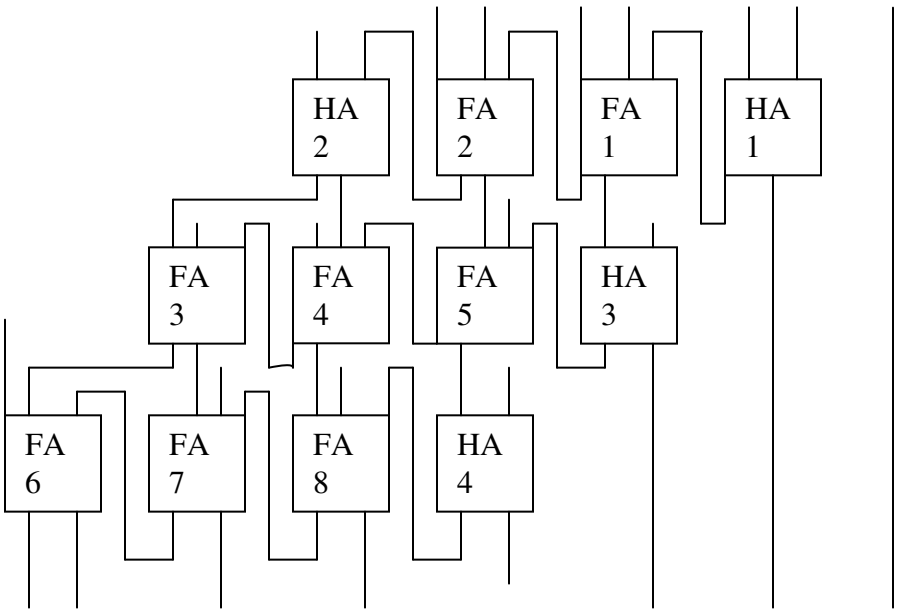
Implementation of 4 bit array multiplier using Verilog HDL and its testing on the Spartan 2 FPGA

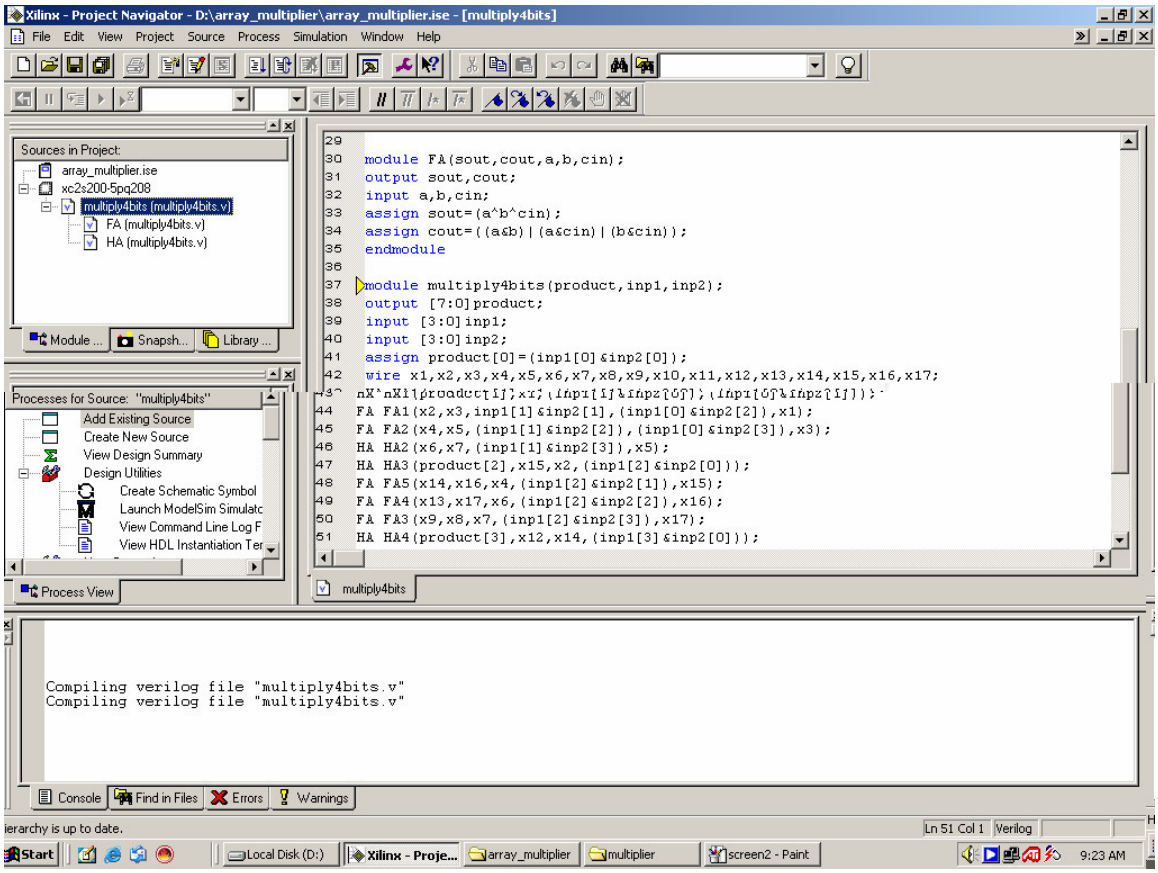
The aim here is to take you through the design and implementation steps of FPGA implementation for 4-bit binary multiplier. The algorithm used here is a simple one that uses repeated addition. Refer to HDL description for Adder and Full Adder given in the text Digital Principles and Applications, 6e by Leach, Malvino and Saha, TMH, 2006.

