# Report

For EE126 assignment5

The goal of this project is to design a pipeline version of a ARM processor that can overcome data-hazard using forwarding and stalling.

A pipeline processor is a processor that carries out one instruction in 5 clock cycles, keep every part of the processor busy with current and latter instruction.

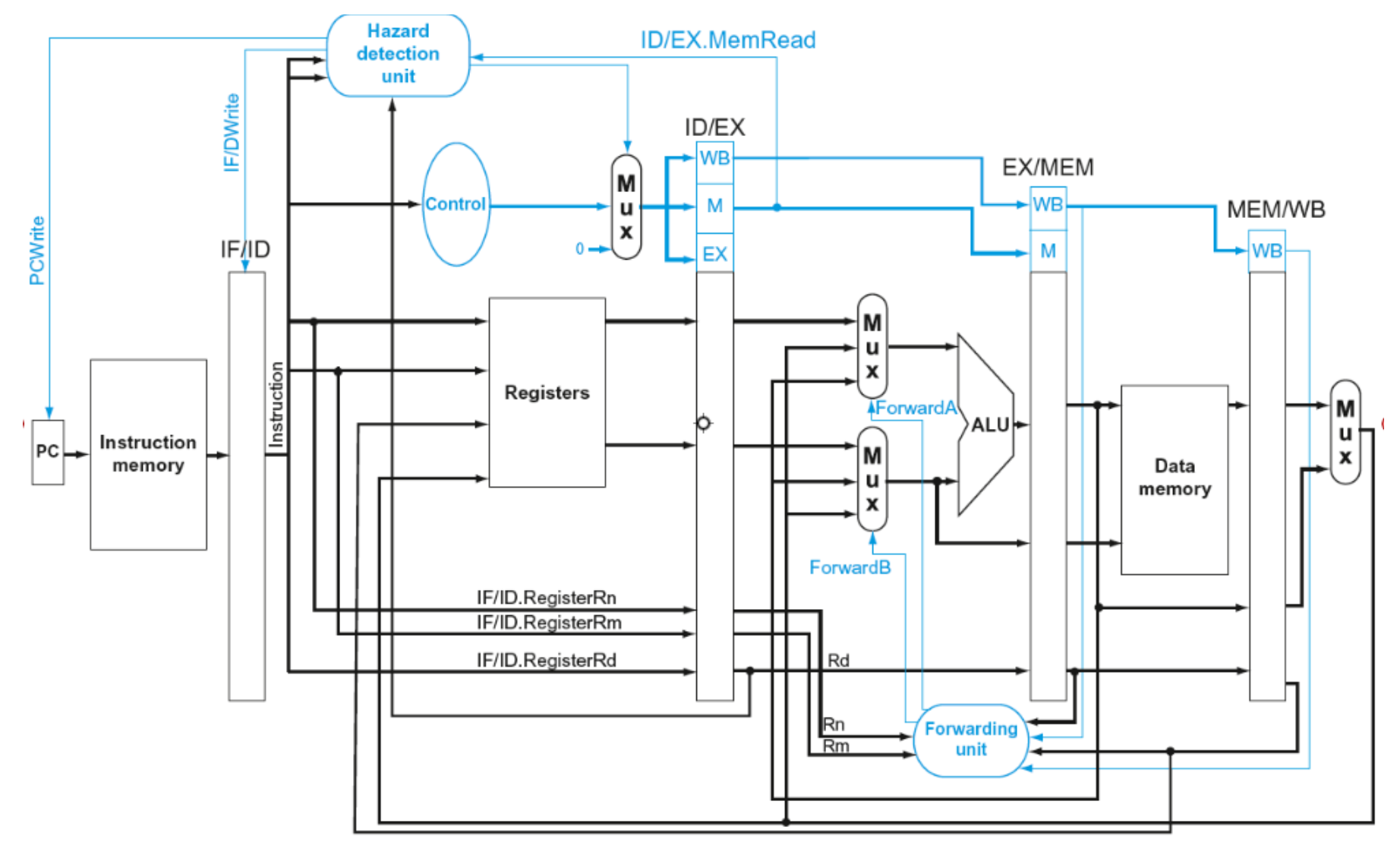
Provide waveforms—along with annotations, labels, and descriptions—that show the successful execution of the program defined later in this document. Be sure to:

