CMPE-630 Digital IC Design
Laboratory Exercise 7

Autolayout Design Techniques (HDL-Layout)

Brandon Key
Performed: 6 Nov 2019
Submitted: 13 Nov 2019

Instructor: Dr. Amlan Ganguly
TAs: Abhishek Vashist
Andrew Fountain
Piers Kwan

By submitting this report, you attest that you neither have given nor have received any assistance (including writing, collecting data, plotting figures, tables or graphs, or using previous student reports as a reference), and you further acknowledge that giving or receiving such assistance will result in a failing grade for this course.

Your Signature: ______________________________

---

1 Abstract

2 Design Methodology and Theory

The world of IC design is a very large and complex one. Engineering time is a critical factor when trying to create designs. An engineer tries to minimize power and cost, while maximizing performance. Modern ICs are very complex and manual engineer effort is not feasible. Fortunately, auto layout tools exists so that non-critical parts can be generated quickly. In this exercise, a single bit ALU and a 16-bit ALU were designed, automatically laid out and then power and timing results were extracted.

For all components designed in this exercise, VHDL was written to describe the functionality of the component. Leonardo Spectrum was used to turn the VHDL into synthesizable logic. The VHDL was then functionally tested with a test bench using Questa Sim.

2.1 Functional Simulation

The ALUs in this exercise had a simple 2 bit op-code which can be seen in Table 1.

Table 1: ALU Operations

OpCode	Operation	Operands
00	AND	A AND B
01	OR	A OR B
10	ADD	A + B
11	SUB	A - B

2.1.1 1 Bit ALU

The 1 Bit ALU designed in this exercise was created from behavioral VHDL (see Listing 10). A Questa Sim simulation was performed to test the functionality of the 1 bit ALU. The test bench can be seen in Listing 9. The test bench went through every op-code and every input. The resulting waveforms can be seen in Figure 1.

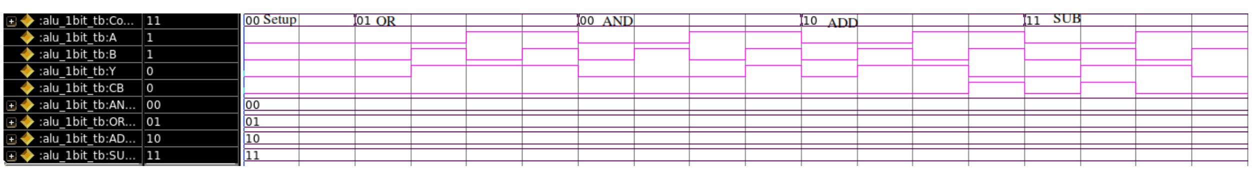

Figure 1: Functional Simulation of 1-bit ALU

The 1 Bit ALU functioned properly.

2.1.2 16 Bit ALU

The 16-bit ALU was created structurally with generically large structures. The VHDL that describes the ALU can be seen in Listing 12. While the 1-bit ALU could have a full test bench that tested every input, the 16-bit ALU was far too large to do the same. Instead, a few test cases were selected to test each function of the ALU. Each component in the ALU was previously tested, so the main goal was to test the setup of the ALU itself. The testbench code can be seen in Listing 4. The AND functionality was tested first and captured in Figure 2.

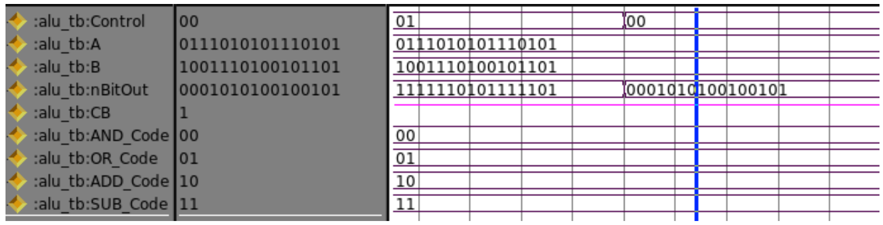

Figure 2: Functional Simulation of 16-bit ALU: AND

Once the and functionality was verified, the OR functionality was tested and recorded in Figure 3.

