**ALU Component**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity ALU is

Port ( a1 : in STD\_LOGIC\_VECTOR (31 downto 0);

a2 : in STD\_LOGIC\_VECTOR (31 downto 0);

alu\_control : in STD\_LOGIC\_VECTOR (3 downto 0);

ALU\_result : out STD\_LOGIC\_VECTOR (31 downto 0);

Zero : out STD\_LOGIC);

end ALU;

architecture Behavioral of ALU is

signal resultX : std\_logic\_vector (31 downto 0);

begin

process (a1,a2, alu\_control)

begin

case alu\_control is

when "0000" => --bitwise and

resultX <= a1 and a2;

when "0001" => --bitwise or

resultX <= a1 or a2;

when "0010" => --addition

resultX <= std\_logic\_vector(unsigned (a1) + unsigned (a2));

when "0110" => --substraction

resultX <= std\_logic\_vector(unsigned (a1) - unsigned (a2));

when "0111" => --set less than

if ( signed (a1)< signed (a2)) then

resultX <= x"00000001";

else

resultX <= x"00000000";

end if;

when "1100" => --logical nor

resultX <= a1 nor a2;

when others => null; --nop

resultX <=x"00000000";

end case;

end process;

--concurrent code

ALU\_result <= resultX;

Zero <='1' when resultX= x"00000000" else

'0';

end Behavioral;

**Test Bench Code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

--USE ieee.numeric\_std.ALL;

ENTITY Tb\_ALU IS

END Tb\_ALU;

ARCHITECTURE behavior OF Tb\_ALU IS

--Inputs

signal tb\_a1 : std\_logic\_vector(31 downto 0) := (others => '0');

signal tb\_a2 : std\_logic\_vector(31 downto 0) := (others => '0');

signal tb\_alu\_control : std\_logic\_vector(3 downto 0) := (others => '0');

--Outputs

signal tb\_ALU\_result : std\_logic\_vector(31 downto 0);

signal tb\_Zero : std\_logic;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: entity work.ALU (Behavioral) PORT MAP (

a1 => tb\_a1,

a2 => tb\_a2,

alu\_control => tb\_alu\_control,

ALU\_result => tb\_ALU\_result,

Zero => tb\_Zero

);

-- Stimulus process

stim\_proc: process

begin

tb\_a1 <= x"00000003";

tb\_a2 <= x"FFFFFFFF";

tb\_alu\_control <= "0000"; --bitwise and

wait for 50 ns;

tb\_alu\_control <= "0001"; --bitwise or

wait for 50 ns;

tb\_alu\_control <= "0010"; --addition

wait for 50 ns;

tb\_alu\_control <= "0110"; --substraction

wait for 50 ns;

tb\_alu\_control <= "0111"; --set less than

wait for 50 ns;

tb\_alu\_control <= "1100"; --bitwise nor

wait for 50 ns;

assert false

report "End"

severity failure;

end process;

END;

**ALU Control**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

--use IEEE.NUMERIC\_STD.ALL;

entity AluControl is

Port ( funct : in STD\_LOGIC\_VECTOR (5 downto 0);

ALUOp : in STD\_LOGIC\_VECTOR (1 downto 0);

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

end AluControl;

architecture Behavioral of AluControl is

begin

Operation (3) <= '0';

Operation (2) <= ALUOp (0) or (ALUOp(1) and funct (1));

Operation (1) <= not ALUOp (1) or not funct (2);

Operation(0) <= (funct (3) or funct (0))and ALUOp (1);

end Behavioral

**Test Bench Code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

--USE ieee.numeric\_std.ALL;

ENTITY Tb\_AluControl IS

END Tb\_AluControl;

ARCHITECTURE behavior OF Tb\_AluControl IS

--Inputs

signal tb\_funct : std\_logic\_vector(5 downto 0) := (others => '0');

signal tb\_ALUOp : std\_logic\_vector(1 downto 0) := (others => '0');

