Announcements

- **Homeworks**
  - Hw #2 will be graded by Tue night
  - Hw #3 solutions, online by Tue noon
  - Hw #4 posted online (due Fri, Nov 21)

- **Lectures this week**
  - Due to holiday on Tue, video of Mon lec will be available on Wed
  - Wed lecture: Logic synthesis (by Sourabh Tandon / Synopsys Inc.)

- **Office hours this week**
  - Monday: 4:30pm – 6:00pm (business as usual)
  - Wed: no office hours (out of town / ICCAD)
    - Wed OH moves to Friday 4:00pm – 5:30pm
Today

- **Midterm**
  - Avg = 60%
  - Std.dev = 13%
  - Max = 90% / Min = 30%

- **Class project**
  - Description online
  - Team sign-up: classwiki

- **Timing analysis**
  - How to make flip-flops
    - Delay and power analysis
  - How to do timing analysis
    - Datapath logic + F/F’s

---

### Two Types of Machines with State (Storage)

Two quite different abstract models:

**Data storage used for computation (Data Flows)**

- In this abstraction, the storage is used to hold data that is being manipulated. In this model the number of bits of state can be enormous, but it does not matter. It is simply the data-set that is being manipulated.
- State is not that important, it is the flow of data that is critical.

**States for sequencing information (Finite State Machines)**

- In this abstraction, the storage is used to hold your place in some decision making process. It indicates where you are, and using this information you decide what to do next.
- The amount of state (number of unique decision points) is finite, and usually limited. One could think about drawing out the ‘decision graph’ showing the possible transitions between states.
**Brief Introduction to Clocking**

- Means to synchronize
  - By allowing events to happen at known timing boundaries, we can sequence these events
- Greatly eases building of FSMs
  - Clock strobe indicates the moment when states are stored
- No need to worry about variable delay through the CL
  - All signals are delayed until the clock edge (clock imposes the worst case delay)

**Positive Feedback: Bi-Stability**

*State storage…*
**Meta-Stability**

Gain should be larger than 1 in the transition region.

**Cross-Coupled Pairs**

NOR-based set-reset

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>Q</th>
<th>(\overline{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q</td>
<td>(\overline{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>

Forbidden State

The “Overpowering” Approach
Writing into a Static Latch

Use the clock as a decoupling signal, that distinguishes between the transparent and opaque states.

Converting into a MUX (static)

Forcing the state (pseudo-static)

Storage Mechanisms

Static

Dynamic

Dynamic mode
**Pseudo-Static Latch**

Dynamic $\xrightarrow{\text{clock}}$ Pseudo-static

Storage Elements – Static Latch

- **Latches**
  - Level sensitive – transparent when H, hold when L
  - Build with pass transistors.
    - Pass D-to-Q when clock is HIGH
    - Connect Q back to itself when clock is LOW
  - (All nodes are driven at all times)
  - Sizing of feedback portion small
    - Do not need big devices to hold data
Transmission Gate Latches

- Simplest implementation
  - only 4 transistors
  - Dynamic when S=1
  - Susceptible to noise

- Basic static latch
  - pull-up/pull-down keeper
  - Conflict at node S whenever new data is written

- Complete implementation
  - Feedback turned off when writing to the latch
  - No conflict
  - Larger clock load

Oklobdzija et al., Wiley 2003

Principal Ways to Build a Register

Ways to design an edge-triggered sequential cell:

**Master-Slave Latches**

**Pulse-Triggered Latch**
Storage Element – Static Flip-Flop

- Master-Slave Flip-flops
  - Edge-triggered – data is sampled at the clock edge

Use 2 latches
- Master passes D-to-Pb when clock is low.
- Slave passes data from Pb to Q when clock is high.
  - Data does not pass D-to-Pb when clock is high.
  - Data that arrives just before the L-H clock transition is at the output.