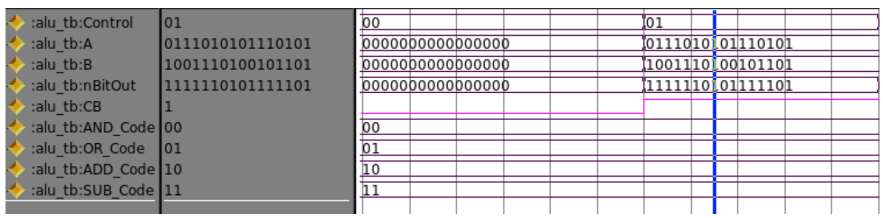

Figure 3: Functional Simulation of 16-bit ALU: OR

x

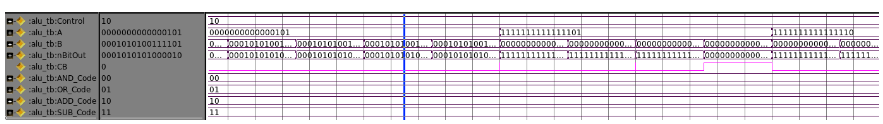

Figure 4: Functional Simulation of 16-bit ALU: Addition

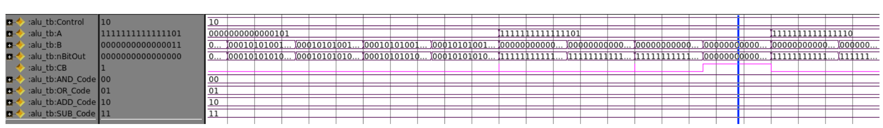

Figure 5: Functional Simulation of 16-bit ALU: Addition with carry

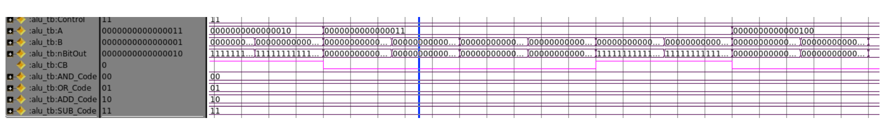

Figure 6: Functional Simulation of 16-bit ALU: Subtraction

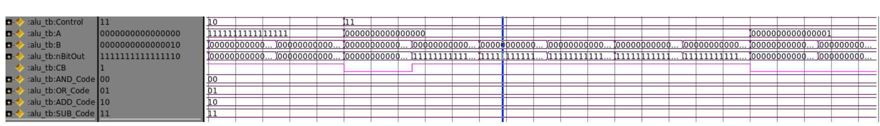

Figure 7: Functional Simulation of 16-bit ALU: Subtraction with negative result

2.2 Schematic

2.2.1 1-Bit ALU

Behavioral

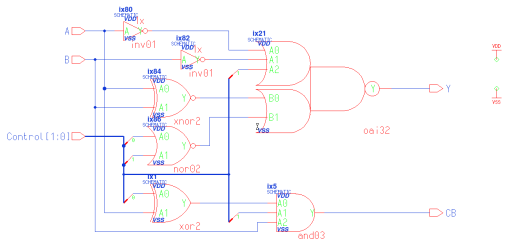

Figure 8: 1 Bit ALU Schematic

2.2.2 n-Bit ALU

Structural

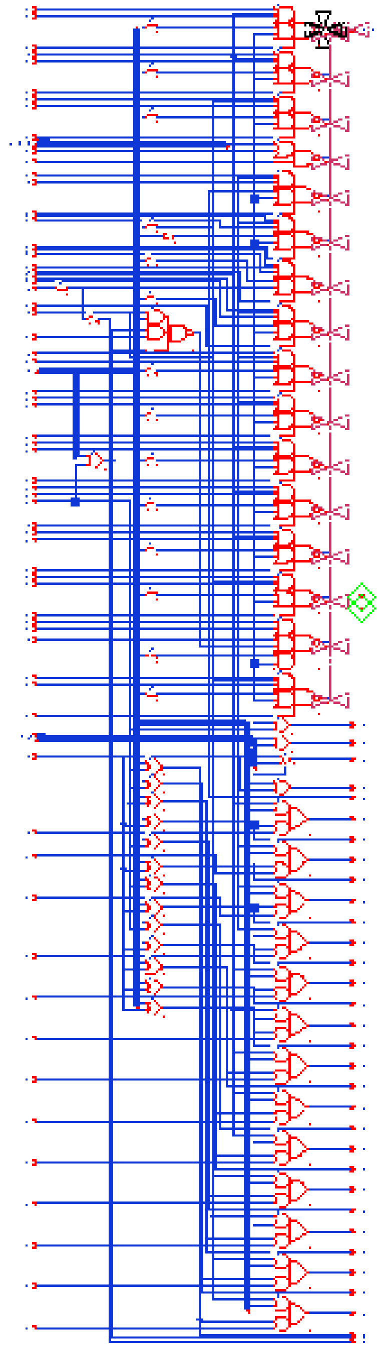

Figure 9: 16 Bit ALU Schematic Page 1

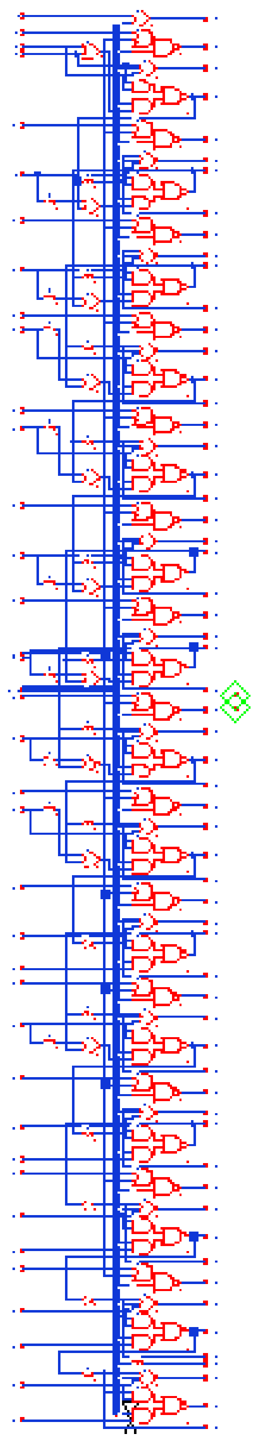

Figure 10: 16 Bit ALU Schematic Page 2

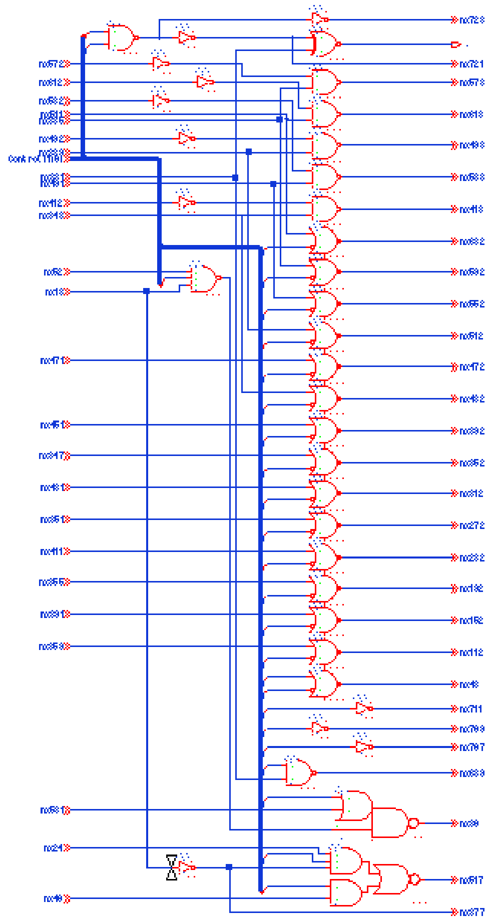

Figure 11: 16 Bit ALU Schematic Page 3

3 Results and Analysis

3.1 Layout

3.1.1 1 Bit ALU

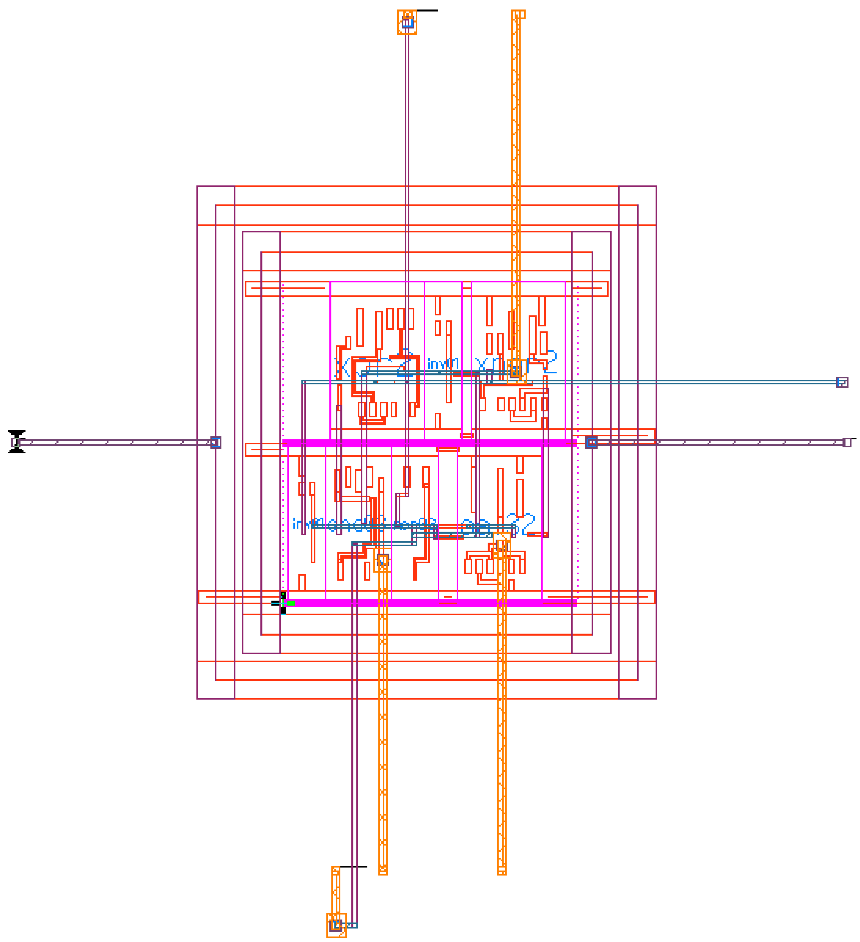

Figure 12: 1 Bit ALU Layout

3.1.2 16 Bit ALU

Area 0.7

Power Routing

- Varying levels of routing completion time
- Slight preference for jogs over via to fill the area.
- Rip
- Under rip options:

  Rips Most Aggressive

  Automatic Rip Passes

  Reroute

- Under Advanced:

    Allow all directions for stubs

    Via Options ¿ Use via generator

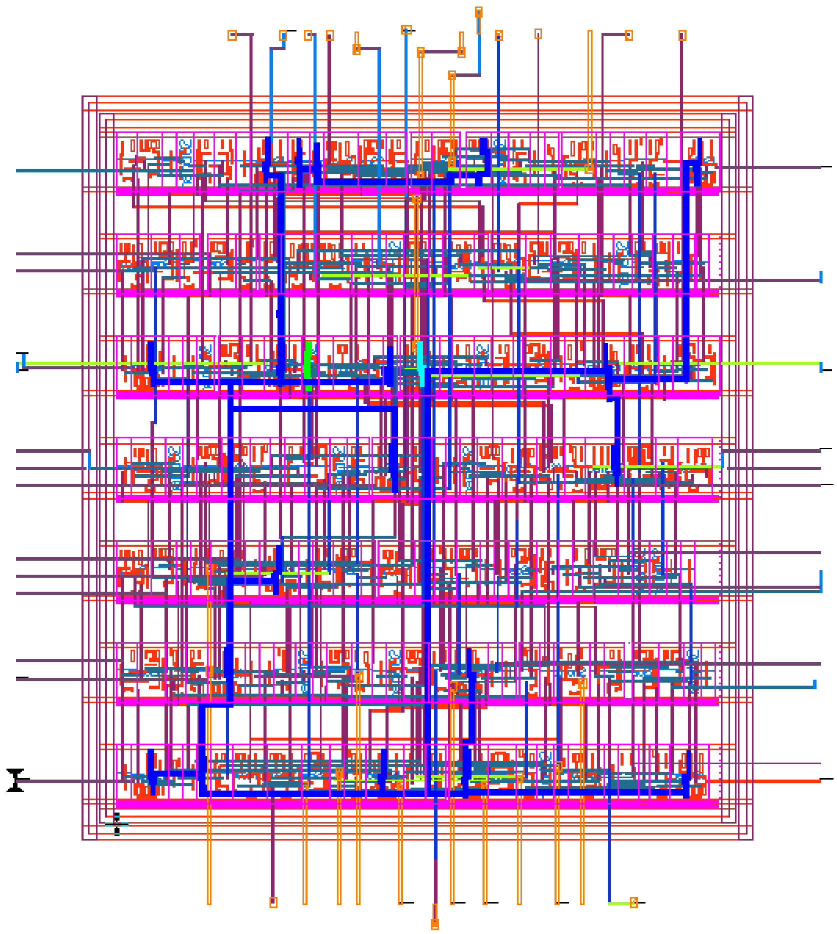

Figure 13: 16 Bit ALU Layout

3.2 Timing

3.2.1 1 Bit ALU

It was found that subtraction was by far the slowest operation, with the timing difference visible in the waveforms.

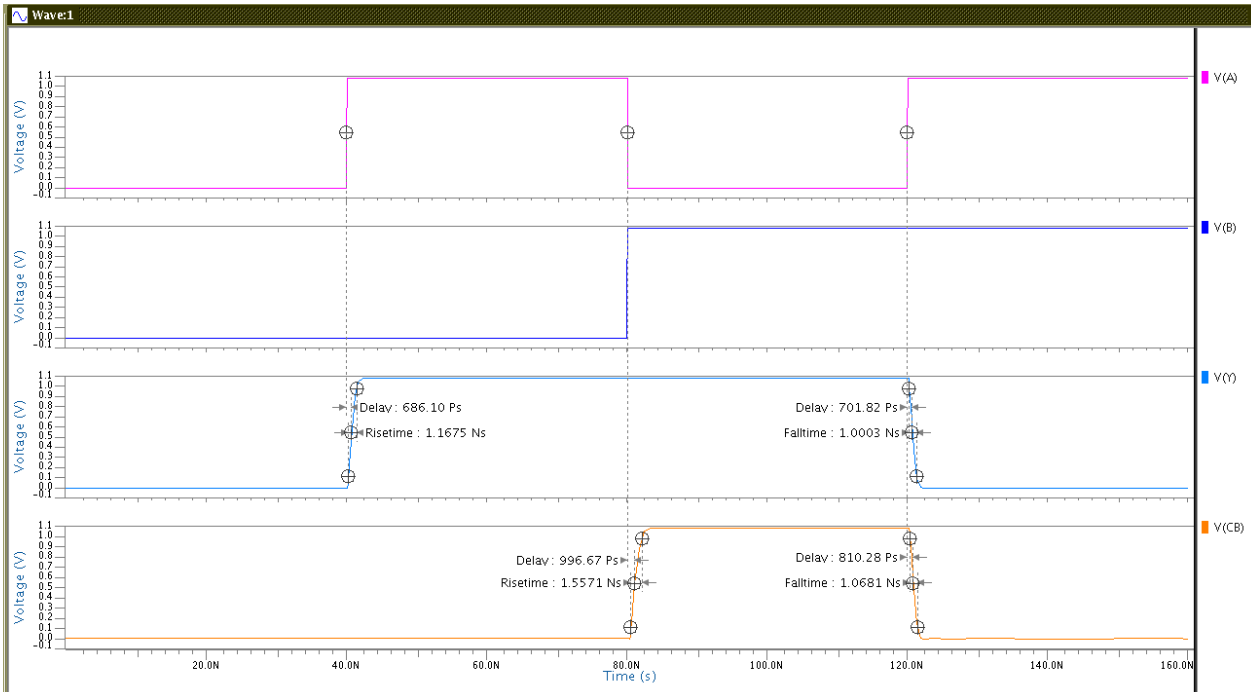

Figure 14: 1 Bit ALU Worst Case Timing Simulation

Table 2: 1-Bit ALU Worst Case Rise Time

Output	**Rise Time (ps)**	**A**	**B**	**Operation**
Y	1167.5	1	0	SUB
Carry	1557.1	0	1	SUB

