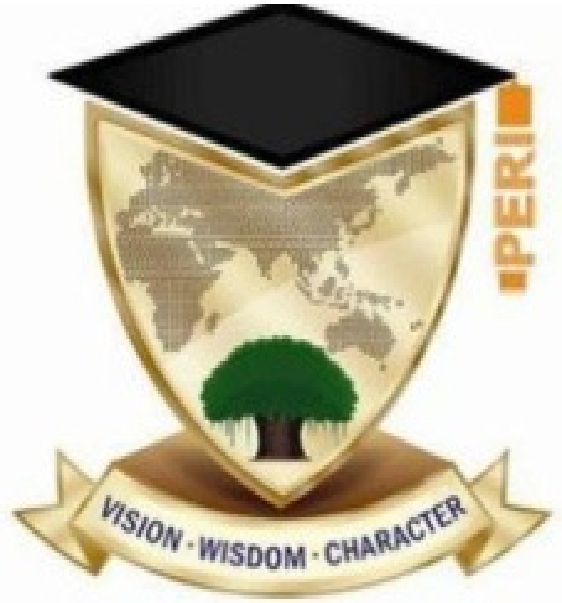
PERI INSTITUTE OF TECHNOLOGY

Mannivakkam, Chennai-600 048

*(Affiliated to **ANNA UNIVERSITY**, Chennai)*



DEPARMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SIXTH SEMESTER

EC 2357 VLSI DESIGN LAB

LAB MANUAL

SYLLABUS
EC2357 VLSI DESIGN LABORATORY

1. Design Entry and simulation of combinational logic circuits (8 bit adders, 4 bit multipliers, address decoders, multiplexers), Test bench creation, functional verification, and concepts of concurrent and sequential execution to be highlighted.
2. Design Entry and simulation of sequential logic circuits (counters, PRBS generators, accumulators). Test bench creation, functional verification, and concepts of concurrent and sequential execution to be highlighted.
3. Synthesis, P&R and Post P&R simulation for all the blocks/codes developed in Expt. No. 1 and No. 2 given above. Concepts of FPGA floor plan, critical path, design gate count, I/O configuration and pin assignment to be taught in this experiment.
4. Generation of configuration/fuse files for all the blocks/codes developed as part of Expt.1. and Expt. 2. FPGA devices must be configured and hardware tested for the blocks/codes developed as part of Expt. 1. and Expt. 2. The correctness of the inputs and outputs for each of the blocks must be demonstrated atleast on oscilloscopes (logic analyzer preferred).
5. Schematic Entry and SPICE simulation of MOS differential amplifier. Determination of gain, bandwidth, output impedance and CMRR.
6. Layout of a simple CMOS inverter, parasitic extraction and simulation.
7. Design of a 10 bit number controlled oscillator using standard cell approach, simulation followed by study of synthesis reports.
8. Automatic layout generation followed by post layout extraction and simulation of the circuit studied in Expt. No.7

Note 1:For Expt. 1 To 4 can be carried out using Altera (Quartus) / Xilinx (Alliance) / ACTEL (Libero) tools.

Note 2: For expt. 5-8 introduce the student to basics of IC design. These have to be carried out using atleast 0.5u CMOS technology libraries. The S/W tools needed Cadence / MAGMA / Tanner.

LIST OF EXPERIMENTS

EX.NO	NAME OF THE EXPERIMENT
1	Study of simulation and synthesis using tools
2	Simulation of basic Logic Gates
3	Simulation of Half Adder and Full Adder
4	Simulation of Half Subtractor and Full Subtractor
5	Simulation of Multiplexer and Demultiplexer
6	Simulation of Encoder and Decoder
7	Simulation of Arithmetic Logic Unit
8	Simulation of 8- bit adder
9	Simulation of 5-bit multiplier
10	Simulation of D Flip Flop and D Latch
11	Simulation of Pseudo Random Binary Sequence
12	Simulation of Accumulator
13	Simulation of Up-Down Counter
14	Study of implementation in FPGA
15	Implementation of basic logic gates in FPGA
16	Implementation of Half Adder and Full Adder in FPGA
17	Implementation of Half Subtractor and Full Subtractor in FPGA
18	Implementation of Multiplexer and Demultiplexer in FPGA
19	Implementation of Encoder and Decoder in FPGA
20	Implementation of D Flip Flop and D Latch in FPGA
21	Implementation of MOS differential amplifier schematic in ORCAD PSPICE

22	Study of simulating a layout in Microwind
23	Implementation of CMOS Inverter layout in Microwind
24	Implementation of CMOS NAND gate layout in Microwind

EXP: NO: 1	STUDY OF SIMULATION AND SYNTHESIS USING TOOLS
DATE:	

AIM:

To study the various synthesis tools used in Xilinx ISE Design Suite 12.1software.

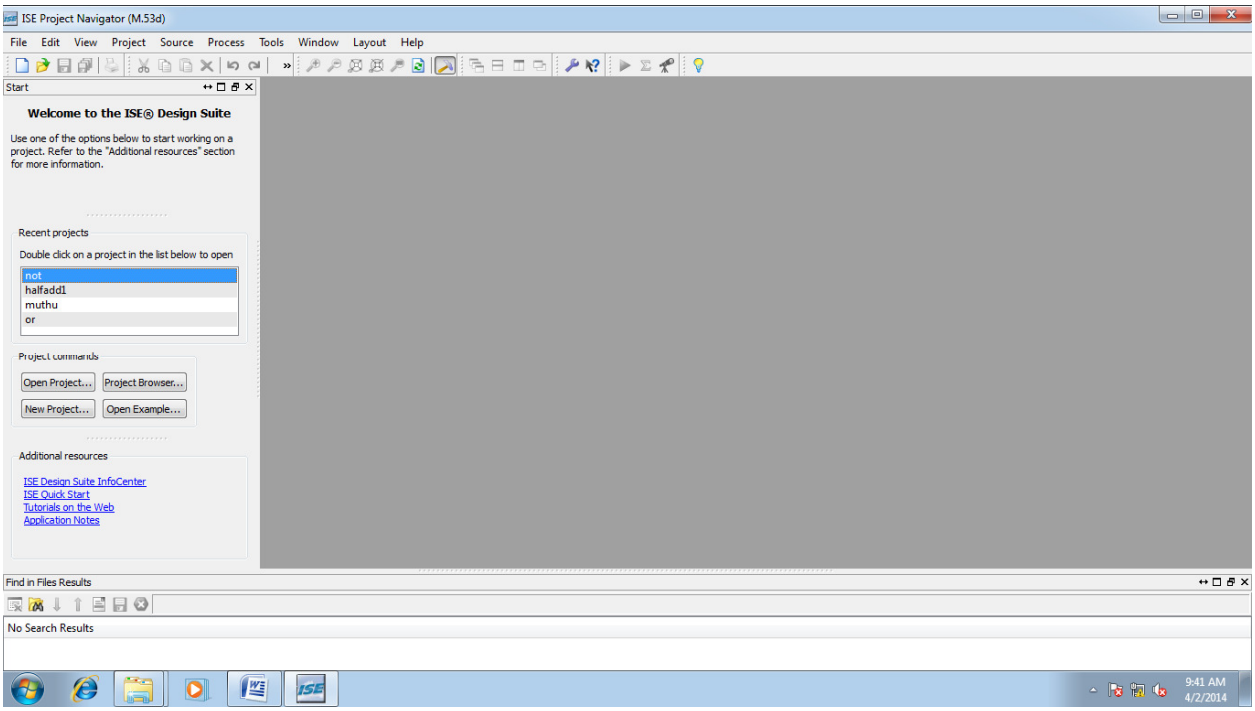
Eg: AND gate

SOFTWARE TOOL REQUIRED:

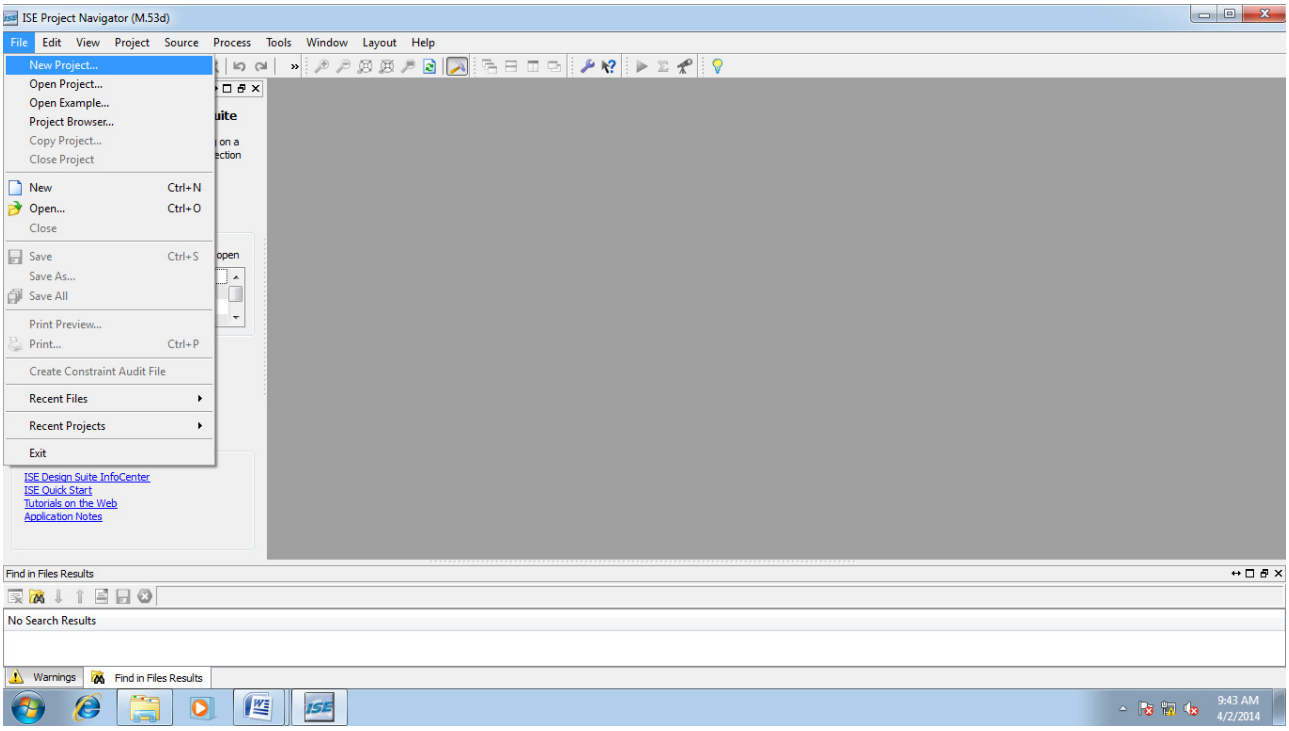
1. Xilinx ISE Design Suite 12.1

PROCEDURE:

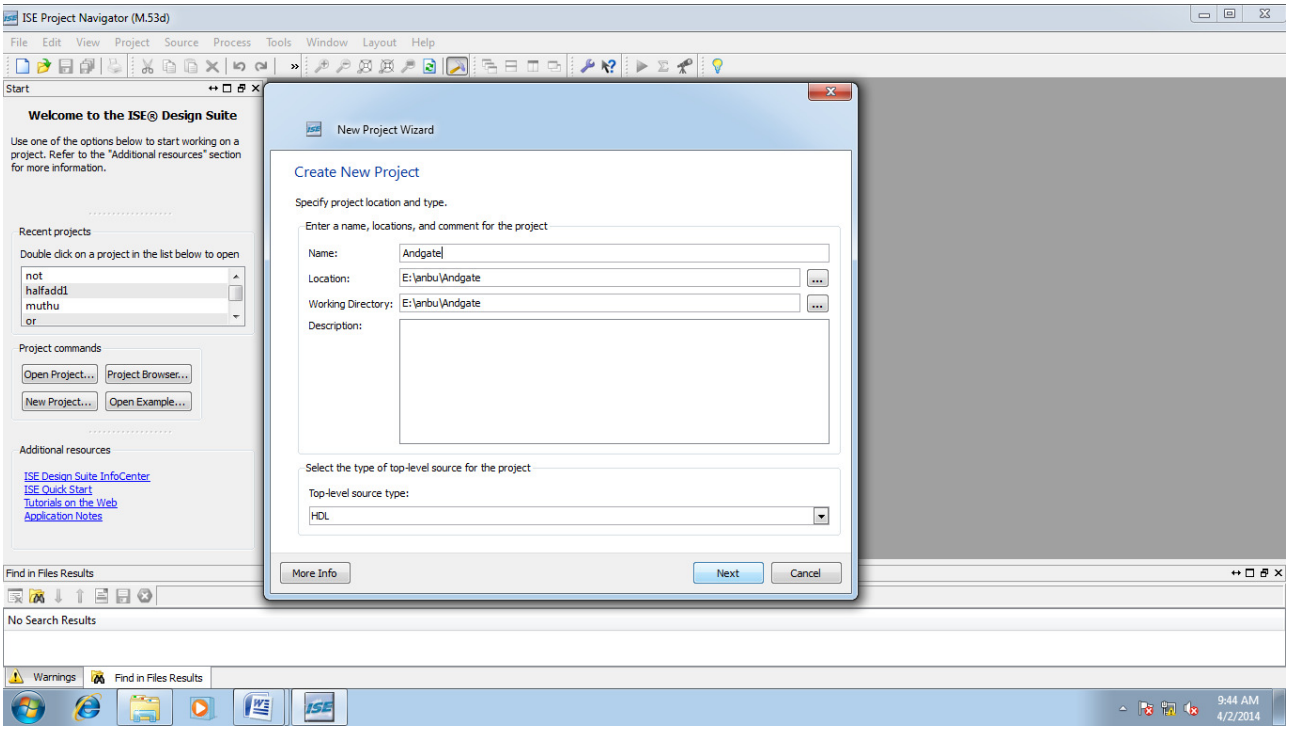
1. **Now start the Xilinx ISE Design Suite 12.1**



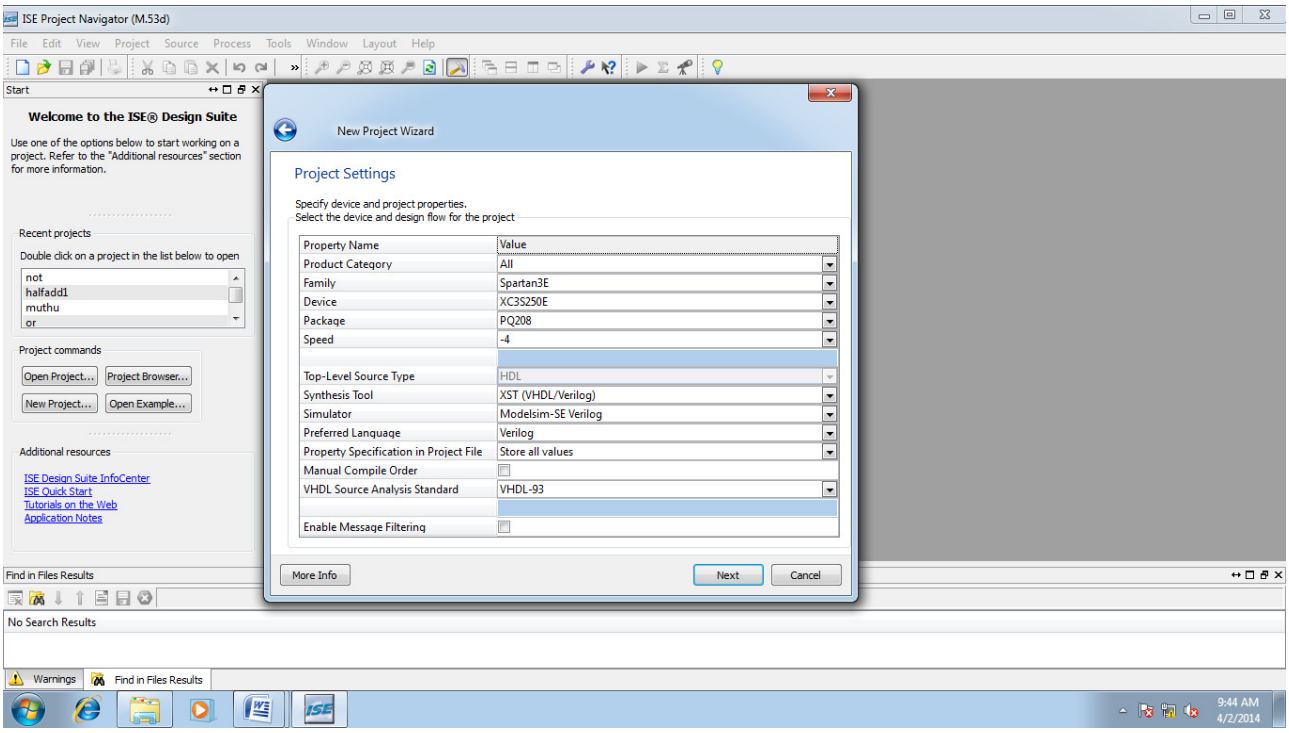
2. Go to file and click new project



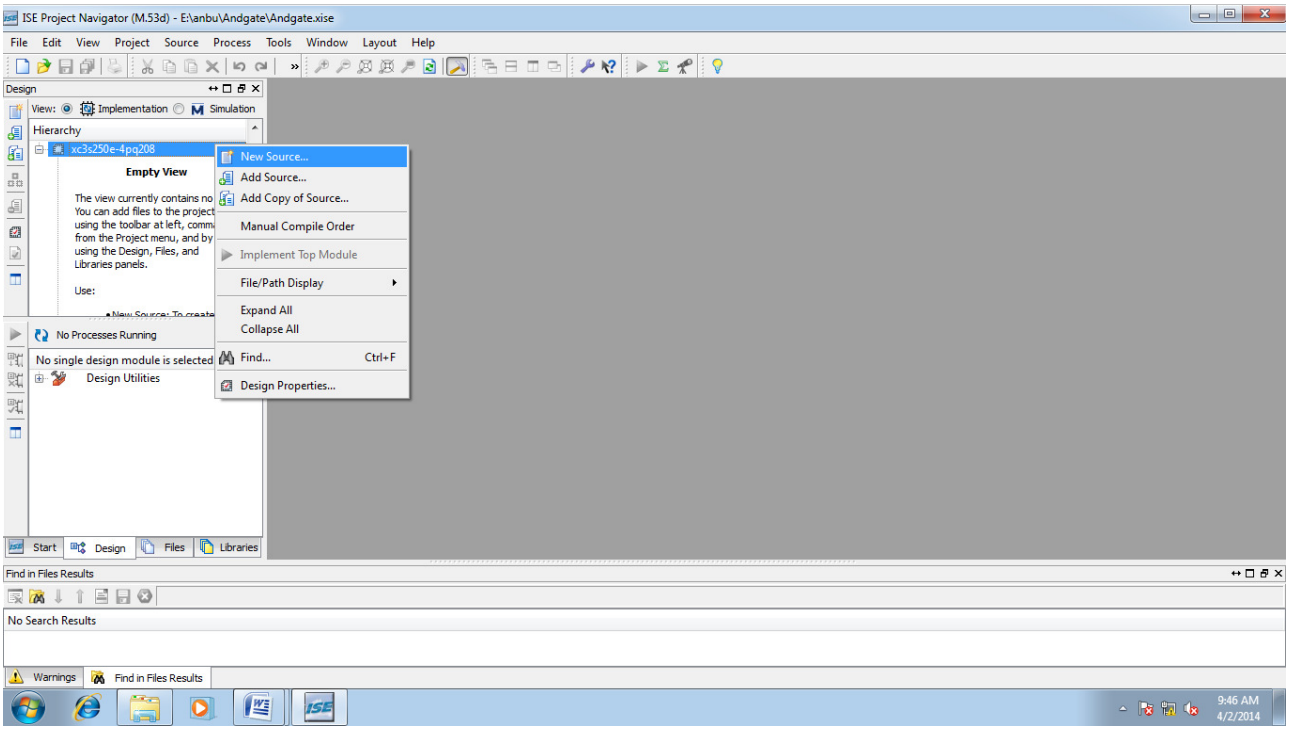
3. Enter the project name and click next



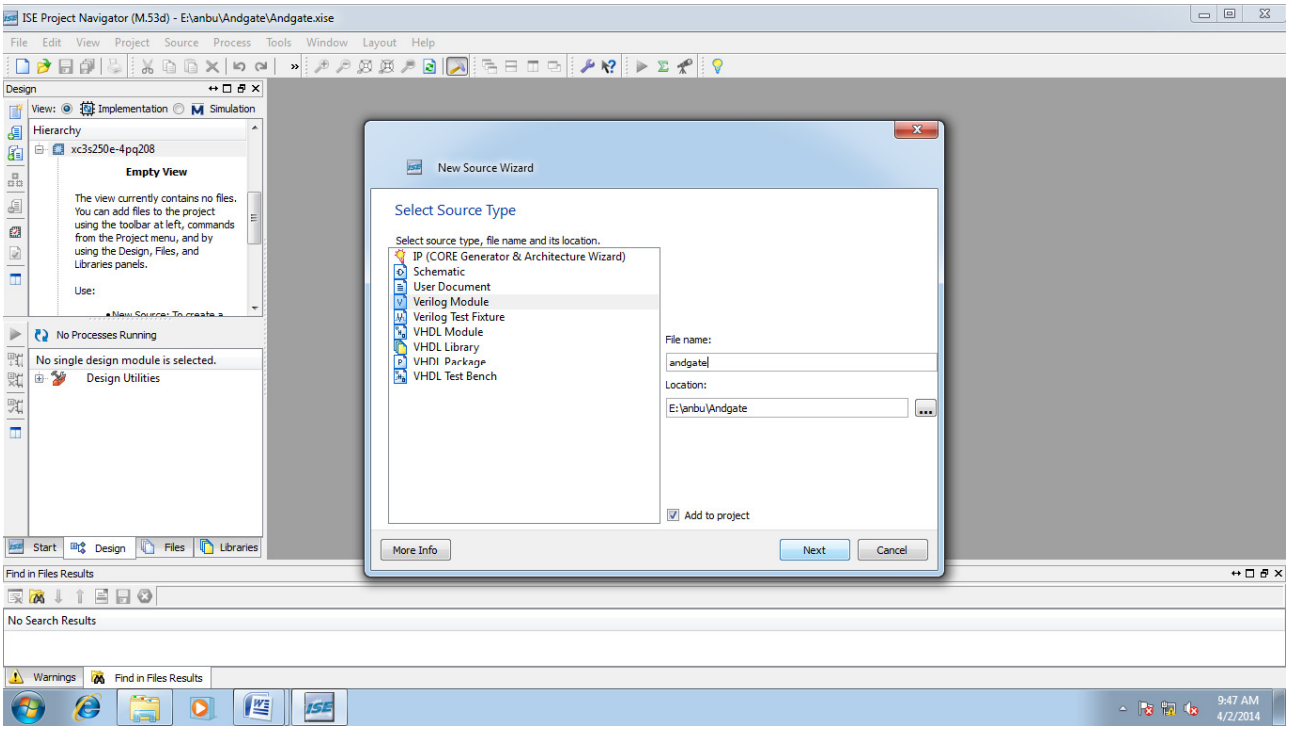
4. **Select the family name is Spartan 3E, speed is -4 and simulator is verilog click next and click Finish.**



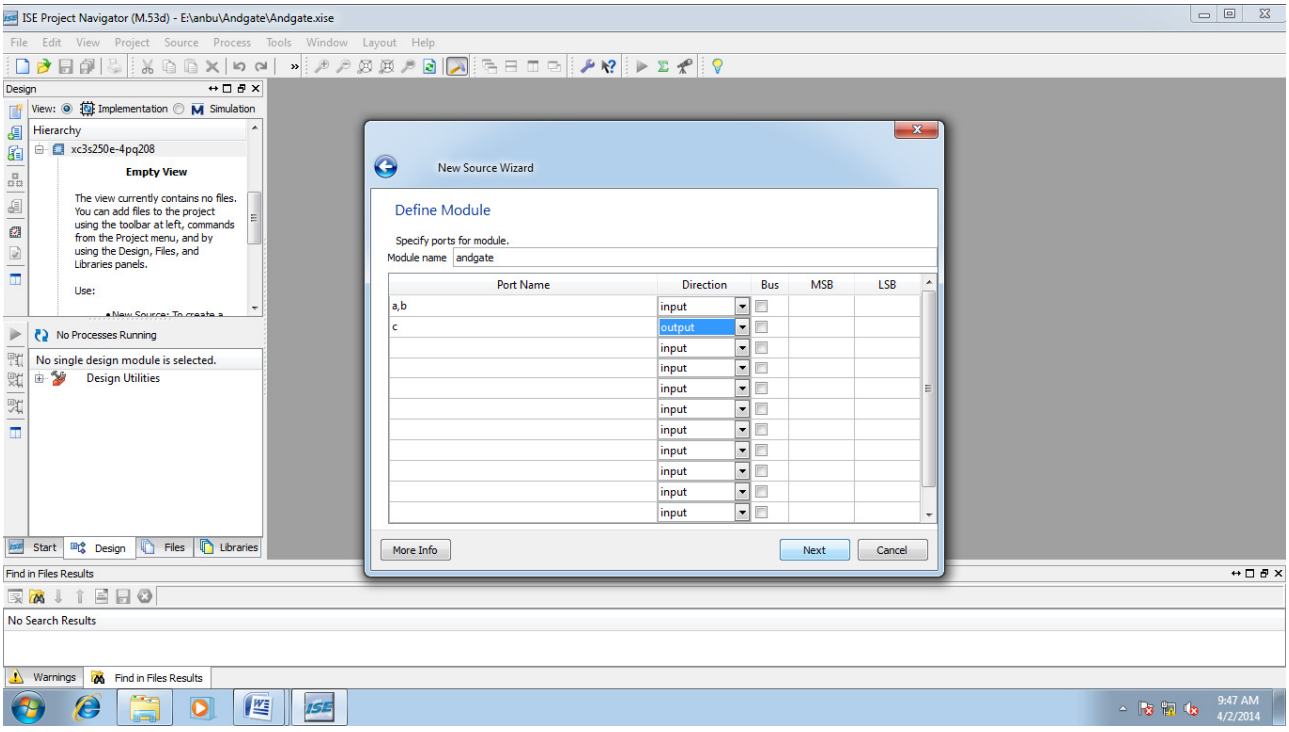
5. **Click new source.**



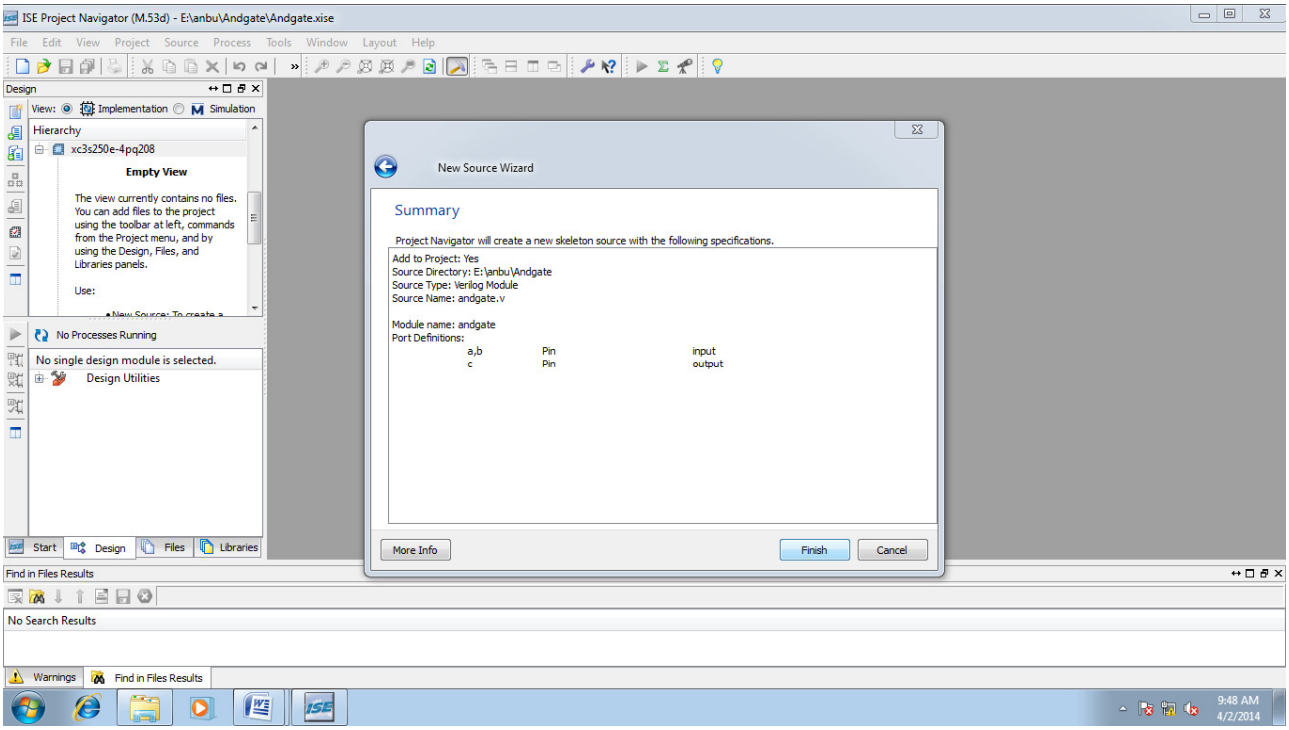
6. Select verilog module and type file name and click next.



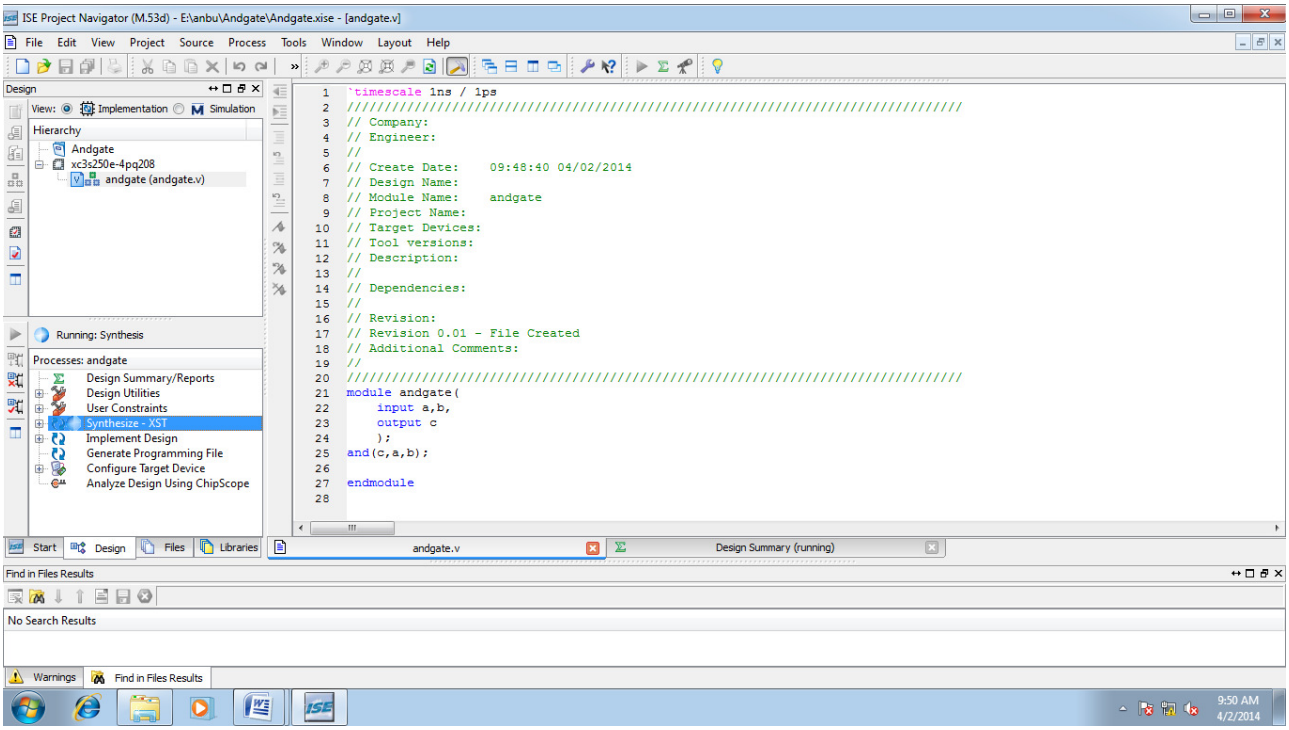
7. Assign input and output port and click next.



