Mumtahid Akash

Professor Zheng Peng

CSC34300

17/04/2022

**Task1:**

**Introduction:**

The lab deals with the implementation and understanding of Instruction decoder and ALU Control unit. These are the sub module or sub functions of Mips 32. Basic task was to understand the concept of decoder and alu control unit. Below is the vhdl code of Instruction decoder and ALU control unit:

**VHDL CODE of Decoder:**

----------------------------------------------------------------------------------

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity DEC is

port(

I\_DEC\_EN : in STD\_LOGIC;

I\_DEC\_OPCODE: in STD\_LOGIC\_VECTOR(5 downto 0);

O\_DEC\_REGDST : out STD\_LOGIC;

O\_DEC\_JUMP : out STD\_LOGIC;

O\_DEC\_BEQ : out STD\_LOGIC;

O\_DEC\_BNE : out STD\_LOGIC;

O\_DEC\_MEMREAD : out STD\_LOGIC;

O\_DEC\_MEMTOREG : out STD\_LOGIC;

O\_DEC\_ALUOP : out STD\_LOGIC\_VECTOR(1 downto 0);

O\_DEC\_MEMWRITE : out STD\_LOGIC;

O\_DEC\_ALUSRC: out STD\_LOGIC;

O\_DEC\_REGWRITE: out STD\_LOGIC

);

end DEC;

architecture Behavioral of DEC is

begin

O\_DEC\_REGDST <= '1' when (I\_DEC\_OPCODE="000000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="001000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000100" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000101" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000010" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="100011" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="101011" and I\_DEC\_EN='1') else '0';

O\_DEC\_JUMP <= '0' when (I\_DEC\_OPCODE="000000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="001000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000100" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000101" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="000010" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="100011" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="101011" and I\_DEC\_EN='1') else '0';

O\_DEC\_BEQ <= '0' when (I\_DEC\_OPCODE="000000" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="001000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000100" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="000101" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000010" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="100011" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="101011" and I\_DEC\_EN='1') else '0';

O\_DEC\_BNE <= '0' when (I\_DEC\_OPCODE="000000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="001000" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="000100" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="000101" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000010" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="100011" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="101011" and I\_DEC\_EN='1') else '0';

O\_DEC\_MEMREAD <= '0' when (I\_DEC\_OPCODE="000000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="001000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000100" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000101" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000010" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="100011" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="101011" and I\_DEC\_EN='1') else '0';

O\_DEC\_MEMTOREG <= '0' when (I\_DEC\_OPCODE="000000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="001000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000100" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000101" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000010" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="100011" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="101011" and I\_DEC\_EN='1') else '0';

O\_DEC\_MEMWRITE <= '0' when (I\_DEC\_OPCODE="000000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="001000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000100" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000101" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000010" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="100011" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="101011" and I\_DEC\_EN='1') else '0';

O\_DEC\_ALUSRC <= '0' when (I\_DEC\_OPCODE="000000" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="001000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000100" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000101" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000010" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="100011" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="101011" and I\_DEC\_EN='1') else '0';

O\_DEC\_REGWRITE <= '1' when (I\_DEC\_OPCODE="000000" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="001000" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000100" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000101" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="000010" and I\_DEC\_EN='1') else

'1' when (I\_DEC\_OPCODE="100011" and I\_DEC\_EN='1') else

'0' when (I\_DEC\_OPCODE="101011" and I\_DEC\_EN='1') else '0';

O\_DEC\_ALUOP <= "10" when (I\_DEC\_OPCODE="000000" and I\_DEC\_EN='1')else

"00" when (I\_DEC\_OPCODE="001000" and I\_DEC\_EN='1')else

"01" when (I\_DEC\_OPCODE="000100" and I\_DEC\_EN='1')else

"11" when (I\_DEC\_OPCODE="000101" and I\_DEC\_EN='1')else

"00" when (I\_DEC\_OPCODE="000010" and I\_DEC\_EN='1')else

"00" when (I\_DEC\_OPCODE="100011" and I\_DEC\_EN='1') else

"00" when (I\_DEC\_OPCODE="101011" and I\_DEC\_EN='1') else "00";

end Behavioral;

**VHDL CODE of ALU CONTROL UNIT (ACU):**

----------------------------------------------------------------------------------

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity ACU is

Port ( I\_ACU\_ALUOp : in STD\_LOGIC\_VECTOR (1 downto 0);

I\_ACU\_Funct : in STD\_LOGIC\_VECTOR (5 downto 0);

