CS152: Computer Systems Architecture
Circuits Recap – Digital Why And How

Sang-Woo Jun
Spring 2023

Large amount of material adapted from MIT 6.004, “Computation Structures”,
and CS 152 Slides by Isaac Scherson
Course outline

- Part 1: The Hardware-Software Interface
  - What makes a ‘good’ processor?
  - Assembly programming and conventions

- Part 2: Recap of digital design
  - Combinational and sequential circuits
  - How their restrictions influence processor design

- Part 3: Computer Architecture
  - Computer Arithmetic
  - Simple and pipelined processors
  - Caches and the memory hierarchy

- Part 4: Computer Systems
  - Operating systems, Virtual memory
The digital abstraction

“Building Digital Systems in an Analog World”
The digital abstraction

- Electrical signals in the real world is analog
  - Continuous signals in terms of voltage, current,

- Modern computers represent and process information using discrete representations
  - Typically binary (bits)
  - Encoded using ranges of physical quantities (typically voltage)
Aside: Historical analog computers

- Computers based on analog principles have existed
  - Uses analog characteristics of capacitors, inductors, resistors, etc to model complex mathematical formulas
    - Very fast differential equation solutions!
    - Example: Solving circuit simulation would be very easy if we had the circuit and was measuring it

- Some modern resurgence as well!
  - Research on sub-modules performing fast non-linear computation using analog circuitry

Why are digital systems desirable?  
Hint: Noise

Polish analog computer AKAT-1 (1959)
Source: Topory

https://semiengineering.com/can-analog-make-a-comeback/
Using voltage digitally

Key idea
- Encode two symbols, “0” and “1” (1 bit) in an analog space
- And use the same convention for every component and wire in system

Problem: There is always noise between transmitter and receiver

Also, noise can accumulate as we pass through more gates

Source: MIT 6.004 2019 L05
Using voltage digitally

- Key idea
  - Encode two symbols, “0” and “1” (1 bit) in an analog space
  - And use the same convention for every component and wire in system

Source: MIT 6.004 2019 L05
Handling noise

- When a signal travels between two entities, **there will be noise**
  - Temperature, electromagnetic fields, interaction with surrounding modules, ...

- What if $V_{\text{out}}$ is barely lower than $V_L$, or barely higher than $V_H$?  
  - Noise may push the signal into invalid range
  - Rest of the system runs into undefined state!

- **Solution:** Output signals use a stricter range than input

<table>
<thead>
<tr>
<th>Enforced output</th>
<th>Valid “0”</th>
<th>Undefined</th>
<th>Valid “1”</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Interpreted input</th>
<th>Valid “0”</th>
<th>Undefined</th>
<th>Valid “1”</th>
</tr>
</thead>
</table>

Noise margin

*Source: MIT 6.004 2019 L05*
Voltage Transfer Characteristic

- Example component: Buffer
  - A simple digital device that copies its input value to its output

- Voltage Transfer Characteristic (VTC):
  - Plot of $V_{out}$ vs. $V_{in}$ where each measurement is taken after any transients have died out.
  - Not a measure of circuit speed!
    - Only determines behavior under static input

- Each component generates a new, “clean” signal!
  - Noise from previous component corrected

Source: MIT 6.004 2019 L05
Benefits of digital systems

- Digital components are “restorative”
  - Noise is cancelled at each digital component
  - Very complex designs can be constructed on the abstraction of digital behavior

- Compare to analog components
  - Noise is accumulated at each component
  - Lay example: Analog television signals! (Before 2000s)
    - Limitation in range, resolution due to transmission noise and noise accumulation
    - Contrary: digital signals use repeaters and buffers to maintain clean signals

Source: “Does TV static have anything to do with the Big Bang?” How it works, 2012
The basic building block: CMOS transistors ("Complementary Metal–Oxide–Semiconductor")

Everything is built as a network of transistors!
The basic building block: CMOS FETs

- Remember CS151 – FETs come in two varieties, and are composed to create Boolean logic

![CMOS NAND Gate diagram](image)

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

Source: MIT 6.004 2019 L09
Making chips out of transistors…?
The basic building block 2: Standard cell library

- **Standard cell**
  - Group of transistor and interconnect structures that provides a boolean logic function
    - Inverter, buffer, AND, OR, XOR, ...
  - For a specific implementation technology/vendor/etc...
  - Also includes physical characteristic information

- **Eventually, chips designs are expressed as a group of standard cells networked via wires**
  - Among what is sent to a fab plant

<table>
<thead>
<tr>
<th>Gate</th>
<th>Delay (ps)</th>
<th>Area (μ²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverter</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Buffer</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>AND2</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>NAND2</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>OR2</td>
<td>55</td>
<td>26</td>
</tr>
<tr>
<td>NOR2</td>
<td>35</td>
<td>16</td>
</tr>
<tr>
<td>AND4</td>
<td>90</td>
<td>40</td>
</tr>
<tr>
<td>NAND4</td>
<td>70</td>
<td>30</td>
</tr>
<tr>
<td>OR4</td>
<td>100</td>
<td>42</td>
</tr>
<tr>
<td>NOR4</td>
<td>80</td>
<td>32</td>
</tr>
</tbody>
</table>

Example:

Various components have different delays and area! The actual numbers are not important right now.