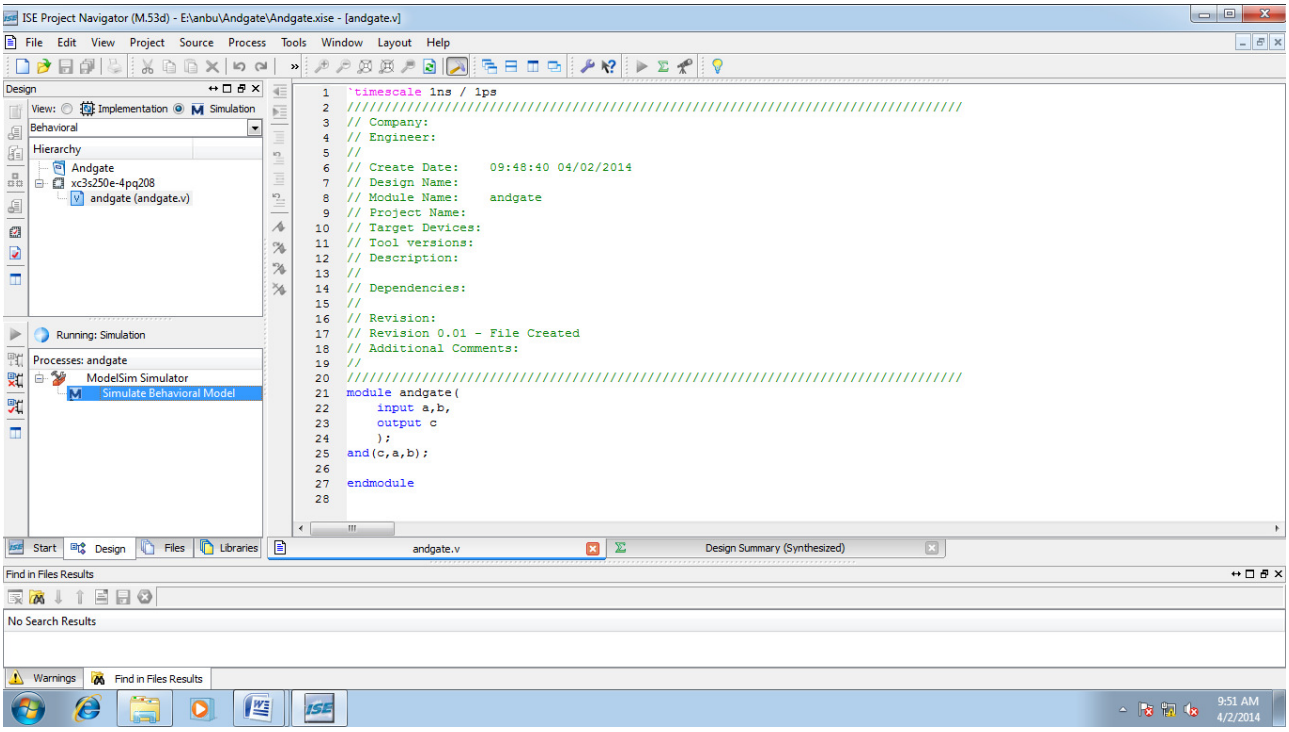
8. Finally the report is shown click finish.



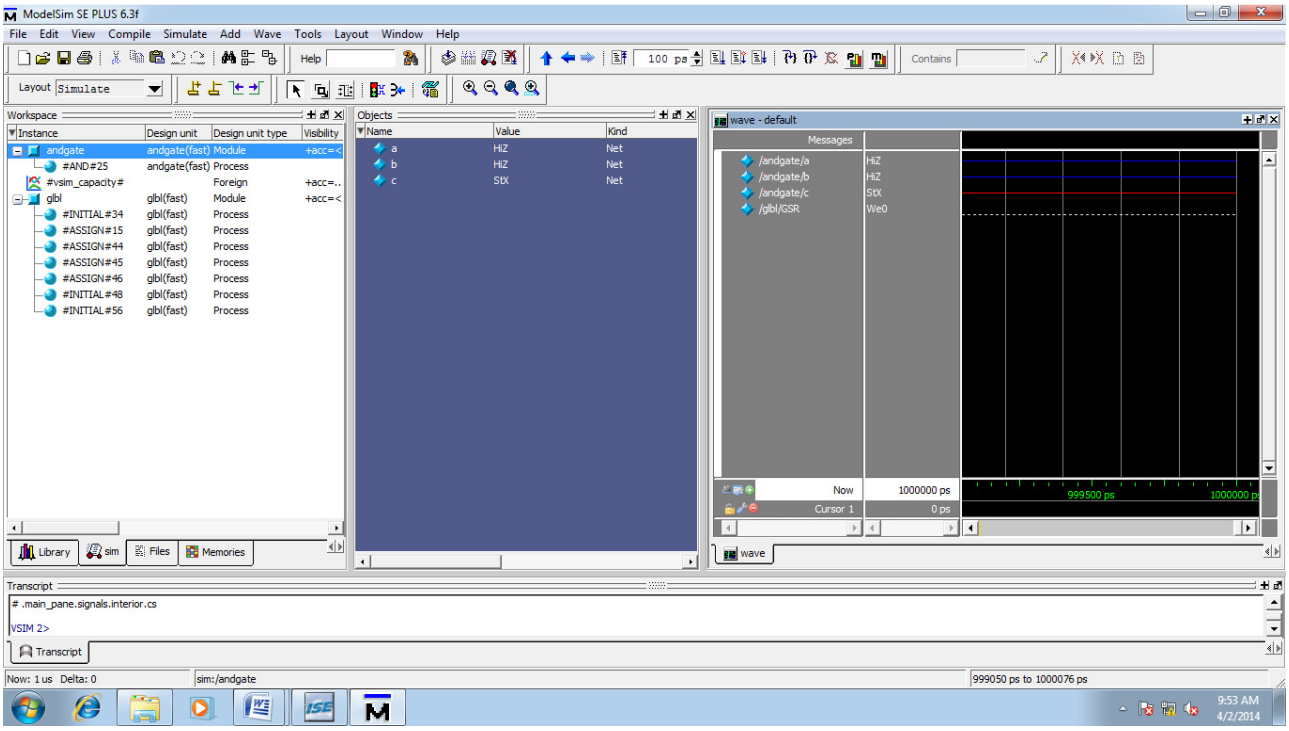
9. Type the program save and click synthesis.



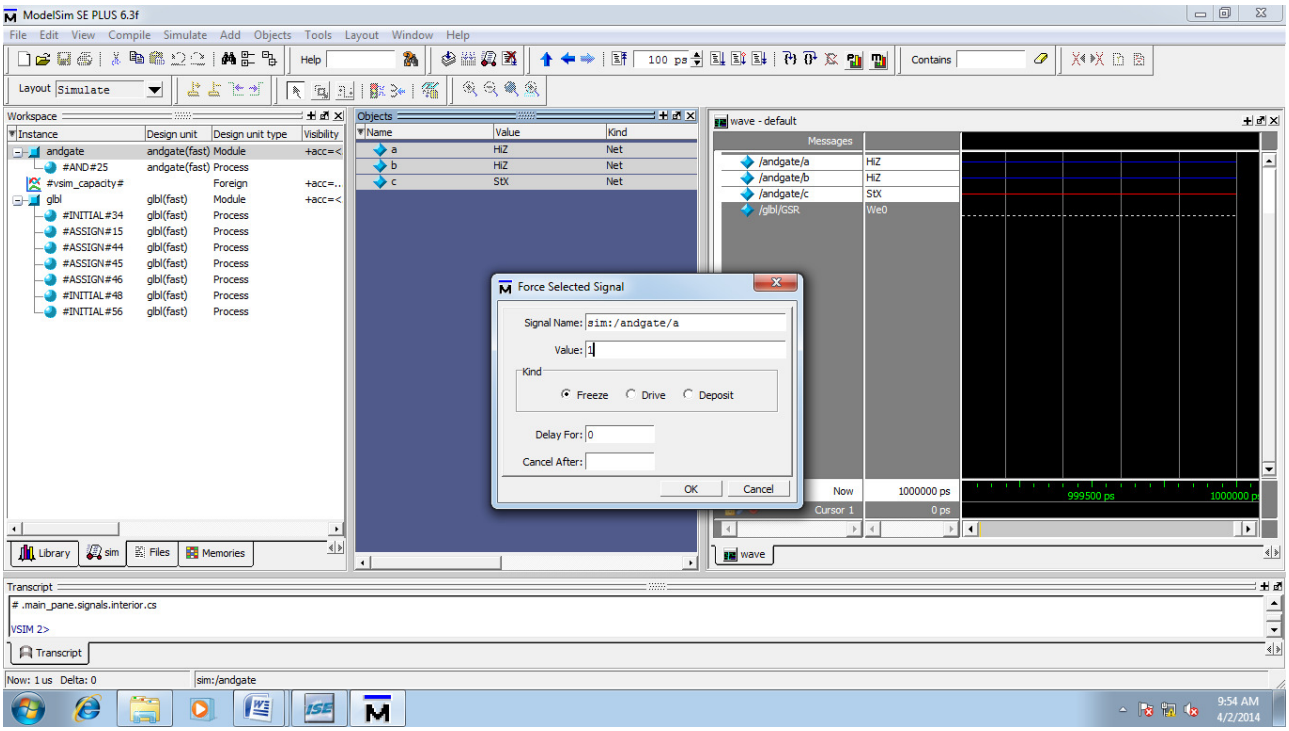
10. To see the output wave form change the source to behavioral simulation and click simulator behavior model in modelsim simulator. And Click No



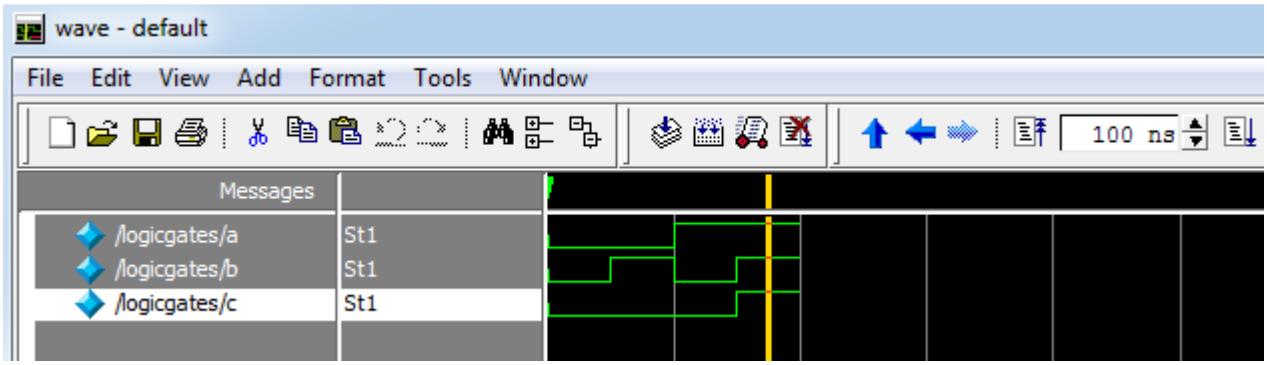
11. Select your file in work area.



12. In object window select input and output signal and provide input.



13. In wave window, click run icon and you can see corresponding output.



RESULT:

Thus the study about various synthesis tools used in the Xilinx ISE Design Suite 12.1 was studied.

EXP.NO:2	SIMULATION OF BASIC LOGIC GATES
DATE:	

AIM:

To write a verilog program for basic logic gates to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Write the functionality of the gates.
6. Terminate the program.

THEORY:

AND GATE:

The AND gate performs logical multiplication which is most commonly known as the AND junction. The operation of AND gate is such that the output is high only when all its inputs are high and when any one of the inputs is low the output is low.

$$Y = a \,\&\, b$$

OR GATE:

The OR gate performs logical addition which is most commonly known as the OR junction. The operation of OR gate is such that the output is high only when any one of its input is high and when both the inputs are low the output is low.

$$Y = a \mid b$$

NOT GATE:

The Inverter performs a basic logic gate function called Inversion or Complementation. The purpose of an inverter is to change one logic level to opposite level. When a high level is applied top an inverter, the low level will appear at the output and vice versa.

Y = ~a

NAND GATE:

The term NAND is derived from the complement of AND. It implies the AND junction with an inverted output. The operation of NAND gate is such that the output is low only when all its inputs are high and when any one of the inputs is low the output is high.

Y = ~(a & b)

NOR GATE:

The term NOR is derived from the complement of OR. It implies the OR junction with an inverted output. The operation of NOR gate is such that the output is high only when all its inputs are low and when any one of the inputs is high the output is low.

Y = ~(a | b)

EX-OR GATE:

The output is high only when the inputs are at opposite level.

Y = a ^ b

EX-NOR GATE:

The output is high only when the inputs are at same level.

Y = ~(a ^ b)

PROGRAM:

Verilog Code for basic logic gates

```
module logicgates(a,b,c,d,e,f,g,h);
input a,b;
output c,d,e,f,g,h;
and(c,a,b);
or(d,a,b);
not(e,a);
nand(f,a,b);
xor(g,a,b);
```

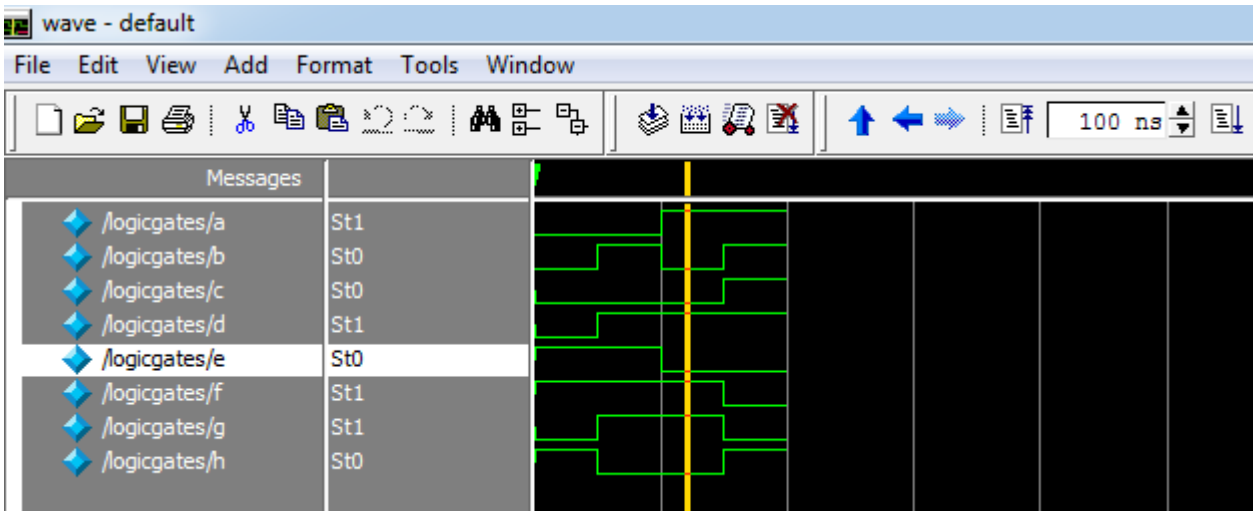
```
xnor(h,a,b);
endmodule
```

PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double
4. clicking on the synthesis in the process window.
5. Perform the functional simulation using Xilinx ISE simulator.
6. The output wave form can be observed in model sim.

SIMULATION REPORT



RESULT:

Thus the verilog program for basic logic gates were written, synthesized and simulated using Xilinx tool.

EXP: NO: 3	SIMULATION OF HALF ADDER AND FULL ADDER
DATE:	

AIM:

To write a verilog program for half adder and full adder to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Terminate the program.

THEORY:

HALF ADDER:

The half adder consists of two input variables designated as Augends and Addend bits. Output variables produce the Sum and Carry. The 'carry' output is 1 only when both inputs are 1 and ,sum' is 1 if any one input is 1. The Boolean expression is given by,

sum = x ^ y

carry = x & y

FULL ADDER:

A Full adder is a combinational circuit that focuses the arithmetic sum of three bits. It consists of 3 inputs and 2 outputs. The third input is the carry from the previous Lower Significant Position. The two outputs are designated as Sum (S) and Carry (C). The binary variable S gives the value of the LSB of the Sum. The output S=1 only if odd number of 1's are present in the input and the output C=1 if two or three inputs are 1.

sum = x ^ y ^ z

carry= (x & y) | (y & z) | (x & z)

PROCEDURE:

Software part

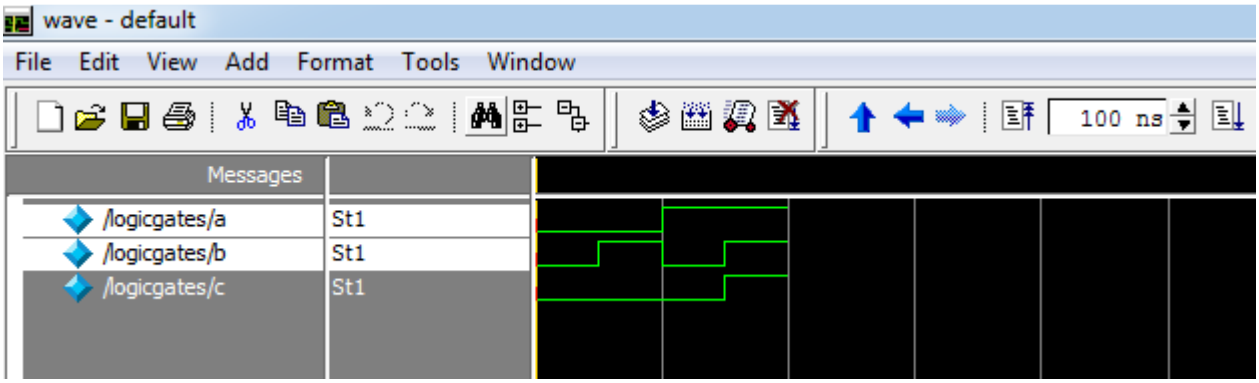
1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output can be observed by using model sim.

PROGRAM:

Verilog code for half adder

```
module halfadder(a, b, sum, carry);
input a;
input b;
output sum;
output carry;
xor(sum,a,b);
and(carry,a,b);
endmodule
```

SIMULATION REPORT

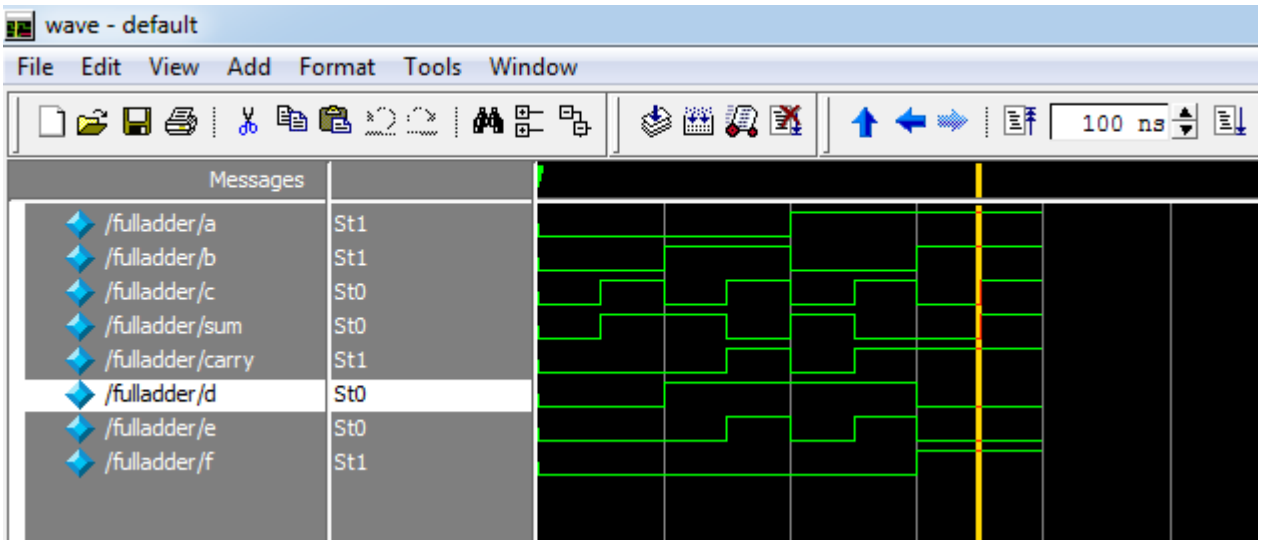


PROGRAM:

Verilog code for full adder

```
module fulladder(a,b,c,d,e,f,sum,carry);
input a,b,c;
output sum,carry,d,e,f;
wire d,e;
xor(d,a,b);
xor(sum,d,c);
and(e,d,c);
and(f,a,b);
or(carry,e,f);
endmodule
```

SIMULATION REPORT



RESULT:

Thus the verilog program for half adder and full adder were written, synthesized and simulated using Xilinx tool.

EXP: NO: 4	SIMULATION OF HALF SUBTRACTOR AND FULL SUBTRACTOR
DATE:	

AIM:

To write a verilog program for half subtractor and full subtractor to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

THEORY:

HALF SUBRACTOR:

The Half subtractor consist of two input variables, the output variables produce the Difference (d) and Borrow (bo). The output 'bo' is 1 only when the input 'a' is at low level and other input 'b' is at higher level. The output'd' is 1 only when only one of the inputs is 1. The Boolean expression for half subtractor is given by,

difference = a ^ b

borrow = a'b

FULL SUBTRACTOR:

A full subtractor is a multiple output combinational logical network which performs a subtraction between two binary bits considering that a '1' might have been borrowed by a lower significant stage. Along with the minuend 'a' and the subtrahend 'b', the third input is the borrow bit 'c', from the previous stage of subtraction. The combinational logic network of the full subtractor thus has three inputs and two outputs. The two outputs produced are the difference bit output'd' and a final borrow 'bo' respectively. The Boolean expression for full subtractor is given by,

difference= a ^ b ^ c

borrow = a'b + a'c + bc

PROCEDURE: (SAME FOR HALF SUBTRACTOR AND FULL SUBTRACTOR)

Software part

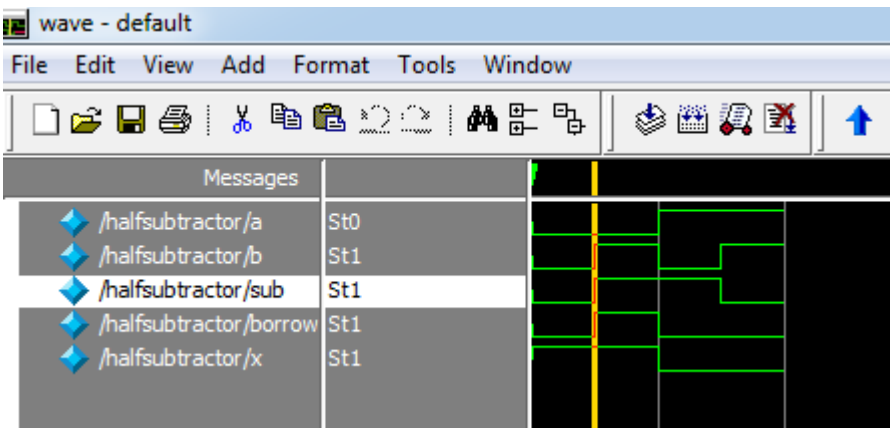
1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Draw the half subtractor and full subtractor circuit by choosing schematic as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output can be observed by using model sim.

PROGRAM:

Verilog code for half subtractor

```
module halfsubtractor(a,b,sub,borrow,x)
input a,b;
output sub,borrow,x;
xor(sub,a,b);
not(x,a);
and(borrow,x,b);
endmodule
```

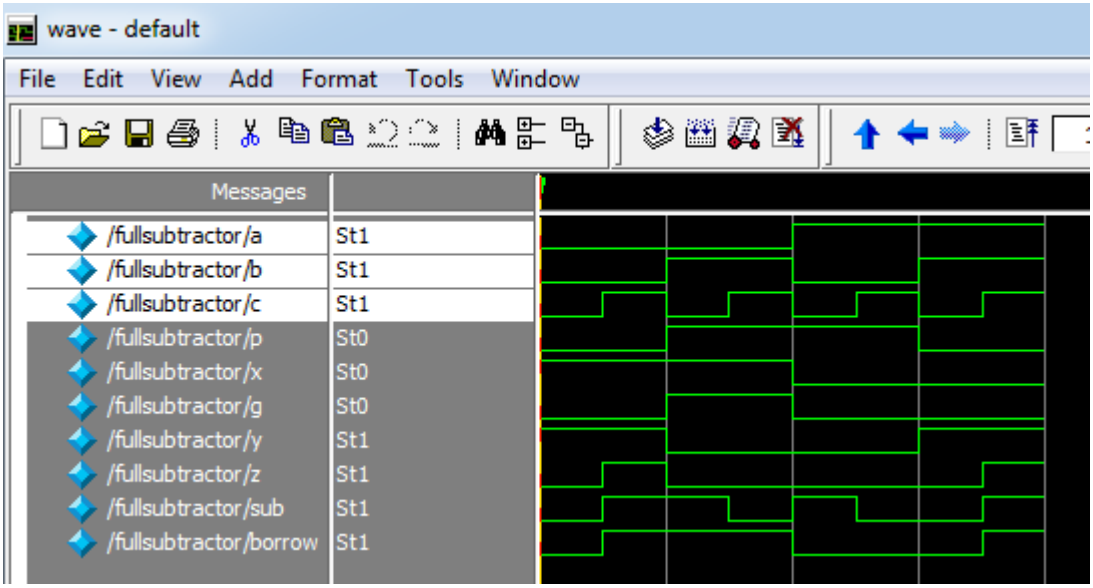
SIMULATION REPORT: (FOR HALF SUBTRACTOR)



Verilog code for full subtractor

```
module fullsubtractor(a,b,c,sub,borrow,p,x,g,y,z);
input a,b,c,p,x,g,y,z;
output sub,borrow;
xor(p,a,b);
not(x,a);
and(g,x,b);
xor(sub,p,c);
not(y,p);
and(z,y,c);
or(borrow,z,g);
endmodule
```

SIMULATION REPORT: (FOR FULL SUBTRACTOR)



RESULT:

Thus the verilog program for half subtractor and full subtractor were written, synthesized and simulated using Xilinx tool.

EXP: NO: 5	SIMULATION OF MULTIPLEXER AND DEMULTIPLEXER
DATE:	

AIM:

To write a verilog program for multiplexer and demultiplexer to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Terminate the program.

THEORY:

MULTIPLEXER

A Multiplexer is a Combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The set of selection of a particular line is controlled by selection lines. Normally there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.

The 4:1 MUX has four inputs I_0, I_1, I_2 and I_3 and select lines S_0 and S_1. The select lines s0 and s1 are decoded to select a particular AND gate. The outputs of the AND gates are applied to a single OR gate that provides the one line output Y.

DEMULTIPLEXER

A Demultiplexer is a Combinational circuit that selects binary information from one of input line and directs it to many output line. The set of selection of a particular output is controlled by selection lines. Normally there are 1 input line and 2^n selection lines whose bit combinations determine the output.

The 1:4 DEMUX has one input and select lines S_0 and S_1. The select lines s0 and s1 are decoded to select a particular AND gate. The outputs of the AND gates provides the various line output Y_1, Y_2, Y_3 and Y_4.

PROCEDURE: (SAME FOR BOTH MUX & DEMUX)

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output can be observed using model sim.

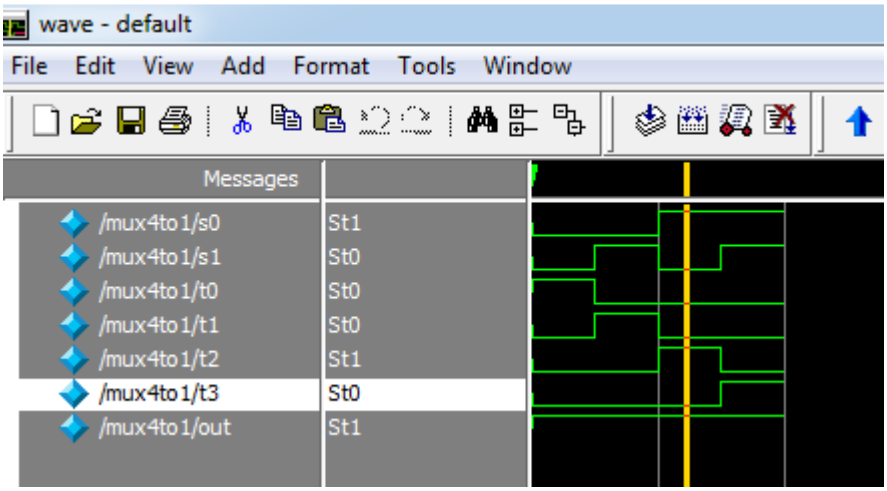
PROGRAM:

Verilog code for Multiplexer

```
module mux4to1(s0,s1,t0,t1,t2,t3,out);
input s0,s1,t0,t1,t2,t3;
output out;
wire s0n,s1n,n1,n2,n3,n4;
not(s0n,s0);
not(s1n,s1);
and(n1,s0n,s1n,t0);
```

```
and(n2,s0,s1n,t1);
and(n3,s0n,s1,t2);
and(n4,s0,s1,t3);
or(out,n1,n2,n3,n4);
endmodule
```

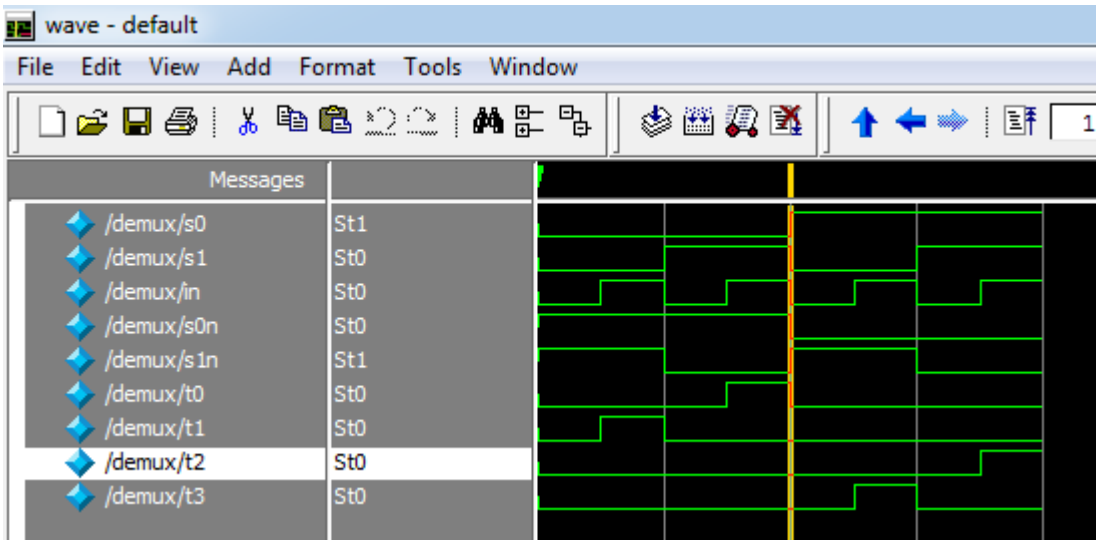
SIMULATION REPORT: (FOR MUX)



VERILOG CODE FOR DEMULTIPLEXER

```
module demux1to4(s0,s1,s0n,s1n,in,t0,t1,t2,t3);
input s0,s1,in;
output s0n,s1n,t0,t1,t2,t3;
wire s0n,s1n;
not(s0n,s0);
not(s1n,s1);
and(t0,s0n,s1,in);
and(t1,s0n,s1n,in);
and(t2,s0,s1,in);
and(t3,s0,s1n,in);
endmodule
```

SIMULATION REPORT: (FOR DEMUX)



RESULT:

Thus the verilog program for multiplexer and demultiplexer were written, synthesized and simulated using Xilinx tool.

EXP: NO: 6	SIMULATION OF ENCODER AND DECODER
DATE:	

AIM:

To write a verilog program for encoder and decoder to synthesize and simulate using Xilinx software tool**.**

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Terminate the program.

THEORY:

ENCODER

An Encoder is a digital circuit that has 2^n (or fewer) input lines and n output lines. The output lines generate the binary the binary code corresponding to the input value. In encoder it is assumed that only one input has a value of 1 at any given time**.**

DECODER

Discrete quantities of information are represented in digital systems by binary codes. A binary code of n bits is capable of representing up to 2^n distinct elements of coded information. A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. If the n bit coded information unused combinations. The decoder may have fewer than 2^n outputs.

The decoder are also called ‘n’ to ‘m’ line decoders, where is less than or equal to 2^n. Their purpose is to generate the 2^n (or fewer) minterms of input variables. The name decoder is also used in conjunction with other code converters such as BCD to SEVEN SEGMENT decoder.

PROCEDURE: (FOR ENCODER & DECODER)

Software part

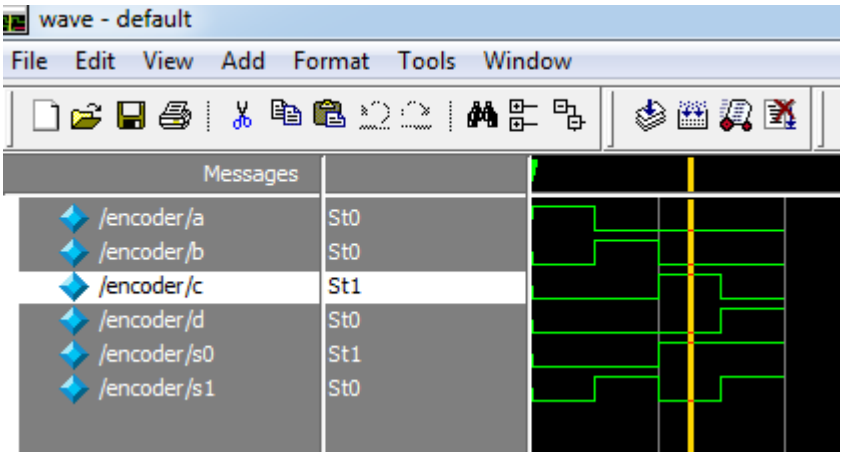
1. Click on the Xilinx ISE Design Suite 12.1 or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output can be observed using model sim.

