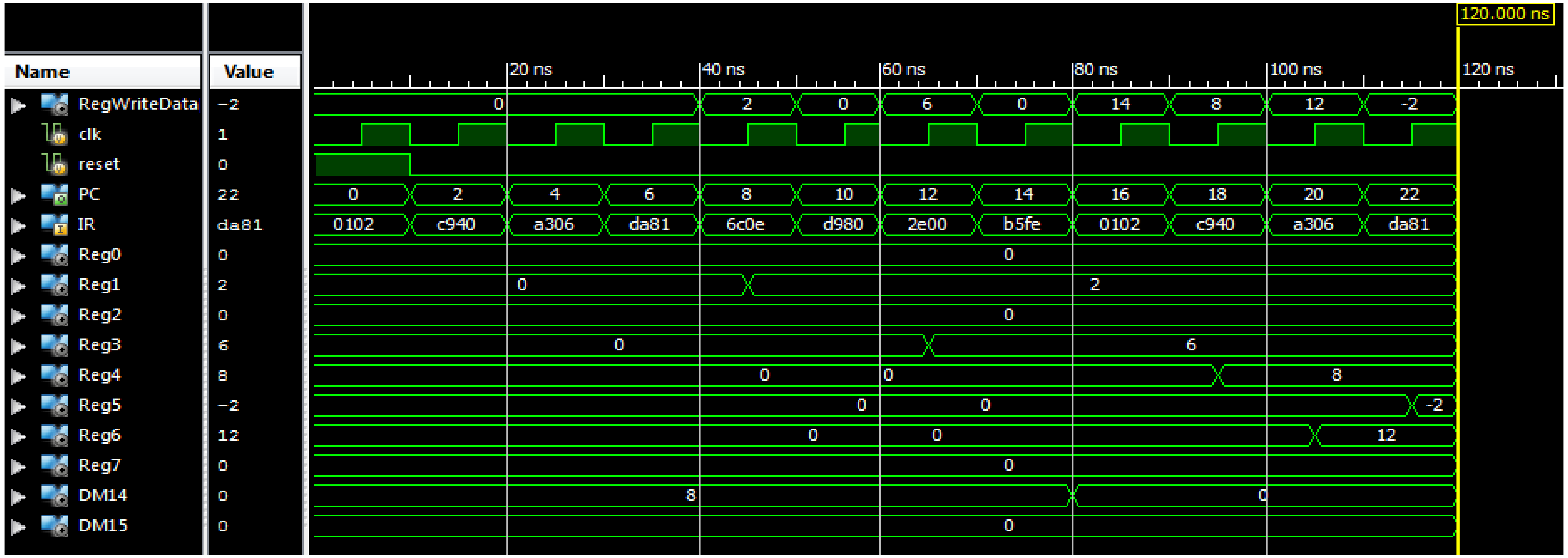
BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI K. K. BIRLA Goa Campus
First Semester 2014-2015, CS F342 Computer Architecture, Lab – 5, 16th October 2014

Design and implement Pipeline Implementation with Data Forwarding and Hazard Detection (as shown in Fig. 1):

Instruction Memory:

lw R1, R0(2)	(0102)
add R2, R1, R1	(c940)
addi R3, R4, 6	(a306)
sub R4, R3, R2	(da81)
sw R4, R5(4)	(6c0e)
add R4, R3, R1	(d980)
lw R6, R5(0)	(2e00)
addi R5, R6, -2	(b5fe)

