Joshua Monson  
ECEN 620   
Assignment 9

**Overview:**

The purpose of this assignment was to improve our **risc\_rpm** processor test bench by adding a cover group with several cover points and cross cover points that gather data to help us understand the functionality that we’ve tested. The coverpoints that were created for this homework help us understand if we’ve thoroughly tested all instructions. For example, the covergroup helps us determine if we’ve tested all source and destination register combinations with all instructions that have source and destination registers. Additionally, the covergroup tells us when we have tested a variety of sequences of instructions and read and written all memory addresses for the read and write instructions.

The remaining contents are:

1. System Verilog Code (that was modified since the last assignment)
2. Full Coverage Report
3. Coverage Report showing Partial Coverage

**SystemVerilog Code:**

**test.sv**

import TbEnvPkg**::\*;**

Instruction cur\_inst**;**

covergroup Cov**();**

opcodes**:**coverpoint cur\_inst**.**opcode **{**

//1. All opcodes have been executed.

bins ops**[]** **=** **{[**0**:**6**],** 9**};**

illegal\_bins bad\_ops**[]** **=** **{[**7**:**8**],[**10**:**15**]};**

//4. All opcodes have been preceeded and followed all other opcodes

bins op\_order**[]** **=** **(**0**,**1**,**2**,**3**,**4**,**5**,**6**,**9**=>**0**,**1**,**2**,**3**,**4**,**5**,**6**,**9**);**

**}**

srcs**:**coverpoint cur\_inst**.**src **{**

bins sregs**[]** **=** **{[**0**:**3**]};**

**}**

dsts**:**coverpoint cur\_inst**.**dst **{**

bins dregs**[]** **=** **{[**0**:**3**]};**

**}**

//2. The source for every opcode that has a source has been 0,1,2,3

opcode\_src**:**cross opcodes**,** srcs **{**

ignore\_bins no\_src **=** binsof**(**opcodes**.**ops**)** intersect **{**0**,**5**,**9**};**

ignore\_bins trans **=** binsof**(**opcodes**.**op\_order**);**

**}**

//3. The destination for every opcode that has a dst has been 0,1,2,3

opcode\_dst**:**cross opcodes**,** dsts **{**

ignore\_bins no\_dst **=** binsof**(**opcodes**.**ops**)** intersect **{**0**,**6**,**9**};**

ignore\_bins trans **=** binsof**(**opcodes**.**op\_order**);**

**}**

//5. Opcodes with both source and destination have had all combs of srcs and dsts

opcode\_src\_dst**:**cross opcodes**,** srcs**,** dsts **{**

ignore\_bins no\_src\_dst **=** binsof**(**opcodes**.**ops**)** intersect **{**0**,**5**,**6**,**9**};**

ignore\_bins trans **=** binsof**(**opcodes**.**op\_order**);**

**}**

addresses**:** coverpoint cur\_inst**.**byte1 iff **(**cur\_inst**.**opcode **!=** NOP **&&**

cur\_inst**.**opcode **!=** ADD **&&**

cur\_inst**.**opcode **!=** SUB **&&**

cur\_inst**.**opcode **!=** AND **&&**

cur\_inst**.**opcode **!=** NOT **&&**

cur\_inst**.**opcode **!=** HALT**){**

bins addrs**[]** **=** **{[**0**:**255**]};**

**}**

WrRdRdiMem**:**cross opcodes**,** addresses **{**

//6 & 7: All mem locations written

ignore\_bins non\_read\_write **=** binsof**(**opcodes**.**ops**)** intersect **{**0**,**1**,**2**,**3**,**4**};**

ignore\_bins trans **=** binsof**(**opcodes**.**op\_order**);**

**}**

endgroup // Cov

class Driver\_cbs\_cov extends Driver\_cbs**;**

Cov ck**;**

**function** new**();**

ck**=**new**();**

**endfunction** // new

virtual **task** pre\_inst**(**ref Instruction I**);**

cur\_inst **=** I**;**

ck**.**sample**();**

**endtask** // pre\_inst

endclass // Driver\_cbs\_cov

program automatic test**(**SPM\_IF**.**TEST mem\_bus**);**

**initial** **begin**

static virtual SPM\_IF**.**TEST vMemBus **=** mem\_bus**;**

Driver\_cbs\_cov cov\_callback**;**

Environment E**;**

cov\_callback **=** new**();**

E **=** new **();**

E**.**build**(**vMemBus**);**

E**.**D**.**callbacks**.**push\_back**(**cov\_callback**);**

E**.**run**(**10000**);**

$display**(**"Test Complete found %d Errors."**,** E**.**D**.**Scb**.**ErrorCounter**);**