--Outputs

signal tb\_Operation : std\_logic\_vector(3 downto 0);

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: entity work.AluControl (Behavioral)

PORT MAP (

funct => tb\_funct,

ALUOp => tb\_ALUOp,

Operation => tb\_Operation

);

-- Stimulus process

stim\_proc: process

begin

--R type command

tb\_ALUOp <= "10";

tb\_funct <= "100100"; --and

wait for 20 ns;

tb\_ALUOp <= "10";

tb\_funct <= "100101"; --or

wait for 20 ns;

tb\_ALUOp <= "10";

tb\_funct <= "100000"; --and

wait for 20 ns;

tb\_ALUOp <= "10";

tb\_funct <= "100010"; --Sub

wait for 20 ns;

tb\_ALUOp <= "10";

tb\_funct <= "101010"; --slt

wait for 20 ns;

--I type command

--Load word command

tb\_ALUOp <= "00";

tb\_funct <= "XXXXXX"; --and

wait for 20 ns;

--Store word command

tb\_ALUOp <= "00";

tb\_funct <= "XXXXXX"; --and

wait for 20 ns;

--Branch equal command

tb\_ALUOp <= "01";

tb\_funct <= "XXXXXX"; --and

wait for 20 ns;

assert false

report "End"

severity failure;

end process;

END;

**Controller Component**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

--use IEEE.NUMERIC\_STD.ALL;

entity Controller is

Port (OpCode: in std\_logic\_vector (5 downto 0); --Instructiob 31 - 26

RegDst : out std\_logic ;

Jump : out std\_logic;

Branch :out std\_logic;

MemRead : out std\_logic;

MemToReg : out std\_logic;

ALUOp : out std\_logic\_vector (1 downto 0);

MemWrite : out std\_logic;

ALUSrc : out std\_logic;

RegWrite : out std\_logic

);

end Controller;

architecture Behavioral of Controller is

begin

process (OpCode)

begin

RegWrite <= '0'; --Deassert for next command

case OpCode is

when "000000" =>--And, or, Add ,Sub, Slt :0x00

RegDst <= '1';

Jump <= '0';

Branch <= '0';

MemRead <= '0';

MemToReg <= '0';

ALUOp <= "10";

MemWrite <= '0';

ALUSrc <= '0';

RegWrite <= '1' after 10 ns;

when "100011" => --Load word (lw) : 0x23

RegDst <= '0';

Jump <= '0';

Branch <= '0';

MemRead <= '1';

MemToReg <= '1';

ALUOp <= "00";

MemWrite <= '0';

ALUSrc <= '1';

RegWrite <= '1' after 10 ns;

when "101011" => --store word (sw) : 0x2B

RegDst <= 'X'; --dont care

Jump <= '0';

Branch <= '0';

MemRead <= '0';

MemToReg <= 'X';

ALUOp <= "00";

MemWrite <= '1';

ALUSrc <= '1';

RegWrite <= '0';

when "000100" => --branch equal (beq) : 0x04

RegDst <= 'X';

Jump <= '0';

Branch <= '1' after 2 ns;

MemRead <= '0';

MemToReg <= 'X';

ALUOp <= "01";

MemWrite <= '0';

ALUSrc <= '0';

RegWrite <= '0';

when "000010" => --Jump (j) : 0x02

RegDst <= 'X';

Jump <= '1';

Branch <= '0';

MemRead <= '0';

MemToReg <= 'X';

ALUOp <= "00";

MemWrite <= '0';

ALUSrc <= '0';

RegWrite <= '0';

when others => null; --Implement other commands down here

RegDst <= '0';

Jump <= '0';

Branch <= '0';

MemRead <= '0';

MemToReg <= '0';

ALUOp <= "00";

MemWrite <= '0';

ALUSrc <= '0';

RegWrite <= '0';

end case;

end process;

end Behavioral;

**Test Bench Code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

--USE ieee.numeric\_std.ALL;

