Performance Characterization

- Delay analysis
- Transistor sizing
- Logical effort
- Power analysis

Delay Definitions

- $$t_{\text{pdr}}$$: rising propagation delay
  - From input to rising output crossing $$V_{\text{DD}}/2$$

- $$t_{\text{pdf}}$$: falling propagation delay
  - From input to falling output crossing $$V_{\text{DD}}/2$$

- $$t_{\text{pd}}$$: average propagation delay
  - $$t_{\text{pd}} = (t_{\text{pdr}} + t_{\text{pdf}})/2$$

- $$t_{\text{r}}$$: rise time
  - From output crossing 0.2 $$V_{\text{DD}}$$ to 0.8 $$V_{\text{DD}}$$

- $$t_{\text{f}}$$: fall time
  - From output crossing 0.8 $$V_{\text{DD}}$$ to 0.2 $$V_{\text{DD}}$$
Simulated Inverter Delay

- Solving differential equations by hand is too hard
- SPICE simulator solves the equations numerically
  - Uses more accurate I-V models too!
- But simulations take time to write

Delay Estimation

- We would like to be able to easily estimate delay
  - Not as accurate as simulation
  - But easier to ask “What if?”
- The step response usually looks like a 1st order RC response with a decaying exponential.
- Use RC delay models to estimate delay
  - C = total capacitance on output node
  - Use effective resistance R
  - So that $t_{pd} = RC$
- Characterize transistors by finding their effective R
  - Depends on average current as gate switches
RC Delay Models

- Use equivalent circuits for MOS transistors
  - Ideal switch + capacitance and ON resistance
  - Unit nMOS has resistance R, capacitance C
  - Unit pMOS has resistance 2R, capacitance C
- Capacitance proportional to width
- Resistance inversely proportional to width

Example: 3-input NAND

- Sketch a 3-input NAND with transistor widths chosen to achieve effective rise and fall resistances equal to a unit inverter (R).
Example: 3-input NAND

- Sketch a 3-input NAND with transistor widths chosen to achieve effective rise and fall resistances equal to a unit inverter (R).
3-input NAND Caps

- Annotate the 3-input NAND gate with gate and diffusion capacitance.
3-input NAND Caps

- Annotate the 3-input NAND gate with gate and diffusion capacitance.

Elmore Delay

- ON transistors look like resistors
- Pullup or pulldown network modeled as \( RC \) ladder
- Elmore delay of RC ladder

\[
t_{pd} \approx \sum_{\text{nodes } i} R_{i\text{-to-source}} C_i
\]

\[
= R_1 C_1 + (R_1 + R_2) C_2 + \ldots + (R_1 + R_2 + \ldots + R_N) C_N
\]
Example: 2-input NAND

• Estimate worst-case rising and falling delay of 2-input NAND driving \( h \) identical gates.

![Diagram of 2-input NAND gate driving \( h \) copies]

Example: 2-input NAND

• Estimate rising and falling propagation delays of a 2-input NAND driving \( h \) identical gates.

![Diagram showing propagation delays: 2C, 6C, 4hC]
Example: 2-input NAND

- Estimate rising and falling propagation delays of a 2-input NAND driving \( h \) identical gates.

\[
t_{pdr} = (6 + 4h)RC
\]
Example: 2-input NAND

- Estimate rising and falling propagation delays of a 2-input NAND driving \( h \) identical gates.

\[
\begin{align*}
\text{Example: 2-input NAND} \\
\text{• Estimate rising and falling propagation delays of a 2-input NAND driving } h \text{ identical gates.}
\end{align*}
\]
Example: 2-input NAND

- Estimate rising and **falling** propagation delays of a 2-input NAND driving $h$ identical gates.

$$t_{pdf} = (2C)(\frac{R}{2}) + [(6 + 4h)C](\frac{R}{2} + \frac{R}{2})$$

$$= (7 + 4h)RC$$

Delay Components

- Delay has two parts
  - *Parasitic delay*
    - 6 or 7 RC
    - Independent of load
  - *Effort delay*
    - 4h RC
    - Proportional to load capacitance
Contamination Delay

- Best-case (contamination) delay can be substantially less than propagation delay.
- Ex: If both inputs fall simultaneously

\[ t_{cdr} = (3 + 2h)RC \]

Diffusion Capacitance

- We assumed contacted diffusion on every s / d.
- Good layout minimizes diffusion area
- Ex: NAND3 layout shares one diffusion contact
  - Reduces output capacitance by 2C
  - Merged uncontacted diffusion might help too
Layout Comparison

- Which layout is better?

