Toronto Metropolitan University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering
& Architectural Science

Course Number	COE 608
Course Title	Computer Organization and Architecture
Semester/Year	W2023
Instructor	Khalid Abdel Hafeez
TA Name	Yasaman Ahmadiadli

Lab 6 Report

Section No.	10
Submission Date	April 12th, 2024
Due Date	April 12th, 2024

Student Name	Student ID	Signature*
Caleb Joseph	xxxx99699	C.J

(Note: remove the first 4 digits from your student ID)

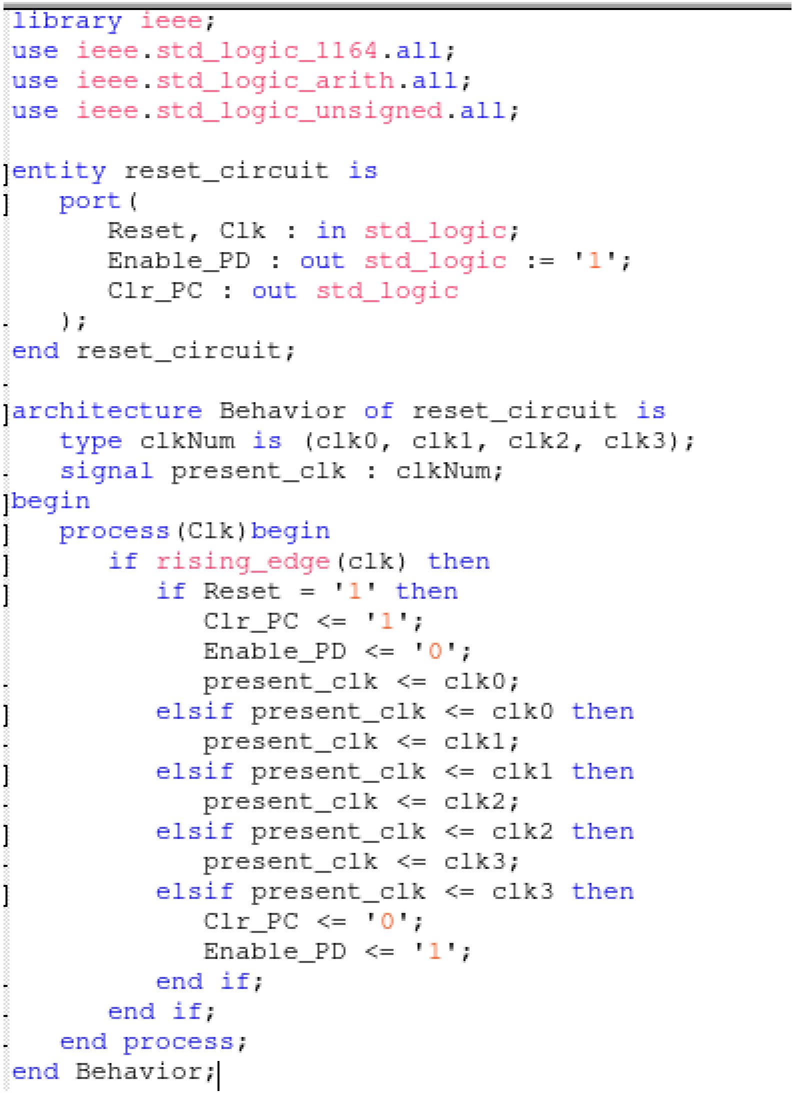
**By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at:http://www.ryerson.ca/content/dam/senate/policies/pol60.pd*

1. Lab Objective:

This lab focused on the practical implementation of the 32-bit CPU via the CPU control unit using VHDL. We aimed to gain hands-on experience creating a control unit that orchestrates the intricate dance of instructions, data, and signals. In addition, we aimed to carry out fetch, decode, execute. This report details the processes involved and highlights key insights into the previous labs data path and control signals for registers, ALU and MUXes.

2. Experiment Details:

VHDL Code:

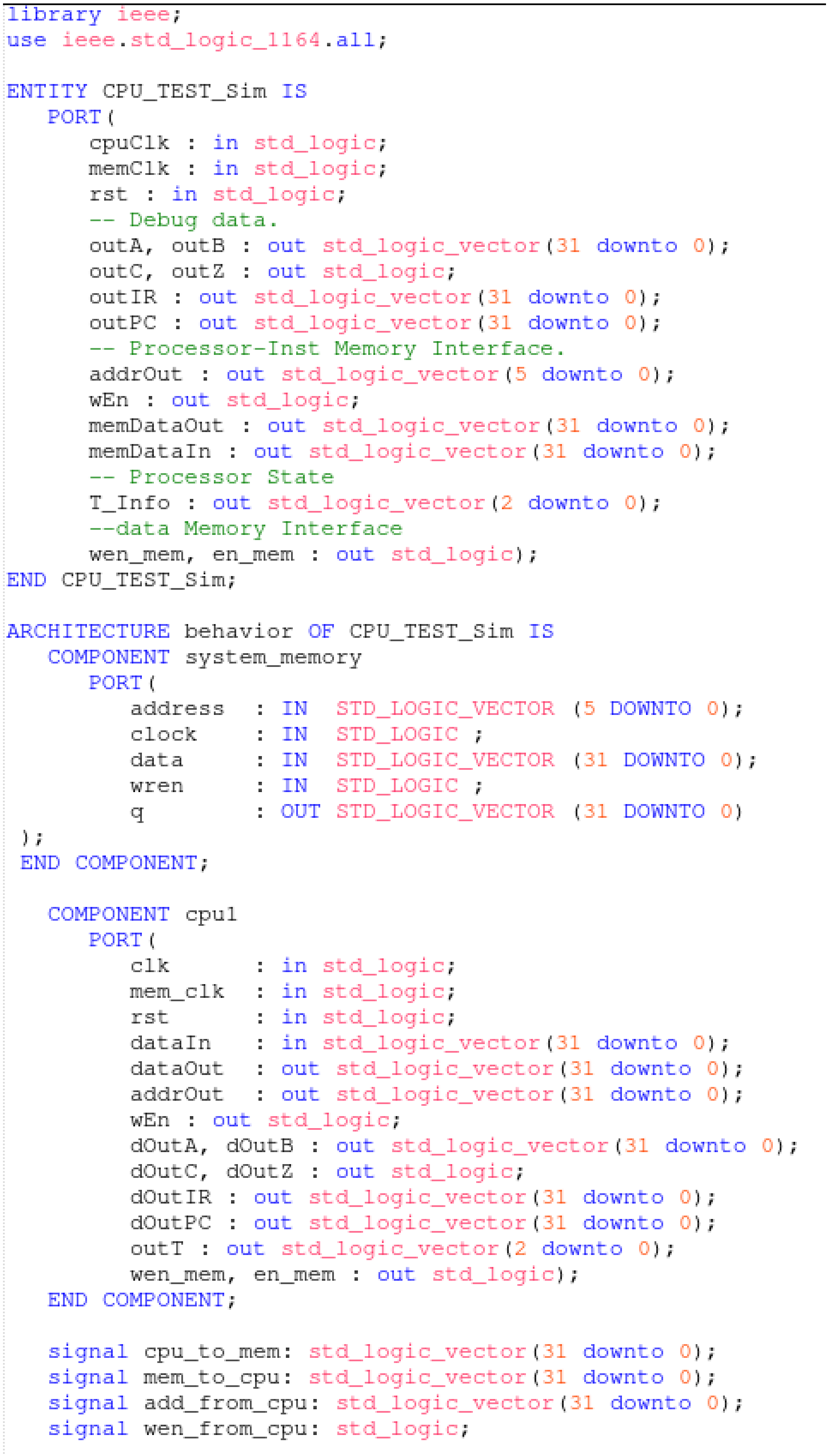


```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity reset_circuit is
    port(
        Reset, Clk : in std_logic;
        Enable_PD : out std_logic := '1';
        Clr_PC : out std_logic
    );
end reset_circuit;

architecture Behavior of reset_circuit is
    type clkNum is (clk0, clk1, clk2, clk3);
    signal present_clk : clkNum;
begin
    process(Clk)begin
        if rising_edge(clk) then
            if Reset = '1' then
                Clr_PC <= '1';
                Enable_PD <= '0';
                present_clk <= clk0;
            elsif present_clk <= clk0 then
                present_clk <= clk1;
            elsif present_clk <= clk1 then
                present_clk <= clk2;
            elsif present_clk <= clk2 then
                present_clk <= clk3;
            elsif present_clk <= clk3 then
                Clr_PC <= '0';
                Enable_PD <= '1';
            end if;
        end if;
    end process;
end Behavior;
```

Figure 1. VHDL code for reset_circuit.



