**NanoProcessor Design**

**Group Number - 43**

# **Group members:**

P.Kobinarth (200307C)

T.Pairavi (200441F)

S.Nisanthan (200432E)

P.Sanujen (200583P)

Y.Sathveegan (200592R)

# **Introduction:**

Diagram, schematic

Description automatically generatedWe 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.

# **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**

Diagram

Description automatically generated

### **Timing Diagram**

**Graphical user interface

Description automatically generated**

## **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**

Diagram

Description automatically generated with medium confidence

### **Timing Diagram**

**Graphical user interface, diagram

Description automatically generated**

## **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**

Diagram, schematic

Description automatically generated

### **Timing Diagram**

**A picture containing timeline

Description automatically generated**

## **2-way-3bit Multiplexer**

* 1. 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 1control 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**

Chart

Description automatically generated

### **Timing Diagram**

**Timeline

Description automatically generated with medium confidence**

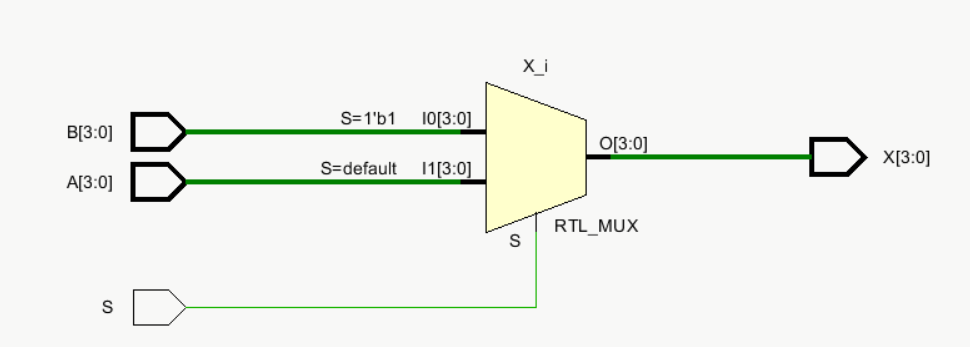
## **2-way-4bit Multiplexer**

* 1. 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 1control 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**

**Graphical user interface, timeline

Description automatically generated**

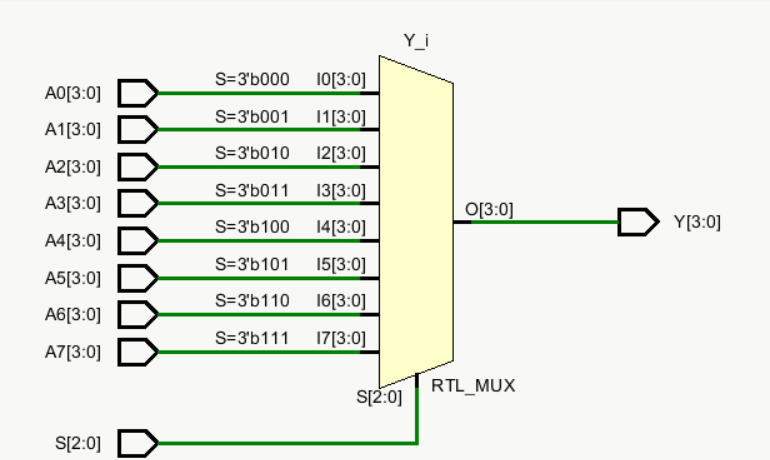
## **8-way-4bit Multiplexer**

* 1. 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 3control bits, and these control bits are used to select one of the *8* groups of *4* bits rather than a single bit.

