A Methodology and a Case Study of Dynamic Power Management for Embedded Systems

Sara Bocchio
STMicroelectronics – Italy

Carlo Brandolese
Simone Corbetta
William Fornaciari
Politecnico di Milano – Italy

FETCH 2012
Outline

Introduction

ReISC III SoC Virtual Platform
- Hardware Layer
- Software Layer

Application
- Functional description
- Customized Platform

Experimental results
- Analysis & Exploration

Conclusions & Future work
Introduction

The goals of the present work are to
1) Develop a flexible, power-enabled, TLM virtual platform
2) Provide models to build a ReISC based SoC
3) Develop a software power management module
4) Perform design space exploration on power management resources

The STMicroelectronics ReISC III SoC
- Integrates a 32bit RISC core
  - With split data/instruction scratchpad memories
  - With DMA support
- Integrates several peripherals
  - Buses, timers, watchdogs, AD/DA, ...
- Supports "system-wide" power modes: RUN, SNOOZE and SLEEP
  - Some peripheral may not work properly in SNOOZE and SLEEP modes
- Supports "local" power modes
  - Influencing subsets of devices
  - Controlled by a dedicated power management module
The ReISC SoC Virtual Platform is structured into:

**Application software**
- The application C source code

**Operating System**
- FreeRTOS source code

**DPM**
- The Dynamic Power Manager
- Custom source code

**Drivers**
- Device drivers source code

**Hardware platform**
- A SystemC TLM model of the SoC
- The embedded ReISC processor ISS

**Power Platform**
- Support SystemC classes for power sensing
Hardware Layer – SoC Architecture

Modeled with:
- SystemC
- Instruction Set Simulator
Hardware Layer – Power Islands

A Power Island is a region of the SoC with
- Independent and scalable power supply
- Independent and scalable clock frequency

The ReISC SoC provides three power islands
- PI0 - Normal
  - Always on
  - Power supply and clock frequency are fixed to the global values
- PI1 - Gated
  - On
  - NoClock
  - Off
- PI2 - Tunable
  - On
  - Snooze
  - Sleep
  - NoClock
  - Off
Hardware Layer – Power Platform

Is a set of SystemC modules implementing power models

- Power FSM
- Analyzers
- Tails

Power FSMs

- Model the devices based on
  - The specific behaviour of the device
  - The specific power island a device belongs to

Analyzers

- Accumulate power consumption
- Available for each device and each power FSM state

Tails

- Log power information over time
- Available per each device
Hardware Layer – Power FSMs

Simple power FSMs
- Mainly depend on the power island
- Do not significantly depend on the behaviour of the specific device

Complex power FSMs
- Depend on the power island
- Depend on the behaviour of the specific device
  - The FSM describing the device behaviour is often a macrostate of the complete power FSM
Hardware Layer – Power Analyzers & Tails

Suppose that a device switches between states A and B as follows:

- **State A (0.5 mW)**: 15ms
- **State B (1.0 mW)**: 10ms
- **State A (0.5 mW)**: 25ms

Analyzers save power/timing data:
- Per each device
- Per each state

Tails provide power profiling:
- Per each device
- Over time
Software Layer – DPM & Drivers

The Application implements

- The functional logic
- The policies for dynamic power management

The Dynamic Power Manager

- Provides an API for controlling the power modes of the different power islands
- These functions use device drivers to configure the specific device belonging to a power island

The Drivers layer controls

- Specific devices
- The power manager special device
The application for the case study is an ECG Holter device

- Samples cardiac pulses & saves pulse traces
- Performs feature extraction & checks for anomalies
- Warns the patient if anomalies are detected

The device is structured as shown below
Customized Platform

Modeled with:
- SystemC
- Instruction Set Simulator

Unused Modules:
- ISS / SystemC
Results – Application Traces without DPM

Execution traces

- Are derived from power-island/device tails
- Without power management
  - They mimic the device-specific power FSM
  - Power-island specific unused states are shown in red

