Efficient Event-driven Simulation of Parallel Processor Architectures

Alexey Kupriyanov, Dmitrij Kissler, Frank Hannig, Jürgen Teich
Hardware-Software-Co-Design
University of Erlangen-Nuremberg, Germany
{kupriyanov, kissler, hannig, teich}@cs.fau.de
Outline

- Motivation
- Generic Processor Array Architecture Model
- Event-driven IS Simulation
  - Data Dependence Graph
  - Simulation Activity Period
- Example (Simulation Activity Schedule)
- Minimization of simulation events
- Methodology Restrictions
- Design and Implementation
  - Weakly Programmable Processor Array Architecture
  - Design Framework
  - Experimental Results and Case Study
- Conclusions
Motivation

- Steadily increasing complexity of processor architectures
- 70-80% of the development cycle is spent in validation
- Fast estimation and simulation methods are essential to explore enormous design spaces of possible architecture/compiler co-designs
Generic Processor Architecture Model

- Multiple Functional Units
- VLIW Memory and PC
- Input Shift Registers or FIFOs
- General Purpose Registers
- Output Registers
Generic Processor Array Model
Event-driven IS Simulation

- The challenge is to simulate only in those regions of the processor array currently performing a computation
  - Examples: PEs responsible for array reconfiguration; image processing algorithms, etc.
- The simulation of a certain processing element is performed only as soon as a simulation event occurs
- A simulation event is a change of the contents in at least one of the input or output registers of a PE
- Cycle-based instruction set simulation is used between events in order to achieve higher simulation speeds as compared to RTL
  - Simplification of the instruction sequencer and the instruction decoder simulation subroutines
  - Acceptable level of granularity for a core simulation: provides detailed functional information such as register values as well as program execution times and other timing information
Data Dependence Graph

VLIW-Program $V$

$$V = \begin{pmatrix} v_{00} & v_{01} & v_{02} \\ v_{10} & v_{11} & v_{12} \\ v_{20} & v_{21} & v_{22} \end{pmatrix}$$

$DDG_1$

$$DDG = (H, E), \ E \subseteq H \times H$$

$$H = H_{in} \cup H_c \cup H_{op} \cup H_{out}$$

$e = (h_1, h_2) \in E$
At the IS level, the PE must be simulated at least during a certain period of several cycles after an event arrives in order to calculate the result and possibly to produce an event for other connected PEs.

**Activity-period** $d(P)$

of a processor element $P$ defines a minimal number of simulation cycles needed to perform the computation after a simulation event arrives and to store the results which can produce the new events for other processor elements.
Example (Simulation Activity Schedule)

Simulation activity

Total PE simulation cycles: 30
Effective PE simulation cycles: 18
Minimization of simulation events (1)

- Not every change in the input registers of the PE is a simulation event.
- Not all input data stored in the input shift registers ID are processed by P.
- Only those input values placed in the last register of the chains effect the output results of P and only at those certain cycles when the PC is pointing to the instruction accessing one of these shift registers.
- Let \( p \) be a pattern of simulation events for a particular input of P.
- The vector \( M_j = (m_0, m_1, \ldots, m_s) \) denotes the event-mask of processor input \( I_j, j = [0, m-1] \).

\[
m_i \left\{ \begin{array}{ll} 
1, & \text{if } \exists e(h_1, h_2): h_1 \in H_{in}, \quad i = [0, s] \\
0, & \text{otherwise}, \quad i = [0, s] 
\end{array} \right.
\]

\[
p_j[i] = M_j[k], \quad k = (i + s + 1 + a_j - l_j \mod(s + 1)) \mod(s + 1), \quad i = [0, C - 1]
\]
Minimization of simulation events (2)

Minimize the number of arriving simulation events

<table>
<thead>
<tr>
<th>Cycle #</th>
<th>PC</th>
<th>ID0</th>
<th>RD0</th>
<th>RD1</th>
<th>OD0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(initial)</td>
<td>0</td>
<td>(0,0,0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(0,0,0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(0,0,0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>(2,0,0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>(2,2,0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>(3,2,2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>(2,3,2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>(2,2,3)</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>(2,2,2)</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>(2,2,2)</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>(0,2,2)</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>(1,0,2)</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>(1,1,0)</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>(0,1,1)</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>(0,0,1)</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>(0,0,0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>(0,0,0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>(0,0,0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>(0,0,0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>(0,0,0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Speed up can only be exploited for PEs…

