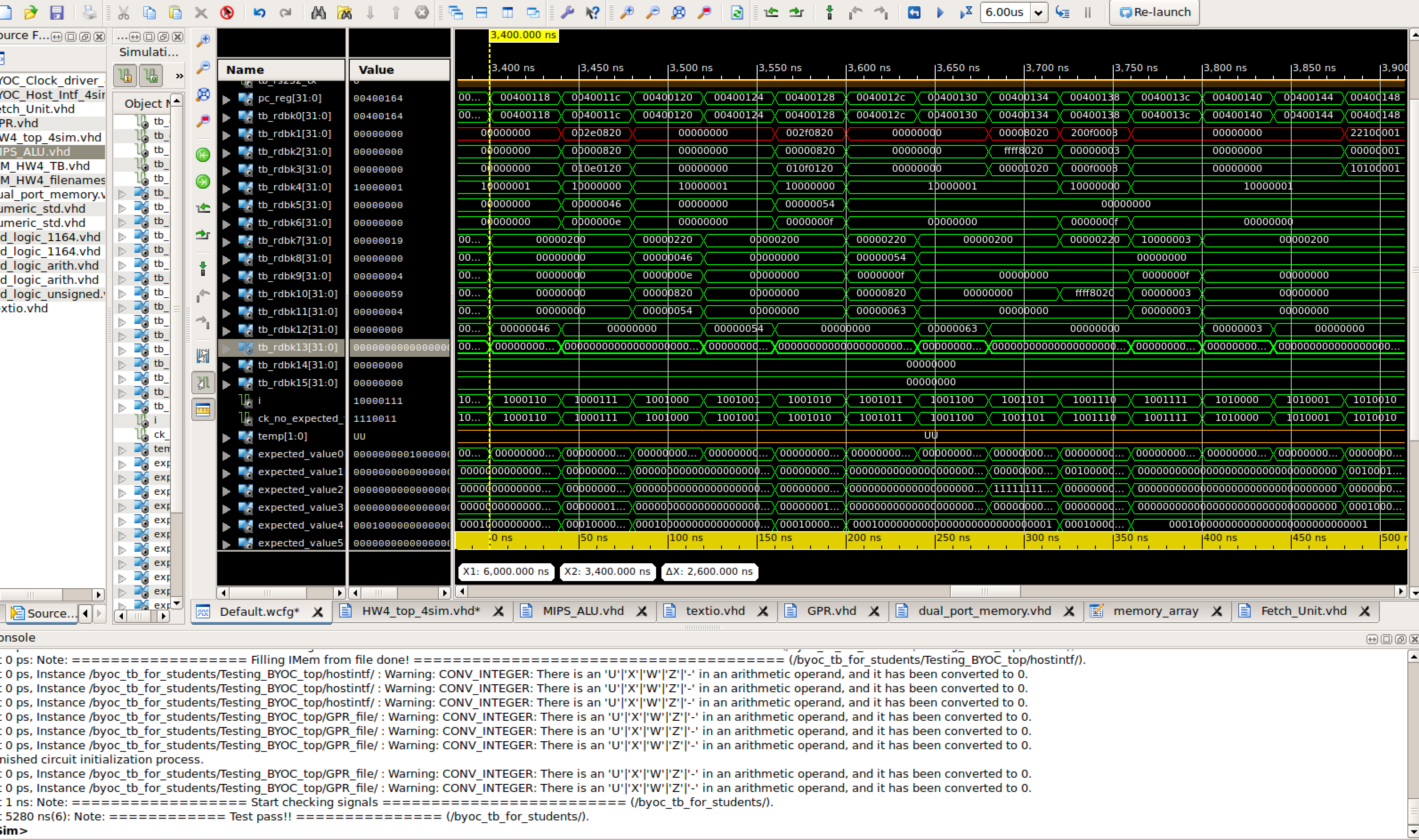
