Imperas S Google Cloud E metrics

RISC-V Compliance & Verification Techniques Processor Cores and Custom Extensions

Simon Davidmann & Lee Moore - Imperas Software Ltd.

Richard Ho - Google, Inc.

Doug Letcher - Metrics Technology, Inc.

Kevin McDermott - Imperas Software Ltd.





Introduction

• DV solution for RISC-V RTL cores



- Collaboration between Google, Imperas and Metrics
- Industry adoption has started
- The basis for RISC-V core verification in:











Agenda

- RISC-V Intro and Status
- Industry requirements for ISA Compliance and Hardware Design Verification
- RISC-V ISA Compliance
- Processor Hardware Design Verification
- RISC-V Instruction Stream Generation
- Hardware Design Verification Flow
- Demonstration walk-through
- Scaling verification using Cloud resources





Agenda

- RISC-V Intro and Status
- Industry requirements for ISA Compliance and Hardware Design Verification
- RISC-V ISA Compliance
- Processor Hardware Design Verification
- RISC-V Instruction Stream Generation
- Hardware Design Verification Flow
- Demonstration walk-through
- Scaling verification using Cloud resources





What is RISC-V

- An open-source hardware Instruction Set Architecture (ISA)
- Started in 2010 at UC Berkeley
- Frozen base user spec released in 2014
- Fifth major RISC ISA design effort out of Berkeley
- Ratification of RISC-V Base ISA and Privileged Specifications, June 2019







RISC-V ushers in new era of silicon design









Technical priorities in 20 focus areas

V Extension (Vector Ops) Task Group

Cryptographic Extension Task Group

Debug Specification Task Group

Fast Interrupts Spec Task Group

Memory Model Spec Task Group

Processor Trace Spec Task Group

Sv128 Specification Task Group

Compliance Task Group



Opcode Space Mgmt Standing Committee Software Standing Committee

Base ISA Ratification Task Group

Privileged ISA Spec Task Group

UNIX-Class Platform Spec Task Group

Formal Specification Task Group

Trusted Execution Env Spec Task Group

B Extension (Bit Manipulation) Task Group

J Extension (Dynam. Translated Lang) Task Group

+ Security Committee and proposed Safety Task Group

P Extension (Packed-SIMD Inst) Task Group





© Accellera Systems Initiative

RISC-V Bit Manipulation Instructions Working Group https://github.com/riscv/riscv-bitmanip

- New, optional, not yet standardized, instruction extension
- The bitmanip instructions extend the RISC-V instruction set to enable efficient bit manipulation
- This includes operations like:
 - counting bits, leading zeros, etc.
 - bit extraction
 - rotations, shifting and reversing
- Current status: specification under review 0.9.0
- Imperas has modeled / maintains and includes in its executable reference model
- Available now: https://github.com/riscv/riscv-bitmanip
 - Includes Imperas free riscvOVPsim simulator as reference







RISC-V Vector Instructions Working Group https://github.com/riscv/riscv-v-spec

- Optional, not yet standardized, instruction extension
- Been developed by RISC-V founders for years
- Specifications available 0.7.1
- GitHub free download reference simulator <u>https://github.com/riscv/riscv-ovpsim</u>
- Imperas has implemented latest draft Vector to riscvOVPsim free simulator
- Imperas is a reference implementation delivered to early users of Vector RTL









More Information:



- <u>https://riscv.org</u>
- <u>www.imperas.com</u>, <u>www.OVPworld.org</u>
- https://github.com/riscv/riscv-compliance (RISC-V compliance suite & reference simulator)
- <u>https://github.com/riscv/riscv-bitmanip</u> (bitmanip spec & reference simulator)
- <u>https://github.com/riscv/riscv-v-spec</u> (vector spec)
- <u>https://github.com/riscv/riscv-ovpsim</u> (free riscvOVPsim reference simulator)





RISC-V Cores - Commercial and Open Source

- Commercial IP Providers
 - SiFive, Andes, Codasip, Syntacore,
- OpenHW Group
 - PULP RI5CY, Ariane /ETH Zurich => Core-V
- CHIPS Alliance
 - SweRV / WD
- lowRISC
 - PULP Zero-riscy /ETH Zurich => Ibex





RISC-V: Flexibility within the ISA framework

Ecosystem support - Commercial and Open Source

- Commercial
 - IAR, Lauterbach, Segger, ExpressLogic, Imperas, ...
- Key Open Source activities include
 - GNU tools, gcc, gdb, …
 - LLVM, LLDB, ...
 - FreeRTOS, Zephyr, Linux, ...







Verification Tools

- Simulation: Imperas (OVPsim), spike, qemu, ...
- Verification: Google (open source test generator), Valtrix (commercial test generator), ...





Agenda

- RISC-V Intro and Status
- Industry requirements for ISA Compliance and Hardware Design Verification
- RISC-V ISA Compliance
- Processor Hardware Design Verification
- RISC-V Instruction Stream Generation
- Hardware Design Verification Flow
- Demonstration walk-through
- Scaling verification using Cloud resources





Compliance for RISC-V is Important

Q: What is meant by "compliance"?

A: The device works within the envelope of the agreed specifications

Q: Is there an easy process or path to follow to develop methodologies/tools for compliance?

A: NO – all established ISAs are single company controlled and those companies work extremely hard on proprietary solutions to ensure that all designs that go out their door work correctly – so RISC-V has to pioneer compliance collectively

Q: What happens if the RISC-V industry builds devices that are not complying with specifications?

A: Users cannot assume that tools like C compilers, operating systems, and application software will be transferable across devices and work correctly



© Accellera Systems Initiative



Imperas key contributor to the RISC-V Compliance Suite

- Compliance suite is 'work in progress'
- Two components
 - Test suites
 - Each suite focuses on a feature set of the RISC-V envelope
 - Initial focus is instructions, user mode spec, e.g. rv32i, rv32im, rv32imc, rv64i, ...
 - Awaiting RISC-V platform specifications to subset privilege spec, before starting privilege suites
 - Framework
 - Make, bash, and scripts
 - Encapsulate compiler tools, linkers, simulators, and targets (Devices Under Test)
 - Includes simulator: as example target, and to generate reference signatures
 - Run: Select suite and target
 - Runs each test, target produces signatures, compares to saved golden reference signature
- Available: www.github.com/riscv/riscv-compliance





Agenda

- RISC-V Intro and Status
- Industry requirements for ISA Compliance and Hardware Design Verification
- RISC-V ISA Compliance
- Processor Hardware Design Verification
- RISC-V Instruction Stream Generation
- Hardware Design Verification Flow
- Demonstration walk-through
- Scaling verification using Cloud resources





ISA Compliance (agenda)

- An Intro to testing RISC-V ISA standard Compliance
- Scope and approaches to ISA compliance
- Components needed
 - Test Framework
 - Tests
 - Assembler/compiler
 - Reference simulator to generate known good results
 - Device Under Test with encapsulation to run it
- Overview of riscvOVPsim compliance reference simulator
- RISC-V Foundation Compliance Suite walk-through and demo
 - Github
 - Flow, tests, select target, run, compare
 - Adding a new target
- Status and roadmap





RISC-V Getting your chip Verified

- Compliance Tests
- DV Tests
- Formal tools
- Simulators
- Emulators
- FPGA's





RISC-V Compliance

- What is Compliance
- What is NOT Compliance
 - Where does Compliance end and DV begin
- Possible Methods





What is Compliance (in RISC-V)

- Participating in the Compliance WG, I have asked to get a definition this has proven difficult.
- There are deep discussions about how and what to test in the compliance working group.
- Discussions frequently drift into 'oh-no that is DV, not compliance', or 'that is too deep/detailed for compliance testing'
- So what can we definitively say is compliance ?





What is Compliance (in RISC-V)

- The device works within the envelope of the agreed specifications
- Compliance is adhering to the definition of the Instruction Set Architecture specification and its intended semantic behavior.
- Proving the ISA behavior should be independent of any understanding of the underlying micro-architectural implementation.





What is NOT Compliance (in RISC-V)

- We should not be concerned about compliance testing a given implementation in order to prove that it may
 - perform out of order execution
 - perform parallel execution (multiple dispatch)
 - execute multi/single cycle instruction(s)
 - have a shallow or deep pipeline
 - implement complex branch predictions
 - perform speculative execution based upon branch prediction
 - *enable contentious mode*
 - implement an instruction through trapped emulation
 - Extension-M : providing Mult but no Div
 - *disable contentious mode*





Compliance Testing Definition

- Compliance is attempting to test all of the possible instruction alternatives, without attempting to test all of the possible values either provided to, or produced by an execution unit, or the data paths being exercised.
- This causes many grey areas of overlaps between compliance and conventional DV, or a simple question:

Where does Compliance end, and DV begin?





Running the RISC-V compliance suite Framework







• For example, is it sufficient to exercise an **add** instruction by the following code snippet

```
la x31, RESULT_ADDR
li x1, 2
li x2, 3
add x3, x2, x1  // x3 = 5
sw x3, x31(0)  // Save to SIGNATURE
```

```
RESULT_ADDR:
.word 0x0
```







• Or should specific bit patterns be exercised, for example a walking 1, or walking 0 in register values

```
la x29, RESULT ADDR
li x30, 0
for il in 0 to 31; do
  li x1, (0x1 << $i1)
  for i2 in 0 to 31; do
   li x2, (0x1 << $i2)
   add x3, x2, x1
   add x31, x30, x29 // Calculate a new (sw) ptr
   sw x3, x31(0)
   addi x30, 4
  done
done
RESULT ADDR:
.word 0x0
```







- Given that we are adding a walking '1' to a 0 value in the first inner loop pass, then we know we are also creating a walking '1' in the result register (for an add instruction).
- This is pretty simple given an 'add' instruction, in the case of a more complex instruction, then to ensure we have a walking '1' in the result, we probably need some coverage metrics to ensure we meet our goals more on that later.





