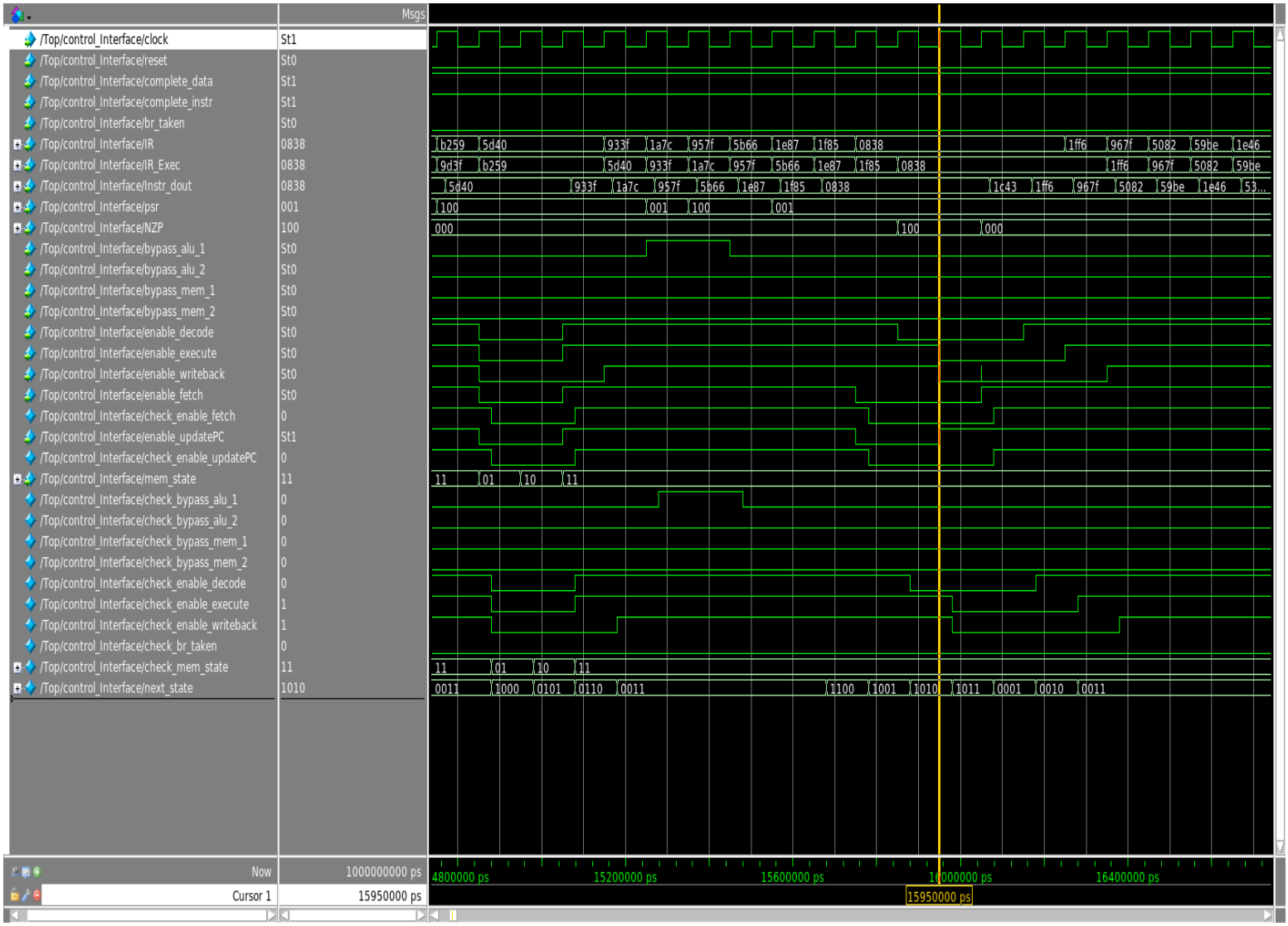
BUG REPORT

By Amogh Sakdeo, Radhika Sakhare, Parth Bhogate

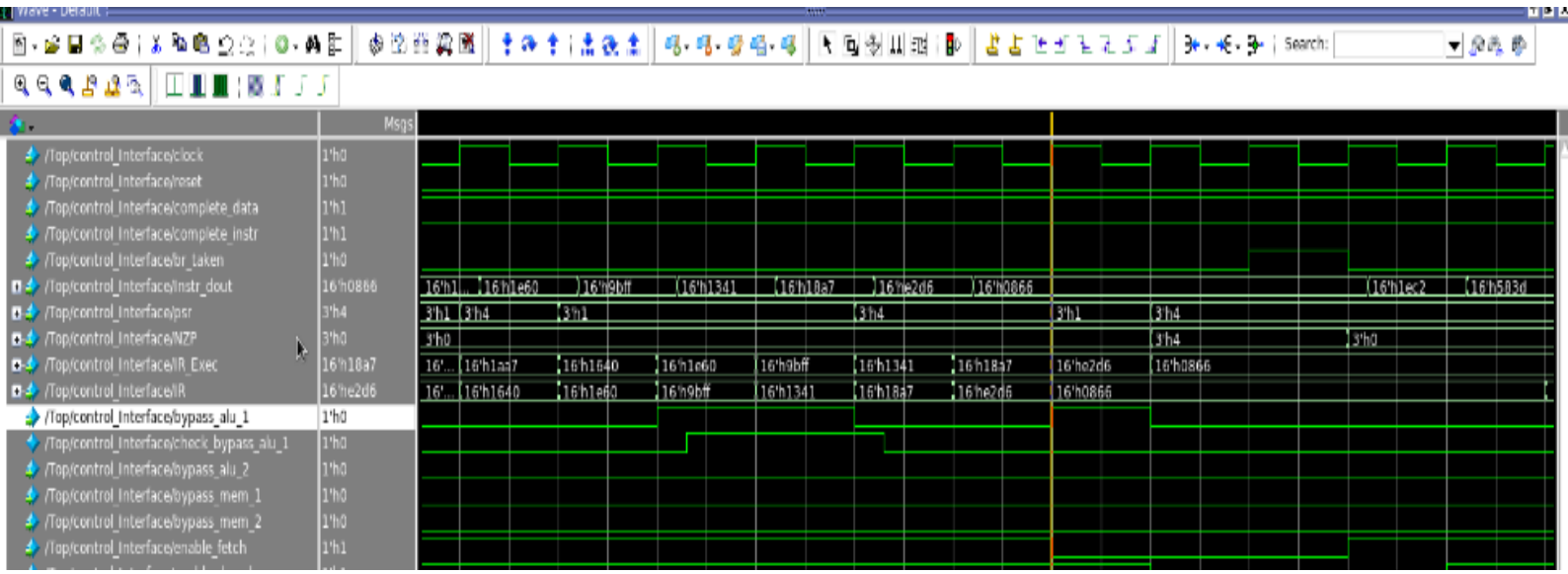
Bug 1 (data_defs_bug1.vp) Not taken branch behaving like a taken branch

The above bug is found in the controller block of the LC3 DUT. For a branch instruction appearing at the Instruction Dout (opcode: 4'b0000), the enable fetch and enable update PC signals go low, followed by enable decode going low in the next clock cycle and enable execute and writeback going low in the following clock cycle. For a not-taken branch, enable fetch and enable update PC then go high in the next clock cycle and others follow in the next cycles. If a branch is taken, it is detected at the execute stage and enable Update PC immediately goes high for the taken address. In this particular bug, for a not taken branch enable update PC goes high when the Instruction is in the execute stage hence staying low for only two clock cycles rather than staying low for three clock cycles.



