Objective

- Understand the intricacies of advanced IC design and development
Objective

- Understand the intricacies of advanced IC design and development

**RTL code**

```verilog
module dff(input clk, input d, output q);
    always @(posedge clk)
    q <= d;
endmodule
```

...
Objective

• Understand the intricacies of advanced IC design and development

RTL code

module dff(input clk, input d, output q);
    always @(posedge clk)
        q <= d;
endmodule

...
PD Contents

1. Moore’s Law and Dennard Scaling
2. Hardware Design Cycle
3. Functional Verification
4. Circuit Design
5. Physical Design
Evaluation

- Presentation on related research work
  - PechaKucha format: 20 slides, 20 seconds/slide

- List of papers posted soon in PD web site
  - http://jarnau.site.ac.upc.edu/PD/

- 20% of final mark
Practical approach

• Theory sessions are useful for lab assignments
  – How to write effective test benches
  – How to verify your design
  – How to synthesize, place and route your design
  – How to estimate max freq and area of your design

• EDA tools for digital synthesis
  – Qflow: http://opencircuitdesign.com/qflow/
PD Contents

1. Moore’s Law and Dennard Scaling
   (a) Transistor basic physics
   (b) Power wall
   (c) Dark silicon

2. Hardware Design Cycle

3. Functional Verification

4. Circuit Design

5. Physical Design
1. Moore’s Law and Dennard Scaling
Moore’s Law
The number of transistors in a dense integrated circuit doubles about every two years.

1. Moore’s Law and Dennard Scaling
Moore’s Law
The number of transistors in a dense integrated circuit doubles about every two years.

Dennard Scaling
As transistors shrink, they become faster, consume less power, and are cheaper to manufacture.

1. Moore’s Law and Dennard Scaling
MOS Transistors

- Metal Oxide Semiconductor

nMOS Transistor

pMOS Transistor
MOS Transistors

nMOS

pMOS
MOS Transistors

nMOS

\[\text{gate} \quad | \quad \text{drain} \quad | \quad \text{source}\]

pMOS

\[\text{gate} \quad | \quad \text{drain} \quad | \quad \text{source}\]

gate = 0 Æ OFF
MOS Transistors

nMOS

gate

source

drain

pMOS

gate

source

drain

gate = 0  OFF

gate = 1  ON
MOS Transistors

nMOS

pMOS

gate = 0 → OFF

gate = 1 → ON
MOS Transistors

nMOS

pMOS

OFF

ON

gate = 0

gate = 1

OFF

ON

gate = 0

gate = 1
CMOS Logic

- Inverter
CMOS Logic

- Inverter
CMOS Logic

- Inverter
CMOS Logic

- Inverter
CMOS Logic

- Inverter

\[ V_{DD} \]
\[ A \]
\[ 0 \]
\[ 1 \]
\[ GND \]

A \rightarrow Y
CMOS Logic

- Inverter

\[ V_{DD} \]

\[ A \]

\[ 0 \]

\[ \text{ON} \]

\[ \text{OFF} \]

\[ Y \]

\[ 1 \]

\[ \text{GND} \]

\[ V_{DD} \]

\[ A \]

\[ \text{GND} \]
CMOS Logic

- Inverter

\[ V_{DD} \]
\[ \text{ON} \]
\[ \text{OFF} \]
\[ \text{GND} \]

\[ A \]
\[ 0 \]
\[ 1 \]

\[ V_{DD} \]
\[ \text{Y} \]
\[ \text{GND} \]

\[ A \]
\[ 1 \]

\[ Y \]
CMOS Logic

- Inverter

\[
\begin{align*}
\text{A} &\quad \text{Y} \\
0 &\quad 1 \\
\text{GND} &
\end{align*}
\]
CMOS Logic