Table 3: 1-Bit ALU Worst Case Fall Time

Output	**Fall Time (ps)**	**A**	**B**	**Operation**
Y	1001.3	1	1	SUB
Carry	1068.1	1	1	SUB

Table 4: 1-Bit ALU Worst Case Propagation Time High to Low

Output	**Tp,HL (ps)**	**A**	**B**	**Operation**
Y	701.8	1	1	SUB
Carry	810.3	1	1	SUB

Table 5: 1-Bit ALU Worst Case Propagation Time Low to High

Output	**Tp,LH (ps)**	**A**	**B**	**Operation**
Y	686.1	1	0	SUB
Carry	996.7	0	1	SUB

3.2.2 16 Bit ALU

Figure 15: Simulation with Incorrect Forces

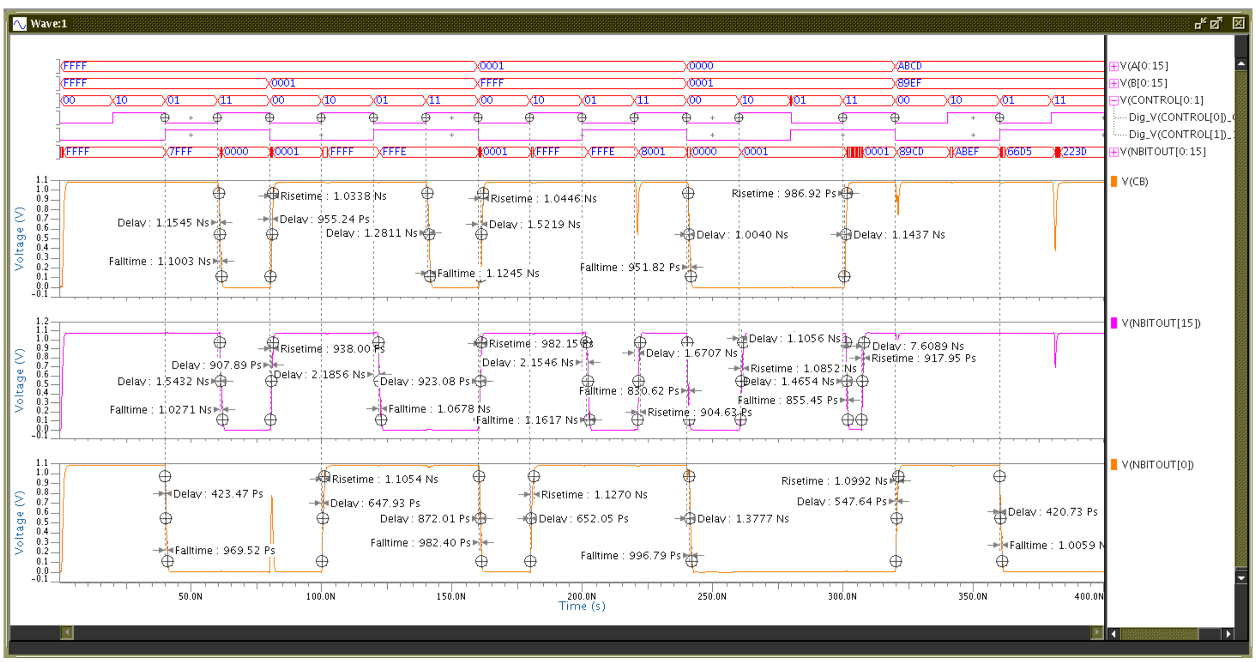

Figure 16: 16-Bit ALU Timing Waveforms

Table 6: 16-Bit ALU Worst Case Rise Time

Input		Output	Rise Time (ps)			
A	**B**	**Y**	**Op**	**Y**	**Op**	**CB**
0x0000	0x0000	Y[15]	11	938.0	11	1033.8
0xFFFF	0xFFFF	Y[15]	11	938.0	01	1124.5
0xFFFF	0x0001	Y[15]	11	982.2	11	1044.6
0x0001	0xFFFF	Y[15]	01	904.63	00	1044.6
0x0000	0x0001	Y[0]	00	1085.2	01	986.9
0xABCD	0x89EF	Y[15]	01	917.9	11	951.8

Table 7: 16-Bit ALU Worst Case Fall Time

Input		Output	Fall Time (ps)			
A	**B**	**Y**	**Op**	**Y**	**Op**	**CB**
0x0000	0x0000	Y[15]	01	1100.3	01	1027.1
0xFFFF	0xFFFF	Y[15]	10	1067.0	10	1033.8
0xFFFF	0x0001	Y[15]	11	830.6	01	1124.5
0x0001	0xFFFF	Y[15]	10	1161.7	11	951.8
0x0000	0x0001	Y[0]	00	996.8	10	1005.9
0xABCD	0x89EF	Y[15]	01	855.5	01	969.5

Table 8: 16-Bit ALU Worst Case Propagation Time High to Low

Input		Output	Tp,HL (ps)			
A	**B**	**Y**	**Op**	**Y**	**Op**	**CB**
0x0000	0x0000	Y[15]	01	907.9	01	1154.5
0xFFFF	0xFFFF	Y[15]	10	2185.6	10	1281.1
0xFFFF	0x0001	Y[15]	11	2154.6	01	1004.0
0x0001	0xFFFF	Y[15]	10	1670.7	11	1143.7
0x0000	0x0001	Y[0]	00	1377.7	10	920.7
0xABCD	0x89EF	Y[15]	01	1465.4	01	872.0

Table 9: 16-Bit ALU Worst Case Propagation Time Low to High

Input		Output	Tp,LH (ps)			
A	**B**	**Y**	**Op**	**Y**	**Op**	**CB**
0x0000	0x0000	Y[15]	11	2185.6	11	955.2
0xFFFF	0xFFFF	Y[15]	11	2154.6	01	1521.9
0xFFFF	0x0001	Y[15]	11	1670.7	11	1143.7
0x0001	0xFFFF	Y[15]	01	1085.2	00	947.6
0x0000	0x0001	Y[0]	00	1099.2	01	1047.9
0xABCD	0x89EF	Y[15]	01	7608.9	11	1127.0

3.3 Power

3.3.1 1 Bit ALU

3.3.2 16 Bit ALU

4 Conclusion

5 Appendix

5.1 VHDL

Listing 1: Controller-16Bit VHDL

```
--------------------------------------------------------------------------------
--Company        : RIT
--Author         : Brandon Key
--Created        : 02/18/2018
--
--Project Name : Lab 3
--File           : Controller_16Bit.vhd
--
--Entity         : Controller_16Bit
--Architecture : behav
--
--Tool Version : VHDL '93
--Description    : *SPECIAL controller, DO NOT USE OUTSIDE THIS PROJECT*
--               : Takes 4 bit control signal bit
--               : Figues out the proper output
-------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
--use work.controlcodes.all;

entity Controller_16Bit is
    generic (n : integer := 16);
    port(
        Control : in std_logic_vector(1 downto 0);

        ADD_SUB_In : in std_logic_vector(N-1 downto 0);
        OR_In      : in std_logic_vector(N-1 downto 0);
        AND_In     : in std_logic_vector(N-1 downto 0);

        ADD_SUB_SEL : out std_logic;

        nBitOut : out std_logic_vector(N-1 downto 0)
        );
end Controller_16Bit;

architecture behav of Controller_16Bit is

    constant AND_Code : std_logic_vector(1 downto 0) := "00";
    constant OR_Code  : std_logic_vector(1 downto 0) := "01";
    constant ADD_Code : std_logic_vector(1 downto 0) := "10";
    constant SUB_Code : std_logic_vector(1 downto 0) := "11";
```

```vhdl
begin

    --Proces to set the select signal when subtraction should occur
    ADD_SUB_SEL_proc: with Control select
        ADD_SUB_SEL <= '1' when SUB_Code,
                       '0' when others;

    nBitOut_proc: with Control select
        nBitOut <= ADD_SUB_In when ADD_Code,
                   ADD_SUB_In when SUB_Code,
                   OR_In when OR_Code,
                   AND_In when AND_Code,
                   (others => '0') when others;

end behav;
```

