System-Level Power/Energy Optimization

1. Sources of Power Dissipation
2. Reducing Power Consumption
3. System Level Power Optimization
4. Dynamic Power Management
5. Mapping and Scheduling for Low Energy
6. Real-Time Scheduling with Dynamic Voltage Scaling

Why is Power Consumption an Issue?

- Portable systems - battery life time!
- Systems with a very limited power budget: Mars Pathfinder, autonomous helicopter, ...
- Desktops and servers: high power consumption
  - raises temperature and deteriorates performance & reliability
  - increases the need for expensive cooling mechanisms
- One of the main difficulties with developing high performance chips is heat extraction.
- High power consumption has economical and ecological consequences.

Sources of Power Dissipation in CMOS Devices

\[ P = \sum \left( \frac{C}{2} \cdot V_{DD} \cdot f \cdot N_{SW} \cdot Q_{SC} \cdot f \cdot N_{SW} \cdot I_{leak} \cdot f \cdot V_{DD} \right) \]

- \( P \) = total power dissipation
- \( C \) = node capacitances
- \( N_{SW} \) = switching activities (number of gate transitions per clock cycle)
- \( f \) = frequency of operation
- \( V_{DD} \) = supply voltage
- \( Q_{SC} \) = charge carried by short circuit current per transition
- \( I_{leak} \) = leakage current

Remember the Design Flow

Informal Specification, Constraints

- Modeling
- Functional Simulation
- Formal Verification

Arch. Selection

- System model
- System architecture
- Mapping

System-level estimation

not OK

Simulation

- Softw. model
- Hardw. model
- Softw. Generation
- Hardw. Synthesis

not OK

Mapped and scheduled model

Simulation

- Softw. blocks
- Hardw. blocks
- Simulation

not OK

Testing

OK

Prototype

OK

Fabrication

Testing

OK

Prototype

OK

Fabrication
Sources of Power Dissipation in CMOS Devices (cont’d)

**CMOS transistor (N-type)**

- **Threshold voltage:**
  - The minimal voltage required at the gate to turn on the transistor

**CMOS inverter**

- **Dynamic power:**
  - Charging and discharging the output load capacitance
  - Momentary short circuits at a gate’s output

**Static power:**

- Subthreshold leakage conduction
- Junction leakage (drain and source to body)

**Static power:**

- Leakage power has been considered negligible compared to dynamic.

**Dynamic power:**

- Total dissipation from leakage is approaching the total from dynamic.

**As technology drops below 65nm:**

- Leakage power is exceeding dynamic.
Sources of Power Dissipation in CMOS Devices (cont’d)

- Leakage power is consumed even if the circuit is idle (standby). The only way to avoid is decoupling from power.
- Short circuit power can be around 10% of total.
- Switching power is still the main source of power consumption.

For the rest of the discussion, we consider mainly switching power. At the end we come back to leakage.

Power and Energy Consumption

\[ P = \frac{1}{2} C V_{DD}^2 f N_{SW} \]

\[ E = P \cdot t = \frac{1}{2} C V_{DD}^2 N_{CY} N_{SW} \]

\( N_{CY} \) = number of cycles needed for the particular task.

- In certain situations we are concerned about power consumption:
  - heat dissipation, cooling:
  - physical deterioration due to temperature.
- Sometimes we want to reduce total energy consumed:
  - battery life.

Reducing Power/Energy Consumption

The main sources:
- Reduce supply voltage
- Reduce switching activity
- Reduce capacitance
- Reduce number of cycles

Reducing Power/Energy Consumption (cont’d)

- Circuit level:
  - Ordering of transistors in gate (influences capacitance).
  - Transistor sizing.
- Logic level:
  - Don’t-care optimization to reduce switching activity.
  - Reduce spurious switching activity by balancing the delays of paths that converge at each gate.
  - Technology mapping.
  - State encoding such that switching activity is minimised: if state \( s \) has a large number of transitions to state \( q \), they should be given uni-distant codes.
  - Encoding to minimise switching activity in arithmetic units or on the bus.
  - Gated clocks: Gate the clocks of circuits (registers, gates, arithmetic units when they are in idle time periods.)
Reducing Power/Energy Consumption (cont’d)