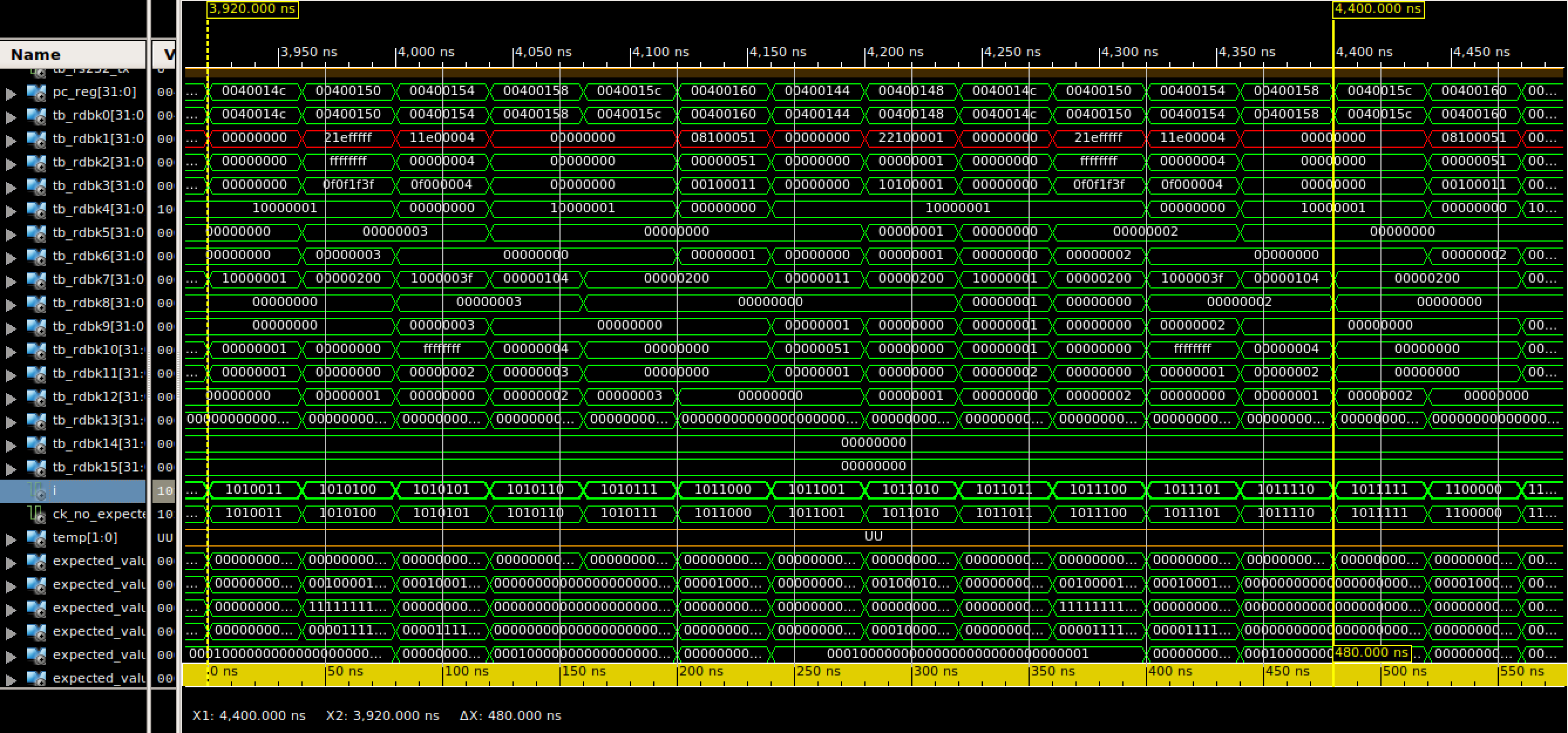
