

**CMPE-630 Digital Integrated Circuit Design**  
**Final Project**  
**Multiply and Accumulate (MAC) Datapath Unit Design**

Brandon Key  
Chris Guarini  
Performed: 9 Dec 2019  
Submitted: 9 Dec 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: \_\_\_\_\_

# Contents

<b>1</b>	<b>Abstract</b>	<b>4</b>
<b>2</b>	<b>Design Methodology and Theory</b>	<b>4</b>
2.1	User Operation . . . . .	4
2.2	Adder . . . . .	5
2.3	Multiplier . . . . .	8
2.4	16-Bit Register . . . . .	10
2.5	32-Bit Register . . . . .	11
2.6	MAC . . . . .	12
2.7	Mux . . . . .	13
2.8	LFSR . . . . .	15
2.9	MISR . . . . .	15
2.10	BIST . . . . .	15
2.11	Schematic . . . . .	15
<b>3</b>	<b>Results and Analysis</b>	<b>33</b>
3.1	Full Adder . . . . .	34
3.2	Multiplier . . . . .	35
3.3	16-Bit Register . . . . .	36
3.4	32-Bit Register . . . . .	37
3.5	Mux . . . . .	38
3.6	MAC with BIST Layout . . . . .	39
3.7	Power . . . . .	43
<b>4</b>	<b>Conclusion</b>	<b>43</b>
<b>5</b>	<b>Appendix</b>	<b>43</b>

---

5.1	VHDL . . . . .	43
5.2	Leonardo Scripts . . . . .	93
5.3	Layout Versus Schematic Results . . . . .	96
5.4	SPICE . . . . .	102
<b>6</b>	<b>References</b>	<b>108</b>

# 1 Abstract

## 2 Design Methodology and Theory

A cornerstone of IC design is the ability to create large, complex designs from smaller more manageable parts. The project outlined in this exercise calls for the design, testing and layout of a multiply and accumulate (MAC) unit, which takes two 16-bit inputs, multiplies them together, adds them to the value stored in a register, and then stores that output back into the register. The final component should contain a built in self test (BIST) that verifies the functionality of the MAC.

The MAC is composed of a carry-save multiplier, ripple carry full-adder, and parallel register. The BIST is implemented through the use of an LFSR for the inputs, an MISR for the output, and a test controller which controls the timing and sets the test passed and test complete outputs. A full diagram of the MAC with BIST can be seen below in Figure 1.

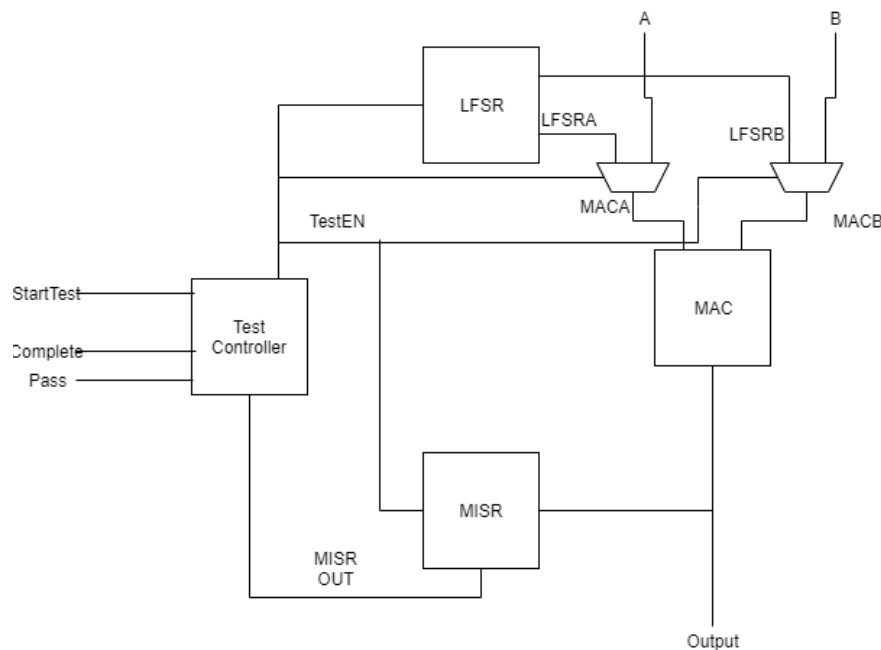


Figure 1: High Level Block Diagram of the MAC with BIST.

### 2.1 User Operation

//TODO Chris, talk about how to use this and go into test mode

## 2.2 Adder

The adder used in this design was 32 bits wide to accept input from the multiplier and the accumulation register.

