Profile Assisted Online System-Level Performance and Power Estimation for Dynamic Reconfigurable Embedded Systems

Jingqing Mu, Roman Lysecky
Department of Electrical and Computer Engineering
University of Arizona
jqmu@ece.arizona.edu, rlysecky@ece.arizona.edu
http://www.ece.arizona.edu/~embedded
Introduction – Hardware/Software Partitioning
(*Design-time/Static Partitioning*)

- **Static HW/SW Partitioning**
  - **Pros:**
    - Speedup of 2X to 10X
    - Energy reduction of 25% to 95%
  - **Cons:**
    - Cannot adapt to changing system/application execution
Introduction – Hardware/Software Partitioning
(Dynamic Reconfiguration and Adaptable Systems)

- Runtime Reconfiguration & FPGAs
  - Reconfigure FPGA for different HW circuits
  - Dynamically adapt HW at runtime as needed
  - Optimize performance or power consumption

- Transmutable Processors [Bauer et al., DAC 2008][Shafique et al., DAC 2009]
  - Reconfigure custom instructions within processor datapath
  - Adapts to non-deterministic application behavior using runtime execution behavior
  - Average speedups of 3.6X or energy reduction of 29%

- Input-Driven Self Configuration [Bruneel et al. DATE 2009]
  - Adapts HW circuit based on actual inputs to the systems (e.g. common input parameters utilized within FIR computation)
Introduction – Hardware/Software Partitioning
(Dynamic Reconfiguration and Adaptable Systems)

• Runtime Reconfiguration & FPGAs (cont.)
  • Dynamic Co-processor Selection [Fu & Compton, FPGA 2005][Fu & Compton, FCCM 2008]
    • Selects among multiple coprocessor alternatives for HW circuits with varying area/speedup tradeoffs
    • Guided by changes in application phase behavior at runtime
Introduction – Hardware/Software Partitioning
(Dynamic Reconfiguration and Adaptable Systems)

- Runtime Reconfiguration & FPGAs (cont.)
  - Warp Processors [Mu & Lysecky, TODAES 2009][Lysecky et al., TODAES 2006]
    - Dynamically translates critical SW kernels to HW to optimize performance or power
    - Performance-driven: Average speedup of 2.5X
    - Low-Power: Average power reduction of 74%

1. Application initially executes on microprocessor
2. Profiler dynamically monitors application execution
3. On-chip optimizer translates kernels to HW within FPGA
4. Reconfigure FPGA and update processor & FPGA V/F
5. Optimized execution 2-10X faster – or – consumes 75% less power
Introduction – Hardware/Software Partitioning
(Dynamically Adaptable Systems)

- Dynamically Adaptable Systems:
  - Provides advantages over statically partitioned/optimized implementations:
    - Data input can affect application execution
    - Execution environment can affect application execution
    - Human-interaction non-deterministically affects execution behavior

- **Accurate runtime execution statistics and estimation methods are needed to guide dynamic reconfiguration, adaptation, and/or optimization methods**
Dynamically Reconfigurable Embedded Systems
(Target Reconfigurable System Architecture)

- Dynamically Adaptable System Architecture:
  - 624 MHz XScale Processor
    - Voltage scalable with discrete voltage and frequency settings
    - 32 KB instruction and data caches
  - FPGA
    - Model on Xilinx Viretx-4 with 2 independent reconfigurable regions
    - Maximum frequency of 175 MHz
      - Frequency scalable in 5MHz increments
      - Note: actual maximum frequency dependent on hardware kernel
  - 32 MB Off-chip Memory
  - Dynamic Application Profiler (DAPprof) [Shankar & Lysecky, DAC 2009]
    - Loop-level profiler utilized to determine dynamic kernel execution statistics
Dynamically Reconfigurable Embedded Systems
(Target Reconfigurable System Architecture)

- Dynamically Adaptable System Architecture
  - Software:
    - Software available for all kernels
  - Hardware:
    - Hardware kernels determined at design time for any application kernel yielding a performance/power improvement over the original software application
    - If kernel is not available in hardware at time of execution, software will be utilized

- Performance-driven:
  - Select subset of kernels to configure within FPGA to maximize performance

- Low-Power:
  - Select kernels to implement within FPGA, voltage/frequency for processor, and frequency for each kernel to minimize power consumption
Profile Assisted System-Level Performance/Power Estimation
(Profile Epoch Based Re-Optimization)

- Dynamic Application Profiler (DAProf)
  - Non-intrusive application profiler
    - Provides loop/kernel-level profiling identifying frequently executed loops
    - Provides execution breakdown of executions and average iterations per execution
    - Greater than 95% accuracy in reported statistics
  - Profile epoch ($pe$) defines the granularity of profile updates (e.g. every 30 ms)
  - Updated profile utilized to re-optimize system using profile assisted estimation framework
Profile Assisted System-Level Performance/Power Estimation
(Estimation Framework)

