Review: Pass Transistor Circuits

- Use pass transistors like switches to do logic
- Inputs drive diffusion terminals as well as gates

- CMOS + Transmission Gates:
  - 2-input multiplexer
  - Gates should be restoring

TG Multiplexer

Transmission Gate XOR

Transmission Gate XOR

TG Full Adder
**Differential TG Logic (DPL)**

- **AND/NAND**: $F = A \oplus B$
- **XOR/XNOR**: $F = AB$

**Complementary Pass-transistor Logic (CPL)**

- **Dual-rail form of pass transistor logic**
- **Avoids need for ratioed feedback**
- **Optional cross-coupling for rail-to-rail swing**

**CPL Properties**

- **Differential** so complementary data inputs and outputs are always available (so don't need extra inverters)
- **Still static**, since the output defining nodes are always tied to $V_{DD}$ or GND through a low resistance path
- **Design is modular**, all gates use the same topology, only the inputs are permuted.
- Simple XOR makes it attractive for structures like *adders*
- Fast (assuming number of transistors in series is small)
- Additional routing overhead for complementary signals
- Still have static power dissipation problems

**CPL Full Adder**
Sequential Circuits

Seqeuncing

- **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

Timing Metrics
System Timing Constraints

- **Combinational Logic**
  - Inputs
  - Outputs

- **State Register**
  - Current State
  - Next State
  - Clock
  - *T* (clock period)

- **Outputs**
  - \( \geq t_{\text{c-q}} + t_{\text{logic}} + t_{\text{sutcdreg}} + t_{\text{cdlogic}} \)
  - \( \geq t_{\text{hold}} \)

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

- **Transmission gate**
Latch Design

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

Latch Design

- 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
  +
  -

Latch Design

- Tristate feedback
  +
  -

Latch Design

- Buffered input
  +
  +

Latch Design

- Buffered input
  +
  +

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

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

```
Latch DQ
φ
en

Enable Multiplexer Design Clock Gating Design
```

Reset

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

```
Latch DQ
φ
en

Reset
```

Set / Reset

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

```
Set / Reset

Set
Reset
```

Sequencing Methods

- Flip-flops
- 2-Phase Latches
- Pulsed Latches

```
Sequencing Methods

Flip-flops
2-Phase Latches
Pulsed Latches
```

Timing Diagrams

Contamination and Propagation Delays

```
Timing Diagrams
Contamination and Propagation Delays
```

Max-Delay: Flip-Flops

```
Max-Delay: Flip-Flops
```

VLSI Design I; A. Milenkovic
Max-Delay: Flip-Flops

\[ t_{ud} \leq T_c = \left( t_{\text{setup}} + t_{\text{pdc}} \right) \]

Max Delay: 2-Phase Latches

\[ t_{ud} \leq T_c = \left( t_{\text{setup}} + t_{\text{pdc}} \right) \]

Max Delay: Pulsed Latches

\[ t_{ud} \leq T_c = \max \left( t_{\text{tsetup}} + t_{\text{tpd}} + t_{\text{tpdq}} \right) \]

Min-Delay: Flip-Flops

\[ t_{ud} \geq C_{\text{L}} \]
Min-Delay: Flip-Flops

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

Min-Delay: 2-Phase Latches

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

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

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

Hold time increased by pulse width

Min-Delay: Pulsed Latches

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

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

Latch
Latch
Latch
Combinational Logic

Borrowing time across half-cycle boundary
Borrowing time across pipeline stage boundary

Loops may borrow time internally but must complete within the cycle

How Much Borrowing?

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

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

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

\[ t_{\text{prop}} \leq T - \left( t_{\text{setup}} + t_{\text{nonoverlap}} \right) \]

Skew: Latches

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

Pulsed Latches
\[ t_{\text{borrow}} \leq \max \left( t_{\text{setup}} - t_{\text{prop}}, t_{\text{setup}} - t_{\text{hold}} \right) \]

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
  - 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

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