NanoProcessor Design

Group Number - 43

Group members:

P.Kobinarth (200307C)
T.Pairavi (200441F)
S.Nisanthan (200432E)
P.Sanujen (200583P)
Y.Sathveegan (200592R)

Introduction:

We designed a very simple microprocessor (hence, called a *nanoprocessor*) capable of executing a simple set of instructions. The block diagram of the nano processor is given in Fig. 1. As the microprocessor only understands machine language, we provide those instructions as a binary value. We will hard code our program to ROM. One of the push buttons should use to reset the PC and Register Bank (this enables us to restart the program at any time). We used the slow clock to drive our nano processor. Therefore, to be able to see the changes as our program executes reduce the clock rate such that it ticks every 2 or 3 seconds.

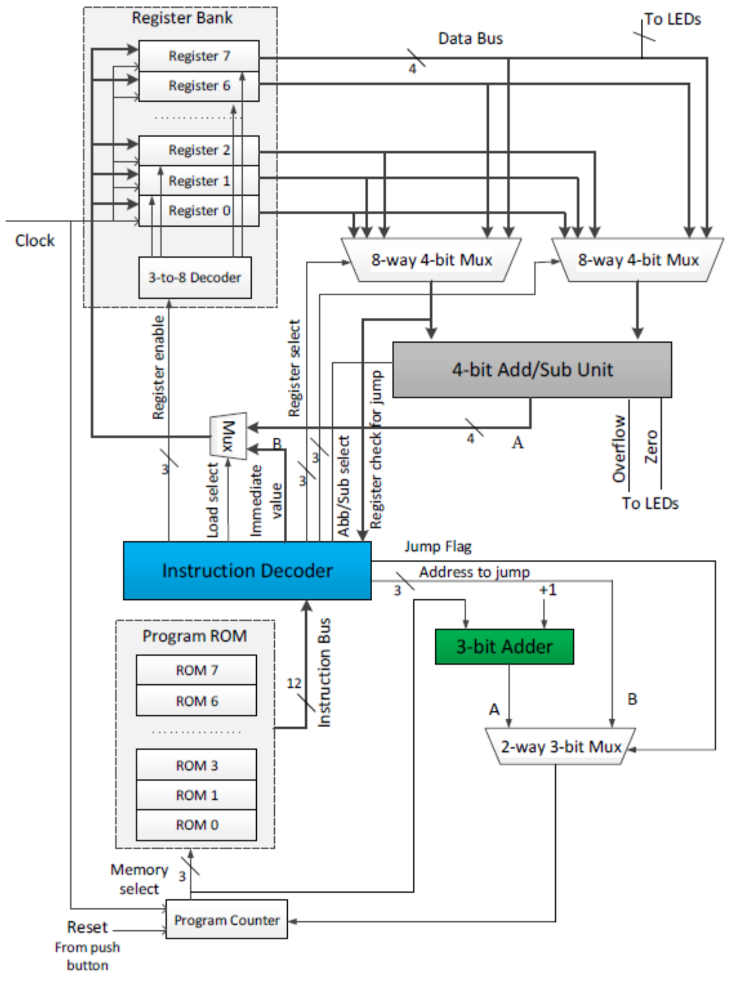


Figure 1 – High-level diagram of the nanoprocessor.

Purposes of this lab (Design a microprocessor):

- Design and develop a 4-bit arithmetic unit that can add and subtract signed integers.
- Decode instructions to activate necessary components on the processor.
- Design and develop k-way b-bit multiplexers or tri-state busses.
- Verify their functionality via simulation and on the development board.

Tasks:

- We will design a very simple nano processor capable of executing a simple set of instructions.
- We need to develop or extend several components.
    1. 4bit Add/Subtract unit
    2. 3-bit adder
    3. 3-bit Program Counter (PC)
    4. K-way b-bit multiplexer
    5. 4-bit register
    6. Register bank
    7. Program ROM
    8. Instruction Decoder
    9. Slow Clock
    10. LookUpTable 16 to 7
    11. NanoProcessor
    12. OurProcesssor(Processor with NanoProcessor, 7 segment display, and SlowClock)

Components

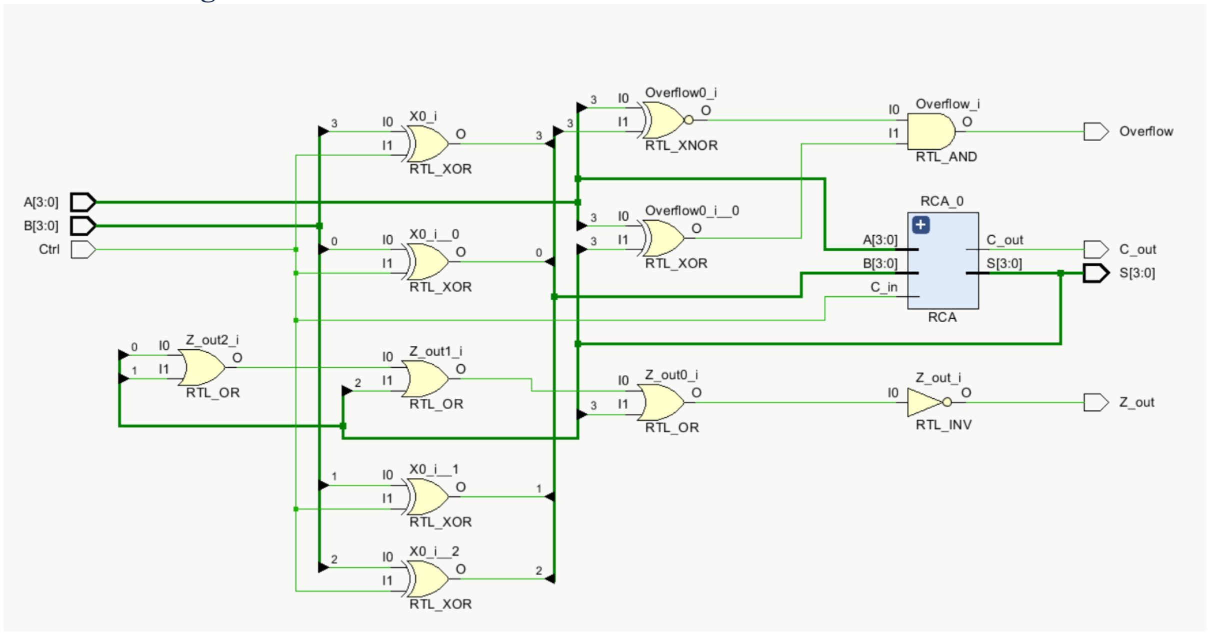
4bit Add/Subract unit

It is capable of adding and subtracting numbers represented using 2's complement. It has an overflow and zero flags.

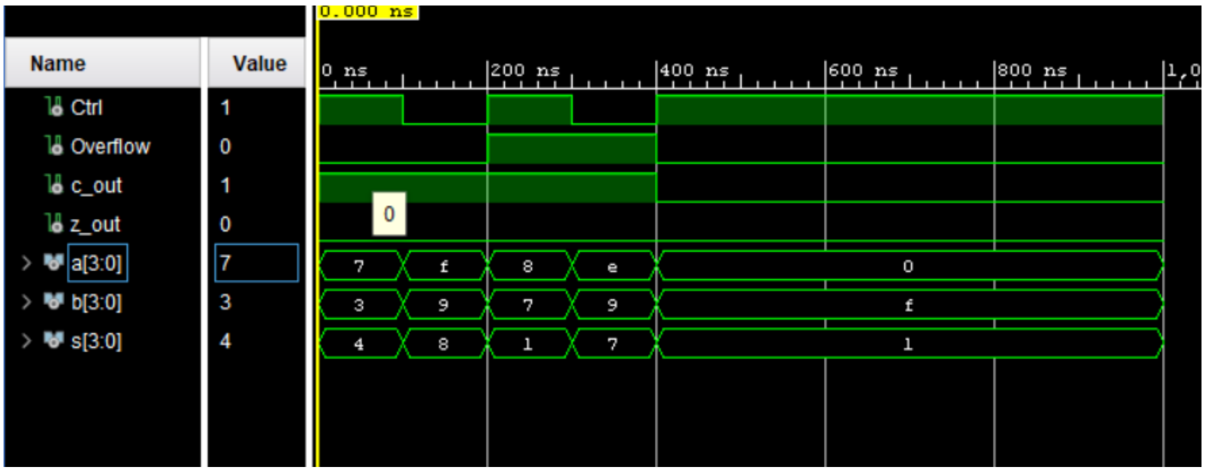
VHDL Code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/09/2022 10:40:02 PM
-- Design Name:
-- Module Name: Add_Sub - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- 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 is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Ctrl : in STD_LOGIC;
           C_out : out STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
           Overflow : out STD_LOGIC;
           Z_out : out STD_LOGIC
           );
end Add_Sub;

architecture Behavioral of Add_Sub is

component RCA
 Port (A : in STD_LOGIC_VECTOR (3 downto 0);
       B : in STD_LOGIC_VECTOR (3 downto 0);
       C_in : in STD_LOGIC;
       S : out STD_LOGIC_VECTOR (3 downto 0);
       C_out : out STD_LOGIC
       );
end component;
SIGNAL X,Y: STD_LOGIC_VECTOR (3 downto 0);

