What do ASIC/FPGA Designers need to know about physics?

- Physics effect:
  - Area $\rightarrow$ cost
  - Delay $\rightarrow$ performance
  - Energy $\rightarrow$ performance & cost

- Ideally, zero delay, area, and energy. However, the physical devices occupy area, take time, and consume energy.

- CMOS process lets us build transistors, wires, connections, and we get capacitors, inductors, and resistors whether or not we want them.
Performance, Cost, Power

- How do we measure performance? operations/sec? cycles/sec?

- Performance is directly proportional to clock frequency. Although it may not be the entire story:

  Ex: CPU performance
  \[ = \text{# instructions} \times \text{CPI} \times \text{clock period} \]
Limitations on Clock Rate

1. Logic Gate Delay

What are typical delay values?

2. Delays in flip-flops

Both times contribute to limiting the clock period.

• What must happen in one clock cycle for correct operation?
  – All signals connected to FF (or memory) inputs must be ready and “setup” before rising edge of clock.
  – For now we assume perfect clock distribution (all flip-flops see the clock at the same time).
Example

Parallel to serial converter circuit

\[ T \geq \text{time}(\text{clk} \rightarrow Q) + \text{time}(\text{mux}) + \text{time(setup)} \]

\[ T \geq \tau_{\text{clk} \rightarrow Q} + \tau_{\text{mux}} + \tau_{\text{setup}} \]
In General ...

For correct operation:

\[ T \geq \tau_{\text{clk} \rightarrow Q} + \tau_{\text{CL}} + \tau_{\text{setup}} \]

for all paths.

• How do we enumerate all paths?
  – Any circuit input or register output to any register input or circuit output?

• Note:
  – “setup time” for outputs is a function of what it connects to.
  – “clk-to-q” for circuit inputs depends on from where it comes.
Modern CMOS gate delays on the order of a few picoseconds. (However, highly dependent on gate design and context.)

Often expressed as FO4 delays (fan-out of 4) - as a process dependent delay metric:

- the delay of an inverter, driven by an inverter 4x smaller than itself, and driving an inverter 4x larger than itself.

- Less than 10ps for a 32nm process. For a 7nm process FO4 is around 2.5ps.
### Process Dependent FO4 Delay

Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm

Aaron Stillmaker\textsuperscript{a,b}, Bevan Baas\textsuperscript{a}

\textsuperscript{a} Department of Electrical and Computer Engineering, University of California, Davis, One Shields Ave., Davis, CA 95616, USA
\textsuperscript{b} Department of Electrical and Computer Engineering, California State University, Fresno, 2320 E. San Ramon Ave., Fresno, CA 93740, USA

Characteristics of different technology nodes [23]. The modeled measurements are for a single inverter in an FO4 chain. The energy value is the average energy required for a single inverter transition from low to high, or high to low.

<table>
<thead>
<tr>
<th>Production Year</th>
<th>Technology Node (nm)</th>
<th>Technology Type</th>
<th>$V_{DD}$ (V)</th>
<th>Simulated Performance of Inverter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Delay (ps)</td>
</tr>
<tr>
<td>1999</td>
<td>180</td>
<td>Bulk</td>
<td>1.8</td>
<td>77.2</td>
</tr>
<tr>
<td>2001</td>
<td>130</td>
<td>Bulk</td>
<td>1.2</td>
<td>34.7</td>
</tr>
<tr>
<td>2004</td>
<td>90</td>
<td>Bulk</td>
<td>1.1</td>
<td>26.5</td>
</tr>
<tr>
<td>2007</td>
<td>65</td>
<td>Bulk</td>
<td>1.1</td>
<td>19.8</td>
</tr>
<tr>
<td>2008</td>
<td>45</td>
<td>High-k</td>
<td>1.1</td>
<td>10.9</td>
</tr>
<tr>
<td>2010</td>
<td>32</td>
<td>High-k</td>
<td>0.97</td>
<td>9.8</td>
</tr>
<tr>
<td>2012</td>
<td>20</td>
<td>Multi-Gate</td>
<td>0.9</td>
<td>9.66</td>
</tr>
<tr>
<td>2013</td>
<td>16\textsuperscript{a}</td>
<td>Multi-Gate</td>
<td>0.86</td>
<td>6.12</td>
</tr>
<tr>
<td>2013</td>
<td>14\textsuperscript{a}</td>
<td>Multi-Gate</td>
<td>0.86</td>
<td>4.02</td>
</tr>
<tr>
<td>2015</td>
<td>10</td>
<td>Multi-Gate</td>
<td>0.83</td>
<td>3.24</td>
</tr>
<tr>
<td>2017</td>
<td>7</td>
<td>Multi-Gate</td>
<td>0.8</td>
<td>2.47</td>
</tr>
</tbody>
</table>

