Elec 374 – Phase 1 report

## Group 2

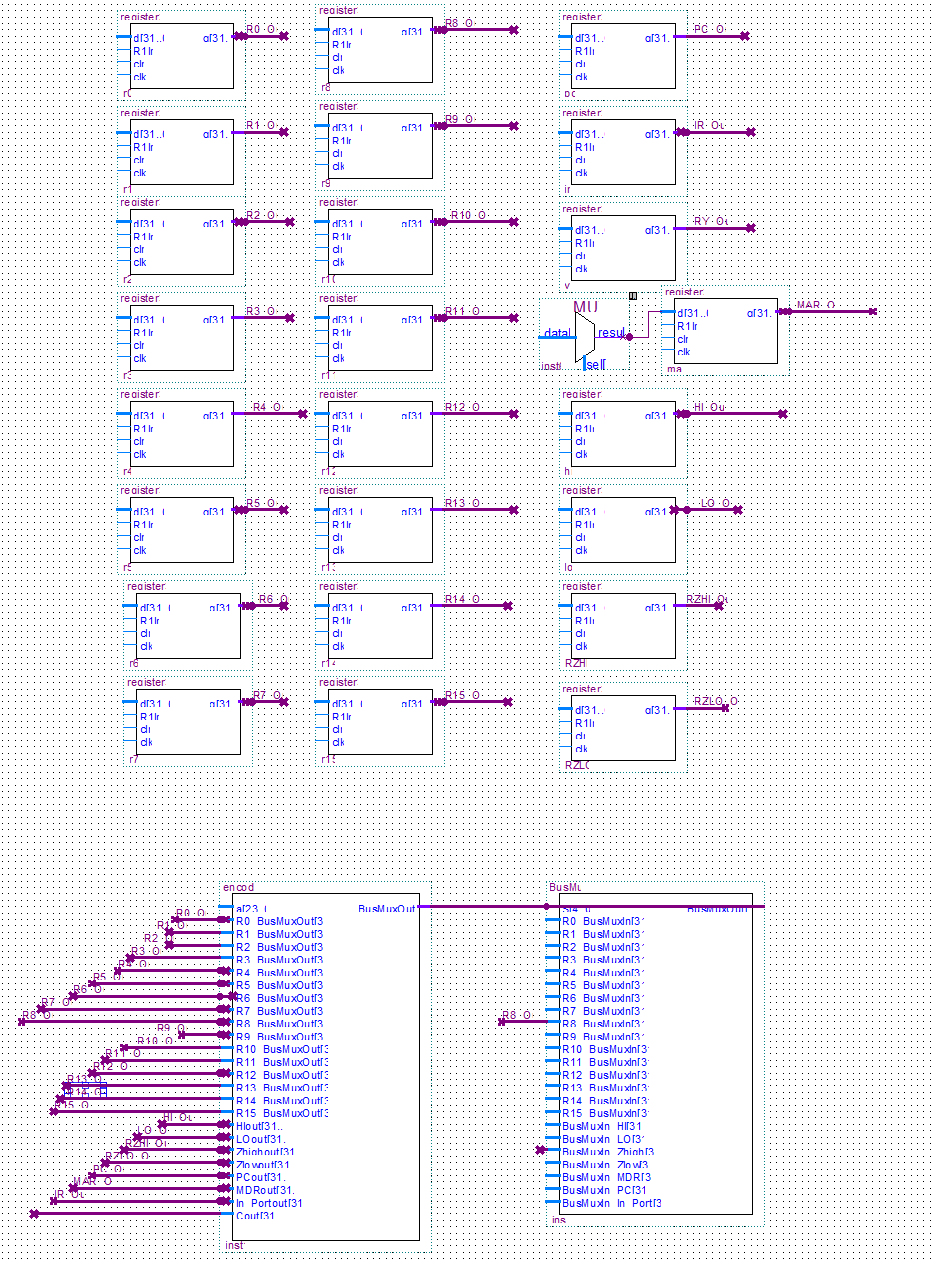
## Tom Heysel

## 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.