Computer Architecture - Project 2
Microprogrammed Control

Vsevolod Syrtsov
syrtsovv@tcd.ie
source: github.com/futurecertificate

Contents

1 VHDL Sources

1.1 Control Address Register

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity CAR is
        Port(         Cmux : in STD_LOGIC_VECTOR(7 downto 0);
                      Smux, reset, Clk : in STD_LOGIC;
                      car_out : out STD_LOGIC_VECTOR(7 downto 0)
                      );
end CAR;

architecture Behavioral of CAR is

COMPONENT ALU
Port (
    A: in std_logic_vector(15 downto 0);
    B: in std_logic_vector(15 downto 0);
    GS: in std_logic_vector(3 downto 0);

    G: out std_logic_vector(15 downto 0);

    C: out std_logic;
    V: out std_logic
    );
END COMPONENT;

    signal Ain, Bin, ALUout : std_logic_vector(15 downto 0);
    signal GS : std_logic_vector(3 downto 0);

begin

alu_op: ALU PORT MAP(
A => Ain,
B => Bin,
GS => GS,
```

```
G => ALUout
);

        process(reset, Cmux, Clk)
        variable car : STD_LOGIC_VECTOR(7 downto 0);

        begin
        if rising_edge(Clk) then
                if(reset = '1') then car := x"C0";
                elsif(Smux = '1') then car := Cmux;
                elsif(Smux = '0') then

                        GS <= x"1";
                        Ain <= x"00" & Cmux;
                        Bin <= x"0000";
                        car := ALUout(7 downto 0);

                end if;
                car_out <= car after 20ns;
        end if;
        end process;
end Behavioral;
```

1.2 Control Memory

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ControlMemory is
        Port(        in_car : in STD_LOGIC_VECTOR(7 downto 0);
                        MW, MM, RW, MD, MB, TB, TA, TD, PL, PI, IL, MC : out STD
                        FS_cm : out STD_LOGIC_VECTOR(4 downto 0);
                        MS_cm : out STD_LOGIC_VECTOR(2 downto 0);
                        NA : out STD_LOGIC_VECTOR(7 downto 0)
                        );
end ControlMemory;

architecture Behavioral of ControlMemory is
        --instantiate an array for each given memory allocation
```

```
        type mem_array is array(0 to 255) of STD_LOGIC_VECTOR(27 downto 0);

begin
        memory_m : process(in_car)
        variable ControlMemory : mem_array := (
        --Module 0
        b"11000000001000000011000000100", --0 load value into register
    x"C020224", --1 ADI -> add the immediate operand
    x"C02000C", --2 LD -> load to register
    x"C020001", --3 SR -> store in register
    x"C020014", --4 INC -> increment the register's value by 1
    x"C0200E4", --5 NOT -> invert
    x"C020024", --6 ADD -> add values from source A and B into destination
    x"0028002", --7 B -> branch unconditionally
    x"009A002", --8 BXX -> branch conditionally if z set (MS = 100)
    x"0000000", --9
        x"0000000", --A
        x"0000000",         --B
    x"0000000", --C
        x"0000000", --D
        x"0000000", --E
        x"0000000", --F

        --Module 1
        x"0000000", --0
        x"0000000", --1
        x"0000000", --2
        x"0000000", --3
        x"0000000", --4
        x"0000000", --5
        x"0000000", --6
        x"0000000", --7
        x"0000000", --8
        x"0000000", --9
        x"0000000", --A
        x"0000000", --B
        x"0000000", --C
        x"0000000", --D
        x"0000000", --E
        x"0000000", --F
```

```
--Module 2
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module 3
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module 4
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
```

```
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module 5
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module 6
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
```

```
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module 7
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module 8
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
```

```
x"0000000", --E
x"0000000", --F

--Module 9
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module A
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module B
x"0000000", --0
```

```
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module C
x"c10c002", --0 IF fetching
x"0030000", --1 Exit signal
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module D
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
```

```
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module E
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
x"0000000", --B
x"0000000", --C
x"0000000", --D
x"0000000", --E
x"0000000", --F

--Module F
x"0000000", --0
x"0000000", --1
x"0000000", --2
x"0000000", --3
x"0000000", --4
x"0000000", --5
x"0000000", --6
x"0000000", --7
x"0000000", --8
x"0000000", --9
x"0000000", --A
```

```
        x"0000000", --B
        x"0000000", --C
        x"0000000", --D
        x"0000000", --E
        x"0000000" --F
        );

variable addr : integer;
variable control_out : STD_LOGIC_VECTOR(27 downto 0);

begin
        addr := conv_integer(in_car);
        control_out := ControlMemory(addr);
        MW <= control_out(0);
        MM <= control_out(1);
        RW <= control_out(2);
        MD <= control_out(3);
        FS_cm <= control_out(8 downto 4);
        MB <= control_out(9);

        TB <= control_out(10);
        TA <= control_out(11);
        TD <= control_out(12);
        PL <= control_out(13);
        PI <= control_out(14);
        IL <= control_out(15);
        MC <= control_out(16);
        MS_cm <= control_out(19 downto 17);
        NA <= control_out(27 downto 20);
end process;

end Behavioral;
```

1.3 Decoder 4 to 9

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder4_9 is
Port (
```

```
LE : in std_logic;
A0 : in std_logic;
A1 : in std_logic;
A2 : in std_logic;
A3 : in std_logic;


Q0 : out std_logic;
Q1 : out std_logic;
Q2 : out std_logic;
Q3 : out std_logic;
Q4 : out std_logic;
Q5 : out std_logic;
Q6 : out std_logic;
Q7 : out std_logic;
Q8 : out std_logic
);

end decoder4_9;
architecture Behavioral of decoder4_9 is
begin
Q0<=((not A0)and (not A1)and (not A2)and(not A3)) and LE;
Q1<=((not A0)and (not A1)and (not A2)and(A3)) and LE;
Q2<=((not A0)and (not A1)and (A2)and(not A3)) and LE;
Q3<=((not A0)and (not A1)and (A2)and(A3)) and LE;
Q4<=((not A0)and (A1)and (not A2)and(not A3)) and LE;
Q5<=((not A0)and (A1)and (not A2)and(A3)) and LE;
Q6<=((not A0)and (A1)and (A2)and(not A3)) and LE;
Q7<=((not A0)and (A1)and (A2)and(A3)) and LE;
Q8<=((A0)and (not A1)and (not A2)and(not A3)) and LE;

