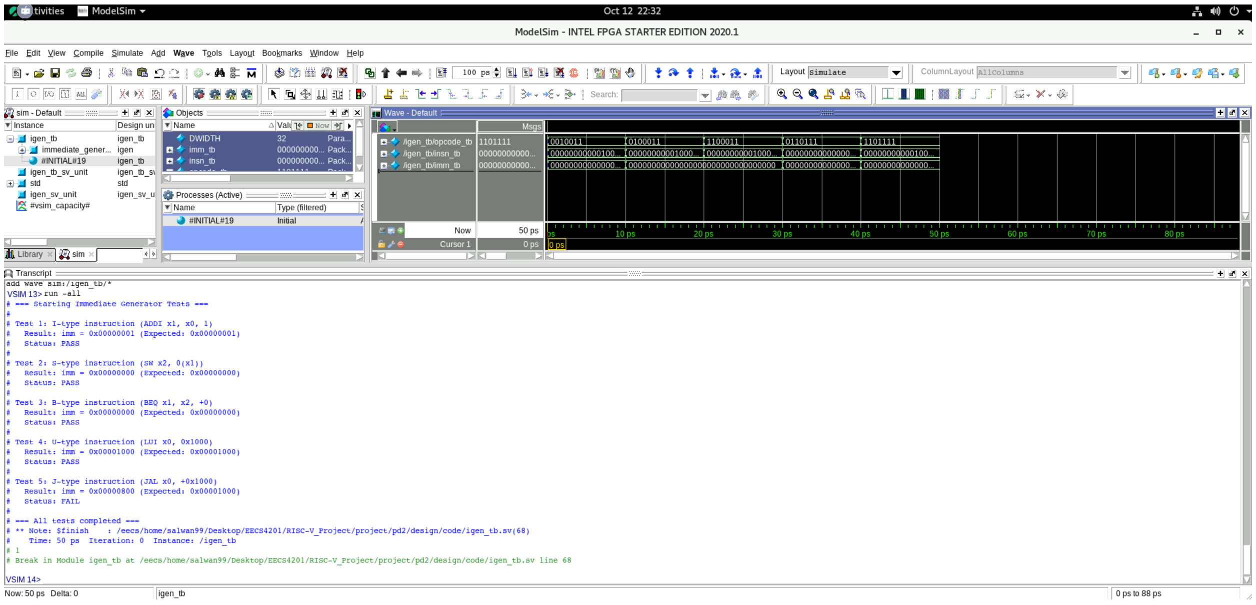
PD2 Report Document

Team Name: MuxMasters
Salwan Aldhahab (218597963)
Mavra Muzmmal (219075969)

EECS 4201 E
Prof. Anirudh M Kaushik
Lassonde School of Engineering
York University

