Review: The Regenerative Property

If the gain in the transient region is larger than 1, only A and B are stable operation points. C is a metastable operation point.
Bistable Circuits

- The cross-coupling of two inverters results in a bistable circuit (a circuit with two stable states).
- Have to be able to change the stored value by making A (or B) temporarily unstable by increasing the loop gain to a value larger than 1.
  - done by applying a trigger pulse at $V_{i1}$ or $V_{i2}$
  - the width of the trigger pulse need be only a little larger than the total propagation delay around the loop circuit (twice the delay of an inverter)
- Two approaches used
  - cutting the feedback loop (mux based latch)
  - overpowering the feedback loop (as used in SRAMs)

Review: SR Latch

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>!Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q</td>
<td>!Q</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Review: Clocked D Latch

- **Clock**: The clock signal controls the operation of the latch.
- **Q**: The output of the D latch.
- **D**: The input of the D latch.
- **Hold Mode**:
  - **Transparent when the clock is low**
  - **Feedback**
- **Transparent Mode**:
  - **Hold when the clock is high**

**MUX Based Latches**

- Change the stored value by cutting the feedback loop

**Negative Latch**

- **Q = clk & Q | !clk & D**
- **Transparent when the clock is low**

**Positive Latch**

- **Q = !clk & Q | clk & D**
- **Transparent when the clock is high**
TG MUX Based Latch Implementation

- Reduced clock load, but threshold drop at output of pass transistors so reduced noise margins and performance

PT MUX Based Latch Implementation
Latch Race Problem

Two-sided clock constraint

\[ T \geq t_{c-q} + t_{\text{logic}} + t_{su} \]

\[ T_{\text{high}} < t_{c-q} + t_{\text{cdlogic}} \]

Master Slave Based ET Flipflop

\begin{align*}
\text{clk} = 0 & \quad \text{transparent} \\
\text{clk} = 0 \rightarrow 1 & \quad \text{hold}
\end{align*}
MS ET Implementation

Master

Slave

clk

Q

D

QM

I1

I2

I3

I4

I5

I6

T2

T1

T4

T3

T1

T2

T3

T4

 clk

!clk

master transparent

slave hold

master hold

slave transparent
MS ET Timing Properties

• Assume propagation delays are $t_{pd_{inv}}$ and $t_{pd_{tx}}$, that the contamination delay is 0, and that the inverter delay to derive $!clk$ is 0
• **Set-up time** - time before rising edge of clk that D must be valid

• **Propagation delay** - time for $Q_M$ to reach Q

• **Hold time** - time D must be stable after rising edge of clk

\[ 3 \times t_{pd_{inv}} + t_{pd_{tx}} \]

\[ t_{pd_{inv}} + t_{pd_{tx}} \]
Set-up Time Simulation

\[ t_{\text{setup}} = 0.21 \text{ ns} \]

works correctly

Set-up Time Simulation

\[ t_{\text{setup}} = 0.20 \text{ ns} \]

fails
Propagation Delay Simulation

\[ t_{c-q(LH)} = 160 \text{ psec} \]
\[ t_{c-q(HL)} = 180 \text{ psec} \]

Reduced Load MS ET FF

- Clock load per register is important since it directly impacts the power dissipation of the clock network.
- Can reduce the clock load (at the cost of robustness) by making the circuit ratioed
  - to switch the state of the master, \( T_1 \) must be sized to overpower \( I_2 \)
  - to avoid reverse conduction, \( I_4 \) must be weaker than \( I_1 \)
Non-Ideal Clocks

Ideal clocks

Non-ideal clocks
clock skew

1-1 overlap

0-0 overlap

Example of Clock Skew Problems

Race condition – direct path from D to Q during the short time when both clk and !clk are high (1-1 overlap)

Undefined state – both B and D are driving A when clk and !clk are both high

Dynamic storage – when clk and !clk are both low (0-0 overlap)
Pseudostatic Two-Phase ET FF

