Agenda

- Recap
- Power reduction on
  - Gate level
  - Architecture level
  - Algorithm level
  - System level
Recap: Problems of Power Dissipation

- Continuously increasing performance demands
- Increasing power dissipation of technical devices
- Today: power dissipation is a main problem

High Power dissipation leads to:

- Reduced time of operation
- Higher weight (batteries)
- Reduced mobility

- High efforts for cooling
- Increasing operational costs
- Reduced reliability
Recap: Consumption in CMOS

- Voltage (Volt, V)
- Current (Ampere, A)
- Energy

Water pressure (bar)
Water quantity per second (liter/s)
Amount of Water

Energy consumption is proportional to capacitive load!
Recap: Energy and Power

- **Power** is height of curve
- **Energy** is area under curve

**Energy** = Power \* time for calculation = Power \* Delay
Recap: Power Equations in CMOS

\[ P = \alpha f C L V_{DD}^2 + V_{DD} I_{\text{peak}} (P_{0\rightarrow 1} + P_{1\rightarrow 0}) + V_{DD} I_{\text{leak}} \]

**Dynamic power**
(≈ 40 - 70% today and decreasing relatively)

**Short-circuit power**
(≈ 10% today and decreasing absolutely)

**Leakage power**
(≈ 20 – 50% today and increasing)
Recap: Levels of Optimization

<table>
<thead>
<tr>
<th>Levels of Optimization</th>
<th>Savings</th>
<th>Speed</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>&gt; 70 %</td>
<td>Seconds</td>
<td>&gt; 50 %</td>
</tr>
<tr>
<td>Algorithm</td>
<td>40-70 %</td>
<td>Minute</td>
<td>25-50 %</td>
</tr>
<tr>
<td>Architecture</td>
<td>25-40 %</td>
<td>Minutes</td>
<td>15-30 %</td>
</tr>
<tr>
<td>Gate</td>
<td>15-25 %</td>
<td>Hour</td>
<td>10-20 %</td>
</tr>
<tr>
<td>Transistor</td>
<td>10-15 %</td>
<td>Hours</td>
<td>5-10 %</td>
</tr>
</tbody>
</table>

Copyright Sill, 2008

Micro transductors ’08, Low Power 2
Recap: Logic Restructuring

- Logic restructuring: changing the topology of a logic network to reduce transitions

\[ P_{0 \rightarrow 1} = P_0 \times P_1 = (1 - P_A P_B) \times P_A P_B \]

- Chain implementation has a lower overall switching activity than tree implementation for random inputs

- **BUT:** Ignores glitching effects

Source: Timmernann, 2007
Recap: Input Ordering

Beneficial: postponing introduction of signals with a high transition rate (signals with signal probability close to 0.5)

\[(1-0.5 \times 0.2) \times (0.5 \times 0.2) = 0.09\]
\[(1-0.2 \times 0.1) \times (0.2 \times 0.1) = 0.0196\]

AND: \[P_{0 \rightarrow 1} = (1 - P_A P_B) \times P_A P_B\]

Source: Irwin, 2000
Recap: Glitching

![Diagram of a logic circuit]

<table>
<thead>
<tr>
<th>ABC</th>
<th>101</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unit Delay

Source: Irwin, 2000
Design Layer: Gate Level

- Basic elements:
  - Logic gates
  - Sequential elements (flipflops, latches)
- Behavior of elements is described in libraries
Device Sizing (= changing gate width)
- Affects input capacitance $C_{\text{in}}$
- Affects load capacitance $C_{\text{load}}$
- Affects dynamic power consumption $P_{\text{dyn}}$

Optimal fanout factor $f$ for $P_{\text{dyn}}$ is smaller than for performance (especially for large loads)
- e.g., for $C_{\text{load}}=20$, $C_{\text{in}}=1$
  - $f_{\text{circuit}} = 20$
  - $f_{\text{opt_energy}} = 3.53$
  - $f_{\text{opt_performance}} = 4.47$

For Low Power: avoid oversizing ($f$ too big) beyond the optimal

Source: Nikolic, UCB
Delay ($t_d$) and dynamic power consumption ($P_{dyn}$) are functions of $V_{DD}$. 

