System-Level Synthesis of Ultra Low-Power WSN Node Controllers

A. Pasha, O. Sentieys, S. Derrien
University of Rennes
IRISA/INRIA

sentieys@irisa.fr
Outline

• Motivation
• Related Work
• Proposed Approach
  • System-Level Model
  • Micro-Task Synthesis
• Experimental Validation
• Conclusion and Future Perspectives
Wireless Sensor Network (WSN)

- Dense network of small nodes
  - Sensing, actuating, control
  - Processing, storage
  - Communication, relaying
- Low cost, low energy
  - Towards *autonomous self-powered* sensor nodes
  - 0.1-1 mW on active period
What are the main sources of energy consumption?
- **Radio**: 30-70mW
- Processor: 5-10mW
Objectives for Energy Reduction

- How can we design an energy-efficient platform for wireless sensor network nodes?
  - Platform = software + hardware + protocols

- (1) Decrease transmission (Tx) power
  - Power-aware signal processing
  - Error detection and correction

- (2) Optimize radio activity (MAC layer)
  - Wake-up interval tuning

- (3) Power optimization of the hardware
  - Co-processing, DVS, power-gating, etc.
Based on low-power MCUs
  - PowWow, ScatterWeb (MSP430) [2]
  - WiseNet (CoolRISC) [3]
  - MICA-II (ATMega128L) [4]
  - Many others [6-8]

Further improvement?
  - Hardware specialization of communication stack
    - Possible dynamic power saving
  - Hardware acceleration of advanced signal processing techniques
    - MIMO, ECC, etc.
    - Communication vs. Computation energy
  - Static power reduction
Power Gating [9]

- Addition of a sleep transistor to between
  - \( V_{DD} \) and logic circuit
  - OR
  - Ground and logic circuit
Outline

• Motivation
• Related Work
• Proposed Approach
  • System-Level Model
  • Micro-Task Generation
• Experimental Validation
• Conclusion and Future Perspectives
Heterogeneous computation
Task Mapping

- Represent WSN application as Task Flow Graph
  - Event-driven execution
  - Run-to-completion semantic
  - Communicating through data-flow and events
- Replace software tasks by concurrent hardware micro-tasks and system-monitor
  - Reduction in dynamic power thanks to specialization
  - Hardware micro-task is power-gated
  - Turned-On only when needed
  - Drastic reduction in static power
- Automatic generation of the global hardware
Task Mapping

System Monitor

Sensor (e.g. heat) → V0 Port
Radio chip (e.g. CC2420) → V0 Port

MT2 → MT3 → MT4 → MT5

Local Memory A' → Flash Memory G → Local Memory B'

MT0

start
powerOn()

ext. event
shutDown()

calcNeigh()

time_OUTA

sendData()

ack_OK

receiveAck()

time_OUTB

sendBeacon()
LoMiTa: ultra-low-power Tasking
Contributions of Current Work

- **Proposal** of a Domain Specific Language (DSL) for system-level model description
- **Design-flow** for hardware system monitor synthesis
  - From given system description in DSL
- **Design-flow** for hardware micro-task synthesis
  - From given task description in ANSI-C
- **Experimental validation**
  - Power-gating applicability to the proposed approach
  - Power benefits of the proposed approach over MCU-based solutions
    - Case-study application
Outline

• Motivation
• Related Work
• Proposed Approach
  • System-Level Model
  • Micro-Task Generation
• Experimental Validation
• Conclusion and Future Perspectives
System-Level Model
• Domain Specific Language (DSL)
• System description (node task graph)
• Developed in Xtext, EMF-based MDE tool
• No-ons of micro-tasks, events, task hierarchies, and priorities, shared memories, and IOs
• Conversion to a formal IR in the form of meta-models
• VHDL description generated for system monitor using code generation utilities

1 EMF = Eclipse Modeling Framework
2 MDE = Model Driven Engineering
3 IR = Intermediate Representation
System Monitor (SM)

- Main features of the system monitor (SM)
  - Replaces the OS present in MCU-based solutions
  - Manages activation of hardware micro-tasks and shared resources
- Ensures the hypothesis:
  - “Two micro-tasks sharing a write-access to a shared resource are never active simultaneously”
- Benefit:
  - Reduced hardware complexity for shared resource access

(a) Access control logic in conventional systems
(b) Access control logic in our system
Outline

• Motivation
• Related Work
• Proposed Approach
  • System-Level Model
  • Micro-Task Synthesis
• Experimental Validation
• Conclusion and Future Perspectives
Hardware Micro-Task

```c
void encipher(unsigned long * v, unsigned long * w)
{
    unsigned long
    y = v[0], z = v[1], sum = 0, delta = 0x9E3779B9,
    a = k[0], b = k[1], c = k[2], d = k[3], n = 32;
    while (n-- > 0)
    {
        sum += delta;
        y += (z << 4) + a + z + sum + (z >> 5) + b;
        z += (y << 4) + c + y + sum + (y >> 5) + d;
    }
    w[0] = y, w[1] = z;
}
```