**end**

endprogram // test

**driver.sv**

class Driver\_cbs**;**

virtual **task** pre\_inst**(**ref Instruction I**);**

**endtask** // pre\_inst

virtual **task** post\_inst**(**ref Instruction I**);**

**endtask** // post\_inst

endclass // Driver\_cbs

class Driver**;**

virtual SPM\_IF**.**TEST dut\_if**;**

Driver\_cbs callbacks**[**$**];**

ScoreBoard Scb**;**

mailbox **#(**Instruction**)** mbx**;**

**event** handshake**;**

**function** new **(**virtual SPM\_IF**.**TEST dif**,**

mailbox **#(**Instruction**)** mb**,** **event** hs**);**

dut\_if **=** dif**;**

mbx **=** mb**;**

Scb **=** new **();**

handshake **=** hs**;**

**endfunction** // new

**function** automatic void initialize**();**

dut\_if**.**cb**.**rst **<=** 1**;**

dut\_if**.**cb**.**data\_out **<=** 8'h0**;**

**endfunction**

**task** automatic reset**();**

initialize**();**

**@**dut\_if**.**cb**;**

dut\_if**.**cb**.**rst **<=** 1**;**

**@**dut\_if**.**cb**;**

**@**dut\_if**.**cb**;**

dut\_if**.**cb**.**rst **<=** 0**;**

**repeat** **(**4**)** **@**dut\_if**.**cb**;**

dut\_if**.**cb**.**rst **<=** 1**;**

**repeat** **(**1**)** **@**dut\_if**.**cb**;**

**endtask**

**function** automatic string getOpcode**(**Instruction I**);**

**case(**I**.**byte0**[**7**:**4**])**

NOP**:** return "NOP"**;**

ADD**:** return "ADD"**;**

SUB**:** return "SUB"**;**

AND**:** return "AND"**;**

NOT**:** return "NOT"**;**

RD**:** return "RD"**;**

WR**:** return "WR"**;**

BR**:** return "BR"**;**

BRZ**:** return "BRZ"**;**

RDI**:** return "RDI"**;**

HALT**:** return "HALT"**;**

**endcase** // case (I.byte0[7:4])

**endfunction** // string

**task** automatic run**(**int count**);**

Instruction I**;**

int i **=** 0**;**

**repeat(**count**)** **begin**

mbx**.**get**(**I**);**

foreach**(**callbacks**[**i**])** callbacks**[**i**].**pre\_inst**(**I**);**

sendInstruction**(**I**,**i**);**

foreach**(**callbacks**[**i**])** callbacks**[**i**].**post\_inst**(**I**);**

**->**handshake**;**

i**++;**

**end** // repeat (count)

**if** **(**$root**.**top**.**DUT**.**Controller**.**state **==** 4'd11**)** **begin**

$display**(**"Error: Processor has halted"**);**

**end**

**endtask** // run

**task** automatic sendInstruction**(input** Instruction I**,** int i**);**

$display**(**"@%0d: Starting Instr #%0d :: %s"**,** $time**,** i**,** getOpcode**(**I**));**

//Fetch 1

Scb**.**fetch1**();**

**@**dut\_if**.**cb**;**

//Fetch 2: Get and Drive I.byte0

Scb**.**fetch2**(**I**);**

dut\_if**.**cb**.**data\_out **<=** I**.**byte0**;**

**@**dut\_if**.**cb**;**

//Decode:

Scb**.**check\_address**(**dut\_if**.**cb**.**address**);**

Scb**.**check\_pc**(**$root**.**top**.**DUT**.**Processor**.**PC\_count**);**

Scb**.**check\_ir**(**$root**.**top**.**DUT**.**Processor**.**instruction**);**

Scb**.**decode**(**I**,**8'hff**);**

**case** **(**I**.**byte0**[**7**:**4**])**

NOP**:** doNOP**(**I**);**

ADD**,** SUB**,** AND**:** doAddSubAnd**(**I**);**

NOT**:** doNot**(**I**);**

RD**,** RDI**:** doRdRdi**(**I**,**8'hff**);**

WR**:** doWr**(**I**);**

BR**,** BRZ**,** HALT**:** doBrBrzHalt**(**I**);**

**endcase** // case (I.byte0[7:4])

**endtask** // sendInstruction

**task** automatic doAddSubAnd**(input** Instruction I**);**

$display**(**"Decoded AddSubAnd Instr"**);**

//Leave Decode State

**@**dut\_if**.**cb**;**

//Leave S\_ext1

**@**dut\_if**.**cb**;**

Scb**.**check\_regs**(**$root**.**top**.**DUT**.**Processor**.**R0\_out**,**

$root**.**top**.**DUT**.**Processor**.**R1\_out**,**

$root**.**top**.**DUT**.**Processor**.**R2\_out**,**

$root**.**top**.**DUT**.**Processor**.**R3\_out**);**

**endtask** // AddSubAnd

**task** automatic doNOP**(input** Instruction I**);**

$display**(**"Decoded NOP"**);**

**@**dut\_if**.**cb**;**

**endtask** // doNOP

**task** automatic doNot**(input** Instruction I**);**

$display**(**"Decoded NOT"**);**

//Leave Decode State

**@**dut\_if**.**cb**;**

Scb**.**check\_regs**(**$root**.**top**.**DUT**.**Processor**.**R0\_out**,**

$root**.**top**.**DUT**.**Processor**.**R1\_out**,**

$root**.**top**.**DUT**.**Processor**.**R2\_out**,**

$root**.**top**.**DUT**.**Processor**.**R3\_out**);**

**endtask** // doNot

**task** automatic doRdRdi**(input** Instruction I**,** byte rdval**);**

$display**(**"Decoded RdRdi Instr"**);**

//Leave Decode State

**@**dut\_if**.**cb**;**

dut\_if**.**cb**.**data\_out **<=** I**.**byte1**;**

**if(**I**.**byte0**[**7**:**4**]** **==** RD**)** **begin**

//Leave RD1

**@**dut\_if**.**cb**;**

dut\_if**.**cb**.**data\_out **<=** rdval**;**

Scb**.**check\_address**(**dut\_if**.**cb**.**address**);**

//Leave RD2

**@**dut\_if**.**cb**;**

Scb**.**check\_address**(**dut\_if**.**cb**.**address**);**

**end**

**else** **begin** //RDI

//Leave RD1

**@**dut\_if**.**cb**;**

Scb**.**check\_address**(**dut\_if**.**cb**.**address**);**

**end** // else: !if(I.byte0[7:4] == RD)

Scb**.**check\_regs**(**$root**.**top**.**DUT**.**Processor**.**R0\_out**,**

$root**.**top**.**DUT**.**Processor**.**R1\_out**,**

$root**.**top**.**DUT**.**Processor**.**R2\_out**,**

$root**.**top**.**DUT**.**Processor**.**R3\_out**);**

**endtask** // doRdRdi

**task** automatic doWr**(input** Instruction I**);**

$display**(**"Decoded Wr"**);**

//Leave Decode State

**@**dut\_if**.**cb**;**

//Leave WR1

dut\_if**.**cb**.**data\_out **<=** I**.**byte1**;**

**@**dut\_if**.**cb**;**

Scb**.**check\_address**(**dut\_if**.**cb**.**address**);**

//Leave WR2

**@**dut\_if**.**cb**;**

Scb**.**check\_address**(**dut\_if**.**cb**.**address**);**

Scb**.**check\_dataIn**(**dut\_if**.**cb**.**data\_in**);**

**endtask**

**task** automatic doBrBrzHalt**(input** Instruction I**);**

$display**(**"Error: Encountered Opcode %d"**,** I**.**byte0**[**7**:**4**]);**