begin
X(0) <= B(0) XOR Ctrl;
X(1) <= B(1) XOR Ctrl;
X(2) <= B(2) XOR Ctrl;
X(3) <= B(3) XOR Ctrl;
RCA_0 : RCA
    PORT MAP (
        A => A,
        B => X,
        C_in => Ctrl,
        S => Y,
        C_out => C_out
    );
Overflow <= (A(3) XNOR (Ctrl XOR B(3))) AND (A(3) XOR Y(3));
S <= Y;
Z_Out <= NOT( Y(0) OR Y(1) OR Y(2) OR Y(3));
end Behavioral;
```

Schematic Diagram



Timing Diagram



3-bit Adder

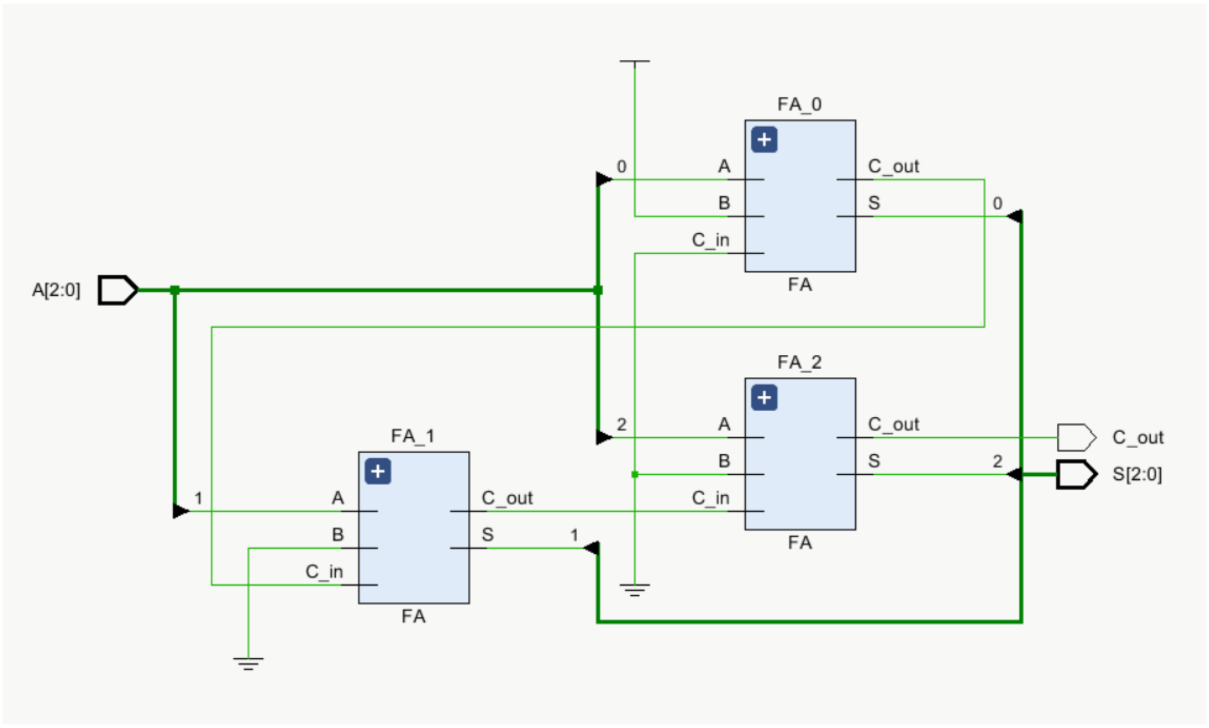
This unit is used to increment the Program Counter

VHDL Code

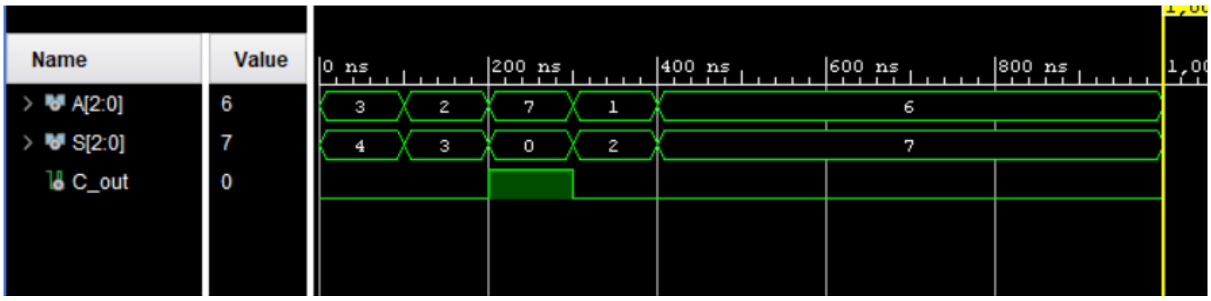
```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/09/2022 11:03:50 PM
-- Design Name:
-- Module Name: Adder_3bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- 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 Adder_3bit is
    Port (
        A : in STD_LOGIC_VECTOR (2 downto 0);
        S: out STD_LOGIC_VECTOR (2 downto 0);
        C_out : out STD_LOGIC
        );
end Adder_3bit;

architecture Behavioral of Adder_3bit 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;
SIGNAL FA0_S, FA0_C, FA1_S, FA1_C, FA2_S, FA2_C, FA3_S, FA3_C : std_logic;

begin
```

```
FA_0 : FA
    port map (
         A => A(0),
         B => '1',
         C_in => '0',
         S => S(0),
         C_Out => FA0_C);
FA_1 : FA
    port map (
        A => A(1),
        B => '0',
        C_in => FA0_C,
        S => S(1),
        C_Out => FA1_C);
FA_2 : FA
    port map (
        A => A(2),
        B => '0',
        C_in => FA1_C,
        S => S(2),
        C_Out => C_out);
end Behavioral;
```

Schematic Diagram



Timing Diagram



3-bit Program Counter

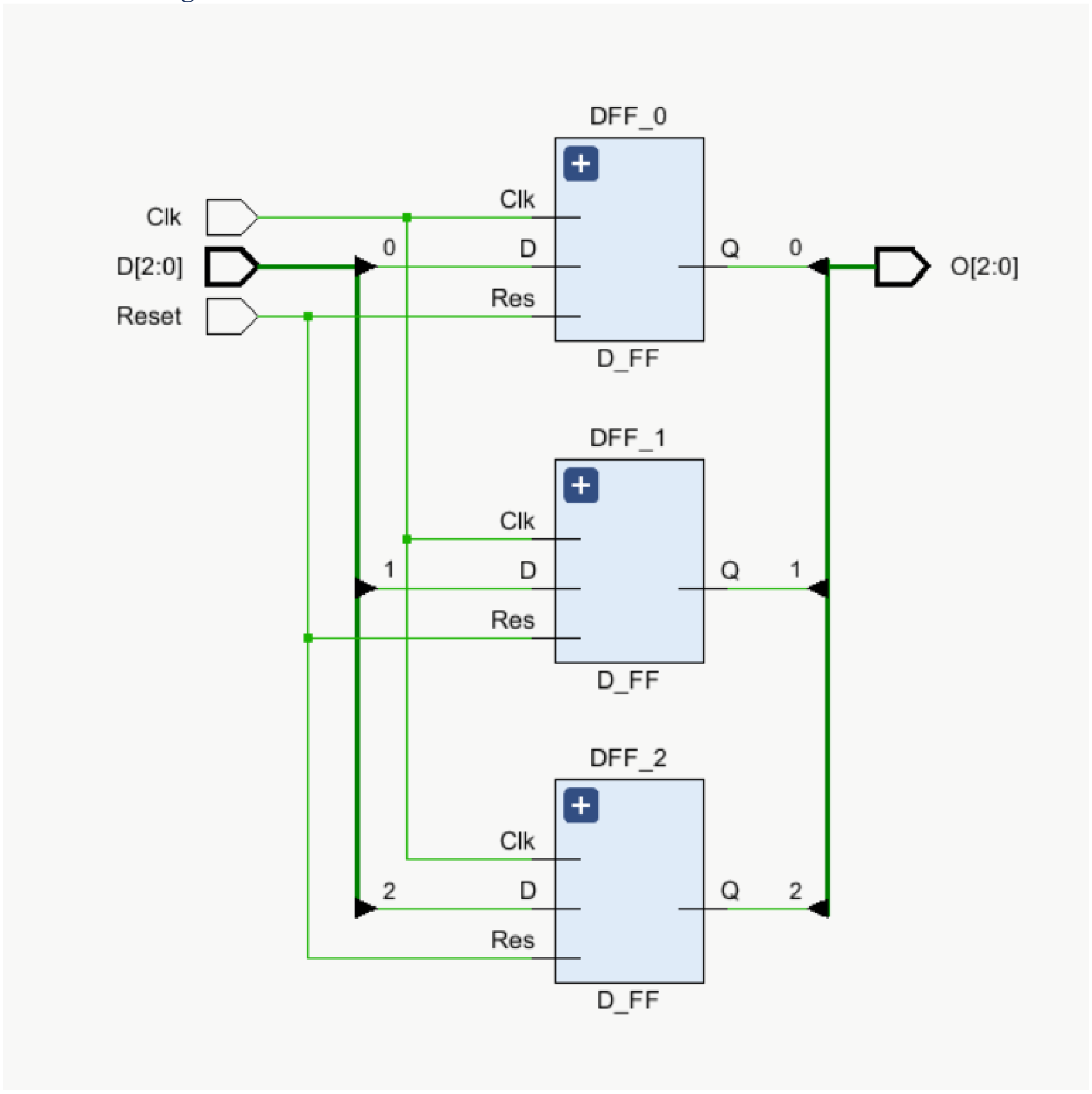
Address of the next instruction to be executed is stored here. It is type of a register.

VHDL Code

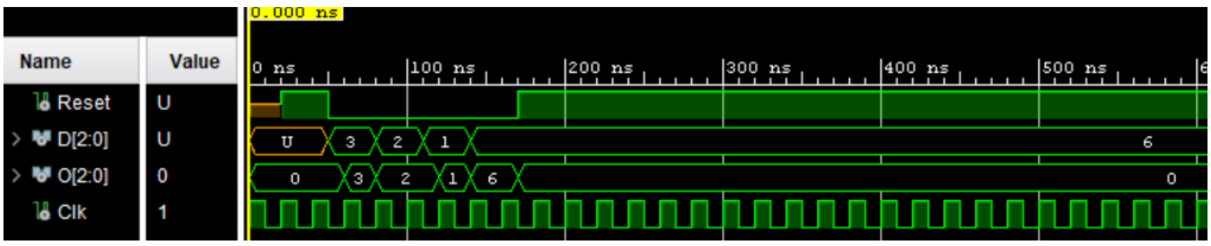
```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/10/2022 06:47:07 PM
-- Design Name:
-- Module Name: PC - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- 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 PC is
   Port ( Clk : in STD_LOGIC;
          Reset : in STD_LOGIC;
          D : in STD_LOGIC_VECTOR (2 downto 0);
          O : out STD_LOGIC_VECTOR (2 downto 0));
end PC;
```