end Behavioral;
```

1.4 Datapath

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity datapath_1B is
Port (

SB: in std_logic_vector(2 downto 0);
AS0: in std_logic;
AS1: in std_logic;
AS2: in std_logic;
AS3: in std_logic;

BS0: in std_logic;
BS1: in std_logic;
BS2: in std_logic;
BS3: in std_logic;

des_A0: in std_logic;
des_A1: in std_logic;
des_A2: in std_logic;
des_A3: in std_logic;

MB_Select: in std_logic;
MD_Select: in std_logic;

Clk: in std_logic;

LE: in std_logic;
FS: in std_logic_vector(4 downto 0);
Data_in: in std_logic_vector(15 downto 0);

MM: in std_logic;
pc_in: in std_logic_vector(15 downto 0);

Add_out : out std_logic_vector(15 downto 0); --Address out
B_out : out std_logic_vector(15 downto 0); --Data Out

V: out std_logic;
C: out std_logic;
Z: out std_logic;
N: out std_logic
);
end datapath_1B;
```

```
architecture Behavioral of datapath_1B is

COMPONENT reg_file PORT(
LE : in std_logic;

AS0 : in std_logic;
AS1 : in std_logic;
AS2 : in std_logic;
AS3 : in std_logic;

BS0 : in std_logic;
BS1 : in std_logic;
BS2 : in std_logic;
BS3 : in std_logic;

des_A0 : in std_logic;
des_A1 : in std_logic;
des_A2 : in std_logic;
des_A3 : in std_logic;

Clk : in std_logic;

D_data : in std_logic_vector(15 downto 0);
B_data : out std_logic_vector(15 downto 0);
A_data : out std_logic_vector(15 downto 0)

);
END COMPONENT;

COMPONENT zero_fill
Port(
        SB : in  STD_LOGIC_VECTOR (2 downto 0);
            constant_out : out STD_LOGIC_VECTOR (15 downto 0)
);
END COMPONENT;

COMPONENT function_unit
Port (
A: in std_logic_vector(15 downto 0);
B: in std_logic_vector(15 downto 0);
FS: in std_logic_vector(4 downto 0);
```

```
V: out std_logic;
C: out std_logic;
Z: out std_logic; --return 1 if 0
N: out std_logic; --return MSB
F: out std_logic_vector(15 downto 0)
);

END COMPONENT;

COMPONENT mux_2_16
port (
In0 : in std_logic_vector(15 downto 0);
In1 : in std_logic_vector(15 downto 0);
s : in std_logic;
Z : out std_logic_vector(15 downto 0)
);
END COMPONENT;

signal A_data, B_data, B_data_mux_src : std_logic_vector(15 downto 0); --Data co
signal FZ, D: std_logic_vector(15 downto 0); --Data coming from function unit (
signal zf_in: std_logic_vector(15 downto 0);


begin

fu: function_unit PORT MAP(
A=> A_data,
B=> B_data_mux_src,
FS => FS,
V => V,
C => C,
Z => Z,
N => N,
F=>FZ
);

zf : zero_fill PORT MAP (
    SB=>SB,
    constant_out =>zf_in
);
```

```
muxB: mux_2_16 PORT MAP(
In0 => B_data,
In1 => zf_in,
s => MB_Select,
Z => B_data_mux_src
);

muxD: mux_2_16 PORT MAP(
In0 => FZ,
In1 => Data_in,
s => MD_Select,
Z => D
);

muxM: mux_2_16 PORT MAP(
In0 => A_data,
In1 =>pc_in,
s => MM,
z => Add_out
);

regf: reg_file PORT MAP(
LE => LE,
AS0 => AS0,
AS1 => AS1,
AS2 => AS2,
AS3 => AS3,

BS0 => BS0,
BS1 => BS1,
BS2 => BS2,
BS3 => BS3,

des_A0 => des_A0,
des_A1 => des_A1,
des_A2 => des_A2,
des_A3 => des_A3,
Clk => Clk,
D_Data => D,
A_Data => A_Data,
```

```
B_Data => B_Data
);

B_Out <= B_data_mux_src after 1 ns;
end Behavioral;
```

1.5 Extender to PC

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity E_PC is
        Port(         SR_SB : in STD_LOGIC_VECTOR(5 downto 0);
                      signed_extension : out STD_LOGIC_VECTOR(15 downto 0)
                      );
end E_PC;

architecture Behavioral of E_PC is
        signal extended_signal : STD_LOGIC_VECTOR(15 downto 0);
begin
        extended_signal(5 downto 0) <= SR_SB;
        extended_signal(15 downto 6) <= "0000000000" when SR_SB(5) = '0' else "1
        signed_extension <= extended_signal;

end Behavioral;
```

1.6 Instruction Register

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity InstructionR is
        Port(         IR : in STD_LOGIC_VECTOR(15 downto 0);
                      IL, Clk : in STD_LOGIC;
                      OPCODE :  out STD_LOGIC_VECTOR(6 downto 0);
                      DR, SA, SB : out STD_LOGIC_VECTOR(2 downto 0)
                      );
end InstructionR;

architecture Behavioral of InstructionR is
```

```
begin
    process(Clk)
    begin
    if(rising_edge(Clk)) then
        if IL = '1' then
                OPCODE <= IR(15 downto 9) after 1ns;
                DR <= IR(8 downto 6) after 1ns;
                SA <= IR(5 downto 3) after 1ns;
            SB <= IR(2 downto 0) after 1ns;
        end if;
     end if;
     end process;
end Behavioral;
```

1.7 Memory - RAM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Memory is
    Port (
            data_in : in std_logic_vector(15 downto 0);
            address_in : in std_logic_vector(15 downto 0);
            mw: in std_logic;
            Clk: in std_logic;
            data_out : out std_logic_vector(15 downto 0));
end Memory;

architecture Behavioral of Memory is
    type mem_array is array(0 to 511) of std_logic_vector(15 downto 0);
begin
    mem_process: process(address_in, data_in,Clk)
    variable data_mem: mem_array := (
    --M-0
    x"0000", --0
    x"0000", -- R0 <= 0
    x"0041", -- R1 <= 1
    x"0082", -- R2 <= 2
    x"00C3", -- R3 <= 3
```

```
x"0104", -- R4 <= 4
x"0145", -- R5 <= 5
x"0186", -- R6 <= 6
x"01C7", -- R7 <= 7
b"0000110001010011", --9 ADD R1, R2, R3
b"0000010100101000", --A R4 <= M[R5]    (LD)
b"0000011000010011", --B R3 => M[R2]    (SR)
b"0000001001010011", --C ADI R1, R2, #3
b"0000100010010000", --D R2 = R2 + 1    (INC)
b"0000101011011000", --E R2 = NOT R2
b"0000111010010001", --F B -> unconditional jump (B) to 010001 (11)

--M-1
x"0C52", --0 R1 = R2 + R2 (ADD)
b"0001000000000010", --1 BC -> conditional branch, z is set (BC)
x"0E00", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",

--M-2
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-3
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-4
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-5
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-6
```

```
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-7
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-8
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-9
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-A
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-B
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-C
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-D
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-E
x"0000", x"0000", x"0000", x"0000",
```

```
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-F
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-10
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-11
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-12
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-13
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-14
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-15
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-16
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
```

```
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-17
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-18
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-19
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-1A
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-1B
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-1C
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-1D
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
--M-1E
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
x"0000", x"0000", x"0000", x"0000",
```

```
    x"0000", x"0000", x"0000", x"0000",
    --M-1F
    x"0000", x"0000", x"0000", x"0000",
    x"0000", x"0000", x"0000", x"0000",
    x"0000", x"0000", x"0000", x"0000",
    x"0000", x"0000", x"0000", x"0000");


