Fast Design Space Exploration Framework with an Efficient Performance Estimation Technique

School of Electrical Engineering and Computer Science
Seoul National University
Seongnam Kwon, Choonseung Lee, Sungchan Kim, Youngmin Yi, Soonhoi Ha
{ksn, cslee, ynwie, ymyi, sha}@iris.snu.ac.kr
1. Introduction
2. Proposed Design Space Exploration framework
3. Performance estimation method of SW function blocks to be used in proposed framework
4. Experimental results
5. Conclusion
Design Space Exploration

- Find an optimal architecture (high performance, low cost) which is most suitable for target application. It includes:
  1. Component mapping
  2. Communication architecture selection
  3. Performance estimation
Current Practice

- Component selection & mapping: manually
- Estimation method of entire system’s performance: (TLM) simulation
- System complexity increases: large number of architecture candidate, MPSoC, NoC?
  - Architecture selection becomes difficult
- Fast and automatic design space exploration framework is needed!
Principle of Orthogonalization[1]


Y-Chart

- Orthogonalization of behavior and architecture

Behavior specification → mapping → Performance evaluation → decision → Architecture specification
Orthogonalization of computation and communication

DSE consist of two steps:

1. Select the processing components and map the function blocks to the components: **Cosynthesis Loop**

2. Determine the optimal communication architecture: **Communication DSE loop**
Contents

1. Introduction
2. Proposed Design Space Exploration framework
3. Performance estimation method of SW function blocks to be used in proposed framework
4. Experimental results
5. Conclusion
**Cosynthesis Loop**

- Select the processing components and map the function blocks to the components
  - **Input:**
    - System behavior is modeled as a composition of function blocks
    - Block performance DB
    - Architecture platform and component library
  - **Output:**
    - Partitioned code
- Performance estimation: the sum of block performances on the mapped component
- This is presented in [6]

Communication DSE Loop

Candidates communication architecture selection
- **Input:**
  - HW/SW cosimulation-generated memory traces
  - Communication architecture library
- **Output:**
  - Optimized communication architecture

Two step design space exploration
1. Queueing analysis: pruning the design space
2. Trace-driven simulation

This will be presented at CODES+ISSS 2004
Proposed Design Space Exploration Framework

- Instruction set simulation for estimation of function blocks
- Block performance DB (name, CPU time, memory access counts)
- Behavior Specification
- Component selection & mapping
- Partitioned code
- HW/SW cosimulation to obtain memory traces
- Communication architecture selection
- Cosynthesis loop
- Global DSE loop
- Communication DSE loop
Contributions

1. A complete design space exploration framework of PeaCE is first presented
2. We propose a static performance estimation method of function blocks to be used in the proposed framework
1. Introduction
2. Proposed Design Space Exploration framework
3. Performance estimation method of SW function blocks to be used in proposed framework
4. Experimental results
5. Conclusion
1. **Objective**
   - Estimate performance of SW block being used in DSE loop
   - Soft-real time application is the target of this work: Multimedia example

2. **Performance variation factors of SW block**
   1. **Architecture features**
      - Ex. Memory access cycle, bus protocol, and so on
   2. **Compiler, Compiler version, Compile option**
   3. **Input data & Interference between blocks**
1. Architecture features

- Two types of Performance information
  - Architecture-independent: CPU time
  - Architecture-dependent: Memory access time
- Performance information: (CPU time, Memory access counts)

Separate architecture-dependent performance information
2. Compiler, Compiler version, Compile option

- Use different database entry
- DB primary key: (Name, parameter value, target processor, compiler op,..)

*Performance equation of entire system*

\[
P_{est} = \sum_{B_k \text{ on CP}} n(k) \times \left\{ P(k, i) + m(k) \times n_m + c(k, l) \times n_c \right\}
\]

- \(B_k\): function block executed on critical path
- \(P(k, i)\): the estimated CPU time on the mapped component \(P_i\)
- \(m(k)\): the memory access counts
- \(c(k, l)\): the communication requirements to the next block \(B_l\)
- \(n(k)\): the number of invocations of block \(B_k\)
- \(n_m, n_c\): memory access overhead and the channel communication overhead
3. Input data & Interference between blocks

Application:

A → B → C

Simulating each block:

A → Performance Information
B → Performance Information
C → Performance Information

Simulating the entire application:

A → B → C → Performance Information

What is a good test vector?