Listing 2: nBitAdderSubtractor-4Bit VHDL

```vhdl
---------------------------------------------------------------------------
--Company      : RIT
--Author       : Brandon Key
--Created      : 02/18/2018
--
--Project Name : Lab 3
--File         : nBitAdderSubtractor_4Bit.vhd
--
--Entity       : nBitAdderSubtractor_4Bit
--Architecture : struct
--
--Tool Version : VHDL '93
--Description  : Entity and structural description of an adder subtractor
--             : SEL = 0 : A+B = Y
--             : SEL = 1 : A-B = Y
---------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

entity nBitAdderSubtractor_4Bit is
    generic (n : integer := 16);
    port(
        A,B : in  std_logic_vector(n-1 downto 0);
        SEL : in  std_logic;
        Y   : out std_logic_vector(n-1 downto 0);
        CB  : out std_logic
        );
end nBitAdderSubtractor_4Bit;

architecture struct of nBitAdderSubtractor_4Bit is

    component full_adder is
        port(A,B,Cin : in  std_logic;
            Sum,Cout : out std_logic
        );
     end component full_adder;

    --Create an array to hold all of the carries
    type carry_array is array (n-1 downto 0) of std_logic;
```

```vhdl
    signal c_array : carry_array;

    signal B_XOR_SEL : std_logic_vector( (n-1) downto 0);

begin

    --Generate the xor statements to be mapped to the full adders
    XORator : for i in 0 to n-1 generate
        B_XOR_SEL(i) <= B(i) xor SEL;
    end generate XORator;

    generate_adders : for i in 0 to n-1  generate
        i_first: if i = 0 generate
            --The first adder gets SEL as the Cin
            adder : full_adder port map(
                A => A(i),
                B => B_XOR_SEL(i),
                Cin => SEL,
                Sum => Y(i),
                Cout => c_array(i)
            );
        end generate i_first;

        i_last : if i = (n-1) generate
            --The last adder doesn't have a carry out
            adder : full_adder port map(
                A => A(i),
                B => B_XOR_SEL(i),
                Cin => c_array(i-1),
                Sum => Y(i),
                Cout =>c_array(i)
            );
        end generate i_last;

        --Middle adders
        i_mid : if (i /= 0) and (i /= (n-1)) generate
            adder : full_adder port map(
                A => A(i),
                B => B_XOR_SEL(i),
                Cin => c_array(i-1),
                Sum => Y(i),
                Cout => c_array(i)
            );
        end generate i_mid;

    end generate generate_adders;

    CB <= c_array(n-1) xor SEL;

end  struct;
```

Listing 3: FullAdder VHDL

```vhdl
-------------------------------------------------------------------------------
--Company       : RIT
--Author        : Brandon Key
--Created       : 02/18/2018
--
--Project Name : Lab 3
```

```
--File          : Full_Adder.vhd
--
--Entity        : Full_Adder
--Architecture  : behav
--
--Tool Version  : VHDL '93
--Description   : Entity and behavural description of a full adder
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Full_Adder is
    port(A,B,Cin : in std_logic;
        Sum,Cout : out std_logic
        );
end Full_Adder;

architecture behav of Full_Adder is
begin
    --uses select assignment to implement the truth table of a full adder


    sum_proc: with std_logic_vector'(Cin&A&B) select
        Sum <= '0' when "000",
               '1' when "001",
               '1' when "010",
               '0' when "011",
               '1' when "100",
               '0' when "101",
               '0' when "110",
               '1' when "111",
               '0' when others;

    Cout_proc: with std_logic_vector'(Cin&A&B) select
        Cout <= '0' when "000",
                '0' when "001",
                '0' when "010",
                '1' when "011",
                '0' when "100",
                '1' when "101",
                '1' when "110",
                '1' when "111",
                '0' when others;

end behav;
```

Listing 4: ALU-16Bit-tb VHDL

```
----------------------------------------------------------------------------------
-- Company: RIT
-- Engineer: Brandon Key
--
-- Create Date:    17:51:58 02/28/2018
-- Design Name:
-- Module Name:    /home/ise/DSDII/Lab/Lab3/SourceCode/ALU_16Bit_tb.vhd
-- Project Name:   Lab3
-- Target Device:
```

```vhdl
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: ALU_16Bit
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
--use work.globals.all;
--use work.controlcodes.all;

ENTITY ALU_16Bit_tb IS
END ALU_16Bit_tb;

ARCHITECTURE behavior OF ALU_16Bit_tb IS

    constant AND_Code : std_logic_vector(1 downto 0) := "00";
    constant OR_Code  : std_logic_vector(1 downto 0) := "01";
    constant ADD_Code : std_logic_vector(1 downto 0) := "10";
    constant SUB_Code : std_logic_vector(1 downto 0) := "11";

    type testRecordArray is array (natural range <>) of std_logic_vector(2 downto 0)
    ;

    constant n:integer := 16;
    -- "Time" that will elapse between test vectors we submit to the component.
    constant TIME_DELTA : time := 50 ns;


    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT ALU_16Bit
    PORT(
         Control : IN  std_logic_vector(1 downto 0);
         A : IN  std_logic_vector(N-1 downto 0);
         B : IN  std_logic_vector(N-1 downto 0);
         nBitOut : OUT  std_logic_vector(N-1 downto 0);
         CB : OUT std_logic
        );
    END COMPONENT;


   --Inputs
   signal Control : std_logic_vector(1 downto 0) := (others => '0');
   signal A : std_logic_vector(N-1 downto 0) := (others => '0');
   signal B : std_logic_vector(N-1 downto 0) := (others => '0');
```

```vhdl
    --Outputs
   signal nBitOut : std_logic_vector(N-1 downto 0);
   signal CB : std_logic;
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name


BEGIN

    -- Instantiate the Unit Under Test (UUT)
   uut: ALU_16Bit
   PORT MAP (
          Control => Control,
          A => A,
          B => B,
          nBitOut => nBitOut,
          CB => CB
        );


   -- Stimulus process
   stim_proc: process
      --create a function to make a vector a strung
      function vec2str(vec : std_logic_vector) return string is
        variable stmp:string(vec'left+1 downto 1);
      begin
        for i in vec'reverse_range loop
          if vec(i) = '1' then
            stmp(i+1) := '1';
          elsif vec(i) = 'U' then
            stmp(i+1) := 'U';
          else
            stmp(i+1) := '0';
          end if;
        end loop;
        return stmp;
      end vec2str;

        procedure check_add(
            constant in1 : in natural;
            constant in2 : in natural;
            constant res_expected : in natural;
            constant CB_expected : in std_logic) is
            variable res : natural;
            begin
            -- Assign values to circuit inputs.
            A <= std_logic_vector(to_unsigned(in1, A'length));
            B <= std_logic_vector(to_unsigned(in2, B'length));
            Control <= ADD_Code;


            wait for TIME_DELTA;
            -- Check output against expected result.
            res := to_integer(unsigned(nBitOut));
            assert ((res = res_expected) and ( CB = CB_expected ))
            report "" & integer'image(in1) & "+" &
                    integer'image(in2) & "=" &
                    integer'image(res_expected) & "!=" &
                    integer'image(res) &
```

```
                "    " &
                "CB exp: " & std_logic'image(CB_expected) &
                "Got: " & std_logic'image(CB)
        severity error;
    end procedure check_add;

    procedure check_sub(
        constant in1 : in natural;
        constant in2 : in natural;
        constant res_expected : in natural;
        constant CB_expected : in std_logic) is
        variable res : natural;
        begin
        -- Assign values to circuit inputs.
        A <= std_logic_vector(to_unsigned(in1, A'length));
        B <= std_logic_vector(to_unsigned(in2, B'length));
        Control <= SUB_Code;


        wait for TIME_DELTA;
        -- Check output against expected result.
        res := to_integer(unsigned(nBitOut));
        assert ((res = res_expected) and ( CB = CB_expected ))
        report "" & integer'image(in1) & "-" &
                integer'image(in2) & "=" &
                integer'image(res_expected) & "!=" &
                integer'image(res) &
                "    " &
                "CB exp: " & std_logic'image(CB_expected) &
                "Got: " & std_logic'image(CB)
        severity error;
    end procedure check_sub;


    procedure check_or(
        constant in1 : in natural;
        constant in2 : in natural;
        constant res_expected : in natural) is
        variable res : natural;
        begin
        -- Assign values to circuit inputs.
        A <= std_logic_vector(to_unsigned(in1, A'length));
        B <= std_logic_vector(to_unsigned(in2, B'length));
        Control <= OR_Code;


        wait for TIME_DELTA;
        -- Check output against expected result.
        res := to_integer(unsigned(nBitOut));
        assert ((res = res_expected) and ( CB = '0' ))
        report "" & integer'image(in1) & "+" &
                integer'image(in2) & "=" &
                integer'image(res_expected) & "!=" &
                integer'image(res) &
                "    " &
                "CB: " & std_logic'image(CB)
        severity error;
    end procedure check_or;
```

```

      procedure check_and(
          constant in1 : in natural;
          constant in2 : in natural;
          constant res_expected : in natural) is
          variable res : natural;
          begin
          -- Assign values to circuit inputs.
          A <= std_logic_vector(to_unsigned(in1, A'length));
          B <= std_logic_vector(to_unsigned(in2, B'length));
          Control <= AND_Code;


          wait for TIME_DELTA;
          -- Check output against expected result.
          res := to_integer(unsigned(nBitOut));
          report "" & integer'image(in1) & "+" &
                 integer'image(in2) & "=" &
                 integer'image(res_expected);

          assert ((res = res_expected) and ( CB = '0' ))
          report  "!=" &
                 integer'image(res) &
                 "     " &
                 "CB: " & std_logic'image(CB)
          severity error;
      end procedure check_and;

  begin


    --wait for the outputs to stabilize
    wait for 100 ns;

    --check_add(4,5,9,0);
    --check_add(65535, 2, 1, 1);
    --check_sub(1234, 234, 1000, 0);
    --check_sub(1, 2, 1, 1);

    control <= OR_Code;
    A <= "0111010101110101";
    B <= "1001110100101101";
    wait for 50 ns;

    control <= AND_Code;
    wait for 50 ns;


    -- Test adder
   for x in (0) to (5) loop
      for y in 5432 to 5438 loop
          control <= ADD_Code;
          A <= std_logic_vector(to_unsigned(x, A'length));
          B <= std_logic_vector(to_unsigned(y, B'length));
          wait for 50 ns;

          assert(nBitOut = std_logic_vector(to_unsigned(x+y, A'length) ) )
          report("Bad Add = " & vec2str(nBitOut)
                 & " expected = " & vec2str( std_logic_vector(to_unsigned(x+y, A'
```

```
length)) )
                    & " A = " & vec2str(A)
                    & " B = " & vec2str(B)
             );
        end loop;
    end loop;

    for x in ((2**N)-3) to ((2**N)-1) loop
        for y in 0 to 3 loop
            control <= ADD_Code;
            A <= std_logic_vector(to_unsigned(x, A'length));
            B <= std_logic_vector(to_unsigned(y, B'length));
            wait for 50 ns;

            assert(nBitOut = std_logic_vector(to_unsigned(x+y, A'length) ) )
            report("Bad Add = " & vec2str(nBitOut)
                    & " expected = " & vec2str( std_logic_vector(to_unsigned(x+y, A'
length)) )
                    & " A = " & vec2str(A)
                    & " B = " & vec2str(B)
             );
        end loop;
    end loop;

    -- Test suber
    for x in 0 to 5 loop
        for y in 0 to 5 loop
        control <= SUB_Code;
        A <= std_logic_vector(to_unsigned(x, A'length));
        B <= std_logic_vector(to_unsigned(y, B'length));
        wait for 50 ns;

        assert(nBitOut = std_logic_vector(to_signed(x-y, A'length)) )
        report("Bad Sub = " & vec2str(nBitOut)
            & " expected = " & vec2str( std_logic_vector(to_signed(x-y, A'length)) )
            & " A = " & vec2str(A)
            & " B = " & vec2str(B)
        );

        end loop;
    end loop;

    for x in 12345 to 12350 loop
        for y in 5 to 7 loop
        control <= SUB_Code;
        A <= std_logic_vector(to_unsigned(x, A'length));
        B <= std_logic_vector(to_unsigned(y, B'length));
        wait for 50 ns;

        assert(nBitOut = std_logic_vector(to_signed(x-y, A'length)) )
        report("Bad Sub = " & vec2str(nBitOut)
            & " expected = " & vec2str( std_logic_vector(to_signed(x-y, A'length)) )
            & " A = " & vec2str(A)
            & " B = " & vec2str(B)
        );

        end loop;
    end loop;