The gate level diagram of the 4 bit array multiplier was obtained as follows : -
(The unconnected inputs are the combinations of the input bits ANDed in pairs. The exact combinations at each adder box can be found out from the Verilog code)

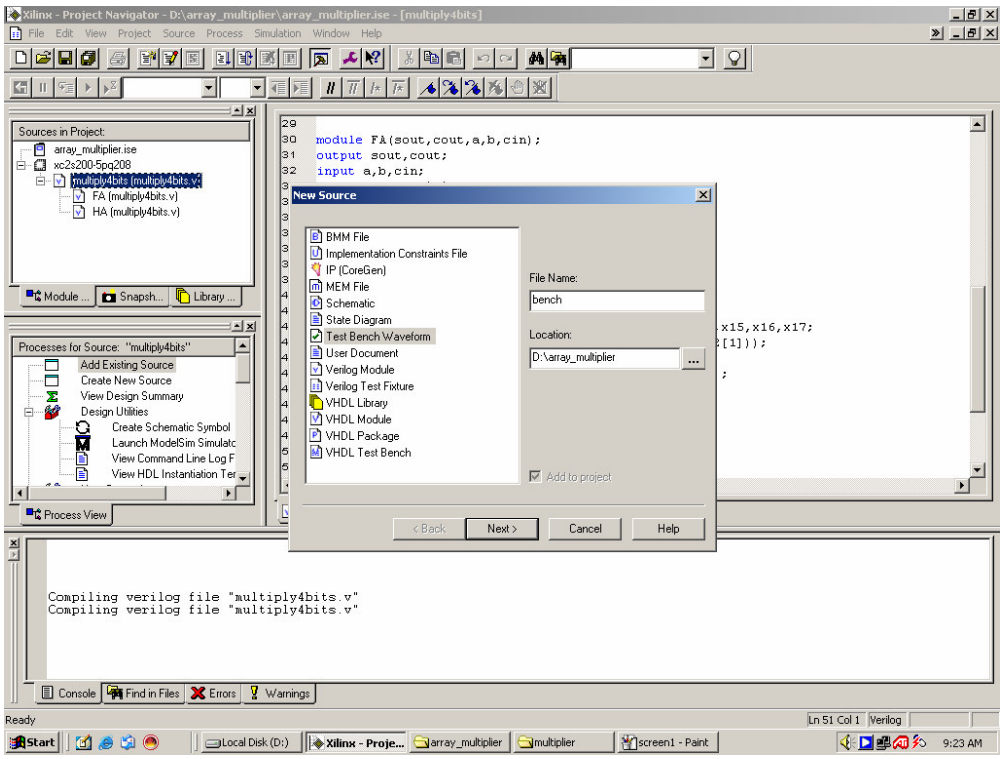


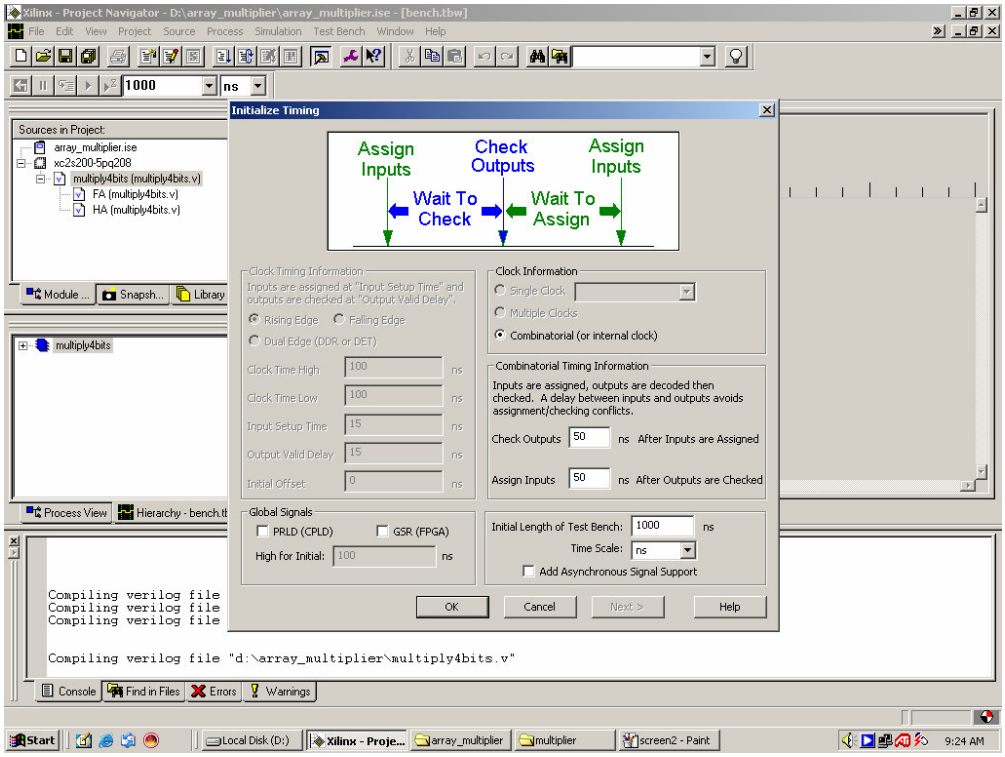
From the diagram , the gate level verilog code is then written in the **Xilinx** text editor :

```
module HA(sout,cout,a,b);
output sout,cout;
input a,b;
assign sout=a^b;
assign cout=(a&b);
endmodule

module FA(sout,cout,a,b,cin);
output sout,cout;
input a,b,cin;
assign sout=(a^b^cin);
assign cout=((a&b)|(a&cin)|(b&cin));
endmodule

module multiply4bits(product,inp1,inp2);
output [7:0]product;
input [3:0]inp1;
input [3:0]inp2;
assign product[0]=(inp1[0]&inp2[0]);
wire x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17;
HA HA1(product[1],x1,(inp1[1]&inp2[0]),(inp1[0]&inp2[1]));
FA FA1(x2,x3,inp1[1]&inp2[1],(inp1[0]&inp2[2]),x1);
FA FA2(x4,x5,(inp1[1]&inp2[2]),(inp1[0]&inp2[3]),x3);
HA HA2(x6,x7,(inp1[1]&inp2[3]),x5);
HA HA3(product[2],x15,x2,(inp1[2]&inp2[0]));
FA FA5(x14,x16,x4,(inp1[2]&inp2[1]),x15);
FA FA4(x13,x17,x6,(inp1[2]&inp2[2]),x16);
FA FA3(x9,x8,x7,(inp1[2]&inp2[3]),x17);
HA HA4(product[3],x12,x14,(inp1[3]&inp2[0]));
FA FA8(product[4],x11,x13,(inp1[3]&inp2[1]),x12);
FA FA7(product[5],x10,x9,(inp1[3]&inp2[2]),x11);
FA FA6(product[6],product[7],x8,(inp1[3]&inp2[3]),x10);
 endmodule
```

