Virtual Prototypes and Open Source Hardware Design in Research and Education

Sallar Ahmadi-Pour (saahm)
sallar@uni-bremen.de
University of Bremen, Group of Computer Architecture

Our AGRA Github
https://github.com/agra-uni-bremen/

All our RISC-V related works
https://agra.informatik.uni-bremen.de/projects/risc-v/
Group of Computer Architecture @ Uni Bremen

- **Circuits and System Descriptions**
  - Specifications
  - Hardware description languages
  - System level description languages (e.g. SystemC, UML, SysML, etc.)
  - Robustness

- **Algorithms and Data Structures**
  - Boolean satisfiability (SAT)
  - Decision diagrams
  - Processing large amounts of data

- **Verification**
  - Model Checking
  - Equivalence checking
  - Debugging

- **Test Pattern Generation**
  - Automatic test pattern generation
  - Design for testability

- **Emerging Technologies**
  - Contactless Side-Channel Attacks
  - Approximate Computing
  - Microfluidic Biochips
  - In-Memory Computing
  - Reversible Logic
Motivation

• System-on-Chips (SoCs) are big and complex
  You wouldn’t want to write thousands of lines of code and then go on towards an endless bug hunt, would you?

• Complex interleaving of Software (SW) and Hardware (HW)

• Errors can happen at all layers of the System
  ➔ Finding and fixing: Early = Cheap, Late = Expensive

• Traditional flow: design and produce HW first, SW later
  • Problems with HW will be fixed by SW or new chip
  • SW can’t be developed aligned with HW and early
Virtual Prototyping

• Utilize abstractions to model aspects of system with different details
  • “What is actually needed to run SW faithfully?”
  • Don’t model every wire, become faster with higher abstractions
• Explore the design space of structure
• Electronic System Level Design Flow
• Executable Specification
Virtual Prototyping

Behavioral
- Systems
- Algorithms
- Register-transfer
- Logic
- Transfer functions

Structural
- CPU, Memory
- Subsystems, Buses
- ALUs, Registers
- Gates, Flipflops
- Transistors
- Polygons
- Cells
- Macros, Floor Plans
- Clusters
- Chips, Physical Partitions

Geometry

Traditional
- Specification
- Hardware
- Software

ESL with VP
- Specification
- Virtual Prototype
- Hardware
- Software

Faster Time to Market
Virtual Prototyping

Behavioral
- Systems
- Algorithms
- Register-transfer
- Logic
- Transfer functions

Structural
- HDL
- CPU, Memory
- Subsystems, Buses
- ALUs, Registers
- Gates, Flipflops
- Transistors
- Polygons
- Cells
- Macros, Floor Plans
- Clusters
- Chips, Physical Partitions

Geometry

Traditional
- Specification
- Hardware
- Software

ESL with VP
- Virtual Prototype
- Specification
- Hardware

Faster Time to Market
Virtual Prototyping

Behavioral
- Systems
- Algorithms
- Register-transfer

Structural
- HDL
- SystemC (TLM, RTL, AMS)
- CPU, Memory
- Subsystems, Buses
- ALUs, Registers
- Gates, Flipflops
- Transistors
- Polygons
- Cells
- Macros, Floor Plans
- Clusters
- Chips, Physical Partitions

Geometry

Traditional
- Specification
- Hardware
- Software

ESL with VP
- Specification
- Virtual Prototype
- Hardware
- Faster Time to Market
Virtual Prototyping

Behavioral
- Systems
- Algorithms
- Register-transfer
- Logic
- Transfer functions
- Transistors
- Polygons
- Cells
- Macros, Floor Plans
- Clusters
- Chips, Physical Partitions

Structural
- CPU, Memory
- Subsystems, Buses
- ALUs, Registers
- Gates, Flipflops
- HDL
- TLM, RTL, AMS

Geometry

Traditional
- Specification
- Hardware
- Software

ESL with VP
- Specification
- Virtual Prototype
- Hardware
- Software

Faster Time to Market
Open Source RISC-V Virtual Prototype

- RV32GC, RV64GC support (i.e. RV32IMAFDC, RV64IMAFDC)
- Implemented in SystemC TLM-2.0
- SW debug capabilities (GDB RSP interface) with for example Eclipse
- FreeRTOS, RIOT OS, Zephyr OS, Linux support
- Peripherals, e.g. Flash ctrl, UART, SPI
- SiFive HiFive Board configuration available
- Many more features

github.com/agra-uni-bremen/riscv-vp
Virtual Breadboard for the Virtual Prototype

- Extends RISC-V VP
- Interactive simulation with off-chip components [Buttons, Switches, LEDs, 7-Segment, ...], Modeled in C++ or Lua
- Mouse, keyboard and visual I/O in graphical user interface
- Build drivers for devices you don’t have yet
- Education on embedded systems (it doesn’t break physically)

https://github.com/agra-uni-bremen/virtual-breadboard
Virtual Breadboard for the Virtual Prototype

- Extends RISC-V VP
- Interactive simulation with off-chip components [Buttons, Switches, LEDs, 7-Segment, …], Modeled in C++ or Lua
- Mouse, keyboard and visual I/O in graphical user interface
- Build drivers for devices you don’t have yet
- Education on embedded systems (it doesn’t break physically)

https://github.com/agra-uni-bremen/virtual-breadboard
Virtual Breadboard for the Virtual Prototype