\textsuperscript{a} The 2013 ITRS report labels a single "16/14" node.
"Path Delay"

For correct operation:

\[
\text{Total Delay} \leq \text{clock\_period} - FF_{\text{setup\_time}} - FF_{\text{clk\_to\_q}}
\]
on all paths.

High-speed processors critical paths (worst case paths) have around 20 FO4 delays.
FO4 Delays per clock period

CPU Clock Periods 1985-2005

Thanks to Francois Labonte, Stanford
This figure shows how compiler optimizations have led to performance boosts in Libquantum.

SPEC 2006

Not surprisingly, specification data gets harder to find the older the processor becomes, especially for those that are no longer made, or worse, whose manufacturers no longer exist. We have been collecting this type of data for three decades and are now releasing it in the form of an open repository of processor specifications. The goal of CPU DB is to aggregate detailed processor specifications into a convenient form and to encourage community participation, both to leverage this information and to keep it accurate and current. CPU DB (cpudb.stanford.edu) is populated with desktop, laptop, and server processors, for which we use SPEC to benchmark performance.

With this open database, you can mine microprocessor trends over the past 40 years. Where did these incredible gains come from? This article sheds some light on this question by examining the improvements to processors from 1971 to 2011. We introduce a new tool—CPDB—that combines the capabilities of the microprocessor registry at Stanford with a collection of processor benchmarks to perform this analysis. We show that these improvements are due to the combined effects of technological scaling, compiler optimization, and software innovation, and that high-performance systems built with off-the-shelf processors are increasingly limited by bottlenecks such as memory latency and interconnect bandwidth. In addition, we provide a methodology to separate the effect of technology scaling from improvements on other frontiers (e.g., architecture and software), allowing the comparison of improvements on other frontiers (e.g., architecture and software), allowing the comparison of the impact of these different sources of improvement on processor performance.

In November 1971, Intel introduced the world's first single-chip microprocessor, the Intel 4004. It had 2,300 transistors, ran at a clock speed of up to 740 KHz, and delivered 10,000 instructions per second while dissipating 0.5 watts. The following four decades witnessed exponential growth in microprocessor performance from improvements in microarchitecture, compiler, and software technologies. While information about current processors is easy to find, it is rarely arranged in a manner that is conducive to understanding the motivation and impact of those improvements. With this database, you can mine microprocessor trends over the past 40 years.

With time and help from the community, we hope to extend the coverage of embedded processors in the database. Employ billions of transistors, include multiple processor cores on a single silicon die, run at clock speeds measured in gigahertz, and deliver more than 4 million times the performance of the original 4004.

Today's microprocessor chips can perform complex tasks such as folding, and computing real-time ballistic trajectories of angry birds. With this open database, you can mine microprocessor trends over the past 40 years.

Andrew Danowitz, Kyle Kelley, James Mao, John P. Stevenson, Mark Horowitz, Stanford University

F04 delays per cycle per cycle is roughly proportional to the amount of computation completed per cycle.
"Gate Delay"

- What determines the actual delay of a logic gate?
- Transistors are not perfect switches - cannot change terminal voltages instantaneously.
- Consider the NAND gate:

\[ \Delta t \propto \frac{C_L}{I} \]

- Current (I) value depends on: process parameters, transistor size
- \( C_L \) models gate output, wire, inputs to next stage (Cap. of Load)
- \( C \) “integrates” I creating a voltage change at output
More on transistor Current

- Transistors actually act like a cross between a resistor and "current source"

\[ I_{sat} \propto \frac{W}{L} \]

- \( I_{sat} \) depends on process parameters (higher for nFETs than for pFETs) and transistor size (layout):

FinFets use multiple "fins" to get wider
Physical Layout determines FET strength

- “Switch-level” abstraction gives a good way to understand the function of a circuit.
  - nFET \((g=1 \rightarrow \text{short circuit} : \text{open})\)
  - pFET \((g=0 \rightarrow \text{short circuit} : \text{open})\)
- Understanding delay means going below the switch-level abstraction to transistor physics and layout details.
Transistors as water valves. (Cartoon physics)

If electrons are water molecules, transistor strengths (W/L) are pipe diameters, and capacitors are buckets ...