```

```
      wait;
   end process;

END;
```

Listing 5: Controller-4Bit VHDL

```
-----------------------------------------------------------------------------------------
--Company       : RIT
--Author        : Brandon Key
--Created       : 02/18/2018
--
--Project Name : Lab 3
--File          : Controller_4Bit.vhd
--
--Entity        : Controller_4Bit
--Architecture : behav
--
--Tool Version : VHDL '93
--Description   : *SPECIAL controller, DO NOT USE OUTSIDE THIS PROJECT*
--              : Takes 4 bit control signal bit
--              : Figues out the proper output
-----------------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
--use work.controlcodes.all;

entity Controller_4Bit is
    generic (n : integer := 16);
    port(
        Control : in std_logic_vector(1 downto 0);

        ADD_SUB_In : in std_logic_vector(N-1 downto 0);
        OR_In      : in std_logic_vector(N-1 downto 0);
        AND_In     : in std_logic_vector(N-1 downto 0);

        ADD_SUB_SEL : out std_logic;

        nBitOut : out std_logic_vector(N-1 downto 0)
        );
end Controller_4Bit;

architecture behav of Controller_4Bit is

    constant AND_Code : std_logic_vector(1 downto 0) := "00";
    constant OR_Code  : std_logic_vector(1 downto 0) := "01";
    constant ADD_Code : std_logic_vector(1 downto 0) := "10";
    constant SUB_Code : std_logic_vector(1 downto 0) := "11";

begin

    --Proces to set the select signal when subtraction should occur
    ADD_SUB_SEL_proc: with Control select
        ADD_SUB_SEL <= '1' when SUB_Code,
```

```
                          '0' when others;

    nBitOut_proc: with Control select
        nBitOut <= ADD_SUB_In when ADD_Code,
                   ADD_SUB_In when SUB_Code,
                   OR_In when OR_Code,
                   AND_In when AND_Code,
                   (others => '0') when others;

end behav;
```

Listing 6: nBitOR-4Bit VHDL

```
--------------------------------------------------------------------------------
--Company        : RIT
--Author         : Brandon Key
--Created        : 1/22/2018
--
--Project Name : Lab 1
--File           : nBitOR_4Bit.vhd
--
--Entity         : nBitOR_4Bit
--Architecture : Dataflow
--
--Tool Version : VHDL '93
--Description    : Entity and structural description of an OR gate
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity nBitOR_4Bit is
    generic (n : integer := 16);
    port(A,B : in std_logic_vector(n-1 downto 0);
        Y : out std_logic_vector(n-1 downto 0)
        );
end nBitOR_4Bit;

architecture Dataflow of nBitOR_4Bit is
    begin
    Y <= A or B;-- bitwise or
end Dataflow;
```

Listing 7: ALU-4Bit VHDL

```
--------------------------------------------------------------------------------
--Company        : RIT
--Author         : Brandon Key
--Created        : 02/18/2018
--
--Project Name : Lab 3
--File           : ALU_4Bit.vhd
--
--Entity         : ALU_4Bit
--Architecture : struct
--
--Tool Version : VHDL '93
--Description    : ALU_4Bit
--------------------------------------------------------------------------------
```

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

package globals is
    constant N : integer := 16;
end globals;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

package controlcodes is
    constant AND_Code : std_logic_vector(1 downto 0) := "00";
    constant OR_Code  : std_logic_vector(1 downto 0) := "01";
    constant ADD_Code : std_logic_vector(1 downto 0) := "10";
    constant SUB_Code : std_logic_vector(1 downto 0) := "11";
end controlcodes;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
use work.controlcodes.all;
use work.globals.all;

entity ALU_4Bit is
    port(
        Control : in std_logic_vector(1 downto 0);
        A,B     : in std_logic_vector(N-1 downto 0);
        nBitOut : out std_logic_vector(N-1 downto 0);
        CB      : out std_logic
        );
end ALU_4Bit;

architecture struct of ALU_4Bit is

    --constant N : integer := 4;

    signal ADD_SUB_Out : std_logic_vector(N-1 downto 0);
    signal OR_Out      : std_logic_vector(N-1 downto 0);
    signal AND_Out     : std_logic_vector(N-1 downto 0);

    signal ADD_SUB_SEL : std_logic;


begin

    nBitAdderSubtractor_4Bit : entity work.nBitAdderSubtractor_4Bit
        generic map (N => N)
        port map ( A => A, B => B, SEL => ADD_SUB_SEL, Y => ADD_SUB_Out, CB => CB);


    nBitOR_4Bit : entity work.nBitOR_4Bit
        generic map (N => N)
        port map ( A => A, B => B, Y => OR_Out);


    nBitAND_4Bit : entity work.nBitAND_4Bit
```

```
            generic map (N => N)
            port map ( A => A, B => B, Y => AND_Out);

        Controller_4Bit : entity work.Controller_4Bit
            generic map (N => N)
            port map(
            Control     => Control,
            ADD_SUB_In  => ADD_SUB_Out,
            OR_In       => OR_Out,
            AND_In      => AND_Out,
            ADD_SUB_SEL => ADD_SUB_SEL,
            nBitOut     => nBitOut
            );

end struct;
```

Listing 8: nBitAND-4Bit VHDL

```
------------------------------------------------------------------------------------
--Company       : RIT
--Author        : Brandon Key
--Created       : 1/22/2018
--
--Project Name  : Lab 1
--File          : nBitAND_4Bit.vhd
--
--Entity        : nBitAND_4Bit
--Architecture  : Dataflow
--
--Tool Version  : VHDL '93
--Description   : Entity and structural description of an AND gate
------------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity nBitAND_4Bit is
    generic (n : integer := 16);
    port(A,B : in std_logic_vector(n-1 downto 0);
        Y : out std_logic_vector(n-1 downto 0)
        );
end nBitAND_4Bit;

architecture Dataflow of nBitAND_4Bit is
    begin
    Y <= A AND B;-- bitwise or
end Dataflow;
```

Listing 9: ALU-1Bit-tb VHDL

```
--------------------------------------------------------------------------------
-- Company: RIT
-- Engineer: Brandon Key
--
-- Create Date: 17:51:58 02/28/2018
-- Design Name:
-- Module Name: /home/ise/DSDII/Lab/Lab3/SourceCode/ALU_1Bit_tb.vhd
-- Project Name: Lab3
-- Target Device:
```

```vhdl
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: ALU_1Bit
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test. Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
--use work.globals.all;
--use work.controlcodes.all;

ENTITY ALU_1Bit_tb IS
END ALU_1Bit_tb;

ARCHITECTURE behavior OF ALU_1Bit_tb IS

    CONSTANT AND_Code : std_logic_vector(1 DOWNTO 0) := "00";
    CONSTANT OR_Code : std_logic_vector(1 DOWNTO 0) := "01";
    CONSTANT ADD_Code : std_logic_vector(1 DOWNTO 0) := "10";
    CONSTANT SUB_Code : std_logic_vector(1 DOWNTO 0) := "11";

    TYPE testRecordArray IS ARRAY (NATURAL RANGE <>) OF std_logic_vector(2 DOWNTO 0)
    ;
    CONSTANT TIME_DELTA : TIME := 50 ns;
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT ALU_1Bit
        PORT (
            Control : IN std_logic_vector(1 DOWNTO 0);
            A : IN std_logic;
            B : IN std_logic;
            Y : OUT std_logic;
            CB : OUT std_logic
        );
    END COMPONENT;
    --Inputs
    SIGNAL Control : std_logic_vector(1 DOWNTO 0) := (OTHERS => '0');
    SIGNAL A : std_logic := '0';
    SIGNAL B : std_logic := '0';

