NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

BARREL SHIFTER

Graduation Project
COM-400

Student : Barış İlhan (20030546)

Supervisor : Mehmet Kadir Özakman

Nicosia-2008

ACKNOWLEDGEMENTS

First of all, I would like to thank to my graduation project supervisor Mr.MEHMET KADİR ÖZAKMAN who is a patient and very appreciating personality.He has guided me with a keen interest and helped me by all means. Dr.MEHMET KADİR ÖZAKMAN ; thanks for your continual support.

Secondly, I would like to thank all my teachers in NEAR EAST UNIVERSITY , including faculty of engineering.Speacially to Department Chairman RAHIB ABIYEV and Advisor Mr.KAAN UYAR

Finally, I would like to thank my family, speacially to my father MEHMET ALI ILHAN for providing both moral and financial support that made the during my training and completion of the Project.

ABSTRACT

Using the Behevioral language VHDL and the latest technology and methods to design and generate the programming file for a Barrel Shifter which is used to floating Point arithmetic operations.The Design perform successfully and learn how to create an electronic design.

In this Project, may not get a chance to do electronic design; preferences is to do programming but nowadays,the electronic design process is to write program such as VHDL/Verilog etc.

Barrel Shifter design proved successfull and left is to program device just clicking and ISE tool.

Nowadays,the device somewhere in any lab ,all to do these using internet and program that device.It cost fee.

Table of Contents

LIST OF ABBREVIATIONS

ISE	Integrated Software Environment
FPGA	Field Programmable Gate Array
VHDL	Very high speed integrated circuit Hardware Description Language
MSB	Most Significant Bit
LSB	Low Significant Bit
DUT	Design Under Test
UUT	Unit Under Test
FEC	Forward Error Correction
DLL	Delay-Locked Loop
DCM	Digital Clock Managers
CPLD	Complex Programmable Logic Devices

CHAPTER ONE: INTRODUCTION

The aim of this project is to design simulate and generate the programming code for a 16 bit BARREL SHIFTER to implement into SPARTAN 3- FPGA device.

This Project describes a way to create a common-clock (synchronous) version of shifting a data word by a given Shift Amount, with the depth and width being adjustable within the VHDL code.

The project consists of introduction, two chapters and conclusion;

Chapter two; presents how to make a project by using ISE 9.1i, description about my project and functions of my project.

Chapter three; giving information about what i m use in project and how it works software properties.

Finally, the conclusion section presents the important results obtained within the project.

CHAPTER TWO: BARREL SHIFTER WITH THE VHDL DESIGN

1.1. Design Description

Describing design is BARREL SHIFTER.

A barrel shifter is a digital circuit that can shift a data word by a specified number of bits. It can be implemented as a sequence of multiplexers. In this implementation, the output of one MUX is connected to the input of the next MUX in a way that depends on the shift distance. The number of multiplexers required is $n*\log_2(n)$, for an n bit word. Four common word sizes and the number of multiplexers needed are listed below:

- 64-bit — $64 * \log_2(64) = 64 * 6 = 384$
- 32-bit — $32 * \log_2(32) = 32 * 5 = 160$
- 16-bit — $16 * \log_2(16) = 16 * 4 = 64$
- 8-bit — $8 * \log_2(8) = 8 * 3 = 24$

For example a four-bit barrel shifter, with inputs A, B, C and D. The shifter can cycle the order of the bits ABCD. That is, it can 'shift' all of the outputs up to three positions to the right (and thus make any cyclic combination of A, B, C and D). The barrel shifter has a variety of applications, including being a vital component in microprocessors (alongside the ALU).

1.1.1. Implementation

Often, the barrel shifter is implemented as a cascade of parallel 2x1 multiplexers. For a four-bit barrel shifter, an intermediate signal is used which shifts by two bits, or passes the same data, based on the value of S[1]. This signal is then shifted by another multiplexer, which is controlled by S[0]:

```
im  = IN, if S[1] == 0
    = IN << 2, if S[1] == 1
OUT = im, if S[0] == 0
    = im << 1, if S[0] == 1
```

Larger barrel shifters have additional stages.

1.1.2. Uses

The barrel shifter is used in floating-point arithmetic hardware. For a floating-point add or subtract operation, the mantissa of the numbers must be aligned, which requires shifting the smaller number to the right, increasing its exponent, until it matches the exponent of the larger number. This is done by subtracting the exponents, and using the barrel shifter to shift the smaller number to the right by the difference, in one cycle. If a simple shifter were used, shifting by *n* bit positions would require *n* clock cycles.

