Low-Power Design

Rene van Leuken
Introduction (1)
Introduction (2)
The Importance of Low Power Electronics (1)

- Effect on the environment
  - Reduces CO$_2$ emission and energy consumption

- Enables new electronic equipment
  - Size reduction
  - Increased battery lifetime
The Importance of Low Power Electronics (2)

- Cooling and System packaging considerations
  - Reduction in system cost
- Power/frequency considerations
  - Enables higher operation frequency
The Importance of Low Power Electronics (3)

• Reliability
  • Reduction in electronic system (component) malfunction
Contact-less Smart Card (1998)

- Wipe distance < 10 cm
- Asynchronous design
- Process 0.35 CMOS, 2 volt (5 metal layers)
- Modules on card: memory, uart, processor, de/crypt, coil
- 8051 processor (async) 40% power reduction
Power dissipation 80C51

Synchronous

Asynchronous
CMOS Power

- The power consumption in digital CMOS circuits
  \[ P_{\text{avg}} = P_{\text{dynamic}} + P_{\text{short-circuit}} + P_{\text{leakage}} \]

- Dynamic Power Consumption
  Charging and Discharging Capacitors

- Short Circuit Currents
  Short Circuit Path between Supply Rails during Switching

- Leakage (Static)
  Leaking diodes and transistors
Switching Energy

\[ \text{Energy/transition} = \alpha \cdot C_L \cdot V_{dd}^2 \]

\[ \text{Power} = \text{Energy/transition} \cdot f = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f \]

\[ \text{Total Power} = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f + \tau \cdot \alpha \cdot V_{dd} \cdot I_{\text{short}} + V_{dd} \cdot I_0 \]

- This basic rule works for all levels of design abstraction
- The key parameters when trying to minimize dynamic power consumption
  - Capacitive load, supply voltage and operating frequency
  - Number of devices
Motivation

• Application domain:
  • Moderately complex programmable systems with run-time or operating system,
  • E.g., laptops, smart phones,......

• Objective:
  • Reduce energy consumed by VLSI chips and peripherals, while preserving performance.
Why Lower Power

• Portable systems:
  • Long battery life.
  • Light weight.
  • Small form factor.

• IC priority list:
  • Power dissipation.
  • Cost.
  • Performance.

• Technology direction:
  • Reduced voltage/power designs based on mature high performance IC technology, high integration to minimize size, cost, power, and speed.
Solutions

- Spectrum of approaches:
  - One end: dissipate power efficiently.
    - Targets designs which care most about performance.
    - Heat Sinks.
    - Fans (Air Cooling).
    - Liquid Cooling.
    - Cryogenic Cooling.
  - Other end: consume less power.
    - Targets designs which are primarily about minimizing power consumption.
Levels for Low Power Design

- System.
- Algorithm.
- Architecture.
- Circuit/logic.
- Technology.

- Design partitioning, Power Down.
- Complexity, Concurrency, Locality, Regularity, Data representation.
- Voltage scaling, Parallelism, Instruction set, Signal correlations.
- Transistor Sizing, Logic optimization, Activity Driven Power Down, Low-swing logic, Adiabatic switching.
- Threshold Reduction, Multi thresholds.
Levels for Low Power Design

Level of Abstraction

- System level
- Behavior level
- RT level
- Logic level
- Transistor level
- Layout level

Expected Saving

- Increasing power savings
- 10-20 x
- 2-5 x
- 20-50%
Reducing Power Consumption

• Algorithm level:
  • The choice of an algorithm is the most highly leveraged decision in meeting the power constraints. The power consumption is strongly correlated to a number of properties that a given algorithm may have.
  • Some algorithmic properties which are critical for selection an algorithm for low power design:
Algorithm level:

- Size measures includes quantities such as the number of nodes, the bit width, the number of I/O operations, the number of operations, and the number of memory accesses.
- **Concurrency** measures the number of operations and interconnect accesses that can be executed concurrently.
- Spatial locality characterizes the degree to which the algorithm has natural clusters of computation, within which significant amounts of computation can be done independently.
- Regularity captures the degree to which common patterns appear.
Reducing Power Consumption

• Architecture level:
  • The architectural level is the design entry point for the large majority of digital designs and design decisions at this level can have dramatic impact on the power budget design.
  • Perhaps the most important strategy for reducing power consumption involves employing concurrent processing at the architecture level. This is a direct trade-off of area and performance for power.