    --Outputs
    SIGNAL Y : std_logic;
    SIGNAL CB : std_logic;
BEGIN
    -- Instantiate the Unit Under Test (UUT)
    uut : ALU_1Bit
    PORT MAP(
```

```
        Control => Control,
        A => A,
        B => B,
        Y => Y,
        CB => CB
    );
    -- Stimulus process
    stim_proc : PROCESS
        --create a function to make a vector a strung
        FUNCTION vec2str(vec : std_logic_vector) RETURN STRING IS
        VARIABLE stmp : STRING(vec'LEFT + 1 DOWNTO 1);
    BEGIN
        FOR i IN vec'reverse_range LOOP
            IF vec(i) = '1' THEN
                stmp(i + 1) := '1';
            ELSIF vec(i) = 'U' THEN
                stmp(i + 1) := 'U';
            ELSE
                stmp(i + 1) := '0';
            END IF;
        END LOOP; RETURN stmp;
    END vec2str;

    BEGIN
        --wait for the outputs to stabilize
        WAIT FOR 100 ns;

        control <= OR_Code;
        A <= '0';
        B <= '0';
        WAIT FOR 50 ns;
        A <= '0';
        B <= '1';
        WAIT FOR 50 ns;
        A <= '1';
        B <= '0';
        WAIT FOR 50 ns;
        A <= '1';
        B <= '1';
        WAIT FOR 50 ns;

        control <= AND_Code;
        A <= '0';
        B <= '0';
        WAIT FOR 50 ns;
        A <= '0';
        B <= '1';
        WAIT FOR 50 ns;
        A <= '1';
        B <= '0';
        WAIT FOR 50 ns;
        A <= '1';
        B <= '1';
        WAIT FOR 50 ns;

        control <= ADD_Code;
        A <= '0';
        B <= '0';
        WAIT FOR 50 ns;
```

```
        A <= '0';
        B <= '1';
        WAIT FOR 50 ns;
        A <= '1';
        B <= '0';
        WAIT FOR 50 ns;
        A <= '1';
        B <= '1';
        WAIT FOR 50 ns;

        control <= SUB_Code;
        A <= '0';
        B <= '0';
        WAIT FOR 50 ns;
        A <= '0';
        B <= '1';
        WAIT FOR 50 ns;
        A <= '1';
        B <= '0';
        WAIT FOR 50 ns;
        A <= '1';
        B <= '1';
        WAIT FOR 50 ns;

        WAIT;
    END PROCESS;

END;
```

Listing 10: ALU-1Bit VHDL

```
---------------------------------------------------------------------------
--Company      : RIT
--Author       : Brandon Key
--Created      : 02/18/2018
--
--Project Name : Lab 3
--File         : ALU.vhd
--
--Entity       : ALU
--Architecture : struct
--
--Tool Version : VHDL '93
--Description  : ALU
---------------------------------------------------------------------------
library  IEEE;
use IEEE.STD_LOGIC_1164.ALL;

package controlcodes is
    constant AND_Code : std_logic_vector(1 downto 0) := "00";
    constant OR_Code  : std_logic_vector(1 downto 0) := "01";
    constant ADD_Code : std_logic_vector(1 downto 0) := "10";
    constant SUB_Code : std_logic_vector(1 downto 0) := "11";
end controlcodes;


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

```
use work.controlcodes.all;

entity ALU_1Bit is
    port(
        Control : in std_logic_vector(1 downto 0);
        A,B     : in std_logic;
        Y       : out std_logic;
        CB      : out std_logic
        );
end ALU_1Bit;

architecture behav of ALU_1Bit is

begin

    Y_proc: with Control select
        Y <= A xor B when ADD_Code,
             A xor B when SUB_Code,
             A or  B when OR_Code,
             A and B when AND_Code,
             '0'     when others;

    CB_proc: with Control select
        CB <= A and B       when ADD_Code,
              (not A) and B when SUB_Code,
              '0'           when OR_Code,
              '0'           when AND_Code,
              '0'           when others;

end behav;
```

Listing 11: ALU-4Bit-tb VHDL

```
--------------------------------------------------------------------------------
-- Company: RIT
-- Engineer: Brandon Key
--
-- Create Date:   17:51:58 02/28/2018
-- Design Name:
-- Module Name:   /home/ise/DSDII/Lab/Lab3/SourceCode/ALU_4Bit_tb.vhd
-- Project Name:  Lab3
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: ALU_4Bit
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
```

```
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
--use work.globals.all;
--use work.controlcodes.all;

ENTITY ALU_4Bit_tb IS
END ALU_4Bit_tb;

ARCHITECTURE behavior OF ALU_4Bit_tb IS

    constant AND_Code : std_logic_vector(1 downto 0) := "00";
    constant OR_Code  : std_logic_vector(1 downto 0) := "01";
    constant ADD_Code : std_logic_vector(1 downto 0) := "10";
    constant SUB_Code : std_logic_vector(1 downto 0) := "11";

    type testRecordArray is array (natural range <>) of std_logic_vector(2 downto 0)
    ;

    constant n:integer := 16;
    -- "Time" that will elapse between test vectors we submit to the component.
    constant TIME_DELTA : time := 50 ns;


    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT ALU_4Bit
    PORT(
         Control : IN  std_logic_vector(1 downto 0);
         A : IN  std_logic_vector(N-1 downto 0);
         B : IN  std_logic_vector(N-1 downto 0);
         nBitOut : OUT  std_logic_vector(N-1 downto 0);
         CB : OUT std_logic
        );
    END COMPONENT;


   --Inputs
   signal Control : std_logic_vector(1 downto 0) := (others => '0');
   signal A : std_logic_vector(N-1 downto 0) := (others => '0');
   signal B : std_logic_vector(N-1 downto 0) := (others => '0');

    --Outputs
   signal nBitOut : std_logic_vector(N-1 downto 0);
   signal CB : std_logic;
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name


BEGIN

    -- Instantiate the Unit Under Test (UUT)
   uut: ALU_4Bit
   PORT MAP (
          Control => Control,
          A => A,
          B => B,
          nBitOut => nBitOut,
          CB => CB
```

```
      );


  -- Stimulus process
  stim_proc: process
     --create a function to make a vector a strung
     function vec2str(vec : std_logic_vector) return string is
       variable stmp:string(vec'left+1 downto 1);
     begin
       for i in vec'reverse_range loop
         if vec(i) = '1' then
           stmp(i+1) := '1';
         elsif vec(i) = 'U' then
           stmp(i+1) := 'U';
         else
           stmp(i+1) := '0';
         end if;
       end loop;
       return stmp;
     end vec2str;

       procedure check_add(
           constant in1 : in natural;
           constant in2 : in natural;
           constant res_expected : in natural;
           constant CB_expected : in std_logic) is
           variable res : natural;
           begin
           -- Assign values to circuit inputs.
           A <= std_logic_vector(to_unsigned(in1, A'length));
           B <= std_logic_vector(to_unsigned(in2, B'length));
           Control <= ADD_Code;


           wait for TIME_DELTA;
           -- Check output against expected result.
           res := to_integer(unsigned(nBitOut));
           assert ((res = res_expected) and ( CB = CB_expected ))
           report "" & integer'image(in1) & "+" &
                  integer'image(in2) & "=" &
                  integer'image(res_expected) & "!=" &
                  integer'image(res) &
                  "     " &
                  "CB exp: " & std_logic'image(CB_expected) &
                  "Got: " & std_logic'image(CB)
           severity error;
       end procedure check_add;

       procedure check_sub(
           constant in1 : in natural;
           constant in2 : in natural;
           constant res_expected : in natural;
           constant CB_expected : in std_logic) is
           variable res : natural;
           begin
           -- Assign values to circuit inputs.
           A <= std_logic_vector(to_unsigned(in1, A'length));
           B <= std_logic_vector(to_unsigned(in2, B'length));
           Control <= SUB_Code;
```

```
            wait for TIME_DELTA;
            -- Check output against expected result.
            res := to_integer(unsigned(nBitOut));
            assert ((res = res_expected) and ( CB = CB_expected ))
            report "" & integer'image(in1) & "-" &
                   integer'image(in2) & "=" &
                   integer'image(res_expected) & "!=" &
                   integer'image(res) &
                   "     " &
                   "CB exp: " & std_logic'image(CB_expected) &
                   "Got: " & std_logic'image(CB)
            severity error;
        end procedure check_sub;


        procedure check_or(
            constant in1 : in natural;
            constant in2 : in natural;
            constant res_expected : in natural) is
            variable res : natural;
            begin
            -- Assign values to circuit inputs.
            A <= std_logic_vector(to_unsigned(in1, A'length));
            B <= std_logic_vector(to_unsigned(in2, B'length));
            Control <= OR_Code;


            wait for TIME_DELTA;
            -- Check output against expected result.
            res := to_integer(unsigned(nBitOut));
            assert ((res = res_expected) and ( CB = '0' ))
            report "" & integer'image(in1) & "+" &
                   integer'image(in2) & "=" &
                   integer'image(res_expected) & "!=" &
                   integer'image(res) &
                   "     " &
                   "CB: " & std_logic'image(CB)
            severity error;
        end procedure check_or;


        procedure check_and(
            constant in1 : in natural;
            constant in2 : in natural;
            constant res_expected : in natural) is
            variable res : natural;
            begin
            -- Assign values to circuit inputs.
            A <= std_logic_vector(to_unsigned(in1, A'length));
            B <= std_logic_vector(to_unsigned(in2, B'length));
            Control <= AND_Code;


            wait for TIME_DELTA;
            -- Check output against expected result.
            res := to_integer(unsigned(nBitOut));
            report "" & integer'image(in1) & "+" &
```

```
                    integer'image(in2) & "=" &
                    integer'image(res_expected);

            assert ((res = res_expected) and ( CB = '0' ))
            report  "!=" &
                    integer'image(res) &
                    "     " &
                    "CB: " & std_logic'image(CB)
            severity error;
        end procedure check_and;

    begin


      --wait for the outputs to stabilize
      wait for 100 ns;

      --check_add(4,5,9,0);
      --check_add(65535, 2, 1, 1);
      --check_sub(1234, 234, 1000, 0);
      --check_sub(1, 2, 1, 1);

      control <= OR_Code;
      A <= "0111010101110101";
      B <= "1001110100101101";
      wait for 50 ns;

      control <= AND_Code;
      wait for 50 ns;


      -- Test adder
     for x in (0) to (5) loop
         for y in 5432 to 5438 loop
             control <= ADD_Code;
             A <= std_logic_vector(to_unsigned(x, A'length));
             B <= std_logic_vector(to_unsigned(y, B'length));
             wait for 50 ns;

             assert(nBitOut = std_logic_vector(to_unsigned(x+y, A'length) ) )
             report("Bad Add = " & vec2str(nBitOut)
                     & " expected = " & vec2str( std_logic_vector(to_unsigned(x+y, A'length)) )
                     & " A = " & vec2str(A)
                     & " B = " & vec2str(B)
               );
         end loop;
     end loop;

     for x in ((2**N)-3) to ((2**N)-1) loop
         for y in 0 to 3 loop
             control <= ADD_Code;
             A <= std_logic_vector(to_unsigned(x, A'length));
             B <= std_logic_vector(to_unsigned(y, B'length));
             wait for 50 ns;

             assert(nBitOut = std_logic_vector(to_unsigned(x+y, A'length) ) )
             report("Bad Add = " & vec2str(nBitOut)
                     & " expected = " & vec2str( std_logic_vector(to_unsigned(x+y, A'
```

```
    length)) )
                    & " A = " & vec2str(A)
                    & " B = " & vec2str(B)
                );
            end loop;
        end loop;

        -- Test suber
        for x in 0 to 5 loop
            for y in 0 to 5 loop
            control <= SUB_Code;
            A <= std_logic_vector(to_unsigned(x, A'length));
            B <= std_logic_vector(to_unsigned(y, B'length));
            wait for 50 ns;

            assert(nBitOut = std_logic_vector(to_signed(x-y, A'length)) )
            report("Bad Sub = " & vec2str(nBitOut)
                & " expected = " & vec2str( std_logic_vector(to_signed(x-y, A'length)) )
                & " A = " & vec2str(A)
                & " B = " & vec2str(B)
            );

            end loop;
        end loop;

        for x in 12345 to 12350 loop
            for y in 5 to 7 loop
            control <= SUB_Code;
            A <= std_logic_vector(to_unsigned(x, A'length));
            B <= std_logic_vector(to_unsigned(y, B'length));
            wait for 50 ns;

            assert(nBitOut = std_logic_vector(to_signed(x-y, A'length)) )
            report("Bad Sub = " & vec2str(nBitOut)
                & " expected = " & vec2str( std_logic_vector(to_signed(x-y, A'length)) )
                & " A = " & vec2str(A)
                & " B = " & vec2str(B)
            );

            end loop;
        end loop;




      wait;
    end process;

END;
```

