Nanoprocessor Design

Project Report

#Group 44

- A.S. Jayathunga - 200265T
- M.R.A.A.K. Gunasinghe - 200196G
- K.A.Anshan Lahiru Kavinda - 200300A
- K.K.A.J.S. Kumarasinghe - 200323V
- W.M.T.B. Weerasekara - 200698X

Assigned Lab Task

Design a 4-bit processor capable of executing four basic instructions and having memory for eight such instructions.

Available Instructions are as follows,

1. Update a register value with the addition of itself and a value of another register. (ADD)
2. Change a value of a register to its two's complementary negative. (NEG)
3. Update a register value with a 4-bit immediate value. (MOVI)
4. Jump to a specified instruction if the value of a particular register equals zero. (JZR)

After completing the design of the processor, verify its functionality by running a simple assembly program to calculate the sum of numbers from 1 to 3. Assembly program must be converted to the relevant machine code of our processor and stored in the memory we designed for our processor. Number of instructions of our assembly program cannot exceed eight since our processor only has the capability of accommodating eight instructions at once. Final result of the program must be displayed on a seven segment display integrated to our processor. Processor should also have two output flags to indicate the current state of the output value of the Arithmetic Unit. One of the flags indicates whether the value is zero and other flag indicates if an overflow condition occurs. These two flags must be connected to two LEDs.

A Brief Description about our implementation of Nanoprocessor

RESET : **Center push button**
Register_7_output : **LED (2 downto 0)**
zero_flag: **LED 14**
overflow(Carry)_flag: **LED 15**

PS : R0 Register in Register Bank is Read Only with the value zero.

Instruction Set

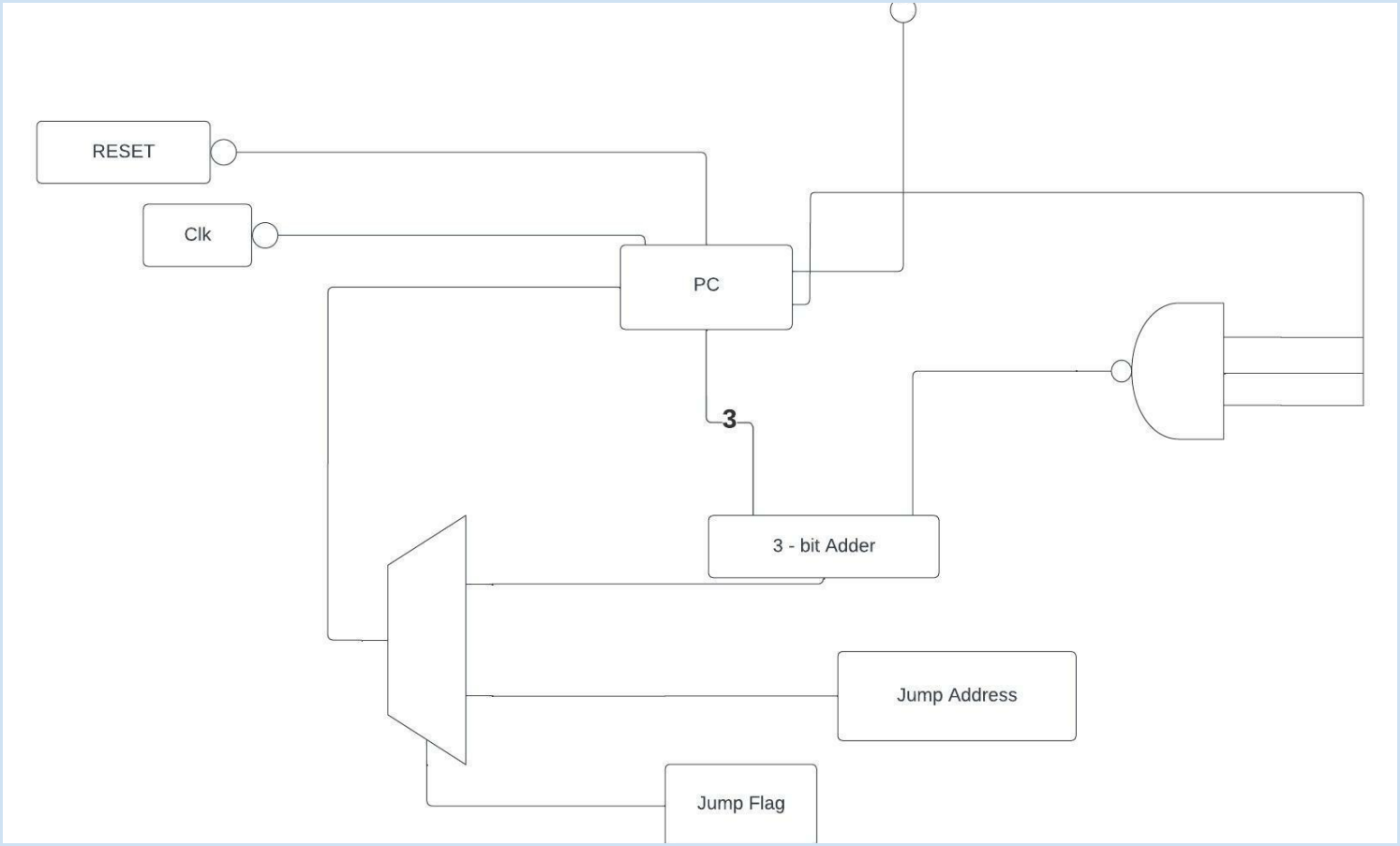
Instruction	**Operation**	**Format (12-bit Instruction)**
MOVI R,d	R ← d	10RRR000dddd
ADD R_a,R_b	R_a ← R_a+R_b	00$R_a$$R_a$$R_a$$R_b$$R_b$$R_b$0000
NEG R	R ← -R	01RRR0000000
JZR R,d	If R==0 : PC ← d Else : PC ← PC+1	11RRR0000ddd
NOP	Do Nothing	000000000000

PS: Here all zeros (000000000000) can be considered as NOP because, even though it is originally an ADD instruction, here addition is performed on the read only register with value zero. Hence it does not manipulate any register value and basically does not make any effect on the processor.

Program counter and 3-bit adder Modification

Program counter value increments by 1 after each clock cycle until it reaches the last value of the Program ROM (7 in this implementation) and stops.

This is achieved by designing the 3-bit incremental adder in such a way that it changes its incremental value from 1 to 0 once its input value gets a specified number (7 in this implementation).



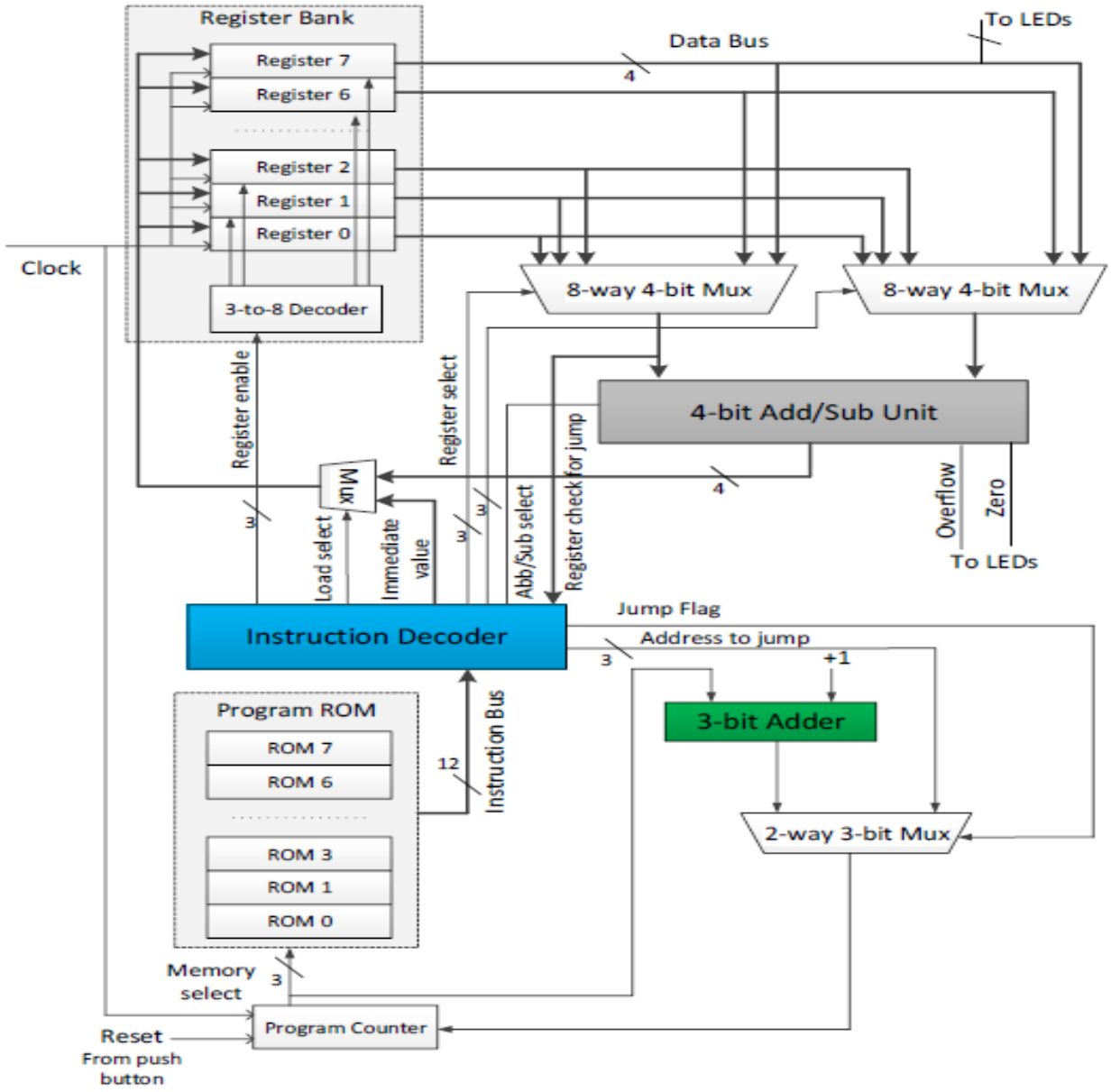
MUX Designs

- 8 way 4-bit MUX was designed using four 8 to 1 MUXs.
- 2 way 3-bit MUX was designed using three 2 to 1 MUXs.
- 2 way 4-bit MUX was designed using one 2 to 1 MUX and one 2 way 3-bit MUX.

Assembly program and its machine code representation of a program to calculate the sum of numbers from 1 to 3.

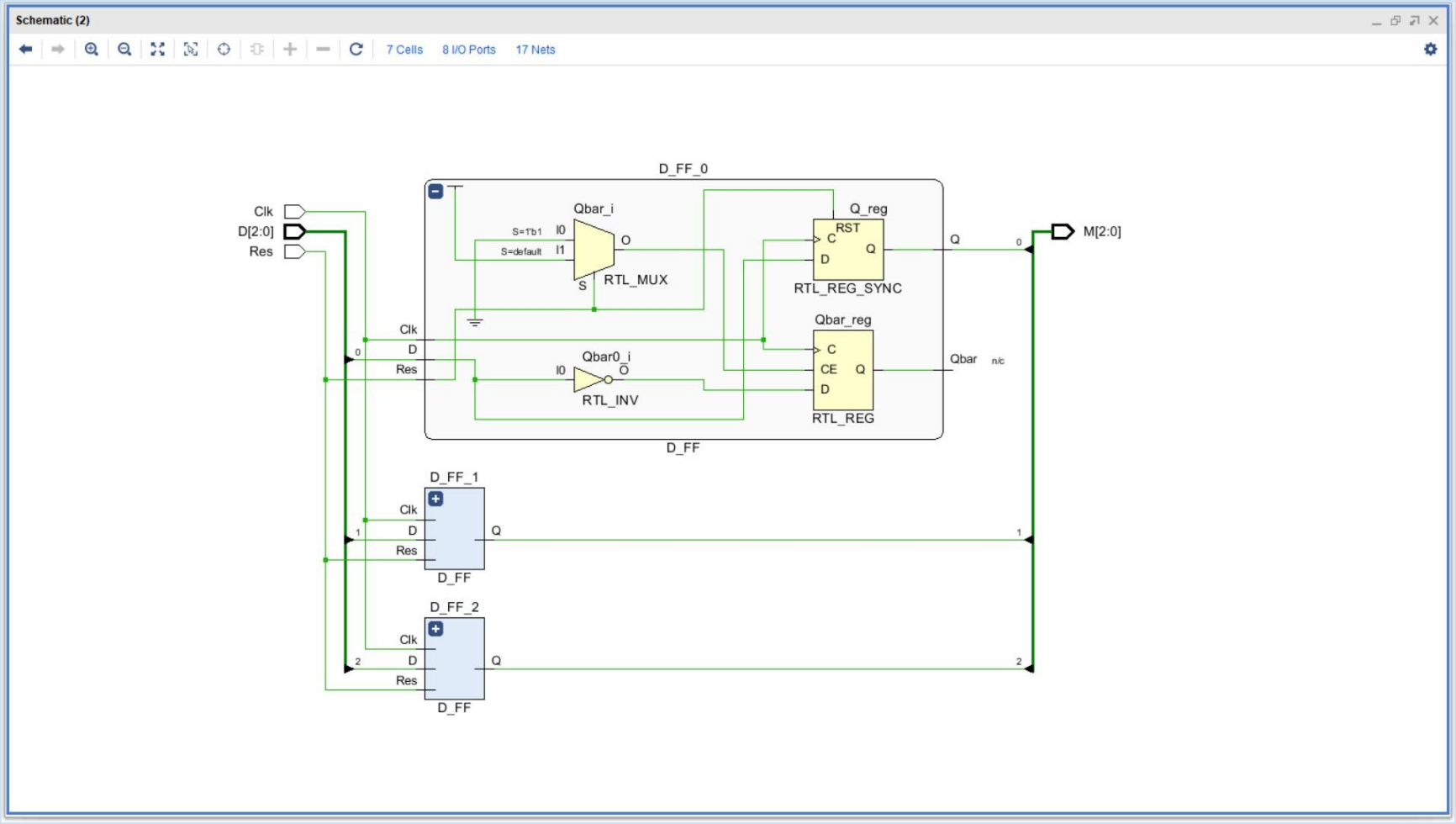
Assembly Code	Machine Code	Operation	Registers Contents
MOVI R1,1	100010000001	R1 <- 1	R7=0,R1=1
MOVI R2,1	100100000001	R2 <- 1	R7=0,R1=1,R2=1
ADD R7,R2	001110100000	R7 <- R7 + R2	R7=1,R1=1,R2=1
ADD R2,R1	000100010000	R2 <- R2 + R1	R7=1,R1=1,R2=2
ADD R7,R2	001110100000	R7 <- R7 + R2	R7=3,R1=1,R2=2
ADD R2,R1	000100010000	R2 <- R2 + R1	R7=3,R1=1,R2=3
ADD R7,R2	001110100000	R7 <- R7 + R2	R7=6,R1=1,R2=3
NOP	000000000000	R0 <- R0 + R0	R7=6,R0=0