- Question: is a walking '1' sufficient ? This will not test a result overflow, underflow etc. so do we need to extend this simple test with some corner case values ?
- for example define a list of special values
 - list = {(-ve largest), (-ve mid), -1, 0, 1, (+ve mid),
 (+ve largest)}
 - list = {0x8000000, 0xC000000, 0xFFFFFFF, 0x0000000, 0x0000001, 0x3FFFFFF, 0x7FFFFFFF}
- Now unroll the lists as the data values





WARNING

• Serious Risk of Encroaching upon DV Territory





• Question: why not just go for the full range of values ?

```
la x29, RESULT ADDR
li x30, 0
for i1 in 0 to 0xFFFFFFF; do
  li x1, $i1
  for i2 in 0 to 0xFFFFFFF; do
   li x2, $i2
   add x3, x2, x1
   add x31, x30, x29
   sw x3, x31(0)
   addi x30, 4
  done
done
RESULT ADDR:
.word 0x0
```







DANGER

• Deep inside DV Territory





- Question, do we also need to exercise each possible register combination
 - rDlist = x31, x30, x29, ...
 - rS1list = x31, x30, x29, ...
 - rS2list = x31, x30, x29, ...
- In which case, why not simply ...





// Allocate 3 registers from pool for BASE, OFFSET, PTR la x(ADDBASE), RESULT ADDR li x(ADDOFF), 0 li x(ADDPTR), 0 foreach rD in rDlist; do foreach rS1 in rS1list; do foreach rS2 in rS2list; do for i1 in 0 to 0xFFFFFFF; do li x(rS1), \$i1 for i2 in 0 to 0xFFFFFFF; do li x(rS2), \$i2 add x(rD), x(rS2), x(rS1)add x(ADDPTR), x(ADDOFF), x(ADDBASE) sw x(rD), x(ADDPTR)(0)addi x(ADDOFF), 4 done done done done done





Game Over

- You just generated too many permutations, and exceeded way out past DV territory
- RD RS1 RS2 RS1v RS2v
- 2**5 x 2**5 x 2**5 x 2**32 x 2**32
- In this case it is a 32bit Architecture change to 64bit, and the permutations increase accordingly
- This is simply not feasible
- remember we will also need to adhere to the framework constraints (what are these ?)




Test framework Constraints

- As yet not clearly defined !
- How much available memory for Code/Data ?
- Effect:
 - How to handle large displacements for Branch, Jump
 - How to handle large immediate offsets in Load/Store
- Result:
 - Tradeoff, Tradeoff, Tradeoff





Compliance goals, For each Instruction

- Attempt to use every possible register as both an input and output (where an output is produced)
- For all immediate values (offset, displacement, shift etc) Attempt to show that all bits in the value have been exercised as all 1's, all 0's and both a 1 and 0, preferably in isolation
 - eg for a 4 bit immediate value, the following possible values
 - -0000, 0001, 0010, 0100, 1000,
 - -1110, 1101, 1011, 0111, 1111
- ie, avoiding simple 2-state toggle : 0000, 1111





Compliance goals, For each Instruction

- Attempt to provide input register values whereby corner cases are exercised
 - -MAX -MID, -MIN, -1, 0, 1, +MIN, +MID, +MAX (FP: +INF, -INF +0, -0)
 - Walking '1', Walking '0' values
- Attempt to produce output register values whereby corner cases are exercised
 - -MAX -MID, -MIN, -1, 0, 1, +MIN, +MID, +MAX (FP: +INF, -INF +0, -0)
 - Walking '1', Walking '0' values





Compliance OUT-OF-SCOPE, do not attempt

- Exercise Register/Value Cross coverage
 - Register indices, eg rs1 rs2 rd cross coverage => 2**5 x 2**5 x 2**5
 - Register values, eg rs1=X rs2=Y cross coverage => 2*32 x 2*32
- Exercise all possible immediate values
 - for a given 2**n immediate, do not exercise 2**n values
 - (Unless) small set, eg shift/rotate amount 2**5
- Branch/Jump Targets
 - TBD, related to the framework constraints (relative to PC is difficult to fully exploit)
- Load/Store offsets
 - Where an address is R + I, the the R can be inversely adjusted in order to offset far load/store
 - eg For a given address X, Rx=(X-4), Imm=4,

** note :

Register+Immediate targets of Load/Store/Branch/Jump, the target address can be brought back into range by rebasing the base pointer register so that an immediate is provided resulting in a valid target address, given a potential out of range intermediate immediate/displacement.



Compliance Test in Memory Structure

- TEXT section
 - Exception Handling Code (**contentious**)
 - Startup Code
 - Test Code (Body of the test)
 - Shutdown Code
- DATA section
 - Debug/logging strings
 - Signature memory (Pre-initialized, Post-extracted)





Compliance Test Semantics

- Loader program to get memory in initial state
 - RTL Simulator \$writemem()
 - ISS readelf()
 - HW debug load
- Execute
 - Method of finish/exit detection
- Extract Signature (results)
- Compare Signature to golden reference





Compliance test Porting to target

- Compilation infrastructure
- Macro body definitions
- Linker script to describe the target memory structure





Compliance Code snippet RV32I - ADD

Addresses for test data and results

- la x1, test_A1_data
- la x2, test_A1_res

Load testdata

lw x3, 0(x1)

Register initialization

li x4, 0 li x5, 1

Test : add <dst-reg>, <src1-reg>, <src2-reg> add x4, x3, x4 add x5, x3, x5

Store results to signature memory for extraction to signature file

- sw x3, 0(x2)
- sw x4, 4(x2)
- sw x5, 8(x2)





Compliance Linker snippet (ibex)

OUTPUT_ARCH("riscv") ENTRY(_start)

SECTIONS

. = 0x00000000; .text.trap : { *(.text.trap) }

. = 0x0000080; .text.init : { *(.text.init) }

. = ALIGN(0x1000); .text : { *(.text) }

```
. = ALIGN(0x1000);
.data : { *(.data) }
.data.string : { *(.data.string)}
.bss : { *(.bss) }
_end = .;
```





Running the RISC-V compliance suite







Running the RISC-V compliance suite

	File Edit View Scrollback Bookmarks Settings Help
shell moore	Check I-CSRRSI-01 0K
File Edit View Scrollback Bookmarks Settings Help	Check I-CSRRW-01 UK
The (OP OF) Target 'riscu()/Psim(spi') has object file read from '/home/meere/WOPK/riscy dy tep/riscy complia	Check I-DELAY SLOTS-01 OK
no (or or) alget is to oversin/ ou has object file fead from / home/moore/work/fisco-dv.to//fisco-domptia	Check I-EBREAK-01 0K
	Check I-ECALL-01 OK
Into (UK_PH) Program Headers:	Check I-ENDIANESS-01 OK
Into (UR_PH) Type Uttset VirtAddr PhysAddr FileSiz MemSiz Flags Align	Check I-FENCE.I-0I OK
Info (OR_PD) LOAD 0x00001000 0x80000000 0x80000000 0x000003c4 0x000003c4 R-E 1000	Check $T_{-1}\Delta I_{-}\Omega I_{-}\Omega K$
Info (OR_PD) LOAD 0x00002000 0x80001000 0x80001000 0x00001204 0x00001204 RW- 1000	Check I-JALR-01 0K
Info (SIGNATURE_DUMP) Found Symbol 'begin_signature' in application at 0x80002030	Check I-LB-01 0K
Info (SIGNATURE DUMP) Found Symbol 'end signature' in application at 0x800020e0	Check I-LBU-01 OK
Info (SIGNATURE DUMP) Signature File enabled, file '/home/moore/WORK/riscv-dv.top/riscv-compliance.ref/work/	Check I-LH-01 OK
rv32i/T-ADD-01.signature.output'	Check I-LHUT-01 UK
Info (SIGNATURE DUMP) Extracting signature from 0x80002030 size 176 bytes	Check I-LW-01 OK
Info (SIGNATHE DIMP) Symbol 'begin signature' at 0x80002030	Check I-MISALIGN JMP-01 OK
Tafe (STONATURE DUMP) Symbol construct at 0x0000200	Check I-MISALIGN_LDST-01 0K
The (STONATORE DUMP) Symbol end Signature at 0x0002000	Check I-NOP-01 OK
Into (SIGNATORE DOMP) Intercept write_tonost . Generate Signature file	Check I-OR-01 OK
+++++++++++++++++++++++++++++++++++++++	Check I-BE size-01 OK
00000010000001800000007fffffff	Check I-RF width-01 OK
800000018000000000000000000000000000000	Check I-RF_x0-01 OK
fffffffe0000000ffffffffffffffff	Check I-SB-01 OK
7fffffff7ffffffffffffffffffffffe	Check I-SH-01 OK
ffffffffffffffffffe8000000	Check I-SLL-01 UK
7ffffff80000001800000080000000	Check I-SLT-01 0K
0000abcd0000001000000ffffffff	Check I-SLTI-01 OK
	Check I-SLTIU-01 OK
	Check I-SLTU-01 OK
	Check I-SRA-01 0K
303238143032381400000000	Check I-SRI-01 OK
lest PASSED	Check I-SRLI-01 OK
Carl moore thath Carl shell means	Check I-SUB-01 OK
Shell : moore	Check I-SW-01 OK
	Check I-XOR-01 OK
	Check I-XURI-01 UK
1. Start with ssh to remote shell in Metrics Cloud Platform	OK: 55/55
	moore@shell-1:~/WORK/riscv-dv.top/riscv-compliance.ref\$
2 Load tests etc. from Git. e.g. compliance suite	moore : hash
	Shell : moore
2 Dup as if in least shall	\sim
	(2019
1 In this second DV(201 and all in the second its	
4. In this example RV32I compliance suite	



ຢ∂⊗ 🗹 👘	999 1997	>							Thu Oct 24, 3:51 PM	Lee Moor
		moore/p:bash		_ 				moore : bash		808
File Edit View	Scrollback	Bookmarks	Settings Help		File Edit View S	Scrollback Bookmarks Set	tings Help			
You have new mail in / [moore@lnx33 pics]\$ []	var/spool/mail/	noore			[moore@lnx33 ~] [moore@lnx33 ~] [moore@lnx33 ~]	\$ make hel^C \$ \$ ∎	Ĭ			
moo	ore/p : bash									
						moore : bash		moore : bash		
						moore , bush		moore : pasir		

- Observing and measuring the Coverage Points
 - Instruction Type
 - Target RD Register usage
 - Source RS1, RS2 Register usage
 - Immediate Value usage (buckets, data points)
 - Register Value usage (buckets, data points)