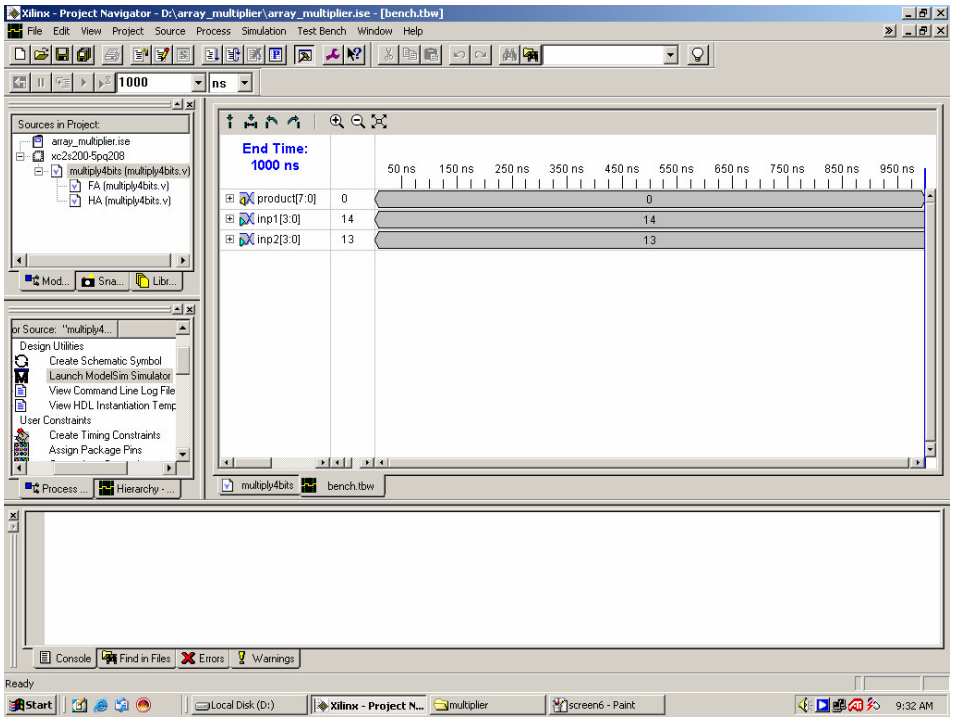


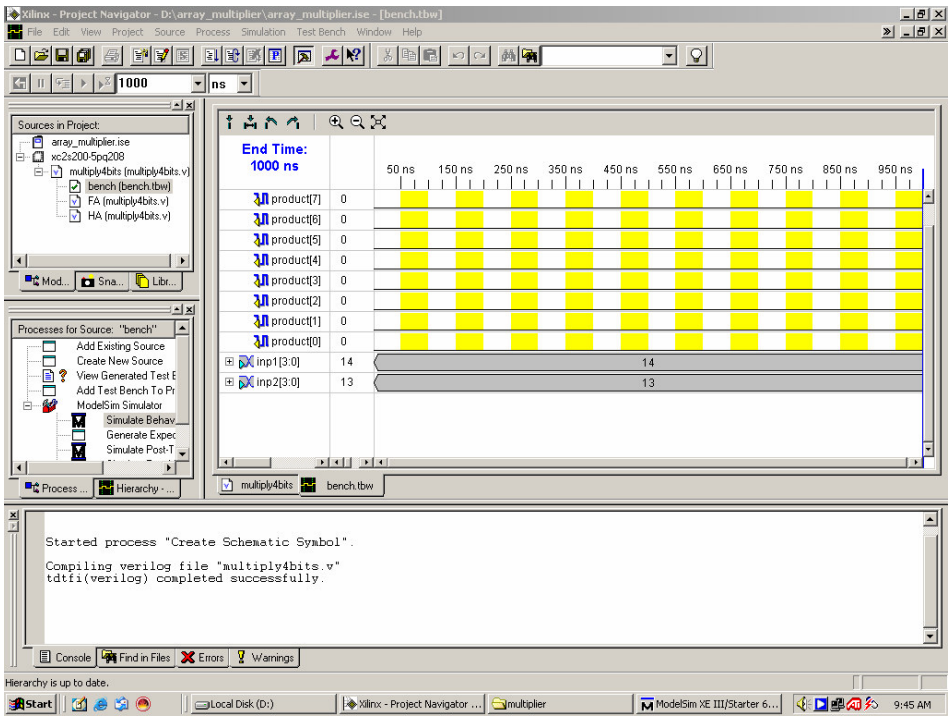
Next the test bench file has to be created to test the design. The test bench file can be created by right clicking on the main module name and selecting the “Add New Source” option.