ENTITY Tb\_Controller IS

END Tb\_Controller;

ARCHITECTURE behavior OF Tb\_Controller IS

--Inputs

signal tb\_OpCode : std\_logic\_vector(5 downto 0) := (others => '0');

--Outputs

signal tb\_RegDst : std\_logic;

signal tb\_Jump : std\_logic;

signal tb\_Branch : std\_logic;

signal tb\_MemRead : std\_logic;

signal tb\_MemToReg : std\_logic;

signal tb\_ALUOp : std\_logic\_vector(1 downto 0);

signal tb\_MemWrite : std\_logic;

signal tb\_ALUSrc : std\_logic;

signal tb\_RegWrite : std\_logic;

BEGIN

-- Instantiate the Unit Under Test (UUT)

U1\_Test : entity work.Controller (Behavioral)

PORT MAP (

OpCode => tb\_OpCode,

RegDst => tb\_RegDst,

Jump => tb\_Jump,

Branch => tb\_Branch,

MemRead => tb\_MemRead,

MemToReg => tb\_MemToReg,

ALUOp => tb\_ALUOp,

MemWrite => tb\_MemWrite,

ALUSrc => tb\_ALUSrc,

RegWrite => tb\_RegWrite

);

-- Stimulus process

stim\_proc: process

begin

tb\_OpCode <= "000000"; --R type command

wait for 50 ns;

tb\_OpCode <= "100011"; --Load Word

wait for 50 ns;

tb\_OpCode <= "101011"; -- Store Word

wait for 50 ns;

tb\_OpCode <= "000100"; --Branch equal

wait for 50 ns;

tb\_OpCode <= "000010"; --Jump

wait for 50 ns;

tb\_OpCode <= "111111"; --unknown

wait for 50 ns;

assert false;

report "End"

severity failure;

end process;

END;

**Data Memory Component**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity DataMemory is

Port ( Address : in STD\_LOGIC\_VECTOR (31 downto 0);

Writedata : in STD\_LOGIC\_VECTOR (31 downto 0);

MemRead : in STD\_LOGIC;

MemWrite : in STD\_LOGIC;

ReadData : out STD\_LOGIC\_VECTOR (31 downto 0)

);

end DataMemory;

architecture Behavioral of DataMemory is

type RAM\_16\_x\_32 is array(0 to 15) of std\_logic\_vector(31 downto 0);

signal DM : RAM\_16\_x\_32 := (

x"00000000", --Assume starts at 0x100010000

x"00000000",

x"00000000",

x"00000000",

x"00000000",

x"00000000",

x"00000000",

x"00000000",

x"00000000",

x"00000000",

x"00000000",

x"00000000",

x"00000000",

x"00000000",

x"00000000",

x"00000000"

);

begin

process (MemWrite, MemRead) --Pulse on write

begin

--Note: 268500992 = 0x10010000

if (MemWrite = '1') then

DM (( to\_integer (unsigned(Address))-268500992)/4) <= WriteData;

end if;

if (MemRead = '1') then

ReadData <= DM ((to\_integer(unsigned(Address))-268500992)/4);

end if;

end process;

end Behavioral;

**Test Bench Code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

--USE ieee.numeric\_std.ALL;

ENTITY Tb\_DataMemory IS

END Tb\_DataMemory;

ARCHITECTURE behavior OF Tb\_DataMemory IS

--Inputs

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

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

signal tb\_MemRead : std\_logic := '0';

signal tb\_MemWrite : std\_logic := '0';

--Outputs

signal tb\_ReadData : std\_logic\_vector(31 downto 0);

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: entity work.DataMemory (Behavioral)

PORT MAP (

Address => tb\_Address,

Writedata => tb\_Writedata,

MemRead => tb\_MemRead,

MemWrite => tb\_MemWrite,

ReadData => tb\_ReadData

);