1. no cycles in DDG
2. no conditional branches
3. There should exist at least one ID and one OD

VLIW-Program $V$

$$V = \begin{pmatrix} v_{00} & v_{01} & v_{02} \\ v_{10} & v_{11} & v_{12} \\ v_{20} & v_{21} & v_{22} \end{pmatrix}$$

...with

- no cycles in DDG
- no conditional branches
- There should exist at least one ID and one OD
Outline

- Motivation
- Generic Processor Array Architecture Model
- Event-driven IS Simulation
  - Data Dependence Graph
  - Simulation Activity Period
- Example (Simulation Activity Schedule)
- Minimization of simulation events
- Methodology Restrictions
- Design and Implementation
  - Weakly Programmable Processor Array Architecture
  - Design Framework
  - Experimental Results and Case Study
- Conclusions
Design and Implementation: WPPA
WPPA Design Framework

- WPPA Simulator

  Application Program (VLIW Assembler)

  WPPA Framework

  MAML Architecture Specification

  Architecture Description

  Simulated HDL

  Generators

  Debugging and Simulation Environment

  Automated Simulator Generation and Compilation

  VHDL Backend

  Synthesizable HDL Generators

  Further Refinement

  Instruction Set

  Simulator Generator

  Test Bench (Stimuli Assignment)

  Simulation Core (C++ Code)

  Simulator Compilation and Execution (Traces Generation)

C++
Experimental Results and Case Study

- **WPPA1**
  - Edge detection algorithm
  - In overall 3 PEs, 21 FUs, 44 registers with input shift regs

- **WPPA2**
  - Edge detection algorithm
  - In overall 6 PEs, 18 FUs, 29 registers

- **WPPA3**
  - FIR filter algorithm
  - In overall 4 PEs, 12 FUs, 80 registers

<table>
<thead>
<tr>
<th>Simulation speed</th>
<th>WPPA1 (3PEs)</th>
<th>WPPA2 (6PEs)</th>
<th>WPPA3 (4PEs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPS</td>
<td>14,4 MIPS</td>
<td>10,7 MIPS</td>
<td>10,4 MIPS</td>
</tr>
<tr>
<td>CPS</td>
<td>686k CPS</td>
<td>592k CPS</td>
<td>864k CPS</td>
</tr>
</tbody>
</table>
Experimental Results and Case Study

- Functional simulation of generated VHDL code
  - Modelsim simulation time of 200k cycles: 359 sec
  - Simulation time using our approach: 0.235 sec (approx. a factor of 1500 compared to RTL Modelsim simulation)

- Generated synthesizable VHDL models
  - Xilinx Virtex-II Pro xc2vp30 FPGA
  - Overall performance for 2 x 3 array (WPPA2) is 2.3 GOPS
  - Processor reconfiguration time: <3 μs
  - Configuration stream size: 200 Byte
  - Operating clock frequency 125 MHz
  - FPGA occupancy: 58%
Conclusions

- Generic Processor Array Architecture Model
- Definition of the simulation model of each PE by a DDG
- Event-driven simulation methodology
- Optimizations
  - Introduction of simulation-event-patterns
  - Minimization of simulation events
  - Determination of activity-periods
- Modeling environment **WPPA Framework**
  - Graphical or XML-based architecture entry
  - Automatic generation of a C++ simulation model
- Case study
  - Simulation of different WPPAs implementing edge detection and FIR algorithms
  - Comparison to existing commercial simulators
Questions?

Efficient Event-driven Simulation of Parallel Processor Architectures

Alexey Kupriyanov
Hardware-Software-Co-Design Chair
University of Erlangen-Nuremberg
Am Weichselgarten 3
91058 Erlangen
Germany

Phone: + 49 9131 85-25156
Fax: + 49 9131 85-25149
Email: kupriyanov@cs.fau.de
URL: http://www12.informatik.uni-erlangen.de