ECE-745: Verification of a 16-bit LC3 Microprocessor

Group 14
Jay Udani
Kunal Buch
Jovin Miranda
Venkata Sai Sashank

Objective:

To devise a layered and re-usable test-bench with object oriented paradigm to verify the 6 stage LC3 Microcontroller with a comprehensive coverage plan.

Tools Required:

A layered test bench is written in System Verilog using object-oriented programming methodologies. Mentor Graphics Modelsim10.3 is used to simulate the test bench

Verification Levels:

The test bench has a golden reference model in which each stage is modeled as a system Verilog class and is driven with the inputs from the DUT. At each stage outputs of golden reference are checked against the DUT's outputs to find bugs that are contained within a stage. Once blocks are individually evaluated the system can be checked at higher level to observe bugs that occur due to interactions between the blocks

Verification Strategy:

The verification environment contains a generator, driver, golden reference model and coverage files. The individual functionality is described as given below:

Instruction & Generator:

These files are used to generate the Instr_dout. The Instruction file contains the constraints used on various parts to obtain the desired instruction sequences.

Driver:

This class drives the inputs to the DUT and golden reference models. A mailbox bounded to 1 is used to transfer the instruction between the generator and the driver. The driver takes the instruction from the generator on a clock cycle when instrmem_rd == 1. This class also has the reset and initialization functions

Goldenref_*:

These files contain the goldenref classes of Fetch, Decode, Execute, Memory Access, Writeback and controller classes. These are modeled based on the project specifications and are evaluated against the clean DUT to guarantee appropriate behavior

Environment:

This class contains the creation, execution and checking calls for all of the above mentioned files.

Bug Report

1) Data_defs_bug1:

   LC3 Stage the bug is in: Control
   Cases: When IR = ST or STI or STR
   Description: While matching for dependency between the consecutive instructions in the execute stage, when IR == ST, STR or STI, the bypass_alu_2 is enabled by checking bit IR [2:0] instead of IR [11:9].

2) Data_defs_bug2:

   LC3 Stage the bug is in: Execute
   Cases: When Branch (BR) is present at IR
   Description: When Branch (BR) is present at IR, NZP [1] is stuck at 0.

3) Data_defs_bug3:

   LC3 Stage the bug is in: Decode
   Description: Signals IR and npc_out are asynchronous instead of synchronous.
   Cases: It is seen in all cases for every instruction

4) Data_defs_bug4:

   LC3 Stage the bug is in: Memory Access
   Description: Mem_out is assigned the value of Dmem_addr instead of Dmem_dout.
   Cases: It is seen in all cases for every instruction

5) Data_defs_bug5:

   LC3 Stage the bug is in: Writeback
   Cases: When the signal enable_writeback being high and IR_Exec is not equal to LEA

Description: When the signal enable_writeback being high and IR_Exec is not equal to LEA, the wires of mem_out and alu_out have been interchanged, hence, affecting the values of PSR, VSR1 and VSR2.

Other interpretation can be given as, mem_out -> alu_out, alu_out -> pc_out and pc_out -> mem_out. But since pc_out = alu_out and hence we do not see the bug in that case.

Coverage Plan

Cover Groups

ALU Instructions:

Cover Group Name: ALU_OPR_cg

Details: This cover group defines the coverage for the ALU instructions of LC3.

Cover Points:

Cover Point Name	Description
Cov_alu_opcode	Covers all the ALU operations i.e. ADD, AND and NOT.
Cov_imm_en	Covers Immediate Enable(INSTR_DOUT[5])
Cov_SR1	Covers all the values of SR1
Cov_SR2	Covers all the values of SR2
Cov_DR	Covers all the values of DR
Cov_imm5	Covers all the values of imm5 qualified with the immediate type instructions.
Xc_opcode_imm_en	Cross Cover point between Cov_alu_opcode & Cov_imm_en.
Xc_opcode_dr_sr1_imm5	Cross-Cover point between Cov_alu_opcode, Cov_SR1, Cov_DR & Cov_imm5 for only immediate mode instructions.
Xc_opcode_dr_sr1_sr2	Cross-Cover point between Cov_alu_opcode, Cov_SR1, Cov_DR & Cov_imm5 for only register mode instructions.
Cov_aluin1	Aluin1 binned into 10 bins
Cov_aluin1_corner	Covers the corner cases of aluin1
Cov_aluin2	Aluin2 binned into 10 bins
Cov_aluin2_corner	Covers the corner cases of aluin2
Xc_opcode_aluin1	Cross-Cover point between Cov_alu_opcode, Cov_aluin1 to cover all the corner case values for each instruction.
Xc_opcode_aluin1_cover	Cross-Cover point between Cov_alu_opcode, Cov_aluin1_corner to cover all the corner case values for each instruction.
Xc_opcode_aluin2	Cross-Cover point between Cov_alu_opcode, Cov_aluin2 to cover all the corner case values for each instruction.
Xc_opcode_aluin2_corner	Cross-Cover point between Cov_alu_opcode, Cov_aluin2_corner to cover all the corner case values for each instruction.
Cov_opr_zero_zero	Covers the corner case when aluin1 and aluin2 are zeros.
Cov_opr_zero_all1	Covers the corner case when aluin1 is zero and aluin2 is all ones
Cov_opr_all1_zero	Covers the corner case when aluin1 is all ones and aluin2 is zero
Cov_opr_all1_all1	Covers the corner case when aluin1 and aluin2 are all ones
Cov_opr_alt01_alt01	Corner scenario of aluin1=5555 and aluin2 =5555