**Behavioral level**

- Schedule and map operations so that number of cycles is minimised (with increased number of switching per clock cycle) ⇒ you can run at slower clock rate ⇒ you can reduce supply voltage.
- Allocate and share modules so that power consumption is reduced (for example, by reducing switching activity)

**Architecture level**

- Specialise instruction set, datapath, register structure to the particular architecture, with power consumption as an optimization goal (see flow in Fö. 9).
  - You have on the chip and you switch only those resources (gates) you really need.
- Reduce power consumption on the bus.
  - lower switching activity: clever encoding, reduce switching activity on the address bus by exploiting correlations;
  - minimise the bus length (capacitance) by optimal module placement;
  - bus segmentation: transform a long heavily loaded global bus into a partitioned set of local bus segments.

Reducing Power/Energy Consumption (cont’d)

- Optimise the memory structure.
  - Memory transfers are extremely power hungry: a memory transfer takes 33 times more energy than an addition!
  - Reducing the number of memory accesses is a very efficient way to save power!
  - Adapt the number of caches, their size and associativity, and the length of the cache line to the application ⇒ reduce number of memory transfers.
  - Interesting trade-off: larger caches consume more power but reduce number of memory transfers ⇒ find the right balance!

Reducing Power/Energy Consumption (cont’d)

- Provide instruction support for Power management:
  - Instructions which allow to put in stand-by or shut down certain parts of the system.
  - Instructions which allow to dynamically fix the supply voltage (dynamic voltage scaling).
Reducing Power/Energy Consumption (cont’d)

☞ System Level

• Static techniques are applied at design time.
  - Compilation for low power: instruction selection considering their power profile, data placement in memory, register allocation.
  - Algorithm design: find the algorithm which is the most power-efficient.
  - Task mapping and scheduling.

• Dynamic techniques are applied at run time.
  - These techniques are applied at run-time in order to reduce power consumption by exploiting idle or low-workload periods.

System Level Power Optimization

Three techniques will be discussed:

1. Dynamic power management: a dynamic technique.
2. Task mapping: a static technique.
3. Task scheduling with dynamic power scaling: static & dynamic.

Dynamic Power Management (DPM)

Decisions:
• Switching among multiple power states:
  • idle
  • sleep
  • run

• Switching among multiple frequencies and voltage levels.

Goal:
• Energy optimization
• QoS constraints satisfied

Dynamic Power Management (cont’d)

Hardware Support (e.g., Intel Xscale Processor)

- RUN: operational
- IDLE: Clocks to the CPU are disabled; recovery is through interrupt.
- SLEEP: Mainly powered off; recovery through wake-up event.
- Other intermediate states: DEEP IDLE, STANDBY, DEEP SLEEP
Dynamic Power Management (cont’d)
☞ DPM techniques are used in laptops, personal digital assistants (PDAs), and other portable appliances in order to shut down or place in stand-by unused devices. The goal is power saving.
☞ DPM techniques are implemented in the operating system (including Windows 2000 running on laptops).
☞ The power breakdown for a laptop computer:
  - 36% of total power consumed by the display
  - 18% by hard-disk
  - 18% by wireless LAN interface
  - 7% by keyboard, mouse, etc.
  - 21% by digital VLSI circuits.

The Basic Concept of DPM
• When there are requests for a device ⇒ the device is busy; otherwise it is idle.
• When the device is idle, it can be shut down to enter a low-power sleeping state.

Power Management Policies
• Power management policies are concerned with predictions related to idle periods:
  - For shut-down: try to predict how long the idle period will be in order to decide if a shut-down should be performed.
  - For wake-up: try to predict when the idle period ends, in order to avoid user delays due to the wake-up.
  It is quite difficult, and often the wake-up is started simply when a request has arrived.
• Typical Policies:
  1. Time-out
  2. Predictive
  3. Stochastic

The Basic Concept of DPM (cont’d)
☞ Changing the power state takes time (several seconds) and extra energy.
  T_{sd} : shutdown delay
  T_{wu} : wake-up delay

Send the device to sleep only if the saved energy justifies the overhead!