• Coverage Results for ADD

- Contact Lee Moore at <u>moore@imperas.com</u>
- For latest results and updates presented at DVCon Europe 2019 in Munich





• Proving the Coverage measurements, by propagation of an intended result

Coverage observes the following

addi x2, x1=0x1, 0x1 // -> x2=0x2

This implies the following points are met for Instruction addi

Target Register [x2] => Covered

Target Register Value [1]=1 => Covered

Source Register [x1] => Covered

Source Register Value [0]=1 => Covered

Immediate Value [0]=1 => Covered





- If our code snippet looks like this, we are gradually improving our coverage right ?
- la x31, ADDR_SIGNATURE

addi x2, x1=0x1, 0x0 // -> x2=0x1

- addi x2, x1=0x1, 0x1 // \rightarrow x2=0x2
- addi x2, x1=0x1, 0x2 // -> x2=0x3
- addi x2, x1=0x1, 0x4 // -> x2=0x5
- addi x2, x1=0x1, 0x8 // -> x2=0x9
- addi x2, x1=0x1, 0xF // -> x2=0x10

sw x2, x31(0) // Store Signature





- Coverage is monitoring the Execution unit, but ignoring the Load/Store Unit, so in fact
- la x31, ADDR_SIGNATURE

addi x2, x1=0x1, 0x0 // -> x2=0x1

- addi x2, x1=0x1, 0x1 // \rightarrow x2=0x2
- addi x2, x1=0x1, 0x2 // -> x2=0x3
- addi x2, x1=0x1, 0x4 // -> x2=0x5
- addi x2, x1=0x1, 0x8 // -> x2=0x9
- addi x2, x1=0x1, 0xF // -> x2=0x10 sw x2, x31(0) //

- DISCARDED
- DISCARDED
- DISCARDED
- DISCARDED
- DISCARDED
- Saved
- by this Store





- Coverage is monitoring the execution, but is unaware of the propagation of the effect, and the observability of the program flow to the persistent storage in the SIGNATURE section
- How can we prove that a reportedly covered item, is observed within the SIGNATURE section ?





- Fault Injection to provide Mutation Testing
- For a given reported coverage point, mutate the instruction to have provided the contribution and test the propagation of that mutation to an observable entry in the SIGNATURE section
- Ref: https://www.geeksforgeeks.org/software-testing-mutation-testing/





• For the given code snippet previously, mutate the operation to a legal alternate la x31, ADDR SIGNATURE

// MUTATE ON :

```
<original-instruction> addi x2, x1=0x1, 0x0
<mutated-instruction> add x0, x0, x0
```

// MUTATE OFF

- addi x2, x1=0x1, 0x1 // \rightarrow x2=0x2 DISCARDED
- addi x2, x1=0x1, 0x2 // \rightarrow x2=0x3 DISCARDED
- addi x2, x1=0x1, 0x4 // \rightarrow x2=0x5 DISCARDED
- addi x2, x1=0x1, 0x8 // \rightarrow x2=0x9 DISCARDED
- addi x2, x1=0x1, 0xF // -> x2=0x10 Saved sw x2, x31(0) // -> by this Store



- The mutated instruction could simply be
 - replace by a NOP : add, x0, x0, x0
- Or, override parts of the decode, for example the decode for an addi is as follows
 - ADDI : 12'b(Imm), 5'b(rs1), 3'b(000), 5'b(rd), 7'b(0010011)
 - The fixed parts of the decode are field [3](000) and field [5](0010011)
 - All other bits affect the immediate values and register indices for source and destination
 - In effect the decode appears thus
 -000001 0011
 - all bits indicated as '.' could be mutated, in order to observe whether this incurs a propagated effect to our SIGNATURE section.





- We need to be careful to ensure that introducing the mutated instructions does not cause a 'fault model explosion', in other words for the ADDI case above there are 22 bits which can be considered for mutation, but it would not be prudent to generate 2**22 fault models
- A pragmatic approach is to create the following fault models
 - NOP Replacement
 - Bit inversion
- This would produce a total of 23 fault models for this one instruction bearing in mind that the same approach is taken for every possible ADDI instruction
- the encoding for
 - addi x2, x1=0x1, 0x1
- would thus be
 - | imm12 | rs1 |dec| rd | dec
 - 000 0010011
 - 0000000001 00001 000 00010 0010011





- Potential Fault Models would be
- | imm12 | rs1 |dec| rd | dec
- 1000000001 0001 000 00010 0010011
- 0**1**000000001 00001 000 00010 0010011
- 00**1**00000001 00001 **000** 00010 **0010011**
- 000**1**0000001 00001 000 00010 0010011
- etc ...





- These fault models are inserted automatically by the Imperas Fault Simulator at runtime without changing the elf/binary
- Having run these fault models on an early version of the compliance suite, yielded some very interesting results, whereby the test engineer was utterly convinced that he had covered a particular scenario, but actually injecting the fault and following the propagation of the program flow proved the result had been in someway masked.
- We found numerous reasons for this, including calculated values not stored in the SIGNATURE region, values in the SIGNATURE overwritten, or default initialization SIGNATURE values, having the same value as calculated values.
- initialized at -1 (0xFFFFFFF) and the value -1 (0xFFFFFFF) stored hence no observable difference.





- Running the Imperas Mutating Fault Simulator on the existing Compliance suite to grade the coverage
- Automating the Fault injection to assert the coverage metrics
- Imperas Simulator
 - injected x faults in y simulations and executed in n seconds (Schrodinger's Cat)
- <Show Running>





File Edit View Scrollback Bookmarks Settings Help

Σ.

33

1 60 🕑

2-

۶.

shell : moore



2.

moore : rlogin

>-

...ationTestFork : bash

Agenda

- RISC-V Intro and Status
- Industry requirements for ISA Compliance and Hardware Design Verification
- RISC-V ISA Compliance
- Processor Hardware Design Verification
- RISC-V Instruction Stream Generation
- Hardware Design Verification Flow
- Demonstration walk-through
- Scaling verification using Cloud resources





Processor Hardware Design Verification (agenda)

- Objective of HW DV for processors
 - Prove beyond (reasonable) doubt design meets objectives
- Available technologies and tools (Dynamic)
 - Reference model
 - Compliance suites, tools, tests
 - Directed test suites
 - Industry benchmarks and Operating Systems
 - Random Instruction Generators
- Formal models, properties libraries, model checkers, equivalence checkers
- ISS, RTL simulators, FPGAs, Emulators, ...





Processor Hardware Design Verification Dynamic Testing approaches

- Prove beyond (reasonable) doubt design meets objectives, using
 - Directed tests
 - Architectural tests
 - Instruction correctness
 - Micro-architectural
 - Pipeline correctness, hazards, speculative execution, branch prediction, multiple issue
 - Compliance
 - Stress
 - Industry Benchmark, eg Coremark, Dhrystones, Linpack
 - Pseudo Random Instruction Stream Generators
 - Architectural tests
 - Micro-architectural
 - Constraint solving & Coverage metrics





Processor Hardware Design Verification Targets for execution

- Instruction Set Simulation (ISS)
- Cycle Accurate Simulation (CAS)
- Virtual Prototype/Platform, System Emulator
- RTL Simulation
- FPGA
- Emulator (FPGA Prototyping)
- Accelerator (Hardware RTL Simulator)
- Combination of Above, eg ISS + Accelerator
- Physical Device





Agenda

- RISC-V Intro and Status
- Industry requirements for ISA Compliance and Hardware Design Verification
- RISC-V ISA Compliance
- Processor Hardware Design Verification
- RISC-V Instruction Stream Generation
- Hardware Design Verification Flow
- Demonstration walk-through
- Scaling verification using Cloud resources





RISC-V Instruction Stream Generation (agenda)



- Motivation
- Existing open source solutions
- Missing pieces
- Google Open Source RISC-V ISG
- Key features
- Generator internal flow





RISC-V Instruction Stream Generation



- Design Verification is the Foundation of Reuse
- Quality is key to reuse
- Stress testing needed to ensure IP will operate correctly when integrated into new design
- DV accounts for >50% of project time for complex chips





Open source RISC-V processor verification

solutions

Verification is one of the key challenges of modern processor development.

riscv-tests

Assembly unit test

A simple test framework focused on sanity testing the basic functionality of each RISC-V instruction. It's a very good starting point to find basic implementation issues.

riscv-torture

Scala-based RISC-V assembly generator

Provides a good mix of hand-written sequences. Supports most RISC-V ISA extensions which makes it very attractive. Simple program structure and fixed privileged mode setting.









Many missing pieces

- Complex branch structure
- MMU stress testing
- Exception scenarios
- Compressed instruction support
- Full privileged mode operation verification
- Coverage model
- ...



Motivation

Build a high quality open DV infrastructure that can be adopted and enhanced by DV engineers to improve the verification quality of RISC-V processors.

Google Cloud



DESIGN AND VE





Google RISC-V Instruction Stream Generation

- High quality SystemVerilog UVM DV infrastructure
- Open source
- Drives a RISC-V core through corner cases and pushes it to the limit



https://github.com/google/riscv-dv




Key Features

01



Randomness

Randomize everything: instruction, ordering, program structure, privileged mode setting, exceptions..

Architecture Aware

The generated program should be able to hit the corner cases of the processor architectural features.

Performance

03

The instruction generator should be scalable to generate a large program in a short period of time.

04

Extendability

Easy to add new instruction sequences, custom instruction extension, custom CSR etc.



accel

SYSTEMS INITIATIVE







Randomness

Instruction level randomization

Cover all possible operands and immediate values of each instruction Example: Arithmetic overflow, divide by zero, long branch, exceptions etc.

Sequence level randomization

Maximize the possibility of instruction orders and dependencies



Program level randomization

Random privileged mode setting, page table organization, program calls













Instruction randomization

Easy part

Arithmetic: ADD, SUB, LUI, MUL, DIV ... Shift: SLLI, SRL, SRLI, SRAI ... Logical: XOR, OR, AND, ANDI ... Compare: SLTI, SLT, SLTU ... Others: FENCE, SFENCE, EBREAK ...