Top Level Design Of Nanoprocessor



Program Counter

Schematic

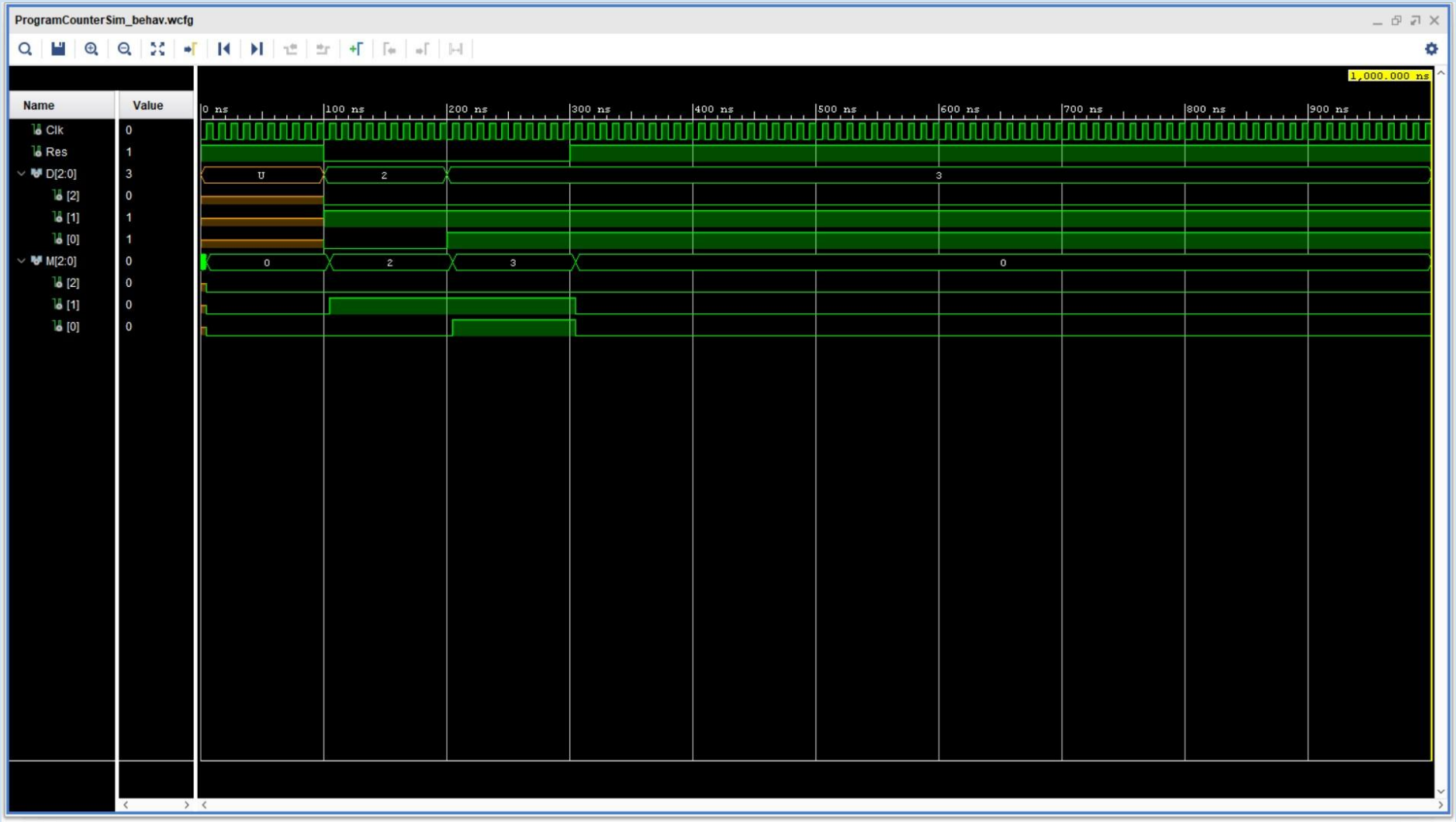


VHDL File

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/07/2022 11:28:56 PM
-- Design Name:
-- Module Name: ProgramCounter - Behavioral
-- Project Name: Nanoprocessor Design
-- Target Devices:
-- Tool Versions:
-- Description: Group 44
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ProgramCounter is
    Port ( D : in STD_LOGIC_VECTOR (2 downto 0);
           Clk : in STD_LOGIC;
           Res : in STD_LOGIC;
           M : out STD_LOGIC_VECTOR (2 downto 0));
end ProgramCounter;
```

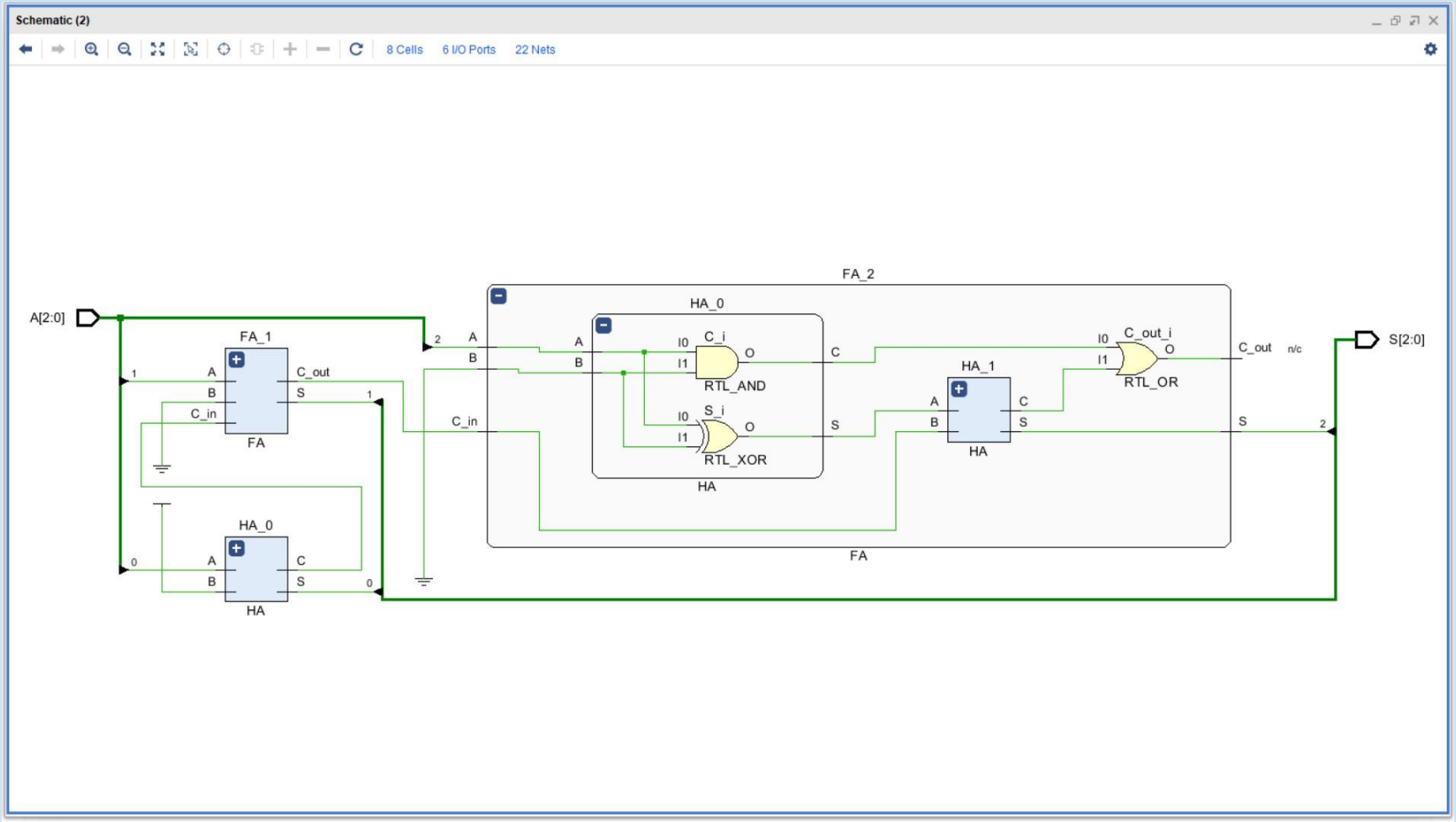
```
architecture Behavioral of ProgramCounter is

component D_FF
    Port ( D : in STD_LOGIC;
       Res : in STD_LOGIC;
       Clk : in STD_LOGIC;
       Q : out STD_LOGIC;
       Qbar : out STD_LOGIC);
end component;

signal D0,D1,D2 : STD_LOGIC;

begin
D_FF_0 : D_FF
    PORT MAP(
        D => D(0),
        Res => Res,
        Clk => Clk,
        Q => D0
    );

 D_FF_1 : D_FF
    PORT MAP(
        D => D(1),
        Res => Res,
        Clk => Clk,
        Q => D1
    );
 D_FF_2 : D_FF
    PORT MAP(
        D => D(2),
        Res => Res,
        Clk => Clk,
        Q => D2
    );
M(0) <= D0;
M(1) <= D1;
M(2) <= D2;
end Behavioral;
```

Timing Diagram



3-bit Ripple Carry Adder (Incremental)

Schematic



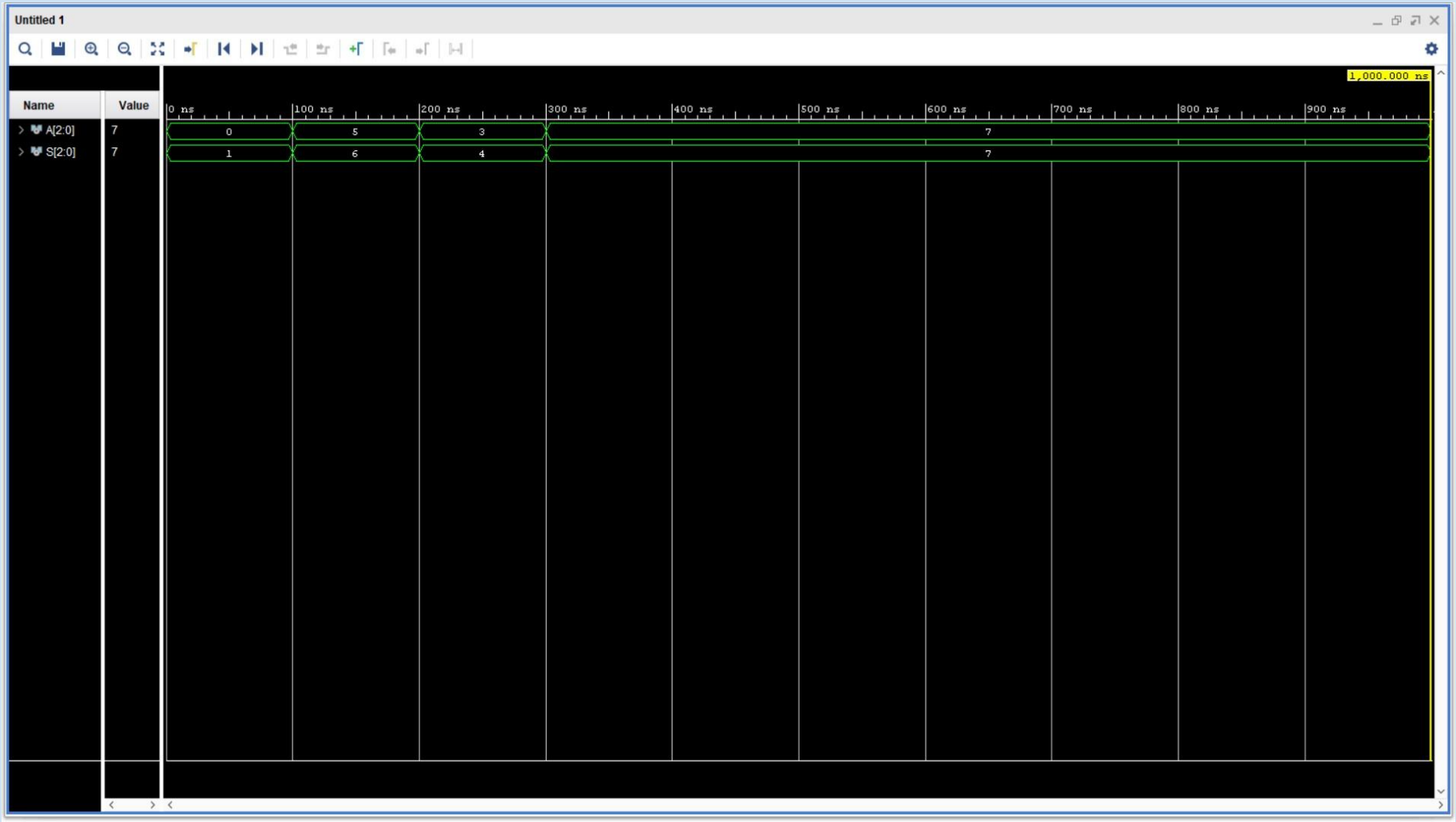
VHDL File

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/07/2022 10:07:28 PM
-- Design Name:
-- Module Name: RCA_3 - Behavioral
-- Project Name: Nanoprocessor Design
-- Target Devices:
-- Tool Versions:
-- Description: Group 44
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity RCA_3 is
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           S : out STD_LOGIC_VECTOR (2 downto 0));
end RCA_3;
```

```
architecture Behavioral of RCA_3 is
component FA
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C_in : in STD_LOGIC;
           S : out STD_LOGIC;
           C_out : out STD_LOGIC);
end component;

component HA
Port ( A : in STD_LOGIC;
       B : in STD_LOGIC;
       S : out STD_LOGIC;
       C : out STD_LOGIC);
end component;

signal HA0_C,FA1_C,FA2_C : STD_LOGIC;
signal B0_in : STD_LOGIC;

begin

HA_0 : HA
    PORT MAP(
        A => A(0),
        B => B0_in, --'1',
        S => S(0),
        C => HA0_C

    );
FA_1 : FA
    PORT MAP(
        A => A(1),
        B => '0',
        C_in => HA0_C,
        S => S(1),
        C_out => FA1_C

    );
```

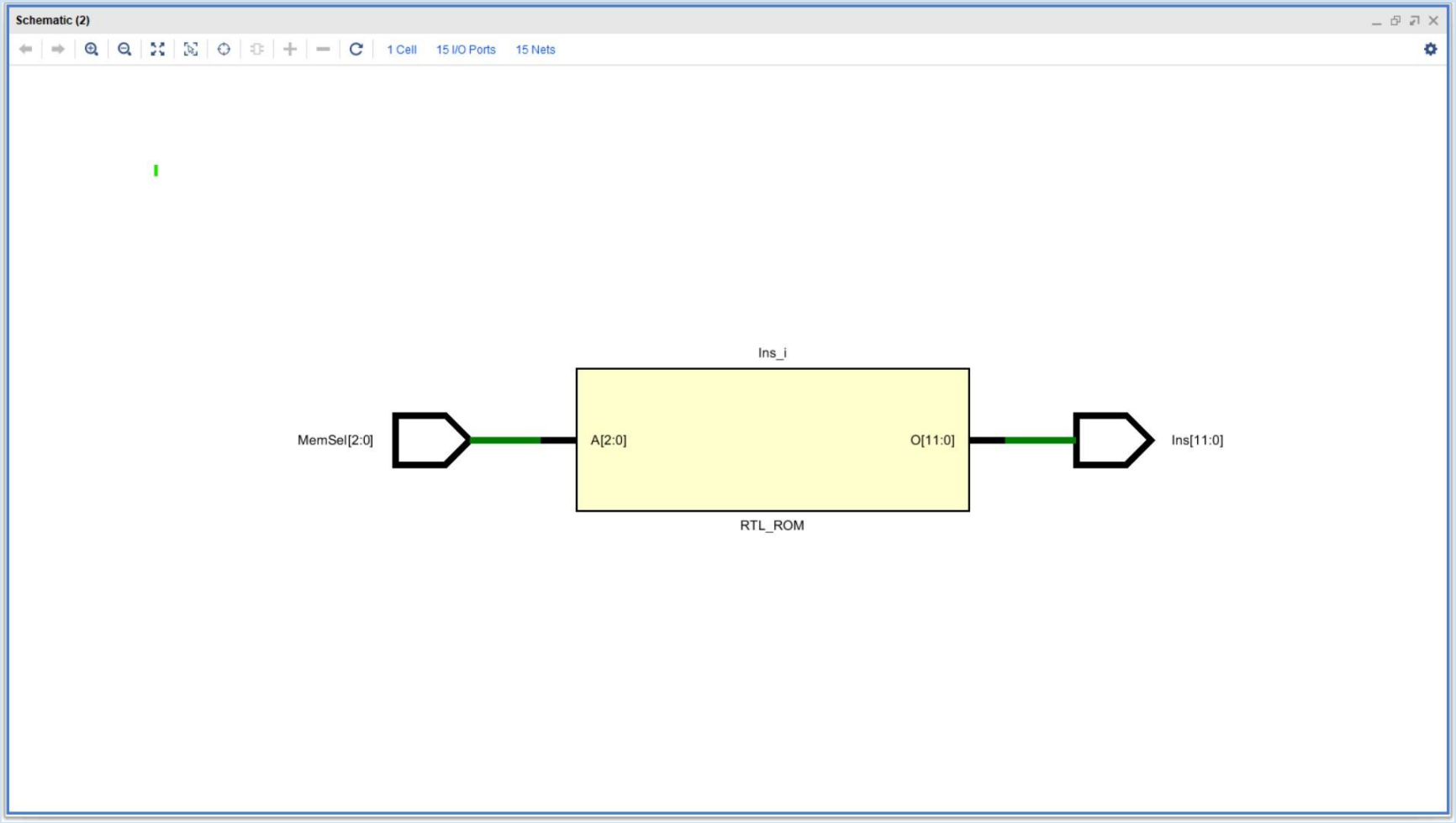
```
FA_2 : FA
    PORT MAP(
        A => A(2),
        B => '0',
        S => S(2),
        C_in => FA1_C,
        C_out => FA2_C
    );

--C_out <= FA2_C;
B0_in <= NOT(A(0)) OR NOT(A(1)) OR NOT(A(2));

end Behavioral;
```