```
architecture Behavioral of PC is
component D_FF
    Port ( D : in STD_LOGIC;
           Res : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Q : out STD_LOGIC
           );
end component;
signal Res_Sig : STD_LOGIC:= '1';
begin
DFF_0 : D_FF
    PORT MAP (
        D => D(0),
        Q => O(0),
        Clk => Clk,
        Res => Res_Sig
    );
DFF_1 : D_FF
    PORT MAP (
        D => D(1),
        Q => O(1),
        Clk => Clk,
        Res => Res_Sig
    );
DFF_2 : D_FF
    PORT MAP (
        D => D(2),
        Q => O(2),
        Clk => Clk,
        Res => Res_Sig
    );
Res_Sig <= Reset;
end Behavioral;
```

Schematic Diagram



Timing Diagram



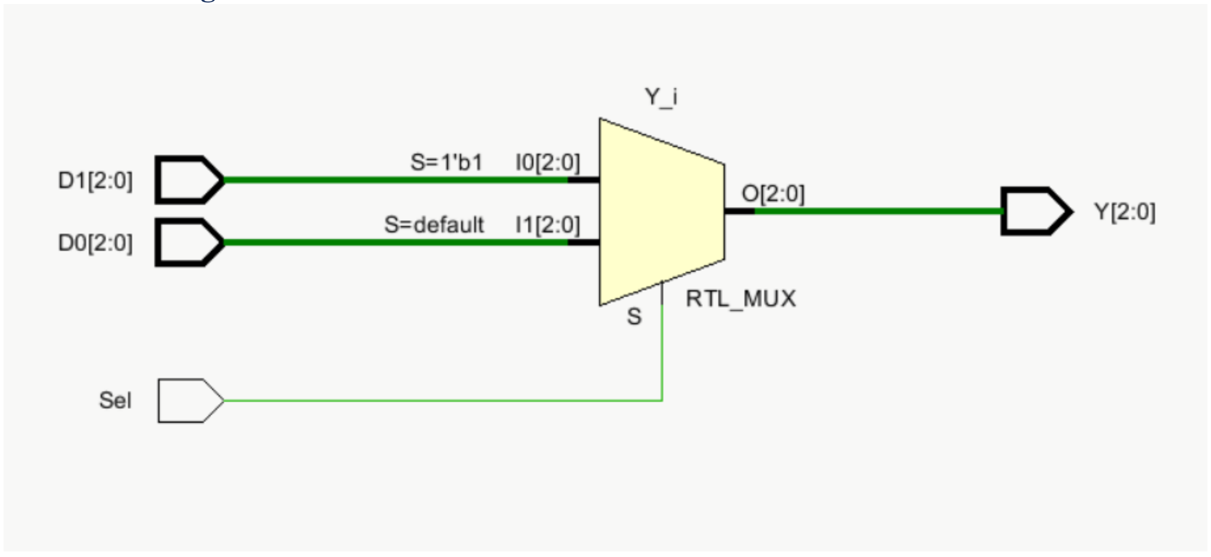
2-way-3bit Multiplexer

It can take in *2*-inputs, each with *3*-bits, rather than a single bit, and the output is a group of *3*-bits. There are 1 control bits, and these control bits are used to select one of the *3* groups of *3* bits rather than a single bit.

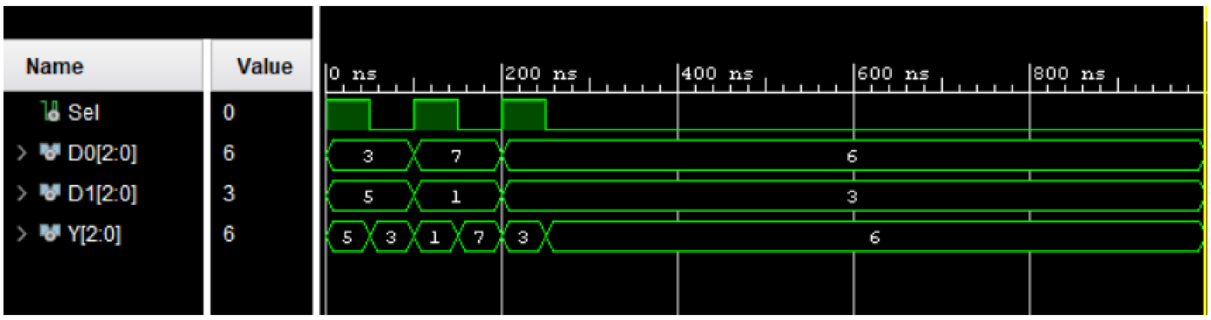
VHDL Code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/09/2022 11:46:26 PM
-- Design Name:
-- Module Name: Mux_2_to_1_3bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- 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_to_1_3bit is
    Port ( Sel : in STD_LOGIC;
           D0 : in STD_LOGIC_VECTOR (2 downto 0);
           D1 : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end Mux_2_to_1_3bit;

architecture Behavioral of Mux_2_to_1_3bit is

begin
     Y <= D1 when (Sel='1') else D0;


end Behavioral;
```

Schematic Diagram



Timing Diagram



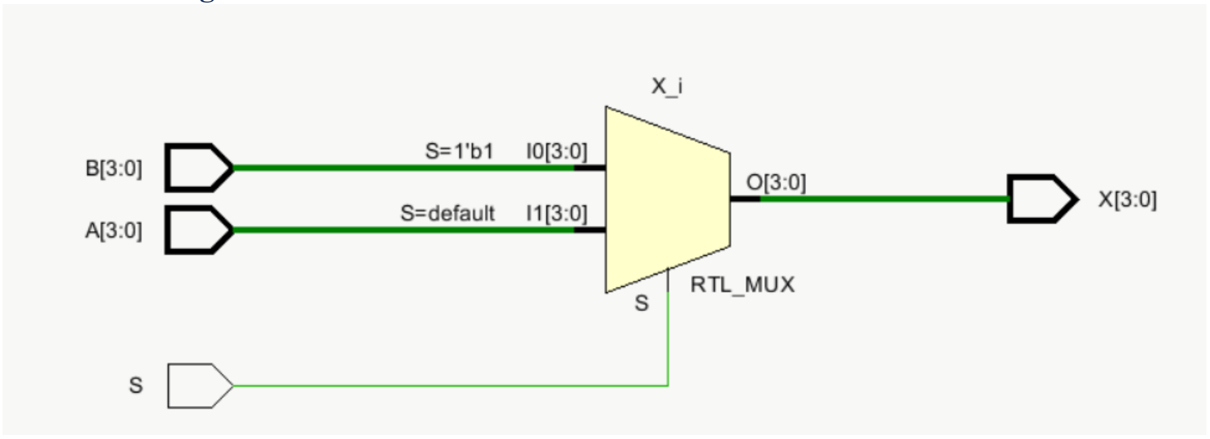
2-way-4bit Multiplexer

It can take in *2*-inputs, each with *4*-bits, rather than a single bit, and the output is a group of *4*-bits. There are 1 control bits, and these control bits are used to select one of the *2* groups of *4* bits rather than a single bit.

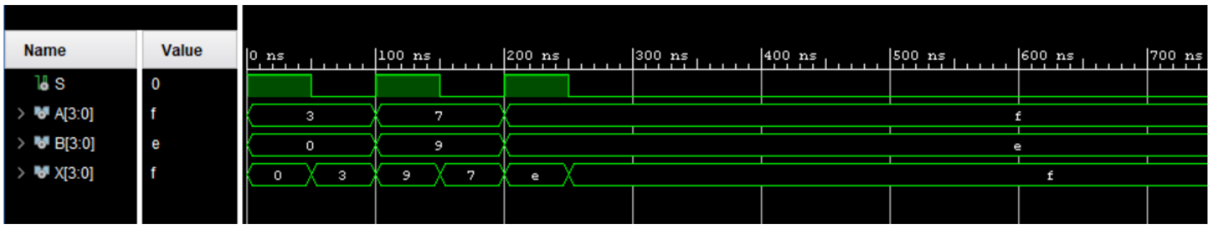
VHDL Code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/09/2022 11:44:45 PM
-- Design Name:
-- Module Name: Mux_2_to_1_4bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- 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_to_1_4bit is
    Port ( S : in STD_LOGIC;
           A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           X : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_2_to_1_4bit;

architecture Behavioral of Mux_2_to_1_4bit is

begin
     X <= B when (S='1') else A;

end Behavioral;
```

Schematic Diagram



Timing Diagram



8-way-4bit Multiplexer

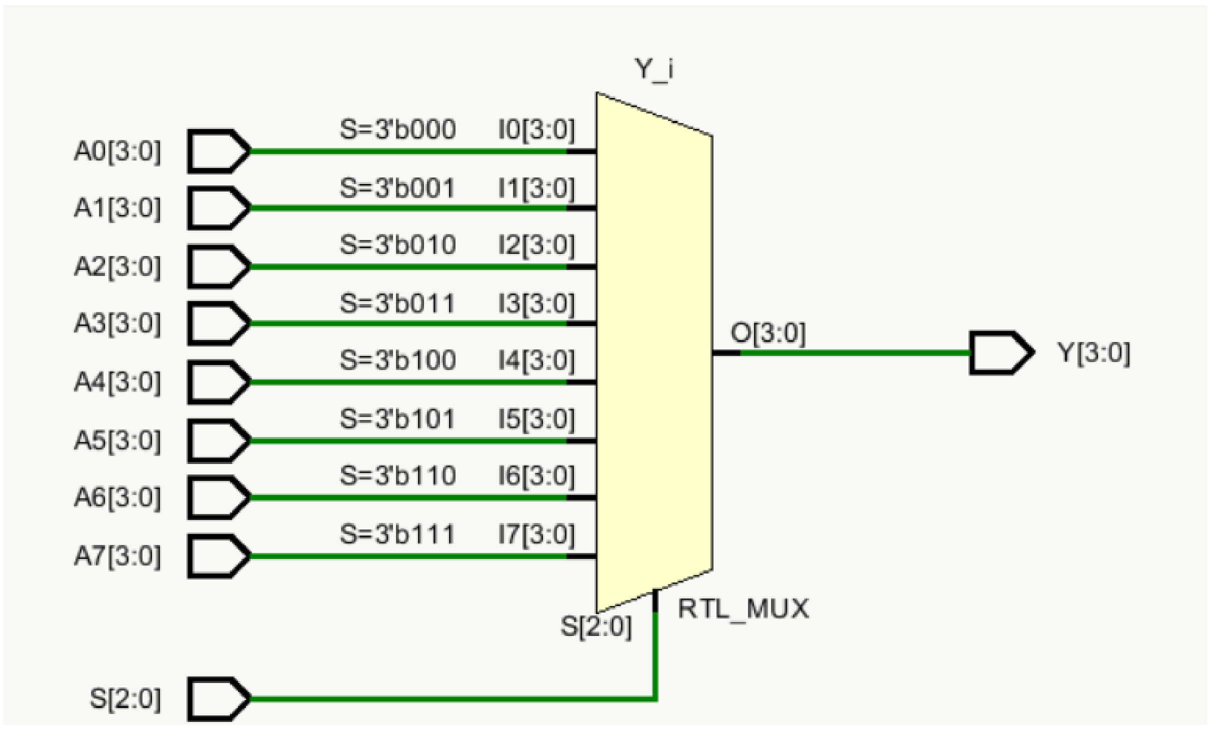
It can take in *8*-inputs, each with *4*-bits, rather than a single bit, and the output is a group of *4*-bits. There are 3 control bits, and these control bits are used to select one of the *8* groups of *4* bits rather than a single bit.

VHDL Code

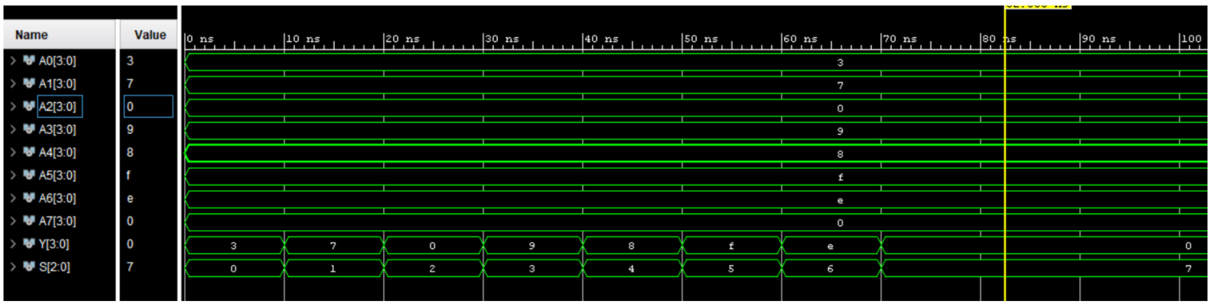
```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/10/2022 10:57:01 AM
-- Design Name:
-- Module Name: Mux_8_to_1 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- 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_to_1_4bit is
   Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
          A0 : in STD_LOGIC_VECTOR (3 downto 0);
          A1 : in STD_LOGIC_VECTOR (3 downto 0);
          A2 : in STD_LOGIC_VECTOR (3 downto 0);
          A3 : in STD_LOGIC_VECTOR (3 downto 0);
          A4 : in STD_LOGIC_VECTOR (3 downto 0);
          A5 : in STD_LOGIC_VECTOR (3 downto 0);
          A6 : in STD_LOGIC_VECTOR (3 downto 0);
          A7 : in STD_LOGIC_VECTOR (3 downto 0);
          Y : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_8_to_1_4bit;

