Propagation Delay of CMOS Gates

(a) Two-input NAND  (b) RC equivalent model

<table>
<thead>
<tr>
<th>Input Data Pattern</th>
<th>Delay (psec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = B = 0 → 1</td>
<td>69</td>
</tr>
<tr>
<td>A = 1, B = 0 → 1</td>
<td>62</td>
</tr>
<tr>
<td>A = 0 → 1, B = 1</td>
<td>50</td>
</tr>
<tr>
<td>A = B = 1 → 0</td>
<td>35</td>
</tr>
<tr>
<td>A = 1, B = 1 → 0</td>
<td>76</td>
</tr>
<tr>
<td>A = 1 → 0, B = 1</td>
<td>57</td>
</tr>
</tbody>
</table>
Propagation delay of Four input NAND Gate

$$t_{pHL} = 0.69(R_1 \cdot C_1 + (R_1 + R_2) \cdot C_2 + (R_1 + R_2 + R_3) \cdot C_3 + (R_1 + R_2 + R_3 + R_4) \cdot C_L)$$
Disadvantages of Complementary CMOS Design

- Increase in complexity
- Larger implementation area
- Propagation delay deteriorates rapidly as a function of fan-in
Disadvantages of Complementary CMOS Design

- Large number of transistors increases overall capacitance of the gate
- Series connection causes slowdown
Design Techniques for Large Fan-in
Transistor Sizing

- Increase the size of transistor

- But increasing the transistor size also increases propagation delay
Progressive Transistor Sizing

$M_1 > M_2 > M_3 > M_N$
Input Reordering

(a) 

(b)
Logic Restructuring
Optimizing Performance in Combinatorial Networks

\[ t_p = t_{p0}\left(1 + \frac{C_{\text{ext}}}{\gamma C_g}\right) = t_{p0}(1 + f/\gamma) \]

\[ t_p = t_{p0}(p + gf/\gamma) \]

- \( t_{p0} \) = intrinsic delay of an inverter
- \( f \) = effective fan-out (ratio between external load and input capacitance), also called electrical effort
- \( p \) = ratio of intrinsic delays of the complex gates and the simple inverter. Function of gate topology as well as layout style
- \( g \) = logical effort
Estimated of $p$ and $g$

<table>
<thead>
<tr>
<th>Gate type</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverter</td>
<td>1</td>
</tr>
<tr>
<td>$n$-input NAND</td>
<td>$n$</td>
</tr>
<tr>
<td>$n$-input NOR</td>
<td>$n$</td>
</tr>
<tr>
<td>$n$-way multiplexer</td>
<td>$2n$</td>
</tr>
<tr>
<td>XOR, NXOR</td>
<td>$n2^{a-1}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gate Type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverter</td>
<td>1</td>
<td></td>
<td></td>
<td>(n+2)/3</td>
</tr>
<tr>
<td>NAND</td>
<td>4/3</td>
<td>5/3</td>
<td></td>
<td>(n+2)/3</td>
</tr>
<tr>
<td>NOR</td>
<td>5/3</td>
<td>7/3</td>
<td></td>
<td>(2n+1)/3</td>
</tr>
<tr>
<td>Multiplexer</td>
<td>2</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>XOR</td>
<td>4</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Optimizing Performance in Combinatorial Networks

gate effort $h = fg$
Optimizing Performance in Combinatorial Networks

Total delay of path through combinatorial logic block

\[ t_p = \sum_{j=1}^{N} t_{p,j} = t_{p0} \sum_{j=1}^{N} \left( p_j + \frac{f_j g_j}{\gamma} \right) \]

Find minimum delay by finding N-1 partial derivative and set them to zero, we get

\[ f_1 g_1 = f_2 g_2 = \ldots = f_N g_N \]

Path logical effort, G can be found by multiplying logical efforts of all the gates

\[ G = \Pi g_i \]
Optimizing Performance in Combinatorial Networks

Path effective fan-out (or electrical effort) $F$, 

$$F = \frac{C_L}{C_{g1}}$$

Some of the drive current is directed along path while some is directed of the path. We need to calculation branching effort as well

$$b = \frac{(C_{on-path} + C_{off-path})}{C_{on-path}}$$

Path Branching effort can be calculated by

$$B = \Pi b_i$$
Optimizing Performance in Combinatorial Networks

Path Electrical effort can be calculated from electrical and branching effort

\[ F = \prod (f_i/b_i) = (\prod f_i)/B \]

Total path effort \( H \) can be defined as

\[ H = \prod h_i = \prod g_i f_i = GFB \]

