ELEC 374 – PHASE 1 REPORT

Group 2

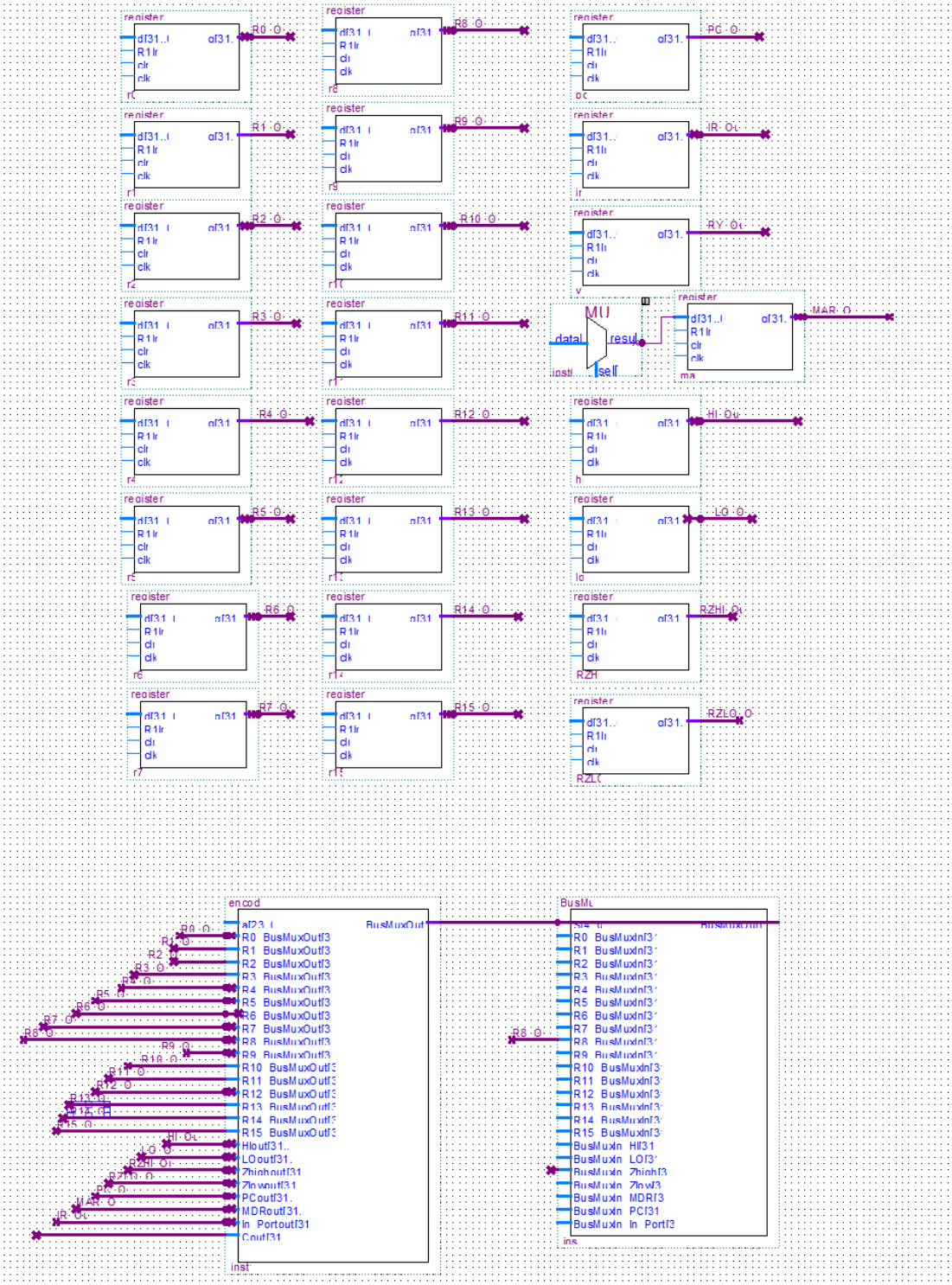
Tom Heysel

Allison Christensen

Matthieu Roux

Simon Day

Current Processor Diagram



VHDL Code

32-5 Bus Encoder

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity encoder is
 port(
    a : in STD_LOGIC_VECTOR(23 downto 0);
    BusMuxOut : out STD_LOGIC_VECTOR(4 downto 0);