    variable addr:integer range 0 to 511;
    begin
    if rising_edge(Clk) then
        addr := conv_integer(address_in(8 downto 0));
        if mw = '1' then
            data_mem(addr) := data_in;
        end if;
        if mw = '0' then
            data_out <= data_mem(addr);
        end if;
    end if;
    end process;
end Behavioral;
```

1.8 Microprogrammed Control

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity Microprogrammed_Control is
  Port (
            data_in: in std_logic_vector(15 downto 0);
            V: in std_logic;
            C: in std_logic;
            Z: in std_logic; --return 1 if 0
            N: in std_logic; --return MSB
            reset: in std_logic;
            Clk: in std_logic;
            pc_out: out std_logic_vector(15 downto 0);
            TD_DR: out std_logic_vector(3 downto 0);
```

```
            TA_SA: out std_logic_vector(3 downto 0);
            TB_SB: out std_logic_vector(3 downto 0);
            SB: out std_logic_vector(2 downto 0);
            mb: out std_logic;
            md: out std_logic;
            rw: out std_logic;
            mm: out std_logic;
            mw: out std_logic;
            fs: out std_logic_vector(4 downto 0)
  );
end Microprogrammed_Control;

architecture Behavioral of Microprogrammed_Control is

COMPONENT InstructionR Port(
        IR : in STD_LOGIC_VECTOR(15 downto 0);
        IL, Clk : in STD_LOGIC;
        OPCODE :  out STD_LOGIC_VECTOR(6 downto 0);
        DR, SA, SB : out STD_LOGIC_VECTOR(2 downto 0)
);
END COMPONENT;

COMPONENT E_PC  Port(
        SR_SB : in STD_LOGIC_VECTOR(5 downto 0);
        signed_extension : out STD_LOGIC_VECTOR(15 downto 0)
);
END COMPONENT;

COMPONENT mux2_1 Port (
    in0 : in  std_logic_vector(7 downto 0);
    in1 : in  std_logic_vector(7 downto 0);
    mc: in std_logic;
    z : out std_logic_vector(7 downto 0)
 );
END COMPONENT;

COMPONENT mux8_1 port (
    MS : in std_logic_vector(2 downto 0);
    in0 : in std_logic;
    in1 : in std_logic;
    in2 : in std_logic;
```

```
    in3 : in std_logic;
    in4 : in std_logic;
    in5 : in std_logic;
    in6 : in std_logic;
    in7 : in std_logic;

    MUXS : out std_logic
);
END COMPONENT;

COMPONENT PCreg Port(
    seAD : in STD_LOGIC_VECTOR(15 downto 0);
        PL, PI, reset, Clk : in STD_LOGIC;
        PC : out STD_LOGIC_VECTOR(15 downto 0)
);
END COMPONENT;

COMPONENT CAR Port(
    Cmux : in STD_LOGIC_VECTOR(7 downto 0);
        Smux, reset, Clk : in STD_LOGIC;
        car_out : out STD_LOGIC_VECTOR(7 downto 0)
);
END COMPONENT;

COMPONENT ControlMemory Port(
    in_car : in STD_LOGIC_VECTOR(7 downto 0);
        MW, MM, RW, MD, MB, TB, TA, TD, PL, PI, IL, MC : out STD_LOGIC;
        FS_cm : out STD_LOGIC_VECTOR(4 downto 0);
        MS_cm : out STD_LOGIC_VECTOR(2 downto 0);
        NA : out STD_LOGIC_VECTOR(7 downto 0)
);
END COMPONENT;


signal IL : std_logic;
signal PL : std_logic;
signal PI : std_logic;
signal MC : std_logic;
signal NA : std_logic_vector(7 downto 0);
signal MS : std_logic_vector(2 downto 0);
signal TD : std_logic;
```

```
signal TB : std_logic;
signal TA : std_logic;

signal notZ : std_logic;
signal notC : std_logic;
signal SR_SB : std_logic_vector(5 downto 0);
signal pad_OPCODE : std_logic_vector(7 downto 0);
signal CAR_in : std_logic_vector(7 downto 0);
signal OPCODE : STD_LOGIC_VECTOR(6 downto 0);
signal DRout : std_logic_vector(2 downto 0);
signal SAout : std_logic_vector(2 downto 0);
signal SBout : std_logic_vector(2 downto 0);
signal muxs_out : std_logic;
signal extended_pc : std_logic_vector(15 downto 0);
signal mem_in : std_logic_vector(7 downto 0);

begin

ir : InstructionR Port Map(
    IR=>data_in,
    IL=>IL,
    Clk=>Clk,
    OPCODE =>OPCODE,
    DR => DRout,
    SA => SAout,
    SB => SBout


);
SR_SB <= DRout & SBout;
epc : E_PC Port Map(
    SR_SB => SR_SB,
    signed_extension => extended_pc
);

pc : PCreg Port Map(
    seAD =>extended_pc,
    PL => PL,
    PI => PI,
    Clk => Clk,
    reset => reset,
```

```
    PC => pc_out

);

pad_OPCODE <= '0' & OPCODE;
muxc : mux2_1 Port Map (
    In0 => NA,
    In1 => pad_OPCODE,
    mc => MC,
    z => CAR_in

);


notZ <= not Z;
notC <= not C;

muxs : mux8_1 Port Map (
    ms => MS,
    in0 => '0',
    in1 => '1',
    in2 => C,
    in3 => V,
    in4 => Z,
    in5 => N,
    in6 => notC,
    in7 => notZ,
    MUXS => muxs_out

);

car0 : CAR Port Map (
    Cmux =>CAR_in,
    Smux => muxs_out,
    reset => reset,
    Clk => Clk,
    car_out => mem_in
);


--MW, MM, RW, MD, MB, TB, TA, TD, PL, PI, IL, MC
```

```
cm : ControlMemory Port Map (
    in_car => mem_in,
    MW => mw,
    MM => mm,
    RW => rw,
    MB => mb,
    MD => md,
    TA => TA,
    TB => TB,
    TD => TD,
    PL => PL,
    PI => PI,
    IL => IL,
    MC => MC,
    FS_cm => fs,
    MS_cm => MS,
    NA =>NA

);
TD_DR(0) <= TD;
TD_DR(3 downto 1) <= DRout;

TA_SA(0) <= TA;
TA_SA(3 downto 1) <= SAout;

TB_SB(0) <= TB;
TB_SB(3 downto 1) <= SBout;

SB<= SBout;

end Behavioral;
```

1.9 Multiplexer 2 to 1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity mux2_1 is
    Port (
            in0 : in  std_logic_vector(7 downto 0);
            in1 : in  std_logic_vector(7 downto 0);
            mc: in std_logic;
```

```
            z : out std_logic_vector(7 downto 0));
end mux2_1;

architecture Behavioral of mux2_1 is
begin
    z <=    in0 after 5 ns when mc = '0' else
            in1 after 5 ns when mc = '1' else
            x"00" after 5 ns;
end Behavioral;
```

1.10 Multiplexer 8 to 1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mux8_1 is
port (
MS : in std_logic_vector(2 downto 0);
in0 : in std_logic;
in1 : in std_logic;
in2 : in std_logic;
in3 : in std_logic;
in4 : in std_logic;
in5 : in std_logic;
in6 : in std_logic;
in7 : in std_logic;

MUXS : out std_logic);

end mux8_1;
architecture Behavioral of mux8_1 is
begin
MUXS <= in0 after 1 ns when MS(2) ='0' and MS(1) ='0' and MS(0) ='0' else
in1 after 1 ns when MS(2) ='0' and MS(1) ='0' and MS(0) ='1' else
in2 after 1 ns when MS(2) ='0' and MS(1) ='1' and MS(0) ='0' else
in3 after 1 ns when MS(2) ='0' and MS(1) ='1' and MS(0) ='1' else
in4 after 1 ns when MS(2) ='1' and MS(1) ='0' and MS(0) ='0' else
in5 after 1 ns when MS(2) ='1' and MS(1) ='0' and MS(0) ='1' else
in6 after 1 ns when MS(2) ='1' and MS(1) ='1' and MS(0) ='0' else
in7 after 1 ns when MS(2) ='1' and MS(1) ='1' and MS(0) ='1' else
```

```
'0' after 1 ns;
end Behavioral;
```

1.11 Multiplexer 9 to 16

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mux9_16 is
Port ( In0, In1, In2, In3, In4, In5, In6, In7, In8 : in std_logic_vector(15 down
S0, S1, S2, S3 : in std_logic;
Z : out std_logic_vector(15 downto 0));

end mux9_16;
architecture Behavioral of mux9_16 is
begin

Z<= In0 after 5 ns when S0='0' and S1='0' and S2 ='0' and S3 = '0' else
In1 after 5 ns when S0='0' and S1='0' and S2 ='0' and S3 = '1' else
In2 after 5 ns when S0='0' and S1='0' and S2 ='1' and S3 = '0' else
In3 after 5 ns when S0='0' and S1='0' and S2 ='1' and S3 = '1' else
In4 after 5 ns when S0='0' and S1='1' and S2 ='0' and S3 = '0' else
In5 after 5 ns when S0='0' and S1='1' and S2 ='0' and S3 = '1' else
In6 after 5 ns when S0='0' and S1='1' and S2 ='1' and S3 = '0' else
In7 after 5 ns when S0='0' and S1='1' and S2 ='1' and S3 = '1' else
In8 after 5 ns when S0='1' and S1='0' and S2 ='0' and S3 = '0' else
"0000000000000000" after 5 ns;