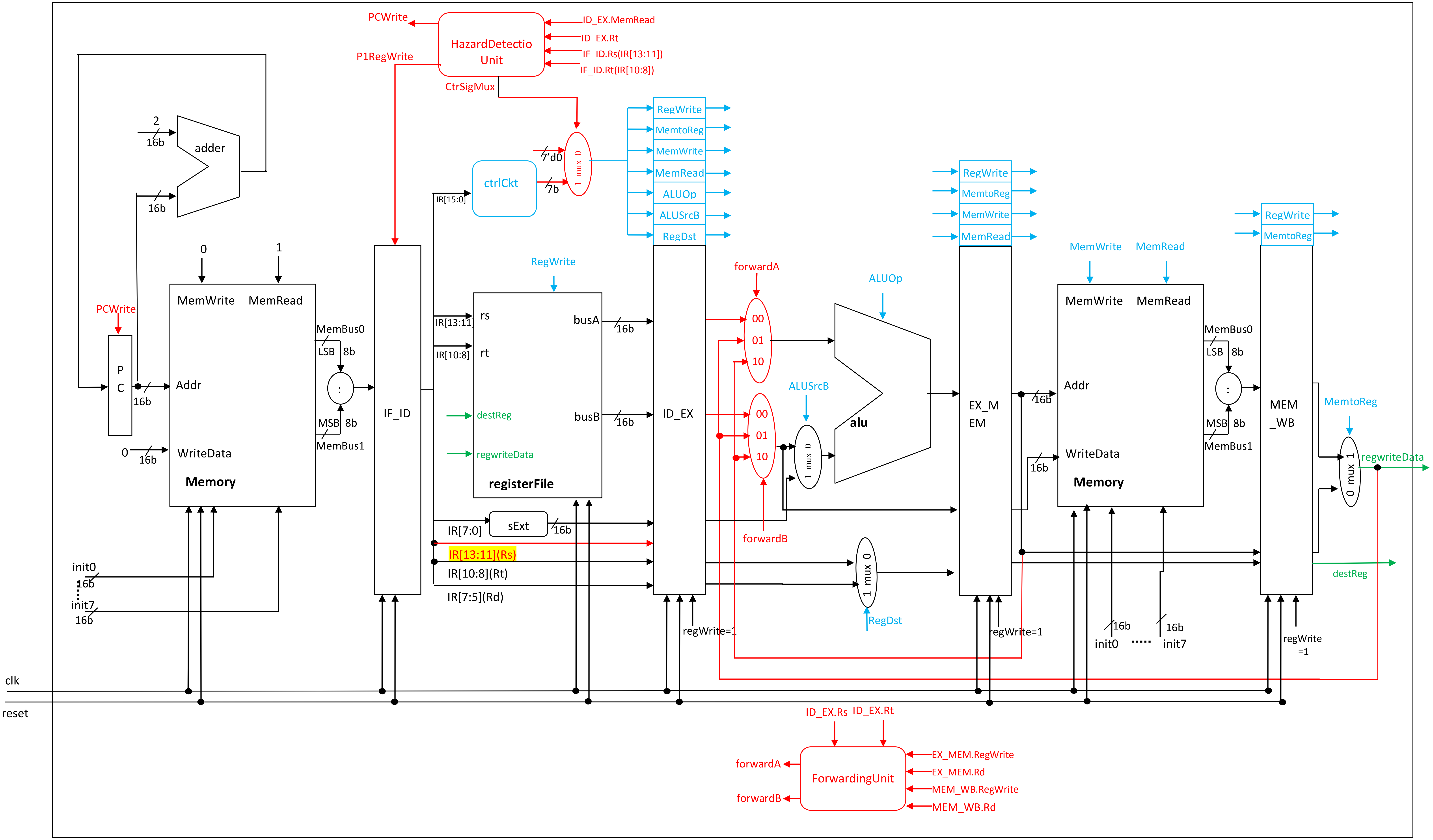
Simulation Results without Data Forwarding and Hazard Detection (Lab04Solution.v):



Additional/Modified Interfaces for Data Forwarding and Hazard Detection:

1. **module ForwardingUnit(**input EX_MEM_RegWrite, input MEM_WB_RegWrite, input [2:0] EX_MEM_Rd, input [2:0] MEM_WB_Rd, input [2:0] ID_EX_Rs, input [2:0] ID_EX_Rt, **output reg [1:0] forwardA, output reg [1:0] forwardB);**
2. **module HazardDetectionUnit(**input ID_EX_MemRead, input [2:0] ID_EX_Rt, input [2:0] IF_ID_Rs, input [2:0] IF_ID_Rt, **output reg PCWrite, output reg P1RegWrite, output reg CtrSigMux);**
3. **module Mux4to1_16bits(**input [15:0] in1, input [15:0] in2, input [15:0] in3,input [15:0] in4, input [1:0] Sel, **output reg [15:0] MuxOut);**
4. **module Mux2to1_7bits(**input [6:0] in1, input [6:0] in2, input Sel, **output reg [6:0] MuxOut);**
5. **module ID_EX_PipelineRegister(**input clk, input reset, input regWrite, input [15:0] BusA, input [15:0] BusB, input [15:0] SignExt8to16Imm, input [2:0] rs, input [2:0] rt, input [2:0] rd, input ALUSrcB, input ALUOp, input RegDst, input MemRead, input MemWrite, input MemToReg, input RegWrite, **output [15:0] P2BusA, output [15:0] P2BusB, output [15:0] P2SignExt8to16Imm, output [2:0] P2rs, output [2:0] P2rt, output [2:0] P2rd, output P2ALUSrcB, output P2ALUOp, output P2RegDst, output P2MemRead, output P2MemWrite, output P2MemToReg, output P2RegWrite);**