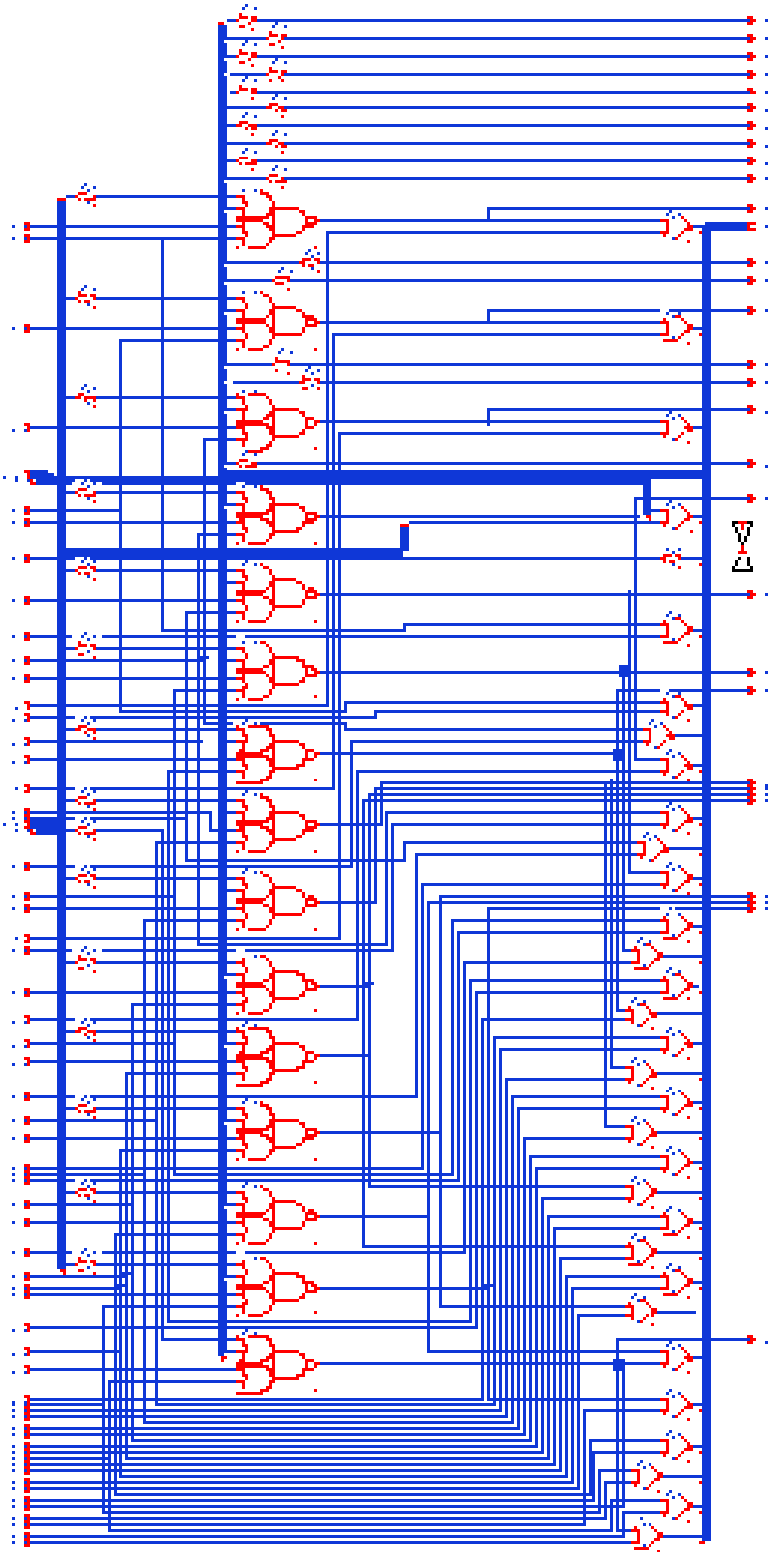


Figure 2: nBitAdder Schematic Page 1

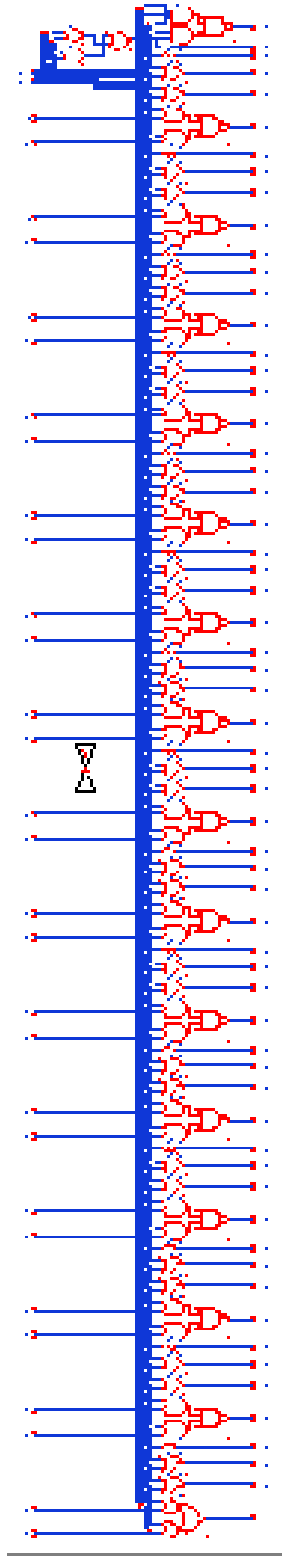


Figure 3: nBitAdder Schematic Page 2

## 2.3 Multiplier

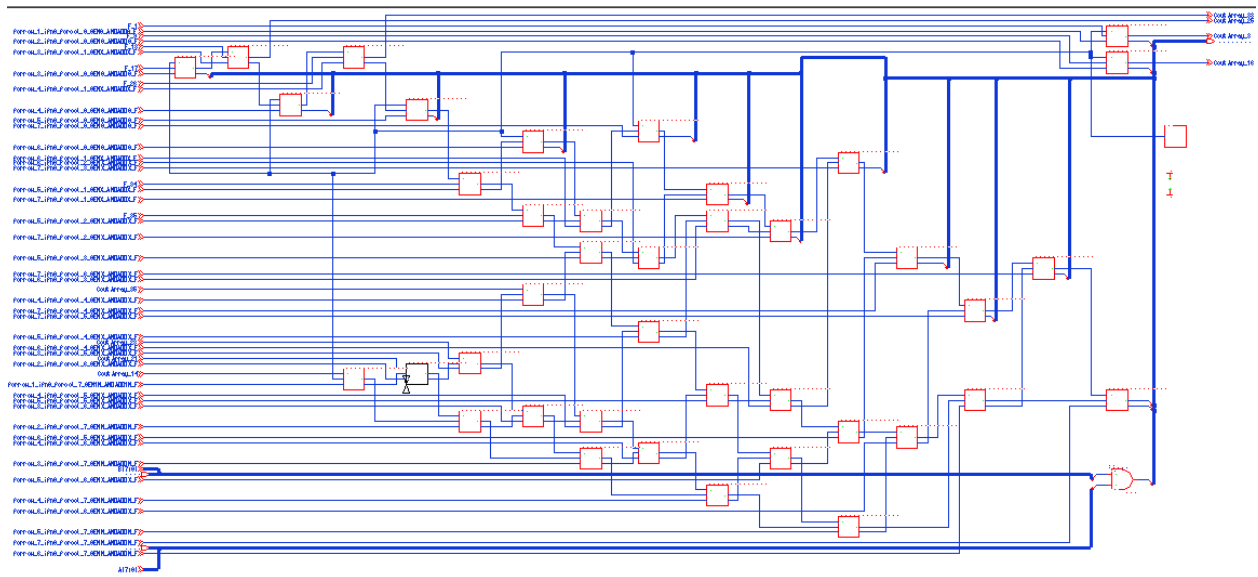


Figure 4: Multiplier Schematic Page 1



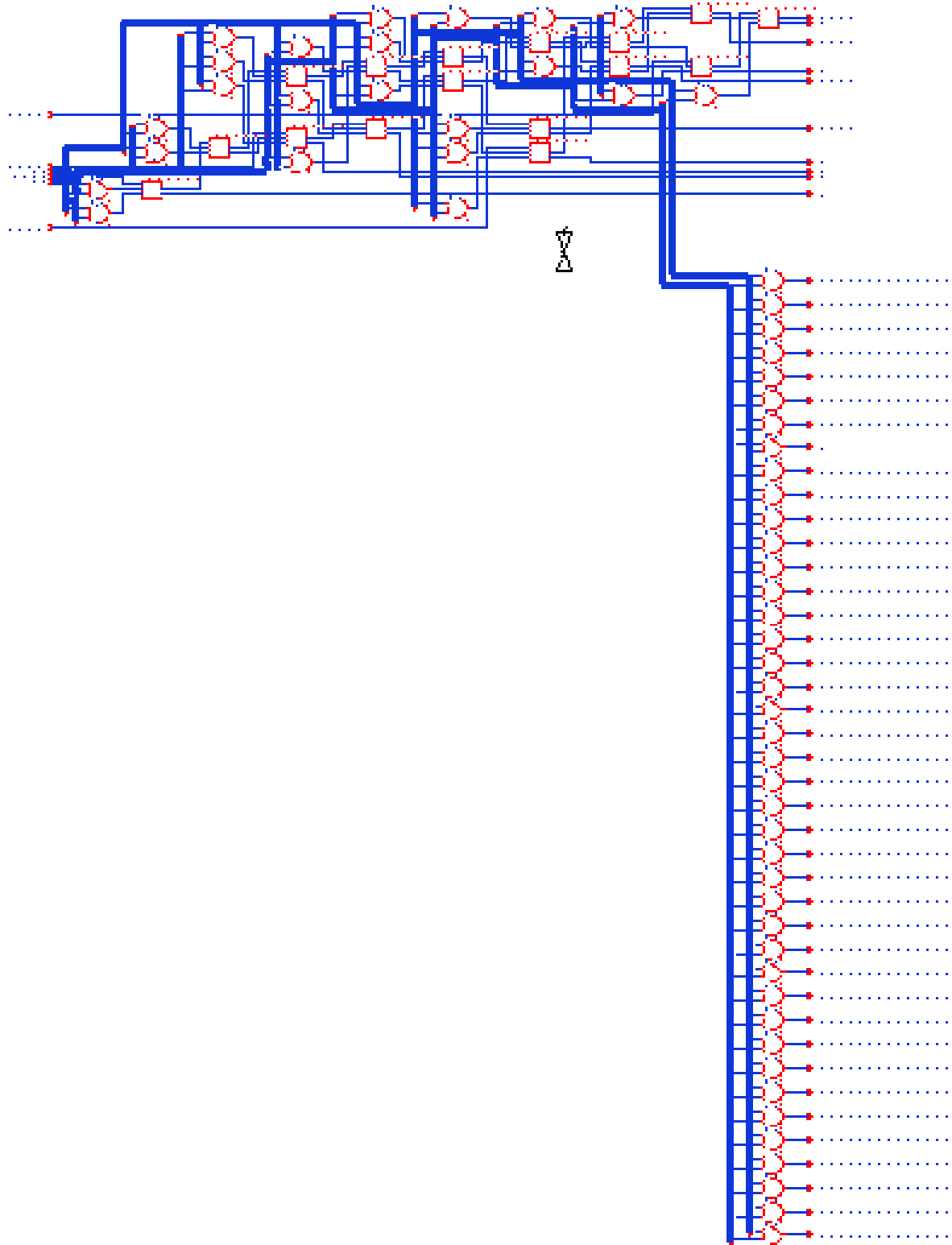


Figure 5: Multiplier Schematic Page 2

## 2.4 16-Bit Register

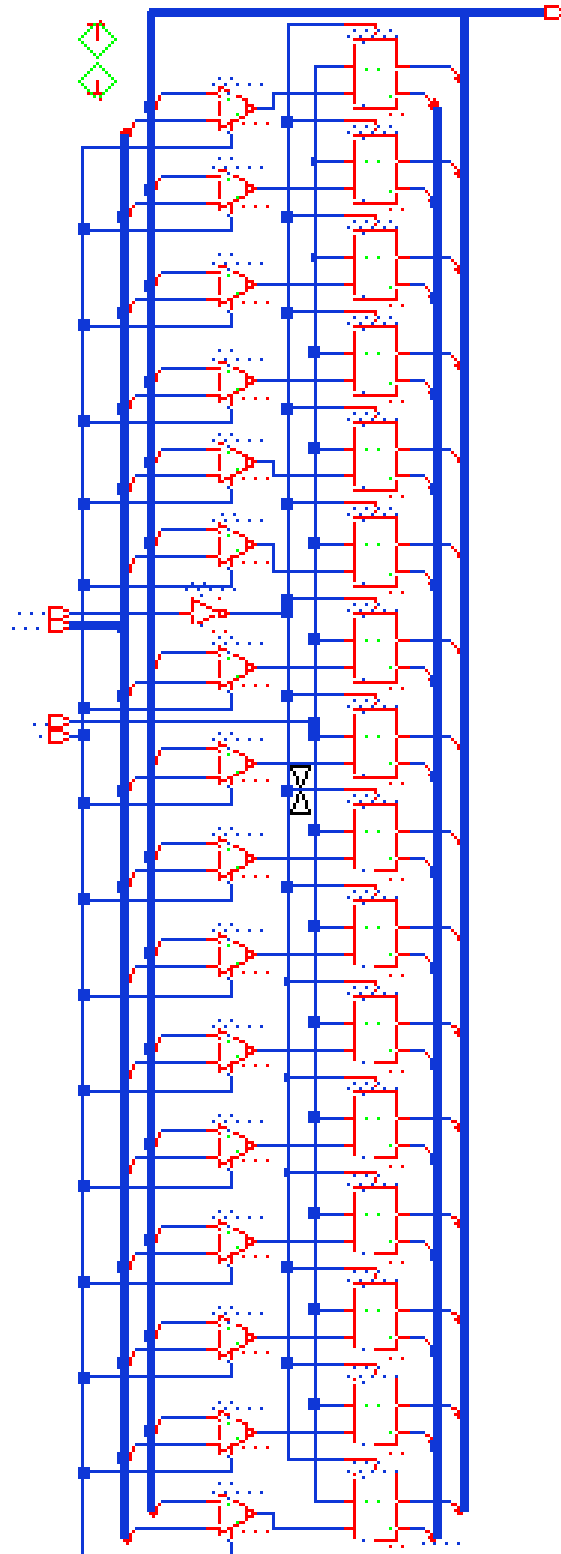


Figure 6: 16 Bit Register Schematic

## 2.5 32-Bit Register

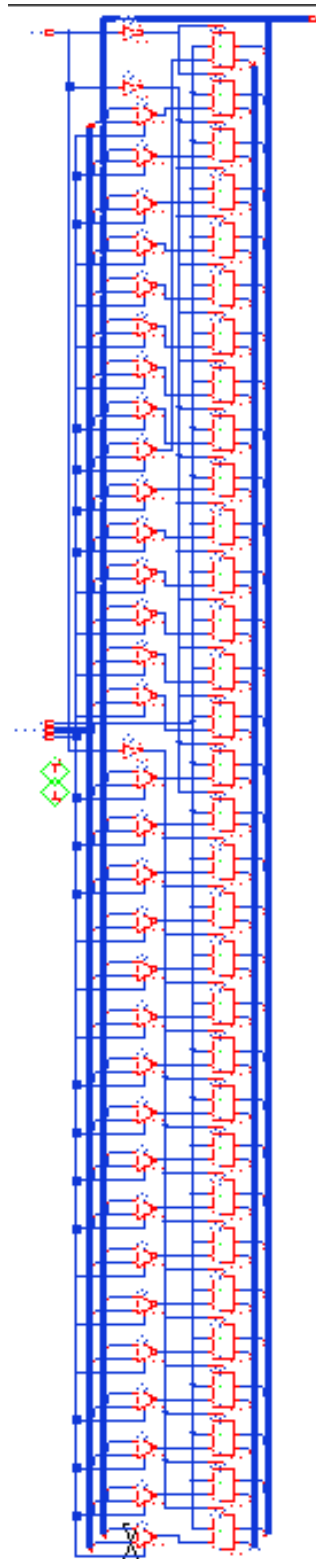


Figure 7: 32 Bit Register Schematic

## 2.6 MAC

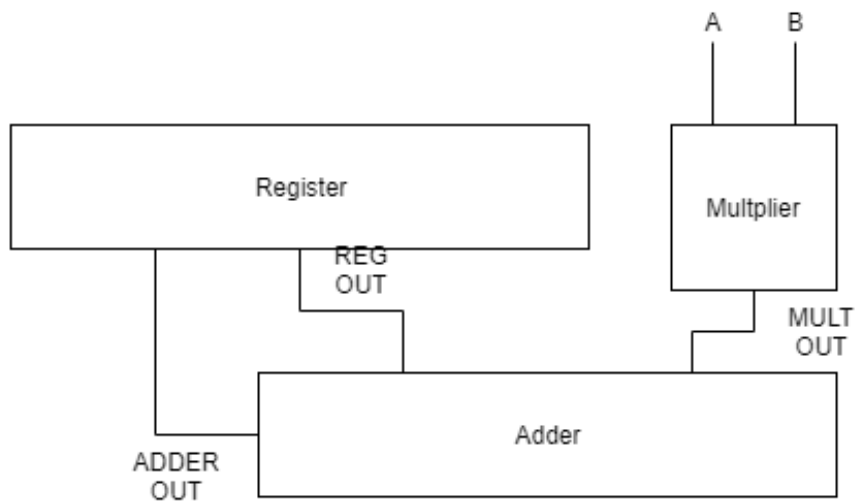


Figure 8: MAC block

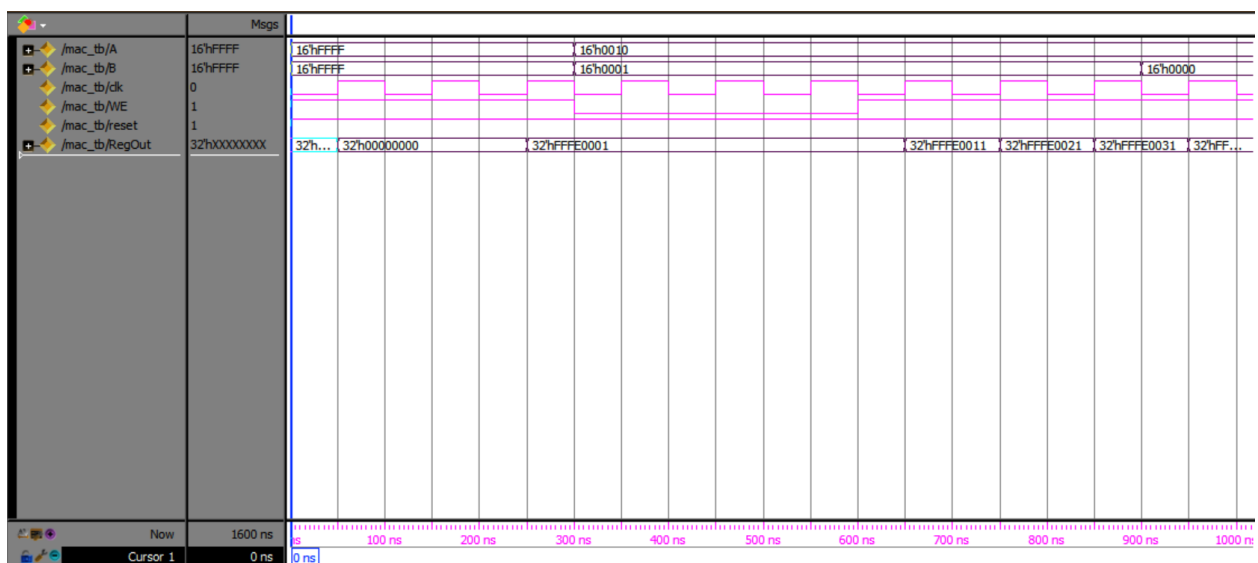


Figure 9: MAC 16bit Test Bench

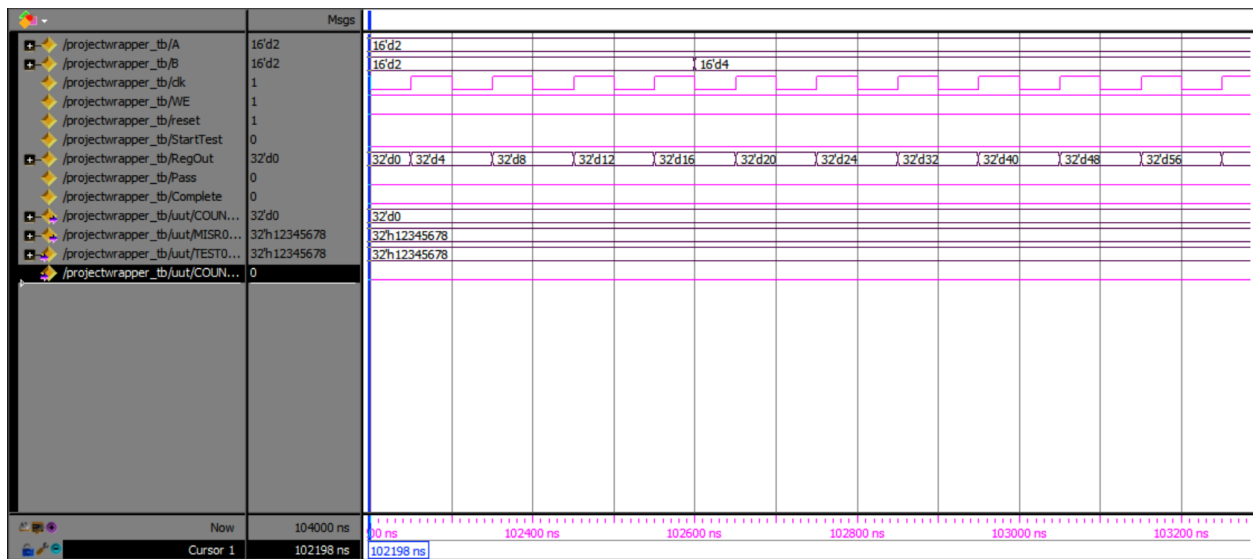


Figure 10: MAC Test Bench

## 2.7 Mux

The multiplexer was used to change the input from the user input to the

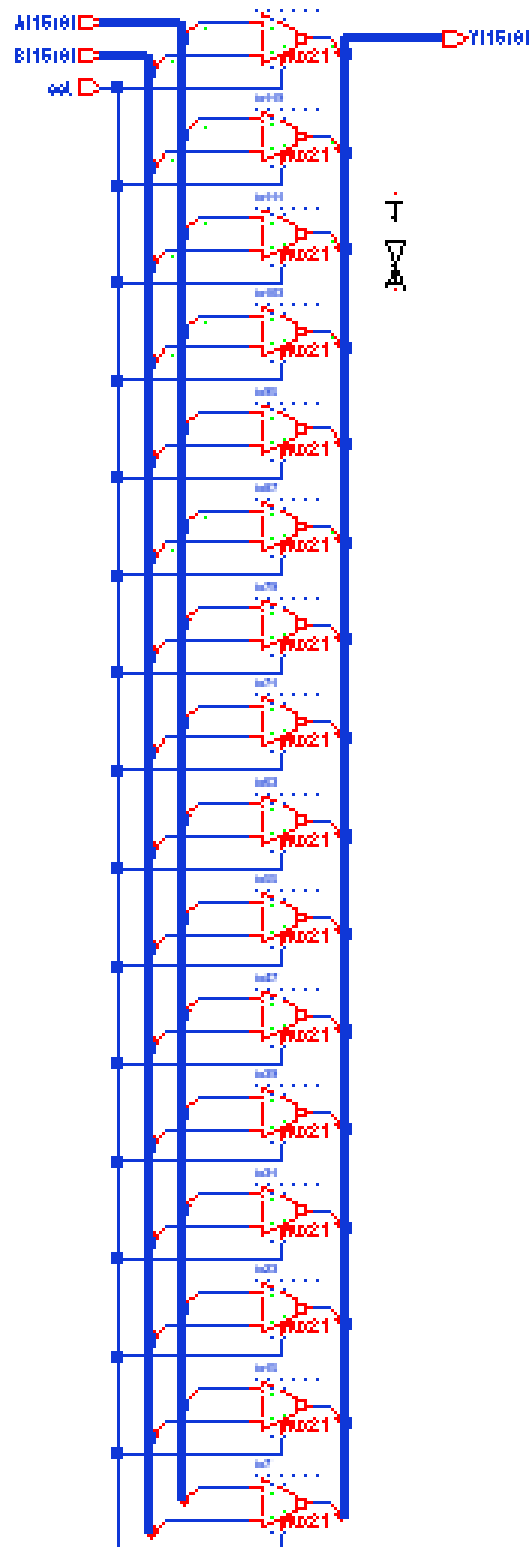


Figure 11: nBitMux 2to1 Schematic

## 2.8 LFSR

## 2.9 MISR

## 2.10 BIST

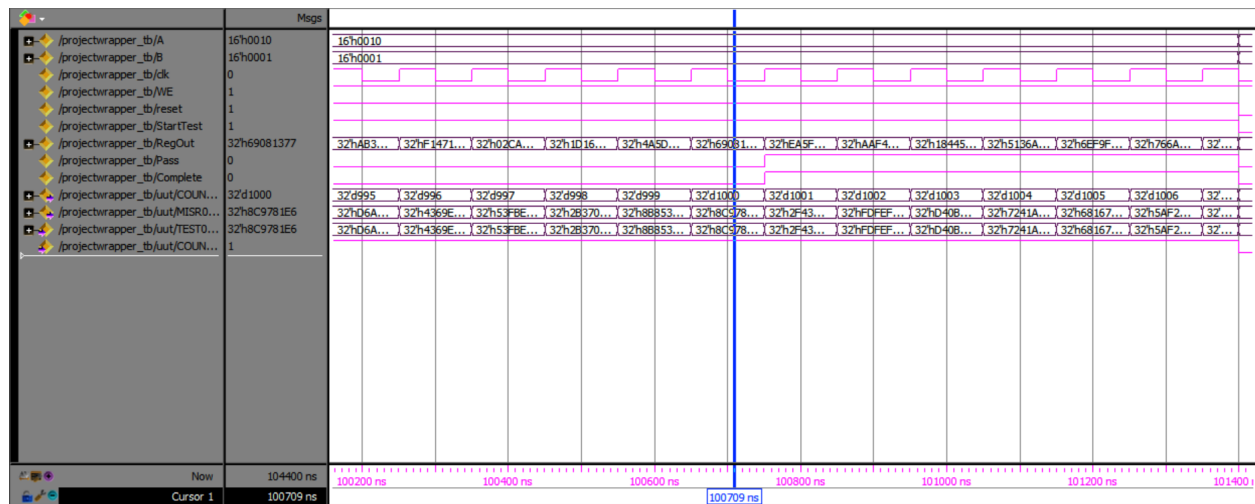


Figure 12: BIST Test Bench

## 2.11 Schematic

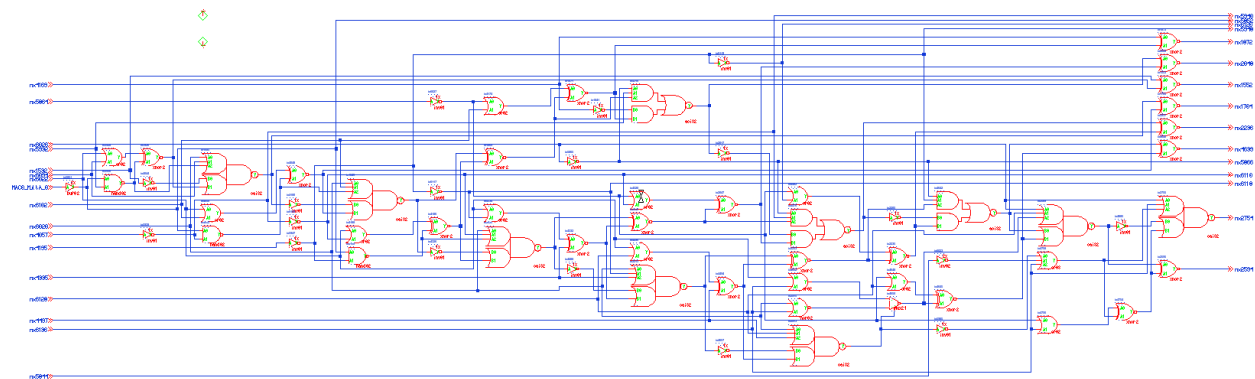
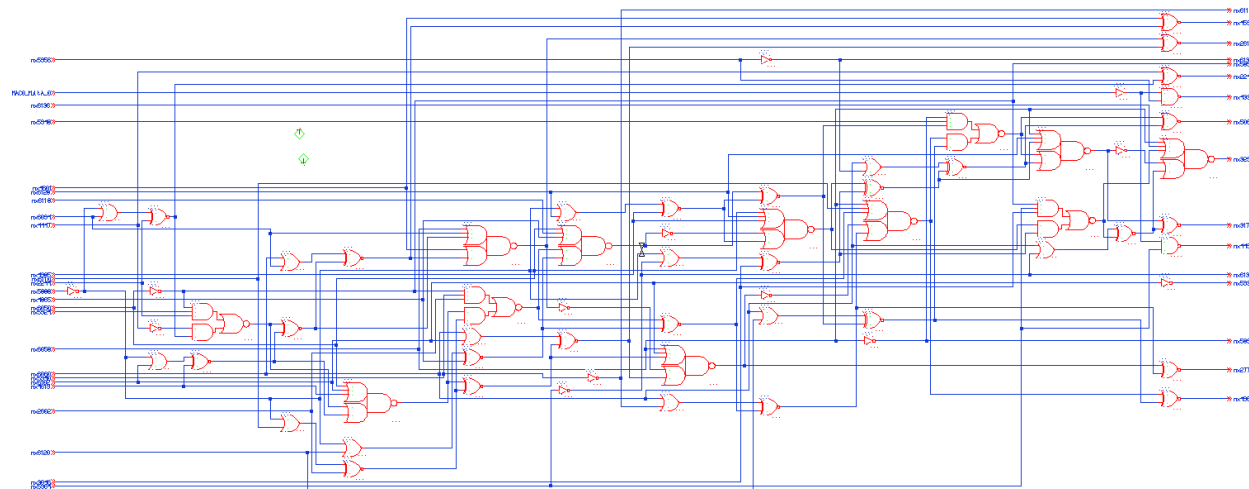
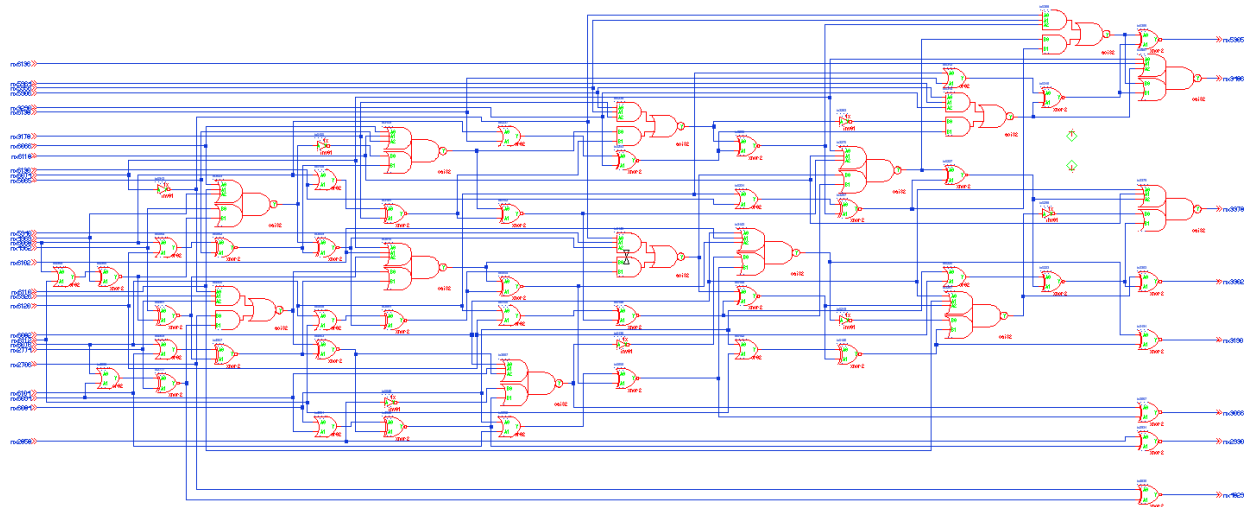


Figure 13: Full Schematic Page 1









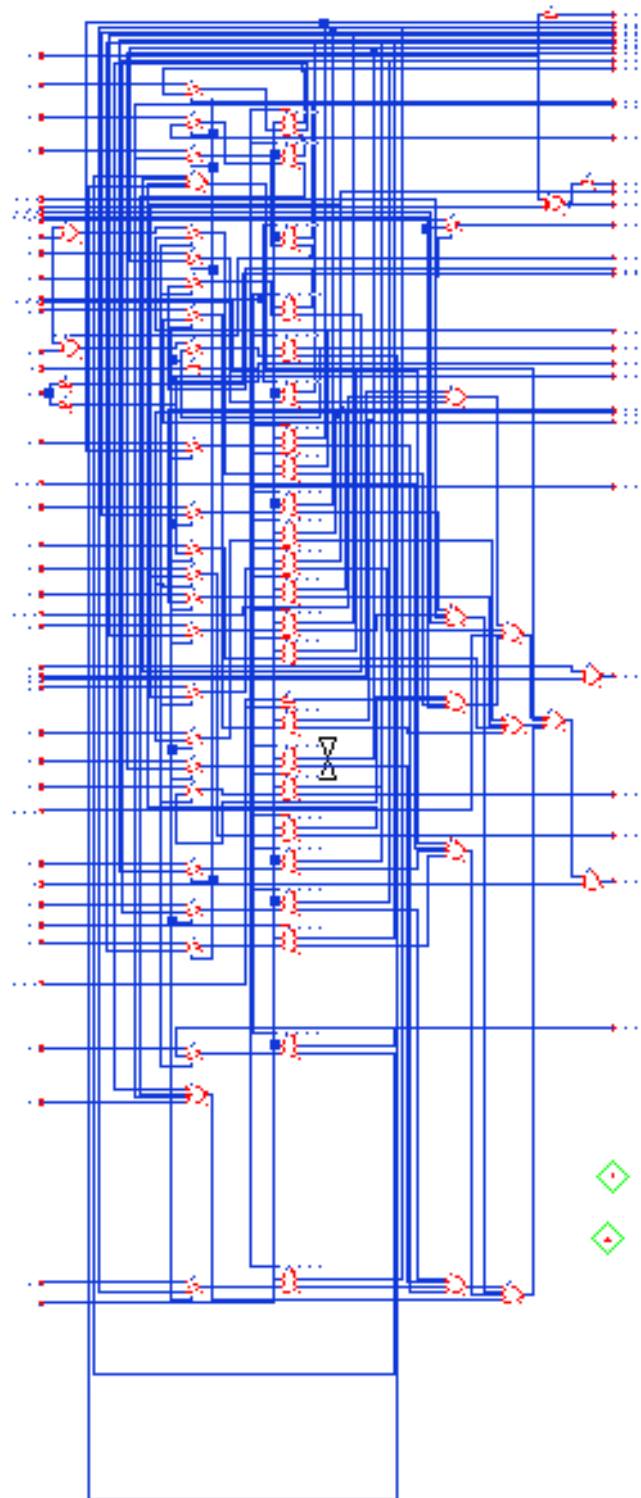


Figure 20: Full Schematic Page 8

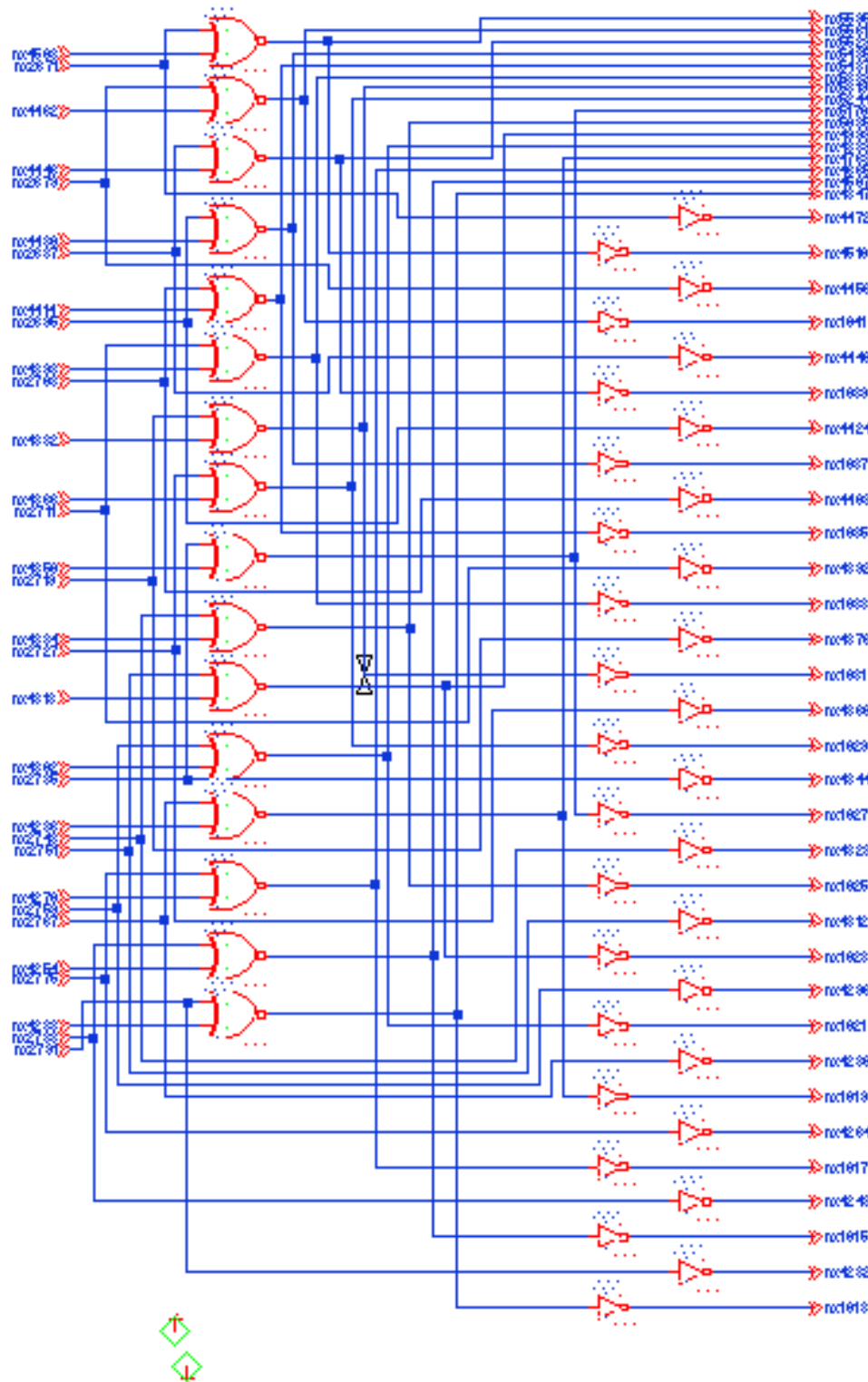


Figure 21: Full Schematic Page 9

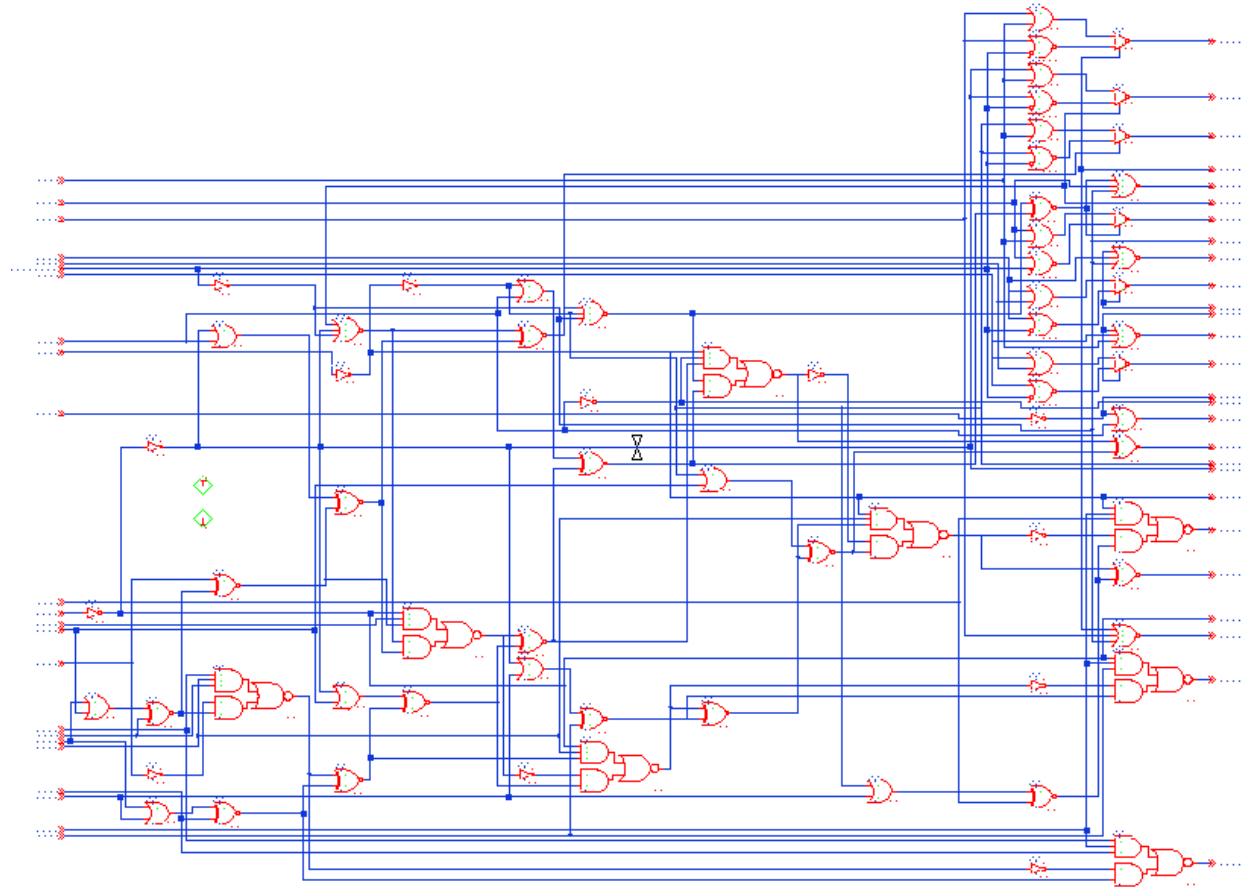


Figure 22: Full Schematic Page 10

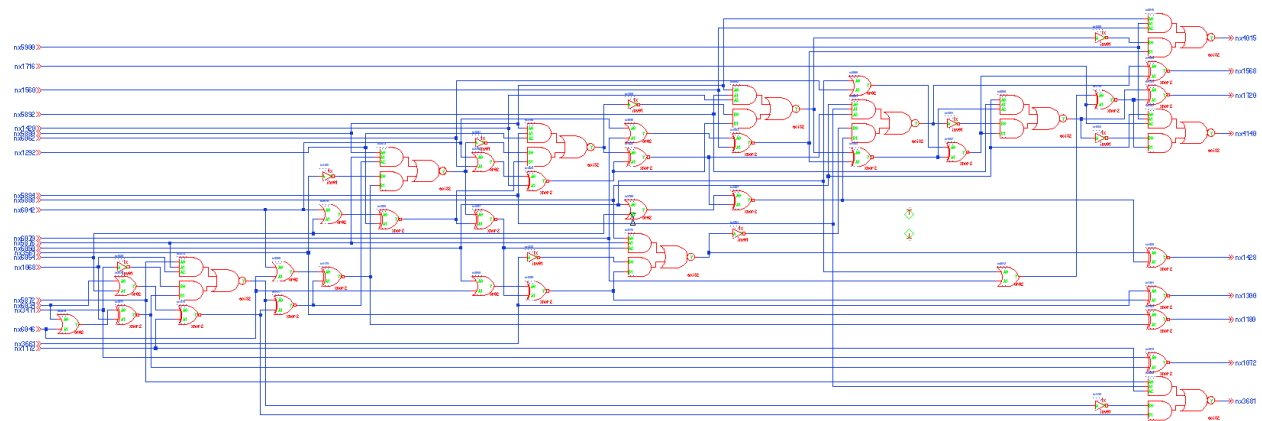
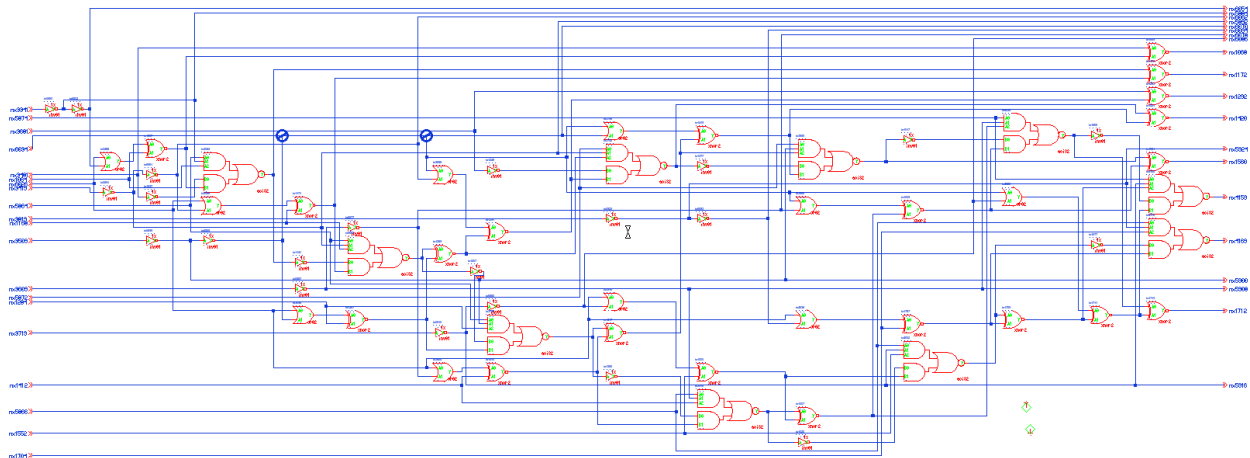
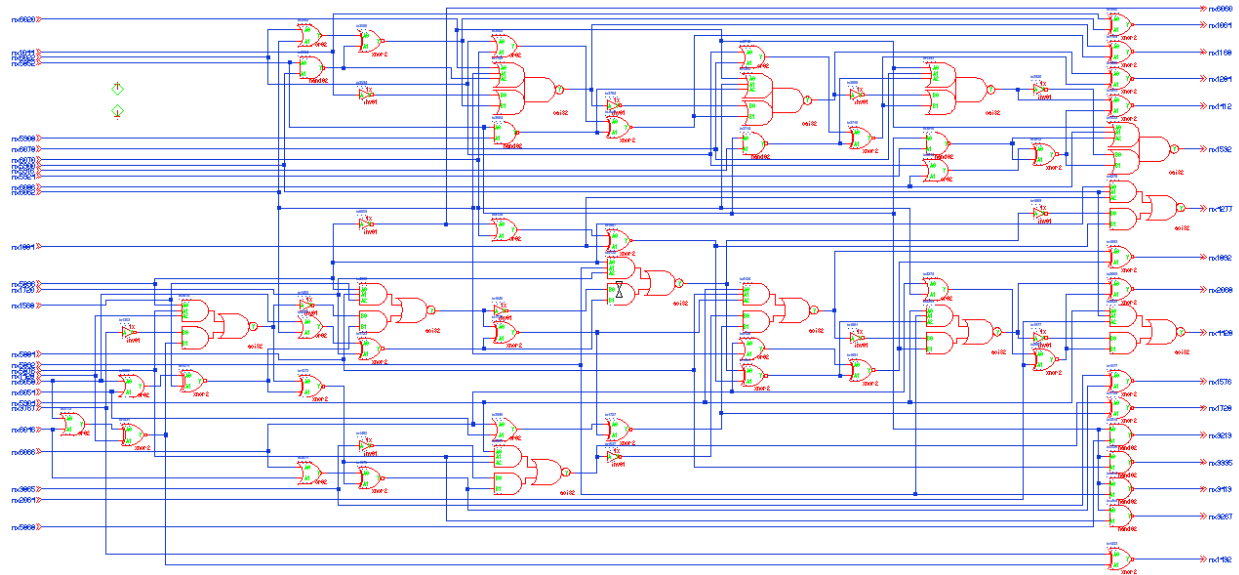