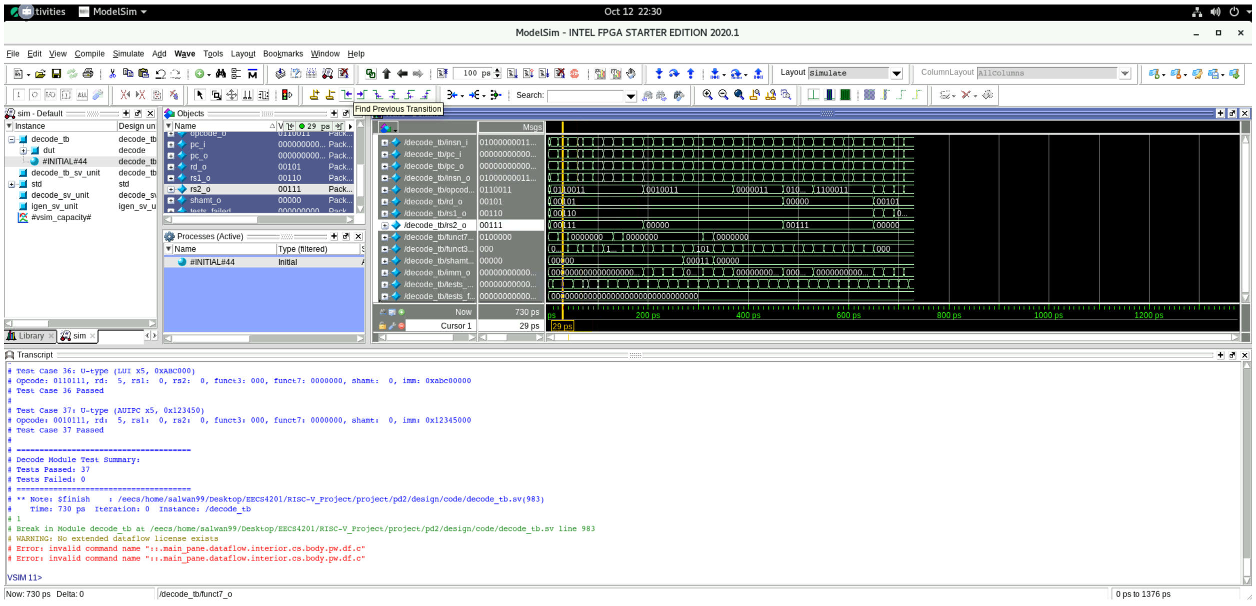
ModelSim Our Testbench Waveform Screenshots:



```
add wave sim:/igen_tb/*
VSIM 13> run -all
# === Starting Immediate Generator Tests ===
#
# Test 1: I-type instruction (ADDI x1, x0, 1)
#   Result: imm = 0x00000001 (Expected: 0x00000001)
#   Status: PASS
#
# Test 2: S-type instruction (SW x2, 0(x1))
#   Result: imm = 0x00000000 (Expected: 0x00000000)
#   Status: PASS
#
# Test 3: B-type instruction (BEQ x1, x2, +0)
#   Result: imm = 0x00000000 (Expected: 0x00000000)
#   Status: PASS
#
# Test 4: U-type instruction (LUI x0, 0x1000)
#   Result: imm = 0x00001000 (Expected: 0x00001000)
#   Status: PASS
#
# Test 5: J-type instruction (JAL x0, +0x1000)
#   Result: imm = 0x00000800 (Expected: 0x00001000)
#   Status: FAIL
#
# === All tests completed ===
# ** Note: $finish    : /eecs/home/salwan99/Desktop/EECS4201/RISC-V_Project/project/pd2/design/code/igen_tb.sv(68)
#    Time: 50 ps  Iteration: 0  Instance: /igen_tb
# 1
# Break in Module igen_tb at /eecs/home/salwan99/Desktop/EECS4201/RISC-V_Project/project/pd2/design/code/igen_tb.sv line 68

VSIM 14>
```

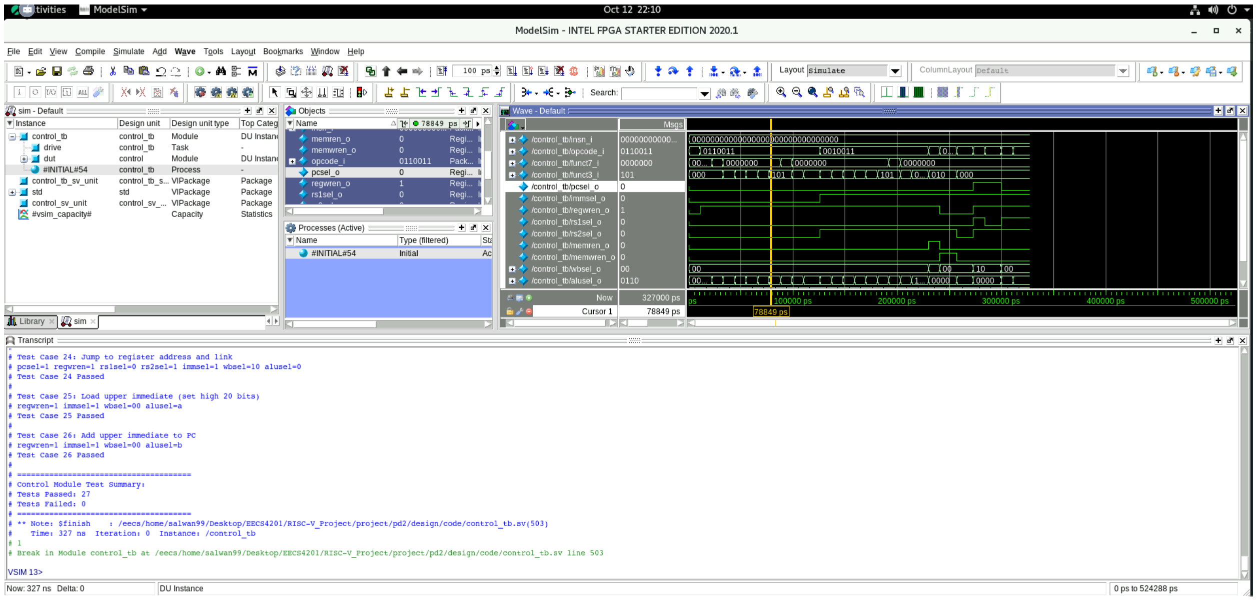
igen_tb.sv



```
# Test Case 36: U-type (LUI x5, 0xABC000)
# Opcode: 0110111, rd:  5, rs1:  0, rs2:  0, funct3: 000, funct7: 0000000, shamt:  0, imm: 0xabc00000
# Test Case 36 Passed
#
# Test Case 37: U-type (AUIPC x5, 0x123450)
# Opcode: 0010111, rd:  5, rs1:  0, rs2:  0, funct3: 000, funct7: 0000000, shamt:  0, imm: 0x12345000
# Test Case 37 Passed
#
# ========================================
# Decode Module Test Summary:
# Tests Passed: 37
# Tests Failed: 0
# ========================================
# ** Note: $finish    : /eecs/home/salwan99/Desktop/EECS4201/RISC-V_Project/project/pd2/design/code/decode_tb.sv(983)
#    Time: 730 ps  Iteration: 0  Instance: /decode_tb
# 1
# Break in Module decode_tb at /eecs/home/salwan99/Desktop/EECS4201/RISC-V_Project/project/pd2/design/code/decode_tb.sv line 983
# WARNING: No extended dataflow license exists
# Error: invalid command name "::.main_pane.dataflow.interior.cs.body.pw.df.c"
# Error: invalid command name "::.main_pane.dataflow.interior.cs.body.pw.df.c"

VSIM 11>
```