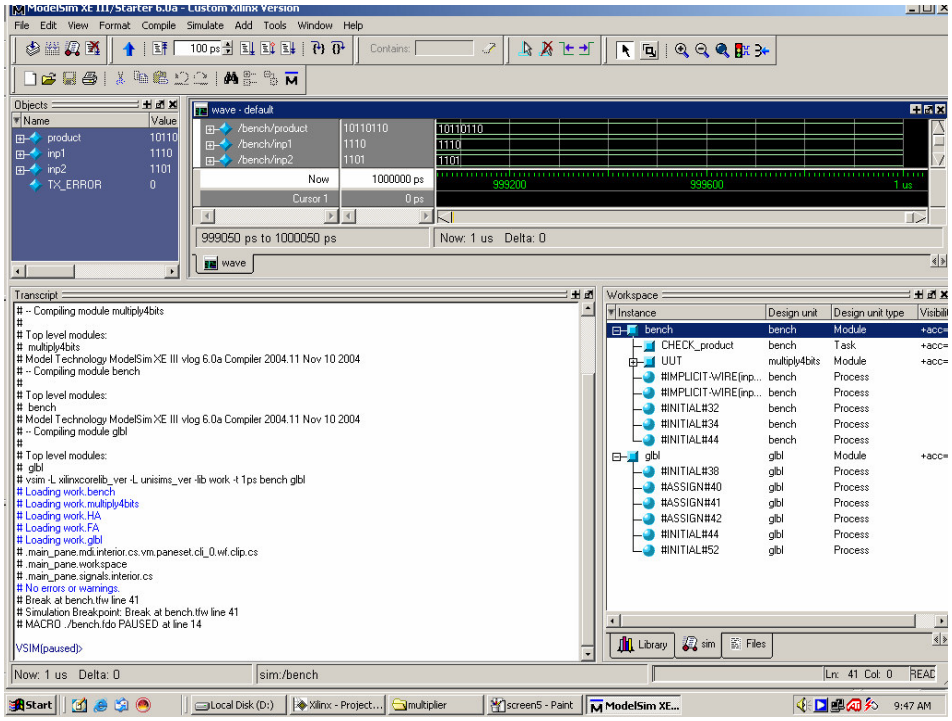


Once the test bench file is created, it is time to apply the inputs in the input box. The inputs can be specified both in binary number format as well as by specifying their waveform. Let for example, the input to given as inp1=14 and inp2=13 :

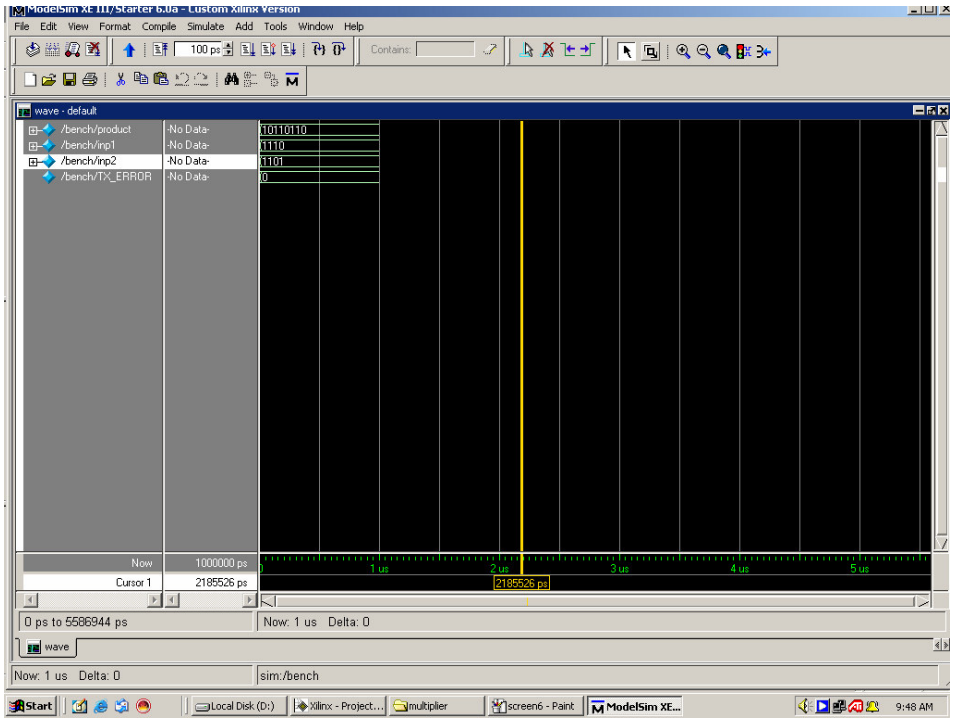




Its time now to simulate the design. To do this the testbench file is saved and from the processes menu in the left the “Simulate Behavioral HDL” option is selected. The code is thus simulated in ModelSim :

