Performance Evaluation of a Reconfigurable Instruction Set Processor

Mehdipour, Farhad
Faculty of Information Science and Electrical Engineering, Kyushu University

Noori, Hamid
Faculty of Information Science and Electrical Engineering, Kyushu University

Honda, Hiroaki
Faculty of Information Science and Electrical Engineering, Kyushu University

Inoue, Koji
Faculty of Information Science and Electrical Engineering, Kyushu University

https://doi.org/10.15017/14879
Performance Evaluation of a Reconfigurable Instruction Set Processor

Farhad Mehdipour, H. Noori, H. Honda, K. Inoue, K. Murakami

Faculty of Information Science and Electrical Engineering, KYUSHU UNIVERSITY, Fukuoka, JAPAN

{farhad@c.csce.kyushu-u.ac.jp}
Outline

• Reconfigurable Instructions Set Processors
• A Combined Analytical and Simulation-Based Model (CAnSO)
  • Model Extraction and Calibration
  • Basic Model Definitions
  • Speedup Formulations
  • Simplification and Calibration
• Experiments
  • Experimental Setup
  • Model Validation
  • Design Space Exploration Using CAnSO
  • Effects of Modifications
• Conclusions and Future Work
Designing Embedded Systems

- Embedded Microprocessors
- Application-Specific Integrated Circuits (ASICs)
- Application-Specific Instruction set Processors (ASIPs)
- Extensible Processors
Extensible Processors

Mechanism

- Acceleration by using CFU
- A hardware is augmented to the base processor
- Executes hot portions of applications
Extensible Processors

- Base processor (BP)'s fixed instruction set + Custom Instructions
- Goals
  - Improving the performance and energy efficiency
  - Maintaining compatibility and flexibility

LD/ST: Load / Store
CFU: Custom Functional Unit

Instruction Dispatcher
Register File

Custom Instructions
BP's Instructions
Custom Instructions

- Instruction set customization ↔ hardware/software partitioning (Identifying critical segments in applications)
- Custom Instructions (CIs) are
  - extracted from critical segments of an application and
  - executed on a Custom Functional Unit (CFU)

Critical segments:
Most frequently executed (Hot) portions of the applications

A CI can be represented as a DFG

1: SUBU    R3, R0, R3
2: ADDU    R10, R0, R0
3: SRA     R8, R10, 0x3
4: SLT     R2, R3, R8
5: BNE     R0,400488, R2

ISOCC 2008@ Busan, South Korea   November 24-25, 2008
Extensible Processors

- **Drawbacks:**
  - Lack of flexibility
  - Long time and cost of designing and verifying
  - Many issues associated with designing a new processor from scratch:
    - longer time-to-market and
    - significant NRE (Non-Recurring Engineering) costs

- **Solution**
  - Using a Reconfigurable Functional Unit (RFU) instead of fixed architecture CFU
Reconfigurable Processors

Microprocessor + Reconfigurable Logic → Reconfigurable Processor
Reconfigurable Instruction Set Processors (RISPs)

- Adding and generating custom instructions after fabrication
- Using a reconfigurable FU(RFU) instead of custom FU
How a RISP Works

RISP Baseline Processor

GPP: General Purpose Processor
RAC=RFU: Reconfigurable Accelerator

A Hot Basic Block
RISP Benefits and Drawbacks

Benefits
- Specialized datapath
- Shared hardware
- Higher Speedup
- Less power consumption

Drawbacks
- More area
- Difficult to use
Performance Evaluation of a RISP

- Performance evaluation of a RISP challenges
  - designing of a RISP architecture
  - optimizing an existing arch. for an objective function

- For a designer
  - obtaining optimum system configuration is desirable
  - a performance analysis in terms of the performance metrics (speedup, area and so on) is required

- Performance evaluation models
  - Structural models: includes empirical studies based on measurements and simulations of the target system
  - Analytical models: incorporates a system (usually simplified) structure to obtain mathematically solvable models
the RAC is responsible for executing almost 30% of dynamic instructions of applications in average
Model Extraction and Utilization

1st phase: Model Extraction

Applications

Cycle-accurate Simulation

Observations from simulation:
- Statistics on various parameters

Analytical Model

Analytical Model Simplification and Calibration

Combined Simulation & Analytical-Based Model (CAnSO)

