
- An easy way to guarantee hold times is to use 2-phase latches with big nonoverlap times
- Call these clocks \( \phi_1, \phi_2 (\text{ph1, ph2}) \)
<table>
<thead>
<tr>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flip-Flops:</strong></td>
</tr>
<tr>
<td>‣ Very easy to use, supported by all tools</td>
</tr>
<tr>
<td><strong>2-Phase Transparent Latches:</strong></td>
</tr>
<tr>
<td>‣ Lots of skew tolerance and time borrowing</td>
</tr>
</tbody>
</table>