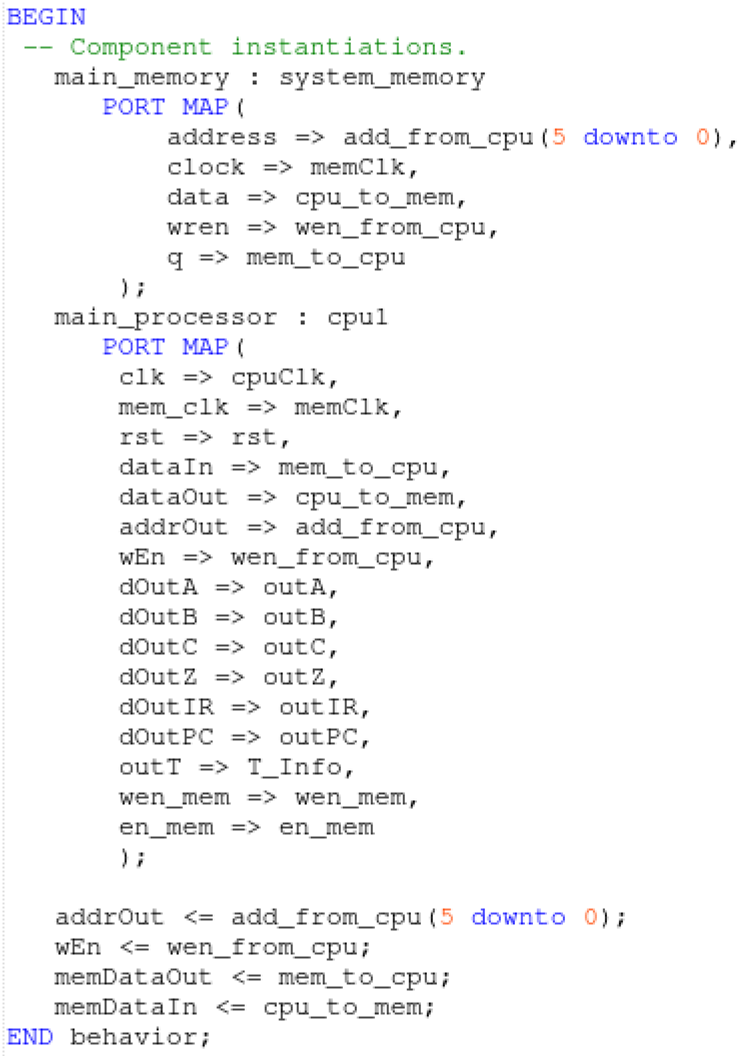
```vhdl
library ieee;
use ieee.std_logic_1164.all;

ENTITY CPU_TEST_Sim IS
   PORT(
      cpuClk : in std_logic;
      memClk : in std_logic;
      rst : in std_logic;
      -- Debug data.
      outA, outB : out std_logic_vector(31 downto 0);
      outC, outZ : out std_logic;
      outIR : out std_logic_vector(31 downto 0);
      outPC : out std_logic_vector(31 downto 0);
      -- Processor-Inst Memory Interface.
      addrOut  : out std_logic_vector(5 downto 0);
      wEn : out std_logic;
      memDataOut  : out std_logic_vector(31 downto 0);
      memDataIn : out std_logic_vector(31 downto 0);
      -- Processor State
      T_Info : out std_logic_vector(2 downto 0);
      --data Memory Interface
      wen_mem, en_mem : out std_logic);
END CPU_TEST_Sim;

ARCHITECTURE behavior OF CPU_TEST_Sim IS
   COMPONENT system_memory
      PORT(
         address   : IN  STD_LOGIC_VECTOR (5 DOWNTO 0);
         clock     : IN  STD_LOGIC ;
         data      : IN  STD_LOGIC_VECTOR (31 DOWNTO 0);
         wren      : IN  STD_LOGIC ;
         q         : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
 );
 END COMPONENT;

   COMPONENT cpu1
      PORT(
         clk       : in std_logic;
         mem_clk   : in std_logic;
         rst       : in std_logic;
         dataIn    : in std_logic_vector(31 downto 0);
         dataOut   : out std_logic_vector(31 downto 0);
         addrOut   : out std_logic_vector(31 downto 0);
         wEn : out std_logic;
         dOutA, dOutB : out std_logic_vector(31 downto 0);
         dOutC, dOutZ : out std_logic;
         dOutIR : out std_logic_vector(31 downto 0);
         dOutPC : out std_logic_vector(31 downto 0);
         outT : out std_logic_vector(2 downto 0);
         wen_mem, en_mem : out std_logic);
   END COMPONENT;

   signal cpu_to_mem: std_logic_vector(31 downto 0);
   signal mem_to_cpu: std_logic_vector(31 downto 0);
   signal add_from_cpu: std_logic_vector(31 downto 0);
   signal wen_from_cpu: std_logic;
```



```
BEGIN
 -- Component instantiations.
   main_memory : system_memory
      PORT MAP(
          address => add_from_cpu(5 downto 0),
          clock => memClk,
          data => cpu_to_mem,
          wren => wen_from_cpu,
          q => mem_to_cpu
       );
   main_processor : cpu1
      PORT MAP(
       clk => cpuClk,
       mem_clk => memClk,
       rst => rst,
       dataIn => mem_to_cpu,
       dataOut => cpu_to_mem,
       addrOut => add_from_cpu,
       wEn => wen_from_cpu,
       dOutA => outA,
       dOutB => outB,
       dOutC => outC,
       dOutZ => outZ,
       dOutIR => outIR,
       dOutPC => outPC,
       outT => T_Info,
       wen_mem => wen_mem,
       en_mem => en_mem
       );

   addrOut <= add_from_cpu(5 downto 0);
   wEn <= wen_from_cpu;
   memDataOut <= mem_to_cpu;
   memDataIn <= cpu_to_mem;
END behavior;
```

Figure 2. cpu_test_sim VHDL code implementation.

VHDL Waveforms:

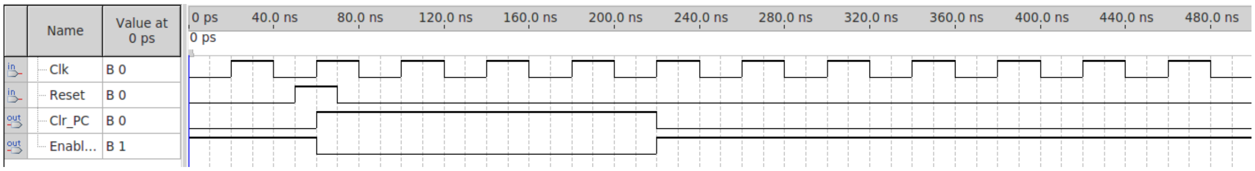


Figure 3. Functional waveform of reset_circuit.

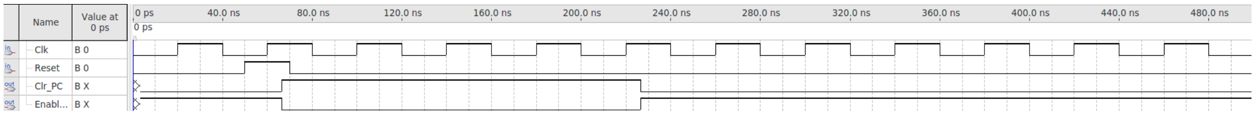
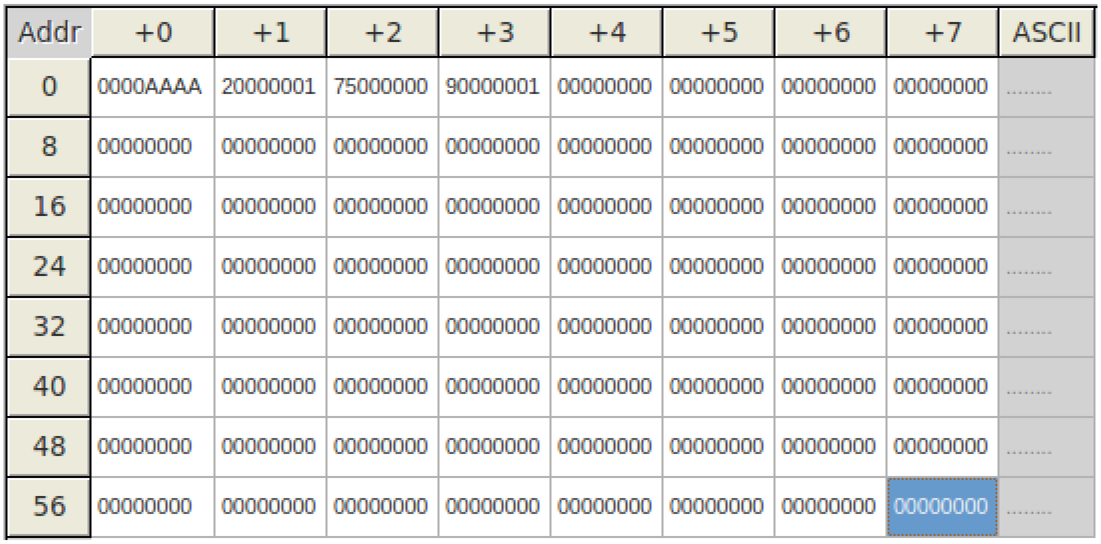


Figure 4. Timing-waveform of reset_circuit.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0000AAAA	20000001	75000000	90000001	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 5. System memory data for LDAI, STA, CLRA, LDA.

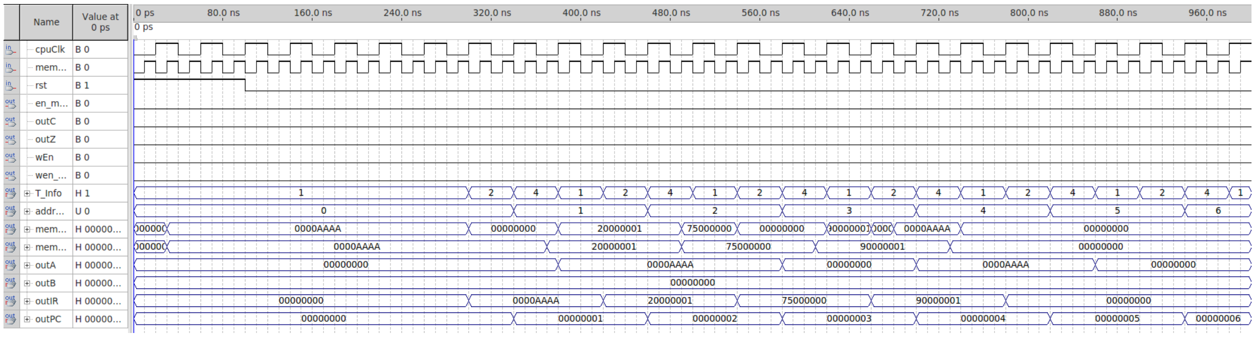


Figure 6. Functional waveform of LDAI, STA, CLRA, LDA.

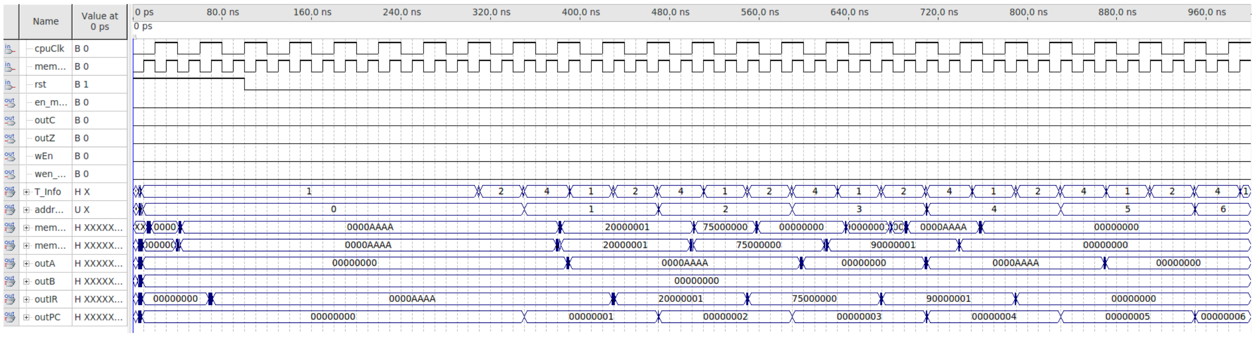
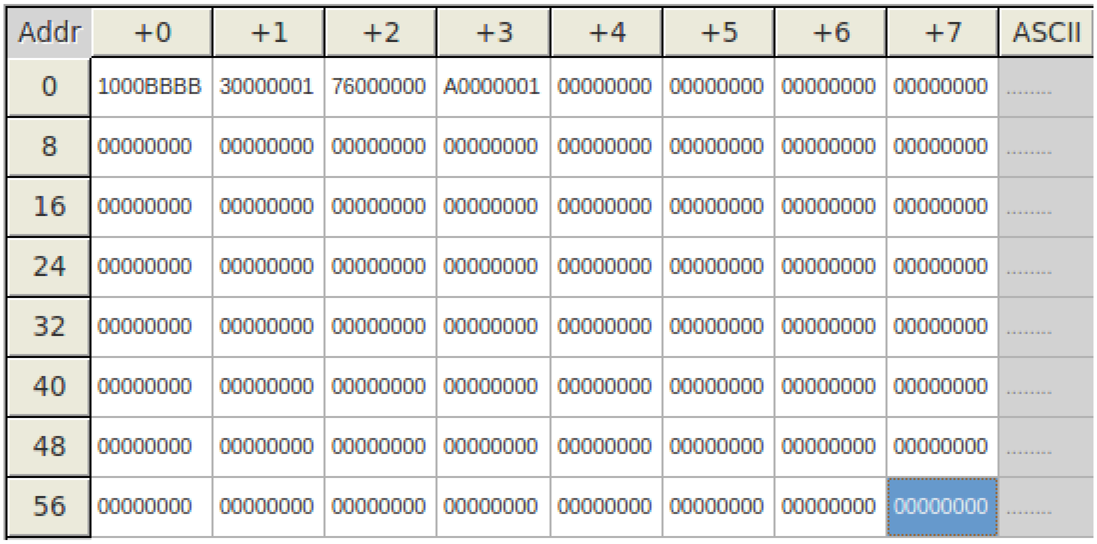


Figure 7. Timing-waveform of LDAI, STA, CLRA, LDA.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	1000BBBB	30000001	76000000	A0000001	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 8. System memory data for LDBI, STB, CLRB, LDB.