-- Stimulus process

stim\_proc: process

begin

--Write two memory locations

tb\_Address <= x"10010000";

tb\_WriteData <= x"11112222";

tb\_MemWrite <= '0';

wait for 10 ns;

tb\_MemWrite <= '1';

wait for 10 ns;

tb\_MemWrite <= '0';

wait for 10 ns;

tb\_Address <= x"10010004";

tb\_WriteData <= x"33334444";

tb\_MemWrite <= '0';

wait for 10 ns;

tb\_MemWrite <= '1';

wait for 10 ns;

tb\_MemWrite <= '0';

wait for 10 ns;

--Read two memory locations

tb\_Address <= x"10010000";

tb\_MemRead <= '0';

wait for 10 ns;

tb\_MemRead <= '1';

wait for 10 ns;

tb\_MemRead <= '0';

wait for 10 ns;

tb\_Address <= x"10010004";

tb\_MemRead <= '0';

wait for 10 ns;

tb\_MemRead <= '1';

wait for 10 ns;

tb\_Memread <= '0';

wait for 10 ns;

assert false

report "End"

severity failure;

end process;

END;

**Instruction Memory Component**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity InstructionMemory is

Port ( ReadAddress : in STD\_LOGIC\_VECTOR (31 downto 0);

Instruction : out STD\_LOGIC\_VECTOR (31 downto 0)

);

end InstructionMemory;

architecture Behavioral of InstructionMemory is

type RAM\_16\_x\_32 is array(0 to 15) of std\_logic\_vector(31 downto 0);

signal IM : RAM\_16\_x\_32 := (x"01285024", --0x0040 0000; and $t2, $t1, $t0

x"018b6825", --0x0040 0004; or $t5, $t4, $t3

x"01285020", --0x0040 0008; add $t2, $t1, $t0

x"01285022", --0x0040 000c; sub $t2, $t1, $t0

x"0149402a", --0x0040 0010; slt $t0, $t2, $t1

x"1211fffb", --0x0040 0014; beq $s0, $s1, L1 (1210fffb for $s1,$s1)

x"01285024", --0x0040 0018; and $t2, $t1, $t0

x"018b6825", --0x0040 001c; or $t5, $t4, $t3

x"01285020", --0x0040 0020; add $t2, $t1, $t0

x"01285022", --0x0040 0024; sub $t2, $t1, $t0

x"0149402a", --0x0040 0028; slt $t0, $t2, $t1

x"08100000", --0x0040 002c; j 0x00400000 => 0000 1000 0001 0000 0000 0000 0000 0000

x"00000000",

x"00000000",

x"00000000",

x"00000000"

);

begin

--Note: 4194304 = 0x0040 0000

Instruction <= x"00000000" when ReadAddress = x"003FFFFC" else

IM ((to\_integer(unsigned(ReadAddress))-4194304)/4);

end Behavioral;

**Test Bench Code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY Tb\_InstructionMemory IS

END Tb\_InstructionMemory;

ARCHITECTURE behavior OF Tb\_InstructionMemory IS

--Inputs

signal tb\_ReadAddress : std\_logic\_vector(31 downto 0) := x"003fffff";

--Outputs

signal tb\_Instruction : std\_logic\_vector(31 downto 0);

BEGIN

-- Instantiate the Unit Under Test (UUT)

U1\_Test: entity work.InstructionMemory (Behavioral)

PORT MAP (

ReadAddress => tb\_ReadAddress,

Instruction => tb\_Instruction

);

-- Stimulus process

stim\_proc: process

begin

--Read 12 commands

for I in 0 to 11 loop

tb\_ReadAddress <= x"00400000" or std\_logic\_vector(to\_unsigned(I\*4,32));

wait for 25 ns;

end loop;

assert false

report "End"

severity failure;

end process;

END;

**Register File Component**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity RegisterFile is

Generic (

B : integer :=32; --Number of bits

W : integer :=5 --Number of address bits

);

Port ( ReadRegister1 : in Std\_logic\_vector (W-1 downto 0);