2nd phase: Model Utilization

Application

Statistics (on various parameters) - from 1st phase

Combined Simulation & Analytical-Based Modeling (using CAnSO)

Estimated Performance Metrics
General Template of a RISP

Baseline Processor:
- CPU Core
- Register File
- Instr. Cache
- Data Cache
- Branch Predictor

RAC Interface:
- RAC
- Configuration Memory

Reconfigurable Instruction Set Processor
Basic Model Definitions

- **Base Processor**
  - an in-order general five-stage RISC processor

- **RAC**
  - a coarse-grained tightly-coupled reconfigurable hardware

- CIs are indexed for direct accessing of the configuration bit-stream

- The content of all registers are sent to the RAC (Shared RF)

- Controlling configurations
  - Hardware-based: starting address of CI and index to the config. Mem. is stored in a CAM for quick retrieval
  - Software-based: starting address of a CI is replaced with a special instruction

- Memory accesses

- Control instructions
Single and Continuous Executions

- **Execution on the Base Processor**
- **Execution on the RAC**
- **Overhead time**

**Single Execution**

**Continuous Execution**

---

ISOCC 2008@ Busan, South Korea  November 24-25,  2008
Speedup Formulation

- Latency of execution of $Ci_i$ instructions on the BP
  \[ f_{RAC} = \frac{\sum_{i=1}^{n_{CI}} (\tau^i_{BP} \times O_i)}{n_{tcc}} \]

- Total no. of executions of $Ci_i$
  \[ f_{BP} = \frac{\sum_{i=1}^{n_{CI}} (\tau^i_{BP} \times O_i)}{n_{tcc}} = 1 - f_{RAC} \]

- Fraction of instructions executing on BP

- Execution time on the RAC
  \[ s_P = \frac{n_{tcc}}{n_{tcc} - \sum_{i=1}^{n_{CI}} (\tau^i_{BP} \times O_i) + \psi(\theta, \tau)} \]

- Overall Speedup
  \[ \psi(\theta, \tau) = \sum_{i=1}^{n_{CI}} \left( \sum_{j \in S_i} (\theta_{ij} \times (\tau_{RAC} + \tau_{OVH})) + \sum_{j \in C_i} (\tau_{RAC} + \tau_{OVH}) + ((\theta_{ij} - 1) \times \tau_{RAC}) \right) \]

- Execution time on the BP

- Latency of RAC and the overhead reconfiguration time
The Effect of CI Length

- **Large CIs**
  - Including more instructions than the no. of available resources in the RAC

- **Temporal Partitioning**
  - Dividing larger CIs to a number of smaller CIs

\[
L = \{k|k \in \{1,\ldots,n_{CI}\}, l_k > n_{FU}\} \\
P = \{p_k|k \in L, p_k = \left\lfloor \frac{l_k}{n_{FU}} \right\rfloor\}
\]

\[
m'_{k \in L} = O_i \times p_k, m'_{k \notin L} = m_i
\]

\[
\theta'_{k \in L} = ((1,\ldots,1), (1,\ldots,1), \ldots, (1,\ldots,1)), \theta'_{k \notin L} = m'_{k \in L}, \theta'_{k \notin L} = \theta_{k \notin L}
\]

\[
S'_{i \in L} = \{1,\ldots,m'_i\}, S'_{i \notin L} = S_i
\]

\[
C'_{i \in L} = \emptyset, C'_{i \notin L} = C_i
\]
Side-Effects

- **Control Instructions**
  - the rate of miss-predicted branches might be reduced → higher speedup

- **Instruction Cache Misses**
  - no need for fetching instructions belonging to the CIs
  - access and miss rates to instruction cache are reduced
  - BP fraction reduces → speedup increases

\[
\begin{align*}
    s_O &= \frac{n_{tcc}}{n_{tcc} - \sum_{x=\{b,i\}} \delta_{xm} \times p_{xm} + \sum_{i=1} \left( \tau_{BP}^i \times O_i \right) + \psi'(\theta', \tau)}
\end{align*}
\]

no. of penalty cycles for branch miss-predictions/cache misses

variation in branch/cache miss-predictions/misses
RF’s Input/Output Ports

- Register file is shared between BP and RAC
- Additional clock cycles for reading/writing from/to the RF

\[
\tau'_{OVH} = \tau_{OVH} + \max(0, \frac{\Delta_i}{\Delta_{reg}}) + \max(0, \frac{\nabla_i}{\nabla_{reg}})
\]
The Assumed RAC Architecture
RAC’s Delay