1.2. Design Flow

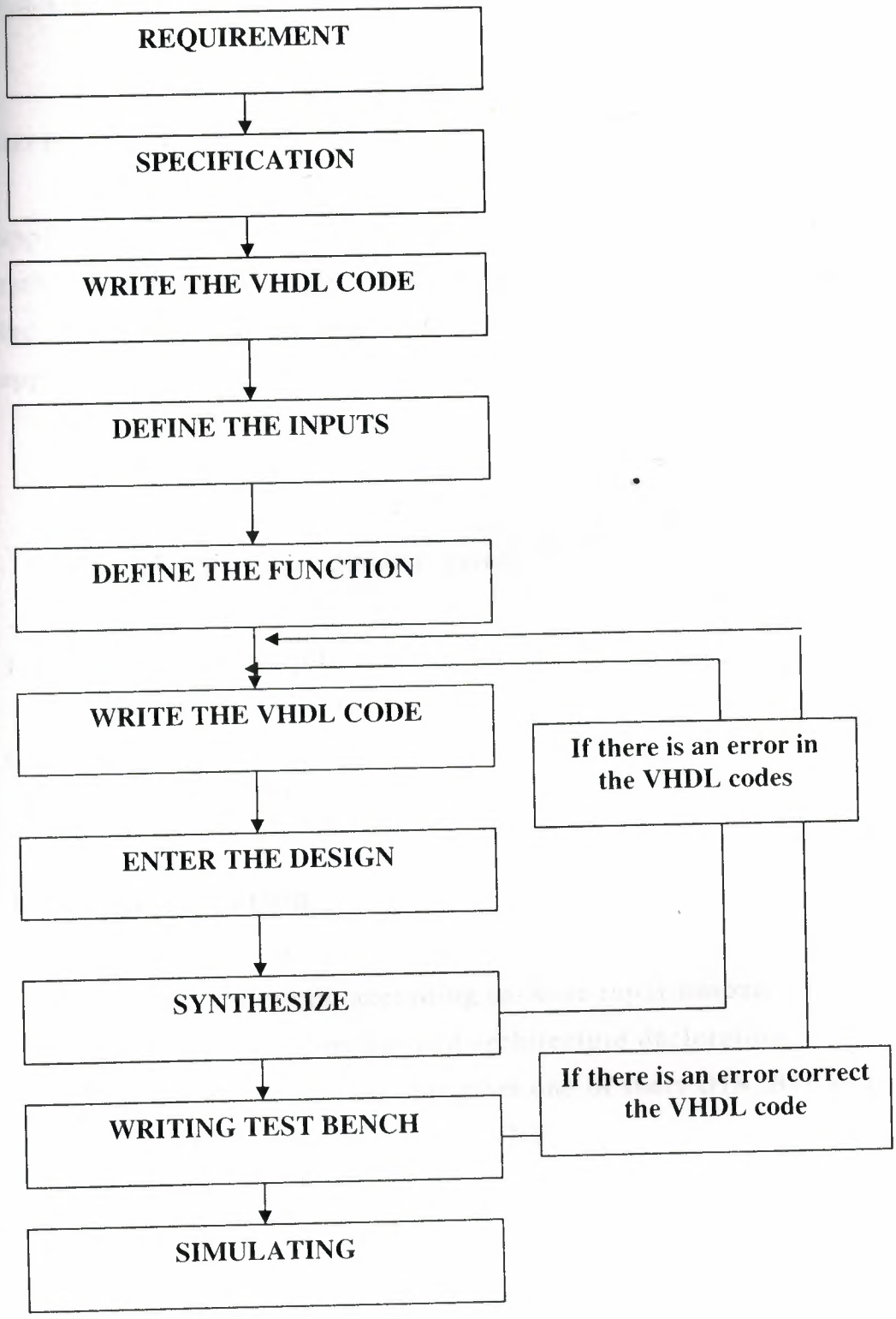

Figure.1 Design Process

1.2.1. Requirement

1.2.1.1. Hardware requirement

Using FGPA(Field Programmable Gate Array) chip that XC3S200 device from family of SPARTAN-3.

FGPA's have traditionally found use in high-speed custom digital applications where designs tend to be more constrained by performance rather than cost.The explosion of integration and reduction in price has led to the more recent widespread use of FPGAs in common embedded applications.

SPARTAN-3 has ;

- 200.000 gates

1.2.1.2. Software Requirement

VHDL (Very high speed integrated circuit Hardware Description Language)

Design used XİLİNX 9.1i

1.2.2. Specification

- Writing the program according to these input-output.
- Using entity,input-output and architecture decleration.
- Defining two signal this program one of them DIN_BIT and DOUT_BIT the other signal is S_INT.