Randomize each instruction individually with bias towards corner cases. (overflow, underflow, compressed instruction)



Tricky part

Branch / jump instruction Need a valid branch/jump target Avoid infinite loop

Load/store/jump instruction

Need an additional instruction to setup the base address The calculated address should be a valid location

CSR instruction

Avoid randomly changing the privileged state Result checking could be a challenge as the privileged CSR behavior could be implementation-specific.







Architecture Aware 01 Branch prediction

O2 MMU (TLB, Cache etc)









© Accellera Systems Initiative





Architecture Aware 03 Issue, execute, commit



It's not just a random stream of instructions, it should be designed to effectively verify the architectural features of the processor.



• CHIPS • ALLIANCE









DESIGN AND VERIFIC

CONFERENCE AND EXHIBITION

IROP

2019



Complete feature list

Supported ISA RV32IMC, RV64IMC

Supported privileged mode User mode, supervisor mode, machine mode

Supported spec version

User level spec 2.20, privileged mode spec 1.10

Supported RTL simulator

VCS, Incisive, Metrics



Test suite

Basic arithmetic instruction test

Random instruction test

MMU stress test

HW/SW interrupt test

Page table exception test

Branch/jump instruction stress test

Interrupt/trap delegation test

Privileged CSR test







Agenda

- RISC-V Intro and Status
- Industry requirements for ISA Compliance and Hardware Design Verification
- RISC-V ISA Compliance
- Processor Hardware Design Verification
- RISC-V Instruction Stream Generation
- Hardware Design Verification Flow
- Demonstration walk-through
- Scaling verification using Cloud resources





Hardware Design Verification Flow (agenda)

- Flow
- Components
 - Google: open source riscv-dv instruction stream generator
 - Imperas: model and simulation golden reference of RISC-V CPU
 - Metrics : SystemVerilog design + UVM simulator for RTL
- Using





Hardware Design Verification Flow



- Google: open source riscv-dv instruction stream generator
- Metrics : SystemVerilog design + UVM simulator for RTL
- Imperas: model and simulation golden reference of RISC-V CPU





accellera

SYSTEMS INITIATIVE

Google SystemVerilog UVM Instruction Stream Generator

C Google Cloud

Open Source SystemVerilog UVM RISC-V Instruction Stream Generator

https://github.com/google/riscv-dv

Keys features:

- Randomness
 - Randomize everything: instruction, ordering, program structure, privileged mode setting, exceptions..
- Architecture-aware
 - Generated program able to hit the corner cases of the processor architectural features
- Performance
 - Instruction generator is scalable to generate a large program in a short period of time
- Extendibility
 - Easy to add new instruction sequences, custom instruction extension, custom CSR etc.





Imperas RISC-V Instruction Set Simulator and full RISC-V Specification Envelope Model



http://www.imperas.com/riscv https://github.com/riscv/riscv-ovpsim

- Industrial quality model and simulator of RISC-V processors for use in compliance, verification and test development
- Complete, fully functional, configurable simulator
 - All 32bit and 64bit features of ratified User and Privilege RISC-V specs
 - Vector extension, version 0.7.1
 - Bit Manipulation extension, version 0.9.0
 - Model source included under Apache 2.0 open source license
- Used as golden reference in RISC-V Compliance Suite and Bit Manipulation group
- Extendibility: easy for user to extend with new instructions and functionality





Metrics (1): SystemVerilog Simulator



accelle

SYSTEMS INITIATIVE

- Complete SystemVerilog IEEE 1800-2012 compliant simulator including UVM
- Includes all the standard features of a modern SystemVerilog simulator including debug, APIs, language and testbench support
- Simulates the testbench, the RTL design, and the populates the coverage models 85

Metrics (2): Full Cloud Platform

- True cloud platform for ASIC and complex FPGA design verification
- On-demand simulation resources
- Modern continuous integration workflow
- Easy access to simulation results from any web browser



=> So the full verification can be done in the Google Cloud using the Metrics Cloud Platform, the Metrics simulator, the Imperas RISC-V reference simulator and the Google Instruction Stream Generator



SYSTEMS INITIATIVE

Agenda

- RISC-V Intro and Status
- Industry requirements for ISA Compliance and Hardware Design Verification
- RISC-V ISA Compliance
- Processor Hardware Design Verification
- RISC-V Instruction Stream Generation
- Hardware Design Verification Flow
- Demonstration walk-through
- Scaling verification using Cloud resources





Demonstration

- Uses RISC-V Core: lowRISC/ibex RTL (RV32IMC)
 - Originated from zero-RI5CY ETH Zurich PULP





DV Flow is controlled by Makefile and bash scripts and includes python scripts



- Compile up SystemVerilog UVM test generator and run it
 - can easily set how many tests to create each run
 - Creates .S files that are then converted to .o
- Run the Imperas ISS to generate reference results
- Compile the SystemVerilog RTL of ibex core and testbench
- Run RTL simulation & record RTL results
- Post-processor run logs and compare





Demo: Configuring flow

- Test generator settings
 - Device Under Test / Processor (ISA, xlen, extensions, modes, mmu, ...)
 - Show settings .sv file
 - Complexity & quantity of tests
 - Show Command line options, testlist.yaml
- Reference ISS riscvOVPsim config
 - Show yaml file for Ibex and config.ic file





Demo show test gen, iss, rtl, compare run

- Show running under Metrics cloud connection
 - make gen
 - make gcc_compile
 - make iss_sim
 - make iss_cmp





And results are simple pass, or detailed fail

- Example of detailed fail:
 - Shows mis-matching instructions
 - Configured here to show 5
 - Full traces etc are kept for review
 - Can dump full VCD for detailed waveform analysis