A “on” p-FET fills up the capacitor with charge.

A “on” n-FET empties the bucket.
The Switch Inverter: Transient Response

\[ V(t) = V_0 e^{-t/RC} \]

\[ t_{1/2} = \ln(2) \times RC \]

\[ t_{pHL} = f(R_{on}C_L) = 0.69 \times R_n C_L \]

(a) Low-to-high

(b) High-to-low
Turning Rise/Fall Delay into Gate Delay

- Cascaded gates:

```
  1  0  1  1  0  1
  0  1  0  1  0  1
```

```
INV1.

V

Vdd/2

INV2.

V

Vdd/2

INV3.

V

Vdd/2

prop. delay = sum of individual prop. delays of gates in series.

In general:
More on gate delay

- Everything that connects to the output of a logic gate (or transistor) contributes capacitance:
  - Transistor drains
  - Interconnection (wires/contacts/vias)
  - Transistor Gates
  - Additionally, wires can contribute additional intrinsic delay
Wire Delay

- Ideally, wires behave as “transmission lines”:
  - signal wave-front moves close to the speed of light
    - ~1 ft/ns
  - Time from source to destination is called the “transit time”.
  - In ICs most wires are short, and the transit times are relatively short compared to the rise/fall times and can be ignored.
  - Not so on PC boards.
**Wires**

- As parallel plate capacitors:
  \[ C \propto \text{Area} = \text{width} \times \text{length} \]

- Wires have some finite resistance, so have distributed \( R \) and \( C \):
  \[ \Delta t \propto r c L^2 \approx r c + 2 r c + 3 r c + \ldots \]

with \( r = \text{res/length} \), \( c = \text{cap/length} \).
Wire Delay

- Even in those cases where the transmission line effect is negligible:
  - Wires possess distributed resistance and capacitance
  - Time constant associated with distributed RC is proportional to the square of the length

- For **short wires** on ICs, resistance is insignificant (relative to effective R of transistors), but C is important.
  - Typically around half of C of gate load is in the wires.

- For **long wires** on ICs:
  - Busses, clock lines, global control signal, etc.
  - Resistance is significant, therefore distributed RC effect dominates.
  - Signals are typically “rebuffered” to reduce delay

- For **long wires** on ICs with high currents:
  - Inductance is also important
Wire Rebuffering

- For **long wires** on ICs:
  - busses, clock lines, global control signal, etc.
  - Resistance is significant, therefore $r_c L^2$ effect dominates.
  - signals are typically “rebuffered” to reduce delay:

  ![Wire Rebuffering Diagram]

  **unbuffered wire** $\Delta t \propto L^2$

  **wire buffered into N sections** $\Delta t \propto N \cdot (L/N)^2 + (N-1) \cdot t_{\text{buffer}}$

  Assuming $t_{\text{buffer}}$ is small, $\Delta t \propto L^2/N$

  **Speedup:** $\propto N$
Flip-Flop delays eat into “time budget”

\[
T \geq \tau_{\text{clk}\rightarrow Q} + \tau_{\text{CL}} + \tau_{\text{setup}}
\]
Recall: Positive edge-triggered flip-flop

**Sampling circuit**

- Setup time results from delay through first latch.
- Clock to Q delay results from delay through second latch.

**Holds value**

- Holds value

**Timing Diagram**

- Input data must be stable in this period.
- Setup time
- Hold time
- "clk-to-q" delay
Sensing: When clock is low

A flip-flop “samples” right before the edge, and then “holds” value.

**Sampling circuit**

**Holds value**

\( \text{clk} = 0 \)
\( \text{clk}' = 1 \)

Will capture new value on posedge.

Outputs last value captured.
Capture: When clock goes high

A flip-flop “samples” right before the edge, and then “holds” value.

\[ \text{clk} = 1 \]
\[ \text{clk}' = 0 \]

Sampling circuit

Remembers value just captured.

Holds value

Outputs value just captured.
