CMOS COMBINATIONAL LOGIC

DELAY
- RC MODELS
- RC DELAY
- LOGICAL EFFORT

ENERGY
AREA

RC MODELS

- $C_{sb}$ capacitors do not actually switch, so ignore
- Lump $C_{dbp} + C_{dnn}$ since from low to constant nodes
- Lump $C_{gsp} + C_{gsn}$ since from high to constant nodes
- Assume pmos mobility 2x nmos high now low mobility
Define \( C \) = gate capacitance of minimum sized NMOS

Define \( R \) = effective resistance of minimum sized NMOS

Define \( K \) = how much wider a transistor is compared to min NMOS

\[
\frac{1}{3} \frac{2R}{k} \quad \frac{1}{3} \frac{2R}{k}
\]

\[
\frac{1}{3} \frac{2kC}{R \cdot k} \quad \frac{kC}{R \cdot k} \quad \frac{2kC}{R \cdot k} \quad \frac{2kC}{R \cdot k}
\]

Input NAND Gate

VDD VDD

\[
\frac{1}{3} \frac{2R}{k} \quad \frac{1}{3} \frac{2R}{k}
\]

\[
A \quad \frac{2kC}{R} \quad \frac{2kC}{R} \quad \frac{2kC}{R}
\]

\[
\frac{1}{3} \frac{2kC}{R \cdot k} \quad \frac{kC}{R \cdot k} \quad \frac{2kC}{R \cdot k} \quad \frac{2kC}{R \cdot k}
\]

IF WE SHARE CONTACT DIFF

PARASITIC CAP

CONTACT NC

SWING DIFFUSION CAP
**RC DECRYPTION**

Input:  
\[ V_{in} \]

Output:  
\[ V_{out} \]

\[ V_{out}(t) = V_{DD} e^{-t/RC} \]

Let \( t = RC \)

\[ = \frac{V}{I} \cdot \frac{Q}{V} = \frac{Q}{I} \]

\[ = \frac{Q}{Q/s} = \text{Seconds} \]

\[ t_{pd} = \text{PROPAGATION DELAY, TIME UNTIL } V_{out} = V_{DD}/2 \]

\[ = CN(2) \cdot R \cdot C_1 \]

Let \( R' = CN(2) \cdot R \)

\[ R' C_1 = 2 R C \]

We usually just assume effective resistance is scaled by \( CN(2) \)

\[ t_{pd} = 2RC \]

\[ V_{out} = \frac{V_{DD}}{2} = V_{DD} e^{-t/\tau} \]

\[ \frac{1}{V_{DD}} = e^{-t/\tau} \]

\[ \ln(\frac{1}{2}) = -\frac{t}{\tau} \]

\[ t = -\tau \cdot \ln(\frac{1}{2}) = \tau \ln(2) \]
RC DELAY: NAND

\[ Y = A \cdot \frac{1}{1 + \frac{1}{B}} \]

\[ A = 1 \quad B = \frac{1}{1 - \frac{1}{1}} \]

\[ C_1 = C \quad C_2 = 3C \]

**Complicates 2nd order model**

**Approximation**

\[ T' = T_1' + T_2' = RC_1 + (R + R)C_2 \]

\[ = RC + (2R)(3C) \]

\[ = RC + 6RC = 7RC \quad (3.5 \times \text{slower than usual}) \]

Best when one \( T' \) much larger than other \( T' \)

Even if \( T_1' = T_2' \), error < 1.5\%

**Generalize to Elmore delay**

\[ t_{PD} = \sum_j R_{ij} C_i \]

Effective resistance of shared path from input to node \( i \) and node \( j \), where \( s \) is output.

\[ t_{PD} = RC + 2RC + 3RC \]

\[ = 6RC \]
Elmore Delay of Trees

Delay of path from $x$ to $y$ is impacted by branch to $z$

Delay of path from $x$ to $z$ is impacted by branch to $y$

For path $x$ to $y$ we also lump $C_2 + C_3$ and use shaded resistance, $R_0 + R_1$

Similarly for path $x$ to $z$ we lump $C_1$ and use shaded resistance, $R_0 + R_1$

This extra term factors in delay of "Branch"

$$t_{pd, xy} = R_0 C_0 + (R_0 + R_1 + R_2) C_1 + (R_0 + R_1) (C_2 + C_3)$$
$$= RC + 3RC + 4RC$$
$$= 8RC$$

$$t_{pd, xz} = R_0 C_0 + (R_0 + R_1 + R_2) C_2 + (R_0 + R_1 + R_3 + R_4) C_3 + (R_0 + R_1) C_1$$
$$= RC + 3RC + 4RC + 2RC$$
$$= 10RC$$
RISE/FALL TIMES: INVERTER

From earlier, \( t_{PD,1 \rightarrow 0} = 2RC \) (unequal rise/fall times)

\[ \frac{1}{2} \frac{1}{RC} \]

\[ \frac{1}{4} \frac{1}{RC} \]

For equal rise/fall times, the effective resistance of pullup must equal effective resistance of pulldown.

If we assume PMOS mobility \( 2x \) worse than NMOS, then PMOS must be \( 2x \) size of NMOS in an inverter for equal rise/fall times.