Clocked Inverter

- Another common element for building storage elements is a clocked-inverter.
  - Not too different from an inverter+pass-gate.
  - Eliminates the extra contact spacing between NMOS and PMOS so it is quite compact
  - But does not pull down/up with both N/P
Static-Latch Layout with Clocked Inverter

- Feedback using minimum size

More Robust Latching Structures

- Be careful with using a pass-gate at the input
  - If we don’t know where the input is coming from… it can be a value lower than ground or higher than Vdd
  - Cause dc current even when pass-gate should be off
- If the output is driven to noisy circuits
  - Disturb the stored node… better if we tap the output separately
Embedded Logic into F/F

- Embed logic into storage elements
  - A 2:1 multiplexer by using clocks that are gated with sel
    - $CkS_1 = ck \& sel_1$; $CkS_{1b} = (ck \& sel_1)'$
    - $CkS_2 = ck \& sel_2$; $CkS_{2b} = (ck \& sel_2)'$
      - If 2:1 mux then sel2 can be sel1'
      - Instead of inverter, use simple logic gates

---

Enabled Flip-Flop (or Latch)

- New data is accepted only when Enable is H
- Otherwise retain old data

- Conceptually, a mux before the F/F. Can use a built-in mux
- Or we can use a “qualified” clock
**Synchronous Reset**

- Some flip-flops or latches embed very useful functions to help its operation
- Synchronous reset assumes that the reset (output L) occurs with the clock transition

**Asynchronous Reset**

- The output is cleared to Low whenever a reset signal arrives
- Need to clear the held value (state)
- A Set/Reset can also be made using feedback NAND gates
C\textsuperscript{2}MOS Flip-Flop

- Uses clocked inverter

One nice feature of this F/F is that it is not sensitive to \( \phi, \phi_b \) skew.

Skew of C\textsuperscript{2}MOS Flip-flop

- Skew of clock and clockb can cause both signals to be HIGH or LOW simultaneously
- Data of pass-transistor F/F could pass through without being latched

- C\textsuperscript{2}MOS F/F will have both pull-up ON or both pull-down ON, but data won’t transition through

\[ \text{over(\text{up}) : 0-0 : pull } \]
\[ \text{1-1 : pull} \]
Charge Injection

- A problem with dynamic nodes is that it is high-impedance and therefore very sensitive to noise.
- Clock/Data switching can easily introduce charge.
  - Often known as clock feedthrough.

\[
\Delta V_{\text{out}} = \Delta V_{\text{in}} \frac{C_{\text{coupl}}}{C_{\text{coupl}} + C_{\text{out}}}
\]

Not a big concern for this F/F, but it can be for other designs.

Charge Sharing

- Similar to injection, charge on floating nodes can share (depending on the gate voltage) degrading the value of the floating nodes.
- Example:

\[
Q_{\text{before}} = V_{\text{dd}} C_p + V_{\text{GND}} C_{\text{out}} = Q_{\text{after}} = V_{\text{final}} C_p + V_{\text{final}} C_{\text{out}}
\]

Be careful because if \( C_{\text{out}} >> C_p, V_{CP} \ll -V_{TP} \).

Note that C^MOS inverter is not designed like this... for this reason.
Summary

- Clocks are a convenient constraint to simplify system design
  - Essentially forces delay of all logic paths to equal $T_{cycle}$

- Storage elements can be built by either (1) using positive feedback to keep the value or (2) using capacitance to dynamically store the value
  - Dynamic elements are faster but are less robust

- Dynamic storage is sensitive to noise
  - Particularly charge injection and charge sharing

Characterizing Timing

- Register
- Latch
Timing Definitions

Clk-Q Delay
Setup Time

(a) $T_{\text{setup}} = 0.21\text{ns}$  
(b) $T_{\text{setup}} = 0.20\text{ns}$

More Precise Setup Time

$C_{\text{L}}$  
$D$  
$Q$

(a)  
(b)
Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)

Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)
Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)

Data \( T_{\text{Setup}-1} \) Clock

\[ t=0 \]
Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)

Setup Time as a Function of Data-to-Clock Delay

When D-to-Q delay is observed, we can see that data can come closer
to the triggering event than we thought.
Data-to-Q is the RELEVANT parameter, NOT Clock-to-Q as many think!
Data-to-Output Delay

- Sum of setup time and Clk-Q delay is the only true measure of performance w.r.t. system speed
- \( T = T_{\text{Clk-Q}} + T_{\text{Logic}} + T_{\text{setup}} + T_{\text{skew}} \)

Setup-Hold Time Illustrations

**Hold-1 case**

- \( D \) to \( Q \) path
- \( T_{\text{Clk-Q}} \) delay
- \( T_{\text{Hold-1}} \) delay
- Clock and Data signals
Setup-Hold Time Illustrations