• Include the most relevant signals in your waveform, including those related to overcoming data hazards

• Point out key events such as

– When the pipeline is stalled

– When values are forwarded  
• Clearly show what pipeline-stage each instruction is in



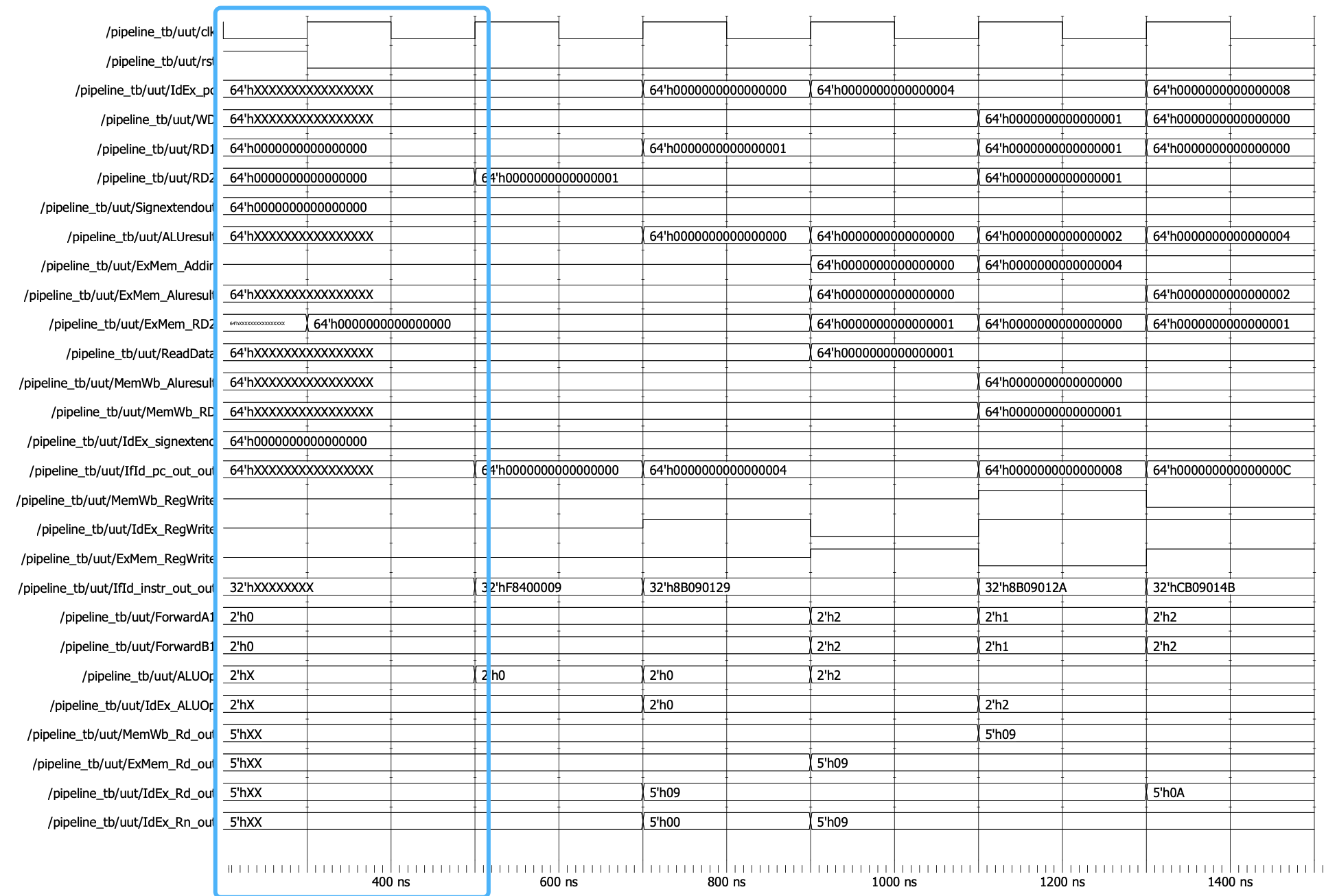
(Processor overview)

In the first CC(Clock Cycle), we set “reset” as ‘1’ to initiate whole processor.

In the second CC, the processor begins to operate test program.