V_{DD} versus Delay and Power
Multiple \( V_{\text{DD}} \)

- Main ideas:
  - Use of different supply voltages within the same design
  - High VDD for critical parts (high performance needed)
  - Low VDD for non-critical parts (only low performance demands)

- At design phase:
  - Determine critical path(s) (see upper next slide)
  - High \( V_{\text{DD}} \) for gates on those paths
  - Lower \( V_{\text{DD}} \) on the other gates (in non-critical paths)
  - For low VDD: prefer gates that drive large capacitances (yields the largest energy benefits)

- Usually two different \( V_{\text{DD}} \) (but more are possible)
Level converters:

- Necessary, when module at lower supply drives gate at higher supply (step-up)
- If gate supplied with $V_{DDL}$ drives a gate supplied with $V_{DDH}$
  - then PMOS never turns off
- Possible implementation:
  - Cross-coupled PMOS transistors
  - NMOS transistor operate on reduced supply
- No need of level converters for step-down change in voltage
- Reducing of overhead:
  - Conversions at register boundaries
  - Embedding of inside flipflop
Data Paths

- Data propagate through different data paths between registers (flipflops - FF)
- Paths mostly differ in propagation delay times
- Frequency of clock signal (CLK) depends on path with longest delay
  ➔ critical path
Data Paths: Slack

A
B
Y
C

time

G1 ready with evaluation
all Inputs of G1 arrived
Slack for G1
delay of G1
Multiple $V_{DD}$ in Data Paths

- Minimum energy consumption when all logic paths are critical (same delay)
- Possible Algorithm: clustered voltage-scaling
  - Each path starts with $V_{DDH}$ and switches to $V_{DDL}$ (blue gates) when slack is available
  - Level conversion in flipflops at end of paths
Design Layer: Architecture Level

- Also known as Register transfer level (RTL)
- Base elements:
  - Register structures
  - Arithmetic logic units (ALU)
  - Memory elements
- Only behavior is described
  (no inner structure)
Clock Gating

- Most popular method for power reduction of clock signals and functional units
- Gate off clock to idle functional units
- Logic for generation of disable signal necessary
  - Higher complexity of control logic
  - Higher power consumption
  - Critical timing critical for avoiding of clock glitches at OR gate output
  - Additional gate delay on clock signal

Source: Irwin, 2000
Clock-Gating in Low-Power Flip-Flop

Source: Agarwal, 2007
Clock Gating cont’d

- Clock gating over consideration of state in Finite-State-Machines (FSM)

Clock Gating: Example

- 90% of FlipFlops clock-gated
- 70% power reduction by clock-gating

Source: M. Ohashi, Matsushita, 2002
Recap: $V_{DD}$ versus Delay and Power

Dynamic Power can be traded by delay
A Reference Datapath

Input → Register → Combinational logic → Register → Output

Supply voltage: $V_{\text{ref}}$
Total capacitance switched per cycle: $C_{\text{ref}}$
Clock frequency: $f_{\text{Clk}}$
Power consumption: $P_{\text{ref}} = C_{\text{ref}} V_{\text{ref}}^2 f_{\text{clk}}$

Source: Agarwal, 2007
Each copy processes every Nth input, operates at reduced voltage.

Supply voltage: \( V_N \leq V_{\text{ref}} \)

\( N = \text{Deg. of parallelism} \)

Source: Agarwal, 2007
Pipelined Architecture

- Reduces the propagation time of a block by factor N
  - Voltage can be reduced at constant clock frequency
- Constant throughput

Functionality:

CLK

Area A

CLK

A/N

A/N

A/N

CLK

Data

CLK
Parallel Architecture: Example

- Reference Data path (for example)

- Critical path delay $T_{adder} + T_{comparator}$ (= 25 ns)
  - $f_{ref} = 40$ MHz
- Total capacitance being switched = $C_{ref}$
- $V_{DD} = V_{ref} = 5V$
- Power for reference datapath = $P_{ref} = C_{ref} \times V_{ref}^2 \times f_{ref}$