1.2.3. Define the Inputs and Outputs

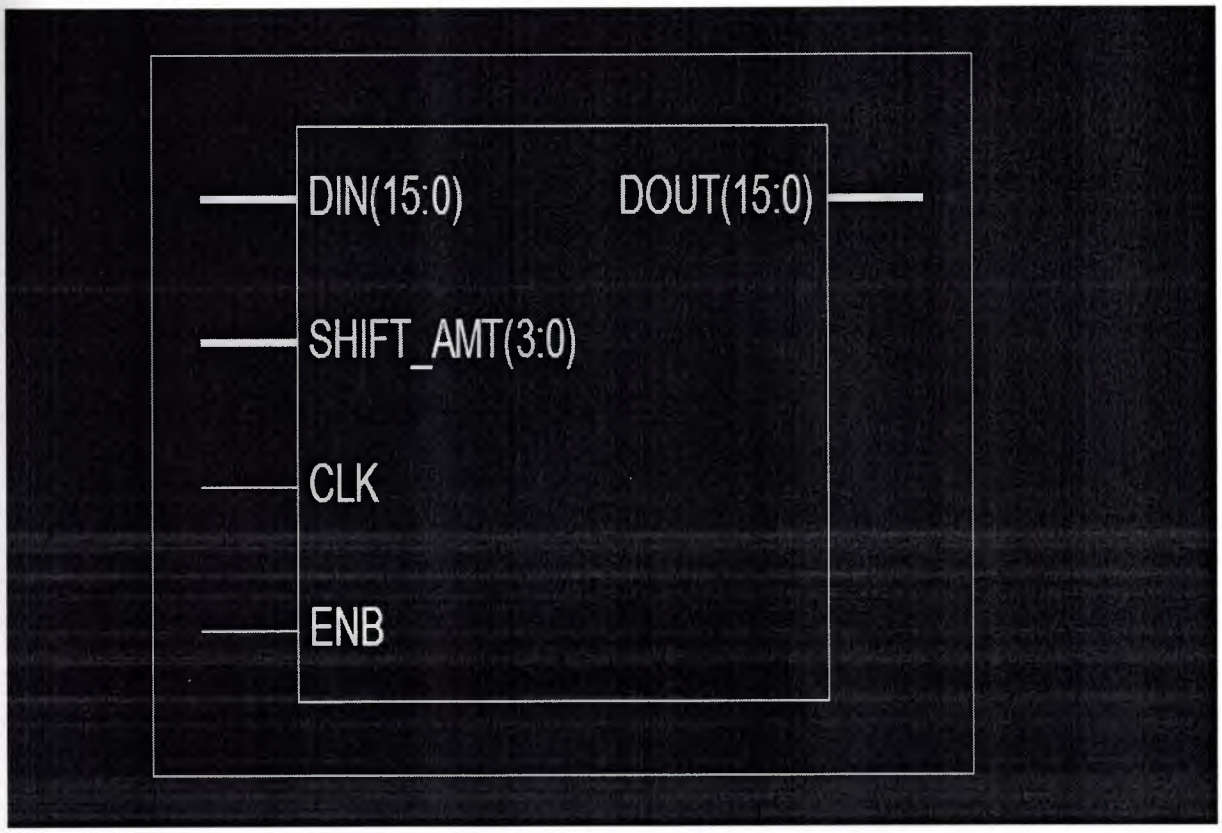

Figure.2 Block Diagram

The inputs and outputs of the BARREL SHIFTER:

Inputs:

- DIN
- CLOCK
- SHIFT_AMT

DIN : 16 BIT ENTERING DATA TO BARREL SHIFTER

CLK : 1 BIT SYSTEM CLOCK

SHIFT_AMT : 4 BIT SHIFT AMOUNT

Outputs:

- DOUT

DOUT : 16 BIT OUTPUT DATA

1.2.4. The Functions of Barrel SHIFTERS

In design data input is rotated right by 4 bit Shift Amount and gives me the Data Output.The rotation function;

- Synchronous with the clock
- Bits are shifted the desired number of bit positions in a single clock cycle
- So that it is a fast operation because of we can shift the data as much as we want in a single clock cycle.

1.2.5.Enter the Design

Design used to entity section of the VHDL code to implement the inputs and outputs.Using the architecture section of the VHDL code to implement the processes.

1.2.5.1. Create an HDL Source

In this section, creating the top-level HDL file for design. Determine the language that i wish to use for the BARREL_SHIFT. Then, continue either to the "Creating a VHDL Source".

1.2.5.2. Creating a VHDL Source

We open the xilinx 9.1 i and we select *file* and *new Project* as figure 3.

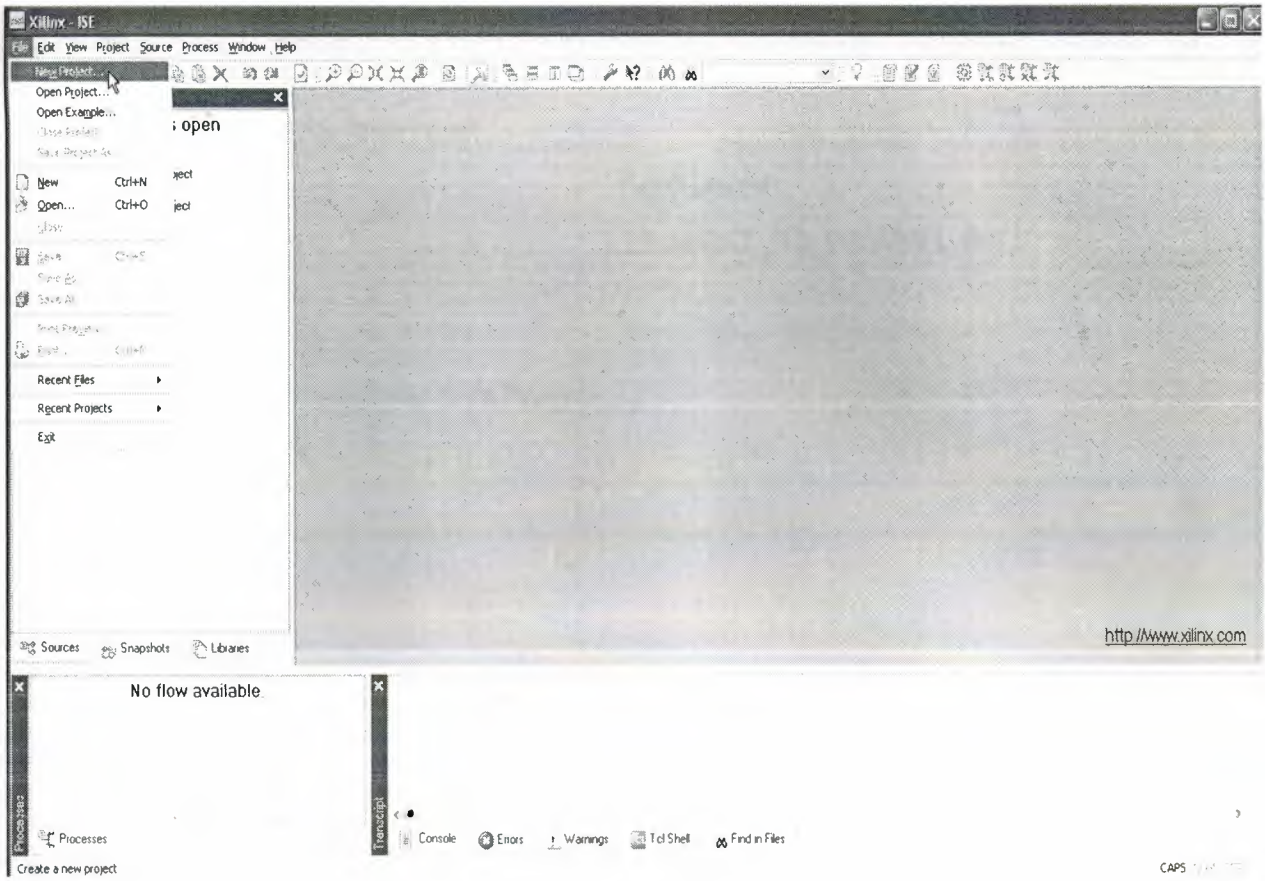

Figure.3 Creating project

And we are writing as Project name '*BARRREL_SHIFT*' as figure 4.