The first stage of LDUR X9, [XZR, 0]:

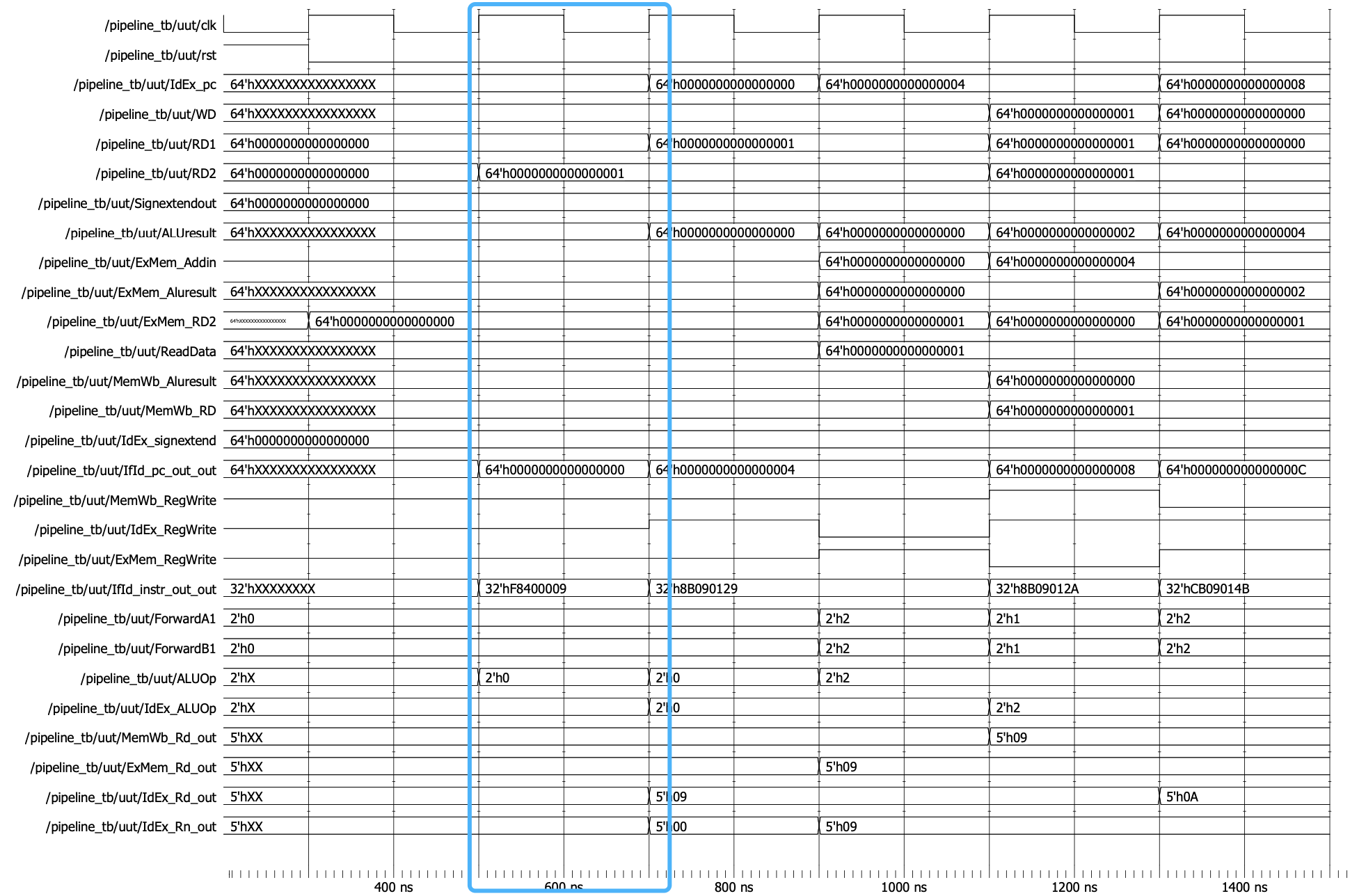
Read the PC and find the corresponding 32 bits address and then store them into if/id.



(First two Clock Cycles )

In the third CC, the second instruction(ADD) begins to operation like the behavior of first one.

The second stage of “LDUR”: hazard unit receive operand, generate signal like PCWrite (set as ‘0’) to stop next instruction ADD from moving forward.