PROGRAM:

Verilog code for Encoder

```
module encoder(a,b,c,d,s0,s1);
input a,b,c,d;
output s0,s1;
or(s0,c,d);
or(s1,b,d);
endmodule
```

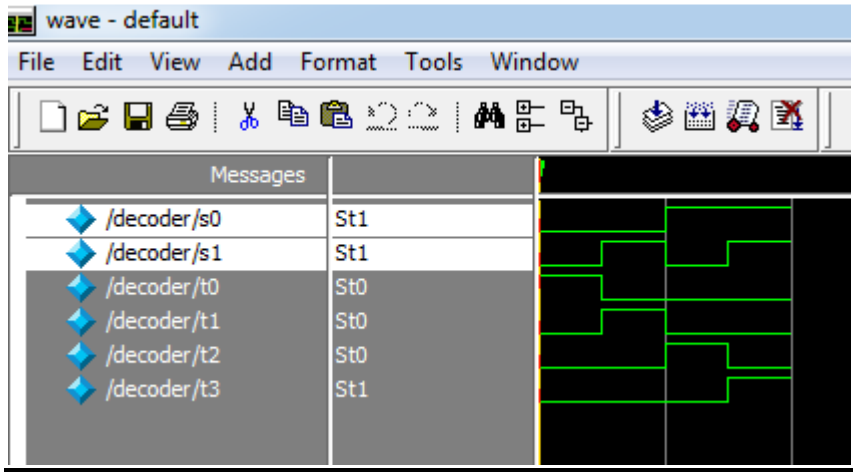
SIMULATION REPORT: (FOR ENCODER)



VERILOG CODE FOR DECODEER

```
module decoder(s0,s1,t0,t1,t2,t3);
input s0,s1;
output t0,t1,t2,t3;
wire s0n,s1n;
not(s0n,s0);
not(s1n,s1);
and(t0,s0n,s1n);
and(t1,s0n,s1);
and(t2,s0,s1n);
and(t3,s0,s1);
endmodule
```

SIMULATION REPORT: (FOR DECODER)



RESULT:

Thus the verilog program for encoder and decoder were written, synthesized and simulated using Xilinx tool.

EXP: NO: 7	SIMULATION OF ARITHMATIC LOGIC UNIT
DATE:	

AIM:

To write a verilog program for ALU to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Terminate the program.

THEORY:

It is a digital circuit that performs arithmetic and logical operations. The ALU is a fundamental building block of the CPU of a computer and even the simplest microprocessors contain of for purposes such as maintaining timers. The processors found inside modern CPU's and graphic processing units accommodate very powerful and very complex ALU's.

PROCEDURE:

Software part

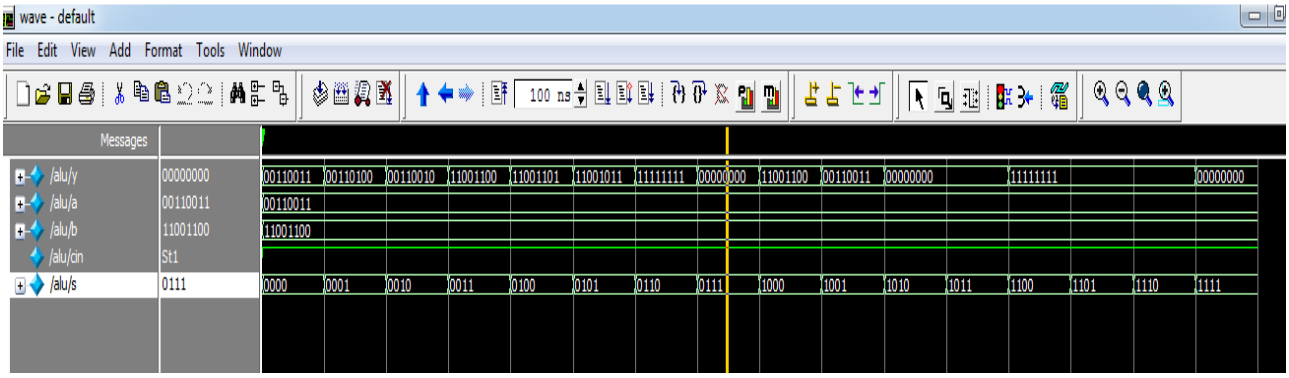
1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output can be observed using model sim.

PROGRAM:

Verilog code for ALU

```
module alu(y,a,b,cin,s);
output [7:0]y;
input [7:0]a,b;
input cin;
input [3:0]s;
reg [7:0]y;
always@(a,b,s)
begin
case(s)
4'b0000:y=a;
4'b0001:y=a+1;
4'b0010:y=a-1;
4'b0011:y=b;
4'b0100:y=b+1;
4'b0101:y=b-1;
4'b0110:y=a+b;
4'b0111:y=a+b+cin;
4'b1000:y=~a;
4'b1001:y=~b;
4'b1010:y=a&b;
4'b1011:y=a/b;
4'b1100:y=~(a&b);
4'b1101:y=~(a/b);
4'b1110:y=a^b;
4'b1111:y=~(a^b);
endcase
end
endmodule
```

SIMULATION REPORT:



RESULT:

Thus the verilog program for ALU were written, synthesized and simulated using Xilinx tool.

EXP: NO: 8	SIMULATION OF 8- BIT ADDER
DATE:	

AIM:

To write a verilog program for 8-Bit Adder to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements for Verilog code.
5. Write the functionality of the gates.
6. Terminate the program.

THEORY :

When you add large numbers carefully together the addition is done digit by digit. The comwter does the same. In the illustration, two 8 –digit binary numbers are being added. The top row contains the first number and the second row the other. Working from the right-hand side, there can be no 'carry' to acd to the sum of the first two digits, so a half adder is sufficient. But for the second and subsequent pairs of digits, full adders must be usee (any carry' is indicated by a f below the adder). The output will be an 8-bit and if the carry is formed that will be shown in cout output value.

PROCEDURE:

Software part

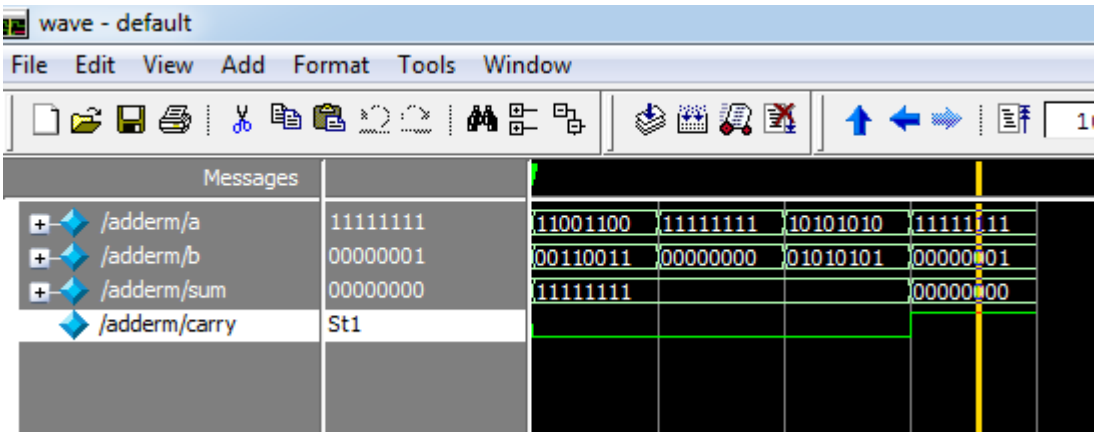
1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double
4. clicking on the synthesis in the process window.
5. Perform the functional simulation using Xilinx ISE simulator.
6. The output wave form can be observed in model sim.

PROGRAM:

Verilog Code for 8 BIT ADDER

```
module adderm(a, b, sum, carry);
input [7:0] a;
input [7:0] b;
output [7:0] sum;
output carry;
wire[8:0]temp;
assign temp=(a+b);
assign sum=temp[7:0];
assign carry=temp[8];
endmodule
```

SIMULATION REPORT :



RESULT:

Thus the verilog program for 8-Bit Adder were written, synthesized and simulated using Xilinx tool.

EXP:NO: 9	SIMULATION OF 5-BIT MULTIPLIER
DATE:	

AIM:

To write a verilog program for 4-Bit multiplier to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Terminate the program.

THEORY:

Multiplication of two elements in the polynomial basis can be accomplished in the normal way of multiplication, but there are a number of ways to speed up muultipication, especially in hardware. In this type the multiplication can done parallel counter and it is generate carry. The multiplication is independent of the carry so we can perform N number of multiplication independent of carry.

PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.

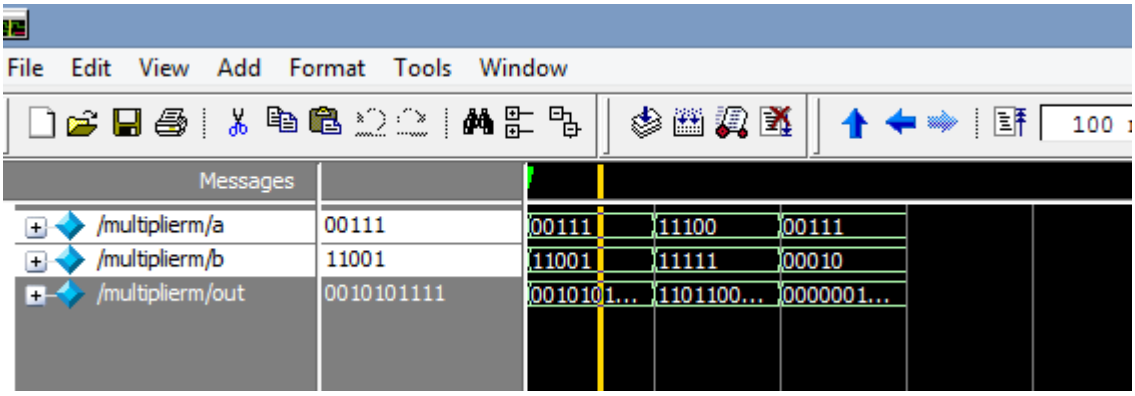
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double
4. clicking on the synthesis in the process window.
5. Perform the functional simulation using Xilinx ISE simulator.
6. The output wave form can be observed in model sim.

PROGRAM:

Verilog code for 4-Bit multiplier

```
module multiplierm(a, b, out);
input [4:0] a;
input [4:0] b;
output [9:0] out;
assign out=(a*b);
endmodule
```

SIMULATION REPORT:



RESULT:

Thus the verilog program for 4-Bit multiplier were written, synthesized and simulated using Xilinx tool.

EXP: NO: 10	SIMULATION OF D FLIP FLOP AND D LATCH
DATE:	

AIM:

To write a verilog program for D-flip flop and D-latch to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Terminate the program.

THEORY:

D-FLIP FLOP:

It has only a single data input. That data input is connected to the S input of RS-flip flop, while the inverse of D is connected to the R input. This prevents that the input combination ever occurs. To allow the flip flop to be in holding state, a D-flip flop has a second input called "clock". The clock input is AND-ed with the D input, such that when clock=0, the R and S inputs of the RS-flip flop are 0 and the state is held.

D-LATCH:

It has only a single data input. That data input is connected to the S input of RS-flip flop, while the inverse of D is connected to the R input. This prevents that the input combination ever occurs. To allow the flip flop to be in holding state, a D-flip flop has a second input called "enable".

The enable input is AND-ed with the D input, such that when enable=0, the R and S inputs of the RS-flip flop are 0 and the state is held.

PROCEDURE:

Software part:

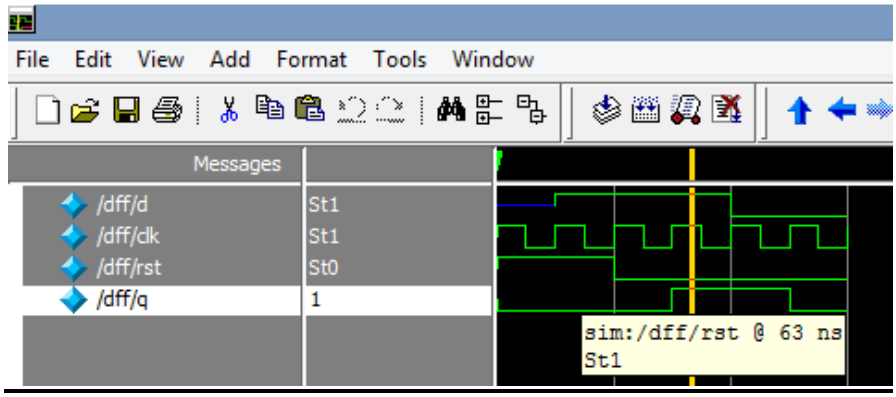
1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output can be observed using model sim.

PROGRAM:

Verilog code for D flipflop

```
module dff(d,clk,rst,q);
input d,clk,rst;
output q;
reg q;
always@(posedge clk or posedge rst)
begin
   if (rst)
   q<=1'b0;
   else
   if(clk)
   q<=d;
end
endmodule
```

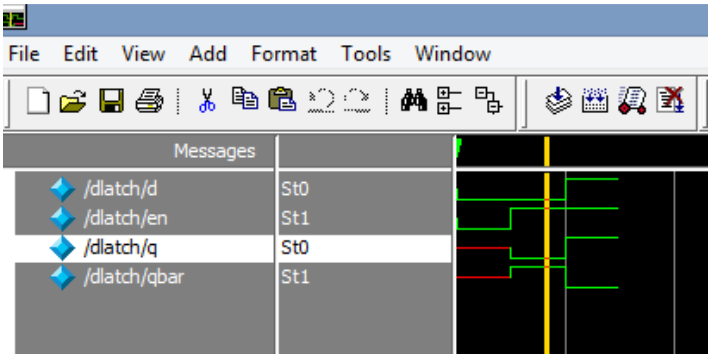
SIMULATION REPORT:



Verilog code for D latch

```
module dlatch(d,en,b,c,a,q,qbar);
input d,en;
output b,c,a,q,qbar;
wire b,c,a,q,qbar;
not(a,d);
nand(b,d,en);
nand(c,a,en);
nand(q,b,qbar);
nand(qbar,c,q);
endmodule
```

SIMULATION REPORT:



RESULT:

Thus the verilog program for D-flip flop and D-latch were written, synthesized and simulated using Xilinx tool.

EXP:NO: 11 DATE:	SIMULATION OF PSEUDO RANDOM BINARY SEQUENCE

AIM:

To write a verilog program for Pseudo Random Binary Sequence to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements for Verilog code.
5. Write the functionality of the gates.
6. Terminate the program.

THEORY :

PRBS :

A PRBS is 'pseudorandom', because, although it is in fact deterministic, it seems to be random in a sense that the value of an element is independent of the values of any of the other elements, similar to real random sequences.

A PRBS can be stretched to infinity by repeating it after elements, this in contrast to most random sequences, such as sequences generated by radioactive decay or by white noISE, that are 'infinite' by nature. The PRBS is more general than the maximum length sequence, which is a special pseudo-random binary sequence of N bits generated as the output of a linear shift register. A maximum length sequence always has a 1/2 duty cycle and its number of elements .

PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output wave form can be observed in model sim.

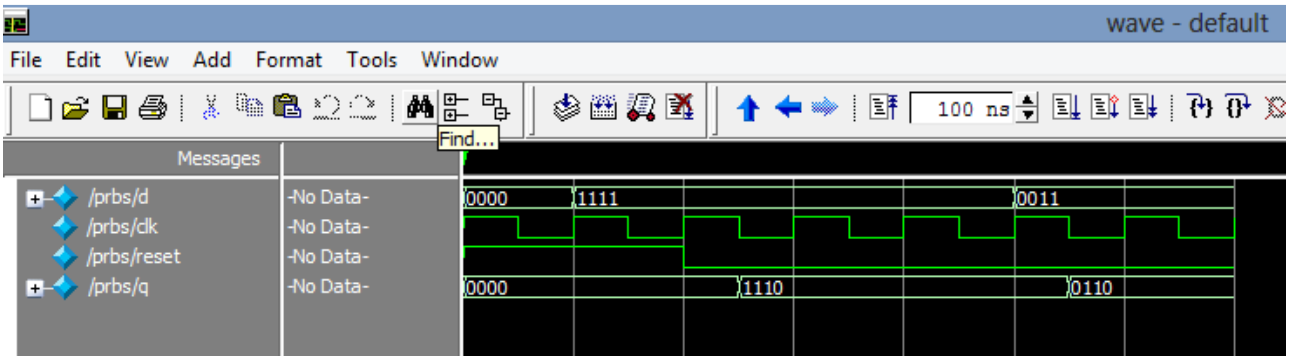
PROGRAM:

Verilog Code for Pseudo Random Binary Sequence

```
module prbs(d, clk, reset, q);
input [3:0] d;
input clk;
input reset;
output [3:0] q;
reg [3:0]q;
reg [3:0]temp;
always @(clk)
begin
if(reset)
q=4'b0000;
else
begin
temp=d;
temp[3]=temp[0]^temp[3];
q=temp<<1;
end
end
```

endmodule

SIMULATION REPORT :



RESULT:

Thus the verilog program for Pseudo Random Binary Sequence were written, synthesized and simulated using Xilinx tool.

EXP: NO: 12	SIMULATION OF ACCUMULATOR
DATE:	

AIM:

To write a verilog program for Accumulator to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements for Verilog code.
5. Write the functionality of the gates.
6. Terminate the program.

THEORY :

ACCUMULATOR:

An accumulator is a register in which intermediate arithmetic and logic results are stored. Without a register like an accumulator, it would be necessary to write the result of each calculation (addition, multiplication, shift, etc.) To main memory, perhaps only to be read right back again for use in the next operation. Access to main memory is slower than access to a register like the accumulator because the technology used for the large main memory is slower (but cheaper) than that used for a register.

The canonical example for accumulator use is summing a list of numbers. The accumulator is initially set to zero, then each number in turn is read and added to the value in the accumulator.

Only when all numbers have been added is the result held in the accumulator written to main memory or to another, non-accumulator, CPU register.

PROGRAM:

Verilog Code for Accumulator

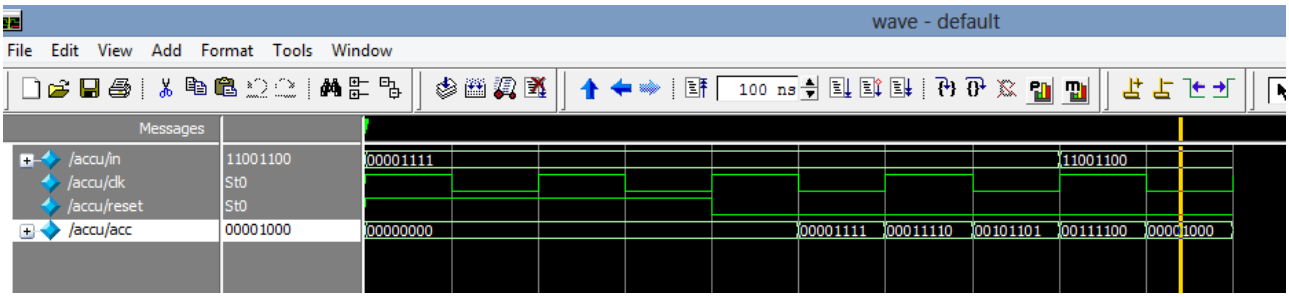
```
module accu (in, acc, clk, reset);
input [7:0] in;
input clk, reset;
output [7:0] acc;
reg [7:0] acc;
always@(clk) begin
if(reset)
acc <= 8'b00000000;
else
acc <= acc + in;
end
endmodule
```

PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output wave form can be observed in model sim.

SIMULATION REPORT :



RESULT:

Thus the verilog program for Accumulator were written, synthesized and simulated using Xilinx tool.

EXP: NO: 13	SIMULATION OF UP-DOWN COUNTER
DATE:	

AIM:

To write a verilog program for Up-Down Counter to synthesize and simulate using Xilinx software tool.

TOOLS REQUIRED:

Software:

1. Xilinx ISE Design Suite 12.1

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements for Verilog code.
5. Write the functionality of the gates.
6. Terminate the program.

THEORY :

UP-DOWN COUNTER :

A counter that can change state in either direction, under the control of an up or down selector input, is known as an up/down counter. When the selector is in the up state, the counter increments its value. When the selector is in the down state, the counter decrements the count. Likewise the counter counts in both the directions continuously until attaining the end of the count. The count is initiated by the positive clock pulse. The counter counts from 0000 to 1111 for up count and 1111 to 0000 for down count.

PROCEDURE:

Software part

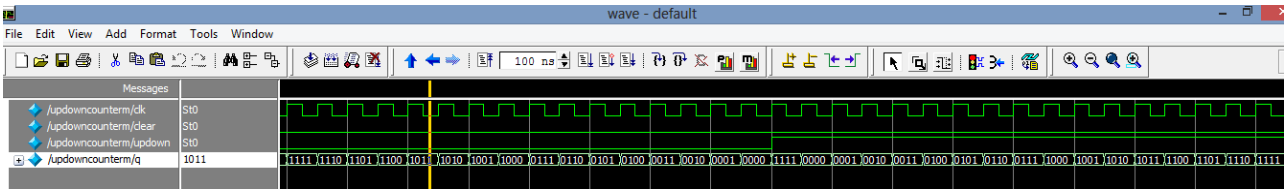
1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. The output wave form can be observed in model sim.

PROGRAM:

Verilog Code for Up-Down Counter

```
module updowncounterm(clk, clear, updown, q);
input clk;
input clear;
input updown;
output [3:0] q;
reg [3:0] q;
always@(posedge clear or posedge clk)
begin
if(clear)
q <=4'b0000;
else if(updown)
q <= q+1'b1;
else
q <= q-1'b1;
end
endmodule
```

SIMULATION REPORT :



RESULT:

Thus the verilog program for Up-Down Counte were written, synthesized and simulated using Xilinx tool.

EXP: NO: 14	STUDY OF IMPLEMENTATION IN FPGA
DATE:	

AIM:

To study the implementation of programs in FPGA.

Eg: AND gate

TOOLS REQUIRED:

SOFTWARE REQUIRED:

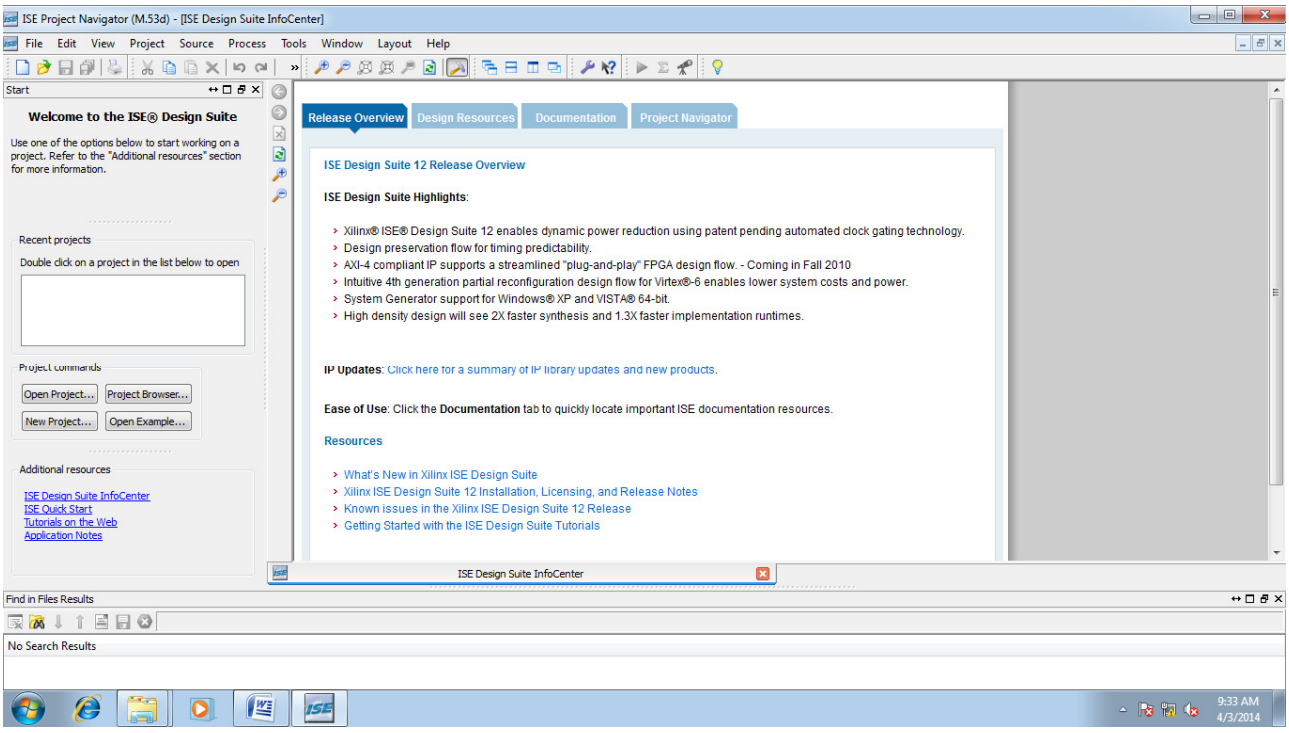
1. Xilinx ISE Design Suite 12.1

HARDWARE REQUIRED:

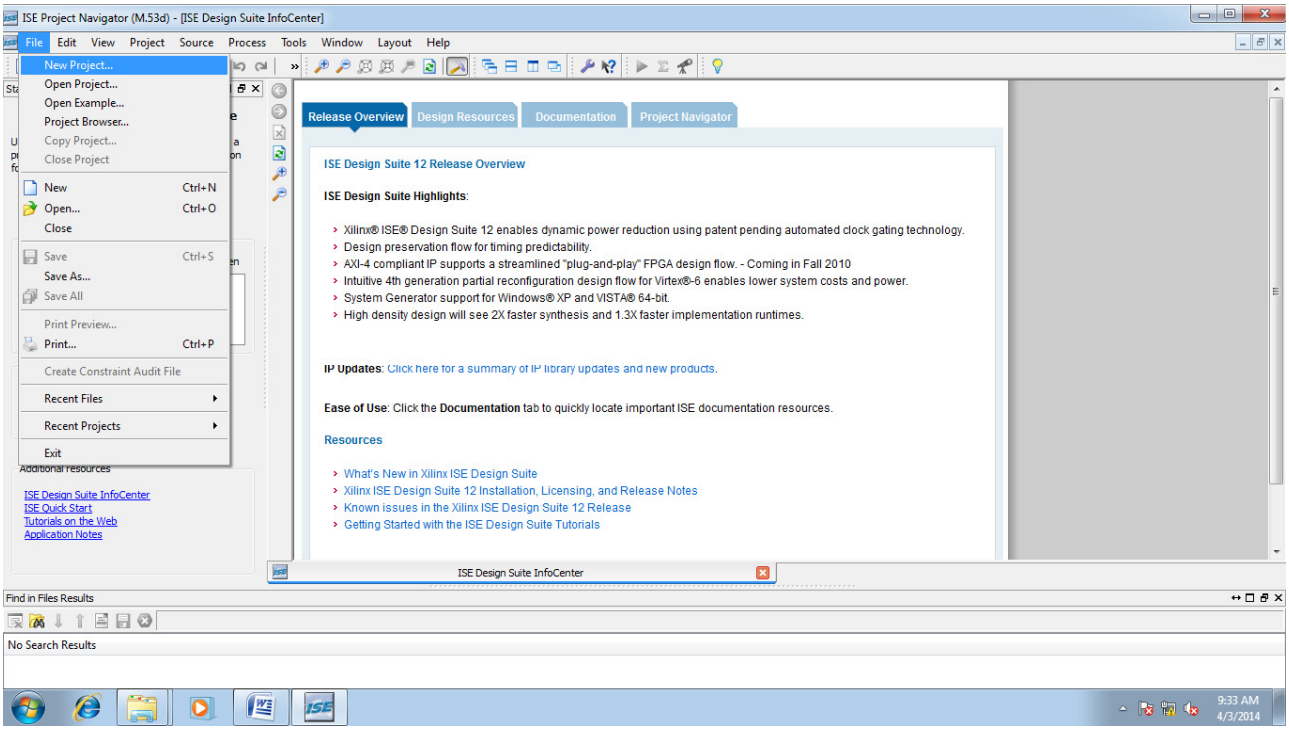
1. XILINX SPARTAN3E FPGA kit.

PROCEDURE:

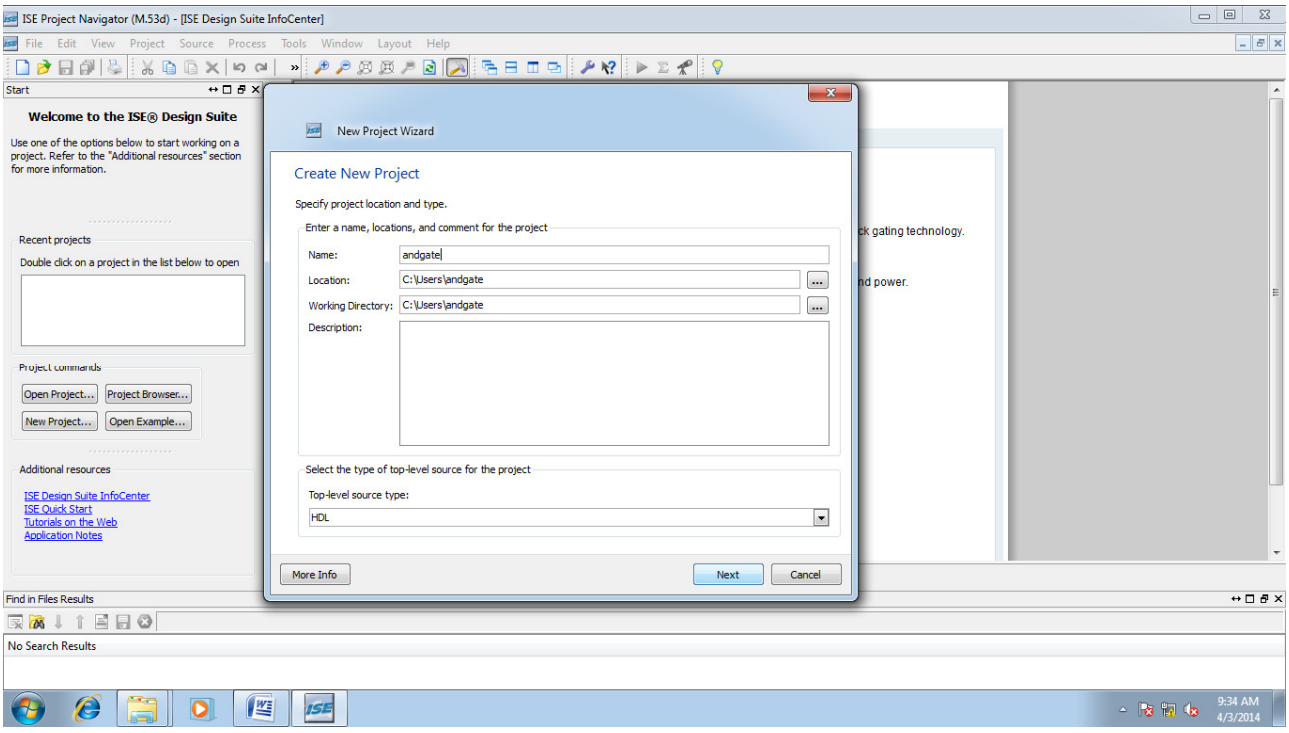
1. **Now start the Xilinx ISE Design Suite 12.1**