RISE/FALL TIMES: 2-INPUT NAND

Size transistors so worst case effective resistance equal in both pull-up and pull-down networks.

\[ \frac{1}{2} \frac{1}{2R} \]

\[ \frac{1}{3} \frac{1}{2R} \]

Assumes worst case where only a single PMOS is pulling up output node.
\[ t_{PD, 0 \to 1} = RC + (R + R) 3C = 7RC \]

\[ t_{PD, 1 \to 0} = (R + R) 3C = 6RC \]

\[ t_{PD, 0 \to 1} = 2RC + 2RC = 4RC \]

\[ t_{PD, 1 \to 0} = RC \]

**Table:**

<table>
<thead>
<tr>
<th></th>
<th>Worst</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Internal capacitor</td>
<td>GRC</td>
<td>GRC</td>
</tr>
<tr>
<td>With Internal capacitor</td>
<td>GRC</td>
<td>7RC</td>
</tr>
</tbody>
</table>

So delay depends on internal capacitance and the order in which inputs arrive on which inputs change.
RISE/FALL TIMES: Zinput Nk

\[ t_{pd, 1 \rightarrow 0} = \frac{R}{2} GC \]

\[ t_{pD} = 3 RC \]

\[ t_{pD} = GNC + RHC \]

\[ t_{pD} = 10 RC \]

\[ t_{pD, 0 \rightarrow 1} = GRC \]

\[ t_{pD} = \frac{R}{2} HC + \left(\frac{C}{2} + \frac{C}{2}\right) GC \]

\[ t_{pD} = 8 RC \]

\[ t_{pD} = GRC \]
<table>
<thead>
<tr>
<th>INVERTER</th>
<th>3RC</th>
<th>3RC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND</td>
<td>6RC</td>
<td>6RC</td>
</tr>
<tr>
<td></td>
<td>GRC</td>
<td>GRC</td>
</tr>
<tr>
<td>NOR</td>
<td>10RC</td>
<td>8RC</td>
</tr>
<tr>
<td></td>
<td>GRC</td>
<td>GRC</td>
</tr>
</tbody>
</table>

Is this a fair comparison? No, we are not normalizing anything across these gates. Need to either normalize:

1) Input gate cap (ie. load on previous gate)
2) Drive strength (ie. effective resistance)

Effective resistance of all 3 gates:

- INV: $T_L$
- NAND: $2R$ — twice the effective resistance
- NOR: $T_L$ — so half the drive current

Here are all three gates sized to have equal rise and fall times and the same drive strength.

$$R_{eff} = R_L$$
$$C_{in} = 3C$$
$$C_{out} = 3C$$
$$t_{pp,100} = 3 RC$$
$$t_{pp,0.01} = 3 RC$$
LARGER GATES

Worst case:

\[ t_{pq, 1 \rightarrow 0} = \frac{2R}{K} \cdot 2K + \left( \frac{R}{K} + \frac{R}{K} \right) 3K = 7RC \]

\[ t_{pq, 0 \rightarrow 1} = \frac{2R}{K} \cdot 2K + \left( \frac{2R}{K} \right) K = 9RC \]

This is the parasitic delay, it is independent of size.

LARGER LOADS
\[ \tau_{PD,1} = \frac{R}{k}KC + \left( \frac{R}{k} + \frac{R}{k} \right) (3K + \beta)C \]

\[ RC + 2 \frac{R}{k} (3K + \beta)C \]

\[ RC + CRC + 2 \frac{R}{k} \beta C \]

\[ TRC + \frac{\beta}{k} ZRC \]

\[ \text{EFFECT DELAY: DEPENDS ON COMPLEXITY OF GATE, SIZE OF GATE AND WHAT IT IS DRIVING} \]

\[ \text{PARASITIC DELAY: INHERENT} \]

\[ \text{Delay when no load is attached, independent of sizing} \]

\[ \text{NOTE: INCREASING \( \beta \), INCREASES EFFECT DELAY} \]

\[ \text{INCREASING \( k \), DECREASES EFFECT DELAY} \]

\[ \text{But will increase \( \beta \) of previous gate!} \]
Deriving Linear Decay Mode (logical effort)

- Abstract all CMOS logic gates as

\[ C_{W} = \alpha C_{T} \]

\[ R_{i} = R_{ui} = R_{di} = \frac{R_{+}}{\alpha} \]

\[ C_{pi} = \alpha C_{pt} \]

\[ t_{pd} = t_{abs} = R_{i} \left( C_{out} + C_{pi} \right) \]

\[ = \frac{R_{+}}{\alpha} \frac{C_{w}}{C_{w}} \left( \frac{C_{out}}{C_{w}} \right) + \frac{R_{+}}{\alpha} \left( \alpha C_{pt} \right) \]

\[ = \frac{R_{+}}{\alpha} \frac{dC_{t}}{C_{w}} \left( \frac{C_{out}}{C_{w}} \right) + \frac{R_{+}}{\alpha} \left( \alpha C_{pt} \right) \]

\[ = R_{+} \frac{C_{t}}{C_{w}} \left( \frac{C_{out}}{C_{w}} \right) + R_{+} C_{pt} \]
\[ a_{AN} = R_t C_t \left( \frac{C_{OUT}}{C_{IN}} \right) + R_t C_{P} \]

\[ a_{AN} \text{ Parasitic delay} \]

\[ a_{AN} \text{ Effort delay} \]

\[ \alpha \text{ is "wisdom" in just } C_{IN} \]

\[ (v + r) = R_{INV} C_{INV} \]

\[ g = \frac{R_t C_t}{R_{INV} C_{INV}} \]

\[ h = \frac{C_{OUT}}{C_{IN}} \]

\[ p = \frac{R_t C_{P}}{R_{INV} C_{INV}} \]

\[ a_{AN} = R_{INV} C_{INV} \left( \frac{R_t C_t}{R_{INV} C_{INV}} \right) \left( \frac{C_{OUT}}{C_{IN}} \right) + R_{INV} C_{INV} \left( \frac{R_t C_{P}}{R_{INV} C_{INV}} \right) \]

\[ a_{AN} = \alpha \left( gh + p \right) \]

\[ a_{AN} = \eta d \]

- **Complexity of gate topology**
- **Ratio of gate time constant to INV time constant**
- **How much worse is gate at producing current compared to INV with same input cap**
- **How much more input cap is necessary to drive same current as min. INV.**

\[ p = \frac{R_t C_t}{R_{INV} C_{INV}} = \frac{C_t}{C_{INV}} \]