Resizing the Inverter

Minimum-sized transistor: $W=3\lambda, L=2\lambda$

To get equal rise and fall times, $\beta_n = \beta_p \Rightarrow W_p = 3W_n$, assuming that electron mobility is three times that of holes: $W_p=9\lambda$.

Sometimes the function being implemented makes resizing unnecessary!
Analyzing the NAND Gate

\[ \beta_{n,\text{eff}} = \frac{1}{\frac{1}{\beta_{n1}} + \frac{1}{\beta_{n2}} + \frac{1}{\beta_{n3}}} \]

Resistances are in series (conductances are in parallel)

If \( \beta_{n1} = \beta_{n2} = \beta_{n3} = \beta_n \) then \( \beta_{n,\text{eff}} = \beta_n/3 \)

- Pull-down circuit has three times resistance, one-third times the conductance

For pull-up, only one transistor has to be on, \( \beta_{p,\text{eff}} = \min\{\beta_{p1}, \beta_{p2}, \beta_{p3}\} \)

If \( \beta_{p1} = \beta_{p2} = \beta_{p3} = \beta_p = \beta_p/3 \) then \( \beta_{p,\text{eff}} = \beta_p \) \( \Rightarrow \) no resizing is necessary

Analyzing the NOR Gate

\[ \beta_{p,\text{eff}} = \frac{1}{\frac{1}{\beta_{p1}} + \frac{1}{\beta_{p2}} + \frac{1}{\beta_{p3}}} \]

Resistances are in series (conductances are in parallel)

If \( \beta_{p1} = \beta_{p2} = \beta_{p3} = \beta_p \) then \( \beta_{p,\text{eff}} = \beta_p/3 \)

- Pull-up circuit has three times resistance, one-third times the conductance

For pull-down, only one transistor has to be on, \( \beta_{n,\text{eff}} = \min\{\beta_{n1}, \beta_{n2}, \beta_{n3}\} \)

If \( \beta_{n1} = \beta_{n2} = \beta_{n3} = \beta_n = 3\beta_p \) then \( \beta_{n,\text{eff}} = 9\beta_p \) \( \Rightarrow \) considerable resizing is necessary

\[ W_p = 9W_n \]
Effect of Series Transistors

Resize the pull-up transistors to make pull-up times equal.
After resizing:
a: $2\beta_p$, b: $2\beta_p$, c: $\beta_p$
Transistor Placement (Series Stack)

How to order transistors in a series stack?

Body effect: \( \delta V_t \propto \sqrt{V_{sb}} \)

- At time \( t = 0 \), \( a=b=c=0 \), \( f=1 \), capacitances are charged
- Ideally \( V_{ta} = V_{tb} = V_{tc} \approx 0.8V \)
- However, \( V_{ta} > V_{tb} > V_{tc} \) because of body effect
- If \( a, b, c \) become 1 at the same time, which transistor will switch on first?
- \( t_c \) will switch on first (\( V_{sb} \) for \( t_c \) is zero), \( C_c \) will discharge, pulling \( V_{sb} \) for \( t_b \) to zero
- If signals arrive at different times, how should the transistors be ordered?
- Design strategy: place latest arriving signal nearest to output-early signals will discharge internal nodes
Some Design Guidelines

• Use NAND gates (instead of NOR) wherever possible
• Placed inverters (buffers) at high fanout nodes to improve drive capability
• Avoid use of NOR completely in high-speed circuits: $A_1 + A_2 + \ldots + A_n = A_1 . A_2 . . . A_n$

Some Design Guidelines

• Use limited fan-in (<10): high fan-in $\Rightarrow$ long series stacks
• Use minimum-sized gates on high fan-out nodes: minimize load presented to driving gate
Logical Effort

• Chip designers face a bewildering array of choices
  – What is the best circuit topology for a function?
  – How many stages of logic give least delay?
  – How wide should the transistors be?