Task in C with a run-to-completion semantic running on a power-gated MCU

SW micro-tasks in C

HW micro-task in VHDL
Micro-Task Synthesis

MicroTask Synthesis Design Flow

Task.c
Compiler Front-end
CDFG-Level IR
Tree-based Instruction Selection and Mapping
Custom Datapath Model
Assembly-Level IR
Register Allocation
Bitwidth Adaptation
Assembly-Level IR
FSM Generation
Datapath Generation
EMF-based RTL Models for FSM and Datapath
Code-Generation Tool
FSM.vhd
Datapath.vhd
Features

- Tree-based instruction selection
  - Polynomial time complexity
  - **WSN-specific** instruction patterns
    - Direct I/O-based operations etc.
  - Not constrained by pre-existing MPU instruction-set
  - **Cost function** was based on number of clock cycles

- Customized hardware generation
  - Task-specific FSM/datapath generation
  - Choice for 8-bit/16-bit datapath

- Based on MDE principles
  - Exploits RT-level architecture meta-models
  - Code generation facilities for several outputs like VHDL, SystemC and behavioral-equivalent C (for simulation)
## Comparison to HLS/ASIP Design

<table>
<thead>
<tr>
<th></th>
<th>HLS design-flow</th>
<th>ASIP design-flow</th>
<th>LoMiTa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datapath Selection</td>
<td>Iterative (mostly)</td>
<td>User-guided (ISA-based)</td>
<td>User-guided (ISA-based)</td>
</tr>
<tr>
<td>Instruction Selection</td>
<td>No</td>
<td>Yes (both DAG/Tree-based)</td>
<td>Yes (Tree-based)</td>
</tr>
<tr>
<td>Hardware</td>
<td>FSM + Datapath</td>
<td>Instruction-set processor</td>
<td>FSM + Datapath</td>
</tr>
<tr>
<td>Application domain</td>
<td>Data-intensive (mostly)</td>
<td>Data-intensive</td>
<td>Control-oriented</td>
</tr>
</tbody>
</table>
Outline

• Motivation
• Related Work
• Proposed Approach
  • System-Level Model
  • Micro-Task Synthesis
• Experimental Validation
• Conclusion and Future Perspectives
Micro-Task Synthesis

• Transistor-level characterization
  • Spice simulations for various sleep transistor sizes, task complexity, etc.
  • Wake-up time and energy models [13]

• Benchmarks
  • Tasks extracted from WSN benchmarks e.g. WiSeNBench [10]
    • Tea (Tiny Encryption Algorithm), crc8, crc16, etc.
  • Control tasks from the OS layer of PowWow platform
    • Radio-link through SPI, geographical routing protocol, etc.

• Case study
  • Developed using the benchmark application/control tasks
Task Flow Graph (Case Study)
Micro-Task Synthesis: Set-Up

• Hardware synthesis and power estimations
  • VHDL generated through the design-flow for case-study tasks
  • Synthesis and gate-level power estimation using Design Compiler from Synopsys
  • 130 nm and 65 nm CMOS library

• Energy gains w.r.t. power and energy consumptions of
  • MSP430F21x2 from datasheet and software profiling
  • MSP430 MCU-core synthesized in the same conditions (without memory and peripherals)
# Dynamic Power Savings (8-bit)

<table>
<thead>
<tr>
<th>Task Name</th>
<th>No. States</th>
<th>Time (µs)</th>
<th>Power (µW)</th>
<th>Energy (pJ)</th>
<th>P. Gain* (x)</th>
<th>E. Gain* (x)</th>
<th>Area (µm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>crc8</td>
<td>71</td>
<td>4.4</td>
<td>30</td>
<td>132</td>
<td>292/32</td>
<td>339/37</td>
<td>5832</td>
</tr>
<tr>
<td>crc16</td>
<td>103</td>
<td>6.4</td>
<td>47</td>
<td>300</td>
<td>187/21</td>
<td>141/15</td>
<td>8732</td>
</tr>
<tr>
<td>tea-dec</td>
<td>586</td>
<td>36.6</td>
<td>84.5</td>
<td>3090</td>
<td>104/11.5</td>
<td>78/8.6</td>
<td>19950</td>
</tr>
<tr>
<td>tea-enc</td>
<td>580</td>
<td>36.2</td>
<td>87.3</td>
<td>3160</td>
<td>101/11</td>
<td>75/8</td>
<td>20248</td>
</tr>
<tr>
<td>calcNeigh</td>
<td>269</td>
<td>16.8</td>
<td>74</td>
<td>1248</td>
<td>118/13</td>
<td>143/22</td>
<td>16873</td>
</tr>
<tr>
<td>sndBeacon</td>
<td>672</td>
<td>42</td>
<td>33</td>
<td>1400</td>
<td>264/29</td>
<td>198.5/21.7</td>
<td>10578</td>
</tr>
<tr>
<td>rcvBeacon</td>
<td>331</td>
<td>21</td>
<td>27.3</td>
<td>565</td>
<td>322/35</td>
<td>247.6/27</td>
<td>5075</td>
</tr>
</tbody>
</table>