Area = $636 \times 833 \: \mu^2$

Source: Irwin, 2000
Parallel Architecture: Example cont’d

- The clock rate can be reduced by half with the same throughput
  \( f_{par} = f_{ref} / 2 \)
- \( V_{par} = V_{ref} / 1.7, \ C_{par} = 2.15 \ C_{ref} \)
- \( P_{par} = (2.15 \ C_{ref}) (V_{ref} / 1.7)^2 (f_{ref} / 2) = 0.36 \ P_{ref} \)

Source: Irwin, 2000
Pipelined Architecture: Example

- \( f_{\text{pipe}} = f_{\text{ref}}, \quad C_{\text{pipe}} = 1.1 \, C_{\text{ref}}, \quad V_{\text{pipe}} = V_{\text{ref}} / 1.7 \)
- Voltage can be dropped while maintaining the original throughput
- \( P_{\text{pipe}} = C_{\text{pipe}} \, V_{\text{pipe}}^2 \quad f_{\text{pipe}} = (1.1 \, C_{\text{ref}}) \, (V_{\text{ref}}/1.7)^2 \quad f_{\text{ref}} = 0.37 \, P_{\text{ref}} \)

Source: Irwin, 2000
## Approximate Trend

<table>
<thead>
<tr>
<th></th>
<th>N-parallel proc.</th>
<th>N-stage pipeline proc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacitance</td>
<td>N*C_{ref}</td>
<td>C_{ref}</td>
</tr>
<tr>
<td>Voltage</td>
<td>V_{ref}/N</td>
<td>V_{ref}/N</td>
</tr>
<tr>
<td>Frequency</td>
<td>f_{ref}/N</td>
<td>f_{ref}</td>
</tr>
<tr>
<td>Dynamic Power</td>
<td>C_{ref}V_{ref}^{2}f_{ref}/N^2</td>
<td>C_{ref}V_{ref}^{2}f_{ref}/N^2</td>
</tr>
<tr>
<td>Chip area</td>
<td>N times</td>
<td>10-20% increase</td>
</tr>
</tbody>
</table>

Guarded Evaluation

- Reduction of switching activity by adding latches at inputs

- Latch preserves previous value of inputs to suppress activity

- Could also use AND gates to mask inputs to zero
  \( = \) forced zero
Identify logical conditions at inputs that are invariant to the output

- Since those inputs don’t affect output, disable input transitions
- Trade area for energy
Precomputation: Design Issues

- Design steps
  1. Selection of precomputation architecture
  2. Determination of precomputed and gated inputs (Register R1 should be much smaller than R2)
  3. Search good implementation for \( g(X) \)
  4. Evaluation of potential energy savings based on input statistics (if savings not sufficient go to step 2 or 3 and try again)

- Also works for multiple output functions where \( g(X) \) is the product of \( g_j(X) \) over all \( j \)

Source: Irwin, 2000
Precomputation: Example

- Binary Comparator

A > B
n-bit binary value comparator
A > B

Can achieve up to 75% power reduction with 3% area overhead and 1 to 5 additional gate delays in worst case path

Source: Irwin, 2000
Various algorithms exist to implement an integer adder

- Ripple, select, skip (x2), Look-ahead, conditional-sum.
- Each with its own characteristics of timing and power consumption.

Source: Mendelson, Intel
### Adder Design

<table>
<thead>
<tr>
<th>Adder Design</th>
<th>Energy (pJ)</th>
<th>Delay (nSec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ripple Carry</td>
<td>117</td>
<td>54.27</td>
</tr>
<tr>
<td>Constant Width Carry Skip</td>
<td>109</td>
<td>28.38</td>
</tr>
<tr>
<td>Variable Width Carry Skip</td>
<td>126</td>
<td>21.84</td>
</tr>
<tr>
<td>Carry Lookahead</td>
<td>171</td>
<td>17.13</td>
</tr>
<tr>
<td>Carry Select</td>
<td>216</td>
<td>19.56</td>
</tr>
<tr>
<td>Conditional Sum</td>
<td>304</td>
<td>20.05</td>
</tr>
</tbody>
</table>

- Adders differ in Energy and delay
  - Different adders for different applications
  - Also true for other units (multiplier, counter, …)