    -- Register Inputs
    R0_BusMuxOut  : IN STD_LOGIC_VECTOR(31 downto 0);
    R1_BusMuxOut  : IN STD_LOGIC_VECTOR(31 downto 0);
    R2_BusMuxOut  : IN STD_LOGIC_VECTOR(31 downto 0);
    R3_BusMuxOut  : IN STD_LOGIC_VECTOR(31 downto 0);
    R4_BusMuxOut  : IN STD_LOGIC_VECTOR(31 downto 0);
    R5_BusMuxOut  : IN STD_LOGIC_VECTOR(31 downto 0);
    R6_BusMuxOut  : IN STD_LOGIC_VECTOR(31 downto 0);
    R7_BusMuxOut  : IN STD_LOGIC_VECTOR(31 downto 0);
    R8_BusMuxOut  : IN STD_LOGIC_VECTOR(31 downto 0);
    R9_BusMuxOut  : IN STD_LOGIC_VECTOR(31 downto 0);
    R10_BusMuxOut : IN STD_LOGIC_VECTOR(31 downto 0);
    R11_BusMuxOut : IN STD_LOGIC_VECTOR(31 downto 0);
    R12_BusMuxOut : IN STD_LOGIC_VECTOR(31 downto 0);
    R13_BusMuxOut : IN STD_LOGIC_VECTOR(31 downto 0);
    R14_BusMuxOut : IN STD_LOGIC_VECTOR(31 downto 0);
    R15_BusMuxOut : IN STD_LOGIC_VECTOR(31 downto 0);

    HIout         : IN STD_LOGIC_VECTOR(31 downto 0);
    LOout         : IN STD_LOGIC_VECTOR(31 downto 0);
    Zhighout      : IN STD_LOGIC_VECTOR(31 downto 0);
    Zlowout       : IN STD_LOGIC_VECTOR(31 downto 0);
    PCout         : IN STD_LOGIC_VECTOR(31 downto 0);
    MDRout        : IN STD_LOGIC_VECTOR(31 downto 0);
    In_Portout    : IN STD_LOGIC_VECTOR(31 downto 0);
    Cout          : IN STD_LOGIC_VECTOR(31 downto 0)
 );
end encoder;

architecture bhv of encoder is
begin
```

```
process(a,
                R0_BusMuxOut ,
                R1_BusMuxOut ,
                R2_BusMuxOut ,
                R3_BusMuxOut ,
                R4_BusMuxOut ,
                R5_BusMuxOut ,
                R6_BusMuxOut ,
                R7_BusMuxOut ,
                R8_BusMuxOut ,
                R9_BusMuxOut ,
                R10_BusMuxOut,
                R11_BusMuxOut,
                R12_BusMuxOut,
                R13_BusMuxOut,
                R14_BusMuxOut,
                R15_BusMuxOut,
                HIout,
                LOout,
                Zhighout,
                Zlowout,
                PCout,
                MDRout,
                In_Portout)

begin
 case a is
     when R0_BusMuxOut => BusMuxOut <= "00000";
     when R1_BusMuxOut    => BusMuxOut <= "00001";
     when R2_BusMuxOut    => BusMuxOut <= "00010";
     when R3_BusMuxOut    => BusMuxOut <= "00011";
     when R4_BusMuxOut    => BusMuxOut <= "00100";
     when R5_BusMuxOut    => BusMuxOut <= "00101";
     when R6_BusMuxOut    => BusMuxOut <= "00110";
     when R7_BusMuxOut    => BusMuxOut <= "00111";
     when R8_BusMuxOut    => BusMuxOut <= "01000";
     when R9_BusMuxOut    => BusMuxOut <= "01001";
     when R10_BusMuxOut   => BusMuxOut <= "01010";
     when R11_BusMuxOut   => BusMuxOut <= "01011";
     when R12_BusMuxOut   => BusMuxOut <= "01100";
     when R13_BusMuxOut   => BusMuxOut <= "01101";
     when R14_BusMuxOut   => BusMuxOut <= "01110";
     when R15_BusMuxOut   => BusMuxOut <= "01111";
     when HIout           => BusMuxOut <= "10000";
     when LOout           => BusMuxOut <= "10001";
```

```
     when Zhighout        => BusMuxOut <= "10010";
     when Zlowout         => BusMuxOut <= "10011";
     when PCout           => BusMuxOut <= "10100";
     when MDRout          => BusMuxOut <= "10101";
     when In_Portout      => BusMuxOut <= "10110";
     when others => BusMuxOut <= "XXXXX";

 end case;
end process;

end bhv;
```

General Purpose Register

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

ENTITY register32 IS PORT(
    d    : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    R1In : IN STD_LOGIC; -- load/enable.
    clr  : IN STD_LOGIC; -- async. clear.
    clk  : IN STD_LOGIC; -- clock.
    q    : OUT STD_LOGIC_VECTOR(31 DOWNTO 0) -- output
);
END register32;

ARCHITECTURE description OF register32 IS

BEGIN
    process(clk, clr)
    begin
        if clr = '1' then
            q <= "00000000";
        elsif rising_edge(clk) then
            if R1In = '1' then
                q <= d;
            end if;
        end if;
    end process;
END description;
```