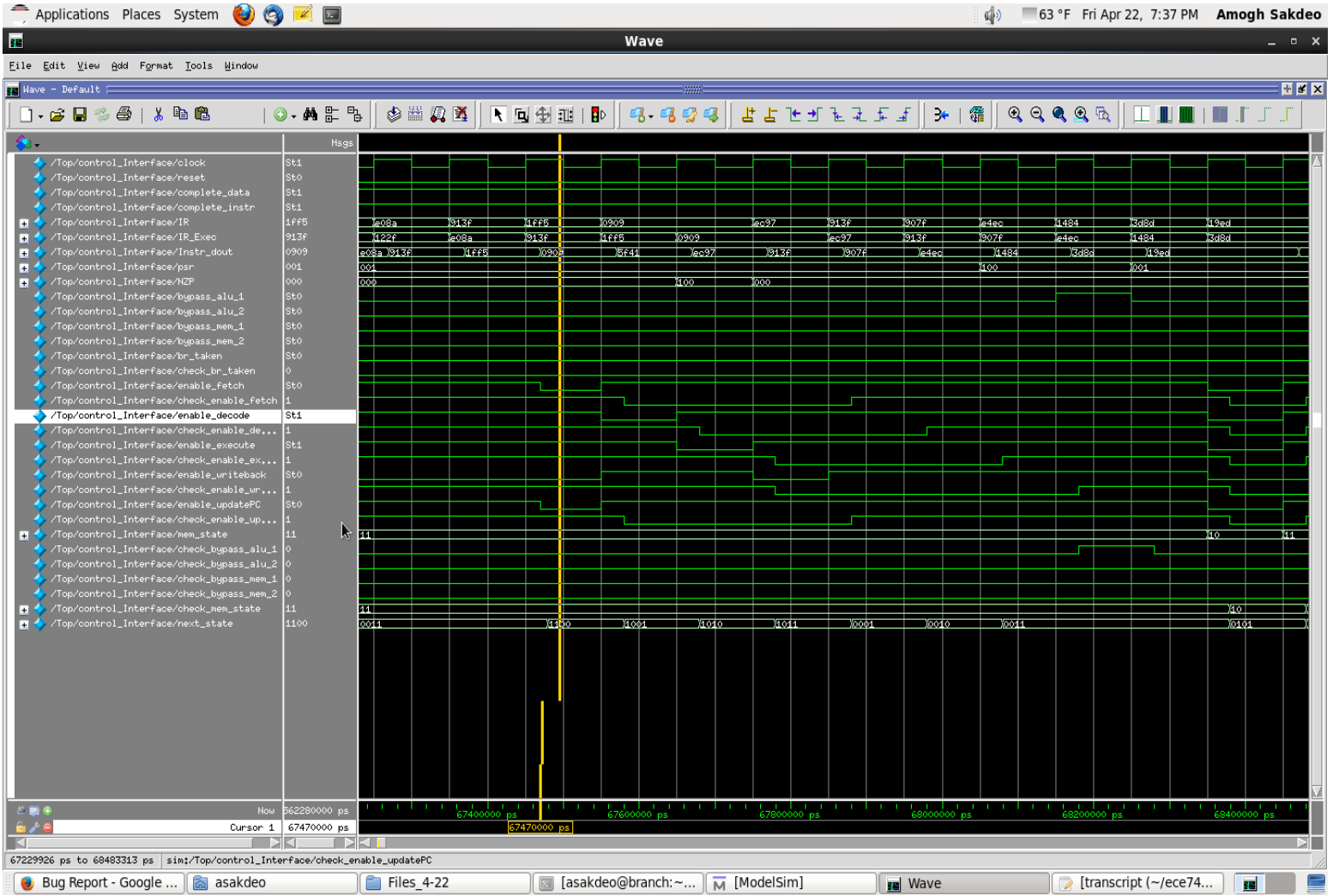
Bug 2 (data_defs_bug2.vp) Branch acting like a Jump checking for dependency IR[8:6]