• Power Reduction Techniques.
  • Low system clocks.
    • High frequencies are generated with on-chip PLLs.
  • High-level of integration (single chip).
    • Avoid off-chip components.
  • Power management: shutdown unused blocks
  • Memory partitioning
    • Selectively enabled blocks
  • Parallelism, pipelining
  • Reduction of global busses
  • Simplification of instruction coding and execution
  • Component minimization
    • Arithmetic
    • Memories/registers
  • Scheduling and allocation
  • Netlist transformations
  • Data/number coding
Parallelism and Power

Consider a CMOS circuit in which most of the power dissipation is active power. In this case, the power dissipation for the circuit is roughly:

\[ P \propto CV^2f \]

where \( C \) is the switched capacitance of the circuit, \( V \) is the supply voltage, and \( f \) is the effective switching frequency of the nodes in the circuit. We can take advantage of the fact that the maximum operating frequency of the circuit is roughly proportional to voltage. If the function implemented in the circuit can be scaled through parallel logic implementation by some factor \( s(>1) \), so that the circuit has more transistors (for more parallelism) by a factor of \( s \), but can achieve the same throughput at frequency reduced by \( s \) we can reduce the operating voltage by roughly \( s \) as well, so we see a significant reduction in power:

\[ P_{\text{new}} \propto (sC)\left(\frac{V}{s}\right)^2\left(\frac{f}{s}\right) = \frac{P_{\text{original}}}{s^2} \]

In practice, \( s \) cannot be increased arbitrarily because of the operation of CMOS circuits degrades as operating voltage approaches transistor-threshold voltage.
Reducing Power Consumption

• Buses.
  • Buses are a significant source of power loss, especially interchip buses, which are often very wide.
  • A chip can expend 15 percent to 20% of its power on interchip drivers.
  • One approach to limiting this swing is to encode the address lines into a Gray code because address changes, particularly from cache refills, are often sequential, and counting in Gray code switch the least number of signals.
  • Transmitting the difference between successive address values achieves a result similar to the Gray code.
  • Compressing the information in address lines further reduces them.
Reducing Power Consumption

• Circuit/Logic level:
  • Clock gating. Widely used to turn off clock tree branches to latches or flip-flops whenever they are not used. Until recently, developers considered gated clocks to be a poor design practice because the clock tree gates can exacerbate clock skew. More accurate timing analyzers and more flexible design tools have made it possible to produce reliable designs with gated clocks.
Reducing Power Consumption

• Circuit/Logic level:
• Asynchronous logic. Because the systems do not have a clock, they save the considerable power that a clock tree requires. Drawback:
  • Need to generate completion signals. This means that additional logic must be used at each register transfer, which can increase the amount of logic and wiring.
  • Testing difficulty and an absence of design tools.
• Optimizing switching activity (example from Rabaey: DIC).
Design Solutions
Clock Gating

• Most effective power optimization technique
• Supported by most of the EDA tools
• Effective at register level as well as at clock network level
• Different approaches:
  • Functional approach
  • Activity-driven
  • Observability Don’t Care-Driven
Clock Gating Principle

- **Goal**
  Disable or suppress transitions from propagating to parts of the clock path (FFs, clock network and logic) under a given IDLE condition.

- **Principle**
  To each sequential functional unit is associated a block CG which inhibits the clock signal when the IDLE condition is true.
  The IDLE condition is computed by function $F_{cg}$.
Clock Gating Implementation

Flip-Flop-Based Design

Simplest way to implement block CG but subject to spikes.

When CLK is low, spikes are filtered by the AND
When CLK is high, spikes are filtered by the Latch

When CLK is high, spikes are filtered by the NOR
When CLK is low, spikes are filtered by the Latch
How effective is Clock-gating?

90% of F/F’s were clock-gated.

70% power reduction by clock-gating alone.

M. Ohashi, Matsushita, ISSCC 2002
Gating the Clock Network
Data Path: Guarded Evaluation

• Applicable to combin. Blocks emb. within logic
• If Y is idle, transparent latches are inserted to all inputs
• Control circuitry is added to determine the IDLE condition
• The IDLE condition is used to disable the latches.
Operand Isolation Example

Design without Operand Isolation
Automatic Operand Isolation

- Stops data feeding into DesignWare arithmetic components, unless output is required.
- Automatically inserts isolation logic.
- Automatically inserts activation logic.

Diagram:
- DATA_1
- DATA_2
- Add_0
- SEL_0
- SEL_1
- mux_0
- mux_1
- reg_0
- AS
Bus Coding

• Advanced SoC characterized by:
  • Long buses with high capacitance and a significant switching activity.

• Techniques proposed:
  • Low swing bus
  • Charge recycling bus
  • Bus pipelining
  • Bus multiplexing
  • Bus encoding
Bus Coding: RTL approach

• Bus coding is more suitable for:
  • Long wires (To/From) memories
  • Memory buses

• Issues:
  • Limited budget for Encoder/Decoder
    • Chose simple implementations
Bus Coding

b(t): Source word

B(t): Code word

Less switching activity
Bus Coding

• Different approaches:
  • **Bus-Invert Coding** and its variants (four)
  • Transition Signaling Code
  • Offset Code
  • **T0 Code** and its variants (four)
  • Limited-Weight Code (ie. One-hot code)
  • **Codebook based code**
  • Etc…
Bus Invert Coding

- The encoding depends on Hamming distance between the present bus value $B(t)$ and the next bus value $B(t+1)$

$$(B(t), \text{INV}(t)) = \begin{cases} 
(b(t), 0) & \text{if } H \leq N/2 \\
(b'(t), 1) & \text{Otherwise}
\end{cases}$$