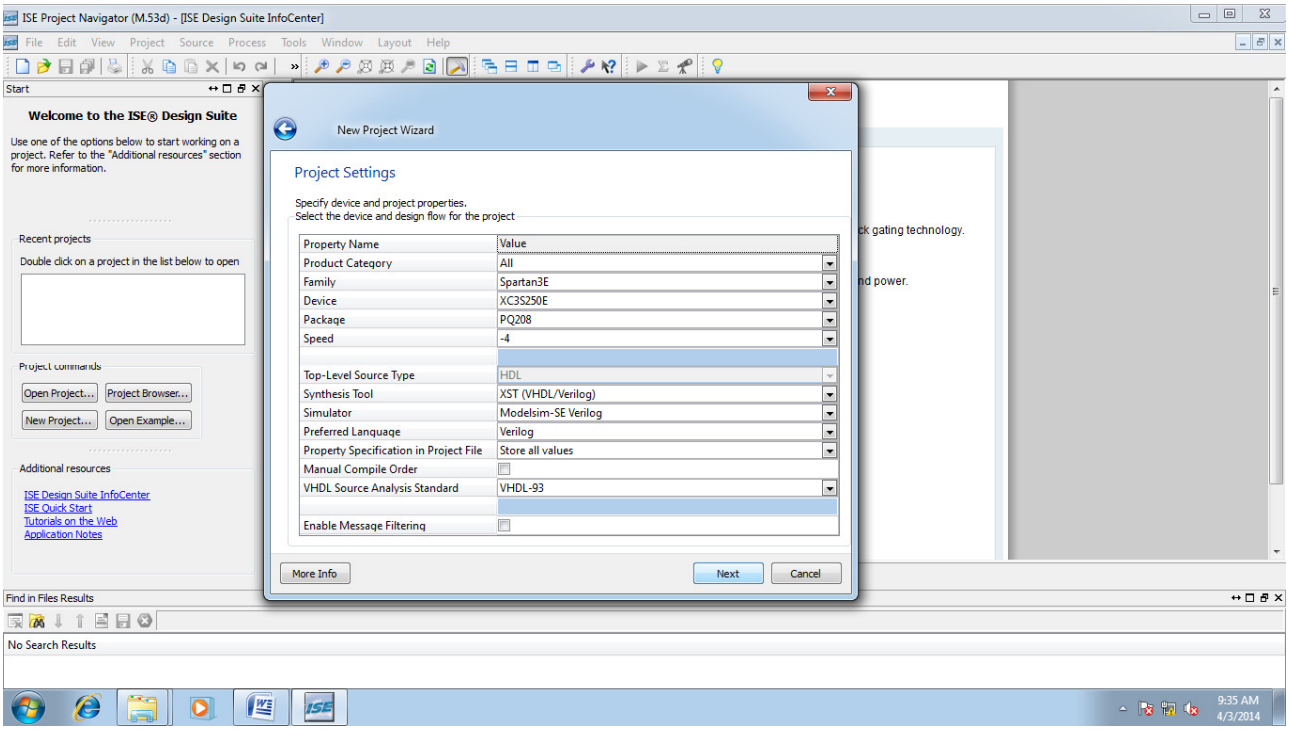
2. **Go to file and click new project**



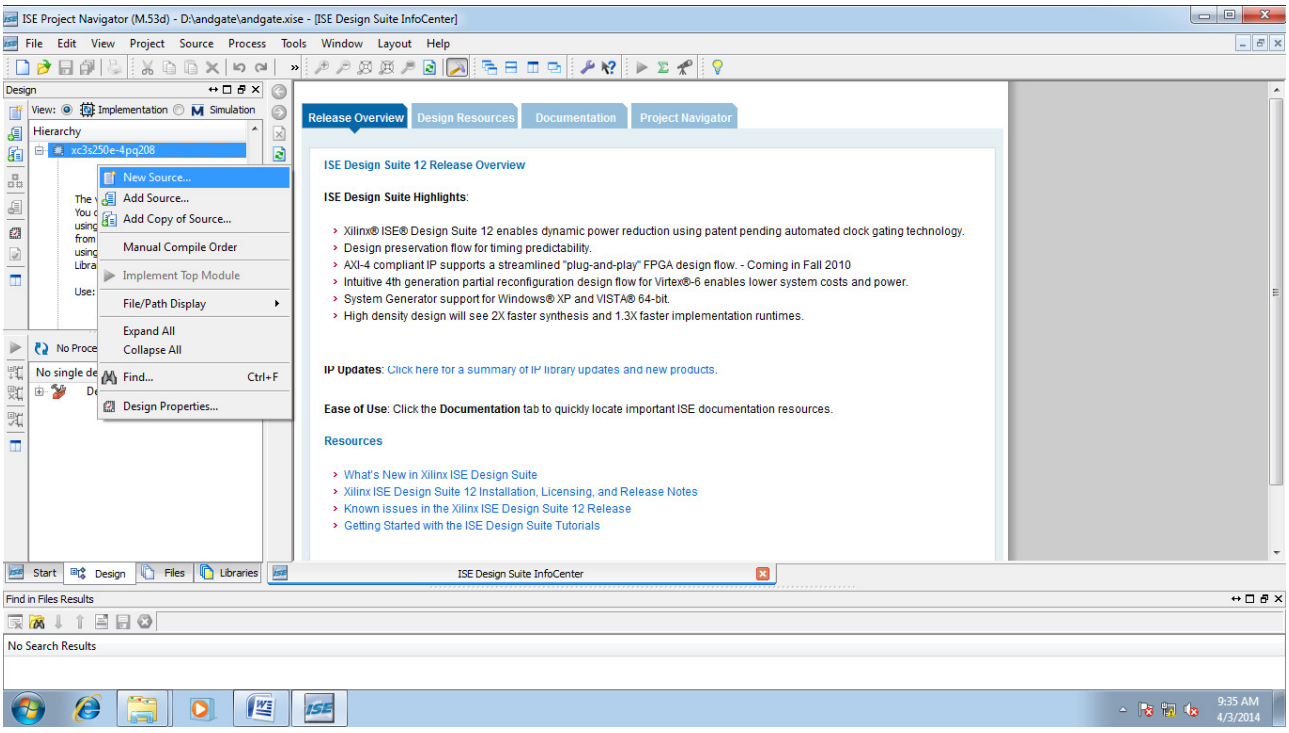
3. **Enter the project name and click next**



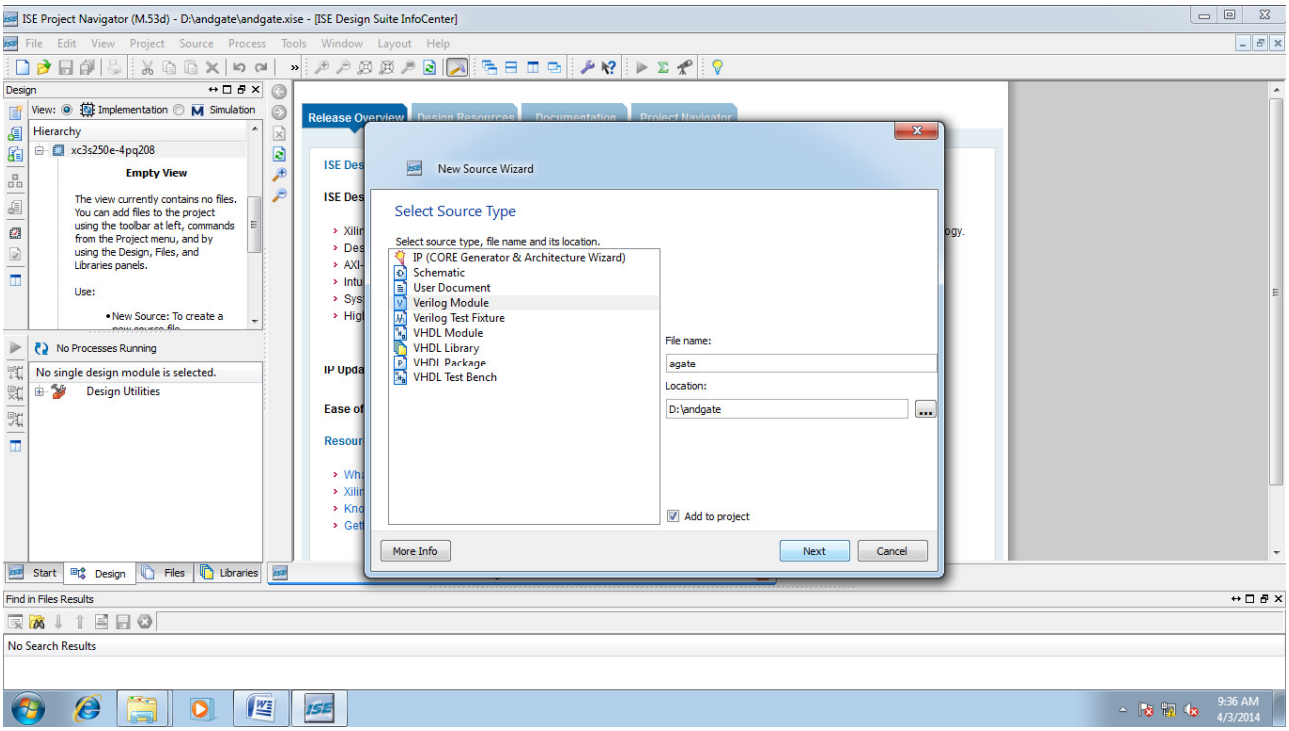
4. **Select the family name is Spartan 3E, speed is -4 and simulator is verilog click next.**



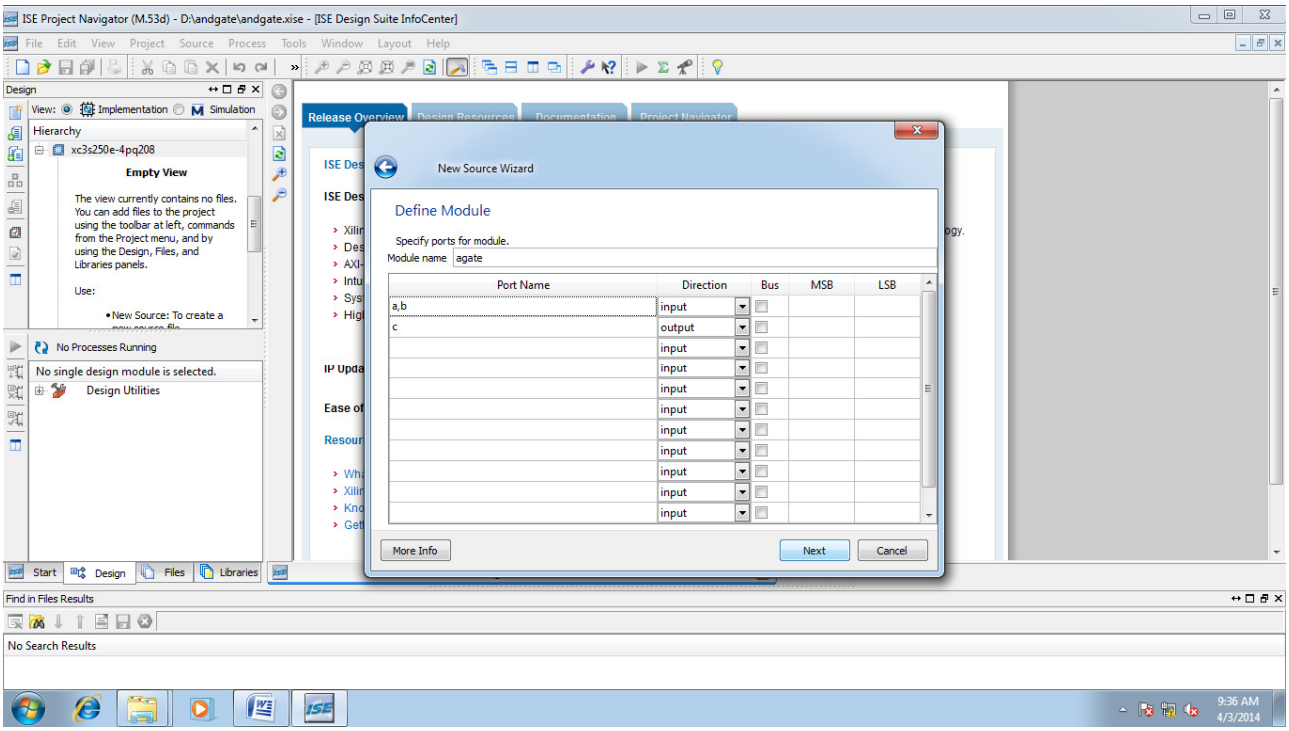
5. **Click new source.**



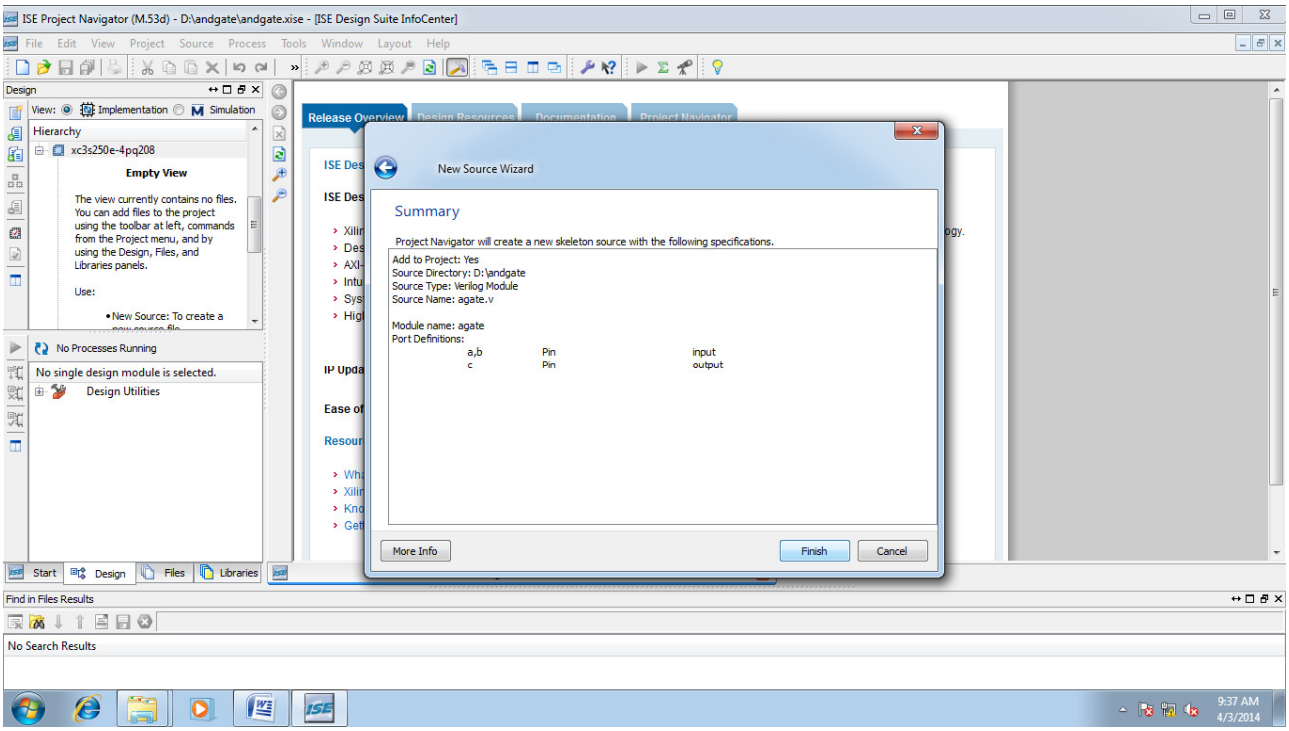
6. **Select verilog module and type file name and click next.**



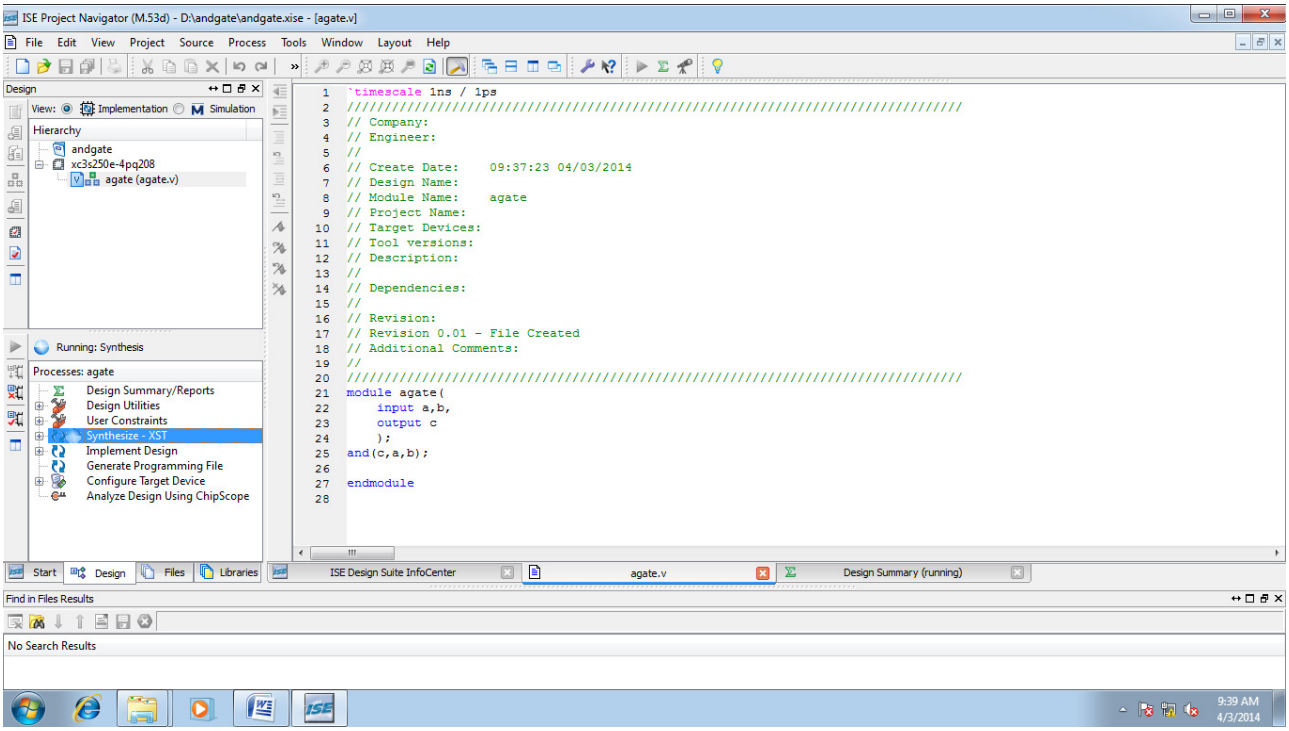
7. **Assign input and output port and click next.**



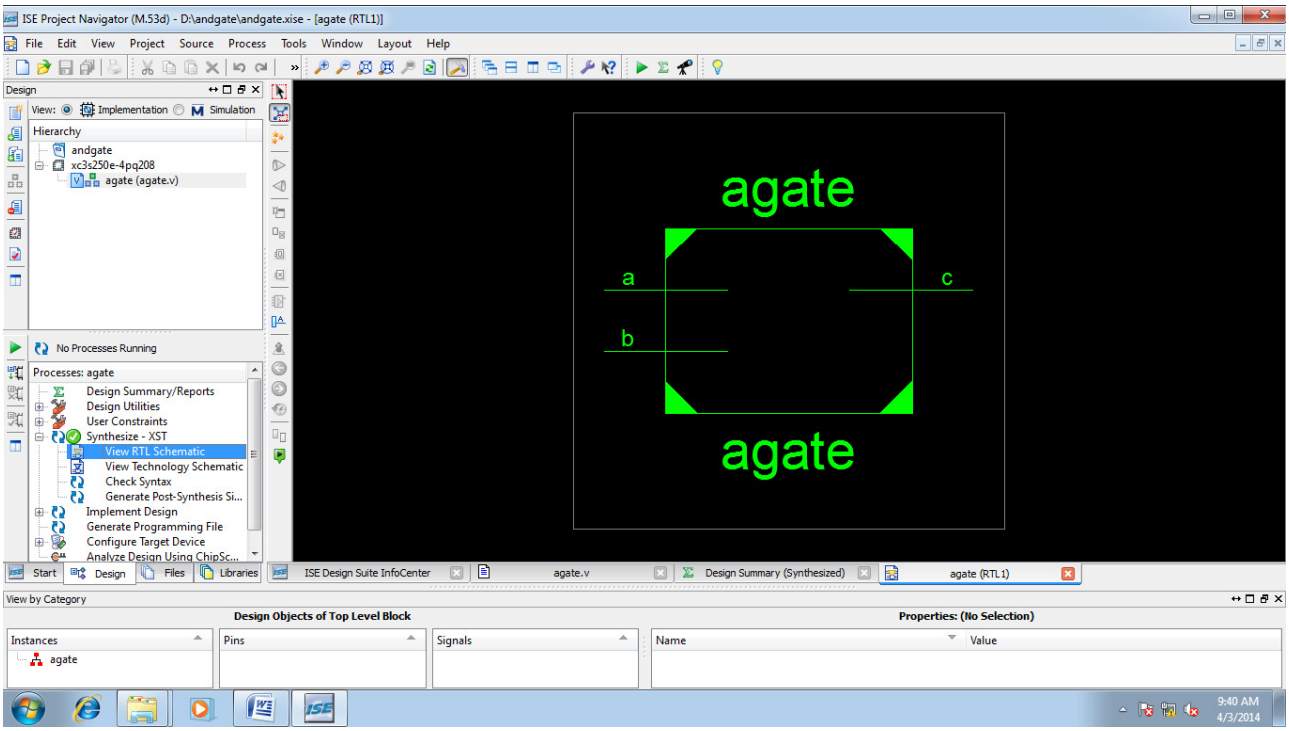
8. **Finally the report is shown click finish and give yes.**



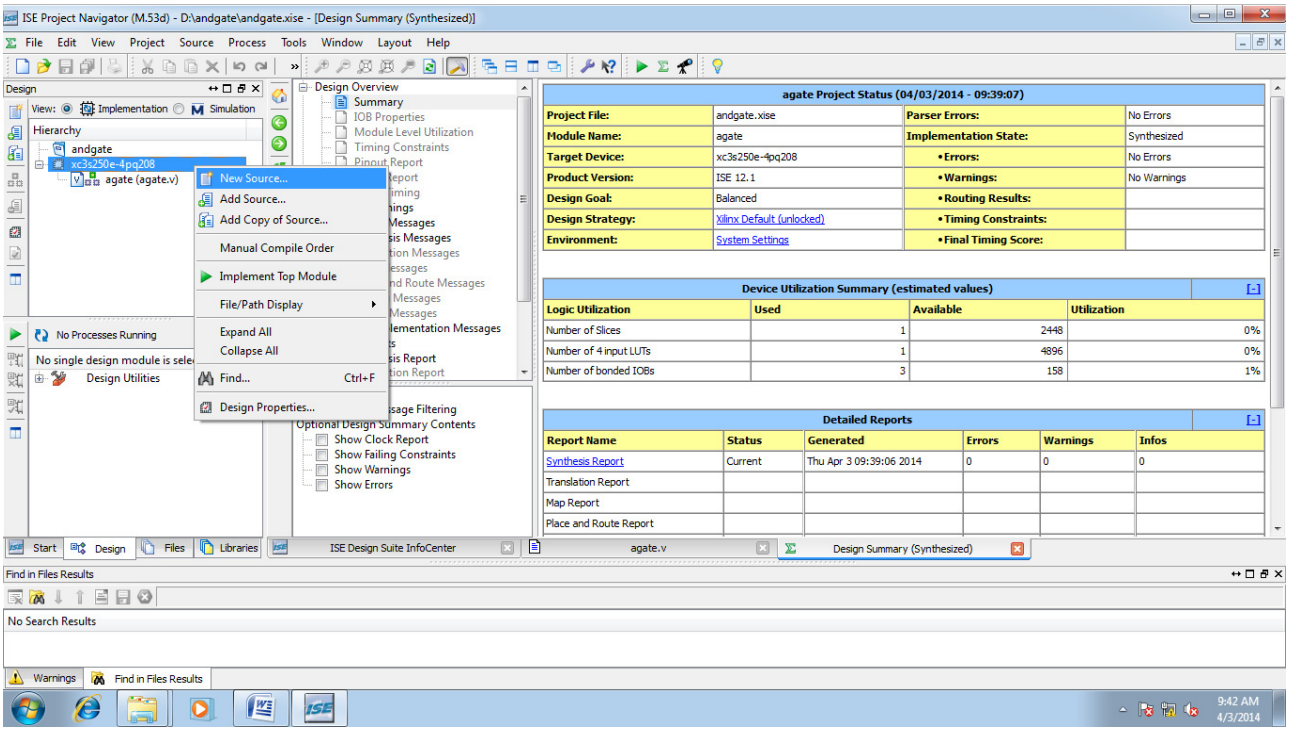
9. **Type the program, save and click synthesis.**



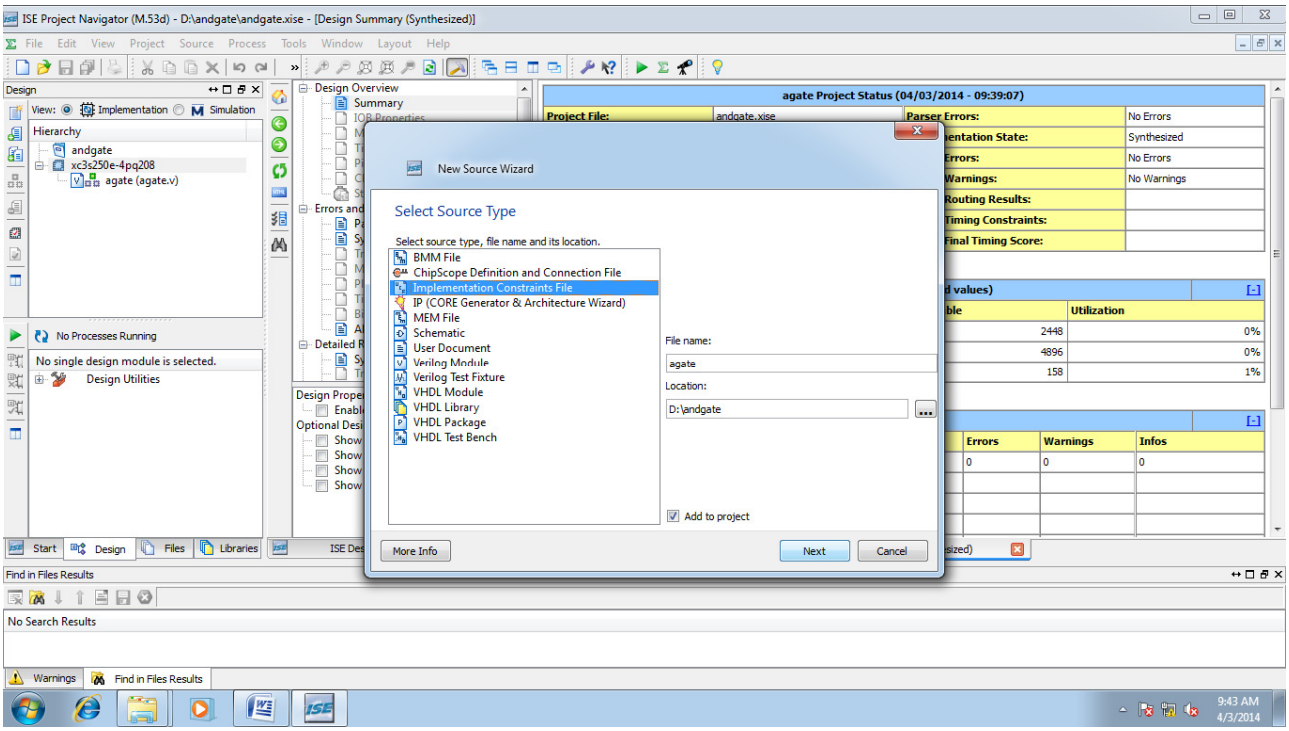
10. Select RTL schematic and view that.



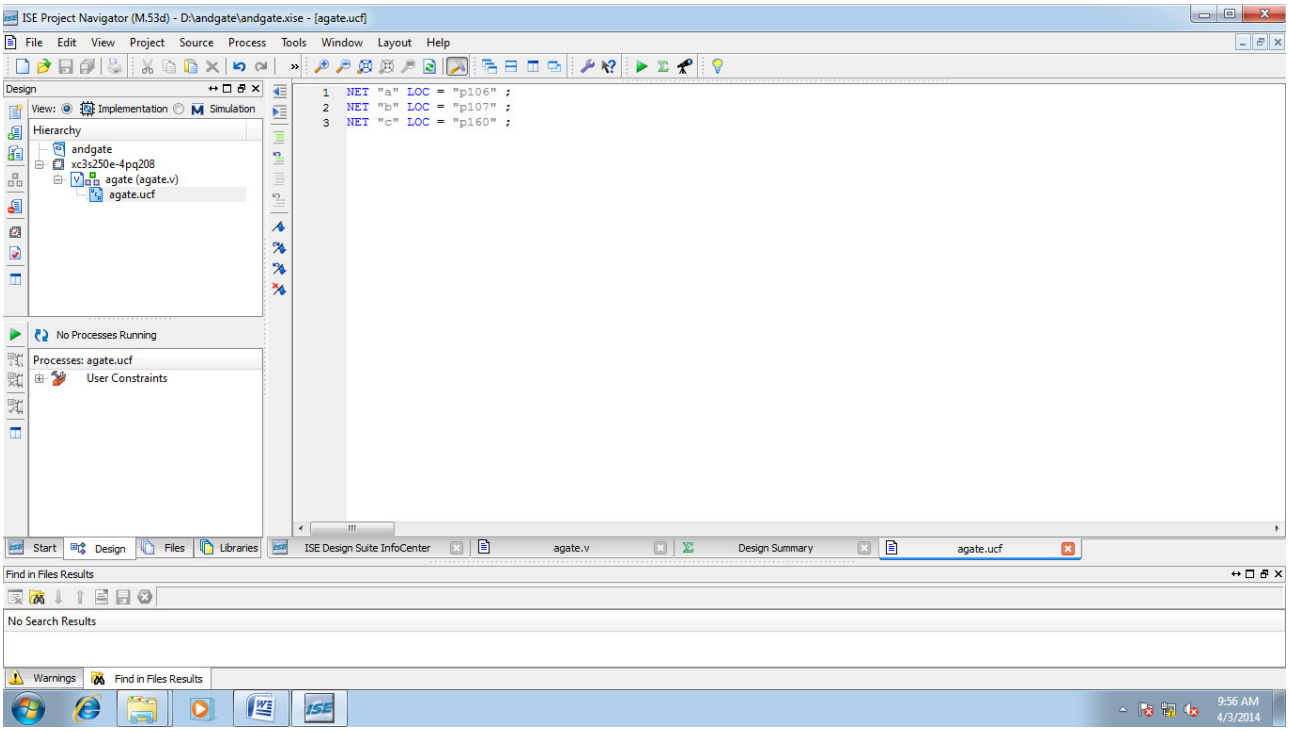
11. Right click on the file name and select New Source



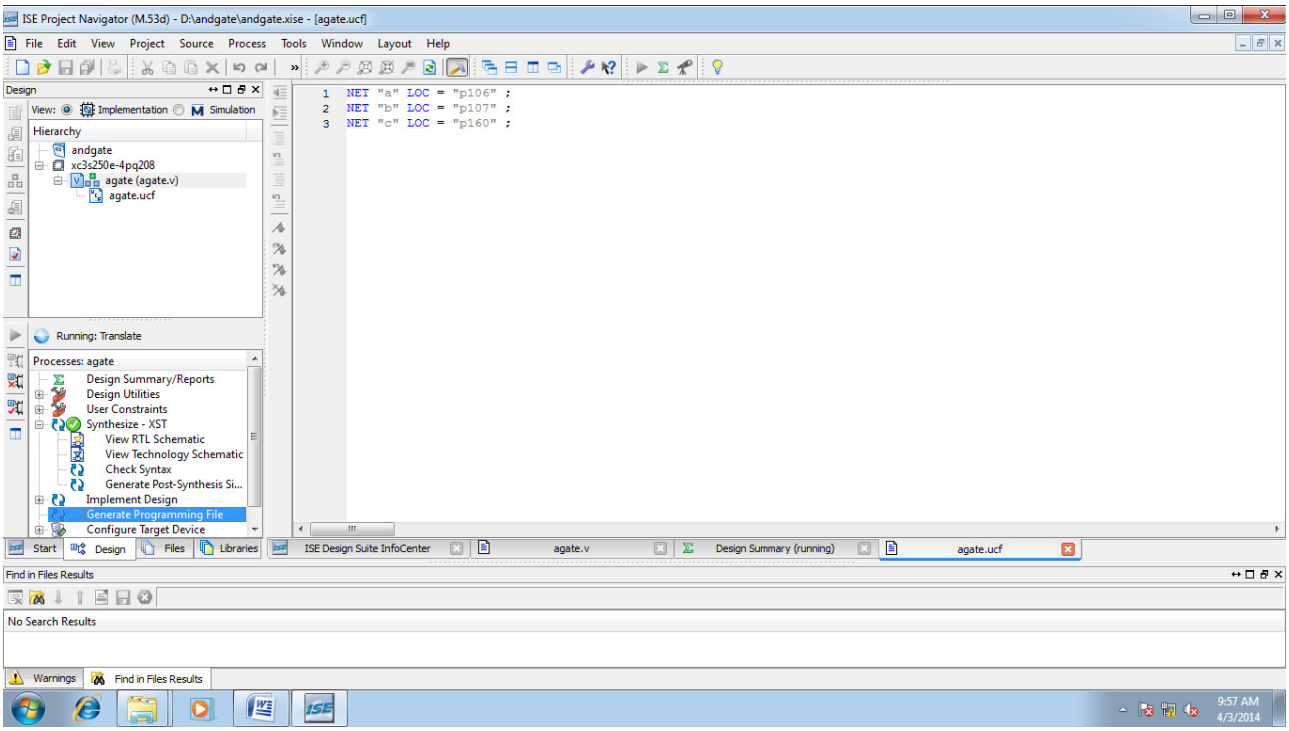
12. Select Implementation Constraints File and type file name and click next.