- Inverter

\[ A \quad 0 \quad \text{ON} \quad \text{OFF} \quad \text{GND} \]

\[ V_{DD} \quad \text{Y} \quad 1 \]

\[ A \quad 1 \quad \text{OFF} \quad \text{ON} \quad \text{GND} \]
CMOS Logic

- Inverter

![CMOS Inverter Diagram](image)
CMOS Logic

- NAND gate
CMOS Logic

- NAND gate
CMOS Logic

- NAND gate
CMOS Logic

- NAND gate
CMOS Logic

- NAND gate
CMOS Logic

- NAND gate
CMOS Logic

- NAND gate
CMOS Logic

- NAND gate
CMOS Logic

- NAND gate
CMOS Logic

- NAND gate
CMOS Logic

- NAND gate
CMOS Logic

- NAND gate
CMOS Logic

- Complementary Metal-Oxide Semicondutor
  - pMOS pull-up network
  - nMOS pull-down network
CMOS Logic

- Multiplexers
CMOS Logic

- Multiplexers
CMOS Logic

- Multiplexers
CMOS Logic

- Multiplexers

![Multiplexer Diagram](image-url)
CMOS Logic

- Multiplexers
CMOS Logic

- Multiplexers
CMOS Logic

- Latches and flip-flops

D latch

D flip-flop
Moore’s Law

- The number of transistors in a dense integrated circuit doubles about every two years
Moore’s Law

- The number of transistors in a dense integrated circuit doubles about every two years
Moore’s Law

- The number of transistors in a dense integrated circuit doubles about every two years
Moore’s Law

- The number of transistors in a dense integrated circuit doubles about every two years

Moore’s Law
Moore’s Law

Transistors per Square Millimeter by Year, 1971–2018. Logarithmic scale. Data from Wikipedia.
Technology

- What do “7nm” or “10nm” technology mean?
Layout Design Rules

- Define how small features can be and how closely they can be reliably packed
- Lambda based rules
Dennard Scaling

- As transistors shrink, they become faster, consume less power, and are cheaper to manufacture.
  - Miniaturization provides smaller and faster transistors
Dennard Scaling

- As transistors shrink, they become **faster**, consume **less power**, and are **cheaper to manufacture**.
  - Miniaturization provides smaller and faster transistors

\[ \text{Area} = S^2 \]
Dennard Scaling

- As transistors shrink, they become faster, consume less power, and are cheaper to manufacture.
  - Miniaturization provides smaller and faster transistors.

\[
\text{Area} = S^2
\]

\[
\text{Area} = 0.5S^2
\]
Dennard Scaling

- As transistors shrink, they become **faster**, consume **less power**, and are **cheaper to manufacture**.
  - Miniaturization provides smaller and faster transistors

\[ \text{Area} = S^2 \]
\[ \text{Area} = 0.5 S^2 \]

\[ \text{Area} = 0.5 S^2 \]

- Transistor dimensions scaled by 30%
Dennard Scaling

- As transistors shrink, they become **faster**, consume **less power**, and are **cheaper to manufacture**.
  - Miniaturization provides smaller and faster transistors

\[
\text{Area} = S^2
\]

\[
\text{Area} = 0.5S^2
\]

- Transistor dimensions scaled by 30%
- Delay reduced by 30%
Dennard Scaling

- As transistors shrink, they become faster, consume less power, and are cheaper to manufacture.
  - Miniaturization provides smaller and faster transistors

\[ \text{Area} = S^2 \]

\[ \text{Area} = 0.5 \times S^2 \]

- Transistor dimensions scaled by 30%
- Delay reduced by 30%
- Frequency increased by ~40%
Performance Scaling

- Smaller transistors are faster
  - ~1.4x higher performance per generation
- More transistors allow more functionality
  - Better branch predictors
  - Larger caches
  - ILP: superscalar, out-of-order execution
  - DLP: vector unit
  - Better memory controller
CPU Performance

CPU Frequency

![CPU Frequency Chart]

- AMD
- Cypress
- DEC
- Fujitsu
- Hitachi
- HP
- IBM
- Intel
- Motorola
- MIPS
- SGI
- Sun
- Cyrix
- HAL
- NexGen