$N$: number of bus lines, $H$: Hamming Distance
Bus Invert Coding

<table>
<thead>
<tr>
<th>Binary (31 Trs)</th>
<th>BIC (19 Trs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00101010</td>
<td>00101010</td>
</tr>
<tr>
<td>00111011</td>
<td>00111011 0</td>
</tr>
<tr>
<td>11010100</td>
<td>00101011 1</td>
</tr>
<tr>
<td>11110100</td>
<td>00001011 1</td>
</tr>
<tr>
<td>00001101</td>
<td>00001101 0</td>
</tr>
<tr>
<td>01110110</td>
<td>10001001 1</td>
</tr>
<tr>
<td>00010001</td>
<td>00010001 0</td>
</tr>
<tr>
<td>10000100</td>
<td>10000100 0</td>
</tr>
</tbody>
</table>

05/13/14
Bus Invert Coding

- Characteristics:
  - Redundant bit consumes power
  - Switching activity on highly capacitive buses is reduced at the expense of additional switching activity in the decoder/encoder
  - Effective when the data to be transmitted is randomly distributed in time (µP ↔ cache)
  - Not efficient for address bus encoding
T0 Code

• Exploit the sequentially of the address buses
• Redundant line INC is added to the bus
• When two addresses to be transmitted are sequential, the address bus is frozen and INC is set to 1
• Zero-Transition for ideally consecutive addresses
T0 Code: Principle

Encoder

$$(B(t), \text{INC}(t))$$

\[
\begin{align*}
(B(t-1), 1) & \quad \text{If } b(t) = b(t-1) + S \\
(b(t), 0) & \quad \text{Otherwise}
\end{align*}
\]

Decoder

$$b(t)$$

\[
\begin{align*}
(B(t-1) + S) & \quad \text{If INC} = 1 \\
B(t) & \quad \text{If INC} = 0
\end{align*}
\]

S may be known by the encoder and the decoder or send on the bus
# T0 Code: example

<table>
<thead>
<tr>
<th>Transition</th>
<th>T0 Code</th>
<th>Binary Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>00000100</td>
<td>00000100 0</td>
</tr>
<tr>
<td>5</td>
<td>00000101</td>
<td>00000100 1 1</td>
</tr>
<tr>
<td>6</td>
<td>00000110</td>
<td>00000100 1 0</td>
</tr>
<tr>
<td>7</td>
<td>00000111</td>
<td>00000100 1 0</td>
</tr>
<tr>
<td>8</td>
<td>00001000</td>
<td>00000100 1 0</td>
</tr>
<tr>
<td>6</td>
<td>00000110</td>
<td>00000110 0 2</td>
</tr>
<tr>
<td>7</td>
<td>00000111</td>
<td>00000110 1 1</td>
</tr>
<tr>
<td>8</td>
<td>00001000</td>
<td>00000110 1 0</td>
</tr>
</tbody>
</table>

**Binary encoding:**

16 Transitions

**T0 encoding:**

4 Transitions
T0 Code: Implementation

Encoder

Decoder
T0 Code: Implementation

Diagram showing the implementation of an encoder and decoder.
T0 Code

- Suitable for address bus encoding when sequential addresses transmitted on the bus dominate.
- The encoder inserts one clock cycle delay
- Extra area and delay
- Power saving achieved if the probability of sequential addresses appearing in the bus is higher than a technology dependent threshold
Dynamic Voltage Scaling (DVS)

Application
Operating System
Control Signals

Controller

Load prediction
Speed setting

Software
Hardware

Normalized workload

Normalized power

Variable V$_{dd}$
Fixed V$_{dd}$

S. Lee et al, DAC, June 2000
Dynamic Voltage and Frequency Scaling (DVFS)

- Tune Both Fclk and $V_{DD}$: DVFS
- Workload-Based Technique: Software-Hardware cooperation

S. Lee et al, DAC, June 2000
Dynamic Voltage and Frequency Scaling (DVFS)

\[ P_{AC} = C_{eff} \cdot V_{DD}^2 \cdot f \]

- **Power** \( \propto V^2 \cdot f \)
- **Hurry-up-and-wait**
  - Busy cycles
  - Idle cycles
- **Frequency scaling (f/2)**
  - Power \( \propto V^2 \cdot \left(\frac{f}{2}\right)\)
- **Voltage and frequency scaling (f/2, V_{DD}/2)**
  - Power \( \propto \left(\frac{V}{2}\right)^2 \cdot \left(\frac{f}{2}\right)\)
Conclusion

• Best way to save power: do nothing!
• Most important equation: to remember:
  • Power = C * V^2 * f
• Slowing clock rate does not reduce energy for fixed operation!
• Ways of reducing power:
  • Pipelining with reduced voltage.
  • Parallelism with reduced voltage.
  • Bus loads.
  • Co-design.