Buses are significant source of power dissipation

- 50% of dynamic power for interconnect switching (Magen, SLIP 04)
- MIT Raw processor’s on-chip network consumes 36% of total chip power (Wang et al. 2003)

Caused by:

- High switching activities
- Large capacitive loading

Source: Irwin, 2000
Bus Power Reduction

- For an n-bit bus: \( P_{bus} = n^* \alpha f_{Clk} C_{load} V_{DD}^2 \)

- Alternative bus structures
  - Segmented buses (lower \( C_{load} \))
  - Charge recovery buses
  - Bus multiplexing (lower \( f_{Clk} \) possible)

- Minimizing bus traffic (\( n \))
  - Code compression
  - Instruction loop buffers

- Minimization of bit switching activity (\( f_{clk} \)) by data encoding

- Minimize voltage swing (\( V_{DD}^2 \)) using differential signaling

Source: Irwin, 2000
Reducing Shared Resources

- Shared resources incur switching overhead
- Local bus structures reduce overhead

Source: Irwin, 2000
Reducing Shared Resources cont’d

- Bus segmentation
  - Another way to reduce shared buses
  - Control of bus segment by controller blocks (B)

Source: Evgeny Bolotin – Jan 2004
Design Layer: Algorithm Level

- Base elements:
  - Functions
  - Procedures
  - Processes
  - Control structures

- Description of design behavior
Coding styles

- Use processor-specific instruction style:
  - Variable types
  - Function calls style
  - Conditionalized instructions (for ARM)

- Follow general guidelines for software coding
  - Use table look-up instead of conditionals
  - Make local copies of global variables so that they can be assigned to registers
  - Avoid multiple memory look-ups with pointer chains
Source-code Transformations

Minimize power-consuming activity:

- **Computation**
  
  - $A * B + A * C \rightarrow A * (B + C)$

- **Communication**
  
  - For $c = 1..N$
    - Receive $(A)$
    - $B = c \times A$

- **Storage**
  
  - For $c = 1..N$
    - $B[c] = A[c] \times D[c]$
    - $F[c] = B[c] - 1$
    - $F[c] = A[c] \times D[c] - 1$
Algorithms can differ in power dissipation

Source: Irwin, 2000
Adaptive Dynamic Voltage Scaling (DVS)

- Slow down processor to fill idle time
- More Delay ➔ lower operational voltage

<table>
<thead>
<tr>
<th>Active</th>
<th>Idle</th>
<th>Active</th>
<th>Idle</th>
<th>3.3 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td></td>
<td></td>
<td></td>
<td>2.4 V</td>
</tr>
</tbody>
</table>

- Runtime Scheduler determines processor speed and selects appropriate voltage
- Transitions delay for frequencies ~150µs
- Potential to realize 10x energy savings
Adaptive DVS: Example

- Task with 100 ms deadline, requires 50 ms CPU time at full speed
  - Normal system gives 50 ms computation, 50 ms idle/stopped time
  - Half speed/voltage system gives 100 ms computation, 0 ms idle
  - Same number of CPU cycles but: $E = C \left(\frac{V_{DD}}{2}\right)^2 = E_{ref} / 4$
  - Dynamic Voltage Scaling adapts voltage to workload

![Diagram showing adaptive DVS example](image)
Basic Elements:

- Complex modules
- Processors
- Calculation and control units
- Sensors
Dynamic Power Management

- Systems are:
  - Designed to deliver peak performance, but …
  - Not needing peak performance most of the time
- Components are idle sometimes
- Dynamic power management (DPM):
  - Puts idle components in low-power non-operational states when idle
- Power manager:
  - Observes and controls the system
  - Power consumption of power manager is negligible
Processor Sleep Modes

- Software power control - power management
  - **DOZE**: Most units stopped except on-chip cache memory (cache coherency)
  - **NAP**: Cache also turned off, PLL still on, time out or external interrupt to resume
  - **SLEEP**: PLL off, external interrupt to resume

Deeper sleep mode consumes less power

Deeper sleep mode requires more latency to resume
## Processor Sleep Modes: Example