PI0: Core

OFF

ON

PI1: ADC

SAMPLE

IDLE

NOCLK

OFF

PI2: UART

RX

TX

IDLE

NOCLK

SNOOZE

SLEEP

OFF
Results – Energy consumption without DPM

The Holter application

- Samples cardiac pulses every 30ms
- Examines samples in a "period"
  - A period consists of 28 pulses, i.e. 840ms
  - All processing is performed on a period

Simulations collect energy consumption for the three power islands

<table>
<thead>
<tr>
<th>PI0</th>
<th>PI1</th>
<th>PI2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Gated</td>
<td>Tunable</td>
</tr>
</tbody>
</table>

Energy consumption (µJ)
From tails and analyzers it can be noticed that

- Most of the power is spent
  - By power island PI0, where the core resides
  - By power island PI1 & PI2 while in their IDLE states

This suggests

- To reduce power supply of PI1 and PI2 while while IDLE
  - UART/ADC events and requests to access GPIOs may need to restore the IDLE state
- To trade-off energy consumption reduction with delays and state-transition energy overheads

To find the optimal power management policy we have

- Augmented the task source code with suitable macros
  - Expanded in the exploration phase to specific power configuration calls
- Described the design space in terms of allowed power modes per PI
  - Using an XML configuration file
- Performed designs space exploration with MOST
  - A general purpose exploration tool developed at Politecnico di Milano
Results – Power management policies exploration

**Augmented code**
```c
void TaskX() {
    TASKX_ENTER
    // Original body
    TASKX_EXIT
}
```

**Configuration**
```c
...#define TASKX_ENTER PI1_Set(IDLE);
#define TASKX_EXIT PI1_Set(NOCLK);
...
```

**Design Space Definition**
```xml
<param name="TASKX_ENTER">
    <val> "PI1_Set(OFF);" </val>
    <val> "PI1_Set(NOCLK);" </val>
    <val> "PI1_Set(IDLE);" </val>
</param>
...```

**Compile & Link**

**MOST Wrapper**

**ReISC SoC Virtual Platform + IS Simulator**

**Analyzers & Tails**

**Time & Energy Figures**

** MOST DSE**

new configuration

**Analysis results**

**Optimal DPM Policy**

---

**Notes:**
- Power management policies exploration
- Design Space Definition
- Augmented code
- Compile & Link
- ReISC SoC Virtual Platform + IS Simulator
- Analyzers & Tails
- Time & Energy Figures
- MOST Wrapper
- Optimal DPM Policy
Results – Energy reduction

Policies

1) No power management: PI1 & PI2 in IDLE mode between device accesses
2) PI2 in SNOOZE mode between UART transmission bursts
3) PI2 in SLEEP mode between UART transmission bursts
4) PI2 in NOCLK mode between UART transmission bursts
5) As 3), but with PI1 in NOCLK when not sampling
6) As 3), but with PI1 in OFF when not sampling
Conclusions

Development of

- A SystemC TLM Virtual SoC Platform based on the ReISC III Core
  - Integrated with ReISC III instruction set simulator
  - Providing power consumption models for devices
- A general power analysis framework
  - Power FSMs
  - Analyzers & Tails
- A Dynamic Power Management module

Biomedical case study: Holter ECG

- Completely implemented in C on FreeRTOS
- Integrated in the simulation and analysis framework

Encouraging results

- Up to 18% energy saving on the controllable power islands
- No loss of performance (throughput & responsiveness) of the application
Future activities

More complex case studies

- Software
  - Higher number of tasks
  - Higher computational workload
  - More complex task communication/synchronization schemes

- Hardware
  - Usage of more devices provided by the platform
  - Support for the ReISC IV core, providing independent DVFS capabilities

Improved models

- More accurate power state machines for devices
  - Better modelling of energy associated to state transitions

- Explicit power modelling of memories
  - Now considered as part of the core power consumption
  - Average values used

- Improved ISA-level and/or source-level software power models
  - Assembly instruction dependent
Thank you