decode_tb.sv



```
# Test Case 24: Jump to register address and link
# pcsel=1 regwren=1 rs1sel=0 rs2sel=1 immsel=1 wbsel=10 alusel=0
# Test Case 24 Passed
#
# Test Case 25: Load upper immediate (set high 20 bits)
# regwren=1 immsel=1 wbsel=00 alusel=a
# Test Case 25 Passed
#
# Test Case 26: Add upper immediate to PC
# regwren=1 immsel=1 wbsel=00 alusel=b
# Test Case 26 Passed
#
# ======================================
# Control Module Test Summary:
# Tests Passed: 27
# Tests Failed: 0
# ======================================
# ** Note: $finish    : /eecs/home/salwan99/Desktop/EECS4201/RISC-V_Project/project/pd2/design/code/control_tb.sv(503)
#    Time: 327 ns  Iteration: 0  Instance: /control_tb
# 1
# Break in Module control_tb at /eecs/home/salwan99/Desktop/EECS4201/RISC-V_Project/project/pd2/design/code/control_tb.sv line 503

VSIM 13>
```

Control_tb.sv

Challenges Faced During PD2 Deliverables:

During the PD2 the first challenge we faced was with instruction memory. When we tried to run the professor's test, memory.sv thrown so many errors. We discovered the problem wasn't the memory file but the command we used to run it, I figured this out after talking with TA. The memory file was never loaded as intended because we were running/compiling with the wrong command line. The warnings disappeared and the memory behvaed as expected after fixing the command (proper file list, include path, and compile order). That was a great reminder that tool setup can look like a logic bug until you verify the flow.

The second main challenge we faced was with the decode stage. At first we wasn'r sure where to put the immediate generator, is it inside the decode or as a separate block in the pd2.sv. We began by inserting it into the decoder; however, because the initial run of the decoder was written sequentially, the outputs were delayed, and the checks contained a large number of zeros. The time problems were resolved after switching to entirely combinational logic, although the tests continued to have decode mismatch problems. Since a few instruction situations were incorrect (and shift/immediate handling needed attention), we were forced to re-audit field extraction and opcodes/funct mappings. The decode outputs eventually matched the specification when those circumstances were fixed.

Writing the testbenches was the last challenge. It required some rounds to figure out a clear strategy, including what to print, how to organize pass/fail tests, and how to cover sample instances. We finally found an order that worked, which accelerated debugging and made errors clear. All things considered, these glitches taught us to keep the decode strictly combinational, evaluate the tool flow early, and create straightforward, repeatable test patterns that highlight errors.