end Behavioral;
```

1.12 Program Counter

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity PCreg is
        Port(         seAD : in STD_LOGIC_VECTOR(15 downto 0);
                        PL, PI, reset, Clk : in STD_LOGIC;
```

```
                        PC : out STD_LOGIC_VECTOR(15 downto 0)
                        );
end PCreg;

architecture Behavioral of PCreg is

COMPONENT ALU
Port (
    A: in std_logic_vector(15 downto 0);
    B: in std_logic_vector(15 downto 0);
    GS: in std_logic_vector(3 downto 0);

    G: out std_logic_vector(15 downto 0);

    C: out std_logic;
    V: out std_logic
    );
END COMPONENT;

    signal Ain, Bin, ALUout : std_logic_vector(15 downto 0);
    signal GS : std_logic_vector(3 downto 0);

begin

alu_op: ALU PORT MAP(
A => Ain,
B => Bin,
GS => GS,

G => ALUout
);


        process(Clk, reset, PL, PI)
        variable PCv : STD_LOGIC_VECTOR(15 downto 0);

        begin
        if(rising_edge(Clk)) then
                if(reset = '1') then PCv := x"0000";
                elsif(PL = '1') then
                    GS  <= x"2";
```

```
                    Ain <= PCv;
                    Bin <= seAD;
                        PCv := ALUout;
                elsif(PI = '1') then
                        GS <= x"1";
                        Ain <= PCv;
                        Bin <= x"0000";
                        PCv := ALUout;

                end if;
                PC <= PCv after 2ns;
        end if;
        end process;

end Behavioral;
```

1.13 Register file

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity reg_file is
Port (
LE : in std_logic;

AS0 : in std_logic;
AS1 : in std_logic;
AS2 : in std_logic;
AS3 : in std_logic;

BS0 : in std_logic;
BS1 : in std_logic;
BS2 : in std_logic;
BS3 : in std_logic;

des_A0 : in std_logic;
des_A1 : in std_logic;
des_A2 : in std_logic;
des_A3 : in std_logic;
```

```
Clk : in std_logic;

D_data : in std_logic_vector(15 downto 0);
B_data : out std_logic_vector(15 downto 0);
A_data : out std_logic_vector(15 downto 0)

);

end reg_file;

architecture Behavioral of reg_file is

COMPONENT reg16
PORT(

D : IN std_logic_vector(15 downto 0);
load : IN std_logic;
Clk : IN std_logic;
Q : OUT std_logic_vector(15 downto 0)
);

END COMPONENT;

COMPONENT mux9_16
PORT(

In0 : IN std_logic_vector(15 downto 0);
In1 : IN std_logic_vector(15 downto 0);
In2 : IN std_logic_vector(15 downto 0);
In3 : IN std_logic_vector(15 downto 0);
In4 : IN std_logic_vector(15 downto 0);
In5 : IN std_logic_vector(15 downto 0);
In6 : IN std_logic_vector(15 downto 0);
In7 : IN std_logic_vector(15 downto 0);
In8 : IN std_logic_vector(15 downto 0);

S0 : IN std_logic;
S1 : IN std_logic;
S2 : IN std_logic;
S3 : IN std_logic;
Z : OUT std_logic_vector(15 downto 0)
```

```
);
END COMPONENT;

COMPONENT mux2_16
PORT(

In0 : IN std_logic_vector(15 downto 0);
In1 : IN std_logic_vector(15 downto 0);
s : IN std_logic;
Z : OUT std_logic_vector(15 downto 0)
);
END COMPONENT;

COMPONENT decoder4_9
PORT(
LE : IN std_logic;
A0 : IN std_logic;
A1 : IN std_logic;
A2 : IN std_logic;
A3 : IN std_logic;
Q0 : OUT std_logic;
Q1 : OUT std_logic;
Q2 : OUT std_logic;
Q3 : OUT std_logic;
Q4 : OUT std_logic;
Q5 : OUT std_logic;
Q6 : OUT std_logic;
Q7 : OUT std_logic;
Q8 : OUT std_logic
);
END COMPONENT;

signal load_reg0, load_reg1, load_reg2, load_reg3, load_reg4, load_reg5, load_re
signal reg0_q, reg1_q, reg2_q, reg3_q, reg4_q, reg5_q, reg6_q, reg7_q, reg8_q :


begin

reg00: reg16 PORT MAP(

D => D_data,
```

```
load => load_reg0,
Clk => Clk,
Q => reg0_q

);

reg01: reg16 PORT MAP(

D => D_data,
load => load_reg1,
Clk => Clk,
Q => reg1_q

);

reg02: reg16 PORT MAP(

D => D_data,
load => load_reg2,
Clk => Clk,
Q => reg2_q

);

reg03: reg16 PORT MAP(

D => D_data,
load => load_reg3,
Clk => Clk,
Q => reg3_q

);

reg04: reg16 PORT MAP(

D => D_data,
load => load_reg4,
Clk => Clk,
Q => reg4_q

);
```

```
reg05: reg16 PORT MAP(

D => D_data,
load => load_reg5,
Clk => Clk,
Q => reg5_q

);

reg06: reg16 PORT MAP(

D => D_data,
load => load_reg6,
Clk => Clk,
Q => reg6_q

);

reg07: reg16 PORT MAP(

D => D_data,
load => load_reg7,
Clk => Clk,
Q => reg7_q

);

reg08: reg16 PORT MAP(

D => D_data,
load => load_reg8,
Clk => Clk,
Q => reg8_q

);


decoder_4_9: decoder4_9 PORT MAP(
LE => LE,
```

```
A0 => des_A0,
A1 => des_A1,
A2 => des_A2,
A3 => des_A3,

Q0 => load_reg0,
Q1 => load_reg1,
Q2 => load_reg2,
Q3 => load_reg3,
Q4 => load_reg4,
Q5 => load_reg5,
Q6 => load_reg6,
Q7 => load_reg7,
Q8 => load_reg8

);


A_mux: mux9_16 PORT MAP(

In0 => reg0_q,
In1 => reg1_q,
In2 => reg2_q,
In3 => reg3_q,
In4 => reg4_q,
In5 => reg5_q,
In6 => reg6_q,
In7 => reg7_q,
In8 => reg8_q,

S0 => AS0,
S1 => AS1,
S2 => AS2,
S3 => AS3,
Z => A_data

);

B_mux: mux9_16 PORT MAP(

In0 => reg0_q,
```

```
In1 => reg1_q,
In2 => reg2_q,
In3 => reg3_q,
In4 => reg4_q,
In5 => reg5_q,
In6 => reg6_q,
In7 => reg7_q,
In8 => reg8_q,

S0 => BS0,
S1 => BS1,
S2 => BS2,
S3 => BS3,
Z => B_data

);