<pre>simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ compare simulation result under /home/simond/git/ibex/dv/uvm/out Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_instr_base_test.0.5 Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_test.0.5.o.log Processed instruction count : 198 Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rll_sim/riscv_instr_base_test.0.4/trace_core_00_0.log Processed instruction count : 6775 Wismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x00000000000000000013c Wismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x80000088 [45] ovpsim : addi sp,sp,-800 -> sp(0x80000e1c) addr:0x0000000000000000000000000000000000</pre>	<pre>simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ compare simulation result under /home/simond/git/ibex/dv/uvm/out compare simulation result under /home/simond/git/ibex/dv/uvm/out rest: /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_orpsim/riscv_instr_base_test.0.5 Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/rl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Mismatch[1:</pre>	MINGW32:~		(<u>2.5</u> 2)		×
<pre>simond@shell-1:-/git/ibex/dv/uvm\$ simond@shell-1:-/git/ibex/dv/uvm\$ simond@shell-1:-/git/ibex/dv/uvm\$ make post_compare ./compare "/home/simond/git/ibex/dv/uvm\$ make post_compare ./compare "/home/simond/git/ibex/dv/uvm/out" compare simulation result under /home/simond/git/ibex/dv/uvm/out/instr_gen/sec_instr_base_test.0.5 Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_test.0.5 Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rit_sim/riscv_instr_base_test.0.5.0.log Processed instruction count : 198 Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Wismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp.0xb -> sp(0x8000bl3c) addr:0x00000008000013c Wismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp.sp.800 -> sp(0x8000elc) addr:0x00000080000140 Wismatch[3]: [45] ibex : addi x4, x0, 0 -> tp(0x00000080000140 Wismatch[4]: [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x00000008000014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783500) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000012c) addr:0x00000008000014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783500) addr:0x80000088 [47] ovpsim : adi s2,s2,986 -> s2(0x8000012c) addr:0x00000008000014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783500) addr:0x80000088 [47] ovpsim : adi s2,s2,986 -> s2(0x8000012c) addr:0x00000008000014c [47] ibex : addi x9, x9, 1369 -> s1(0x81783500) addr:0x80000088 [47] ovpsim : adi s2,s2,986 -> s2(0x8000012c) addr:0x00000008000014c [47] ovpsim : adi s2,s2,986 -> s2(0x8000012c) addr:0x00000008000014c [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x80000088 [47] ovpsim : adi s2,s2,986 -> s2(0x8000012c) addr:0x00000008000014c [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x80000008c [47] o</pre>	<pre>simond@shell-1:-/git/ibex/dv/uvm\$ simond@shell-1:-/git/ibex/dv/uvm\$ simond@shell-1:-/git/ibex/dv/uvm\$ simond@shell-1:-/git/ibex/dv/uvm\$ simond@shell-1:-/git/ibex/dv/uvm\$ simond@shell-1:-/git/ibex/dv/uvm\$ compare "home/simond/git/ibex/dv/uvm/out" compare simulation result under /home/simond/git/ibex/dv/uvm/out/ Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/asm_tests/riscv_instr_base_test.0.5 Processing ovpsim log: /home/simond/git/ibex/dv/uvm/out/rl_sim/riscv_instr_base_test.0.5.o.log Processed instruction count : 198 Processing ibex log: /home/simond/git/ibex/dv/uvm/out/rl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 675 Wismatch[1:</pre>	simond@shell-1:~/git	/ibex/dv/uvm\$			
<pre>simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ make post_compare ./compare "/home/simond/git/ibex/dv/uvm/out" compare simulation result under /home/simond/git/ibex/dv/uvm/out Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/asm_tests/riscv_instr_base_test.0.S Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/rlstr_gen/riscv_ovpsim/riscv_instr_base_test.0.S Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Wismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x000000008000013c Wismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000b12c) addr:0x0000000000000014c Wismatch[3]: [45] ibex : addi x4, x0, 0 -> tp(0x0000000000000000014c Wismatch[4]: [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x00000000000014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x8000014c) addr:0x0000000000000014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x8000014c) addr:0x0000000000000014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x8000012c) addr:0x000000000000014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x8000012c) addr:0x8000000000000000000014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x8000012c) addr:0x800000000000000000000000000000000000</pre>	<pre>simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ make post_compare ./compare "/home/simond/git/ibex/dv/uvm/out" compare simulation result under /home/simond/git/ibex/dv/uvm/out Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/asm_tests/riscv_instr_base_test.0.5 Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/rit_gen/riscv_ovpsim/riscv_instr_base_test.0.5.o.log Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp.0xb -> sp(0x8000b13c) addr:0x00000008000013c Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [45] ovpsim : auidi sp.sp.800 -> sp(0x8000aelc) addr:0x000000080000140 Mismatch[3]: [46] ibex : addi x4, x0, 0 -> tp(0x0000000080000148 Mismatch[4]: [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x000000080000142 Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000a8 [46] ovpsim : auipc s2,2x986 -> s2(0x8000014c) addr:0x00000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000a8 [47] ovpsim : addi s2,x2,986 -> s2(0x8000014c) addr:0x00000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,x2,986 -> s2(0x80000526) addr:0x00000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,x2,986 -> s2(0x80000526) addr:0x000000088000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,x2,986 -> s2(0x80000526) addr:0x000000088000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,x2,986 -> s2(0x80000526) addr:0x000000088000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovp</pre>	simond@shell-1:~/git	/ibex/dv/uvm\$			
<pre>simond@shell-1:~/git/ibex/dv/uvm\$ make post_compare //compare i/home/simond/git/ibex/dv/uvm/out" compare simulation result under /home/simond/git/ibex/dv/uvm/out Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/asm_tests/riscv_instr_base_test.0.S Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_test.0.S.o.log Processed instruction count : 198 Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rll_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Wismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x80000008000013c Wismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp800 -> sp(0x8000e1c) addr:0x80000080000140 Wismatch[3]: [45] ibex : addi x1, x0, 61 -> ra(0xfc2e4277) addr:0x8000008c [45] ovpsim : addi sp,sp800 -> sp(0x8000e1c) addr:0x80000080000140 Mismatch[3]: [45] ibex : addi x4, x0, 0 -> tp(0x0000000080000148 Mismatch[4]: [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x80000004c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x8178359) addr:0x80000088 [47] ovpsim : auipc s2,0x0 -> s2(0x8000012c) addr:0x800000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x8178359) addr:0x8000000a [47] ovpsim : auipc s2,0x80 -> s2(0x8000025c) addr:0x800000000000000000000000000000000000</pre>	<pre>simond@shell-1:~/git/ibex/dv/uvm\$ simond@shell-1:~/git/ibex/dv/uvm\$ make post_compare //compare i/home/simond/git/ibex/dv/uvm/out" compare simulation result under /home/simond/git/ibex/dv/uvm/out Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_instr_base_test.0.S Processing ovpsim log: /home/simond/git/ibex/dv/uvm/out/rinstr_gen/riscv_ovpsim/riscv_instr_base_test.0.So.log Processed instruction count : 198 Processing ibex log: /home/simond/git/ibex/dv/uvm/out/rll_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Wismatch[1]:</pre>	simond@shell-1:~/git.	/ibex/dv/uvm\$			
<pre>simond@shell-1:~/git/ibex/dv/uvm\$ make post_compare ./compare "/home/simond/git/ibex/dv/uvm/out" compare simulation result under /home/simond/git/ibex/dv/uvm/out Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_instr_base_test.0.S Processed instruction count : 198 Processed instruction count : 198 Processed instruction count : 6775 Mismatch[]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x000000008000013c Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000e1c) addr:0x0000000080000140 Mismatch[3]: [45] ibex : addi x4, x0, 0 -> tp(0x0000000080000140 Mismatch[4]: [45] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ovpsim : mul a3,a2,s8 -> a3(0x000000000) addr:0x800000088 [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x000000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x00000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x80000088 [47] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x00000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x80000088 [47] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x00000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x00000008000015c Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 8 tests PASSED, 1 tests FAILED 8 imond@shell-1:-://dit/ibex/dv/uvm\$</pre>	<pre>simond@shell-1:~/git/ibex/dv/uvm\$ make post_compare ./compare "/home/simond/git/ibex/dv/uvm/out" compare simulation result under /home/simond/git/ibex/dv/uvm/out Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/asm_tests/riscv_instr_base_test.0.S Processed instruction count : 198 Processed instruction count : 675 Wismatch[]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000B13c) addr:0x80000008000013c Wismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000B12c) addr:0x00000008000013c Wismatch[3]: [45] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [45] ovpsim : mul a3,a2,s8 -> ap(0x8000000) addr:0x800000080 [45] ovpsim : mul a3,a2,s8 -> ap(0x8000000) addr:0x800000080 [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x8000008044c Wismatch[5]: [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x80000080800014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000520) addr:0x80000080800014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000520) addr:0x80000008060014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000520) addr:0x80000080800014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000520) addr:0x800000080800014c Wismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000520) addr:0x800000808000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 0 tests PASSED, 1 tests FAILED simond@shell-1:-x/git/ibex/dv/uvm\$</pre>	simond@shell-1:~/git.	/ibex/dv/uvm\$			
<pre>./compare "/home/simond/git/ibex/dv/uvm/out" compare simulation result under /home/simond/git/ibex/dv/uvm/out Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_test.0.S.o.log Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 198 Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x000000008000013c Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000e1c) addr:0x80000080e00140 Mismatch[3]: [45] ibex : addi x4, x0, 0 -> tp(0x00000000000000000000000000000000000</pre>	<pre>./compare "/home/simond/git/ibex/dv/uvm/out" compare simulation result under /home/simond/git/ibex/dv/uvm/out Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/asm_tests/riscv_instr_base_test.0.5 Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 198 Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp.0xb -> sp(0x8000b13c) addr:0x0000000000000000000000000000000000</pre>	simond@shell-1:~/git.	/ibex/dv/uvm\$ make post_compare			
