A Flexible Hybrid Simulation Platform Targeting Multiple Configurable Processors SoC

*Hao Shen  Frédéric Pétrot
System Level Synthesis Group, TIMA Laboratory, CNRS/Grenoble INP/UJF
43, Avenue Félix Viallet, 38031, Grenoble, France
Outline

- Introduction of Configurable Processor
- Hybrid Platform Overview
- Hybrid Platform Realization Details
- Experimental Results
- Conclusion and Future Works
Outline

Introduction of Configurable Processor

Hybrid Platform Overview

Hybrid Platform Realization Details

Experimental Results

Conclusion and Future Works
Configurable Processors

![Diagram showing the trade-off between Performance/Gate and Power Efficiency and Flexibility and Time-to-Market for Configurable Processors, GPP, DSP, Configurable Processor, and ASIC. The diagram illustrates that Configurable Processors offer a balance between these two extremes.]

**Tool chain supports**

- Automatic generation of dedicated compilers for extended instruction sets
- Automatic generation of simulation and profiling environments
- RTL code available for synthesis
MPSoC Architecture

How to program an MPSoC with configurable processors?

- Different processors with different configurations
- The operating system and device driver porting difficulty
Motivation of the Hybrid Simulation

Modifications during the design space exploration
- MPSoc architecture modification
- Processor configuration modification
- Hardware module (I/O device) realization and interfaces modification

Hardware dependent Software (HdS) requirements
- Flexibility (as less assembly code as possible & no hard coded I/O address)
- Very easy to be integrated with the SystemC TLM modules

Solution: Hybrid Simulation
- Realize the HAL and Drivers with SystemC modules
Outline

- Introduction of Configurable Processor
- Hybrid Platform Overview
- Hybrid Platform Realization Details
- Experimental Results
- Conclusion and Future Works
What Is Hybrid Simulation Platform?

Semi-hosting functional call
- A specific instruction that is only valid within the ISS
- The ISS calls a host hooking function (handler)
- Used by lots of simulators such as the SPIM MIPS simulator

Realize most of HAL APIs with SystemC modules
- Avoid processor related assembly codes
- Avoid Hds porting and modification works

Integrate I/O devices with their drivers
- Avoid inconsistent register access
- Avoid the use of I/O addresses
- Compatible with SystemC TLM
Hybrid Simulation MPSoC Platform

- Applications + OS
- HAL APIs
- Hardware Abstraction Layer (Target Assembly Codes)
- Instruction Set Simulators
  - ICache
  - DCache
- Memory
- Periph

- Applications + OS
- HAL APIs
- Hybrid Instruction Set Simulators
  - Context Switch
  - Boot Code
  - I/O Access
  - Lock
  - Interrupt Handler

- Memory
- Periph
- TLM SystemC
Hybrid Simulation Platform Details

Checking and modifying processor statuses

- The ISS should provide the register get and set functions
  - `get_registers` and `set_registers`
- The simulation platform should provide the memory access functions
  - `peek_physical` and `poke_physical`

Context switch

- **APIs**: `HAL_SAVE_REGISTERS`, `HAL_CONTEXT_INIT`, `HAL_CONTEXT_SWITCH` and `HAL_CONTEXT_LOAD`
- Assembly code => SystemC realization

I/O devices

- Register access => TLM functional call
HAL APIs Realization

Applications and Operating System

```c
void scheduler_commute(pthread_t self){
    ...
    HAL_CONTEXT_SWITCH(self->reg, thread->reg);
}
```

HAL APIs Wrapper

```c
#define HAL_CONTEXT_SWITCH(store_addr, load_addr) xt_iss_simcall(HOST_HAL_CONTEXT_SWITCH, 0, store, load, 0, 0)
```

SystemC HAL APIs Realization

```c
int host_hal_context_switch(int store_addr, int load_addr){
    map<string, sc_unsigned> current=core.get_all_registers();
    saveRegWindow(context_map[store_addr], current);
    loadRegWindow(context_map[load_addr], current);
    core.set_all_registers(current);
}
```
Driver Hybrid Simulation

Applications

HAL+Driver

ReadB(DMA_STATUS ...);
WriteB(DMA_ADDR ...);

Instruction Set Simulators

ICache  DCache

DMA Device with SystemC

0x00  Control Registers
0x08  Address Registers
......
0x40  Status Registers

Applications

HAL+Driver

DMA_Reset();
DMA_Copy(from, to...);

Hybrid Instruction Set Simulators

DMA Device with TLM

DMA_Reset()
DMA_Copy()
......
DMA_Check_Status()
Outline

Introduction of Configurable Processor

Hybrid Platform Overview

Hybrid Platform Realization Details

Experimental Results

Conclusion and Future Works
Motion-JPEG Decoder Example

Motion-JPEG Application
- A multimedia format with separately compressed JPEG images

Mutek operating system with SMP support
- Open source and lightweight
- POSIX Thread APIs

MPSoC architecture modeling
- Xtensa LX2 configurable processor
- Xtensa SystemC (XTSC) platform
Accuracy of Hybrid Platform

Estimation accuracy of the hybrid simulation platform

<table>
<thead>
<tr>
<th></th>
<th>Hybrid Platform</th>
<th>Traditional ISS Platform</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended Instructions</td>
<td>1.71</td>
<td>1.53</td>
<td>10%</td>
</tr>
<tr>
<td>Acceleration Ratio</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DMA Transfer Example

Memory copy using the DMA device

- Use `peek_physical` and `poke_physical` for memory access
- Timing annotation based on transfer size

```cpp
int ta_dma::dma_ss_copy ( xtsc_core &core,
int from, int to, int size)
{
    // Copy original data to DMA buffer.
    core.peek_physical(from, size, buff);
    // Consume time for this copy operation
    consume(size/4*time_per_word);
    // Copy DMA buffer to Target Memory
    core.poke_physical(to, size, buff);
    return size;
}
```
Development Effort

**Hybrid simulation can save development effort**

- Use *Source Lines Of Code (SLOC)* to evaluate design effort
- Useful for both HdS and driver development
- Simplify the I/O device modeling

**SLOC for both hybrid and traditional simulation platform**

<table>
<thead>
<tr>
<th>Module Name</th>
<th>SLOC of the hybrid platform</th>
<th>SLOC of the traditional platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA</td>
<td>79</td>
<td>267</td>
</tr>
<tr>
<td>context</td>
<td>202</td>
<td>257 (206 lines of asm code)</td>
</tr>
<tr>
<td>stack</td>
<td>48</td>
<td>14 (6 lines of asm code)</td>
</tr>
</tbody>
</table>
Outline

Introduction of Configurable Processor

Hybrid Platform Overview

Hybrid Platform Realization Details

Experimental Results

Conclusion and Future Works
Conclusion and Future Works

A hybrid MPSoC simulation platform

- Simplify the Hds adaptation for processor and architecture modifications
- Simplify the I/O device modeling and the driver adaptation
- Specially useful for early design stages

Future works

- Hds adaptation for Linux OS
- Open source hybrid ISS: such as QEMU
Questions & Answers

Hao.Shen@imag.fr