Figure 23: Full Schematic Page 11



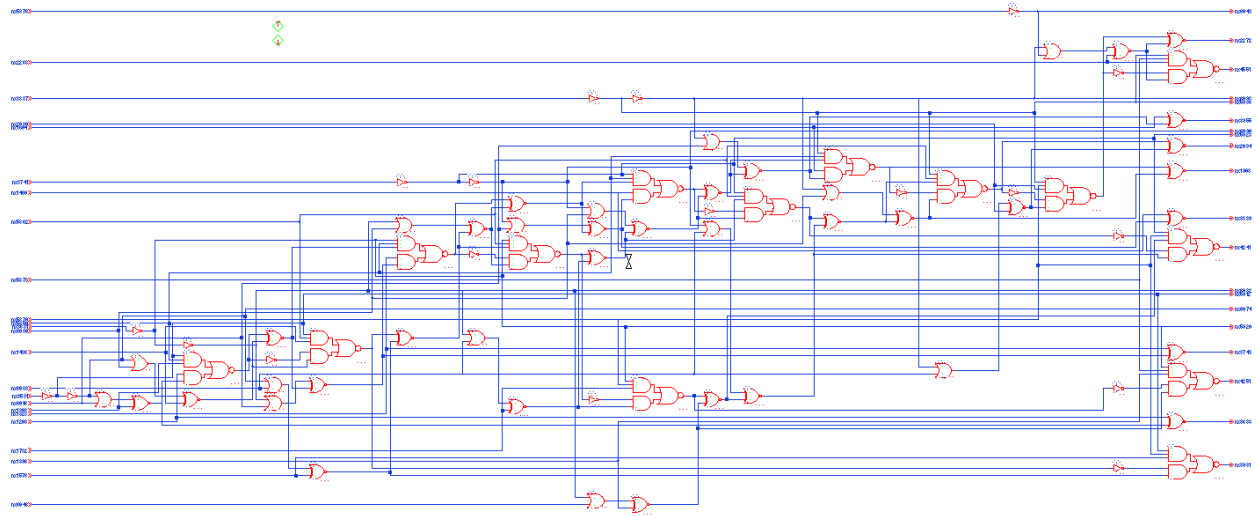


Figure 26: Full Schematic Page 14

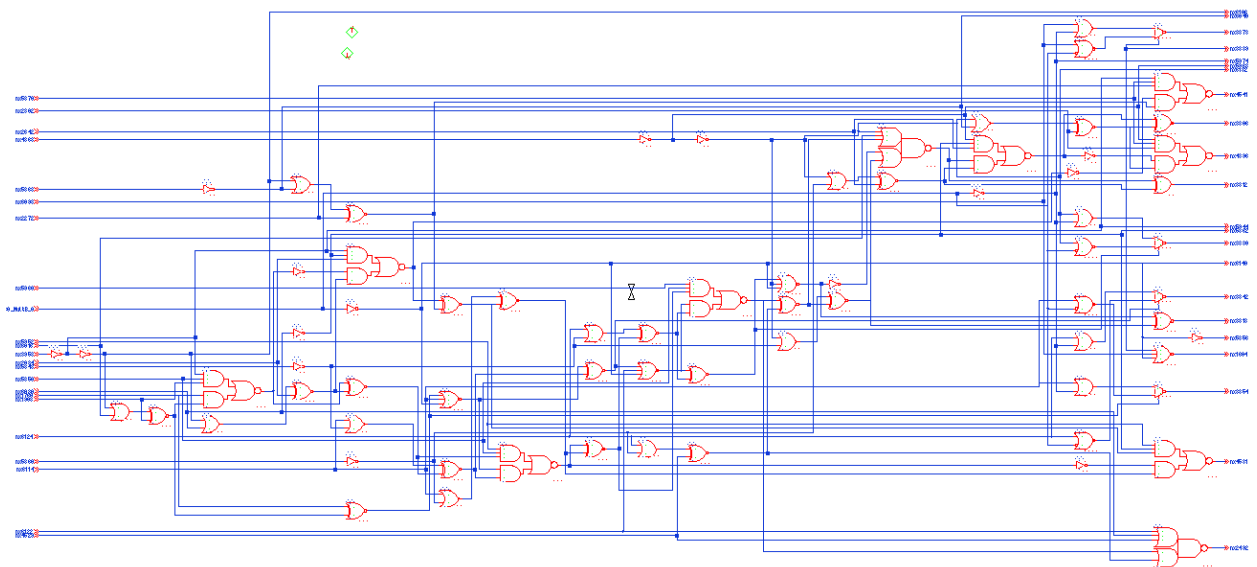


Figure 27: Full Schematic Page 15

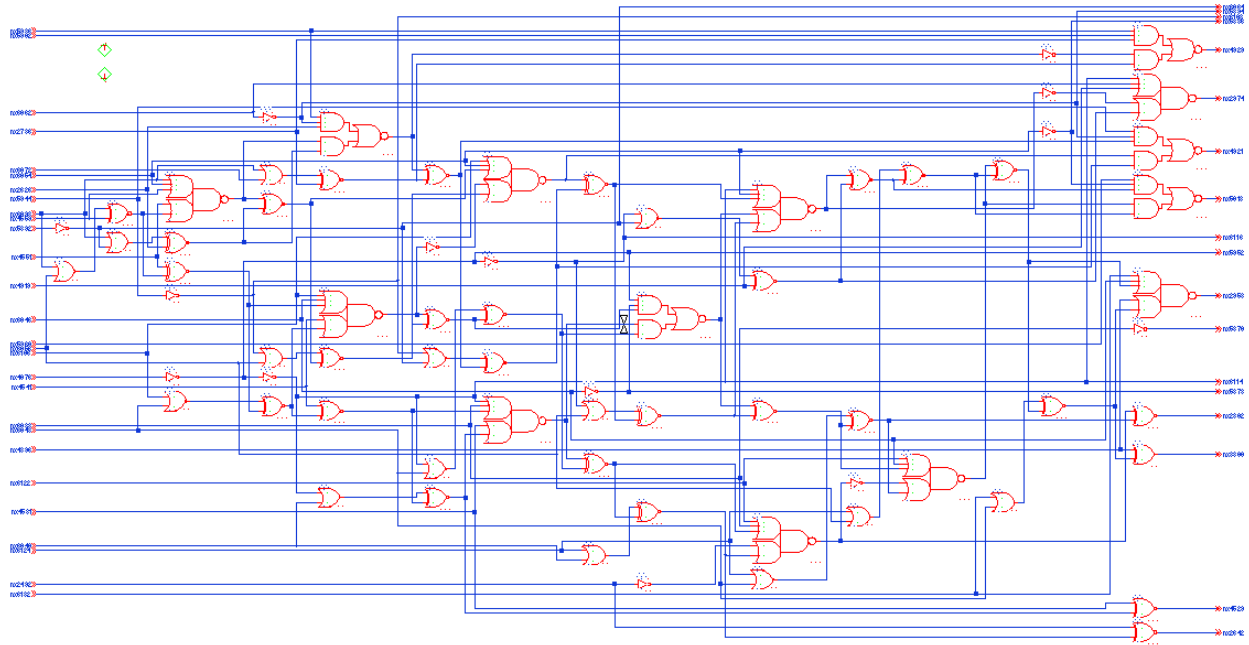


Figure 28: Full Schematic Page 16

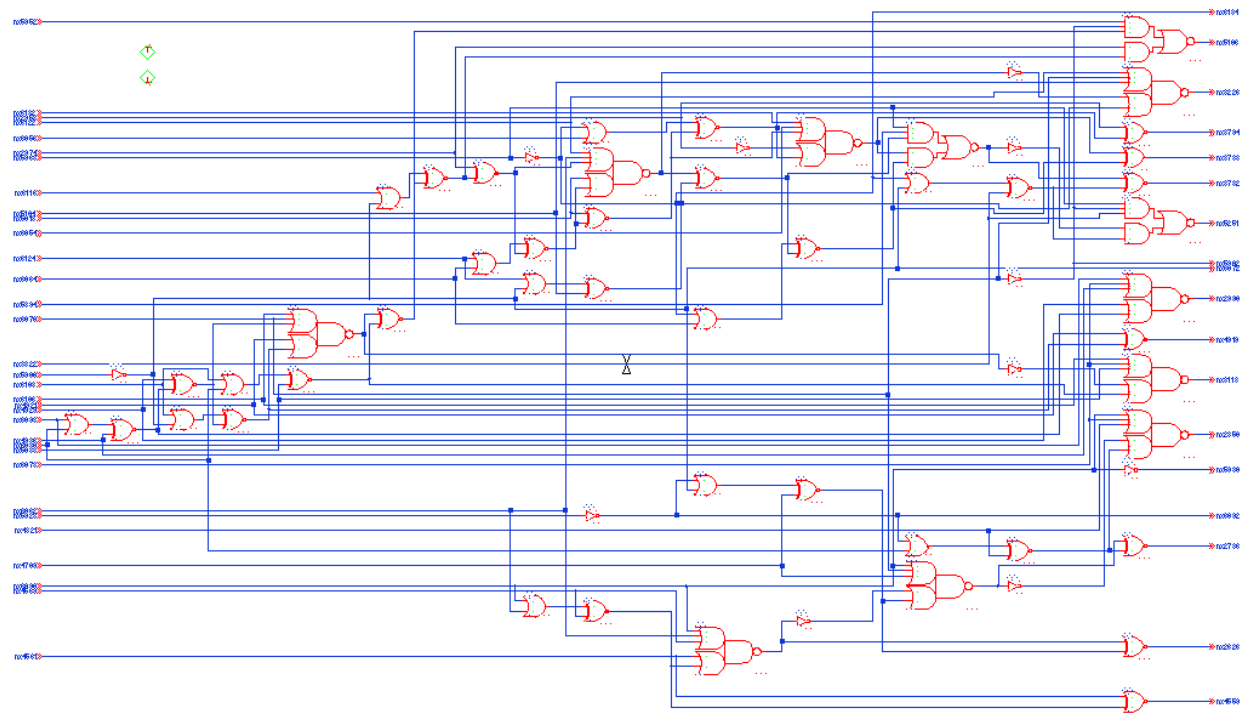


Figure 29: Full Schematic Page 17



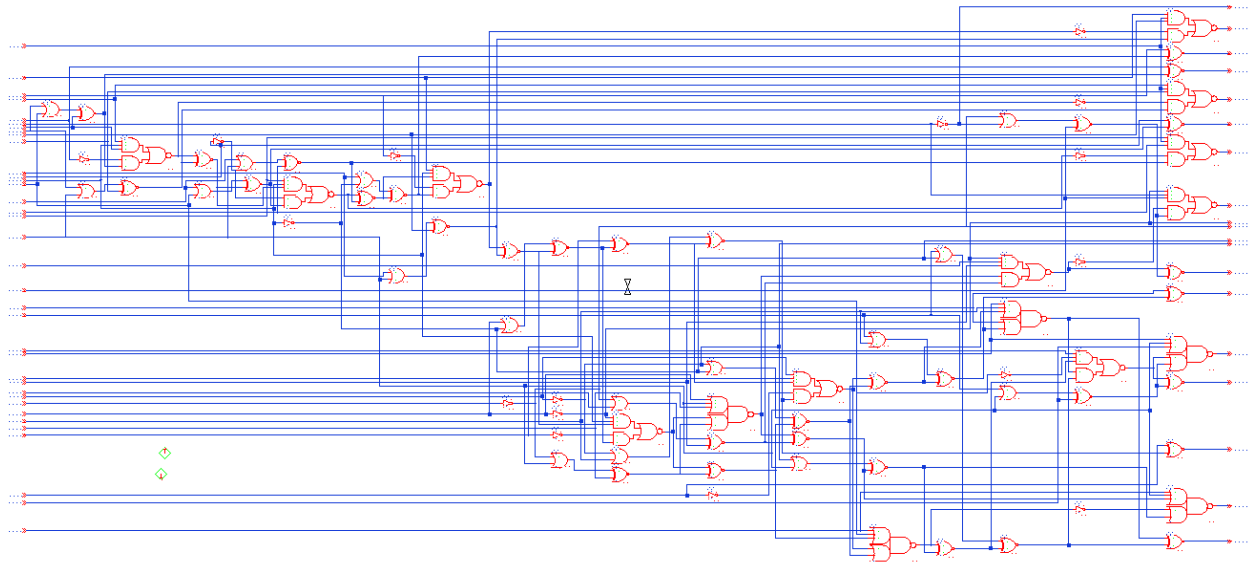


Figure 30: Full Schematic Page 18

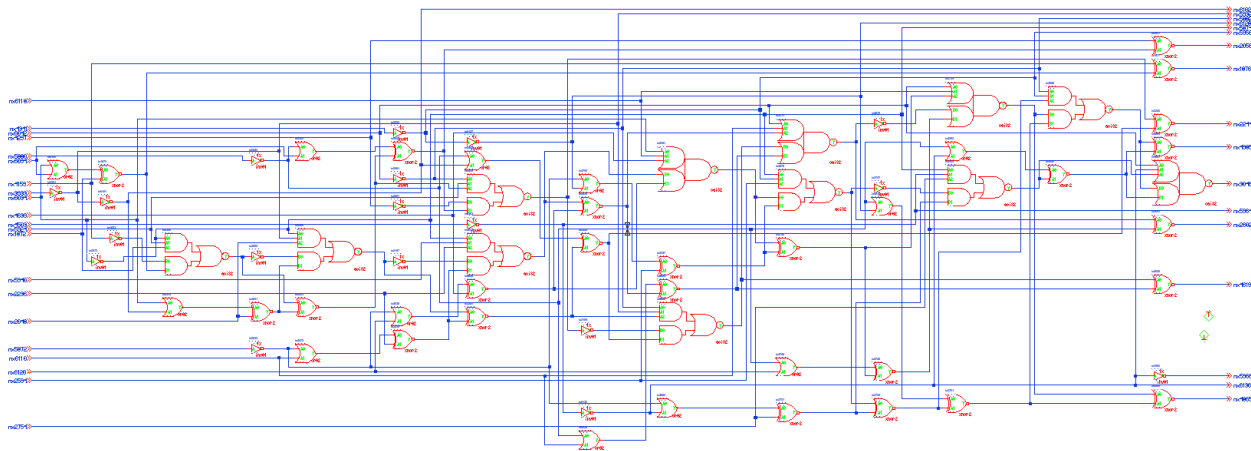


Figure 31: Full Schematic Page 19

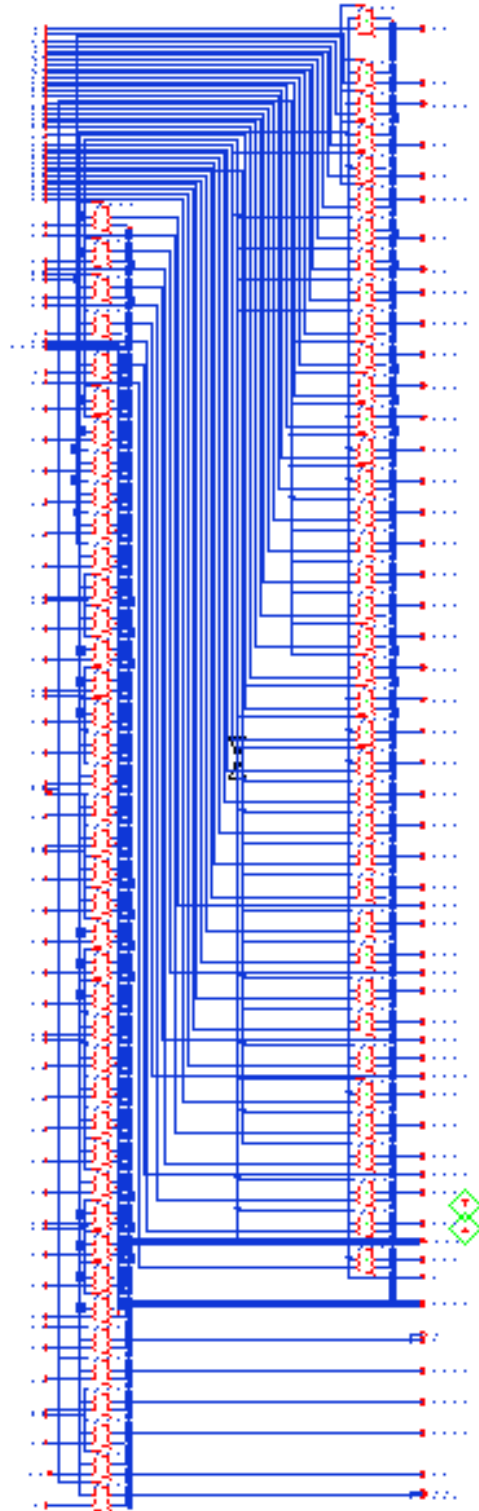


Figure 32: Full Schematic Page 20

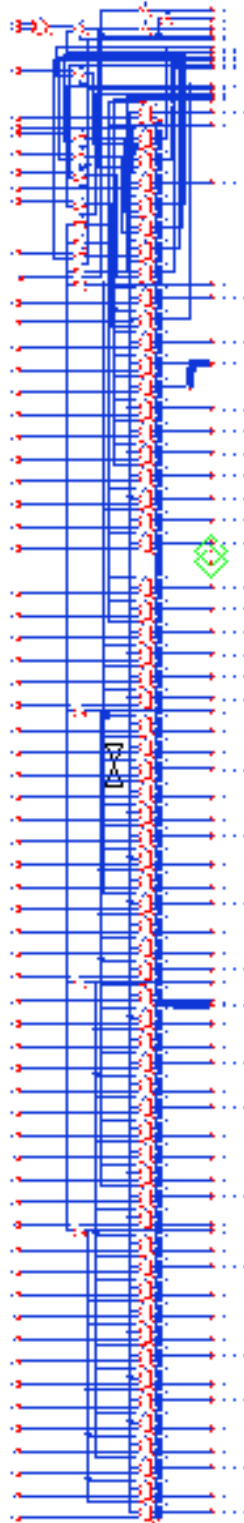


Figure 33: Full Schematic Page 21

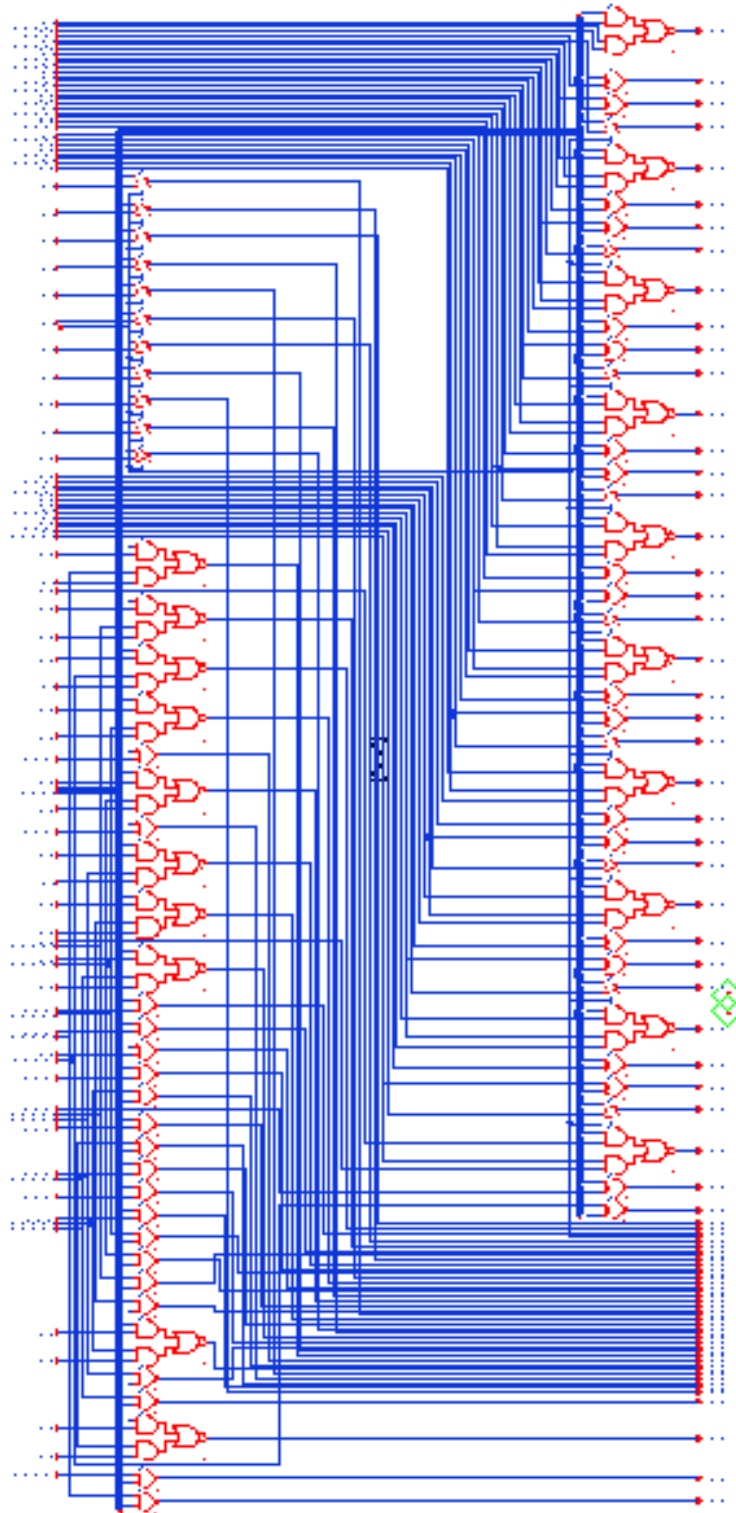


Figure 34: Full Schematic Page 22

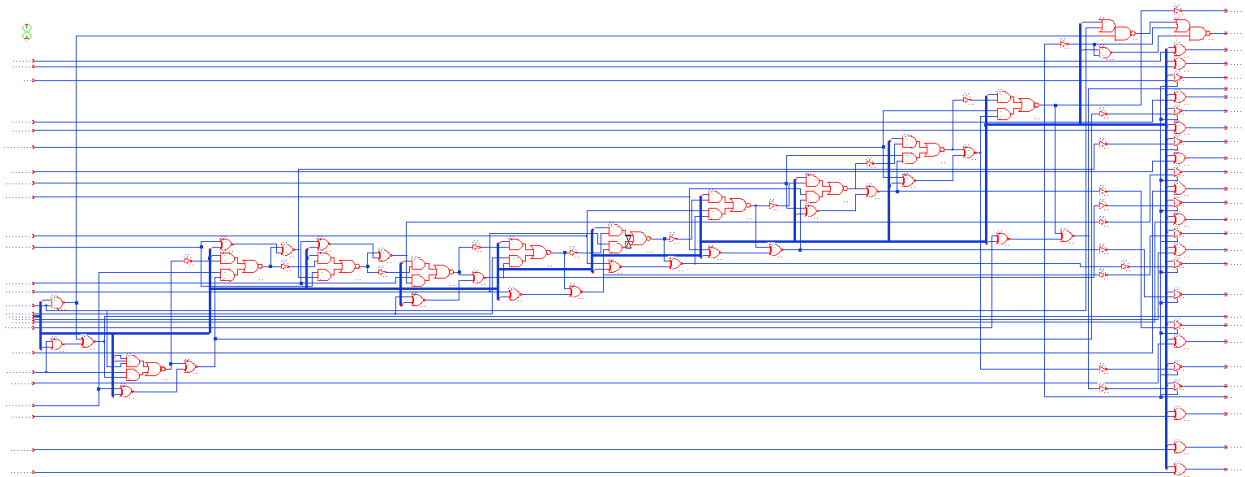


Figure 35: Full Schematic Page 23

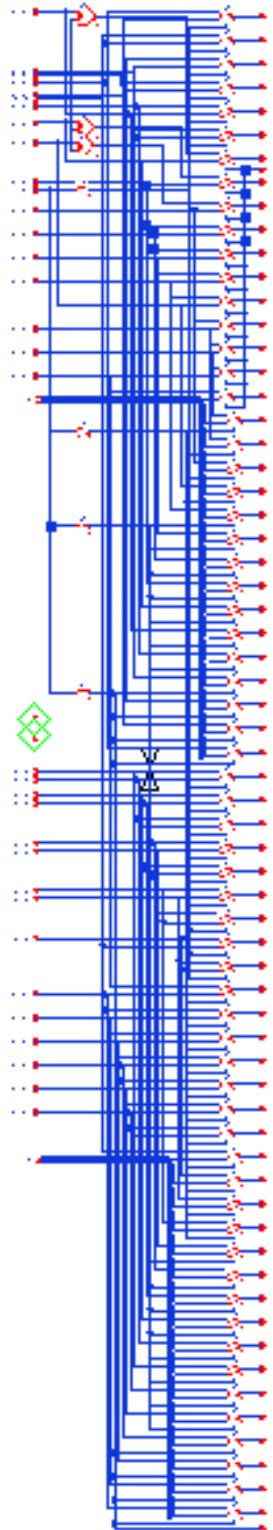


Figure 36: Full Schematic Page 24

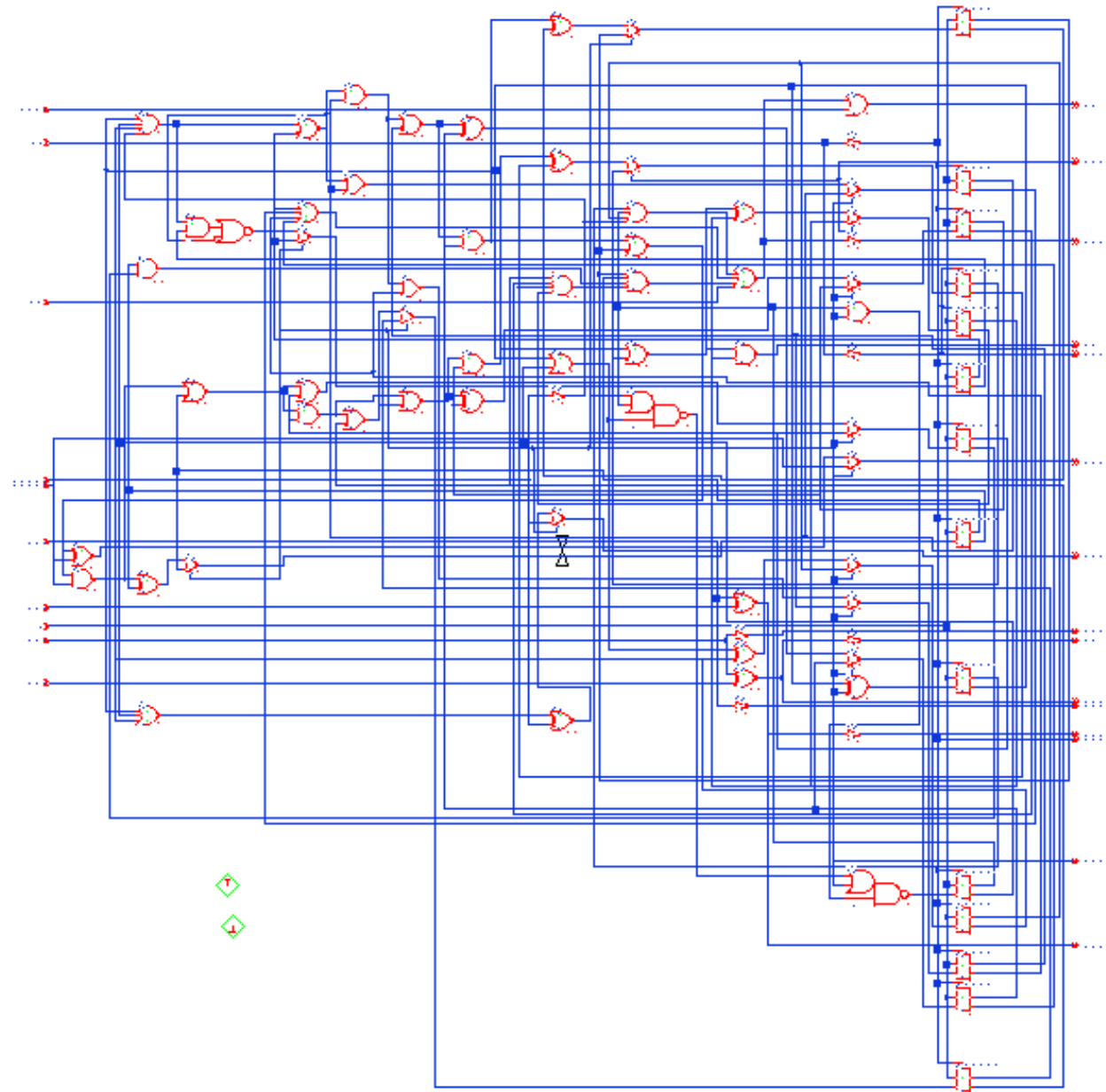


Figure 37: Full Schematic Page 25

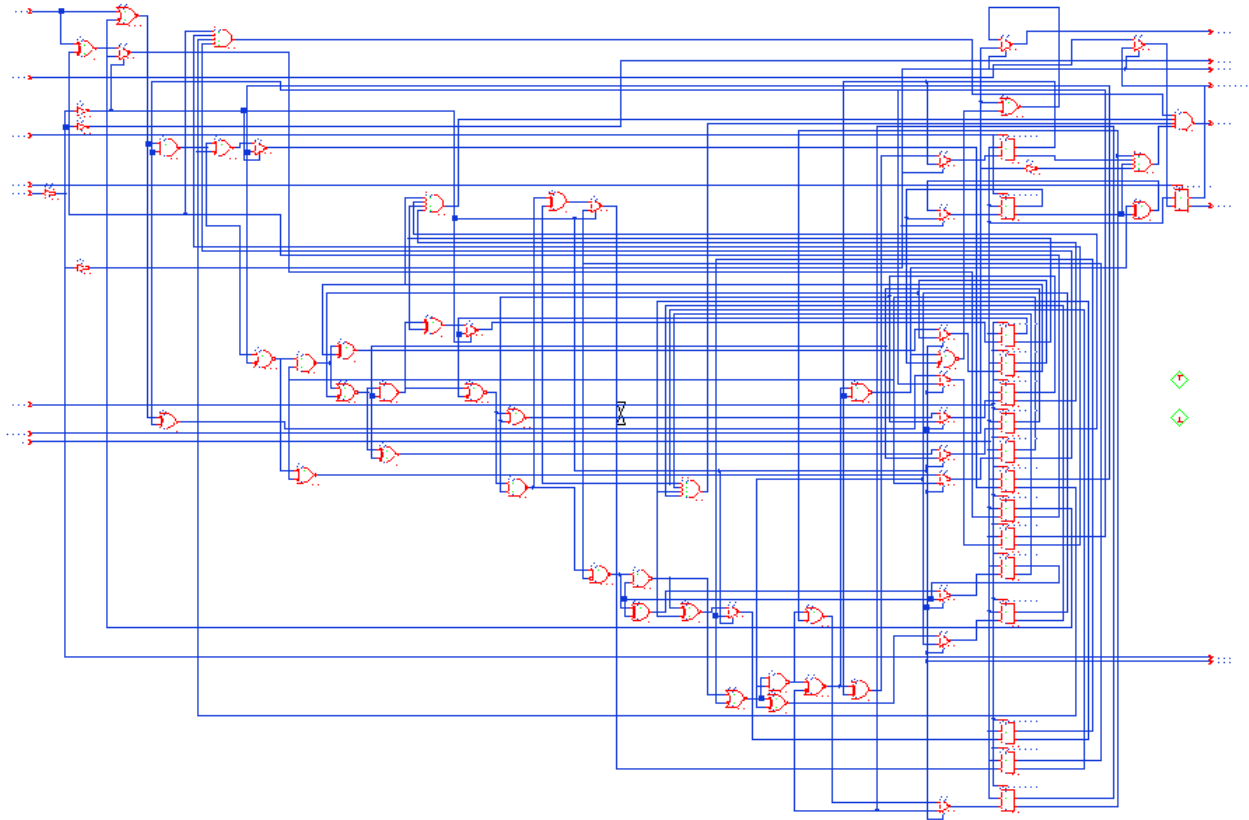


Figure 38: Full Schematic Page 26

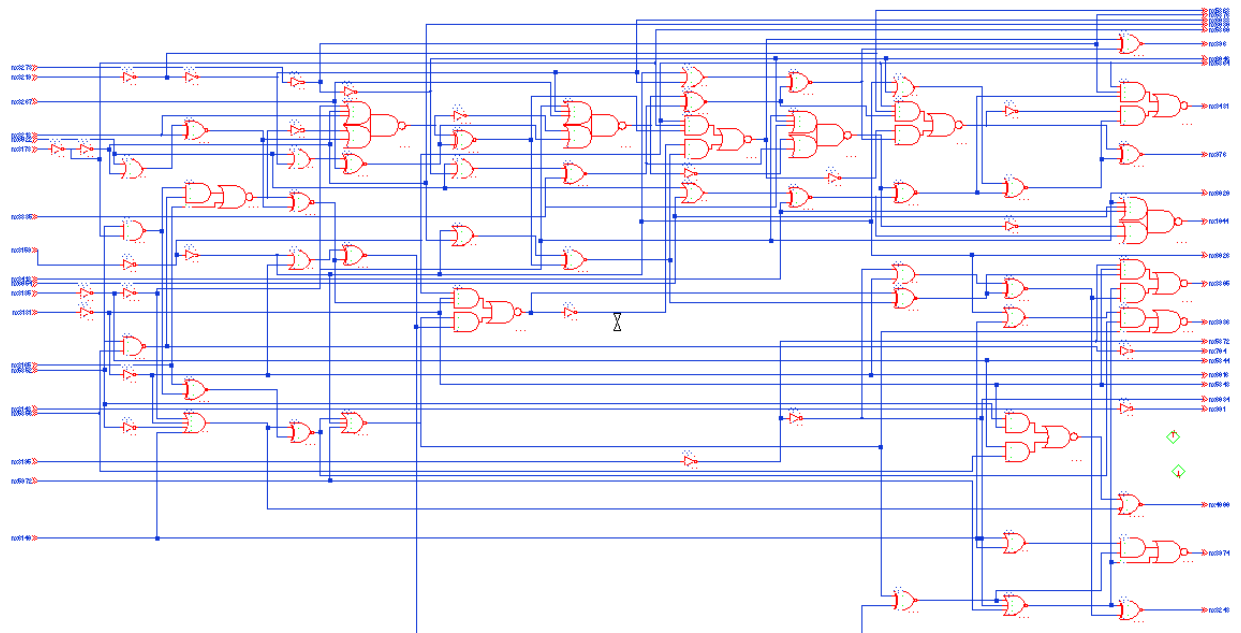


Figure 39: Full Schematic Page 27



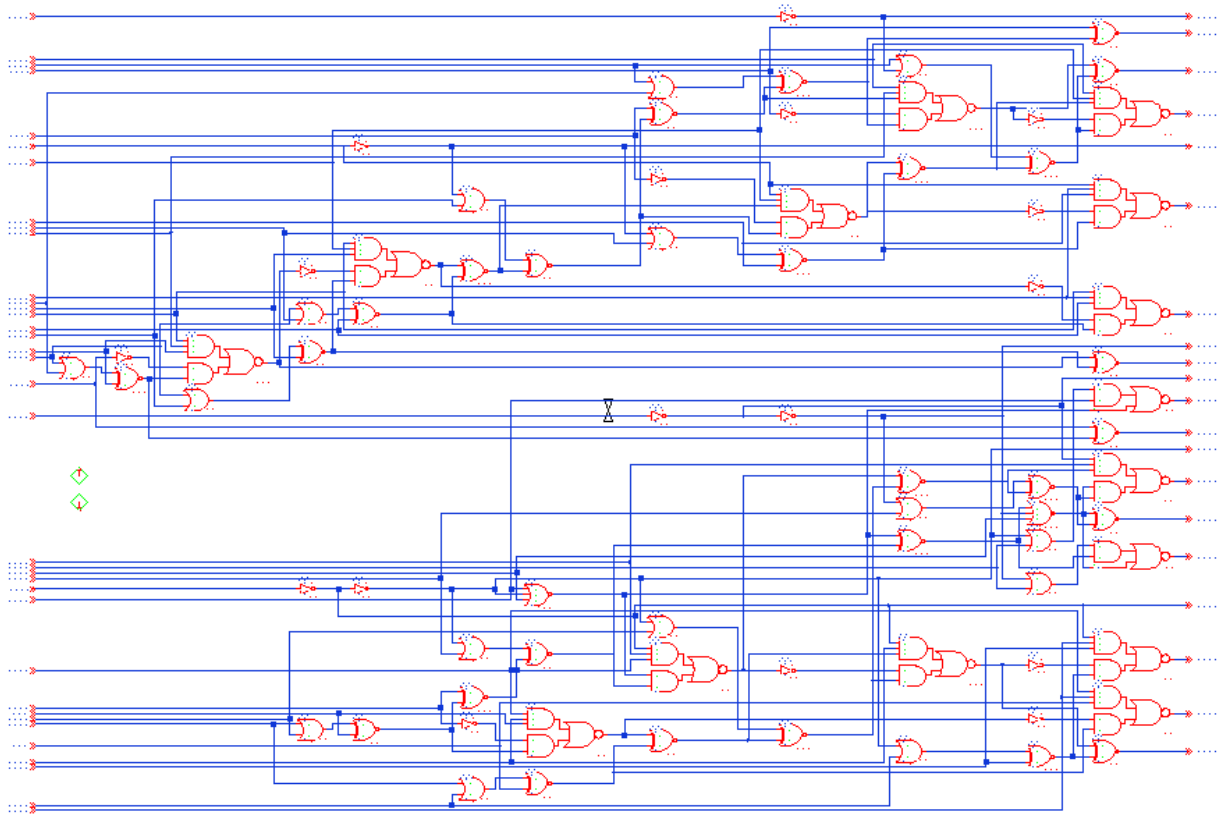


Figure 40: Full Schematic Page 28

### 3 Results and Analysis

The MAC was initially laid out structurally; components were laid out and turned into cells that would then be connected together. This was done to limit the complexity of the final design.