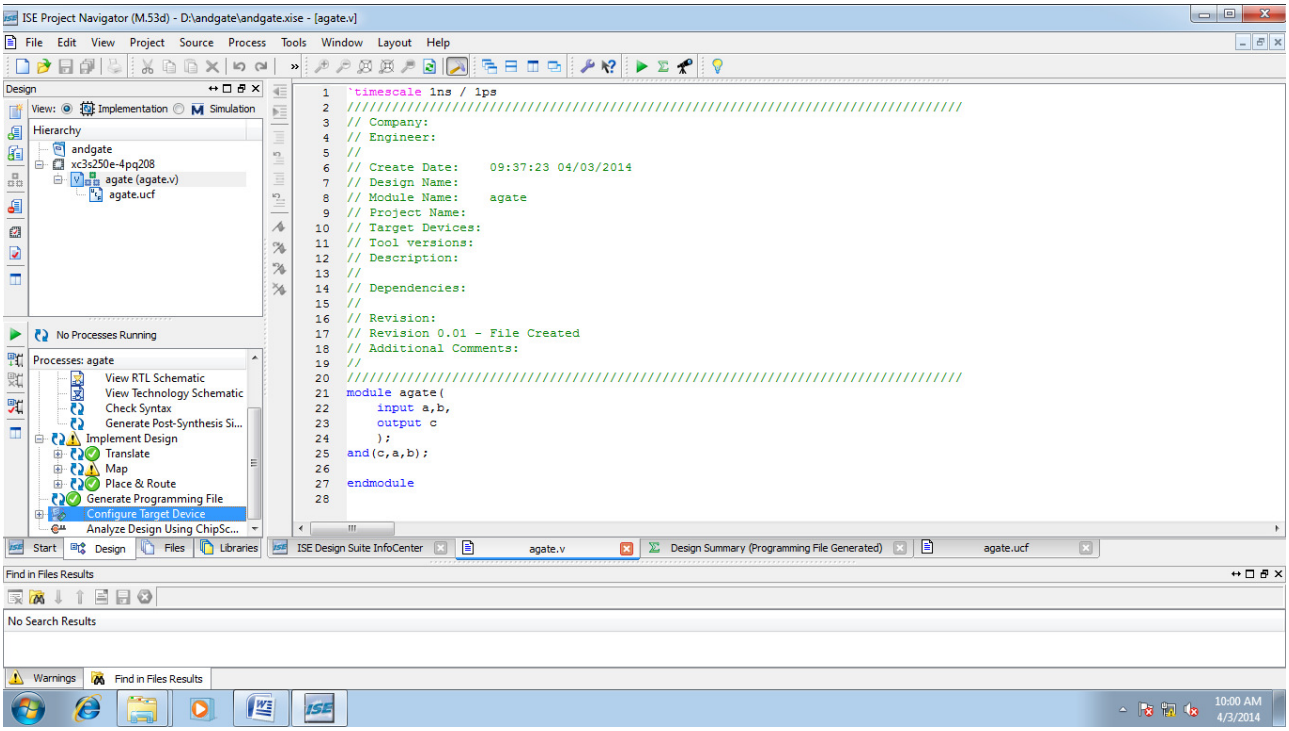
13. Type the Netlist and click save



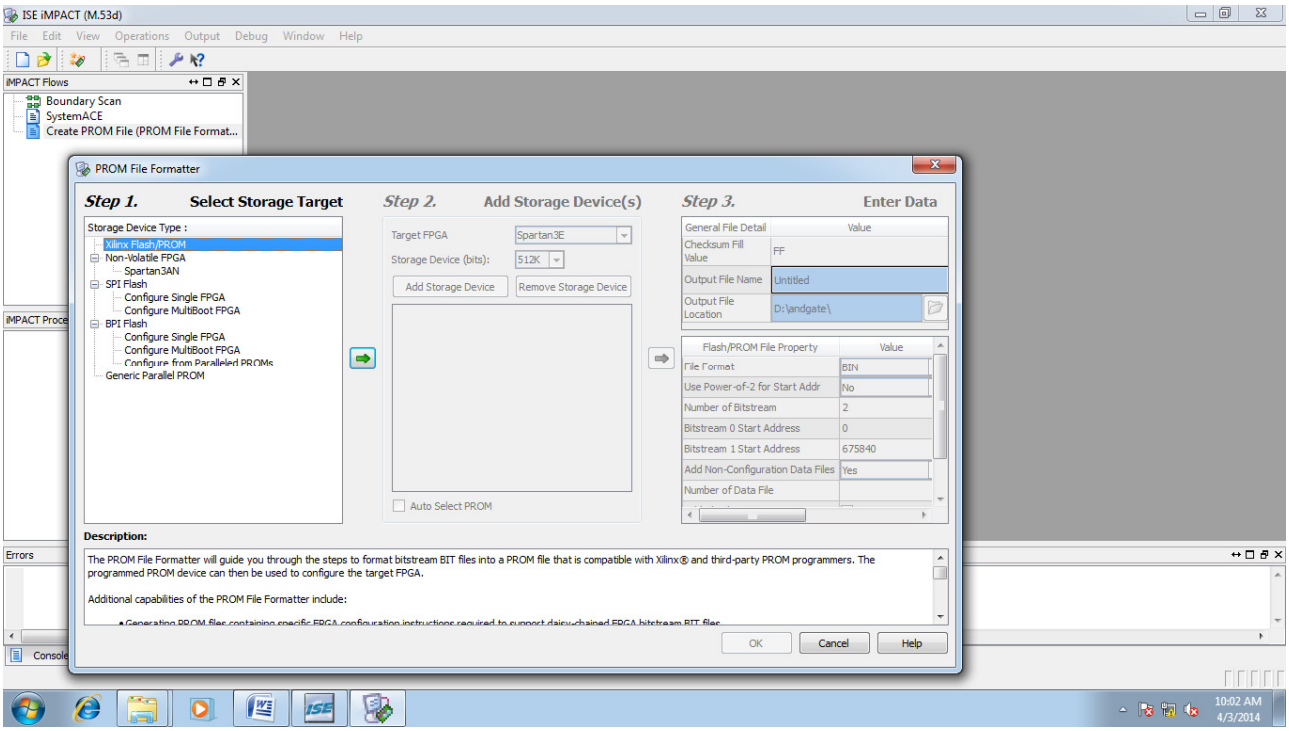
14. Then double click Generate Programming File.



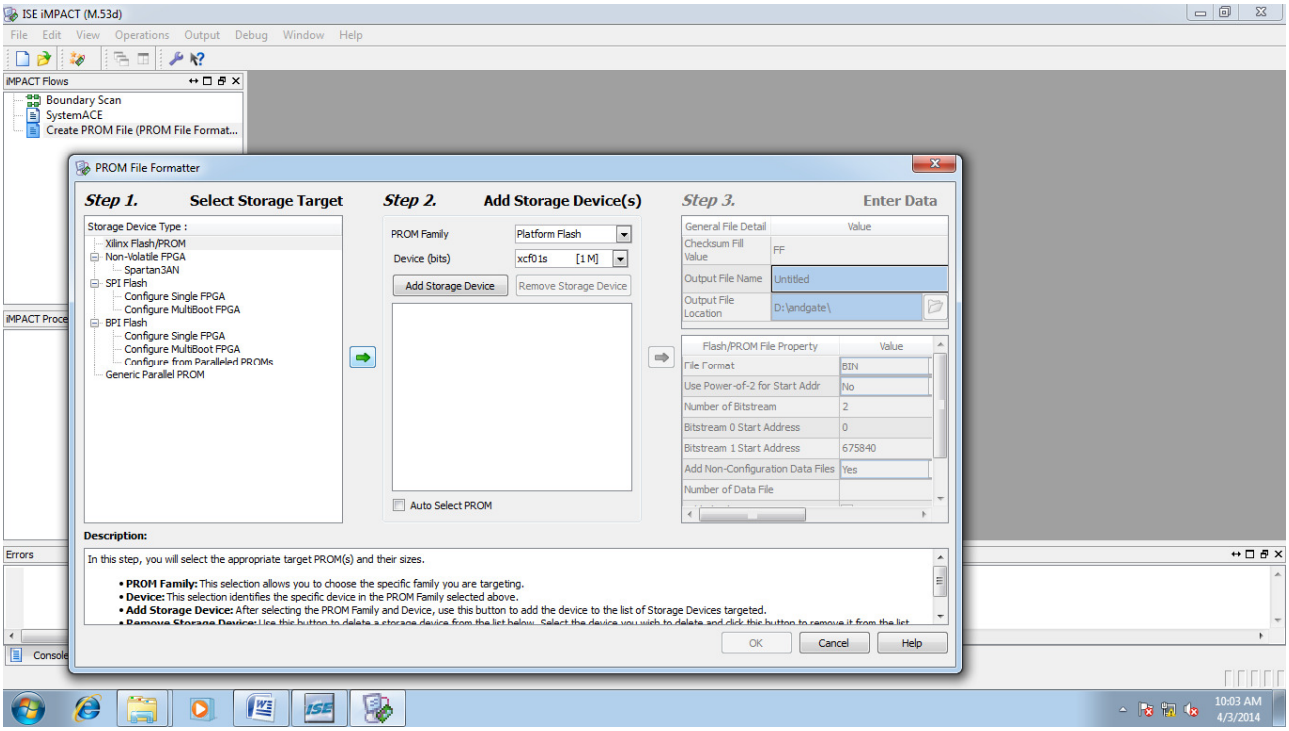
15. Double click Configure Target Device and click OK.



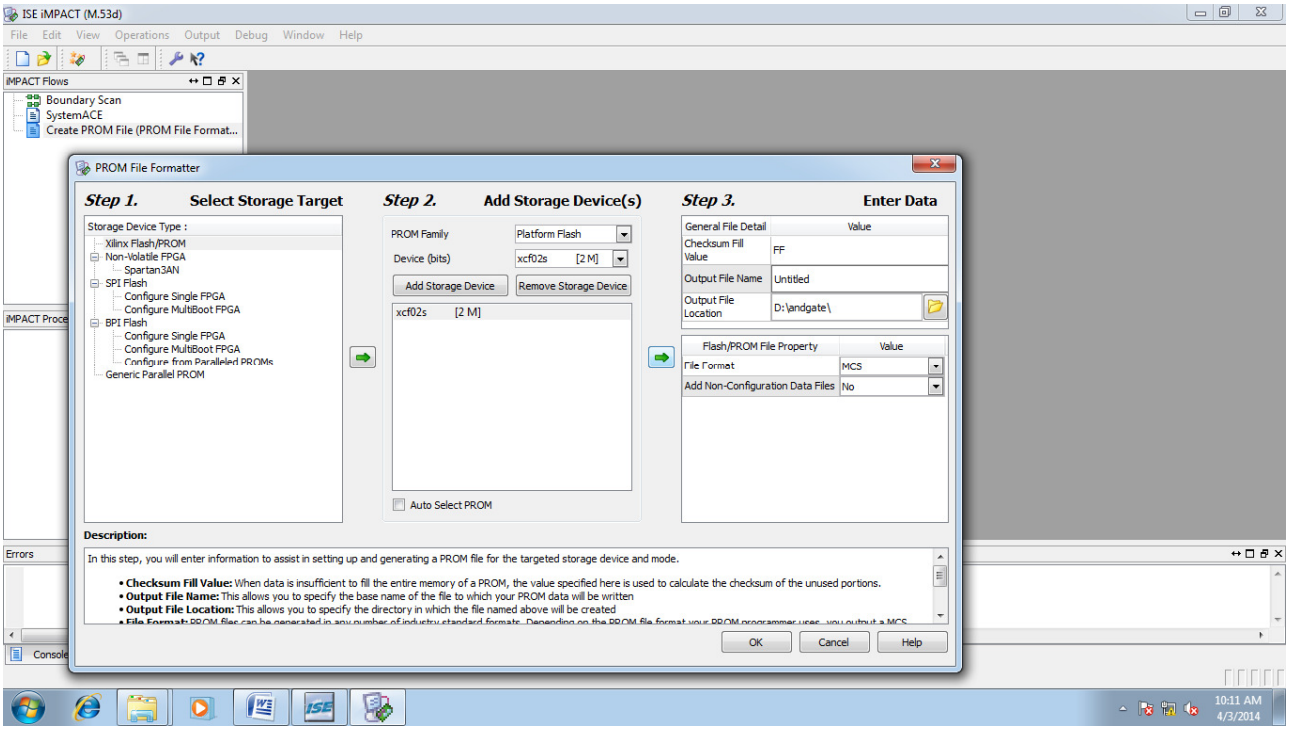
16. Double click Create PROM File in the ISE iMPACT window.



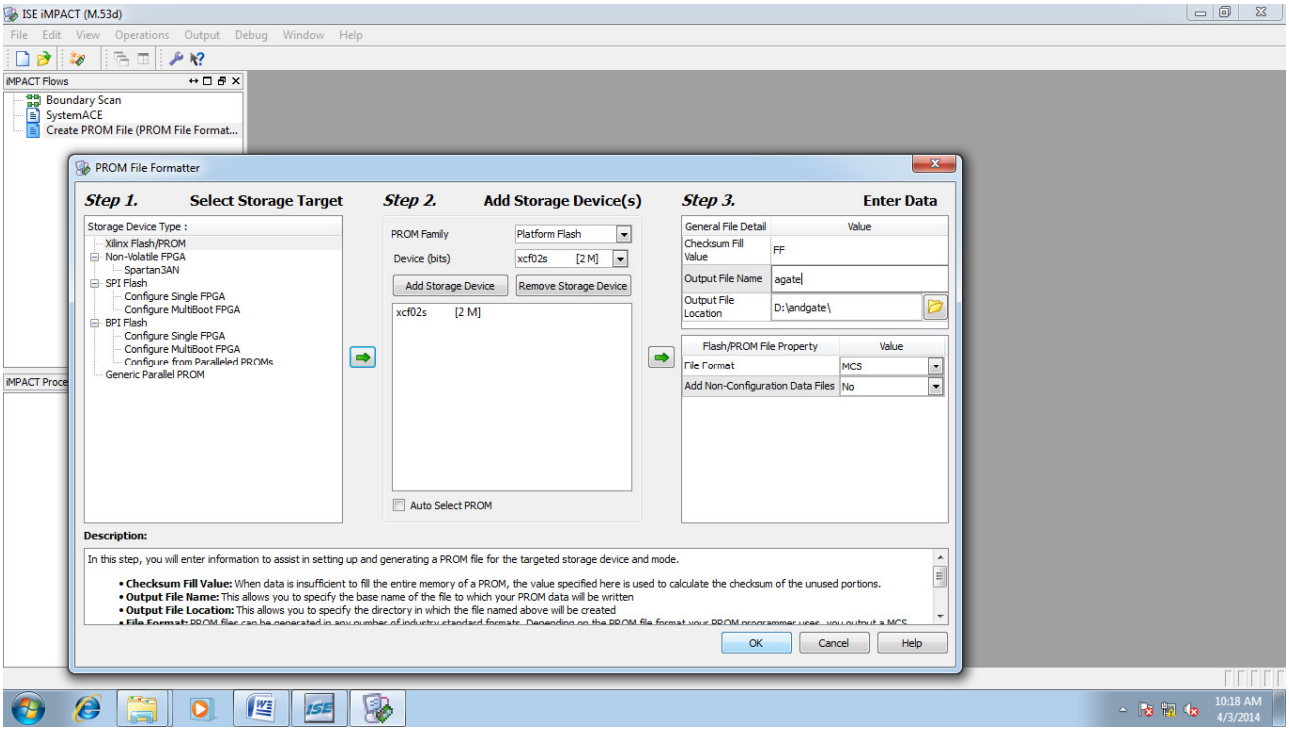
17. Select Storage Target Device as Xilinx Flash PROM and click forward.



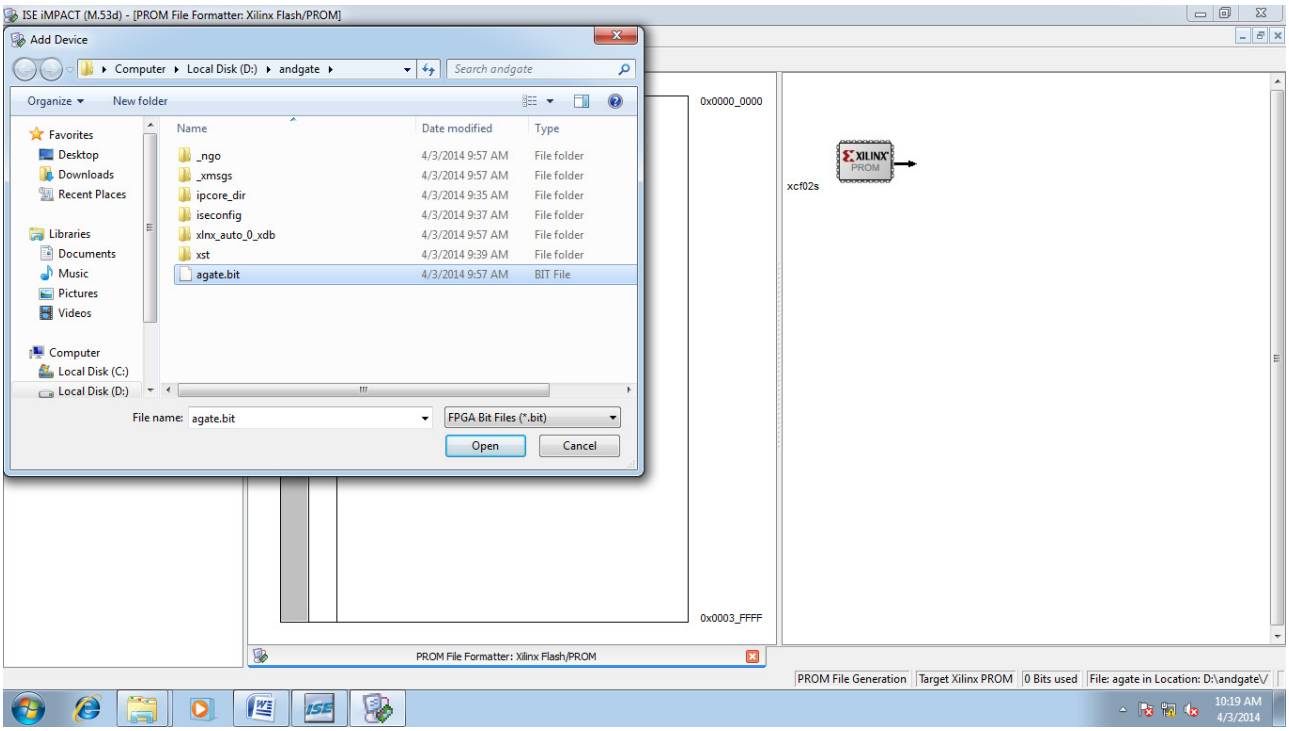
18. Add storage Device as xcf01s [2 M] and click forward.



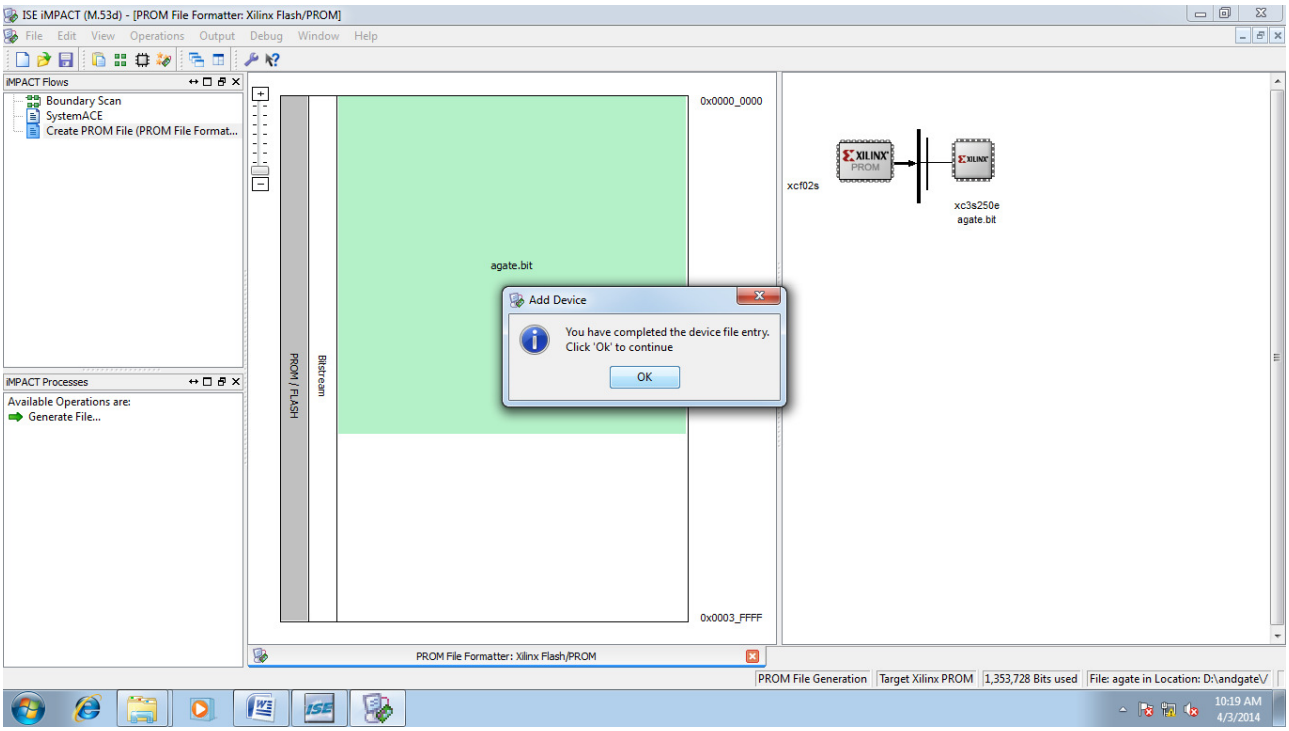
19. Type Output File Name and Location and click OK.



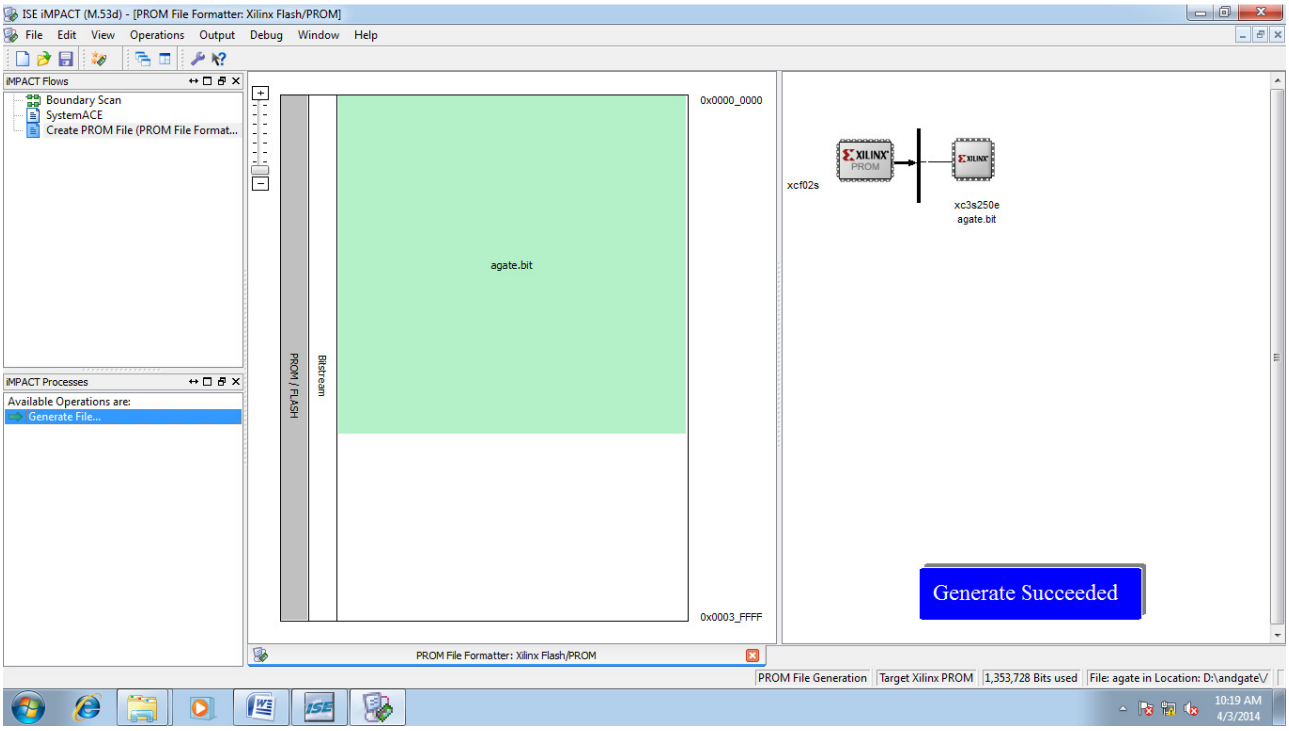
20. Select the corresponding .bit file and click Open.



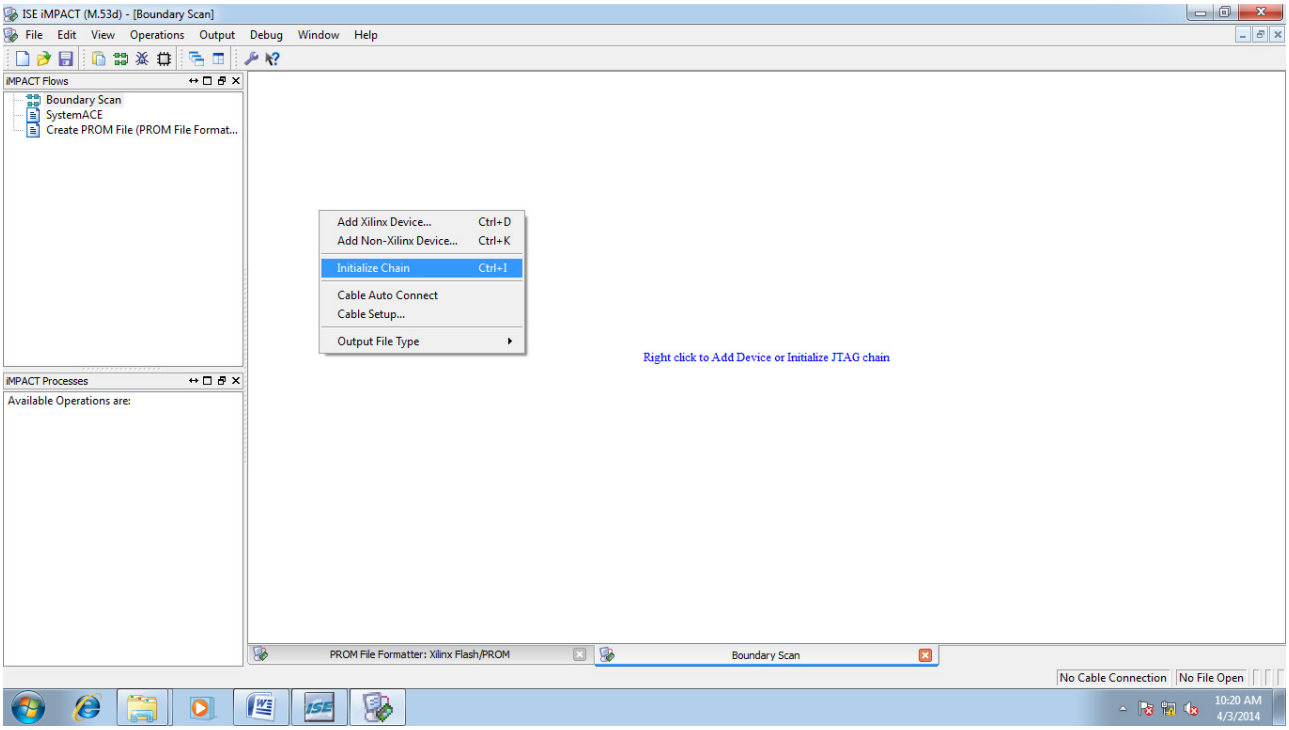
21. Click No to Add Another Device and Click OK.



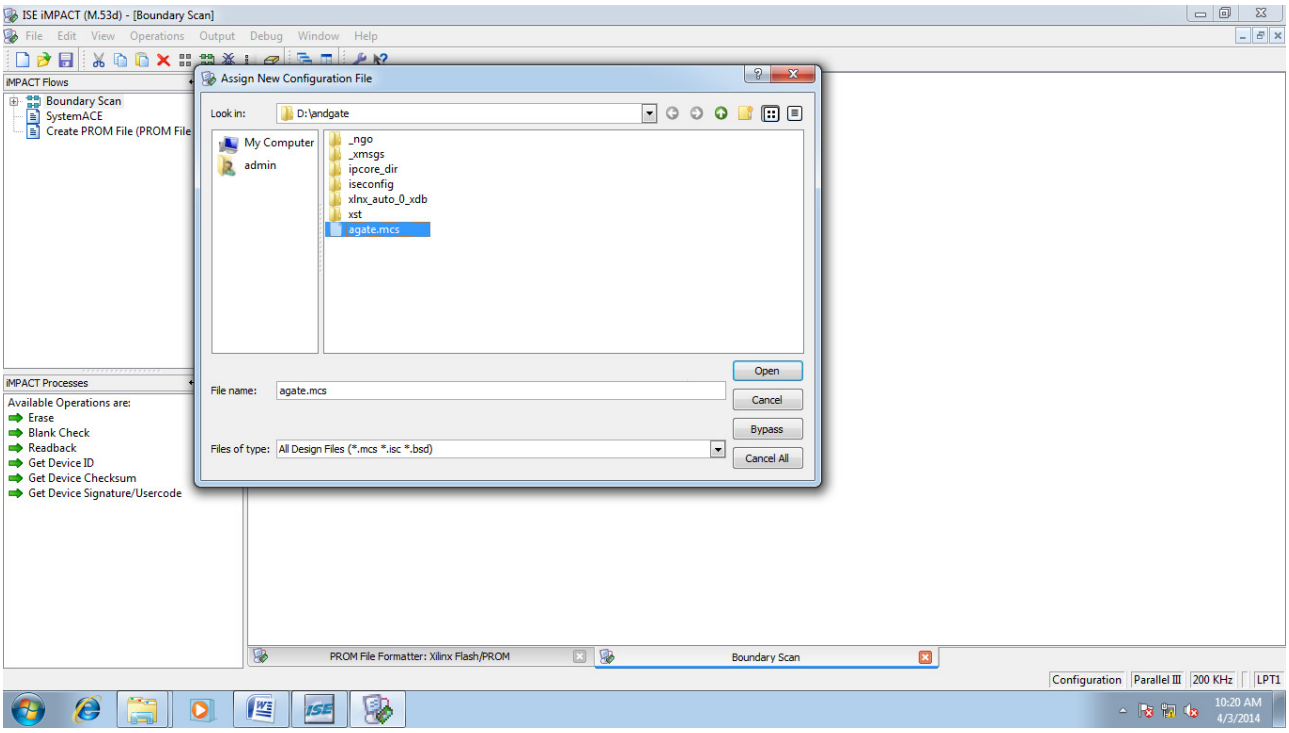
22. Double click Generate File.



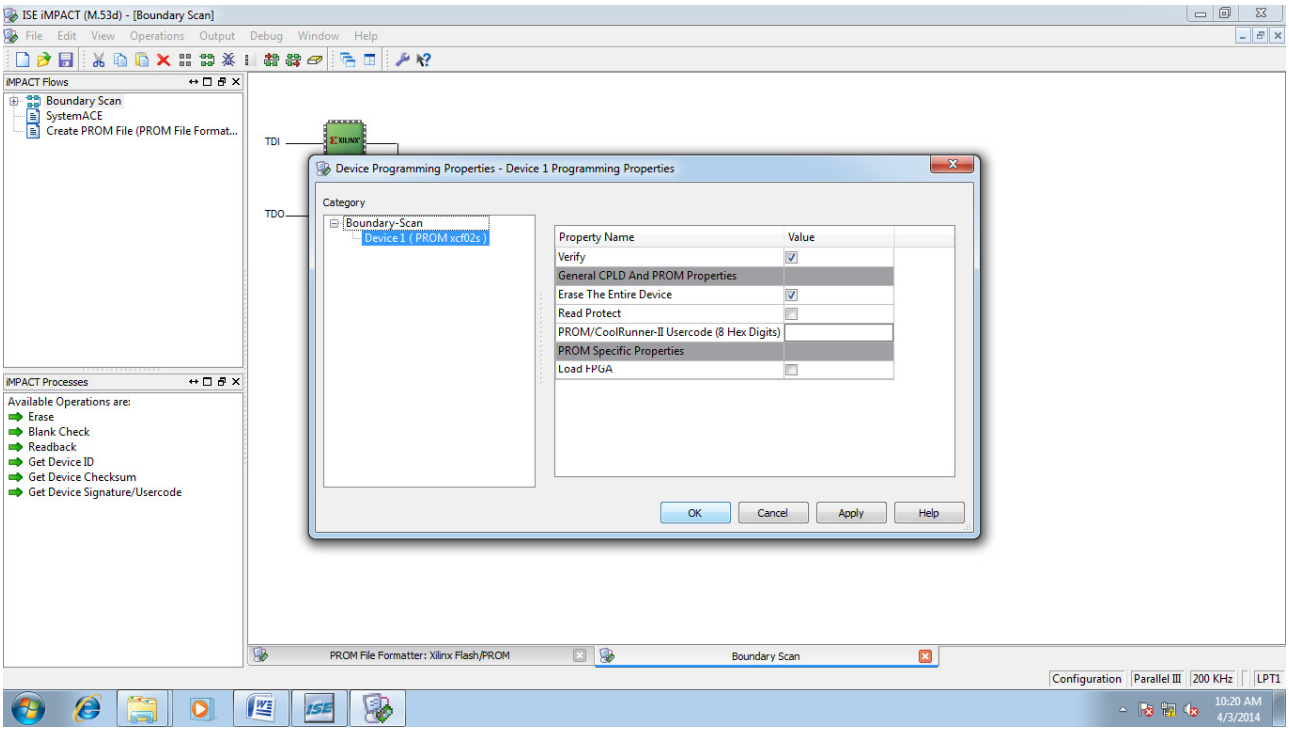
23. Double click Boundary Scan and Right click on the window and select Initialize Chain.