- **Kernel Implementation for each Kernel ($K_i$)**
  - Execution Cycles per Iteration ($K_i\text{Cycles/Iter}$)
  - Communication Cycles per Execution ($K_i\text{CommCycles}$)
  - Maximum Frequency ($\text{MAX}(F_{Ki})$)
  - Static Power Consumption ($P_{Ki,\text{static}}$)
  - Dynamic Power Consumption ($P_{Ki,\text{MAX}(FKi)}$)
  - Reconfiguration Time ($T_{Ki,\text{Reconf}}$)
  - Reconfiguration Power ($P_{Ki,\text{Reconf}}$)
  - SW Execution Estimate ($\text{ExecPath}_i$)

- **Execution Profile for each Kernel ($K_i$)**
  - Kernel Executions ($\text{Execs}_i$)
  - Average Iterations per Execution ($\text{AvgIter}_i$)

- **FPGA Configuration**
  - Current Kernels in FPGA ($R_{\text{current}}$)

- **Processor/Cache Statistics**
  - Average Cycles per Instruction ($\text{CPI}$)
  - Instruction Cache Hits (I$\text{Hits}_i$)/Misses (I$\text{Misses}_i$)
  - Data Cache Hits (D$\text{Hits}_i$)/Misses (D$\text{Misses}_i$)
  - Data Cache Write Backs (D$\text{WriteBacks}_i$)
  - Data Cache Replacements (D$\text{Replacements}_i$)

- **Kernel Selection and V/F Configuration**
  - Microprocessor Voltage/Frequency ($V_{\mu P}/F_{\mu P}$)
  - Kernel Selection for next Epoch ($R_{\text{next}}$)
  - Kernel ($K_i$) Frequency ($F_{Ki}$)

- **Decision Variables**

- **Estimated Performance/Power for Next Profile Epoch**
  - Power ($P[pe_{i+1},pe_{i+2}]$)
  - Speedup ($S[pe_{i+1},pe_{i+2}]$)

- **Static Information**
- **Dynamic Information**
Profile Assisted System-Level Performance/Power Estimation

(Performance Estimation)

- Performance Estimation
  - Estimate speedup starting with Amdahl’s Law considering kernels implemented within FPGA
    \[ S_{HW/SW[p_{i+1}, p_{i+2}]} = \frac{1}{1 - \sum \%Exec_i + \sum \%Exec_i \text{Speedup}_i} \]
  - Utilize estimate of percentage of execution time for each kernel in previous profile epoch (%Exec_i)
    - Estimated using dynamic profile and estimate of software execution cycles based on static analysis of application binary
    - **Average Execution Path:**
      \[ \%Exec_i = \frac{\text{AvgExecPath}_i \times \text{Exec}_i \times \text{AvgIters}_i}{\sum \text{AvgExecPath}_i \times \text{Exec}_i \times \text{AvgIters}_i} \]
    - **Maximum Execution Path:**
      \[ \%Exec_i = \frac{\text{MaxExecPath}_i \times \text{Exec}_i \times \text{AvgIters}_i}{\sum \text{MaxExecPath}_i \times \text{Exec}_i \times \text{AvgIters}_i} \]
Profile Assisted System-Level Performance/Power Estimation
(Performance Estimation)

- Performance Estimation (cont.)
  - Kernel speedup estimation consider both hardware execution cycles and communication cycles:

  \[
  \text{Speedup}_i = \frac{K_i \text{(SW Time)}}{K_i \text{Time} + K_i \text{CommTime}}
  \]

  - Normalize performance estimation to software:

  \[
  \frac{HW}{SW}_{\text{Time}} = \frac{1}{\text{Speedup}_{HW/\text{SW}}} = 1 - \sum \%\text{Exec}_i + \sum \frac{\%\text{Exec}_i}{\text{Speedup}_i}
  \]

  - Estimate impact of reconfiguration overhead:

  \[
  S_{HW/\text{SW}[\text{pe}_i, \text{pe}_{i+1}]} = 1 \times \%\text{ReconfTime} + S_{HW/\text{SW}[\text{pe}_i, \text{pe}_{i+1}]} \times (1 - \%\text{ReconfTime})
  \]

  - Percentage of reconfiguration time (\%\text{ReconfTime}) for the next profile epoch is estimated based on which new kernels must be configured with the FPGA:

  \[
  \%\text{ReconfTime} = \frac{\sum T_{K_i \text{Reconf}}}{|\text{pe}|}, \forall K_i \not\in R_{\text{Current}}
  \]
Profile Assisted System-Level Performance/Power Estimation
(Power Estimation)

- Power Estimation
  - Power Estimation utilizes performance estimation to determine overall system power including microprocessor, instruction cache, data cache, FPGA, and off-chip memory