- Extends RISC-V VP
- Interactive simulation with off-chip components [Buttons, Switches, LEDs, 7-Segment, ...], Modeled in C++ or Lua
- Mouse, keyboard and visual I/O in graphical user interface
- Build drivers for devices you don’t have yet
- Education on embedded systems (it doesn’t break physically)

https://github.com/agra-uni-bremen/virtual-breadboard
Virtual Prototype in the Loop – VP and real HW

https://github.com/agra-uni-bremen/virtual-bus
Virtual Prototype in the Loop – VP and real HW

<table>
<thead>
<tr>
<th>RISC-V VP</th>
<th>FPGA Chip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulated on host computer</td>
<td>Physical peripherals</td>
</tr>
</tbody>
</table>

https://github.com/agra-uni-bremen/virtual-bus
Virtual Prototype in the Loop – VP and real HW

https://github.com/agra-uni-bremen/virtual-bus
MicroRV32 – From VP to RTL

- Use VP to model HW Platform, then utilize as reference for RTL implementation
- Utilize SpinalHDL for OSS based HW development
  - Generate Verilog, VHDL
  - Simulate
  - Synthesize for FPGA
- RV32IMC (configurable), for FPGAs and ASIC
- Binary compatible VP target for fast SW development
- Multi-cycle data path as robust and lightweight design
MicroRV32 – From VP to RTL

- MicroRV32 SoC runs around 20 MHz (Core itself reaches around 43 MHz)
- On HX8K FPGA requires 26% (RV32I) - 44% (RV32IMC) of area
- Built with OSS tools, made open source
- Also put into ASIC (commercial 28 nm node), waiting for packaging

github.com/agra-uni-bremen/microrv32
MicroRV32 – From VP to RTL

- MicroRV32 SoC runs around 20 MHz (Core itself reaches around 43 MHz)
- On HX8K FPGA requires 26% (RV32I) - 44% (RV32IMC) of area
- Built with OSS tools, made open source
- Also put into ASIC (commercial 28 nm node), waiting for packaging

github.com/agra-uni-bremen/microrv32
Symbolic Execution

**Motivation:** Automatically explore all paths inside a piece of software

- Instead of concrete values for variables use *symbolic values*

- Symbolic values represent a set of possible concrete values (i.e. path conditions)
  - Represented as formulas, e.g. $a > 5 & a < 10$
  - During symbolic execution the values are constrained more and more
Symbolic Execution

**Motivation:** Automatically explore all paths inside a piece of software

- Instead of concrete values for variables use *symbolic values*
- Symbolic values represent a set of possible concrete values (i.e. path conditions)
  - Represented as formulas, e.g. $a > 5 \& a < 10$
  - During symbolic execution the values are constrained more and more

```c
void myfunc( int a ) {
    if ( a > 8 )
    // ...
    else
    // ...
    if ( a < 5 )
    // ...
    else
    // ...
}
```
Symbolic Execution

Motivation: Automatically explore all paths inside a piece of software

- Instead of concrete values for variables use **symbolic values**
- Symbolic values represent a set of possible concrete values (i.e. path conditions)
  - Represented as formulas, e.g. \( a > 5 \) & \( a < 10 \)
  - During symbolic execution the values are constrained more and more
Symbolic Execution

**Motivation:** Automatically explore all paths inside a piece of software

- Instead of concrete values for variables use *symbolic values*
- Symbolic values represent a set of possible concrete values (i.e. path conditions)
  - Represented as formulas, e.g. $a > 5 \& a < 10$
  - During symbolic execution the values are constrained more and more
Symbolic Execution

**Motivation:** Automatically explore all paths inside a piece of software

- Instead of concrete values for variables, use *symbolic values*
- Symbolic values represent a set of possible concrete values (i.e. path conditions)
  - Represented as formulas, e.g. \( a > 5 \) \& \( a < 10 \)
  - During symbolic execution the values are constrained more and more
SymEx-VP – Test and Verify Embedded Software

• Apply idea from symbolic execution to embedded software
• Combine with concrete inputs (i.e. concolic testing)
• Build upon RISC-V VP to faithfully simulate the hardware interaction
• **Results:** Discovered various bugs in RIOT OS modules

github.com/agra-uni-bremen/symex-vp
SymEx-VP – Test and Verify Embedded Software

- Visualize the traces generated with SymEx-VP in 3D
- Gain more insights and better understanding of SymEx-VP results
- Explore and study multiple traces
- Information is encoded in color, shape, orientation, etc.

github.com/agra-uni-bremen/symex-3d
Conclusion

• VPs not only interesting for industry, but also research and education
• Build *virtually* anything, no need (yet) for real HW (and its fast)
• Enable powerful methods of SW development in HW design
• Accessible for anyone
  • Extensible (merge requests always welcome)
  • No need for hardware
  • Verify what you build early on
Virtual Prototypes and Open Source Hardware Design in Research and Education

Sallar Ahmadi-Pour (saahm)
sallar@uni-bremen.de
University of Bremen, Group of Computer Architecture

Our AGRA Github
https://github.com/agra-uni-bremen/

All our RISC-V related works
https://agra.informatik.uni-bremen.de/projects/risc-v/