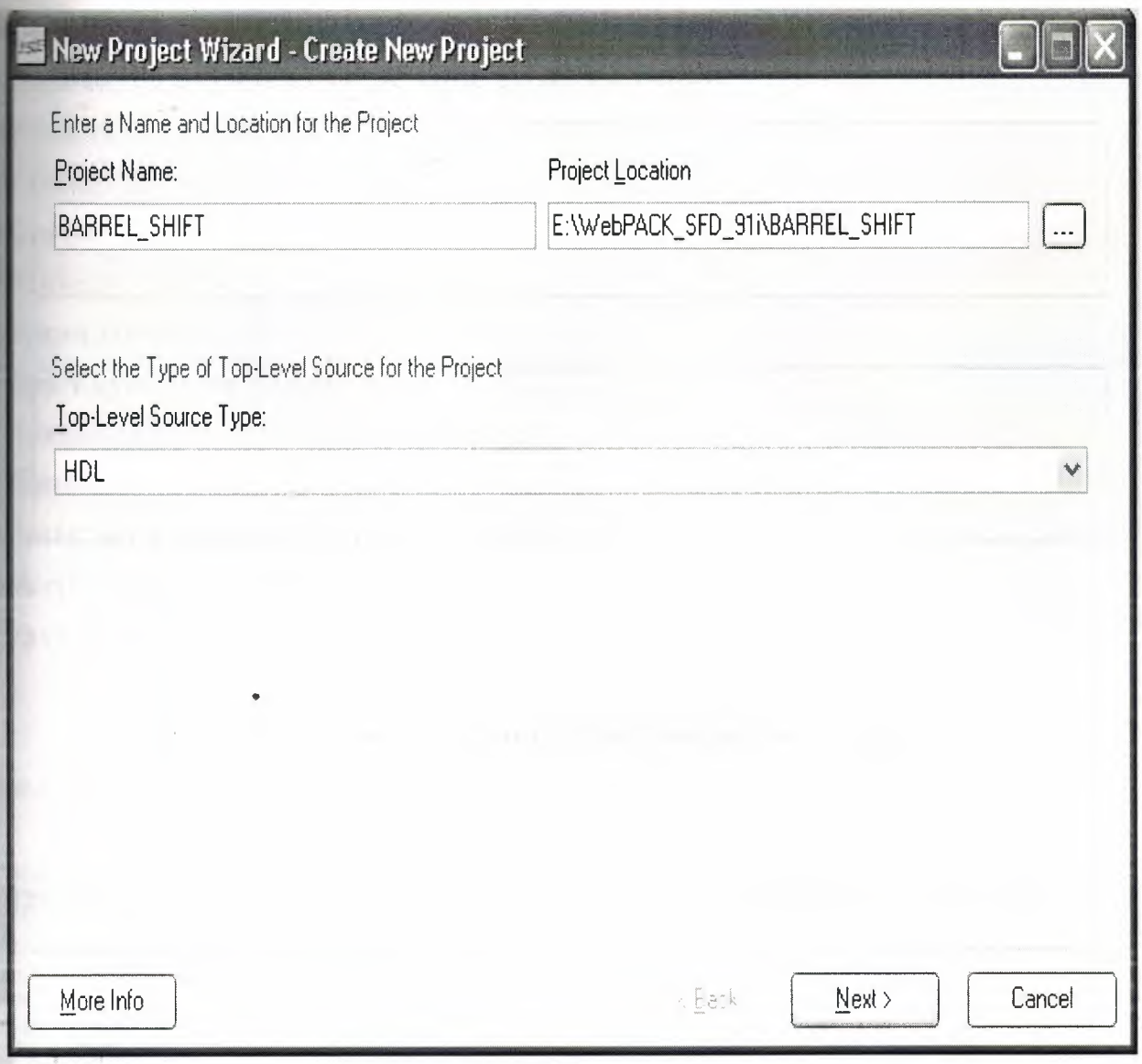

Figure.4 Entering project name

And click next.Then it shows the Device properties;new ISE project which will target the FPGA device on the SPARTAN3

To create a new project:

1. Select File > New Project... The New Project Wizard appears.
2. Type BARREL_SHIFT in the Project Name field.
3. Enter or browse to a location (directory path) for the new project. A BARREL_SHIFT
subdirectory is created automatically.
4. Verify that HDL is selected from the Top-Level Source Type list.

5. Click Next to move to the device properties page.

6. Fill in the properties in the table as shown below:

♦Product Category: All

♦Family: Spartan3

♦Device: XC3S200

♦Package: FT256

♦Speed Grade: -4

♦Top-Level Source Type: HDL

♦Synthesis Tool: XST (VHDL/Verilog)

♦Simulator: ISE Simulator (VHDL/Verilog)

♦Preferred Language: Verilog (or VHDL)

♦Verify that Enable Enhanced Design Summary is selected.

Leave the default values in the remaining fields.

When the table is complete, your project properties will look like the following figure 5.