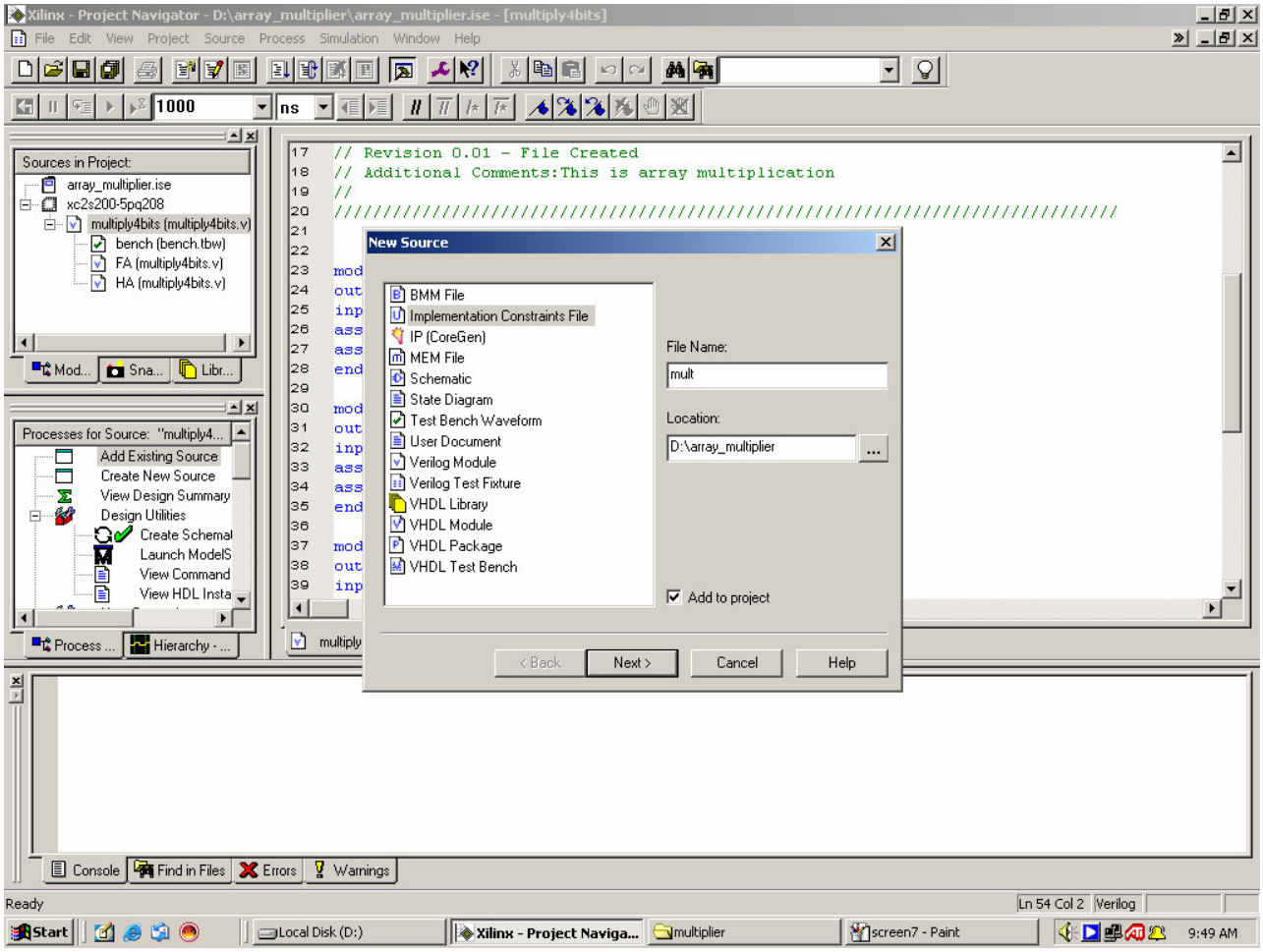


The “wave “ window is then maximized since that is where the output waveform will be coming. Here we see the product = 10110110 in binary = 182 in decimal.