2-1 MUX for MDR

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

entity mux is port
( in1   :in std_logic_vector(31 downto 0);
  in2   :in std_logic_vector(31 downto 0);
  cs:in std_logic;
  output :OUT std_logic_vector(31 downto 0)
);
END mux;

ARCHITECTURE description OF mux IS

BEGIN
    process(in1, in2, cs)
    begin
        if cs = '0' then
            output <= in1;
        elsif cs = '1' then
            output <= in2;
        end if;
    end process;
END description;
```

BUS MUX

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;


--  Entity Declaration

ENTITY BusMux IS
-- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
PORT
(
S : IN STD_LOGIC_VECTOR(4 downto 0);
-- Register Inputs
R0_BusMuxIn  : IN STD_LOGIC_VECTOR(31 downto 0);
R1_BusMuxIn  : IN STD_LOGIC_VECTOR(31 downto 0);
R2_BusMuxIn  : IN STD_LOGIC_VECTOR(31 downto 0);
R3_BusMuxIn  : IN STD_LOGIC_VECTOR(31 downto 0);
R4_BusMuxIn  : IN STD_LOGIC_VECTOR(31 downto 0);
R5_BusMuxIn  : IN STD_LOGIC_VECTOR(31 downto 0);
R6_BusMuxIn  : IN STD_LOGIC_VECTOR(31 downto 0);
R7_BusMuxIn  : IN STD_LOGIC_VECTOR(31 downto 0);
R8_BusMuxIn  : IN STD_LOGIC_VECTOR(31 downto 0);
R9_BusMuxIn  : IN STD_LOGIC_VECTOR(31 downto 0);
R10_BusMuxIn : IN STD_LOGIC_VECTOR(31 downto 0);
R11_BusMuxIn : IN STD_LOGIC_VECTOR(31 downto 0);
R12_BusMuxIn : IN STD_LOGIC_VECTOR(31 downto 0);
R13_BusMuxIn : IN STD_LOGIC_VECTOR(31 downto 0);
R14_BusMuxIn : IN STD_LOGIC_VECTOR(31 downto 0);
R15_BusMuxIn : IN STD_LOGIC_VECTOR(31 downto 0);

-- Other register Inputs
BusMuxIn_HI    : IN STD_LOGIC_VECTOR(31 downto 0);
BusMuxIn_LO    : IN STD_LOGIC_VECTOR(31 downto 0);
BusMuxIn_Zhigh : IN STD_LOGIC_VECTOR(31 downto 0);
BusMuxIn_Zlow  : IN STD_LOGIC_VECTOR(31 downto 0);
BusMuxIn_MDR   : IN STD_LOGIC_VECTOR(31 downto 0);
--Program Counter Input
BusMuxIn_PC       : IN STD_LOGIC_VECTOR(31 downto 0);
--???
BusMuxIn_In_Port  : IN STD_LOGIC_VECTOR(31 downto 0);

-- Output Port (What will go on the Bus)
BusMuxOut  : OUT STD_LOGIC_VECTOR(31 downto 0)
);
-- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
```

```
END BusMux;

architecture BusMux_architecture of BusMux is
begin
    process(
                R0_BusMuxIn,
                R1_BusMuxIn,
                R2_BusMuxIn,
                R3_BusMuxIn,
                R4_BusMuxIn,
                R5_BusMuxIn,
                R6_BusMuxIn,
                R7_BusMuxIn,
                R8_BusMuxIn,
                R9_BusMuxIn,
                R10_BusMuxIn,
                R11_BusMuxIn,
                R12_BusMuxIn,
                R13_BusMuxIn,
                R14_BusMuxIn,
                R15_BusMuxIn,
                BusMuxIn_HI ,
                BusMuxIn_LO ,
                BusMuxIn_Zhigh,
                BusMuxIn_Zlow ,
                BusMuxIn_PC ,
                BusMuxIn_In_Port,
                BusMuxIn_MDR )
    begin
            case S is
                 when "00000"  => BusMuxOut <= R0_BusMuxIn;
                 when "00001"  => BusMuxOut <= R1_BusMuxIn;
                 when "00010"  => BusMuxOut <= R2_BusMuxIn;
                 when "00011"  => BusMuxOut <= R3_BusMuxIn;
                 when "00100"  => BusMuxOut <= R4_BusMuxIn;
                 when "00101"  => BusMuxOut <= R5_BusMuxIn;
                 when "00110"  => BusMuxOut <= R6_BusMuxIn;
                 when "00111"  => BusMuxOut <= R7_BusMuxIn;
                 when "01000"  => BusMuxOut <= R8_BusMuxIn;
                 when "01001"  => BusMuxOut <= R9_BusMuxIn;
                 when "01010"  => BusMuxOut <= R10_BusMuxIn;
                 when "01011"  => BusMuxOut <= R11_BusMuxIn;
                 when "01100"  => BusMuxOut <= R12_BusMuxIn;
                 when "01101"  => BusMuxOut <= R13_BusMuxIn;
```

```
                when "01110"  => BusMuxOut <= R14_BusMuxIn;
                when "01111"  => BusMuxOut <= R15_BusMuxIn;
                when "10000"  => BusMuxOut <= BusMuxIn_HI;
                when "10001"  => BusMuxOut <= BusMuxIn_LO;
                when "10010"  => BusMuxOut <= BusMuxIn_Zhigh;
                when "10011"  => BusMuxOut <= BusMuxIn_Zlow;
                when "10100"  => BusMuxOut <= BusMuxIn_PC;
                when "10101"  => BusMuxOut <= BusMuxIn_MDR;
                when "10110"  => BusMuxOut <= BusMuxIn_In_Port;
                when others => BusMuxOut <= "XXXXX";
            end case;
    end process;
end architecture;
```

