Low Power Design

Dr. Paul D. Franzon

Outline
1. Power consumption in CMOS
2. Strategies to reduce power consumption
3. Strategies to reduce energy/operation

References
- Smith and Franzon, Chapter 11
- Weste and Harris, Principles of CMOS VLSI Design, A Systems Perspective
Objectives

• Identify why low power design is important.
• Understand the difference between low power design and low energy design.
• Understand static and dynamic power sources in static CMOS circuits.
• Identify techniques to reduce power consumption and energy per operation


**Power Consumption**

Why is power consumption important?

- **Battery powered devices**
  - Maximize battery life
- **Limit requirements of cooling**
  - Temperature control of chip is important
    - Typically want chip transistor temperature to be ~85 – 105 C
      - Higher temperatures slow chips down, increases transistor leakage, increases transistor “wearout”
      - On/off thermal cycles fatigue solder and other connections reducing reliability
  - Plastic packaging is 10x cheaper than ceramic packaging but can only dissipate 1 - 2 W
  - Fans, heat pipes etc. are needed and get more expensive as heat flux increases
- An issue in most designs even in mobile devices!
Power Consumption

...Why is power consumption important?
• As power requirements go up, the cost of delivering the power to the chips increases
  • Copper bus-bars
  • Power regulators/supplies etc.
    ♦ Efficiency ~ 80% - 95% ⇒ lose 20% of power even before it gets to the chip
    ♦ Before that roughly ~30% of nations power consumption in the electrical grid is spent on overcoming distribution losses
• Green systems
  • About 1% of worlds power goes to electronics, and is increasing
  • About 20% of home power goes to electronics, and is increasing
  • Greenhouse gas issues
Power Consumption

Power is increasingly the MAJOR constraint on system performance

- The major constraint used to be #transistors/die. Now it is power.
- Battery driven Wireless systems
  - System Performance is limited by how much data can be communicated and processed on a single battery charge
    - Want this capacity to permit a battery life of more than a day
- Desktop systems
  - Without attention, power consumption of circa 2018 CPUs would be 1 kw or more ➞ Impossible to air cool
  - Without attention, power consumptions of chips like HDTV decoders would be 100+W ➞ Expensive to cool
- Server farms and Supercomputers
  - Without attention, power consumption of next generation server farms would be > 20 MW ➞ requires multiple power substations, not one
**Dark Silicon**

**Performance per unit power**

- Systems increasingly limited by power consumption, not number of transistors
- \( \Rightarrow \) "Dark Silicon": Most of the chip will be OFF to meet thermal limits
**Energy per Operation**

Can always scale power by slowing down device
- E.g. Reducing processing capability (GOPS = Giga Operations Per Second) by reducing clock rate, also reduces power proportionally
- Useful when device in “idle” mode or power not otherwise needed

But many systems have performance requirements as well
- Have to complete so many operations to complete task
  - Slowing down processor only slows down task completion
- Might even have real time constraints

Measuring Energy per Operation rather than power gives truer picture of efficiency of a design
- E.g. 1 GOPS @ 1 GHz @ 1 W vs. 0.1 GOPS @ 100 MHz @ 0.1 W
- Both have same energy / operation ($10^{-9}$ J/operation (1 nJ/op))
Power Consumption

Static CMOS Circuits:

- Static Power (whether circuit switching or not)
  - Leakage
    - Sub-threshold drain to source
    - Some libraries include low-leakage cells, or cells that can be switching to a low leakage state
- Dynamic Power (when logic transitions occur)
  - ‘through’ current small during switching
  - Toggling power when output node changes logic state
**CMOS Circuit**

Circuit during switching event

- E.g. Inverter driving a load:
  - Power dissipated in resistors in 010 cycle = potential energy stored and released on capacitor during that cycle
    \[ Q = CV_{dd} \]
    \[ E = QV_{dd} = CV_{dd}^2 \]
    \[ P = \frac{E}{T} = CV_{dd}^2/T \]
    \[ = N_{\text{switch}} CV_{dd}^2 f_{\text{clock}} \]

- Alternative derivation:
  - When \( V_{out} \rightarrow 1 \), energy dissipated in top resistor:
    \[ E = \int_0^{V_{dd}} (V_{dd} - V_{out})Idt = \int_0^{V_{dd}} (V_{dd} - V_{out})CdV_{out} = \frac{CV_{dd}^2}{2} \]
    \[ \Rightarrow I = C \frac{dV_{out}}{dt} \]
Minimizing Power Consumption

Power consumption in a CMOS module:

\[
\text{Power} = \sum N_{\text{switch}} f V_{\text{dd}}^2 C_{\text{load}} + \text{leakage power}
\]

- Dynamic Power
- Static Power

- Sum over all nodes in circuit
- \( f \) = clock frequency
- \( N_{\text{switch}} \) = average % of clock periods in which node toggles (I.e. 010 or 101)
- \( C_{\text{load}} \) = capacitance of node

