COMBINATIONAL LOGIC DYNAMICS

Fast Complex Gates: Design Technique 1
- Transistor sizing
  - as long as fan-out capacitance dominates
- Progressive sizing
  - Distributed RC line
  - M1 > M2 > M3 > … > MN
  - (the fet closest to the output is the smallest)
  - Can reduce delay by more than 20%; decreasing gains as technology shrinks

Fast Complex Gates: Design Technique 2
- Transistor ordering

Fast Complex Gates: Design Technique 3
- Alternative logic structures

Fast Complex Gates: Design Technique 4
- Isolating fan-in from fan-out using buffer insertion

Fast Complex Gates: Design Technique 5
- Reducing the voltage swing
  - $t_{PER} = 0.69 \left( \frac{3}{4} C_L V_{swing} \right) / I_{DSATn}$
  - $t_{PER} = 0.69 \left( \frac{3}{4} C_L V_{DD} \right) / I_{DSATn}$
  - Linear reduction in delay
  - Also reduces power consumption
  - But the following gate is much slower!
  - Or requires use of “sense amplifiers” on the receiving end to restore the signal level (memory design)
Sizing Logic Paths for Speed

- Frequently, input capacitance of a logic path is constrained.
- Logic also has to drive some capacitance.
- Example: ALU load in an Intel’s microprocessor is 0.5pF.
- How do we size the ALU datapath to achieve maximum speed?
- We have already solved this for the inverter chain — can we generalize it for any type of logic?

Buffer Example

\[ \text{Delay} = \sum_{i=1}^{N} \left( t_i + g_i \cdot f_i \right) \text{ (in units of } t_{\text{inv}}) \]

For given \( N \), \( C_i/C_0 = C/C_0 \).
To find \( N \), \( C_i/C_0 \approx 4 \).
How to generalize this to any logic path?

Logical Effort

\[ \text{Delay} = k \cdot R_{\text{load}} C_{\text{out}} \left( 1 + \frac{C_i}{C_{\text{inv}}} \right) \]

\( p \) = intrinsic delay \( (3kR_{\text{unit}}C_{\text{unit}}) \) - gate parameter \( \propto W \)
\( g \) = logical effort \( (kR_{\text{unit}}C_{\text{unit}}) \) - gate parameter \( \propto W \)
\( f \) = effective fanout

Normalize everything to an inverter:
\( R_{\text{inv}} = 1, R_{\text{inv}} = 1 \)
Divide everything by \( t_{\text{inv}} \)
(everything is measured in unit delays \( t_{\text{inv}} \))
Assume \( \gamma = 1 \).

Delay in a Logic Gate

Gate delay:
\[ d = h + p \]
Effort delay:
\[ h = g \cdot f \]

Logical effort is a function of topology, independent of sizing.
Effective fanout (electrical effort) is a function of load/gate size.

Logical Effort

- Inverter has the smallest logical effort and intrinsic delay of all static CMOS gates.
- Logical effort of a gate presents the ratio of its input capacitance to the inverter capacitance when sized to deliver the same current.
- Logical effort increases with the gate complexity.

Intrinsic Delay

- Inverter has the smallest intrinsic delay and of all static CMOS gates.
- Intrinsic delay of a gate presents the ratio of its output capacitance to the inverter output capacitance when sized to deliver the same current.
- Intrinsic delay increases with the gate complexity.
Logical Effort

Logical effort is the ratio of input capacitance of a gate to the input capacitance of an inverter with the same output current.

Logical Effort of Gates

- Fan-out ($f$)
- Normalized delay ($d$)

Intrinsic Delay

Effort Delay

Add Branching Effort

- Branching effort: $C_{\text{off-path}}$ is the branch capacitance
- $b = C_{\text{on-path}} + C_{\text{off-path}} / C_{\text{on-path}}$

Multistage Networks

- Stage effort: $h_i = g_i f_i$
- Path electrical effort: $F = C_{\text{out}} / C_{\text{in}}$
- Path logical effort: $G = g_1 g_2 \cdots g_N$
- Branching effort: $B = b_1 b_2 \cdots b_N$
- Path effort: $H = G F B$
- Path delay $D = \sum d_i = \sum p_i + \sum h_i$

Optimum Effort per Stage

When each stage bears the same effort:

- $h^N = H$
- $h = \sqrt[N]{H}$

Stage efforts: $g_1 f_1 = g_2 f_2 = \cdots = g_N f_N$

Effective fanout of each stage: $f_i = h / g_i$

Minimum path delay:

$$\hat{D} = \sum (g_i f_i + p_i) = NH^{1/N} + P$$
Optimal Number of Stages

For a given load and given input capacitance of the first gate, find optimal number of stages and optimal sizing.

\[
D = NH^{1/N} + Np_{in}
\]

\[
\frac{\partial D}{\partial N} = -\frac{H^{1/N}}{N} \ln(H^{1/N}) + \frac{H^{1/N}}{N} + p_{in} = 0
\]

Substitute ‘best stage effort’ \( h = H^{1/N} \)