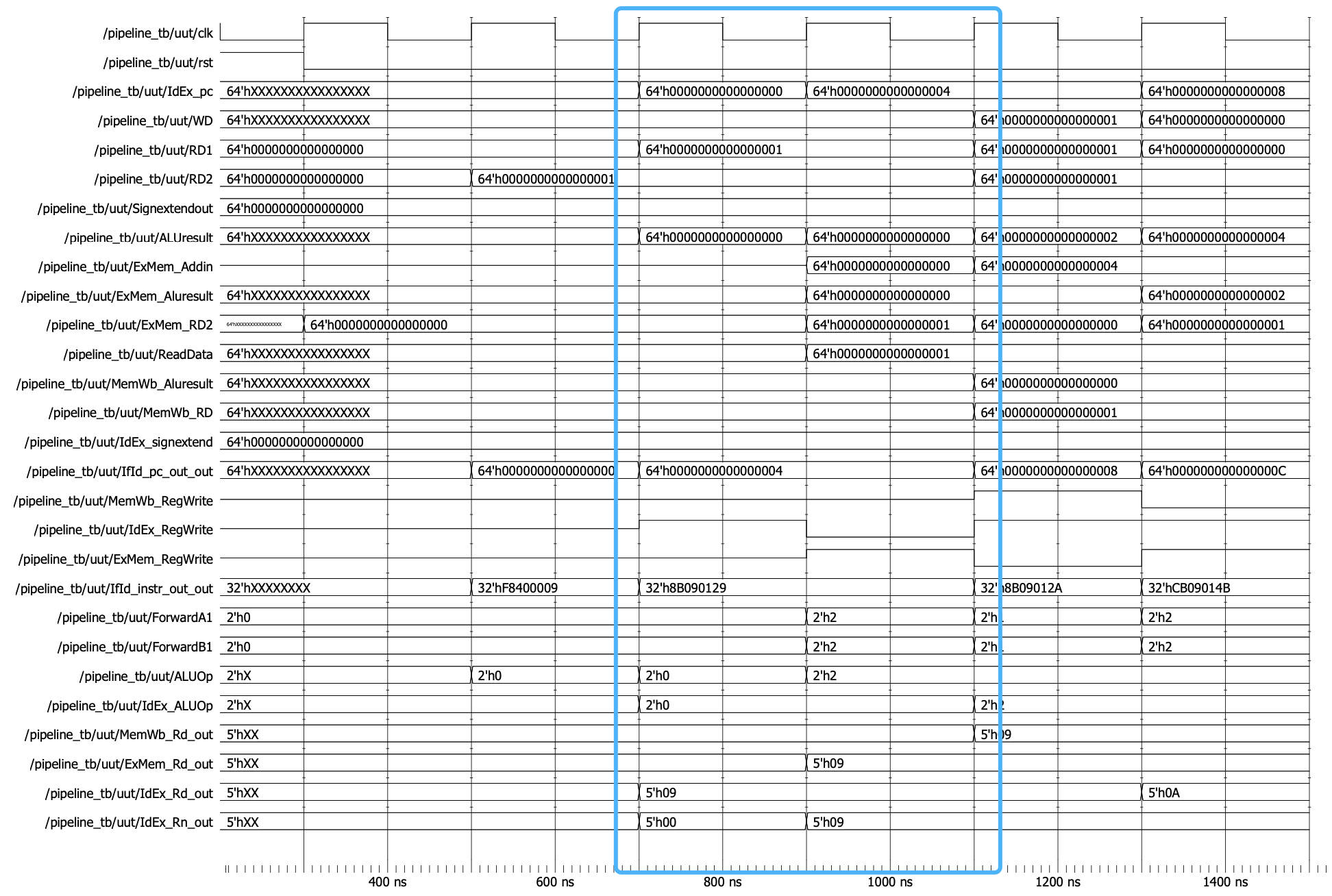
(The third Clock Cycle )

Combine the forth and fifth CC together to see what’s going on under the influence of hazard unit.

In the third stage of LDUR, it starts to compute memory address, and forth stage, it load the value of X9. In the meantime, the second instruction(ADD) should be in ID stage, but because of the hazard unit detect the potential issue, so it stalls it for one cycle, and at the point of 900ns, the second instruction enter ID stage.

Obviously, the value of x9 is just read from memory by LDUR, so the ADD instruction get a wrong value from register(which is data hazard).