$display**(**"This Opcode is not yet supported by the Test Bench Env"**);**

$finish**;**

**endtask**

endclass**;**

**Generator.sv**

`define SV\_RAND\_CHECK**(**r**)** \

do **begin** \

**if** **(!(**r**))** **begin** \

$display**(**"%s:%0d: Randomization failed \"%s\""**,** \

`\_\_FILE\_\_**,** `\_\_LINE\_\_**,** `"r`"**);\**

$finish**;** \

**end** \

**end** **while(**0**)**

class Instruction**;**

rand bit **[**7**:**0**]** byte0**;**

rand bit **[**7**:**0**]** byte1**;**

bit **[**3**:**0**]** opcode**;**

bit **[**1**:**0**]** src**;**

bit **[**1**:**0**]** dst**;**

//Don't allow the Generator to Create BR, BRZ, or HALT Inst

constraint c\_byte0 **{** byte0**[**7**:**4**]** **!=** BR**;**

byte0**[**7**:**4**]** **!=** BRZ**;**

byte0**[**7**:**4**]** **!=** HALT**;**

byte0**[**7**:**4**]** **!=** 4'hA**;**

byte0**[**7**:**4**]** **!=** 4'hB**;**

byte0**[**7**:**4**]** **!=** 4'hC**;**

byte0**[**7**:**4**]** **!=** 4'hD**;**

byte0**[**7**:**4**]** **!=** 4'hE**;** **};**

//Don't Randomize byte1 if single byte inst

constraint c\_byte1 **{** **(** byte0**[**7**:**4**]** **==** NOP **||**

byte0**[**7**:**4**]** **==** ADD **||**

byte0**[**7**:**4**]** **==** SUB **||**

byte0**[**7**:**4**]** **==** AND **||**

byte0**[**7**:**4**]** **==** NOT **||**

byte0**[**7**:**4**]** **==** HALT**)** **->** **(**byte1 **==** 8'd0**);};**

**function** void post\_randomize**();**

opcode **=** byte0**[**7**:**4**];**

src **=** byte0**[**3**:**2**];**

dst **=** byte0**[**1**:**0**];**

**endfunction** // post\_randomize

endclass // Instruction

class Generator**;**

Instruction I**;**

mailbox **#(**Instruction**)** mbx**;**

**event** handshake**;**

**function** new **(**mailbox **#(**Instruction**)** m**,** **event** hs**);**

mbx **=** m**;**

I **=** new **();**

handshake **=** hs**;**

**endfunction** // new

**task** run**(**int count**);**

Instruction i**;**

**repeat(**count**)** **begin**

`SV\_RAND\_CHECK**(**I**.**randomize**());**

i**=**new I**;**

mbx**.**put**(**i**);**

**wait(**handshake**.**triggered**);**

**end**

**endtask**

endclass // Generator

**scoreboard.sv**

class ScoreBoard**;**

int ErrorCounter**;**

bit **[**7**:**0**]** pc**;**

bit **[**7**:**0**]** ir**;**

bit **[**7**:**0**]** r**[**4**];**

byte DataInQueue**[**$**];**

byte AddressQueue**[**$**];**

**function** new **();**

ErrorCounter **=** 0**;**

pc **=** 0**;**

ir **=** 0**;**

r**[**0**]=**0**;** r**[**1**]=**0**;** r**[**2**]=**0**;** r**[**3**]=**0**;**

**endfunction** // new

**function** automatic void incr\_pc**();**

pc**++;**

$display**(**"incr pc: %02h"**,** pc**);**

**endfunction** // incr\_pc

**function** automatic void update\_pc**(**int val**);**

pc **=** val**;**

**endfunction** // update\_pc

**function** automatic void update\_ir**(**Instruction I**);**

ir **=** I**.**byte0**;**

**endfunction** // update\_ir

**function** automatic void update\_addrq**(input** bit **[**7**:**0**]** v**);**

AddressQueue**.**push\_back**(**v**);**

$display**(**"Updating Address Queue: %02h %p"**,** v**,** AddressQueue**);**

**endfunction** // update\_addrq

**function** automatic void fetch1**();**

//The value of the pc should next appear on the

// address line.