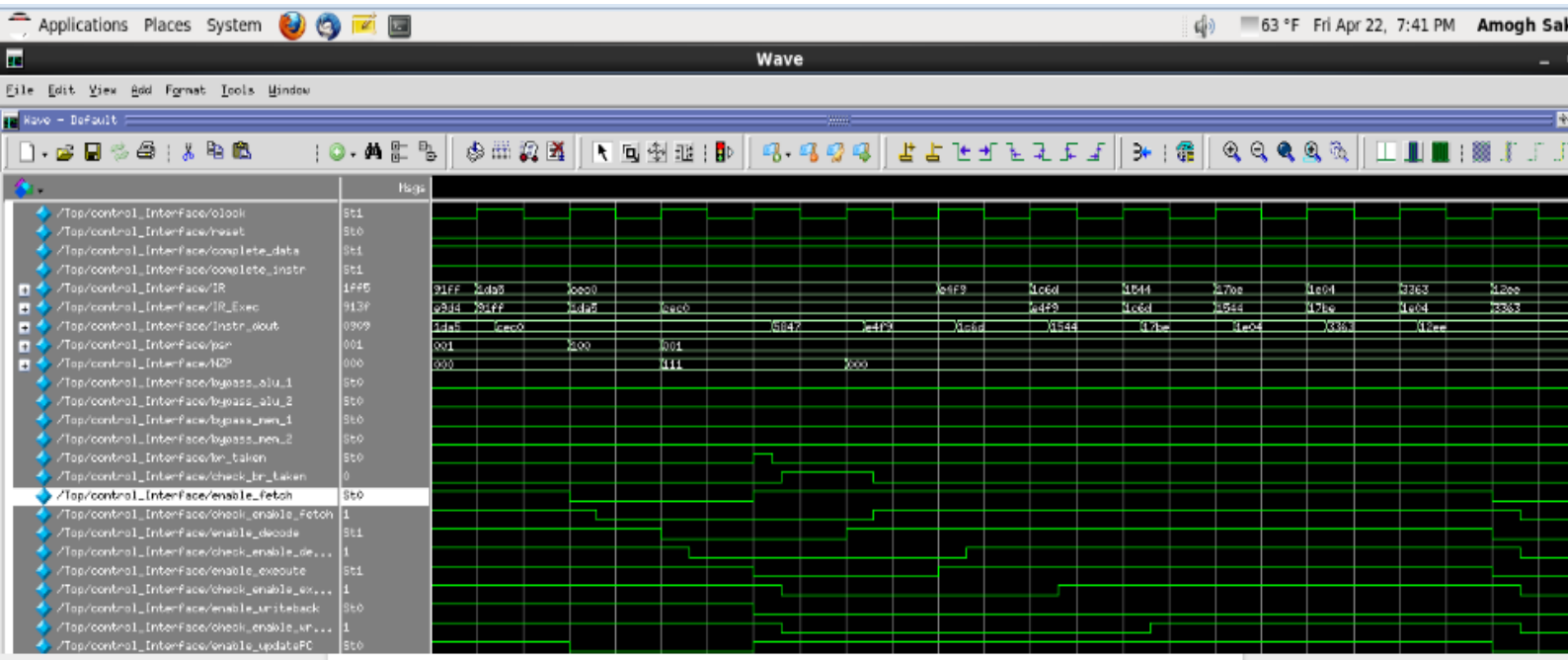
Bug 2 is found in controller block of the LC3 DUT. When a branch instruction is encountered at IR, there is no need to check for the bypass ALU signals since branch checks the value of previous register being worked upon and decides whether the Branch is taken or not. However in this particular bug, Bypass ALU1 goes high at times and it is not expected. It can be seen that Branch is behaving like a Jump and whenever IR[8:6] matches with IR_Exec[11:9] bypass ALU1 goes high implying that it is checking for a condition of destination followed by source register so that it can take the value from the Bypass of Execute stage. Instruction sequence: 16'h5000 followed by 16'h0400 can help detect.