• Logical effort is a method to make these decisions
  – Uses a simple model of delay
  – Allows back-of-the-envelope calculations
  – Helps make rapid comparisons between alternatives
  – Emphasizes remarkable symmetries

Delay in a Logic Gate

• Express delays in process-independent unit
\[ d = \frac{d_{\text{abs}}}{\tau} \]
\[ \tau = 3RC \]

– 12 ps in 180 nm process
– 40 ps in 0.6 μm process
Delay in a Logic Gate

- Express delays in process-independent unit
  \[ d = \frac{d_{abs}}{\tau} \]
- Delay has two components
  \[ d = f + p \]

- Effort delay \( f = gh \) (a.k.a. stage effort)
  - Again has two components
Delay in a Logic Gate

- Express delays in process-independent unit
  \[ d = \frac{d_{abs}}{\tau} \]
- Delay has two components
  \[ d = f + p \]
- Effort delay \( f = gh \) (a.k.a. stage effort)
  - Again has two components
  - \( g \): logical effort
    - Measures relative ability of gate to deliver current
    - \( g = 1 \) for inverter
  - \( h \): electrical effort = \( C_{out} / C_{in} \)
    - Ratio of output to input capacitance
    - Sometimes called fanout
Delay in a Logic Gate

- Express delays in process-independent unit
  \[ d = \frac{d_{\text{abs}}}{\tau} \]

- Delay has two components
  \[ d = f + p \]

- Parasitic delay \( p \)
  - Represents delay of gate driving no load
  - Set by internal parasitic capacitance

\[ d = f + p = gh + p \]
Delay Plots

\[ d = f + p \]
\[ = gh + p \]

Computing Logical Effort

- **Definition:** Logical effort is the ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current.
- **Measure from delay vs. fanout plots**
- **Or estimate by counting transistor widths**

\[
\begin{align*}
C_n & = 3 \\
g & = 3/3 \\
C_n & = 4 \\
g & = 4/3 \\
C_n & = 5 \\
g & = 5/3
\end{align*}
\]
Catalog of Gates

- **Logical effort of common gates**

<table>
<thead>
<tr>
<th>Gate type</th>
<th>Number of inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverter</td>
<td>1</td>
</tr>
<tr>
<td>NAND</td>
<td>4/3  5/3  6/3  (n+2)/3</td>
</tr>
<tr>
<td>NOR</td>
<td>5/3  7/3  9/3  (2n+1)/3</td>
</tr>
<tr>
<td>Tristate / mux</td>
<td>2  2  2  2  2</td>
</tr>
</tbody>
</table>

- **Parasitic delay of common gates**
  - In multiples of $p_{inv}$ (≈1)

<table>
<thead>
<tr>
<th>Gate type</th>
<th>Number of inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverter</td>
<td>1</td>
</tr>
<tr>
<td>NAND</td>
<td>2  3  4  n</td>
</tr>
<tr>
<td>NOR</td>
<td>2  3  4  n</td>
</tr>
<tr>
<td>Tristate / mux</td>
<td>2  4  6  8  2n</td>
</tr>
<tr>
<td>XOR, XNOR</td>
<td>4  6  8</td>
</tr>
</tbody>
</table>
Example: Ring Oscillator

• Estimate the frequency of an N-stage ring oscillator

Logical Effort: \( g = \)
Electrical Effort: \( h = \)
Parasitic Delay: \( p = \)
Stage Delay: \( d = \)
Frequency: \( f_{osc} = \)