Gate effort that minimizes the path delay is

\[ h = \sqrt[\sqrt{F/G}]{H} = \sqrt{H} \]

Minimum delay of path is given by:

\[ D = \tau_{p0} \left( \sum_{j=1}^{N} P_j + \frac{N}{\gamma} \sqrt[H]{H} \right) \]
Optimizing Performance in Combinatorial Networks

If $s_1$ is the sizing factor of first gate in the chain

$$g_2 s_2 C_{\text{ref}} = \left(\frac{f_1}{b_1}\right) g_1 s_1 C_{\text{ref}}$$

For gate $i$ in the chain, this yields

$$s_i = (g_1 s_1/g_i) \prod_{j=1}^{i-1} (f_j/g_j)$$
Example

\[ F = \frac{C_L}{C_{g1}} = 5 \]
\[ G = 1 \times \left(\frac{5}{3}\right)^2 \times 1 = \frac{25}{9} \]
\[ B = 1 \text{ (no branching)} \]
\[ H = GFB = \frac{125}{9} \]
\[ h = 4 \sqrt{\frac{125}{9}} = 1.93 \]
\[ f_1 = 1.93 \]
\[ f_2 = 1.93 \times \left(\frac{3}{5}\right) = 1.16 \]
\[ f_3 = 1.16 \]
\[ f_4 = 1.93 \]
\[ a = \frac{f_1 g_1}{g_2} = 1.16 \]
\[ b = \frac{f_1 f_2 g_1}{g_3} = 1.34 \]
\[ c = \frac{f_1 f_2 f_3 g_1}{g_4} = 2.60 \]
Power Consumption in CMOS Logic Gates

Dynamic power dissipation,

\[ \alpha_{0 \rightarrow 1} C_L V_{DD}^2 f. \]

- \( \alpha_{0 \rightarrow 1} \) is switching activity
  - Static component based on topology
  - Dynamic component based on timing behaviour (glitching)

\[ \alpha_{0 \rightarrow 1} = p_0 \cdot p_1 = p_0 \cdot (1 - p_0) \]

\[ \alpha_{0 \rightarrow 1} = \frac{N_0}{2^N} \cdot \frac{N_1}{2^N} = \frac{N_0 \cdot (2^N - N_0)}{2^{2N}} \]

- \( p_0 \) is probability when output is zero
- \( p_1 \) is probability when output is one
Example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

\[ a_0 \rightarrow 1 = \frac{N_0 \cdot (2^N - N_0)}{2^{2N}} = \frac{3 \cdot (2^2 - 3)}{2^2 \cdot 2} = \frac{3}{16} \]
Signal Statistics

Let, \( p_a \) and \( p_b \) be the probabilities that the inputs A and B are one. Assume inputs are not correlated. The probability the the output node is 1 is given by

\[
p_1 = (1-p_a)(1-p_b)
\]

Probability of transition from 0 to 1 is

\[
\alpha_{0\rightarrow1} = p_0 p_1 = (1-(1-p_a)(1-p_b))(1-p_a)(1-p_b)
\]

<table>
<thead>
<tr>
<th>Operation</th>
<th>( \alpha_{0\rightarrow1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>( (1-p_Ap_B)p_ap_B )</td>
</tr>
<tr>
<td>OR</td>
<td>( (1-p_A)(1-p_B)(1-(1-p_A)(1-p_B)) )</td>
</tr>
<tr>
<td>XOR</td>
<td>[1 - (p_A + p_B - 2p_ap_B)(p_A + p_B - 2p_ap_B)]</td>
</tr>
</tbody>
</table>
Intersignal Correlations

(a) Logic circuit without reconvergent fanout

(b) Logic circuit with reconvergent fanout
Dynamic or Glitching Transitions
Design Techniques to Reduce Switching Activity
Logic Restructuring

<table>
<thead>
<tr>
<th></th>
<th>$O_1$</th>
<th>$O_2$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$ (chain)</td>
<td>1/4</td>
<td>1/8</td>
<td>1/16</td>
</tr>
<tr>
<td>$p_0 = 1-p_1$ (chain)</td>
<td>3/4</td>
<td>7/8</td>
<td>15/16</td>
</tr>
<tr>
<td>$p_{b=1}$ (chain)</td>
<td>3/16</td>
<td>7/64</td>
<td>15/256</td>
</tr>
<tr>
<td>$p_1$ (tree)</td>
<td>1/4</td>
<td>1/4</td>
<td>1/16</td>
</tr>
<tr>
<td>$p_0 = 1-p_1$ (tree)</td>
<td>3/4</td>
<td>3/4</td>
<td>15/16</td>
</tr>
<tr>
<td>$p_{b=1}$ (tree)</td>
<td>3/16</td>
<td>3/16</td>
<td>15/256</td>
</tr>
</tbody>
</table>