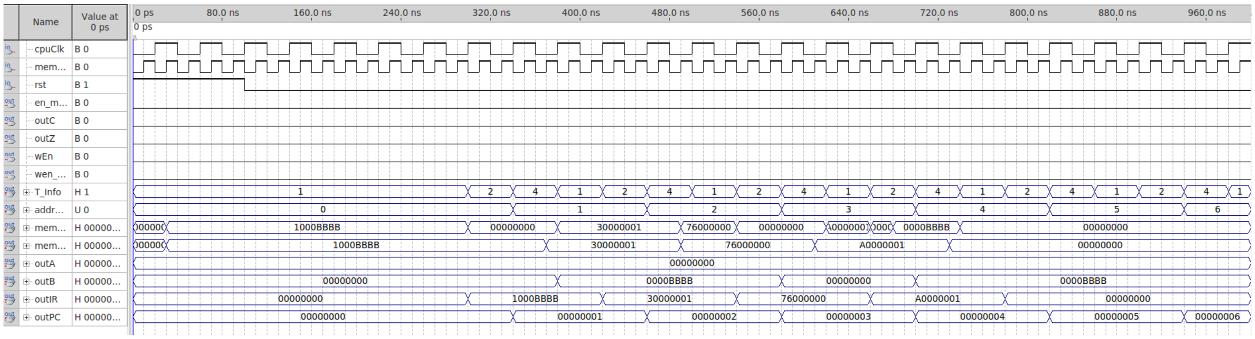


Figure 9. Functional waveform of LDBI, STB, CLRB, LDB.

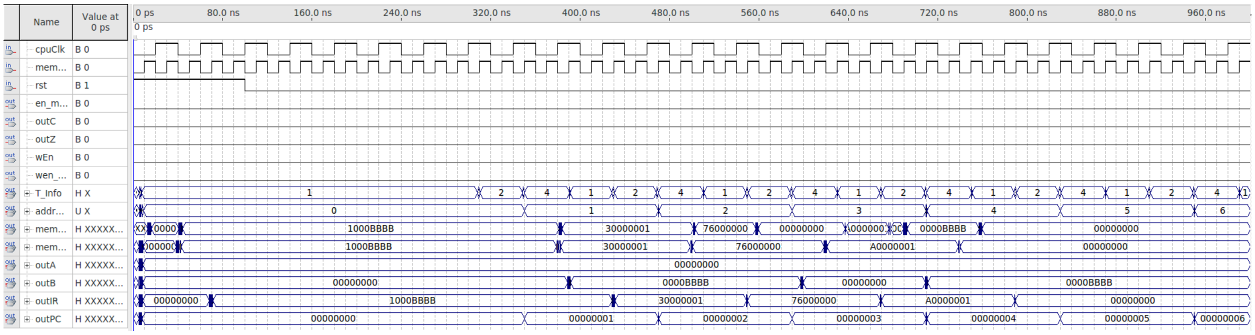
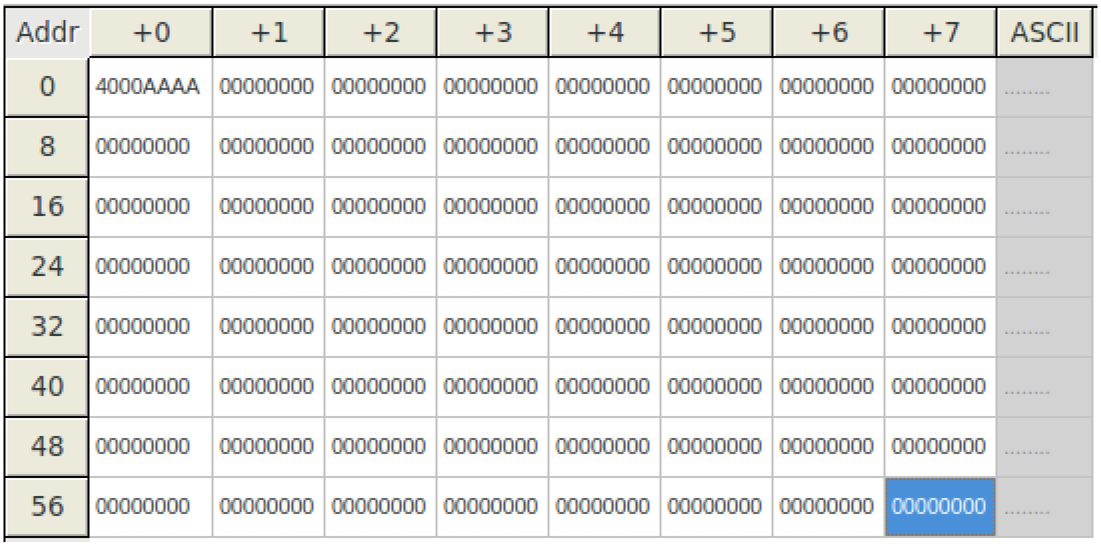


Figure 10. Timing-waveform of LDBI, STB, CLRB, LDB.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	4000AAAA	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 11. System memory data for LUI.

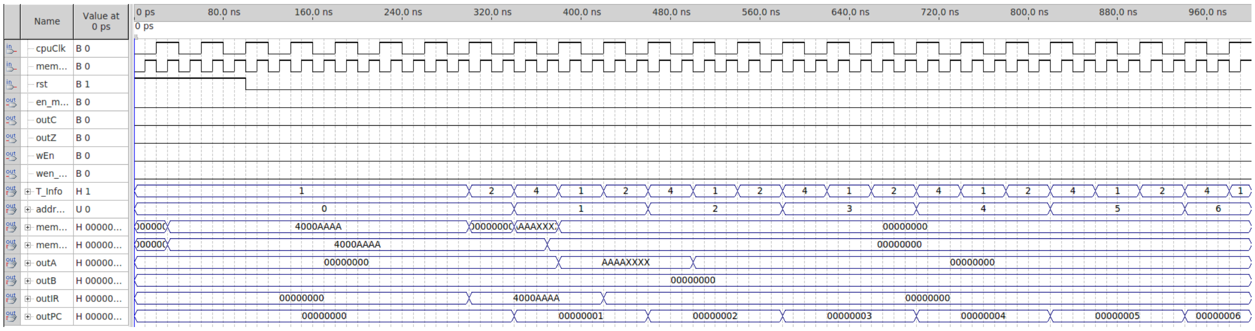


Figure 12. Functional waveform of LUI.

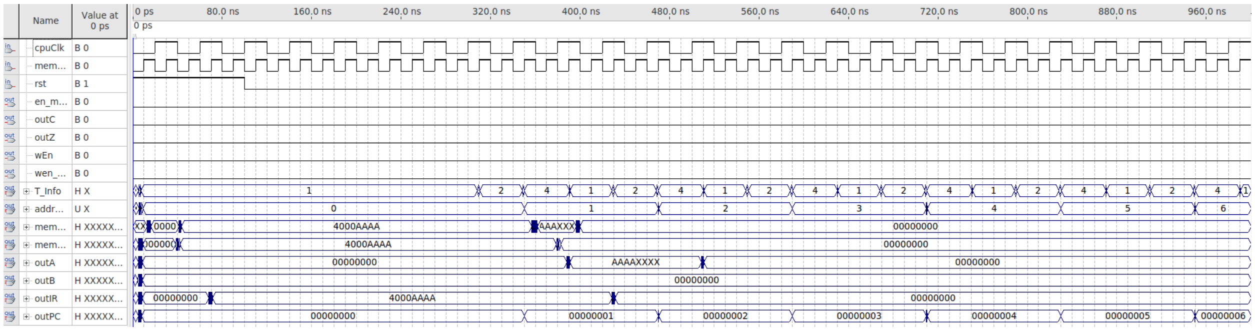
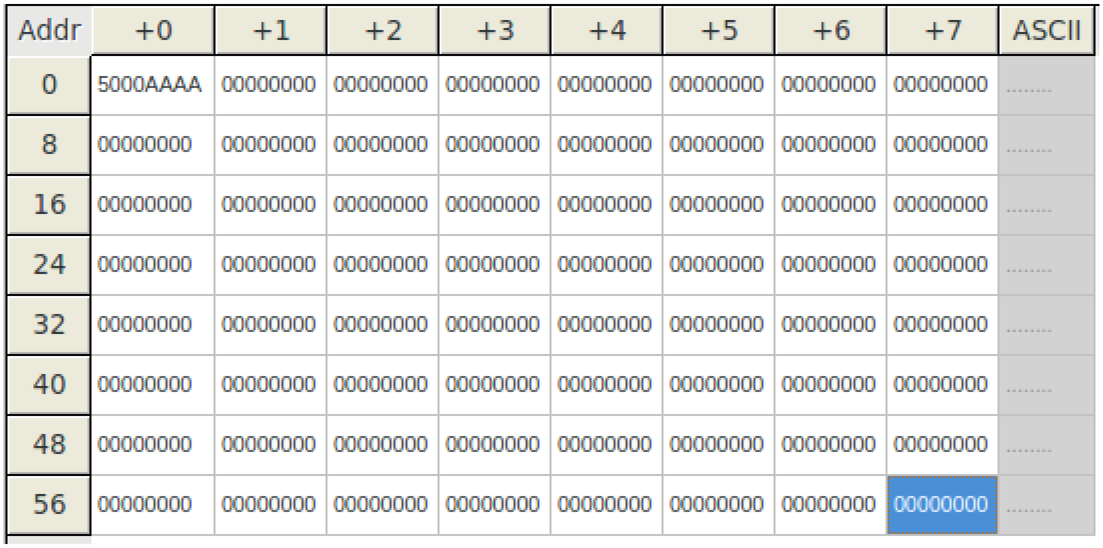


Figure 13. Timing-waveform of LUI.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	5000AAAA	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 14. System memory data for JMP.