The main Problems:
• Don’t shut down such that delays occur too frequently.
• Don’t shut down such that the savings due to the sleeping are smaller than the power overhead of the state changes.
**Time-out Policy**

- **It is assumed that**, after a device is idle for a period $\tau$ (the interval $T_1 - T_2$ on slide 22), it will stay idle for at least a period which makes it efficient to shut down.
  - **Drawbacks:** you waste energy during the period $\tau$ (compared to instantaneous shut-down).
  - **Policies:**
    - Fixed time-out period: you set the value of $\tau$, which then stays constant.
    - Adjusted at run-time: increase or decrease $\tau$, depending on the length of previous idle periods.

**Predictive Policy**

- The length of an idle period is predicted. If the prediction is for an idle period long enough, the shut-down is performed immediately (no time interval $T_1 - T_2$ on slide 22).

**Example Policy**

- L-shaped distribution for idle period followed by long idle periods.
- Busy periods longer than a threshold $\theta$ are followed by short idle periods.

**Stochastic Policy**

- Predictions are based on Markov models: requests and power state transitions of the device are modelled as probabilistic state machines.
  - The power manager observes the arriving requests, the request queue and the device $\Rightarrow$ generates shutdown commands.

**Mapping and Scheduling for Low Energy**

- For many embedded systems DPM techniques, like presented before, cannot be applied:
  - They have no devices like hard-disk, no (or small) display $\Rightarrow$ VLSI is a main source of power dissipation.
  - They have time constraints $\Rightarrow$ we have to keep deadlines (usually we cannot afford shut-down and wake-up times).
  - The operating system is small $\Rightarrow$ no sophisticated techniques at run-time.
  - The application is known at design time $\Rightarrow$ we know a lot about the application already at design time.

- Static techniques can be used (applied at design time). Mapping and scheduling for low energy are important!
Consider a mapping: 

\[ \text{Task} \quad \text{Worst} \quad \text{Energy} \]

<table>
<thead>
<tr>
<th>Task</th>
<th>( \mu p3 )</th>
<th>( \mu p4 )</th>
<th>( \mu p3 )</th>
<th>( \mu p4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>T2</td>
<td>7</td>
<td>9</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>T3</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>T4</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>T5</td>
<td>10</td>
<td>13</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>T6</td>
<td>17</td>
<td>21</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>T7</td>
<td>10</td>
<td>14</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>T8</td>
<td>15</td>
<td>19</td>
<td>14</td>
<td>9</td>
</tr>
</tbody>
</table>

Comm. times and energy: 

- \( \mu p3 \): \( \tau_1, \tau_3, \tau_6, \tau_7, \tau_8 \) 
  - C1-2: \( t = 1 \); \( E = 3 \) 
  - C3-5: \( t = 2 \); \( E = 5 \) 
- \( \mu p4 \): \( \tau_2, \tau_4, \tau_5, \tau_8 \) 
  - C4-8: \( t = 1 \); \( E = 3 \) 
  - C5-7: \( t = 1 \); \( E = 3 \) 

Execution time: 52; Energy consumed: 75.

The second mapping with \( \tau_8 \) on \( \mu p4 \) consumes less energy; 

Assume that we have a maximum allowed delay = 60. 

This second mapping is preferable, even if it is slower!
Real-Time Scheduling with Dynamic Voltage Scaling

☞ The energy consumed by a task, due to switching power (slide 6):
\[ E = \frac{1}{2} CV^2 \cdot N_{GW} \cdot N_{SW} \cdot N_{CY} \]
where:
- \( E \) is the energy consumed by a task,
- \( C \) is the capacitance of the gates in the circuit,
- \( V^2 \) is the supply voltage,
- \( N_{GW} \) is the number of gate transitions per clock cycle,
- \( N_{SW} \) is the number of switch transitions per clock cycle,
- \( N_{CY} \) is the number of cycles needed for the task.

☞ Reducing supply voltage \( V_{DD} \) is the most efficient way to reduce energy consumption.

☞ The frequency at which the processor can be operated depends on \( V_{DD} \):
\[ f = k \frac{(V_{DD} - V_t)^2}{V_{DD}} \]
where:
- \( f \) is the frequency,
- \( k \) is a circuit dependent constant,
- \( V_t \) is the threshold voltage.

The execution time of the task:
\[ t_{exec} = N_{CY} \cdot \frac{V_{DD}}{k(V_{DD} - V_t)^2} \]

Real-Time Scheduling with Dynamic Voltage Scaling (cont'd)

The scheduling problem (F09/10, slide 8):
Which task to execute at a certain moment on a certain processor so that time constraints are fulfilled?

