Chapter 4
Low-Power VLSI Design

Jin-Fu Li
Advanced Reliable Systems (ARES) Lab.
Department of Electrical Engineering
National Central University
Jhongli, Taiwan
Outline

• Introduction
• Low-Power Gate-Level Design
• Low-Power Architecture-Level Design
• Algorithmic-Level Power Reduction
• RTL Techniques for Optimizing Power
Introduction

• Most SOC design teams now regard power as one of their top design concerns

• Why low-power design?
  – Battery lifetime (especially for portable devices)
  – Reliability

• Power consumption
  – Peak power
  – Average power
Overview of Power Consumption

• Average power consumption
  – Dynamic power consumption
  – Short-circuit power consumption
  – Leakage power consumption
  – Static power consumption

• Dynamic power dissipation during switching
Overview of Power Consumption

• Generic representation of a CMOS logic gate for switching power calculation

\[ P_{avg} = \frac{1}{T} \left[ \int_0^{T/2} V_{out} \left( -C_{load} \frac{dV_{out}}{dt} \right) dt + \int_{T/2}^T (V_{DD} - V_{out}) (C_{load} \frac{dV_{out}}{dt}) dt \right] \]
Overview of Power Consumption

• The average power consumption can be expressed as

\[ P_{avg} = \frac{1}{T} C_{load} V_{DD}^2 = C_{load} V_{DD}^2 f_{CLK} \]

• The node transition rate can be slower than the clock rate. To better represent this behavior, a node transition factor \( \alpha_T \) should be introduced

\[ P_{avg} = \alpha_T C_{load} V_{DD}^2 f_{CLK} \]

• The switching power expressed above are derived by taking into account the output node load capacitance
Overview of Power Consumption

The generalized expression for the average power dissipation can be rewritten as

\[ P_{\text{avg}} = \left( \sum_{i=1}^{\text{#ofnodes}} \alpha_{Ti} C_i V_i \right) V_{DD} f_{CLK} \]
Gate-Level Design – Technology Mapping

• The objective of logic minimization is to reduce the boolean function.

• For low-power design, the signal switching activity is minimized by restructuring a logic circuit.

• The power minimization is constrained by the delay, however, the area may increase.

• During this phase of logic minimization, the function to be minimized is

\[ \sum_{i} P_i (1 - P_i) C_i \]
Gate-Level Design – Technology Mapping

• The first step in technology mapping is to decompose each logic function into two-input gates.

• The objective of this decomposition is to minimizing the total power dissipation by reducing the total switching activity.

\[
\begin{align*}
\alpha &= 0.0384 \\
\alpha &= 0.0196 \\
\alpha &= 0.0099 \\
\alpha &= 0.0099 \\
\alpha &= 0.1875
\end{align*}
\]
Gate-Level Design – Phase Assignment

High activity node

A
B
C

A
B
C

High activity node
Gate-Level Design – Pin Swapping

Switching activity

Switching activity
Gate-Level Design – *Glitching Power*

- **Glitches**
  - spurious transitions due to imbalanced path delays

- A design has more balanced delay paths
  - has fewer glitches, and thus has less power dissipation

- Note that there will be no glitches in a dynamic CMOS logic

![Diagram](image-url)
Gate-Level Design – *Glitching Power*

- A chain structure has more glitches
- A tree structure has fewer glitches

![Diagram of a chain structure](image)

![Diagram of a tree structure](image)
Gate-Level Design – Precomputation

Precomputation Logic

Combinational Logic
Gate-Level Design – Precomputation

Diagram:

- A\langle n-1 \rangle
- B\langle n-1 \rangle
- A\langle n-2:0 \rangle
- B\langle n-2:0 \rangle

Precomputation logic

Enable

REG R1

1-bit Comparator (MSB)

REG R2

(n-1)-bit Comparator

REG R3

REG R4

F

EE4012 VLSI Design
National Central University
Gate-Level Design – Gating Clock

Fail DFT rule checking

Add control pin to solve DFT violation problem
Gate-Level Design – Input Gating
Clock-Gating in Low-Power Flip-Flop

Source: Prof. V. D. Agrawal
Reduced-Power Shift Register

Flip-flops are operated at full voltage and half the clock frequency.