Timing Diagram



Program Rom

Schematic

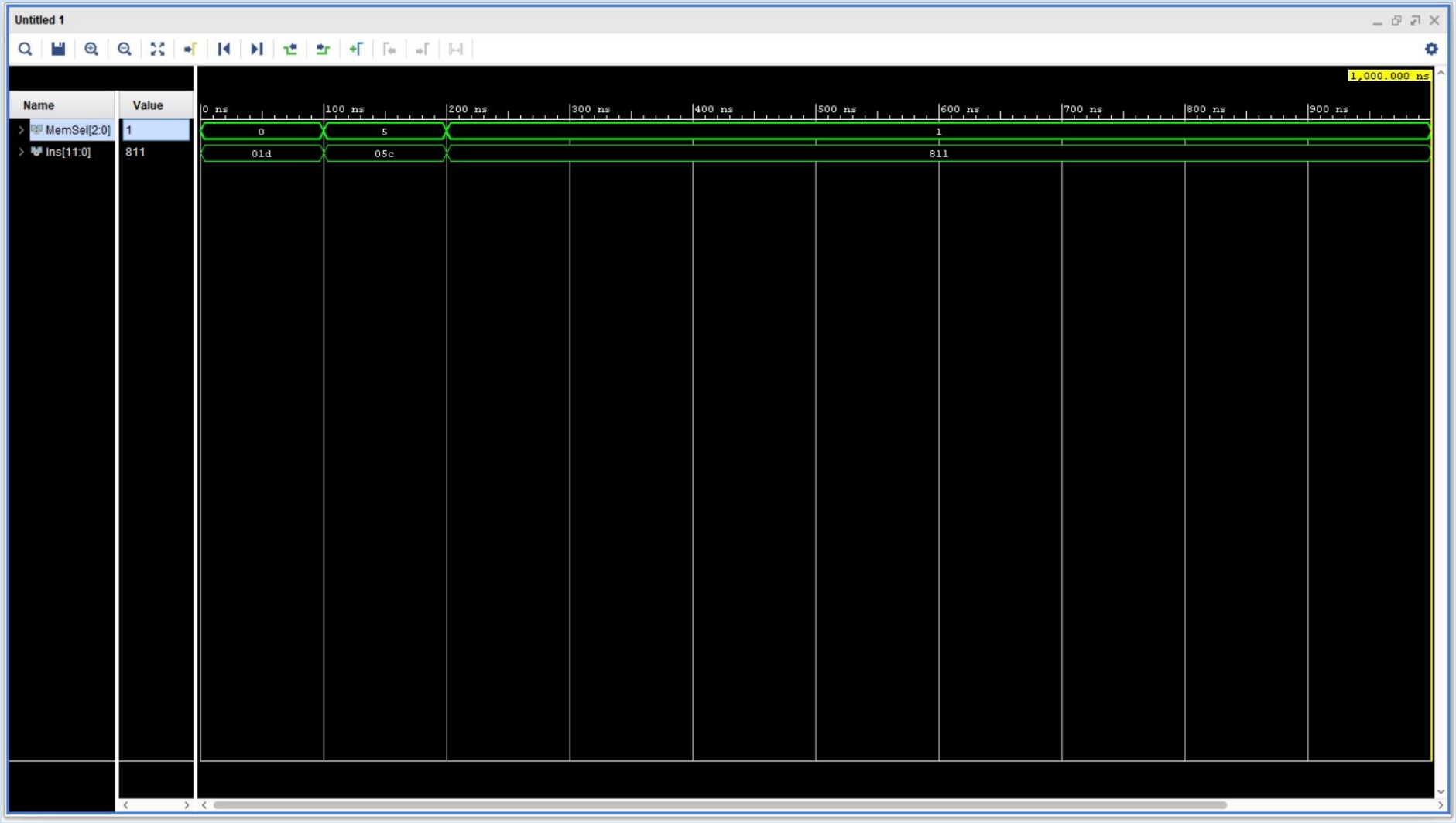


VHDL File

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/08/2022 07:24:52 AM
-- Design Name:
-- Module Name: ProgramRom - Behavioral
-- Project Name: Nanoprocessor Design
-- Target Devices:
-- Tool Versions:
-- Description: Group 44
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ProgramRom is
    Port ( Ins : out STD_LOGIC_VECTOR (11 downto 0);
           MemSel : in STD_LOGIC_VECTOR (2 downto 0));
end ProgramRom;
```

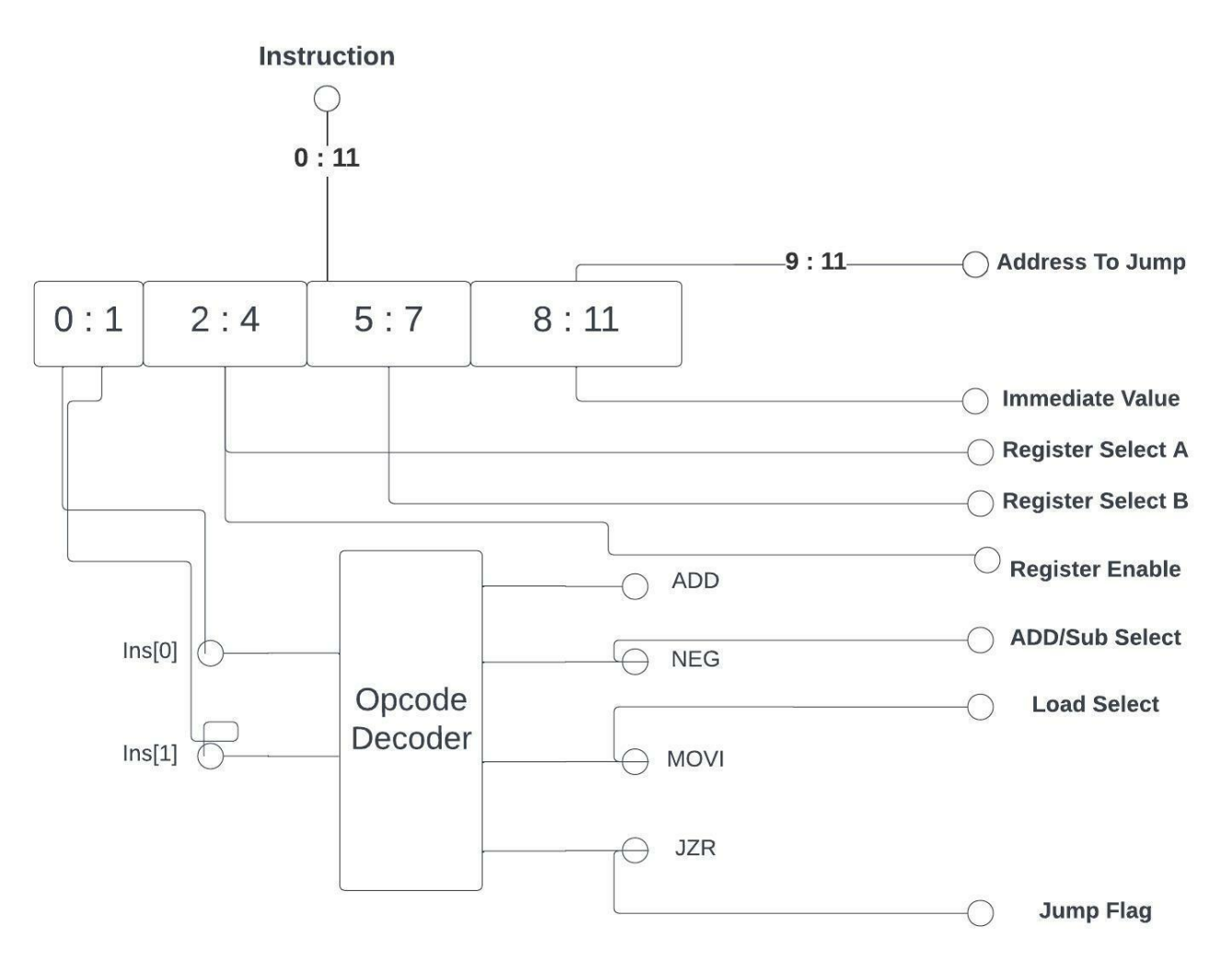
```
architecture Behavioral of ProgramRom is

type rom_type is array (0 to 7) of std_logic_vector(11 downto 0);
signal program_ROM : rom_type := (
--"101110000000", --ROM 0, MOVI R7,0 ; R7 <- 0        : R7=0              : 000000011101
"100010000001", --ROM 0, MOVI R1,1 ; R1 <- 1         : R7=0,R1=1         : 100000010001
"100100000001", --ROM 1, MOVI R2,1 ; R2 <- 1         : R7=0,R1=1,R2=1    : 100000001001
"001110100000", --ROM 2, ADD R7,R2 ; R7 <- R7 + R2   : R7=1,R1=1,R2=1    : 000001011100
"000100010000", --ROM 3, ADD R2,R1 ; R2 <- R2 + R1   : R7=1,R1=1,R2=2    : 000010001000
"001110100000", --ROM 4, ADD R7,R2 ; R7 <- R7 + R2   : R7=3,R1=1,R2=2    : 000001011100
"000100010000", --ROM 5, ADD R2,R1 ; R2 <- R2 + R1   : R7=3,R1=1,R2=3    : 000010001000
"001110100000", --ROM 6, ADD R7,R2 ; R7 <- R7 + R2   : R7=6,R1=1,R2=3    : 000001011100
"000000000000"  --ROM 7, NOP ;  Do Nothing
);

begin

Ins <= program_ROM(to_integer(unsigned(MemSel)));


end Behavioral;
```

Timing Diagram

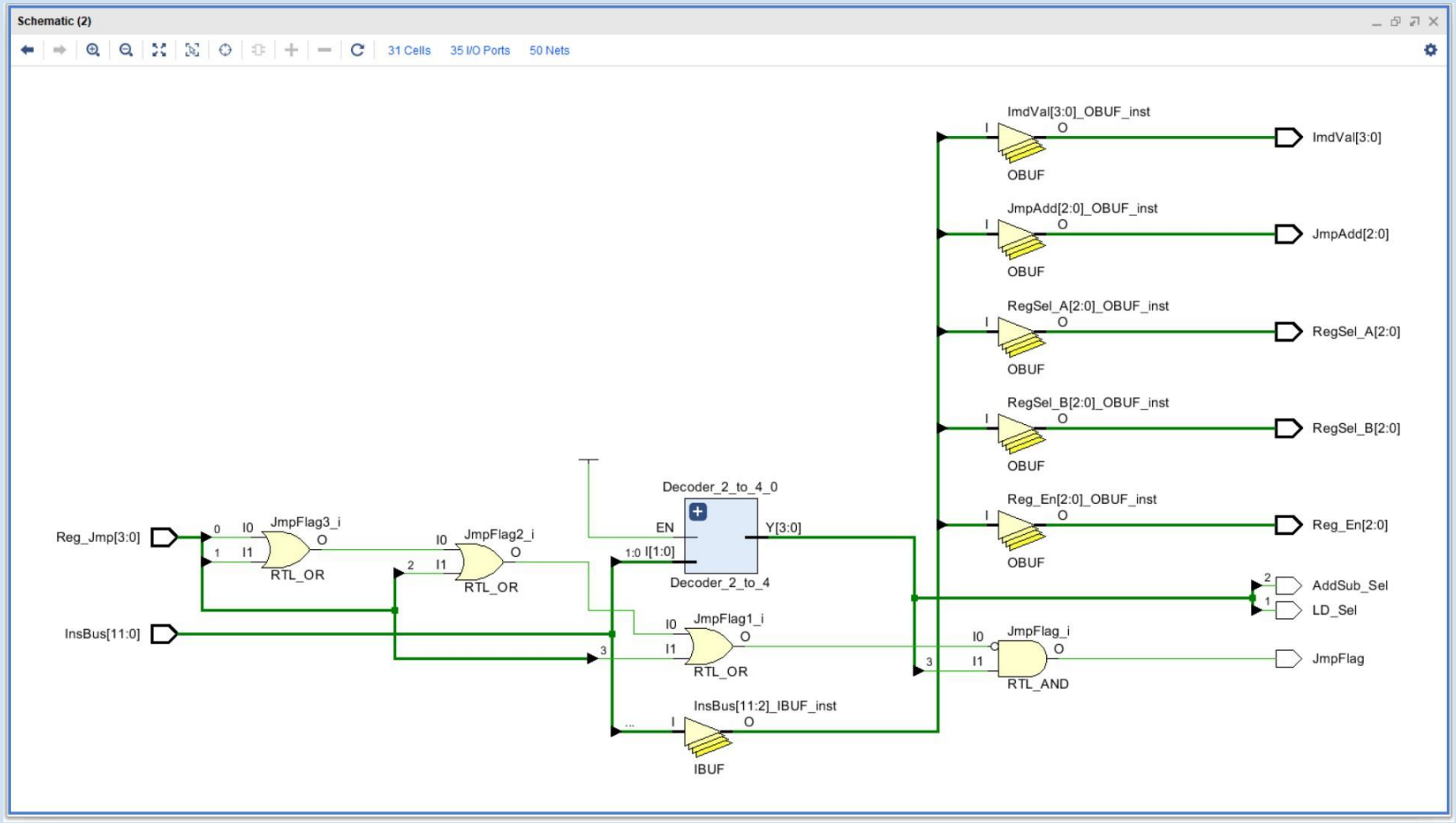


Instruction Decoder

Block Diagram



Schematic

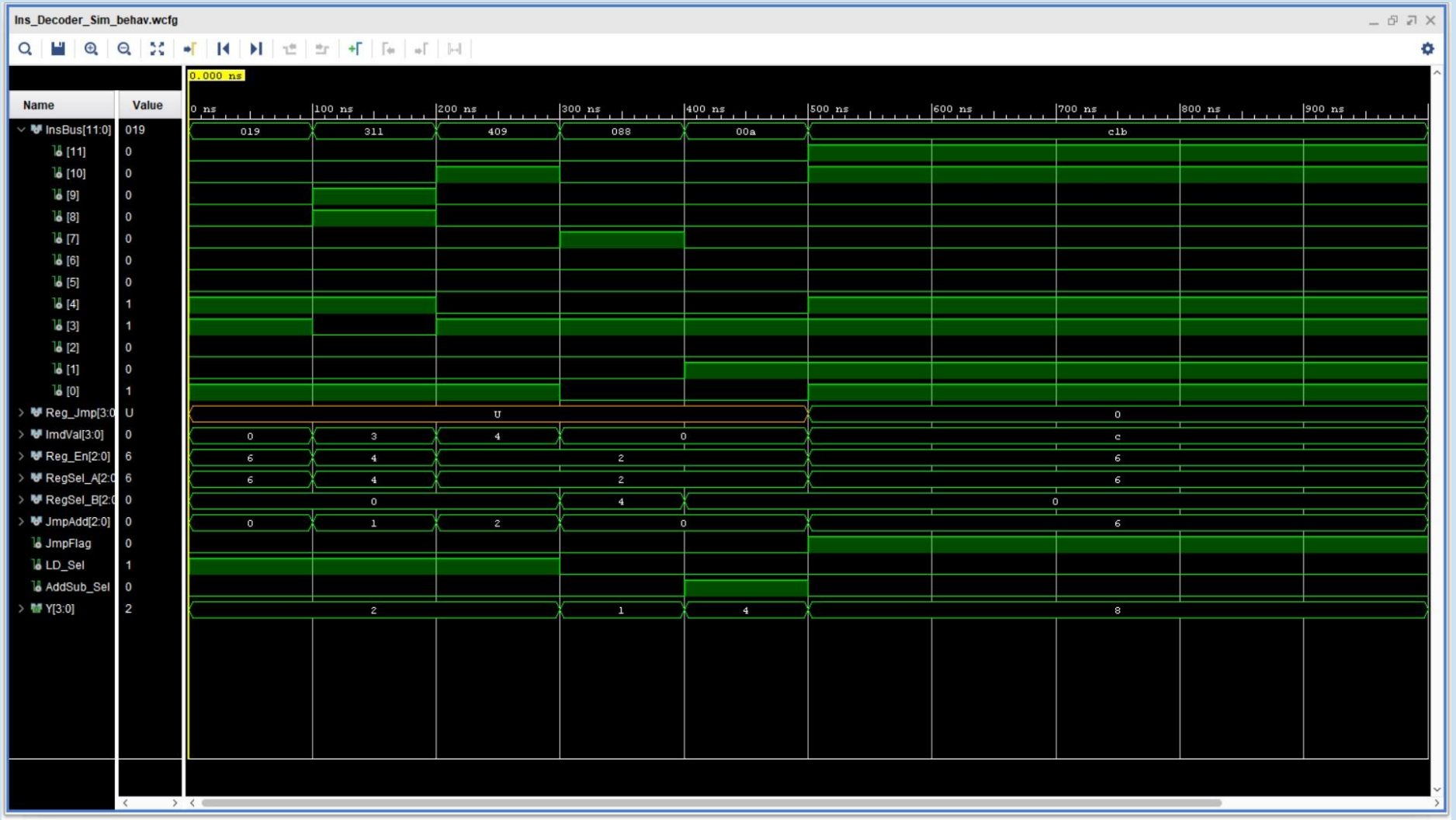

Schematic (2)
31 Cells
35 I/O Ports
50 Nets
ImdVal[3:0]_OBUF_inst
OBUF
ImdVal[3:0]
JmpAdd[2:0]_OBUF_inst
OBUF
JmpAdd[2:0]
RegSel_A[2:0]_OBUF_inst
OBUF
RegSel_A[2:0]
RegSel_B[2:0]_OBUF_inst
OBUF
RegSel_B[2:0]
Reg_En[2:0]_OBUF_inst
OBUF
Reg_En[2:0]
Decoder_2_to_4_0
EN
Y[3:0]
I[1:0]
Decoder_2_to_4
Reg_Jmp[3:0]
JmpFlag3_i
RTL_OR
JmpFlag2_i
RTL_OR
InsBus[11:0]
JmpFlag1_i
RTL_OR
JmpFlag_i
RTL_AND
JmpFlag
AddSub_Sel
LD_Sel
InsBus[11:2]_IBUF_inst
IBUF

VHDL File