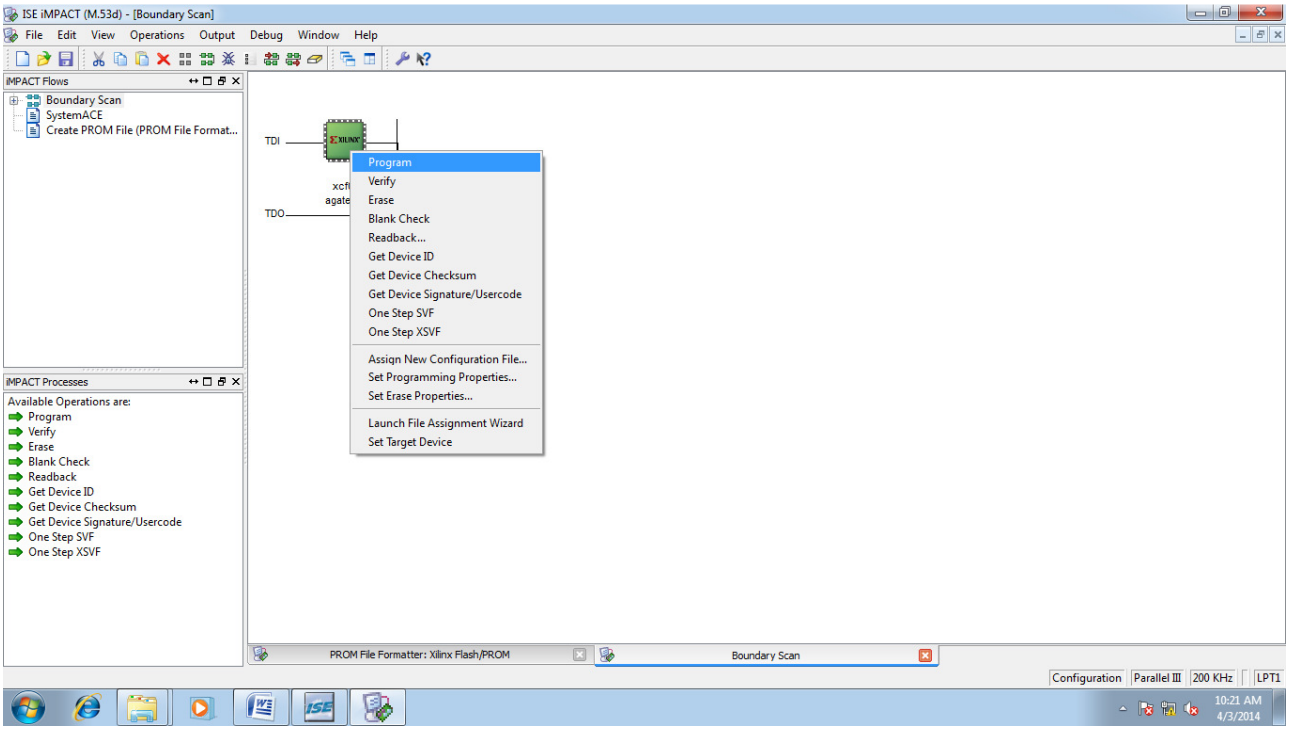
24. Now Select the corresponding .mcs file and click open.



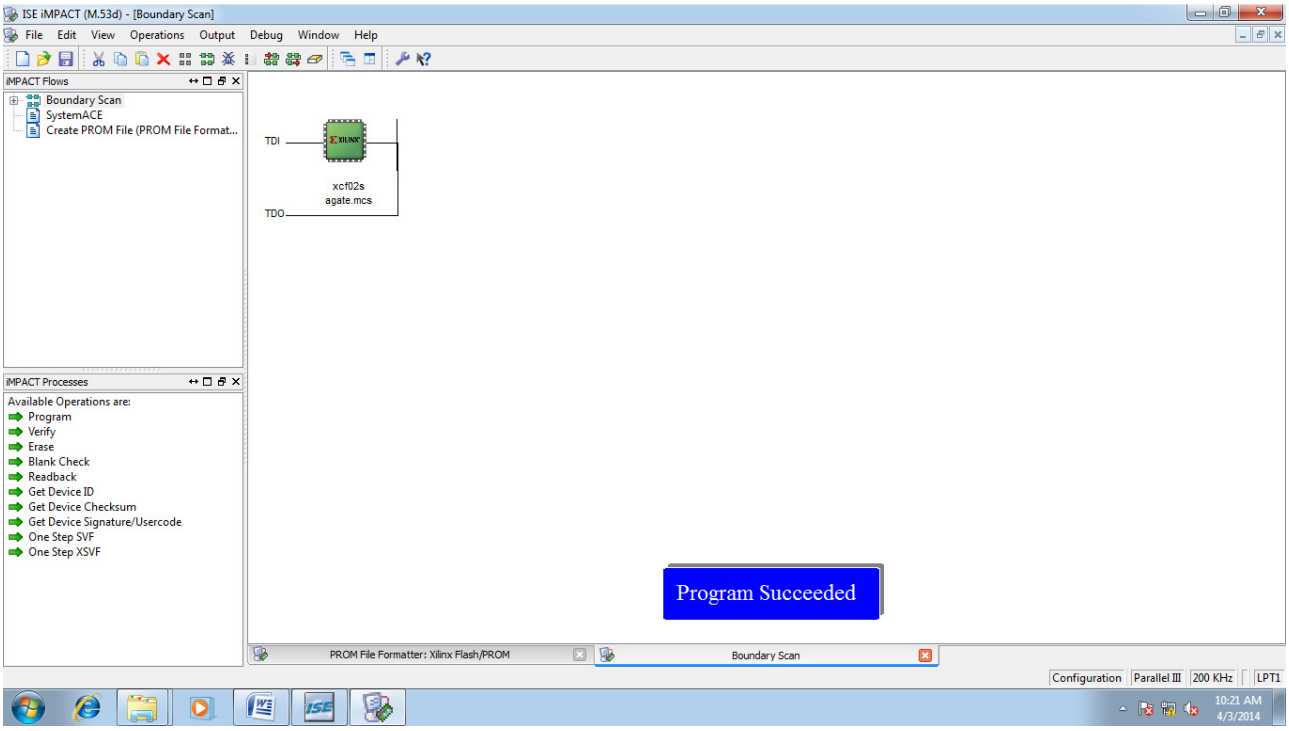
25. Click OK in the Device Programming Properties window.



26. Download the Program on to the kit by Right clicking on the device icon and select program.



27. After the program got downloaded successfully, change corresponding DIP switchto see the output.



RESULT:

Thus the study about various synthesis tools used in the Xilinx 9.2i download in FPGA is studied.

EXP:NO:15	IMPLEMENTATION OF BASIC LOGIC GATES IN FPGA
DATE:	

AIM:

To write a verilog program for basic logic gates to synthesize and implement in FPGA board using Xilinx tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

HARDWARE:

1. XILINX SPARTAN3E FPGA kit

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Write the functionality of the gates.
6. Terminate the program.

THEORY:

AND GATE:

The AND gate performs logical multiplication which is most commonly known as the AND junction. The operation of AND gate is such that the output is high only when all its inputs are high and when any one of the inputs is low the output is low.

$$Y = a \,\&\, b$$

OR GATE:

The OR gate performs logical addition which is most commonly known as the OR junction. The operation of OR gate is such that the output is high only when any one of its input is high and when both the inputs are low the output is low.

$$Y = a \mid b$$

NOT GATE:

The Inverter performs a basic logic gate function called Inversion or Complementation. The purpose of an inverter is to change one logic level to opposite level. When a high level is applied top an inverter, the low level will appear at the output and vice versa.

Y = ~a

NAND GATE:

The term NAND is derived from the complement of AND. It implies the AND junction with an inverted output. The operation of NAND gate is such that the output is low only when all its inputs are high and when any one of the inputs is low the output is high.

Y = ~(a & b)

NOR GATE:

The term NOR is derived from the complement of OR. It implies the OR junction with an inverted output. The operation of NOR gate is such that the output is high only when all its inputs are low and when any one of the inputs is high the output is low.

Y = ~(a | b)

EX-OR GATE:

The output is high only when the inputs are at opposite level.

Y = a ^ b

EX-NOR GATE:

The output is high only when the inputs are at same level.

Y = ~(a ^ b)

PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. Open a new UCF file and lock the pins of the design with FPGA I/O pins.
6. Implement the design by double clicking on the implementation tool selection.

7. Create programming file (i.e., bit file) for downloading into the specified device.

Hardware part

1. Connect the power supply cable to the FPGA kit using power supply adapter.
2. Connect FPGA board to parallel port of PC using parallel port cable.
3. Connect FRC1 of FPGA board with the switches (i/ps) of GPIO card - II using FRC cable.
4. Connect FRC2 of FPGA board with the LEDs (o/ps) of GPIO card - II using FRC cable.
5. To simulation go to model sim and verify the output waveform.(for simulation execute the program with test bench)
6. To varying the input port in the FPGA board to verify the output.(for FPGA implementation use only the main poogram)

PROGRAM:

Verilog Code for basic logic gates

```
module logicgates(a,b,c,d,e,f,g,h);
input a,b;
output c,d,e,f,g,h;
and(c,a,b);
or(d,a,b);
not(e,a);
nand(f,a,b);
xor(g,a,b);
xnor(h,a,b);
endmodule
```

USER CONSTRAIN:

NET "a" LOC = "p106";
NET "b" LOC = "p107";
NET "c" LOC = "p160";
NET "d" LOC = "p161";
NET "e" LOC = "p162";
NET "f" LOC = "p163";
NET "g" LOC = "p164"

RESULT:

Thus the verilog program for basic logic gates were written, synthesized and implemented in FPGA device.

EXP: NO: 16	IMPLEMENTATION OF HALF ADDER AND FULL ADDER IN FPGA
DATE:	

AIM:

To write a verilog program for half adder and full adder to synthesize and implement in FPGA board using Xilinx tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

HARDWARE:

1. XILINX SPARTAN3E FPGA kit

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Terminate the program.

THEORY:

HALF ADDER:

The half adder consists of two input variables designated as Augends and Addend bits. Output variables produce the Sum and Carry. The 'carry' output is 1 only when both inputs are 1 and ,sum' is 1 if any one input is 1. The Boolean expression is given by,

sum = x ^ y

carry = x & y

FULL ADDER:

A Full adder is a combinational circuit that focuses the arithmetic sum of three bits. It consists of 3 inputs and 2 outputs. The third input is the carry from the previous Lower Significant Position. The two outputs are designated as Sum (S) and Carry (C). The binary variable S gives the value of the LSB of the Sum. The output S=1 only if odd number of 1's are present in the input and the output C=1 if two or three inputs are 1.

$$sum = x \wedge y \wedge z$$

$$carry = (x \,\&\, y) \mid (y \,\&\, z) \mid (x \,\&\, z)$$

PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. Open a new UCF file and lock the pins of the design with FPGA I/O pins.
6. Implement the design by double clicking on the implementation tool selection.
7. Create programming file (i.e., bit file) for downloading into the specified device.

Hardware part

1. Connect the power supply cable to the FPGA kit using power supply adapter.
2. Connect FPGA board to parallel port of PC using parallel port cable.
3. Connect FRC1 of FPGA board with the switches (i/ps) of GPIO card - II using FRC cable.
4. Connect FRC2 of FPGA board with the LEDs (o/ps) of GPIO card - II using FRC cable.
5. To simulation go to model sim and verify the output waveform.(for simulation execute the program with test bench)
6. To varying the input port in the FPGA board to verify the output.(for FPGA implementation use only the main poogram)

PROGRAM:

Verilog code for half adder

```
module halfadder(a, b, sum, carry);
input a;
input b;
output sum;
output carry;
xor(sum,a,b);
and(carry,a,b);
endmodule
```

USER CONSTRAIN:

```
NET "a" LOC = "p106";
NET "b" LOC = "p107";
NET "sum" LOC = "p160";
NET "carry" LOC = "p161";
```

PROGRAM:

Verilog code for full adder

```
module fulladder(a,b,c,sum,carry);
input a,b,c;
output sum,carry;
wire d,e,f;
xor(d,a,b);
xor(sum,d,c);
and(e,d,c);
and(f,a,b);
or(carry,e,f);
```

endmodule

USER CONSTRAINT:

NET “a” LOC = “p106”;

NET “b” LOC = “p107”;

NET “c” LOC = “p108”;

NET “sum” LOC = “p160”;

NET”carry” LOC=”p161”;

RESULT:

Thus the verilog program for half adder and full adder were written, synthesized and implemented in FPGA device**.**

EXP: NO: 17	IMPLEMENTATION OF HALF SUBTRACTOR AND FULL SUBTRACTOR IN FPGA
DATE:	

AIM:

To write a verilog program for half subtractor and full subtractor to synthesize and implement in FPGA board using Xilinx tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

HARDWARE:

1. XILINX SPARTAN3E FPGA kit

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Terminate the program.

THEORY:

HALF SUBRACTOR:

The Half subtractor consist of two input variables, the output variables produce the Difference (d) and Borrow (bo). The output 'bo' is 1 only when the input 'a' is at low level and other input 'b' is at higher level. The output'd' is 1 only when only one of the inputs is 1. The Boolean expression for half subtractor is given by,

difference = a ^ b

borrow = a'b

FULL SUBTRACTOR:

A full subtractor is a multiple output combinational logical network which performs a subtraction between two binary bits considering that a '1' might have been borrowed by a lower significant stage. Along with the minuend 'a' and the subtrahend 'b', the third input is the borrow bit 'c', from the previous stage of subtraction. The combinational logic network of the full subtractor thus has three inputs and two outputs. The two outputs produced are the difference bit output'd' and a final borrow 'bo' respectively. The Boolean expression for full subtractor is given by,

difference = a ^ b ^ c

borrow = a'b + a'c + bc

PROCEDURE: (SAME FOR HALF SUBTRACTOR AND FULL SUBTRACTOR)

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Draw the half subtractor and full subtractor circuit by choosing schematic as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. Open a new UCF file and lock the pins of the design with FPGA I/O pins.
6. Implement the design by double clicking on the implementation tool selection.
7. Create programming file (i.e., bit file) for downloading into the specified device.

Hardware part

1. Connect the power supply cable to the FPGA kit using power supply adapter.
2. Connect FPGA board to parallel port of PC using parallel port cable.
3. Connect FRC1 of FPGA board with the switches (i/ps) of GPIO card - II using FRC cable.
4. Connect FRC2 of FPGA board with the LEDs (o/ps) of GPIO card - II using FRC cable.
5. To simulation go to model sim and verify the output waveform.(for simulation execute the program with test bench)

6. To varying the input port in the FPGA board to verify the output.(for FPGA implementation use only the main poogram)

PROGRAM:

Verilog code for half subtractor

```
module halfsubtractor(a,b,sub,borrow,);
input a,b;
output sub,borrow;
xor(sub,a,b);
not(x,a);
and(borrow,x,b);
endmodule
```

USER CONSTRAIN:

```
NET "a" LOC = "p106";
NET "b" LOC = "p107";
NET "sub" LOC = "p160";
NET "borrow" LOC = "p161" ;
```

PROGRAM:

Verilog code for full subtractor

```
module fullsubtractor(a,b,c,sub,borrow);
input a,b,c;
output sub,borrow;
xor(p,a,b);
not(x,a);
and(g,x,b);
xor(sub,p,c);
not(y,p);
and(z,y,c);
or(borrow,z,g);
```

endmodule

USER CONSTRAIN:

NET "a" LOC = "p106";

NET "b" LOC = "p107";

NET "c" LOC = "p108";

NET "sub" LOC = "p160";

NET ”borrow” LOC=”p161”

RESULT:

Thus the verilog program for half subtractor and full subtractor were written, synthesized and implemented in FPGA device.

EXP: NO: 18	IMPLEMENTATION OF MULTIPLEXER AND DEMULTIPLEXER IN FPGA
DATE:	

AIM:

To write a verilog program for multiplexer and demultiplexer to synthesize and implement in FPGA board using Xilinx tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

HARDWARE:

1. XILINX SPARTAN3E FPGA kit

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Terminate the program.

THEORY:

MULTIPLEXER

A Multiplexer is a Combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The set of selection of a particular line is controlled by selection lines. Normally there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.

The 4:1 MUX has four inputs I_0, I_1, I_2 and I_3 and select lines S_0 and S_1. The select lines s0 and s1 are decoded to select a particular AND gate. The outputs of the AND gates are applied to a single OR gate that provides the one line output Y.

DEMULTIPLEXER

A Demultiplexer is a Combinational circuit that selects binary information from one of input line and directs it to many output line. The set of selection of a particular output is controlled by selection lines. Normally there are 1 input line and 2^n selection lines whose bit combinations determine the output.

The 1:4 DEMUX has one input and select lines S_0 and S_1. The select lines s0 and s1 are decoded to select a particular AND gate. The outputs of the AND gates provides the various line output Y_1, Y_2, Y_3 and Y_4.

PROCEDURE: (SAME FOR BOTH MUX & DEMUX)

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. Open a new UCF file and lock the pins of the design with FPGA I/O pins.
6. Implement the design by double clicking on the implementation tool selection.
7. Create programming file (i.e., bit file) for downloading into the specified device.

Hardware part

1. Connect the power supply cable to the FPGA kit using power supply adapter.
2. Connect FPGA board to parallel port of PC using parallel port cable.
3. Connect FRC1 of FPGA board with the switches (i/ps) of GPIO card - II using FRC cable.
4. Connect FRC2 of FPGA board with the LEDs (o/ps) of GPIO card - II using FRC cable.
5. To simulation go to model sim and verify the output waveform.(for simulation execute the program with test bench)
6. To varying the input port in the FPGA board to verify the output.(for FPGA implementation use only the main poogram)

PROGRAM:

Verilog code for Multiplexer

```
module mux4to1(s0,s1,t0,t1,t2,t3,out);
input s0,s1,t0,t1,t2,t3;
output out;
wire s0n,s1n,n1,n2,n3,n4;
not(s0n,s0);
not(s1n,s1);
and(n1,s0n,s1n,t0);
and(n2,s0,s1n,t1);
and(n3,s0n,s1,t2);
and(n4,s0,s1,t3);
or(out,n1,n2,n3,n4);
endmodule
```

USER CONSTRAIN:

```
NET "s0" LOC = "p1066";
NET "s1" LOC = "p107";
NET "t0" LOC = "p108";
NET "t1" LOC = "p109";
NET "t2" LOC = "p112";
NET "t3" LOC = "p113";
NET"out" LOC="p160"
```

VERILOG CODE FOR DEMULTIPLEXER

```
module demux1to4(s0,s1,in,t0,t1,t2,t3);
input s0,s1,in;
output t0,t1,t2,t3;
wire s0n,s1n;
```

```
not(s0n,s0);
not(s1n,s1);
and(t0,s0n,s1,in);
and(t1,s0n,s1n,in);
and(t2,s0,s1,in);
and(t3,s0,s1n,in);
endmodule
```

USER CONSTRAIN:

```
NET "s0" LOC = "p106";
NET "s1" LOC = "p107";
NET "in" LOC = "p108";
NET "t0" LOC = "p160";
NET "t1" LOC = "p161";
NET "t2" LOC = "p162";
NET "t3" LOC = "p163";
```

RESULT:

Thus the verilog program for mux and demux were written, synthesized and implemented in FPGA device.

EXP: NO: 19	www.vidyarthiplus.com **IMPLEMENTATION OF ENCODER AND DECODER IN FPGA**
DATE:	

AIM:

To write a verilog program for encoder and decoder to synthesize and implement in FPGA board using Xilinx tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

HARDWARE:

1. XILINX SPARTAN3E FPGA kit

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Terminate the program.

THEORY:

ENCODER

An Encoder is a digital circuit that has 2^n (or fewer) input lines and n output lines. The output lines generate the binary the binary code corresponding to the input value. In encoder it is assumed that only one input has a value of 1 at any given time.

DECODER

Discrete quantities of information are represented in digital systems by binary codes. A binary code of n bits is capable of representing up to 2^n distinct elements of coded information. A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. If the n bit coded information unused combinations. The decoder may have fewer than 2^n outputs.

PROCEDURE: (FOR ENCODER & DECODER)

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. Open a new UCF file and lock the pins of the design with FPGA I/O pins.
6. Implement the design by double clicking on the implementation tool selection.
7. Create programming file (i.e., bit file) for downloading into the specified device.

Hardware part

1. Connect the power supply cable to the FPGA kit using power supply adapter.
2. Connect FPGA board to parallel port of PC using parallel port cable.
3. Connect FRC1 of FPGA board with the switches (i/ps) of GPIO card - II using FRC cable.
4. Connect FRC2 of FPGA board with the LEDs (o/ps) of GPIO card - II using FRC cable.
5. To simulation go to model sim and verify the output waveform.(for simulation execute the program with test bench)
6. To varying the input port in the FPGA board to verify the output.(for FPGA implementation use only the main poogram)

PROGRAM:

Verilog code for Encoder

```
module encoder(a,b,c,d,s0,s1);
input a,b,c,d;
output s0,s1;
or(s0,c,d);
or(s1,b,d);
endmodule
```

USER CONSTRAIN:

NET "a" LOC = "p106"
NET "b" LOC = "p107";
NET "c" LOC = "p108";
NET "d" LOC = "p109";
NET"s0" LOC="p160"
NET "s1" LOC="p161";

VERILOG CODE FOR DECODEER

```
module decoder(s0,s1,t0,t1,t2,t3);
input s0,s1;
output t0,t1,t2,t3;
wire s0n,s1n;
not(s0n,s0);
not(s1n,s1);
and(t0,s0n,s1n);
and(t1,s0n,s1);
and(t2,s0,s1n);
and(t3,s0,s1);
endmodule
```

USER CONSTRAIN:

NET "s0" LOC = "p106";
NET "s1" LOC = "p107";
NET "t0" LOC = "p160";
NET "t1" LOC = "p161";
NET "t2" LOC = "p162";
NET "t3" LOC = "p163";

RESULT:

Thus a verilog program for encoder and decoder was synthesized and implemented in FPGA board using Xilinx tool.

EXP: NO: 20	IMPLEMENTATION OF D FLIP FLOP AND D LATCH IN FPGA
DATE:	

AIM:

To write a verilog program for D-flip flop and D-latch to synthesize and implement in FPGA board using Xilinx tool.

TOOLS REQUIRED:

SOFTWARE:

1. Xilinx ISE Design Suite 12.1

HARDWARE:

1. XILINX SPARTAN3E FPGA kit.

ALGORITHM:

1. Start the program.
2. Declare the input and output variables.
3. Declare the output as register data type.
4. Use PROCEDURAL construct statements (behavioral modeling) for Verilog code.
5. Terminate the program.

THEORY:

D-FLIP FLOP:

It has only a single data input. That data input is connected to the S input of RS-flip flop, while the inverse of D is connected to the R input. This prevents that the input combination ever occurs. To allow the flip flop to be in holding state, a D-flip flop has a second input called “clock”. The clock input is AND-ed with the D input, such that when clock=0, the R and S inputs of the RS-flip flop are 0 and the state is held.

D-LATCH:

It has only a single data input. That data input is connected to the S input of RS-flip flop, while the inverse of D is connected to the R input. This prevents that the input combination ever occurs. To allow the flip flop to be in holding state, a D-flip flop has a second input called “enable”. The enable input is AND-ed with the D input, such that when enable=0, the R and S inputs of the RS-flip flop are 0 and the state is held.

PROCEDURE:

Software part

1. Click on the Xilinx ISE Design Suite 12.1or Xilinx Project navigator icon on the desktop of PC.
2. Write the Verilog code by choosing HDL as top level source module.
3. Check syntax, view RTL schematic and note the device utilization summary by double clicking on the synthesis in the process window.
4. Perform the functional simulation using Xilinx ISE simulator.
5. Open a new UCF file and lock the pins of the design with FPGA I/O pins.
6. Implement the design by double clicking on the implementation tool selection.
7. Create programming file (i.e., bit file) for downloading into the specified device.

Hardware part

1. Connect the power supply cable to the FPGA kit using power supply adapter.
2. Connect FPGA board to parallel port of PC using parallel port cable.
3. Connect FRC1 of FPGA board with the switches (i/ps) of GPIO card - II using FRC cable.
4. Connect FRC2 of FPGA board with the LEDs (o/ps) of GPIO card - II using FRC cable.
5. To simulation go to model sim and verify the output waveform.(for simulation execute the program with test bench)
6. To varying the input port in the FPGA board to verify the output.(for FPGA implementation use only the main poogram)

PROGRAM:

Verilog code for D flipflop

```
module dff(d,clk,rst,q);
input d,clk,rst;
output q;
reg q;
always@(posedge clk or posedge rst)
begin
   if (rst)
   q<=1'b0;
   else
   if(clk)
   q<=d;
end
endmodule
```

USER CONSTRAIN:

```
NET "d" LOC = "p106";
NET "clk" LOC = "p107";
NET "rst" LOC = "p108";
NET "q" LOC = "p160";
```

Verilog code for D latch

```
module dlatch(d,en,b,c,a,q,qbar);
input d,en;
output b,c,a,q,qbar;
wire b,c,a,q,qbar;
not(a,d);
nand(b,d,en);
nand(c,a,en);
nand(q,b,qbar);
```

```
nand(qbar,c,q);
endmodule
```

USER CONSTRAIN:

```
NET "d" LOC = "p106";
NET "en" LOC = "p107";
NET "a" LOC = "p160";
NET "b" LOC = "p161";
NET "c" LOC = "p162";
NET "q" LOC = "p163";
NET "qbar" LOC = "p164";
```

RESULT:

Thus the verilog program for D flip flop and D latch were written, synthesized and implemented in FPGA device.

EXP: NO: 21	www.vidyarthiplus.com **IMPLEMENTATION OF MOS DIFFERENTIAL AMPLIFIER SCHEMATIC IN ORCAD PSPICE**
DATE:	

AIM:

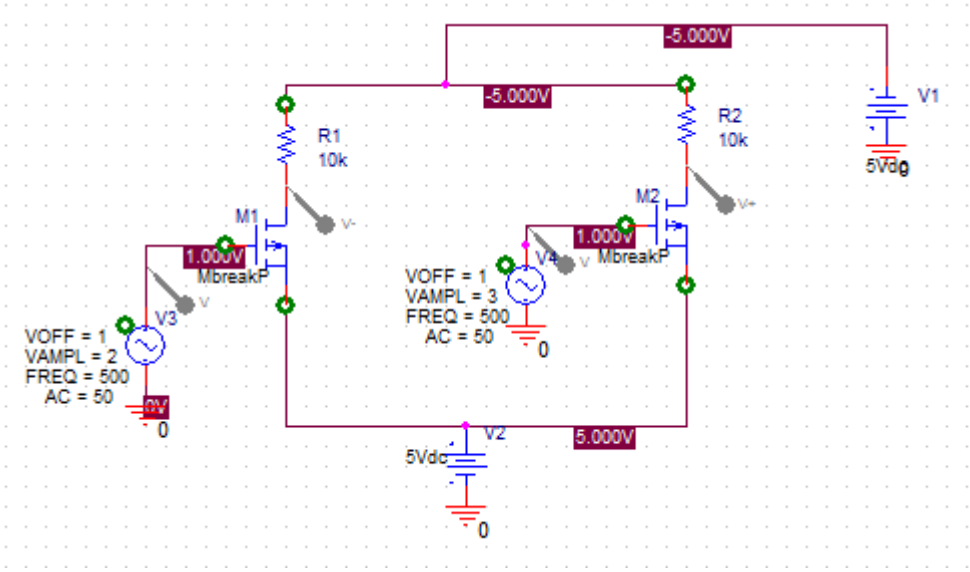
To design, synthesize, simulate and implement the MOS differential amplifier schematic in ORCAD PSPICE.

TOOLS REQUIRED:

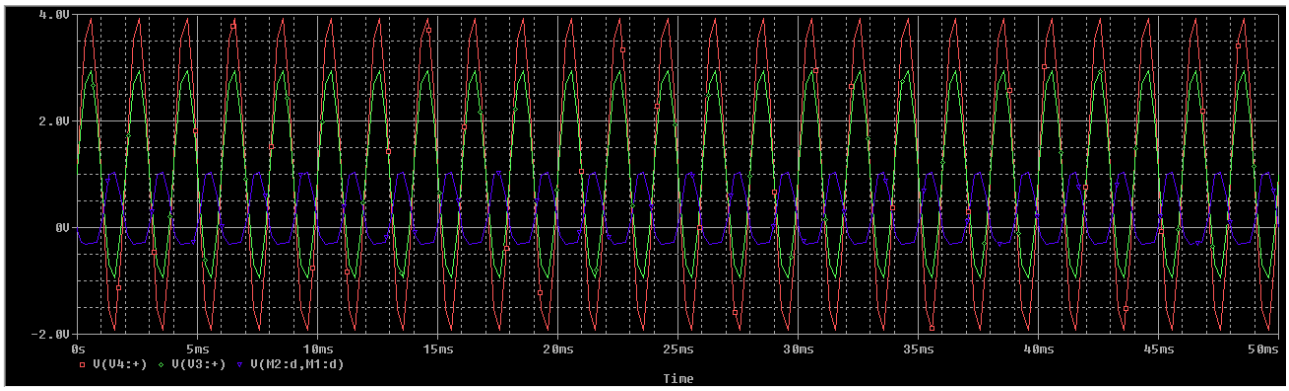
SOFTWARE:

1. ORCAD PSPICE

CIRCUIT DIAGRAM:



OUTPUT:



RESULT:

Thus the CMOS inverter schematic form was designed and stimulated using ORCAD PSIPCE.

EXP: NO: 22	www.vidyarthiplus.com **STUDY OF SIMULATING LAYOUT IN MICROWIND**
DATE:	

AIM:

To study the simulating schematic in Microwind.

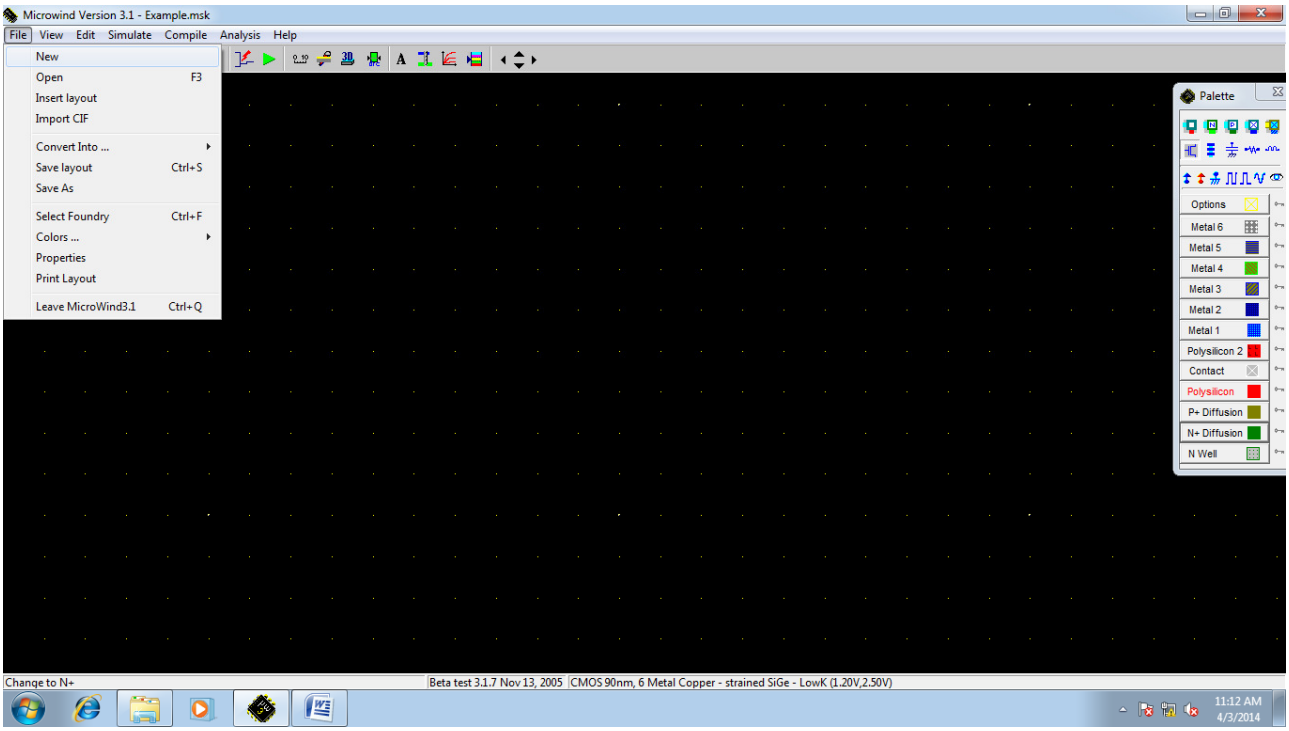
TOOLS REQUIRED:

SOFTWARE :

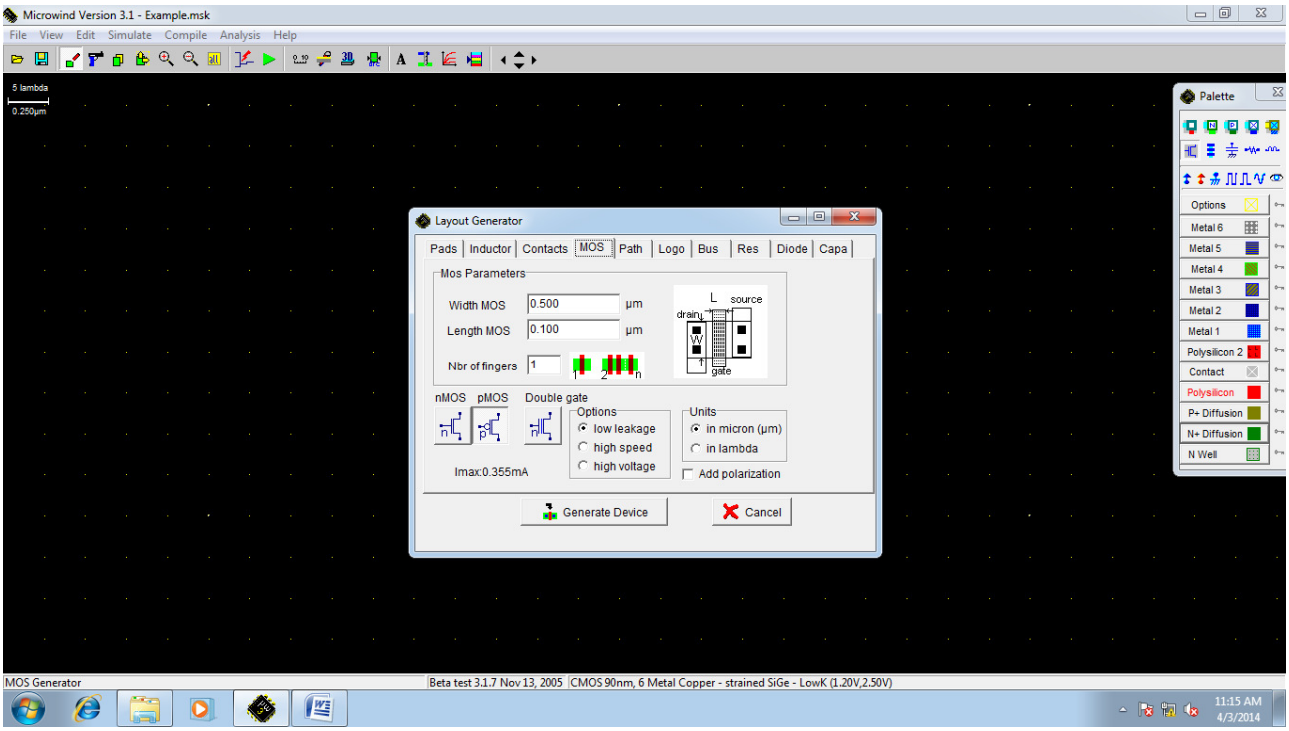
1. MICROWIND V3.1

PROCEDURE:

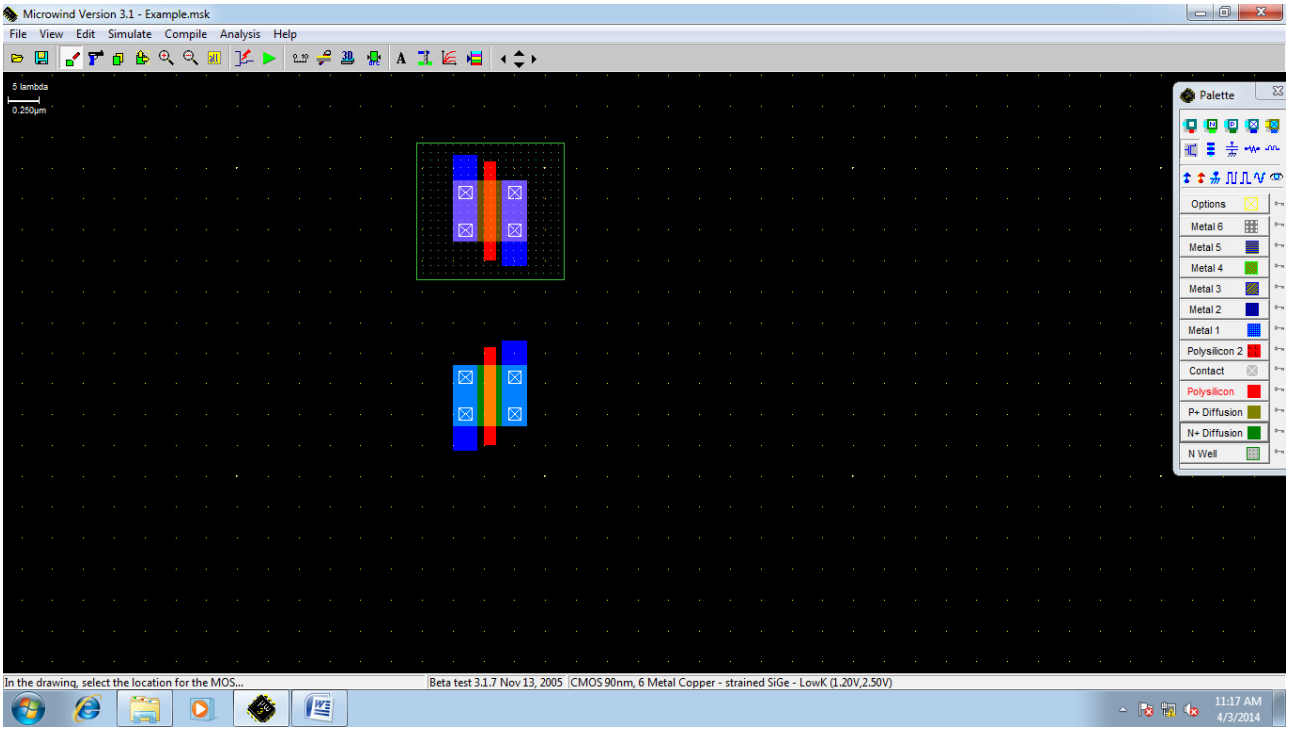
1. Double click on Microwind and go to file→new.