Listing 12: ALU-16Bit VHDL

```
-----------------------------------------------------------------------------------
--Company       : RIT
--Author        : Brandon Key
--Created       : 02/18/2018
--
--Project Name  : Lab 3
--File          : ALU_16Bit.vhd
```

```
--
--Entity        : ALU_16Bit
--Architecture  : struct
--
--Tool Version  : VHDL '93
--Description   : ALU_16Bit
-------------------------------------------------------------------------------

library  IEEE;
use IEEE.STD_LOGIC_1164.ALL;

package globals is
    constant N : integer := 16;
end globals;


library  IEEE;
use IEEE.STD_LOGIC_1164.ALL;

package controlcodes is
    constant AND_Code : std_logic_vector(1 downto 0) := "00";
    constant OR_Code  : std_logic_vector(1 downto 0) := "01";
    constant ADD_Code : std_logic_vector(1 downto 0) := "10";
    constant SUB_Code : std_logic_vector(1 downto 0) := "11";
end controlcodes;


library  IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
use work.controlcodes.all;
use work.globals.all;

entity  ALU_16Bit  is
    port(
        Control : in std_logic_vector(1 downto 0);
        A,B     : in std_logic_vector(N-1 downto 0);
        nBitOut : out std_logic_vector(N-1 downto 0);
        CB      : out std_logic
        );
end  ALU_16Bit;

architecture  struct of  ALU_16Bit  is

    --constant N : integer := 4;

    signal ADD_SUB_Out : std_logic_vector(N-1 downto 0);
    signal OR_Out      : std_logic_vector(N-1 downto 0);
    signal AND_Out     : std_logic_vector(N-1 downto 0);

    signal ADD_SUB_SEL : std_logic;


begin

    nBitAdderSubtractor_16Bit : entity work.nBitAdderSubtractor_16Bit
        generic map (N => N)
        port map ( A => A, B => B, SEL => ADD_SUB_SEL, Y => ADD_SUB_Out, CB => CB);

```

```
    nBitOR_16Bit : entity work.nBitOR_16Bit
        generic map (N => N)
        port map ( A => A, B => B, Y => OR_Out);


    nBitAND_16Bit : entity work.nBitAND_16Bit
        generic map (N => N)
        port map ( A => A, B => B, Y => AND_Out);

    Controller_16Bit : entity work.Controller_16Bit
        generic map (N => N)
        port map(
        Control     => Control,
        ADD_SUB_In  => ADD_SUB_Out,
        OR_In       => OR_Out,
        AND_In      => AND_Out,
        ADD_SUB_SEL => ADD_SUB_SEL,
        nBitOut     => nBitOut
        );

end struct;
```

Listing 13: nBitOR-16Bit VHDL

```
---------------------------------------------------------------------------
--Company       : RIT
--Author        : Brandon Key
--Created       : 1/22/2018
--
--Project Name : Lab 1
--File          : nBitOR_16Bit.vhd
--
--Entity        : nBitOR_16Bit
--Architecture : Dataflow
--
--Tool Version : VHDL '93
--Description   : Entity and structural description of an OR gate
---------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity nBitOR_16Bit is
    generic (n : integer := 16);
    port(A,B : in std_logic_vector(n-1 downto 0);
        Y : out std_logic_vector(n-1 downto 0)
        );
end nBitOR_16Bit;

architecture Dataflow of nBitOR_16Bit is
    begin
    Y <= A or B;-- bitwise or
end Dataflow;
```

Listing 14: nBitAdderSubtractor-16Bit VHDL

```
---------------------------------------------------------------------------
--Company       : RIT
```

```vhdl
--Author        : Brandon Key
--Created       : 02/18/2018
--
--Project Name : Lab 3
--File          : nBitAdderSubtractor_16Bit.vhd
--
--Entity        : nBitAdderSubtractor_16Bit
--Architecture : struct
--
--Tool Version : VHDL '93
--Description   : Entity and structural description of an adder subtractor
--              : SEL = 0 : A+B = Y
--              : SEL = 1 : A-B = Y
--------------------------------------------------------------------------------

library  IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

entity  nBitAdderSubtractor_16Bit  is
    generic (n : integer  := 16);
    port(
        A,B : in  std_logic_vector(n-1  downto  0);
        SEL : in  std_logic;
        Y   : out std_logic_vector(n-1  downto  0);
        CB  : out std_logic
        );
end  nBitAdderSubtractor_16Bit;

architecture  struct of  nBitAdderSubtractor_16Bit  is

    component full_adder is
        port(A,B,Cin : in  std_logic;
            Sum,Cout : out  std_logic
        );
     end component full_adder;

    --Create an array to hold all of the carries
    type carry_array  is  array (n-1  downto  0) of std_logic;
    signal c_array : carry_array;

    signal B_XOR_SEL : std_logic_vector( (n-1) downto 0);

begin

    --Generate the xor statements to be mapped to the full adders
    XORator : for i in 0 to n-1 generate
        B_XOR_SEL(i) <= B(i) xor SEL;
    end generate XORator;

    generate_adders : for i in 0 to n-1  generate
        i_first: if i = 0 generate
            --The first adder gets SEL as the Cin
            adder : full_adder port map(
                A => A(i),
                B => B_XOR_SEL(i),
                Cin => SEL,
                Sum => Y(i),
                Cout => c_array(i)
```

```
            );
        end generate i_first;

        i_last : if i = (n-1) generate
            --The last adder doesn't have a carry out
            adder : full_adder port map(
                A => A(i),
                B => B_XOR_SEL(i),
                Cin => c_array(i-1),
                Sum => Y(i),
                Cout =>c_array(i)
            );
        end generate i_last;

        --Middle adders
        i_mid : if (i /= 0) and (i /= (n-1)) generate
            adder : full_adder port map(
                A => A(i),
                B => B_XOR_SEL(i),
                Cin => c_array(i-1),
                Sum => Y(i),
                Cout => c_array(i)
            );
        end generate i_mid;

    end generate generate_adders;

    CB <= c_array(n-1) xor SEL;

end struct;
```

Listing 15: nBitAND-16Bit VHDL

```
---------------------------------------------------------------------------
--Company       : RIT
--Author        : Brandon Key
--Created       : 1/22/2018
--
--Project Name : Lab 1
--File          : nBitAND_16Bit.vhd
--
--Entity        : nBitAND_16Bit
--Architecture : Dataflow
--
--Tool Version : VHDL '93
--Description   : Entity and structural description of an AND gate
---------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity nBitAND_16Bit is
    generic (n : integer := 16);
    port(A,B : in std_logic_vector(n-1 downto 0);
        Y : out std_logic_vector(n-1 downto 0)
        );
end nBitAND_16Bit;

architecture Dataflow of nBitAND_16Bit is
```

```
    begin
    Y  <= A AND B;-- bitwise or
end  Dataflow;
```