```
-- Engineer:
--
-- Create Date: 07/10/2022 03:21:17 PM
-- Design Name:
-- Module Name: Ins_Decoder - Behavioral
-- Project Name: Nanoprocessor Design
-- Target Devices:
-- Tool Versions:
-- Description: Group 44
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Ins_Decoder is
    Port ( InsBus : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Jmp : in STD_LOGIC_VECTOR (3 downto 0);
           Reg_En : out STD_LOGIC_VECTOR (2 downto 0);
           RegSel_A : out STD_LOGIC_VECTOR (2 downto 0);
           RegSel_B : out STD_LOGIC_VECTOR (2 downto 0);
           ImdVal : out STD_LOGIC_VECTOR (3 downto 0);
           JmpAdd : out STD_LOGIC_VECTOR (2 downto 0);
           JmpFlag : out STD_LOGIC;
           LD_Sel : out STD_LOGIC;
           AddSub_Sel : out STD_LOGIC);
end Ins_Decoder;
```

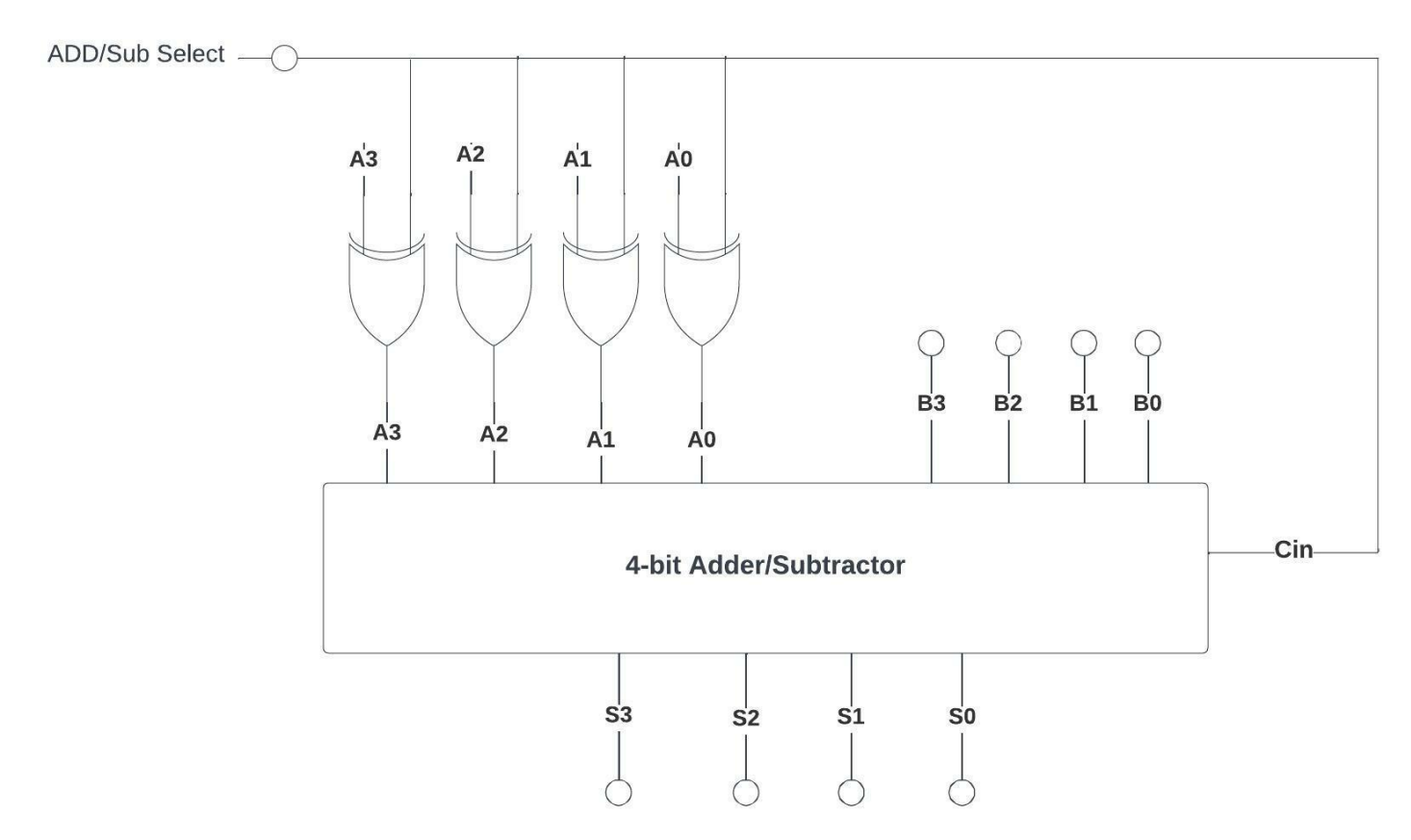
```
architecture Behavioral of Ins_Decoder is
component Decoder_2_to_4
Port ( I : in STD_LOGIC_VECTOR (1 downto 0);
       EN : in STD_LOGIC;
       Y : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal op_codes : STD_LOGIC_VECTOR (3 downto 0);
signal MOV,ADD,NEG,JZR : STD_LOGIC;
begin
Decoder_2_to_4_0:Decoder_2_to_4
    PORT MAP(
        I => InsBus(1 downto 0),
        EN => '1',
        Y => op_codes
    );

ADD <= op_codes(0);
NEG <= op_codes(2);
MOV <= op_codes(1);
JZR <= op_codes(3);

Reg_En <= InsBus(4 downto 2);
RegSel_A <= InsBus(4 downto 2);
RegSel_B <= InsBus(7 downto 5);
ImdVal <= InsBus(11 downto 8);
JmpAdd <= InsBus(11 downto 9);

AddSub_Sel <= NEG;
LD_Sel <= MOV;
JmpFlag <= NOT(Reg_Jmp(0) OR Reg_Jmp(1) OR Reg_Jmp(2) OR Reg_Jmp(3)) AND JZR;


end Behavioral;
```

Timing Diagram

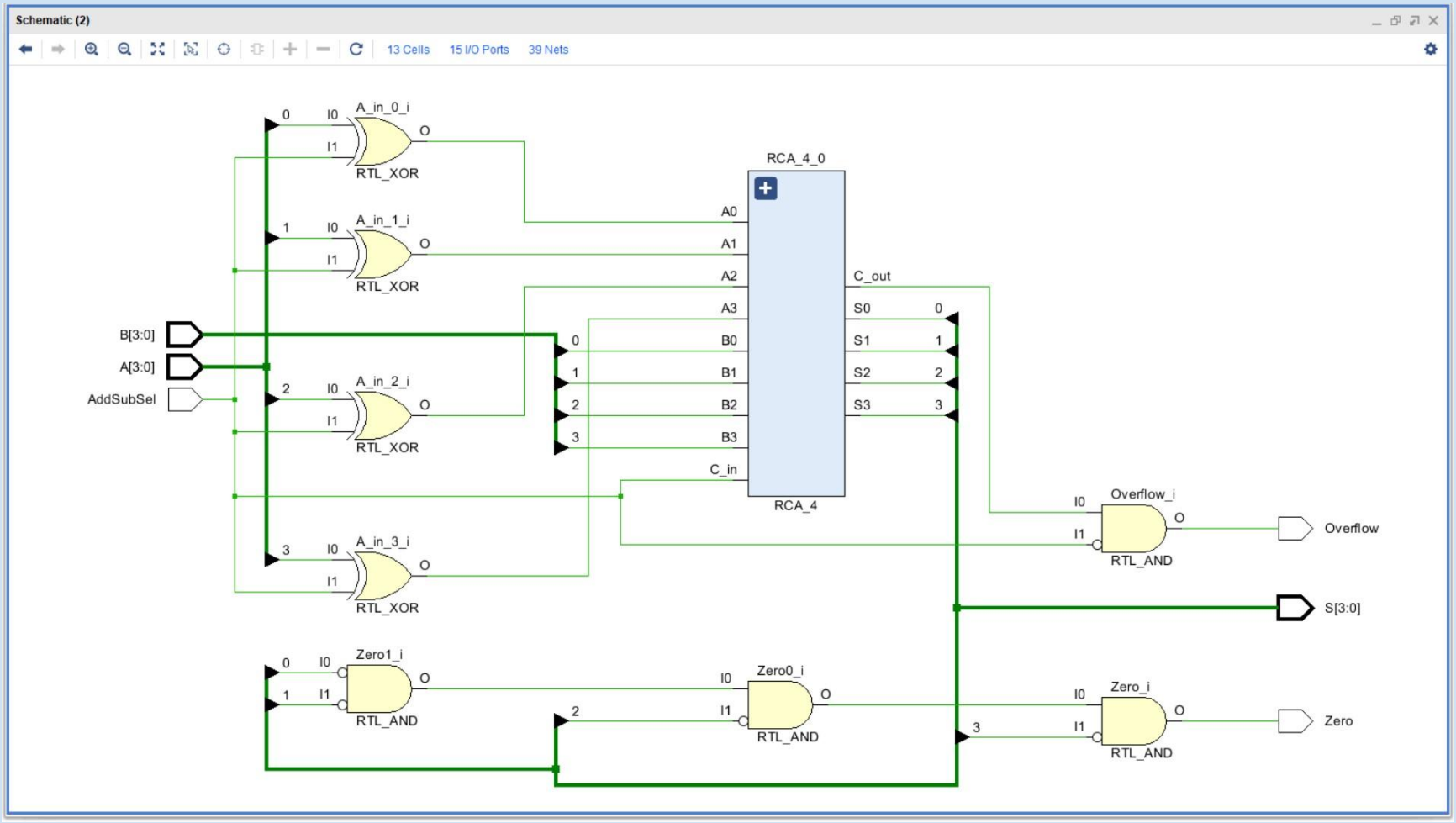


Arithmetic Unit

Block Diagram



Schematic


Schematic (2)
13 Cells
15 I/O Ports
39 Nets
A_in_0_i
A_in_1_i
A_in_2_i
A_in_3_i
RTL_XOR
RCA_4_0
RCA_4
A0
A1
A2
A3
B0
B1
B2
B3
C_in
C_out
S0
S1
S2
S3
B[3:0]
A[3:0]
AddSubSel
Overflow_i
RTL_AND
Overflow
S[3:0]
Zero1_i
Zero0_i
Zero_i
Zero

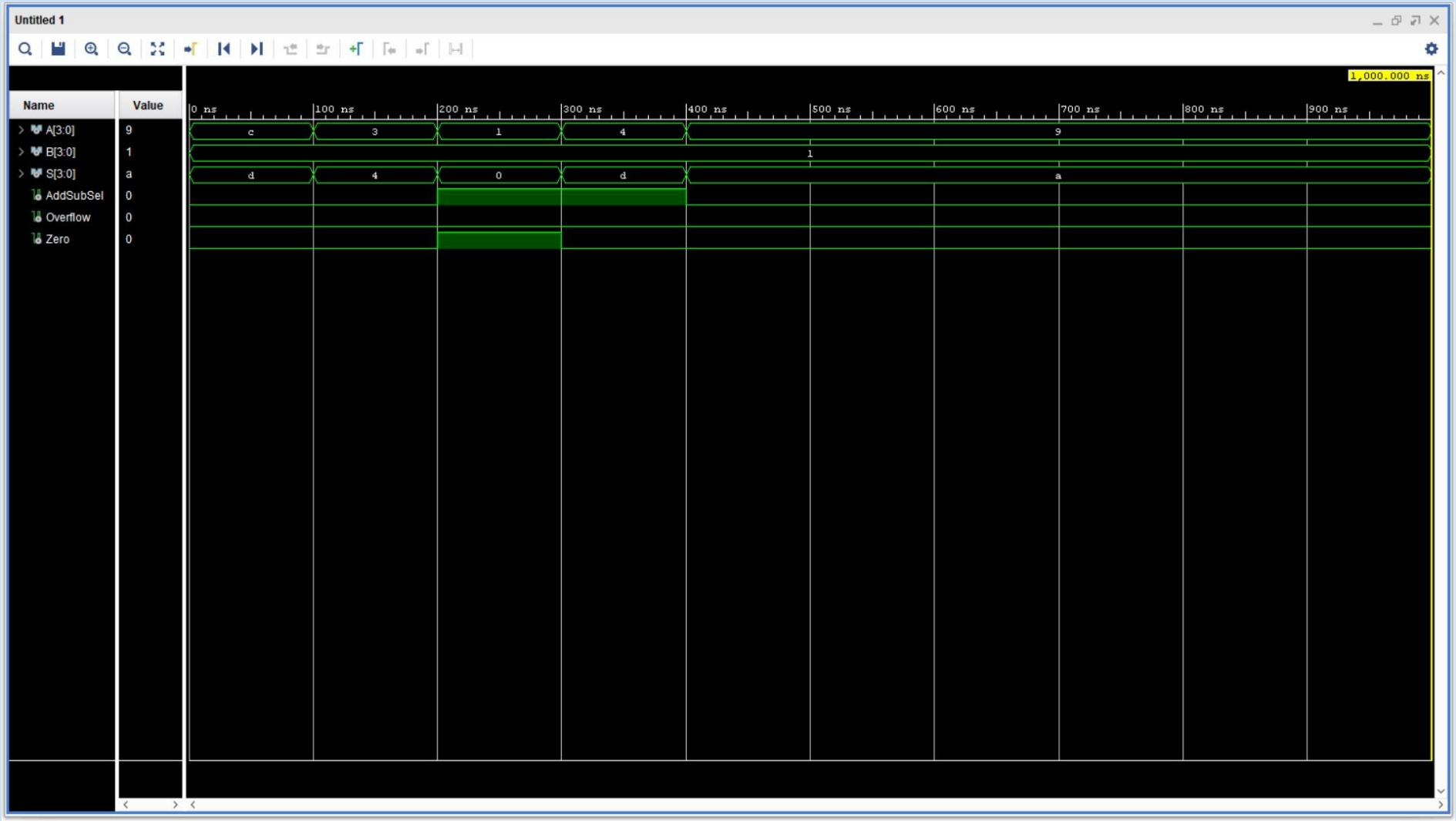
VHDL File

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/08/2022 07:30:19 PM
-- Design Name:
-- Module Name: Add_Sub_Unit_4 - Behavioral
-- Project Name: Nanoprocessor Design
-- Target Devices:
-- Tool Versions:
-- Description: Group 44
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity Add_Sub_Unit_4 is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           AddSubSel : in STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
           Overflow : out STD_LOGIC;
           Zero : out STD_LOGIC);
end Add_Sub_Unit_4;
```

```
architecture Behavioral of Add_Sub_Unit_4 is
component RCA_4
Port ( A0 : in STD_LOGIC;
       A1 : in STD_LOGIC;
       A2 : in STD_LOGIC;
       A3 : in STD_LOGIC;
       B0 : in STD_LOGIC;
       B1 : in STD_LOGIC;
       B2 : in STD_LOGIC;
       B3 : in STD_LOGIC;
       C_in : in STD_LOGIC;
       S0 : out STD_LOGIC;
       S1 : out STD_LOGIC;
       S2 : out STD_LOGIC;
       S3 : out STD_LOGIC;
       C_out : out STD_LOGIC);
end component;

signal RCA_out : STD_LOGIC_VECTOR (3 downto 0);
signal A_in : STD_LOGIC_VECTOR (3 downto 0);
signal RCA_carry : STD_LOGIC;

begin

RCA_4_0 : RCA_4
    PORT MAP(
        A0 => A_in(0),
        A1 => A_in(1),
        A2 => A_in(2),
        A3 => A_in(3),
        B0 => B(0),
        B1 => B(1),
        B2 => B(2),
        B3 => B(3),
        C_in => AddSubSel,
        S0 => RCA_out(0),
        S1 => RCA_out(1),
        S2 => RCA_out(2),
        S3 => RCA_out(3),
        C_out => RCA_carry
    );
```