\( N_{\text{switch}} \)

- Clock: \( 100\% \)
- Maximum for glitch-free logic: \( 50\% \)
- Logic typically has \( N_{\text{switch}} \sim 0.1 \)
Minimizing Energy

Energy = ∫Power.dt

Energy consumption in a CMOS module:

\[
\text{Energy} = \sum_{\text{cycles}} \sum_{\text{nodes}} N_{\text{switch}} Vdd^2 C_{\text{load}} + \text{leakage power} \times \text{time}
\]

- \(N_{\text{switch}}\) = average % of clock periods in which node toggles (I.e. 010 or 101)
- \(C_{\text{load}}\) = capacitance of node
Speed – Energy Tradeoff

Running at lowest possible voltage not always the best strategy

Optimum energy\_delay (around 600 – 800mV)
Scenarios

1. Low power in idle mode
   - Part of chip is idle, e.g. CPU waiting for a keystroke
     - Reduce Vdd, reduce fclock to units required to stay on but not required to operate at peak performance
       - Referred to as *Dynamic Voltage Frequency Scaling*
     - Synthesis support: Libraries characterized at multiple Vdd’s
     - Note: Can’t change to and from these states instantly
.. Scenarios

2. Zero power in off state

• Unit goes through periods when it is not needed
  • Example: Camera JPEG decoder in smart phone
• Requirement:
  • Reduce dynamic power to zero (including for clock)
  • In advanced nodes, reduce leakage power to zero
• Solutions:
  ◆ Turn off unneeded units by removing their clock
    ➔ Note might result in loss of state in those units
    ➔ Referred to as clock gating
   /logo Designed by clock tree designer, not inserted in RTL
    ➔ Sometimes combined with sleep transistors that reduce leakage current
    by blocking the power supply
  ◆ Note: Can’t change to and from these states instantly
**Sleep transistors**

Leakage power is consumed whether clock is on or off

- Significant percentage of total power
- One solution:
  - Put low leakage (high threshold voltage) transistors around circuit to be turned off
  - Turn these transistors on while operating
  - Turn transistors on while operating
  - Referred to as “sleep transistors”

- Another solution:
  - Use gates with high threshold transistors off critical path
  - (Using high Vt transistors makes circuit slower)
Scenarios

3. Minimize energy/operation without a performance constraint
   • Examples. Many biomedical processors running from a long-life battery, electronic wrist-watches, hearing aids, long-life low-activity sensors
   • Goal: Minimize total power – accept performance at that power
     ➔ Operate at lowest Vdd possible
   • Requires cell library characterized at that low Vdd
   • At low Vdd’s the difference between tmin and tmax for logic can become very large
     • Complicates timing convergence
   • Extreme: Vdd < Vt “Sub-threshold” voltage
     • Further complicates timing convergence
   • For practical reasons many low power chips designed at 0.6 – 1 V
… Scenarios

4. Minimize energy/operation or energy/task within a performance target constraint

E.g. Most future processors
“operation” = micro-operation
“task” = complete function (e.g. 1024 point FFT)
• Might lead to choosing Vdd at optimum energy.delay point
• Want to minimize “wasted” power
  • i.e. Power NOT contributing to end calculation
• Want to minimize
  \[ \text{Energy} = \sum_{\text{cycles}} \sum_{\text{nodes}} N_{\text{switch}} Vdd^2 C_{\text{load}} + \text{leakage power} \]
• Minimize:
  \[ \sum_{\text{micro-operations}} \text{required to complete task, weighted by energy/micro-op} \]
  ➔ Examples of micro-operations:
  Add, Multiple, Memory Store, Register store, Move data, etc.
• Minimize “useless” toggling
Summary – Sub-module 1

Reasons to reduce power consumption in a design

- Battery life
- Cost of cooling
- Limited AC power supply
- Reducing “dark silicon”

Low-power scenarios

- Chip has situation with idle or near idle resources
  - Clock gating, Dynamic Voltage/Frequency Scaling
- Longest battery life without a performance constraint → Lowest Vdd, possibly sub-threshold
- Power constraint with a performance constraint
  → Minimize energy/operation at reasonable clock frequency

\[
\text{Power} = \sum N_{\text{switch}} \cdot f \cdot V_{\text{dd}}^2 \cdot C_{\text{load}} + \text{leakage power}
\]
# Energy / Micro-operation Examples