<pre>compare simulation result under /home/simond/git/ibex/dv/uvm/out Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/asm_tests/iscv_instr_base_test.0.S Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_test.0.S.o.log Processed instruction count : 198 Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000bl3c) addr:0x000000008000013c Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000ae1c) addr:0x000000080000140 Mismatch[3]: [45] ibex : addi x4, x0, 0 -> tp(0x000000080000148 Mismatch[4]: [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x000000080000142 Mismatch[5]: [47] ibex : addi x9, y9, 1369 -> s1(0x8178359) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x00000008000014c Mismatch[5]: [47] ibex : addi x9, y9, 1369 -> s1(0x8178359) addr:0x800000ac [47] ovpsim : addi s2,52,986 -> s2(0x8000014c) addr:0x00000008000014c Mismatch[5]: [47] ibex : addi x9, y9, 1369 -> s1(0x8178359) addr:0x800000ac [47] ovpsim : addi s2,52,986 -> s2(0x80000526) addr:0x000000080000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 8 tests PASSED, 1 tests FAILED 8 imond@fabell-1:-//eit/ibex/dv/uvms _</pre>	<pre>compare simulation result under /home/simond/git/ibex/dv/uwm/out Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/asm_tests/riscv_instr_base_test.0.5 Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/rll_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 198 Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x0000000008000013c Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000e1c) addr:0x0000000080000140 Mismatch[3]: [45] ibex : addi x4, x0, 0 -> tp(0x0000000080000148 Mismatch[4]: [45] ibex : addi x4, x0, 0 -> tp(0x0000000080000148 Mismatch[4]: [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x00000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000az [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x00000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000az [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x000000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000az [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x000000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000az [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x0000000080000150 Compare clibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 0 tests PASSED, 1 tests FAILED simond@shell-1:~/git/ibex/dv/uvm\$</pre>	./compare "/home/sim	ond/git/ibex/dv/uvm/out"			
<pre>Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/asm_tests/riscv_instr_base_test.0.S Processing ovpsim log: /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_test.0.So.log Processed instruction count : 198 Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x000000008000013c Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [45] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [45] ibex : addi x4, x0, 0 -> tp(0x000000080000140 Mismatch[3]: [45] ibex : addi x4, x0, 0 -> tp(0x000000080000148 Mismatch[4]: [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x80000088 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x000000000000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x8000014c) addr:0x000000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x000000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x8000526) addr:0x000000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x8000526) addr:0x000000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x8000526) addr:0x0000000080000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 8 tests PASSED, 1 tests FAILED 8 imond/git/bex/dv/uvm\$</pre>	<pre>Test: /home/simond/git/ibex/dv/uvm/out/instr_gen/asm_tests/riscv_instr_base_test.0.5 Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_test.0.5.o.log Processed instruction count : 198 Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x80000008000013c Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000e1c) addr:0x0000000000000000000000000000000000</pre>	compare simulation r	esult under /home/simond/git/ibex/dv/uvm/out			
Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_test.0.S.o.log Processed instruction count : 198 Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x00000008000013c Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000e1c) addr:0x000000080000140 Mismatch[3]: [45] ibex : addi x4, x0, 0 -> tp(0x000000000 addr:0x8000009c [45] ovpsim : mul a3,a2,58 -> a3(0x00000000) addr:0x80000008d [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x80000008d [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x80000008d [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x80000008d [47] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x80000008d [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x800000008d [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x800000000000000000000000000000000000	<pre>Processing ovpsim log : /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_test.0.S.o.log Processed instruction count : 198 Processed instruction count : 6775 Mismatch[1: [43] ibex :</pre>	Test: /home/simond/g	it/ibex/dv/uvm/out/instr_gen/asm_tests/riscv_instr_base_test.0.S			
<pre>Processed instruction count : 198 Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x000000008000013c Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000ae1c) addr:0x0000000000000000000000000000000000</pre>	Processed instruction count : 198 Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x000000000000000 Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000e1c) addr:0x0000000000000000000000000000000000	Processing ovpsim lo	g : /home/simond/git/ibex/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_t	est.0.S.	o.log	
<pre>Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x000000008000013c Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000ae1c) addr:0x000000000000000000000000000000000 Mismatch[3]: [45] ibex : addi x4, x0, 0 -> tp(0x00000000000000000000 [45] ovpsim : mul a3,a2,s8 -> a3(0x00000000) addr:0x00000000000000000 [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x800000a8 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x0000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x80000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x0000000000000000000000000000000000</pre>	<pre>Processing ibex log : /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_00_0.log Processed instruction count : 6775 Mismatch[1: [43] ibex :</pre>	Processed instruction	i count : 198			
<pre>Processed instruction count : 6775 Mismatch[1]: [43] ibex :</pre>	Processed instruction count : 6775 Mismatch[1]: [43] ibex : lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088 [43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x00000008000013c Mismatch[2]: [44] ibex : addi x1, x1, 631 -> ra(0xfc2e4277) addr:0x8000008c [44] ovpsim : addi sp,sp,-800 -> sp(0x8000ae1c) addr:0x0000000000 addr:0x80000096 [45] ibex : addi x4, x0, 0 -> tp(0x000000000) addr:0x80000096 [45] ovpsim : mul a3,a2,s8 -> a3(0x00000000) addr:0x0000000080000148 Mismatch[4]: [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x800000a8 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x0000000000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x80000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x0000000080000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch ∂ tests PASSED, 1 tests FAILED simond@shell-1:~/git/ibex/dv/uvm\$_	Processing ibex log	: /home/simond/git/ibex/dv/uvm/out/rtl_sim/riscv_instr_base_test.0/trace_core_	00_0.log	8	
<pre>Mismatch[1]: [43] ibex :</pre>	<pre>Mismatch[1]: [43] ibex :</pre>	Processed instruction	i count : 6775			
<pre>[43] ibex :</pre>	<pre>[43] ibex :</pre>	Mismatch[1]:				
<pre>[43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x0000000000000000000000000000000000</pre>	<pre>[43] ovpsim : auipc sp,0xb -> sp(0x8000b13c) addr:0x0000000000000000000000000000000000</pre>	[43] ibex :	lui x1, 0xfc2e4000 -> ra(0xfc2e4000) addr:0x80000088			
<pre>Mismatch[2]: [44] ibex :</pre>	<pre>Mismatch[2]: [44] ibex :</pre>	[43] ovpsim : auipc	sp,0xb -> sp(0x8000b13c) addr:0x00000008000013c			
<pre>[44] ibex :</pre>	<pre>[44] ibex :</pre>	Mismatch[2]:				
<pre>[44] ovpsim : add1</pre>	<pre>[44] ovpsim : add1</pre>	[44] ibex :	addi x1, x1, 631 -> ra(0x+c2e4277) addr:0x8000008c			
<pre>Mismatch[3]: [45] ibex :</pre>	<pre>Mismatch[3]: [45] ibex : addi x4, x0, 0 -> tp(0x00000000) addr:0x80000096 [45] ovpsim : mul a3,a2,s8 -> a3(0x000000000) addr:0x000000080000148 Mismatch[4]: [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x800000a8 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x000000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x0000000080000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 8 tests PASSED, 1 tests FAILED simond@shell-1:~/git/ibex/dv/uvm\$</pre>	[44] ovpsim : addi	sp,sp,-800 -> sp(0x8000ae1c) addr:0x0000000000000140			
<pre>[45] iDeX : add1 X4, X0, 0 -> tp(0X000000000 addr:0X80000000 [45] ovpsim : mul a3,a2,s8 -> a3(0X000000000) addr:0X0000000080000148 Mismatch[4]: [46] ibex : lui x9, 0X81783000 -> s1(0X81783000) addr:0X800000a8 [46] ovpsim : auipc s2,0x0 -> s2(0X8000014c) addr:0X000000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0X81783559) addr:0X800000ac [47] ovpsim : addi s2,s2,986 -> s2(0X80000526) addr:0X0000000080000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 0 tests PASSED, 1 tests FAILED simond@shell-1:~/eit/ibex/dy/uvm\$ _</pre>	<pre>[45] 10eX : add1 X4, X0, 0 -> tp(0X000000000 addr:0X80000000 [45] ovpsim : mul a3,a2,s8 -> a3(0X000000000) addr:0X0000000080000148 Mismatch[4]: [46] ibex : lui x9, 0X81783000 -> s1(0X81783000) addr:0X800000a8 [46] ovpsim : auipc s2,0X0 -> s2(0X8000014c) addr:0X000000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0X81783559) addr:0X800000ac [47] ovpsim : addi s2,s2,986 -> s2(0X80000526) addr:0X0000000080000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 0 tests PASSED, 1 tests FAILED simond@shell-1:~/git/ibex/dv/uvm\$ _</pre>	Mismatch[3]:				
<pre>[45] ovpsim : mul</pre>	<pre>[45] ovpSim : mul</pre>	[45] 1Dex :	add1 X4, X0, 0 -> tp(0X000000000140) addr:0X80000096			
<pre>Mismatch[4]: [46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x800000a8 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x000000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x0000000080000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 0 tests PASSED, 1 tests FAILED simond@shell-1:~/eit/ibex/dv/uvm\$ _</pre>	[46] ibex : lui x9, 0x81783000 -> s1(0x81783000) addr:0x800000a8 [46] ovpsim : auipc s2,0x0 -> s2(0x8000014c) addr:0x000000008000014c Mismatch[5]: addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x000000080000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch θ tests PASSED, 1 tests FAILED simond@shell-1:~/git/ibex/dv/uvm\$	[45] OVPSIM : MUI	a3,a2,58 -> a3(0x00000000) auur:0x000000000000000148			
<pre>[46] lbcx . lul x9, 0xel/85060 -> S1(0xel/85060 addr:0x80606088 [46] ovpsim : auipc s2,0x0 -> S2(0x806004c) addr:0x0000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x000000080000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 0 tests PASSED, 1 tests FAILED simond@shell-1:~/eit/ibex/dy/uvm\$ _</pre>	<pre>[46] lbcx . lul x9, 0x1/35000 -> S1(0x1/35000) addr.0x800000aa [46] ovpsim : auipc s2,0x0 -> S2(0x8000014c) addr:0x00000008000014c Mismatch[5]: [47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x000000080000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 0 tests PASSED, 1 tests FAILED simond@shell-1:~/git/ibex/dv/uvm\$ _</pre>	MISMALCH[4]:	$100 \times 0^{-0} \times 0^{-$			
<pre>[40] 00pSim : addpc = 52,000 = 7 52(00000014C) addr.00000000000000000000000000000000000</pre>	<pre>[40] 00pSim : addpc = 52,000 = 7 52(00000014C) addr.00000000000000000000000000000000000</pre>	[40] IDEX .	101 X3, 0X01/05000 -/ S1(0X01/05000) 4001.0X00000006			
<pre>[47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x0000000000000000000000000000000000</pre>	<pre>[47] ibex : addi x9, x9, 1369 -> s1(0x81783559) addr:0x800000ac [47] ibex : addi s2,s2,986 -> s2(0x80000526) addr:0x0000000080000150 Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 0 tests PASSED, 1 tests FAILED simond@shell-1:~/git/ibex/dv/uvm\$ _</pre>	Micmatch[5]	32,000 -7 32(0,00000141) 8001.00000000000000141			
<pre>[47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x0000000000000000000000000000000000</pre>	<pre>[47] ovpsim : addi s2,s2,986 -> s2(0x80000526) addr:0x0000000000000000000000000000000000</pre>	[47] ihex :	addi x9 x9 1369 -> s1(0x81783559) addr:0x800000ac			
Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 0 tests PASSED, 1 tests FAILED simond@shell-1:~/git/ibex/dy/uvm\$	Compare (ibex vs ovpsim) result[FAILED]: 43 matched, 64 mismatch 0 tests PASSED, 1 tests FAILED simond@shell-1:~/git/ibex/dv/uvm\$ _	[47] ovpsim : addi	52.52.986 -> 52(0x8000526) addr:0x000000080000150			
0 tests PASSED, 1 tests FAILED simond@shell-1:~/git/ibex/dv/uvm\$	0 tests PASSED, 1 tests FAILED simond@shell-1:~/git/ibex/dv/uvm\$ _	Compare (ibex vs ovp	sim) result[FATLED]: 43 matched, 64 mismatch			
simond@shell-1:~/git/ibex/dv/uvm\$	simond@shell-1:~/git/ibex/dv/uvm\$ _	0 tests PASSED, 1 te	sts FAILED			
		simond@shell-1:~/git	/ibex/dv/uvm\$			