31 stage ring oscillator in 0.6 \( \mu \)m process has frequency of ~ 200 MHz

Frequency: \( f_{osc} = \frac{1}{2N*d} = \frac{1}{4N} \)
Example: FO4 Inverter

- Estimate the delay of a fanout-of-4 (FO4) inverter

Logical Effort: \( g = 1 \)
Electrical Effort: \( h = 4 \)
Parasitic Delay: \( p = 1 \)
Stage Delay: \( d = 5 \)

The FO4 delay is about
- 200 ps in a 0.6 \( \mu \)m process
- 60 ps in an 180 nm process
- \( f/3 \) ns in an \( f \) \( \mu \)m process
Multistage Logic Networks

• Logical effort generalizes to multistage networks

• *Path Logical Effort*  
  \[ G = \prod g_i \]

• *Path Electrical Effort*  
  \[ H = \frac{C_{\text{out-path}}}{C_{\text{in-path}}} \]

• *Path Effort*  
  \[ F = \prod f_i = \prod g_i h_i \]

• Can we write \( F = GH \)?
Paths that Branch

• No! Consider paths that branch:

\[ G = 1 \]
\[ H = \frac{90}{5} = 18 \]
\[ GH = 18 \]
\[ h_1 = \frac{(15 + 15)}{5} = 6 \]
\[ h_2 = \frac{90}{15} = 6 \]
\[ F = g_1 g_2 h_1 h_2 = 36 = 2GH \]
Branching Effort

- Introduce *branching effort*
  - Accounts for branching between stages in path
    \[ b = \frac{C_{\text{on path}} + C_{\text{off path}}}{C_{\text{on path}}} \]
    \[ B = \prod h_i \]
    \[ \prod h_i = BH \]

- Now we compute the path effort
  - \( F = GBH \)

Multistage Delays

- Path Effort Delay \( D_F = \sum f_i \)
- Path Parasitic Delay \( P = \sum p_i \)
- Path Delay \( D = \sum d_i = D_F + P \)
Designing Fast Circuits

\[ D = \sum d_i = D_F + P \]

- Delay is smallest when each stage bears same effort
  \[ \hat{f} = g_i h_i = F^{1/N} \]
- Thus minimum delay of N stage path is
  \[ D = NF^{1/N} + P \]
- This is a key result of logical effort
  - Find fastest possible delay
  - Doesn’t require calculating gate sizes

Gate Sizes

- How wide should the gates be for least delay?
  \[ \hat{f} = gh = g \frac{C_{out}}{C_{in}} \]
  \[ \Rightarrow C_{in_i} = \frac{g_i C_{out_i}}{\hat{f}} \]
- Working backward, apply capacitance transformation to find input capacitance of each gate given load it drives.
- Check work by verifying input cap spec is met.
Example: 3-stage path

- Select gate sizes x and y for least delay from A to B.

Example: 3-stage path

Logical Effort \( G = \)
Electrical Effort \( H = \)
Branching Effort \( B = \)
Path Effort \( F = \)
Best Stage Effort \( f = \)
Parasitic Delay \( P = \)
Delay \( D = \)
Example: 3-stage path

Logical Effort \[ G = \frac{4}{3} \times \frac{5}{3} \times \frac{5}{3} = \frac{100}{27} \]

Electrical Effort \[ H = \frac{45}{8} \]

Branching Effort \[ B = 3 \times 2 = 6 \]

Path Effort \[ F = GBH = 125 \]

Best Stage Effort \[ \sqrt[3]{F} = \sqrt[3]{125} = 5 \]

Parasitic Delay \[ P = 2 + 3 + 2 = 7 \]

Delay \[ D = 3 \times 5 + 7 = 22 = 4.4 \text{ FO4} \]

Example: 3-stage path

- Work backward for sizes
  \[ y = \]
  \[ x = \]

\[ x \]
\[ y \]
\[ A \]
\[ 8 \]
Example: 3-stage path

- Work backward for sizes
  \[ y = 45 \times \frac{5}{3} \div 5 = 15 \]
  \[ x = (15\times2) \times \frac{5}{3} \div 5 = 10 \]