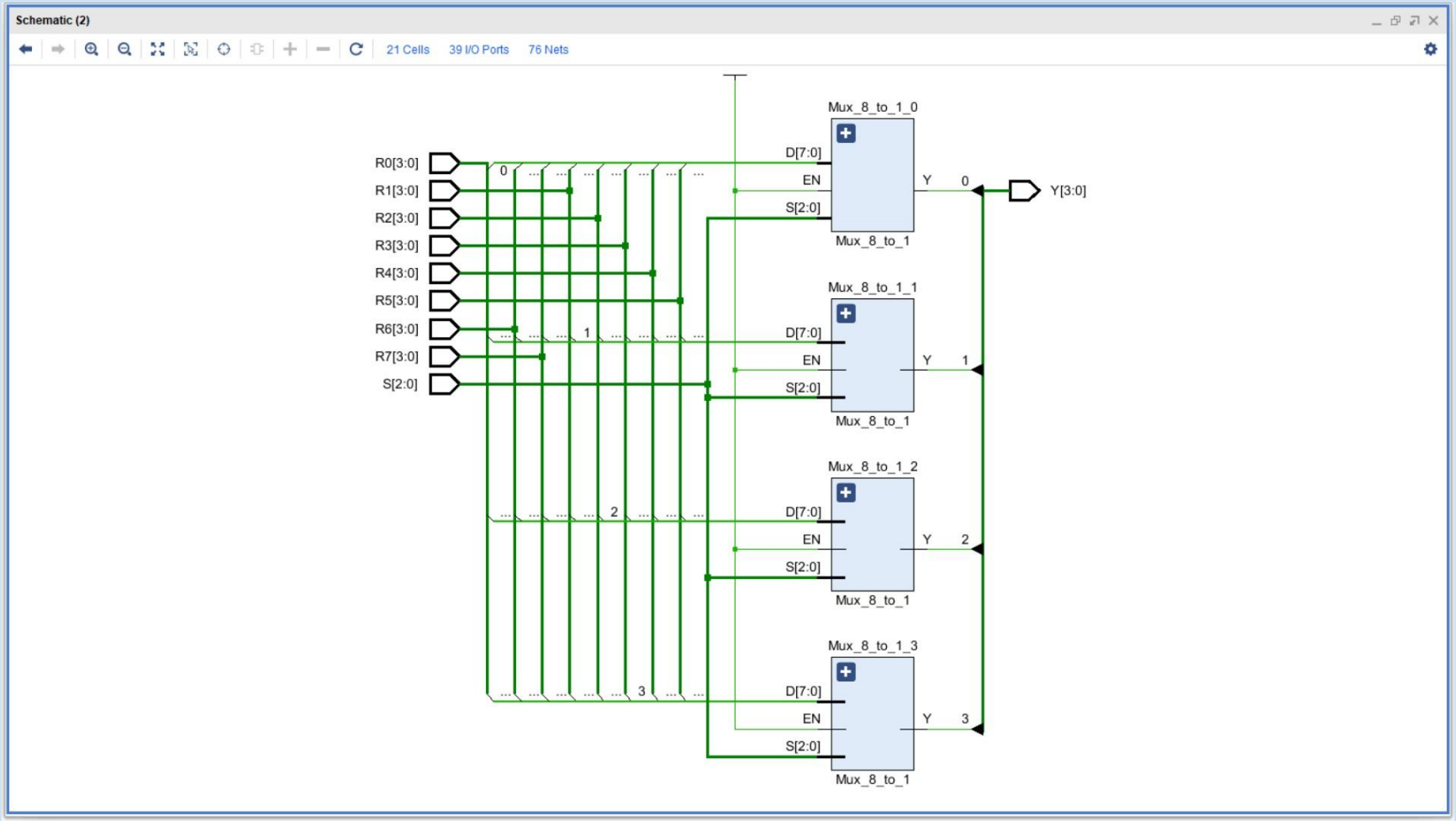
```
A_in(0) <= A(0) XOR AddSubSel;
A_in(1) <= A(1) XOR AddSubSel;
A_in(2) <= A(2) XOR AddSubSel;
A_in(3) <= A(3) XOR AddSubSel;


S(0) <= RCA_out(0);
S(1) <= RCA_out(1);
S(2) <= RCA_out(2);
S(3) <= RCA_out(3);

Overflow <= RCA_carry AND NOT(AddSubSel);

Zero <= NOT(RCA_out(0)) AND NOT(RCA_out(1)) AND NOT(RCA_out(2)) AND NOT(RCA_out(3));

end Behavioral;
```

Timing Diagram



8 Way 4-bit Multiplexer

Schematic



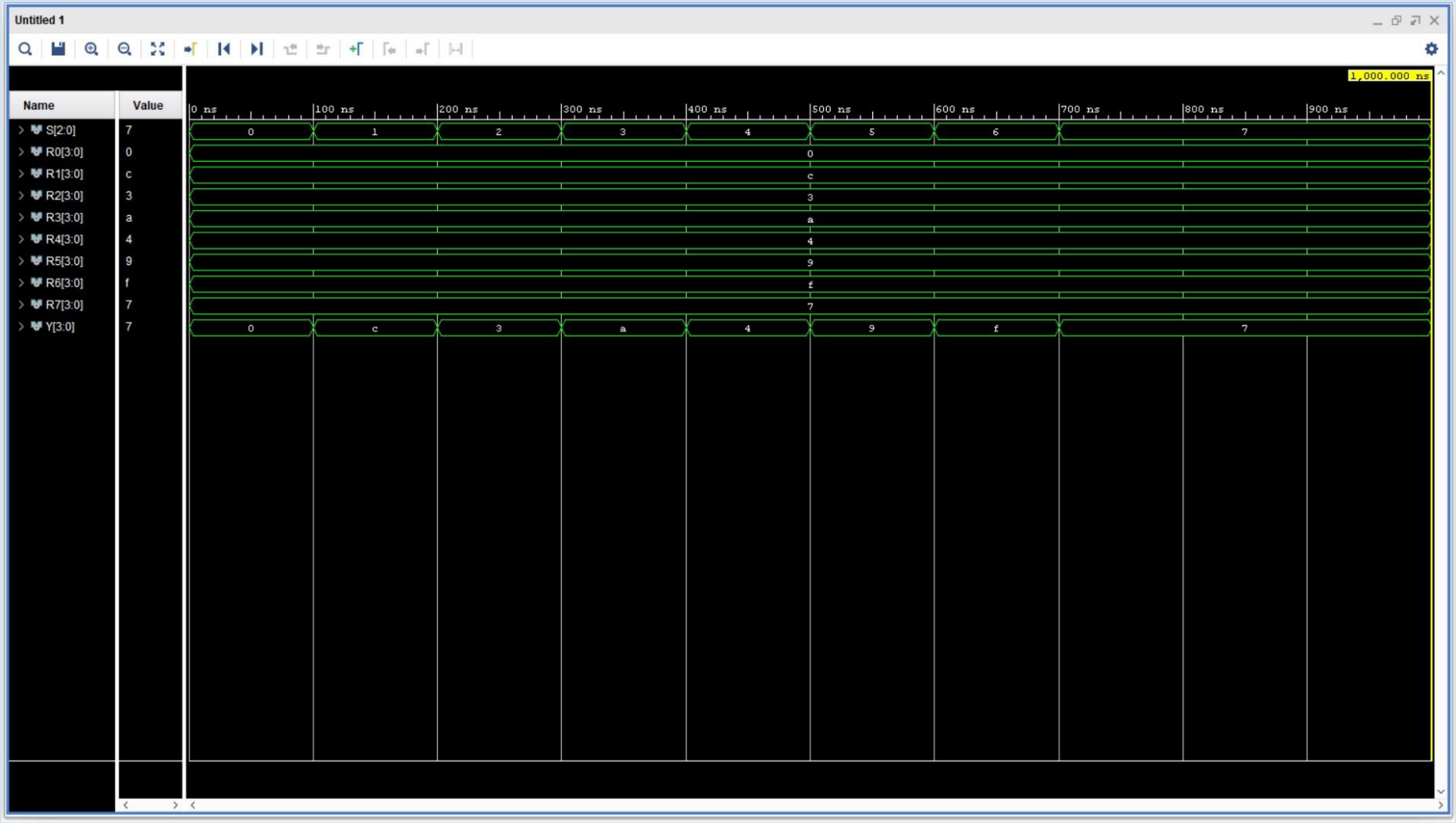
VHDL File

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/07/2022 08:01:08 PM
-- Design Name:
-- Module Name: Mux_8_way_4_bit - Behavioral
-- Project Name: Nanoprocessor Design
-- Target Devices:
-- Tool Versions:
-- Description: Group 44
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity Mux_8_way_4_bit is
    Port ( R0 : in STD_LOGIC_VECTOR (3 downto 0);
           R1 : in STD_LOGIC_VECTOR (3 downto 0);
           R2 : in STD_LOGIC_VECTOR (3 downto 0);
           R3 : in STD_LOGIC_VECTOR (3 downto 0);
           R4 : in STD_LOGIC_VECTOR (3 downto 0);
           R5 : in STD_LOGIC_VECTOR (3 downto 0);
           R6 : in STD_LOGIC_VECTOR (3 downto 0);
           R7 : in STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_8_way_4_bit;
```

```
architecture Behavioral of Mux_8_way_4_bit is
component Mux_8_to_1
Port (      D : in STD_LOGIC_VECTOR (7 downto 0);
            S : in STD_LOGIC_VECTOR (2 downto 0);
            EN : in STD_LOGIC;
            Y : out STD_LOGIC);
end component;

begin

Mux_8_to_1_0 : Mux_8_to_1
    PORT MAP(
        D(0) => R0(0),
        D(1) => R1(0),
        D(2) => R2(0),
        D(3) => R3(0),
        D(4) => R4(0),
        D(5) => R5(0),
        D(6) => R6(0),
        D(7) => R7(0),
        S => S,
        EN => '1',
        Y => Y(0)
        );
Mux_8_to_1_1 : Mux_8_to_1
    PORT MAP(
        D(0) => R0(1),
        D(1) => R1(1),
        D(2) => R2(1),
        D(3) => R3(1),
        D(4) => R4(1),
        D(5) => R5(1),
        D(6) => R6(1),
        D(7) => R7(1),
        S => S,
        EN => '1',
        Y => Y(1)
    );
```

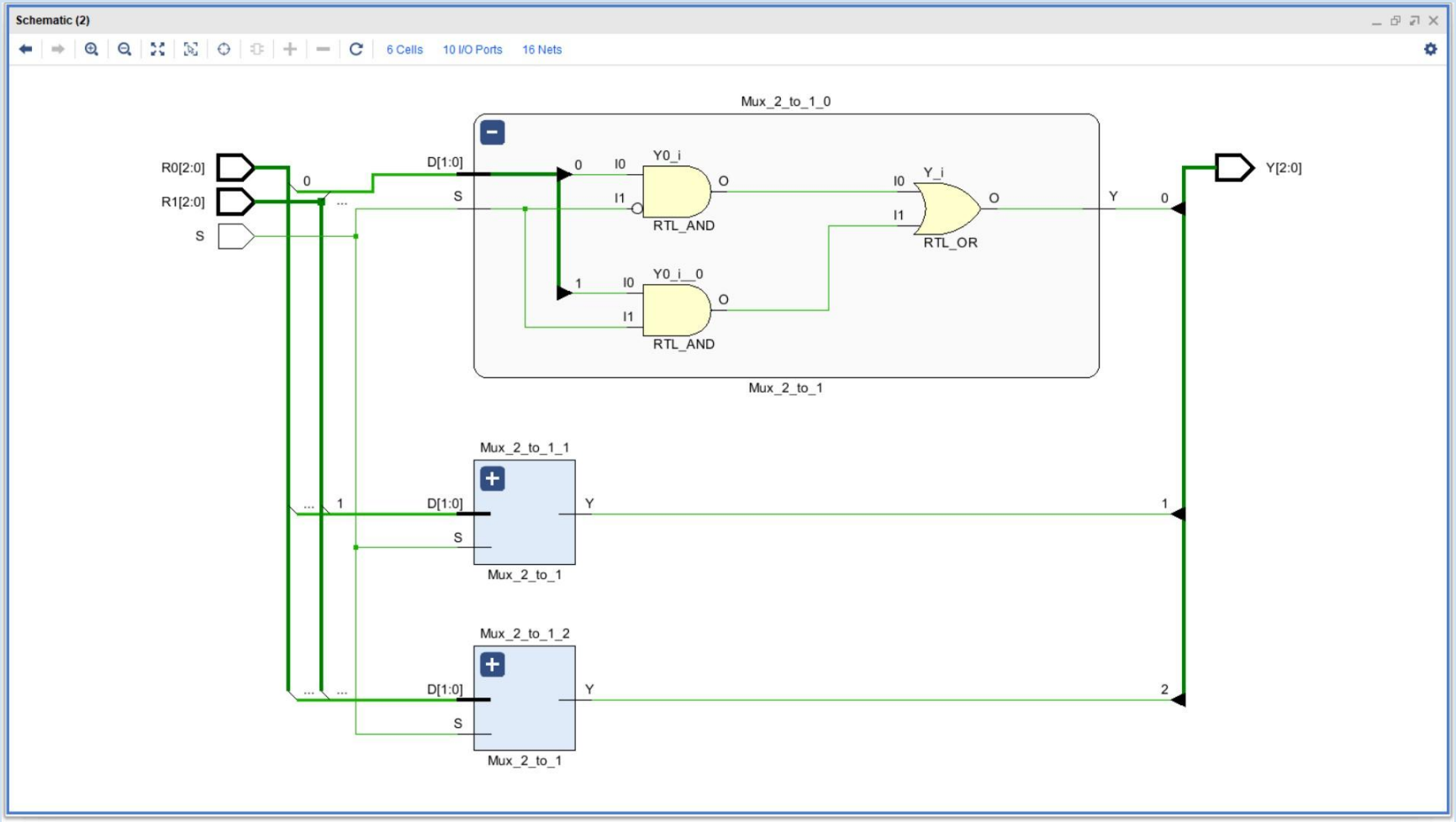
```
Mux_8_to_1_2 : Mux_8_to_1
    PORT MAP(
        D(0) => R0(2),
        D(1) => R1(2),
        D(2) => R2(2),
        D(3) => R3(2),
        D(4) => R4(2),
        D(5) => R5(2),
        D(6) => R6(2),
        D(7) => R7(2),
        S => S,
        EN => '1',
        Y => Y(2)
        );
Mux_8_to_1_3 : Mux_8_to_1
    PORT MAP(
            D(0) => R0(3),
            D(1) => R1(3),
            D(2) => R2(3),
            D(3) => R3(3),
            D(4) => R4(3),
            D(5) => R5(3),
            D(6) => R6(3),
            D(7) => R7(3),
            S => S,
            EN => '1',
            Y => Y(3)
            );