Cov_opr_alt01_alt10	Corner scenario of aluin1=5555 and aluin2 =AAAA
Cov_opr_alt10_alt01	Corner scenario of aluin1=AAAA and aluin2 =5555
Cov_opr_alt10_alt10	Corner scenario of aluin1=AAAA and aluin2 =AAAA
Cov_aluin1_pos_neg	Corner scenario when aluin1 is positive and negative
Cov_aluin2_pos_neg	Corner scenario when aluin2 is positive and negative
x_aluin1_aluin2_pos_neg	Cross-Cover point between Cov_**aluin1**_pos_neg and Cov_**aluin2**_pos_neg to cover all the cases

Order of Instructions:

Cover group name: opr_seq_cg:

Details: This cover group defines the coverage for all the possible combinations for the different types of instructions

Bin Name	Description
add_seq	Covers the sequence where ADD instruction is followed by any other type of instruction (ALU, MEM, CTRL)
and_seq	Covers the sequence where AND instruction is followed by any other type of instruction (ALU, MEM, CTRL)
not_seq	Covers the sequence where NOT instruction is followed by any other type of instruction (ALU, MEM, CTRL)
ld_seq	Covers the sequence where LD instruction is followed by ALU or CTRL instruction
ldr_seq	Covers the sequence where LDR instruction is followed by ALU or CTRL instruction
ldi_seq	Covers the sequence where LDI instruction is followed by ALU or CTRL instruction
lea_seq	Covers the sequence where LEA instruction is followed by any other type of instruction (ALU, MEM, CTRL)
st_seq	Covers the sequence where ST instruction is followed by ALU or CTRL instruction
str_seq	Covers the sequence where STR instruction is followed by ALU or CTRL instruction
sti_seq	Covers the sequence where STI instruction is followed by ALU or CTRL instruction
br_seq	Covers the sequence where BR instruction is followed by ALU or MEM instruction

jmp_seq	Covers the sequence where BR instruction is followed by ALU or MEM instruction

Control Instructions:

Coverage group name - CTRL_OPR_cg
Details: This cover group defines the coverage for the Control operations of LC3.

Bin Name	Description
Cov_ctrl_opcode	Covers all the CTRL operations ie BR and JMP
Cov_BaseR	Covers all the values of the BaseR.
Cov_NZP	Covers all the values of NZP register
Cov_PSR	Covers all the values of PSR register
Cov_PCoffset9	Covers all the values of PCoffset9, binned to 10 bins
Cov_PCoffset9_c	Covers all the corner case values of PCoffset9
Xc_NZP_PSR	Cross cover point between Cov_NZP and Cov_PSR

Memory Instructions:

Coverage group name - MEM_OPR_cg
Details: This cover group defines the coverage for the Memory operations of LC3.

Bin Name	Description
Cov_mem_opcode	Covers all the MEM operations
Cov_BaseR	Covers all the values of the BaseR.
Cov_SR_*	Covers all the values of SR register
Cov_DR_*	Covers all the values of DR register
Cov_PCoffset6_*	Covers all the values of PCoffset6, binned to 10 bins
Cov_PCoffset6_c_*	Covers all the corner case values of PCoffset6
Cov_PCoffset9_*	Covers all the values of PCoffset9, binned to 10 bins
Cov_PCoffset9_c_*	Covers all the corner case values of PCoffset9
xc_BaseR_DR_offset6_*, xc_BaseR_SR_offset6_*	Cross covers BaseR, DR, SR and offset values

Summary:

Number of Instructions: 750,000

Coverage: 100%

No. of bins: 4363

Coverage Directives: 32

ModelSim Coverage Report

Number of tests run:	1
Passed:	1
Warning:	0
Error:	0
Fatal:	0

List of tests included in report...

List of global attributes included in report...

Coverage Summary by Structure:

Design Scope	Coverage
LC3_test_top	100.00%
project_test_top_sv_unit	100.00%
Coverage	100.00%

Coverage Summary by Type:

Total Coverage:					100.00%	100.00%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Covergroups	4363	4363	0	1	100.00%	100.00%
Directives	32	32	0	1	100.00%	100.00%

Report generated by ModelSim on Sat 15 Apr 2017 09:19:44 PM EDT