ReadRegister2 : in Std\_logic\_vector (W-1 downto 0);

WriteRegister : in Std\_logic\_vector (W-1 downto 0);

WriteData : in Std\_logic\_vector (B-1 downto 0);

RegWrite : in Std\_logic;

ReadData1 : out Std\_logic\_vector (B-1 downto 0);

ReadData2 : out Std\_logic\_vector (B-1 downto 0)

);

end RegisterFile;

architecture Behavioral of RegisterFile is

type reg\_file\_type is array (0 to 2\*\*W-1) of

Std\_logic\_vector (B-1 downto 0);

signal array\_reg : reg\_file\_type := ( x"00000000", --$zero

x"11111111", --$at

x"22222222", --$v0

x"33333333", --$v1

x"44444444", --$a0

x"55555555", --$a1

x"66666666", --$a2

x"77777777", --$a3

x"88888888", --$t0

x"99999999", --$t1

x"aaaaaaaa", --$t2

x"bbbbbbbb", --$t3

x"cccccccc", --$t4

x"dddddddd", --$t5

x"eeeeeeee", --$t6

x"ffffffff", --$t7

x"00000000", --$s0

x"11111111", --$s1

x"22222222", --$s2

x"33333333", --$s3

x"44444444", --$s4

x"55555555", --$s5

x"66666666", --$s6

x"77777777", --$s7

x"88888888", --$t8

x"99999999", --$t9

x"aaaaaaaa", --$k0

x"bbbbbbbb", --$k1

x"10008000", --$gp

x"7FFFF1EC", --$sp

x"eeeeeeee", --$fp

x"ffffffff" --$ra

);

begin

process (RegWrite) --pulse on write

begin

if (RegWrite ='1') then

array\_reg(to\_integer (unsigned(WriteRegister))) <= WriteData;

end if;

end process;

--read port

ReadData1 <= array\_reg(to\_integer(unsigned(ReadRegister1)));

ReadData2 <= array\_reg(to\_integer(unsigned(ReadRegister2)));

end Behavioral;

**Test Bench Code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY Tb\_RegisterFile IS

END Tb\_RegisterFile;

ARCHITECTURE behavior OF Tb\_RegisterFile IS

--Inputs

signal tb\_ReadRegister1 : std\_logic\_vector(4 downto 0) := (others => '0');

signal tb\_ReadRegister2 : std\_logic\_vector(4 downto 0) := (others => '0');

signal tb\_WriteRegister : std\_logic\_vector(4 downto 0) := (others => '0');

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

signal tb\_RegWrite : std\_logic := '0';

--Outputs

signal tb\_ReadData1 : std\_logic\_vector(31 downto 0);

signal tb\_ReadData2 : std\_logic\_vector(31 downto 0);

BEGIN

-- Instantiate the Unit Under Test (UUT)

U1\_Reg: entity work.RegisterFile (Behavioral)

PORT MAP (

ReadRegister1 => tb\_ReadRegister1,

ReadRegister2 => tb\_ReadRegister2,

WriteRegister => tb\_WriteRegister,

WriteData => tb\_WriteData,

RegWrite => tb\_RegWrite,

ReadData1 => tb\_ReadData1,

ReadData2 => tb\_ReadData2

);

-- Stimulus process

stim\_proc: process

begin

--Reading all 32 Registers

-- for I in 0 to 30 loop

-- tb\_ReadRegister1 <= std\_logic\_vector (to\_unsigned(I,5));

-- tb\_ReadRegister2 <= std\_logic\_vector (to\_unsigned(I+1,5));

-- wait for 25 ns;

-- end loop;

--Writing a register

tb\_WriteRegister <= "01000"; --$t0 register

tb\_WriteData <= x"a5a5a5a5";

wait for 50 ns;

tb\_RegWrite <= '1';

wait for 5 ns;

tb\_RegWrite <= '0';

wait for 5 ns;

assert false

report "End"

severity failure;

end process;

END;