Tanmay Radke
000823912
Grp 24

Project 2B - Functional Coverage Report

A Entire functional coverage is divided into six covergroups namely:

1 Top
2 Controller
3 Fetch_cov
4 Decode_cov
5 Execute_cov
6 WB_cov
7 MA_cov

Each coverage is now explained in greater depth.

I) Covergroup Top

This covergroup deals with the top signals of the DUT. Functional Coverage for Instr_dout , Data_dout , complete_data , complete_instr , reset are considered in this covergroup.

a) Instr_dout

The following instructions are covered -
ADD Register , Register,Register and **ADD Register , Register , immediate**
AND Register , Register,Register and **AND Register , Register , immediate**
NOT , LD,LDR,LDI,ST,STR,STI,JMP,BRACH

Illegal bins are assigned for Instr_dout[15:12] = 4,8,13,15 as these values shud never occur during the simulation

```
bins zeros = {16'b0};
bins ones = {16'hffff};
bins special11 = {16'hff00};
bins special22 = {16'hf0f0};
bins special33 = {16'b1100110011001100};
bins special44 = {16'b1010101010101010};
```

The following six outputs are used to check for stuck at zero and one problem.
Also to get various sequences of Inputs the following sequences are being checked

```
bins sequence7 = (1,5,9 => 1,5,9 => 1,5,9); // 3 ALU continious ALU instruction
```

bins sequence6 = (3,7,11,2,6,10 => 3,7,11,2,6,10 => 3,7,11,2,6,10); //Three continious memory operations
bins sequence1 = (0,12 => 0,12 => 0,12 => 0,12); // Control followed by control

bins sequence8 = (3,7,11,2,6,10 => 3,7,11,2,6,10 => 1,5,9); //Mem Mem Alu
bins sequence9 = (3,7,11,2,6,10 => 1,5,9 => 1,5,9); //Mem Alu Alu

bins sequence10 = (1,5,9 => 3,7,11,2,6,10 => 3,7,11,2,6,10); //Alu Mem Mem
bins sequence11 = (1,5,9 => 1,5,9 => 3,7,11,2,6,10); //Alu Alu Mem

bins sequence2 = (6,12 => 0,12); //Memory followed by control (Most Important)
bins sequence3 = (10 => 0,12);
bins sequence4 = (3,7 => 0,12);
bins sequence5 = (11 => 0,12);

bins sequence12 = (3,7,11,2,6,10 => 1,5,9 => 0,12); // Mem Alu Cont

b) Reset

bins all = (1 => 1 => 1=> 1 => 1 =>0); The following transition is checked for. Reset remains high for 5 continious samples and then goes down to zero

c) Complete_data and Complete_instr has to be high through the simulation. Hence Illegal bins checking for their zero value is checked.

Thus Top covergroup takes care on the inputs coming into the DUT.

B) Controller – The Heart of the LC3 Microcontroller

a)This is the most important covergroup as it delas with all the inputs and outputs of the Controller block

IR_Exec , IR and Instr_dout are checked for all the possible instructions.

Then sr1 and sr2 (separate for ALU and stores) from IR is calculated and checked for all the possible values.

dr is calculated from IR_Exec and checked for the different values.

Various values of NZP are covered and sequence (1,2,3,4,5,6,7 => 0) is checked to see when whether NZP is set to 0 synchronously for non branch instruction

psr is checked for its three possible values.

illegal_bins bad_val = {0,3,5,6,7};

for psr is declared as psr can never have thes =e values.

Add bit Instr_dout[5] is also covered for 0 and 1 value.

Cross coverage for IR_Exec_cov ,sr1_IR_cov ,dr_IR_Exec_cov , IR_decode_con_cov , sr2_IR_cov is taken
To check whether conditions where various are asserted at the proper time.

Then coverpoints for alu1_bypass , alu2_bypass , mem1_bypass , mem2_bypass are declared to check whether they go to 0,s and one.

Out_bypass is a cross betweenIR , IR_Exec ,bypass signals to check whether LEA followed by a 2 register dependence, ALU followed by dependant instructions , memory instruction followed by a dependant instruction.

```
illegal_bins bad_value1 = binsof(alu1_by_cov.one)&&binsof(mem1_by_cov.one);
illegal_bins bad_value2 = binsof(alu2_by_cov.one)&&binsof(mem2_by_cov.one);
```

Thus memory and bypass signas cannot be high at the same time.

Also various sequence of transitions for enables for various combinations of IR_Exec , IR and Instr _dout .

Similarly coverage fr br_taken , memstate is done.

C) Covergroup Fetch_cov.

This covergroup deals with the Fetch Block of DUT. The coverpoints include for taddr , npc , pc , instrmem_rd ,enable_fectch..Coverage for reset values and allfs is checked.

cross temp_enable_fetch_cov , instrmem_rd_fetch_cov is calculated to check whether the DUT takes instrmem_rd values as 1's and z's properly.

D)Covergroup Decode_cov.

This covergroup deals with all the output of the Decode block. All the valid values for W_Control, Mem_Control , E_Control are checked. Bins for illegal values are created to check whether the above signals take values other than the expected.

E)Execute_cov

This covergroup checks whether the outputs of the execute block takes all the possible values. Its being checked whether sr1 , sr2 and dr take all values from 0 to 1.

Similarly it is being checked if aluout and pcout cover the following values in order to check at stuck at 0/1 faults and bus swapping faults

```
bins zeros = {16'b0};
bins ones = {16'hffff};
bins special11 = {16'hff00};
bins special22 = {16'hf0f0};
bins special33 = {16'b1100110011001100};
bins special44 = {16'b1010101010101010};
```

Similarly coverage for mem_Control and W_Control is being checked.

F)WB_cov

The psr value is checked for 1,4,2 and illegal_bis are created for 0,3,5,6,7.

The various values for VSR1 and VSR2 are checked to find stuck at and swapping bus faults.

```
bins zeros = {16'b0};
bins ones = {16'hffff};
bins special11 = {16'hff00};
bins special22 = {16'hf0f0};
bins special33 = {16'b1100110011001100};
bins special44 = {16'b1010101010101010};
```

G)MA_cov

Memout value is checked for the six special cases to find stuck at 0 and 1 problem and bus swapping problem.

DMem_Addr and Dmem_din is checked for additional combination of all z as this s the value it shud have in memstae 3.

DMem_rd cecks for 0 , 1, z .

Memstate is checked for 0 – 3 states with the following transitions sequence checked

```
bins sequence1 = (3 =>0 =>3);
bins sequence2 = (3 =>2 =>3);
```

```
bins sequence3 = (3 =>1 => 0 =>3);
bins sequence4 = (3 =>1 => 2 =>3);
bins sequence5 = (3 =>3 =>3);


memstate_cx : cross memstate1_cov, DMem_rd_MemAccess_cov,
DMem_din_MemAccess_cov, DMem_addr_MemAccess_cov{
```

Now as stated above a cross coverage between above coverpoints is done to check for a cross between input and output . At memstate = 0 ,1 Dmem_din = 0 , for memstate = 3 , Dmem_addr , DMem_din and Dmem_rd are z's . All these combinations are checked.

Conclusion)

Almost 100% coverage is obtained for the considered coverage points