And... what about functional coverage you may ask...

- Sneak preview...
- New Open Source SystemVerilog UVM Coverage component added to flow
 - Adds coverage groups / bins in UVM testbench
 - Configure coverpoints based on DUT, eg RV32IMC, RV32IMAFDC, ...
- Post processes a test's execution trace
- Collates coverage from many runs
- Uses SystemVerilog UVM simulator's coverage analysis to report, review
 - Works with Cadence, Mentor, Synopsys, Metrics



Mentor Questa Coverage views

🗧 Covergroups :=====											
Name		Class Type	Coverage	Goal	% of Goal	Status Inclu	ded Merge_instanc	es Get_in:	st_coverage		
/riscv_instr_pkg/rise	cv_instr_cover_group		76.24%								
TYPE add_cg		riscv_instr_co	100.00%	100	100.00%	V	8	auto(1)			
TYPE addi co		riscy_instr_co	99.55%	100	99 55%		*	auto(1)		1	
TYPE lui cg		riscv instr co	80.00%	100	80.00%			auto(1)			
💽 🗾 TYPE auipc_cg		riscv_instr_co	80.00%	100	80.00%		a	auto(1)			
😨 🗾 TYPE sra_cg		riscv_instr_co	100.00%	100	100.00%	×.	e	auto(1)			
TYPE SIL_CG		riscv_instr_co	100.00%	100	100.00%		6	auto(1)			
TYPE srai co		riscv_instr_co	0.00%	100	0.00%	ľ,	-	auto(1)			
😟 🗾 TYPE sili_cg		riscv_instr_co	100.00%	100	100.00%	Image: A state of the state	ā	auto(1)			
🔁 🗾 TYPE srli_cg		riscv_instr_co	100.00%	100	100.00%	\checkmark	ē	auto(1)			
TYPE xor_cg		riceu inetr on	100.0004	100	100.0044			uito/13			
TYPE and co	Covergroups				77		9/1 		les constil		
TYPE xori_cg	* Name					Class Type	Coverage	Goal	% of Goal	Status	Included
🗄 🗾 TYPE ori_cg	Instr_pkg/riscv_instr_pkg/riscv_inst	r_cover_grou	р				76.249				
TYPE andi_cg	🚊 🗾 TYPE add_cg					riscv_instr_co.	100.00%	100	100.00%		
TYPE situ cg	🕀 🗾 CVP add cg::cp rs	1				riscv instr co.	. 100.00%	100	100.00%		
🗉 🗾 TYPE siti_cg	🕀 🗾 CVP add cg::cp rs	2				riscv instr co.	. 100.00%	100	100.00%		
TYPE sltiu_cg	🕀 🗾 CVP add cg::cp ro	1				riscv instr co.	. 100.00%	100	100.00%		
TYPE bne_cg	💽 🗾 CVP add cg::cp rs	1 sign				riscv instr co.	100.00%	100	100.00%		
😟 🗾 TYPE blt_cg	+ CVP add_cg::cp_rs	2_sign				riscv_instr_co.	100.00%	100	100.00%		
TYPE bltu cg	🗄 🗾 CVP add_cg::cp_rd	t_sign				riscv_instr_co.	100.00%	100	100.00%		
🕣 🗾 TYPE bgeu_cg	🛨 🗾 CVP add_cg::cp_g	pr harzard				riscv instr co.	100.00%	100	100.00%		
TYPE Ib_cg	🕁 🗾 CROSS add_cg::cp	_sign_cross				riscv_instr_co.	. 100.00%	100	100.00%		
TYPE IW_cg	🔄 🗾 TYPE sub_cg					riscv_instr_co.	. 100.00%	100	100.00%		
TYDE Ibu og	😟 🗾 TYPE addi_cg					riscv_instr_co.	. 99.55%	100	99.55%		
	🔁 🗾 TYPE lui_cg					riscv_instr_co.	. 80.00%	100	80.00%		
	🕒 🗾 TYPE auipc_cg					riscv_instr_co.	. 80.00%	100	80.00%		
	🔄 🗾 TYPE sra_cg					riscv_instr_co.	. 100.00%	100	100.00%		
	🔁 🗾 TYPE sll_cg					riscv_instr_co.	100.00%	100	100.00%		
	🕒 🗾 TYPE srl_cg					riscv_instr_co.	. 100.00%	100	100.00%		
	🕒 🗾 TYPE srai_cg					riscv_instr_co.	. 0.00%	100	0.00%		
	🔄 🗾 TYPE slli_cg					riscv_instr_co.	. 100.00%	100	100.00%		
	连 🗾 TYPE srli_cg					riscv_instr_co.	. 100.00%	100	100.00%		
	🕒 🗾 TYPE xor_cg					riscv_instr_co.	. 100.009	100	100.00%		
	🔁 🗾 TYPE or_cg					riscv_instr_co.	. 100.00%	100	100.00%		
	🔁 🗾 TYPE and_cg					riscv_instr_co.	. 100.00%	100	100.00%		
	🕀 🗾 TYPE xori_cg					riscv_instr_co.	. 100.00%	100	100.00%		
	🕒 🗾 TYPE ori cg					riscv instr co.	. 100.00%	100	100.00%		

ararount -											
ergroups .		Class Tumo	Coverage	Goal	% of Goal	Status	Included	Morgo inc	tancos	Got inst coup	
e kiesu instrukteriesu instruccus	r 97000	Class Type	Coverage 76 2406	Ouai	70 01 00as	Status	Included	merge_ma	sances	Ger_inst_cove	
TYPE add co	r_group	riscy instr co	100.00%	100	100.00%		1		auto/1	n.	
CVP add co::cp rs1		riscy instr co	100.00%	100	100.00%		1		unio(x	<i>.</i> ,	
CVP add cg::cp_rs2		riscv instr co	100.00%	100	100.00%		1				
🗄 🗾 CVP add_cg::cp_rd		riscv_instr_co	100.00%	100	100.00%		1				
💿 🗾 CVP add_cg::cp_rs1_sigr	1	riscv_instr_co	100.00%	100	100.00%		1				
CVP add_cg::cp_rs2_sigr	1	riscv_instr_co	100.00%	100	100.00%		√				
CVP add_cg::cp_rd_sign		riscv_instr_co	100.00%	100	100.00%		1				
B bin auto[POSITIVE]			7176	1	100.00%		1				
B bin autoiNEGATIVE	tord	ricov instr oo	100 00%	100	100.00%		1				
B) bin auto[NO_HAZARD]	zaiu	hscv_hsu_co	10832	100	100.00%		1				
B) bin auto[RAW_HAZAR]	DI		728	1	100.00%		1				
B bin auto[W/ Covera				n 17							
B) bin auto[W/	neups :			Close	Turne.	Courseage	Cool	04 of Cool	Ctature	Included	Morgo instanc
CROSS add_				Ciass	Type	Coverage	OUai	70 01 00al	Jaalus	includeu	merge_instanc
-B) bin <auto[p is<="" td="" ⊒=""><td>cv_instr_pkg/riscv_instr_cover</td><td>_group</td><td></td><td></td><td></td><td>76.249</td><td></td><td>100.001</td><td></td><td></td><td></td></auto[p>	cv_instr_pkg/riscv_instr_cover	_group				76.249		100.001			
-B) bin <auto[n td="" 💻<="" 🖃=""><td>TYPE add_cg</td><td></td><td></td><td>riscv_i</td><td>nstr_co</td><td>100.009</td><td>6 10</td><td>0 100.009</td><td>10</td><td></td><td></td></auto[n>	TYPE add_cg			riscv_i	nstr_co	100.009	6 10	0 100.009	10		
B bin <auto[p< td=""><td>CVP add_cg::cp_rs1</td><td></td><td></td><td>riscv_i</td><td>nstr_co</td><td>100.009</td><td>10</td><td>0 100.004</td><td>10</td><td>V.</td><td></td></auto[p<>	CVP add_cg::cp_rs1			riscv_i	nstr_co	100.009	10	0 100.004	10	V.	
B) bin cauto[0	CVP add_cg::cp_rs2			riscv_i	nstr_co	100.009	6 10	0 100.009	10		
B) bin <auto[p< td=""><td>CVP add_cg::cp_rd</td><td></td><td></td><td>riscv_i</td><td>nstr_co</td><td>100.009</td><td>0 10</td><td>J 100.009</td><td>20</td><td>V</td><td></td></auto[p<>	CVP add_cg::cp_rd			riscv_i	nstr_co	100.009	0 10	J 100.009	20	V	
B) bin <autofp< td=""><td>B bin auto[ZERO]</td><td></td><td></td><td></td><td></td><td>32</td><td>9</td><td>1 100.009</td><td>Pb</td><td></td><td></td></autofp<>	B bin auto[ZERO]					32	9	1 100.009	Pb		
B) bin <auto[n< td=""><td>B bin auto[RA]</td><td></td><td></td><td></td><td></td><td>31</td><td>3</td><td>1 100.009</td><td>10</td><td>V</td><td></td></auto[n<>	B bin auto[RA]					31	3	1 100.009	10	V	
TYPE sub_cg	b) bin auto[SP]					31		1 100.009	10	× .	
TYPE addi_cg	B) bin auto[GP]					40	/ 	1 100.009	10		
TYPE lui_cg	B) bin auto[1P]					45	3	1 100.004	10	× .	
TYPE auipc_cg	Din auto[10]					34	0	100.009	10	×.	
TYPE sra_cg	B) bin auto[11]					38	9	1 100.009	PO	× 1	
TYPE SI_CU	B) bin auto[12]					42	5	100.009	10	× .	
TYPE srai co	B) bin auto[50]					40.	5	1 100.009	20	× 1	
auto(B) bin auto[31]					40	9	1 100.009	10		
outo(B) bin auto[A1]					40.	2	1 100.005	16		
auto(.	B) bin auto[A2]					37	7	1 100.009	16		
auto(B) bin auto[A3]					39	6	1 100.009	V6		
auto(B) bin auto[A4]					35	2	1 100.009	h		
suto/	B) bin auto[A5]					41	2	1 100.009	16		
duto(.	B) bin auto[A6]					35	R	1 100.009	6		
auto(.	B) bin auto[A7]					38	7	1 100.009	6		
auto(B) bin auto[S2]					42	9	1 100.009	10		
auto(B) bin auto[S3]					38	2	1 100.009	16		
auto(B) bin auto[S4]					36	в	1 100.009	16	×	
auto(.	B) bin auto[S5]					34	в	1 100.009	16		
auto(B bin auto[S6]					44	2	100.009	10		
auto(B) bin auto[S7]					32	3	1 100.009	96	1	
auto	B) bin auto[S8]					39	9	1 100.009	16		
autol.	B) bin auto[S9]					41	4	1 100.009	16		
auto(.	B) bin auto[S10]					38	2	1 100.009	10	Image: A start of the start	
auto(B) bin auto[S11]					44	В	1 100.009	10		
auto(B) bin auto[T3]					41	5	1 100.009	16	1	
X	B) bin auto[T4]					41	5	100.009	16		
	B) bin auto[T5]					35-	4	1 100.009	16	\checkmark	
	B] bin auto[T6]					31	2	1 100.009	16	\checkmark	
	CVP add_cg::cp_rs1_sign			riscv_i	nstr_co	100.009	6 10	0 100.009	16		
-	nnia Car norma hhe GVO			riscy in	nstr co	100.009	6 10	1 100 000	6		





▼ Nam

Agenda

- RISC-V Intro and Status
- Industry requirements for ISA Compliance and Hardware Design Verification
- RISC-V ISA Compliance
- Processor Hardware Design Verification
- RISC-V Instruction Stream Generation
- Hardware Design Verification Flow
- Demonstration walk-through
- Scaling verification using Cloud resources





Metrics Cloud Platform makes it all much simpler...