The scheduling problem with voltage scaling:
Which task to execute at a certain moment on a certain processor, and at which voltage level, so that time constraints are fulfilled and energy consumption is minimised?

☞ The problem: reducing supply voltage extends execution time!

Variable Voltage Processors

- Several supply voltage levels are available.
- Supply voltage can be fixed by the application (operating system) through execution of particular instructions.
- Frequency is automatically adjusted to the current supply voltage.
- Several processors with variable voltage levels are already available. There will be more and more in the near future.

The Basic Principle

We consider a single task \( \tau \):
- total computation: \( 10^9 \) execution cycles.
- deadline: 25 seconds.
- processor nominal (maximum) voltage: 5V.
- energy: 40 nJ/cycle at nominal voltage.
- processor speed: 50MHz (50\times10^6 cycles/sec) at nominal voltage.

\[ V^2 \]
\[ \text{slack} \]
\[ 10^9 \text{cycles} \]
\[ E_{total} = 40 \text{J} \]
\[ t_{exec} = 20 \text{sec} \]
The Basic Principle (cont’d)

Let’s make it slower!

- $V_{DD} = 2.5V$
  - energy: $40 \times 2.5^2 / 5^2 = 10nJ/cycle$.
  - speed: $50 \times 2.5 / 5 = 25MHz$

\[ \begin{align*}
  E_{total} &= 32.5 J \\
  t_{max} &= 25 sec
\end{align*} \]

The Basic Principle (cont’d)

- $V_{DD} = 4V$
  - energy: $40 \times 4^2 / 5^2 = 25nJ/cycle$.
  - speed: $50 \times 4 / 5 = 40MHz$

\[ \begin{align*}
  E_{total} &= 25 J \\
  t_{max} &= 25 sec
\end{align*} \]

If a processor uses a single supply voltage and completes a program just on deadline, the energy consumption is minimised.

Consider two tasks $\tau_1, \tau_2$:

- Computation
  - $\tau_1$: $250 \times 10^6$ execution cycles; $\tau_2$: $750 \times 10^6$ execution cycles;
  - Deadline: 25 seconds.
  - Processor nominal (maximum) voltage: 5V.
  - Energy:
    - 40 nJ/cycle at nominal voltage.
    - 25 nJ/cycle at $V_{DD} = 4V$.
- Processor speed:
  - 50MHz (50 $\times 10^6$ cycles/sec) at nominal voltage.
  - 40MHz at $V_{DD} = 4V$. 

Find the voltage so that the tasks just meet their deadline => you have minimised energy consumption!
Considering Task Particularities

Energy consumed by a task:

\[ E = \frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot N_{SW} \cdot N_{SW} \]

Average energy consumed by task per cycle:

\[ E_{CT} = \frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot N_{SW} \]

Often tasks differ from each other in terms of executed operations ⇒ \(N_{SW}\) and \(C\) differ from one task to the other.

The average energy consumed per cycle differs from task to task.

Considering Task Particularities (cont'd)

Consider two tasks \(\tau_1, \tau_2\):

- Computation
  - \(\tau_1\): 250\( \times 10^6\) execution cycles; \(\tau_2\): 750\( \times 10^6\) execution cycles;
  - Deadline: 25 seconds.
  - Processor nominal (maximum) voltage: 5V.
  - Processor speed:
    - 50MHz (50\( \times 10^6\) cycles/sec) at nominal voltage.
    - 40MHz at \(V_{DD} = 4V\).
    - 25MHz at \(V_{DD} = 2.5V\).
- Energy \(\tau_1\):
  - 50 nJ/cycle at \(V_{DD} = 5V\).
  - 32 nJ/cycle at \(V_{DD} = 4V\).
  - 12.5 nJ/cycle at \(V_{DD} = 2.5V\).
- Energy \(\tau_2\):
  - 12.5 nJ/cycle at \(V_{DD} = 5V\).
  - 8 nJ/cycle at \(V_{DD} = 4V\).
  - 3 nJ/cycle at \(V_{DD} = 2.5V\).

