EECS 151/251A
Spring 2018
Digital Design and Integrated Circuits

Instructors:
Nick Weaver & John Wawrzynek

Lecture 10
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
  $= \# \text{ instructions} \times CPI \times \text{clock period}$
Limitations on Clock Rate

1 Logic Gate Delay

What are typical delay values?

- 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).

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

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

\[ T \geq \tau_{\text{clk} \rightarrow Q} + \tau_{\text{mux}} + \tau_{\text{setup}} \]
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 where it comes from.
Modern CMOS gate delays on the order of a few picoseconds. (However, highly dependent on gate context.)

- Often expressed as FO4 delays (fan-out of 4) - as a process independent delay metric:
  - the delay of an inverter, driven by an inverter 4x smaller than itself, and driving an inverter 4x larger than itself.
  - For a 90nm process FO4 is around 20ps. Less than 10ps for a 32nm process.
"Path Delay"

- For correct operation:
  \[ \text{Total Delay} \leq \text{clock\_period} - \text{FF}_{\text{setup\_time}} - \text{FF}_{\text{clk\_to\_q}} \]
on all paths.

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

CPU Clock Periods 1985-2005

MIPS 2000
5 pipeline stages

Pentium Pro
10 pipeline stages

Pentium 4
20 pipeline stages

Historical limit: about 12

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

SPEC 2006

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

CPU DB: Recording History

With time and help from the community, we hope to extend the coverage of embedded processors in the database.

Although 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 (cpudb.stanford.edu) is to aggregate detailed processor information and to keep it accurate and current. CPU DB is populated with information about commercial processors from 17 manufacturers and placed in an extensible database collected by Stanford's Electronic Design Automation Lab.

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 provide a methodology to separate the effect of technology scaling from improvements on other frontiers (e.g., architecture and software), allowing the comparison of machines built in different technologies. To demonstrate the utility of this data and analysis, we use introducing CPU DB (cpudb.stanford.edu), an open and extensible database collected by Stanford's Electronic Design Automation Lab.

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 60,000 instructions per second while dissipating 0.5 watts. The following four decades witnessed exponential growth in compute power, a trend that has enabled applications as diverse as climate modeling, protein folding, and computing real-time ballistic trajectories of angry birds. Today's microprocessor chips 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.

Where did these incredible gains come from? This article sheds some light on this question by examining how the performance of processor chips is measured, the distribution of processor speeds and performance, and how compiler optimizations contribute to processor performance.

F04 Delays Per Cycle for Processor Designs

F04 delay 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:

  - Current (I) value depends on: process parameters, transistor size

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

  - \( C_L \) 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 act like a cross between a resistor and “current source”

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

\[ I_{SAT} \propto \frac{W}{L} \]
Physical Layout determines FET strength

- “Switch-level” abstraction gives a good way to understand the function of a circuit.
  - nFET (g=1 ? short circuit : open)
  - pFET (g=0 ? short circuit : open)
- Understanding delay means going below the switch-level abstraction to transistor physics and layout details.
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 – Dynamic Model (Simplified)

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

Diagram showing the switch with labels for G, S, D, C_s, C_G, and R_{on}.
Switch Sizing

What happens if we make a switch $W$ times larger (wider)

$|V_{GS}| \geq |V_T|$

$C_S W$ $R_{on} W$ $C_D W$

$S \quad G \quad W \quad D$

$V_{GS}$
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

The pull-up switch (PMOS)

Minimum-size switch

Sized for symmetry

General sizing
Inverter Parasitic Model

\[ C_{in} = 3WC_G \]

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

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

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
Turning Rise/Fall Delay into Gate Delay

- Cascaded gates:

“transfer curve” for inverter.
The Switch Inverter: Transient Response

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

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

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

\( \begin{align*} \text{(a) Low-to-high} \quad & \text{\hspace{2cm}} \text{(b) High-to-low} \\ \end{align*} \)
More on $C_L$

- Everything that connects to the output of a logic gate (or transistor) contributes capacitance:
  - Transistor drains
  - Interconnection (wires/contacts/vias)
  - Transistor Gates
Inverter with Load Capacitance

\[ V_{in} = 3WC_G \]

\[ V_{out} = R_N \frac{W}{C_{int}} \]

\[ C_{int} = 3WR_N \]

\[ C_L \]

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

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

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

\[ = t_{inv}(\gamma + \frac{C_L}{C_{in}}) = t_0(\gamma + f) \]

\[ f = \text{fanout} = \text{ratio between load and input capacitance of gate} \]
Inverter Delay Model

\[ t_p = t_{inv}(\gamma + f) \]

- \( t_{inv} \) technology constant
  - Can be dropped from expression
  - Delay unit-less variable (expressed in unit delays)

Question: how does transistor sizing (W) impact 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 clock period 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 finite resistance, so have distributed R and C:

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

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:

\[ v_1, v_2, v_3, v_4 \]
Recall: Positive edge-triggered flip-flop

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

Sampling circuit

Holds value
Sensing: When clock is low

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

Sampling circuit

Holds value

\[ D \rightarrow Q \]

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

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.