metrics





• Complete solution for DV







Metrics: integrated GitLab

	https://gitlab.demo.metrics.ca/lowRISC/	/ibex	©	0 1	4	111	
itLab Projects - Groups Ac	tivity Milestones Snippets 🏼 🎢	•	 This project Sear 	ch 9, 07	n	ല	۰ 🕲
l ibex	lowRISC > lbex > Details						
Overview							
Details							
Activity		ibex v					
Cycle Analytics	lbex is a sm	nall 32 bit RISC-V CPU core (RV32IMC/EMC) with a two stage pipelin riscy. Forked from: https://github.com/lowRISC/ibex	e, previously known as	zero-			
Repository	☆ Star 0 Y Fo	ork 0 SSH = git@gitlab.demo.metrics.ca:low	6 • • +	* 🌲 Global *			
() Issues (0)	Files (3 MB) Commits (947)	Branches (2) Tags (4) Readme Apache License 2.0 Contribut	ion guide CI configur	ation Add Change	log		
Ŋ Merge Requests (0)	master v ibex	¢/ + ~	History	Q, Find file	-		
Ci/CD	d [¶] *•4 [™] *•4 [™] * [™] Aimee Sutton (Metrics	nd for demo re:test status s AE) authored 2 days ago		⊙ ff474d4e	ю		
👗 Snippets	Name	Last commit		Last upda	ite		
O Settings	M doc	Merge branch 'master' of https://github.com/lowRISC/i.		2 days a	90		
	dv/uvm	Merge branch 'master' into metrics-integration		2 days a	90		
	metrics	Removed work-around for demo re:test status		2 days a	ço		
<< Collapse sidebar	🖿 rti	Added a covergroup/instruction coverage		3 days a	90		

metrics







Metrics: built in Continuous Integration to run jobs

metrics

€ → ଫ ŵ	🛈 🔒 http	s://gitlab.demo.metrics.c	a/lowRISC/lbex/pipelines			🖂 🕁	\mathbf{F}	III\ 🖸 🗄
🖊 GitLab 🛛 Projects 🗸	Groups Activity	Milestones Snippets	1		# ~ m	is project Search	9. D7 11. (s 🌚-
l ibex	IdwRISC > Ibex	Pipelines						
Overview	All 7 P	Pending 0 Running	0 Finished 4 Branches Tags			Run Pipeline	Clear runner caches	CI Lint
Repository	Status	Pipeline	Commit	Stages				
Registry Issues (0)	skipped	#174 by 🐴 Tatest	$\begin{array}{l} & \underbrace{\forall metrics-integr.}_{q_{12}^{(2)}} \leftrightarrow d7c82e7b \\ & \underbrace{\forall}_{q_{12}^{(2)}}^{(2)} & Updates to functional coverage col \end{array}$	۲				þ
Merge Requests	i skipped	#173 by 🎄	$\begin{array}{l} \forall \text{metrics-integr.} \Leftrightarrow \text{ff474d4e} \\ \overset{\text{GP}}{_{2,3}} \text{ Removed work-around for demo re} \end{array}$	۲				þ
Pipelines Jobs	(skipped	#172 by 🎄 Intest		۲				þ
Schedules Environments	() failed	#171 by 🚋 latest	<pre>> pulpissimo-vl ← d4d46971</pre>			⊘ 00:00:09 ∰ a day ago		C
Clusters Charts	() failed	#170 by 排 Natest	◆ pulpissino-v1 ~ 71cb9878 update headers	۲		 ⊘ 00:00:10 m a day ago 		C
🖱 Wiki	() failed	#169 by 👍 latest	> pulpissino-v1.0 ↔ b99e74ef Removed non-ASCII characters	۲		⊘ 00:00:06 ∰ a day ago		c
★ Collapse sidebar	() failed	#168 by 掛 latest	♥ pulpino-v1.0.0 -> 38934581 # fixed fetch fifo [broken after last co	۲		⊘ 00:00:07 ∰ a day ago		C

• Push of a button ('Run Pipeline') to run jobs (that we ran from shells earlier)





Metrics: browse access to all results

metrics

imetrics era bit Result	ts GitLab	Settings			Admin	Help / Feedback	۲	Metrics
st Runs								
Show Archived Results		Info	Time	Coverage		Artifacts	Rank	
2 Filters Added		riscv_instr_base_test_2019-06-14_12-21-30_1793695	00:12:12	15.68%	83.33%	Log 🖙	24	
Regression Run ID == x riscv_instr_base_regr_2019-06-14_12		5ee0: 20884	Jun 14 19 12:06	functional	assertion	wave 12		
Status == x Passed		 riscv_instr_base_test_2019-06-14_12-21-32_13296344 Seed: 12987 	00:13:40 Jun 14 '19 12:06	9.8% functional	83.33% assertion	Log ය Wave ය	18	
Status •	Prev	I: Next				ltems per page	50 •	
== • Running •								
Add Filter								

• Showing test runs and summary coverage



Metrics: can drill down into details

metrics

::: metrics "CTA		Help / Feedback
25 logs selected	Test run riscy_instr_base_test_2019-06-14_12-21-14_13460266 an hour ago	
Q. FAILED (*) Aa 48 results in 25 logs Clear Log riscv. instr_base_test_2019-06-14_12-21? 2 950 Compare result(FAILED): 86 matched, 17 951: 0 tests PASSED, 1 tests FAILED Log riscv_instr_base_test_2019-06-14_12-21?	<pre>03/329 =1:5:mulation terminated by 5:mish at time 100000000 (/mLx-it0w/buit0/10ex/dv/uvm/tb/core_ibex_tb_top.sv:sl); 03/330 Run directory: /mux-flow/build/repo/dv/uvm/out/rtl_sim/riscv_instr_base_test.0 03/331 System timescale is l0ps / l0ps 03/332 DSim version: 20190115.17.0 (b:S *c:125 h:bab967811) 03/333 Random seed: 12100 03/333 Test: /mux-flow/build/repo/dv/uvm/out/instr_gen/asm_tests/riscv_instr_base_test.0.S 03/336 Processing spike log : /mux-flow/build/repo/dv/uvm/out/instr_gen/spike_sim/riscv_instr_base_test.0.S.o.log 03/337 Processing ovpsim log : /mux-flow/build/repo/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_test.0.S.o.log 03/338 Processing ibex log : /mux-flow/build/repo/dv/uvm/out/instr_gen/riscv_ovpsim/riscv_instr_base_test.0.S.o.log 03/338 Processing ibex log : /mux-flow/build/repo/dv/uvm/out/instr_gen/riscv_instr_base_test.0.S.o.log 03/339 Processing ibex log : /mux-flow/build/repo/dv/uvm/out/instr_gen/riscv_instr_base_test.0.S.o.log</pre>	
S03 Compare result[FAILED]: 85 matched, 14 804 0 tests PASSED, 1 tests FAILED Log	637330 Processing lock (og , how room)abroarter epoyov and outrice improver instruction absector (accector cog 637340 Mismatch[1]: 637344 [62] ibex : [w x10, 12(x2) x1=80080000 x1=80080000 x2=80098000 x3=80080000 x3=80080000 x3=80080000 x3=800800000 x3=8008000000000000000000000000000000000	98 x4=00000000
riscv_instr_base_test_2019-06-14_12-21 ² 429 Compare result[FAILED]: 87 matched, 10 430 0 tests PASSED, 1 tests FAILED	637342 [62] ovpsin : mv s7,s4 → s7(0x88000000) addr:0x80000002f6 637343 Mismatch[2]:	99 x4≈06606609
 Log riscv_instr_base_test_2019-06-14_12-21² 637356 Compare result[FAILED]: 96 matches 637357 0 tests PASSED, 1 tests FAILED 	637345 [63] ovpsin : andi a3,a3,6 -> a3(0x000000000) addr:0x0000000000000000000000000000000000	99 x4=00000009
Log riscv_instr_base_test_2019-06-14_12-21	637350 [65] ibex : sra x29, x4, x0 x1=00000009 x1=00000009 x2=8000aelc x3=00000009 x3=0000000 637351 [65] ovpsin : and a1,a1,s0 -> a1(8x00000000) addr:ex0000000008a	x4=00000009
Log riscv_instr_base_test_2019-06-14_12-212	637352 Mismatch[5]: 637353 [66] 1bex : add x14, x0, x9 x1=00000009 x1=00000009 x2=8000ae1c x3=00000009 x3=0000000	y9 x4=86086889
Log riscv_instr_base_test_2019-06-14_12-212 Log riscv_instr_base_test_2019-06-14_12-212	C37354 [66] ovpsin : sub s1,51,00 → s1(0xff9fe700) addr:0x0000000000000000000000000000000000	

• For example see the saved logs – so can see all detail of run





Metrics: can show functional coverage

metrics

iii metrics BETA	Results GitLab	Settings	Ad	lmin Help / Feedback	Metrics PE
riscv_instr_base_regr_2019-0 Coverage Merged Sources	06-16_15-17-59 / Fur	ctional Coverage			
▼CI instr_cov.0	84.31%	100% 1 /mux-flow/bu	ild/ibex/rtl/ibex_tracer.sv:110		
CP instruction	84.31%	100% 1			
BIN instr_lui	2913	101			
BIN instr_auipc	875	101			
BIN instr_jal	531	101			
BIN instr_jalr	463	101			
BIN instr_beq	1180	101			
BIN instr_bne	554	101			
BIN instr_blt	120	101			
BIN instr_bge	183	101			
PIN instr bltu	152	101			







Metrics: can even see detailed contribution of each test including functional coverage

metrics 🕬 🖿 ibex Results GitLab Settin	25		Admin	Help / Feedback	Metrics PE
cov_instr_base_regr_2019-06-14_10-22-13 / Functiona	l Coverage				
	ID	Functional Contribution			
	testRuns/riscv_instr_base_test_2019-06-14_10-33-40_12086081	46.07%			
	testRuns/rlscv_instr_base_test_2019-06-14_10-33-22_15784441	4.9%			
46.07	testRuns/riscv_instr_base_test_2019-06-14_10-32-53_3736851	3.92%			
	testRuns/riscv_instr_base_test_2019-06-14_10-33-20_3355917	2.95%			
	testRuns/riscv_instr_base_test_2015-06-14_10-33-50_1220507	2.94%			
	testRuns/riscv_instr_base_test_2019-06-14_10-33-57_10581450	2.93%			
	testRuns/riscv_instr_base_test_2019-06-14_10-33-32_8076881	1.96%			
	testRuns/riscv_instr_base_test_2019-06-14_10-33-25_13658844	1.96%			
	testRuns/riscv_instr_base_test_2019-06-14_10-32-34_741142	1.96%			
	testRuns/rhscv_instr_base_test_2019-06-14_10-34-32_15594444	0.99%			



metrics



TÎS

Metrics: includes top level overview dashboard



• Allows management overview of status of verification







Status

- The integrated DV infrastructure from Google, Metrics, Imperas is a work in progress
- Clearly shows direction and focus





Summary

• DV solution for RISC-V RTL cores



- Collaboration between Google, Imperas and Metrics
- Industry adoption has started
- The basis for RISC-V core verification in:











Questions