- All FUs in the RAC implement similar operations

- Each mux receives
  - all outputs of the FUs in upper rows and
  - Outputs from its adjacent FUs at the same row

\[
\tau_{RAC}^w_h = \sum_{i=1}^{h} \tau_{FU} + \sum_{i=1}^{h-1} \tau_{MUX}_i^k, \quad k \in \{0,1,\ldots,w\}
\]

\[
\psi(\theta,\tau) = \sum_{i=1}^{n_C} \left( \sum_{j \in S_i} (\theta_{ij} \times (\tau_{RAC} + \tau_{OVH})) + \sum_{j \in C_i} ((\tau_{RAC} + \tau_{OVH}) + (\theta_{ij} - 1) \times \tau_{RAC}) \right)
\]
Control instructions are not supported

Reduction in instruction cache accesses as well as cache misses
  - average reduction in access to i-cache is almost 17%
  - average i-cache miss rate is almost 3%.

Average i-Cache Accesses: 17%
Misses: 3%
Simplification and Calibration

Fractions of Single & Continuous Executions

the ratio of single to continuous execution $\rightarrow 43\%$

$$s_o = \left\{ \frac{n_{tcc}^*}{n_{tcc}^* - \delta_{im} \times p_{im} - \sum_{i=1}^{n_{DF}} i \times \rho_{BP} \times O_{i}^*} \right\} + \psi^* \left( \theta^*, \tau_{RAC}^H + \tau_{OVH} \right)$$

$$\psi^* \left( \theta', \tau_{RAC}^H + \tau_{OVH} \right) = \sum_{i=1}^{n_{CI}} O_{i}^* \times \left( \alpha^* \times \tau_{OVH}^H + \tau_{RAC}^H \right)$$
Experimental Setup

- Fourteen applications of Mibench
  - automotive, security, consumer, network, telecommunication

- CIs (DFGs) are extracted from applications

- Simplescalar’s cycle-accurate simulator is extended to simulate a reconfigurable instruction set processor

- Model Establishment
  - simulating all applications
  - collecting required information
  - model simplification and calibration

~ 4 hours to completion on a PC: Dual Core, Intel 6600@2400Mhz, 2GB RAM
Model Validation

- **Average variation= 2%**

![Graph showing model validation results with cycle-accurate simulation, CAnSO, and uncalibrated CAnSO.](graph.png)

- **Cycle-accurate simulation**
- **CAnSO**
- **Uncalibrated CAnSO**

Results indicate a consistent average variation of 2% between the models, with specific performance benchmarks highlighted for various tasks.
Design Space Exploration Using CAnSO

- The design of a RAC including different components entails a multitude of design parameters
- Examining 100 design points using 14 applications:
  - Simulation: 17 days
  - CAnSO: 4 hours
- Using CAnSO, re-simulation is not needed after establishing the model

ISOCC 2008@ Busan, South Korea  November 24-25,  2008
Using CAnSO for Design Space Exploration of the RAC

Increasing the width of RAC increases speedup

Width > 6: no more speedup is achievable

the small heights $\rightarrow$ very low speedup

Height > 5: RAC's longer critical path delay $\rightarrow$ speedup declines
Effect of Modifications

Applying modification to the design:
- Small time is required for repeating the simulation
- Each iteration of the CAnSO takes less than a minute

![Graph showing speedup for different benchmarks with and without modification]
CONCLUSION

- Reconfigurable instruction set processors
- A combined analytical and simulation-based model (CAnSO)
- Suitable for exploring a large design space for the accelerator
- Sufficient flexibility in a rapid evaluation of modified target architectures
- Substantially reduce the design or optimization time while preserving a reasonable accuracy
- Proves less than 2% variation in evaluation results
- Uncalibrated CAnSO depicts 22% difference in average

Future work:
- Expanding CAnSO to support control instructions
- Considering more complicated RAC architectures