Extensible and Configurable RISC-V based Virtual Prototype

Vladimir Herdt[1]
Daniel Große[1,2]
Hoang M. Le[1]
Rolf Drechsler[1,2]

[1]University of Bremen & [2]DFKI Bremen, Germany
vherdt@informatik.uni-bremen.de

CONFIRM
16ES0565

edaclusterforschung

SyDe
graduate school, funded by
German Excellence Initiative

2018-09-17

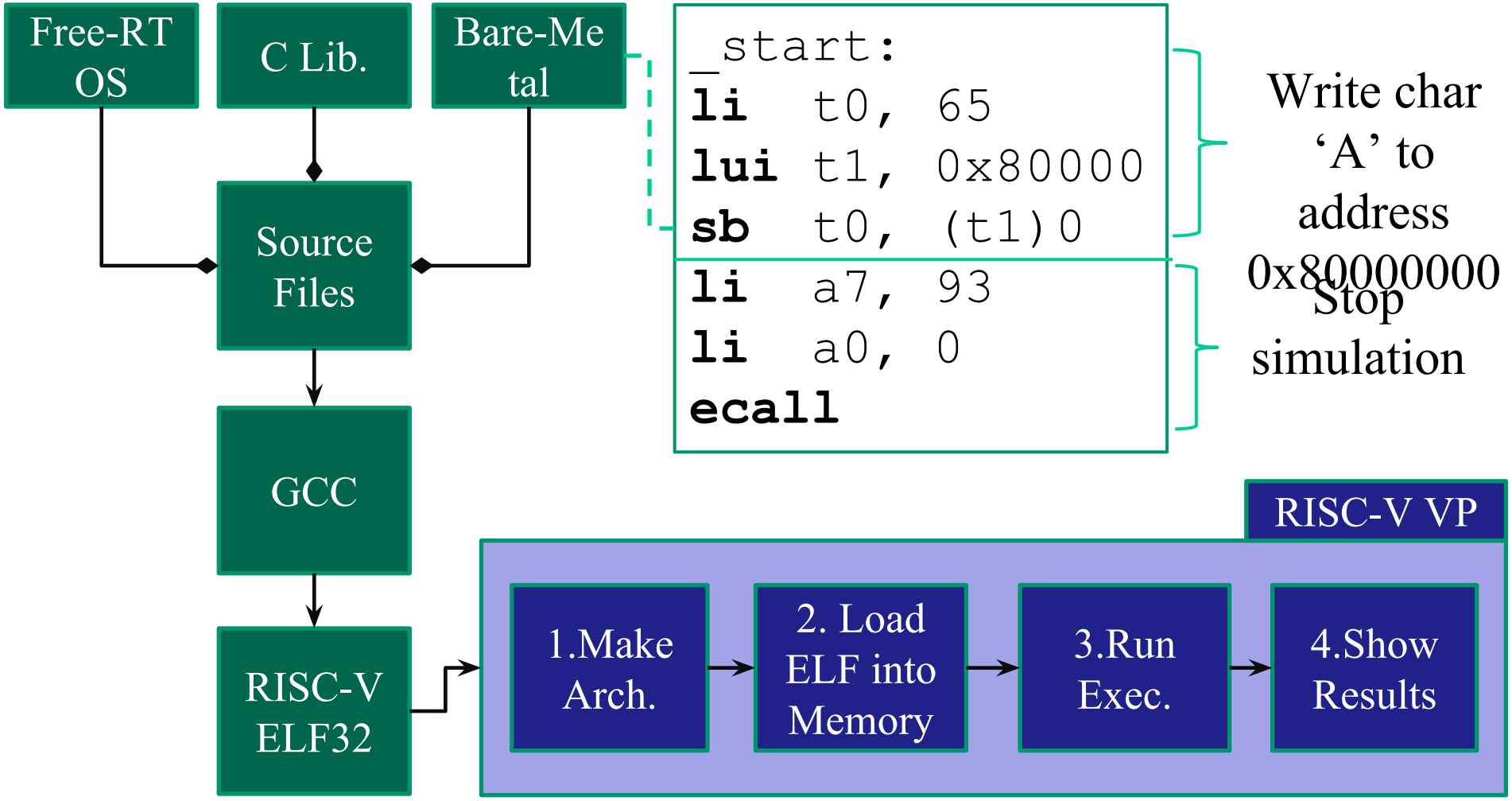
VP Overview (1)