Rate of diff cap/mv vs gate cap/mv

Crudeley assume 1 for INV
TEMPLATES

\[
\begin{array}{cccc}
R_{\text{eff}} & R & R & R \\
C_{\text{in}} & 3C & 4C & 5C \\
\frac{R \cdot C}{R \cdot C} & \frac{R \cdot C}{R \cdot C} & \frac{R \cdot C}{R \cdot C} & \frac{R \cdot C}{R \cdot C} \\
1 & 4/3 & \frac{5}{3} & \frac{5}{3} \\
\frac{R + C}{R + C} & \frac{R + C}{R + C} & \frac{R + C}{R + C} & \frac{R + C}{R + C} \\
1 & \frac{6}{3} = 2 & \frac{6}{3} = 2 & \frac{6}{3} = 2
\end{array}
\]

RECALL \( d = gh + p \) (LINEAR DELAY MODEL)

Plot \( d \) as a function of \( h \)

\[
\text{NOR: } d = \frac{5}{3}h + 2 \\
\text{NANO: } d = \frac{4}{3}h + 2 \\
\text{INV: } d = h + 1
\]

Parasitic delay \[ h = \frac{C_{\text{par}}}{C_{\text{in}}}) \]

Why we prefer NANO vs. NOR? NOR has higher delay for same electrical effort.
Many, Many Approximations

- Extraneous Delay
- $p = 1$ for $\mu V$ (i.e. $1 \mu m$ diff cap $\approx 1 \mu m$ gate cap)
- Ignore internal parasitic cap
- Equal rise and fall time
- $\mu_0 = \frac{1}{2}\mu_1$
- Ignore actual rise/fall times
- Ignore worst arrival time
- Ignore velocity saturation

Still reasonably good results when using logical effort for sizing even in modern technologies. News designers build the right intuition.
MULTISTAGE LOGIC NETWORK

RAW DELAY IS THE SUM OF THE DELAY OF EACH STAGE

\[ D = \sum d_i = \sum g_i h_i + \sum p_i \]

KEY QUESTIONS:
- How should we size gates to minimize total delay?
- How should we change the topology to minimize delay?

Let's develop some metrics that are independent of sizing

RAW LOGICAL EFFORT = RAW ELECTRICAL EFFORT

\[ G = \prod g_i \quad \quad \quad \quad H = \frac{C_{\text{out}}}{C_{\text{in}}} \]

So for the above example:

\[ G = 1 \cdot \frac{5}{3} \cdot \frac{4}{3} \cdot 1 = \frac{20}{9} = 2.22 \]

\[ H = \frac{20C}{10C} = 2 \]

RAW EFFORT IS THE PRODUCT OF STAGE EFFORTS

\[ F = \prod f_i = \prod g_i h_i \]
So since stage effort \( f = gh \), does path effort \( F = GH \)?

Consider simple example:

\[
G = 1 \times 1 = 1
\]

\[
H = 90\text{c} / 5\text{c} = 18
\]

\[
F = \Pi gh_i = 1 \times 6 \times 1 \times 6 = 36
\]

But \( GH = 18 \neq 36 \)

So in this example, \( F = 2GH \)

(A we call this the "branching" effort)

Key idea is some of the drive current is directed off the path we are analyzing. Recall empire delay for trees.

\[
\text{stage brown ch. } \quad b = \frac{\text{Conpam} + \text{Conpam}}{\text{Conpam}}
\]

\[
\text{path brown ch. } \quad T_b = \Pi b_i
\]

For above example \( b = \frac{15+15}{15} = 2 \)

So path effort

\[
F = \Pi f_i = \Pi g_i h_i = G \cdot B \cdot H
\]

Note that path effort depends on circuit topology and loading of entire path but not size of transitions in network.

Also note that path effort does not change if add or remove inverters!
With these meeting we can now answer the two earlier questions.

Q1: How should we size gates to minimize total delay?

Start with path delay equation:

\[ D = \sum_i d_i = \sum_i g_i h_i + \sum_i p_i \]

Independent variables are \( h_i \)'s (i.e. internal gate swings). We want to choose \( h_i \)'s to minimize \( D \). So we can take the partial derivative of \( D \) with respect to \( h_i \)'s, set to zero, and solve for optimum \( h_i \)'s.

Consider two stage path

\[ \text{Gate 1} \rightarrow \text{Gate 2} \]

Assume \( C_1 \) and \( C_2 \) are given

- Input cap \( C_i \)
- Logical eff \( g_i \)
- Parasitic delay \( p_i \)

\[ D = (g_i h_i + p_i) + (g_2 h_2 + p_2) \]

Note that \( h_1 \) and \( h_2 \) are constrained since \( C_1 \) and \( C_2 \) are given and input cap of gate 2 is output cap of gate 1.

\[ h_1 = \frac{C_2}{C_1}, \quad h_2 = \frac{C_3}{C_2} \]

\[ H = h_1 h_2 = \frac{C_3}{C_1} \]

\( H \) is a constant since \( C_1 \) and \( C_2 \) are given

Substitute \( h_2 = H h_1 \) into delay equation

\[ D = (g_i h_i + p_i) + \left( \frac{g_2 H}{h_1} h_1 + p_2 \right) \]
Take partial derivative with respect to only variable \( h_1 \),

\[
\frac{dD}{dh_1} = g_1 - \frac{g_2 h}{h_1^2} = 0
\]

\[g_1 h_1^2 = g_2 h\]
\[g_1 h_2 = g_2 h_1 h_2\]
\[g_1 h_1 = g_2 h_2\]
\[f_1 = f_2\]

Delay is minimized when stage effort is same in both stages!

This generalizes to fans with any number of stages and fans with branching effort.

Fastest design always equalizes effort in each stage.