Bug 3(data_defs_bug3.vp) Wrong Enable signal sequences for Control Instruction

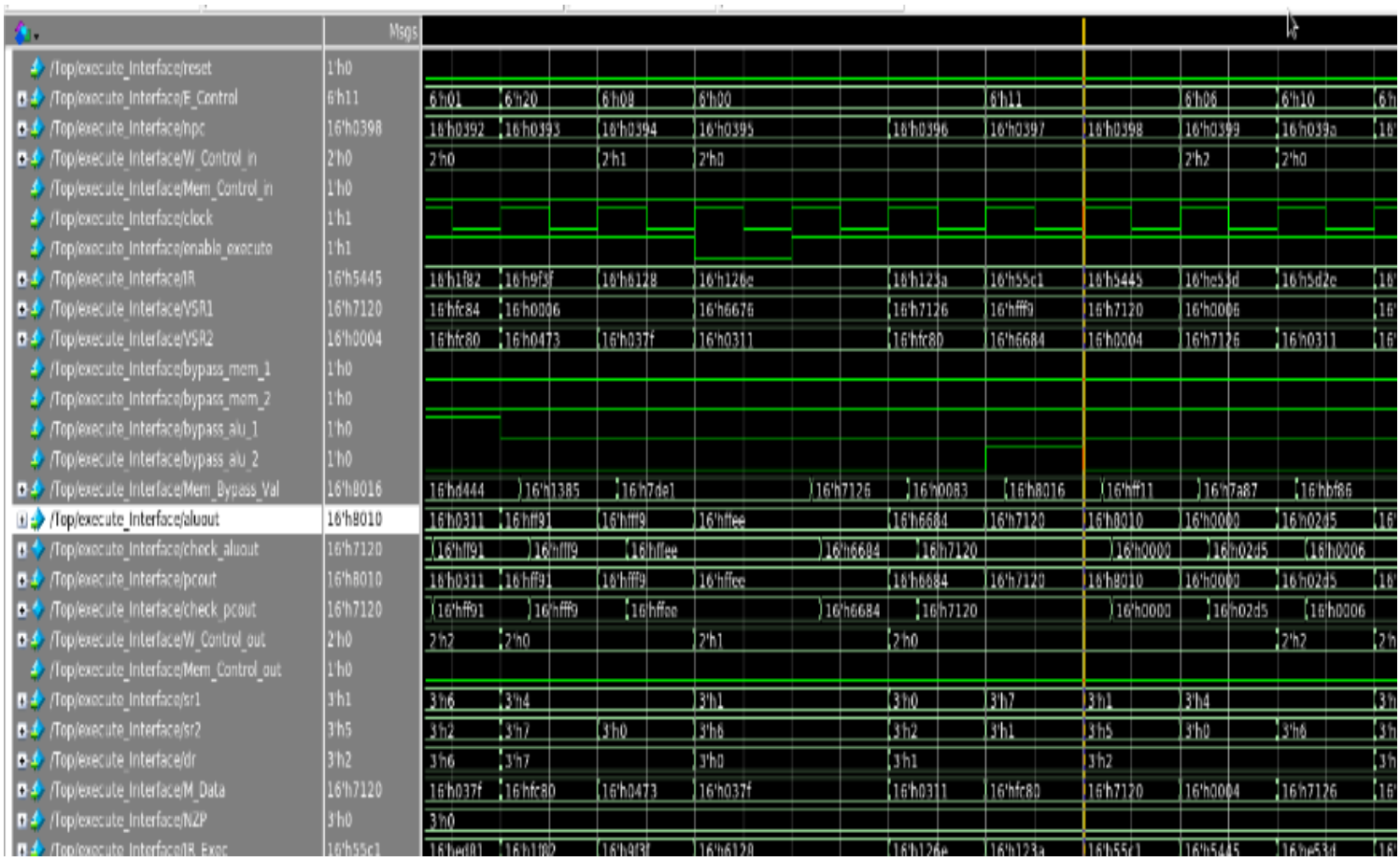
This bug is in the controller block of the DUT. For a control instruction appearing at Instr_dout enable fetch and update Pc go low, in the next clock cycle enable decode goes low and in the next clock cycle enable execute and writeback go low. In the following clock cycle all the enables start going high one by one. If it is a taken branch or a jmp enable goes high one clock cycle early (in the execute stage). However for this bug, when a first control instruction appears and is a taken one enable fetch also goes high along with enable update PC after 2 clock cycles. Because of which next instruction appears on the instr_dout and we don't see br_taken flag going high. All the other enables start becoming high in sequence after fetch becomes high in different clock cycles. Also enable writeback when it goes low is stuck at 0 until it encounters a next control instruction at the instr_dout. For a second control instruction enable fetch and update PC go low asynchronously and stay low only for one clock cycle, and other enable signals keep following the same behaviour in successive clock cycles in sequence. Since Enable fetch goes high so soon, instr_dout value changes early and the DUT does not compute Br_taken flag. These two sequences are then alternated for the further control instructions.