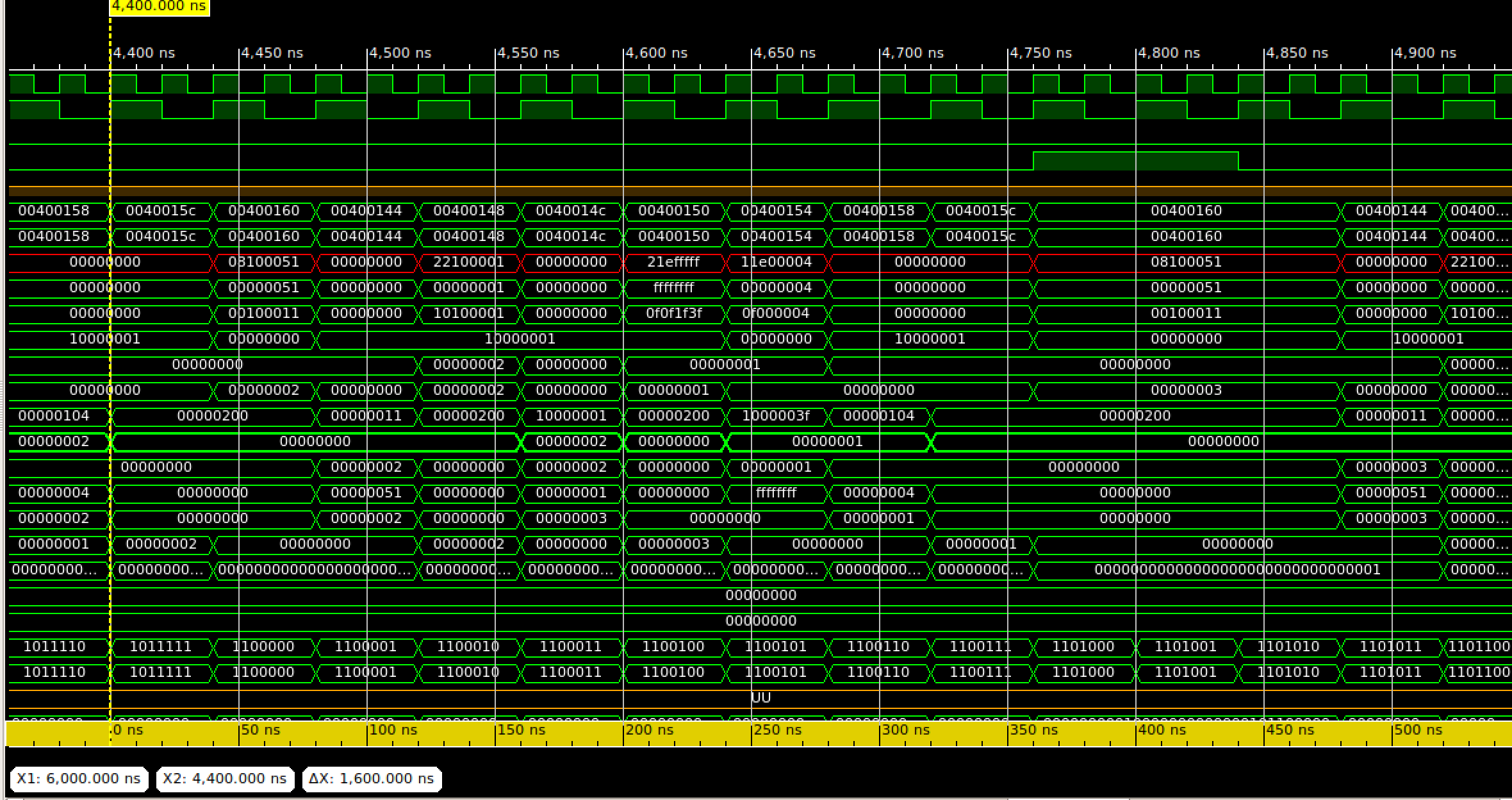
