COE 405, Term 062

Design & Modeling of Digital Systems

HW# 3 Solution

Due date: Monday, April 16, 2007

Q.1. You are required to model an ALU that has the following entity description:

```
Entity ALU is
        Generic (N: Natural :=4);
        Port (A, B: IN Bit_Vector(N-1 Downto 0);
            Cin: IN Bit;
            Sel: IN Bit_Vector(2 Downto 0);
            C: OUT Bit_Vector(N-1 Downto 0);
            Cout, SignF, OverflowF, ZeroF: OUT BIT
            );
End;
```

The ALU performs one of eight different functions according to selection line inputs as shown in the table given below:

Sel	Function
000	**C=A+B**
001	**C=A+B+Cin**
010	**C=A-B**
011	**C=A-B-Cin**
100	**C=B+1**
101	**C=B-1**
110	**C=B**
111	**C=2*B**

The four flags Cout, SignF, OverflowF and ZeroF are computed according to the result. Note that the Cout flag is considered a borrow when a subtraction operation is performed.

(i) Model the following two functions, “+” and “-“, to support addition and subtraction on Bit_Vector. Model the functions by converting Bit_Vector type to Integer, perform the required operation in integer and then convert the result back to Bit_Vector type. Assume that the returned result has length one extra bit more than the inputs to return the carry out.

Function "+" (l, r : Bit_Vector) RETURN Bit_vector IS

Function "-" (l, r : Bit_Vector) RETURN Bit_vector IS

```
Subtype SInteger is Integer Range -2**(N+1)to 2**(N+1)-1;
Function Bin2Int(Bin: Bit_Vector) return SInteger is
                variable SUM: SInteger:=0;
        begin
                -- convert to integer as unsigned
                For I IN 0 To (Bin'Length - 1) Loop
                        if Bin(I)='1' then
                                SUM := SUM + (2**I);
                        end if;
                End Loop;
                return SUM;

        end Bin2Int;

Procedure Int2Bin (Int: IN SInteger; Bin : OUT BIT_VECTOR)  IS
        Variable Tmp: SInteger;
        Constant size: Natural := Bin'length;
Begin
        Tmp := Int;
        if (Tmp < 0) Then
                Tmp :=2**size+Tmp;
        End If;
        For I IN 0 To (Bin'Length - 1) Loop
                If ( Tmp MOD 2 = 1) Then
                        Bin(I) := '1';
                Else Bin(I) := '0';
                End If;
                Tmp := Tmp / 2;
        End Loop;
End Int2Bin;

Function        "+" ( l,r : Bit_Vector ) RETURN Bit_vector IS
        Variable IRes: SInteger;
        Variable Result: Bit_Vector(l'length downto 0);
Begin
        IRes := Bin2Int(l) + Bin2Int(r);
        Int2Bin(IRes, Result);
        Return Result;
End "+";
Function        "-" ( l,r : Bit_Vector ) RETURN Bit_vector IS
        Variable IRes: SInteger;
        Variable Result: Bit_Vector(l'length downto 0);
Begin
        IRes := Bin2Int(l) - Bin2Int(r);
        Int2Bin(IRes, Result);
        Return Result;
End "-";
```