end Behavioral;
```

1.14 Project 2 Top Level

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity top_level is
  Port (
            reset: in std_logic;
            Clk: in std_logic
  );
end top_level;

architecture Behavioral of top_level is

COMPONENT Microprogrammed_Control Port(
   data_in: in std_logic_vector(15 downto 0);
   V: in std_logic;
   C: in std_logic;
   Z: in std_logic; --return 1 if 0
```

```
    N: in std_logic; --return MSB
    reset: in std_logic;
    Clk: in std_logic;
    pc_out: out std_logic_vector(15 downto 0);
    TD_DR: out std_logic_vector(3 downto 0);
    TA_SA: out std_logic_vector(3 downto 0);
    TB_SB: out std_logic_vector(3 downto 0);
    SB: out std_logic_vector(2 downto 0);
    mb: out std_logic;
    md: out std_logic;
    rw: out std_logic;
    mm: out std_logic;
    mw: out std_logic;
    fs: out std_logic_vector(4 downto 0)
);
END COMPONENT;

COMPONENT datapath_1B Port (

SB: in std_logic_vector(2 downto 0);
AS0: in std_logic;
AS1: in std_logic;
AS2: in std_logic;
AS3: in std_logic;

BS0: in std_logic;
BS1: in std_logic;
BS2: in std_logic;
BS3: in std_logic;

des_A0: in std_logic;
des_A1: in std_logic;
des_A2: in std_logic;
des_A3: in std_logic;

MB_Select: in std_logic;
MD_Select: in std_logic;

Clk: in std_logic;

LE: in std_logic;
```

```
FS: in std_logic_vector(4 downto 0);
Data_in: in std_logic_vector(15 downto 0);

MM: in std_logic;
pc_in: in std_logic_vector(15 downto 0);

Add_out : out std_logic_vector(15 downto 0); --Address out
B_out : out std_logic_vector(15 downto 0); --Data Out

V: out std_logic;
C: out std_logic;
Z: out std_logic;
N: out std_logic
);
END COMPONENT;

COMPONENT Memory port (
            data_in : in std_logic_vector(15 downto 0);
            address_in : in std_logic_vector(15 downto 0);
            mw: in std_logic;
            Clk: in std_logic;
            data_out : out std_logic_vector(15 downto 0)
);

END COMPONENT;


--MC outputs



    signal pc_out:  std_logic_vector(15 downto 0);
    signal TD_DR:  std_logic_vector(3 downto 0);
    signal TA_SA:  std_logic_vector(3 downto 0);
    signal TB_SB:  std_logic_vector(3 downto 0);
    signal SB: std_logic_vector(2 downto 0);
    signal mb:  std_logic;
    signal md:  std_logic;
    signal rw:  std_logic;
    signal mm:  std_logic;
    signal mw:  std_logic;
```

```
    signal fs:  std_logic_vector(4 downto 0);

--Datapath outputs

    signal Add_out :  std_logic_vector(15 downto 0);
    signal B_out :  std_logic_vector(15 downto 0);

    signal V:  std_logic;
    signal C:  std_logic;
    signal Z:  std_logic;
    signal N:  std_logic;

--Memory Outputs
    signal data_out : std_logic_vector(15 downto 0);


begin

mc : Microprogrammed_Control Port Map(
    Clk =>Clk,
    reset => reset,
    data_in =>data_out,
    V=>V,
    C=>C,
    Z=>Z,
    N=>N,
    pc_out=>pc_out,
    TD_DR => TD_DR,
    TA_SA=>TA_SA,
    TB_SB =>TB_SB,
    SB=>SB,
    mb=>mb,
    md =>md,
    rw=>rw,
    mm=>mm,
    mw=>mw,
    fs=>fs
);

dpb : datapath_1B Port Map(
    Clk =>Clk,
```

```
    Add_out => Add_out,
    B_out => B_out,
    V=>V,
    C=>C,
    Z=>Z,
    N=>N,
    SB=>SB,
    AS0=>TA_SA(0),
    AS1=>TA_SA(1),
    AS2=>TA_SA(2),
    AS3=>TA_SA(3),
    BS0=>TB_SB(0),
    BS1=>TB_SB(1),
    BS2=>TB_SB(2),
    BS3=>TB_SB(3),
    des_a0=>TD_DR(0),
    des_a1=>TD_DR(1),
    des_a2=>TD_DR(2),
    des_a3=>TD_DR(3),
    mb_select=>mb,
    md_select=>md,
    le=>rw,
    fs=>fs,
    data_in=>data_out,
    mm=>mm,
    pc_in=>pc_out
);

mem : Memory Port Map (
    Clk => Clk,
    address_in =>Add_out,
    data_in=>B_out,
    mw=>mw,
    data_out =>data_out
);
end Behavioral;
```

1.15 Zero Fill

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity zero_fill is
    Port (
        SB : in  STD_LOGIC_VECTOR (2 downto 0);
            constant_out : out STD_LOGIC_VECTOR (15 downto 0));
end zero_fill;

architecture Behavioral of zero_fill is

begin
constant_out(2 downto 0) <= SB;
constant_out(15 downto 3) <= "0000000000000";
end Behavioral;
```

2 Component Test Benches

2.1 Control Address Register

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_car IS
END tb_car;

ARCHITECTURE behavior OF tb_car IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT CAR
    Port(        Cmux : in STD_LOGIC_VECTOR(7 downto 0);
                        Smux, reset, Clk : in STD_LOGIC;
                        car_out : out STD_LOGIC_VECTOR(7 downto 0)
                        );
    END COMPONENT;


   --Inputs
   signal Cmux : std_logic_vector(7 downto 0) := (others => '0');
   signal Smux : std_logic := '0';
   signal reset : std_logic := '0';
   signal Clk : std_logic := '0';
         --Outputs
   signal car_out : std_logic_vector(7 downto 0);

BEGIN
 clk_process :process
   begin
                Clk <= '0';
                wait for 5ns;
                Clk <= '1';
                wait for 5ns;
   end process;


        -- Instantiate the Unit Under Test (UUT)
```

```
    uut: CAR PORT MAP (
           Cmux => Cmux,
           Smux => Smux,
           reset => reset,
           car_out => car_out,
           Clk => Clk
         );

    -- Stimulus process
    stim_proc: process
    begin
                  wait for 5ns;
                  reset <= '1';

                  wait for 30ns;
                  reset <= '0';

                  wait for 30ns;
                  Cmux <= x"01";

                  wait for 30ns;
                  Cmux <= x"DA";
                  Smux <= '1';

       wait;
    end process;

END;
```

2.2 Control Memory

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_ControlMemory IS
END tb_ControlMemory;

ARCHITECTURE behavior OF tb_ControlMemory IS

    -- Component Declaration for the Unit Under Test (UUT)
```

```
COMPONENT ControlMemory
PORT(
     in_car : IN  std_logic_vector(7 downto 0);
     MW : OUT  std_logic;
     MM : OUT  std_logic;
     RW : OUT  std_logic;
     MD : OUT  std_logic;
     MB : OUT  std_logic;
     TB : OUT  std_logic;
     TA : OUT  std_logic;
     TD : OUT  std_logic;
     PL : OUT  std_logic;
     PI : OUT  std_logic;
     IL : OUT  std_logic;
     MC : OUT  std_logic;
     FS_cm : OUT  std_logic_vector(4 downto 0);
     MS_cm : OUT  std_logic_vector(2 downto 0);
     NA : OUT  std_logic_vector(7 downto 0)
    );
END COMPONENT;


--Inputs
signal in_car : std_logic_vector(7 downto 0) := (others => '0');

      --Outputs
signal MW : std_logic;
signal MM : std_logic;
signal RW : std_logic;
signal MD : std_logic;
signal MB : std_logic;
signal TB : std_logic;
signal TA : std_logic;
signal TD : std_logic;
signal PL : std_logic;
signal PI : std_logic;
signal IL : std_logic;
signal MC : std_logic;
signal FS_cm : std_logic_vector(4 downto 0);
signal MS_cm : std_logic_vector(2 downto 0);
signal NA : std_logic_vector(7 downto 0);
```

```
BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: ControlMemory PORT MAP (
          in_car => in_car,
          MW => MW,
          MM => MM,
          RW => RW,
          MD => MD,
          MB => MB,
          TB => TB,
          TA => TA,
          TD => TD,
          PL => PL,
          PI => PI,
          IL => IL,
          MC => MC,
          FS_cm => FS_cm,
          MS_cm => MS_cm,
          NA => NA
        );

   -- Stimulus process
   stim_proc: process
   begin
                wait for 10ns;
                in_car <= x"00";

                wait for 10ns;
                in_car <= x"08";

                wait for 10ns;
                in_car <= x"0D";

                wait;
   end process;