- RV32IM(A) + machine mode CSRs
- Implemented in SystemC/C++
  - TLM-2.0 compliant
  - approx. 3000 LOC (w/o comments, blanks)
- **Open Source**
  - Github links to our RISC-V VP projects on http://www.systemc-verification.org/riscv-vp
  - MIT license
- Overview paper at FDL 2018
  http://www.informatik.uni-bremen.de/agra/doc/konf/2018FDL_RISCV_VP.pdf

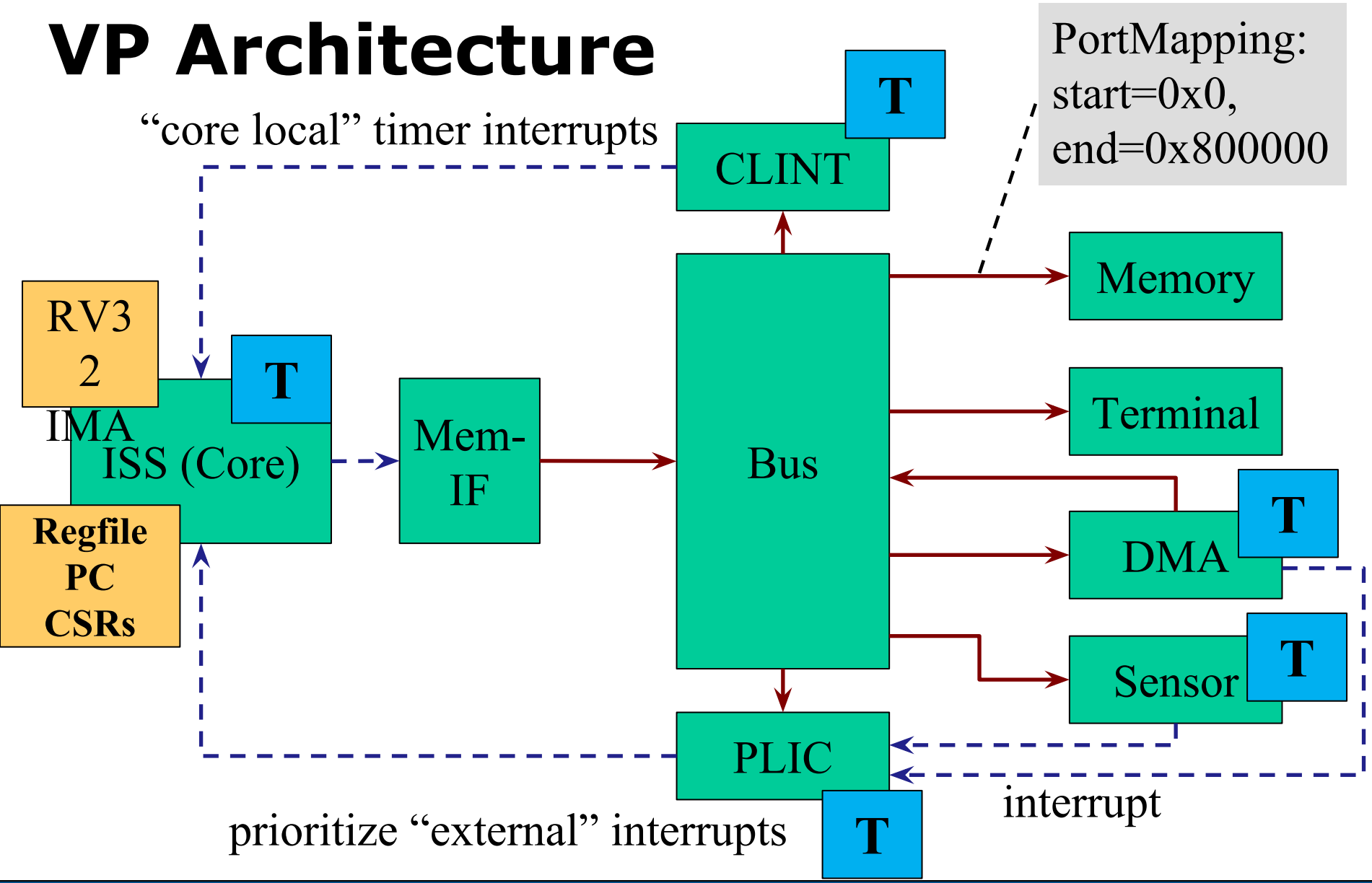
VP Overview (2)