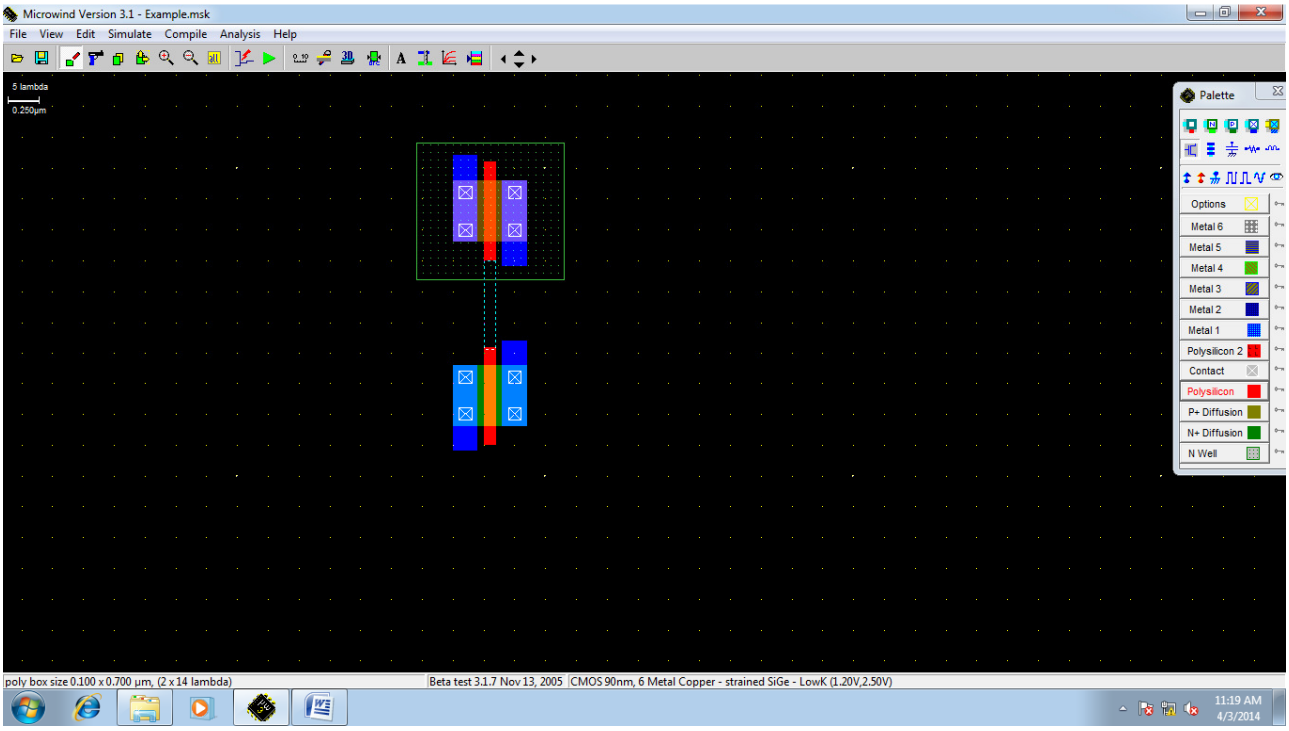
2. **Select MOS Generator and select PMOS device and click Generate Device.**



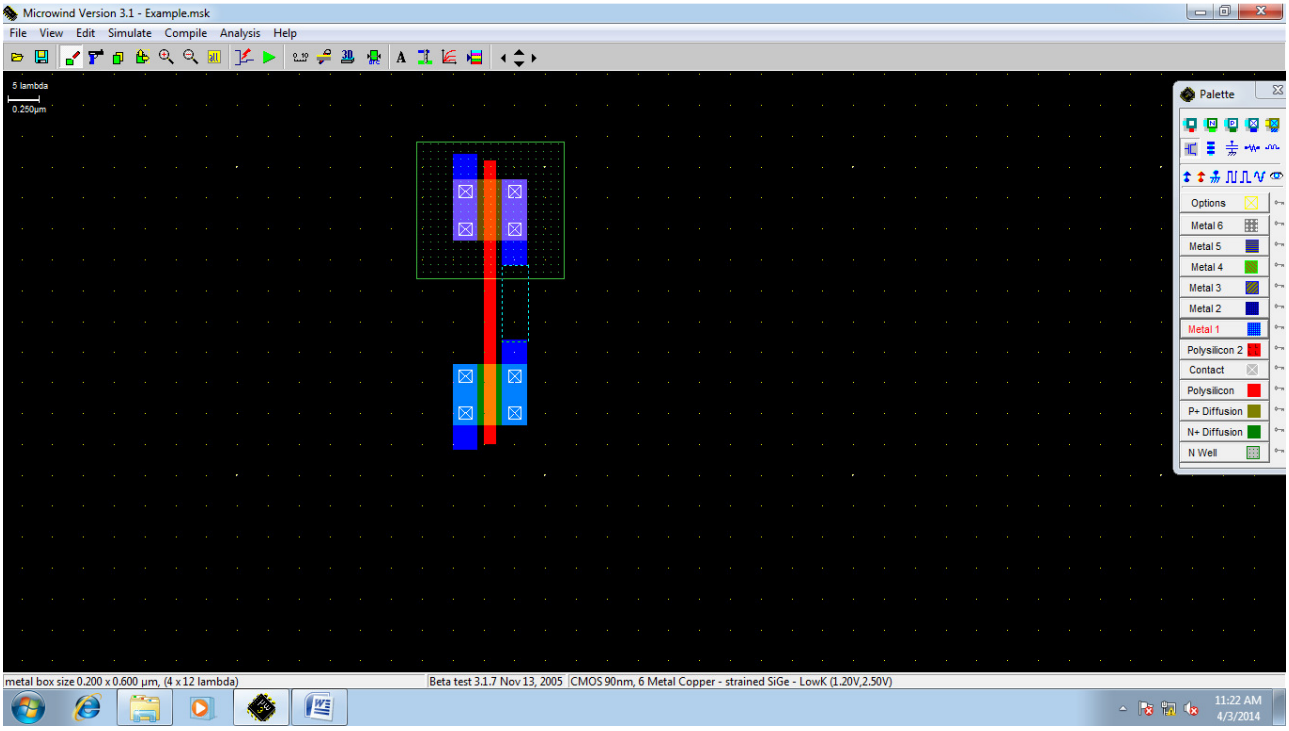
3. **Similarly generate NMOS device and place it.**



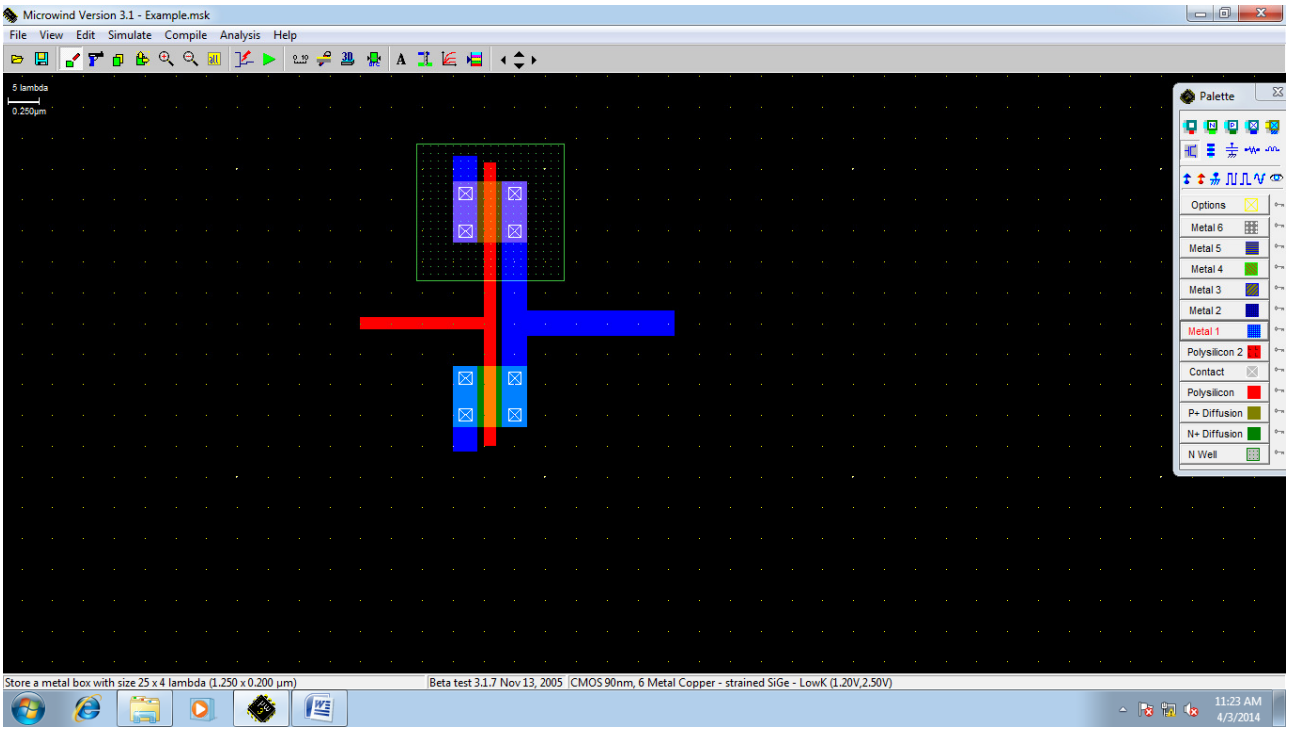
4. **Select Polysilicon from the Palette and connect the gate of PMOS and NMOS.**



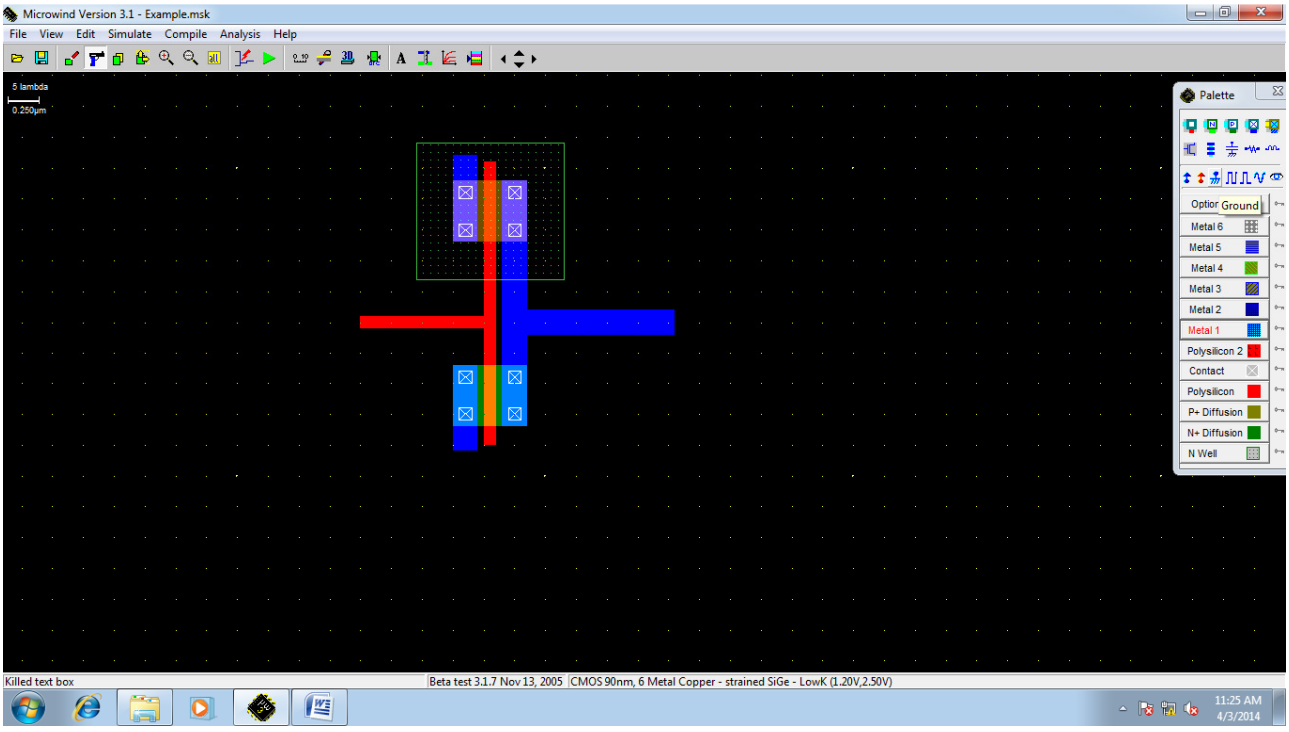
5. **Now Connect the two transistors using a metal1 layer from the Palette.**

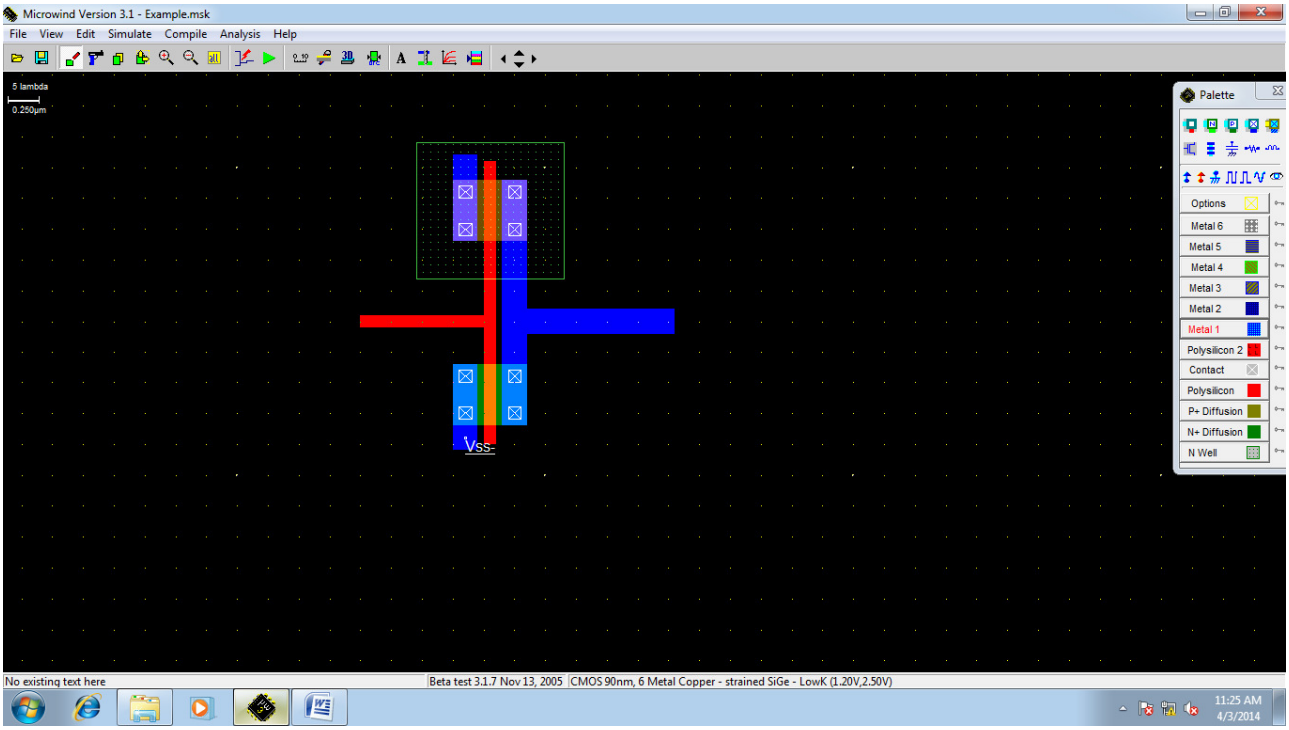


6. **Now make connections for input and output using corresponding layers.**

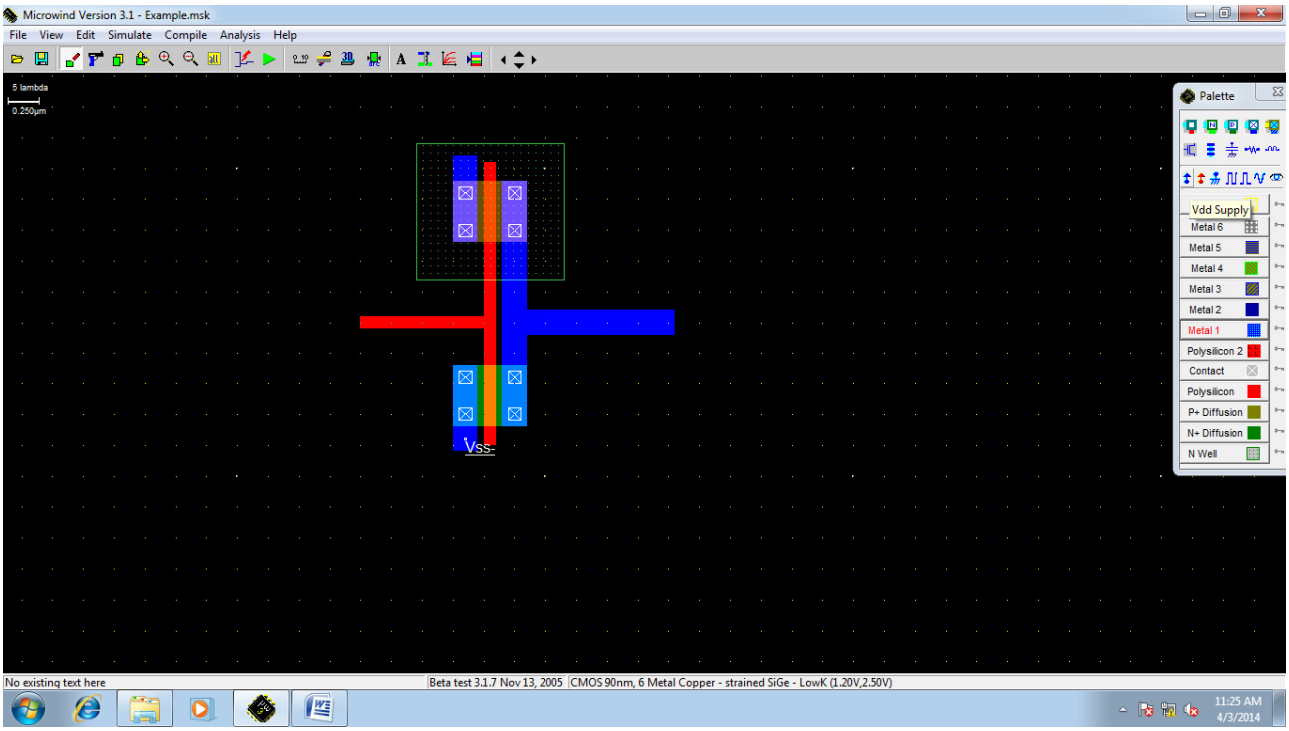


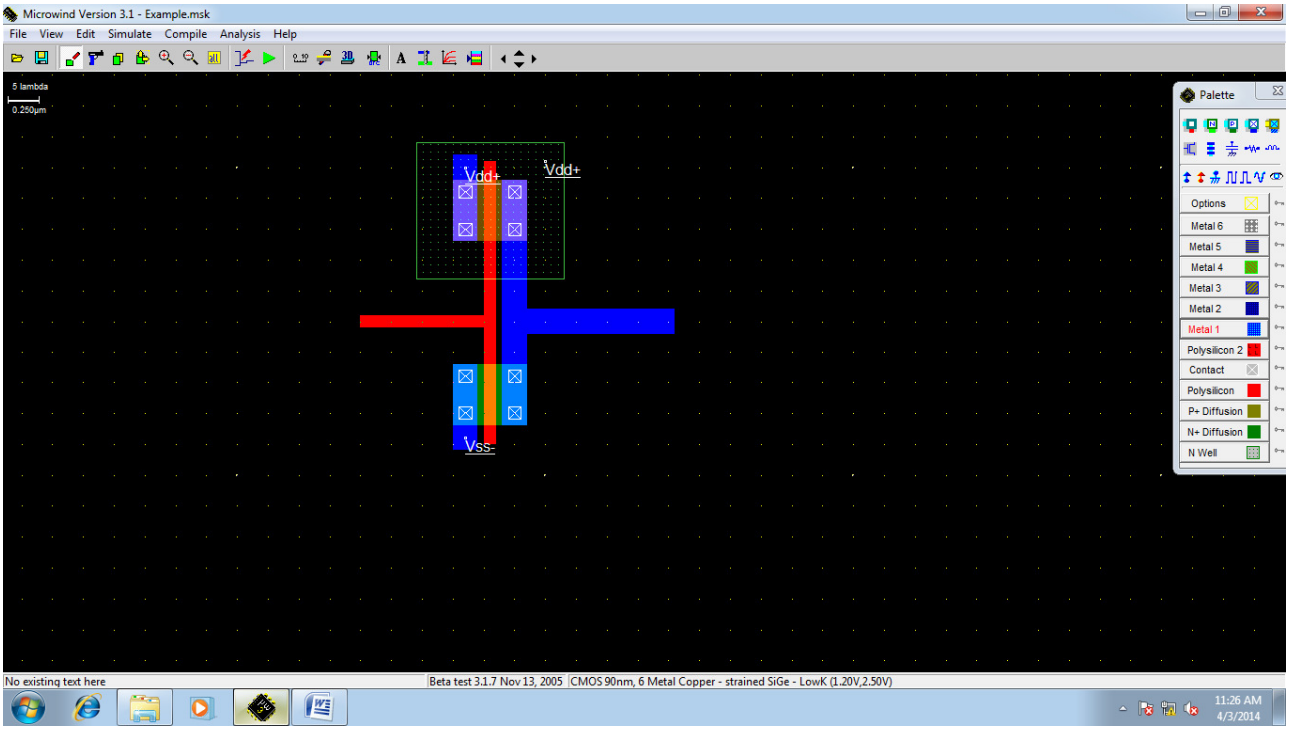
7. **Choose Ground from Palette and place it.**



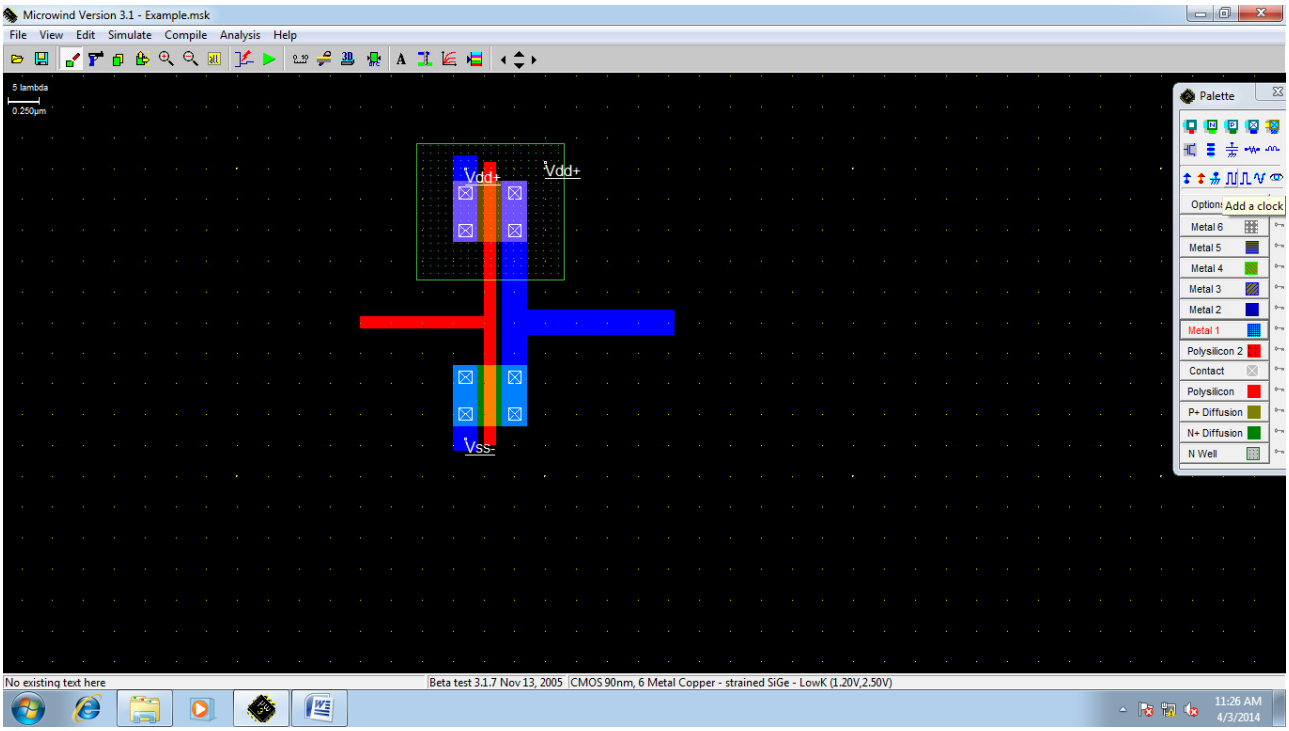


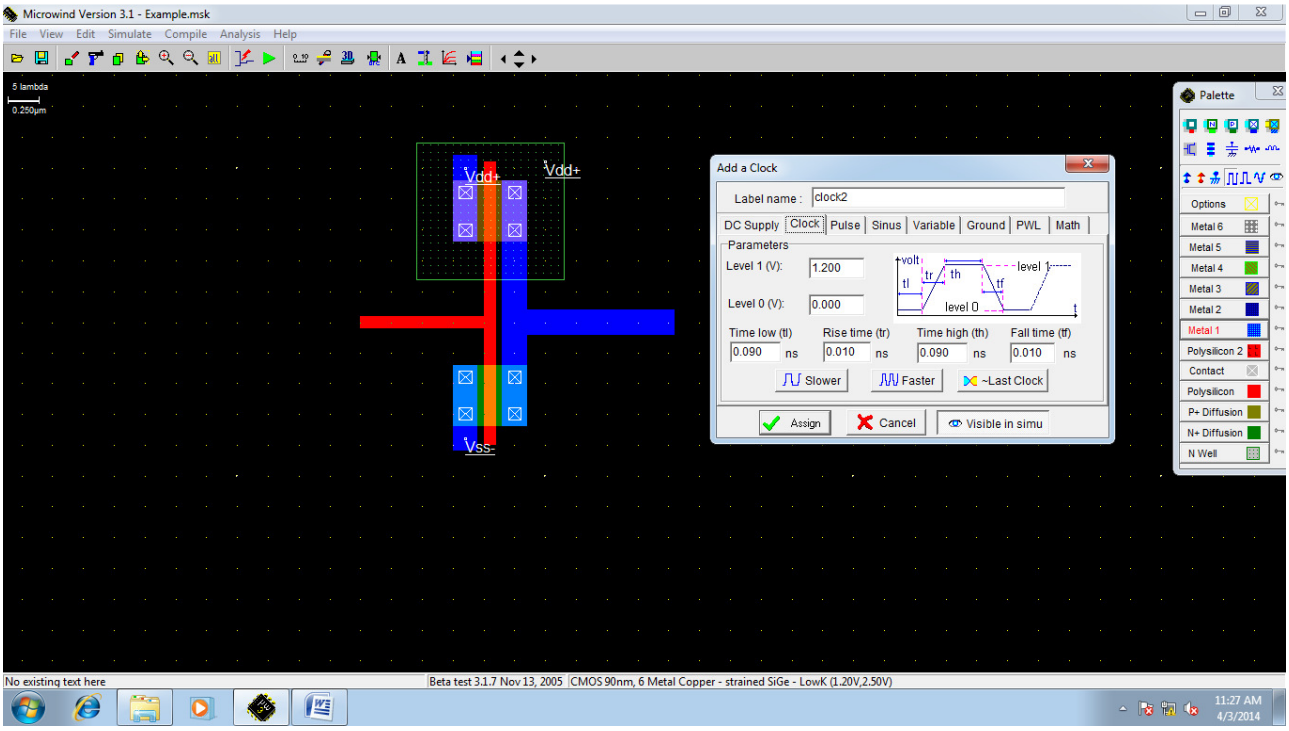
8. Choose supply from the Palette and place it.