$display**(**"@%t:fetch1: %02h"**,** $time**,** pc**);**

update\_addrq**(**pc**);**

**endfunction** // fetch1

**function** automatic void fetch2**(**Instruction I**);**

$display**(**"@%t:fetch2"**,** $time**);**

update\_ir**(**I**);**

incr\_pc**();**

**endfunction** // fetch2

**function** automatic void readbyte2**(**Instruction I**);**

$display**(**"readbyte2 I.byte1=%02x"**,** I**.**byte1**);**

update\_addrq**(**pc**);**

update\_addrq**(**I**.**byte1**);**

incr\_pc**();**

**endfunction** // readbyte2

**function** automatic void decode**(**Instruction I**,** bit **[**7**:**0**]** rdval**);**

bit **[**3**:**0**]** opcode **=** I**.**byte0**[**7**:**4**];**

bit **[**1**:**0**]** src **=** I**.**byte0**[**3**:**2**];**

bit **[**1**:**0**]** dst **=** I**.**byte0**[**1**:**0**];**

$display**(**"@%0d SB: Decode"**,** $time**);**

**case** **(**I**.**byte0**[**7**:**4**])**

NOP**:** **;**

ADD**:** r**[**dst**]** **=** r**[**src**]** **+** r**[**dst**];**

SUB**:** r**[**dst**]** **=** r**[**dst**]** **-** r**[**src**];**

AND**:** r**[**dst**]** **=** r**[**src**]** **&** r**[**dst**];**

NOT**:** r**[**dst**]** **=** **~**r**[**src**];**

RD**:** **begin**

r**[**dst**]** **=** rdval**;**

readbyte2**(**I**);**

**end**

RDI**:begin**

update\_addrq**(**pc**);**

incr\_pc**();**

r**[**dst**]** **=** I**.**byte1**;**

**end**

WR**:** **begin**

readbyte2**(**I**);**

DataInQueue**.**push\_back**(**r**[**src**]);**

**end**

BR**,**BRZ**,**HALT**:** **;**

**endcase** // case (I.byte0)

**endfunction** // decode

**function** automatic void check\_pc**(**bit **[**7**:**0**]** cpc**);**

compare\_value**(**"PC"**,** cpc**,** pc**);**

**endfunction** // check\_pc

**function** automatic void check\_ir**(**bit **[**7**:**0**]** cir**);**

compare\_value**(**"IR"**,** cir**,** ir**);**

**endfunction** // check\_ir

**function** automatic void check\_regs**(**bit **[**7**:**0**]** cr0**,**

bit **[**7**:**0**]** cr1**,**

bit **[**7**:**0**]** cr2**,**

bit **[**7**:**0**]** cr3**);**

compare\_value**(**"r0"**,** cr0**,** r**[**0**]);**

compare\_value**(**"r1"**,** cr1**,** r**[**1**]);**

compare\_value**(**"r2"**,** cr2**,** r**[**2**]);**

compare\_value**(**"r3"**,** cr3**,** r**[**3**]);**

**endfunction** // check\_regs

**function** automatic void check\_address**(**bit **[**7**:**0**]** caddress**);**

compare\_qvalue**(**"address"**,** "AddressQueue"**,** caddress**,** AddressQueue**);**

**endfunction** // check\_address

**function** automatic void check\_dataIn**(**bit **[**7**:**0**]** cdataIn**);**

compare\_qvalue**(**"cdataIn"**,** "DataInQueue"**,** cdataIn**,** DataInQueue**);**

**endfunction** // check\_address

**function** automatic void compare\_qvalue**(**string name**,** string qname**,** bit **[**7**:**0**]** actual**,** ref byte queue**[**$**]);**

$display**(**"Checking Q=%p"**,** queue**);**

**if(**queue**.**size**()** **==** 0**)**

$display**(**"Error: %s is empty!"**,** qname**);**

**else**

compare\_value**(**name**,** actual**,** queue**.**pop\_front**());**

**endfunction** // compare\_qvalue

**function** automatic void compare\_value**(**string name**,** bit **[**7**:**0**]** actual**,** bit **[**7**:**0**]** expected**);**