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy/µOp</th>
<th>Energy/µOp @ 7 nm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Computation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit MAC (32 nm)</td>
<td>19.4 pJ/op</td>
<td>7 pJ/op</td>
</tr>
<tr>
<td>11 nm 0.4 V core</td>
<td>200 pJ/op</td>
<td>142 pJ/op</td>
</tr>
<tr>
<td><strong>Storage (32 bits)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDR 3 DRAM</td>
<td>4.8 nJ/word</td>
<td>Scales with new stds</td>
</tr>
<tr>
<td>Optimized DRAM bank</td>
<td>96 pJ / word</td>
<td>Does not scale</td>
</tr>
<tr>
<td>90 nm 2-way cache</td>
<td>107 pJ/word</td>
<td>28 pJ/op</td>
</tr>
<tr>
<td>90 nm Pipeline Register</td>
<td>12 pJ</td>
<td>4 pJ/op</td>
</tr>
<tr>
<td>90 nm Register File</td>
<td>17 pJ</td>
<td>5 pJ/op</td>
</tr>
<tr>
<td>32 nm 64 kB SRAM cache</td>
<td>163 pJ/word</td>
<td>60 pJ/op</td>
</tr>
<tr>
<td>32 nm 512 KB STT-RAM cache (R/W)</td>
<td>174 / 316 pJ</td>
<td>63 pJ/op (W does not scale)</td>
</tr>
<tr>
<td><strong>Data Motion (32 bits)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 mV High speed IO</td>
<td>37 pJ/word</td>
<td>Scales slowly</td>
</tr>
<tr>
<td>On-chip (2.5 mm)</td>
<td>8.7 pJ/word</td>
<td>Scales with V^2</td>
</tr>
<tr>
<td>3DIC – Through Silicon Via (TSV)</td>
<td>3.5 pJ/word</td>
<td>Scales with V^2</td>
</tr>
<tr>
<td>3DIC – Face to face (F2F)</td>
<td>&lt; 1 pJ/word</td>
<td>Scales with V^2</td>
</tr>
</tbody>
</table>

MAC = Multiply - Accumulate
Typical Power Distribution

- Clock typically consumes 20% - 40% of total power
  - Why clock gating is valuable when turning modules off
- On-chip interconnect consumes 20% - 40% of total power
  - Employ architectures that reduce average wire length
- Leakage power is a big % of memory power

Note: Leakage power gets worse with technology scaling

Exemplar power distribution
3DIC

Stacking and interconnect of chips vertically using microbumps and Through Silicon Vias

Potentially 1 μm pitch

Face to Face

Back to Back

TSVs (typically 25 μm pitch)
Estimating Power Through CAD

Can estimate power through the CAD tools
  • After place and route, know C_node for each node
  • Know Vdd, f
  • What is N_switch for each node?

Method 1:
  • Assume a fixed N_switch for each node for rough estimate of power consumption

Method 2:
  • Use logic simulation to calculate N_switch for each node (interpolate for nodes not in RTL if using RTL simulation)
Reducing Power through Smart Design

1. Use algorithms that require fewer high energy micro-operations
   - Examples:
     • Signal processing algorithms that require fewer multiplications
     • Integer multiply instead of floating point if appropriate
     • Algorithms that compress data into SRAMs to reduce number of off-chip DRAM operations

2. Avoid high-energy micro-operations in detailed design
   - E.g. Uses muxes instead of arithmetic units, shift units etc.
   - Reduce number of registers
   - Reduce number of memory accesses

3. Avoid “wasted toggling”
   - E.g. Designs where high energy units operate with data changes that are not needed
   - Remember: Combinational Logic Evaluates whenever inputs change

4. Use synthesis tools that focuses on minimizing power, not area
Reducing “Useless” toggling at logic level

Example:
reg [31:0] A, B, D;
always@(posedge clock)
  begin
    if (C) D <= A+B;
    else D <= A;
  end

Possible ways to reduce power:
**Toggling Reduction**

If C is low a lot...

```verilog
assign E=C?A:0;
assign F=C?B:0;
always@(posedge clock)
  if (C) D <= E+F;
  else D <= A;
```

*Only useful if C is low more than 50% of the time.*
Other Alternatives

- Store previous value of A and B in a register
  - Used instead of 0 input to mux
  - Must consider power overhead of register (including extra Cload on clock)
  - Not likely to be beneficial here
  - Might be beneficial for a larger design (e.g. multiplier)

Energy in MULT wasted if B does not select new mult output and inputs change

Overhead of FF acceptable here if B does not change a lot

FFs freezes logic inputs to MULT to prevent useless toggling
Summary – Low Power Sub-module 2

Energy / micro-operation varies considerably with the micro-op
  • DRAM > SRAM > Arithmetic > Register
  • Off-chip > On-chip > 3DIC

Minimizing Energy / Operation or Task via good design
  • Algorithms that minimize higher energy micro-operations
  • Logic and control structures that minimize the additional micro-ops beyond those required to complete the required steps in a task
    • E.g. Adding pipeline registers increases energy / op
  • Minimize “wasted” power or toggling that does not contribute to the steps required to complete a task
    • E.g. Changing the inputs of a large combinational logic unit causes toggling and power consumption even if that calculation is not being used – work out a way to freeze such inputs when not needed