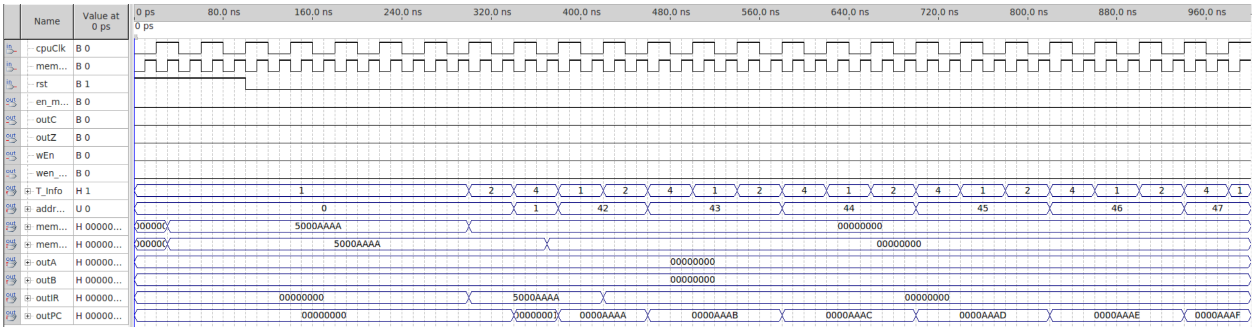


Figure 15. Functional waveform of JMP.

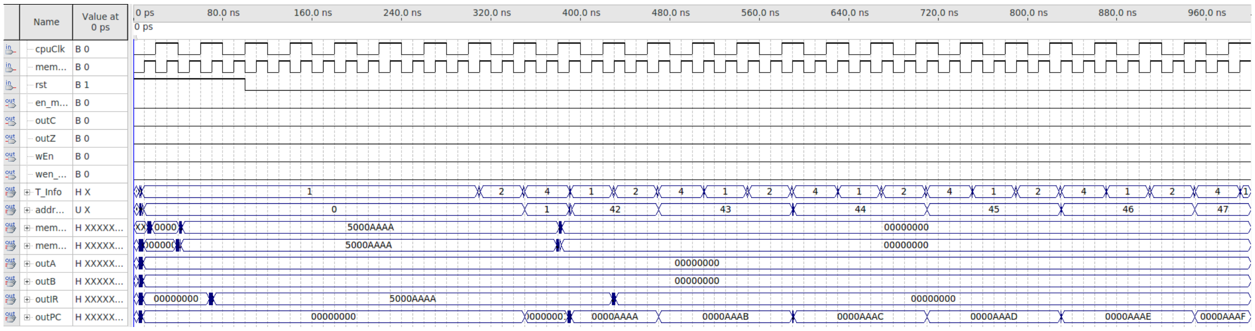
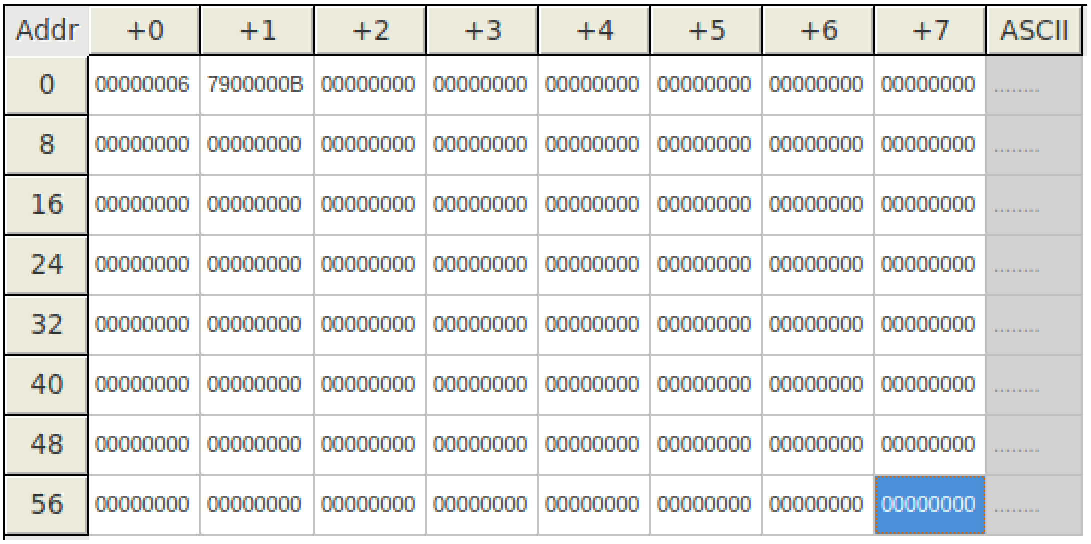


Figure 16. Timing-waveform of JMP.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000006	7900000B	00000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 17. System memory data for ANDI.

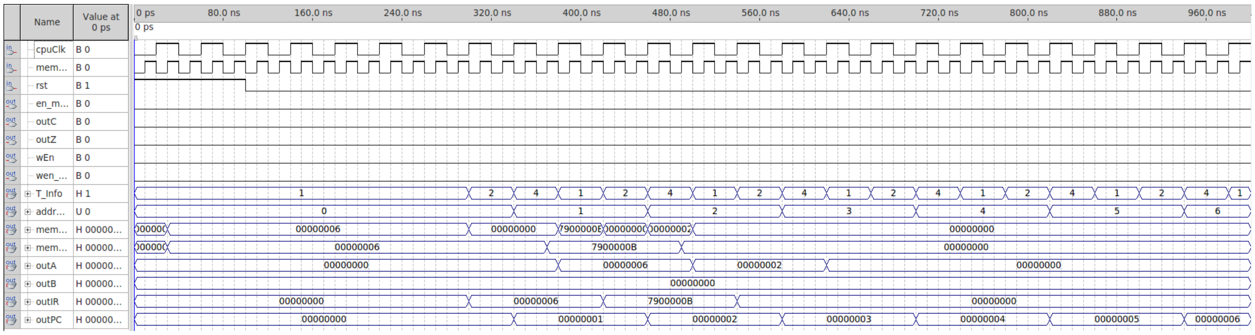


Figure 18. Functional waveform of ANDI.

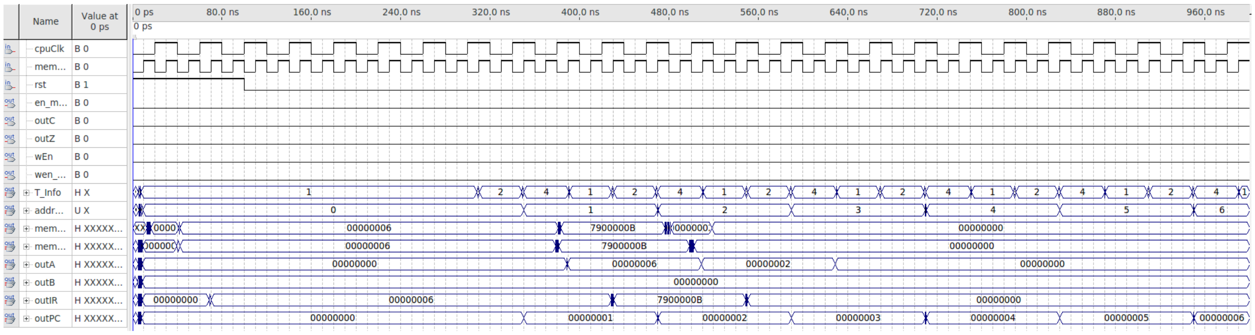
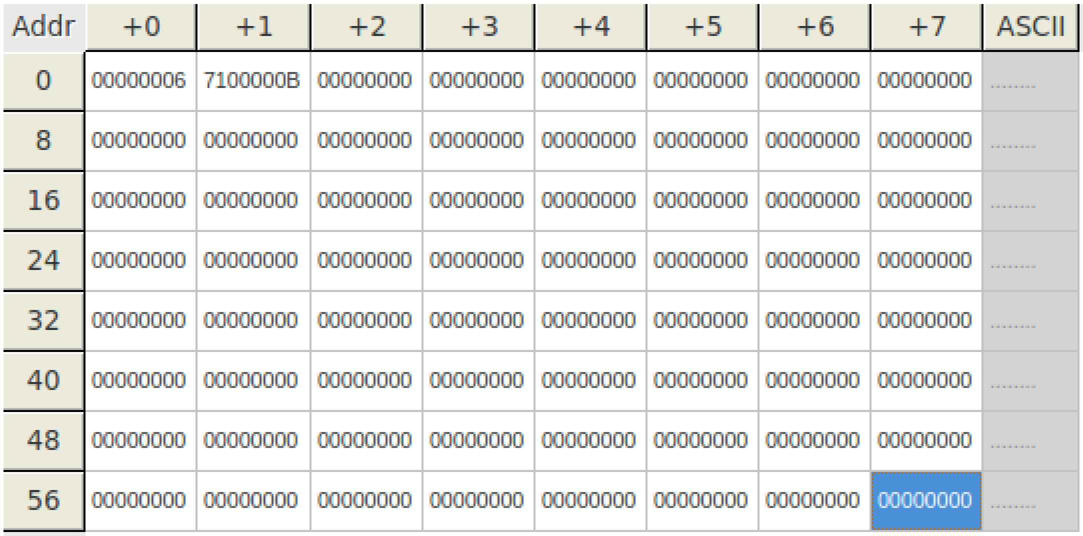


Figure 19. Timing-waveform of ANDI.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000006	7100000B	00000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 20. System memory data for ADDI.

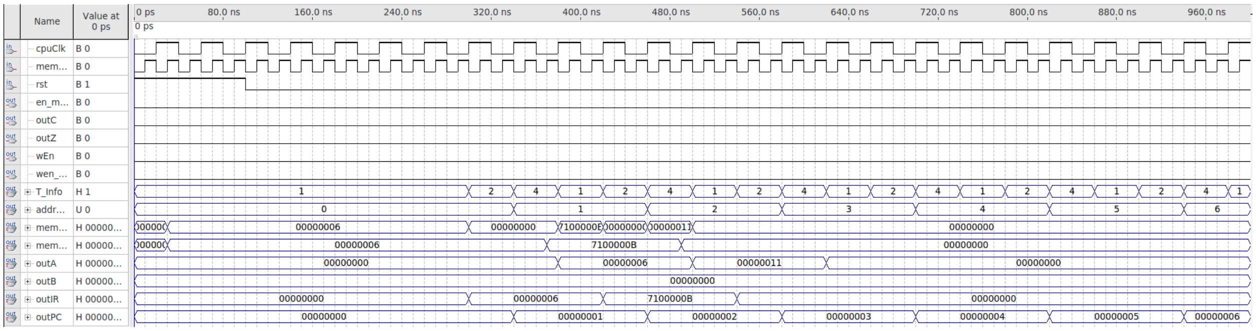


Figure 21. Functional waveform of ADDI.