architecture Behavioral of Mux_8_to_1_4bit is

begin
```

```
    process(A0,A1,A2,A3,A4,A5,A6,A7,S)
    begin
    case S is
    when "000" => Y <= A0;
    when "001" => Y <= A1;
    when "010" => Y <= A2;
    when "011" => Y <= A3;
    when "100" => Y <= A4;
    when "101" => Y <= A5;
    when "110" => Y <= A6;
    when "111" => Y <= A7;
    when others => NULL;
    end case;
    end process;

end Behavioral;
```

Schematic Diagram



Timing Diagram



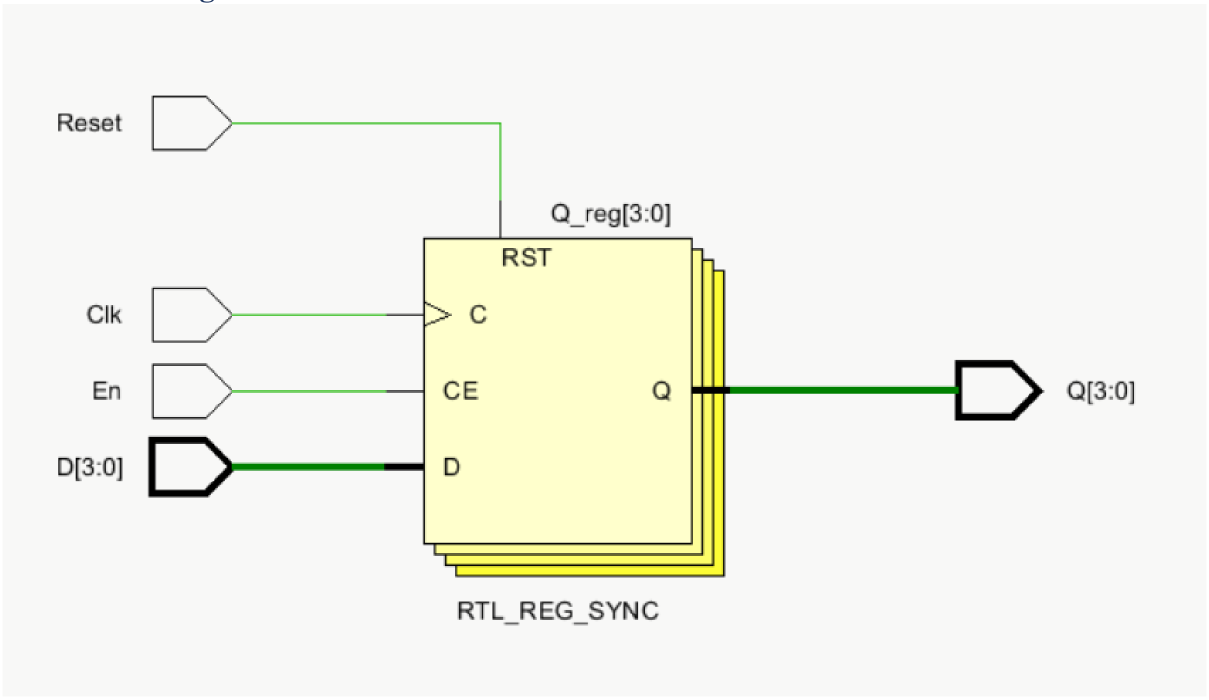
4-bit Register

It can hold a 4bit data

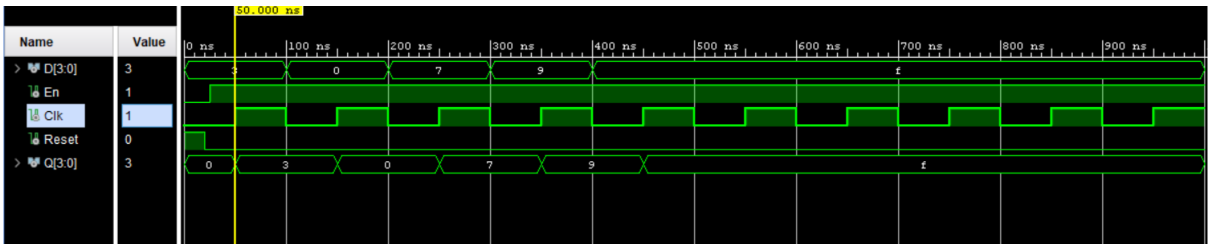
VHDL Code

```vhdl
----------------------------------------------------------------------------------
-- Company: 
-- Engineer: 
-- 
-- Create Date: 07/09/2022 10:54:25 PM
-- Design Name: 
-- Module Name: Register_4bit - Behavioral
-- Project Name: 
-- Target Devices: 
-- Tool Versions: 
-- Description: 
-- 
-- 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 Register_4bit is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           En : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (3 downto 0));
end Register_4bit;

architecture Behavioral of Register_4bit is

begin
    process (Clk,Reset)
    begin
        if Reset = '1' then
            Q <= "0000";
        elsif (rising_edge(Clk)) then
            if En = '1' then
                Q <= D;
            end if;
        end if;
    end process;

end Behavioral;
```

Schematic Diagram



Timing Diagram



Register Bank

It contains 4, 8-bit registers with a 3 to 8 decoder to enable the registers.

VHDL Code

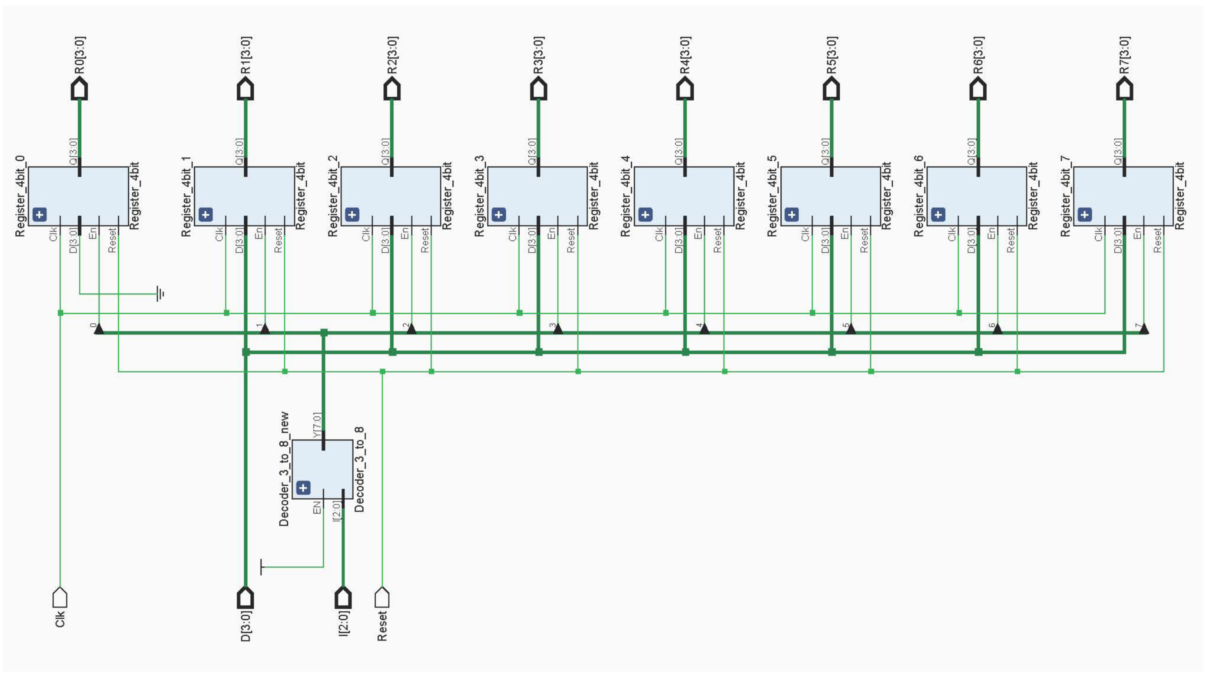
```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/09/2022 11:22:25 PM
-- Design Name:
-- Module Name: Register_Bank - Behavioral
-- Project Name:
-- Target Devices:
```