* Gains w.r.t. the power and energy consumptions of an MSP430F21x2 (datasheet) and an open source MSP430 MCU-core
## Dynamic Power Savings (16-bit)

<table>
<thead>
<tr>
<th>Task Name</th>
<th>No. States</th>
<th>Time (µs)</th>
<th>Power (µW)</th>
<th>Energy (pJ)</th>
<th>P. Gain* (x)</th>
<th>E. Gain* (x)</th>
<th>Area (µm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>crc8</td>
<td>71</td>
<td>4.4</td>
<td>55</td>
<td>242.6</td>
<td>159/17.4</td>
<td>185/20</td>
<td>10348</td>
</tr>
<tr>
<td>crc16</td>
<td>73</td>
<td>4.6</td>
<td>55</td>
<td>251</td>
<td>159/17.5</td>
<td>168/18</td>
<td>10280</td>
</tr>
<tr>
<td>tea-dec</td>
<td>308</td>
<td>19</td>
<td>153</td>
<td>2940</td>
<td>58/6</td>
<td>82/9</td>
<td>27236</td>
</tr>
<tr>
<td>tea-enc</td>
<td>306</td>
<td>19</td>
<td>152</td>
<td>2910</td>
<td>58/6</td>
<td>81/9</td>
<td>27069</td>
</tr>
<tr>
<td>calcNeigh</td>
<td>269</td>
<td>17</td>
<td>142</td>
<td>2392</td>
<td>62/6.7</td>
<td>74/8</td>
<td>24745</td>
</tr>
<tr>
<td>sndBeacon</td>
<td>672</td>
<td>42</td>
<td>58</td>
<td>2440</td>
<td>151.5/16.5</td>
<td>114/12</td>
<td>14863</td>
</tr>
<tr>
<td>rcvBeacon</td>
<td>332</td>
<td>21</td>
<td>50</td>
<td>1036</td>
<td>176/18</td>
<td>135/15</td>
<td>9485</td>
</tr>
</tbody>
</table>
8-bit vs. 16-bit Micro-Task

Dynamic Power Comparison (Y-axis units = in µW)

Silicon Area Comparison (Y-axis units = in µm²)

Dynamic Energy Comparison (Y-axis units = in pJ)
Static Power Savings

- Static power of the MSP430 = 1.54 µW
  - Estimated static power of a micro-task = ~317 nW
    - Smaller data-memory + power gating
    - 4x to 5x reduction in static power

- Adds on to the overall energy gain
  - Dynamic energy gain = 198.5 (for sndBeacon task)
  - Total energy gain = 279
• Synthesis performed for case-study example
• Dynamic power: 12 μW (8.8 mW for MSP430)
• Static power: 606 nW (1.54 μW for MSP430)
  • Could be reduced to 150 nW if low-voltage libraries are used
• Area: 1710 μm² (approx. 214 Nand-gates in 130 nm)
Prototype chip

- CMOS 65nm, ST
- MSP430+PG MAC accelerator
- Received last week!
Outline

• Motivation
• Related Work
• Proposed Approach
  • System-Level Model
  • Micro-Task Generation
• Experimental Validation
• Conclusion and Future Perspectives
Conclusion

• WSN is a fast growing technology
  • Enormous potential for applications domains
• Power is the most important constraint for WSN
  • Small form-factor and requirements for autonomy
• Current solutions using low-power MCUs have limitations
  • High average power consumption and low energy efficiency
• Hardware specialization of control and computation subsystem with a technology to reduce static power is promising
• We proposed “LoMiTa” a design-flow based on hardware specialization and power-gating
Conclusion

• System-level design-flow for system monitor synthesis
  • From given system description in DSL
• Design-flow for hardware micro-task synthesis
  • From given task description in ANSI-C
• Estimates showed important power savings
  • One to two orders for dynamic power
  • 4x to 5x reduction in static power
  • 50% reduction in wake-up delays
Power-Gated Reconfigurable Structures

• Study of power-gated reconfigurable architecture
  • To provide re-programmability and further save silicon area

• Targets the feasibility of porting power gating to fine-grain reconfigurable architectures

• PowWow version II might consist of a general purpose power-gated MCU with a reconfigurable SoC using LoMiTa design-flow


References