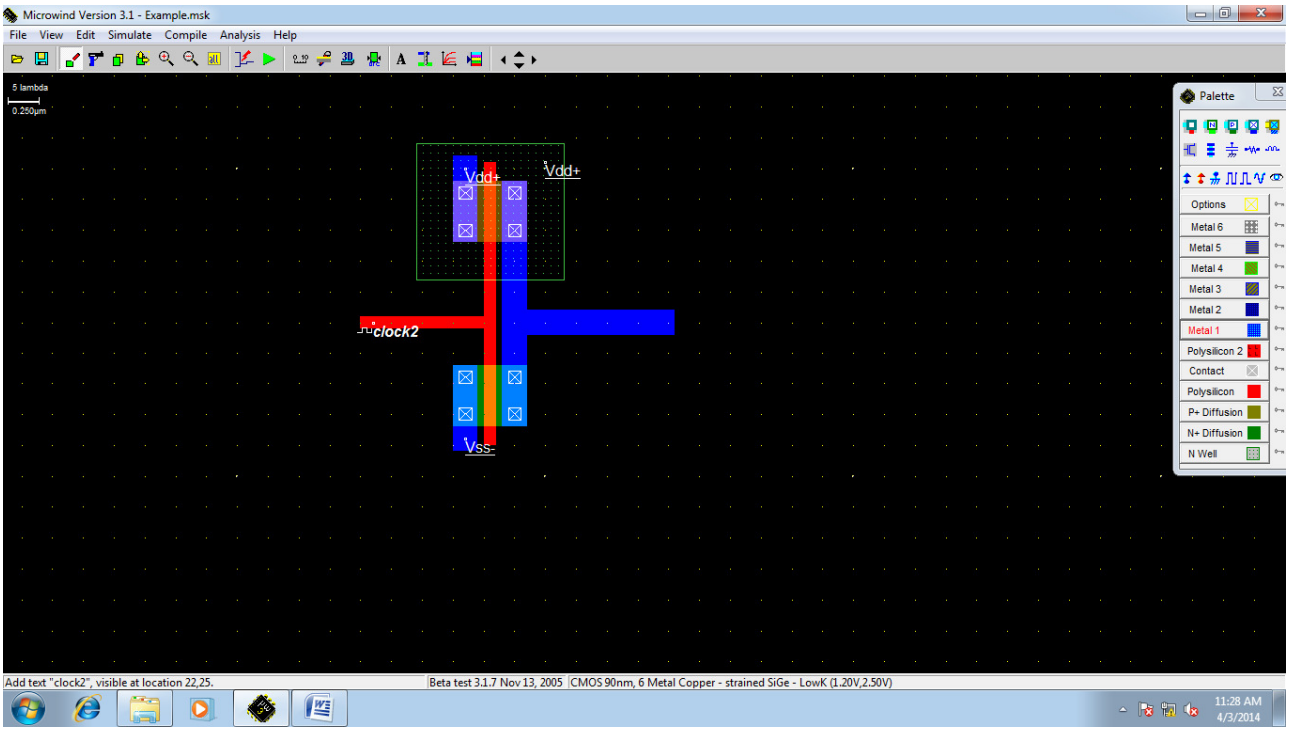




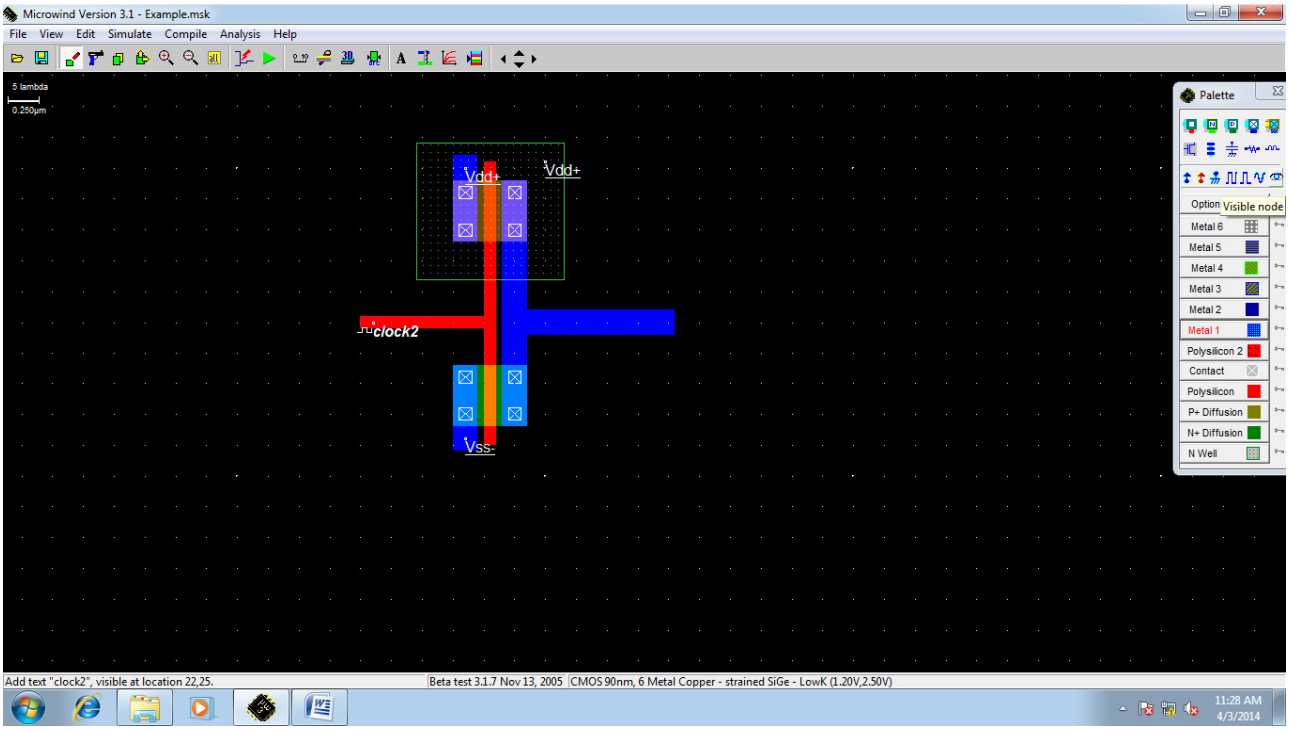
8. Now provide input, by choosing Add a Clock from the Palette, select assign and place it at the input side.

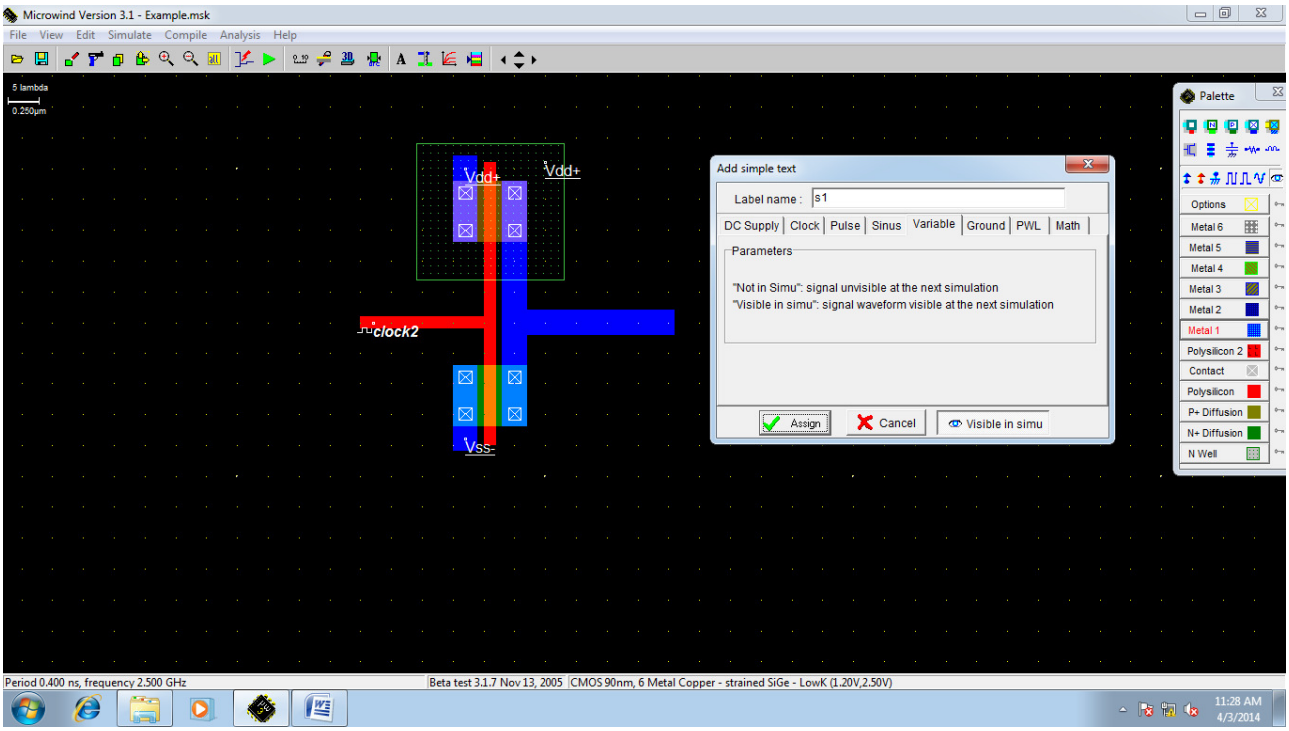


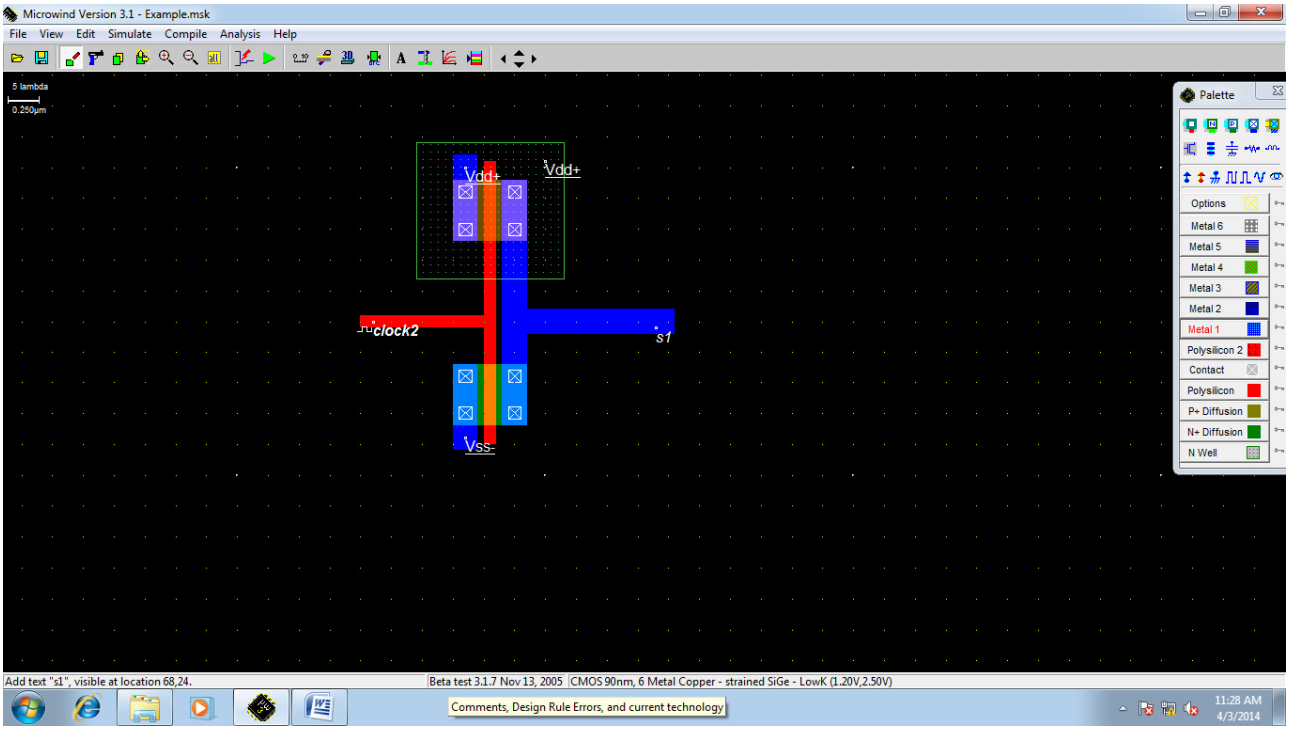

Microwind Version 3.1 - Example.msk
File View Edit Simulate Compile Analysis Help
Vdd+
Vdd+
Vss-
Add a Clock
Label name : clock2
DC Supply Clock Pulse Sinus Variable Ground PWL Math
Parameters
Level 1 (V): 1.200
Level 0 (V): 0.000
Time low (tl) 0.090 ns
Rise time (tr) 0.010 ns
Time high (th) 0.090 ns
Fall time (tf) 0.010 ns
Slower
Faster
~Last Clock
Assign
Cancel
Visible in simu
Palette
Options
Metal 6
Metal 5
Metal 4
Metal 3
Metal 2
Metal 1
Polysilicon 2
Contact
Polysilicon
P+ Diffusion
N+ Diffusion
N Well
No existing text here
Beta test 3.1.7 Nov 13, 2005 CMOS 90nm, 6 Metal Copper - strained SiGe - LowK (1.20V,2.50V)
11:27 AM
4/3/2014


Microwind Version 3.1 - Example.msk
File View Edit Simulate Compile Analysis Help
Vdd+
Vdd+
clock2
Vss-
Palette
Options
Metal 6
Metal 5
Metal 4
Metal 3
Metal 2
Metal 1
Polysilicon 2
Contact
Polysilicon
P+ Diffusion
N+ Diffusion
N Well
Add text "clock2", visible at location 22,25.
Beta test 3.1.7 Nov 13, 2005 CMOS 90nm, 6 Metal Copper - strained SiGe - LowK (1.20V,2.50V)
11:28 AM
4/3/2014

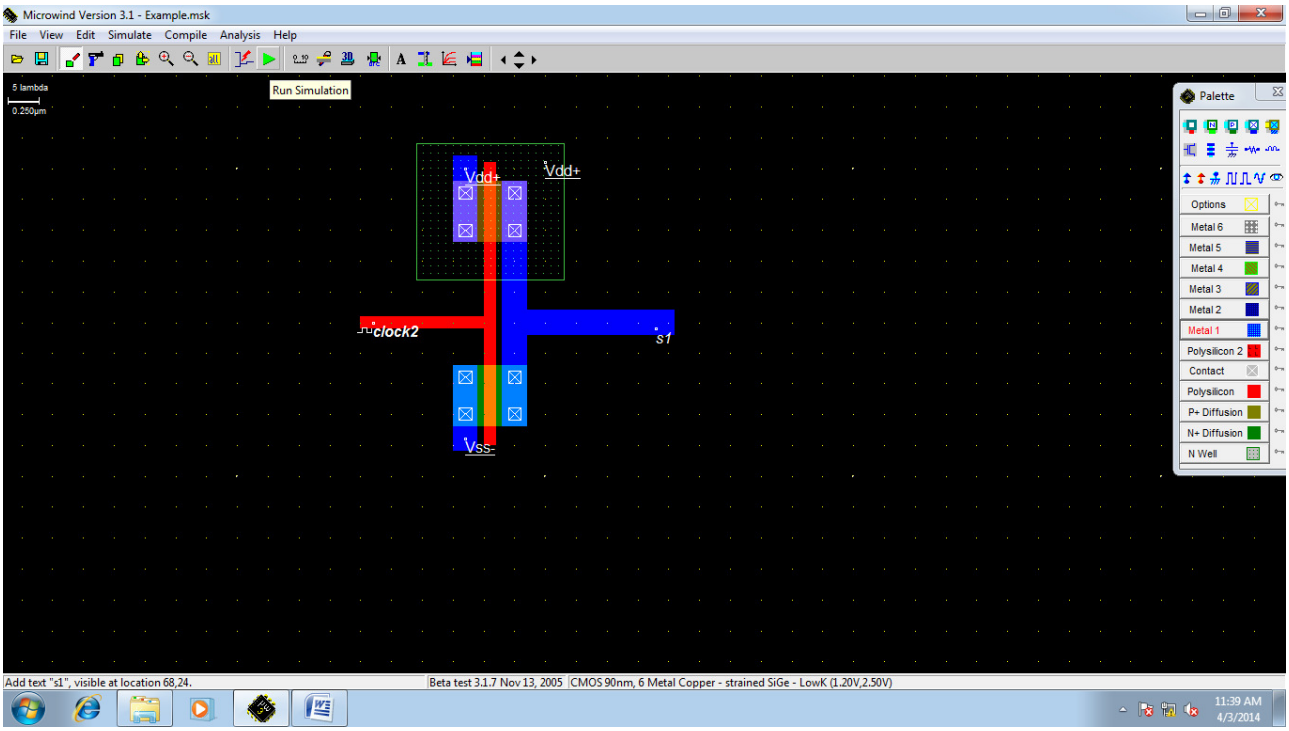
9. **Now output is placed by choosing Visible node symbol from the Palette and click assign to place it at the output port.**



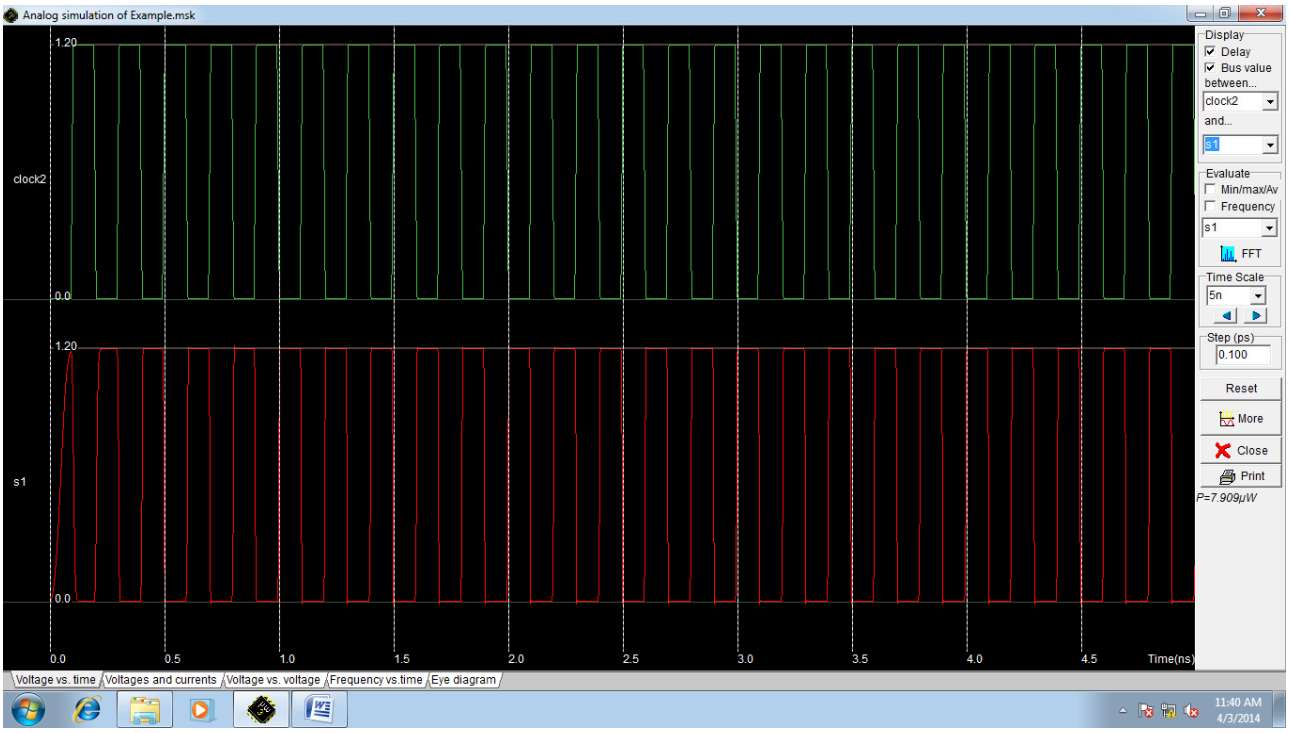




10. Select Run Simulation from the Menu bar.



11. Output waveform is generated in a new window.



RESULT:

Thus the study of simulating layout in Microwind is studied.

EXP: NO: 23	IMPLEMENTATION OF CMOS INVERTER LAYOUT IN MICROWIND
DATE:	

AIM:

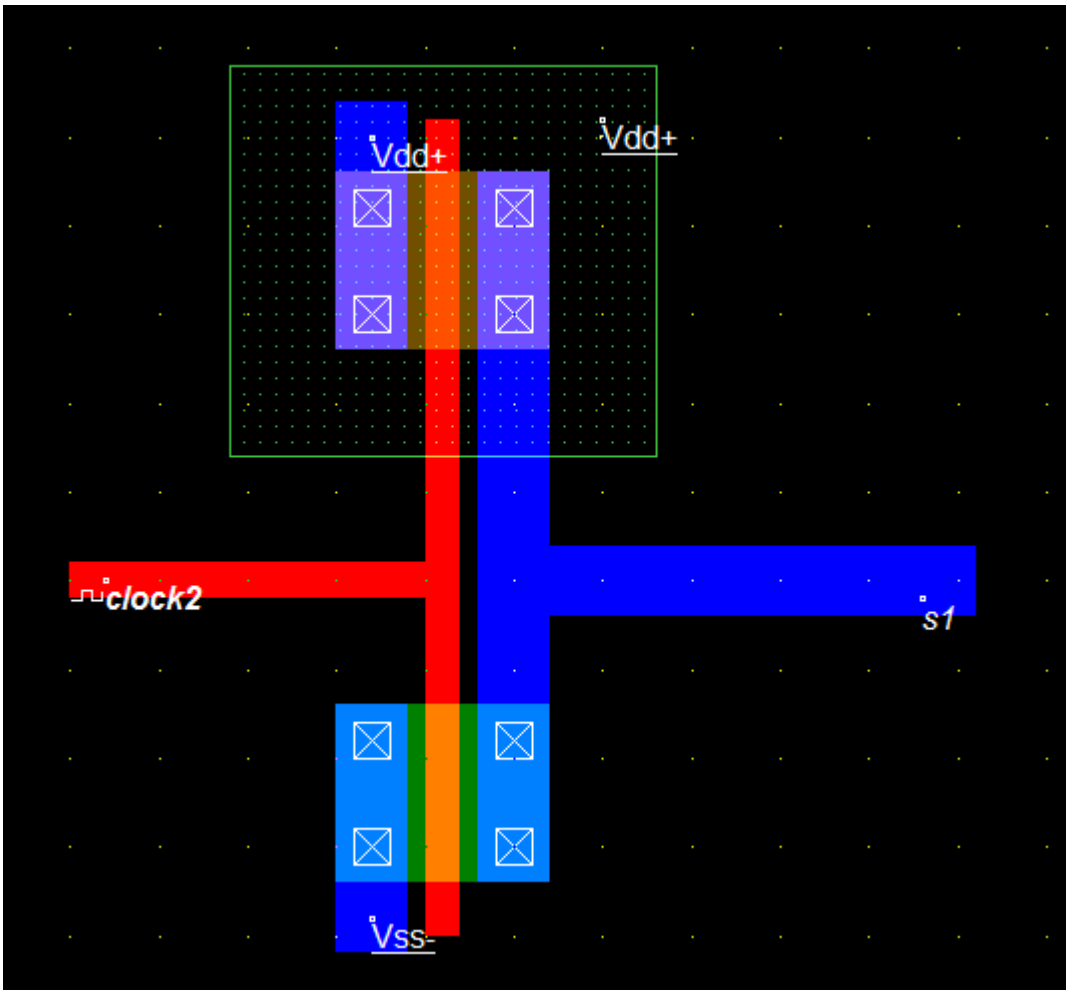
To design, synthesize, simulate and implement the CMOS inverter in layout form using Microwind.

TOOLS REQUIRED:

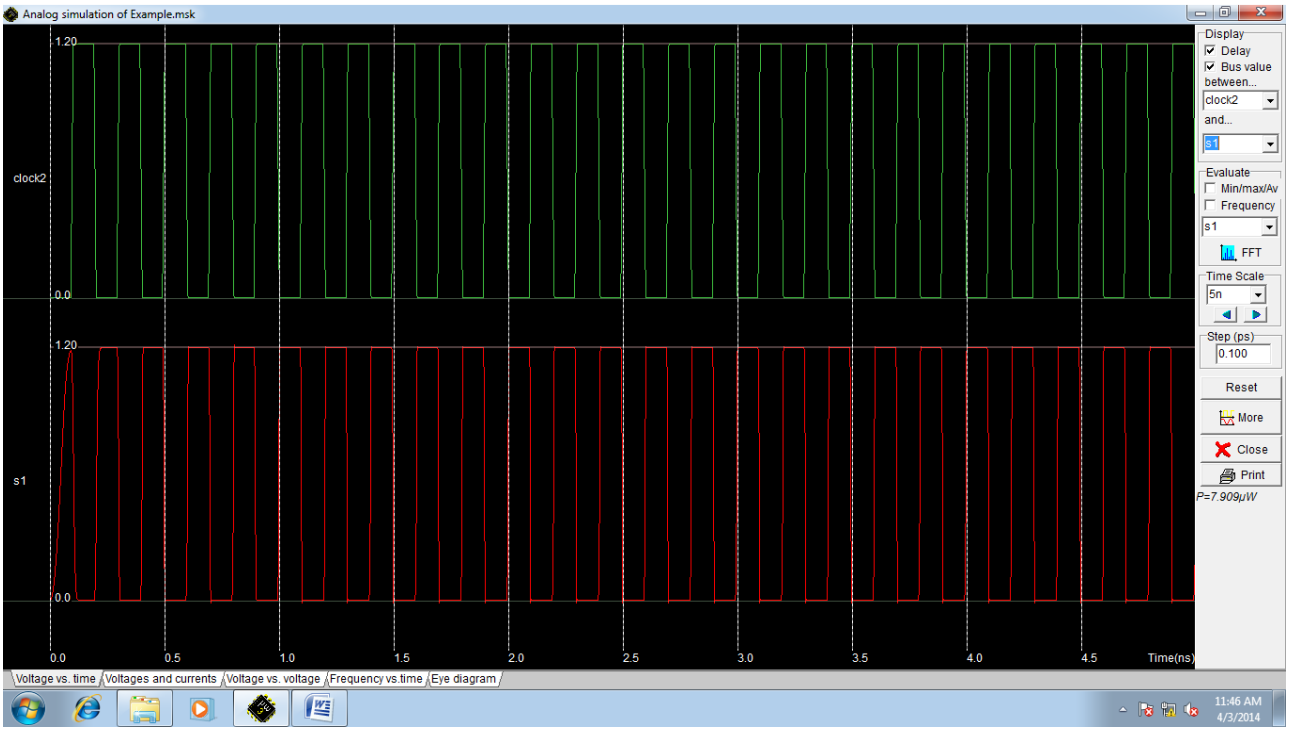
SOFTWARE:

1. MICROWIND V3.1

CIRCUIT DIAGRAM:



OUTPUT:



RESULT:

Thus the CMOS inverter schematic form was designed and simulated using Microwind.

EXP: NO: 24 DATE:	IMPLEMENTATION OF CMOS NAND GATE LAYOUT IN MICROWIND

AIM:

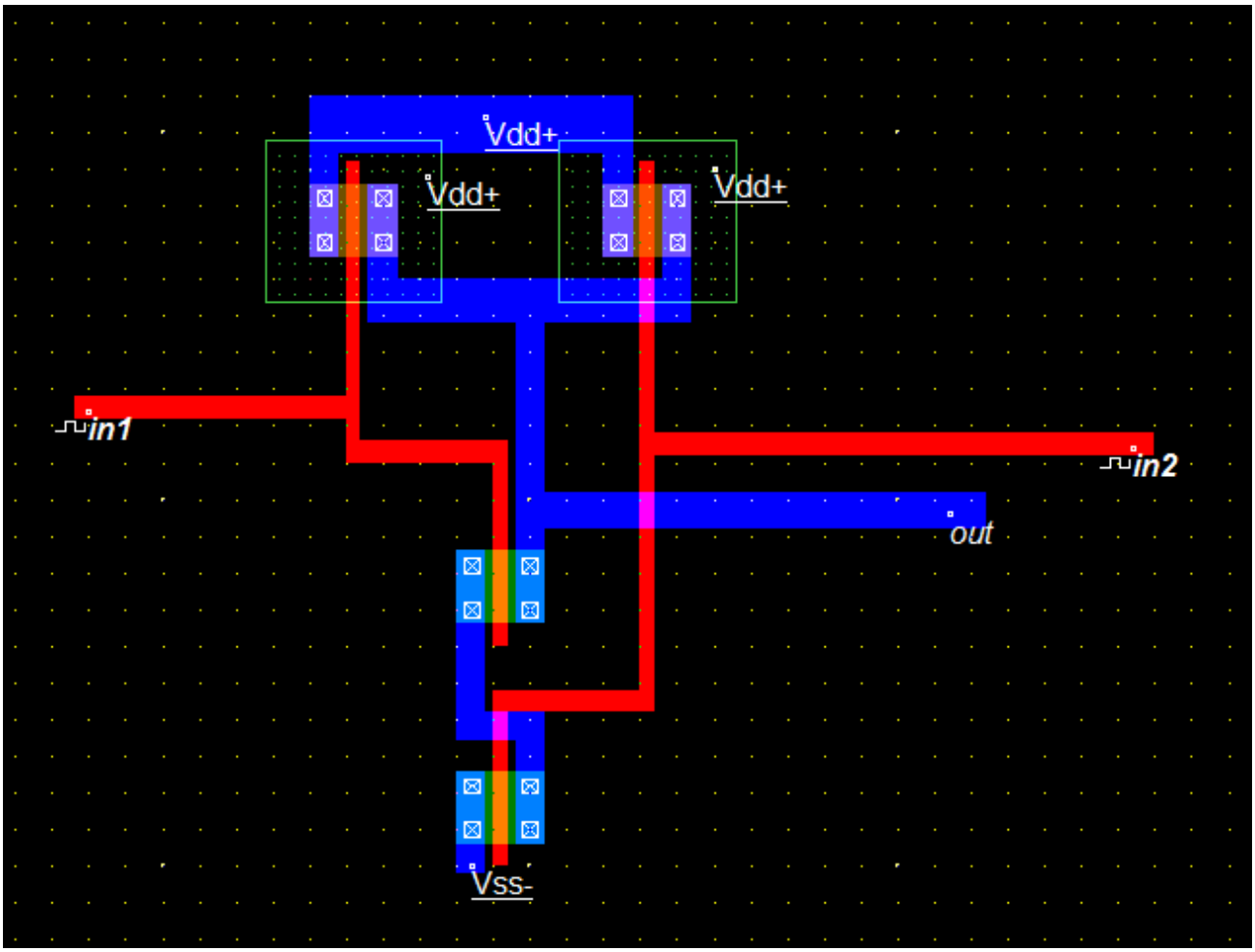
To design, synthesize, simulate and implement the CMOS NAND gate in layout form using Microwind.

TOOLS REQUIRED:

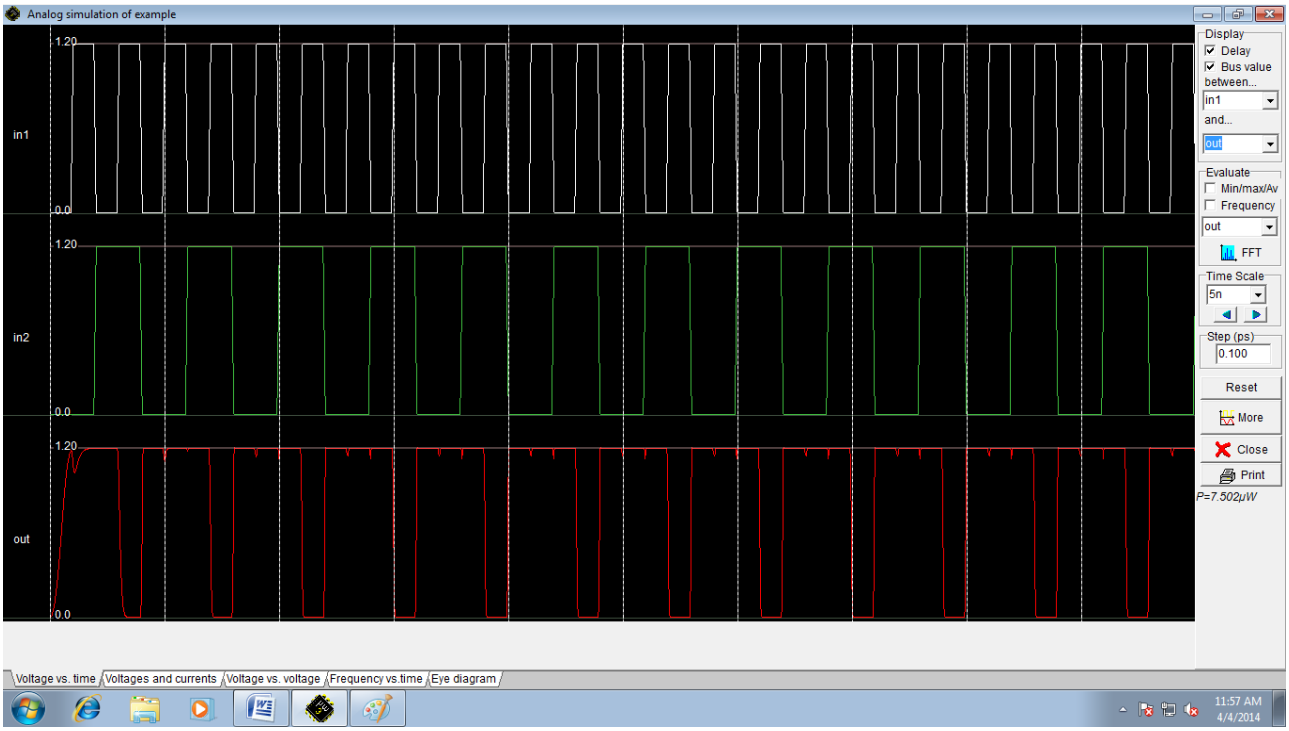
SOFTWARE:

1. MICROWIND V3.1

CIRCUIT DIAGRAM:



OUTPUT:



RESULT:

Thus the CMOS NAND gate schematic form was designed and simulated using Microwind.