- **PowerPC sleep modes**

<table>
<thead>
<tr>
<th>Mode</th>
<th>66Mhz</th>
<th>80Mhz</th>
</tr>
</thead>
<tbody>
<tr>
<td>No power mgmt</td>
<td>2.18W</td>
<td>2.54W</td>
</tr>
<tr>
<td>Dynamic power mgmt</td>
<td>1.89W</td>
<td>2.20W</td>
</tr>
<tr>
<td>DOZE</td>
<td>307mW</td>
<td>366mW</td>
</tr>
<tr>
<td>NAP</td>
<td>113mW</td>
<td>135mW</td>
</tr>
<tr>
<td>SLEEP</td>
<td>89mW</td>
<td>105mW</td>
</tr>
<tr>
<td>SLEEP without PLL</td>
<td>18mW</td>
<td>19mW</td>
</tr>
<tr>
<td>SLEEP without clock</td>
<td>2mW</td>
<td>2mW</td>
</tr>
</tbody>
</table>

- 10 cycles to wake up from SLEEP
- 100us to wake up from SLEEP+

Source: Irwin, 2000
Transmeta LongRun

- Applies adaptive DVS
- LongRun policies:
  - Detection of different workload scenarios
  - Based on runtime performance information
- After detection ➔ accordingly adaptation of:
  - Processor supply voltage
  - Processor frequency
  - Clock frequency always within limits required by supply voltage to avoid clock skew problems
- Use of core frequency/voltage hard coded operating points

➔ Best trade-off between performance and power possible
Transmeta LongRun cont’d

Source: Transmeta
Transmeta LongRun: Example

Source: Transmeta
Battery aware design

- Non-linear effects influence life time of batteries
- “Rate Capacity”
  - If discharging currents higher than allowed
  - Real capacity goes under nominal capacity
- “Battery Recovery”
  - Pulsed discharge increases nominal capacity
  - Based on recovery times
  - (as long there is no rate capacity effect)

Source: Timmermann, 2007
Battery aware design cont’d

Diffusion Model from - Rakhmatov, Vrudula et al.

- Analytically very sound but computationally intensive
- Cannot be used for online scheduling decisions.
Battery aware design: Example 1

- Performance of a bipolar lead-acid battery subjected to six current impulses. Pulse length=3 ms, rest period=22 ms.

Battery aware design: Example 2

<table>
<thead>
<tr>
<th>Profile</th>
<th>Aver. Current [mA]</th>
<th>Battery lifetime [ms]</th>
<th>Specif. energy [Wh/Kg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>123.8</td>
<td>357053</td>
<td>15.12</td>
</tr>
<tr>
<td>B</td>
<td>124.2</td>
<td>536484</td>
<td>18.58</td>
</tr>
</tbody>
</table>

⇒ Minimum average current ≠ Maximum battery life time

Source: Timmermann, 2007
Backup
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.

Clock can be stopped when (Xk, Sk) combination occurs.
Trend: Interconnects

Interconnects

Propagation delays of global wires will be a multiple of the clock cycle.

Example (very optimistic):
6–10 clock cycles in 50nm technology
[Benini, 2002]
Bus Multiplexing

- Number of bus transitions per cycle
  \[ = 2 \left( 1 + \frac{1}{2} + \frac{1}{4} + \ldots \right) = 4 \]

Source: Irwin, 2000
Resource Sharing and Activity II
Bus Multiplexing

- Sharing of long data buses with time multiplexing
- Example:
  - $S_1$ uses even cycles
  - $S_2$ odd

Source: Irwin, 2000
Correlated Data Streams

For a shared (multiplexed) bus advantages of data correlation are lost (bus carries samples from two uncorrelated data streams)

- Bus sharing should not be used for positively correlated data streams
- Bus sharing may prove advantageous in a negatively correlated data stream (where successive samples switch sign bits) - more random switching

Source: Irwin, 2000
Disadvantages of Bus Multiplexing

- If data bus is shared, advantages of data correlation are lost (bus carries samples from two uncorrelated data streams)
- Bus sharing should not be used for positively correlated data streams
- Bus sharing may prove advantageous in a negatively correlated data stream (where successive samples switch sign bits) - more random switching
Adaptive DVS cont’d

- Implementation

![Diagram showing Workflow Filter, FIFO Input Buffer, Variable Power-Speed System, and Power-Speed Control Knob]