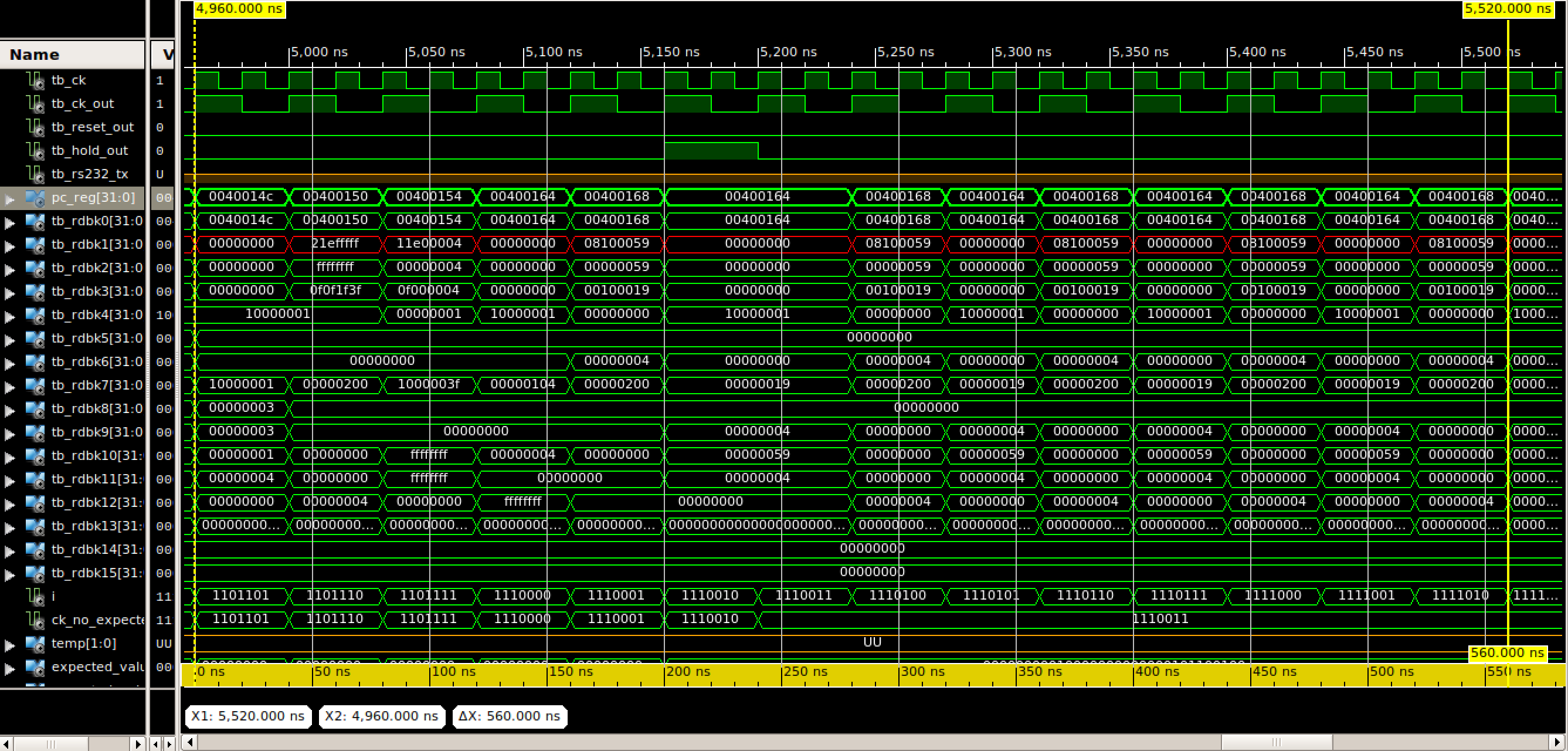
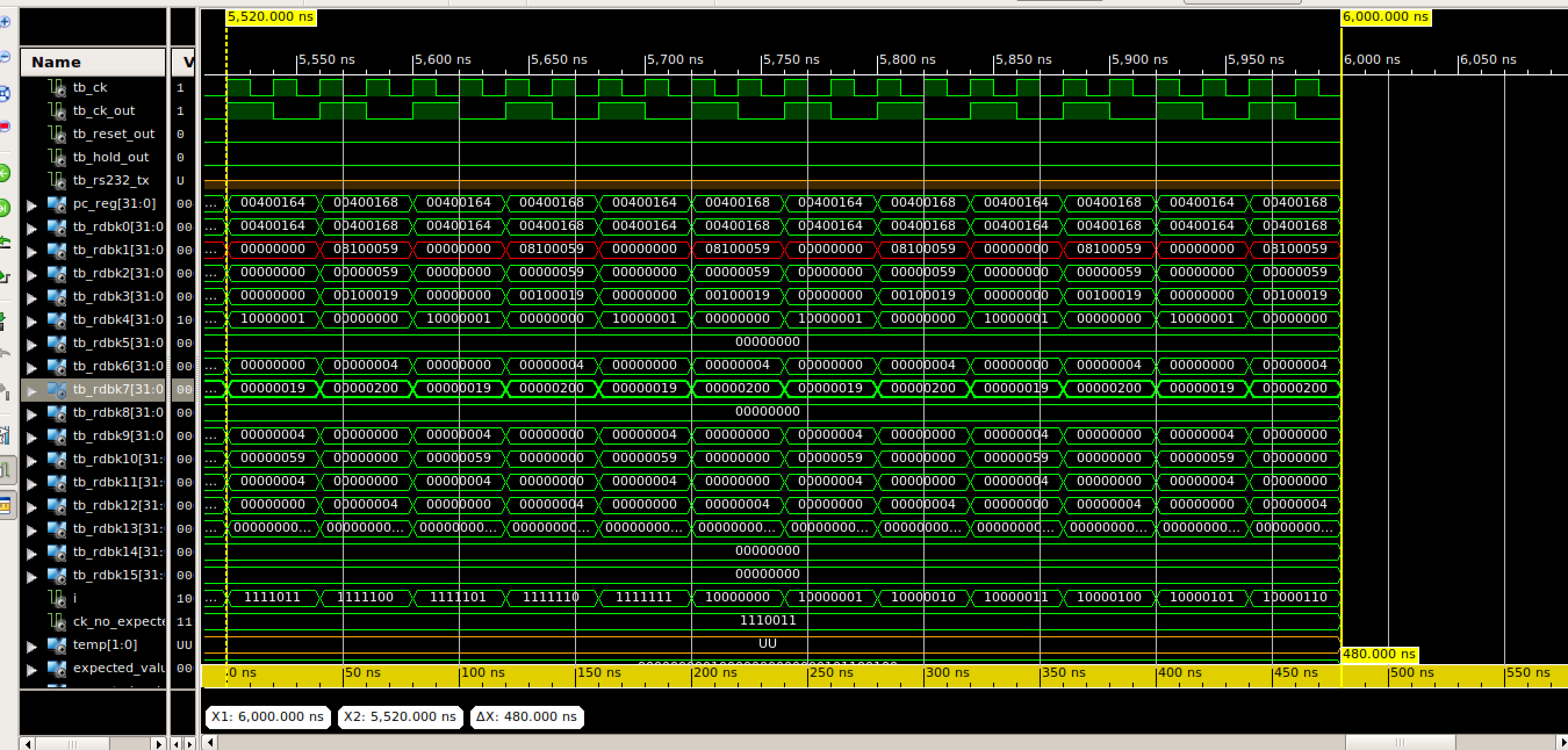
HW 4 – Simulation Report