O\_ACUCTL : out STD\_LOGIC\_VECTOR (3 downto 0));

end ACU;

architecture Behavioral of ACU is

begin

process(I\_ACU\_ALUOp,I\_ACU\_Funct)

begin

case I\_ACU\_ALUOp is

when "00" => O\_ACUCTL <="0010";

when "01" => O\_ACUCTL <="0110";

when others =>

case I\_ACU\_Funct is

when "100000" => O\_ACUCTL <=

"0010";

when "100010" => O\_ACUCTL<=

"0110";

when "100100" => O\_ACUCTL <=

"0000";

when "100101" => O\_ACUCTL <=

"0001";

when "101010" => O\_ACUCTL<=

"0111";

when others => O\_ACUCTL <=

"----";

end case;

end case;

end process;

end Behavioral;

**Test Bench:**

--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY TEST\_MODULE IS

PORT (

O\_REGDST : OUT std\_logic;

O\_JUMP : OUT std\_logic;

O\_BEQ : OUT std\_logic;

O\_BNE : OUT std\_logic;

O\_MEMREAD : OUT std\_logic;

O\_MEMTOREG : OUT std\_logic;

O\_MEMWRITE : OUT std\_logic;

O\_ALUSRC : OUT std\_logic;

O\_REGWRITE : OUT std\_logic;

O\_ALUCTL : OUT std\_logic\_vector(3 downto 0)

);

END TEST\_MODULE;

ARCHITECTURE behavior OF TEST\_MODULE IS

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

COMPONENT DEC

PORT(

I\_DEC\_EN : IN std\_logic;

I\_DEC\_OPCODE : IN std\_logic\_vector(5 downto 0);

O\_DEC\_REGDST : OUT std\_logic;

O\_DEC\_JUMP : OUT std\_logic;

O\_DEC\_BEQ : OUT std\_logic;

O\_DEC\_BNE : OUT std\_logic;

O\_DEC\_MEMREAD : OUT std\_logic;

O\_DEC\_MEMTOREG : OUT std\_logic;

O\_DEC\_ALUOP : OUT std\_logic\_vector(1 downto 0);

O\_DEC\_MEMWRITE : OUT std\_logic;

O\_DEC\_ALUSRC : OUT std\_logic;

O\_DEC\_REGWRITE : OUT std\_logic

);

END COMPONENT;

COMPONENT ACU

PORT(

I\_ACU\_ALUOp : in STD\_LOGIC\_VECTOR (1 downto 0);

I\_ACU\_Funct : in STD\_LOGIC\_VECTOR (5 downto 0);

O\_ACUCTL : out STD\_LOGIC\_VECTOR (3 downto 0)

);

END COMPONENT;

--Inputs

signal I\_DEC\_EN : std\_logic := '0';

signal I\_EN : std\_logic:= '0';

signal I\_INSTR : std\_logic\_vector(31 downto 0):=(others=>'0');

--Outputs

signal O\_DEC\_ALUOP : std\_logic\_vector(1 downto 0);

-- No clocks detected in port list. Replace <clock> below with

-- appropriate port name

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut\_DEC: DEC PORT MAP (

I\_DEC\_EN => I\_EN,

I\_DEC\_OPCODE => I\_INSTR(31 downto 26),

O\_DEC\_REGDST => O\_REGDST,

O\_DEC\_JUMP => O\_JUMP,

O\_DEC\_BEQ => O\_BEQ,

O\_DEC\_BNE => O\_BNE,

O\_DEC\_MEMREAD => O\_MEMREAD,

O\_DEC\_MEMTOREG => O\_MEMTOREG,

O\_DEC\_ALUOP => O\_DEC\_ALUOP,

O\_DEC\_MEMWRITE => O\_MEMWRITE,

O\_DEC\_ALUSRC => O\_ALUSRC,

O\_DEC\_REGWRITE => O\_REGWRITE

);

uut\_ACU: ACU PORT MAP (

I\_ACU\_ALUOp => O\_DEC\_ALUOP,

I\_ACU\_Funct => I\_INSTR(5 downto 0),

O\_ACUCTL => O\_ALUCTL

);

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

I\_INSTR<=X"00123A00";

wait for 100 ns;

I\_EN<='1';

wait for 100 ns;

I\_INSTR<=X"00123A00";

wait for 100 ns;

I\_INSTR<=X"20123A00";

wait for 100 ns;

I\_INSTR<=X"10123A00";

wait for 100 ns;

I\_INSTR<=X"14123A00";

wait for 100 ns;

I\_INSTR<=X"08123A00";

wait for 100 ns;

I\_INSTR<=X"8C123A00";

wait for 100 ns;

I\_INSTR<=X"AC123A00";

wait for 100 ns;

I\_INSTR<=X"20123A00";

wait for 100 ns;

I\_INSTR<=X"00123A20";

wait for 100 ns;

I\_INSTR<=X"20123A22";

wait for 100 ns;