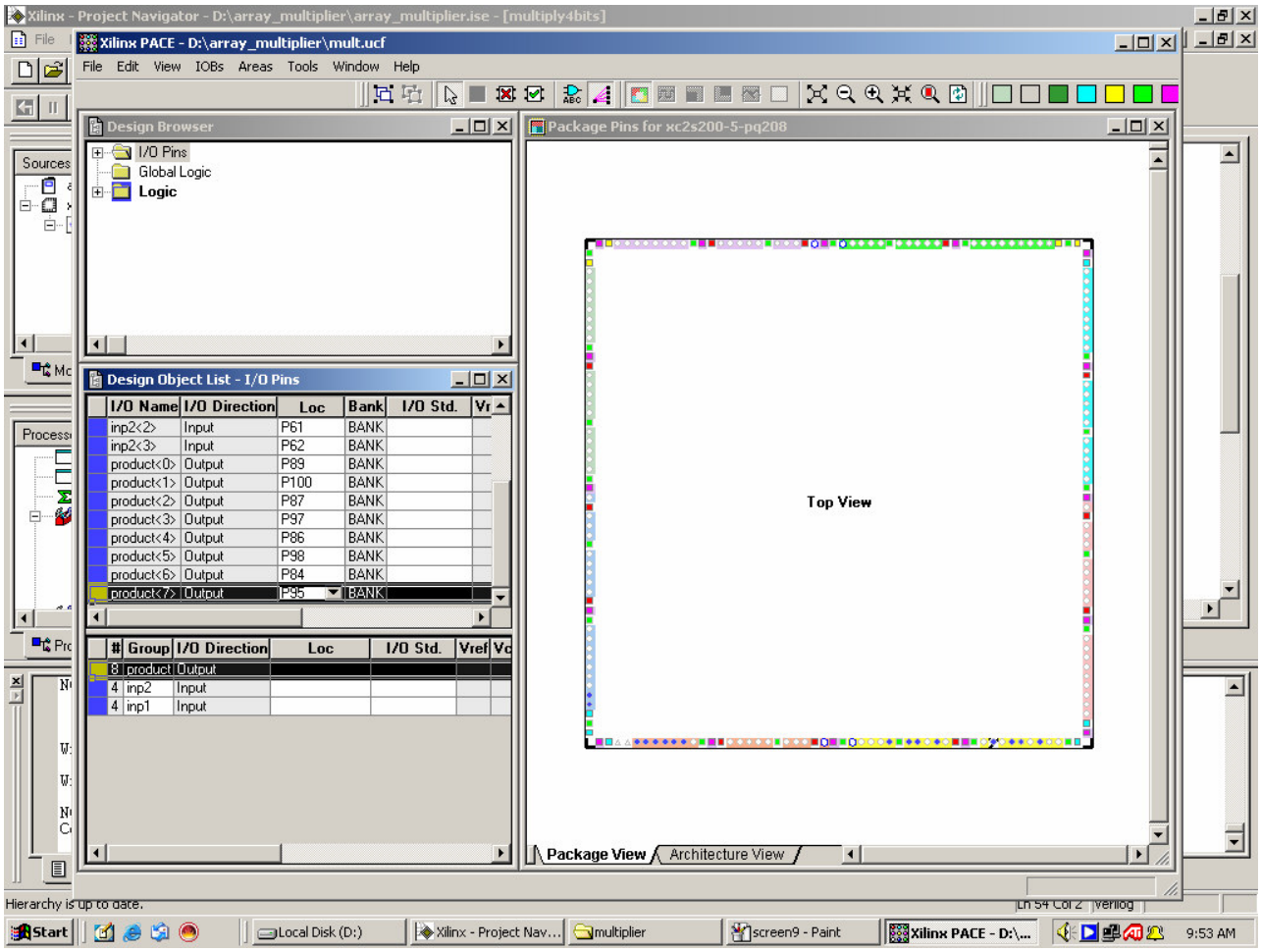


It’s time now to download the design on the Spartan 2 FPGA.

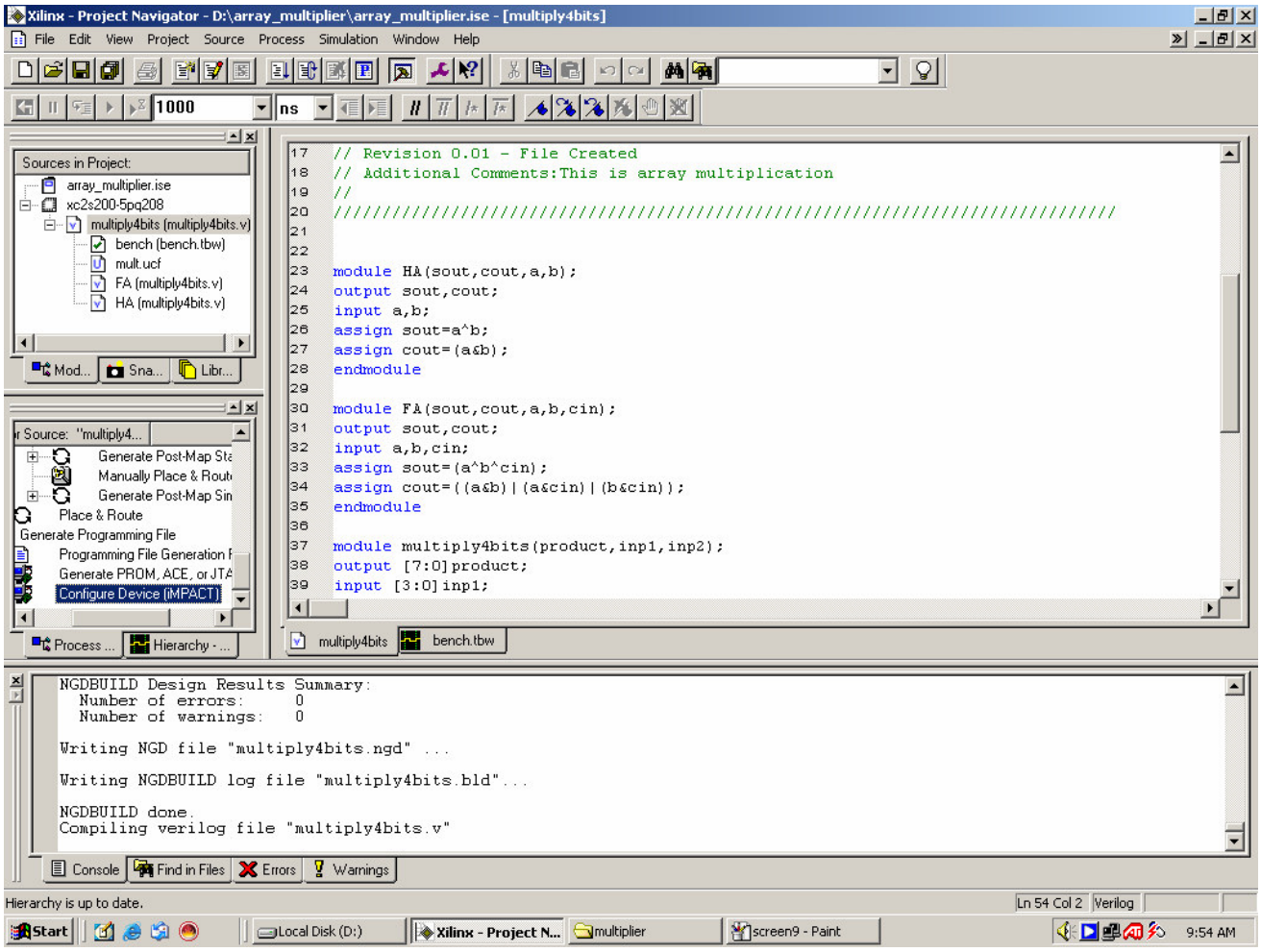
To do so first the “Implementation Constraint File “ (.ucf) file is created by adding a “New Source “ from the main module.