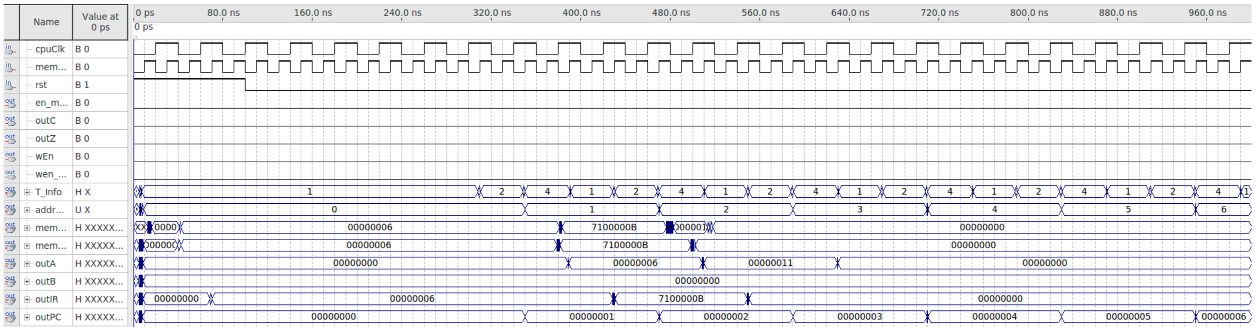
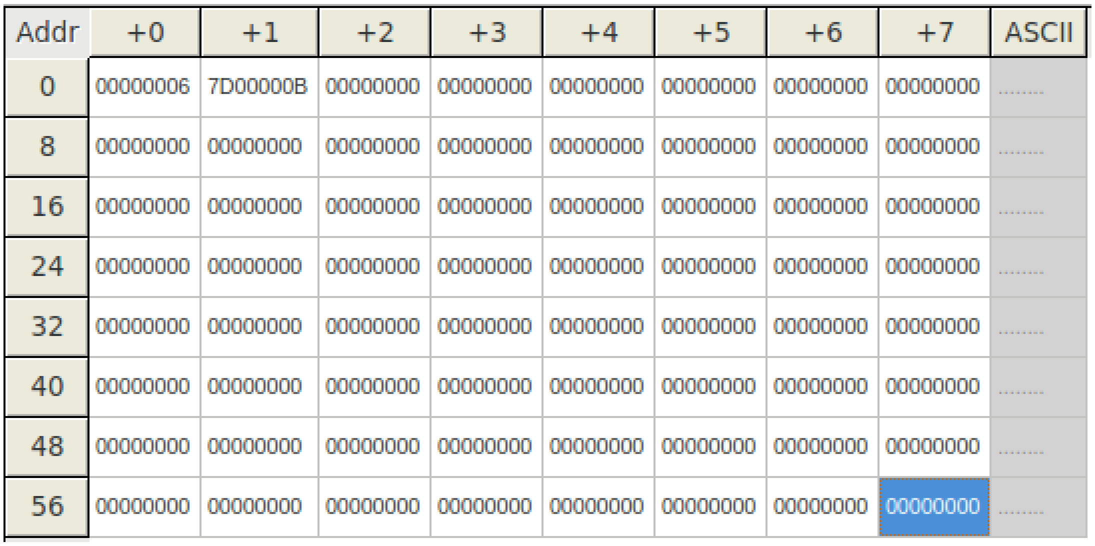


Figure 22. Timing-waveform of ADDI.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000006	7D00000B	00000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 23. System memory data for ORI.

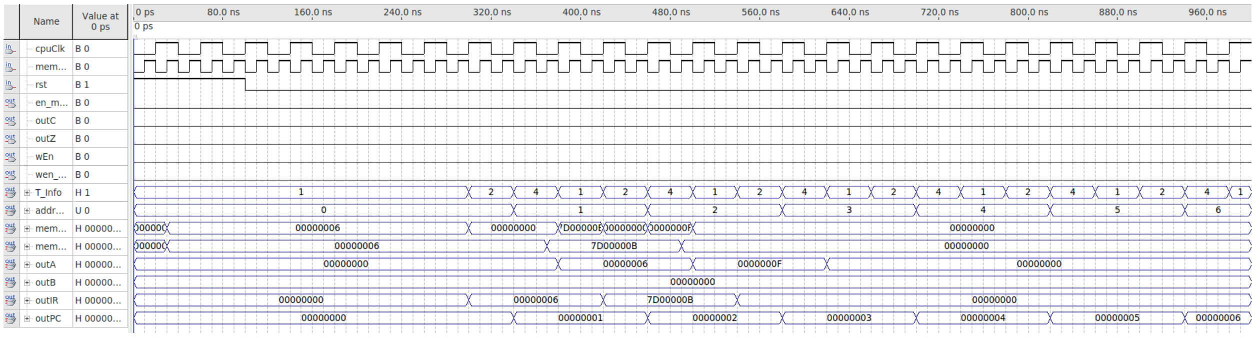


Figure 24. Functional waveform of ORI.

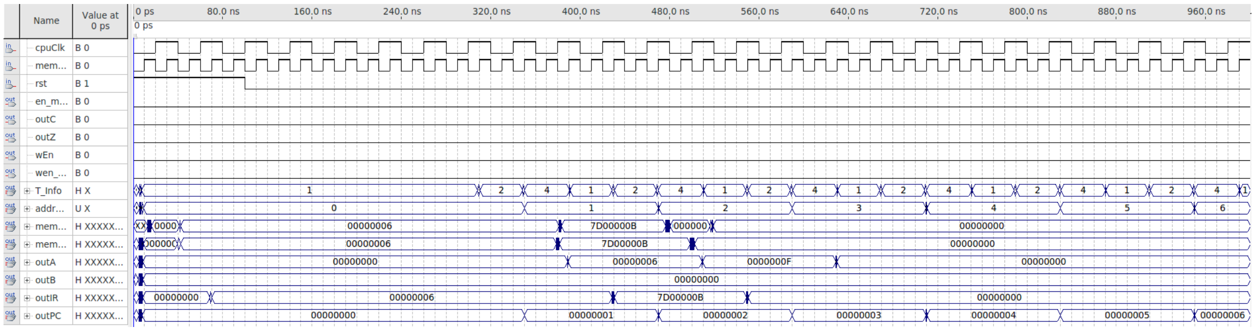
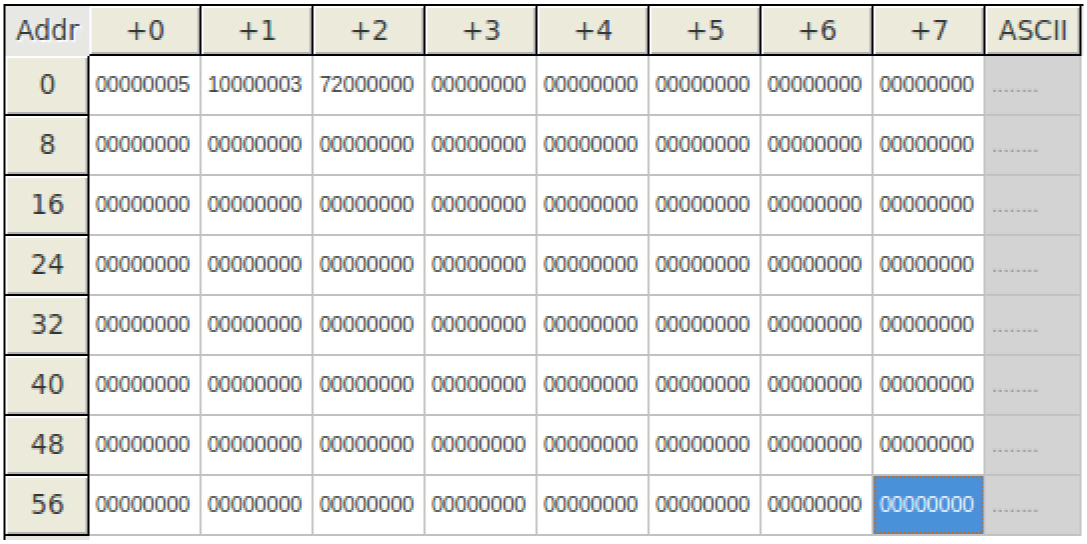


Figure 25. Timing-waveform of ORI.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000005	10000003	72000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 26. System memory data for SUB.

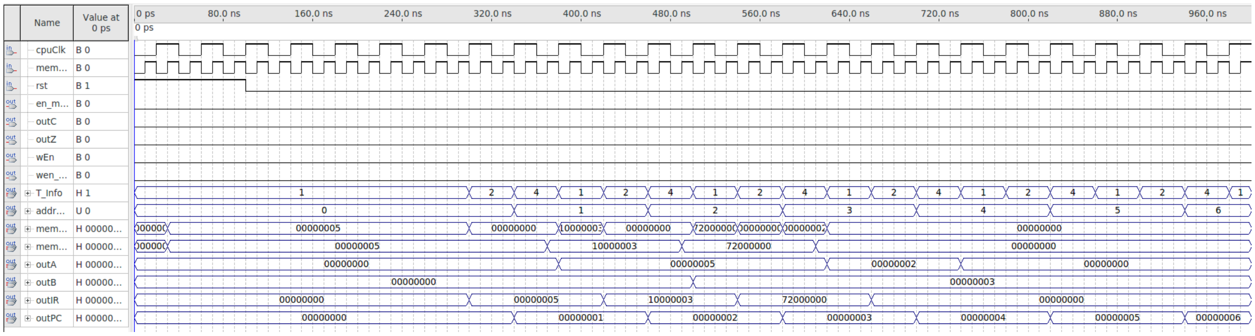


Figure 27. Functional waveform of SUB.

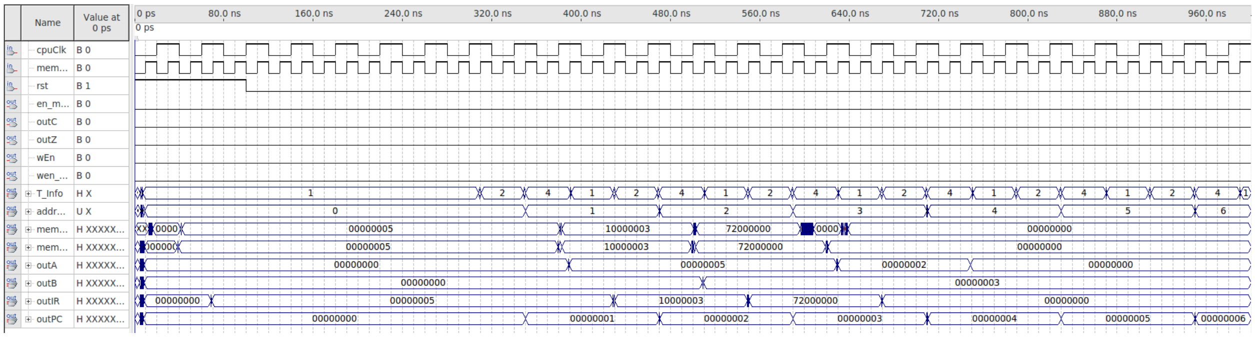
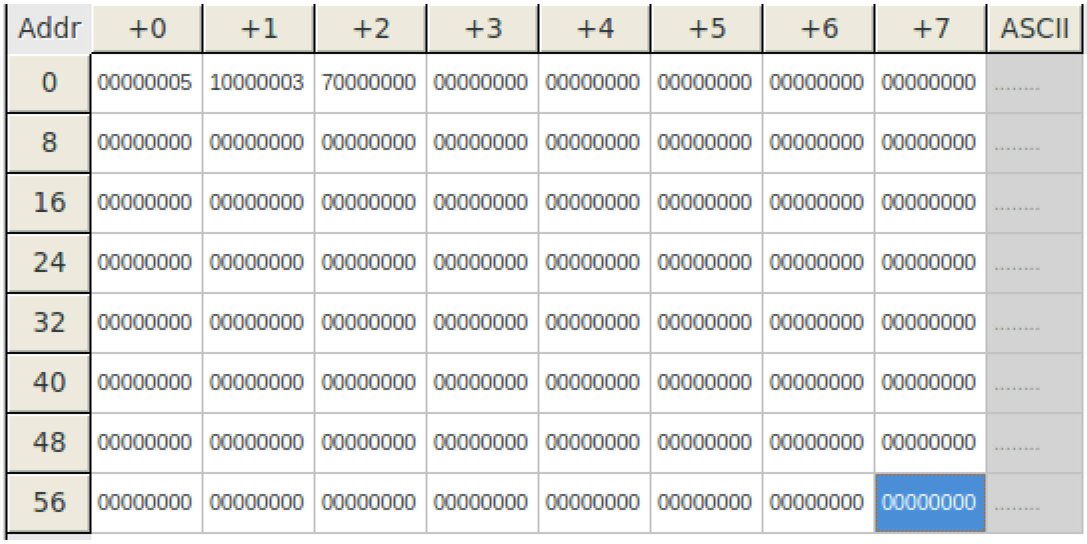


Figure 28. Timing-waveform of SUB.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000005	10000003	70000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 29. System memory data for ADD.