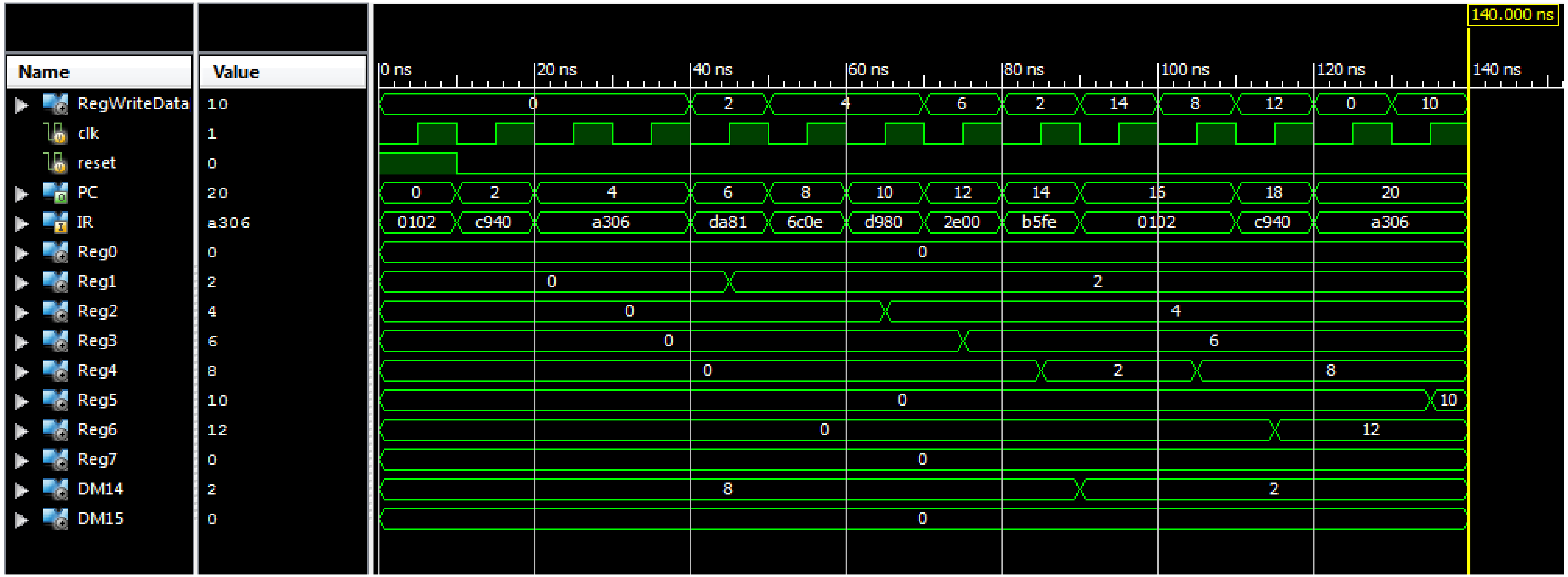
Fig. 1] Pipeline Implementation with Data forwarding and Hazard Detection



Test Bench:

```
module testBench;
        reg clk, reset;
        wire [15:0] RegWriteData;
        TopModule uut (.clk(clk), .reset(reset), .RegWriteData(RegWriteData));
        always
                #5 clk=~clk;
        initial begin
                clk = 0; reset = 1;
                #10 reset=0;
                #130 $finish;
        end
endmodule
```

Simulation Results with Data Forwarding and Hazard Detection (as per Fig. 1):



140.000 ns
Name
Value
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
RegWriteData 10 | 0 | 2 | 4 | 6 | 2 | 14 | 8 | 12 | 0 | 10
clk 1
reset 0
PC 20 | 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20
IR a306 | 0102 | c940 | a306 | da81 | 6c0e | d980 | 2e00 | b5fe | 0102 | c940 | a306
Reg0 0 | 0
Reg1 2 | 0 | 2
Reg2 4 | 0 | 4
Reg3 6 | 0 | 6
Reg4 8 | 0 | 2 | 8
Reg5 10 | 0 | 10
Reg6 12 | 0 | 12
Reg7 0 | 0
DM14 2 | 8 | 2
DM15 0 | 0