**if(**actual **!=** expected**)** **begin**

$display**(**"@%08d: Error: Found Unexpected Value for %s: Expected: %02X Actual: %02X"**,** $time**,** name**,** expected**,** actual**);**

ErrorCounter**++;**

**end**

//else

//$display("@%08d: %s is correct", $time, name);

**endfunction** // compare\_value

Endclass

**Complete Coverage Report**

COVERGROUP COVERAGE:  
---------------------------------------------------------------------------------------------------

Covergroup Metric Goal/ Status

At Least

----------------------------------------------------------------------------------------------------

TYPE /Alu\_RISC\_v\_unit/Cov 100.0% 100 Covered

Coverpoint Cov::opcodes 100.0% 100 Covered //Requirements 1 & 4

Coverpoint Cov::srcs 100.0% 100 Covered

Coverpoint Cov::dsts 100.0% 100 Covered

Coverpoint Cov::addresses 100.0% 100 Covered

Cross Cov::opcode\_src 100.0% 100 Covered //Requirement 2

Cross Cov::opcode\_dst 100.0% 100 Covered //Requirement 3

Cross Cov::opcode\_src\_dst 100.0% 100 Covered //Requirement 5

Cross Cov::WrRdRdiMem 100.0% 100 Covered //Requirements 6 & 7

Covergroup instance Alu\_RISC\_v\_unit::Driver\_cbs\_cov::ck

100.0% 100 Covered

Coverpoint opcodes 100.0% 100 Covered

covered/total bins: 72 72

missing/total bins: 0 72

illegal\_bin bad\_ops[7] 0 ZERO

illegal\_bin bad\_ops[8] 0 ZERO

illegal\_bin bad\_ops[10] 0 ZERO

illegal\_bin bad\_ops[11] 0 ZERO

illegal\_bin bad\_ops[12] 0 ZERO

illegal\_bin bad\_ops[13] 0 ZERO

illegal\_bin bad\_ops[14] 0 ZERO

illegal\_bin bad\_ops[15] 0 ZERO

bin ops[0] 642 1 Covered

bin ops[1] 657 1 Covered

bin ops[2] 624 1 Covered

bin ops[3] 650 1 Covered

bin ops[4] 626 1 Covered

bin ops[5] 2318 1 Covered

bin ops[6] 2229 1 Covered

bin ops[9] 2254 1 Covered

bin op\_order[9=>9] 517 1 Covered

bin op\_order[9=>6] 501 1 Covered

bin op\_order[9=>5] 483 1 Covered

bin op\_order[9=>4] 146 1 Covered

bin op\_order[9=>3] 149 1 Covered

bin op\_order[9=>2] 169 1 Covered

bin op\_order[9=>1] 152 1 Covered

bin op\_order[9=>0] 137 1 Covered

bin op\_order[6=>9] 497 1 Covered

bin op\_order[6=>6] 498 1 Covered

bin op\_order[6=>5] 569 1 Covered

bin op\_order[6=>4] 128 1 Covered

bin op\_order[6=>3] 130 1 Covered

bin op\_order[6=>2] 135 1 Covered

bin op\_order[6=>1] 134 1 Covered

bin op\_order[6=>0] 138 1 Covered

bin op\_order[5=>9] 509 1 Covered

bin op\_order[5=>6] 515 1 Covered

bin op\_order[5=>5] 543 1 Covered

bin op\_order[5=>4] 148 1 Covered

bin op\_order[5=>3] 165 1 Covered

bin op\_order[5=>2] 142 1 Covered

bin op\_order[5=>1] 142 1 Covered

bin op\_order[5=>0] 154 1 Covered

bin op\_order[4=>9] 152 1 Covered

bin op\_order[4=>6] 147 1 Covered

bin op\_order[4=>5] 132 1 Covered

bin op\_order[4=>4] 43 1 Covered

bin op\_order[4=>3] 40 1 Covered

bin op\_order[4=>2] 29 1 Covered

bin op\_order[4=>1] 48 1 Covered

bin op\_order[4=>0] 34 1 Covered

bin op\_order[3=>9] 141 1 Covered

bin op\_order[3=>6] 147 1 Covered

bin op\_order[3=>5] 148 1 Covered

bin op\_order[3=>4] 44 1 Covered

bin op\_order[3=>3] 43 1 Covered

bin op\_order[3=>2] 40 1 Covered

bin op\_order[3=>1] 45 1 Covered

bin op\_order[3=>0] 42 1 Covered

bin op\_order[2=>9] 149 1 Covered

bin op\_order[2=>6] 130 1 Covered

bin op\_order[2=>5] 132 1 Covered

bin op\_order[2=>4] 53 1 Covered

bin op\_order[2=>3] 47 1 Covered

bin op\_order[2=>2] 37 1 Covered

bin op\_order[2=>1] 36 1 Covered

bin op\_order[2=>0] 40 1 Covered

bin op\_order[1=>9] 143 1 Covered

bin op\_order[1=>6] 152 1 Covered

bin op\_order[1=>5] 154 1 Covered

bin op\_order[1=>4] 35 1 Covered

bin op\_order[1=>3] 44 1 Covered

bin op\_order[1=>2] 35 1 Covered

bin op\_order[1=>1] 51 1 Covered

bin op\_order[1=>0] 43 1 Covered

bin op\_order[0=>9] 146 1 Covered

bin op\_order[0=>6] 139 1 Covered

bin op\_order[0=>5] 157 1 Covered

bin op\_order[0=>4] 29 1 Covered

bin op\_order[0=>3] 32 1 Covered

bin op\_order[0=>2] 36 1 Covered

bin op\_order[0=>1] 49 1 Covered

bin op\_order[0=>0] 54 1 Covered

The cover bins checking that all write, read, and rdi instructions have written all memory locations have been excluded due to length.

**Partially Coverage Report**

COVERGROUP COVERAGE:

----------------------------------------------------------------------------------------------------

Covergroup Metric Goal/ Status

At Least

----------------------------------------------------------------------------------------------------

