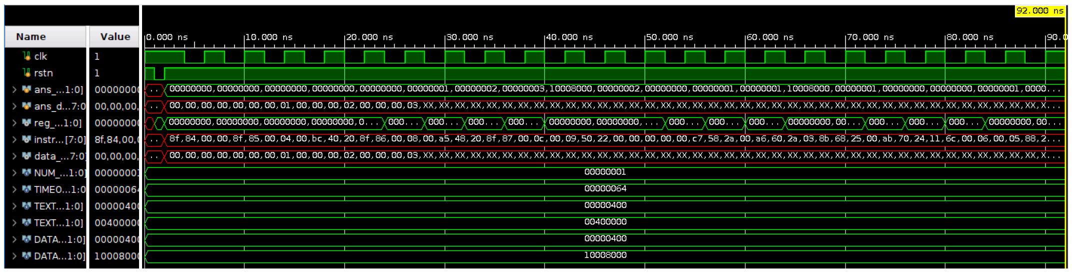
Report

1. Experimental Result

a. Show the waveform screen shot of the test we provided.



b. What other cases you've tested? Why you choose them?

Test addi function.

20 11 00 01 // addi $11, $0, 0x0001

20 12 00 02 // addi $12, $0, 0x0002

20 13 00 03 // addi $13, $0, 0x0003

2. Answer the following Questions.

a. For each code sequence below, state whether it must stall, can avoid stalls using only forwarding, or can execute without stalling or forwarding.

Sequence 1	Sequence 2	Sequence 3
lw $t0,0($t0)	add $t1,$t0,$t0	addi $t1,$t0,#1
add $t1,$t0,$t0	addi $t2,$t0,#5	addi $t2,$t0,#2
	addi $t4,$t1,#5	addi $t3,$t0,#2
		addi $t3,$t0,#4
		addi $t5,$t0,#5

Sequence 1

Since the second instruction depends on the result of the first instruction (the value loaded into $t0), forwarding is necessary to avoid stalls. Without forwarding, a stall would occur. So, it can avoid stalls using only forwarding.

Sequence 2

The second and third instructions do not depend on each other, but the third instruction depends on the result of the first instruction. Therefore, forwarding is required to avoid a stall for the third instruction. The second instruction can execute without stalling or forwarding, but the third instruction must stall or use forwarding to avoid stalls.

Sequence 3

Each instruction in this sequence is independent of the others. There are no dependencies between instructions, so no stalls are necessary, and forwarding is not required. Therefore, this sequence can execute without stalling or forwarding.

b. Explain the difference between throughput and latency.

   Throughput measures the overall rate of task completion or data processing within a system over time, while latency measures the time delay experienced by individual tasks or operations. Higher throughput indicates greater processing capacity, while lower latency indicates faster response times. Both metrics are essential for evaluating the performance and efficiency of computing systems.

c. A group of students were debating the efficiency of the five-stage pipeline when one student pointed out that not all instructions are active in every stage of the pipeline. After deciding to ignore the effects of hazards, they made the following five statements. Which ones are correct? Explain why or why not.

   Ans: 2 ,3 and 4 are correct.

   i. Allowing jumps, branches, and ALU instructions to take fewer stages than the five required by the load instruction will increase pipeline performance under all circumstances.

      False. Since the 'load' instruction still requires five cycles, it will not change, and we need to consider the maximum cycle count. Therefore, the performance remains unchanged.

   ii. Trying to allow some instructions to take fewer cycles does not help, since the throughput is determined by the clock cycle; the number of pipe stages per instruction affects latency, not throughput.

      True. The total time is (IC - 1 + number of stages) * cycle time. Generally, IC is much larger than the number of stages, so the stages can be ignored. Therefore, the overall performance (throughput) depends on the cycle time. Reducing the number of stages only affects the latency of a single instruction.

iii. You cannot make ALU instructions take fewer cycles because of the writeback of the result, but branches and jumps can take fewer cycles, so there is some opportunity for improvement.
True. ALU instructions typically involve multiple stages, including the execution stage and the writeback stage where the result is written back to a register. It's challenging to reduce the cycle time for ALU instructions due to dependencies and the need for proper data forwarding. However, branches and jumps can often be optimized to take fewer cycles, especially with efficient branch prediction techniques.

iv. Instead of trying to make instructions take fewer cycles, we should explore making the pipeline longer, so that instructions take more cycles, but the cycles are shorter. This could improve performance.
True. Making the pipeline longer can indeed improve performance. This is because the time taken for each cycle (clock cycle time) is reduced, allowing more instructions to be processed in the same amount of time. However, this approach also increases the complexity.