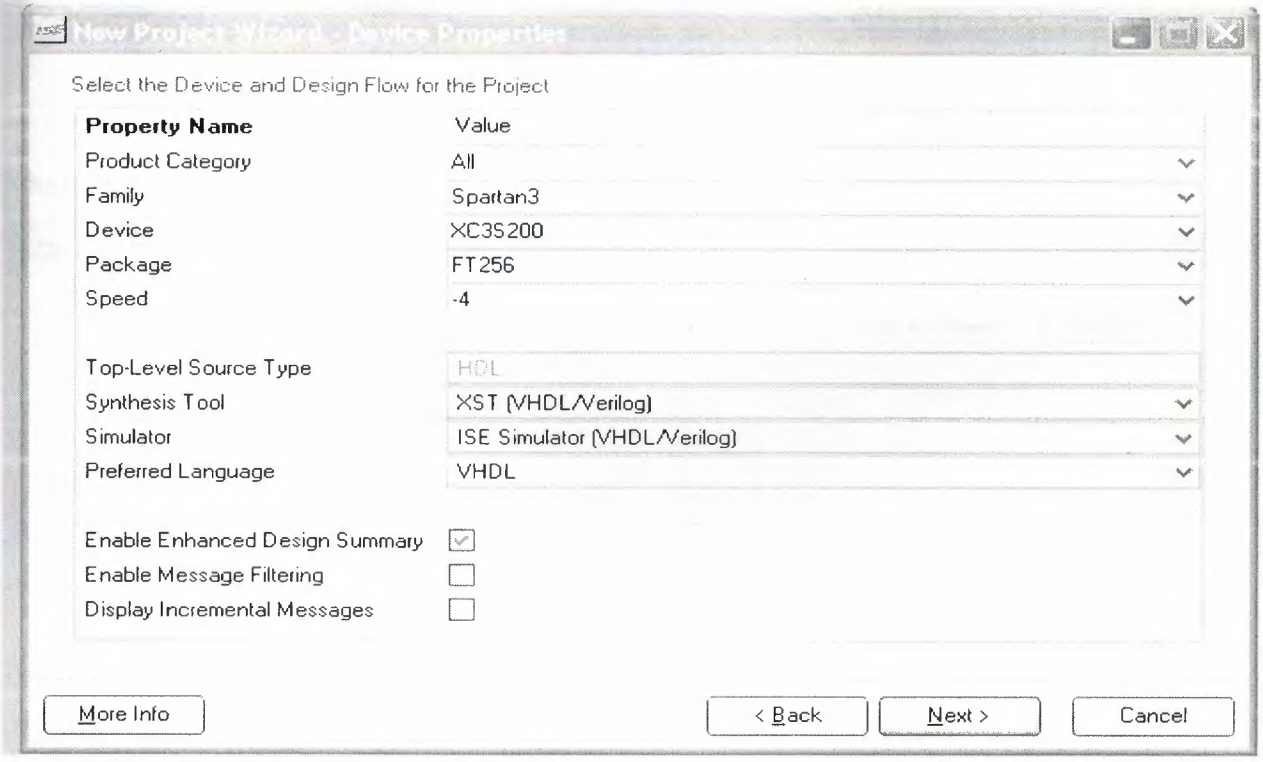

Figure.5 Device properties

Then click next.