Clock Frequency (MHz) vs. Year
CPU Frequency

Power wall!
CMOS Power

- Dynamic Power
  - Switching power

- Static Power (Leakage)
  - Power dissipated even when not switching
  - Around one third of the total power in current technology

\[
P_{\text{dyn}} = \alpha CV^2 f
\]

\[
P_{\text{sta}} = I_{\text{leak}} V
\]

\[
P_{\text{total}} = P_{\text{dyn}} + P_{\text{sta}}
\]
CMOS Power

- **Static Power**
  - Gate leakage ($I_{\text{gate}}$)
  - Subthreshold leakage ($I_{\text{sub}}$)
  - Junction leakage ($I_{\text{junct}}$)
  - Depends on the temperature
CMOS Power

Power Wall
Power Wall and ILP Limits

- Cannot increase performance by raising the frequency due to thermal constraints
- Diminishing returns in ILP
  - Computer architects optimized superscalar out-of-order CPUs for decades
- But Moore’s law still provides more and more transistors
- What can we do with the extra transistors?
Multi-core Processors

https://www.guru3d.com/articles-summary/amd-ryzen-5-2400g-review,2.html
Multi-core Processors

- Use extra transistors to include multiple cores in the same chip
- Exploit TLP (Thread-Level Parallelism)
- Programmer has to manually extract parallelism
CPU Trends

40 Years of Microprocessor Trend Data

Transistors (thousands)
Single-Thread Performance (SpecINT x 10^3)
Frequency (MHz)
Typical Power (Watts)
Number of Logical Cores

Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp
End of Dennard Scaling

- According to Dennard scaling, power density remains constant
  - Smaller transistors consume less power
  - If frequency is not increased, total power remains the same
- Dennard scaling did not take subthreshold leakage into account
- In current technology, more transistors result in more power
  - Power density increases even at the same frequency!

\[ P = \alpha CV^2 f + I_{\text{leak}} V \]
Power Density

Dark Silicon

- Part of the chip must be powered off due to thermal constraints
  - 22 nm: 21% of the chip powered off
  - 8 nm: ~50% of the chip powered off
Multicore Limitations

- Increasing the number of cores is no longer an effective solution to improve performance
  - Due to the dark silicon problem, not all the cores can be powered at the same time
  - Diminishing returns in TLP
- But Moore’s law still provides more and more transistors
- What can we do with the extra transistors?
Hardware Accelerators

- Welcome to the golden age of hardware accelerators!

![Diagram of AMD "ZEN" x86 CPU Cores and Infinity Fabric](https://www.hotchips.org/hc30/1conf/1.05_AMD_APU_AMD_Raven_HotChips30_Final.pdf)
Hardware Accelerators

Qualcomm Snapdragon 835 [1]

- Snapdragon X16 LTE modem
- Adreno 540 Graphics Processing Unit (GPU)
- Display Processing Unit (DPU)
- Video Processing Unit (VPU)
- Wi-Fi
- Hexagon DSP
- Qualcomm Spectra 180 Camera
- Qualcomm® Aqstic Audio
- Kryo 280 CPU
- Qualcomm® IZat™ Location
- Qualcomm Haven Security

NVIDIA “Parker” SoC [2]

- Pascal
- Geforce GPU
- (256 CUDA Cores)
- ARM v8 CPU COMPLEX
  (2x Denver 2 + 4x A57)
  Coherent HMP

1. https://www.notebookcheck.net/Qualcomm-Snapdragon-835-SoC-Benchmarks-and-Specs.207842.0.html
ASIC

• Application Specific Integrated Circuit

https://ngcodec.com/markets-cloud-transcoding/
Software Overheads

```c
for (i = 0; i < 4; i++)
    v[i] *= 2.0f;
```
Software Overheads

for (i = 0; i < 4; i++)
    v[i] *= 2.0f;

la    $t0, v
la    $t1, v+16
li    $t2, 0x40000000
mtc1  $t2, $f2
for:  lwc1  $f0, 0($t0)
    mul.s $f0, $f0, $f2
    swc1  $f0, 0($t0)
    addiu $t0, $t0, 4
    bne   $t0, $t1, for
Software Overheads

```c
for (i = 0; i < 4; i++)
    v[i] *= 2.0f;
```

```assembly
la    $t0, v
la    $t1, v+16
li    $t2, 0x40000000
mtc1  $t2, $f2
for: lwc1  $f0, 0($t0)
    mul.s $f0, $f0, $f2
    swc1  $f0, 0($t0)
    addiu $t0, $t0, 4
    bne   $t0, $t1, for
```

https://en.wikibooks.org/wiki/Microprocessor_Design/Pipelined_Processors
Software Overheads

```c
for (i = 0; i < 4; i++)
    v[i] *= 2.0f;
```

```
la    $t0, v
la    $t1, v+16
li    $t2, 0x40000000
mtc1  $t2, $f2
```