I.D. 204200026  
I.D. 201322708  
I.D. 300267390

3.1 – run Simulation – Test Pass !

3.2

3.2 in the excel file

3.3. Support for branching operations made available by change to the fetch unit

-- PC\_source decoder (create the PC\_source signal)

process(opcode, PC\_source, Rs\_equals\_Rt\_pID, funct) --sensitive to changes in the opcode

begin

case opcode is

when b"000010" => PC\_source <= b"11"; --j

when b"000011" => PC\_source <= b"11"; --jal

when b"000100" =>

if(Rs\_equals\_Rt\_pID = '1') then

PC\_source <= b"01"; --beq

else

-- do nothing

PC\_source <= b"00";

end if;

when b"000101" =>

if(Rs\_equals\_Rt\_pID = '1') then

PC\_source <= b"01"; --bne

else

-- do nothing

PC\_source <= b"00";

end if;

when b"000000" =>

if funct = b"001000" then

PC\_source <= b"10"; -- jr

else

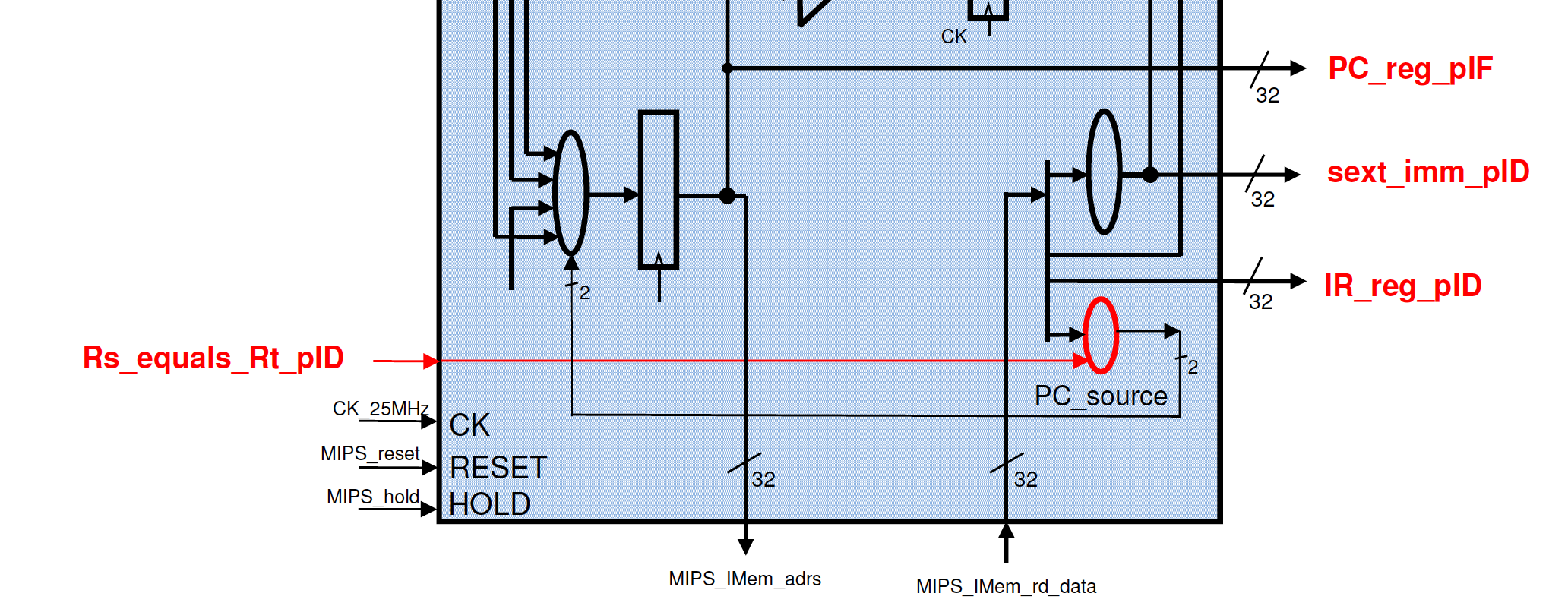
PC\_source <= b"00"; -- all other cases

end if;

when others => PC\_source <= b"00" ; -- all other commands

end case;

end process;

The introduction of the “Rs\_equsl\_Rt\_pID” signal and the addition of “if” (checking if Rs == Rt) conditions as seen above allow support the branching operations. The PC\_Source signal is set by the incoming opcode *AND* the result of the comparison of values done in the ALU. According to each instruction (be it either BEQ or BNE) the PC\_Source is set accordingly, if a jump should occur.   
As also explained in the ex. documents illustration: 

3.4 2 NOP operations are used in the code since in a pipelined MIPS cpu, where each cycle introduces another operation (IF) processed in the different phases, we must make sure we don’t cause hazards. The hazrad that will occur here if we don’t use 2 null operations is that the calculations done in the next command (after each command followed by 2 nops) is dependant on the WB values put into the registers used in it. In other words, since the next operations are always fetched in each cycle, to use the result stored in a register we must wait until it is written, and only then we can use its value in a next instruction.   
Therefore:  
if we want to use the values written to registers , we must wait until the WB phase is complete (2 CPU cycles: IF→ID→EX→WB, we must delay for 2 cycles until WB is done).  
(in our case, we write to $at, $v1 etc…) meaning that using the values stored in these registers before 2 nop commands will cause a data hazard and incorrect results. (incorrect in terms of logic, not hardware functioning)

3.5. The program does not check the sub, or, and, xor and slt instructions implemented in our R-type instruction set. This does not mean that the operations performed by the commands are not used since bneq, and beq use substraction to check for equality.