### 3.1 Full Adder

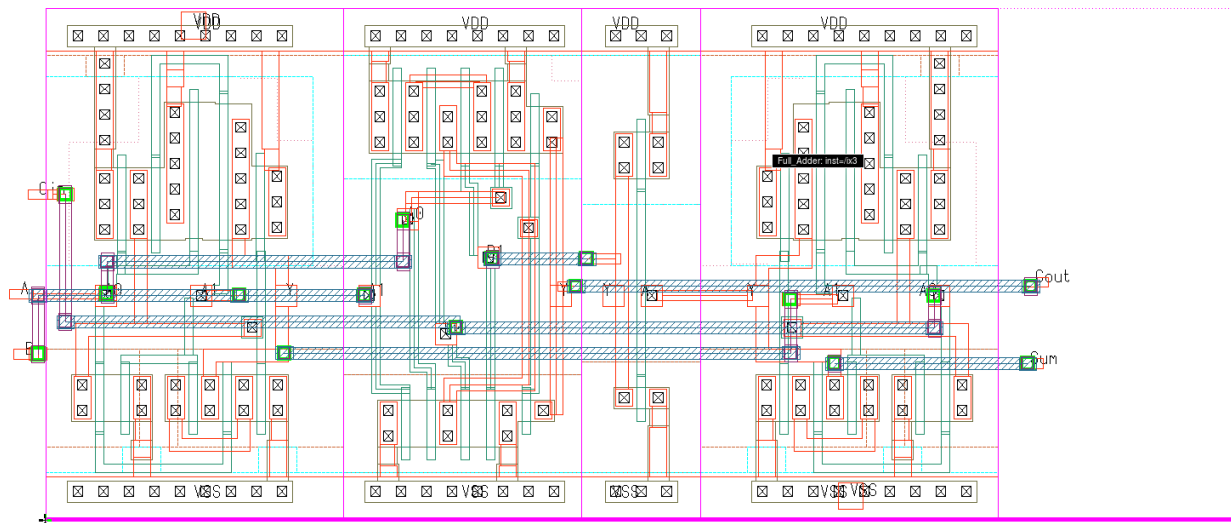


Figure 41: Full Adder Layout

### 3.2 Multiplier

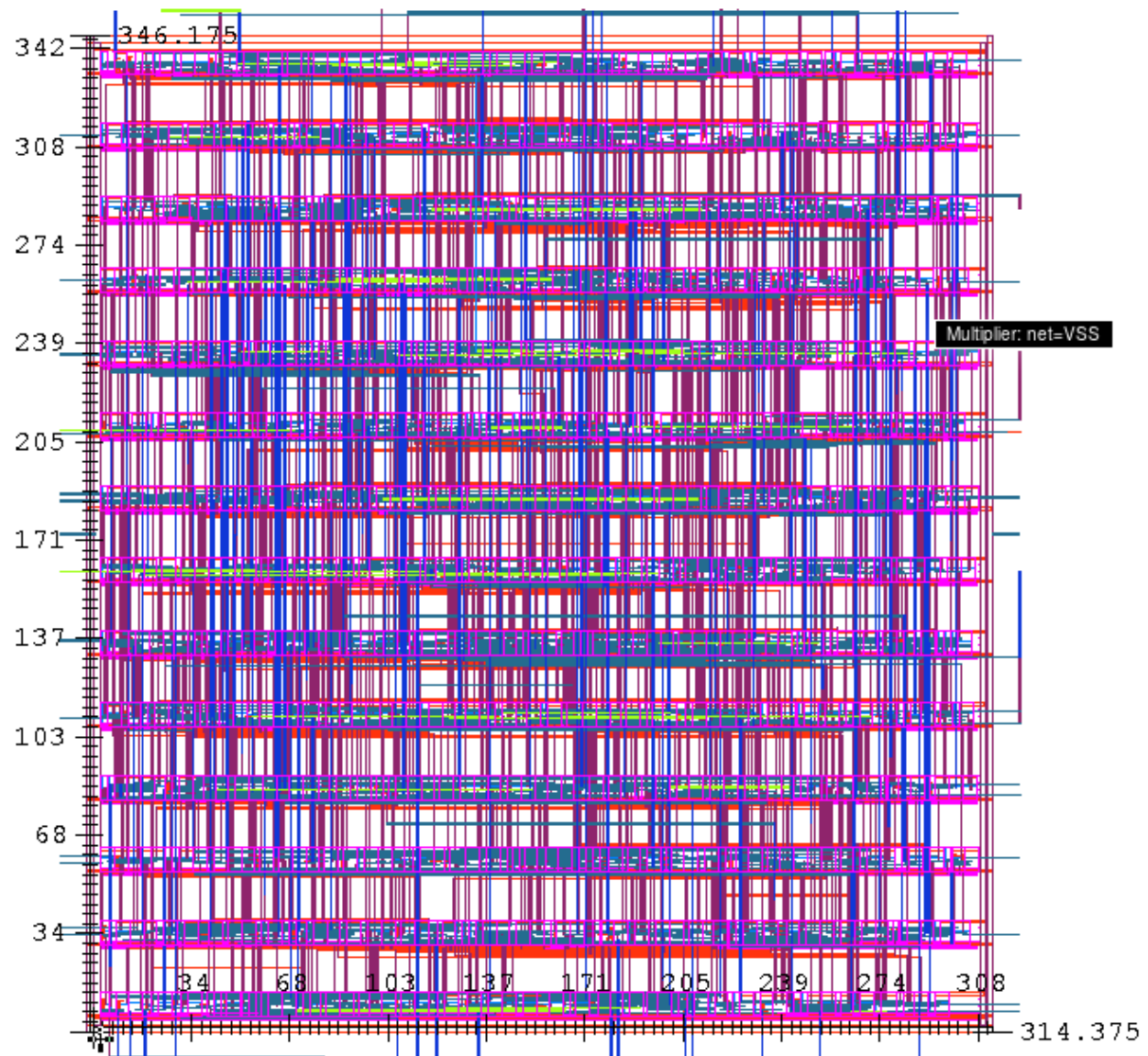


Figure 42: Multiplier Layout

### 3.3 16-Bit Register

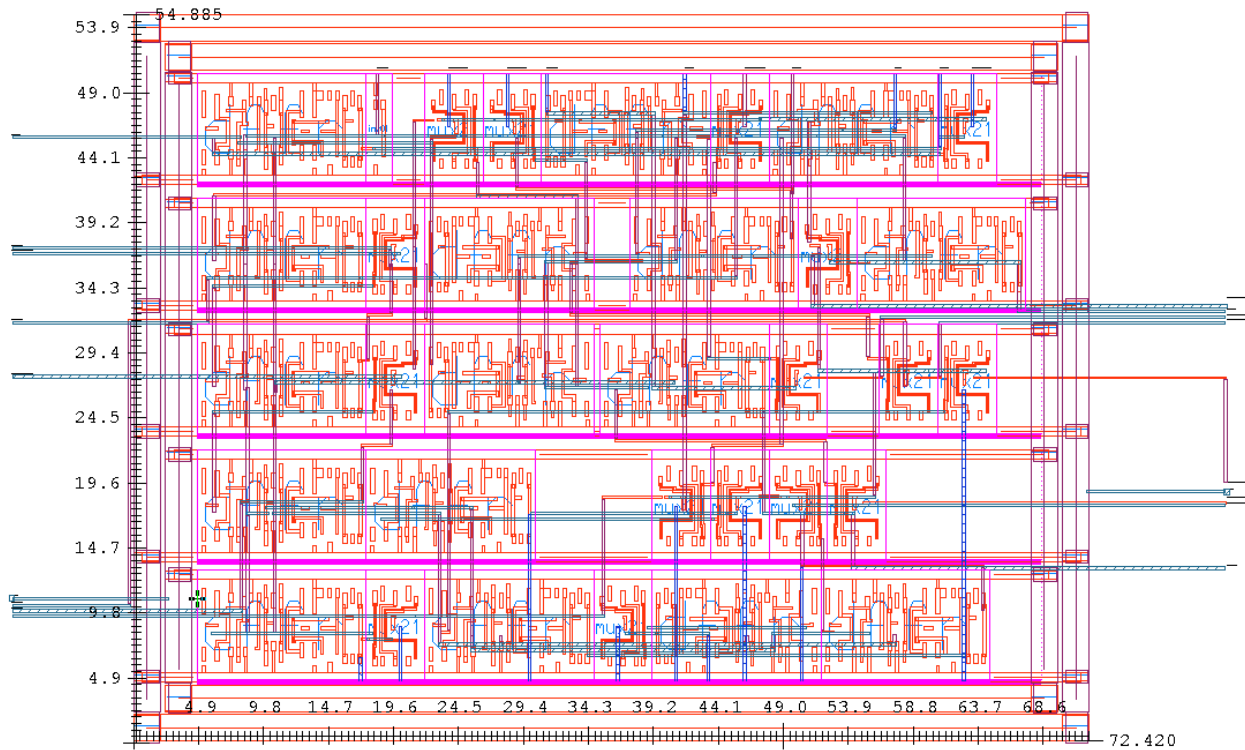


Figure 43: nBitRegister 16 Bit Layout

### 3.4 32-Bit Register

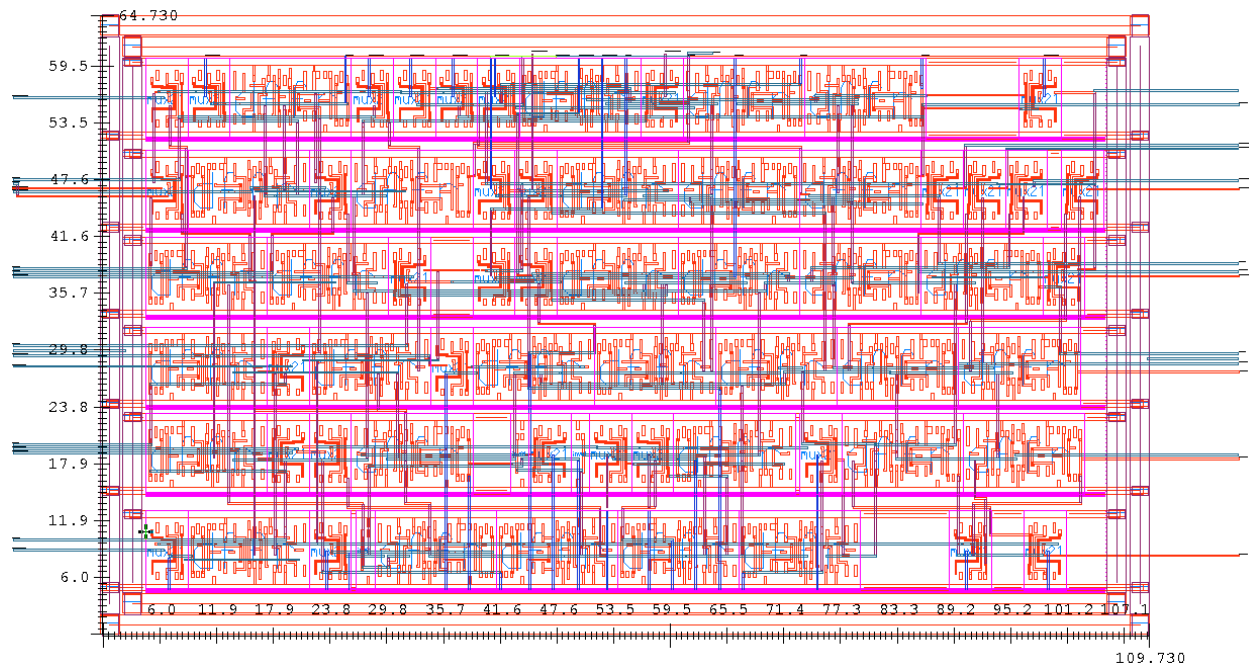


Figure 44: nBitRegister 32 Bit Layout

### 3.5 Mux

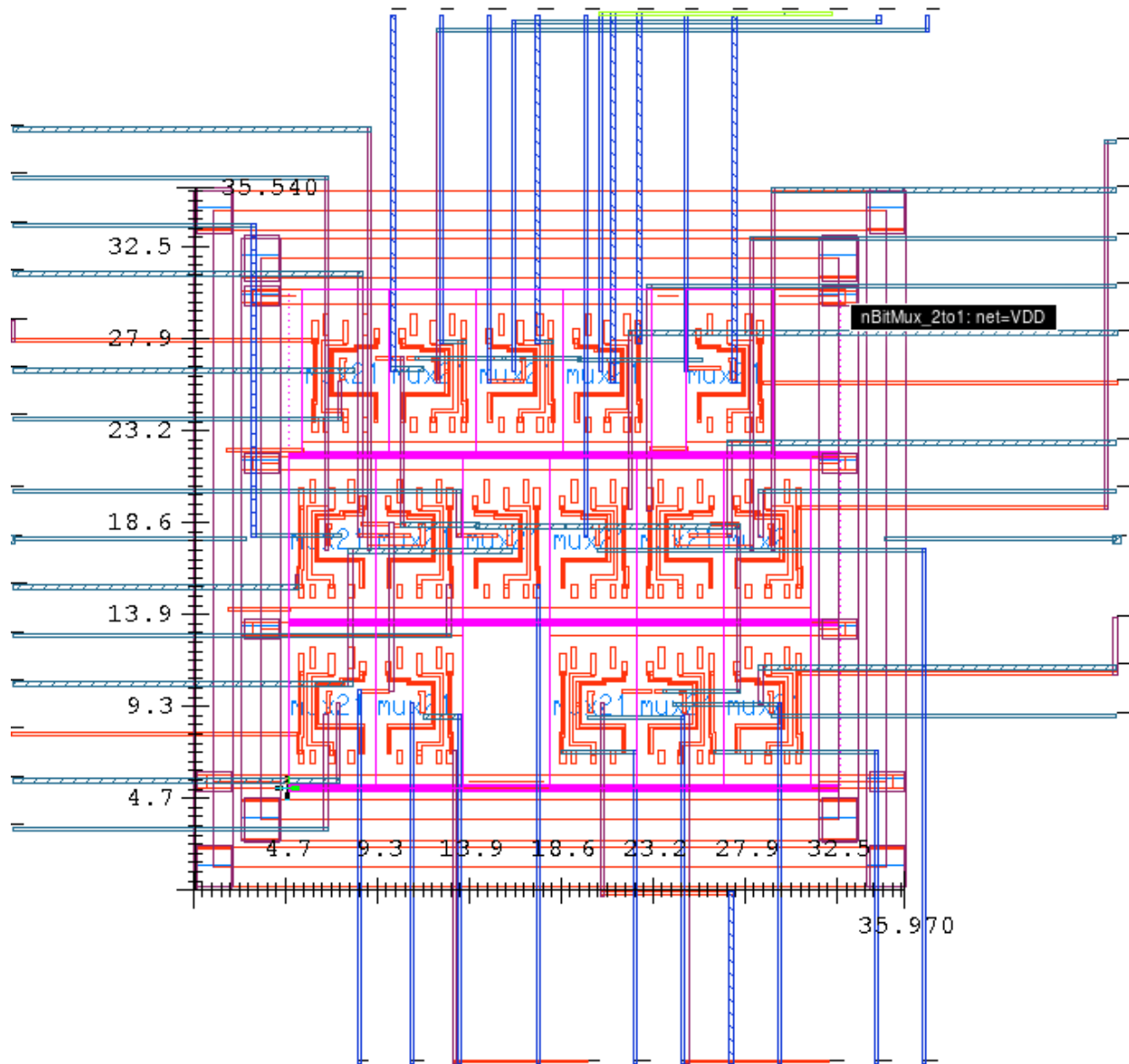


Figure 45: nBitMux 2to1 Layout

### 3.6 MAC with BIST Layout

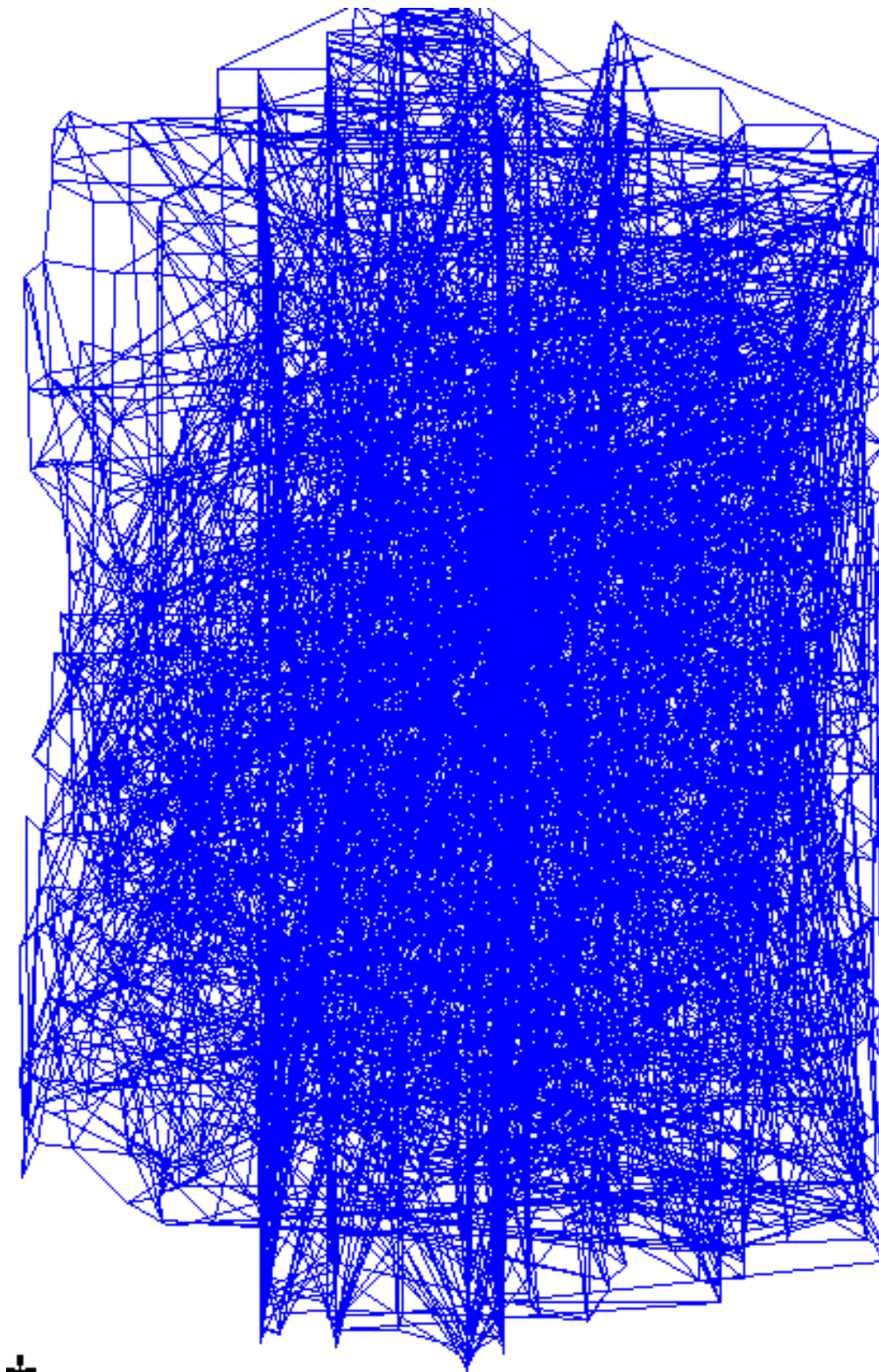


Figure 46: Pre-Layout



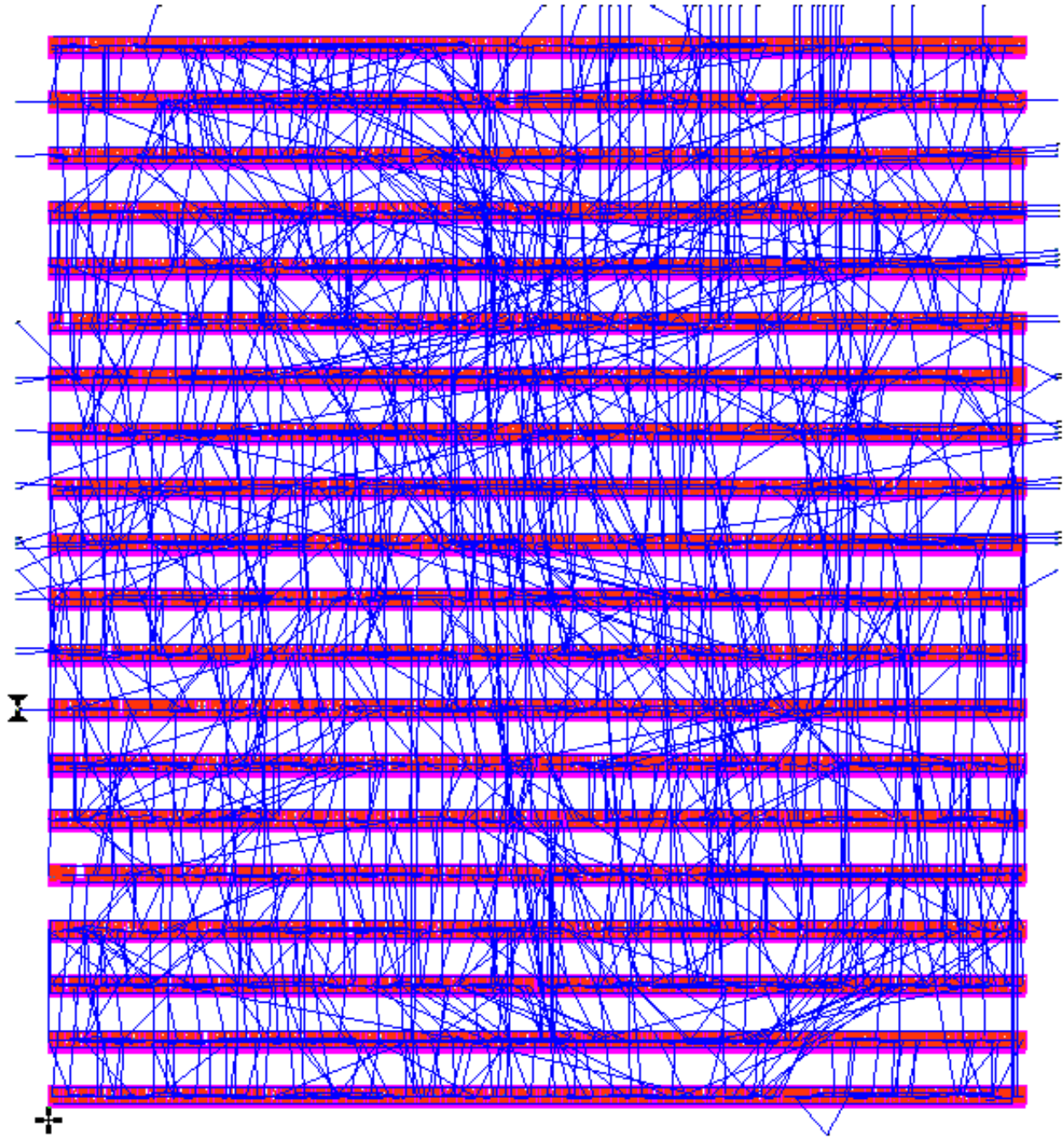


Figure 47: Standard Cells Placement



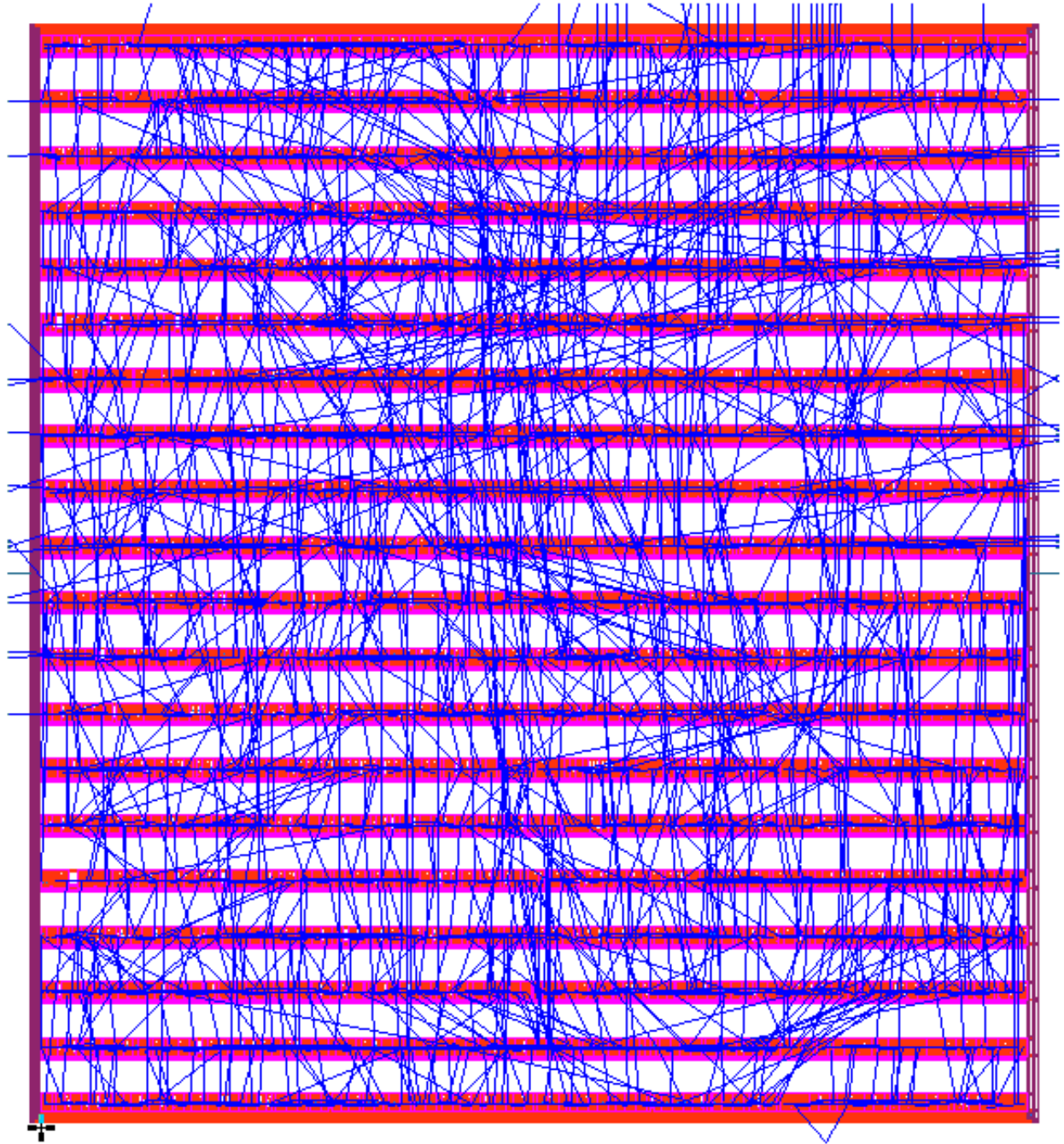


Figure 48: Power Route

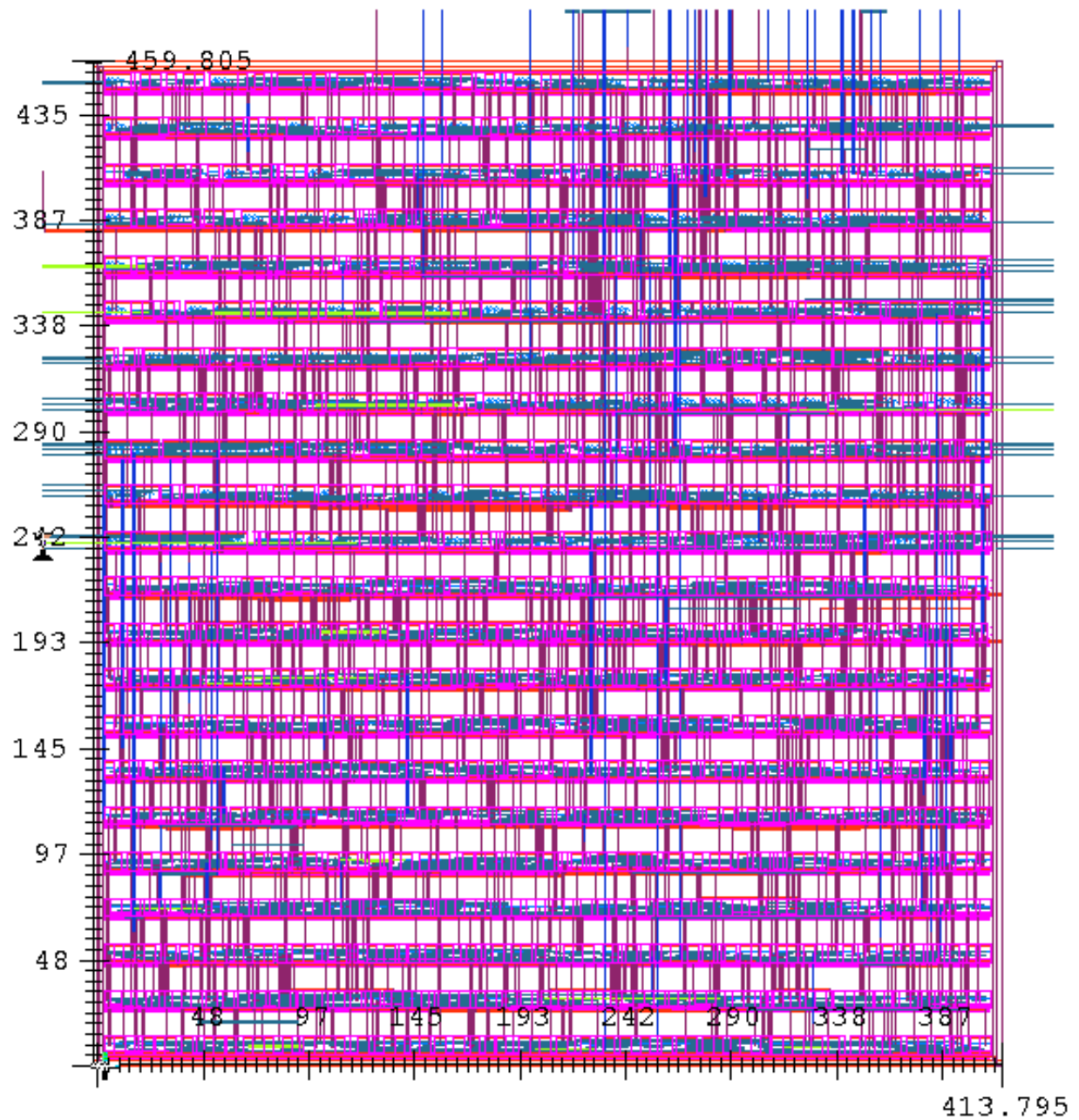


Figure 49: Full Layout

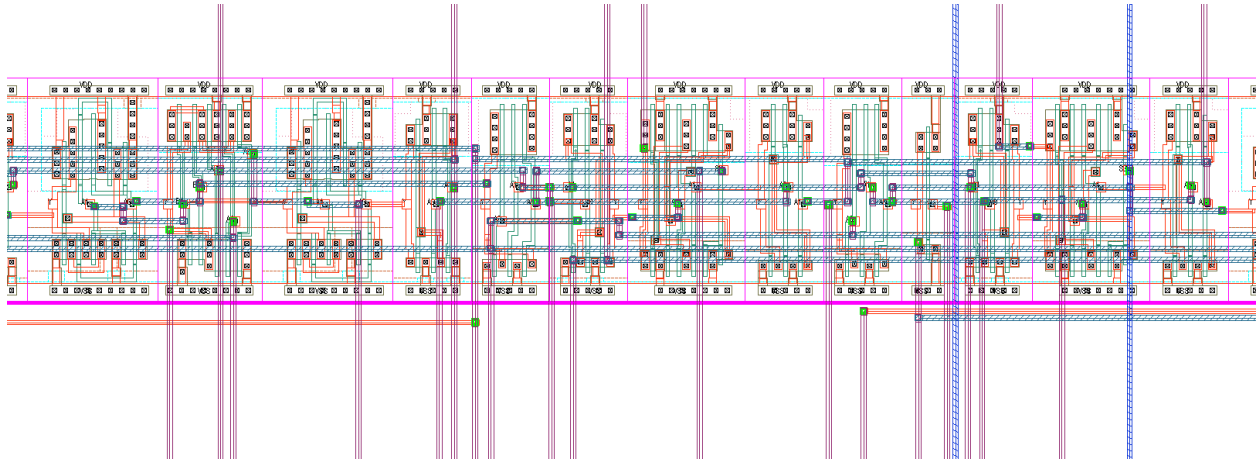


Figure 50: Full Layout Close Up View

### 3.7 Power

Power was measured with an Eldo simulation based on the layout. To measure static power, the average power was measured while the circuit was not active but powered. The static power was measured to be 1.87nW and was recorded in Table 1.

Dynamic power was measured by recording the maximum power measured while the circuit was changing as many transistors as possible. To activate as many transistor as possible, multiple, random inputs were supplied to the circuit for a long period of time (2000us). The maximum power was found to be 16.03mW, which was recorded in Table 1.

Table 1: Simulated Power for MAC

Measured Power (W)	
Static	1.8787E-06
Dynamic	1.6029E-02

It is clear that for this circuit, dynamic power far exceeds the static power. This shows that arithmetic operations draw a lot of power. This is mostly due to their high activity factor and their high speed requirements.

## 4 Conclusion

## 5 Appendix

### 5.1 VHDL

Listing 1: MAC tb VHDL

```

0  -- Testbench created online at:
    -- www.doulos.com/knowhow/perl/testbench-creation/
    -- Copyright Doulos Ltd
    -- SD, 03 November 2002

5  library IEEE;
    use IEEE.Std_logic_1164.all;
    use IEEE.Numeric_Std.all;

    entity MAC_tb is
10 end;

    architecture bench of MAC_tb is

        constant N : integer := 32;

15    component MAC
        generic( N : integer := 32);
        Port ( A : in  STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
              B : in  STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
20          clk : in  STD_LOGIC;
              WE : in  STD_LOGIC;
              reset : in  STD_LOGIC;
              RegOut : out  STD_LOGIC_VECTOR (N-1  downto 0));
    end component;

25    signal A: STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
    signal B: STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
    signal clk: STD_LOGIC;
    signal WE: STD_LOGIC;
30    signal reset: STD_LOGIC;
    signal RegOut: STD_LOGIC_VECTOR (N-1  downto 0);

    begin

35    -- Insert values for generic parameters !!
        uut: MAC generic map ( N      => 32)
            port map ( A      => A,
                      B      => B,
40                  clk     => clk ,
                      WE     => WE,
                      reset  => reset ,
                      RegOut => RegOut );

        clk_proc : process
45        begin
            if clk = '0' then
                clk <= '1';
            else
                clk <= '0';
50            end if;
            wait for 50 ns;
        end process;

        stimulus: process
55        begin

            -- Put initialisation code here

```

```

    WE <= '1';
    reset <= '1';
    A <= "1111111111111111";
    B <= "1111111111111111";

    wait for 300 ns;
    WE <= '0';
    A <= "0000000000010000";
    B <= "0000000000000001";
    wait for 300 ns;
    WE <= '1';

    wait for 300 ns;
    B <= "0000000000000000";

    wait for 300 ns;
    reset <= '0';

    -- Put test bench stimulus code here

    wait;
end process;

end;
```

Listing 2: ProjectWrapper tb VHDL

```

0 library IEEE;
  use IEEE.Std_logic_1164.all;
  use IEEE.Numeric_Std.all;

  entity ProjectWrapper_tb is
5  end;

  architecture bench of ProjectWrapper_tb is

    constant N : integer := 32;

10   component ProjectWrapper
      generic( N : integer := 32);
      Port ( A : in  STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
            B : in  STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
            clk : in STD_LOGIC;
            WE : in STD_LOGIC;
            reset : in STD_LOGIC;
            StartTest : in STD_LOGIC;
            RegOut : out STD_LOGIC_VECTOR (N-1  downto 0);
            Pass : out STD_LOGIC;
            Complete : out STD_LOGIC
20          );
    end component;

25   signal A: STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
   signal B: STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
   signal clk: STD_LOGIC;
   signal WE: STD_LOGIC;
   signal reset: STD_LOGIC;
```

```

30  signal StartTest: STD_LOGIC;
    signal RegOut: STD_LOGIC_VECTOR (N-1 downto 0);
    signal Pass : STD_LOGIC;
    signal Complete : STD_LOGIC;

35  begin

    -- Insert values for generic parameters !!
    uut: ProjectWrapper generic map ( N      => 32)
                        port map ( A      => A,
36                               B      => B,
40                               clk     => clk ,
                               WE      => WE,
                               reset    => reset ,
                               StartTest => StartTest ,
45                               RegOut  => RegOut ,
                               Pass => Pass ,
                               Complete => Complete );

    clk_proc : process
50  begin
        if clk = '0' then
            clk <= '1';
        else
            clk <= '0';
55  end if;
        wait for 50 ns;
    end process;

    stimulus: process
60  begin

        -- Put initialisation code here

65  -- Put test bench stimulus code here

    -- Put initialisation code here
    WE <= '1';
    reset <= '0';
    StartTest <= '1';
    A <= "000000000000010";
    B <= "000000000000010";
    wait for 300 ns;
    WE <= '0';
    wait for 300 ns;
    reset <= '1';
    A <= "000000000010000";
    B <= "000000000000001";
75  WE <= '1';
    wait for 100800 ns;
    reset <= '0';
    StartTest <= '0';
    A <= "000000000000010";
    B <= "000000000000010";
80  wait for 600 ns;
    reset <= '1';
    wait for 600 ns;
85

```

```

    B <= "0000000000000100";

    wait;
end process;

end;

```

Listing 3: FullAdder 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 behavioral 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
    --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",
50      '0' when others;

end   behav;

```

Listing 4: FA 1bit VHDL

```

0  -- Company:
   -- Engineer:
   --
   -- Create Date:      08:18:41 03/02/2017
5  -- Design Name:
   -- Module Name:      FA_1bit - Behavioral
   -- Project Name:
   -- Target Devices:
   -- Tool versions:
10  -- Description:
   --
   -- Dependencies:
   --
   -- Revision:
15  -- Revision 0.01 - File Created
   -- Additional Comments:
   --
   --
20  library IEEE;
   use IEEE.STD_LOGIC_1164.ALL;

   -- Uncomment the following library declaration if using
   -- arithmetic functions with Signed or Unsigned values
   --use IEEE.NUMERIC_STD.ALL;

25  -- Uncomment the following library declaration if instantiating
   -- any Xilinx primitives in this code.
   --library UNISIM;
   --use UNISIM.VComponents.all;

30  entity FA_1bit is
      Port ( A : in  STD_LOGIC;
            B : in  STD_LOGIC;
            Cin : in  STD_LOGIC;
35            S : out STD_LOGIC;
            Cout : out STD_LOGIC);
end FA_1bit;

architecture Behavioral of FA_1bit is
40
begin
    S<=((A xor B) xor Cin); --Sum
    Cout<=((A xor B) and Cin) or (A and B);--Cout

45
end Behavioral;

```



Listing 5: AND2 VHDL

```

0  ---
--- Company:
--- Engineer:
---
--- Create Date:      15:02:42 03/15/2017
5  --- Design Name:
--- Module Name:      AND2 - Behavioral
--- Project Name:
--- Target Devices:
--- Tool versions:
10 --- Description:
---
--- Dependencies:
---
--- Revision:
15 --- Revision 0.01 - File Created
--- Additional Comments:
---

library IEEE;
20 use IEEE.STD_LOGIC_1164.ALL;

--- Uncomment the following library declaration if using
--- arithmetic functions with Signed or Unsigned values
---use IEEE.NUMERIC_STD.ALL;

25 --- Uncomment the following library declaration if instantiating
--- any Xilinx primitives in this code.
---library UNISIM;
---use UNISIM.VComponents.all;

30 entity AND2 is
    Port ( A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          F : out STD_LOGIC);
35 end AND2;

architecture Behavioral of AND2 is

begin
40     F <= A AND B;

end Behavioral;

```

Listing 6: nBitRegister VHDL

```

0  ---
---Company      : RIT
---Author       : Brandon Key
---Created      : 2/8/2017
---
5  ---Project Name : Lab 2
---File         : nBitRegister.vhd
---
---Entity       : nBitRegister
---Architecture : struct
10 ---Revision    :

```

```

15  --Rev 0.01      : 2/8/2017
    --
    --Tool Version : VHDL '93
    --Description  : Entity and behavioral description of an n-bit register
    --
    --Notes        :
    -----

20  library ieee;
    use ieee.std_logic_1164.all;

    entity nBitRegister is
        generic (n : integer := 32);
25      Port (
          nBitIn : in std_logic_vector(n-1 downto 0); -- n bits to store in the
            register
          WE      : in std_logic; -- Active high write enable
          Reset   : in std_logic; -- Async reset, disabled when low
          clk     : in std_logic;
30      Y : out std_logic_vector(n-1 downto 0) -- 1 output, n bits wide
        );
    end nBitRegister;

35  architecture behav of nBitRegister is

    begin

40      output_proc : process (clk, Reset) begin

          if Reset = '0' then
              Y <= (others => '0');
          elsif clk'event and clk = '1' then
45              if WE = '1' then
                  Y <= nBitIn;
              end if;
          end if;

          end process output_proc;

50      end behav;

```

Listing 7: Shifter VHDL

```

0  -----
    -- Company:
    -- Engineer:
    --
    -- Create Date:    09:15:01 03/02/2017
5  -- Design Name:
    -- Module Name:    Shifter - Behavioral
    -- Project Name:
    -- Target Devices:
    -- Tool versions:
10 -- Description:

```

```

--
-- Dependencies:
--
-- Revision:
15 -- Revision 0.01 - File Created
-- Additional Comments:
--

```

---

```

library IEEE;
20 use IEEE.STD_LOGIC_1164.ALL;
use IEEE.MATHREAL.ALL;
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if using
25 -- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
30 --library UNISIM;
--use UNISIM.VComponents.all;

entity Shifter is
    generic( N : integer:=16; Namnt : integer :=integer(ceil(log2(real(16)))));
35
    Port ( A : in  STD_LOGIC_VECTOR (N-1 downto 0);
          amnt : in  STD_LOGIC_VECTOR (Namnt-1 downto 0);
          Control : in  STD_LOGIC_VECTOR (3 downto 0);
          output : out  STD_LOGIC_VECTOR (N-1 downto 0));
40 end Shifter;

--1100 LSL
--1101 LSR
45 --1110 ASR
architecture Behavioral of Shifter is
    signal temp : integer;
begin
    proc1 : process(Control,amnt,A)
50     begin
        if control ="1101" then--LSR
            for i in integer range 0 to N-1 loop
                temp<=to_integer(unsigned(amnt));--convert amnt to unsigned integer
            for indexing
                if i+temp > N-1 then--bits at leftmost
55                 output(i) <='0';
                else
                    output(i)<=A(i+temp);--right Shift
                end if;
            end loop;

60
            elsif control ="1100" then --LSL
                for i in integer range 0 to N-1 loop
                    temp<=to_integer(unsigned(amnt));--convert amnt to unsigned integer
                for indexing
                    if i-temp < 0 then --rightmost bits
65                     output(i) <='0';
                    else
                        output(i)<=A(i-temp);--left shift

```

```

        end if;
    end loop;
else --ASR
    for i in integer range 0 to N-1 loop
        temp<=to_integer(unsigned(amnt));--convert amnt to unsigned integer