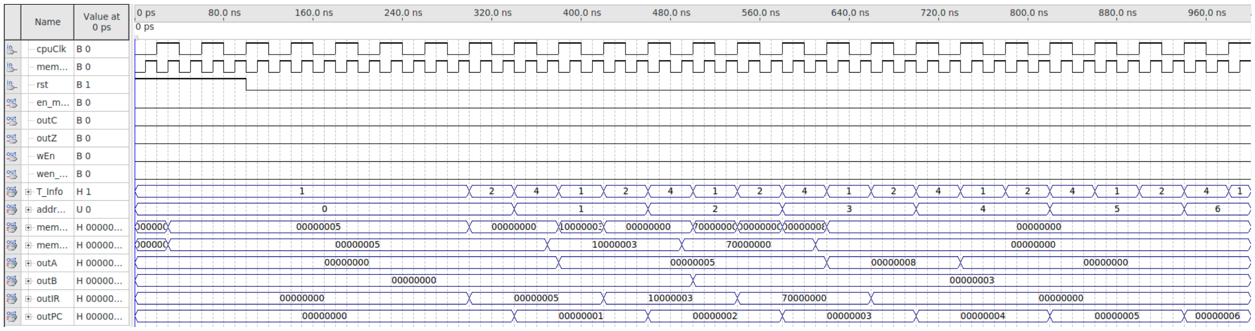


Figure 30. Functional waveform of ADD.

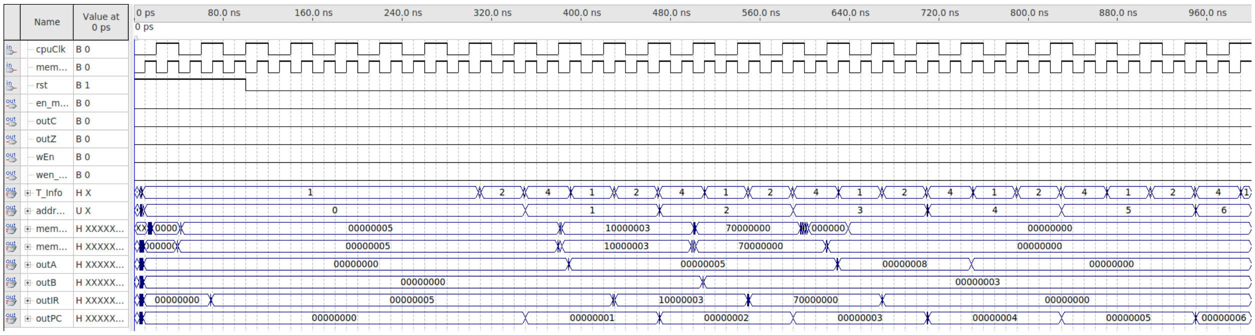
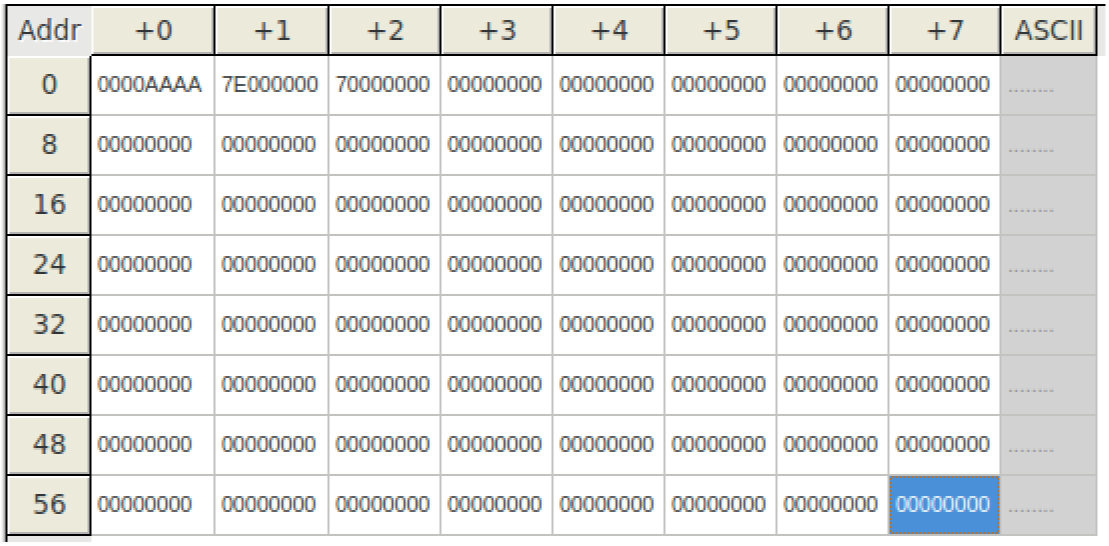


Figure 31. Timing-waveform of ADD.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0000AAAA	7E000000	70000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 32. System memory data for DECA.

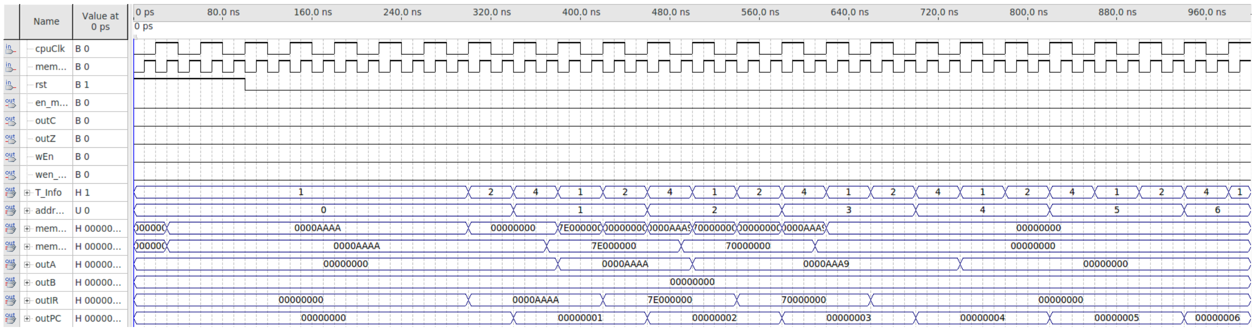


Figure 33. Functional waveform of DECA.

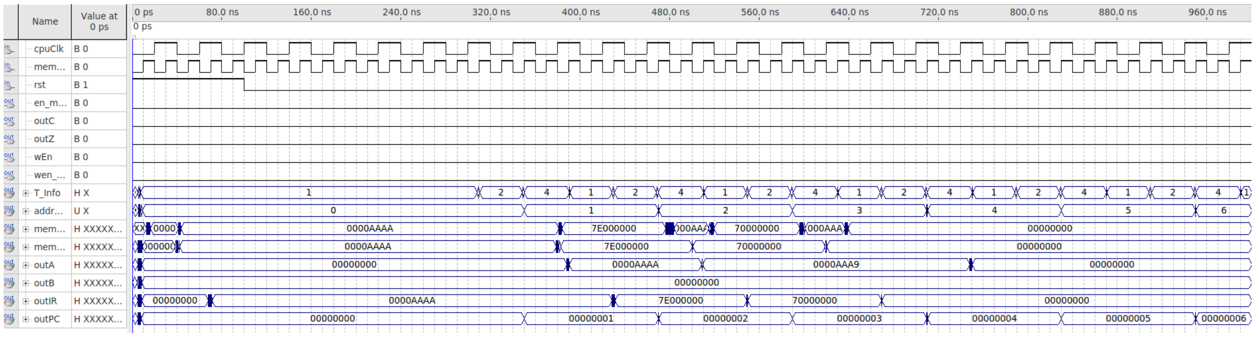
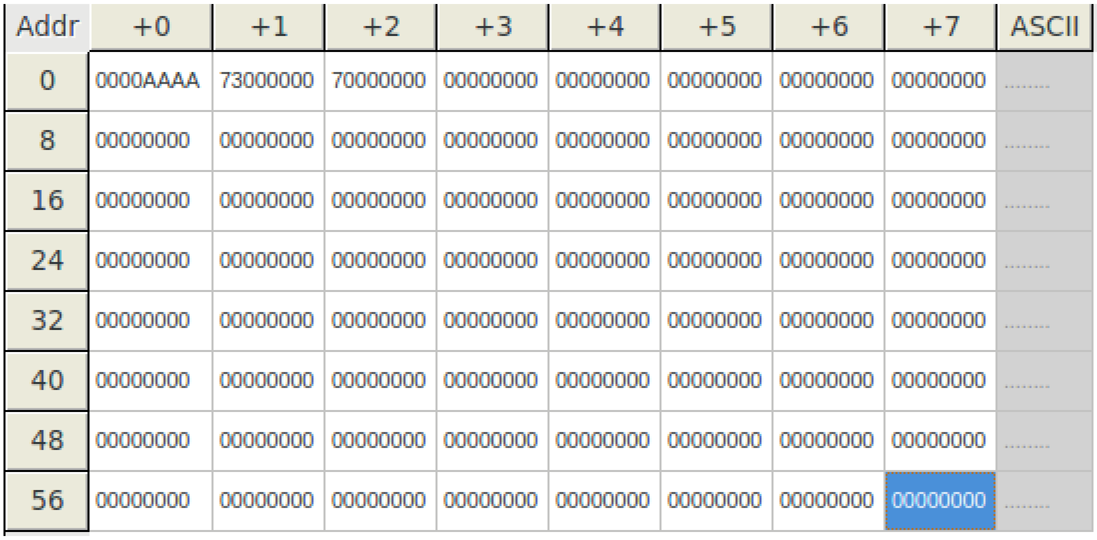


Figure 34. Timing-waveform of DECA.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0000AAAA	73000000	70000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 35. System memory data for INCA.