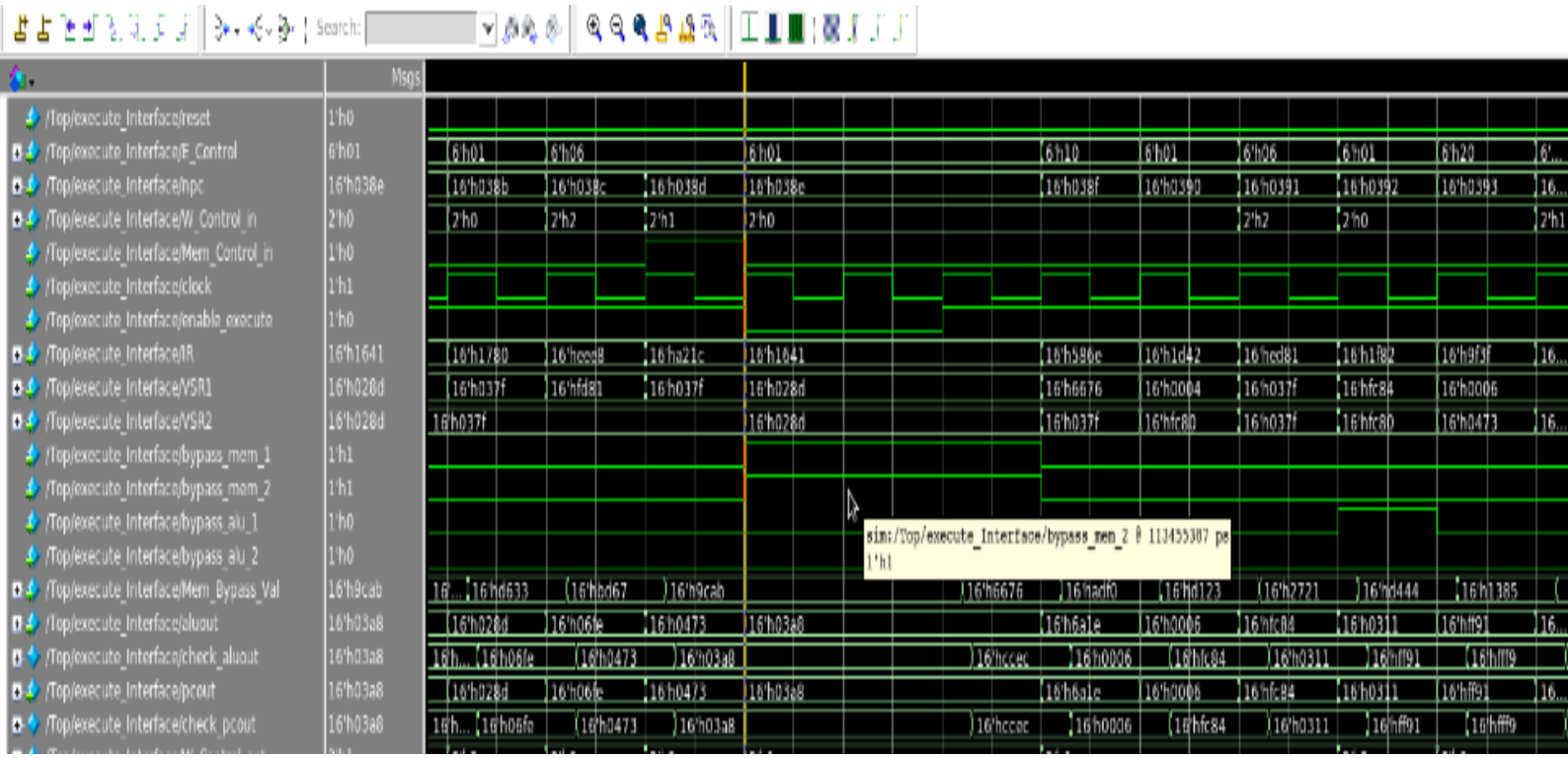



Applications Places System
63 °F Fri Apr 22, 7:41 PM Amogh Sal
Wave
File Edit View Add Format Tools Window
Wave - Default
Msgs
/Top/control_Interface/clock
/Top/control_Interface/reset
/Top/control_Interface/complete_data
/Top/control_Interface/complete_instr
/Top/control_Interface/IR
/Top/control_Interface/IR_Exec
/Top/control_Interface/Instr_dout
/Top/control_Interface/pc
/Top/control_Interface/NZP
/Top/control_Interface/bypass_alu_1
/Top/control_Interface/bypass_alu_2
/Top/control_Interface/bypass_mem_1
/Top/control_Interface/bypass_mem_2
/Top/control_Interface/br_taken
/Top/control_Interface/check_br_taken
/Top/control_Interface/enable_fetch
/Top/control_Interface/check_enable_fetch
/Top/control_Interface/enable_decode
/Top/control_Interface/check_enable_de...
/Top/control_Interface/enable_execute
/Top/control_Interface/check_enable_ex...
/Top/control_Interface/enable_writeback
/Top/control_Interface/check_enable_wr...
/Top/control_Interface/enable_updatePC

Bug 4 (data_defs_bug4.vp) Bypass ALU2 and Bypass mem2 values interchanged

This bug is found in the Execute block of LC3 DUT. When Bypass ALU1 or Bypass ALU2 is high, the values ALUout and PCout should be calculated from VSR1 or VSR2 respectively since they reflect the values from the ALU bypasses. Similarly when Bypass mem1 or bypass mem2 is high, ALUout and PCout should be calculated from the mem_bypass_val since it reflects the value from mem bypass. However in this particular case when Bypass ALU2 is high, value from Mem_bypass_val is being taken for calculating ALUout and PCout. Similarly, when Mem_bypass 2 is high, the