Flip Flop delays:

CLK == 0
Sense D, but Q outputs old value.

CLK 0->1
Capture D, pass value to Q

Note: with too much fanout, second stage could fail to capture data properly. Often output is rebuffed.
Some state elements have positive hold time requirements.

- How can this be?
- Fast paths from one state element to the next can create a violation. (Think about shift registers!)
- CAD tools do their best to fix violations by inserting delay (buffers).
  - Of course, if the path is delayed too much, then cycle time suffers.
  - Difficult because buffer insertion changes layout, which changes path delay.
Timing Analysis and Logic Delay

Some path somewhere in the design has the longest delay and is therefore the “critical path.”
Components of Combinational Path Delay

1. # of levels of logic
2. Internal cell delay
3. Wire delay
4. Cell input capacitance
5. Cell fanout
6. Cell output drive strength
Who controls the delay in ASIC?

<table>
<thead>
<tr>
<th></th>
<th>foundary engineer (TSMC)</th>
<th>Library Developer (Aristan)</th>
<th>CAD Tools (DC, IC Compiler)</th>
<th>Designer (you!)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. # of levels</td>
<td></td>
<td></td>
<td>synthesis</td>
<td>HDL design</td>
</tr>
<tr>
<td>2. Internal cell delay</td>
<td>physical parameters</td>
<td>cell topology, trans sizing</td>
<td>cell selection</td>
<td></td>
</tr>
<tr>
<td>3. Wire delay</td>
<td>physical parameters</td>
<td></td>
<td>place &amp; route</td>
<td>layout</td>
</tr>
<tr>
<td>4. Cell input capacitance</td>
<td>physical parameters</td>
<td>cell topology, trans sizing</td>
<td>cell selection</td>
<td></td>
</tr>
<tr>
<td>5. Cell fanout</td>
<td></td>
<td></td>
<td>synthesis</td>
<td>HDL design</td>
</tr>
<tr>
<td>6. Cell drive strength</td>
<td>physical parameters</td>
<td>transistor sizing</td>
<td>cell selection</td>
<td></td>
</tr>
</tbody>
</table>
Timing Closure: Searching for and beating down the critical path

Must consider all connected register pairs, paths, plus from input to register, plus register to output.

- Design tools help in the search.
- Synthesis tools work to meet clock constraint, report delays on paths,
  - Special static timing analyzers accept a design netlist and report path delays,
  - and, of course, simulators can be used to determine timing performance.

Tools that are expected to do something about the timing behavior (such as synthesizers), also include provisions for specifying input arrival times (relative to the clock), and output requirements (set-up times of next stage).
Timing Analysis, real example

The critical path

Most paths have hundreds of picoseconds to spare.

From "The circuit and physical design of the POWER4 microprocessor", IBM J Res and Dev, 46:1, Jan 2002, J.D. Warnock et al.
Timing Optimization

As an ASIC/FPGA designer you get to choose:

- The algorithm
- The Microarchitecture (block diagram)
- The HDL description of the CL blocks (number of levels of logic)
- Where to place registers and memory (the pipelining)
- Overall floorplan and relative placement of blocks
Circuit retiming

Critical path is 5 (ignore FF delay for now).
We want to improve it without changing circuit semantics.

Add a register, move one circle.
Performance improves by 20%.

Logic Synthesis tools can do this in simple cases.
Want to retime to here, however, delay cannot be added to the loop without changing the semantics of the logic. Because of this, many retiming tools stop at loops. Therefore move registers to after loop.
This is the retimed solution that many retiming aware tools will stop at.