<table>
<thead>
<tr>
<th>Addressing</th>
<th>Instruction</th>
<th>Read</th>
<th>R/W</th>
<th>Read</th>
<th>Write</th>
<th>ADD/SUB</th>
<th>MUL</th>
<th>Bytes</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>I$</td>
<td>D$</td>
<td>RF</td>
<td>RF</td>
<td>MM</td>
<td></td>
<td>R MM</td>
<td>W MM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>8</td>
<td>32</td>
<td>12</td>
<td>16</td>
<td>4</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

https://en.wikibooks.org/wiki/Microprocessor_Design/Pipelined_Processors
Software Overheads

for (i = 0; i < 4; i++)
    v[i] *= 2.0f;

Accesses to TLB
TLB misses
ROB accesses
Issue queue...

<table>
<thead>
<tr>
<th></th>
<th>Read I$</th>
<th>R/W D$</th>
<th>Read RF</th>
<th>Write RF</th>
<th>ADD/SUB MUL</th>
<th>Bytes R MM</th>
<th>Bytes W MM</th>
</tr>
</thead>
<tbody>
<tr>
<td>for: lwc1 $f0, 0($t0)</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>mul.s $f0, $f0, $f2</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>swc1 $f0, 0($t0)</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>addiu $t0, $t0, 4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>bne $t0, $t1, for</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>20</td>
<td>8</td>
<td>32</td>
<td>12</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
</tbody>
</table>

https://en.wikibooks.org/wiki/Microprocessor_Design/Pipelined_Processors
ASIC vs CPU

<table>
<thead>
<tr>
<th>MUL</th>
<th>Bytes R MM</th>
<th>Bytes W MM</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>
ASIC vs CPU

Main Memory

![Diagram showing Main Memory with read and write operations](image-url)

<table>
<thead>
<tr>
<th>MUL</th>
<th>Bytes R MM</th>
<th>Bytes W MM</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation</th>
<th>Read I$</th>
<th>R/W D$</th>
<th>Read RF</th>
<th>Write RF</th>
<th>ADDI/SUB</th>
<th>MUL</th>
<th>Bytes R MM</th>
<th>Bytes W MM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>Write</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32</td>
<td>12</td>
</tr>
<tr>
<td>ADDI/SUB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>MUL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>16</td>
</tr>
</tbody>
</table>

89 / 98
ASIC vs CPU

ASIC does much less work, so it is faster and consumes less energy
Hardware Accelerators

Figure 1. Die area breakdown of Apple’s systems on chips (SoCs). (a) A6 (iPhone 5), (b) A7 (iPhone 5s), and (c) A8 (iPhone 6). More than half of the die area is dedicated to specialized IP blocks.

Hardware Accelerators

Sea of Accelerators

Hardware Accelerators

- ASICs provide higher performance and energy-efficiency
- But they are less programmable and have large development costs
- Only accelerate in hardware applications that are widely popular and compute intensive
- Example: machine learning accelerators
Google TPU


- 92 TOPS
- 28 MB on-chip
- 15x – 30x speedup
- 30x – 80x higher Perf/W
Turing Lecture at ISCA 2018

A New Golden Age for Computer Architecture:
Domain-Specific Hardware/Software Co-Design, Enhanced Security, Open Instruction Sets, and Agile Chip Development

John Hennessy and David Patterson
Stanford and UC Berkeley
13 June 2018

https://www.youtube.com/watch?v=3LVeEjsn8Ts

https://www.acm.org/hennessy-patterson-turing-lecture
What about the future?

- Hardware accelerators will start providing diminishing returns in the future
- Any solution?
  - Quantum computing
  - DNA computing
  - Photonic computing
  - Superconducting computing
Bibliography