\[ E_{total} = 32nJ/cycle \times 250 \times 10^6 \text{cycles} + 32nJ/cycle \times 750 \times 10^6 \text{cycles} \]

\[ E_{total} = 14 J \]

\[ E_{total} = 12.5nJ/cycle \times 250 \times 10^6 \text{cycles} + 12.5nJ/cycle \times 750 \times 10^6 \text{cycles} \]

\[ E_{total} = 12.5 J \]

\[ V^2 \times 250 \times 10^6 \text{cycles} \]

\[ 750 \times 10^6 \text{cycles} \]

\[ E_{total} = 14 J \]

\[ V^2 \times 250 \times 10^6 \text{cycles} \]

\[ 750 \times 10^6 \text{cycles} \]

\[ E_{total} = 12.5 J \]
Considering Task Particularities (cont’d)

☞ If power consumption per cycle is not constant (but differs from task to task), the rule on slide 33 is not true any more.
Voltage levels have to be reduced with priority for those tasks which have a larger energy consumption per cycle.

☞ One particular voltage level has to be established for each task, so that deadlines are just satisfied.

Discrete Voltage Levels

☞ Practical microprocessors can work only at a finite number of discrete voltage levels.

The “ideal” voltage $V_{\text{ideal}}$, determined for a certain task does not exist.

☞ A task is supposed to run for time $t_{\text{exe}}$ at the voltage $V_{\text{ideal}}$.

On the particular processor the two closest available neighbours to $V_{\text{ideal}}$ are: $V_1 < V_{\text{ideal}} < V_2$.

You have minimised the energy if you run the task for time $t_1$ at voltage $V_1$ and for $t_2$ at voltage $V_2$, so that $t_1 + t_2 = t_{\text{exe}}$.

Scheduling Policies

☞ The techniques described here, in order to find optimal voltage levels for real-time tasks, can be applied both with:

• Static cyclic scheduling

• Priority-based scheduling

The Pitfalls with Ignoring Leakage

$$E = NC \cdot C_{\text{tap}} \cdot V_{\text{bus}} \cdot (V_{\text{dd}} - K_1 \cdot e^{C_1 \cdot t_{\text{run}}} - K_2 \cdot e^{C_2 \cdot t_{\text{run}}} - K_3 \cdot e^{C_3 \cdot t_{\text{run}}} - e^{C_4 \cdot t_{\text{run}}} + V_{\text{bus}} \cdot t_{\text{run}}) \cdot t$$

Minimise this and ignore the rest!
The Pitfalls with Ignoring Leakage

1. We don’t optimize global energy but only a part of it!
2. We can get it even very wrong and increase energy consumption!

\[ E = NC C_{\text{eff}} V^2 + L_{\text{crit}} (V_{\text{dd}} + K_1 e^{-K_2 V_{\text{dd}} + K_3 - V_{\text{bs}}}) f \]

- Dynamic decreases with \( V_{\text{dd}} \) regardless of increased time.
- Leakage decreases with \( V_{\text{dd}} \) but growth with time!

\[ E = NC C_{\text{eff}} V^2 + L_{\text{crit}} (V_{\text{dd}} + K_1 e^{-K_2 V_{\text{dd}} + K_3 - V_{\text{bs}}}) f \]

Leakage energy

Dynamic energy

Dynamic + Leakage

Critical point!

If you go beyond this with \( V_{\text{dd}} \) energy grows
Summary

• Power consumption becomes a central issue for embedded systems design.
• Power/energy consumption can be reduced by reducing supply voltage, switching activity, switched capacitance, number of executed cycles.
• There are means at all levels of the design to reduce power consumption: circuit, logic, behavioral, architecture, system level.
• At system level we distinguish dynamic techniques (applied during run-time) and static techniques (applied at design time).

Summary (cont’d)

• Dynamic power management is implemented by the operating system, and is mainly used in portable appliances to shut down or place in stand by unused devices.
• Typical policies for power management are: time-out, predictive, and stochastic.
• Both at task mapping and at scheduling, design decisions can be made with have a huge impact on power/energy consumption.
• Real-time scheduling in the context of processors with voltage scaling is extremely interesting. The main trade-off is voltage level vs. execution time. One has to find the optimal voltage levels such that energy consumption is reduced and deadlines are still fulfilled.