    for indexing
        if A(N-1)='1' then--negative
            if i+temp > N-1 then--leftmost bits
                output(i)<='1';--preserve the negative sign
            else
                output(i)<=A(i+temp);-- Right Shift
            end if;
        else--Positive
            if i+temp > N-1 then --leftmost bits
                output(i)<='0';
            else
                output(i)<=A(i+temp);--right shift
            end if;
        end if;
    end loop;
    end if;
end process;
end Behavioral;

```

Listing 8: TestController VHDL

```

0  -- Company:
1  -- Engineer:
2  --
3  -- Create Date:    14:56:40 03/15/2017
4  -- Design Name:
5  -- Module Name:    Multiplier - Behavioral
6  -- Project Name:
7  -- Target Devices:
8  -- Tool versions:
9  -- Description:
10 --
11 -- Dependencies:
12 --
13 -- Revision:
14 -- Revision 0.01 - File Created
15 -- Additional Comments:
16 --
17
18
19
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.numeric_std.all;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx primitives in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;

```

```

entity TestController is
    generic( N : integer := 32);
35   Port ( clk : in STD_LOGIC;
          StartTest : in STD_LOGIC;
          reset_n : in STD_LOGIC;
          Count : in STD_LOGIC_VECTOR (N-1 downto 0);
          MISR_IN : in STD_LOGIC_VECTOR (N-1 downto 0);
40          Complete : out STD_LOGIC;
          Pass : out STD_LOGIC;
          TestEN: out STD_LOGIC
        );
end TestController;
45
architecture Datapath of TestController is

    signal complete_v, pass_v : STD_LOGIC;

50 begin

    PassProc : process (clk, reset_n) begin

        if reset_n = '0' then
55            Pass_v <= '0';
        elsif rising_edge(clk) then
            if (count = "000000000000000000000000111101000" and MISR_IN = "
10001100100101111000000111100110") or Pass_v = '1' then
                Pass_v <= '1';
            else
60                Pass_v <= '0';
            end if;
        end if;
    end process;

    CompleteProc : process (clk, reset_n) begin
65        if reset_n = '0' then
            Complete_v <= '0';
        elsif rising_edge(clk) then
            if count = "000000000000000000000000111101000" or Complete_v = '1' then
70                Complete_v <= '1';
            else
                Complete_v <= '0';
            end if;
        end if;
    end process;

75    TestProc : process (clk, reset_n) begin
        if reset_n = '0' then
            TestEN <= '0';
        elsif rising_edge(clk) then
80            if StartTest = '1' then
                TestEN <= '1';
            else
                TestEN <= '0';
            end if;
        end if;
    end process;

    --Assign outputs
90    Complete <= Complete_v;

```

```

    Pass <= Pass_v;

95 end Datapath;

```

Listing 9: Subtractor VHDL

```

0  --- Company:
1  --- Engineer:
2  ---
3  --- Create Date:    10:15:15 03/19/2017
4  --- Design Name:
5  --- Module Name:    Subtractor - Behavioral
6  --- Project Name:
7  --- Target Devices:
8  --- Tool versions:
9  --- Description:
10 ---
11 --- Dependencies:
12 ---
13 --- Revision:
14 --- Revision 0.01 - File Created
15 --- Additional Comments:
16 ---
17
18
19 library IEEE;
20 use IEEE.STD_LOGIC_1164.ALL;
21
22 --- Uncomment the following library declaration if using
23 --- arithmetic functions with Signed or Unsigned values
24 ---use IEEE.NUMERIC_STD.ALL;
25
26 --- Uncomment the following library declaration if instantiating
27 --- any Xilinx primitives in this code.
28 ---library UNISIM;
29 ---use UNISIM.VComponents.all;
30
31 entity Subtractor is
32     generic(N : integer :=16);
33     Port ( A : in  STD_LOGIC_VECTOR (15 downto 0);
34           B : in  STD_LOGIC_VECTOR (15 downto 0);
35           Output : out STD_LOGIC_VECTOR (15 downto 0));
36 end Subtractor;
37
38
39 architecture Behavioral of Subtractor is
40 ---Component Declarations
41 ---Adder, to add 1
42 component Ripple_Carry_FA is
43     generic(N : integer :=16);---Number of bits in A and B
44     Port ( A : in  STD_LOGIC_VECTOR (N-1 downto 0);
45           B : in  STD_LOGIC_VECTOR (N-1 downto 0);
46           Cin : in STD_LOGIC;
47           Sum : out STD_LOGIC_VECTOR (N-1 downto 0);
48           Cout : out STD_LOGIC);
49 end component;

```

```

50  --Logic, for bitwise not
    component Logic_Unit is
        generic( N : integer :=16);
        Port ( A : in  STD_LOGIC_VECTOR (N-1 downto 0);
              B : in  STD_LOGIC_VECTOR (N-1 downto 0);
55         Control : in  STD_LOGIC_VECTOR (3 downto 0);
              output : out  STD_LOGIC_VECTOR (N-1 downto 0));
    end component;

--Signal declarations
60 signal logicOut, negative : STD_LOGIC_VECTOR(N-1 downto 0);
signal logicControl : STD_LOGIC_VECTOR(3 downto 0):="1001";--Set to bitwise NOT
signal one : STD_LOGIC_VECTOR(N-1 downto 0) := "0000000000000001";
signal Cout : STD_LOGIC;--Not used

65 begin
--Convert Input 2 to a negative number
--Bitwise not input2
    LOGIC : Logic_Unit
        generic map(N=>N)
70     port map(A=>B, B=>A, Control=>logicControl, output=>logicOut);--A=>B because
        it does bitwise NOT only on A

--Add 1
    ADD1 : Ripple_Carry_FA
        generic map(N=>N)
75     port map(A=>logicOut, B=>one, Cin=>'0', Sum=>negative, Cout=>Cout);

--Add the positive A with the newly created negative B
    ADD2 : Ripple_Carry_FA
        generic map(N=>N)
80     port map(A=>A, B=>negative, Cin=>'0', Sum=>Output, Cout=>Cout);
--output now has the result of A-B

end Behavioral;

```

Listing 10: Controller VHDL

```

0  -----
--Company      : RIT
--Author       : Brandon Key
--Created      : 03/29/2018
--
5  --Project Name : Lab 5
--File         : Controller.vhd
--
--Entity       : Controller
--Architecture : behav
10 --
--Tool Version : VHDL '93
--Description  : Contoller For BIST
-----

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

20 entity Controller is

```

```

    port(
        start_BIST : in std_logic;

        clk : in std_logic;
        rst_n : in std_logic;

        is_testing : out std_logic;
        mul_input_ctrl : out std_logic;
        EN_LFSR : out std_logic;
        rst_n_LFSR : out std_logic;
        EN_MISR : out std_logic;
        rst_n_MISR : out std_logic
    );
end Controller;

architecture struct of Controller is

    type state_type is (multiply, start_test, testing, test_hold);
    signal state, next_state : state_type := multiply;

    signal counter : integer;

begin

    --update the state to the next_state
    state_proc : process (clk, rst_n) begin
        if rst_n = '0' then
            state <= multiply;
        elsif rising_edge(clk) then
            state <= next_state;
        end if;
    end process state_proc;

    --How to change the state
    next_state_proc : process (clk, rst_n) begin
        if rst_n = '0' then
            next_state <= multiply;
        elsif rising_edge(clk) then
            case (next_state) is
                when multiply =>
                    if start_BIST = '1' then
                        next_state <= start_test;
                    else
                        next_state <= multiply;
                    end if;

                when testing =>
                    counter <= counter + 1;
                    if counter = 255 then
                        next_state <= test_hold;
                    else
                        next_state <= testing;
                    end if;

                when start_test =>
                    next_state <= testing;
                    counter <= 0;
                when test_hold =>
                    if start_BIST = '1' then

```



```

80         next_state <= test_hold;
           else
             next_state <= multiply;
           end if;
       when others =>
85         next_state <= multiply;
       end case;

       end if;
   end process next_state_proc;

90
   --Outputs for the states
   out_proc : process (clk) begin
       if rising_edge(clk) then
           case (state) is
95             when multiply =>
                 is_testing <= '0';
                 mul_input_ctrl <= '1';
                 EN_LFSR <= '0';
                 rst_n_LFSR <= '0';
100                EN_MISR <= '0';
                 rst_n_MISR <= '1';

             when start_test =>
                 is_testing <= '1';
                 mul_input_ctrl <= '0';
105                EN_LFSR <= '1';
                 rst_n_LFSR <= '1';
                 EN_MISR <= '0';
                 rst_n_MISR <= '0';

             when testing =>
                 is_testing <= '1';
                 mul_input_ctrl <= '0';
110                EN_LFSR <= '1';
                 rst_n_LFSR <= '1';
                 EN_MISR <= '1';
                 rst_n_MISR <= '1';

             when test_hold =>
                 is_testing <= '0';
                 mul_input_ctrl <= '0';
115                EN_LFSR <= '0';
                 rst_n_LFSR <= '1';
                 EN_MISR <= '0';
                 rst_n_MISR <= '1';

             when others =>
                 is_testing <= '0';
                 mul_input_ctrl <= '1';
120                EN_LFSR <= '0';
                 rst_n_LFSR <= '0';
                 EN_MISR <= '0';
                 rst_n_MISR <= '0';

           end case;
       end if;
125   end process out_proc;

   end struct;

```

Listing 11: ProjectWrapper VHDL

```

0 library IEEE;
  use IEEE.STD_LOGIC_1164.ALL;
  use IEEE.NUMERIC_STD.ALL;

  -- Uncomment the following library declaration if using
  -- arithmetic functions with Signed or Unsigned values
  --use IEEE.NUMERIC_STD.ALL;

  -- Uncomment the following library declaration if instantiating
  -- any Xilinx primitives in this code.
10 --library UNISIM;
  --use UNISIM.VComponents.all;

  entity ProjectWrapper is
    generic( N : integer := 32);
    Port ( A : in  STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
15         B : in  STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
          clk : in  STD_LOGIC;
          WE : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          StartTest : in  STD_LOGIC;
          RegOut : out  STD_LOGIC_VECTOR (N-1  downto 0);
          Pass : out  STD_LOGIC;
          Complete : out  STD_LOGIC
20         );
  end ProjectWrapper;

  architecture Behavioral of ProjectWrapper is
    --COMPONENT DECLARATIONS
    component MAC is
30      generic( N : integer := 32);
      Port ( A : in  STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
            B : in  STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
            clk : in  STD_LOGIC;
            WE : in  STD_LOGIC;
            reset : in  STD_LOGIC;
            RegOut : out  STD_LOGIC_VECTOR (N-1  downto 0));
    end component;

    component LFSR_32_4 is
40      generic (N : integer := 32);
      port(
        clk : in  std_logic;
        rst_n : in  std_logic;
        en : in  std_logic;
        bit_pattern : out  std_logic_vector(N-1  downto 0)
45      );
    end component;

    component MISR_32_4 is
50      generic (N : integer := 32);
      port(
        MISR_in : in  std_logic_vector(N-1  downto 0);
        clk : in  std_logic;
        rst_n : in  std_logic;

```

```

55         en      : in std_logic;
           MISR_out : out std_logic_vector(N-1 downto 0)
           );
end component;

60 component nBitMux_2to1 is
   generic (n : integer := 16);
   port(
       A,B : in  std_logic_vector(n-1 downto 0);
       sel : in  std_logic;
65       Y  : out std_logic_vector(n-1 downto 0)
       );
end component;

component TestController is
70   generic( N : integer := 32);
   Port ( clk : in STD_LOGIC;
         StartTest : in STD_LOGIC;
         reset_n : in STD_LOGIC;
         Count : in  STD_LOGIC_VECTOR (N-1 downto 0);
75         MISR_IN : in  STD_LOGIC_VECTOR (N-1 downto 0);
         Complete : out STD_LOGIC;
         Pass : out STD_LOGIC;
         TestEN: out STD_LOGIC
         );
80 end component;

component Counter is
   generic( N : integer := 32);
   Port ( clk : in STD_LOGIC;
85         TestEN : in STD_LOGIC;
         reset : in STD_LOGIC;
         Count : out  STD_LOGIC_VECTOR (N-1 downto 0));
end component;