Best Number of Stages

- How many stages should a path use?
  - Minimizing number of stages is not always fastest
- Example: drive 64-bit datapath with unit inverter

\[ D = \]
**Best Number of Stages**

- How many stages should a path use?
  - Minimizing number of stages is not always fastest
- Example: drive 64-bit datapath with unit inverter

\[
D = NF^{1/N} + P \\
= N(64)^{1/N} + N
\]

**Derivation**

- Consider adding inverters to end of path
  - How many give least delay?

\[
D = NF^{1/N} + \sum_{i=1}^{\eta} p_i + (N - \eta) p_{\text{inv}}
\]

\[
\frac{\partial D}{\partial N} = -F^{1/N} \ln F^{1/N} + F^{1/N} + p_{\text{inv}} = 0
\]

- Define best stage effort \( \rho = F^{1/N} \)

\[
p_{\text{inv}} + \rho \left(1 - \ln \rho \right) = 0
\]
Best Stage Effort

\[ p_{\text{inv}} + \rho (1 - \ln \rho) = 0 \]

- has no closed-form solution

- Neglecting parasitics \((p_{\text{inv}} = 0)\), we find \(\rho = 2.718\)

- For \(p_{\text{inv}} = 1\), solve numerically for \(\rho = 3.59\)

---

Review of Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Stage</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of stages</td>
<td>1</td>
<td>(N)</td>
</tr>
<tr>
<td>logical effort</td>
<td>(g)</td>
<td>(G = \prod g_i)</td>
</tr>
<tr>
<td>electrical effort</td>
<td>(h = \frac{c}{C_c})</td>
<td>(H = \sum_{\text{path}} h)</td>
</tr>
<tr>
<td>branching effort</td>
<td>(b = \frac{r_{\text{par}}}{})</td>
<td>(B = \sum b_i)</td>
</tr>
<tr>
<td>effort</td>
<td>(f = gh)</td>
<td>(F = G(RH))</td>
</tr>
<tr>
<td>effort delay</td>
<td>(f)</td>
<td>(D_f = \sum f)</td>
</tr>
<tr>
<td>parasitic delay</td>
<td>(p)</td>
<td>(P = \sum p_i)</td>
</tr>
<tr>
<td>delay</td>
<td>(d = f + p)</td>
<td>(D = \sum d_i = D_f + P)</td>
</tr>
</tbody>
</table>
Method of Logical Effort

1) Compute path effort \[ F = GBH \]
2) Estimate best number of stages \[ N = \log_4 F \]
3) Sketch path with \( N \) stages
4) Estimate least delay \[ D = NF^{1/3} + P \]
5) Determine best stage effort \[ \hat{f} = F^{1/3} \]
6) Find gate sizes

\[ C_{in_i} = \frac{g_i}{\hat{f}} C_{out_i} \]

Limits of Logical Effort

- Chicken and egg problem
  - Need path to compute \( G \)
  - But don’t know number of stages without \( G \)
- Simplistic delay model
  - Neglects input rise time effects
- Interconnect
  - Iteration required in designs with wire
- Maximum speed only
  - Not minimum area/power for constrained delay
Summary

- Logical effort is useful for thinking of delay in circuits
  - Numeric logical effort characterizes gates
  - NANDs are faster than NORs in CMOS
  - Paths are fastest when effort delays are ~4
  - Path delay is weakly sensitive to stages, sizes
  - But using fewer stages doesn’t mean faster paths
  - Delay of path is about \( \log_4 F \) FO4 inverter delays
  - Inverters and NAND2 best for driving large caps
- Provides language for discussing fast circuits
  - But requires practice to master

Power and Energy

- Power is drawn from a voltage source attached to the \( V_{DD} \) pin(s) of a chip.