END;
```

2.3 Decoder 4 to 9

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity tb4_9dec is

end tb4_9dec;

architecture Behavioral of tb4_9dec is
COMPONENT decoder4_9
PORT(
LE : IN std_logic;
A0 : IN std_logic;
A1 : IN std_logic;
A2 : IN std_logic;
A3 : IN std_logic;
Q0 : OUT std_logic;
Q1 : OUT std_logic;
Q2 : OUT std_logic;
Q3 : OUT std_logic;
Q4 : OUT std_logic;
Q5 : OUT std_logic;
Q6 : OUT std_logic;
Q7 : OUT std_logic;
Q8 : OUT std_logic
);
END COMPONENT;

     --Inputs
   signal LE : std_logic := '1';

   signal A0 : std_logic := '0';
   signal A1 : std_logic := '0';
   signal A2 : std_logic := '0';
   signal A3 : std_logic := '0';


         --Outputs
   signal Q0 : std_logic;
```

```
   signal Q1 : std_logic;
   signal Q2 : std_logic;
   signal Q3 : std_logic;
   signal Q4 : std_logic;
   signal Q5 : std_logic;
   signal Q6 : std_logic;
   signal Q7 : std_logic;
   signal Q8 : std_logic;

begin
--Unit Under Testing
uut: decoder4_9 PORT MAP (
          LE => LE,
          Q0 => Q0,
          Q1 => Q1,
          Q2 => Q2,
          Q3 => Q3,
          Q4 => Q4,
          Q5 => Q5,
          Q6 => Q6,
          Q7 => Q7,
          Q8 => Q8,

          A0 => A0,
          A1 => A1,
          A2 => A2,
          A3 => A3

        );
stim_proc: process
    begin
                wait for 10ns;
                A0 <= '0';
                A1 <= '0';
                A2 <= '0';
                A3 <= '0';

        wait for 10ns;
                A0 <= '0';
                A1 <= '0';
                A2 <= '0';
```

```
        A3 <= '1';

wait for 10ns;
        A0 <= '0';
        A1 <= '0';
        A2 <= '1';
        A3 <= '0';

wait for 10ns;
        A0 <= '0';
        A1 <= '0';
        A2 <= '1';
        A3 <= '1';

wait for 10ns;
        A0 <= '0';
        A1 <= '1';
        A2 <= '0';
        A3 <= '0';

wait for 10ns;
        A0 <= '0';
        A1 <= '1';
        A2 <= '0';
        A3 <= '1';

wait for 10ns;
        A0 <= '0';
        A1 <= '1';
        A2 <= '1';
        A3 <= '0';

wait for 10ns;
        A0 <= '0';
        A1 <= '1';
        A2 <= '1';
        A3 <= '1';

wait for 10ns;
        A0 <= '1';
        A1 <= '0';
```

```
                A2 <= '0';
                A3 <= '0';


    end process;
end Behavioral;
```

2.4 Extender to Program Counter

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_extendo IS
END tb_extendo;

ARCHITECTURE behavior OF tb_extendo IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT E_PC
    PORT(
         SR_SB : IN  std_logic_vector(5 downto 0);
         signed_extension : OUT  std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal SR_SB : std_logic_vector(5 downto 0) := (others => '0');

         --Outputs
   signal signed_extension : std_logic_vector(15 downto 0);

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: E_PC PORT MAP (
          SR_SB => SR_SB,
          signed_extension => signed_extension
        );
```

```
   -- Stimulus process
   stim_proc: process
   begin
                wait for 50ns;
                SR_SB <= "000001";

                wait for 50ns;
                SR_SB <= "100001";

      wait;
   end process;

END;
```

2.5 Instruction Register

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_ir IS
END tb_ir;

ARCHITECTURE behavior OF tb_ir IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT InstructionR
    Port(        IR : in STD_LOGIC_VECTOR(15 downto 0);
                      IL, Clk : in STD_LOGIC;
                      OPCODE :  out STD_LOGIC_VECTOR(6 downto 0);
                      DR, SA, SB : out STD_LOGIC_VECTOR(2 downto 0)
                      );
    END COMPONENT;


   --Inputs
   signal IR : std_logic_vector(15 downto 0) := (others => '0');
   signal IL : std_logic := '0';
   signal Clk : std_logic := '0';
         --Outputs
```

```
   signal OPCODE : std_logic_vector(6 downto 0);
   signal DR : std_logic_vector(2 downto 0);
   signal SA : std_logic_vector(2 downto 0);
   signal SB : std_logic_vector(2 downto 0);

BEGIN
 clk_process :process
   begin
                Clk <= '0';
                wait for 5ns;
                Clk <= '1';
                wait for 5ns;
   end process;
        -- Instantiate the Unit Under Test (UUT)
   uut: InstructionR PORT MAP (
          IR => IR,
          IL => IL,
          OPCODE => OPCODE,
          DR => DR,
          SA => SA,
          SB => SB,
          Clk => Clk
        );

   -- Stimulus process
   stim_proc: process
   begin
                wait for 10ns;
                IR <= "1111111000010001";

                wait for 5ns;
                IL <= '1';

                wait for 10ns;
                IR <= "0000000100010000";
                IL <= '0';

                wait for 5ns;
                IL <= '1';

      wait;
```

```
   end process;

END;
```

2.6 Memory

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_mem_module IS
END tb_mem_module;

ARCHITECTURE behavior OF tb_mem_module IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT Memory
    Port (
            data_in : in std_logic_vector(15 downto 0);
            address_in : in std_logic_vector(15 downto 0);
            mw: in std_logic;
            Clk : in std_logic;
            data_out : out std_logic_vector(15 downto 0)
            );
    END COMPONENT;


   --Inputs
   signal address_in : std_logic_vector(15 downto 0) := (others => '0');
   signal data_in : std_logic_vector(15 downto 0) := (others => '0');
   signal mw : std_logic := '0';
   signal Clk : std_logic := '0';

         --Outputs
   signal data_out : std_logic_vector(15 downto 0);

BEGIN
  clk_process :process
   begin
                Clk <= '0';
                wait for 5ns;
```

```
            Clk <= '1';
            wait for 5ns;
end process;

     -- Instantiate the Unit Under Test (UUT)
uut: Memory PORT MAP (
         Clk => Clk,
         address_in => address_in,
         data_in => data_in,
         mw => mw,
         data_out => data_out
     );

-- Stimulus process
stim_proc: process
begin

     address_in <= x"0000";


            wait for 10ns;
            mw <='1';
            data_in <= x"0E00";

            wait for 10 ns;
            mw <= '0';
            wait for 20ns;

            address_in <= x"0000";

            wait for 20ns;
            address_in <= x"0006";

            wait for 20ns;
            address_in <= x"0007";

            wait for 20ns;
            address_in <= x"0007";
```

```
      wait;
   end process;

END;
```

2.7 Program Counter

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_pc IS
END tb_pc;

ARCHITECTURE behavior OF tb_pc IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT PCreg
    PORT(
         seAD : in STD_LOGIC_VECTOR(15 downto 0);
         PL : IN  std_logic;
         PI : IN  std_logic;
         reset : IN  std_logic;
         Clk : IN std_logic;
         PC : out STD_LOGIC_VECTOR(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal seAD : std_logic_vector(15 downto 0) := (others => '0');
   signal PL : std_logic := '0';
   signal PI : std_logic := '0';
   signal reset : std_logic := '0';
   signal Clk : std_logic := '0';
         --Outputs
   signal PC : std_logic_vector(15 downto 0);

BEGIN
    clk_process :process
   begin
```

```
                Clk <= '0';
                wait for 5ns;
                Clk <= '1';
                wait for 5ns;
    end process;


        -- Instantiate the Unit Under Test (UUT)
    uut: PCreg PORT MAP (
           seAD => seAD,
           PL => PL,
           PI => PI,
           reset => reset,
           Clk => Clk,
           PC => PC
         );

    -- Stimulus process
    stim_proc: process
    begin
                wait for 5ns;
                reset <= '1';
                seAD <= x"0000";

                wait for 5ns;
                reset <= '0';

                wait for 5ns;
                PI <= '1';
                seAD <= x"0002";

                wait for 20ns;
                PI <= '0';
                PL <= '1';
                seAD <= x"000F";

       wait;
    end process;

END;
```