(ii) Write a behavioral model for modeling the ALU using the developed functions in (i).

```
Architecture BM of ALU is
Begin

    Process(A, B, Cin, Sel)
            Variable Tmp: Bit_Vector (A'length Downto 0);
            Variable Tmp2: Bit_Vector (A'length-1 Downto 0);
            Variable Tmp3 : Bit_Vector (A'length Downto 0);
            Variable Zero: Bit_Vector (A'length-1 Downto 0) := (N-1 Downto 0=>'0');

    Begin
    Case Sel is
            When "000" =>   Tmp := A + B;
                            Cout <= Tmp(N); C <= Tmp(N-1 Downto 0);
                            if (Tmp(N-1 Downto 0) = Zero) Then
                                    ZeroF <= '1';
                            else
                                    ZeroF <= '0';
                            end if;
                            if ( (A(N-1) = B(N-1)) AND (Tmp(N-1) /= A(N-1)) )          Then
                                    OverflowF <= '1';
                            else
                                    OverflowF <= '0';
                            End if;
                            Signf <= Tmp(N-1);
            When "001" =>   Tmp  := A + B;
                            Tmp2 := ((N-1 Downto 1=>'0')&Cin);
                            Tmp3 := Tmp(N-1 Downto 0) + Tmp2;
                            C <= Tmp3(N-1 Downto 0);
                            If (Tmp(N)='1' OR Tmp3(N)='1') Then
                                    Cout <= '1';
                            else
                                    Cout <= '0';
                            end if;
                            if (Tmp3(N-1 Downto 0) = Zero) Then
                                    ZeroF <= '1';
                            else
                                    ZeroF <= '0';
                            end if;
                            if ( (A(N-1) = B(N-1)) AND (Tmp3(N-1) /= A(N-1)) )         Then
                                    OverflowF <= '1';
                            else
                                    OverflowF <= '0';
                            End if;
                            Signf <= Tmp3(N-1);
            When "010" =>   Tmp := A - B;
                            Cout <= Tmp(N); C <= Tmp(N-1 Downto 0);
                            if (Tmp(N-1 Downto 0) = Zero) Then
                                    ZeroF <= '1';
                            else
                                    ZeroF <= '0';
                            end if;
                            if ( (A(N-1) /= B(N-1)) AND (Tmp(N-1) /= A(N-1)) )         Then
                                    OverflowF <= '1';
                            else
```

```
                OverflowF <= '0';
            End if;
            Signf <= Tmp(N-1);
When "011" => Tmp := A - B;
            Tmp2 := ((N-1 Downto 1=>'0')&Cin);
            Tmp3 := Tmp(N-1 Downto 0) - Tmp2;
            C <= Tmp3(N-1 Downto 0);
            If (Tmp(N)='1' OR Tmp3(N)='1') Then
                Cout <= '1';
            else
                Cout <= '0';
            end if;
            if (Tmp3(N-1 Downto 0) = Zero) Then
                ZeroF <= '1';
            else
                ZeroF <= '0';
            end if;
            if ( (A(N-1) /= B(N-1)) AND (Tmp3(N-1) /= A(N-1)) )        Then
                OverflowF <= '1';
            else
                OverflowF <= '0';
            End if;
            Signf <= Tmp3(N-1);
When "100" =>
            Tmp2 := ((N-1 Downto 1=>'0')&'1');
            Tmp3 := B + Tmp2;
            C <= Tmp3(N-1 Downto 0);
            Cout <= Tmp3(N);
            if (Tmp3(N-1 Downto 0) = Zero) Then
                ZeroF <= '1';
            else
                ZeroF <= '0';
            end if;
            if ( (B(N-1) = Tmp2(N-1)) AND (Tmp3(N-1) /= B(N-1)) )      Then
                OverflowF <= '1';
            else
                OverflowF <= '0';
            End if;
            Signf <= Tmp3(N-1);
When "101" =>
            Tmp2 := ((N-1 Downto 1=>'0')&'1');
            Tmp3 := B - Tmp2;
            C <= Tmp3(N-1 Downto 0);
            Cout <= Tmp3(N);
            if (Tmp3(N-1 Downto 0) = Zero) Then
                ZeroF <= '1';
            else
                ZeroF <= '0';
            end if;
            if ( (B(N-1) /= Tmp2(N-1)) AND (Tmp3(N-1) /= B(N-1)) )     Then
                OverflowF <= '1';
            else
                OverflowF <= '0';
            End if;
            Signf <= Tmp3(N-1);
When "110" =>
            Cout <= '0'; C <= B;
            if (B(N-1 Downto 0) = Zero) Then
```

```
                    ZeroF <= '1';
                else
                    ZeroF <= '0';
                end if;

                OverflowF <= '0';
                Signf <= B(N-1);
When "111" =>   Tmp := B + B;
                Cout <= Tmp(N); C <= Tmp(N-1 Downto 0);
                if (Tmp(N-1 Downto 0) = Zero) Then
                    ZeroF <= '1';
                else
                    ZeroF <= '0';
                end if;
                if ( (A(N-1) = B(N-1)) AND (Tmp(N-1) /= A(N-1)) )        Then
                    OverflowF <= '1';
                else
                    OverflowF <= '0';
                End if;
                Signf <= Tmp(N-1);


    End Case;

    End Process;


End;
```

(iii) Write a test bench for testing the n-bit ALU assuming that the input arguments are read from an input file and that the output will be stored in an output file. Use TEXTIO package for this purpose. Apply the following values for testing the correct operation of a 4-bit ALU:

ALU Select	Input A	Input B	Cin
000	5	2	
000	-8	7	
000	7	7	
000	-7	-2	
000	-1	1	
000	-1	-1	
001	-1	1	1
001	-1	-1	0
001	-1	-1	1
001	-1	0	1
010	3	4	
010	-8	7	
010	-7	-1	
010	-7	2	
011	-7	1	1
011	3	2	1
011	-8	1	1
100		-1	

100		1	
100		7	
101		0	
101		-1	
101		-8	
101		7	
110		-1	
110		7	
111		-1	
111		3	
111		7	

The output should be stored in the output file using the following format:

ALU Operation	Input A	Input B	Result	Cout	Signf	OverflowF	ZeroF
C=A+B	5	2	7	0	0	0	0

```
USE STD.TEXTIO.ALL;

Entity ALU_test is
End;

Architecture Test of ALU_test is
    Component ALU
    Generic (N: Natural :=4);
    Port (A, B: IN Bit_Vector(N-1 Downto 0);
          Cin: IN Bit;
         Sel: IN Bit_Vector(2 Downto 0);
        C: OUT Bit_Vector(N-1 Downto 0);
       Cout, SignF, OverflowF, ZeroF: OUT BIT
      );
    End Component;

    For ALL: ALU Use Entity work.alu(BM);

    Constant N: Positive :=4;
    Constant K: Positive :=30;
    Constant Period: Time := 100 ns;


    TYPE Integers IS ARRAY (NATURAL RANGE <>) of INTEGER;
    TYPE Vectors IS ARRAY (NATURAL RANGE <>) of Bit_Vector(2 downto 0);
    TYPE Bits IS ARRAY (NATURAL RANGE <>) of Bit;
```

```
Procedure Int2Bin (Int: IN Integer; Bin : OUT BIT_VECTOR)  IS
        Variable Tmp: Integer;
        Constant size: Natural := Bin'length;
Begin
        Tmp := Int;
        if (Tmp < 0) Then
                Tmp :=2**size+Tmp;
        End If;
        For I IN 0 To (Bin'Length - 1) Loop
                If ( Tmp MOD 2 = 1) Then
                        Bin(I) := '1';
                Else Bin(I) := '0';
                End If;
                Tmp := Tmp / 2;
        End Loop;
End Int2Bin;
Procedure Apply_Data (
                Signal Target: OUT Bit_Vector;
                Constant Values: IN Integers;
                Constant Period: IN Time) IS
                Variable Buf: Bit_Vector(Target'range);
Begin
                For I IN 0 To Values'length-1 Loop
                        Int2Bin (Values(I), Buf);
                        Target <= Transport Buf After I * Period;
                End Loop;
End Apply_Data;

Procedure Apply_Data (
                Signal Target: OUT Bit_Vector;
                Constant Values: IN Vectors;
                Constant Period: IN Time) IS
Begin
                For I IN 0 To Values'length-1 Loop
                        Target <= Transport Values(I) After I * Period;
                End Loop;
End Apply_Data;
Procedure Apply_Data (
                Signal Target: OUT Bit;
                Constant Values: IN Bits;
                Constant Period: IN Time) IS
Begin
                For I IN 0 To Values'length-1 Loop
                        Target <= Transport Values(I) After I * Period;
                End Loop;
End Apply_Data;

function Bin2Int(Bin: Bit_Vector) return integer is
```

```
        variable SUM: INTEGER:=0;
begin
        -- convert to integer as unsigned
        For I IN 0 To (Bin'Length - 1) Loop
                if Bin(I)='1' then
                        SUM := SUM + (2**I);
                end if;
        End Loop;
        -- if negative
        if (Bin(Bin'Length -1)='1') then
                -- 2's complement
                SUM := 2**(Bin'Length)-SUM;
                -- set the negaive sign
                SUM := -SUM;
        end if;
        return SUM;

end Bin2Int;

Signal A, B, C: Bit_Vector(N-1 Downto 0);
Signal Cin, Cout, SignF, OverflowF, ZeroF: Bit;
Signal Sel: Bit_Vector(2 Downto 0);
Signal First, Second: Integers(0 to K-1);
Signal SelA: Vectors(0 to k-1);
Signal Carryin: Bits(0 to k-1);
Begin
Process
File Infile : Text IS IN "alu_input.txt";
Variable My_Line : Line;
Variable val: Integer;
Variable sval: Bit_Vector(2 downto 0);
Variable cval : Bit;


File outFile: Text IS OUT "alu_output.txt";
Variable write_line: Line;
Variable Str: String(1 to 67);
Variable Str2: String(1 to 13);
Variable Str3: String(1 to 85);


Variable i,j: integer :=0;
Begin
        While Not ( Endfile(Infile) )  Loop

                Readline( Infile, My_Line);          -- read a line from the input file
                Read( My_Line, sval);                -- read select value
                SelA(i) <= sval;
```

```
                    Case sval is
                    When "000" | "010" =>
                            Read( My_Line, val);        -- read A value from the line
                            First(i)<= val;
                            Read( My_Line, val);        -- read B value from the line
                            Second(i)<= val;
                            Carryin(i) <= '0';

                    When "001" | "011" =>
                            Read( My_Line, val);        -- read A value from the line
                            First(i)<= val;
                            Read( My_Line, val);        -- read B value from the line
                            Second(i)<= val;
                            Read( My_Line, cval);     -- read Cin value from the line
                            Carryin(i) <= cval;

                    When Others =>
                            First(i)<= 0;
                            Read( My_Line, val);        -- read B value from the line
                            Second(i)<= val;
                            Carryin(i) <= '0';

                    End Case;
                    i := i + 1;
            End Loop;

    -- added code to write sum's into a file ---
            wait for 10 ns;
            Str :="ALU Operation"&ht&"Input A"&ht&"Input
B"&ht&"Cin"&ht&"Result"&ht&"Cout"&ht&"Signf"&ht&"OverflowF"&ht&"Zero
F";
            Write(write_line,Str);
            Writeline(outFile, write_line);
            Str3 :="-------------------------------------------------------------------------------
-----";
            Write(write_line,Str3);
            Writeline(outFile, write_line);
            j:=0;
            while (j < i) loop
                    sval := SelA(j);
                    Case Sval is
                    When "000" =>
