Alex Olson | aleolson@pdx.edu
Nurettin Can Orbegi | orbegi@pdx.edu
Matty Baba Allos | matty@pdx.edu
ECE 593 Spring 2019

Verification Plan of a MIPS-16 Processor

Background:

For the ECE 593 final project, our team of three validators will work on verifying the a MIPS-16 Processor design. The processor is a simplified MIPS five-stage processor. The processor supports 13 instructions and using 8 general purpose registers. Our goal is to functionally verify the design for correctness using pre-silicon methodologies.

Verification Goal:

The design is composed of five functional units (ID, IF, EX, MEM, WB) and auxiliary modules (ALU, data memory, instruction memory, hazard detection unit, register file). Our verification goal is to test the majority of the design within our given timeframe of four weeks. Due to size of the design and the short timeframe, we decided to prioritize our verification based on what we believe are the most important stages and units. We will verify, the ID and WB stages, ALU unit, and hazard detection unit. In the future we hope to verify all five stages and chip-level testing when all units are connected. With that in mind, we will at least provide some basic chip-level testing for this stage of verification.

Verification Requirements:

Verification Levels

- Verify ID and WB stages, ALU unit and hazard detection unit at the unit level
- Verify that the design works at the chip level (top level of this design). This will be limited for this stage of verification process.

Functions

The chip has 13 total instructions that we will verify that includes: NOP, ADD, SUB, AND, OR, XOR, Shift Right, Shift Left, Shift Right Logical (SRU), ADD Immediate, Load, Store, and Branch. Each unit also has a reset (active high) signal to check as well.

Specific Tests & Methods

1. Type of Verification
    - Design will be tested as gray box.
2. Verification Strategy
    - Randomly generated operations will be performed.
3. Abstraction Level
    - Stimulus will send at the interface of the module.
4. Checking
    - We will use reference model for checking.

Coverage

- Make sure to test all possible opcodes
- Multiple instructions in the pipeline
- Make sure we test a case with stall
- Make sure we test a branch case
- Register based instructions
- Memory based instructions

Scenarios

- Fill up the pipeline
- Send a single instruction
- Generate a stall
- Generate a branch
- Generate a memory request
- Multi-cycle operation followed by single-cycle
- Back to back operations that are performed from same registers
- Operations those writes back to registers then operations those reads from same registers.
- Loading data from memory to register
- Saving data from register to memory

Project Management:

Tools

- QuestaSim
- SystemVerilog
- Python
- Github (collaboration)

Risk/Dependencies

- Tight schedule to verify according to plan and pass the class
- People are busy with other work(classes)

Resources

- 3 Students,
- Compute Systems
- 3 QuestaSim licenses

Schedule

May 10, 2019 - Finish the first draft of the verification plan
May 17, 2019 - Submit a waveform showing that the testbench can drive deterministic cases
May 24, 2019 - Submit the current design and testbench, include coverage
May 31, 2019 - Submit the final design, testbench, and design report

ID stage verification plan

Verification Levels

The ID stage will be verify at the unit level

Functions

- Verify
    - Reset
    - Unit is enabled
    - Able to accept instructions for decoding when unit is enabled
    - Able to read register file
    - Able to detect hazards and insert pipeline bubbles
    - Able to output correctly decoded instructions to the next stage.

Specific Tests & Methods

1. Type of Verification
    - The unit will be tested as black box.
2. Verification Strategy
    - We will use the constrained random stimulus to exercise the DUV
3. Abstraction Level
    - Stimulus will send at the interface of the module.
4. Checking
    - Golden vector - ID -> to checking registers (GPRs)

Coverage

- Reset
- Instruction decode enable signal
- Possible instructions,
- Possible source 1 addresses,
- Possible source 2 addresses,
- Instruction decode enable signal
- Possible instructions,
- Possible source 1 addresses,
- Possible source 2 addresses,
- Hazard detection outputs
- Branch offset and branch_taken

Scenarios

- When reset is asserted, outputs should be zero
- When instruction decode enable signal is asserted, design should produce outputs
- For each operation (NOP, ADD, SUB, AND, OR, XOR, SL, SR, SRU, ADDI, LD, ST, BZ), design produces different pipeline_reg_out bus. Check the bus for each operation
- Back to back operations from same registers for each operations mentioned above.
- Send branch instructions, observe branch offset.

WB stage verification plan

Verification Levels

The WB stage will be verified at the unit level

Functions

- Verify
    - Reset
    - Read pipeline registers and write back to register file
    - Read pipeline registers and write back to EX stage

Specific Tests & Methods

5. Type of Verification
    - The unit will be tested as black box.
6. Verification Strategy
    - We will use the constrained random stimulus to exercise the DUV
7. Abstraction Level
    - Stimulus will send at the interface of the module.
8. Checking
    - Golden vector

Coverage

- Write enable
- Write back destinations
- Write back data

Scenarios

- Send in pipeline registers and observe the out to the register file is correct
- Test when the register file is busy
- Write back to the same register back-to-back

ALU unit verification plan

Verification Levels

The ALU unit will be verify at the unit level

Functions

- Verify
    - All commands and operations
    - Various data on both sources.
    - Handling invalid commands

Specific Tests & Methods

9. Type of Verification
    - The unit will be tested as black box.
10. Verification Strategy
    - We will use the constrained random stimulus to exercise the DUV. The proportions will be:

Value	0x0000	0xFFFF	$random
Probability	0.25	0.25	0.50

11. Abstraction Level
    - Stimulus will send at the interface of the module.
12. Checking
    - Reference model

Coverage

- All operations (NOP, ADD, SUB, AND, OR, XOR, SL, SR, SRU, ADDI, LD, ST, BZ)
- Constrained random range of data values on both inputs and output (two 16-bit inputs and one 16-bit output).
- Single cycle operations
- Multi-cycle operations
- Transitions between two different operations (e.g single cycle to multi-cycle)

Scenarios

- Send in a combination of all valid commands and data
- Send in an invalid operation and observe no output.
- Back to back operations

Hazard detection unit verification plan

Verification Levels

The Hazard detection unit will be verified at the unit level

Functions

- Verify
    - Pipeline stall can be generated
    - Pipeline hazard detected correctly

Specific Tests & Methods

13. Type of Verification
    - The unit will be tested as black box.
14. Verification Strategy
    - We will use the constrained random stimulus to exercise the DUV
15. Abstraction Level
    - Stimulus will send at the interface of the module.
16. Checking
    - Reference model

Coverage

- Hazard detected from all possible stages
- Pipeline stall generated
- Detect hazard from possible decoding sources

Scenarios

- Generate a hazard from sources being used by other stages
- Single source hazard
- Double source hazard
- Ex stage hazard
- Mem stage hazard
- Wb stage hazard