- **Components:**
  - Core, Bus, Memory,
  - Interrupt Controller,
  - Peripherals (Sensor, Timer, DMA, Terminal)
- **Supports:**
  - Interrupts, Syscalls, CLIB, GCOV
  - **Recently:** GDB, FreeRTOS, FAT32
- **Ongoing:**
  - Tübingen timing simulation integration
  - 64 Bit support (first prototype available)
  - Verification: VP model and SW running on VP

VP-based Simulation for RISC-V



VP Architecture



Backup

RISC-V (1)

- Completely open ISA that is freely available
- No license costs involved
- Efficient and versatile design
- High-performance to small embedded devices
- Widely adopted

RISC-V (2)

- Mandatory Integer Instruction Set "I" (~47 instrs.)
  - 32/64/128 Bit

- + Optional Extensions
  - "M", "A", "F", "D", etc.

RV32IMAFD = RV32G

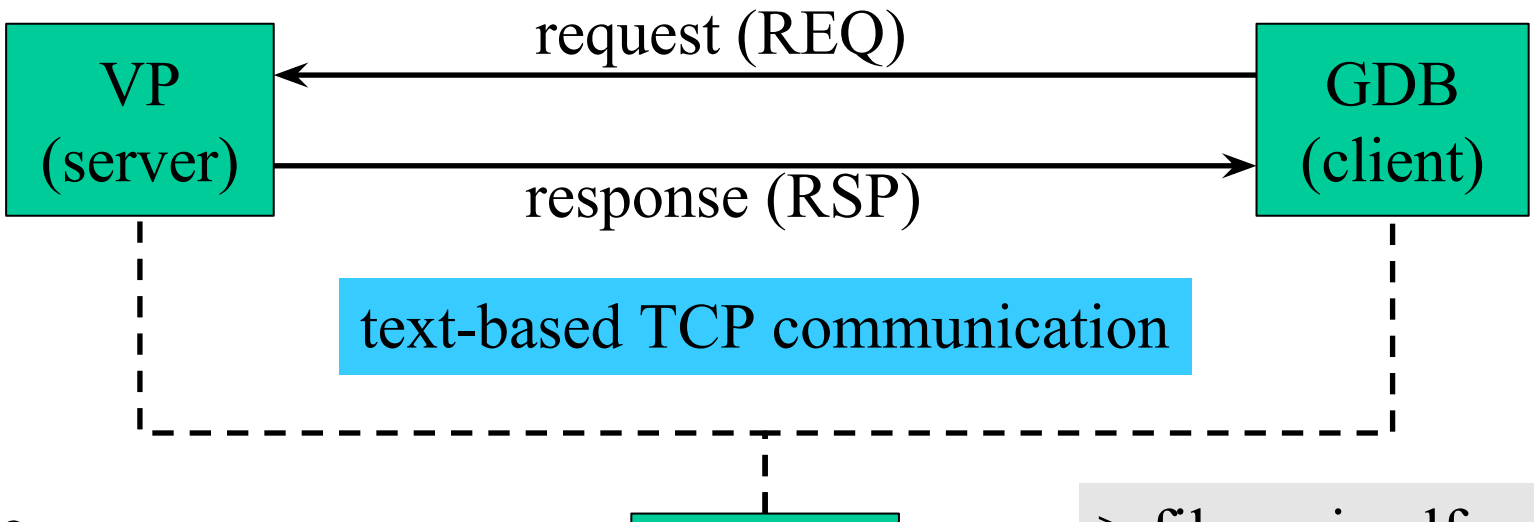
- Control and Status Registers (CSRs) and Environment Interaction