                            str2 := "C=A+B       ";
                            Write(write_line,str2);
                            Write(write_line,ht);
                            Write(write_line, First(j));
                            Write(write_line,ht);
                            Write(write_line, Second(j));
```

```
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, Bin2Int(C));
            Write(write_line,ht);
            Write(write_line, Cout);
            Write(write_line,ht);
            Write(write_line, Signf);
            Write(write_line,ht);
            Write(write_line, OverflowF);
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, ZeroF);
            Writeline(outFile, write_line);
        When "001" =>
            str2 := "C=A+B+Cin    ";
            Write(write_line,str2);
            Write(write_line,ht);
            Write(write_line, First(j));
            Write(write_line,ht);
            Write(write_line, Second(j));
            Write(write_line,ht);
            Write(write_line, Carryin(j));
            Write(write_line,ht);
            Write(write_line, Bin2Int(C));
            Write(write_line,ht);
            Write(write_line, Cout);
            Write(write_line,ht);
            Write(write_line, Signf);
            Write(write_line,ht);
            Write(write_line, OverflowF);
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, ZeroF);
            Writeline(outFile, write_line);
        When "010" =>
            str2 := "C=A-B        ";
            Write(write_line,str2);
            Write(write_line,ht);
            Write(write_line, First(j));
            Write(write_line,ht);
            Write(write_line, Second(j));
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, Bin2Int(C));
            Write(write_line,ht);
            Write(write_line, Cout);
            Write(write_line,ht);
            Write(write_line, Signf);
```

```
            Write(write_line,ht);
            Write(write_line, OverflowF);
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, ZeroF);
            Writeline(outFile, write_line);
        When "011" =>
            str2 := "C=A-B-Cin    ";
            Write(write_line,str2);
            Write(write_line,ht);
            Write(write_line, First(j));
            Write(write_line,ht);
            Write(write_line, Second(j));
            Write(write_line,ht);
            Write(write_line, Carryin(j));
            Write(write_line,ht);
            Write(write_line, Bin2Int(C));
            Write(write_line,ht);
            Write(write_line, Cout);
            Write(write_line,ht);
            Write(write_line, Signf);
            Write(write_line,ht);
            Write(write_line, OverflowF);
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, ZeroF);
            Writeline(outFile, write_line);
        When "100" =>
            str2 := "C=B+1        ";
            Write(write_line,str2);
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, Second(j));
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, Bin2Int(C));
            Write(write_line,ht);
            Write(write_line, Cout);
            Write(write_line,ht);
            Write(write_line, Signf);
            Write(write_line,ht);
            Write(write_line, OverflowF);
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, ZeroF);
            Writeline(outFile, write_line);
        When "101" =>
            str2 := "C=B-1        ";
```

```
            Write(write_line,str2);
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, Second(j));
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, Bin2Int(C));
            Write(write_line,ht);
            Write(write_line, Cout);
            Write(write_line,ht);
            Write(write_line, Signf);
            Write(write_line,ht);
            Write(write_line, OverflowF);
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, ZeroF);
            Writeline(outFile, write_line);
        When "110" =>
            str2 := "C=B         ";
            Write(write_line,str2);
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, Second(j));
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, Bin2Int(C));
            Write(write_line,ht);
            Write(write_line, Cout);
            Write(write_line,ht);
            Write(write_line, Signf);
            Write(write_line,ht);
            Write(write_line, OverflowF);
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, ZeroF);
            Writeline(outFile, write_line);
        When "111" =>
            str2 := "C=2*B       ";
            Write(write_line,str2);
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, Second(j));
            Write(write_line,ht);
            Write(write_line,ht);
            Write(write_line, Bin2Int(C));
            Write(write_line,ht);
            Write(write_line, Cout);
            Write(write_line,ht);
```

```
                    Write(write_line, Signf);
                    Write(write_line,ht);
                    Write(write_line, OverflowF);
                    Write(write_line,ht);
                    Write(write_line,ht);
                    Write(write_line, ZeroF);
                    Writeline(outFile, write_line);
                End Case;
                j := j + 1;
                wait for Period;
            End loop;

            wait;
    End Process;

    Apply_data(A, First, Period);
    Apply_data(B, Second, Period);
    Apply_data(Cin, Carryin, Period);
    Apply_data(Sel, SelA, Period);
    CUT: ALU Generic Map (N) Port Map (A, B, Cin, Sel, C, Cout, SignF,
OverflowF, ZeroF);
End;
```