For a general fan, optimal delay is:

\[\hat{f} = F^{1/3}\]

Take total fan effort and divide evenly across \( n \) stages

\[\hat{D} = N F^{1/3} + P\]

**Memo for optimal sizing**

1. Calculate fan effort \( F = G B H \)
2. Calculate effort for each stage \( \hat{f} = F^{1/3} \)
3. Estimate min delay of optimal sizing \( \hat{D} = N \hat{f} + P \)
4. Starting with last stage, work backwards sizing each gate

\[\hat{f} = gh = g \frac{C_{out}}{C_{in}}\]
\[C_{in} = \frac{g}{\hat{f}} C_{out}\]
Example of Optimal Sizing

\[ F = G \cdot D \cdot M = \left( \frac{5}{3} \times \frac{4}{3} \times 1 \right) \left( 1.1 \times 2 \times 1 \right) \left( \frac{20C}{10C} \right) = 2.22 \times 2 \times 2 = 8.89 \]

Step 1)

\[ F = F^{1/0} = 1.73 \]

Step 2)

\[ \hat{D} = 4 \cdot (1.73) + (1 + 2 + 2 + 1) = 6.92 + 6 = 12.92 \]

\[ \hat{D}_{obs} = 12.92 \text{ V} \]

Step 3)

\[ C_{in} = \frac{1}{1.73} \times 20C = 11.56 \text{ C} \]

\[ C_{in} = \frac{4/3}{1.73} \times 11.56 \text{ C} \times 2 = 17.81 \text{ C} \]

\[ C_{1,2} = \frac{5/3}{1.73} 17.81 \text{ C} = 17.16 \text{ C} \]

\[ C_{1,2} = \frac{1}{1.73} 17.16 \text{ C} = 9.97 \text{ C} \]

Close to given input caps of 10C
This assumes we can size gate arbitrarily. In a full custom design, what if using standard cell lib?

Assume standard cell lib with following gates

\[ \text{INV} x_1, \text{INV} x_2, \text{INV} x_3, \text{INV} x_4, \text{INV} x_5, \text{INV} x_6 \]
\[ \text{NAND} x_1, \text{NAND} x_2, \text{NAND} x_4 \]
\[ \text{NOR} x_1, \text{NOR} x_2, \text{NOR} x_4 \]

What does \( x_1, x_2, x_4 \) mean? \( x_2 \) means twice the drive strength of an \( x_1 \) inverter. So \( \text{NAND} x_2 \) means \( \beta = 2 \)

Assume we have determined optimal sizing is \( C_{in} \), how do we figure out which cell to use?

\[ g = \frac{P}{C} \]

If we assume \( P = \frac{P_{max}}{C_{in}} \)

\[ g = \frac{C_{in}}{C} \Rightarrow C = g C_{in} \]

\[ C_{in} = \alpha C \]

So \( C = \alpha g C_{in} \)

\[ \alpha = \frac{C_{in}}{g C_{in}} \quad \text{since} \quad C_{in} = \Phi C \]
Given optimum $C_{in}$ from before, what is $\alpha$?

<table>
<thead>
<tr>
<th></th>
<th>$g$</th>
<th>$\frac{11.56}{1.3}$</th>
<th>$\frac{17.81}{4.3}$</th>
<th>$\frac{17.16}{3.3}$</th>
<th>$\frac{9.92}{1.3}$</th>
<th>Gate?</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.56 C</td>
<td>1</td>
<td>8.85</td>
<td>4.45</td>
<td>5.55</td>
<td>3.3</td>
<td>INV x 4</td>
</tr>
<tr>
<td>17.01 C</td>
<td>4/3</td>
<td>17.81</td>
<td>20.05</td>
<td>17.16</td>
<td>9.92</td>
<td>NaN x 4</td>
</tr>
<tr>
<td>17.16 C</td>
<td>5/3</td>
<td>17.61</td>
<td>19.3</td>
<td>16.9</td>
<td>9.92</td>
<td>NaN x 4</td>
</tr>
<tr>
<td>9.92 C</td>
<td>1</td>
<td>9.92</td>
<td>17.81</td>
<td>17.16</td>
<td>9.92</td>
<td>INV x 3</td>
</tr>
</tbody>
</table>

Must be < 10 C for your constant

Recalculate actual delay given these gates

First calculate actual $C_{in}$ for each and cell gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>$C_{in}$ = $\alpha \cdot g \cdot C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV x 4</td>
<td>$C_{in} = \alpha \cdot g \cdot 3C = 4 \cdot 1.3 \cdot 3C = 12C$</td>
</tr>
<tr>
<td>NON x 4</td>
<td>$C_{in} = \alpha \cdot g \cdot 3C = 4 \cdot 4/3 \cdot 3C = 16C$</td>
</tr>
<tr>
<td>NA x 4</td>
<td>$C_{in} = \alpha \cdot g \cdot 3C = 4 \cdot 5/3 \cdot 3C = 20C$</td>
</tr>
<tr>
<td>INV x 3</td>
<td>$C_{in} = \alpha \cdot g \cdot 3C = 3 \cdot 1.3 \cdot 3C = 9C$</td>
</tr>
</tbody>
</table>

Now use RAM delay equation

\[ D = \sum gh + \sum p \]

\[ = \left( \frac{1.20}{12} \right) + \left( \frac{4}{3} \cdot \frac{12}{16} \right) + \left( \frac{5}{3} \cdot \frac{16}{20} \right) + \left( \frac{1.20}{9} \right) + \left( 1 + 2 + 2 + 1 \right) \]

\[ = 1.67 + 2 + 1.53 + 2.22 + 6 \]

\[ = 12.22 + 6 \]

\[ = 18.22 \]

Compare with previous which is 17.92 off by 2.3%
Q2: How should we change topology to minimize delay?

Assume we want to implement an eight input AND gate. Calculate minimum delay assuming optimal sizing for the following three topologies assuming $H=1$ and $H=12$.