  - Processor:
    \[
    P_{\mu P} = P_{\mu P(Active)} \cdot \mu P\%_{Active} + P_{\mu P(Idle)} \cdot \mu P\%_{Idle}
    \]
    \[
    \mu P\%_{Active} = \text{Speedup}_{\text{HW/SW}} \cdot \left( 1 - \sum \text{Exec}_i + \frac{\sum (\text{Exec}_i \cdot K_{\text{CommTime}})}{\text{Speedup}_i \cdot (K_{\text{CommTime}} + K_{\text{Time}})} \right)
    \]

- FPGA:
  \[
  P_{\text{FPGA}} = \sum P_{K_i(Static)} + \sum (P_{K_i(Dynamic), F_{K_i}} \cdot K_{\%_{Active}})
  \]
  - Dynamic power consumption for each kernel is scaled to the kernel frequency \((F_{K_i})\)
    \[
    P_{K_i(Dynamic), F_{K_i}} = P_{K_i(\text{MAX}(F_{K_i}))} \cdot \frac{F_{K_i}}{\text{MAX}(F_{K_i})}
    \]
Profile Assisted System-Level Performance/Power Estimation

(Power Estimation)

• Power Estimation (cont.)

  • Instruction and Data Cache:
    • Utilize a modified eCACTI cache model to estimate power consumption of individual cache accesses

\[
P_{DS} = \begin{bmatrix}
P_{DSRH} \times D_{\%RH} + P_{DSRM} \times D_{\%RM} + \\
P_{DSWH} \times D_{\%WH} + P_{DSWM} \times D_{\%WM} + \\
P_{DS\text{Idle}} \times D_{\%\text{Idle}}
\end{bmatrix}
\]

\[
P_{IS} = P_{ISRH} \times I_{\%RH} + P_{ISRM} \times I_{\%RM} \times I_{\%Miss} + P_{IS\text{Idle}} \times I_{\%\text{Idle}}
\]

• Off-Chip Memory:
  • Use cache access statistics to estimate off-chip memory accesses:

\[
P_{\text{Mem}} = P_{\text{Mem(Static)}} + \text{MemReads} \times P_{\text{Mem(Read)}} + \text{MemWrites} \times P_{\text{Mem(Write)}}
\]
Experimental Results

(Experimental Setup)

- Experimental Setup
  - Analyze accuracy and fidelity of profile assisted online estimation framework
    - Utilize estimation to perform both performance-driven and low-power optimization
    - Consider several single task applications from EEMBC, MediaBench, and PowerStone
  - Compare performance and power estimates with accurate simulation based performance and power analysis
    - XEEMU simulator utilized for XScale performance and power analysis [Herczeg et al., PATMOS 2007]
    - XEEMU validated using physical measurements to have average performance and power errors of 3% and 1.6%
  - Compare dynamic kernel selection and V/F configuration with optimal kernel selection determined through exhaustive simulation
Experimental Results
(Performance Estimation Accuracy & Fidelity)

• Performance Estimation
  • Maximum Execution Path provides best performance estimate
    • Average accuracy of 82%
    • akin to estimate best case performance improvement
    • Average Execution Path achieves better accuracy for some applications

• Perfect fidelity
  • Kernels selection using online estimation is equivalent to optimal kernel selection
  • Average *actual* speedup of 4.2X (2.2X without *brev*)
Experimental Results
(Power Consumption Accuracy & Fidelity)

• Power Consumption Estimation
  • Average Execution Path calculation provide best overall performance estimate
    • Average accuracy of 76%
    • Worst case accuracy of 66%

• Perfect fidelity
  • Kernels selection and V/F configuration using online estimation is equivalent to optimal
  • Average actual power reduction of 58%
Experimental Results
(Benefits of Dynamic vs. Static Partitioning)

- Dynamic optimization (even for a fixed application) can provide better power optimization compared to statically configured implementation
  - Execution behavior can change significantly with different task execution and application phases
  - Average power reduction of 31% (compared to original software)
  - Average power reduction of 14% (compared to optimal static configuration)
    - Maximum instantaneous power reduction of 25%

MULTITASKED APPLICATION (BREV, G3FAX, MATMUL, TBLOOK)
Conclusions and Future Work

• Conclusions
  • Profile assisted online system-level estimation framework provides an efficient method for estimating performance and power consumption of dynamically reconfigurable embedded systems
  • Capable of dynamically reducing power consumption in response to changing application demands
  • Average accuracy of 82% and 76% for performance and power consumption, respectively
  • Performance improves of 4.2X or power reduction of 58% compared to software only execution

• Future Work
  • Currently developing online methods for adjusting speedup and power estimation for dynamically changing multitasked applications
  • Develop fast search heuristic for online kernel selection and voltage/frequency scaling
Thank You.