Resulting output file from running the test bench:

ALU Operation	Input A	Input B	Cin	Result	Cout	Signf	OverflowF	ZeroF
-								
C=A+B	5	2		7	0	0	0	0
C=A+B	-8	7		-1	0	1	0	0
C=A+B	7	7		-2	0	1	1	0
C=A+B	-7	-2		7	1	0	1	0
C=A+B	-1	1		0	1	0	0	1
C=A+B	-1	-1		-2	1	1	0	0
C=A+B+Cin	-1	1	1	1	1	0	0	0
C=A+B+Cin	-1	-1	0	-2	1	1	0	0
C=A+B+Cin	-1	-1	1	-1	1	1	0	0
C=A+B+Cin	-1	0	1	0	1	0	0	1
C=A-B	3	4		-1	1	1	0	0
C=A-B	-8	7		1	0	0	1	0
C=A-B	-7	-1		-6	1	1	0	0
C=A-B	-7	2		7	0	0	1	0
C=A-B-Cin	-7	1	1	7	0	0	1	0
C=A-B-Cin	3	2	1	0	0	0	0	1
C=A-B-Cin	-8	1	1	6	0	0	1	0
C=B+1		-1		0	1	0	0	1
C=B+1		1		2	0	0	0	0
C=B+1		7		-8	0	1	1	0
C=B-1		0		-1	1	1	0	0
C=B-1		-1		-2	0	1	0	0
C=B-1		-8		7	0	0	1	0
C=B-1		7		6	0	0	0	0
C=B		-1		-1	0	1	0	0

C=B	7	7	0	0	0	0
C=2*B	-1	-2	1	1	0	0
C=2*B	3	6	0	0	0	0
C=2*B	7	-2	0	1	1	0

(iv) Define a package called HW3 where you store all used types, subtypes, functions and procedures inside the package and use the package when needed.

```
Package HW3 is
    Constant N: Positive :=4;
    Subtype SInteger is Integer Range -2**(N+1)to 2**(N+1)-1;
    TYPE Integers IS ARRAY (NATURAL RANGE <>) of INTEGER;
    TYPE Vectors IS ARRAY (NATURAL RANGE <>) of Bit_Vector(2 downto 0);
    TYPE Bits IS ARRAY (NATURAL RANGE <>) of Bit;
    Function Bin2Int(Bin: Bit_Vector) return SInteger;
    Procedure Int2Bin (Int: IN SInteger; Bin : OUT BIT_VECTOR);
    Function        "+" ( l,r : Bit_Vector ) RETURN Bit_vector;
    Function        "-" ( l,r : Bit_Vector ) RETURN Bit_vector;

    Procedure Apply_Data (
                    Signal Target: OUT Bit_Vector;
                    Constant Values: IN Integers;
                    Constant Period: IN Time);
    Procedure Apply_Data (
                    Signal Target: OUT Bit_Vector;
                    Constant Values: IN Vectors;
                    Constant Period: IN Time);



    Procedure Apply_Data (
                    Signal Target: OUT Bit;
                    Constant Values: IN Bits;
                    Constant Period: IN Time);


    Function Bin2Ints(Bin: Bit_Vector) return integer;

End;

Package Body HW3 is
    Function Bin2Int(Bin: Bit_Vector) return SInteger is
            variable SUM: SInteger:=0;
    begin
            -- convert to integer as unsigned
            For I IN 0 To (Bin'Length - 1) Loop
                    if Bin(I)='1' then
                            SUM := SUM + (2**I);
                    end if;
            End Loop;
            return SUM;

    end Bin2Int;

    Procedure Int2Bin (Int: IN SInteger; Bin : OUT BIT_VECTOR)  IS
            Variable Tmp: SInteger;
            Constant size: Natural := Bin'length;
    Begin
```

```
        Tmp := Int;
        if (Tmp < 0) Then
                Tmp :=2**size+Tmp;
        End If;
        For I IN 0 To (Bin'Length - 1) Loop
                If ( Tmp MOD 2 = 1) Then
                        Bin(I) := '1';
                Else Bin(I) := '0';
                End If;
                Tmp := Tmp / 2;
        End Loop;
End Int2Bin;

Function        "+" ( l,r : Bit_Vector ) RETURN Bit_vector IS
        Variable IRes: SInteger;
        Variable Result: Bit_Vector(l'length downto 0);
Begin
        IRes := Bin2Int(l) + Bin2Int(r);
        Int2Bin(IRes, Result);
Return Result;
End "+";

Function        "-" ( l,r : Bit_Vector ) RETURN Bit_vector IS
        Variable IRes: SInteger;
        Variable Result: Bit_Vector(l'length downto 0);
Begin
        IRes := Bin2Int(l) - Bin2Int(r);
        Int2Bin(IRes, Result);
        Return Result;
End "-";
Procedure Apply_Data (
                Signal Target: OUT Bit_Vector;
                Constant Values: IN Integers;
                Constant Period: IN Time) IS
                Variable Buf: Bit_Vector(Target'range);
Begin
                For I IN 0 To Values'length-1 Loop
                        Int2Bin (Values(I), Buf);
                        Target <= Transport Buf After I * Period;
                End Loop;
End Apply_Data;

Procedure Apply_Data (
                Signal Target: OUT Bit_Vector;
                Constant Values: IN Vectors;
                Constant Period: IN Time) IS
Begin
                For I IN 0 To Values'length-1 Loop
                        Target <= Transport Values(I) After I * Period;
                End Loop;
End Apply_Data;
Procedure Apply_Data (
                Signal Target: OUT Bit;
                Constant Values: IN Bits;
                Constant Period: IN Time) IS
Begin
                For I IN 0 To Values'length-1 Loop
                        Target <= Transport Values(I) After I * Period;
```

```
                End Loop;
    End Apply_Data;

    Function Bin2Ints(Bin: Bit_Vector) return integer is
        variable SUM: INTEGER:=0;
    begin
        -- convert to integer as unsigned
        For I IN 0 To (Bin'Length - 1) Loop
            if Bin(I)='1' then
                SUM := SUM + (2**I);
            end if;
        End Loop;
        -- if negative
        if (Bin(Bin'Length -1)='1') then
            -- 2's complement
            SUM := 2**(Bin'Length)-SUM;
            -- set the negaive sign
            SUM := -SUM;
        end if;
        return SUM;

    end Bin2Ints;

End;
```