<table>
<thead>
<tr>
<th>Topology</th>
<th>$N_F^{1/4}$</th>
<th>$P$</th>
<th>$\hat{D}$</th>
<th>$N_F^{1/4}$</th>
<th>$P$</th>
<th>$\hat{D}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-NAND</td>
<td>3.65</td>
<td>9</td>
<td>12.65</td>
<td>12.64</td>
<td>9</td>
<td>21.64</td>
</tr>
<tr>
<td>2x 4-NAND</td>
<td>3.65</td>
<td>6</td>
<td>9.65</td>
<td>12.64</td>
<td>6</td>
<td>18.64</td>
</tr>
<tr>
<td>4x 2-NAND</td>
<td>5.25</td>
<td>7</td>
<td>12.25</td>
<td>9.77</td>
<td>7</td>
<td>16.77</td>
</tr>
</tbody>
</table>
Optimal number of stages using invariances?

\[ \dot{D} = NF^{\frac{1}{2}} + Np_{in} \]

\[ \frac{d}{dn} \dot{D} = F^{\frac{1}{2}} - F^{\frac{1}{2}} \ln \left( F^{\frac{1}{2}} \right) + p_{in} = 0 \]

If \( p_{in} = 0 \)

\[ F^{\frac{1}{2}} - F^{\frac{1}{2}} \ln \left( F^{\frac{1}{2}} \right) = 0 \]

\[ \ln \left( F^{\frac{1}{2}} \right) = 1 \]

\[ F^{\frac{1}{2}} = e \]

So in other words, \( F = e \)

So if we assume \( p_{in} = 0 \), optimal number of stages results in stage effort of \( e \) for every stage. Since \( g = 1 \) for an invaer, this means \( h = 2.178 \) for every stage.

If \( p_{in} = 1 \) then we need to solve following nonlinear eq:

\[ F^{\frac{1}{2}} - F^{\frac{1}{2}} \ln \left( F^{\frac{1}{2}} \right) + 1 = 0 \]

Let \( \rho = F^{\frac{1}{2}} \) where \( \rho \) is optimal

\[ 1 + \rho \left( 1 - \ln \rho \right) = 0 \]

We can find numerically that \( \rho \approx 3.59 \)

So optimal number of stages results in stage effort of 3.59 when we take into account parasitics. We can roughly approximate 3.59 to be 4.
\[ F^{\frac{1}{2}} = 4 \]

\[ \log\left( F^{\frac{1}{2}} \right) = \log(4) \]

\[ \frac{1}{2} \log(F) = \log(4) \]

\[ \hat{N} = \frac{\log(F)}{\log(4)} = \log_4(F) \]

So optimal number of stages for inverter cascade is roughly:

\[ \hat{N} = \log_4(F) \]

Since \( g=1 \) and \( B=1 \) for inverter cascade

\[ \hat{N} = \left( \log_4(4) \right) \]

Not too bad of an estimate even for realistic PAMS or qAMX that are not inverters.
Logical effort can help give us intuition on how to size gates and choose a topology to minimize delay but it has many limitations.

To deal with more complicated scenarios we can also write the delay equations for each gate in the system and minimize the latest arrival time.

**Example**

![Diagram](image)

Let's write our linear delay equation as a function of $d$.

$$d = gh + p$$

$$g = \frac{R_f C_t}{n_{in} C_{in}}$$

$$C_{in} = \alpha_f C_t$$

$$C_t = \frac{C_{in}}{\alpha_f}$$

$$d = \frac{C_{in} \cdot C_{out}}{d(C_{in})} + p$$

$$g = \frac{C_{in}}{d(C_{in})}$$

Delay as function of $d$ (strobe cell drive).

Recall $C_{in} = 3g\alpha_f$.
Now write delay equations for each stage:

<table>
<thead>
<tr>
<th>Stage</th>
<th>$d_0$</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$a$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$a_3$</td>
</tr>
<tr>
<td>2</td>
<td>$g$</td>
<td>$g_1$</td>
<td>$g_2$</td>
<td>$g_3$</td>
</tr>
<tr>
<td>3</td>
<td>$C_w$</td>
<td>$c_{w_1}$</td>
<td>$c_{w_2}$</td>
<td>$c_{w_3}$</td>
</tr>
<tr>
<td>$p$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
d_0 = \frac{(4d_1 + 7d_2)}{3} + 1 = \frac{4}{3}d_1 + \frac{7}{3}d_2 + 1
\]

\[
d_1 = \frac{(7d_2)}{3d_1} + 2 = \frac{7}{3}d_2 + 2
\]

\[
d_2 = \frac{(10 + 3d_3)}{3d_2} + 3 = \frac{10}{d_2} + \frac{d_3}{d_2} + 3
\]

\[
d_3 = \frac{(36)}{3d_3} + 1 = \frac{12}{d_3} + 1
\]

**Arrival Times**

\[
t_0 = d_0
\]

\[
t_1 = \max(t_0, d_1) + d_1
\]

\[
t_2 = \max(t_0, t_1) + d_2
\]

\[
t_3 = t_2 + d_3
\]

\[
t_3 = \max(t_0, t_1) + d_2 + d_3
\]

\[
= \max(d_0, \max(d_0, d_1) + d_1) + d_2 + d_3
\]
\[ t_3 = \max (d_0, \max (d_0, d_1) + d_1) + d_2 + d_3 \]

**minimize** \( t_3 \) **subject to** above constraints with \( d_0, d_2, d_3 \) as the independent variables

Actually in summary we really want to minimize area (on every) subject to constraint on \( t_3 \).

So we could craft optimization problem to be minimize sum of \( d_1, d_2, d_3 \) \((\text{max for area})\) subject to constraint:

\[ t_{\text{cloth}} > \max (d_0, \max (d_0, d_{\text{air}}) + d_{\text{air}}) + d_2 + d_3 \]

\( \text{clock phase constraint} \)
Energy

Energy is a measure of work.

Power is the rate at which work is done.

\[ E = \int_0^T P(t) \, dt \]

\[ P = \frac{\Delta E}{\Delta t} \]

Electric Potential

Electric Potential at a position per unit charge.

\[ (\text{volts, } 1 \text{V} = 1 \text{J/C}) \]

\[ \Delta V = \frac{\Delta E}{Q} \]

Current

Rate at which charge flows past position.

\[ (\text{amps, } 1 \text{A} = 1 \text{C/s}, I = \frac{Q}{\Delta t}) \]

Power

Rate at which electrical energy is supplied or consumed.

\[ (\text{watts, } 1 \text{W} = 1 \text{J/s}, P = \frac{\Delta E}{\Delta t} = \frac{\Delta V \cdot Q}{Q/s} = VI) \]
Energy Stored on Capacitor

\[ E_c = \int_0^\infty p(t)dt = \int_0^\infty v(t)I(t)dt \]

\[ = \int_0^\infty \frac{dQ}{dt} dt = \int_0^\infty \frac{CVdd}{dv} dv = \frac{1}{2} CVdd^2 \]

\[ \int_0^2 dV = \frac{1}{2} Vdd^2 \]

So on \( t \to 0 \) input transition \( \frac{1}{2} CVdd^2 \) stored on capacitor, thus energy is released on \( 0 \to 1 \) transition.