Logical Effort

<table>
<thead>
<tr>
<th>Number of Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate Type</td>
</tr>
<tr>
<td>Inverter</td>
</tr>
<tr>
<td>NAND</td>
</tr>
<tr>
<td>NOR</td>
</tr>
<tr>
<td>Mux</td>
</tr>
<tr>
<td>XOR</td>
</tr>
</tbody>
</table>

Example: Optimize Path

Example: Optimize Path

Example – 8-input AND

Intrinsic delays

8+1 4+2 2+2+2+1

Fan out is not known here

Method of Logical Effort

- Compute the path effort: \( H = GBF \)
- Find the best number of stages \( N \sim \log_2 H \)
- Compute the stage effort \( h = H^{1/N} \)
- Sketch the path with this number of stages
- Work either from either end, find sizes:

\[
C_{in} = C_{out} \frac{g}{h}
\]

### Summary

<table>
<thead>
<tr>
<th>Term</th>
<th>Stage expression</th>
<th>Path expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical effort</td>
<td>( \tilde{g} )</td>
<td>( \tilde{g} = \prod \tilde{b} )</td>
</tr>
<tr>
<td>Electrical effort</td>
<td>( \tilde{h} )</td>
<td>( \tilde{h} = \prod \tilde{h} )</td>
</tr>
<tr>
<td>Branching effort</td>
<td>( \tilde{f} )</td>
<td>( \tilde{f} = \prod \tilde{f} )</td>
</tr>
</tbody>
</table>

### Ratio Based Logic

#### Ratio Based Logic

- **Resistive Load**
  - \( V_{OL} = \frac{R_{IN} V_{DD}}{R_{IN} + R_{L}} \)
  - Asymmetrical response
  - Static power consumption
  - \( t_{PLH} = 0.69 R_L C_L V_{DD} \)

#### Ratio Based Logic Problems

- **Problems with Resistive Load**
  - \( I_L = (V_{DD} - V_{out}) / R_L \)
  - Charging current drops rapidly once \( V_{out} \) starts to rise

- **Solution:** Use a current source!
  - A variable current is independent of voltage
  - Reduces \( t_{PLH} \) by 25%

### Active Loads

- **Depletion mode NMOS load**
  - \( V_{GS} = 0 \)
  - \( I_L \sim \frac{(R_{IN} \text{ load})}{2} (\left| V_{TH} \right|) \)
  - Deviates from ideal current source
    - Channel length modulation
    - Body effect
      - \( V_{SS} \approx V_{DD} \)
      - \( V_{SS} \) varies with \( V_{out} \)
      - \( V_{out} \) reduces \( \left| V_{TH} \right| \), hence, gets smaller for increasing \( V_{out} \)

- **PMOS Load**
  - \( V_{OL} = V_{DD} \)
  - \( V_{QL} \) varies with \( V_{out} \)
  - \( V_{out} \) reduces \( \left| V_{TH} \right| \), hence, gets smaller for increasing \( V_{out} \)
Active Loads

Pseudo-NMOS load
- No body effect, $V_{SB} = 0V$
- $V_{GS} = -V_{DD}$, higher load current
- $I_L = \left(\frac{k_p}{2}\right)(V_{DD} - |V_{Tn}|)^2$
- Larger $V_{GS}$ causes pseudo-NMOS load to leave saturation mode sooner than NMOS

Load Lines of Ratioed Gates

Pseudo-NMOS

\[
V_{OL} = \frac{V_{DD} - |V_{Tn}|}{k_n} - \frac{V_{OL}}{k_p}
\]

\[
V_{OH} = \frac{V_{DD}}{k_n} + \frac{V_{OL}}{k_p}
\]

Pseudo-NMOS VTC

Noise margin low is significantly reduced compared to CMOS

Pseudo-NMOS NAND Gate

$C_{L\text{pseudo}} = 0.5 C_{L\text{CMOS}}$ (Fan-out of 1)

Improved Loads

Adaptive Load
Improved Loads (2)

Differential Cascode Voltage Switch Logic (DCVSL)

DCVSL Example

XOR-XXOR gate

DCVSL Transient Response

Pass-Transistor Logic

• N transistors
• No static consumption

Example: AND Gate

NMOS-Only Logic
Threshold voltage loss causes static power consumption. NMOS has higher threshold than PMOS (body effect).

**Restorer Sizing**

- Upper limit on restorer size
- Pass-transistor pull-down can have several transistors in stack

**Solution 2: Single Transistor Pass Gate with $V_T = 0$**

- If pass transistors have $V_T = 0$, the output does not require level restorer but there is a leakage current.