end Behavioral;
```

Timing Diagram



2 way 3-bit Multiplexer

Schematic

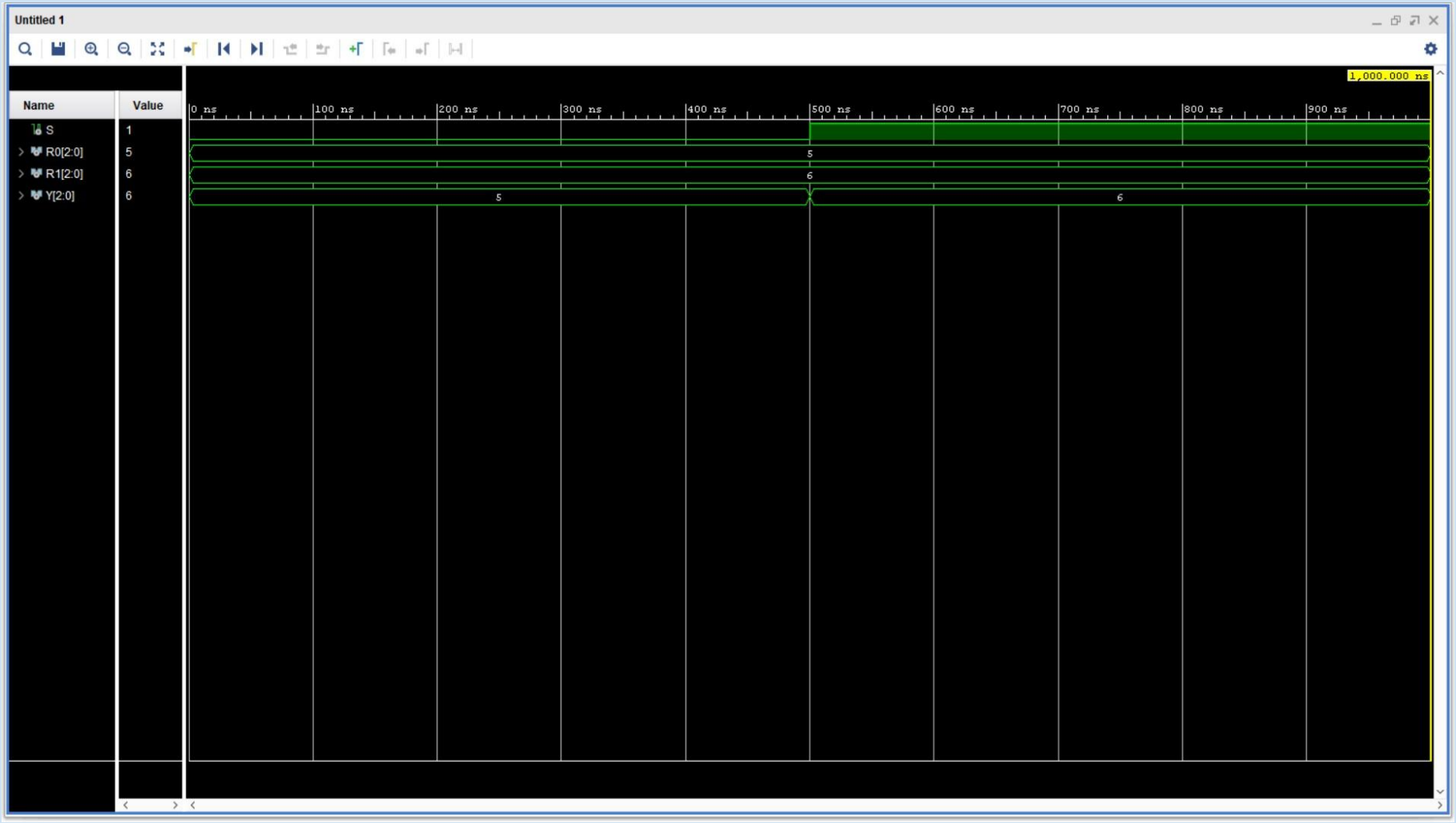


VHDL File

```
--
-- Create Date: 07/07/2022 09:27:35 PM
-- Design Name:
-- Module Name: Mux_2_way_3_bit - Behavioral
-- Project Name: Nanoprocessor Design
-- Target Devices:
-- Tool Versions:
-- Description: Group 44
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity Mux_2_way_3_bit is
    Port ( R0 : in STD_LOGIC_VECTOR (2 downto 0);
           R1 : in STD_LOGIC_VECTOR (2 downto 0);
           S : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end Mux_2_way_3_bit;

architecture Behavioral of Mux_2_way_3_bit is
component Mux_2_to_1
    Port ( D : in STD_LOGIC_VECTOR (1 downto 0);
      S : in STD_LOGIC;
      Y : out STD_LOGIC);
end component;
```

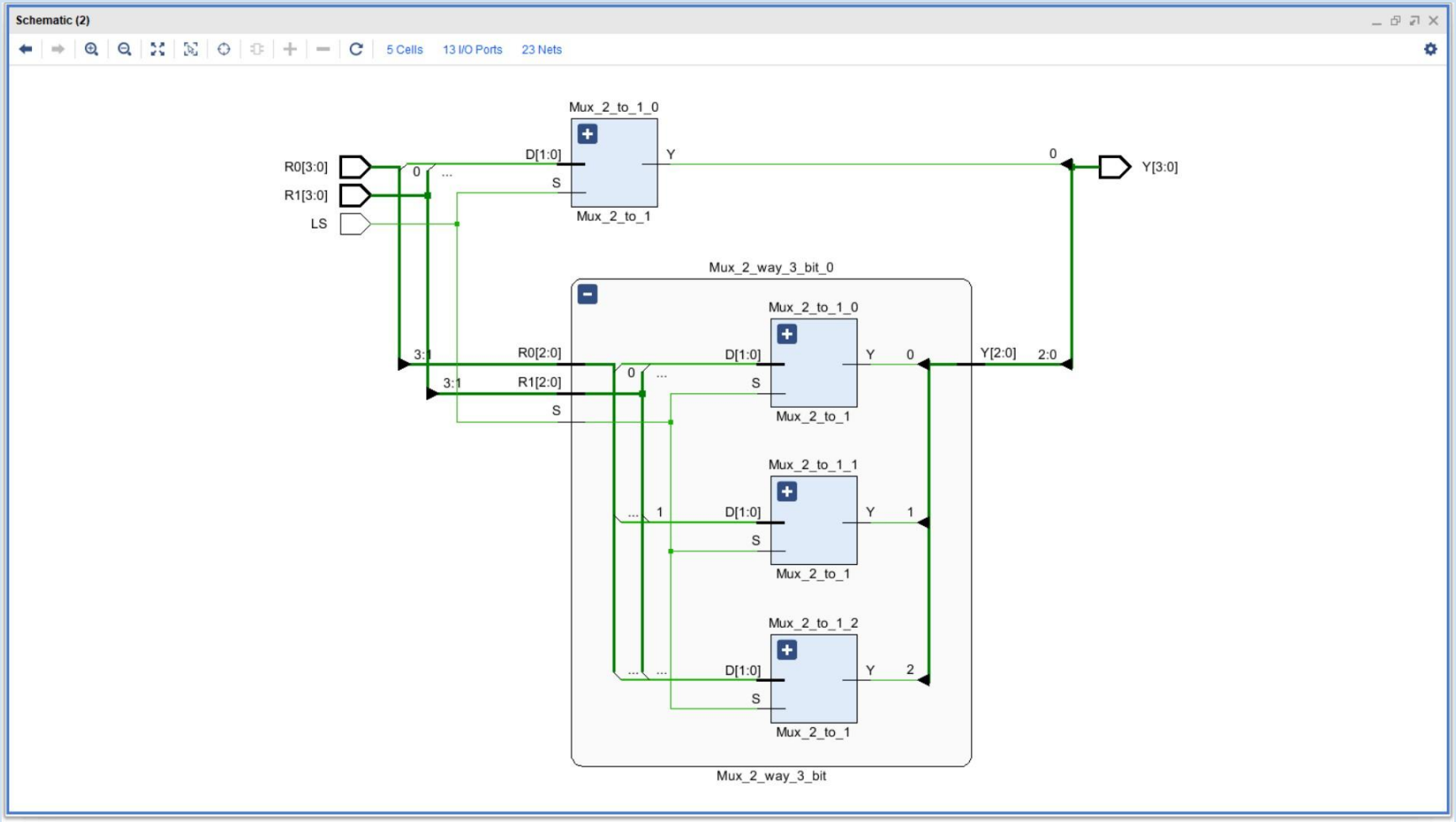
```
begin

Mux_2_to_1_0 : Mux_2_to_1
    PORT MAP(
     D(0) => R0(0),
     D(1) => R1(0),
     S => S,
     Y => Y(0)
    );

Mux_2_to_1_1 : Mux_2_to_1
    PORT MAP(
     D(0) => R0(1),
     D(1) => R1(1),
     S => S,
     Y => Y(1)
    );

Mux_2_to_1_2 : Mux_2_to_1
    PORT MAP(
     D(0) => R0(2),
     D(1) => R1(2),
     S => S,
     Y => Y(2)
    );



end Behavioral;
```

Timing Diagram



2 way 4-bit Multiplexer

Schematic

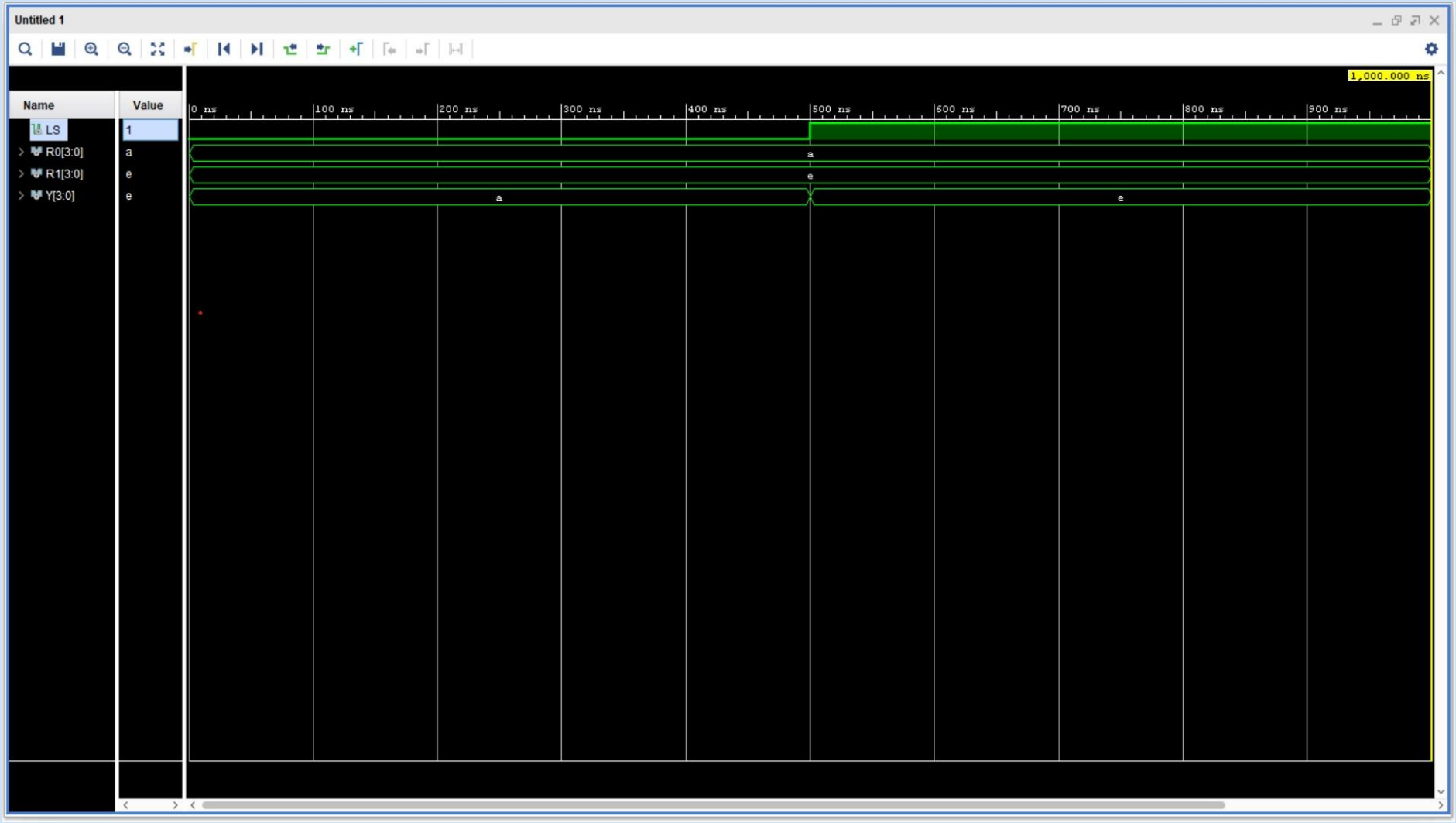


VHDL File

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/07/2022 10:45:47 PM
-- Design Name:
-- Module Name: Mux_2_way_4_bit - Behavioral
-- Project Name: Nanoprocessor Design
-- Target Devices:
-- Tool Versions:
-- Description: Group 44
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity Mux_2_way_4_bit is
    Port ( R0 : in STD_LOGIC_VECTOR (3 downto 0);
           R1 : in STD_LOGIC_VECTOR (3 downto 0);
           LS : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_2_way_4_bit;
```