Energy Delivered from Power Supply

\[ E_{supply} = \int_0^\infty p(t)dt = \int_0^\infty vddI(t)dt \]

\[ = vdd \int_0^\infty \frac{dQ}{dt} dt = vdd \int_0^\infty \frac{CVdd}{dv} dv = \frac{CVdd}{2} \]

\[ \int_0^\infty dv = Vdd \]

Heat energy dissipated as heat in PMOS

Other will dissipates as heat in NMOS

On avg. each bit transistor requires \( \frac{1}{2} CVdd^2 \)

\( \Rightarrow \) probability of bit toggle
**Power**

\[ P_{tot} = P_{switching} + P_{static} \]

\[ = \alpha f \frac{1}{2} CV_{dd}^2 + V_{dd} I_{off} \]

(Note: the number of transistors, sometimes just 1 or 2 transistors, but won't be a factor of \( \frac{1}{2} \))
**Compare Energy**

Need to find total switched cap in worst case.

\[ C_{\text{sw, }.g} = \frac{1}{1.8} \times 10 = 5.6 \]

\[ C_{\text{n, }.g} = \frac{513}{1.8} \times 10 = 9.3 \]

\[ C_{\text{g}} = 1.8 \times 5.6 = 10.4 \]

\[ C_{\text{n, g}} = \frac{2}{1.8} \times 9.3 = 10.3 \]

\[ C_{\text{total, g}} = C_{\text{sw, g}} + B \cdot C_{\text{n, g}} \]

\[ = 5.6 + 8 \times 10.3 \]

\[ = 88.8 \text{ C} \]

To determine parasitic cap, need to understand how gate cap is distributed across transistors.
\[ E = \alpha \frac{1}{2} CV_{dd}^2, \text{ assume } d = 0.1 \text{ and } V_{dd} = 1 \text{ V for both, so only difference is amount of switched capacitance} \]

for 8-input nano topology

\[ C_{\text{tot}} = C_{\text{tot},g} + C_{\text{tot},p} = 88.8 + (5.6 + 24.96) = 119.36 \text{ fF} \]

for 1-input nano topology

\[ C_{\text{tot}} = C_{\text{tot},g} + C_{\text{tot},p} = 101 + (11.16 + 22.58) = 132.94 \text{ fF} \]

So second topology requires \(100\%\) more energy in the worst case where all capacitance is switched. This ignores the energy required for switching the output load.

These energy estimates are in units of \(\text{fF} \times \text{V}^2\) (i.e., the gate capacitance for a minimum N-well transistor). What if we want to estimate the absolute energy in joules?

Page 312, Table B.5 in Weste & Harris

\( C \) in IBM 90nm = 1-2 fF/\( \mu \)m \( V_{dd} = 1 \text{ V} \)

Assume \( W_{min} = 9 \lambda \)

\[ W = 4 \lambda \text{ too small, now assume } 9 \lambda \]

\[ \lambda = 45 \text{ nm} \]

\[ W_{min} = 9 \times 45 = 405 \text{ nm} = 0.4 \text{ \( \mu \)m} \]

Since \( C_g = 1-2 \text{ fF/}\mu \text{m} \) and \( W_{min} = 0.4 \text{ \( \mu \)m} \) est \( C = 0.5 \text{ fF} \)

for first topology (assume \( d = 0.1 \))

\[ E = \alpha \frac{1}{2} CV_{dd}^2 = \frac{1}{2} (120 \text{ C}) \frac{0.5 \text{ fF}}{c} (1V)^2 = 3 \text{ fJ} \]

\[ P = \alpha \frac{1}{2} CV_{dd}^2 = (0.5 \times 10^{-9}) (30 \times 10^{-15}) = 1.5 \text{ } 10^{-17} \text{ W} \]

\[ \approx 500 \text{ } \mu \text{W} \]
Activity factors

- Previous example just used fixed $d = 0.1$ for all nodes.
- More accurate to track activity factors w.r.t. topology

- Assumes inputs have completely random data

\[
P_i = \text{Prob. node is 1}
\]
\[
\bar{P}_i = 1 - P_i = \text{Prob. node is 0}
\]

\[
\alpha = \frac{P_i \bar{P}_i + \bar{P}_i \bar{P}_i}{\bar{P}_i \bar{P}_i}
\]

\[
\alpha' = \frac{P_i P_i}{P_i P_i}
\]

\[
\alpha = 0.5
\]

\[
\alpha' = 0.25 \quad \alpha' = \frac{1}{2} \alpha
\]

NAND 2

\[
\alpha'_{out} = \bar{P}_{o, out} P_{out} = (P_A P_B) (1 - P_A P_B)
\]

NAND B

\[
\alpha'_{out} = (P_A) (1 - P_B) = (0.0039) (0.996) = 0.0039
\]

Assume $P_A = P_B = ... = P$
NAND 2

Probability output is zero

\[ a_{\text{out}}' = \overline{a_1} \cdot \overline{a_2} = (p_a \cdot p_b)(1-p_a \cdot p_b) \]

Probability output is one

Output of NAND 2 is zero if both inputs are one

Output of NAND 2 is one if any inputs are zero

Assume inputs are random data

\[ p_a = 0.5 \quad p_b = 0.5 \]

\[ a_{\text{out}}' = (p_a \cdot p_b)(1-p_a \cdot p_b) = (0.5 \cdot 0.5) \cdot (1-0.5 \cdot 0.5) = (0.25) \cdot (1-0.25) = 0.25 \cdot 0.75 = 0.1875 \]

NAND 3

\[ a_{\text{out}}' = \overline{a_1} \cdot \overline{a_2} \cdot \overline{a_3} = (p_{a_1} \cdot p_{a_2}) \cdot (1-p_{a_3}) \]