**Solution 3: Transmission Gate**

Complementary Pass Transistor Logic
Resistance of Transmission Gate

\[ R_{eq} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2}} \]

Pass-Transistor Based Multiplexer

Transmission Gate XOR

\[ A \oplus B = \overline{A} \cdot B + A \cdot \overline{B} \]

Delay in Transmission Gate Networks

Transmission Gate Full Adder

\[ C_{out} = \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot C \]

Delay Optimization

- Delay of RC chain
  \[ t_D = 0.69 \sum_{i=0}^{n} C_i R_i - 0.69 C_{eq} \frac{m+1}{2} \]
- Delay of buffered chain
  \[ t_D = 0.69 \sum_{i=0}^{n} C_i R_i - 0.69 C_{eq} \frac{m+1}{2} + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \]

\[ W_{opt} = \frac{1}{\sqrt{3}} \cdot \frac{1}{C_{eq}} \]

Similar delays for sum and carry
Dynamic CMOS

- In static circuits at every point in time (except when switching) the output is connected to either GND or \( V_{DD} \) via a low resistance path.
  - fan-in of \( n \) requires \( 2n \) (\( n \) N-type + \( n \) P-type) devices

- Dynamic circuits rely on the temporary storage of signal values on the capacitance of high impedance nodes.
  - requires \( n + 2 \) (\( n \) N-type + 1 P-type) transistors

Dynamic Gate

Two phase operation
Precharge \( (\text{Clk} = 0) \)
Evaluate \( (\text{Clk} = 1) \)

Conditions on Output

- Once the output of a dynamic gate is discharged, it cannot be charged again until the next precharge operation.
- Inputs to the gate can make at most one transition during evaluation.
- Output can be in the high impedance state during and after evaluation (PDN off), state is stored on \( C_L \)

Properties of Dynamic Gates

- Logic function is implemented by the PDN only
  - number of transistors is \( N + 2 \) (versus \( 2N \) for static complementary CMOS)
- Full swing outputs \( (V_{OL} = \text{GND} \text{ and } V_{OH} = V_{DD}) \)
- Non-ratioed - sizing of the devices does not affect the logic levels
- Faster switching speeds
  - reduced load capacitance due to lower input capacitance \( (C_i) \)
  - reduced load capacitance due to smaller output loading \( (C_{out}) \)
  - no \( I_{sc} \), so all the current provided by PDN goes into discharging \( C_L \)

Properties of Dynamic Gates

- Overall power dissipation usually higher than static CMOS
  - no static current path ever exists between \( V_{DD} \) and GND (including \( P_{sc} \))
  - no glitching
  - higher transition probabilities
  - extra load on Clk
- PDN starts to work as soon as the input signals exceed \( V_{Tn} \), so \( V_{IL} \) and \( V_{IH} \) equal to \( V_{Tn} \)
  - low noise margin (NM)
- Needs a precharge/evaluate clock

Issues in Dynamic Design 1: Charge Leakage

Leakage sources
Dominant component is subthreshold current
Solution to Charge Leakage

Same approach as level restorer for pass-transistor logic

Issues in Dynamic Design 2: Charge Sharing

Charge stored originally on $C_L$ is redistributed (shared) over $C_a$ and $C_b$ leading to reduced robustness

Charge Sharing Example

Charge stored originally on $C_L$ is redistributed (shared) over $C_a$ and $C_b$ leading to reduced robustness

Precharge internal nodes using a clock-driven transistor (at the cost of increased area and power)

Issues in Dynamic Design 3: Backgate Coupling

Output connected to a NAND gate and Out2 pulls down Out1 through capacitive coupling
### Backgate Coupling Effect

![Graph showing voltage vs. time for Backgate Coupling Effect]

- Coupling between Out and Clk input of the precharge device due to the gate to drain capacitance. So voltage of Out can rise above $V_{DD}$. The fast rising (and falling edges) of the clock couple to Out.

### Issues in Dynamic Design 4: Clock Feedthrough

![Clock Feedthrough diagram]

- More noise is generated as a result of clock coupling

### Other Effects

- Capacitive coupling between output wires pulls down pre-stored charges
- Substrate coupling
- Minority charge injection
- Supply noise (negative ground bounce may discharge the output)

### Cascading Dynamic Gates

![Cascading Dynamic Gates diagram]

- Only $0 \rightarrow 1$ transitions allowed at inputs!
- So do not connect these gates directly

### Domino Logic

![Domino Logic diagram]

- Here we guarantee proper $0$ to $1$ transitions between gates
Why Domino?

Properties of Domino Logic

- Only non-inverting logic can be implemented
- Very high speed
  - static inverter can be skewed, only L-H transition
  - so make PMOS of inverter stronger
  - Input capacitance reduced – smaller logical effort

Designing with Domino Logic

Footless Domino

The first gate in the chain needs a foot switch
Precharge is rippling – short-circuit current
A solution is to delay the clock for each stage

Differential (Dual Rail) Domino

Solves the problem of non-inverting logic

np-CMOS

Only 0 → 1 transitions allowed at inputs of PDN
Only 1 → 0 transitions allowed at inputs of PUN

Possible coupling in longer runs to dynamic node
NORA Logic

WARNING: Very sensitive to noise!

Example: Full Adder

\[ C_0 = AB + C_i(A+B) \]

A Revised Adder Circuit

24 transistors