90 --SIGNAL DECLARATIONS
signal MACA, MACB : STD_LOGIC_VECTOR ((N/2) - 1 downto 0);
signal MACOUT : STD_LOGIC_VECTOR (N-1 downto 0);
signal LFSROUT : std_logic_vector(N-1 downto 0);
signal MISR_out, CounterOut : std_logic_vector(N-1 downto 0);
95 signal TestEN : STD_LOGIC;

begin
   --Map those ports
   MAC0 : MAC
       generic map(N => 32)
100   port map ( A => MACA, B => MACB,
               clk => clk, WE => WE, reset => reset,
               RegOut => MACOUT);

   LFSR0 : LFSR_32_4
105   generic map (N => 32)
   port map( clk => clk, rst_n => reset, en => TestEN,
             bit_pattern => LFSROUT);

   MUXA : nBitMux_2to1
110   generic map (N => 16)
   port map (A => A, B => LFSROUT(N-1 downto N/2),
             sel => TestEN, Y => MACA);

```

```

115     MUXB : nBitMux_2to1
         generic map (N => 16)
         port map (A => B, B => LFSROUT((N/2) - 1 downto 0),
                   sel => TestEN, Y => MACB);

120     MISR0 : MISR_32_4
         generic map(N => 32)
         port map( MISR_in => MACOUT, clk => clk,
                   rst_n => reset, en => TestEN,
                   MISR_out => MISR_out);

125     TEST0 : TestController
         generic map( N => 32)
         port map(clk => clk, StartTest => StartTest,
                   reset_n => reset, Count => CounterOut,
                   MISR_IN => MISR_out, Complete => Complete,
130     Pass => Pass, TestEN => TestEN);

     COUNT0 : Counter
         generic map( N => 32)
         port map( clk => clk, TestEN => TestEN, reset => reset,
135     Count => CounterOut);

     RegOut <= MACOUT;

end Behavioral;

```

Listing 12: Counter VHDL

```

0  ---
--- Company:
--- Engineer:
---
--- Create Date:    14:56:40 03/15/2017
5  --- Design Name:
--- Module Name:    Multiplier - Behavioral
--- Project Name:
--- Target Devices:
--- Tool versions:
10 --- Description:
---
--- Dependencies:
---
--- Revision:
15 --- Revision 0.01 - File Created
--- Additional Comments:
---
---
20 library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

--- Uncomment the following library declaration if using
--- arithmetic functions with Signed or Unsigned values
25 ---use IEEE.NUMERIC_STD.ALL;

--- Uncomment the following library declaration if instantiating
--- any Xilinx primitives in this code.
---library UNISIM;

```

```

30 --use UNISIM.VComponents.all;

entity Counter is
    generic( N : integer := 32);
    Port ( clk : in STD_LOGIC;
35         TestEN : in STD_LOGIC;
         reset : in STD_LOGIC;
         Count : out STD_LOGIC_VECTOR (N-1 downto 0));
end Counter;

40 architecture Behavioral of Counter is
    --COMPONENT DECLARATIONS

    component nBitAdder is
45         generic (n : integer := 32);
        port(
            A,B : in std_logic_vector(N-1 downto 0);
            Y : out std_logic_vector(N-1 downto 0);
            CB : out std_logic
50         );
    end component;

    component nBitRegister_32 is
        generic (n : integer := 32);
55         Port (
            nBitIn : in std_logic_vector(n-1 downto 0); -- n bits to store in the
            register
            WE : in std_logic; -- Active high write enable
            Reset : in std_logic; -- Async reset , disabled when low
            clk : in std_logic;
60             Y : out std_logic_vector(n-1 downto 0) -- 1 output , n bits wide
        );
    end component;

    --SIGNAL DECLARATIONS
65     signal RegOut, RegIn : STD_LOGIC_VECTOR(N-1 downto 0);
    signal cout : STD_LOGIC;

begin

70     --16 bit adder, always adds 1 to value in register
    ADDER0 : nBitAdder
        generic map(N => 32)
        port map(A => RegOut, B => "00000000000000000000000000000001", Y => RegIn,
        CB => cout);

75     --Holds the current counter value
    REG0 : nBitRegister_32
        generic map(N => 32)
        port map(nBitIn => RegIn, clk => clk, WE => TestEN, Reset => reset, Y =>
        RegOut);

80     --Map output
    Count <= RegOut;

end Behavioral;

```

Listing 13: BIST tb VHDL

```

0  ---Company      : RIT
   ---Author       : Brandon Key
   ---Created      : 03/29/2018
   ---
5  ---Project Name : Lab 5
   ---File         : BIST_tb.vhd
   ---
   ---Entity       : BIST_tb
   ---Architecture : behav
10  ---
   ---Tool Version : VHDL '93
   ---Description  : BIST
   ---
15  LIBRARY ieee;
   USE ieee.std_logic_1164.ALL;
   use ieee.numeric_std.all;

   --- Uncomment the following library declaration if using
   --- arithmetic functions with Signed or Unsigned values
20  ---USE ieee.numeric_std.ALL;

   ENTITY BIST_tb IS
   END BIST_tb;

25  ARCHITECTURE behavior OF BIST_tb IS

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

       COMPONENT Wrapper
30      PORT(
         mul_input : IN  std_logic_vector(7 downto 0);
         start_BIST : IN  std_logic;
         disp_Sig   : IN  std_logic;
         clk        : IN  std_logic;
35         rst_n     : IN  std_logic;
         unused_anode : OUT std_logic;
         hund_anode  : OUT std_logic;
         tens_anode   : OUT std_logic;
         ones_anode   : OUT std_logic;
40         CAn : OUT std_logic;
         CBn : OUT std_logic;
         CCn : OUT std_logic;
         CDn : OUT std_logic;
         CEn : OUT std_logic;
45         CFn : OUT std_logic;
         CGn : OUT std_logic;
         is_Testing : OUT std_logic;
         mul_disp   : OUT std_logic_vector(7 downto 0);
         sig_disp   : OUT std_logic_vector(7 downto 0)
50     );
       END COMPONENT;

       ---Inputs
55  signal mul_input : std_logic_vector(7 downto 0) := (others => '0');
  signal start_BIST : std_logic := '0';

```

```

signal disp_Sig : std_logic := '0';
signal clk : std_logic := '0';
signal rst_n : std_logic := '0';

--Outputs
signal unused_anode : std_logic;
signal hund_anode : std_logic;
signal tens_anode : std_logic;
signal ones_anode : std_logic;
signal CAn : std_logic;
signal CBn : std_logic;
signal CCn : std_logic;
signal CDn : std_logic;
signal CEn : std_logic;
signal CFn : std_logic;
signal CGn : std_logic;
signal is_Testing : std_logic;
signal mul_disp : std_logic_vector(7 downto 0);
signal sig_disp : std_logic_vector(7 downto 0);

-- Clock period definitions
constant clk_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)
 uut: Wrapper PORT MAP (
    mul_input => mul_input,
    start_BIST => start_BIST,
    disp_Sig => disp_Sig,
    clk => clk,
    rst_n => rst_n,
    unused_anode => unused_anode,
    hund_anode => hund_anode,
    tens_anode => tens_anode,
    ones_anode => ones_anode,
    CAn => CAn,
    CBn => CBn,
    CCn => CCn,
    CDn => CDn,
    CEn => CEn,
    CFn => CFn,
    CGn => CGn,
    is_Testing => is_Testing,
    mul_disp => mul_disp,
    sig_disp => sig_disp
  );

-- Clock process definitions
clk_process : process
begin
    clk <= '0';
    wait for clk_period/2;
    clk <= '1';
    wait for clk_period/2;
end process;

-- Stimulus process

```

```

stim_proc: process
begin
  -- hold reset state for 100 ns.
  rst_n <= '0';
120   wait for 100 ns;
  rst_n <= '1';
  wait for clk_period*3;
  -- insert stimulus here

125   start_BIST <= '1';
  wait for clk_period*10;
  start_BIST <= '0';
  wait for clk_period*30;

130   for i in 0 to 5 loop
    start_BIST <= '1';
    mul_input <= std_logic_vector(to_unsigned(i, mul_input'length));
    wait for clk_period*2;
    start_BIST <= '0';
135    wait for clk_period*2;
  end loop;

  wait for clk_period*270;

140   start_BIST <= '1';
  wait for clk_period*300;
  start_BIST <= '0';
  wait for clk_period*10;

145   wait;
end process;

END;

```

Listing 14: Multiplier VHDL

```

0  -- Company:
  -- Engineer:
  --
  -- Create Date:    14:56:40 03/15/2017
5  -- Design Name:
  -- Module Name:    Multiplier - Behavioral
  -- Project Name:
  -- Target Devices:
  -- Tool versions:
10 -- Description:
  --
  -- Dependencies:
  --
  -- Revision:
15 -- Revision 0.01 - File Created
  -- Additional Comments:
  --

library IEEE;
20 use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

```



```

25  -- Uncomment the following library declaration if using
    -- arithmetic functions with Signed or Unsigned values
    --use IEEE.NUMERIC.STD.ALL;

    -- Uncomment the following library declaration if instantiating
    -- any Xilinx primitives in this code.
    --library UNISIM;
30  --use UNISIM.VComponents.all;

entity Multiplier is
    generic( N : integer :=32);
    Port ( A : in  STD_LOGIC_VECTOR ((N/2)-1 downto 0);
35      B : in  STD_LOGIC_VECTOR ((N/2)-1 downto 0);
        Product : out  STD_LOGIC_VECTOR (N-1 downto 0));
end Multiplier;

architecture Behavioral of Multiplier is
40  --COMPONENT DECLARATIONS
    component ANDADD is
        Port ( A : in  STD_LOGIC;
              B : in  STD_LOGIC;
              D : in  STD_LOGIC;
45      Cin : in  STD_LOGIC;
          Sum : out  STD_LOGIC;
          Cout : out  STD_LOGIC);

    end component;

50  component AND2 is
        Port ( A : in  STD_LOGIC;
              B : in  STD_LOGIC;
              F : out  STD_LOGIC);
    end component;
55  --SIGNAL DECLARATIONS
    signal Cin : STD_LOGIC := '0';
    signal Cout: STD_LOGIC := '0';
    signal F : STD_LOGIC_VECTOR(N*N downto 0);--interconnections within multiplier
    between adders and AND gates
    signal CoutArray : STD_LOGIC_VECTOR(N*N downto 0);--Signals used by the Cout
    between the 1 bit full adders
60 begin

    forrow : for i in integer range 0 to (N/2)-1 generate --Loop down the levels
        if0 : if i = 0 generate --top level, just AND gates
65      ANDOGEN : for j in integer range 0 to (N/2)-1 generate
          AND0 : AND2
              port map(A=>A(i), B=>B(j), F=>F(j));
          end generate ANDOGEN;
          Product(0)<=F(0);--Assign first product
70      end generate if0;
        ifn0 : if i>0 generate--everything else
            forcol : for j in integer range 0 to (N/2)-1 generate

                --Generate the first full adder / and gate combo of the row
75      GEN0 : if j = 0 generate
          ANDADD0 : ANDADD
              port map(A=>A(i), B=>B(j), D=>F((N/2)*(i-1)+(j+1)), Cin=>
Cin, Cout=>CoutArray((N/2)*i+j), Sum=>F((N/2)*i+j));--Tons of 2D arrays as 1D
arrays

```

```

80      —D=>The sum of the full adder in the previous row and
next column (i-1) (j+1).
      —Cout=>Corresponding coordinate in CoutArray for this
full adder [i,j].
      —Sum=> Corresponding coordinate in F array for this
full adder [i,j].

      —Assign the sum of the first adder of the row to its
respective product bit
      Product(i)<=F((N/2)*i+j);
      end generate GEN0;

85      —Last full adder of the first row, D has to be 0
GENIN : if i=1 AND j=((N/2)-1) generate
      ANDADDN : ANDADD
      port map(A=>A(i),B=>B(j),D=>Cin,Cin=>CoutArray((N/2)*i+(
j-1)),Cout=>CoutArray((N/2)*i+j),Sum=>F((N/2)*i+j));
      —D=>Cin, which is equal to 0
      —Cin=>Cout of previous full adder in same row (j-1)
      —Cout=>Corresponding coordinate in CoutArray for
this full adder [i,j].
      —Sum=> Corresponding coordinate in F array for this
full adder [i,j].
      end generate GENIN;

95      —Generate the last full adder / and gate combo of the row
GENN : if i/=1 AND j= ((N/2)-1) generate
      ANDADDN : ANDADD
      port map(A=>A(i),B=>B(j),D=>CoutArray((N/2)*(i-1)+j),Cin
=>CoutArray((N/2)*i+(j-1)),Cout=>CoutArray((N/2)*i+j),Sum=>F((N/2)*i+j)); —Map
the last full adder in line
      —D=>The Cout of the last full adder of the previous
row (i-1)
      —Cin=>Cout of previous full adder in same row (j-1)
      —Cout=>Corresponding coordinate in CoutArray for
this full adder [i,j].
      —Sum=> Corresponding coordinate in F array for this
full adder [i,j].

105      —Assign Sum and Cout to Product of the last row's last full
adder
      GENNI : if i=(N/2)-1 generate
      Product(N-1)<=CoutArray((N/2)*i+j); —Product bit from
Cout
      Product(i+j)<=F((N/2)*i+j); —Product bit from Sum
      end generate GENNI;
      end generate GENN;

110      —Generate all the other full adders / AND gate combos
GENX : if j/=((N/2)-1) AND j>0 generate
      ANDADDX : ANDADD
      port map(A=>A(i), B=>B(j), D=>F((N/2)*(i-1)+(j+1)), Cin
=>CoutArray((N/2)*i+(j-1)), Cout=>CoutArray((N/2)*i+j),Sum=>F((N/2)*i+j));
      —D=>The Cout of the last full adder of the previous
row (i-1)
      —Cin=>Cout of previous full adder in same row (j-1)
      —Cout=>Corresponding coordinate in CoutArray for

```

```

125   this full adder [i,j].
      --Sum=> Corresponding coordinate in F array for this
      full adder [i,j].

      --Assign Product bits the sum bits of the last row of full
adders
      GENXI : if i=((N/2)-1) generate
130         Product(i+j)<=F((N/2)*i+j);--Assign the sum to the
      respective product bit
      end generate GENXI;

      end generate GENX;

      end generate forcol;
      end generate ifn0;
      end generate forrow;

135 end Behavioral;

```

Listing 15: LFSR 32 4 VHDL

```

0  --Company      : RIT
   --Author       : Brandon Key
   --Created      : 03/08/2018
   --
5  --Project Name : Lab 5
   --File         : LFSR_32_4.vhd
   --
   --Entity       : LFSR_32_4
   --Architecture : behav
10 --
   --Tool Version : VHDL '93
   --Description  : LFSR_32_4 8 bit output, 4 tap LFSR.
   --
15 library IEEE;
   use IEEE.STD_LOGIC_1164.ALL;
   use IEEE.numeric_std.all;
   use IEEE.STD_LOGIC_UNSIGNED.ALL;

20 entity LFSR_32_4 is
   generic (N : integer := 32);
   port(
       clk      : in std_logic;
       rst_n    : in std_logic;
25      en      : in std_logic;
       bit_pattern : out std_logic_vector(N-1 downto 0)
   );
   end LFSR_32_4;

30 architecture behav of LFSR_32_4 is

   signal internal_reg : std_logic_vector(N-1 downto 0);

   constant SEED : std_logic_vector(N-1 downto 0) := x"12345678";
35

```

```

begin

    bit_pattern <= internal_reg;

    --update the state to the next_state
40  the_proc : process (clk, rst_n) begin
        if rst_n = '0' then
            internal_reg <= SEED;
        elsif rising_edge(clk) then
45            if en = '1' then
                --taps at 32,20,26,25
                internal_reg(0) <= internal_reg(1);
                internal_reg(1) <= internal_reg(2);
                internal_reg(2) <= internal_reg(3);
                internal_reg(3) <= internal_reg(4);
                internal_reg(4) <= internal_reg(5);
                internal_reg(5) <= internal_reg(6);
                internal_reg(6) <= internal_reg(7);
                internal_reg(7) <= internal_reg(8);
                internal_reg(8) <= internal_reg(9);
                internal_reg(9) <= internal_reg(10);
                internal_reg(10) <= internal_reg(11);
                internal_reg(11) <= internal_reg(12);
                internal_reg(12) <= internal_reg(13);
                internal_reg(13) <= internal_reg(14);
                internal_reg(14) <= internal_reg(15);
                internal_reg(15) <= internal_reg(16);
                internal_reg(16) <= internal_reg(17);
                internal_reg(17) <= internal_reg(18);
                internal_reg(18) <= internal_reg(19);
                internal_reg(19) <= internal_reg(20) xor internal_reg(0);
                internal_reg(20) <= internal_reg(21);
                internal_reg(21) <= internal_reg(22);
                internal_reg(22) <= internal_reg(23);
                internal_reg(23) <= internal_reg(24);
                internal_reg(24) <= internal_reg(25) xor internal_reg(0);
                internal_reg(25) <= internal_reg(26) xor internal_reg(0);
                internal_reg(26) <= internal_reg(27);
                internal_reg(27) <= internal_reg(28);
                internal_reg(28) <= internal_reg(29);
                internal_reg(29) <= internal_reg(30);
                internal_reg(30) <= internal_reg(31);
                internal_reg(31) <= internal_reg(0);

                end if;
            end if;
        end process the_proc;

85 end  behav;

```

Listing 16: LFSR 8 4 VHDL

```

0  -----
--Company      : RIT
--Author       : Brandon Key
--Created      : 03/08/2018
--
5  --Project Name : Lab 5

```

```

--File      : LFSR_8_4.vhd
--
--Entity     : LFSR_8_4
--Architecture : behav
10 --
--Tool Version : VHDL '93
--Description  : LFSR_8_4 8 bit output, 4 tap LFSR.
-----

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

20 entity LFSR_8_4 is
   port(
       clk    : in std_logic;
       rst_n  : in std_logic;
       en     : in std_logic;
       bit_pattern : out std_logic_vector(7 downto 0)
   );
end LFSR_8_4;

architecture behav of LFSR_8_4 is
30     signal internal_reg : std_logic_vector(7 downto 0);

     constant SEED : std_logic_vector(7 downto 0) := x"6A";

35     begin

         bit_pattern <= internal_reg;

         --update the state to the next-state
40     the_proc : process (clk, rst_n) begin
         if rst_n = '0' then
             internal_reg <= SEED;
         elsif rising_edge(clk) then
             if en = '1' then
45                 --taps at 7,5,4,3
                 internal_reg(0) <= internal_reg(1);
                 internal_reg(1) <= internal_reg(2);
                 internal_reg(2) <= internal_reg(3);
                 internal_reg(3) <= internal_reg(4) xor internal_reg(0);
50                 internal_reg(4) <= internal_reg(5) xor internal_reg(0);
                 internal_reg(5) <= internal_reg(6) xor internal_reg(0);
                 internal_reg(6) <= internal_reg(7);
                 internal_reg(7) <= internal_reg(0);

                 end if;
55             end if;
         end process the_proc;

end behav;

```

Listing 17: MAC VHDL

```

0 --
-- Company:

```

```

-- Engineer:
--
-- Create Date:      14:56:40 03/15/2017
5 -- Design Name:
-- Module Name:      Multiplier -- Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
10 -- Description:
--
-- Dependencies:
--
-- Revision:
15 -- Revision 0.01 -- File Created
-- Additional Comments:
--

```

---

```

library IEEE;
20 use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
25 --use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
30 --use UNISIM.VComponents.all;

entity MAC is
    generic( N : integer := 32);
    Port ( A : in  STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
35         B : in  STD_LOGIC_VECTOR ((N/2) - 1  downto 0);
        clk : in  STD_LOGIC;
        WE : in  STD_LOGIC;
        reset : in  STD_LOGIC;
        RegOut : out  STD_LOGIC_VECTOR (N-1  downto 0));
40 end MAC;

architecture Behavioral of MAC is
    --COMPONENT DECLARATIONS
    component Multiplier is
45         generic( N : integer := 32);
        Port ( A : in  STD_LOGIC_VECTOR ((N/2)-1  downto 0);
            B : in  STD_LOGIC_VECTOR ((N/2)-1  downto 0);
            Product : out  STD_LOGIC_VECTOR (N-1  downto 0));
        end component;

50     component nBitAdder is
        generic (n : integer := 32);
        port(
            A,B : in  std_logic_vector(N-1  downto 0);
55             Y : out std_logic_vector(N-1  downto 0);
            CB : out std_logic
        );
        end component;

60     component nBitRegister is

```

```

        generic (n : integer := 32);
        Port (
            nBitIn : in std_logic_vector(n-1 downto 0); -- n bits to store in the
register
            WE      : in std_logic; -- Active high write enable
            Reset   : in std_logic; -- Async reset , disabled when low
            clk     : in std_logic;
            Y : out std_logic_vector(n-1 downto 0) -- 1 output , n bits wide
        );
    end component;

    component nBitRegister_32 is
        generic (n : integer := 32);
        Port (
            nBitIn : in std_logic_vector(n-1 downto 0); -- n bits to store in the
register
            WE      : in std_logic; -- Active high write enable
            Reset   : in std_logic; -- Async reset , disabled when low
            clk     : in std_logic;
            Y : out std_logic_vector(n-1 downto 0) -- 1 output , n bits wide
        );
    end component;

    component nBitRegister_16 is
        generic (n : integer := 16);
        Port (
            nBitIn : in std_logic_vector(n-1 downto 0); -- n bits to store in the
register
            WE      : in std_logic; -- Active high write enable
            Reset   : in std_logic; -- Async reset , disabled when low
            clk     : in std_logic;
            Y : out std_logic_vector(n-1 downto 0) -- 1 output , n bits wide
        );
    end component;

    --SIGNAL DECLARATIONS
    signal MultA, MultB : STD_LOGIC_VECTOR((N/2)-1 downto 0);
    signal Product : STD_LOGIC_VECTOR(N-1 downto 0);
    signal adderA, adderB, adderOut : STD_LOGIC_VECTOR(N-1 downto 0);
    signal cout : STD_LOGIC;

begin

    RegMultInA : nBitRegister_16
        generic map( N => 16)
        port map(nBitIn => A,
            WE => '1', clk => clk, Reset => reset,
            Y => MultA
        );

    RegMultInB : nBitRegister_16
        generic map( N => 16)
        port map(nBitIn => B,
            WE => '1', clk => clk, Reset => reset,
            Y => MultB
        );

    MULT1 : Multiplier
        generic map( N => 32)

```

```

    port map(A => MultA, B => MultB, Product => Product);

    RegMultOut : nBitRegister_32
120     generic map( N => 32)
    port map(nBitIn => Product, WE => '1', Reset => reset, clk => clk, Y =>
    adderB);

    BigBoyReg : nBitRegister_32
125     generic map( N => 32)
    port map(nBitIn => adderOut, WE => WE, Reset => reset, clk => clk, Y =>
    adderA);

    ADD1 : nBitAdder
    generic map ( N => 32)
    port map( A => adderA, B => adderB, Y => adderOut, CB => cout);
130

    RegOut <= adderA;
end Behavioral;

```

Listing 18: MISR 32 4 VHDL

```

0  -----
  --Company       : RIT
  --Author        : Brandon Key
  --Created       : 03/08/2018
  --
5  --Project Name : Lab 5
  --File          : MISR_32_4.vhd
  --
  --Entity        : MISR_32_4
  --Architecture  : behav
10  --
  --Tool Version  : VHDL '93
  --Description   : MISR_32_4 32 bit output, 4 tap MISR.
  -----

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

20 entity MISR_32_4 is
    generic (N : integer := 32);
    port(
        MISR_in : in std_logic_vector(N-1 downto 0);
        clk     : in std_logic;
25        rst_n  : in std_logic;
        en      : in std_logic;
        MISR_out : out std_logic_vector(N-1 downto 0)
    );
end MISR_32_4;

30 architecture behav of MISR_32_4 is

    signal internal_reg : std_logic_vector(N-1 downto 0);

35    constant SEED : std_logic_vector(N-1 downto 0) := x"12345678";

    begin

```



```

MISR_out <= internal_reg;

--update the state to the next_state
the_proc : process (clk, rst_n) begin
    if rst_n = '0' then
        internal_reg <= SEED;
    elsif rising_edge(clk) then
        if en = '1' then
            --taps at 32,20,26,25
            internal_reg(0) <= internal_reg(1) xor MISR_in(0);
            internal_reg(1) <= internal_reg(2) xor MISR_in(1);
            internal_reg(2) <= internal_reg(3) xor MISR_in(2);
            internal_reg(3) <= internal_reg(4) xor MISR_in(3);
            internal_reg(4) <= internal_reg(5) xor MISR_in(4);
            internal_reg(5) <= internal_reg(6) xor MISR_in(5);
            internal_reg(6) <= internal_reg(7) xor MISR_in(6);
            internal_reg(7) <= internal_reg(8) xor MISR_in(7);
            internal_reg(8) <= internal_reg(9) xor MISR_in(8);
            internal_reg(9) <= internal_reg(10) xor MISR_in(9);
            internal_reg(10) <= internal_reg(11) xor MISR_in(10);
            internal_reg(11) <= internal_reg(12) xor MISR_in(11);
            internal_reg(12) <= internal_reg(13) xor MISR_in(12);
            internal_reg(13) <= internal_reg(14) xor MISR_in(13);
            internal_reg(14) <= internal_reg(15) xor MISR_in(14);
            internal_reg(15) <= internal_reg(16) xor MISR_in(15);
            internal_reg(16) <= internal_reg(17) xor MISR_in(16);
            internal_reg(17) <= internal_reg(18) xor MISR_in(17);
            internal_reg(18) <= internal_reg(19) xor MISR_in(18);
            internal_reg(19) <= internal_reg(20) xor MISR_in(19) xor
internal_reg(0);
            internal_reg(20) <= internal_reg(21) xor MISR_in(20);
            internal_reg(21) <= internal_reg(22) xor MISR_in(21);
            internal_reg(22) <= internal_reg(23) xor MISR_in(22);
            internal_reg(23) <= internal_reg(24) xor MISR_in(23);
            internal_reg(24) <= internal_reg(25) xor MISR_in(24) xor
internal_reg(0);
            internal_reg(25) <= internal_reg(26) xor MISR_in(25) xor
internal_reg(0);
            internal_reg(26) <= internal_reg(27) xor MISR_in(26);
            internal_reg(27) <= internal_reg(28) xor MISR_in(27);
            internal_reg(28) <= internal_reg(29) xor MISR_in(28);
            internal_reg(29) <= internal_reg(30) xor MISR_in(29);
            internal_reg(30) <= internal_reg(31) xor MISR_in(30);
            internal_reg(31) <= internal_reg(0) xor MISR_in(31);
        end if;
    end if;
end process the_proc;

end behav;

```

Listing 19: nBitRegister tb VHDL

```

-- Company:
-- Engineer:
--
-- Create Date:    16:49:10 02/08/2018

```

```

5  -- Design Name:
-- Module Name:    /home/ise/DSDII/Lab/Lab2/Project/lab2/nBitRegister_tb.vhd
-- Project Name:   lab2
-- Target Device:
-- Tool versions:
10 -- Description:
--
-- VHDL Test Bench Created by ISE for module: nBitRegister
--
-- Dependencies:
15 --
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
20 -- 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
25 -- simulation model.

```

---

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.numeric_std.all;
30
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

35 ENTITY nBitRegister_tb IS
END nBitRegister_tb;

ARCHITECTURE behavior OF nBitRegister_tb IS

40     constant N : integer := 4;

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

    COMPONENT nBitRegister
45     generic (n : integer := 32);
    PORT(
        nBitIn : in std_logic_vector(n-1 downto 0); -- n bits to store in the
        register
        WE      : in std_logic; -- Active high write enable
        Reset   : in std_logic; -- Async reset, disabled when low
50     clk      : in std_logic;
        Y : out std_logic_vector(n-1 downto 0) -- 1 output, n bits wide
    );
    END COMPONENT;

55
--Inputs
signal nBitIn : std_logic_vector(n-1 downto 0) := (others => '0');
signal WE : std_logic := '0';
signal Reset : std_logic := '0';
60 signal clk : std_logic := '0';

--Outputs

```

```

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

-- Clock period definitions
constant clk_period : time := 100 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)
 uut: nBitRegister
  generic map (N => N)
  PORT MAP (
    nBitIn => nBitIn ,
    WE => WE,
    Reset => Reset ,
    clk => clk ,
    Y => Y
  );

-- Clock process definitions
 clk_process : process
 begin
   clk <= '0';
   wait for clk_period/2;
   clk <= '1';
   wait for clk_period/2;
 end process;

-- Stimulus process
 stim_proc: process
 begin
   -- hold reset state for 100 ns.
   Reset <= '0';
   wait for (1*clk_period + 15 ns);
   Reset <= '1';
   wait for clk_period*1;
   --Setup Complete, Time to load

--Load each register 1 by 1
   WE <= '1';
   wait for clk_period;
   nBitIn <= x"D";
   wait for clk_period;
   WE <= '0';
   nBitIn <= x"E";
   wait for clk_period;

   WE <= '1';
   wait for clk_period;
   for i in 0 to 15 loop
     nBitIn <= std_logic_vector(to_unsigned(i, nBitIn'length));
     wait for clk_period*1;
   end loop;

   wait;
 end process;

END;