Output is zero if all 3 inputs are one

Output is one if any inputs are zero

\[ a_{\text{out}}' = (p_{a_1} \cdot p_{a_2}) \cdot (1-p_{a_3}) = (0.5^2) \cdot (1-0.5^3) = (0.25) \cdot (1-0.125) = 0.25 \cdot 0.875 = 0.21875 \]
During lecture today, I mentioned that _adding_ inverters can sometimes _reduce_ the path delay. This might seems counter intuitive based on what you learned in ECE 2300. In ECE 2300, gates had a _constant_ delay. So every inverter might always have a delay of 1 tau, and every NAND2 gate might always have a delay of 2 tau. In fact, we used a similar simplification when estimating the critical path in ECE 4750. If we assume a constant delay model, then adding a pair of inverters would indeed _always_ slow down the path delay. Adding a pair of inverters would simply increase the total propagation delay.

Based on what we have learned in ECE 5745 so far, it should be clear that the constant delay model is a significant oversimplification. The delay of a gate depends on many things including its size, the load capacitance at the output, when inputs arrive, the rise/fall time of the inputs, layout details, etc. Our RC modeling and method of logical effort use a _linear_ delay model which is a little more reasonable than a _constant_ delay model (but of course is still a significant simplification!). So the delay of a gate is:

\[ d = gh + p \]

The logical effort \( g \) and the parasitic delay \( p \) depend only on the template, while the electrical effort \( h \) depends on both the size of the gate (\( C_{in} \)) and the load capacitance at the output (\( C_{out} \)).

Let’s look in more detail at the example we were discussing in lecture to demonstrate how _adding_ inverters can sometimes _reduce_ the path delay. Assume after synthesis we have the following two-gate path:

```
--|NAND
 |NAND---I>o----.---
--|NAND
 4C -- 1000C
NAND2X1 INVX4 ---
  
```

So we have a X1 two-input NAND gate (NAND2X1) and a X4 inverter driving a load of 1000C. The synthesis tool optimized the design assuming the inverter was driving a modest load, but after place-and-route, it turned out that the inverter has to drive a cross-chip global wire and thus a very large fixed capacitance.

What is the delay of this two-gate path?

\[
D = ( g_0 h_0 + g_1 h_1 ) + ( p_0 + p_1 ) \\
= ( \frac{4}{3} \times 12/4 + 1 \times 1000/12 ) + ( 2 + 1 ) \\
= ( 4 + 83.3 ) + 3 \\
= 90.3 \text{ tau}
\]

Recall that the minimum delay will occur when the stage effort is equal across all stages. Notice that the stage effort of the two stages is not even close to being equal which suggests this sizing is suboptimal.

The place-and-route tool can potentially reduce the path delay using "buffer resizing". So let’s assume the tool wants to increase the size of the inverter. Let’s use logical effort to figure out the optimal sizing.

\[
F = GHB = \frac{4}{3} \times 1000/4 \times 1 = 333 \\
f' = F^{1/N} = (333)^{1/2} = 18.25 \\
D' = N\times F^{1/N} + P = 2\times18.25 + ( 2 + 1 ) = 39.5 \text{ tau}
\]
Note that I am using $f'$ instead of $f$ "hat" and $D'$ instead of $D$ "hat". The path delay is significantly lower if we can resize the inverter. Let’s figure out how large the final inverter needs to be to achieve this optimal delay.

$$\text{C}_\text{in},1 = (g/f') \times \text{C}_\text{load} = (1/18.25) \times 1000 = 54.8\text{C}$$

That is a pretty big inverter! The inverter’s NMOS would be 18.27 times the minimum width and the inverter’s PMOS would be 36.53 times the minimum width. Assume our standard cell library has an INVX1, INVX2, INVX4, INVX8, INVX16, INVX32, and INVX64. Let’s choose the INVX16 for the final inverter (which is a little smaller than the optimal full-custom sizing).

![Schematic of the circuit](image)

What is the new delay of the new path?

$$D = (g_0h_0 + g_1h_1) + (p_0 + p_1)$$
$$= (4/3 \times 48/4 + 1 \times 1000/48) + (2 + 1)$$
$$= (16 + 20.8) + 3$$
$$= 39.8 \tau$$

The delay using the standard cell is a little slower and the stage effort is not exactly balanced, but buffer resizing does still significantly reduce the path delay.

The place-and-route tool can potentially further reduce the path delay using "buffer insertion". Let’s quickly estimate the optimal number of stages.

$$\log_4(F) = \log_4(333) = 4.2$$

So the rough estimate of the optimal number of stages is 4, but we are only using two stages. Let’s add two INVX1 gates at the end of the path to see if that helps.

![Schematic of the circuit with buffer insertion](image)

$$D = (g_0h_0 + g_1h_1 + g_2h_2 + g_3h_3) + (p_0 + p_1 + p_2 + p_3)$$
$$= (4/3 \times 48/4 + 1 \times 3/48 + 1 \times 3/3 + 1 \times 1000/3) + (2 + 1 + 1 + 1)$$
$$= (16 + 0.0625 + 1 + 333.3) + 5$$
$$= 355 \tau$$

Yeow -- this is a bad idea. The delay is 9x worse! Instead of driving the large load capacitance with an INVX16, now we are driving this large load capacitance with an INVX1. Very bad idea. What if we add two more INVX16 gates at the end of the path?
This is better than the original design, but slower than the optimized two-gate design with buffer resizing. The key is that we don’t want to add more inverters. We want to add more inverters and then properly resize the gates to ensure we are balancing the stage efforts appropriately. We can just use the method of logical effort to find the optimal delay and the optimal sizing.

\[
F = \frac{GHB}{4/3 \times 1000/4} = 333
\]

\[
f' = F^{(1/N)} = (333)^{(1/4)} = 4.27
\]

\[
D' = N \times F^{(1/N)} + P = 4 \times 4.27 + (2 + 1 + 1 + 1) = 22.1 \text{ tau}
\]