*This is also the optimal solution when the initial values of the registers are not given.*

If the registers have the same initial condition, they can be combined and moved into the loop.
Retiming Example

If the registers have the same initial condition, they can be combined and moved into the loop

\[
\tau_{\text{Clk->Q}} = 10 \\
\tau_{\text{Setup}} = 20
\]

The register can be moved through the NAND gate, producing, a register at each input. The total delay in the loop is unchanged. Care must be taken to properly set the initial conditions of the registers.

Critical Path Delay = 110
Floorplanning: essential to meet timing.

(Intel XScale 80200)
Timing Analysis Tools

- Static Timing Analysis: Tools use delay models for gates and interconnect. Traces through circuit paths.
  - Cell delay model capture
    - For each input/output pair, internal delay (output load independent)
    - Output dependent delay
  - Standalone tools (PrimeTime) and part of logic synthesis.
  - Back-annotation takes information from results of place and route to improve accuracy of timing analysis.
  - DC in "topographical mode" uses preliminary layout information to model interconnect parasitics.
    - Prior versions used a simple fan-out model of gate loading.
Standard cell characterization

<table>
<thead>
<tr>
<th>Path</th>
<th>1.2V - 125°C</th>
<th>1.6V - 40°C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$In1 - t_{pLH}$</td>
<td>$0.073+7.98C+0.317T$</td>
<td>$0.020+2.73C+0.253T$</td>
</tr>
<tr>
<td>$In1 - t_{pHL}$</td>
<td>$0.069+8.43C+0.364T$</td>
<td>$0.018+2.14C+0.292T$</td>
</tr>
<tr>
<td>$In2 - t_{pLH}$</td>
<td>$0.101+7.97C+0.318T$</td>
<td>$0.026+2.38C+0.255T$</td>
</tr>
<tr>
<td>$In2 - t_{pHL}$</td>
<td>$0.097+8.42C+0.325T$</td>
<td>$0.023+2.14C+0.269T$</td>
</tr>
<tr>
<td>$In3 - t_{pLH}$</td>
<td>$0.120+8.00C+0.318T$</td>
<td>$0.031+2.37C+0.258T$</td>
</tr>
<tr>
<td>$In3 - t_{pHL}$</td>
<td>$0.110+8.41C+0.280T$</td>
<td>$0.027+2.15C+0.223T$</td>
</tr>
</tbody>
</table>

- Each library cell (FF, NAND, NOR, INV, etc.) and the variations on size (strength of the gate) is fully characterized across temperature, loading, etc.

- Power Supply Line ($V_{DD}$)
- Delay in (ns)!!

3-input NAND cell
(from ST Microelectronics):
C = Load capacitance
T = input rise/fall time

Ground Supply Line (GND)
Modeling Gate Delay
**Inverter Transient Response**

\[ V(t) = V_0 e^{-t/RC} \]

\[ t_{1/2} = \ln(2) \times RC \]

\[ t_{pHL} = f(R_{on}C_L) = 0.69 R_n C_L \]

(a) Low-to-high

(b) High-to-low
The Switch – Dynamic Model (Simplified)

\[ |V_{GS}| \geq |V_T| \]
Switch Sizing

What happens if we make a MOSFET \( W \) times larger (wider)

\[ |V_{GS}| \geq |V_T| \]

\[ C_G W \]

\[ C_S W \]

\[ R_{on}/W \]

\[ C_D W \]
Switch Parasitic Model

The pull-down switch (NMOS)

Minimum-size switch

Sizing the transistor (factor $W$)

We assume transistors of minimal length (or at least constant length). $R$’s and $C$’s in units of per unit width.
Switch Parasitic Model

For traditional CMOS processes, pFETs are ~twice as resistive as nFETS. (Mobility of holes is 1/2 that of electrons).

The pull-up switch (PMOS)

- Minimum-size switch
  - $V_{in}$
  - $R_P = 2R_N$
  - $C_G$
  - $V_{out}$
  - $C_D$

- Sized for symmetry
  - $V_{in}$
  - $R_N$
  - $2C_G$
  - $V_{out}$
  - $2C_D$

- General sizing
  - $V_{in}$
  - $R_N/W$
  - $V_{out}$
  - $2W C_G$
  - $2W C_D$
Inverter Parasitic Model

\[ C_{in} = 3WC_G \]

\[ C_{int} = 3WC_D = 3W\gamma C_G \]

\[ t_p = 0.69 \left( \frac{R_N}{W} \right) (3W\gamma C_G) = 0.69(3\gamma)R_N C_G \]