5.2 SPICE

Listing 16: 1Bit ALU SPICE

```
* Example circuit file for simulating PEX

.OPTION DOTNODE
.HIER /

.INCLUDE "/home/bxk5113/Pyxis_SPT_HEP/ic_projects/Pyxis_SPT/digicdesign/ALU_1Bit/ALU_1Bit.cal/ALU_1Bit.pex.netlist"

.LIB /home/bxk5113/Pyxis_SPT_HEP/ic_reflibs/tech_libs/generic13/models/lib.eldo TT

* - Instantiate your parasitic netlist and add the load capacitor
** FORMAT :
* XLAYOUT [all inputs as listed by the ".subckt" line in the included netlist, in the order that they appear there] [name of the subcircuit as listed in the included netlist]
XLAYOUT CB Y A B CONTROL[1] CONTROL[0] ALU_1Bit
C1 Y 0 120f
C2 CB 0 120f


* - Analysis Setup - DC sweep
* FORMAT : .DC [name] [low] [high] [step]
*.DC VFORCE_A 0 1.2 0.01

* - Analysis Setup - Trans
* FORMAT : .TRAN [start time] [end time] [time step]
.TRAN 0 160n 0.001n

* --- Forces
* FORMAT -- PULSE : [name] [port] [reference (0 means ground)] PULSE [low] [high] [delay] [fall time] [rise time] [pulse width] [period]
*
* FORMAT -- DC    : [name] [port] [reference (0 means ground)] DC [voltage]
*

VFORCE_A A 0 PULSE (0 1.08 40n 0.1n 0.1n 40n 80n)
VFORCE_B B 0 PULSE (0 1.08 80n 0.1n 0.1n 80n 160n)
VFORCE_C1 CONTROL[1] 0 DC 1.08
VFORCE_C0 CONTROL[0] 0 DC 1.08

VFORCE_VDD VDD 0 DC 1.08
VFORCE_VSS VSS 0 DC 0

* --- Waveform Outputs
.PLOT TRAN V(A)
.PLOT TRAN V(B)
.PLOT TRAN V(CONTROL[1])
.PLOT TRAN V(CONTROL[0])
.PLOT TRAN V(Y)
```

```
.PLOT TRAN V(CB)

* --- Params
.TEMP 125
```

Listing 17: 6Bit ALU SPICE

```
* Example circuit file for simulating PEX

.OPTION DOTNODE
.HIER /

.INCLUDE "/home/bxk5113/Pyxis_SPT_HEP/ic_projects/Pyxis_SPT/digicdesign/ALU_16Bit/ALU_16Bit.cal/ALU_16Bit.pex.netlist"

.LIB /home/bxk5113/Pyxis_SPT_HEP/ic_reflibs/tech_libs/generic13/models/lib.eldo TT

* - Instantiate your parasitic netlist and add the load capacitor
** FORMAT :
* XLAYOUT [all inputs as listed by the ".subckt" line in the included netlist, in the order that they appear there] [name of the subcircuit as listed in the included netlist]
XLAYOUT CB NBITOUT[15] NBITOUT[14] NBITOUT[13] NBITOUT[12] NBITOUT[11] NBITOUT[10] NBITOUT[9] NBITOUT[8] NBITOUT[7] NBITOUT[6] NBITOUT[5] NBITOUT[4] NBITOUT[3] NBITOUT[2] NBITOUT[1] NBITOUT[0] A[15] A[14] A[13] A[12] A[11] A[10] A[9] A[8] A[7] A[6] A[5] A[4] A[3] A[2] A[1] A[0] B[15] B[14] B[13] B[12] B[11] B[10] B[9] B[8] B[7] B[6] B[5] B[4] B[3] B[2] B[1] B[0] CONTROL[1] CONTROL[0] ALU_16Bit

* Output Capacitance
C_CB CB 0 120f
C_NBITOUT[15] NBITOUT[15] 0 120f
C_NBITOUT[14] NBITOUT[14] 0 120f
C_NBITOUT[13] NBITOUT[13] 0 120f
C_NBITOUT[12] NBITOUT[12] 0 120f
C_NBITOUT[11] NBITOUT[11] 0 120f
C_NBITOUT[10] NBITOUT[10] 0 120f
C_NBITOUT[9] NBITOUT[9] 0 120f
C_NBITOUT[8] NBITOUT[8] 0 120f
C_NBITOUT[7] NBITOUT[7] 0 120f
C_NBITOUT[6] NBITOUT[6] 0 120f
C_NBITOUT[5] NBITOUT[5] 0 120f
C_NBITOUT[4] NBITOUT[4] 0 120f
C_NBITOUT[3] NBITOUT[3] 0 120f
C_NBITOUT[2] NBITOUT[2] 0 120f
C_NBITOUT[1] NBITOUT[1] 0 120f
C_NBITOUT[0] NBITOUT[0] 0 120f



* - Analysis Setup - DC sweep
* FORMAT : .DC [name] [low] [high] [step]
*.DC VFORCE__A 0 1.2 0.01

* - Analysis Setup - Trans
* FORMAT : .TRAN [start time] [end time] [time step]
.TRAN 0 400n 0.001n

* --- Forces
* FORMAT -- PULSE : [name] [port] [reference (0 means ground)] PULSE [low] [high] [delay] [fall time] [rise time] [pulse width] [period]
```

```
*
* FORMAT -- DC     : [name] [port] [reference (0 means ground)] DC [voltage]
*

VFORCE_C1 CONTROL[1] 0 PULSE (0 1.08 40n 0.1n 0.1n 40n 80n)
VFORCE_C0 CONTROL[0] 0 PULSE (0 1.08 20n 0.1n 0.1n 20n 40n)

VFORCE_VDD VDD 0 DC 1.08
VFORCE_VSS VSS 0 DC 0

VFORCE_A[0] A[0] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011001 R
VFORCE_A[1] A[1] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011000 R
VFORCE_A[2] A[2] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011001 R
VFORCE_A[3] A[3] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011000 R
VFORCE_A[4] A[4] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011001 R
VFORCE_A[5] A[5] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011000 R
VFORCE_A[6] A[6] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011001 R
VFORCE_A[7] A[7] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011001 R
VFORCE_A[8] A[8] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011001 R
VFORCE_A[9] A[9] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011001 R
VFORCE_A[10] A[10] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011000 R
VFORCE_A[11] A[11] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011000 R
VFORCE_A[12] A[12] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011001 R
VFORCE_A[13] A[13] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011001 R
VFORCE_A[14] A[14] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011000 R
VFORCE_A[15] A[15] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011101 R
VFORCE_B[0] B[0] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010101 R
VFORCE_B[1] B[1] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010100 R
VFORCE_B[2] B[2] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010100 R
VFORCE_B[3] B[3] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010100 R
VFORCE_B[4] B[4] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010101 R
VFORCE_B[5] B[5] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010100 R
VFORCE_B[6] B[6] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010100 R
VFORCE_B[7] B[7] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010101 R
VFORCE_B[8] B[8] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010101 R
VFORCE_B[9] B[9] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010101 R
VFORCE_B[10] B[10] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010101 R
VFORCE_B[11] B[11] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010100 R
VFORCE_B[12] B[12] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010101 R
VFORCE_B[13] B[13] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010101 R
VFORCE_B[14] B[14] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 010101 R
VFORCE_B[15] B[15] 0 PBIT 0 1.08 0 0 0.01n 0 0.01n 80n 011111 R





* --- Waveform Outputs
.PLOT TRAN V(CB)
.PLOT TRAN V(NBITOUT[15])
.PLOT TRAN V(NBITOUT[14])
.PLOT TRAN V(NBITOUT[13])
.PLOT TRAN V(NBITOUT[12])
.PLOT TRAN V(NBITOUT[11])
.PLOT TRAN V(NBITOUT[10])
.PLOT TRAN V(NBITOUT[9])
.PLOT TRAN V(NBITOUT[8])
.PLOT TRAN V(NBITOUT[7])
.PLOT TRAN V(NBITOUT[6])
```

```
.PLOT TRAN V(NBITOUT[5])
.PLOT TRAN V(NBITOUT[4])
.PLOT TRAN V(NBITOUT[3])
.PLOT TRAN V(NBITOUT[2])
.PLOT TRAN V(NBITOUT[1])
.PLOT TRAN V(NBITOUT[0])
.PLOT TRAN V(A[15])
.PLOT TRAN V(A[14])
.PLOT TRAN V(A[13])
.PLOT TRAN V(A[12])
.PLOT TRAN V(A[11])
.PLOT TRAN V(A[10])
.PLOT TRAN V(A[9])
.PLOT TRAN V(A[8])
.PLOT TRAN V(A[7])
.PLOT TRAN V(A[6])
.PLOT TRAN V(A[5])
.PLOT TRAN V(A[4])
.PLOT TRAN V(A[3])
.PLOT TRAN V(A[2])
.PLOT TRAN V(A[1])
.PLOT TRAN V(A[0])
.PLOT TRAN V(B[15])
.PLOT TRAN V(B[14])
.PLOT TRAN V(B[13])
.PLOT TRAN V(B[12])
.PLOT TRAN V(B[11])
.PLOT TRAN V(B[10])
.PLOT TRAN V(B[9])
.PLOT TRAN V(B[8])
.PLOT TRAN V(B[7])
.PLOT TRAN V(B[6])
.PLOT TRAN V(B[5])
.PLOT TRAN V(B[4])
.PLOT TRAN V(B[3])
.PLOT TRAN V(B[2])
.PLOT TRAN V(B[1])
.PLOT TRAN V(B[0])
.PLOT TRAN V(CONTROL[1])
.PLOT TRAN V(CONTROL[0])


* --- Params
.TEMP 125
```