```
-- Tool Versions:
-- Description:
--
-- 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 Register_Bank is
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           D : in STD_LOGIC_VECTOR (3 downto 0);
           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);
           I : in STD_LOGIC_VECTOR (2 downto 0));
end Register_Bank;

architecture Behavioral of Register_Bank is

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;

component Register_4bit
    PORT ( D : in STD_LOGIC_VECTOR (3 downto 0);
           En : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (3 downto 0) );
end component;

signal EN,Reg_S : STD_LOGIC:='1';
signal Y : STD_LOGIC_VECTOR(7 downto 0);

begin
Decoder_3_to_8_new:Decoder_3_to_8
```

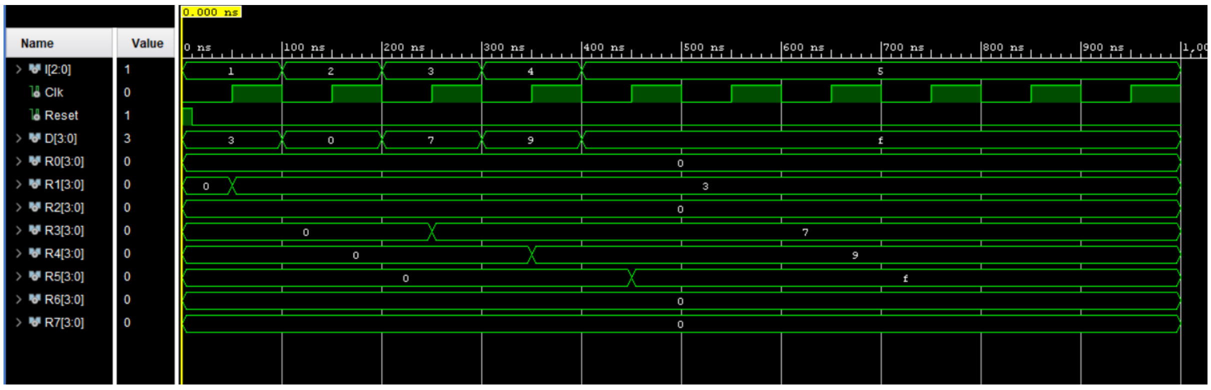
```
 PORT MAP (
       I=>I,
       EN=>EN,
       Y=>Y
       );

Register_4bit_0 : Register_4bit
 PORT MAP(
    D => "0000",
    En => Y(0),
    Clk => Clk,
    Reset => Reg_S,
    Q => R0
    );

Register_4bit_1 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(1),
    Clk => Clk,
    Reset => Reg_S,
    Q => R1
    );

Register_4bit_2 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(2),
    Clk => Clk,
    Reset => Reg_S,
    Q => R2
    );

Register_4bit_3 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(3),
    Clk => Clk,
    Reset => Reg_S,
    Q => R3
    );

Register_4bit_4 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(4),
    Clk => Clk,
    Reset => Reg_S,
    Q => R4
    );

Register_4bit_5 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(5),
    Clk => Clk,
    Reset => Reg_S,
    Q => R5
    );
```

```
Register_4bit_6 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(6),
    Clk => Clk,
    Reset => Reg_S,
    Q => R6
    );

Register_4bit_7 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(7),
    Clk => Clk,
    Reset => Reg_S,
    Q => R7
    );
Reg_S <= Reset;
end Behavioral;
```

Schematic Diagram



Timing Diagram



Program ROM

This is a ROM (Read Only Memory) which has programmed with certain instructions to run the whole processor as needed. The instructions will be transferred to Instruction Decoder to decode.

Assembly code:

```
0 => MOVI R1, 3;
1 => MOVI R2, 1;
2 => NEG R2;
3 => ADD R7, R1;
4 => ADD R1, R2;
5 => JZR R1, 7;
6 => JZR R0, 3;
7 => JZR R0, 7;
```

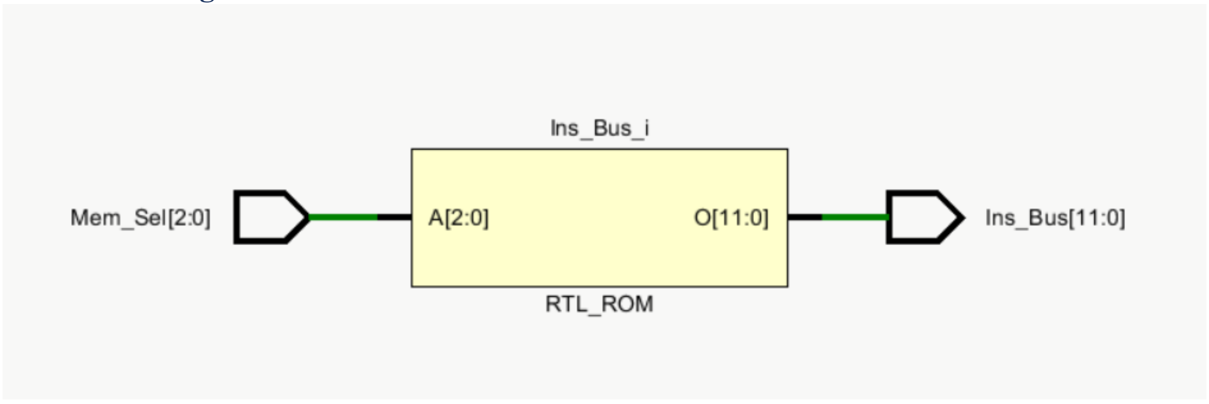
Machine Code

ROM			Instruction											
S2	**S1**	**S0**	**I (11)**	**I (10)**	**I (9)**	**I (8)**	**I (7)**	**I (6)**	**I (5)**	**I (4)**	**I (3)**	**I (2)**	**I (1)**	**I (0)**
0	0	0	1	0	0	0	1	0	0	0	0	0	1	1
0	0	1	1	0	0	1	0	0	0	0	0	0	0	1
0	1	0	0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	1	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	1	0	0	0	0	0
1	0	1	1	1	0	0	1	0	0	0	0	1	1	1
1	1	0	1	1	0	0	0	0	0	0	0	0	1	1
1	1	1	1	1	0	0	0	0	0	0	0	1	1	1

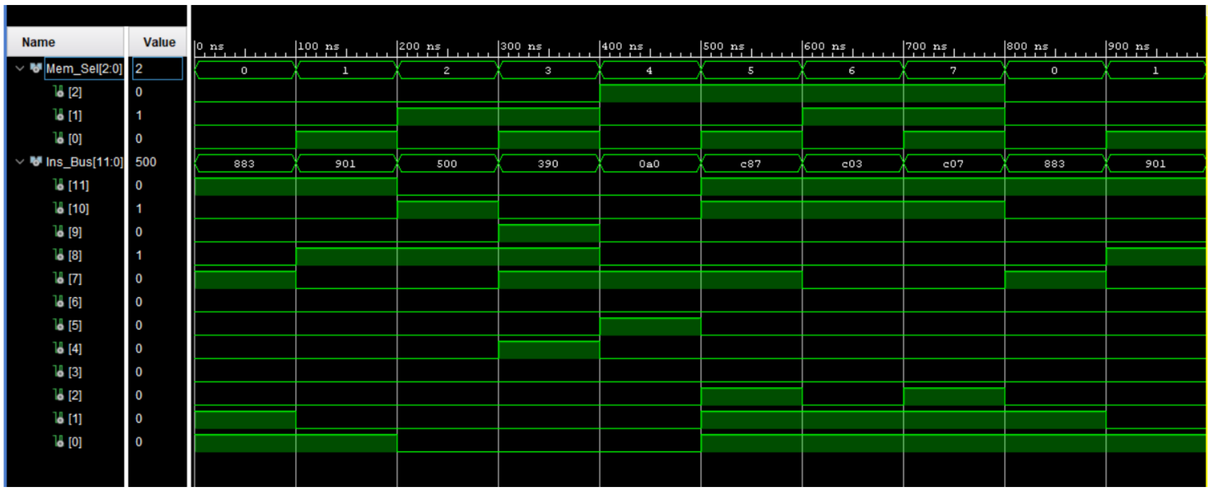
VHDL Code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/10/2022 09:19:40 AM
-- Design Name:
-- Module Name: Program_ROM - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- 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 Program_ROM is
    Port ( Mem_Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Ins_Bus : out STD_LOGIC_VECTOR (11 downto 0));
end Program_ROM;

architecture Behavioral of Program_ROM is
type rom_type is array (0 to 7) of STD_LOGIC_VECTOR(11 downto 0);
    signal sevenSegment_ROM : rom_type := (
        "100010000011",--Move R1, 3
        "100100000001",--Move R2, 1
        "010100000000",-- Neg R2
        "001110010000",--Add R7, R1
        "000010100000",--Add R1, R2
        "110010000111",--JZR R1, 7
        "110000000011",--JZR R0, 3
        "110000000111"


    );
begin
Ins_Bus <= sevenSegment_ROM(to_integer(unsigned(Mem_Sel)));
end Behavioral;
```