Xilinx Pace opens with a top view of the FPGA package on the right and input boxes for specifying the input and output ports on the left. The pin numbers of the input and the output are obtained from the manual accompanying the device, and entered in the input boxes in the desired order.



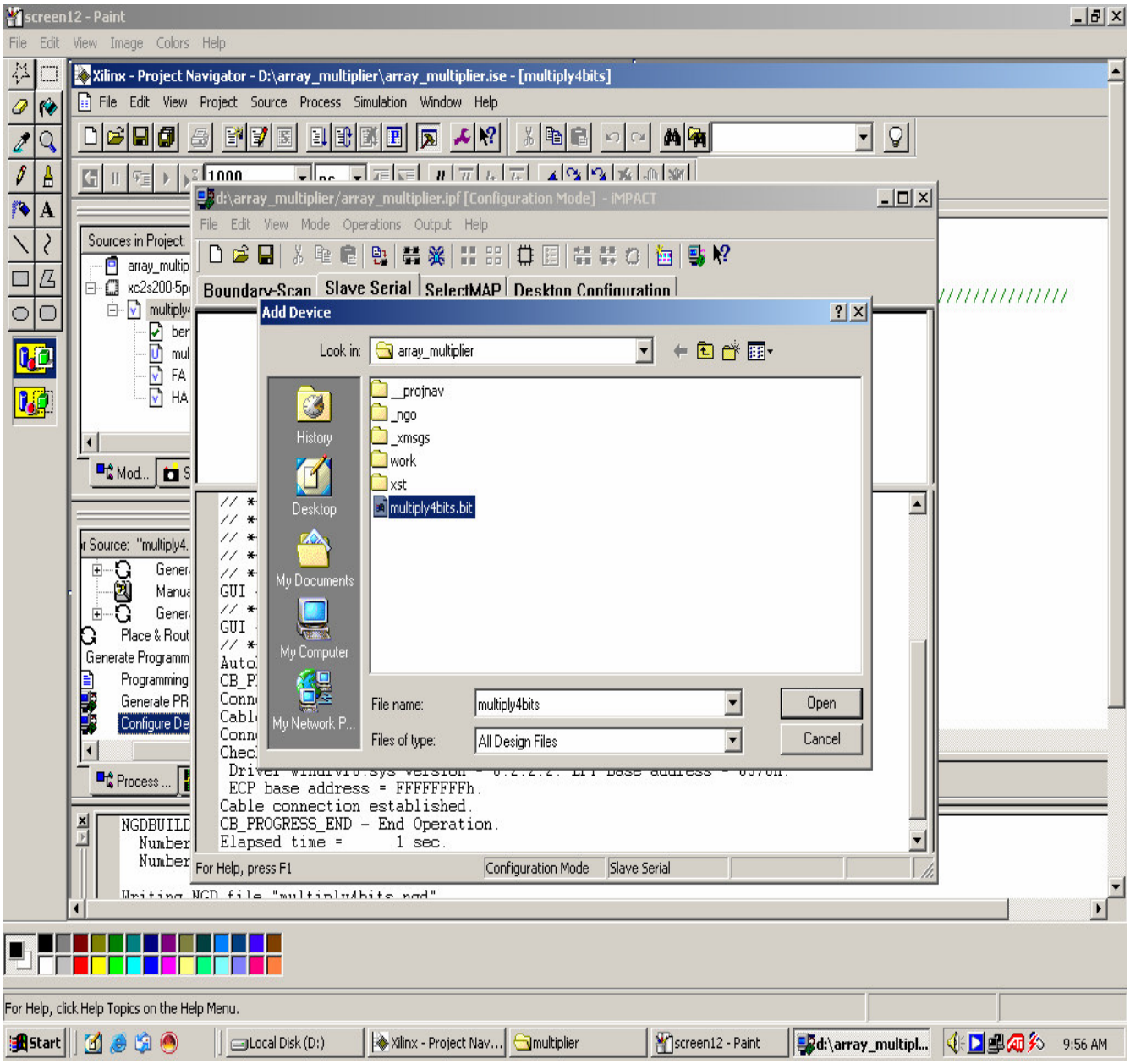
The next step in the download process is configuring the device according to our specifications . To do this the “Configure Device (IMPACT)” option is double clicked in the “ Process” box on the left.