Test bench and analysis environment should be built for each function block.

Proposed Performance Estimation

Common Performance Estimation

Break Point: Code Augmentation
**Solutions**

**Code augmentation**

- Compiler may move or eliminate augmented code:
  - some technique is needed

```c
... func_A();
func_B();
...

extern int Start_func();
extern void End_func();
...
if (Start_func()==true) func_A();
End_func();
if (Start_func()==true) func_B();
End_func();
...
```
1. Introduction
2. Proposed Design Space Exploration framework
3. Performance estimation method of SW function blocks to be used in proposed framework
4. Experimental results
5. Conclusion
Experimental environment

- System: H.263 Encoder
- Target processor: ARM720T
- ISS: ARMulator
- Input: QCIF(176*144) format 50 frames
  - Input file: FOREMAN
- Cache miss penalty:
  - Sequential Read: 2 Cycles
  - Nonsequential Read: 5 Cycles
- Write through cache/Write buffer
  ➔ assume that write penalty is zero
## CPU execution time

<table>
<thead>
<tr>
<th>Block name</th>
<th>Count</th>
<th>WCET</th>
<th>Total Exec. time</th>
<th>WCET/Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable length coding</td>
<td>50</td>
<td>485795</td>
<td>21904583</td>
<td>1.11</td>
</tr>
<tr>
<td>Motion Compensation</td>
<td>50</td>
<td>2608646</td>
<td>48842573</td>
<td>2.67</td>
</tr>
<tr>
<td>IDCT</td>
<td>29700</td>
<td>1947</td>
<td>13122365</td>
<td>4.41</td>
</tr>
<tr>
<td>Quantization</td>
<td>29700</td>
<td>1366</td>
<td>35738978</td>
<td>1.14</td>
</tr>
<tr>
<td>DCT</td>
<td>29700</td>
<td>2841</td>
<td>69555894</td>
<td>1.21</td>
</tr>
<tr>
<td>Motion Estimation</td>
<td>50</td>
<td>24777951</td>
<td>986623904</td>
<td>1.26</td>
</tr>
<tr>
<td>Etc.</td>
<td></td>
<td></td>
<td>68412069</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>50</td>
<td></td>
<td>1244200366</td>
<td></td>
</tr>
</tbody>
</table>

 Executes time of entire system: 1,244,538,245

 error: 0.05%
• **What is a good performance number?**
How deadline miss can be reduced?

With only one decoded image buffer….
- I type: small computation ➔ store early-decoded image
- P type: large computation ➔ using saved time in I type

If worst case block times of FOREMAN is used as performance value…
- Over-estimated amounts: 14% ~ 104%
- Some value between avg. time and worst time should be selected!
DivX Player Example

- Behavior specification
  - Top specification
    - AviReader
    - H263FRDivx
    - MADStream

Top specification

Inner blocks
DivX Player Example

- Instruction set simulation for estimation of blocks

![Performance Result]
DivX Player Example

- Component selection & mapping
- Architecture selection
- Candidate PEs Tables
- The Gantt Chart
DivX Player Example

- Memory traces generation using HW/SW cosimulation

**HW (IDCT):** ModelSim

**SW: ARMulator**

**Memory Trace**
Communication architecture selection

Candidate comm. architecture generation & evaluation

Result: optimized comm. architecture

Performance of candidate architectures

Intermediate graph

Final graph
1. Introduction
2. Proposed Design Space Exploration framework
3. Performance estimation method of SW function blocks to be used in proposed framework
4. Experimental results
5. Conclusion
Conclusion

🔹 Proposed DSE framework: two loops
  - Cosynthesis loop
  - Communication DSE loop

🔹 Simulation-based SW performance estimation method. This is adaptable to consider varying:
  - Architecture features
  - Compiler, Compiler option
  - Input data & Influence between blocks
Thank You!!