TYPE /Alu\_RISC\_v\_unit/Cov 50.5% 100 Uncovered

type\_option.weight=1

type\_option.goal=100

type\_option.comment=

type\_option.strobe=0

type\_option.merge\_instances=0

Coverpoint Cov::opcodes 48.6% 100 Uncovered //Requirements 1 &4

Coverpoint Cov::srcs 100.0% 100 Covered

Coverpoint Cov::dsts 100.0% 100 Covered

Coverpoint Cov::addresses 13.6% 100 Uncovered

Cross Cov::opcode\_src 65.0% 100 Uncovered //Requirement 2

Cross Cov::opcode\_dst 55.0% 100 Uncovered //Requirement 3

Cross Cov::opcode\_src\_dst 17.1% 100 Uncovered //Requirement 5

Cross Cov::WrRdRdiMem 4.5% 100 Uncovered Requirement 6 & 7

Covergroup instance \/Alu\_RISC\_v\_unit::Driver\_cbs\_cov::ck

50.5% 100 Uncovered

option.name=\/Alu\_RISC\_v\_unit::Driver\_cbs\_cov::ck

Coverpoint opcodes 48.6% 100 Uncovered

covered/total bins: 35 72

missing/total bins: 37 72

option.weight=1

option.goal=100

option.comment=

option.at\_least=1

option.auto\_bin\_max=64

option.detect\_overlap=0

illegal\_bin bad\_ops[7] 0 ZERO

illegal\_bin bad\_ops[8] 0 ZERO

illegal\_bin bad\_ops[10] 0 ZERO

illegal\_bin bad\_ops[11] 0 ZERO

illegal\_bin bad\_ops[12] 0 ZERO

illegal\_bin bad\_ops[13] 0 ZERO

illegal\_bin bad\_ops[14] 0 ZERO

illegal\_bin bad\_ops[15] 0 ZERO

bin ops[0] 3 1 Covered

bin ops[1] 4 1 Covered

bin ops[2] 2 1 Covered

bin ops[3] 3 1 Covered

bin ops[4] 2 1 Covered

bin ops[5] 15 1 Covered

bin ops[6] 12 1 Covered

bin ops[9] 9 1 Covered

bin op\_order[9=>9] 2 1 Covered

bin op\_order[9=>6] 0 1 ZERO

bin op\_order[9=>5] 5 1 Covered

bin op\_order[9=>4] 0 1 ZERO

bin op\_order[9=>3] 1 1 Covered

bin op\_order[9=>2] 0 1 ZERO

bin op\_order[9=>1] 1 1 Covered

bin op\_order[9=>0] 0 1 ZERO

bin op\_order[6=>9] 1 1 Covered

bin op\_order[6=>6] 4 1 Covered

bin op\_order[6=>5] 3 1 Covered

bin op\_order[6=>4] 1 1 Covered

bin op\_order[6=>3] 1 1 Covered

bin op\_order[6=>2] 1 1 Covered

bin op\_order[6=>1] 1 1 Covered

bin op\_order[6=>0] 0 1 ZERO

bin op\_order[5=>9] 4 1 Covered

bin op\_order[5=>6] 5 1 Covered

bin op\_order[5=>5] 1 1 Covered

