Fidelity of native-based performance models for Design Space Exploration

Fernando Herrera and Eugenio Villar
University of Cantabria
Outline

- DSE
- Fidelity in DSE
- Estimation Technologies: Native Simulation
- Fidelity in Native Simulation
- VIPPE
- Conclusion
Design Space Exploration

- Crucial Activity in Electronic System-Level Design!!

- Algorithms
- Architectural Mappings
- Number of Processors
- Processor Frequencies
- Task Periods
- Bus Width
- Cache Sizes
- Processor Types

16-Mar-16
Fernando Herrera / University of Cantabria
# solutions:

- 5 applications, 2 algorithms on app1, 2 algorithms on app2, 4 period combinations, 1-4 processors, 3 processor types, 3 frequencies, all possible application mappings
- Design Space: $2 \times 2 \times 4 \times 3 \times (3^4) \times (3^5) = 944784$
- ~1 million solutions
- 5 days analysis:
  $\rightarrow$ 200,000 solutions/day $\rightarrow$ >23 alternatives/sec
DSE requirements

- Exploration speed
- “Sufficient” Accuracy: Fidelity in DSE context
Report from the Fidelity Implementation Study Group

D.C Gross (Boeing)

“The degree to which a model or simulation reproduces the state and behavior of a real world object or the perception of a real world object, feature, condition, or chosen standard in a measurable or perceivable manner; a measure of the realism of a model or simulation; faithfulness.

Fidelity should generally be described with respect to the measures, standards or perceptions used in assessing or stating it. See accuracy, sensitivity, precision, resolution, repeatability, model/simulation validation.”
DSE requirements (I): Exploration Speed

**Assessment**
- Analytical vs Simulation
- Simulation Speed

**Search algorithm.** Decide:
- Next point to be explored
- End of exploration

- **Assessment transition**
  - Data Exchange with exploration algorithm
  - Configure the model to be explored
### Simulation-based Performance Assessment Technologies

<table>
<thead>
<tr>
<th>Method</th>
<th>Speed Up</th>
<th>Accuracy Degradation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional Execution</td>
<td>100000</td>
<td></td>
</tr>
<tr>
<td>Timed native co-simulation</td>
<td>10000</td>
<td>?</td>
</tr>
<tr>
<td>Timed binary translation (QUEMU+, OVP+)</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>ISS (instructions)</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>ISS with microarch. (cycle accurate)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Pin&amp;Cycle Accurate (HDL simulator)</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

• Simulation-based Performance Assessment Technologies
Sufficient Accuracy

- Assessed
- Actual
DSE requirements (II)

- Insufficient Accuracy

![Graph showing energy and response time with assessed and actual markers.](image-url)
DSE requirements (II)

- Insufficient Accuracy

![Diagram showing response time and energy with assessed and actual marked with circles and stars.](image-url)
**Fidelity in DSE (as accuracy)**

- Depends on the amount of solutions and the constellation amplitude (= density)
- One (Our) Criteria: <10% vs instructions accurate technologies

<table>
<thead>
<tr>
<th>Method</th>
<th>Speed Up</th>
<th>Accuracy Degradation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional Execution</td>
<td>100000</td>
<td></td>
</tr>
<tr>
<td>Timed native co-simulation</td>
<td>10000</td>
<td></td>
</tr>
<tr>
<td>Timed binary translation (QUEMU+, OVP+)</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>ISS (instructions)</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>ISS with microarch. (cycle accurate)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Pin&amp;Cycle Accurate (HDL simulator)</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
- Instrument the source code
  - For performance cost accounting
  - Generating memory access traces
- Simulation speed: $\sim (\text{native program speed}) \times \alpha \ (\alpha < 2)$
- Overhead of accounting and trace generation

Native Simulation
In using the **same source code**

In the **assessment of performance**, in the placing of events in the simulated time, considering the effects and penalties of:
- SDK effect (compiler)
- RTOS (if any)
- hardware architecture

Not (or not necessarily) in:
- Placement of events on the emulated time axis (time decoupling)
- the behaviour of sw platform and details relevant to SW development, e.g., in detecting every type of illegal memory access in the target platform.
Native Simulation: Fidelity Challenges

- **Input code:**
  - custom parsing & reconstruction vs LLVM
  - Object/Legacy Libraries [Henkel, CODES+ISSS’12]

- **SDK (Compiler) Complexity**
  - Optimizations [Rosenstiel, DAC11] [Henkel, DATE’12]
  - Target dependent headers

- **Platform Complexity:**
  - Target Processor
    - Word width, ISA [Posadas, 2006]
    - Internal architecture [Gerslauer, CODES’13][Rosenstiel, ESWEEK’14]
  - RTOS [Posadas, 2006][Gerlauer, DATE’11, ACM TECS 2014]
  - Cache model [Posadas, ASP-DAC’11][Rosenstiel, DATE 12] [Gerslauer, ESLyn’13]
  - Memory hierarchies [Posadas, SIES’10]
  - Platform architecture [Posadas, 2009][Rosenstiel, DATE’11]
  - Cache coherence
Parallel Native Simulation

- P: # host cores, M: # target cores, T: # application(s) threads

- Exploit the underlying parallelism of the host machine simulating each processing element in one core
  - Potential speed up: $\min(P,M,T)$
  - $>1$ as long as $M>1 \land P>1 \land T>1$
  - Also in other technologies

- Thread-based parallelization (VIPPE)
  - Potential speed up: $\min(P,T)$
  - $>1$ as long as $P>1 \land T>1$, even if $M=1$
  - Breaks the $M$ speed-up wall, as gain can be $P$ if $P > M \land T>M$

- Challenges cache estimation
• Parallel native simulation
• Compilation branches: LLVM and source compiler
  • Optimizations
• Platform modelling
  • Processor modelling: armv7, sparc, LEON, microblaze
  • I&D Cache model
  • Heterogeneous Multi-core Platform Architecture
• Connection to SystemC
• Performance reports: time, energy, power
• GUIs
Conclusions

- Native Simulation: key simulation technology for simulation-based Design Space Exploration of complex embedded systems

- Fidelity required

- Fidelity requires information regarding the target processor, the cross-development kit, and the target architecture.

- VIPPE: last generation of native simulation tool with Fidelity for Design Space exploration

- More soon in http://vippe.teisa.unican.es