value of previous ALUout is taken implying that it is acting like ALU bypass 2. Instruction sequence 16'h50c0 followed by 16'h1ab0 will help us detect this bug. The following waveforms show the two conditions that have been mentioned for this bug




Search:
Msgs
/Top/execute_Interface/reset
/Top/execute_Interface/E_Control
/Top/execute_Interface/npc
/Top/execute_Interface/WI_Control_in
/Top/execute_Interface/Mem_Control_in
/Top/execute_Interface/clock
/Top/execute_Interface/enable_execute
/Top/execute_Interface/IR
/Top/execute_Interface/VSR1
/Top/execute_Interface/VSR2
/Top/execute_Interface/bypass_mem_1
/Top/execute_Interface/bypass_mem_2
/Top/execute_Interface/bypass_alu_1
/Top/execute_Interface/bypass_alu_2
/Top/execute_Interface/Mem_Bypass_Val
/Top/execute_Interface/aluout
/Top/execute_Interface/check_aluout
/Top/execute_Interface/pcout
/Top/execute_Interface/check_pcout
sim:/Top/execute_Interface/bypass_mem_2 @ 113455307 ps
1'h1

Bug 5 (data_defs_bug5.vp) Z flag of NZP status register stuck at 0

This bug has been found in the execute block of the LC3 DUT. For a branch, every-time the NZP is calculated based on the value of Instruction register. IR[11:9] has the value of NZP associated with the branch instruction. For a JMP instruction the value of NZP is 3'b111, the DUT models this behaviour. For rest of the instructions, the value of NZP is 3'b000 which is also being modelled by the DUT. However for a branch instruction with IR[11:9] as 3'b010, 3'b000 is getting modelled by the DUT while for rest of the flags (3'b001, 3'b100), DUT works as expected. Thus NZP[1] is stuck at 0 implying that the zero flag of NZP is stuck at 0.