Source: MIT 6.004 2019 L06
Aside: Describing chips for foundries

- GDSII, OASIS file formats
- Depicts many standard cells connected via multiple wire layers

Source: File:Silicon_chip_3d.png, Tgrebinski, File:Wikipediaoasisimage 2.png (Wikipedia)
CS152: Computer Systems Architecture
Digital Circuit Design Recap

Sang-Woo Jun
Spring 2023

Large amount of material adapted from MIT 6.004, “Computation Structures”, Morgan Kaufmann “Computer Organization and Design: The Hardware/Software Interface: RISC-V Edition”, and CS 152 Slides by Isaac Scherson
Combinational and sequential circuits

- Two types of digital circuits
  - Combinational circuit
    - Output is a function of current input values
      - output = f(input)
      - Output depends exclusively on input
  - Sequential circuit
    - Have memory ("state")
      - Output depends on the "sequence" of past inputs
What constitutes combinational circuits

1. Input
2. Output
3. Functional specifications
   - The value of the output depending on the input
   - Defined in many ways!
   - Boolean logic, truth tables, hardware description languages,

4. Timing specifications
   HINTED AT IN CS51/CS151
   - Given dynamic input, how does the output change over time?
Timing specifications of combinational circuits

- Propagation delay ($t_{PD}$)
  - An upper bound on the delay from valid inputs to valid outputs
  - Restricts how fast input can be consumed
    (Too fast input $\rightarrow$ output cannot change in time, or undefined output)

A good circuit has low $t_{PD}$
$\rightarrow$ Faster input
$\rightarrow$ Higher performance

How do we get low $t_{PD}$?

Source: MIT 6.004 2019 L05
Timing specifications of combinational circuits

- Contamination delay ($t_{CD}$)
  - A lower bound on the delay between input change to output starting to change
    - Does not mean output has stable value!
  - Guarantees that output will not change within this timeframe regardless of what happens to input

Example: Inverter

Source: MIT 6.004 2019 L05
Back to propagation delay of combinational circuits

- A chain of logic components has additive delay
  - The “depth” of combinational circuits is important
- The “critical path” defines the overall propagation delay of a circuit

**Example:** A full adder

\[
\text{Critical path of three components: } t_{PD} = t_{PD}(\text{xor}2) + t_{PD}(\text{and}2) + t_{PD}(\text{or}2)
\]
Sequential circuits

- Combinational circuits on their own are not very useful
- Sequential logic has memory ("state")
  - State acts as input to internal combinational circuit
  - Subset of the combinational circuit output updates state
Synchronous sequential circuits

- “Synchronous”: all operations are aligned to a shared clock signal
  - Speed of the circuit determined by the delay of its longest critical path
  - For correct operation, all paths must be shorter than clock speed
  - Either simplify logic, or reduce clock speed!
Timing constraints of state elements

- Synchronous state elements also add timing complexities
  - Beyond propagation delay and contamination delay
- Propagation delay ($t_{PD}$) of state elements
  - Rising edge of the clock to valid output from state element
- Contamination delay ($t_{CD}$)
  - State element output should not change for $t_{CD}$ after clock change
- Setup time ($t_{SETUP}$)
  - State element should have held correct data for $t_{SETUP}$ before clock edge
- Hold time ($t_{HOLD}$)
  - Input to state element should hold correct data for $t_{HOLD}$ after clock edge
Timing behavior of state elements

Meeting the **setup time** constraint

- “Processing must fit in clock cycle”
- After rising clock edge,
- \( t_{PD}(\text{State element 1}) + t_{PD}(\text{Combinational logic}) + t_{SETUP}(\text{State element 2}) \)
- must be **smaller** than the clock period

Otherwise, “timing violation”
Meeting the hold time constraint

- "Processing should not effect next state too early"
- After rising clock edge,
- $t_{CD}(\text{State element 1}) + t_{CD}(\text{Combinational logic})$ must be larger than $t_{\text{HOLD}}(\text{State element 2})$

Guaranteed time output will not change
Setup and hold time window

If any constraint is violated, state may hold wrong data!
Real-world implications

- Constraints are met via Computer-Aided Design (CAD) tools
  - Cannot do by hand!
  - Given a high-level representation of function, CAD tools will try to create a physical circuit representation that meets all constraints

- Rule of thumb: Meeting **hold time** is typically not difficult
  - e.g., Adding a bunch of buffers can add enough $t_{CD}$ (Sequential Circuit)

- Rule of thumb: Meeting **setup time** is often difficult
  - Somehow construct shorter critical paths, or
  - reduce clock speed (We want to avoid this!)

How do we create shorter critical paths for the same function?
Simplified introduction to placement/routing

- Mapping state elements and combinational circuits to limited chip space
  - Also done via CAD tools
  - May add significant propagation delay to combinational circuits

- Example:
  - Complex combinational circuits 1 and 2 accessing state A
  - Spatial constraints push combinational circuit 4 far from state A
  - Path from B to A via 4 is now very long!

- Rule of thumb:
  - One comb. should not access too many state
  - One state should not be used by too many comb.
Looking back:
Why are register files small?

- Why are register files 32-element? Why not 1024 or more?

Hierarchical design of an 8x1 multiplexer:

Propagation delay increases with more registers!
Real-world example

- Back in 2002 (When frequency scaling was going strong, but larger FETs)
  - Very high frequency (multi-GHz) meant:
  - ... setup time constraint could tolerate
  - ... up to 8 inverters in its critical path
  - Such stringent restrictions!

  Can we even fit a 32-bit adder there? No!