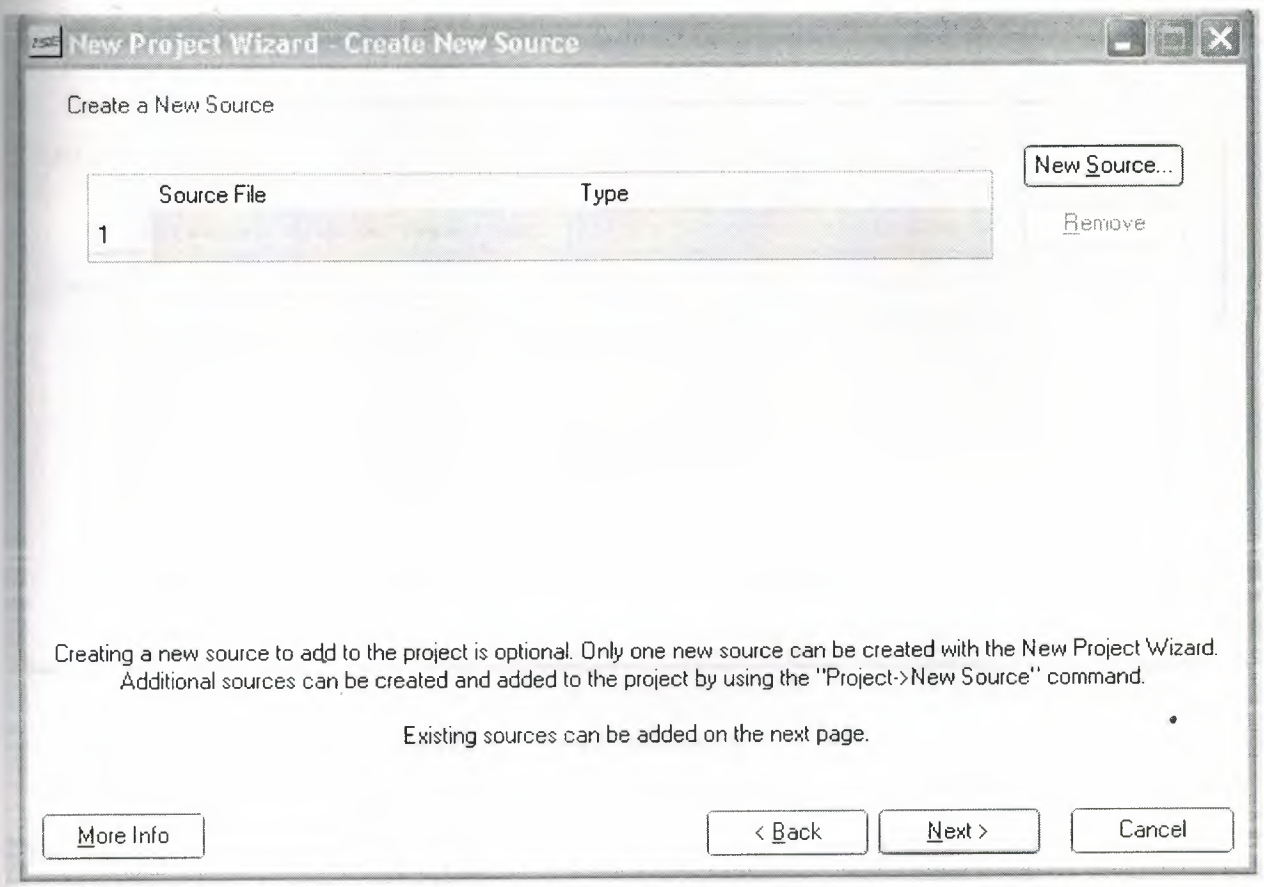

Figure.6 Creating project

Then we again click next as figure 7.

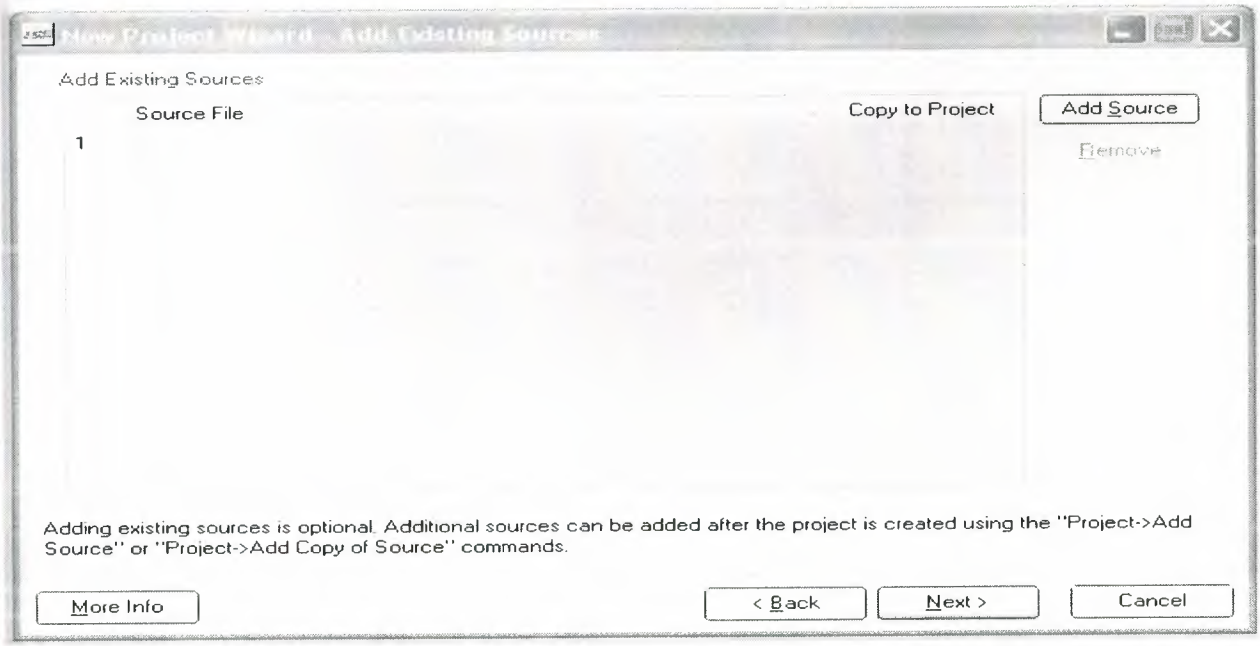

Figure.7 Adding sources

Again click next.After that; it shows the our Project summary as below;