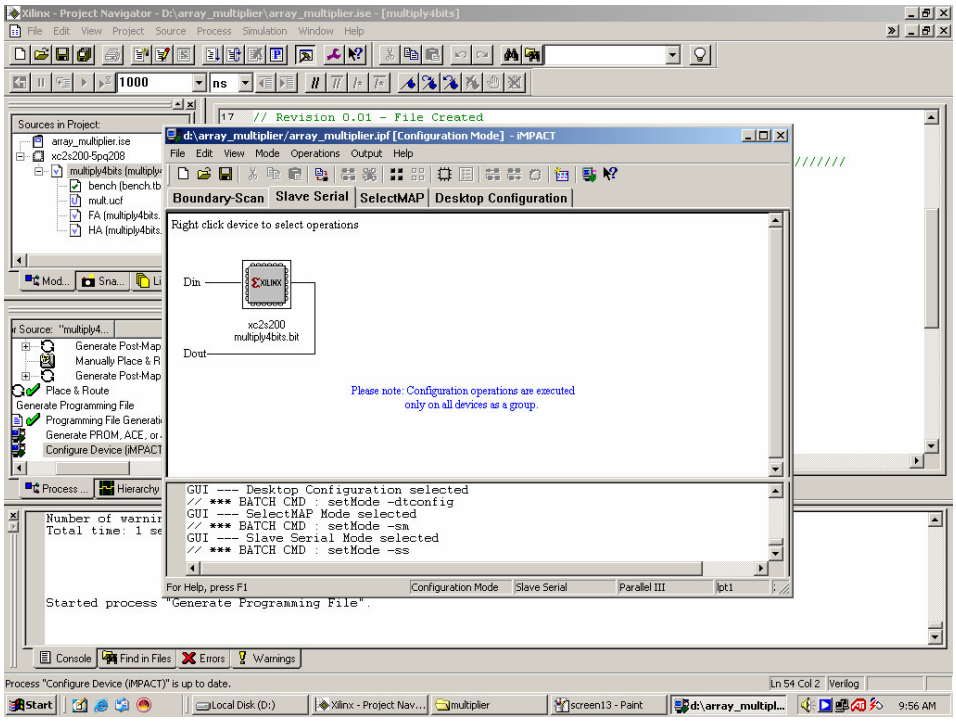
Xilinx then starts configuring the device according to the demands of our program. The progress of the process in shown in the bottom right as percentages.

The different modes of operation include the “Slave Serial mode “ and the “JTAG” (Joint Test Action Group) mode. Here the Slave Serial Mode is used ,and the check box for the Slave Serial Mode is selected when asked the mode has to be specified.

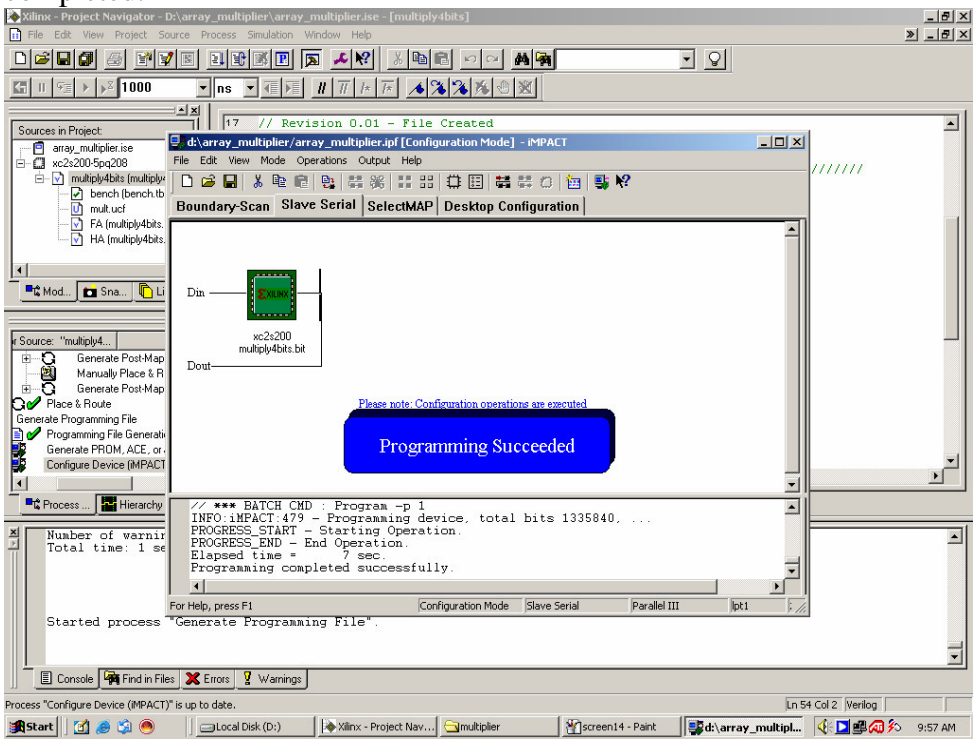
The bit file of our design is selected when the necessary input box comes up :



The device is configured and a top view of the device comes up.



On left clicking on the picture of the device, the last step of the programming is completed.



Now the inputs are to be made available to 4+4 = 8 pins of the FPGA configured as input in previous step and output will be available at 8 pins configured as output. The FPGA unit now behaves like a 4-bit multiplier.