Drain and gate capacitance of transistor are directly related by process (\(\gamma \approx 1\))

\[ C_D = \gamma C_G \]

Intrinsic delay of inverter independent of size
Inverter with Load Capacitance

\[ t_p = 0.69 \left( \frac{R_N}{W} \right) (C_{int} + C_L) \]

\[ = 0.69 \left( \frac{R_N}{W} \right) (3W\gamma C_G + C_L) \]

\[ = 0.69 \left( 3\gamma R_N C_G \right) \left( 1 + \frac{C_L}{\gamma C_{in}} \right) \]

\[ = t_{p0} \left( 1 + \frac{C_L}{\gamma C_{in}} \right) = t_{p0} \left( 1 + \frac{f}{\gamma} \right) \]

\[ \gamma = \text{fanout} = \text{ratio of load capacitance (C_L) to and input capacitance (C_{in})} \]
Inverter Delay Model

Delay linearly proportional to fanout, $f$. For $f=0$, delay is $t_{inv}$

$$t_p = t_{p0}(1+f/\gamma)$$

Question: how does transistor sizing ($W$) impact delay?
Derive the formula for gate propagation as a function of fanout, $f$ (as with inverter)

We derive the equations based on the input connected to the transistor closest to the output ($A$), assuming the $B$ input had been set to 1 (for a long time)

So we can fairly compare to the inverter, size the transistors so that the capacitance of each input is equivalent to the input capacitance of the inverter

Assume that the resistance of the pFET is twice that of the nFET ($R_p = 2R_n$) if the pFET and nFET have the same width

Size the transistors so that the rise time and fall times are equivalent

For the 2 transistors in series, ignore the capacitance at their shared node
Solution setup:

- To keep the pullup and pulldown delays the same, $W_p = W_n$, because $R_p = 2R_n$.
- Remember, inverter had input C of $3C_G$, with $W_p = 2W_n$.
- Therefore, here we increase widths by $\frac{3}{2}$ relative to the inverter, so
- $R$ changes by $\frac{2}{3}$ (shown in figure).
2-input NAND Gate

Solve:

\[ C_{int} = (6/2)C_D + (3/2)C_D = (9/2)C_D \]
\[ t_p = 0.69 \cdot 2 \left( \frac{2R_N}{3W} \right) (C_{int} + C_L) \]
\[ \quad = 0.69 \left( \frac{4R_N}{3W} \right) \left( \frac{9}{2} \gamma W C_G + C_L \right) \]
\[ \quad = 0.69 \left( \frac{R_N}{W} \right) \left( 6\gamma W C_G + \frac{4}{3}C_L \right) \]
\[ \quad = 0.69 \left( \frac{R_N}{W} \right) 3\gamma W C_G \left( 2 + \frac{4C_L}{3\gamma (3W C_G)} \right) \]
\[ \quad = [0.69 \cdot 3R_N\gamma C_G] \left( 2 + \frac{4C_L}{3\gamma C_{IN}} \right) \]
\[ \quad = t_{p0} \left( 2 + \frac{4f}{3\gamma} \right) \]
2-input NOR Gate

\[ W_p = 4W_n \]

\[ C_{int} = 2 \times \left( \frac{3}{5} / C_D + \frac{12}{5} C_D \right) = \left( \frac{18}{5} \right) C_D \]

\[ t_p = 0.69 \left( \frac{3R_N}{5W} \right) (C_{int} + C_L) \]
\[ = 0.69 \left( \frac{5R_N}{3W} \right) \left( \frac{18}{5} \gamma W C_G + C_L \right) \]
\[ = 0.69 \left( \frac{R_N}{W} \right) \left( 6\gamma W C_G + \frac{5C_L}{3} \right) \]
\[ = 0.69 \left( \frac{R_N}{W} \right) 3\gamma W C_G \left( 2 + \frac{5C_L}{3 (3\gamma W C_G)} \right) \]
\[ = \left[ 0.69 \cdot 3R_N \gamma C_G \right] \left( 2 + \frac{5C_L}{3 \gamma C_{IN}} \right) \]
\[ = t_{p0} \left( 2 + \frac{5f}{3\gamma} \right) \]
The y-intercepts for NAND and NOR are both twice that of the inverter. The NAND line has a gradient $4/3$ that of the inverter (steeper); for NOR it is $5/3$ (steepest).

What about gates with more than 2-inputs?

Look at 4-input NAND:

$$t_p = t_{p0} \left( 4 + \frac{2f}{\gamma} \right)$$

intercept

slope
Adding Wires to gate delay

- Wires have finite resistance, so have distributed R and C:

\[ \Delta t \propto r c L^2 \approx r c + 2r c + 3r c + \ldots \]

- Wire propagation delay is around half of what it would be if R and C were “lumped”: \( t_p = 0.38(rL \times cL) = 0.38rcL^2 \)