So we have further reduced the delay by using a combination of buffer insertion and buffer resizing. Let’s figure out how large each inverter needs to be to achieve this optimal delay.

\[
C_{in,3} = \left(\frac{g}{f'}\right) \times C_{load} = \left(\frac{1}{4.27}\right) \times 1000 = 234C
\]

\[
C_{in,2} = \left(\frac{g}{f'}\right) \times C_{in,3} = \left(\frac{1}{4.27}\right) \times 234 = 55C
\]

\[
C_{in,1} = \left(\frac{g}{f'}\right) \times C_{in,2} = \left(\frac{1}{4.27}\right) \times 55 = 13C
\]

Yeow -- that is a big final inverter! Our INVX64 has a C_in of 192C so that will be the best we can do. Let’s size our inverters as follows:

\[
\text{---|NAND}
\]

\[
\text{NAND---I>o----I>o----I>o----I>o----.---}
\]

\[
\text{---|NAND}
\]

\[
4C \text{ --- 1000C}
\]

\[
\text{NAND2X1 INVX16 INVX16 INVX16 ---}
\]

\[
| \quad V
\]

And now let’s calculate the delay again:

\[
D = (g0 \times h0 + g1 \times h1 + g2 \times h2 + g3 \times h3) + (p0 + p1 + p2 + p3)
\]

\[
= (4/3 \times 48/4 + 1 \times 96/48 + 1 \times 192/96 + 1 \times 1000/192) + (2 + 1 + 1 + 1)
\]

\[
= (16 + 2 + 2 + 5.2) + 5
\]

\[
= 27.2 \text{ tau}
\]

The original design without buffer insertion/resizing had a delay of 90.3 tau while the new design with buffer insertion/resizing has a delay of only 27.2, and improvement of 3.3x! So clearly _adding_ inverters can _reduce_ the path delay, but (as we saw with some of our counter-examples) this is only true if you properly size the gates!
It is useful in our pen-and-paper analysis to have a good estimate for some of the technology parameters in our target process. For example, when estimating the power and energy of a circuit, we need to know the supply voltage and gate capacitance (since all of our switched capacitance estimates will be in units of C). West & Harris provides a methodology for using SPICE simulations to estimate various technology parameters. We can also look in the .lib file for our 45nm standard cell library, since this file was itself generated from many SPICE simulations.

The following snippet shows the entry in the .lib file for our canonical inverter (INV_X1). We can see that the nominal supply voltages is 1.1V and that the total input gate capacitance for this inverter is estimated to be 1.7fF.

```plaintext
library (NangateOpenCellLibrary) {
...
/* Units Attributes */
time_unit               : "1ns";
leakage_power_unit      : "1nW";
voltage_unit            : "1V";
current_unit            : "1mA";
pulling_resistance_unit : "1kohm";
capacitive_load_unit    : (1,ff);
/* Op Conditions */
nom_process             : 1.00;
nom_temperature         : 25.00;
nom_voltage             : 1.10;
****************************************************************
Module     : INV_X1
Cell Descr : Combinational cell (INV_X1) with drive strength X1
****************************************************************

cell (INV_X1) {
drive_strength    : 1;
area              : 0.532000;

...}

pin (A) {
direction         : input;
related_power_pin : "VDD";
related_ground_pin: "VSS";
capacitance       : 1.700230;
fall_capacitance  : 1.549360;
rise_capacitance  : 1.700230;
}
...}
```

However, we need an estimate for C which is the gate cap for the NMOS in this canonical inverter. We can figure out C if we take a closer look at the layout and SPICE deck for this inverter. The following page shows the layout for an INV_X1, INV_X2, and INV_X4 gate along with the corresponding SPICE deck for an INV_X1 and INV_X2 gate.

Notice how the layout uses multiple parallel "fingers" to implement a single larger "logical" transistor. So an X2 gate has two fingers and an X4 has four fingers. The SPICE deck has the exact length and width of each transistors (we could also just measure the layout). Notice that both transistors have a width of 50nm even though this is a 45nm process! It is not unusual for standard-cell libraries to use slightly longer than
minimum transistors, since this geometry offers a nice compromise between performance and power consumption. The PMOS width is 630nm and the NMOS width is 415nm. Notice that the PMOS is definitely not twice the width of the NMOS (it is only 630/415 = 1.5x). This is probably because the mobility of an NMOS transistor is not exactly 2x the mobility of a PMOS transistor, and also because the standard-cell library is choosing to offer slightly unequal rise/fall times to offer reduced energy and area. Also notice that the NMOS in this inverter is 415/45 is about 9x the technology node size. This is ratio is a very reasonable size.

With this information we can now estimate C. We know the total gate cap for an INV_X1 gate is 1.7fF, and we know the ratio of how much of that gate cap comes from the NMOS is 415/(415+630) = 0.4. So C is 0.4 * 1.7fF = 0.68fF. To make our analysis simpler we will just roughly estimate the supply voltage as 1V and C as 0.5fF.
.SUBCKT INV_X1 A ZN VDD VSS
*.PININFO A:I ZN:O VDD:P VSS:G
*.EQN ZN=!A
M_i_0 ZN A VSS VSS NMOS_VTL W=0.415U L=0.050U
M_i_1 ZN A VDD VDD PMOS_VTL W=0.630U L=0.050U
.ENDS

.SUBCKT INV_X2 A ZN VDD VSS
*.PININFO A:I ZN:O VDD:P VSS:G
*.EQN ZN=!A
M_i_0_0_x2_0 ZN A VSS VSS NMOS_VTL W=0.415U L=0.050U
M_i_1_0_x2_0 ZN A VDD VDD PMOS_VTL W=0.630U L=0.050U
.ENDS

.SUBCKT INV_X4 A ZN VDD VSS
*.PININFO A:I ZN:O VDD:P VSS:G
*.EQN ZN=!A
M_i_0_0_x2_1 VSS A ZN VSS NMOS_VTL W=0.415U L=0.050U
M_i_1_0_x2_1 VDD A ZN VDD PMOS_VTL W=0.630U L=0.050U
.ENDS