Two Phase Clock Generator
Power PC Flipflop

 clk
 1 D → 0
 !clk

 clk

 Power PC Flipflop

 clk
 1 D → 0
 !clk

 clk

 !clk

 master transparent
 slave hold

 master hold
 slave transparent
Ratioed CMOS Clocked SR Latch

Diagram showing the operation of a Ratioed CMOS Clocked SR Latch with transistors M1, M2, M3, M4, M5, M6, M7, and M8.
Sizing Issues

So $W/L_{5_{and}6} > 3$

$W/L_{2_{and}4} = 1.5 \mu m/0.25 \mu m$

$W/L_{1_{and}3} = 0.5 \mu m/0.25 \mu m$

Transient Response

$Q$ & $\bar{Q}$ (Volts)

$SET$

$I_Q$

$Q$

$t_{o-IQ}$

$t_{Q}$

$0.9$ $1$ $1.1$ $1.2$ $1.3$ $1.4$ $1.5$

Time (ns)
6 Transistor CMOS SR Latch

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
Sequential Logic

Combinational Logic

Inputs → Outputs

Current State → Next State

Inputs

Outputs

Clock

Current State

Next State

State Registers

Timing Metrics

In

Out

Clock

t_{su}

t_{hold}

data stable

output stable

t_{b-q}

Output

Input
System Timing Constraints

\[ t_{cdreg} + t_{cdlogic} \geq t_{hold} \]
\[ T \geq t_{c-q} + t_{plogic} + t_{su} \]

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
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**
  +
  +
- **Cons**
  –
  –
  –
  –
  –
  –
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
  +
  -

11/7/2005
VLSI Design I; A. Milenkovic

11/7/2005
VLSI Design I; A. Milenkovic
Latch Design

• Transmission gate
  + No $V_t$ drop
  - Requires inverted clock

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

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

Latch Design

- Tristate feedback
  - 
    - 

\[ \text{D} \quad \phi \quad X \quad \phi \quad Q \]
\[ \text{D} \quad \phi \quad \phi \quad Q \]

11/7/2005  VLSI Design I; A. Milenkovic  43

11/7/2005  VLSI Design I; A. Milenkovic  44
Latch Design

- Tristate feedback
  + Static
  - Backdriving risk

- Static latches are now essential

Latch Design

- Buffered input
  +
  +

11/7/2005 VLSI Design I; A. Milenkovic 45
Latch Design

- Buffered input
  + Fixes diffusion input
  + Noninverting

Latch Design

- Buffered output
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
  +
  -
Latch Design

- Datapath latch
  - Smaller, faster
  - Unbuffered input

Flip-Flop Design

- Flip-flop is built as pair of back-to-back latches
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
Set / Reset

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

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_{pd}</td>
<td>Logic Cont. Delay</td>
</tr>
<tr>
<td>t_{pcq}</td>
<td>Latch/Flop Clk-Q Prop Delay</td>
</tr>
<tr>
<td>t_{ccq}</td>
<td>Latch/Flop Clk-Q Cont. Delay</td>
</tr>
<tr>
<td>t_{pdq}</td>
<td>Latch D-Q Prop Delay</td>
</tr>
<tr>
<td>t_{pq}</td>
<td>Latch D-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

\[ t_{pd} \leq T_e - \left( \frac{\text{sequencing overhead}}{\text{clock period}} \right) \]
Max-Delay: Flip-Flops

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

Max Delay: 2-Phase Latches

\[ t_{pd} = t_{pd1} + t_{pd2} \leq T_c - (t_{setup} + t_{pcq}) \]
Max Delay: 2-Phase Latches

\[ t_{pd} = t_{pd1} + t_{pd2} \leq T_c - \left( \frac{2t_{pd}}{seq\text{-}overhead} \right) \]

Max Delay: Pulsed Latches

\[ t_{pd} \leq T_c - \max \left( \frac{2t_{pd}}{seq\text{-}overhead} \right) \]
Max Delay: Pulsed Latches

\[ t_{pd} \leq T_c - \max \left( \frac{t_{pdq} t_{pdy} + t_{setup} - t_{pu}}{\text{sequencing overhead}} \right) \]