Gate Driving long wire and other gates

\[ R_w = r_w L, \quad C_w = c_w L \]

\[ t_p = 0.69 R_{dr} C_{int} + 0.69 R_{dr} C_w + 0.38 R_w C_w + 0.69 R_{dr} C_{fan} + 0.69 R_w C_{fan} \]

\[ = 0.69 R_{dr} (C_{int} + C_{fan}) + 0.69 (R_{dr} c_w + r_w C_{fan}) L + 0.38 r_w c_w L^2 \]
Driving Large Loads

- Large fanout nets: clocks, resets, memory bit lines, off-chip
- Relatively small driver results in long rise time (and thus large gate delay)

Strategy:

- How to optimally scale drivers?
- Optimal trade-off between delay per stage and total number of stages?
Driving Large Loads

- For some given $C_L$:
  - How many stages are needed to minimize delay?
  - How to size the inverters?

- Get fastest delay if build one very big inverter
  - So big that delay is set only by self-loading

- Likely not the solution you’re interested in
  - Someone has to drive this inverter (a big inverter has a large input capacitance!) …
Delay Optimization

- First assume given:
  - A fixed number of inverters
  - The size* of the first inverter
  - The size of the load that needs to be driven
- What is the minimal delay of the inverter chain

*note: When we talk about inverter (or gate) “size”, we refer to the width of the transistors making up the circuit.
- Delay for the $j$-th inverter stage:

$$t_{p,j} = t_{p0} \left(1 + \frac{C_{g,j+1}}{\gamma C_{g,j}}\right) = t_{p0}(1 + f_j/\gamma)$$

- Total delay of the chain:

$$t_p = \sum_{j=1}^{N} t_{p,j} = t_{p0} \sum_{j=1}^{N} \left(1 + \frac{C_{g,j+1}}{\gamma C_{g,j}}\right)$$

$$C_{g,N+1} = C_L$$
Optimum Delay and Number of Stages

- Each inverter should be sized up by the same factor $f$ with respect to the preceding inverter.
- Therefore each stage has the same delay.
- Given $C_{g,1}$ and $C_L$.

\[
f = \sqrt[\sqrt[N]{F}]{C_L/C_{g,1}} = \sqrt[\sqrt[N]{F}]{F}
\]

- Where $F$ represents the overall fan-out of the circuit.
- The minimal delay through the chain is

\[
t_p = N \cdot t_{p0}(1 + \sqrt[\sqrt[N]{F}]{F}/\gamma)
\]
Example

\[ C_L = 8 \ C_1 \]

\[ C_L / C_1 \] has to be evenly distributed across \( N = 3 \) stages:
Delay Optimization

- Now assume given:
  - The size of the first inverter
  - The size of the load that needs to be driven
- Minimize delay by finding optimal number and sizes of gates
- So, need to find $N$ that minimizes:

$$t_p = N \cdot t_{p0}(1 + \sqrt[1]{F/\gamma}), \quad F = \frac{C_L}{C_{g,1}}$$
Finding optimal fanout per stage

\[ t_p = N \cdot t_{p0}(1 + \sqrt[4]{F/\gamma}), \quad F = \frac{C_L}{C_g,1} \]

- Differentiate w.r.t. \( N \) and set \( = 0 \):

\[ \gamma + \sqrt[4]{F} - \frac{\sqrt[4]{F} \ln F}{N} = 0 \]

\[ \Rightarrow f = e^{(1+\gamma/f)} \]

- Closed form only if:

\[ \gamma = 0 \quad \Rightarrow \quad N = \ln(F), \quad f = e \]
Optimum Effective Fanout $f$

- Optimum $f$ for given process defined by $\gamma$

$$f = e^{(1+\gamma/f)}$$

For $\gamma = 1$

$$f_{opt} = 3.6$$
In Practice: Plot of Total Delay

- Why the shape?
- Curves very flat for $f > 2$
  - Simplest/most common choice: $f = 4$

[Hodges, p.281]
Normalized Delay As a Function of $F$

$$t_p = N \cdot t_{p0} \left(1 + \frac{N \sqrt{F/\gamma}}{\gamma}\right), \quad F = \frac{C_L}{C_g,1}$$

<table>
<thead>
<tr>
<th>$F$</th>
<th>Unbuffered</th>
<th>Two Stage</th>
<th>Inverter Chain</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>11</td>
<td>8.3</td>
<td>8.3</td>
</tr>
<tr>
<td>100</td>
<td>101</td>
<td>22</td>
<td>16.5</td>
</tr>
<tr>
<td>1000</td>
<td>1001</td>
<td>65</td>
<td>24.8</td>
</tr>
<tr>
<td>10,000</td>
<td>10,001</td>
<td>202</td>
<td>33.1</td>
</tr>
</tbody>
</table>

$(\gamma = 1)$

[Rabaey: page 210]