bin op\_order[5=>4] 1 1 Covered

bin op\_order[5=>3] 1 1 Covered

bin op\_order[5=>2] 0 1 ZERO

bin op\_order[5=>1] 0 1 ZERO

bin op\_order[5=>0] 3 1 Covered

bin op\_order[4=>9] 0 1 ZERO

bin op\_order[4=>6] 1 1 Covered

bin op\_order[4=>5] 1 1 Covered

bin op\_order[4=>4] 0 1 ZERO

bin op\_order[4=>3] 0 1 ZERO

bin op\_order[4=>2] 0 1 ZERO

bin op\_order[4=>1] 0 1 ZERO

bin op\_order[4=>0] 0 1 ZERO

bin op\_order[3=>9] 1 1 Covered

bin op\_order[3=>6] 1 1 Covered

bin op\_order[3=>5] 1 1 Covered

bin op\_order[3=>4] 0 1 ZERO

bin op\_order[3=>3] 0 1 ZERO

bin op\_order[3=>2] 0 1 ZERO

bin op\_order[3=>1] 0 1 ZERO

bin op\_order[3=>0] 0 1 ZERO

bin op\_order[2=>9] 1 1 Covered

bin op\_order[2=>6] 1 1 Covered

bin op\_order[2=>5] 0 1 ZERO

bin op\_order[2=>4] 0 1 ZERO

bin op\_order[2=>3] 0 1 ZERO

bin op\_order[2=>2] 0 1 ZERO

bin op\_order[2=>1] 0 1 ZERO

bin op\_order[2=>0] 0 1 ZERO

bin op\_order[1=>9] 0 1 ZERO

bin op\_order[1=>6] 0 1 ZERO

bin op\_order[1=>5] 1 1 Covered

bin op\_order[1=>4] 0 1 ZERO

bin op\_order[1=>3] 0 1 ZERO

bin op\_order[1=>2] 0 1 ZERO

bin op\_order[1=>1] 2 1 Covered

bin op\_order[1=>0] 0 1 ZERO

bin op\_order[0=>9] 0 1 ZERO

bin op\_order[0=>6] 0 1 ZERO

bin op\_order[0=>5] 3 1 Covered

bin op\_order[0=>4] 0 1 ZERO

bin op\_order[0=>3] 0 1 ZERO

bin op\_order[0=>2] 0 1 ZERO

bin op\_order[0=>1] 0 1 ZERO

bin op\_order[0=>0] 0 1 ZERO

Coverpoint srcs 100.0% 100 Covered

covered/total bins: 4 4

missing/total bins: 0 4

option.weight=1

option.goal=100

option.comment=

option.at\_least=1

option.auto\_bin\_max=64

option.detect\_overlap=0

bin sregs[0] 12 1 Covered

bin sregs[1] 7 1 Covered

bin sregs[2] 15 1 Covered

bin sregs[3] 16 1 Covered

Coverpoint dsts 100.0% 100 Covered

covered/total bins: 4 4

missing/total bins: 0 4

option.weight=1

option.goal=100

option.comment=

option.at\_least=1

option.auto\_bin\_max=64

option.detect\_overlap=0

bin dregs[0] 8 1 Covered

bin dregs[1] 16 1 Covered

bin dregs[2] 13 1 Covered

bin dregs[3] 13 1 Covered