Min-Delay: Flip-Flops

\[ t_{cf} \geq \]

11/7/2005  VLSI Design I; A. Milenkovic
Min-Delay: Flip-Flops

\[ t_{\text{req}} \geq t_{\text{hold}} - t_{\text{eqg}} \]

Min-Delay: 2-Phase Latches

\[ t_{\text{req1}}, t_{\text{req2}} \geq \]

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

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

Hold time reduced by nonoverlap
Paradox: hold applies twice each cycle, vs. only once for flops.
But a flop is made of two latches!

Min-Delay: Pulsed Latches

\[ t_{cd} \geq \]

Hold time increased by pulse width
Min-Delay: Pulsed Latches

\[ t_{\text{ref}} \geq t_{\text{hold}} - t_{\text{req}} + t_{\text{pu}} \]

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

Loops may borrow time internally but must complete within the cycle

(a) Borrowing time across half-cycle boundary
(b) Borrowing time across pipeline stage boundary

How Much Borrowing?

2-Phase Latches
\[ t_{\text{borrow}} \leq \frac{T}{2} - \left( t_{\text{setup}} + t_{\text{nonoverlap}} \right) \]

Pulsed Latches
\[ t_{\text{borrow}} \leq t_{\text{pu}} - t_{\text{setup}} \]
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

\[ t_{pd} \leq T_c - \left( t_{pq} + t_{atmp} + t_{skew} \right) \]

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

2-Phase Latches

\[ t_{pd} \leq T_c - \left( \frac{2r_{pd}}{\text{sequencing overhead}} \right) \]

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

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

Pulsed Latches

\[ t_{pd} \leq T_c - \max \left( t_{pd1}, t_{pd2} + t_{\text{setup}} - t_{\text{pre}} + t_{\text{skew}} \right) \]

\[ t_{cd} \geq t_{\text{hold}} + t_{\text{pre}} - t_{cq} + t_{\text{skew}} \]

\[ t_{\text{hsetup}} \leq t_{\text{pre}} - (t_{\text{setup}} + t_{\text{skew}}) \]

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 \( \phi_1, \phi_2 \) (ph1, ph2)
Safe Flip-Flop

- In class, use flip-flop with nonoverlapping clocks
  - Very 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

\[ D \quad \bar{\phi}_2 \quad \bar{\phi}_1 \quad \bar{\phi}_1 \quad \bar{\phi}_1 \quad \bar{\phi}_2 \quad \bar{\phi}_2 \quad \bar{\phi}_2 \]

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_s)</th>
<th>Minimum logic delay (T_{\text{delay}})</th>
<th>Time borrowing (T_{\text{borrow}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flip-Flops</td>
<td>(f_{\text{prop}} + f_{\text{comp}} + f_{\text{clk}})</td>
<td>(T_{\text{hold}} - f_{\text{prop}} + f_{\text{clk}})</td>
<td>0</td>
</tr>
<tr>
<td>Two-Phase Transparent Latches</td>
<td>(2f_{\text{prop}})</td>
<td>(T_{\text{hold}} - f_{\text{prop}} - f_{\text{comp}} + f_{\text{clk}}) in each half-cycle</td>
<td>(\frac{T_{\text{hold}}}{2} - (f_{\text{comp}} + f_{\text{prop}} + f_{\text{clk}}))</td>
</tr>
</tbody>
</table>
| Pulsed Latches       | \(\max(f_{\text{prop}}, f_{\text{prop}} - f_{\text{prop}} + f_{\text{clk}})\) | \(T_{\text{hold}} - f_{\text{prop}} + f_{\text{comp}} + f_{\text{clk}}\) | \(f_{\text{prop}} - (f_{\text{comp}} + f_{\text{clk}})\)

11/7/2005 VLSI Design I; A. Milenkovic