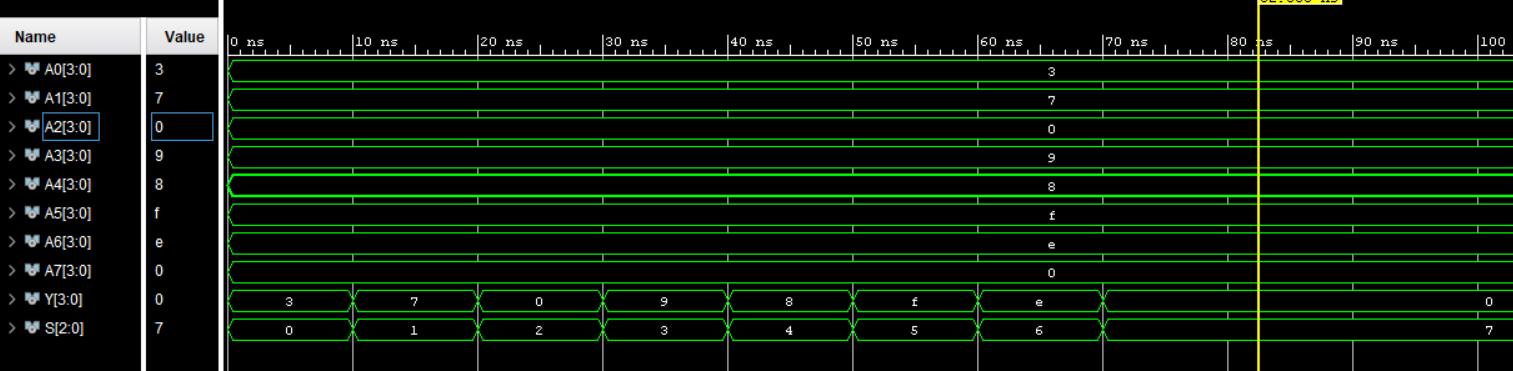
### **VHDL Code**

|  |
| --- |
| ----------------------------------------------------------------------------------  -- 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**

|  |
| --- |
| ----------------------------------------------------------------------------------  -- 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**