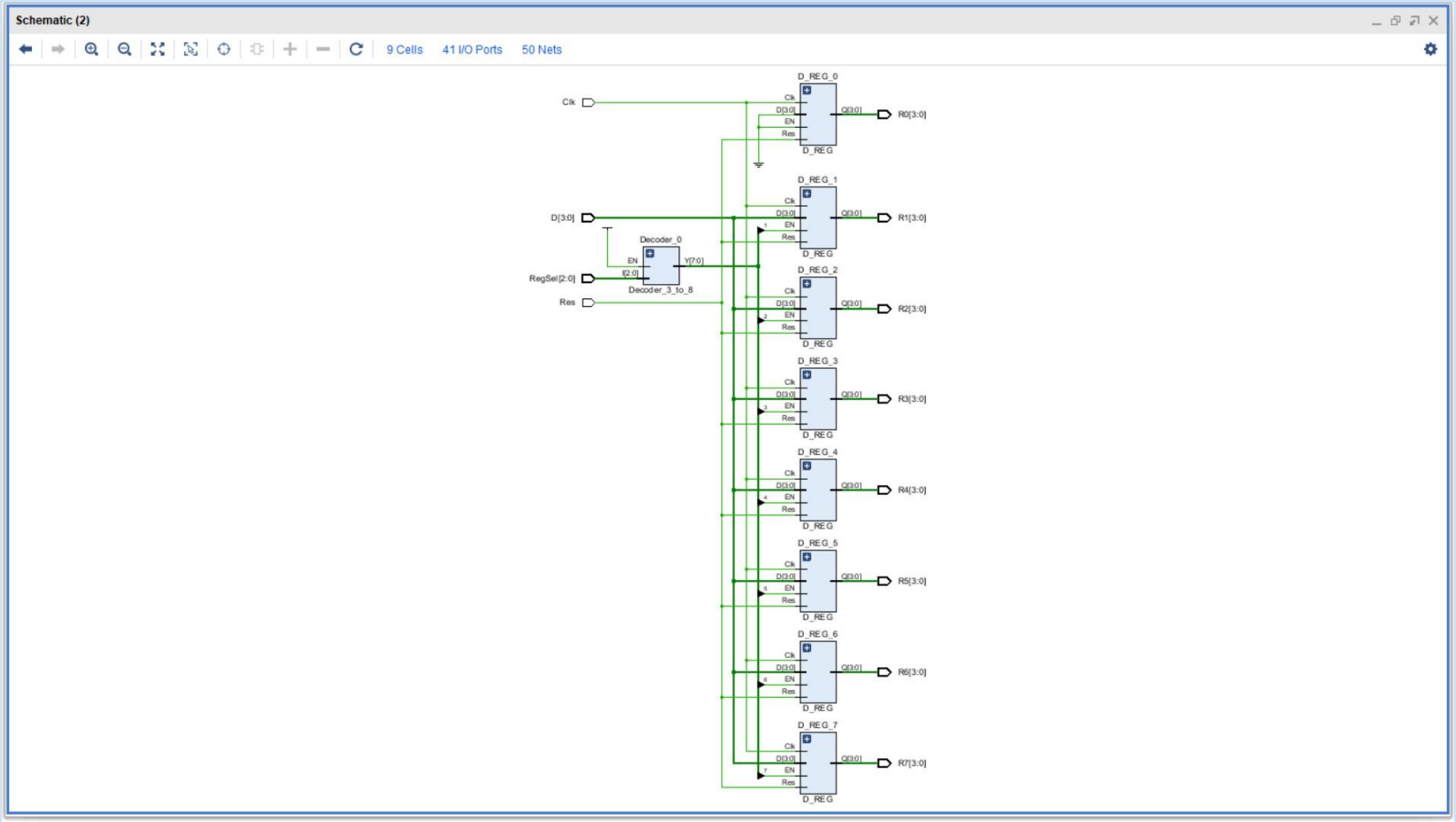
```
architecture Behavioral of Mux_2_way_4_bit is

component Mux_2_way_3_bit
    Port ( R0 : in STD_LOGIC_VECTOR (2 downto 0);
           R1 : in STD_LOGIC_VECTOR (2 downto 0);
           S : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component Mux_2_to_1
    Port ( D : in STD_LOGIC_VECTOR (1 downto 0);
           S : in STD_LOGIC;
           Y : out STD_LOGIC);
end component;

begin

Mux_2_way_3_bit_0 : Mux_2_way_3_bit
    PORT MAP(
        R0 => R0(3 downto 1),
        R1 => R1(3 downto 1),
        S => LS,
        Y => Y(3 downto 1)

    );


Mux_2_to_1_0 : Mux_2_to_1
    PORT MAP(
        D(0) => R0(0),
        D(1) => R1(0),
        S => LS,
        Y => Y(0)
    );

end Behavioral;
```

Timing Diagram



Register Bank

Schematic



VHDL File

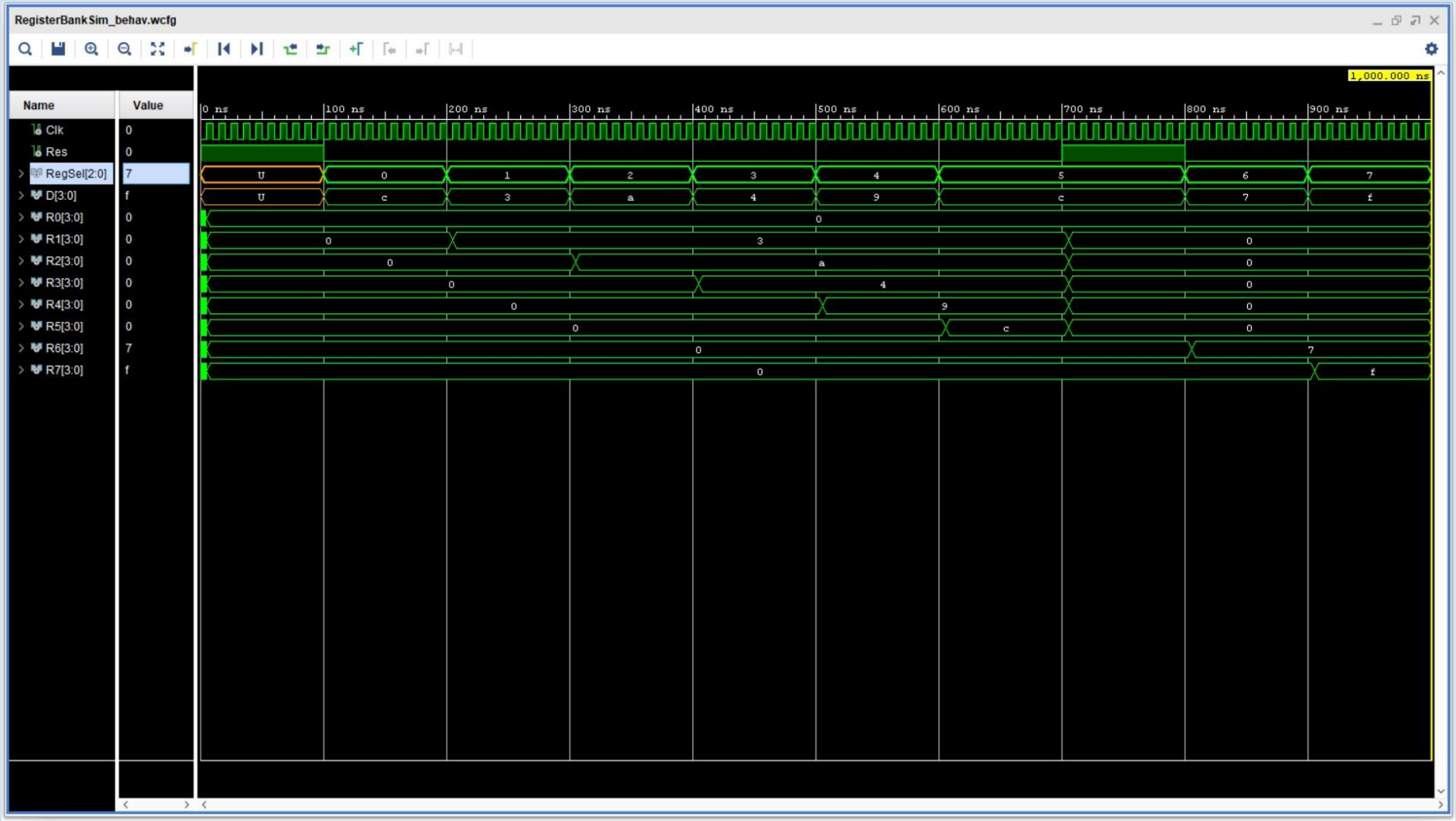
```
--
-- Create Date: 07/07/2022 03:28:47 PM
-- Design Name:
-- Module Name: RegisterBank - Behavioral
-- Project Name: Nanoprocessor Design
-- Target Devices:
-- Tool Versions:
-- Description: Group 44
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity RegisterBank is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           Clk : in STD_LOGIC;
           RegSel : in STD_LOGIC_VECTOR (2 downto 0);
           Res : in STD_LOGIC;
           R0 : out STD_LOGIC_VECTOR (3 downto 0);
           R1 : out STD_LOGIC_VECTOR (3 downto 0);
           R2 : out STD_LOGIC_VECTOR (3 downto 0);
           R3 : out STD_LOGIC_VECTOR (3 downto 0);
           R4 : out STD_LOGIC_VECTOR (3 downto 0);
           R5 : out STD_LOGIC_VECTOR (3 downto 0);
           R6 : out STD_LOGIC_VECTOR (3 downto 0);
           R7 : out STD_LOGIC_VECTOR (3 downto 0));
end RegisterBank;
```

```
architecture Behavioral of RegisterBank is

--component Slow_Clk
--Port ( Clk_in : in STD_LOGIC;
--       Clk_out : out STD_LOGIC);
--end component;

component D_REG
        Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
       EN : in STD_LOGIC;
       Clk : in STD_LOGIC;
       Q : out STD_LOGIC_VECTOR (3 downto 0);
       Res : in STD_LOGIC);
end component;

component Decoder_3_to_8
    Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
           EN : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR (7 downto 0));
end component;

signal Y_out : STD_LOGIC_VECTOR (7 downto 0);

begin


Decoder_0 : Decoder_3_to_8
    PORT MAP (
        I => RegSel,
        EN => '1',
        Y => Y_out


    );
```

```
D_REG_0 : D_REG
    PORT MAP (
        D => "0000",
        EN => '0',--read only Register
        Clk => Clk,
        Q => R0,
        Res => Res

    );
D_REG_1 : D_REG
    PORT MAP (
        D => D,
        EN => Y_out(1),
        Clk => Clk,
        Q => R1,
        Res => Res

    );
D_REG_2 : D_REG
    PORT MAP (
        D => D,
        EN => Y_out(2),
        Clk => Clk,
        Q => R2,
        Res => Res

    );

D_REG_3 : D_REG
    PORT MAP (
        D => D,
        EN => Y_out(3),
        Clk => Clk,
        Q => R3,
        Res => Res

    );
```

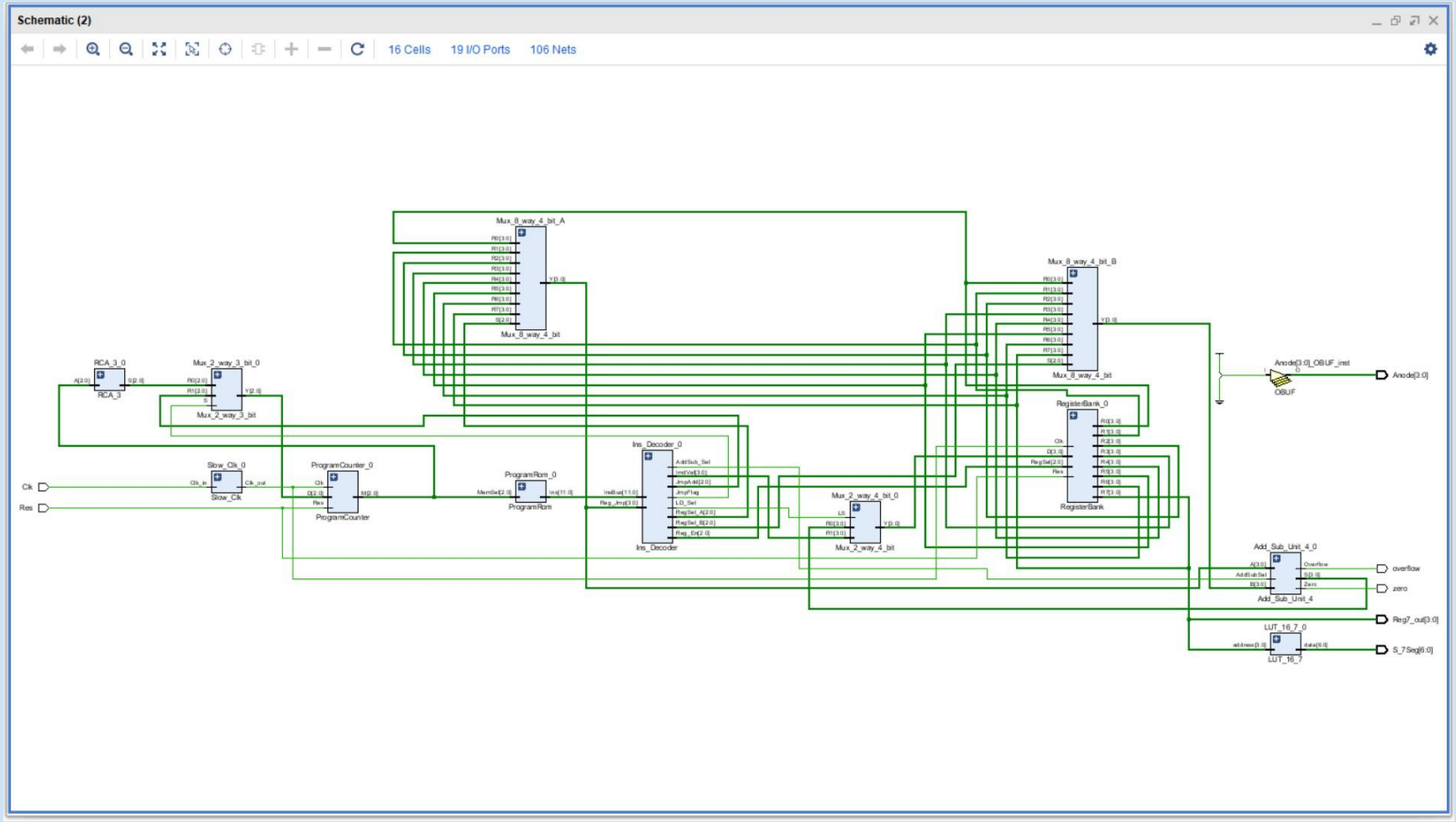
```
D_REG_4 : D_REG
    PORT MAP (
        D => D,
        EN => Y_out(4),
        Clk => Clk,
        Q => R4,
        Res => Res

    );

D_REG_5 : D_REG
        PORT MAP (
            D => D,
            EN => Y_out(5),
            Clk => Clk,
            Q => R5,
            Res => Res
        );
D_REG_6 : D_REG
    PORT MAP (
        D => D,
        EN => Y_out(6),
        Clk => Clk,
        Q => R6,
        Res => Res

    );

D_REG_7 : D_REG
    PORT MAP (
        D => D,
        EN => Y_out(7),
        Clk => Clk,
        Q => R7,
        Res => Res

    );

end Behavioral;
```

Timing Diagram



Nanoprocessor

Schematic



VHDL File

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 07/20/2022 04:42:18 PM
6  -- Design Name:
7  -- Module Name: Nanoprocessor - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description: Group44
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34 entity Nanoprocessor is
35     Port ( Clk : in STD_LOGIC;
36            Res : in STD_LOGIC;
37            Reg7_out : out STD_LOGIC_VECTOR (3 downto 0);
38            S_7Seg : out STD_LOGIC_VECTOR (6 downto 0);
39            Anode : out STD_LOGIC_VECTOR (3 downto 0);
40            overflow : out STD_LOGIC;
41            zero : out STD_LOGIC);
42 end Nanoprocessor;
43
```

```
44  architecture Behavioral of Nanoprocessor is
45
46  component Slow_Clk
47  Port ( Clk_in : in STD_LOGIC;
48         Clk_out : out STD_LOGIC);
49  end component;
50
51
52  component ProgramCounter
53  Port ( D : in STD_LOGIC_VECTOR (2 downto 0);
54         Clk : in STD_LOGIC;
55         Res : in STD_LOGIC;
56         M : out STD_LOGIC_VECTOR (2 downto 0));
57  end component;
58
59  component Mux_2_way_3_bit
60  Port ( R0 : in STD_LOGIC_VECTOR (2 downto 0);
61         R1 : in STD_LOGIC_VECTOR (2 downto 0);
62         S : in STD_LOGIC;
63         Y : out STD_LOGIC_VECTOR (2 downto 0));
64  end component;
65
66  component RCA_3
67  Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
68         S : out STD_LOGIC_VECTOR (2 downto 0));
69  end component;
70
71  component ProgramRom
72  Port ( Ins : out STD_LOGIC_VECTOR (11 downto 0);
73         MemSel : in STD_LOGIC_VECTOR (2 downto 0));
74  end component;
```