Timing Model

- Simple Instruction-based
  - Fixed execution times for each instruction
  - Easy to configure
- TLM blocking transactions
  - b_transport(*tlm::generic_payload* &payload, *sc_core::sc_time* &delay)
  - Peripherals increment the delay parameter
- More precise models can be integrated

GDB Integration

RSP Interface



main.c

```
1: int main() {
2: int a = 4;
3: a++;
4: return a;
5: }
```

REQ: $m111c4,4#f7

RSP: +$05000000#85

```
> file main.elf
> target remote :5005
> b main.c:4
> c
> print(a)
```

FreeRTOS + Eclipse GDB

Debug

- fdl-18-sensor Default [C/C++ Remote Application]
  - main
    - Thread #1 <main> (Suspended : Step)
      - dump_sensor_data() at main.c:22 0x100d4
      - main() at main.c:33 0x1016c
  - riscv32-unknown-elf-gdb (8.0.50.20170808)

Variables | Breakpoints | Registers | Modules

Name	Type	Value
i	int	6

main.c | irq.c | bootstrap.S

```
 9  _Bool has_sensor_data = 0;
10
11  void sensor_irq_handler() {
12      has_sensor_data = 1;
13  }
14
15  void dump_sensor_data() {
16      while (!has_sensor_data) {
17          asm volatile ("wfi");
18      }
19      has_sensor_data = 0;
20
21      for (int i=0; i<64; ++i) {
22          *TERMINAL_ADDR = *(SENSOR_INPUT_ADDR + i);
23      }
24  }
25
26  int main() {
27      register_interrupt_handler(2, sensor_irq_handler);
28
```

Outline | Disassembly

Enter location here

```
21              for (int i=0; i<64; ++i) {
000100cc:   sw      zero,-20(s0)
000100d0:   j       0x100fc <dump_sensor_data+96>
22              *TERMINAL_ADDR = *(SENSOR_INPUT_ADDR + i);
000100d4:   lui     a4,0x50000
000100d8:   lw      a5,-20(s0)
000100dc:   add     a4,a4,a5
000100e0:   lui     a5,0x20000
000100e4:   lbu     a4,0(a4) # 0x50000000
000100e8:   andi    a4,a4,255
000100ec:   sb      a4,0(a5) # 0x20000000
21              for (int i=0; i<64; ++i) {
000100f0:   lw      a5,-20(s0)
000100f4:   addi    a5,a5,1
000100f8:   sw      a5,-20(s0)
000100fc:   lw      a4,-20(s0)
00010100:   li      a5,63
```

https://github.com/agra-uni-bremen/riscv-freertos

Performance Optimization

- DMI for main memory access
  (core concept: char* access to memory)
  - DMI for instruction fetching
  - DMI for data access