The below blue rectangle shows that PC address remains the same before being sent to decode.

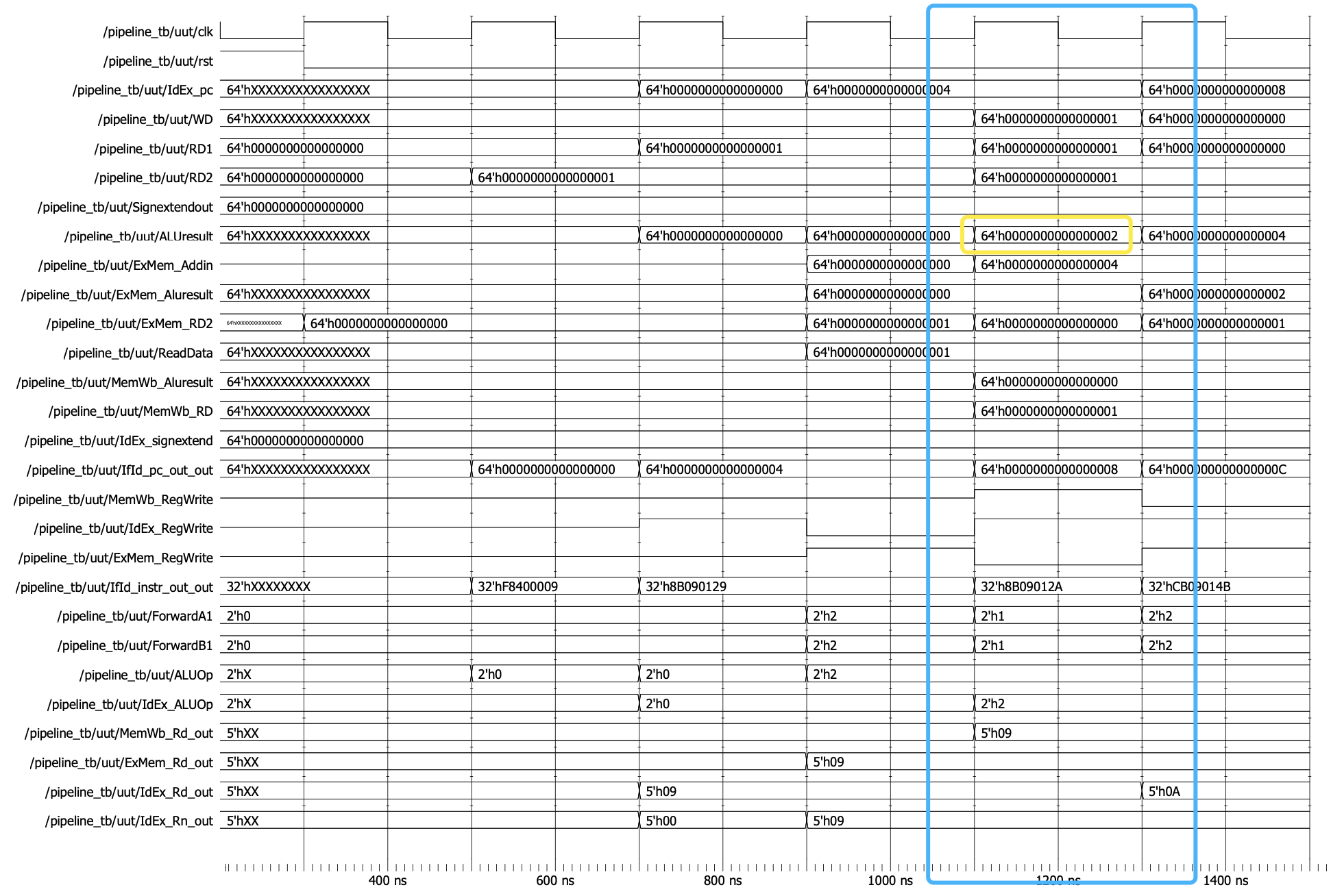


(The forth and fifth Clock Cycles)

In the sixth CC, LDUR enter WB stage, and the second instruction in the EX stage.

To fix the incorrect caused by data hazard, forward unit take effect in this cycle. It compare the RD of ADD instruction and the RD of LDUR, finding that they are the same, so it forwards the value from memory to ALU input.