ALU

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

entity alu is
    Port (  input_a     : in signed(31 downto 0);
            input_b     : in signed(31 downto 0);
            selection   : in STD_LOGIC_VECTOR (2 downto 0);
            AluOut      : out signed(3 downto 0));
end alu;

architecture Behavioral of alu is
    begin
    process(input_a, input_b, selection)
        begin
        case selection is
            when "000" => AluOut<= input_a and input_b; -- AND gate
            when "001" => AluOut<= input_a or input_b;  -- OR gate
            when "010" => AluOut<= not input_a ;        -- NOT gate
            when "011" => AluOut<= input_a xor input_b; -- XOR gate
            when "100" => AluOut<= input_a + input_b;   -- addition
            when "101" => AluOut<= input_a - input_b;   -- subtraction
            when "110" => AluOut<= -- multiplication
            when "111" => AluOut<= -- division
            when others => NULL;
        end case;
    end process;
end Behavioral;
```

Booth Algorithm

```
LIBRARY ieee;

entity booth_multiplier is
    port (
        a_input :   in  std_logic_vector(31 downto 0);  -- Input A
        b_input :   in  std_logic_vector(31 downto 0);  -- Input B
        output  :   out std_logic_vector(63 downto 0);  -- Output
    );
end booth_multiplier

architecture booth_multiplier of booth_multiplier is
    begin
        process(a_input, b_input)
        variable result, temp, to_result :   std_logic_vector(63 downto 0);
        variable ADD, SUB, ZEROES        :   std_logic_vector(31 downto 0); --
Purpose of the se variables is given below
        begin
            ADD     := a_input;             -- Used for when A is to be added
            SUB     := (0 - a_input);       -- Used for when A is to be
subrstracted
            result  := x"0000000000000000"  -- Makes the result 0 at start
            ZEROES  := x"00000000"

            for i in 0 to 31 loop
                if (i =0) then
                    if (b_input(0) = 1) then
                        to_result(31 downto 0) := toSub;
                    end if;
                else
                    if(b_input(i) = '1' and b_input(i-1) = '0') then
                        to_result := ZEROES & SUB;
                    elsif (b_input(i) = '0' and b_input(i-1) = '0') then
                        to_result:= ZEROES & ADD;
                        else
                            to_result := ZEROES & ZEROES;
                        end if;
                    end if;

                --Sign Extension
                    if (to_result(31) = '1') then
                        to_result(63 downto 32) := x"FFFFFFFF";
                    elsif (to_result(31) = '0') then
```

```
                    to_result(63 downto 32) := x"00000000";
                end if;
                to_result := STD_LOGIC_VECTOR(SHIFT_LEFT(UNSIGNED(to_result),
i));
                result := result + to_result;

        end loop;
```

Test Benches

Had our team been able to compile our project successfully, we would have simulated our design using Modelsim. In the Phase 1 instruction set, we were given a sample testbench template in VHDL for the logical AND instruction which we would have been able to use for other instructions with some verification and revision. Our first steps in writing testbenches would be focused on clock generation, adjusting some of the constants in the template to match the target clock rates. With the new signals defined, we would proceed to modify the logic within some given process. The sequential statements inside a process operate on the values they are at immediately before the process begins. To connect these signals and test our device design, we would have started with a reset pulse and a short delay to allow us to see that the machine can be held in this idle state. As we move forward with the remainder of the project, following these steps in writing our testbench should allow us to test the basic functionality of our design and to examine the signals to see that they have been generated correctly.