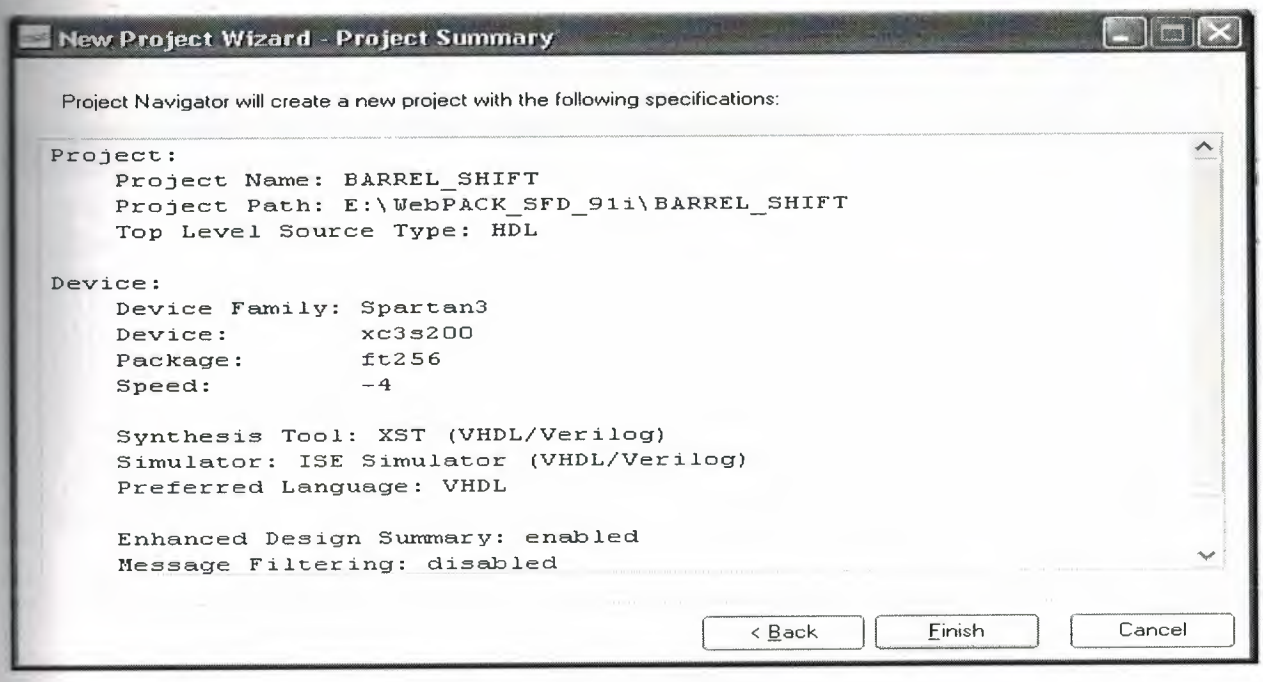

Figure.8 Project summary

Finally we click finish and we adding new source at VHDL module as below;

Figure.9 Creating new source

After that we type BARREL_SHIFT as file name and we select the VHDL Module then we click next as below;

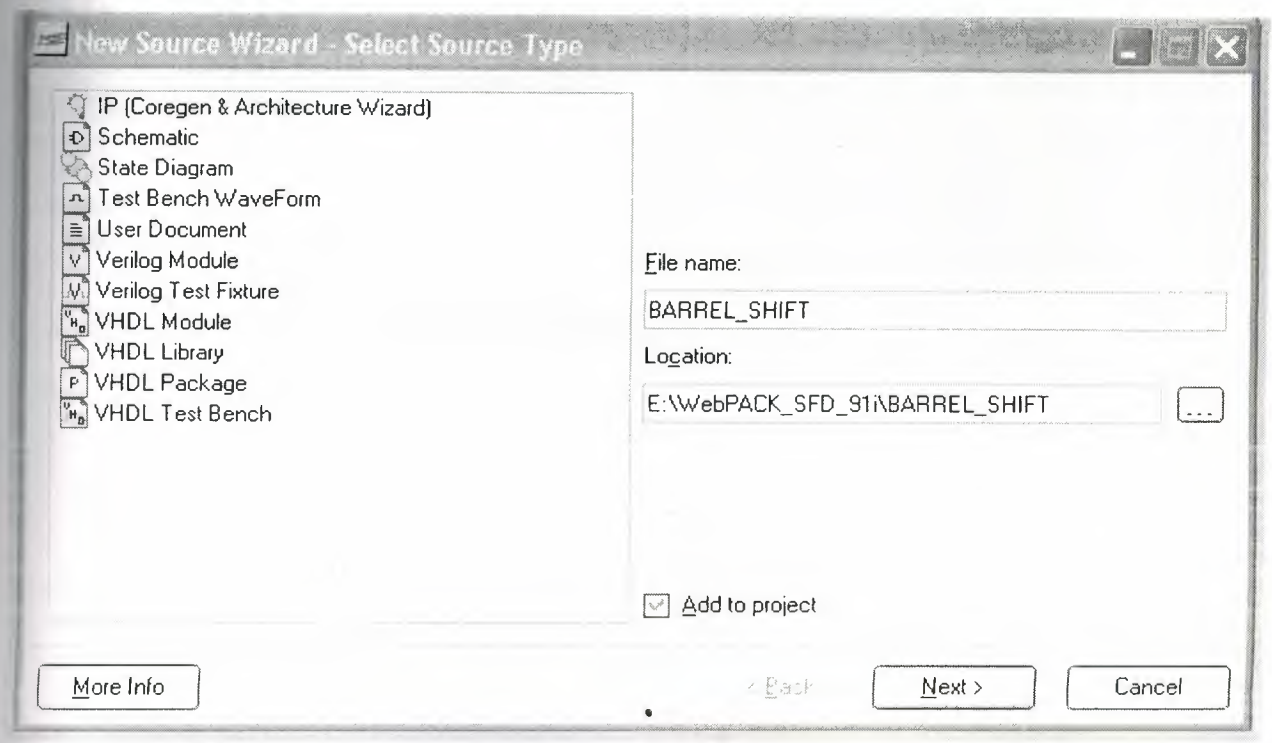

Figure.10 Selection module

In define Module,declare the ports for the counter design by filling in the port information as shown in Figure 11.