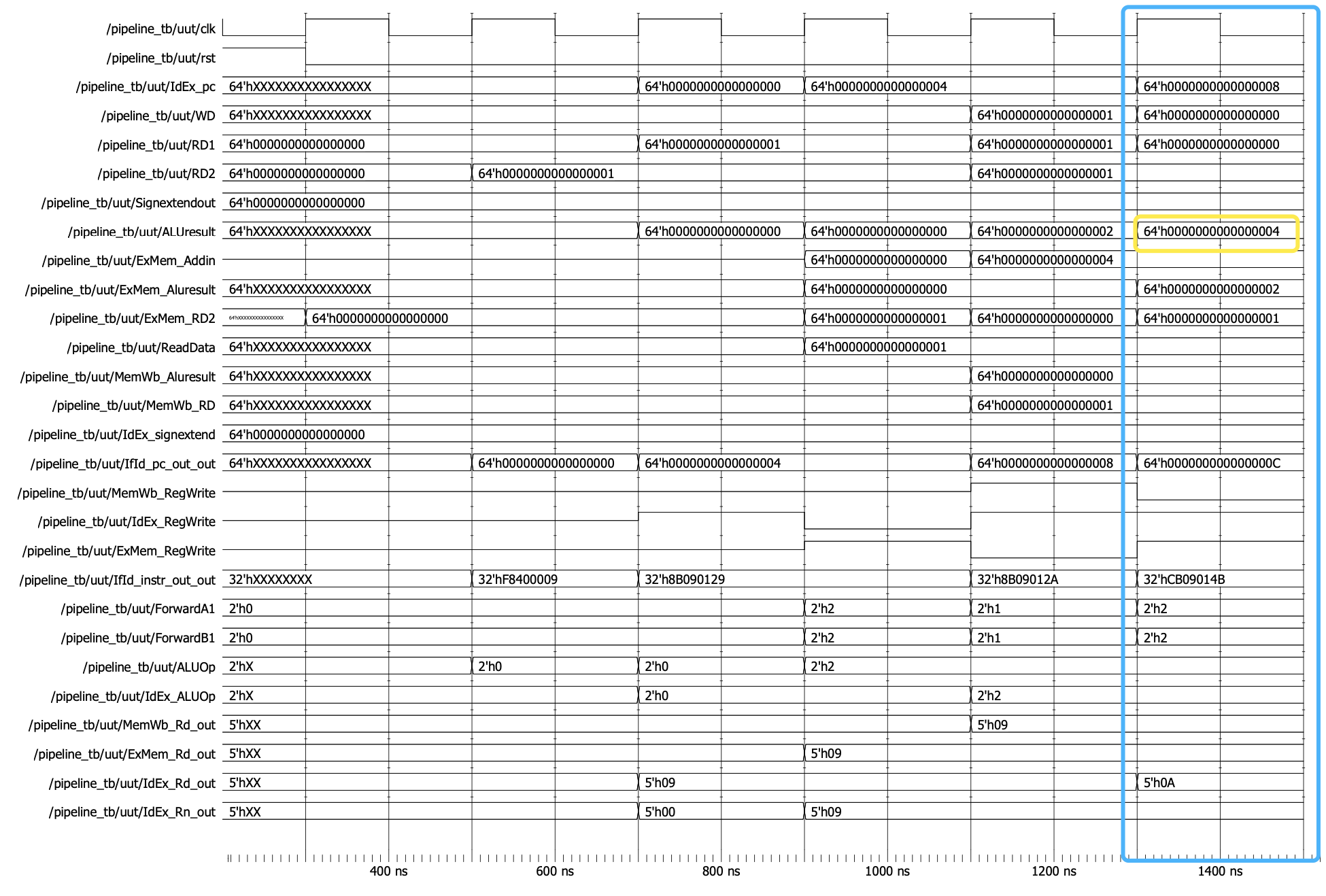
From the yellow rectangle, the correct value comes out.



(The sixth Clock Cycle)

In the seventh CC, the second instruction is in the MEM stage with output of the value of X9. while the third instruction is behind one cycle, and unfortunately it uses X9 as Rn. As previous saying, data hazard rise again.

The forward unit use forwardA and forwardB as choice signal to select previous ALU result as input of ALU and get the correct result shows in the yellow rectangle.