**Sampling circuit**
- \( D \)
- \( Q \)
- \( \text{clk} \)
- \( \text{clk'} \)

**Holds value**
- \( \text{clk} \)
- \( \text{clk'} \)
- \( Q \)

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

**Remembers value just captured.**
**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
Timing Analysis and Logic Delay

- Critical path: the slowest path between any two storage elements.
- A microprocessor fetches instructions and updates all functional units.
- If our clock period $T >$ worst-case delay through $CL$, does this ensure correct operation?

Register:

An Array of Flip-Flops

If our clock period $T >$ worst-case delay through $CL$, does this ensure correct operation?

Combinational Logic

If our clock period $T >$ worst-case delay through $CL$, does this ensure correct operation?

If our clock period $T >$ worst-case delay through $CL$, does this ensure correct operation?
Flip-Flop delays eat into “time budget”

\[
T \geq \tau_{\text{clk}\rightarrow Q} + \tau_{\text{CL}} + \tau_{\text{setup}}
\]
Clock skew also eats into “time budget”

As $T \to 0$, which circuit fails first?

$T \geq T_{CL} + T_{\text{setup}} + T_{\text{clk} \to Q} + \text{worst case skew}$. 

\[
\begin{align*}
T & \geq T_{\text{CL}} + T_{\text{setup}} + T_{\text{clk} \to Q} + \text{worst case skew}.
\end{align*}
\]
Clock Tree
Delays,
IBM “Power”
CPU

- Grid
- Tuned sector trees
- Sector buffers

Delay

Buffer level 2
Buffer level 1
Clock Tree Delays, IBM Power
Clock Skew (cont.)

- Note reversed buffer.
- In this case, clock skew actually provides *extra time* (adds to the effective clock period).
- This effect has been used to help run circuits as higher clock rates. Risky business!
Components of 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?

<table>
<thead>
<tr>
<th># of levels</th>
<th>foundary engineer (TSMC)</th>
<th>Library Developer (Aritsan)</th>
<th>CAD Tools (DC, IC Compiler)</th>
<th>Designer (you!)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.</strong></td>
<td></td>
<td></td>
<td>synthesis</td>
<td>RTL</td>
</tr>
<tr>
<td><strong>2. Internal cell delay</strong></td>
<td>physical parameters</td>
<td>cell topology, trans sizing</td>
<td>cell selection</td>
<td></td>
</tr>
<tr>
<td><strong>3. Wire delay</strong></td>
<td>physical parameters</td>
<td></td>
<td>place &amp; route</td>
<td>layout generator</td>
</tr>
<tr>
<td><strong>4. Cell input capacitance</strong></td>
<td>physical parameters</td>
<td>cell topology, trans sizing</td>
<td>cell selection</td>
<td>instantiation</td>
</tr>
<tr>
<td><strong>5. Cell fanout</strong></td>
<td></td>
<td></td>
<td>synthesis</td>
<td>RTL</td>
</tr>
<tr>
<td><strong>6. Cell drive strength</strong></td>
<td>physical parameters</td>
<td>transistor sizing</td>
<td>cell selection</td>
<td>instantiation</td>
</tr>
</tbody>
</table>
Timing Analysis
What is the smallest $T$ that produces correct operation? Or, can we meet a target $T$?

<table>
<thead>
<tr>
<th>$f$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MHz</td>
<td>1 $\mu$s</td>
</tr>
<tr>
<td>10 MHz</td>
<td>100 ns</td>
</tr>
<tr>
<td>100 MHz</td>
<td>10 ns</td>
</tr>
<tr>
<td>1 GHz</td>
<td>1 ns</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.

Timing Optimization

As an ASIC/FPGA designer you get to choose:

- The algorithm
- The Microarchitecture (block diagram)
- The RTL 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
How to retime logic

Critical path is 5. We want to improve it without changing circuit semantics.

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

Circles are combinational logic, labelled with delays.

Figure 1: A small graph before retiming. The nodes represent logic delays, with the inputs and outputs passing through mandatory, fixed registers. The critical path is 5.

Figure 2: The example in Figure 2 after retiming. The critical path is reduced from 5 to 4.