2.8 Project 2 Top Level

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity tb_top is
end tb_top;

architecture Behavioral of tb_top is

COMPONENT top_level Port (
            reset: in std_logic;
            Clk: in std_logic
  );
END COMPONENT;

signal Clk :std_logic := '0';
signal reset : std_logic := '0';

begin
   clk_process :process
   begin
                Clk <= '0';
                wait for 5ns;
                Clk <= '1';
                wait for 5ns;
   end process;

   uut: top_level PORT MAP (
        Clk=>Clk,
        reset=>reset
   );

   stim_proc: process
   begin
   wait for 200 ns;
   reset <= '0';

   wait for 200 ns ;
   reset <= '1';
```

```
   wait for 200 ns;
   reset <='0';

   end process;

end Behavioral;
```

2.9 Register File

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;


ENTITY tbreg_file IS
END tbreg_file;

ARCHITECTURE behavior OF tbreg_file IS


COMPONENT reg_file
PORT(
LE : in std_logic;

AS0 : in std_logic;
AS1 : in std_logic;
AS2 : in std_logic;
AS3 : in std_logic;

BS0 : in std_logic;
BS1 : in std_logic;
BS2 : in std_logic;
BS3 : in std_logic;

des_A0 : in std_logic;
des_A1 : in std_logic;
des_A2 : in std_logic;
des_A3 : in std_logic;

Clk : in std_logic;

D_data : in std_logic_vector(15 downto 0);
```

```
B_data : out std_logic_vector(15 downto 0);
A_data : out std_logic_vector(15 downto 0)

        );
    END COMPONENT;


        --Inputs
        signal LE : std_logic := '1';

        signal AS0 : std_logic := '0';
        signal AS1 : std_logic := '0';
        signal AS2 : std_logic := '0';
        signal AS3 : std_logic := '0';

        signal BS0 : std_logic := '0';
        signal BS1 : std_logic := '0';
        signal BS2 : std_logic := '0';
        signal BS3 : std_logic := '0';


        signal des_A0 : std_logic := '0';
        signal des_A1 : std_logic := '0';
        signal des_A2 : std_logic := '0';
        signal des_A3 : std_logic := '0';


        signal data_src : std_logic := '0';
        signal Clk : std_logic := '0';
        signal D_data : std_logic_vector(15 downto 0) := (others => '0');

        --Outputs

        signal A_data : std_logic_vector(15 downto 0);
        signal B_data : std_logic_vector(15 downto 0);


BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: reg_file PORT MAP (
```

```
        LE => LE,

        AS0 => AS0,
        AS1 => AS1,
        AS2 => AS2,
        AS3 => AS3,

        BS0 => BS0,
        BS1 => BS1,
        BS2 => BS2,
        BS3 => BS3,

        des_A0 => des_A0,
        des_A1 => des_A1,
        des_A2 => des_A2,
        des_A3 => des_A3,

        Clk => Clk,
        D_data => D_data,
        A_data => A_data,
        B_data => B_data

    );

-- Clock process definitions
clk_process :process
begin
            Clk <= '0';
            wait for 5ns;
            Clk <= '1';
            wait for 5ns;
end process;

-- Stimulus process
stim_proc: process
begin
            wait for 10ns;
            des_a1 <= '0';
            des_a2 <= '0';
            des_a3 <= '0';
            D_data <= x"0000";
```

```
                wait for 10ns;
                des_a1 <= '0';
                des_a2 <= '0';
                des_a3 <= '1';
                D_data <= x"1111";

                wait for 10ns;
                des_a1 <= '0';
                des_a2 <= '1';
                des_a3 <= '0';
                D_data <= x"2222";

                wait for 10ns;
                AS2 <= '1';
                AS3 <= '1';
                des_a1 <= '0';
                des_a2 <= '1';
                des_a3 <= '1';
                D_data <= x"3333";

                wait for 10ns;
                AS2 <= '0';
                AS3 <= '0';
                des_a1 <= '1';
                des_a2 <= '0';
                des_a3 <= '0';
                D_data <= x"4444";

                wait for 10ns;
                des_a0 <= '0';
                des_a1 <= '0';
                des_a2 <= '0';
                des_a3 <= '0';
                D_data <= x"5555";

    end process;

END;
```

2.10 Zero Fill

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY zf_tb IS
END zf_tb;

ARCHITECTURE behavior OF zf_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT zero_fill
    PORT(
         SB : IN  std_logic_vector(2 downto 0);
         constant_out : OUT  std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal SB : std_logic_vector(2 downto 0) := (others => '0');

         --Outputs
   signal constant_out : std_logic_vector(15 downto 0);

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: zero_fill PORT MAP (
          SB => SB,
          constant_out => constant_out
        );