```

Listing 20: nBitAdder VHDL

```

0  -----
--Company      : RIT
--Author       : Brandon Key
--Created      : 02/18/2018
--
5  --Project Name : Lab 3
--File         : nBitAdder.vhd
--
--Entity       : nBitAdder
--Architecture : struct
10 --
--Tool Version : VHDL '93
--Description  : Entity and structural description of an adder subtractor
--              : SEL = 0 : A+B = Y
--              : SEL = 1 : A-B = Y
15 -----

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

entity nBitAdder is
    generic (n : integer := 32);
    port(
25     A,B : in  std_logic_vector(n-1 downto 0);
        Y  : out std_logic_vector(n-1 downto 0);
        CB : out std_logic
    );
end nBitAdder;
30

architecture struct of nBitAdder is

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

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

begin

45    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(
50                 A => A(i),
                 B => B(i),
                 Cin => '0',
                 Sum => Y(i),
                 Cout => c_array(i)
55             );
        end generate i_first;

```

```

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

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

    80      end generate generate_adders;

        CB <= c_array(n-1);

    end struct;

```

Listing 21: nBitRegister 16 VHDL

```

0  -----
--Company      : RIT
--Author       : Brandon Key
--Created      : 2/8/2017
--
5  --Project Name : Lab 2
--File         : nBitRegister_16.vhd
--
--Entity       : nBitRegister_16
--Architecture : struct
10 --Revision    :
--Rev 0.01     : 2/8/2017
--
--Tool Version : VHDL '93
--Description  : Entity and behavioral description of an n-bit register
15 --
--Notes        :
-----

library ieee;
20 use ieee.std_logic_1164.all;

entity nBitRegister_16 is
    generic (n : integer := 16);
25     Port (
        nBitIn : in std_logic_vector(n-1 downto 0); -- n bits to store in the
        register

```

```

WE      : in std_logic; -- Active high write enable
Reset   : in std_logic; -- Async reset , disabled when low
clk      : in std_logic;
30      Y : out std_logic_vector(n-1 downto 0) -- 1 output , n bits wide
    );
end nBitRegister_16;

35 architecture behav of nBitRegister_16 is

begin

40     output_proc : process (clk , Reset) begin

        if Reset = '0' then
            Y <= (others => '0');
        elsif clk'event and clk = '1' then
45             if WE = '1' then
                 Y <= nBitIn;
            end if;
        end if;

50     end process output_proc;

end behav;

```

Listing 22: MISR 8 4 VHDL

```

0  -----
--Company      : RIT
--Author       : Brandon Key
--Created      : 03/08/2018
--
5  --Project Name : Lab 5
--File         : MISR_8_4.vhd
--
--Entity       : MISR_8_4
--Architecture : behav
10 --
--Tool Version : VHDL '93
--Description  : MISR_8_4 8 bit output, 4 tap MISR.
-----

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

20 entity MISR_8_4 is
    port(
        MISR_in : in std_logic_vector(7 downto 0);
        clk      : in std_logic;
        rst_n    : in std_logic;
25        en      : in std_logic;
        MISR_out : out std_logic_vector(7 downto 0)
    );

```

```

end MISR_8_4;

30 architecture behav of MISR_8_4 is

    signal internal_reg : std_logic_vector(7 downto 0);

    constant SEED : std_logic_vector(7 downto 0) := x"6A";

35 begin

    MISR_out <= internal_reg;

    --update the state to the next state
    the_proc : process (clk, rst_n) begin
        if rst_n = '0' then
            internal_reg <= SEED;
        elsif rising_edge(clk) then
45             if en = '1' then
                --taps at 7,5,4,3
                internal_reg(0) <= internal_reg(1) xor MISR_in(0);
                internal_reg(1) <= internal_reg(2) xor MISR_in(1);
                internal_reg(2) <= internal_reg(3) xor MISR_in(2);
50                 internal_reg(3) <= internal_reg(4) xor MISR_in(3) xor
                    internal_reg(0);
                internal_reg(4) <= internal_reg(5) xor MISR_in(4) xor
                    internal_reg(0);
                internal_reg(5) <= internal_reg(6) xor MISR_in(5) xor
                    internal_reg(0);
                internal_reg(6) <= internal_reg(7) xor MISR_in(6);
                internal_reg(7) <= internal_reg(0) xor MISR_in(7);
55             end if;
        end if;
    end process the_proc;

60 end behav;

```

Listing 23: nBitMux 2to1 VHDL

```

0  -----
   --Company       : RIT
   --Author        : Brandon Key
   --Created       : 3/29/2018
   --
   --Project Name  : Lab 5
   --File          : nBitMux_2to1.vhd
   --
   --Entity        : nBitMux_2to1
   --Architecture  : Dataflow
10  -----
   --Tool Version  : VHDL '93
   --Description   : Arbitrary width 2 to 1 mux
   -----

15 library IEEE;
   use IEEE.STD_LOGIC_1164.ALL;

   entity nBitMux_2to1 is
       generic (n : integer := 16);

```

```

20     port (
        A,B : in  std_logic_vector(n-1  downto  0);
        sel : in  std_logic;
        Y   : out std_logic_vector(n-1  downto  0)
    );
25 end  nBitMux_2to1;

architecture Dataflow of nBitMux_2to1 is
begin

30     --update the state to the next_state
    the_proc : process (sel, A, B) begin
        case sel is
            when '0' =>
                Y <= A;
35             when others =>
                Y <= B;
            end case;
        end process the_proc;
40 end  Dataflow;

```

Listing 24: ANDADD VHDL

```

0  ---
--- Company:
--- Engineer:
---
--- Create Date:    15:25:28 03/15/2017
5  --- Design Name:
--- Module Name:    ANDADD - Behavioral
--- Project Name:
--- Target Devices:
--- Tool versions:
10 --- Description:
---
--- Dependencies:
---
--- Revision:
15 --- Revision 0.01 - File Created
--- Additional Comments:
---
---
library IEEE;
20 use IEEE.STD_LOGIC_1164.ALL;

--- Uncomment the following library declaration if using
--- arithmetic functions with Signed or Unsigned values
---use IEEE.NUMERIC_STD.ALL;
25 ---
--- Uncomment the following library declaration if instantiating
--- any Xilinx primitives in this code.
---library UNISIM;
---use UNISIM.VComponents.all;
30
entity ANDADD is
    Port ( A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          D : in  STD_LOGIC;

```



```

35      Cin : in  STD_LOGIC;
        Sum : out STD_LOGIC;
        Cout : out STD_LOGIC);
end ANDADD;

40 architecture Behavioral of ANDADD is

    --Component Declarations
    component AND2 is
        Port ( A : in  STD_LOGIC;
45          B : in  STD_LOGIC;
              F : out STD_LOGIC);
    end component;

    component FA_1bit is
50      Port ( A : in  STD_LOGIC;
              B : in  STD_LOGIC;
              Cin : in  STD_LOGIC;
              S : out STD_LOGIC;
              Cout : out STD_LOGIC);
55    end component;

    --Signal Assignments
    signal F : STD_LOGIC;
begin
60    AND0 : AND2
        port map(A=>A, B=>B, F=>F);

    FA : FA_1bit
65    port map(A=>F, B=>D, Cin=>Cin, S=>Sum, Cout=>Cout);

end Behavioral;

```

Listing 25: Ripple Carry FA VHDL

```

0  -- Company:
    -- Engineer:
    --
    -- Create Date:      08:27:52 03/02/2017
5  -- Design Name:
    -- Module Name:      Ripple_Carry_FA - Behavioral
    -- Project Name:
    -- Target Devices:
    -- Tool versions:
10 -- Description:
    --
    -- Dependencies:
    --
    -- Revision:
15 -- Revision 0.01 - File Created
    -- Additional Comments:
    --

20 library IEEE;
    use IEEE.STD_LOGIC_1164.ALL;

```

```

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC.STD.ALL;
25
-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
30
entity Ripple_Carry_FA is
    generic(N : integer :=16);--Number of bits in A and B
    Port ( A : in  STD_LOGIC_VECTOR (N-1 downto 0);
          B : in  STD_LOGIC_VECTOR (N-1 downto 0);
35          Cin : in STD_LOGIC;
          Sum : out STD_LOGIC_VECTOR (N-1 downto 0);
          Cout : out STD_LOGIC);
end Ripple_Carry_FA;

40 architecture Behavioral of Ripple_Carry_FA is
    --Interim signal used for carry ins and outs of the 1 bit full adders
    --The MSB of C is carry out
    signal C : std_logic_vector(N downto 1);

45    component FA_1bit is
        Port ( A : in  STD_LOGIC;
              B : in  STD_LOGIC;
              Cin : in  STD_LOGIC;
              S : out  STD_LOGIC;
50              Cout : out  STD_LOGIC);
    end component;

begin

55    GEN_ADD : for i in N-1 downto 0 generate
        --Generate the first full adder, which is special because it takes Cin
        FA0_GEN : if(i=0) generate
            FA0: FA_1bit
                port map(A=>A(i),B=>B(i), Cin=>Cin, Cout=>C(1),S=>Sum(i));
60        end generate FA0_GEN;

        --Generate the other adders, the last bit of C is carry out
        FAX_GEN : if(i>0) generate
            FAX: FA_1bit
65                port map(A=>A(i), B=>B(i), Cin=>C(i), Cout=>C(i+1), S=>Sum(i));
            end generate FAX_GEN;

        end generate GEN_ADD;
70
end Behavioral;

```

Listing 26: nBitRegister 32 VHDL

```

0
--Company      : RIT
--Author       : Brandon Key
--Created      : 2/8/2017
--

```

```

5  --Project Name : Lab 2
   --File       : nBitRegister_32.vhd
   --
   --Entity      : nBitRegister_32
   --Architecture : struct
10  --Revision   :
   --Rev 0.01    : 2/8/2017
   --
   --Tool Version : VHDL '93
   --Description  : Entity and behavioral description of an n-bit register
15  --
   --Notes       :
   -----

library ieee;
20 use ieee.std_logic_1164.all;

entity nBitRegister_32 is
   generic (n : integer := 32);
25   Port (
       nBitIn : in std_logic_vector(n-1 downto 0); -- n bits to store in the
       register
       WE      : in std_logic; -- Active high write enable
       Reset   : in std_logic; -- Async reset , disabled when low
       clk     : in std_logic;
30       Y : out std_logic_vector(n-1 downto 0) -- 1 output , n bits wide
   );
end nBitRegister_32;

35 architecture behav of nBitRegister_32 is

begin

40   output_proc : process (clk , Reset) begin

       if Reset = '0' then
           Y <= (others => '0');
       elsif clk'event and clk = '1' then
45           if WE = '1' then
               Y <= nBitIn;
           end if;
       end if;

50   end process output_proc;

end behav;

```

Listing 27: ALU Wrapper VHDL

```

0  -----
   -- Company:
   -- Engineer:
   --
   -- Create Date:    13:41:10 03/18/2017

```

```

5  -- Design Name:
-- Module Name:      ALU_Wrapper - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
10 -- Description:
--
-- Dependencies:
--
-- Revision:
15 -- Revision 0.01 - File Created
-- Additional Comments:
--

```

---

```

library IEEE;
20 use IEEE.STD_LOGIC_1164.ALL;
use IEEE.MATHREAL.ALL;
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if using
25 -- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
30 --library UNISIM;
--use UNISIM.VComponents.all;

entity ALU_Wrapper is
    Port ( input1 : in  STD_LOGIC_VECTOR (15 downto 0);
35         input2 : in  STD_LOGIC_VECTOR (15 downto 0);
        control : in  STD_LOGIC_VECTOR (3  downto 0);
        output  : out STD_LOGIC_VECTOR (15 downto 0));
end ALU_Wrapper;

40 architecture Behavioral of ALU_Wrapper is
--Component Declarations
    component Multiplier is
        generic( N : integer :=16);
        Port ( A : in  STD_LOGIC_VECTOR ((N/2)-1 downto 0);
45             B : in  STD_LOGIC_VECTOR ((N/2)-1 downto 0);
            Product : out STD_LOGIC_VECTOR (N-1 downto 0));
    end component;

    component Logic_Unit is
50         generic( N : integer :=16);
        Port ( A : in  STD_LOGIC_VECTOR (N-1 downto 0);
            B : in  STD_LOGIC_VECTOR (N-1 downto 0);
            Control : in  STD_LOGIC_VECTOR (3  downto 0);
            output : out STD_LOGIC_VECTOR (N-1 downto 0));
55     end component;

    component Shifter is
        generic( N : integer:=16; Namnt : integer :=integer(ceil(log2(real(16)))));
60
        Port ( A : in  STD_LOGIC_VECTOR (N-1 downto 0);
            amnt : in  STD_LOGIC_VECTOR (Namnt-1 downto 0);
            Control : in  STD_LOGIC_VECTOR (3  downto 0);
            output : out STD_LOGIC_VECTOR (N-1 downto 0));

```

```

end component;

65
component Ripple_Carry_FA is
    generic(N : integer :=16);--Number of bits in A and B
    Port ( A : in  STD_LOGIC_VECTOR (N-1 downto 0);
          B : in  STD_LOGIC_VECTOR (N-1 downto 0);
70          Cin : in STD_LOGIC;
          Sum : out STD_LOGIC_VECTOR (N-1 downto 0);
          Cout : out STD_LOGIC);
end component;

75
component Subtractor is
    generic(N : integer :=16);
    Port ( A : in  STD_LOGIC_VECTOR (15 downto 0);
          B : in  STD_LOGIC_VECTOR (15 downto 0);
          Output : out STD_LOGIC_VECTOR (15 downto 0));
80
end component;

--Signal Declarations

    signal Product, Sum, ShiftOut, LogicOut, Difference : STD_LOGIC_VECTOR(15 downto
85      0);
    signal Cout : STD_LOGIC;
begin
    --map multiplier
    MULT : Multiplier
        generic map(N=>16)
90        port map(A=>input1(7 downto 0), B=>input2(7 downto 0), Product=>Product);

    --Map Logic Unit
    LOGIC : Logic_Unit
        generic map(N=>16)
95        port map( A=>input1, B=>input2, Control=>Control, output=>LogicOut);

    --Map Shifter
    SHIFT : Shifter
        generic map(N=>16, Namnt=>4)
100        port map(A=>input1, amnt=>input2(3 downto 0), Control=>Control, output=>
        ShiftOut);

    --Map Adder
    ADD : Ripple_Carry_FA
        generic map(N=>16)
105        port map(A=>input1, B=>input2, Cin=>'0', Cout=>Cout, Sum=>Sum);

    --Map Subtractor
    SUB : Subtractor
        generic map(N=>16)
110        port map(A=>input1, B=>input2, Output=>Difference);

    --End mapping
    -----

115
    ALUPROC : process(input1, input2, control) begin
        C1: case control is
            when "0100"=> output<=Sum;--ADD
            when "0101"=> output<=Difference;--SUB
            when "0110"=> output<=Product;--MUL
120            when "1100"=> output<=ShiftOut;--SLL

```

```

        when "1101" => output<=ShiftOut;--SRL
        when "1110" => output<=ShiftOut;--SRA
        when others=> output<=LogicOut;--OR,NOT,AND,XOR
    end case C1;
125 end process;
end Behavioral;

```

Listing 28: nBitAdderSubtractor 16Bit VHDL

```

0  -----
--Company      : RIT
--Author       : Brandon Key
--Created      : 02/18/2018
--
5  --Project Name : Lab 3
--File         : nBitAdderSubtractor_16Bit.vhd
--
--Entity       : nBitAdderSubtractor_16Bit
--Architecture : struct
10 --
--Tool Version : VHDL '93
--Description  : Entity and structural description of an adder subtractor
--              : SEL = 0 : A+B = Y
--              : SEL = 1 : A-B = Y
15 -----

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
20
entity nBitAdderSubtractor_16Bit is
    generic (n : integer := 16);
    port(
25         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;
30
architecture struct of nBitAdderSubtractor_16Bit is

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

    --Create an array to hold all of the carries
40    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);

45 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;

```

```

50  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
55         adder : full_adder port map(
                A => A(i),
                B => B_XOR_SEL(i),
                Cin => SEL,
                Sum => Y(i),
60                Cout => c_array(i)
            );
        end generate i_first;

        i_last : if i = (n-1) generate
65         --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),
70                Sum => Y(i),
                Cout => c_array(i)
            );
        end generate i_last;

75         --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),
80                Cin => c_array(i-1),
                Sum => Y(i),
                Cout => c_array(i)
            );
        end generate i_mid;

85     end generate generate_adders;

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

90 end struct;

```

Listing 29: ALU TESTBENCH VHDL

```

0  ---
--- Company:
--- Engineer:
---
--- Create Date:    10:35:10 03/19/2017
5  --- Design Name:
--- Module Name:    G:/DSD2/Lab3/Xilinx/Lab3/ALU_TESTBENCH.vhd
--- Project Name:   Lab3
--- Target Device:
--- Tool versions:
10 --- Description:
---
--- VHDL Test Bench Created by ISE for module: ALU_Wrapper
---
--- Dependencies:

```

```

15  --
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
20  -- 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
25  -- simulation model.

```

---

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

30  -- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY ALU_TESTBENCH IS
35  END ALU_TESTBENCH;

ARCHITECTURE behavior OF ALU_TESTBENCH IS

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

40
    COMPONENT ALU_Wrapper
    PORT(
        input1 : IN    std_logic_vector(15 downto 0);
        input2 : IN    std_logic_vector(15 downto 0);
45        control : IN   std_logic_vector(3  downto 0);
        output  : OUT   std_logic_vector(15 downto 0)
    );
    END COMPONENT;

50
    --Inputs
    signal input1 : std_logic_vector(15 downto 0) := (others => '0');
    signal input2 : std_logic_vector(15 downto 0) := (others => '0');
    signal control : std_logic_vector(3  downto 0) := (others => '0');
55
    --Outputs
    signal output : std_logic_vector(15 downto 0);
    -- No clocks detected in port list. Replace <clock> below with
    -- appropriate port name

60

BEGIN

65
    -- Instantiate the Unit Under Test (UUT)
    uut: ALU_Wrapper PORT MAP (
        input1 => input1,
        input2 => input2,
        control => control,
70        output => output
    );

    --0100 ADD

```



```

75  --0101 SUB
    --0110 MUL
    --1000 OR
    --1001 NOT
    --1010 AND
    --1011 XOR
80  --1100 SLL
    --1101 SRL
    --1110 SRA

    -- Stimulus process
85  stim_proc: process
    begin
        -- insert stimulus here

        --Test ADD
90  control<="0100";

        --ADD1
        input1<="1010101010101010";
        input2<="0101010101010101";
95  wait for 50 ns;
        assert output="1111111111111111"
            report "ADD1 failed, expected 1111111111111111, got: " & integer'image(
to_integer(unsigned(output)));

        --ADD2
100  input1<="1111111111111111";
        input2<="0000000000000000";
        wait for 50 ns;
        assert output="1111111111111111"
            report "ADD2 failed, expected 1111111111111111, got: " & integer'image(
to_integer(unsigned(output)));
105  --
        --ADD3
        input1<="0000000000110011";
        input2<="0000000000010010";
            --"0000000000000000"
110  wait for 50 ns;
        assert output="0000000001000101"
            report "ADD3 failed, expected 0000000001000101, got: " & integer'image(
to_integer(unsigned(output)));

        --Test SUB
115  control<="0101";

        --SUB1
        input1<="1111111111111111";
        input2<="1111111111111111";
120  --"0000000000000000"
        wait for 50 ns;
        assert output="0000000000000000"
            report "SUB1 failed, expected 0000000000000000, got: " & integer'image(
to_integer(unsigned(output)));

        --SUB2
125  input1<="0000000001000000";--64
        input2<="0000000000010010";--18
            --"0000000000101110"--46

```

```

130     wait for 50 ns;
    assert output="0000000000101110"
        report "SUB2 failed , expected 0000000000101110, got: " & integer'image(
to_integer(unsigned(output)));

    --SUB3
    input1<="0000000000000111";--7
135    input2<="0000000000010000";--16
        --"111111111110111"--(-)9
    wait for 50 ns;
    assert output="111111111110111"
        report "SUB3 failed , expected 111111111110111, got: " & integer'image(
to_integer(unsigned(output)));
140

    --TEST MUL
    control<="0110";

    --MUL1
145    input1<="1111111111111111";
    input2<="0000000000000000";
        --"0000000000000000"
    wait for 50 ns;
    assert output="0000000000000000"
150    report "MUL1 failed , expected 0000000000000000, got: " & integer'image(
to_integer(unsigned(output)));

    --MUL2
    input1<="0000000000000100";--4
    input2<="0000000000000101";--5
155    --"0000000000010100"--20
    wait for 50 ns;
    assert output="0000000000010100"
        report "MUL2 failed , expected 0000000000010100, got: " & integer'image(
to_integer(unsigned(output)));

    --Test OR
160    control<="1000";

    --OR1
    input1<="1111010101000011";
165    input2<="0011100100110111";
        --"1111110101110111"
    wait for 50 ns;
    assert output="0011000100000011"
        report "OR1 failed , expected 0011000100000011, got: " & integer'image(
to_integer(unsigned(output)));
170

    --TEST NOT
    control<="1001";

    --NOT1
175    input1<="1111010101000011";
    input2<="0000000000000000";
        --"0000101010111100"
    wait for 50 ns;
    assert output="0000101010111100"
180    report "NOT1 failed , expected 0000101010111100, got: " & integer'image(
to_integer(unsigned(output)));

```

```

--TEST AND
control<="1010";

185
--AND1
input1<="1111010101000011";
input2<="0011100100110111";
--"0011000100000011"
wait for 50 ns;
190
assert output="0011000100000011"
    report "AND1 failed , expected 0011000100000011, got: " & integer'image(
to_integer(unsigned(output)));

--TEST XOR
control<="1011";

195
--XOR1
input1<="1111010101000011";
input2<="0011100100110111";
--"1100110001110100"
200
wait for 50 ns;
assert output="1100110001110100"
    report "XOR1 failed , expected 1100110001110100, got: " & integer'image(
to_integer(unsigned(output)));

--TEST SLL
205
control<="1100";

--SLL1
input1<="1111010101000011";
input2<="0000000000000100";
--"0101010000110000"
210
wait for 50 ns;
assert output="0101010000110000"
    report "SLL1 failed , expected 0101010000110000, got: " & integer'image(
to_integer(unsigned(output)));

--TEST SRL
215
control<="1101";

--SRL1
input1<="1111010101000011";
input2<="0000000000000100";
--"0000111101010100"
220
wait for 50 ns;
assert output="0000111101010100"
    report "SRL1 failed , expected 0000111101010100, got: " & integer'image(
to_integer(unsigned(output)));

225
--TEST SRA
control<="1110";

--SRA1
input1<="1111010101000011";
input2<="0000000000000100";
--"1111111101010100"
230
wait for 50 ns;
assert output="1111111101010100"
    report "SRA1 failed , expected 1111111101010100, got: " & integer'image(
235
to_integer(unsigned(output)));

```

```

        wait;
    end process;

END;
```

Listing 30: Logic Unit VHDL

```

0  ---
1  --- Company:
2  --- Engineer:
3  ---
4  --- Create Date:      08:51:09 03/02/2017
5  --- Design Name:
6  --- Module Name:      Logic_Unit - Behavioral
7  --- Project Name:
8  --- Target Devices:
9  --- Tool versions:
10 --- Description:
11 ---
12 --- Dependencies:
13 ---
14 --- Revision:
15 --- Revision 0.01 - File Created
16 --- Additional Comments:
17 ---
18 ---
19 ---
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 --- Uncomment the following library declaration if using
24 --- arithmetic functions with Signed or Unsigned values
25 ---use IEEE.NUMERIC_STD.ALL;
26
27 --- Uncomment the following library declaration if instantiating
28 --- any Xilinx primitives in this code.
29 ---library UNISIM;
30 ---use UNISIM.VComponents.all;
31
32 entity Logic_Unit is
33     generic( N : integer :=16);
34     Port ( A : in  STD_LOGIC_VECTOR (N-1 downto 0);
35           B : in  STD_LOGIC_VECTOR (N-1 downto 0);
36           Control : in  STD_LOGIC_VECTOR (3 downto 0);
37           output : out STD_LOGIC_VECTOR (N-1 downto 0));
38 end Logic_Unit;
39
40 ---Control:
41 ---1000 : or
42 ---1001 : not
43 ---1010 : AND
44 ---1011 : XOR
45 architecture Behavioral of Logic_Unit is
46
47 begin
48     proc1 : process(control, A, B)
49     begin
50         ---Depending on control, will do specific commands
51         if control ="1000" then ---bitwise OR
52             F0 : for i in 0 to N-1 loop
53                 output(i)<=A(i) OR B(i);
54             end loop;
55         end if;
56     end process;
57 end;
```

```

        end loop;

        elsif control = "1001" then --bitwise NOT
55      F1 : for i in 0 to N-1 loop
            output(i) <= NOT A(i);
        end loop;

        elsif control = "1010" then --bitwise AND
60      F2 : for i in 0 to N-1 loop
            output(i) <= A(i) AND B(i);
        end loop;

        elsif control = "1011" then --bitwise XOR
65      F3 : for i in 0 to N-1 loop
            output(i) <= A(i) XOR B(i);
        end loop;
    end if;
    end process;
70

end Behavioral;

```

## 5.2 Leonardo Scripts

Listing 31: Multiplier Spectrum Script

```

0 set exclude_gates {PadInC PadOut}
load_library ~/Pyxis_SPT_HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
  hdl_libs/gdk.syn
read {../SourceCode/FA_1bit.vhd ../SourceCode/AND2.vhd ../SourceCode/ANDADD.vhd ../
  SourceCode/Multiplier.vhd }
ungroup * -hierarchy
set sdf_write_flat_netlist TRUE
5 optimize
write -format verilog ./Output/Multiplier.v
write -format vhdl ./Output/Multiplier.vhdl
write -format sdf ./Output/Multiplier.sdf

```

Listing 32: LFSR 32 4 Spectrum Script

```

0 set exclude_gates {PadInC PadOut}
load_library ~/Pyxis_SPT_HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
  hdl_libs/gdk.syn
read {../SourceCode/LFSR_32_4.vhd }
set sdf_write_flat_netlist TRUE
optimize
5 write -format verilog ./Output/LFSR_32_4.v
write -format vhdl ./Output/LFSR_32_4.vhdl
write -format sdf ./Output/LFSR_32_4.sdf

```

Listing 33: MISR 32 4 Spectrum Script

```

0 set exclude_gates {PadInC PadOut}
load_library ~/Pyxis_SPT_HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
  hdl_libs/gdk.syn
read {../SourceCode/MISR_32_4.vhd }

```

```

set sdf_write_flat_netlist TRUE
optimize
5 write -format verilog ./Output/MISR_32_4.v
write -format vhdl ./Output/MISR_32_4.vhdl
write -format sdf ./Output/MISR_32_4.sdf

```

Listing 34: ProjectWrapper Spectrum Script

```

0 set exclude_gates {PadInC PadOut}
load_library ~/Pyxis.SPT.HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
hdl_libs/gdk.syn
read {../SourceCode/AND2.vhd ../SourceCode/ANDADD.vhd ../SourceCode/FA_1bit.vhd ../
SourceCode/FullAdder.vhd ../SourceCode/LFSR_32_4.vhd ../SourceCode/MISR_32_4.vhd
../SourceCode/Multiplier.vhd ../SourceCode/nBitRegister_16.vhd ../SourceCode/
nBitRegister_32.vhd ../SourceCode/nBitAdder.vhd ../SourceCode/Counter.vhd ../
SourceCode/TestController.vhd ../SourceCode/nBitMux_2to1.vhd ../SourceCode/MAC.
vhd ../SourceCode/ProjectWrapper.vhd }
ungroup * -hierarchy
set sdf_write_flat_netlist TRUE
5 optimize
write -format verilog ./Output/ProjectWrapper.v
write -format vhdl ./Output/ProjectWrapper.vhdl
write -format sdf ./Output/ProjectWrapper.sdf

```

Listing 35: nBitAdder Spectrum Script

```

0 set exclude_gates {PadInC PadOut}
load_library ~/Pyxis.SPT.HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
hdl_libs/gdk.syn
read {../SourceCode/FullAdder.vhd ../SourceCode/nBitAdder.vhd }
set sdf_write_flat_netlist TRUE
optimize
5 write -format verilog ./Output/nBitAdder.v
write -format vhdl ./Output/nBitAdder.vhdl
write -format sdf ./Output/nBitAdder.sdf

```

Listing 36: nBitAdderSubtractor Spectrum Script

```

0 set exclude_gates {PadInC PadOut}
load_library ~/Pyxis.SPT.HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
hdl_libs/gdk.syn
read {../SourceCode/FullAdder.vhd ../SourceCode/nBitAdder.vhd }
set sdf_write_flat_netlist TRUE
optimize
5 write -format verilog ./Output/nBitAdder.v
write -format vhdl ./Output/nBitAdder.vhdl
write -format sdf ./Output/nBitAdder.sdf

```

Listing 37: MAC BIST Spectrum Script

```

0 set exclude_gates {PadInC PadOut}
load_library ~/Pyxis.SPT.HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
hdl_libs/gdk.syn
read {./Output/LFSR_32_4.vhdl ./Output/MISR_32_4.vhdl ./Output/nBitAdder.vhdl ./
Output/Multiplier.vhdl ./Output/nBitRegister_16.vhdl ./Output/nBitRegister_32.
vhdl ./Output/nBitMux_2to1.vhdl ../SourceCode/MAC.vhd }
set sdf_write_flat_netlist TRUE
optimize

```

```

5 write -format verilog ./Output/MAC.v
  write -format vhdl ./Output/MAC.vhdl
  write -format sdf ./Output/MAC.sdf

```

Listing 38: nBitRegister 16 Spectrum Script