Schematic Diagram



Timing Diagram



Instruction Decoder

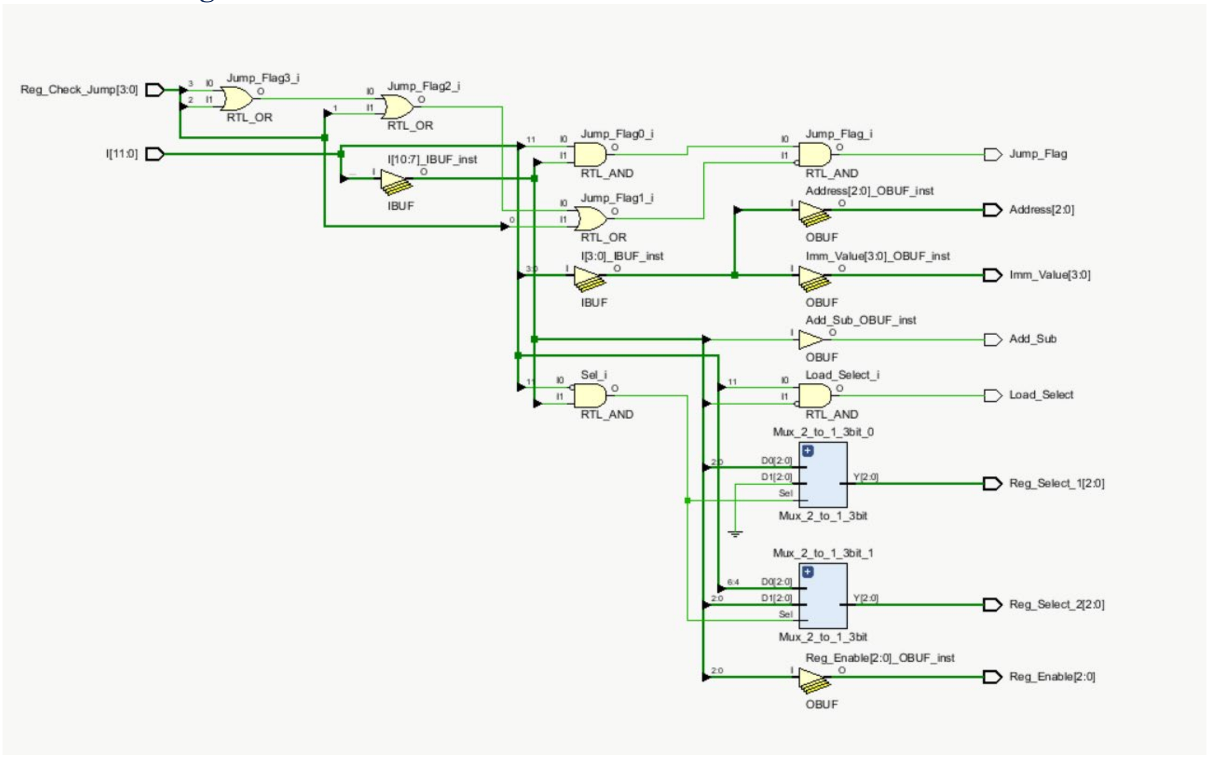
It activates the necessary components based on the instruction we wished to execute.

VHDL Code

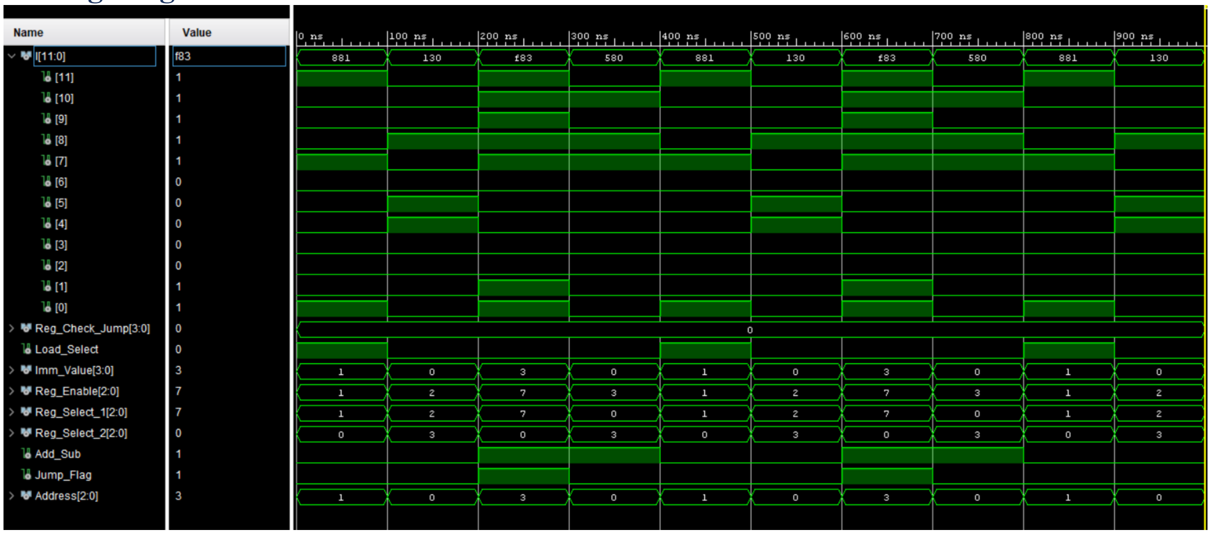
```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/10/2022 09:23:35 PM
-- Design Name:
-- Module Name: Instruction_Decoder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
```

```vhdl
-- 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 Instruction_Decoder is
    Port ( I : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
           Load_Select : out STD_LOGIC;
           Imm_Value : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_1 : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_2 : out STD_LOGIC_VECTOR (2 downto 0);
           Add_Sub : out STD_LOGIC;
           Jump_Flag : out STD_LOGIC;
           Address : out STD_LOGIC_VECTOR (2 downto 0)
           );
end Instruction_Decoder;

architecture Behavioral of Instruction_Decoder is
component Mux_2_to_1_3bit
    Port ( Sel : in STD_LOGIC;
           D0 : in STD_LOGIC_VECTOR (2 downto 0);
           D1 : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end component;
signal Ins : STD_LOGIC_VECTOR (1 downto 0);
signal RegA : STD_LOGIC_VECTOR (2 downto 0);
signal RegB : STD_LOGIC_VECTOR (2 downto 0);
signal Data : STD_LOGIC_VECTOR (3 downto 0);
signal Sel : STD_LOGIC;
begin
Ins <= I(11 downto 10);
RegA <= I(9 downto 7);
RegB <= I(6 downto 4);
Data <= I(3 downto 0);
Mux_2_to_1_3bit_0 : Mux_2_to_1_3bit
    Port map(
      Sel => Sel,
      D0 => RegA,
      D1 => "000",
      Y => Reg_Select_1
    );
Mux_2_to_1_3bit_1 : Mux_2_to_1_3bit
    Port map(
      Sel => Sel,
      D0 => RegB,
      D1 => RegA,
        Y => Reg_Select_2
    );
Sel <= NOT(Ins(1)) AND Ins(0);
Load_Select <= Ins(1) AND NOT (Ins(0)) ;
Add_Sub <= Ins(0);
Jump_Flag  <=  Ins(1)  AND  Ins(0)  AND  NOT(  Reg_Check_Jump(3)  OR  Reg_Check_Jump(2)  OR
Reg_Check_Jump(1)OR Reg_Check_Jump(0));
Reg_Enable <= RegA;
Imm_Value <= Data;
Address <= Data(2 downto 0);

end Behavioral;
```

Schematic Diagram



Timing Diagram



Name	Value
I[11:0]	f83
[11]	1
[10]	1
[9]	1
[8]	1
[7]	1
[6]	0
[5]	0
[4]	0
[3]	0
[2]	0
[1]	1
[0]	1
Reg_Check_Jump[3:0]	0
Load_Select	0
Imm_Value[3:0]	3
Reg_Enable[2:0]	7
Reg_Select_1[2:0]	7
Reg_Select_2[2:0]	0
Add_Sub	1
Jump_Flag	1
Address[2:0]	3

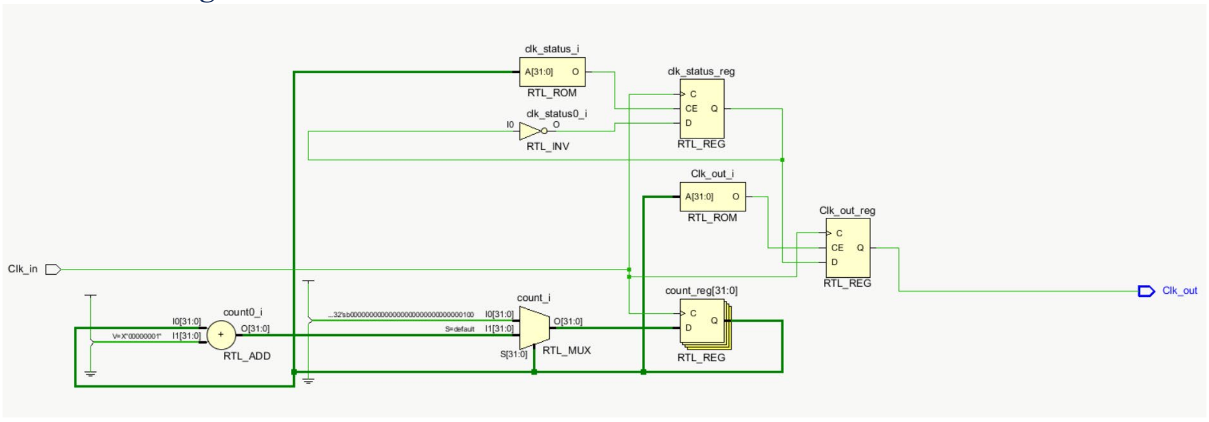
Slow Clock

It used to slow down the input clock.

VHDL Code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06/24/2022 12:19:06 PM
-- Design Name:
-- Module Name: Slow_Clk - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- 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 Slow_Clk is
    Port ( Clk_in : in STD_LOGIC;
           Clk_out : out STD_LOGIC);
end Slow_Clk;

architecture Behavioral of Slow_Clk is

signal count : integer := 1;
signal clk_status : std_logic := '0';

begin

    process (Clk_in) begin
        if (rising_edge(Clk_in)) then
            count <= count + 1; -
            if (count = 100000000)then
                clk_status <= not clk_status;
                Clk_out <= clk_status;
                count <= 1;
            end if;
        end if;
    end process;