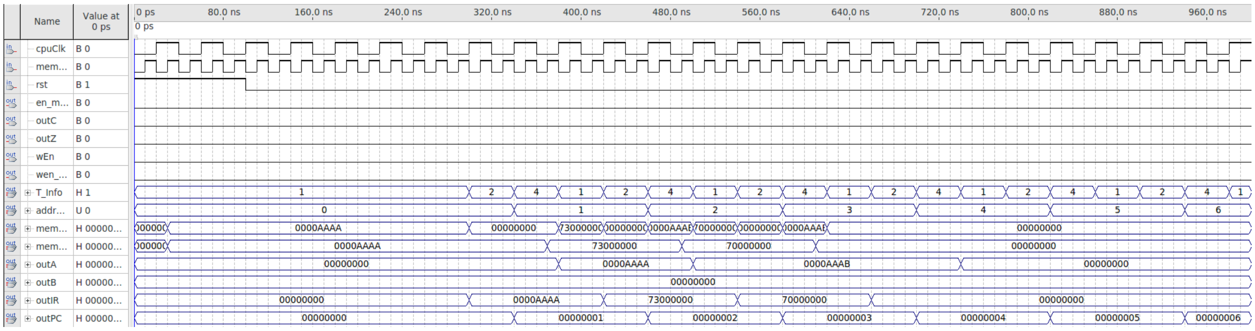


Figure 36. Functional waveform of INCA.

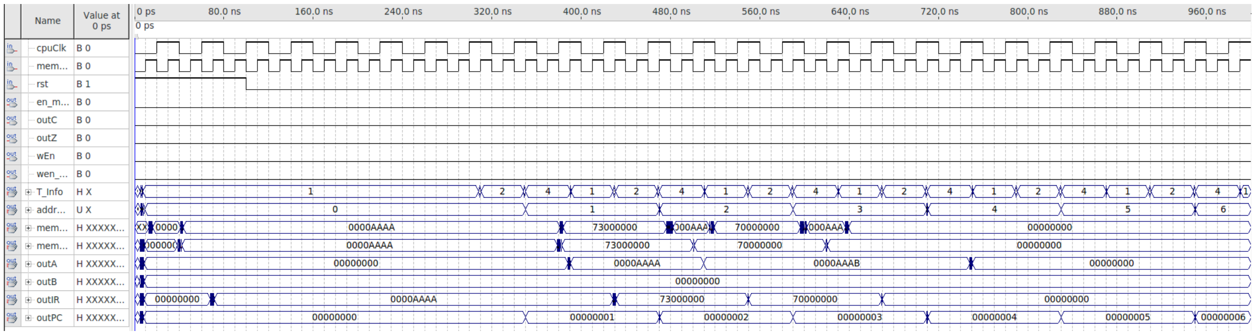
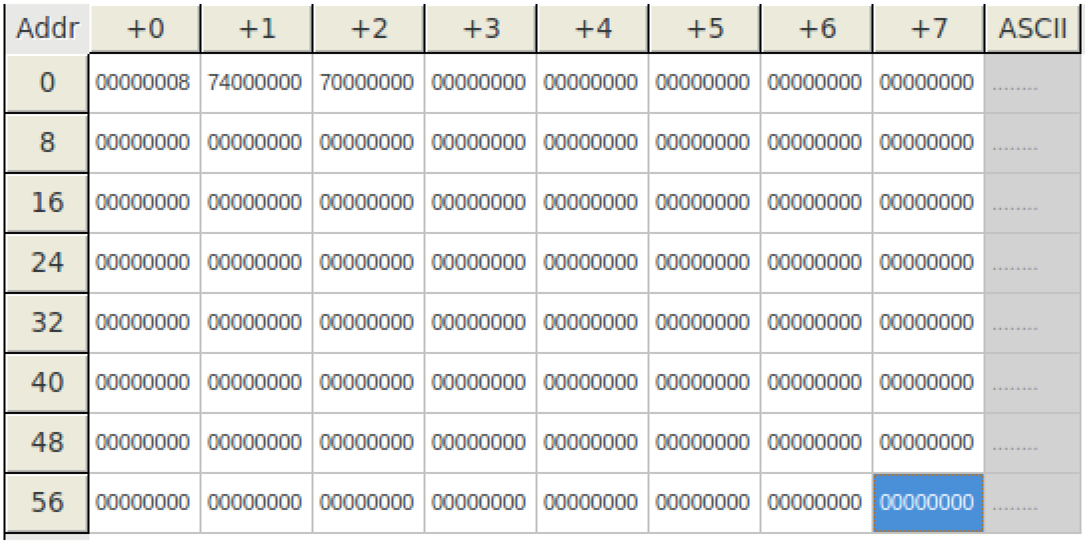


Figure 37. Timing-waveform of INCA.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000008	74000000	70000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 38. System memory data for ROL.

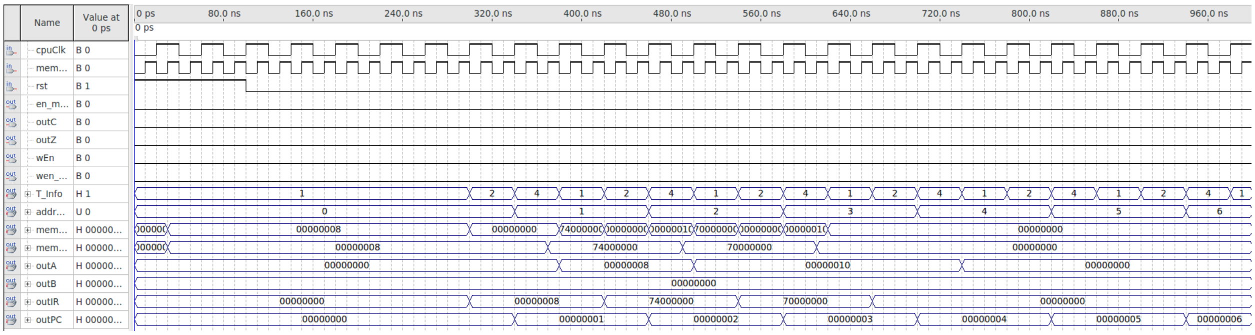


Figure 39. Functional waveform of ROL.

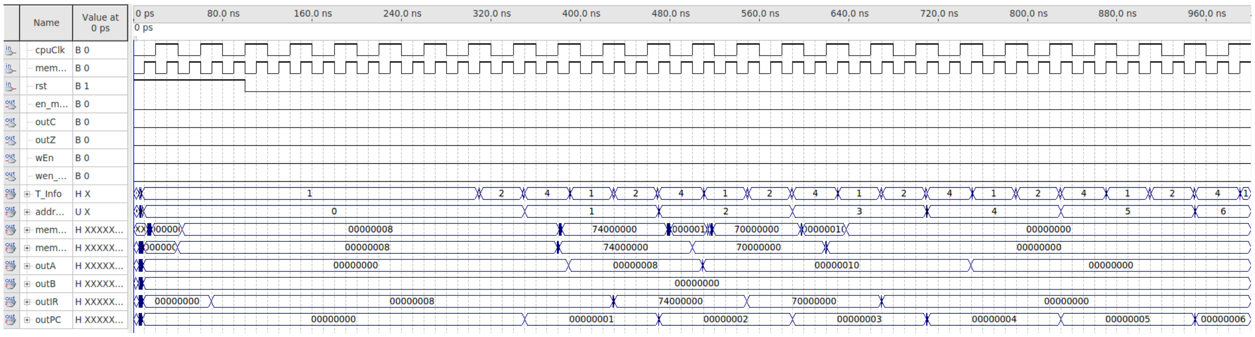
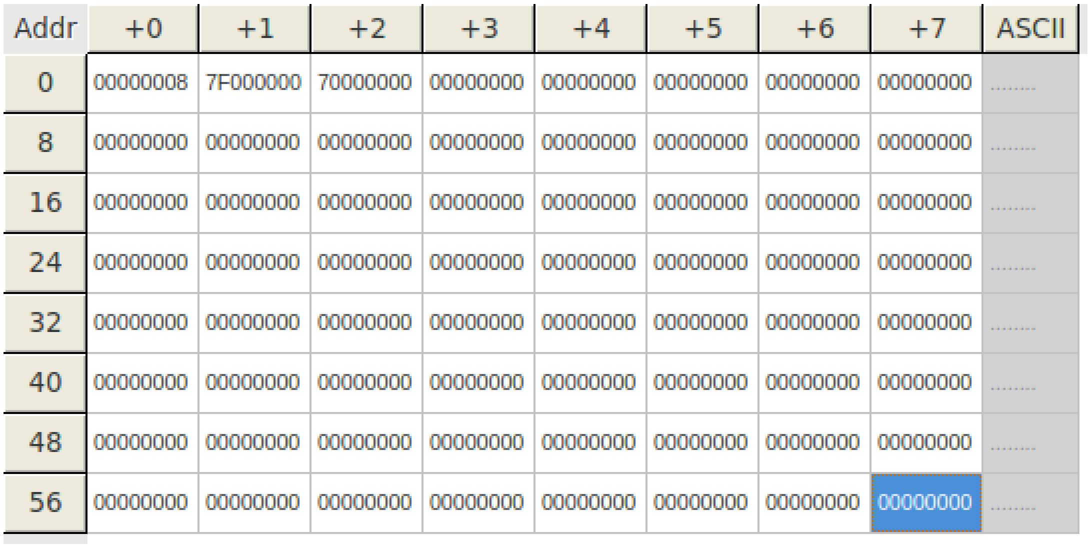


Figure 40. Timing-waveform of ROL.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000008	7F000000	70000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 41. System memory data for ROR.

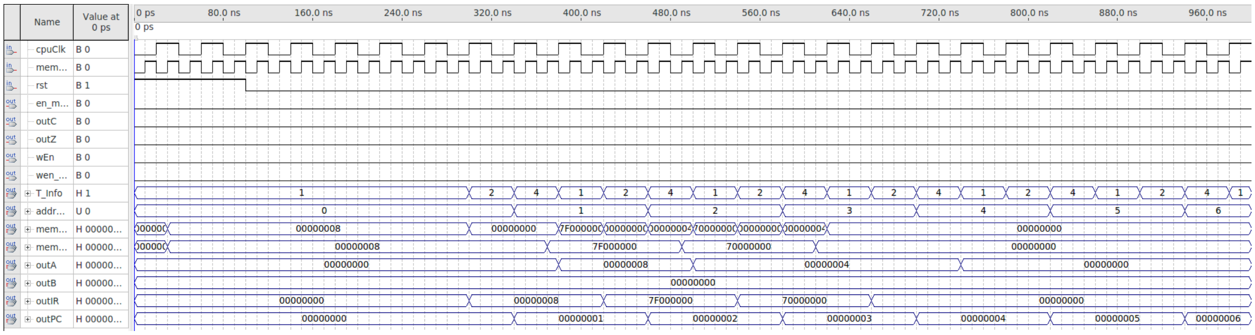


Figure 42. Functional waveform of ROR.