```

0 set exclude_gates {PadInC PadOut}
  load_library ~/Pyxis.SPT.HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
    hdl_libs/gdk.syn
  read {../SourceCode/nBitRegister_16.vhd }
  set sdf_write_flat_netlist TRUE
  optimize
5 write -format verilog ./Output/nBitRegister_16.v
  write -format vhdl ./Output/nBitRegister_16.vhdl
  write -format sdf ./Output/nBitRegister_16.sdf

```

Listing 39: MAC Spectrum Script

```

0 set exclude_gates {PadInC PadOut}
  load_library ~/Pyxis.SPT.HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
    hdl_libs/gdk.syn
  read {../SourceCode/FA_1bit.vhd ../SourceCode/AND2.vhd ../SourceCode/ANDADD.vhd ../
    SourceCode/Multiplier.vhd ../SourceCode/nBitAdder.vhd ../SourceCode/
    nBitRegister_16.vhd ../SourceCode/nBitRegister_32.vhd ../SourceCode/MAC.vhd }
  ungroup * -hierarchy
  set sdf_write_flat_netlist TRUE
5 optimize
  write -format verilog ../SourceCode/MAC.v
  write -format vhd ../SourceCode/MAC.vhd
  write -format sdf ../SourceCode/MAC.sdf

```

Listing 40: nBitMux 2to1 Spectrum Script

```

0 set exclude_gates {PadInC PadOut}
  load_library ~/Pyxis.SPT.HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
    hdl_libs/gdk.syn
  read {../SourceCode/nBitMux_2to1.vhd }
  set sdf_write_flat_netlist TRUE
  optimize
5 write -format verilog ./Output/nBitMux_2to1.v
  write -format vhdl ./Output/nBitMux_2to1.vhdl
  write -format sdf ./Output/nBitMux_2to1.sdf

```

Listing 41: nBitRegister 32 Spectrum Script

```

0 set exclude_gates {PadInC PadOut}
  load_library ~/Pyxis.SPT.HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
    hdl_libs/gdk.syn
  read {../SourceCode/nBitRegister_32.vhd }
  set sdf_write_flat_netlist TRUE
  optimize
5 write -format verilog ./Output/nBitRegister_32.v
  write -format vhdl ./Output/nBitRegister_32.vhdl
  write -format sdf ./Output/nBitRegister_32.sdf

```

Listing 42: MAC (copy) Spectrum Script

```

0 set exclude_gates {PadInC PadOut}

```

```
load_library ~/Pyxis_SPT_HEP/ic_reflibs/external_libs/GDKgates/GDKgates_utilities/
    hdl_libs/gdk.syn
read {./Output/nBitAdder.vhdl ./Output/Multiplier.vhdl ./Output/nBitRegister_16.vhdl
    ./Output/nBitRegister_32.vhdl ../SourceCode/MAC.vhd }
set sdf_write_flat_netlist TRUE
optimize
write -format verilog ./Output/MAC.v
write -format vhdl ./Output/MAC.vhdl
write -format sdf ./Output/MAC.sdf
```

### 5.3 Layout Versus Schematic Results

Listing 43: MAC with BIST LVS Results

```
#####  
##                                ##  
##          C A L I B R E      S Y S T E M        ##  
##                                ##  
##          L V S    R E P O R T                ##  
##                                ##  
#####  
  
REPORT FILE NAME:           lvs.report  
LAYOUT NAME:               /home/bxk5113/Pyxis_SPT_HEP/ic_projects/Pyxis_SPT/  
                           digicdesign/ProjectWrapper/ProjectWrapper.cal/lay.net ('ProjectWrapper')  
SOURCE NAME:              /home/bxk5113/Pyxis_SPT_HEP/ic_projects/Pyxis_SPT/  
                           digicdesign/ProjectWrapper/ProjectWrapper.cal/ProjectWrapper.calibre.src.net ('  
                           ProjectWrapper')  
RULE FILE:                 /home/bxk5113/Pyxis_SPT_HEP/ic_projects/Pyxis_SPT/  
                           digicdesign/ProjectWrapper/ProjectWrapper.cal/_LVS_  
CREATION TIME:             Fri Dec 6 11:22:53 2019  
CURRENT DIRECTORY:         /home/bxk5113/Pyxis_SPT_HEP/ic_projects/Pyxis_SPT/  
                           digicdesign/ProjectWrapper/ProjectWrapper.cal  
USER NAME:                 bxk5113  
CALIBRE VERSION:           v2013.4_26.18      Fri Dec 13 14:55:29 PST 2013  
  
OVERALL COMPARISON RESULTS  
  
#                               ##### - -  
#                               #     *  *  
# CORRECT                       # |  
#                               # \---/  
#                               #####
```



*****		
CELL SUMMARY		
*****		
40	Result	Layout
	CORRECT	ProjectWrapper
45		Source
		ProjectWrapper
*****		
LVS PARAMETERS		
*****		
50	o LVS Setup :	
	// LVS COMPONENT TYPE PROPERTY	
55	// LVS COMPONENT SUBTYPE PROPERTY	
	// LVS PIN NAME PROPERTY	
	LVS POWER NAME	"VD33" "AVDDB" "DVDD" "VDDG" "AVDDG" "
	AHVDD" "AVDDBG" "AHVDD" "VDD5V" "DHVDD" "TAVDDPST"	
		"TAVD33PST" "VDWELL" "AHVDDG" "AVDWELL" "
	AVDDR" "VDDSA" "TAVDD" "VDDPST" "TAVD33"	
		"AHVDDR" "HVDDWELL" "AHVDDWELL" "VDD" "
	AVDD"	
60	LVS GROUND NAME	"DVSS" "VSSG" "AVSSG" "AHVSS" "AVSSBG" "
	AHVSSB" "DHVSS" "TAVSSPST" "AHVSSG" "AVSSR"	
		"VS33" "TAVSS" "VSSPST" "VSSUB" "AVSSUB" "
	AHVSSR" "GND" "AGND" "HVSSUB" "VSS" "AHVSSUB"	
		"AVSS" "AVSSB"
	LVS CELL SUPPLY	NO
	LVS RECOGNIZE GATES	ALL
65	LVS IGNORE PORTS	NO
	LVS CHECK PORT NAMES	YES
	LVS IGNORE TRIVIAL NAMED PORTS	NO
	LVS BUILTIN DEVICE PIN SWAP	YES
	LVS ALL CAPACITOR PINS SWAPPABLE	YES
70	LVS DISCARD PINS BY DEVICE	NO
	LVS SOFT SUBSTRATE PINS	NO
	LVS INJECT LOGIC	YES
	LVS EXPAND UNBALANCED CELLS	YES
	LVS FLATTEN INSIDE CELL	NO
75	LVS EXPAND SEED PROMOTIONS	NO
	LVS PRESERVE PARAMETERIZED CELLS	NO
	LVS GLOBALS ARE PORTS	YES
	LVS REVERSE WL	NO
	LVS SPICE PREFER PINS	YES
80	LVS SPICE SLASH IS SPACE	YES
	LVS SPICE ALLOW FLOATING PINS	YES
	// LVS SPICE ALLOW INLINE PARAMETERS	
	LVS SPICE ALLOW UNQUOTED STRINGS	NO
	LVS SPICE CONDITIONAL LDD	NO
85	LVS SPICE CULL PRIMITIVE SUBCIRCUITS	NO
	LVS SPICE IMPLIED MOS AREA	NO

```

// LVS SPICE MULTIPLIER NAME
LVS SPICE OVERRIDE GLOBALS          NO
LVS SPICE REDEFINE PARAM            NO
90  LVS SPICE REPLICATE DEVICES      NO
LVS SPICE SCALE X PARAMETERS        NO
LVS SPICE STRICT WL                 NO
// LVS SPICE OPTION
LVS STRICT SUBTYPES                 NO
95  LVS EXACT SUBTYPES               NO
LAYOUT CASE                         NO
SOURCE CASE                         NO
LVS COMPARE CASE                    NO
LVS DOWNCASE DEVICE                 NO
100 LVS REPORT MAXIMUM                50
LVS PROPERTY RESOLUTION MAXIMUM     65536
// LVS SIGNATURE MAXIMUM
// LVS FILTER UNUSED OPTION
// LVS REPORT OPTION
105 LVS REPORT UNITS                  YES
// LVS NON USER NAME PORT
// LVS NON USER NAME NET
// LVS NON USER NAME INSTANCE
// LVS IGNORE DEVICE PIN
110
// Reduction

LVS REDUCE SERIES MOS               NO
LVS REDUCE PARALLEL MOS             YES
115 LVS REDUCE SEMI SERIES MOS        NO
LVS REDUCE SPLIT GATES              NO
LVS REDUCE PARALLEL BIPOLAR         YES
LVS REDUCE SERIES CAPACITORS        YES
LVS REDUCE PARALLEL CAPACITORS      YES
120 LVS REDUCE SERIES RESISTORS       YES
LVS REDUCE PARALLEL RESISTORS       YES
LVS REDUCE PARALLEL DIODES          YES
LVS REDUCTION PRIORITY              PARALLEL

125 LVS SHORT EQUIVALENT NODES       NO

// Trace Property

TRACE PROPERTY r(rm1) r r 0.2
130 TRACE PROPERTY r(rm2) r r 0.2
TRACE PROPERTY r(rm3) r r 0.2
TRACE PROPERTY r(rm4) r r 0.2
TRACE PROPERTY r(rm5) r r 0.2
TRACE PROPERTY r(rm6) r r 0.2
135 TRACE PROPERTY r(rm7) r r 0.2
TRACE PROPERTY r(rm8) r r 0.2
TRACE PROPERTY mimcap_g13 a a 0
TRACE PROPERTY mimcap_g13 m m 0
TRACE PROPERTY r(rndiffs) w w 0
140 TRACE PROPERTY r(rndiffs) l l 0
TRACE PROPERTY r(rpdiffs) w w 0
TRACE PROPERTY r(rpdiffs) l l 0
TRACE PROPERTY r(rndiffwo) w w 0
TRACE PROPERTY r(rndiffwo) l l 0
145 TRACE PROPERTY r(rpdiffwo) w w 0

```

```

TRACE PROPERTY r(rpdifwo) 1 1 0
TRACE PROPERTY r(rnwod) w w 0
TRACE PROPERTY r(rnwod) 1 1 0
TRACE PROPERTY r(rnwsti) w w 0
150 TRACE PROPERTY r(rnwsti) 1 1 0
TRACE PROPERTY r(rnpolylo) w w 0
TRACE PROPERTY r(rnpolylo) 1 1 0
TRACE PROPERTY r(rppolylo) w w 0
TRACE PROPERTY r(rppolylo) 1 1 0
155 TRACE PROPERTY r(rnpolyhi) w w 0
TRACE PROPERTY r(rnpolyhi) 1 1 0
TRACE PROPERTY r(rppolyhi) w w 0
TRACE PROPERTY r(rppolyhi) 1 1 0
TRACE PROPERTY mn(nmos) 1 1 0
160 TRACE PROPERTY mn(nmos) w w 0
TRACE PROPERTY mn(nmos_na) 1 1 0
TRACE PROPERTY mn(nmos_na) w w 0
TRACE PROPERTY mn(nmos_lvt) 1 1 0
TRACE PROPERTY mn(nmos_lvt) w w 0
165 TRACE PROPERTY mn(nmos_hvt) 1 1 0
TRACE PROPERTY mn(nmos_hvt) w w 0
TRACE PROPERTY mn(nmos_33) 1 1 0
TRACE PROPERTY mn(nmos_33) w w 0
TRACE PROPERTY mn(nmos_na33) 1 1 0
170 TRACE PROPERTY mn(nmos_na33) w w 0
TRACE PROPERTY mp(pmos) 1 1 0
TRACE PROPERTY mp(pmos) w w 0
TRACE PROPERTY mp(pmos_lvt) 1 1 0
TRACE PROPERTY mp(pmos_lvt) w w 0
175 TRACE PROPERTY mp(pmos_hvt) 1 1 0
TRACE PROPERTY mp(pmos_hvt) w w 0
TRACE PROPERTY mp(pmos_33) 1 1 0
TRACE PROPERTY mp(pmos_33) w w 0
TRACE PROPERTY d(ndiode) a a 0.5
180 TRACE PROPERTY d(ndiode_33) a a 0.5
TRACE PROPERTY d(pdiod) a a 0.5
TRACE PROPERTY d(pdiod_33) a a 0.5
TRACE PROPERTY d(nwdiode) a a 0.5
TRACE PROPERTY c(nmosvar) lr lr 0
185 TRACE PROPERTY c(nmosvar) wr wr 0
TRACE PROPERTY mp(pmos_rf25) rl rl 0
TRACE PROPERTY mp(pmos_rf25) nr nr 0
TRACE PROPERTY mp(pmos_rf) rl rl 0
TRACE PROPERTY mp(pmos_rf) nr nr 0
190 TRACE PROPERTY mn(nmos_rf) nr nr 0
TRACE PROPERTY mn(nmos_rf25) nr nr 0
TRACE PROPERTY c(moscap_rf) nr nr 0
TRACE PROPERTY c(moscap_rf25) nr nr 0
TRACE PROPERTY c(mimcap) lt lt 0
195 TRACE PROPERTY c(xjvar) nr nr 0
TRACE PROPERTY spiral_inductor_lvs nr nr 0

200 CELL COMPARISON RESULTS ( TOP LEVEL )

# ##### - -

```

```

# # # # * *
# # # CORRECT # |
# # # # # \---/
# #####

```

LAYOUT CELL NAME: ProjectWrapper  
 SOURCE CELL NAME: ProjectWrapper

#### INITIAL NUMBERS OF OBJECTS

	Layout	Source	Component Type
Ports:	72	72	
Nets:	10344	10344	
Instances:	9943	9943	MN (4 pins)
	9943	9943	MP (4 pins)
Total Inst:	19886	19886	

#### NUMBERS OF OBJECTS AFTER TRANSFORMATION

	Layout	Source	Component Type
Ports:	72	72	
Nets:	4859	4859	
Instances:	201	201	MN (4 pins)
	777	777	MP (4 pins)
	582	582	SPDW_2.1 (4 pins)
	83	83	SPDW_3.2 (6 pins)
	6	6	SPUP_2.1 (4 pins)
	30	30	SPUP_2.2 (5 pins)
	141	141	SPUP_3.2 (6 pins)
	2001	2001	_invv (4 pins)
	113	113	_invx2v (4 pins)
	662	662	_nand2v (5 pins)
	1	1	_nand3v (6 pins)
	14	14	_nand4v (7 pins)
	606	606	_nor2v (5 pins)
	16	16	_nor3v (6 pins)
	8	8	_nor4v (7 pins)
	1128	1128	_sdw2v (4 pins)
	141	141	_sdw3v (5 pins)
	1586	1586	_smp2v (4 pins)
	83	83	_smp3v (5 pins)
Total Inst:	8179	8179	

\*\*\*\*\*  
 INFORMATION AND WARNINGS  
 \*\*\*\*\*

	Matched Layout	Matched Source	Unmatched Layout	Unmatched Source	Component Type
Ports :	72	72	0	0	
Nets :	4859	4859	0	0	
Instances :	201	201	0	0	MN(NMOS)
	777	777	0	0	MP(PMOS)
	582	582	0	0	SPDW_2.1
	83	83	0	0	SPDW_3.2
	6	6	0	0	SPUP_2.1
	30	30	0	0	SPUP_2.2
	141	141	0	0	SPUP_3.2
	2001	2001	0	0	_invv
	113	113	0	0	_invx2v
	662	662	0	0	_nand2v
	1	1	0	0	_nand3v
	14	14	0	0	_nand4v
	606	606	0	0	_nor2v
	16	16	0	0	_nor3v
	8	8	0	0	_nor4v
	1128	1128	0	0	_sdw2v
	141	141	0	0	_sdw3v
	1586	1586	0	0	_smp2v
	83	83	0	0	_smp3v
Total Inst :	8179	8179	0	0	

o Initial Correspondence Points:

Ports: VDD VSS WE REGOUT[23] CLK REGOUT[24] REGOUT[17] REGOUT[18] REGOUT  
 [20]  
 REGOUT[19] REGOUT[26] REGOUT[25] REGOUT[21] RESET REGOUT[22] REGOUT  
 [27]  
 REGOUT[28] REGOUT[30] REGOUT[16] REGOUT[31] REGOUT[15] REGOUT[29]  
 REGOUT[14]  
 PASS REGOUT[13] REGOUT[9] REGOUT[12] A[12] COMPLETE REGOUT[11]  
 REGOUT[10] A[14]  
 REGOUT[8] REGOUT[7] REGOUT[0] B[7] A[8] B[8] A[13] A[10] A[9]  
 REGOUT[1] B[0]  
 STARTTEST A[15] A[11] REGOUT[4] A[6] B[11] B[13]

\*\*\*\*\*  
 SUMMARY  
 \*\*\*\*\*

```

*****
Total CPU Time:      0 sec
315 Total Elapsed Time: 0 sec

```

## 5.4 SPICE

Listing 44: layout test SPICE

```

0 * Example circuit file for simulating PEX

.OPTION DOTNODE
.HIER /

5 .INCLUDE "/home/bxk5113/Pyxis_SPT_HEP/ic_projects/Pyxis_SPT/digicdesign/
   ProjectWrapper/ProjectWrapper.cal/ProjectWrapper.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
10 ** 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 COMPLETE PASS REGOUT[31] REGOUT[30] REGOUT[29] REGOUT[28] REGOUT[27] REGOUT
[26] REGOUT[25] REGOUT[24] REGOUT[23] REGOUT[22] REGOUT[21] REGOUT[20] REGOUT[19]
REGOUT[18] REGOUT[17] REGOUT[16] REGOUT[15] REGOUT[14] REGOUT[13] REGOUT[12]
REGOUT[11] REGOUT[10] REGOUT[9] REGOUT[8] REGOUT[7] REGOUT[6] REGOUT[5] REGOUT[4]
REGOUT[3] REGOUT[2] REGOUT[1] REGOUT[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] CLK RESET STARTTEST WE
ProjectWrapper

* Output Capacitance
15 C.REGOUT[31] REGOUT[31] 0 120 f
C.REGOUT[30] REGOUT[30] 0 120 f
C.REGOUT[29] REGOUT[29] 0 120 f
C.REGOUT[28] REGOUT[28] 0 120 f
C.REGOUT[27] REGOUT[27] 0 120 f
20 C.REGOUT[26] REGOUT[26] 0 120 f
C.REGOUT[25] REGOUT[25] 0 120 f
C.REGOUT[24] REGOUT[24] 0 120 f
C.REGOUT[23] REGOUT[23] 0 120 f
C.REGOUT[22] REGOUT[22] 0 120 f
25 C.REGOUT[21] REGOUT[21] 0 120 f
C.REGOUT[20] REGOUT[20] 0 120 f
C.REGOUT[19] REGOUT[19] 0 120 f
C.REGOUT[18] REGOUT[18] 0 120 f
C.REGOUT[17] REGOUT[17] 0 120 f
30 C.REGOUT[16] REGOUT[16] 0 120 f
C.REGOUT[15] REGOUT[15] 0 120 f
C.REGOUT[14] REGOUT[14] 0 120 f
C.REGOUT[13] REGOUT[13] 0 120 f
C.REGOUT[12] REGOUT[12] 0 120 f
35 C.REGOUT[11] REGOUT[11] 0 120 f

```

```

C.REGOUT[10] REGOUT[10] 0 120 f
C.REGOUT[9] REGOUT[9] 0 120 f
C.REGOUT[8] REGOUT[8] 0 120 f
C.REGOUT[7] REGOUT[7] 0 120 f
40 C.REGOUT[6] REGOUT[6] 0 120 f
C.REGOUT[5] REGOUT[5] 0 120 f
C.REGOUT[4] REGOUT[4] 0 120 f
C.REGOUT[3] REGOUT[3] 0 120 f
C.REGOUT[2] REGOUT[2] 0 120 f
45 C.REGOUT[1] REGOUT[1] 0 120 f
C.REGOUT[0] REGOUT[0] 0 120 f

* - Analysis Setup - DC sweep
50 * 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]
55 .TRAN 0 2000n 0.05n

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

VFORCE_VDD VDD 0 DC 1.08
VFORCE_VSS VSS 0 DC 0

65 VFORCE_CLK CLK 0 PULSE (0 1.08 25n 0.1n 0.1n 25n 50n)

VFORCE_RESET RESET 0 pw1 (120n 1.08 120.1n 0 )
VFORCE_WE WE 0 DC 1.08

70 .SIGBUS A[15:0] VHI=1.08 VLO=0 TRISE=0.1n TFALL=0.1n TDELAY=210n THOLD=200n BASE=
    HEXA PATTERN 0002 0003 P
.SIGBUS B[15:0] VHI=1.08 VLO=0 TRISE=0.1n TFALL=0.1n TDELAY=210n THOLD=200n BASE=
    HEXA PATTERN 0003 0004 P

75 * --- Waveform Outputs
.PLOT TRAN V(COMPLETE)
.PLOT TRAN V(PASS)
.PLOT TRAN V(REGOUT[31])
.PLOT TRAN V(REGOUT[30])
80 .PLOT TRAN V(REGOUT[29])
.PLOT TRAN V(REGOUT[28])
.PLOT TRAN V(REGOUT[27])
.PLOT TRAN V(REGOUT[26])
.PLOT TRAN V(REGOUT[25])
85 .PLOT TRAN V(REGOUT[24])
.PLOT TRAN V(REGOUT[23])
.PLOT TRAN V(REGOUT[22])
.PLOT TRAN V(REGOUT[21])
.PLOT TRAN V(REGOUT[20])
90 .PLOT TRAN V(REGOUT[19])
.PLOT TRAN V(REGOUT[18])

```

```

. PLOT TRAN V(REGOUT[17])
. PLOT TRAN V(REGOUT[16])
. PLOT TRAN V(REGOUT[15])
95 . PLOT TRAN V(REGOUT[14])
. PLOT TRAN V(REGOUT[13])
. PLOT TRAN V(REGOUT[12])
. PLOT TRAN V(REGOUT[11])
. PLOT TRAN V(REGOUT[10])
100 . PLOT TRAN V(REGOUT[9])
. PLOT TRAN V(REGOUT[8])
. PLOT TRAN V(REGOUT[7])
. PLOT TRAN V(REGOUT[6])
. PLOT TRAN V(REGOUT[5])
105 . PLOT TRAN V(REGOUT[4])
. PLOT TRAN V(REGOUT[3])
. PLOT TRAN V(REGOUT[2])
. PLOT TRAN V(REGOUT[1])
. PLOT TRAN V(REGOUT[0])
110 . 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])
115 . 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])
120 . 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])
125 . 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])
130 . 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])
135 . 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])
140 . PLOT TRAN V(B[1])
. PLOT TRAN V(B[0])
. PLOT TRAN V(CLK)
. PLOT TRAN V(RESET)
. PLOT TRAN V(STARTTEST)
145 . PLOT TRAN V(WE)

* — Params
. TEMP 125

```

150



```
* — Power Measurement
.measure tran static_pwr AVG power from=90ns to=160ns
.measure tran inst_pwr MAX power from=90ns to=160ns
```

Listing 45: power test SPICE

```
0 * Example circuit file for simulating PEX

.OPTION DOTNODE
.HIER /

5 .INCLUDE "/home/bxk5113/Pyxis_SPT_HEP/ic_projects/Pyxis_SPT/digicdesign/
  ProjectWrapper/ProjectWrapper.cal/ProjectWrapper.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
10 ** 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 COMPLETE PASS REGOUT[31] REGOUT[30] REGOUT[29] REGOUT[28] REGOUT[27] REGOUT
[26] REGOUT[25] REGOUT[24] REGOUT[23] REGOUT[22] REGOUT[21] REGOUT[20] REGOUT[19]
REGOUT[18] REGOUT[17] REGOUT[16] REGOUT[15] REGOUT[14] REGOUT[13] REGOUT[12]
REGOUT[11] REGOUT[10] REGOUT[9] REGOUT[8] REGOUT[7] REGOUT[6] REGOUT[5] REGOUT[4]
REGOUT[3] REGOUT[2] REGOUT[1] REGOUT[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] CLK RESET STARTTEST WE
ProjectWrapper

* Output Capacitance
15 C.REGOUT[31] REGOUT[31] 0 120 f
C.REGOUT[30] REGOUT[30] 0 120 f
C.REGOUT[29] REGOUT[29] 0 120 f
C.REGOUT[28] REGOUT[28] 0 120 f
C.REGOUT[27] REGOUT[27] 0 120 f
20 C.REGOUT[26] REGOUT[26] 0 120 f
C.REGOUT[25] REGOUT[25] 0 120 f
C.REGOUT[24] REGOUT[24] 0 120 f
C.REGOUT[23] REGOUT[23] 0 120 f
C.REGOUT[22] REGOUT[22] 0 120 f
25 C.REGOUT[21] REGOUT[21] 0 120 f
C.REGOUT[20] REGOUT[20] 0 120 f
C.REGOUT[19] REGOUT[19] 0 120 f
C.REGOUT[18] REGOUT[18] 0 120 f
C.REGOUT[17] REGOUT[17] 0 120 f
30 C.REGOUT[16] REGOUT[16] 0 120 f
C.REGOUT[15] REGOUT[15] 0 120 f
C.REGOUT[14] REGOUT[14] 0 120 f
C.REGOUT[13] REGOUT[13] 0 120 f
C.REGOUT[12] REGOUT[12] 0 120 f
35 C.REGOUT[11] REGOUT[11] 0 120 f
C.REGOUT[10] REGOUT[10] 0 120 f
C.REGOUT[9] REGOUT[9] 0 120 f
C.REGOUT[8] REGOUT[8] 0 120 f
C.REGOUT[7] REGOUT[7] 0 120 f
40 C.REGOUT[6] REGOUT[6] 0 120 f
C.REGOUT[5] REGOUT[5] 0 120 f
C.REGOUT[4] REGOUT[4] 0 120 f
```

```

C.REGOUT[3] REGOUT[3] 0 120 f
C.REGOUT[2] REGOUT[2] 0 120 f
45 C.REGOUT[1] REGOUT[1] 0 120 f
C.REGOUT[0] REGOUT[0] 0 120 f

* - Analysis Setup - DC sweep
50 * 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]
55 .TRAN 0 600n 0.1n

* --- Forces
* FORMAT --- PULSE : [name] [port] [reference (0 means ground)] PULSE [low] [high] [
    delay] [fall time] [rise time] [pulse width] [period]
*
60 * 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)
65 VFORCE_VDD VDD 0 DC 1.08
VFORCE_VSS VSS 0 DC 0

VFORCE_CLK CLK 0 PULSE (0 1.08 25n 0.1n 0.1n 25n 50n)
70 VFORCE_RESET RESET 0 pw1 (120n 1.08 120.1n 0 )

.SIGBUS A[15:0] VHI=1.08 VLO=0 TRISE=0.1n TFALL=0.1n TDELAY=210n THOLD=200n BASE=
    HEXA PATTERN EFAB 3FD6 P
75 .SIGBUS B[15:0] VHI=1.08 VLO=0 TRISE=0.1n TFALL=0.1n TDELAY=210n THOLD=200n BASE=
    HEXA PATTERN 8C5F 0004 P

* --- Waveform Outputs
.PLOT TRAN V(COMPLETE)
80 .PLOT TRAN V(PASS)
.PLOT TRAN V(REGOUT[31])
.PLOT TRAN V(REGOUT[30])
.PLOT TRAN V(REGOUT[29])
.PLOT TRAN V(REGOUT[28])
85 .PLOT TRAN V(REGOUT[27])
.PLOT TRAN V(REGOUT[26])
.PLOT TRAN V(REGOUT[25])
.PLOT TRAN V(REGOUT[24])
.PLOT TRAN V(REGOUT[23])
90 .PLOT TRAN V(REGOUT[22])
.PLOT TRAN V(REGOUT[21])
.PLOT TRAN V(REGOUT[20])
.PLOT TRAN V(REGOUT[19])
.PLOT TRAN V(REGOUT[18])
95 .PLOT TRAN V(REGOUT[17])
.PLOT TRAN V(REGOUT[16])
.PLOT TRAN V(REGOUT[15])
.PLOT TRAN V(REGOUT[14])

```

```

100 .PLOT TRAN V(REGOUT[13])
    .PLOT TRAN V(REGOUT[12])
    .PLOT TRAN V(REGOUT[11])
    .PLOT TRAN V(REGOUT[10])
    .PLOT TRAN V(REGOUT[9])
    .PLOT TRAN V(REGOUT[8])
105 .PLOT TRAN V(REGOUT[7])
    .PLOT TRAN V(REGOUT[6])
    .PLOT TRAN V(REGOUT[5])
    .PLOT TRAN V(REGOUT[4])
    .PLOT TRAN V(REGOUT[3])
110 .PLOT TRAN V(REGOUT[2])
    .PLOT TRAN V(REGOUT[1])
    .PLOT TRAN V(REGOUT[0])
    .PLOT TRAN V(A[15])
    .PLOT TRAN V(A[14])
115 .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])
120 .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])
125 .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])
130 .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])
135 .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])
140 .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])
145 .PLOT TRAN V(CLK)
    .PLOT TRAN V(RESET)
    .PLOT TRAN V(STARTTEST)
    .PLOT TRAN V(WE)

150 * — Params
    .TEMP 125

    * — Power Measurement
155 .measure tran static_pwr AVG power from=220ns to=50ns
    .measure tran inst_pwr MAX power from=10ns to=600ns

```

## 6 References

Key, Brandon A. *CMPE 260 Laboratory Exercise 3 Arithmetic Logic Unit*. CMPE 260 Laboratory Exercise 3 Arithmetic Logic Unit.

//TODO add BIST from DSD II //TODO Added Chris's DSD II lab