end Behavioral;
```

Schematic Diagram



LUT 16 to 7

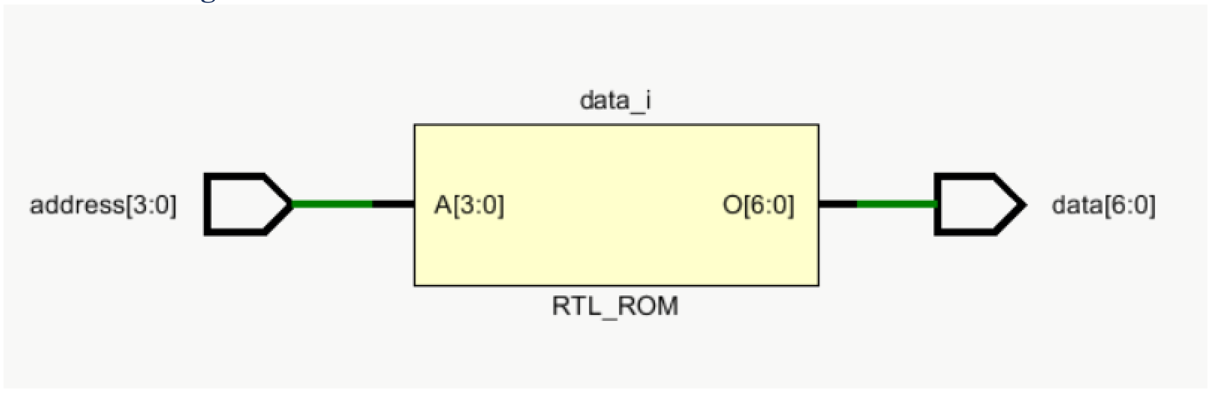
7-segment Display is the component that shows the output in a LED screen on a BASYS-3 board. The output from the processor is mapped to this component as input and the stored data in an inbuilt ROM for the current input address will be the output from the display through LEDs.

VHDL Code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06/25/2022 02:04:46 PM
-- Design Name:
-- Module Name: LUT_16_7 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- 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 LUT_16_7 is
   Port ( address : in STD_LOGIC_VECTOR (3 downto 0);
          data : out STD_LOGIC_VECTOR (6 downto 0));
end LUT_16_7;

architecture Behavioral of LUT_16_7 is
type rom_type is array (0 to 15) of std_logic_vector(6 downto 0);
signal sevenSegment_ROM : rom_type := (
"1000000", -- 0
"1111001", -- 1
"0100100", -- 2
"0110000", -- 3
"0011001", -- 4
"0010010", -- 5
"0000010", -- 6
"1111000", -- 7
"0000000", -- 8
"0010000", -- 9
"0001000", -- a
"0000011", -- b
"1000110", -- c
"0100001", -- d
"0000110", -- e
"0001110" -- f
);
begin
data <= sevenSegment_ROM(to_integer(unsigned(address)));

end Behavioral;
```

Schematic Diagram



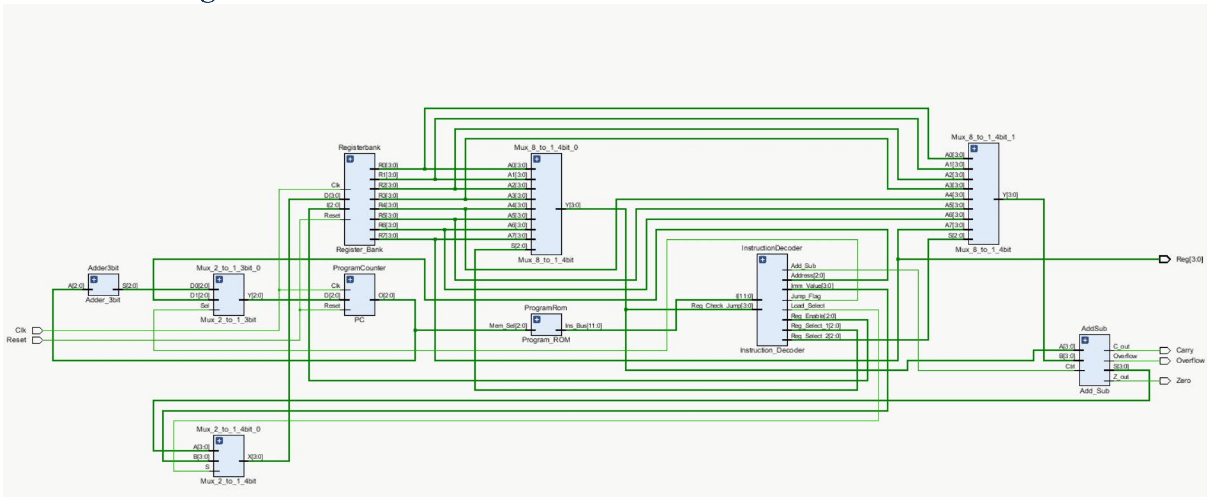
NanoProcessor

VHDL Code

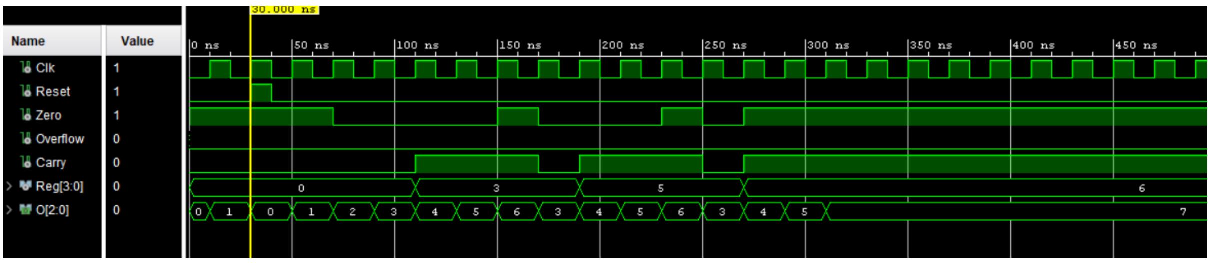
```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/11/2022 12:31:02 PM
-- Design Name:
-- Module Name: NanoProcessor - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- 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 NanoProcessor is
    Port (Clk : in STD_LOGIC;
          Reset : in STD_LOGIC;
          Reg : out STD_LOGIC_VECTOR (3 downto 0);
          Zero : out STD_LOGIC;
          Overflow : out STD_LOGIC;
          Carry: out STD_LOGIC  );
end NanoProcessor;

architecture Behavioral of NanoProcessor is
component Add_Sub
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Ctrl : in STD_LOGIC;
           C_out : out STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
           Overflow : out STD_LOGIC;
           Z_out : out STD_LOGIC
           );
end component;

component Register_Bank
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           D : in STD_LOGIC_VECTOR (3 downto 0);
           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);
           I : in STD_LOGIC_VECTOR (2 downto 0));
end component;
component Adder_3bit
    Port (
           A : in STD_LOGIC_VECTOR (2 downto 0);
           S: out STD_LOGIC_VECTOR (2 downto 0);
           C_out : out STD_LOGIC
           );
end component;
component PC
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           D : in STD_LOGIC_VECTOR (2 downto 0);
           O : out STD_LOGIC_VECTOR (2 downto 0));
end component;
component Instruction_Decoder
    Port ( I : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
           Load_Select : out STD_LOGIC;
           Imm_Value : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_1 : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_2 : out STD_LOGIC_VECTOR (2 downto 0);
           Add_Sub : out STD_LOGIC;
           Jump_Flag : out STD_LOGIC;
           Address : out STD_LOGIC_VECTOR (2 downto 0)
           );
end component;
component Mux_2_to_1_3bit
    Port ( Sel : in STD_LOGIC;
           D0 : in STD_LOGIC_VECTOR (2 downto 0);
           D1 : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end component;
component Mux_2_to_1_4bit
    Port ( S : in STD_LOGIC;
           A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           X : out STD_LOGIC_VECTOR (3 downto 0));
end component;
component Mux_8_to_1_4bit
    Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
           A0 : in STD_LOGIC_VECTOR (3 downto 0);
           A1 : in STD_LOGIC_VECTOR (3 downto 0);
           A2 : in STD_LOGIC_VECTOR (3 downto 0);
           A3 : in STD_LOGIC_VECTOR (3 downto 0);
           A4 : in STD_LOGIC_VECTOR (3 downto 0);
           A5 : in STD_LOGIC_VECTOR (3 downto 0);
           A6 : in STD_LOGIC_VECTOR (3 downto 0);
           A7 : in STD_LOGIC_VECTOR (3 downto 0);
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end component;
component Program_ROM
   Port ( Mem_Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Ins_Bus : out STD_LOGIC_VECTOR (11 downto 0));
end component;

signal Load_Select,Add_Sub_Selector,Jump_Flag,Overflow_0,Z_out : STD_LOGIC;
signal Ins_Bus : STD_LOGIC_VECTOR (11 downto 0);
signal  O,b,Add_Out,Reg_Enable,Reg_Select_1,Reg_Select_2,c_out_0,S0  :  STD_LOGIC_VECTOR  (2
downto 0);
signal Address : STD_LOGIC_VECTOR (2 downto 0);
signal Imm_Value,R0,R1,R2,R3,R4,R5,R6,R7,S,X,Y1,Y2 : STD_LOGIC_VECTOR (3 downto 0);
begin
ProgramCounter : PC
   Port Map(
      Clk => Clk,
      Reset => Reset,
      D => b,
      O => O
   );
ProgramRom : Program_Rom
   Port Map(
      Ins_Bus => Ins_Bus,
      Mem_Sel => O
   );
InstructionDecoder: Instruction_Decoder
   Port Map(
     I => Ins_Bus,
     Reg_Check_Jump => Y1,
     Load_Select => Load_Select,
     Imm_Value  => Imm_Value,
     Reg_Enable => Reg_Enable,
     Reg_Select_1 => Reg_Select_1,
     Reg_Select_2 => Reg_Select_2,
     Add_Sub => Add_Sub_Selector,
     Jump_Flag => Jump_Flag,
     Address => Address
   );
Mux_2_to_1_4bit_0 : Mux_2_to_1_4bit
   Port Map (
     S => Load_Select,
     A => S,
     B => Imm_Value,
     X => X
   );
Registerbank : Register_Bank
   Port Map (
     Clk => Clk,
     Reset => Reset,
     D => X,
     R0 => R0,
     R1 => R1,
     R2 => R2,
     R3 => R3,
     R4 => R4,
     R5 => R5,
     R6 => R6,
     R7 => R7,
     I => Reg_Enable
```

```
    );
Mux_8_to_1_4bit_0 : Mux_8_to_1_4bit
    Port Map (
       S => Reg_Select_1,
       A0 => R0,
       A1 => R1,
       A2 => R2,
       A3 => R3,
       A4 => R4,
       A5 => R5,
       A6 => R6,
       A7 => R7,
       Y => Y1
    );
Mux_8_to_1_4bit_1 : Mux_8_to_1_4bit
    Port Map (
       S => Reg_Select_2,
       A0 => R0,
       A1 => R1,
       A2 => R2,
       A3 => R3,
       A4 => R4,
       A5 => R5,
       A6 => R6,
       A7 => R7,
       Y => Y2
    );
AddSub : Add_Sub
    Port Map (
       A => Y1,
       B => Y2,
       Ctrl => Add_Sub_Selector,
       C_out => Carry,
       S => S,
       Overflow => Overflow_0,
       Z_out => Z_out
    );
Mux_2_to_1_3bit_0 : Mux_2_to_1_3bit
    Port map (
       Sel => Jump_Flag,
       D0 => Add_Out,
       D1 => Address,
       Y => b
    );
Adder3bit : Adder_3bit
    Port Map(
       A => O,
       S => Add_Out
    );
Reg <= R7;
Zero <= Z_out;
Overflow <= Overflow_0;

end Behavioral;
```

Schematic Diagram



Timing Diagram



Name	Value
Clk	1
Reset	1
Zero	1
Overflow	0
Carry	0
Reg[3:0]	0
O[2:0]	0

```
Reg[3:0]  0  3  5  6
O[2:0]    0  1  0  1  2  3  4  5  6  3  4  5  6  3  4  5  7
```

Note : Here O is program counter. Even though reset is not set to 1 program will start immediately.