(v) Synthesize the modeled ALU in (ii) using Xilinx Project Navigator and report on the total equivalent gate count for design after mapping and the longest delay in the design based on Post-Map static timing report.

Total equivalent gate=609
Longest delay in the design=11.787 ns .

(vi) Remodel the functions in (i), “+” and “-“, based on performing the operation using a ripple carry add like functionality. Change the ALU model based on the use of these two newly modeled functions and reapply the same test bench modeled in (iii) to verify the correct functionality of the ALU.

```
Architecture BM2 of ALU is

Function    "+" ( l,r : Bit_Vector ) RETURN Bit_vector IS
    Variable Sum: Bit_Vector(l'length downto 0);
    Variable P, G: Bit_Vector(l'length-1 downto 0);
      Variable C: Bit_Vector(l'length downto 0);
Begin
    C(0) := '0';
    For i in 0 to N-1 Loop
        P(i) := l(i) XOR r(i);
        G(i) := l(i) AND r(i);
        Sum(i) := P(i) XOR C(i);
        C(i+1)  := G(i) OR (P(i) AND C(i));
    End Loop;

    Sum(l'length) := C(l'length);
    Return Sum;
End "+";

Function    "-" ( l,r : Bit_Vector ) RETURN Bit_vector IS
```

```
    Variable Sum: Bit_Vector(l'length downto 0);
    Variable P, G: Bit_Vector(l'length-1 downto 0);
      Variable C: Bit_Vector(l'length downto 0);
Begin
    C(0) := '1';
    For i in 0 to N-1 Loop
            P(i) := l(i) XOR Not r(i);
            G(i) := l(i) AND Not r(i);
            Sum(i) := P(i) XOR C(i);
            C(i+1)  := G(i) OR (P(i) AND C(i));
    End Loop;

    Sum(l'length) := C(l'length);
    Return Sum;
End "-";

Begin

    Process(A, B, Cin, Sel)
            Variable Tmp: Bit_Vector (A'length Downto 0);
            Variable Tmp2: Bit_Vector (A'length-1 Downto 0);
            Variable Tmp3 : Bit_Vector (A'length Downto 0);
            Variable Zero: Bit_Vector (A'length-1 Downto 0) := (N-1 Downto 0=>'0');

    Begin
    Case Sel is
            When "000" =>   Tmp := A + B;
                            Cout <= Tmp(N); C <= Tmp(N-1 Downto 0);
                            if (Tmp(N-1 Downto 0) = Zero) Then
                                    ZeroF <= '1';
                            else
                                    ZeroF <= '0';
                            end if;
                            if ( (A(N-1) = B(N-1)) AND (Tmp(N-1) /= A(N-1)) )          Then
                                    OverflowF <= '1';
                            else
                                    OverflowF <= '0';
                            End if;
                            Signf <= Tmp(N-1);
            When "001" =>   Tmp  := A + B;
                            Tmp2 := ((N-1 Downto 1=>'0')&Cin);
                            Tmp3 := Tmp(N-1 Downto 0) + Tmp2;
                            C <= Tmp3(N-1 Downto 0);
                            If (Tmp(N)='1' OR Tmp3(N)='1') Then
                                    Cout <= '1';
                            else
                                    Cout <= '0';
                            end if;
                            if (Tmp3(N-1 Downto 0) = Zero) Then
                                    ZeroF <= '1';
                            else
                                    ZeroF <= '0';
                            end if;
                            if ( (A(N-1) = B(N-1)) AND (Tmp3(N-1) /= A(N-1)) )          Then
                                    OverflowF <= '1';
                            else
                                    OverflowF <= '0';
                            End if;
```

```
                Signf <= Tmp3(N-1);
When "010" =>   Tmp := A - B;
                Cout <= Not Tmp(N); C <= Tmp(N-1 Downto 0);
                if (Tmp(N-1 Downto 0) = Zero) Then
                        ZeroF <= '1';
                else
                        ZeroF <= '0';
                end if;
                if ( (A(N-1) /= B(N-1)) AND (Tmp(N-1) /= A(N-1)) )          Then
                        OverflowF <= '1';
                else
                        OverflowF <= '0';
                End if;
                Signf <= Tmp(N-1);
When "011" =>   Tmp := A - B;
                Tmp2 := ((N-1 Downto 1=>'0')&Cin);
                Tmp3 := Tmp(N-1 Downto 0) - Tmp2;
                C <= Tmp3(N-1 Downto 0);
                If (Tmp(N)='1' OR Tmp3(N)='1') Then
                        Cout <= '0';
                else
                        Cout <= '0';
                end if;
                if (Tmp3(N-1 Downto 0) = Zero) Then
                        ZeroF <= '1';
                else
                        ZeroF <= '0';
                end if;
                if ( (A(N-1) /= B(N-1)) AND (Tmp3(N-1) /= A(N-1)) )         Then
                        OverflowF <= '1';
                else
                        OverflowF <= '0';
                End if;
                Signf <= Tmp3(N-1);
When "100" =>
                Tmp2 := ((N-1 Downto 1=>'0')&'1');
                Tmp3 := B + Tmp2;
                C <= Tmp3(N-1 Downto 0);
                Cout <= Tmp3(N);
                if (Tmp3(N-1 Downto 0) = Zero) Then
                        ZeroF <= '1';
                else
                        ZeroF <= '0';
                end if;
                if ( (B(N-1) = Tmp2(N-1)) AND (Tmp3(N-1) /= B(N-1)) )       Then
                        OverflowF <= '1';
                else
                        OverflowF <= '0';
                End if;
                Signf <= Tmp3(N-1);
When "101" =>
                Tmp2 := ((N-1 Downto 1=>'0')&'1');
                Tmp3 := B - Tmp2;
                C <= Tmp3(N-1 Downto 0);
                Cout <= Not Tmp3(N);
                if (Tmp3(N-1 Downto 0) = Zero) Then
                        ZeroF <= '1';
                else
```

```
                    ZeroF <= '0';
                end if;
                if ( (B(N-1) /= Tmp2(N-1)) AND (Tmp3(N-1) /= B(N-1)) )    Then
                    OverflowF <= '1';
                else
                    OverflowF <= '0';
                End if;
                Signf <= Tmp3(N-1);
        When "110" =>
                Cout <= '0'; C <= B;
                if (B(N-1 Downto 0) = Zero) Then
                    ZeroF <= '1';
                else
                    ZeroF <= '0';
                end if;

                OverflowF <= '0';
                Signf <= B(N-1);
        When "111" =>  Tmp := B + B;
                Cout <= Tmp(N); C <= Tmp(N-1 Downto 0);
                if (Tmp(N-1 Downto 0) = Zero) Then
                    ZeroF <= '1';
                else
                    ZeroF <= '0';
                end if;
                if ( (A(N-1) = B(N-1)) AND (Tmp(N-1) /= A(N-1)) )          Then
                    OverflowF <= '1';
                else
                    OverflowF <= '0';
                End if;
                Signf <= Tmp(N-1);

    End Case;

    End Process;

End;
```