I\_INSTR<=X"10123A24";

wait for 100 ns;

I\_INSTR<=X"14123A25";

wait for 100 ns;

I\_INSTR<=X"08123A2A";

wait for 100 ns;

I\_INSTR<=X"8C123A20";

wait for 100 ns;

I\_INSTR<=X"AC123A22";

wait for 100 ns;

I\_INSTR<=X"20123A2A";

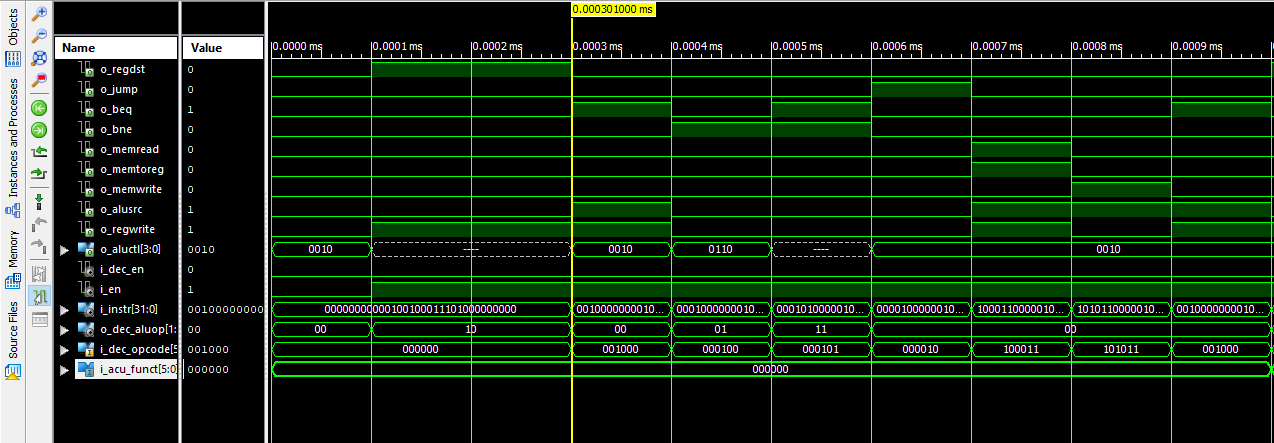
-- insert stimulus here

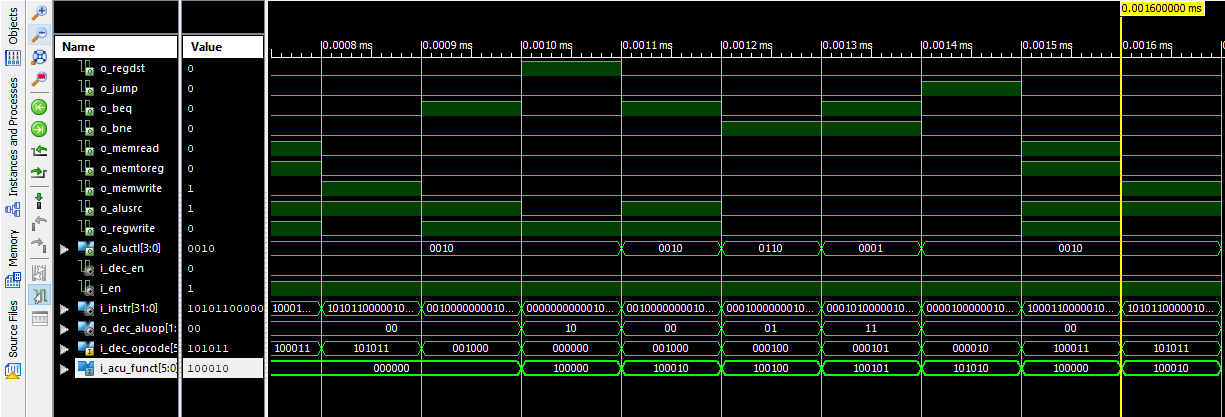
wait;

end process;

END;

**Simulation Waveform:**





Rushdi, A. M. and Al-Yahya, H. A., "A Boolean minimization procedure using the variable-

entered Karnaugh map and the generalized consensus concept", International Journal of

Electronics, Vol. 87, No. 7, (2000), pp. 769-794.

Rushdi, A. M. and Al-Yahya, H. A., "A Boolean minimization procedure using the variable-

entered Karnaugh map and the generalized consensus concept", International Journal of

Electronics, Vol. 87, No. 7, (2000), pp. 769-794.

Rushdi, A. M. and Al-Yahya, H. A., "A Boolean minimization procedure using the variable-

entered Karnaugh map and the generalized consensus concept", International Journal of

Electronics, Vol. 87, No. 7, (2000), pp. 769-794.