Diagram

Description automatically generated

### **Timing Diagram**

**Graphical user interface

Description automatically generated**

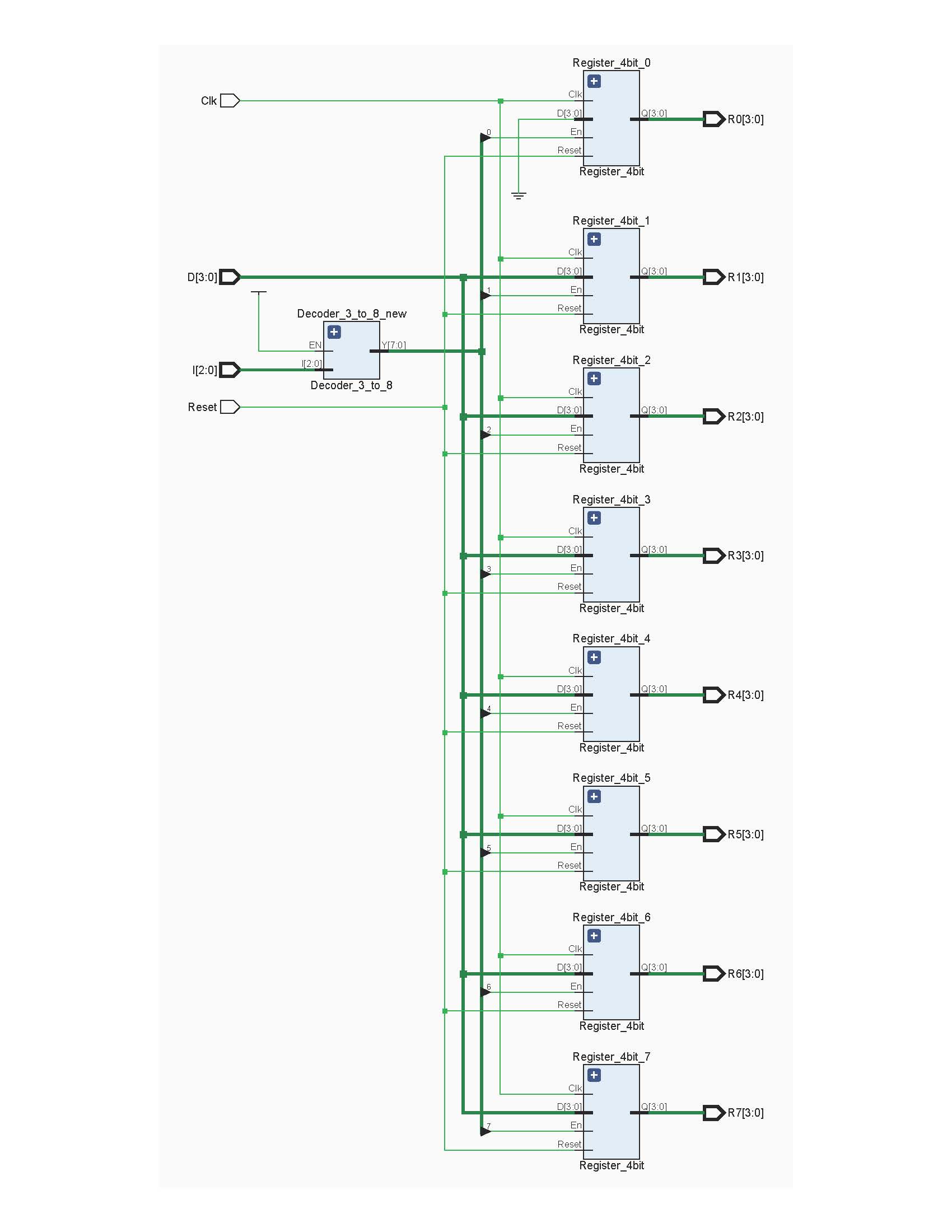
## **Register Bank**

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**

**A picture containing timeline

Description automatically generated**

## **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;

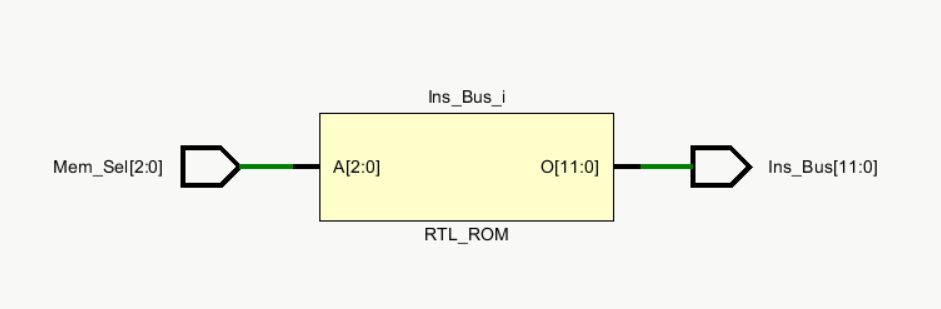
**Machine Code**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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**

**A picture containing chart

Description automatically generated**

## **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  -- 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**

Diagram, schematic

Description automatically generated

### **Timing Diagram**

****

## **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**

Diagram, schematic

Description automatically generated

## **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**

Diagram

Description automatically generated

## **NanoProcessor**

### **VHDL Code**

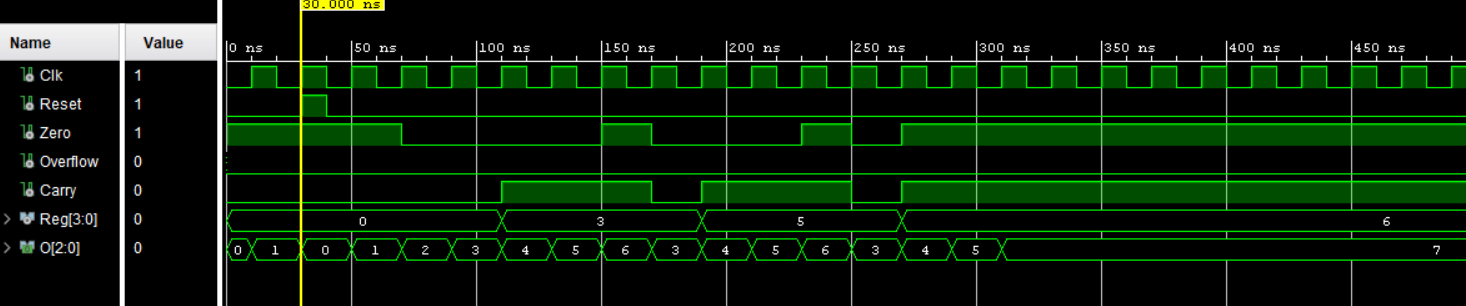
|  |
| --- |
| ----------------------------------------------------------------------------------  -- Company:  -- Engineer:  --  -- Create Date: 07/11/2022 12:31:02 PM  -- Design Name:  -- Module Name: NanoProcessor - Behavioral  -- Project Name:  -- Target Devices:  -- Tool Versions:  -- Description:  --  -- 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**

Diagram, schematic

Description automatically generated

### **Timing Diagram**

****

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**

Diagram

Description automatically generated

### **Timing Diagram**

**Graphical user interface

Description automatically generated**

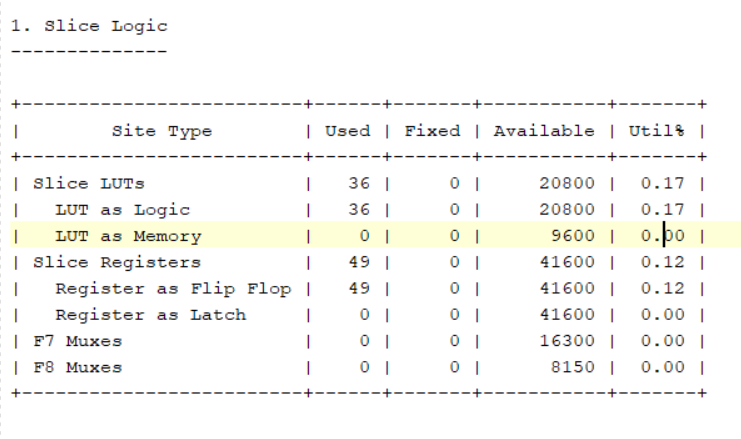
# **Contributions of each member**

|  |  |
| --- | --- |
| **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.**

****

**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.

♥♥♥♥♥♥♥♥♥