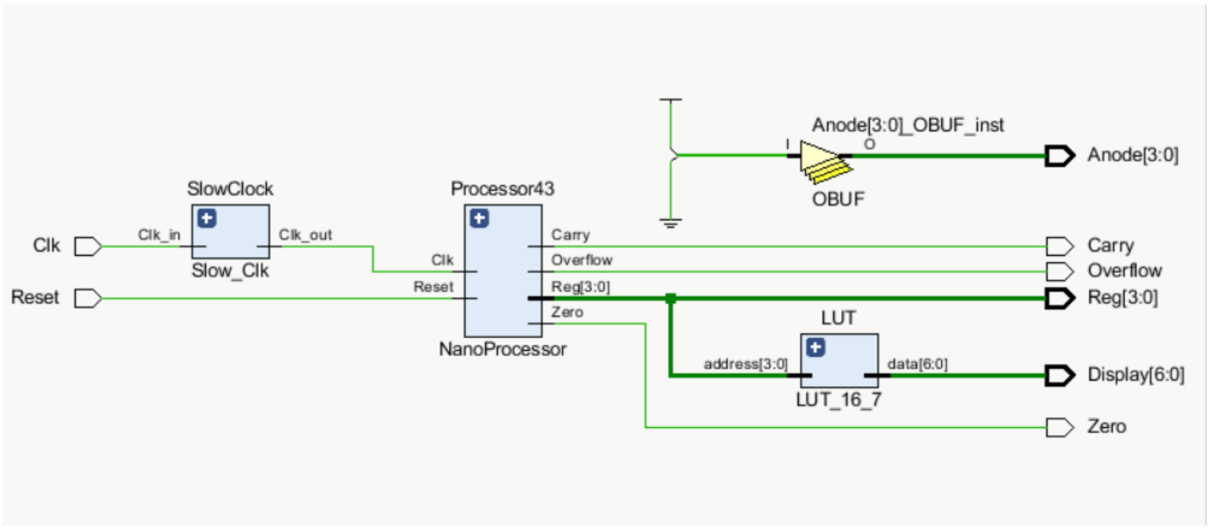
OurProcessor

It is a nano processor with slow clock and 7 segment display combined.

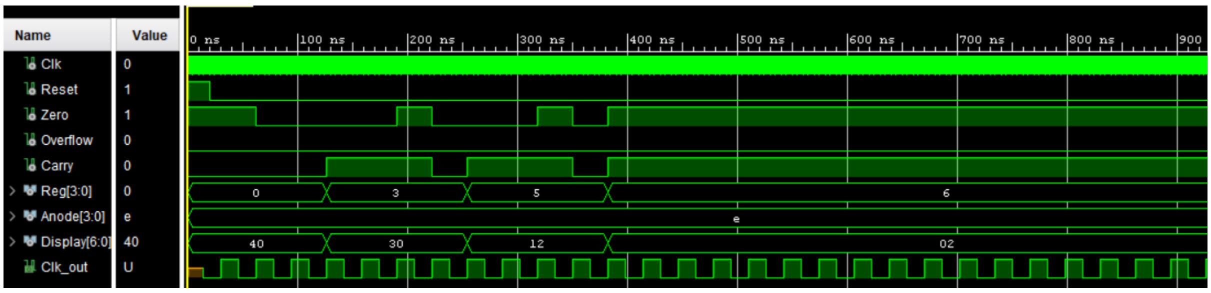
VHDL Code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 07/14/2022 11:16:09 PM
-- Design Name:
-- Module Name: OurProcessor - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- 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 OurProcessor is
    Port (Clk : in STD_LOGIC;
          Reset : in STD_LOGIC;
          Reg : out STD_LOGIC_VECTOR (3 downto 0);
          Zero : out STD_LOGIC;
          Overflow : out STD_LOGIC;
          Display : out STD_LOGIC_VECTOR (6 downto 0);
          Anode: out STD_LOGIC_VECTOR (3 downto 0);
          Carry: out STD_LOGIC );
end OurProcessor;

architecture Behavioral of OurProcessor is
component LUT_16_7
    Port ( address : in STD_LOGIC_VECTOR (3 downto 0);
           data : out STD_LOGIC_VECTOR (6 downto 0));
end component;
component NanoProcessor
    Port (Clk : in STD_LOGIC;
          Reset : in STD_LOGIC;
          Reg : out STD_LOGIC_VECTOR (3 downto 0);
          Zero : out STD_LOGIC;
          Overflow : out STD_LOGIC;
          Carry: out STD_LOGIC );
end component;
component Slow_Clk
    Port ( Clk_in : in STD_LOGIC;
           Clk_out : out STD_LOGIC);
end component;
signal R: STD_LOGIC_VECTOR (3 downto 0);
signal Clk_out : STD_LOGIC;
begin
SlowClock : Slow_Clk
    Port map (
        Clk_in => Clk,
        Clk_out => Clk_out
    );
Processor43 : NanoProcessor
    Port map (
        Clk => Clk_out,
        Reset => Reset,
        Reg => R,
        Zero => Zero,
        Overflow => Overflow,
        Carry => Carry
    );
LUT :LUT_16_7
    Port map (
        address => R,
        Data => Display
    );
Reg <= R;
Anode <= "1110";

end Behavioral;
```

Schematic Diagram



Timing Diagram



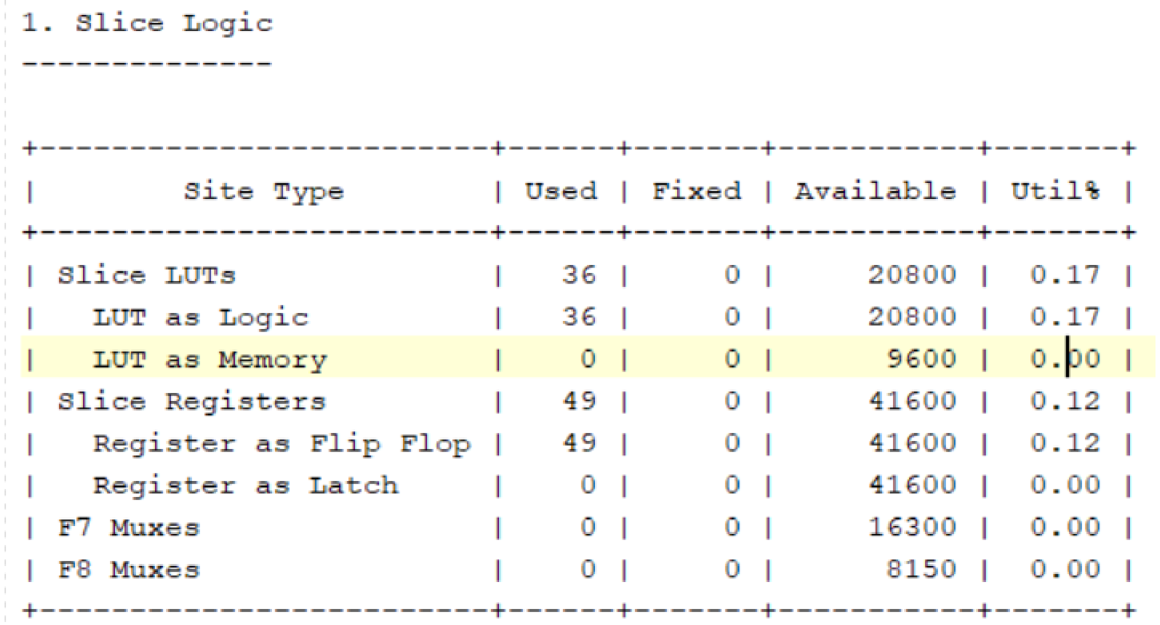
Contributions of each member

Name	Components
P.Kobinarth	4bit Add/ Subract unit 3-bit Adder Instruction Decoder (All)
T.Pairavi	2-way-3bit Multiplexer 2-way-4bit Multiplexer 8-way-4bit Multiplexer Instruction Decoder (All)
S.Nisanthan	4-bit Register Register bank Instruction Decoder (All)
P.Sanujen	Program ROM LUT 16 to 7 Instruction Decoder (All)
Y.Sathveegan	3-bit Programming Counter Slow Clock Instruction Decoder (All)

Resource consumption optimizations to the basic program designs,

- We created the instruction decoder without any use of clock input. Initially, we created the instruction decoder which will decode the instruction for the rising edge of the clock. At that time we used conditional statements to decode the instructions.
- After that we designed the instruction decoder without any usage of clock. We analyzed the logic implementation which will be correct according to all four instructions. As a result, the instruction will be decoded immediately as the instruction arrives in the input port and the time delay for decoding has been removed.
- In order to decode the negation statement we used the 2-way-3bit multiplexer for sending data to the register selector because we have to send the data to the second 8-way-4bit multiplexer.

Final LUT/FF counts.



```
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 |
+-------------------------+------+-------+-----------+-------+
```

Conclusions from the lab

- This circuit required the creation/extension of various components.
- We created a 4-bit arithmetic unit that can only handle smaller tasks, such as adding and subtracting 4-bit signed numbers. We are limited to working with integers between -7 to 8.
- Instead of connecting so many wires, we used busses which simplified the design circuit.
- We recalled several past lab activities in order to develop the nano processor (4bit RCA-Lab3, D flipflop-Lab5, 3 to 8 decoder, multiplexer-Lab4, ROM-based LUT-Lab7) which make our work easier.
- We hardcoded our program to ROM because microprocessor only understands the machine language.
- Middle push button is used to reset the PC and register bank.
- We used the slow clock to drive our nano processor in order to be able to detect changes.
- By verifying each and every component's functionality via simulation and on the development board, we confirmed the proper work of nano processor.
- Through this work, we gained a wide knowledge about how microprocessor worked internally. Our teamwork abilities, such as coordination and communication, are improved through this project. Five of us divided up the tasks, which we later merged to create a nano processor.

♥♥♥♥♥♥♥♥♥