   -- Stimulus process
   stim_proc: process
   begin
                wait for 10ns;
                SB <= "110";
                wait;
   end process;
```

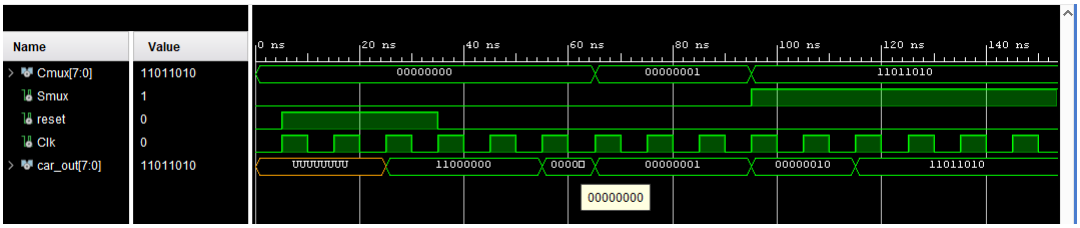
```
END;
```

3 Testbench Results

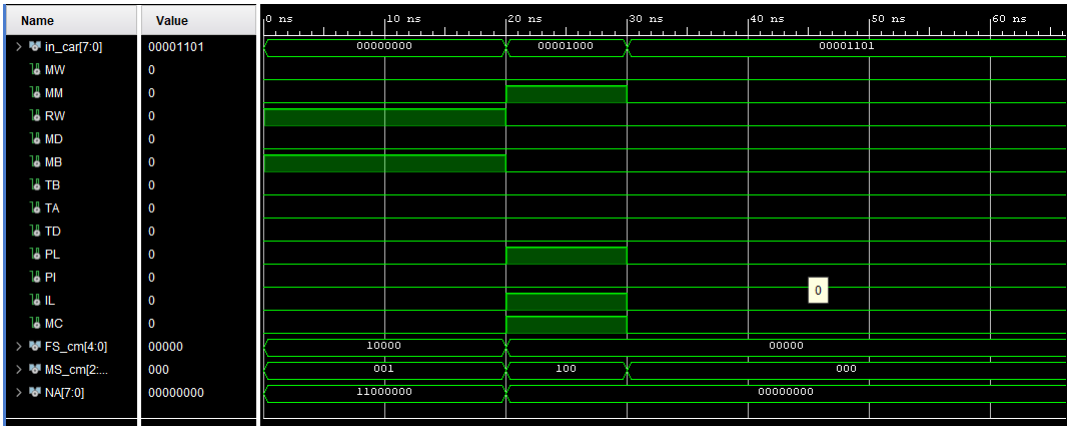
The testbenches show the results of the operands performed by each component within project. The top level shows the progression of data outputs over time. Each module functions as described in documentation, and test benches for the CPU datapaths, logic, and memory seem to pertain to what results should be for operands undergoing various operations. The register file has been modified with an additional register, with selection bits increased from 3 to 4 to accommodate for it. All operating components use 16-bit size inputs and outputs. The memory module - external RAM and control memory each have 16-bit size inputs that utilise 9 least significant bits of the values provided in order to traverse the memory array.

Microcoded operations have been implemented for control memory as per the table provided in lecture notes. Each change and resulting operation in time via the register testbench is reflected by all the instructions - namely ADI, LDR, STR, ADI, INC, NOT, B, BXX. These instructions hold various flags to parameterize the components in the project, to accomplish their described operation through modifying the contents of registers over time and provide directed instructions to each part of the project.

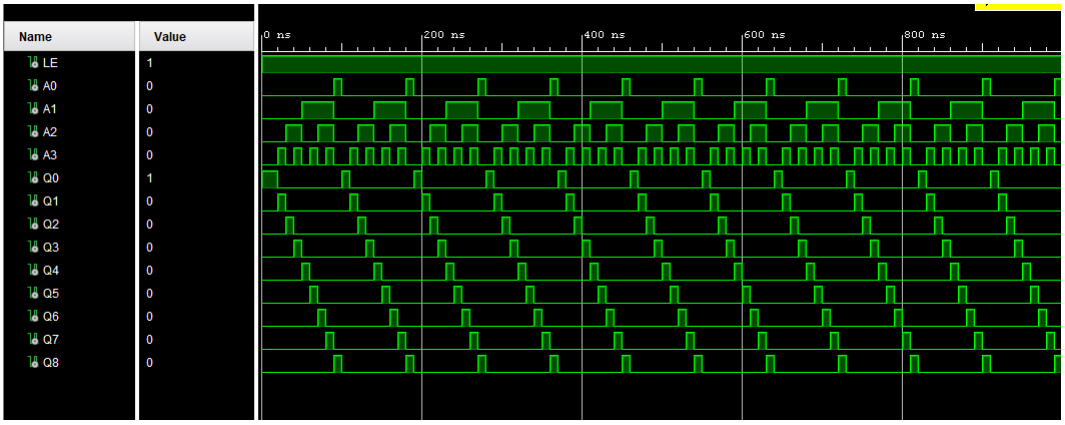
3.1 Control Address Register



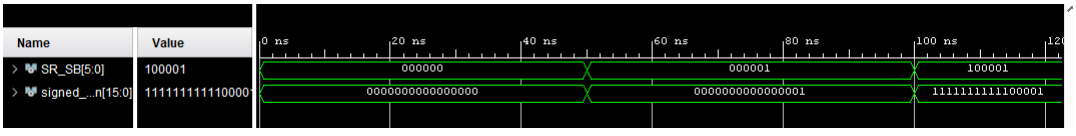
3.2 Control Memory



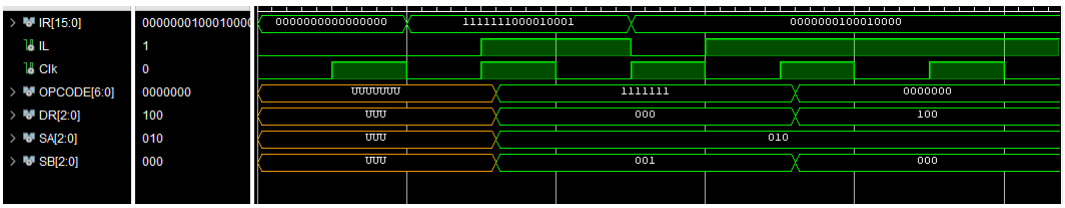
3.3 Decoder



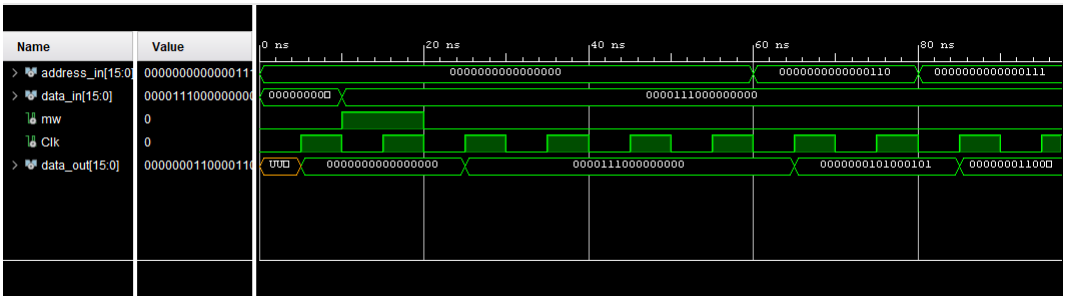
3.4 Extender



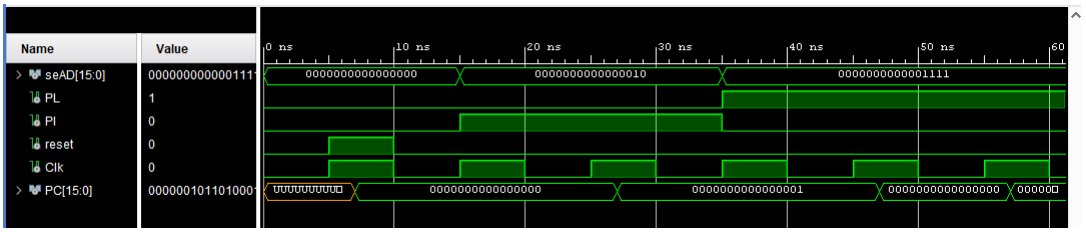
3.5 Instruction Register



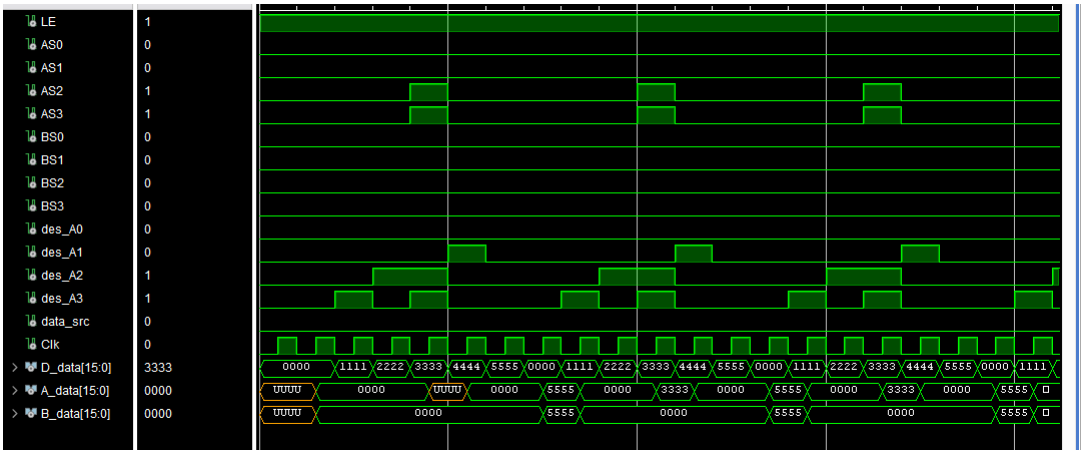
3.6 Memory



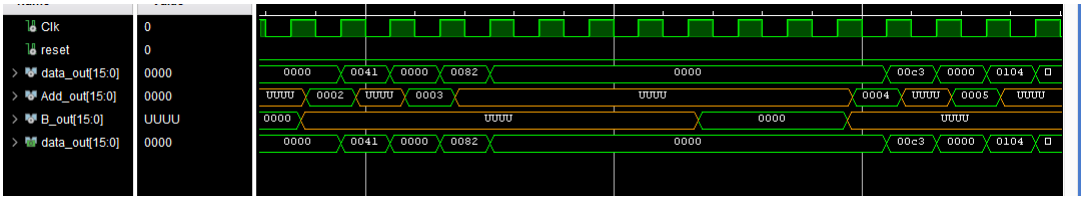
3.7 Program Counter



3.8 Register File



3.9 Top Level Project



3.10 Zero Fill