Running the test bench on thos model of the ALU produced the results shown below which are identical to the first model:

ALU Operation	Input A	Input B	Cin	Result	Cout	Signf	OverflowF	ZeroF
C=A+B	5	2		7	0	0	0	0
C=A+B	-8	7		-1	0	1	0	0
C=A+B	7	7		-2	0	1	1	0
C=A+B	-7	-2		7	1	0	1	0
C=A+B	-1	1		0	1	0	0	1
C=A+B	-1	-1		-2	1	1	0	0
C=A+B+Cin	-1	1	1	1	1	0	0	0
C=A+B+Cin	-1	-1	0	-2	1	1	0	0
C=A+B+Cin	-1	-1	1	-1	1	1	0	0
C=A+B+Cin	-1	0	1	0	1	0	0	1
C=A-B	3	4		-1	1	1	0	0
C=A-B	-8	7		1	0	0	1	0
C=A-B	-7	-1		-6	1	1	0	0
C=A-B	-7	2		7	0	0	1	0
C=A-B-Cin	-7	1	1	7	0	0	1	0

C=A-B-Cin	3	2	1	0	0	0	0	1
C=A-B-Cin	-8	1	1	6	0	0	1	0
C=B+1		-1		0	1	0	0	1
C=B+1		1		2	0	0	0	0
C=B+1		7		-8	0	1	1	0
C=B-1		0		-1	1	1	0	0
C=B-1		-1		-2	0	1	0	0
C=B-1		-8		7	0	0	1	0
C=B-1		7		6	0	0	0	0
C=B		-1		-1	0	1	0	0
C=B		7		7	0	0	0	0
C=2*B		-1		-2	1	1	0	0
C=2*B		3		6	0	0	0	0
C=2*B		7		-2	0	1	1	0

(vii) Synthesize the modeled ALU in (vi) using Xilinx Project Navigator and report on the total equivalent gate count for design after mapping and the longest delay in the design based on Post-Map static timing report. Compare the gate count and maximum delay obtained with that obtained in (v). What are your observations and conclusions?