Source: Prof. V. D. Agrawal
Power Consumption of Shift Register

16-bit shift register, 2μ CMOS

<table>
<thead>
<tr>
<th>Deg. Of parallelism</th>
<th>Freq (MHz)</th>
<th>Power (μW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33.0</td>
<td>1535</td>
</tr>
<tr>
<td>2</td>
<td>16.5</td>
<td>887</td>
</tr>
<tr>
<td>4</td>
<td>8.25</td>
<td>738</td>
</tr>
</tbody>
</table>

\[ P = C'V_{DD}^2f/n \]


Source: Prof. V. D. Agrawal
Assume that With the same 16x16 multiplier, the power supply can be reduced from $V_{ref}$ to $V_{ref}/1.83$.

$$P_{parallel} = 2.2C_{ref} \left( \frac{V_{ref}}{1.83} \right)^2 \frac{f_{ref}}{2} = 0.33P_{ref}$$
The hardware between the pipeline stages is reduced then the reference voltage $V_{\text{ref}}$ can be reduced to $V_{\text{new}}$ to maintain the same worst case delay. For example, let a 50MHz multiplier is broken into two equal parts as shown below. The delay between the pipeline stages can be remained at 50MHz when the voltage $V_{\text{new}}$ is equal to $V_{\text{ref}}/1.83$

$$P_{\text{pipeline}} = 1.2C_{\text{ref}} \left( \frac{V_{\text{ref}}}{1.83} \right)^2 f_{\text{ref}} = 0.36P_{\text{ref}}$$
Retiming is a transformation technique used to change the locations of delay elements in a circuit without affecting the input/output characteristics of the circuit.

Two versions of an IIR filter.
Architecture-Level Design – Retiming

Retiming for pipeline design

```
REG  C1 (6ns)  C2 (2ns)  REG  C3 (4ns)
```

```
REG  C1 (6ns)  REG  C2 (2ns)  C3 (4ns)
```

$f_{\text{ref}}$
Architecture-Level Design – *Retiming*

Clock cycle is 4 gate delays

Clock cycle is 2 gate delays
Architecture-Level Design – Power Management

Architecture -- Level Design –– Power Management

C2
C1
C1_FREEZE
C2_FREEZE

C2
C1
C1_FREEZE
C2_FREEZE
Architecture-Level Design – Bus Segmentation

• Avoid the sharing of resources
  – Reduce the switched capacitance

• For example: a global system bus
  – A single shared bus is connected to all modules, this structure results in a large bus capacitance due to
    * The large number of drivers and receivers sharing the same bus
    * The parasitic capacitance of the long bus line

• A segmented bus structure
  – Switched capacitance during each bus access is significantly reduced
  – Overall routing area may be increased
Architecture-Level Design – *Bus Segmentation*

![Bus Segmentation Diagram](image-url)
Minimization the switching activity, at high level, is one way to reduce the power dissipation of digital processors.

One method to minimize the switching signals, at the algorithmic level, is to use an appropriate coding for the signals rather than straight binary code.

The table shown below shows a comparison of 3-bit representation of the binary and Gray codes.

<table>
<thead>
<tr>
<th>Binary Code</th>
<th>Gray Code</th>
<th>Decimal Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>011</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>010</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>110</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>111</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>101</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>100</td>
<td>7</td>
</tr>
</tbody>
</table>
State Encoding for a Counter

- Two-bit binary counter:
  * State sequence, 00 → 01 → 10 → 11 → 00
  * Six bit transitions in four clock cycles
  * $6/4 = 1.5$ transitions per clock

- Two-bit Gray-code counter
  * State sequence, 00 → 01 → 11 → 10 → 00
  * Four bit transitions in four clock cycles
  * $4/4 = 1.0$ transition per clock

- Gray-code counter is more power efficient.


Source: Prof. V. D. Agrawal
Binary Counter: Original Encoding

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>a  b</td>
<td>A  B</td>
</tr>
<tr>
<td>0  0</td>
<td>0  1</td>
</tr>
<tr>
<td>0  1</td>
<td>1  0</td>
</tr>
<tr>
<td>1  0</td>
<td>1  1</td>
</tr>
<tr>
<td>1  1</td>
<td>0  0</td>
</tr>
</tbody>
</table>

\[ A = a'b + ab' \]
\[ B = a'b' + ab' \]