Chain structure

Tree structure
Input ordering

\[(1 - 0.5 \times 0.2) (0.5 \times 0.2) = 0.09\]

\[(1 - 0.2 \times 0.1) (0.2 \times 0.1) = 0.0196\]
Time-multiplexing Resources

(a) parallel data transmission

(b) serial data transmission
Glitch Reduction by balancing signal paths

(a) Network sensitive to glitching

(b) Glitch-free network
Ratioed Logic

(a) generic

(b) pseudo-NMOS
Features

- Lesser number of transistors compared to Complementary approach (N+1 compared to 2N)
- Nominal High Voltage $V_{OH}$ is $V_{DD}$
- Nominal low voltage is not 0V
- Reduced Noise Margin
- Increase in static power dissipation.

\[ V_{OL} \approx \frac{k_p (-V_{DD} - V_{TP}) \cdot V_{DSAT}}{k_n (V_{DD} - V_{TN})} \approx \frac{\mu_p \cdot W_p}{\mu_n \cdot W_n} \cdot |V_{DSAT}| \]

\[ P_{low} = V_{DD} I_{low} \approx V_{DD} \cdot k_p \left( (-V_{DD} - V_{TP}) \cdot V_{DSATp} - \frac{V_{DSATp}^2}{2} \right) \]
Pseudo NMOS Inverter

<table>
<thead>
<tr>
<th>Size</th>
<th>$V_{OL}$</th>
<th>Static Power Dissipation</th>
<th>$t_{PH}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.693V</td>
<td>564μW</td>
<td>14ps</td>
</tr>
<tr>
<td>2</td>
<td>0.273V</td>
<td>298μW</td>
<td>56ps</td>
</tr>
<tr>
<td>1</td>
<td>0.133V</td>
<td>160μW</td>
<td>123ps</td>
</tr>
<tr>
<td>0.5</td>
<td>0.064V</td>
<td>80μW</td>
<td>268ps</td>
</tr>
<tr>
<td>0.25</td>
<td>0.031V</td>
<td>41μW</td>
<td>569ps</td>
</tr>
</tbody>
</table>
NAND vs NOR in Pseudo-NMOS

Which one would you prefer to implement in pseudo-NMOS?
Building Better Loads

- **Differential Logic**
  - Requires each input is provided by complementary format

- **Positive Feedback**
  - Ensure that load device is turned off when not needed

One example is Differential Cascode Voltage Switch Logic (DCVSL)
Single-Ended vs Differential

- Differential reduces number of transistors required by 2
- Differential removes need for inverter stage
- Differential need more wires for interconnects
- High Dynamic power dissipation
Pass-Transistor Logic

Advantages

- Fewer transistors required

Disadvantages

- NMOS is poor in pulling a node to $V_{DD}$
- NMOS experience body effect when pulling voltage to high

\[
V_x = V_{DD} - (V_{tn0} + \gamma ((\sqrt{2\phi_f} + V_x) - \sqrt{2\phi_f}))
\]
Voltage Swing for Pass Transistor Circuits

(a) Swing on $Y = V_{DD} - V_{Th1} - V_{Th2}$

(b) Swing on $Y = V_{DD} - V_{Th1}$
Energy of Pass-Transistor Logic

\[ E_{0 \rightarrow 1} = \int_{0}^{T} P(t) dt = V_{DD} \int_{0}^{T} i_{\text{supply}}(t) dt = V_{DD} \int_{0}^{(V_{DD} - V_{Tn})} C_{L} dV_{\text{out}} = C_{L} \cdot V_{DD} \cdot (V_{DD} - V_{Tn}) \]
Differential Pass-Transistor Logic (CPL or DPL)

(a) Basic concept
Properties of DPL

- Both input and output signal and its complement available.
- Output either connected to VDD or GND via low-resistance path
- Since design is almost similar. It makes it easier for modular design library
Robust and Efficient Pass-Transistor Design
Level Restoration
Level Restoration

(a) general concept

(b) XOR/XNOR gate
Multiple Threshold Transistors
Transmission Gate Logic

(a) Circuit

(b) Symbolic representation

C = V_{DD}

A = V_{DD}

B (initially at 0)

C = 0

(a) charging node B

(a) discharging node B

C = V_{DD}

A = 0

B (initially at V_{DD})

C = 0