```
76  component Ins_Decoder
77  Port ( InsBus : in STD_LOGIC_VECTOR (11 downto 0);
78         Reg_Jmp : in STD_LOGIC_VECTOR (3 downto 0);
79         Reg_En : out STD_LOGIC_VECTOR (2 downto 0);
80         RegSel_A : out STD_LOGIC_VECTOR (2 downto 0);
81         RegSel_B : out STD_LOGIC_VECTOR (2 downto 0);
82         ImdVal : out STD_LOGIC_VECTOR (3 downto 0);
83         JmpAdd : out STD_LOGIC_VECTOR (2 downto 0);
84         JmpFlag : out STD_LOGIC;
85         LD_Sel : out STD_LOGIC;
86         AddSub_Sel : out STD_LOGIC);
87  end component;
88
89  component Mux_2_way_4_bit
90  Port ( R0 : in STD_LOGIC_VECTOR (3 downto 0);
91         R1 : in STD_LOGIC_VECTOR (3 downto 0);
92         LS : in STD_LOGIC;
93         Y : out STD_LOGIC_VECTOR (3 downto 0));
94  end component;
95
96  component RegisterBank
97  Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
98         Clk : in STD_LOGIC;
99         RegSel : in STD_LOGIC_VECTOR (2 downto 0);
100        Res : in STD_LOGIC;
101        R0 : out STD_LOGIC_VECTOR (3 downto 0);
102        R1 : out STD_LOGIC_VECTOR (3 downto 0);
103        R2 : out STD_LOGIC_VECTOR (3 downto 0);
104        R3 : out STD_LOGIC_VECTOR (3 downto 0);
105        R4 : out STD_LOGIC_VECTOR (3 downto 0);
106        R5 : out STD_LOGIC_VECTOR (3 downto 0);
107        R6 : out STD_LOGIC_VECTOR (3 downto 0);
108        R7 : out STD_LOGIC_VECTOR (3 downto 0));
109 end component;
```

```
component Mux_8_way_4_bit
Port ( R0 : in STD_LOGIC_VECTOR (3 downto 0);
       R1 : in STD_LOGIC_VECTOR (3 downto 0);
       R2 : in STD_LOGIC_VECTOR (3 downto 0);
       R3 : in STD_LOGIC_VECTOR (3 downto 0);
       R4 : in STD_LOGIC_VECTOR (3 downto 0);
       R5 : in STD_LOGIC_VECTOR (3 downto 0);
       R6 : in STD_LOGIC_VECTOR (3 downto 0);
       R7 : in STD_LOGIC_VECTOR (3 downto 0);
       S : in STD_LOGIC_VECTOR (2 downto 0);
       Y : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component Add_Sub_Unit_4
Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
       B : in STD_LOGIC_VECTOR (3 downto 0);
       AddSubSel : in STD_LOGIC;
       S : out STD_LOGIC_VECTOR (3 downto 0);
       Overflow : out STD_LOGIC;
       Zero : out STD_LOGIC);
end component;

component LUT_16_7
Port ( address : in STD_LOGIC_VECTOR (3 downto 0);
       data : out STD_LOGIC_VECTOR (6 downto 0));
end component;
```

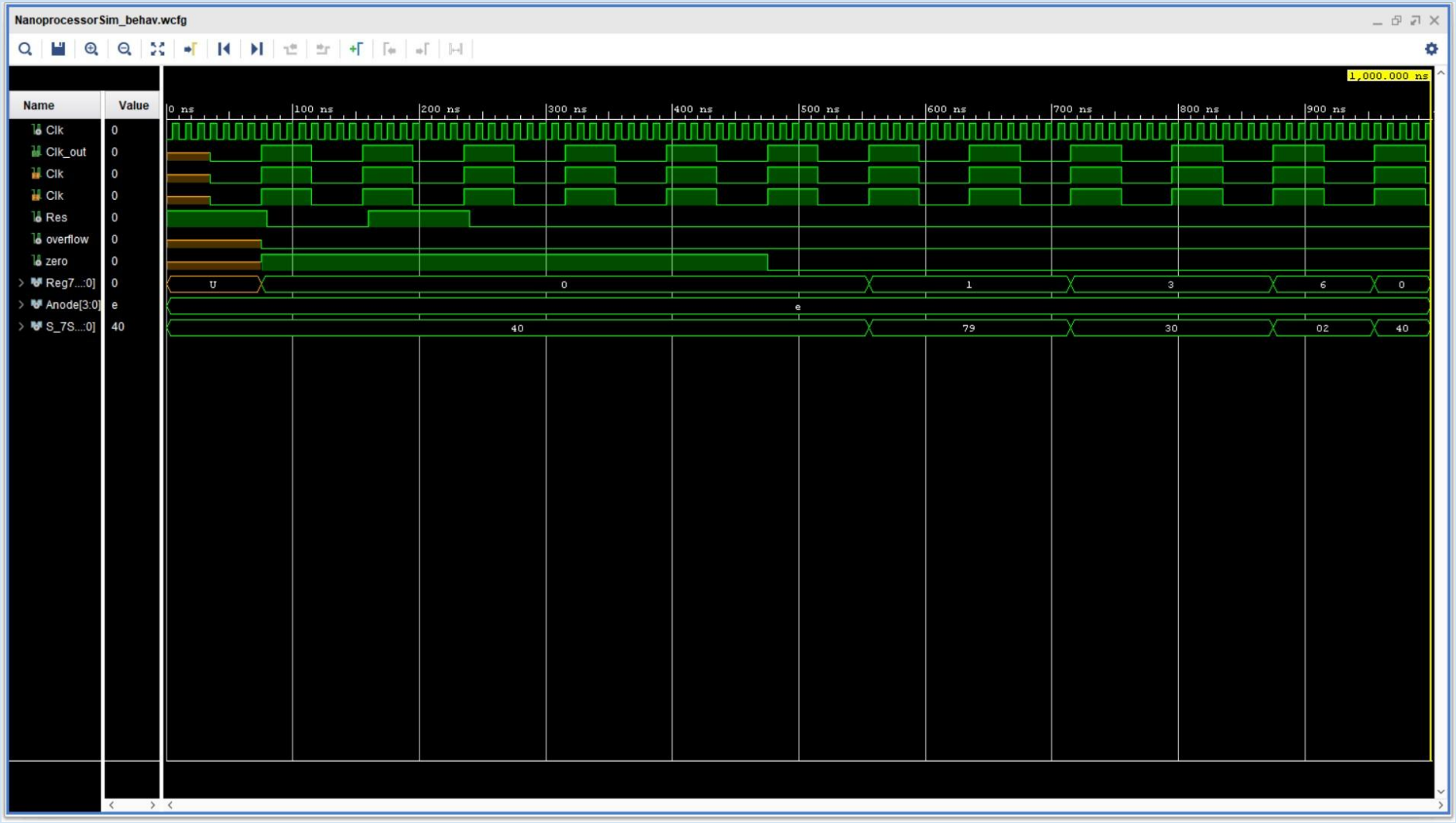
```
139
140  signal slow_Clk_out : STD_LOGIC;
141  signal RegSel_A_out,RegSel_B_out : STD_LOGIC_VECTOR (2 downto 0);
142  signal R0_out,R1_out,R2_out,R3_out,R4_out,R5_out,R6_out,R7_out : STD_LOGIC_VECTOR (3 downto 0);
143  signal M_out,RCA_3_out,Mux_out_3bit,Jmp_Add_out,Reg_En_out : STD_LOGIC_VECTOR (2 downto 0);
144  signal Jmp_flag_out,LD_Sel_out,AddSub_Sel_out : STD_LOGIC;
145  signal Ins_out : STD_LOGIC_VECTOR (11 downto 0);
146  signal Mux_out_4bit_8way_A,Mux_out_4bit_8way_B,Mux_out_4bit_2way,AddSub_out,ImdVal_out : STD_LOGIC_VECTOR (3 downto 0);
147
148  begin
149
150  Slow_Clk_0:Slow_Clk
151      PORT MAP(
152          Clk_in => Clk,
153          Clk_out => slow_Clk_out
154      );
155
156  ProgramCounter_0:ProgramCounter
157      PORT MAP(
158          D => Mux_out_3bit,
159          Clk => slow_Clk_out,
160          Res => Res,
161          M => M_out
162      );
163  Mux_2_way_3_bit_0:Mux_2_way_3_bit
164      PORT MAP(
165          R0 => RCA_3_out,
166          R1 => Jmp_Add_out,
167          S => Jmp_flag_out,
168          Y => Mux_out_3bit
169      );
170  RCA_3_0 :RCA_3
171      PORT MAP(
172          A => M_out,
173          S => RCA_3_out
174      );
175
176  ProgramRom_0:ProgramRom
177      PORT MAP(
178          Ins => Ins_out,
179          MemSel => M_out
180      );
```

```
181
182 Ins_Decoder_0:Ins_Decoder
183     PORT MAP(
184         InsBus => Ins_out,
185         Reg_Jmp => Mux_out_4bit_8way_A,
186         JmpAdd => Jmp_Add_out,
187         JmpFlag => Jmp_flag_out,
188         ImdVal => ImdVal_out,
189         LD_Sel => LD_Sel_out,
190         Reg_En => Reg_En_out,
191         RegSel_A => RegSel_A_out,
192         RegSel_B => RegSel_B_out,
193         AddSub_Sel => AddSub_Sel_out
194     );
195 Mux_2_way_4_bit_0:Mux_2_way_4_bit
196     PORT MAP(
197         R0 => AddSub_out,
198         R1 => ImdVal_out,
199         LS => LD_Sel_out,
200         Y => Mux_out_4bit_2way
201     );
202 RegisterBank_0:RegisterBank
203     PORT MAP(
204         Clk => slow_Clk_out,
205         Res => Res,
206         D => Mux_out_4bit_2way,
207         RegSel => Reg_En_out,
208         R0 => R0_out,
209         R1 => R1_out,
210         R2 => R2_out,
211         R3 => R3_out,
212         R4 => R4_out,
213         R5 => R5_out,
214         R6 => R6_out,
215         R7 => R7_out
216     );
```

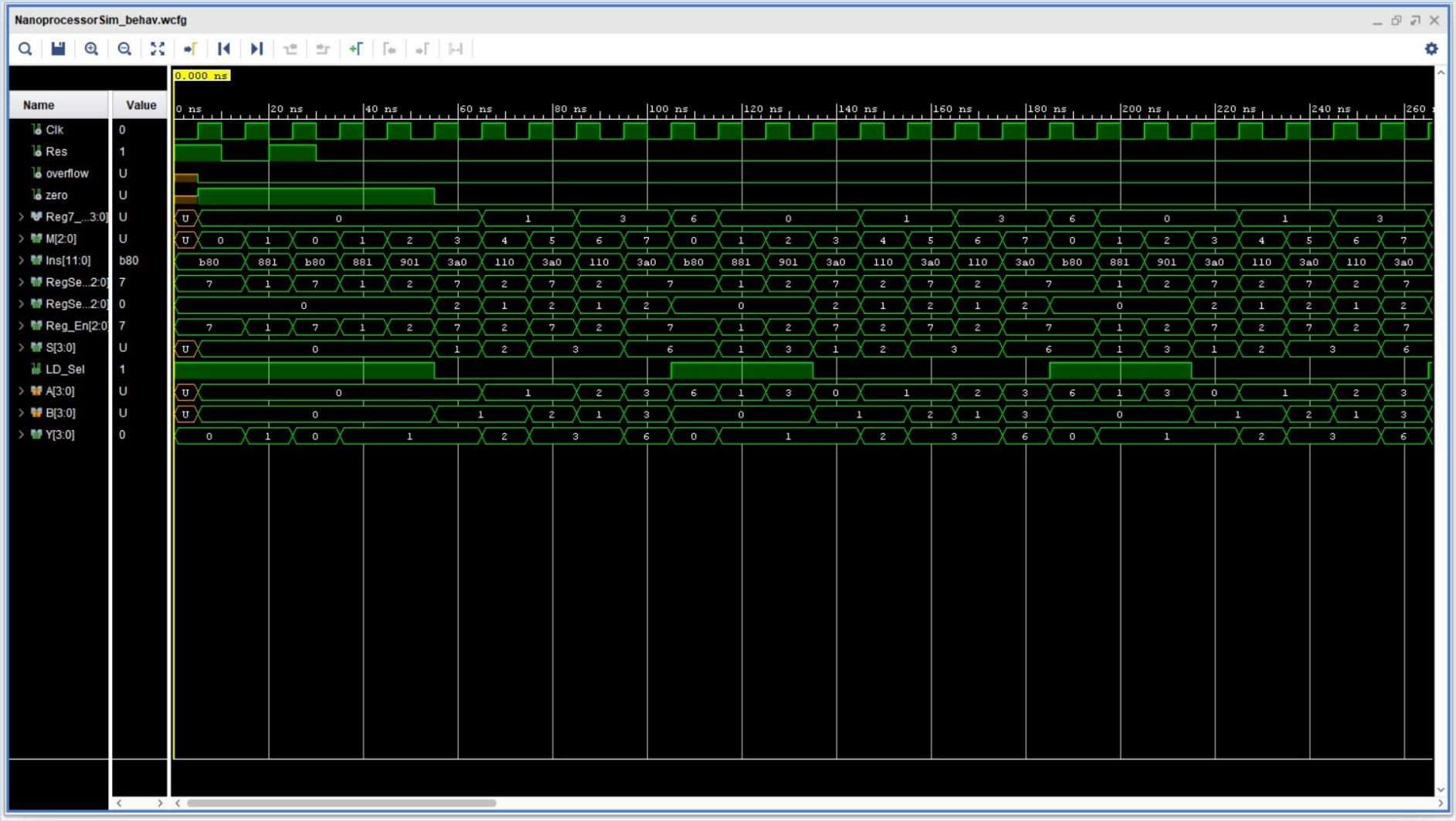
```
Mux_8_way_4_bit_A : Mux_8_way_4_bit
    PORT MAP(
        R0 => R0_out,
        R1 => R1_out,
        R2 => R2_out,
        R3 => R3_out,
        R4 => R4_out,
        R5 => R5_out,
        R6 => R6_out,
        R7 => R7_out,
        S => RegSel_A_out,
        Y => Mux_out_4bit_8way_A
    );
Mux_8_way_4_bit_B : Mux_8_way_4_bit
        PORT MAP(
            R0 => R0_out,
            R1 => R1_out,
            R2 => R2_out,
            R3 => R3_out,
            R4 => R4_out,
            R5 => R5_out,
            R6 => R6_out,
            R7 => R7_out,
            S => RegSel_B_out,
            Y => Mux_out_4bit_8way_B
        );
Add_Sub_Unit_4_0:Add_Sub_Unit_4
    PORT MAP(
        A => Mux_out_4bit_8way_A,
        B => Mux_out_4bit_8way_B,
        AddSubSel => AddSub_Sel_out,
        S => AddSub_out,
        overflow => overflow,
        zero => zero

    );

LUT_16_7_0:LUT_16_7
    PORT MAP(
        address => R7_out,
        data => S_7Seg
    );
```

```
261
262 Reg7_out <= R7_out;
263 Anode <= "1110";
264
265 end Behavioral;
266
```

Timing Diagram



Timing Diagram with Internal Signals (Used For Debugging)



Conclusion from the Lab

- More Instructions can be added by increasing the size of an instruction.
- Program Rom can be expanded to store larger programs with more instructions.
- “Jump if not zero” Instruction must be added in order to write assembly programs with conditional loops.
- This is a 4-bit architecture nanoprocessor; modern computers come with 64-bit architecture.
- We can write programs in assembly language directly to memory if we use an assembler to convert assembly language to relevant machine code of our processor architecture.
- Efficiency of the processor is determined by the maximum clock frequency the processor can support without any component of the machine being damaged.

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	36	0	20800	0.17
LUT as Logic	36	0	20800	0.17
LUT as Memory	0	0	9600	0.00
Slice Registers	49	0	41600	0.12
Register as Flip Flop	49	0	41600	0.12
Register as Latch	0	0	41600	0.00
F7 Muxes	0	0	16300	0.00
F8 Muxes	0	0	8150	0.00

* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt_design after synthesis, if not already completed, for a more realistic count.

7. Primitives

Ref Name	Used	Functional Category
FDRE	49	Flop & Latch
OBUF	17	IO
LUT4	17	LUT
LUT5	12	LUT
LUT6	9	LUT
CARRY4	8	CarryLogic
LUT3	4	LUT

IBUF	2	IO
LUT1	1	LUT
BUFG	1	Clock
+----------+------+---------------------+

Contribution of members to the project

A.S. JAYATHUNGA - 200265T

- Design and simulation of Instruction Decoder.

M.R.A.A.K. Gunasinghe - 200196G

- Design and Simulation of 3-bit Ripple Carry Adder.

K.A.Anshan Lahiru Kavinda - 200300A

- Design and Simulation of Register Bank.

K.K.A.J.S. Kumarasinghe - 200323V

- Design and Simulation of 8 way 4 bit Multiplexer.

W.M.T.B.Weerasekara - 200698X

- Design and Simulation of Add/Sub unit.

The rest of the components' designing, integration of components, testing, and debugging were done collaboratively with the participation of all the members of the group.