- Instantaneous Power: \( P(t) = i_{DD}(t) V_{DD} \)

- Energy:
  \[
  E = \int_0^T P(t) dt = \int_0^T i_{DD}(t) V_{DD} dt
  \]

- Average Power:
  \[
  P_{avg} = \frac{E}{T} = \frac{1}{T} \int_0^T i_{DD}(t) V_{DD} dt
  \]
Dynamic Power

- Dynamic power is required to charge and discharge load capacitances when transistors switch.
- One cycle involves a rising and falling output.
- On rising output, charge \( Q = CV_{DD} \) is required
- On falling output, charge is dumped to GND
- This repeats \( T f_{sw} \) times over an interval of \( T \)

Dynamic Power Cont.

\[
P_{\text{dynamic}} =
\]
**Dynamic Power Cont.**

\[
P_{\text{dynamic}} = \frac{1}{T} \int_0^T i_{DD}(t) V_{DD} \, dt
\]

\[
= \frac{V_{DD}}{T} \int_0^T i_{DD}(t) \, dt
\]

\[
= \frac{V_{DD}}{T} \left[ T f_{sw} CV_{DD} \right]
\]

\[
= CV_{DD}^2 f_{sw}
\]

---

**Activity Factor**

- Suppose the system clock frequency = \( f \)
- Let \( f_{sw} = \alpha f \), where \( \alpha = \) activity factor
  - If the signal is a clock, \( \alpha = 1 \)
  - If the signal switches once per cycle, \( \alpha = \frac{1}{2} \)
  - Dynamic gates:
    - Switch either 0 or 2 times per cycle, \( \alpha = \frac{1}{2} \)
  - Static gates:
    - Depends on design, but typically \( \alpha = 0.1 \)
- Dynamic power: \[ P_{\text{dynamic}} = \alpha CV_{DD}^2 f \]
Short Circuit Current

- When transistors switch, both nMOS and pMOS networks may be momentarily ON at once
- Leads to a blip of “short circuit” current.
- < 10% of dynamic power if rise/fall times are comparable for input and output

Example

- 200 Mtransistor chip
  - 20M logic transistors
    - Average width: 12 $\lambda$
  - 180M memory transistors
    - Average width: 4 $\lambda$
  - 1.2 V 100 nm process
  - $C_g = 2 \text{ fF/}$μm
Dynamic Example

- Static CMOS logic gates: activity factor = 0.1
- Memory arrays: activity factor = 0.05 (many banks!)

- Estimate dynamic power consumption per MHz. Neglect wire capacitance and short-circuit current.

\[
\begin{align*}
C_{\text{logic}} &= (20 \times 10^6)(12\lambda)(0.05\mu m / \lambda)(2f E / \mu n) = 24 \text{ nF} \\
C_{\text{mem}} &= (180 \times 10^6)(4\lambda)(0.05\mu m / \lambda)(2f E / \mu n) = 72 \text{ nF} \\
P_{\text{dynamic}} &= \left[ 0.1C_{\text{logic}} + 0.05C_{\text{mem}} \right](1.2)^2 f = 8.6 \text{ mW/MHz}
\end{align*}
\]
Static Power

- Static power is consumed even when chip is quiescent.
  - Ratioed circuits burn power in fight between ON transistors
  - Leakage draws power from nominally OFF devices

\[ I_{ds} = I_{ds0} e^{\frac{V_{gs}-V_{th}}{V_{t}r}} \left[ 1 - e^{\frac{-V_{ds}}{V_{t}r}} \right] \]

\[ V_{r} = V_{r0} - \eta V_{ds} + \gamma \left( \sqrt{\phi_s} + V_{sb} - \sqrt{\phi_s} \right) \]

Ratio Example

- The chip contains a 32 word x 48 bit ROM
  - Uses pseudo-nMOS decoder and bitline pullups
  - On average, one wordline and 24 bitlines are high
- Find static power drawn by the ROM
  - \( \beta = 75 \mu A/V^2 \)
  - \( V_{tp} = -0.4V \)
Ratio Example