Source: Prof. V. D. Agrawal
Binary Counter: Gray Encoding

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

A = a’b + ab
B = a’b’ + a’b

Source: Prof. V. D. Agrawal
### Three-Bit Counters

<table>
<thead>
<tr>
<th>Binary</th>
<th>No. of toggles</th>
<th>Gray-code</th>
<th>State</th>
<th>No. of toggles</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>-</td>
<td>000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>001</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>2</td>
<td>011</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>010</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>110</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>111</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>2</td>
<td>101</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>100</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>3</td>
<td>000</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Average Transitions/clock = 1.75

Average Transitions/clock = 1

Source: Prof. V. D. Agrawal
N-Bit Counter: Toggles in Counting Cycle

- Binary counter: $T(\text{binary}) = 2(2^N - 1)$
- Gray-code counter: $T(\text{gray}) = 2^N$
- $T(\text{gray})/T(\text{binary}) = 2^{N-1}/(2^N - 1) \rightarrow 0.5$

<table>
<thead>
<tr>
<th>Bits</th>
<th>$T(\text{binary})$</th>
<th>$T(\text{gray})$</th>
<th>$T(\text{gray})/T(\text{binary})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4</td>
<td>0.6667</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>8</td>
<td>0.5714</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>16</td>
<td>0.5333</td>
</tr>
<tr>
<td>5</td>
<td>62</td>
<td>32</td>
<td>0.5161</td>
</tr>
<tr>
<td>6</td>
<td>126</td>
<td>64</td>
<td>0.5079</td>
</tr>
<tr>
<td>$\infty$</td>
<td>-</td>
<td>-</td>
<td>0.5000</td>
</tr>
</tbody>
</table>

Source: Prof. V. D. Agrawal
FSM State Encoding

Transition probability based on PI statistics

Expected number of state-bit transitions:

\[ 2(0.3+0.4) + 1(0.1+0.1) = 1.6 \]

\[ 1(0.3+0.4+0.1) + 2(0.1) = 1.0 \]

State encoding can be selected using a power-based cost function.

Source: Prof. V. D. Agrawal
FSM: Clock-Gating

- Moore machine: Outputs depend only on the state variables.
  - If a state has a self-loop in the state transition graph (STG), then clock can be stopped whenever a self-loop is to be executed.

Source: Prof. V. D. Agrawal
Clock-Gating in Moore FSM


Source: Prof. V. D. Agrawal
Bus Encoding for Reduced Power

- Example: Four bit bus
  - $0000 \rightarrow 1110$ has three transitions.
  - If bits of second pattern are inverted, then $0000 \rightarrow 0001$ will have only one transition.

- Bit-inversion encoding for $N$-bit bus:

![Graph showing number of bit transitions after inversion encoding vs. total number of bit transitions.](source: Prof. V. D. Agrawal)
Bus-Inversion Encoding Logic

Sent data → Polarity decision logic → Polarity bit → Bus register → Received data


Source: Prof. V. D. Agrawal
RTL-Level Design – Signal Gating

Simple Decoder

module decoder (a, sel);
    input  [1:0] a;
    output [3:0] sel;
    reg     [3:0] sel;
    always @(a) begin
        case (a)
            2’b00: sel=4’b0001;
            2’b01: sel=4’b0010;
            2’b10: sel=4’b0100;
            2’b11: sel=4’b1000;
        endcase
    end
endmodule

Decoder with enable

module decoder (en,a, sel);
    input en;
    input  [1:0] a;
    output [3:0] sel;
    reg     [3:0] sel;
    always @({en,a}) begin
        case ({en,a})
            3’b100: sel=4’b0001;
            3’b101: sel=4’b0010;
            3’b110: sel=4’b0100;
            3’b111: sel=4’b1000;
            default: sel=4’b0000;
        endcase
    end
endmodule
RTL-Level Design – *Datapath Reordering*

**Initial**

- A<B
- Mux
- Glitchy
- Stable

**Reordered**

- A<B
- Mux
- Glitchy
- Stable
RTL-Level Design – Memory Partition

pre_addr 8 \( \longrightarrow \) q addr[7:0] \\
clk \( \longrightarrow \) 

din addr write \\
dout noe

addr[7:1] \( \downarrow \) 

noe write \\
addr din \\
dout 128x32

addr0 

128x32 

dout 32 

MUX 32 dout
RTL-Level Design – Memory Partition

• Application-driven memory partition
RTL-Level Design – Memory Partition

- A power-optimal partitioned memory organization