Total equivalent gate= 504.
Longest delay in the design=10.768 ns.

We noticed that this implementation has resulted in less area and less delay. This is because the conversion function from binary to integer in the first ALU has been implemented in hardware which is a costly solution.

(viii) Remodel the functions in (i), "+" and "-", based on performing the operation using a cascaded 4-bit carry-look-ahead like functionality. Change the ALU model based on the use of these two newly modeled functions and reapply the same test bench modeled in (iii) to verify the correct functionality of the ALU.

The two remodeled functions are shown below and the ALU architecture is exactly the same as the one given in (vi).

```
Function     "+" ( l,r : Bit_Vector ) RETURN Bit_vector IS
    Variable Sum: Bit_Vector(l'length downto 0);
    Variable P, G: Bit_Vector(l'length-1 downto 0);
     Variable C: Bit_Vector(l'length downto 0);
Begin
    C(0) := '0';
    For i in 0 to N-1 Loop
            P(i) := l(i) XOR r(i);
            G(i) := l(i) AND r(i);
    End Loop;


    For i in 0 to (N/4)-1 Loop
            C(i*4+1)  := G(i*4+0) OR (P(i*4+0) AND C(i*4+0));
            C(i*4+2)  := G(i*4+1) OR (P(i*4+1) AND G(i*4+0)) OR (P(i*4+1) AND P(i*4+0)
AND C(i*4+0));
            C(i*4+3)  := G(i*4+2) OR (P(i*4+2) AND G(i*4+1)) OR (P(i*4+2) AND P(i*4+1)
AND G(i*4+0)) OR (P(i*4+2) AND P(i*4+1) AND P(i*4+0) AND C(i*4+0));
```

```
            C(i*4+4)  := G(i*4+3) OR (P(i*4+3) AND G(i*4+2)) OR (P(i*4+3) AND P(i*4+2)
AND G(i*4+1)) OR (P(i*4+3) AND P(i*4+2) AND P(i*4+1) AND G(i*4+0)) OR (P(i*4+3) AND
P(i*4+2) AND P(i*4+1) AND P(i*4+0) AND C(i*4+0));
    End Loop;

    For i in 0 to N-1 Loop
            Sum(i) := P(i) XOR C(i);
    End Loop;

    Sum(l'length) := C(l'length);
    Return Sum;
End "+";

Function     "-" ( l,r : Bit_Vector ) RETURN Bit_vector IS
    Variable Sum: Bit_Vector(l'length downto 0);
    Variable P, G: Bit_Vector(l'length-1 downto 0);
      Variable C: Bit_Vector(l'length downto 0);
Begin
    C(0) := '1';
    For i in 0 to N-1 Loop
            P(i) := l(i) XOR NOT r(i);
            G(i) := l(i) AND NOT r(i);
    End Loop;


    For i in 0 to (N/4)-1 Loop
            C(i*4+1)  := G(i*4+0) OR (P(i*4+0) AND C(i*4+0));
            C(i*4+2)  := G(i*4+1) OR (P(i*4+1) AND G(i*4+0)) OR (P(i*4+1) AND P(i*4+0)
AND C(i*4+0));
            C(i*4+3)  := G(i*4+2) OR (P(i*4+2) AND G(i*4+1)) OR (P(i*4+2) AND P(i*4+1)
AND G(i*4+0)) OR (P(i*4+2) AND P(i*4+1) AND P(i*4+0) AND C(i*4+0));
            C(i*4+4)  := G(i*4+3) OR (P(i*4+3) AND G(i*4+2)) OR (P(i*4+3) AND P(i*4+2)
AND G(i*4+1)) OR (P(i*4+3) AND P(i*4+2) AND P(i*4+1) AND G(i*4+0)) OR (P(i*4+3) AND
P(i*4+2) AND P(i*4+1) AND P(i*4+0) AND C(i*4+0));
    End Loop;

    For i in 0 to N-1 Loop
            Sum(i) := P(i) XOR C(i);
    End Loop;

    Sum(l'length) := C(l'length);
    Return Sum;
End "-";
```

(ix) Synthesize the modeled ALU in (viii) using Xilinx Project Navigator and report on the total equivalent gate count for design after mapping and the longest delay in the design based on Post-Map static timing report. Compare the gate count and maximum delay obtained with that obtained in (vii). What are your observations and conclusions?

Total equivalent gate= 528.
Longest delay in the design=12.970 ns.
The obtained area is slightly more than the ripple carry adder and less than the first ALU implementation. However, the delay obtained is the largest while it should be less than the RCA ALU. This is due to the mapping process in FPGAs.