- The chip contains a 32 word x 48 bit ROM
  - Uses pseudo-nMOS decoder and bitline pullups
  - On average, one wordline and 24 bitlines are high
- Find static power drawn by the ROM
  - $\beta = 75 \mu A/V^2$
  - $V_{tp} = -0.4V$
- Solution:
  $$I_{pull-up} = \beta \left( V_{DD} - |V_s| \right)^2 = 24\mu A$$
  $$P_{pull-up} = V_{DD}I_{pull-up} = 29\mu W$$
  $$P_{static} = (31 + 24)P_{pull-up} = 1.6 \text{ mW}$$

Leakage Example

- The process has two threshold voltages and two oxide thicknesses.
- Subthreshold leakage:
  - 20 nA/\mu m for low $V_t$
  - 0.02 nA/\mu m for high $V_t$
- Gate leakage:
  - 3 nA/\mu m for thin oxide
  - 0.002 nA/\mu m for thick oxide
- Memories use low-leakage transistors everywhere
- Gates use low-leakage transistors on 80% of logic
Leakage Example Cont.

• Estimate static power:

  – High leakage: \((20 \times 10^6)(0.2)(12\lambda)(0.05\mu m/\lambda) = 2.4 \times 10^6 \mu m\)
  – Low leakage: \((20 \times 10^6)(0.8)(12\lambda)(0.05\mu m/\lambda) + \\
    (180 \times 10^6)(4\lambda)(0.05\mu m/\lambda) = 45.6 \times 10^6 \mu m\)

\[ I_{\text{static}} = \left(2.4 \times 10^6 \mu m\right)\left[\frac{(20nA/\mu m)}{2} + \frac{3nA/\mu m}{2}\right] + \\
\left(45.6 \times 10^6 \mu m\right)\left[\frac{(0.02nA/\mu m)}{2} + \frac{0.002nA/\mu m}{2}\right] \]

\[ = 32mA \]

\[ P_{\text{static}} = I_{\text{static}}V_{DD} = 38mW \]
Leakage Example Cont.

- Estimate static power: \(20 \times 10^6 \times 0.2 \times (12 \lambda) \times (0.05 \mu m / \lambda) = 2.4 \times 10^6 \mu W\)
  - High leakage: \(20 \times 10^6 \times 0.8 \times (12 \lambda) \times (0.05 \mu m / \lambda)\)
  - Low leakage: \(180 \times 10^6 \times 4 \lambda \times (0.05 \mu m / \lambda) = 45.6 \times 10^6 \mu W\)

\[ I_{\text{static}} = \left(2.4 \times 10^6 \mu W\right) \left[\frac{20\text{nA/}\mu m}{2 + (3\text{nA/}\mu m)}\right] + \left(45.6 \times 10^6 \mu W\right) \left[\frac{0.02\text{nA/}\mu m}{2 + (0.002\text{nA/}\mu m)}\right] = 32\text{mA}\]

\[ P_{\text{static}} = I_{\text{static}} V_{DD} = 38\text{mW} \]

- If no low leakage devices, \(P_{\text{static}} = 749\text{mW} (!)\)

Low Power Design

- Reduce dynamic power
  - \(\alpha\):
  - \(C\):
  - \(V_{DD}\):
  - \(f\):

- Reduce static power
Low Power Design

• Reduce dynamic power
  – \( \alpha \): clock gating, sleep mode
  – \( C \):
  – \( V_{DD} \):
  – \( f \):

• Reduce static power

Low Power Design

• Reduce dynamic power
  – \( \alpha \): clock gating, sleep mode
  – \( C \):
  – \( V_{DD} \):
  – \( f \):

• Reduce static power
Low Power Design

• Reduce dynamic power
  – α: clock gating, sleep mode
  – C: small transistors (esp. on clock), short wires
  – V_{DD}: lowest suitable voltage
  – f:

• Reduce static power
Low Power Design

• Reduce dynamic power
  – \(\alpha\): clock gating, sleep mode
  – C: small transistors (esp. on clock), short wires
  – \(V_{DD}\): lowest suitable voltage
  – f: lowest suitable frequency

• Reduce static power
  – Selectively use ratioed circuits
  – Selectively use low \(V_t\) devices
  – Leakage reduction:
    stacked devices, body bias, low temperature