Hold-1 case

Clock

Data

Time

Setup-Hold Time Illustrations

Hold-1 case

Clock

Data

Time
Setup-Hold Time Illustrations

Hold-1 case

Clock Data

Hold-1 case

Clock Data

⇒
**Clk-Q Delay vs. Setup and Hold Times**

Data-Clk vs. Clk-Output

- Setup
- Hold
- Sampling Window

**Impact of Vdd on the Sampling Window**

Sampling window determines the minimum required duration of data signal
Pulsed Latch (Flip-flop) Example

Hybrid Latch – Flip-flop (HLFF), AMD K-6 and K-7:

- Transparent to D only when Clk and Clk \_1 are both high
- Limited clock uncertainty absorption
- Small D→Q delay
- Small clock load

\[ \text{Partovi et al. 1996} \]
HLFF Analysis

Oklobdzija et al.
Digital System Clocking, Wiley’03

Logic Representation of HLFF
**Sense Amplifier Flip-flop (SAFF)**

**Principle**

- **DQ**: Data Input
- **Q**: Data Output
- **S**: Set Input
- **R**: Reset Input
- **D**: Clock Input

Montanaro et al. 1996

---

**Energy Consumption**

The slide shows a graph comparing energy consumption per transition for different types of latches and flip-flops. The graph includes bars representing energy consumption in fJ (femtojoules) for M-S Latches and Flip-Flops (including pulse-gen) across various technologies:

- MSL
- CMOS
- SDFF
- HLFF
- M-SAFF

The graph highlights the energy efficiency of different technologies and shows how energy consumption varies with different circuit implementations.
Timing Analysis

- Timing parameters
- Clock nonidealities (skew and jitter)
  - Impact of Clk skew on timing
  - Impact of Clk jitter on timing

Timing (Cycle Time & Race Margin)

Cycle time: $T_{\text{Clk}} > t_{c-q} + t_{\text{logic}} + t_{su}$
Race margin: $t_{\text{hold}} < t_{c-q, cd} + t_{\text{logic, cd}}$
Clock Nonidealities

- 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
  - for level-sensitive clocking

Clock Skew and Jitter

- Both skew and jitter affect the effective cycle time
- Only skew affects the race margin
Clock Skew

- Earliest occurrence of Clk edge: Nominal – \( \frac{T_{sk}}{2} \)
- Latest occurrence of Clk edge: Nominal + \( \frac{T_{sk}}{2} \)
- Insertion delay
- Clk delay
- Max Clk skew

Sources of Skew and Jitter

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

Devices

Clock Generation

\( T_{sk} \)
Positive Skew

Launching edge arrives before the receiving edge

Negative Skew

Receiving edge arrives before the launching edge
Positive and Negative Skew

(a) Positive skew

(b) Negative skew

Impact of Clock Skew on Timing

Cycle time (Long Path)

\[ t_{c-q} + t_{\text{logic}} + t_{su} < T_{\text{Clk}} + \delta \]

\[ T_{\text{Clk}} > t_{c-q} + t_{\text{logic}} + t_{su} - \delta \]
Impact of Clock Skew on Timing

**Race immunity (Short Path)**

Data must not arrive before this time

\[ t_{c-q,cd} + t_{\text{logic},cd} > t_{\text{hold}} + \delta \]

\[ t_{\text{hold}} + \delta < t_{c-q,cd} + t_{\text{logic},cd} \]

Impact of Clock Jitter

\[ t_{c-q,cd} \]

\[ t_{\text{logic},cd} \]

\[ t_{\text{hold}} \]

\[ t_{\text{jitter}} \]
Impact of Clock Jitter on Timing

**Cycle time (Late-Early Problem)**

- Latest point of launching
- Earliest arrival of next cycle

\[ t_{c-q} + t_{\text{logic}} + tsu < T_{\text{Clk}} - t_{\text{jitter}} - t_{\text{jitter}} \]

\[ T_{\text{Clk}} > t_{c-q} + t_{\text{logic}} + tsu + 2t_{\text{jitter}} \]

Impact of Skew and Jitter on Timing

- **Cycle time**
  - Positive skew improves performance
  - Negative skew reduces performance
  - Jitter reduces performance

\[ T_{\text{Clk}} > t_{c-q} + t_{\text{logic}} + tsu - \delta + 2t_{\text{jitter}} \]

- **Race Margin**
  - Skew reduces race margin
  - Jitter reduces acceptable skew

\[ t_{\text{hold}} + 2t_{\text{jitter}} + \delta < t_{c-q,cd} + t_{\text{logic,cd}} \]