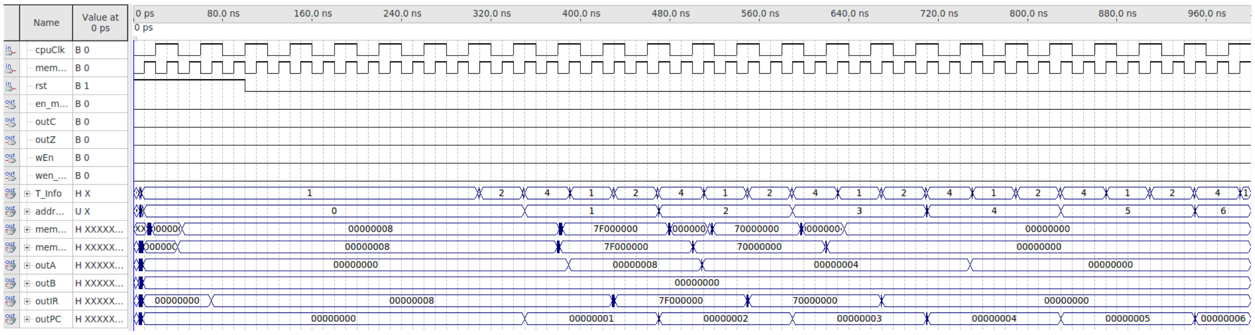
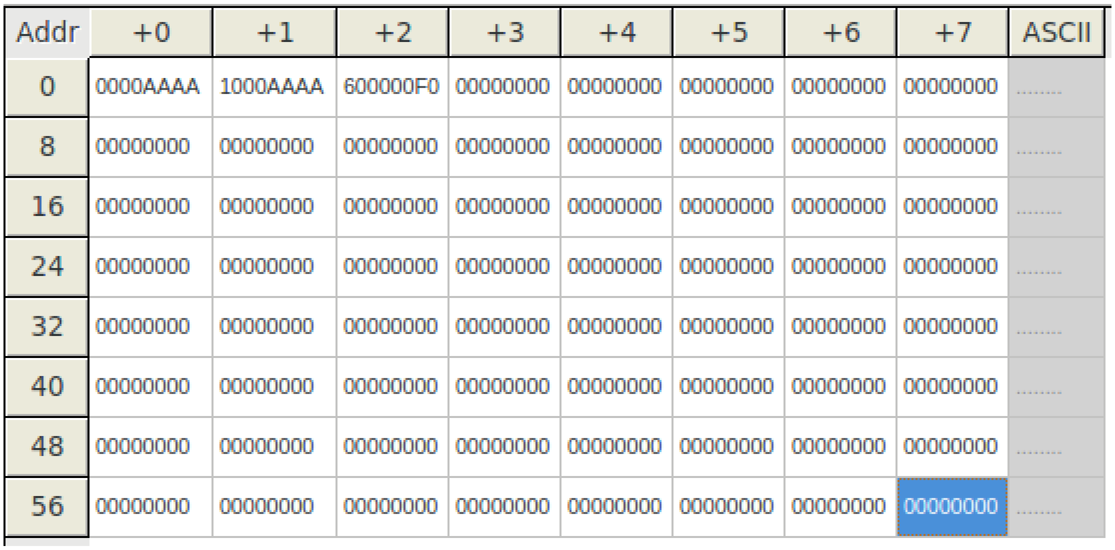


Figure 43. Timing-waveform of ROR.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0000AAAA	1000AAAA	600000F0	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 44. System memory data for BEQ.

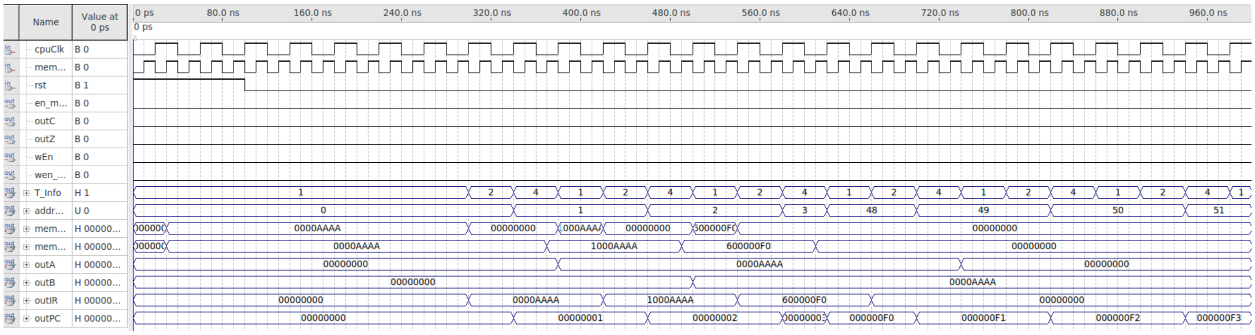


Figure 45. Functional waveform of BEQ.

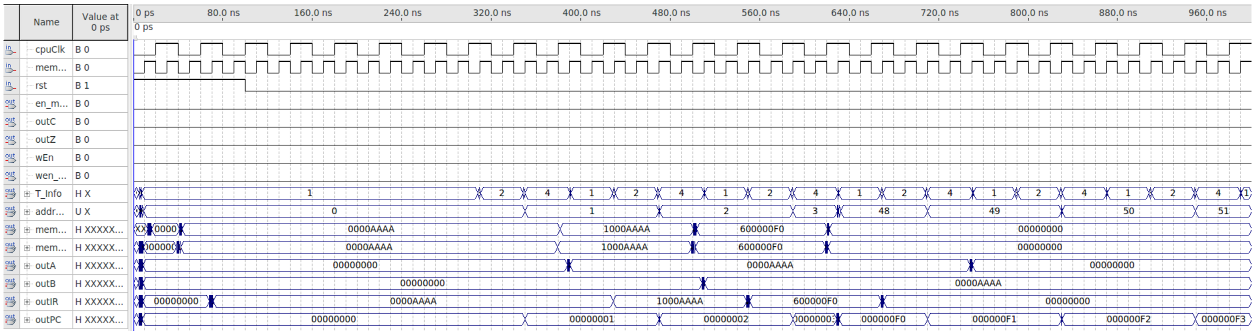
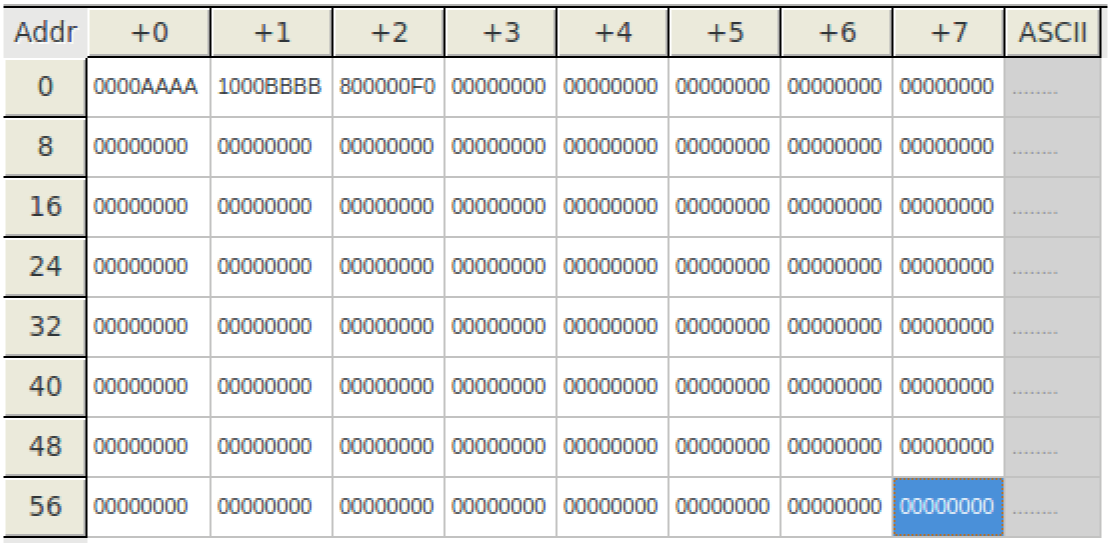


Figure 46. Timing-waveform of BEQ.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0000AAAA	1000BBBB	800000F0	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 47. System memory data for BNE.

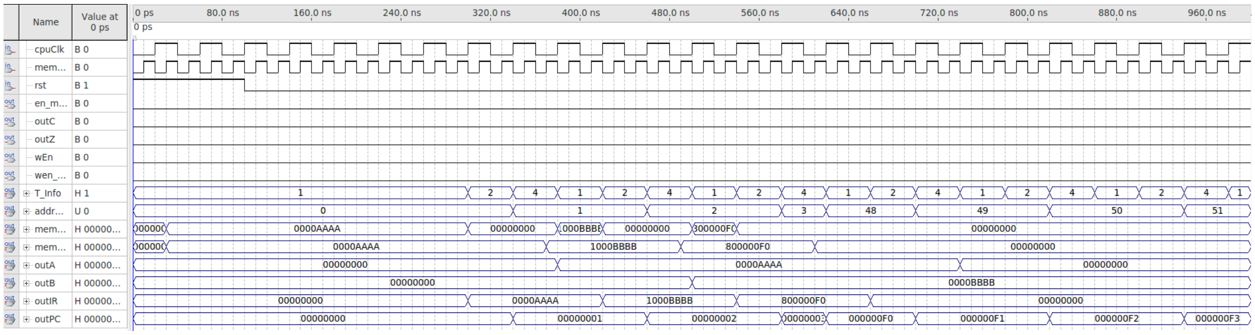


Figure 48. Functional waveform of BNE.

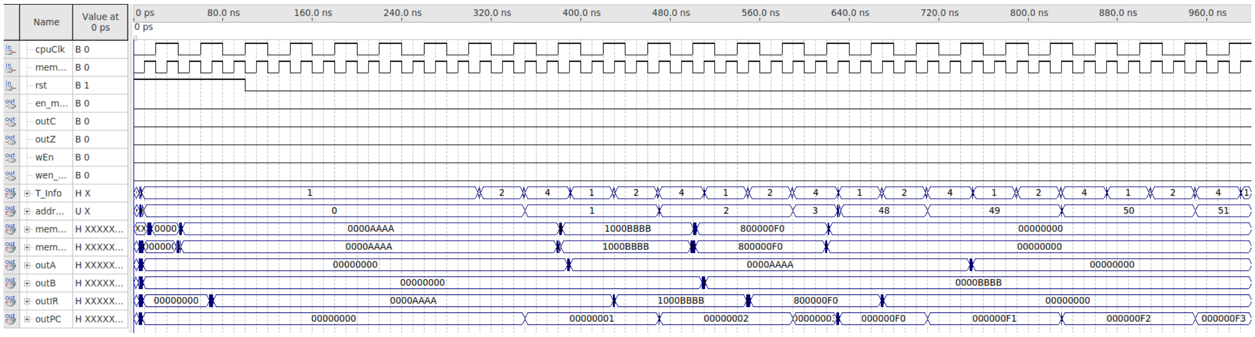


Figure 49. Timing-waveform of BNE.

4. References:

Hafeez, A. K (2023). COE608 CPU Specification [Ebook] (pp. 1-*). Retrieved from https://courses.torontomu.ca/d2l/le/content/836647/viewContent/5466481/View

Hafeez, A. K (2023). COE608 Lab 6 Manual [Ebook] (pp. 1-*). Retrieved from https://courses.torontomu.ca/d2l/le/content/836647/viewContent/5466483/View

Al-Qawasmi M. (2023). COE608 Lab 6 Tutorial [Ebook] (pp. 1-*). Retrieved from https://courses.torontomu.ca/d2l/le/content/836647/viewContent/5466484/View