Logic Synthesis tools can do this in simple cases.
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.
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.
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?
For some given $C_L$:
- How many stages are needed to minimize delay?
- How to size the inverters?
Anyone want to guess the solution?
Careful about Optimization Problems

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

- Likely not the problem you’re interested in
  - Someone has to drive this inverter…
Delay Optimization Problem #1

- You are given:
  - A fixed number of inverters
  - The size of the first inverter
  - The size of the load that needs to be driven

- Your goal:
  - Minimize the delay of the inverter chain

- Need model for inverter delay vs. size
Apply to Inverter Chain

\[
\begin{align*}
    t_p &= t_{p1} + t_{p2} + \ldots + t_{pN} \\
    t_{pj} &= t_{inv} \left( \gamma + \frac{C_{in,j+1}}{C_{in,j}} \right) \\
    t_p &= \sum_{j=1}^{N} t_{p,j} = t_{inv} \sum_{i=1}^{N} \left( \gamma + \frac{C_{in,j+1}}{C_{in,j}} \right), \quad C_{in,N+1} = C_L
\end{align*}
\]
Optimal Sizing for Given N

- Delay equation has $N-1$ unknowns, $C_{in,2}, \ldots, C_{in,N}$

- To minimize the delay, find $N-1$ partial derivatives:

\[
 t_P = \ldots + t_{inv} \frac{C_{in,j}}{C_{in,j-1}} + t_{inv} \frac{C_{in,j+1}}{C_{in,j}} + \ldots
\]

\[
 \frac{dt_p}{dC_{in,j}} = t_{inv} \frac{1}{C_{in,j-1}} - t_{inv} \frac{C_{in,j+1}}{C_{in,j}^2} = 0
\]
Result: every stage has **equal fanout** (f):

\[
\frac{C_{in,j}}{C_{in,j-1}} = \frac{C_{in,j+1}}{C_{in,j}}
\]

Size of each stage is geometric mean of two neighbors:

\[
C_{in,j} = \sqrt{C_{in,j-1}C_{in,j+1}}
\]

Equal fanout $\rightarrow$ every stage will have same delay
Optimum Delay and Number of Stages

- When each stage has same fanout $f$:
  \[ f^N = F = \frac{C_L}{C_{in,1}} \]
- Fanout of each stage:
  \[ f = \sqrt[N]{F} \]
- Minimum path delay:
  \[ t_p = N t_{inv} \left( \gamma + \sqrt[N]{F} \right) \]
Example

\[ C_L = 8 C_1 \]

\( C_L/C_1 \) has to be evenly distributed across \( N = 3 \) stages:
Delay Optimization Problem #2

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

$$t_p = N t_{\text{inv}} \left( \gamma + \sqrt[2^n]{C_L / C_{\text{in}}} \right)$$
Untangling the Optimization Problem

- Rewrite N in terms of fanout/stage $f$:

$$f^N = \frac{C_L}{C_{in}} \rightarrow N = \frac{\ln \left( \frac{C_L}{C_{in}} \right)}{\ln f}$$

$$t_p = N t_{inv} \left( \left( \frac{C_L}{C_{in}} \right)^{1/N} + \gamma \right) = t_{inv} \ln \left( \frac{C_L}{C_{in}} \right) \left( \frac{f + \gamma}{\ln f} \right)$$

$$\frac{\partial t_p}{\partial f} = t_{inv} \ln \left( \frac{C_L}{C_{in}} \right) \cdot \frac{\ln f - 1 - \gamma / f}{\ln^2 f} = 0$$

$$f = \exp \left( 1 + \frac{\gamma}{f} \right)$$  (no explicit solution)

For $\gamma = 0$, $f = e$, $N = \ln \left( \frac{C_L}{C_{in}} \right)$
Optimum Effective Fanout $f$

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

\[ f = \exp(1 + \frac{\gamma}{f}) \]

- $f_{opt} = 3.6$ for $\gamma = 1$
Why the shape?
Curves very flat for $f > 2$
- Simplest/most common choice: $f = 4$

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

\[ t_p = Nt_{inv} \left( \gamma + \sqrt[\gamma]{F} \right), \quad F = \frac{C_L}{C_{in}} \]

<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]
Conclusion

‣ Timing Optimization: You start with a target on clock period. What control do you have?

‣ Biggest effect is RTL manipulation.
  ‣ i.e., how much logic to put in each pipeline stage.
  ‣ We will be talking later about how to manipulate RTL for better timing results.

‣ In most cases, the tools will do a good job at logic/circuit level:
  ‣ Logic level manipulation
  ‣ Transistor sizing
  ‣ Buffer insertion
  ‣ But some cases may be difficult and you may need to help

‣ The tools will need some help at the floorpan and layout