- Temporal decoupling in CPU core
  - Evaluate different local time quantums

Performance Evaluation

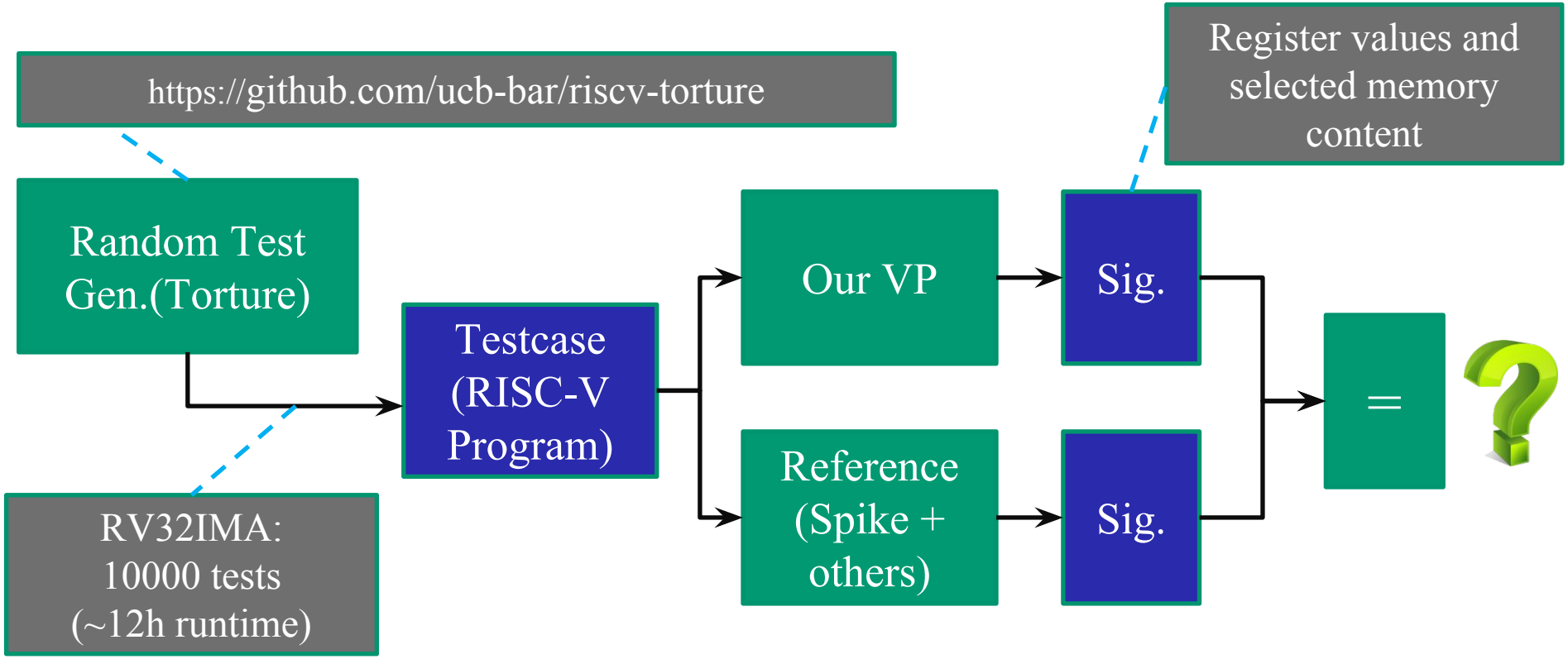
- Simulation time in seconds (T.O. set to 4 hours)
- VP ~15-20 million instructions per second (AMD 2.8GHz)

Bench-mark	RTL Sim	VP Sim	+i_dmi	+d_dmi	+q10	+q100
mergesort/s	56.70	0.39	0.35	0.35	0.32	0.29
primes/s	823.11	1.73	1.01	0.96	0.59	0.49
qsort/s	64.50	0.40	0.35	0.34	0.31	0.30
sha512/s	1307.23	3.23	1.87	1.57	0.90	0.71
mergesort	T.O.	197.32	107.89	86.48	41.17	27.77
primes	T.O.	2400.32	1214.71	1089.36	542.46	387.09
qsort	T.O.	1204.98	698.50	510.70	262.93	116.73
sha512	T.O.	2773.60	1556.02	1302.75	616.10	432.52

VP (ISS) Testing

- RISC-V ISA tests from Berkeley: passed (57 tests, RV32IMA)

https://github.com/riscv/riscv-tests



Conclusions

- Configurable and Extensible RISC-V based Virtual Prototype
- Future work:
  - 64 Bit Support and ISA Extensions (e.g. compressed instructions)
  - Verification
    - RISC-V VP Model:
      Symbolic simulation using SISSI [1]
      UVM / CRAVE [2]
    - SW running on RISC-V VP:
      Symbolic execution to check user assertions
  - Enhanced performance and power estimation

[1] Verifying SystemC using Intermediate Verification Language and Stateful Symbolic Simulation (TCAD 2018)
[2] CRAVE: An Advanced Constrained RAndom Verification Environment for SystemC (SoC 2012)

Extensible and Configurable RISC-V based Virtual Prototype

http://www.systemc-verification.org/riscv-vp

Vladimir Herdt[1]
Daniel Große[1,2]
Hoang M. Le[1]
Rolf Drechsler[1,2]

[1]University of Bremen & [2]DFKI Bremen, Germany
vherdt@informatik.uni-bremen.de

CONFIRM
16ES0565

edaclusterforschung

SyDe
graduate school, funded by
German Excellence Initiative