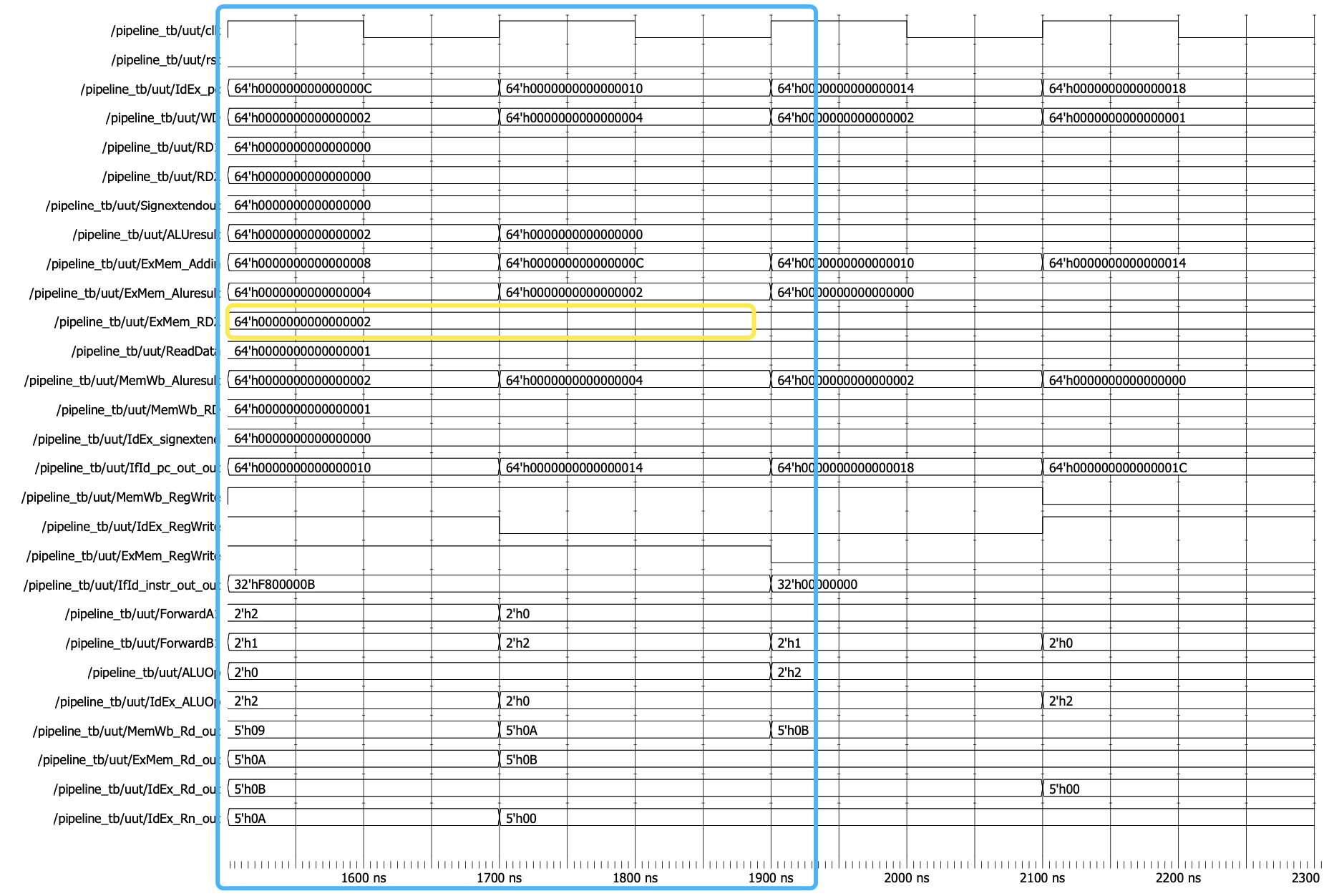
(The seventh Clock Cycle )

The sequential cycle is the same as before, using forward unit to fix the potential data hazard caused by SUB following ADD.

Let’s focus on STUR instruction then.

The STUR should store X11 value to memory, but the value of X11 was calculated by SUB instruction just now. So the data hazard comes again.

Forward unit also by detecting two instructions RDs determines that forward the result of ALU back to its input to make sure it get the right value to store. Similarly, the second store instruction use the value forwarded from mem/wb register to compete the behavior correctly.(yellow rectangle)



(ALU & Store data hazard )