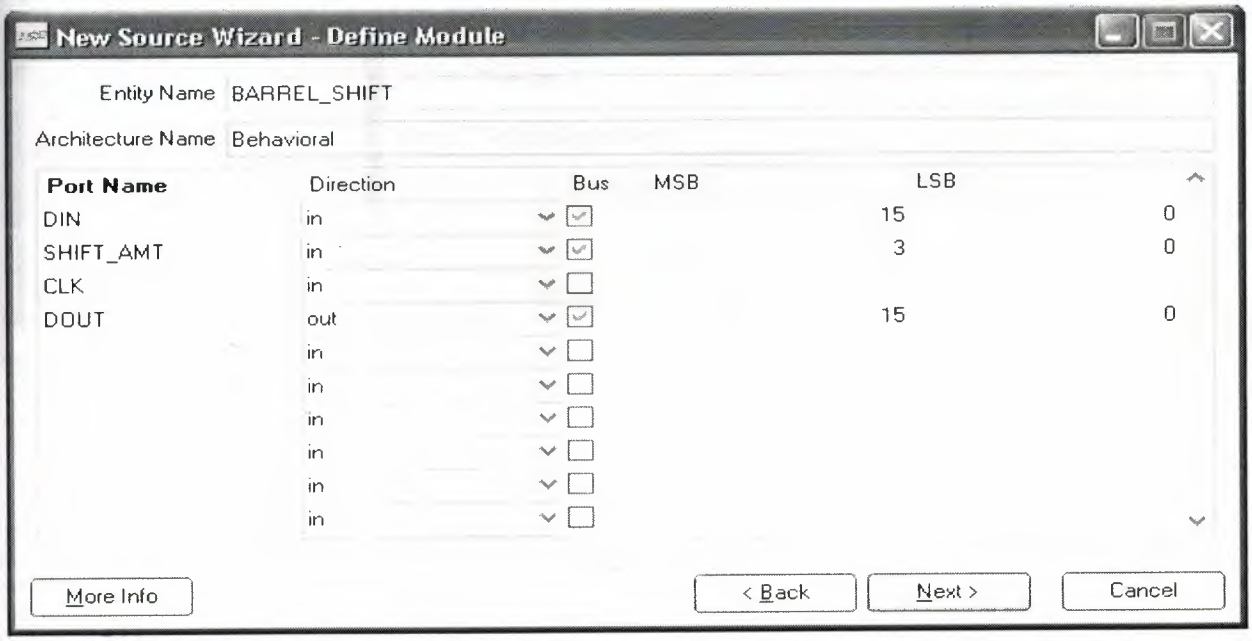

Figure.11 Defining inputs-outputs

and click next..

And our VHDL code is as in figure 12.

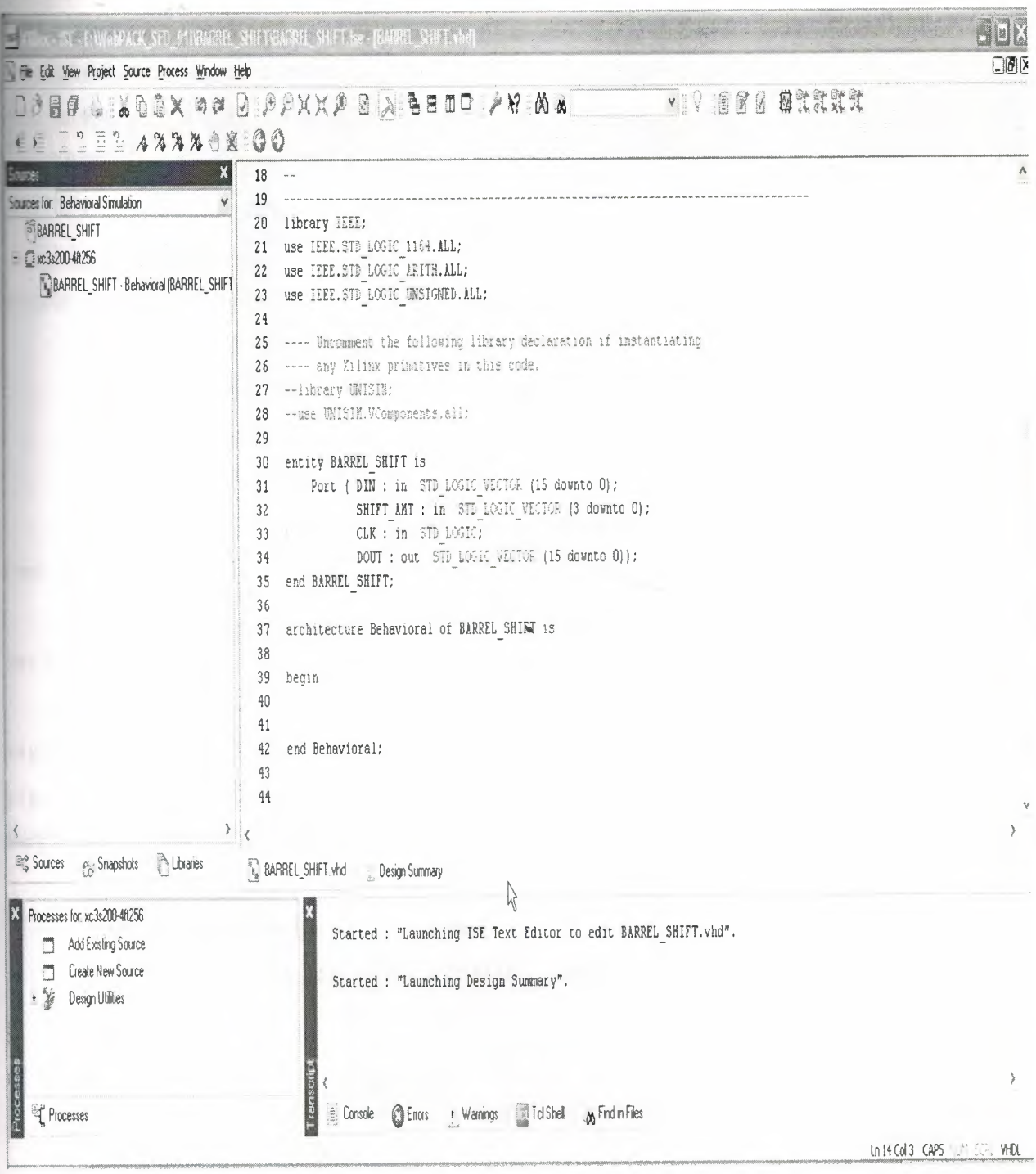

Figure.12 VHDL code

Then writing the VHDL code in source shown below:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
use IEEE.numeric_std.all;

entity BARREL_SHIFT is

      port( DIN : in std_logic_vector(15 downto 0);
          SHIFT_AMT   : in std_logic_vector(3 downto 0);
          CLK, ENB : in std_logic;
          DOUT : out std_logic_vector (15 downto 0));
end BARREL_SHIFT;

architecture RTL of BARREL_SHIFT is

signal DIN_BIT, DOUT_BIT : bit_vector (15 downto 0);
signal S_INT : integer;
begin

      DIN_BIT <= to_bitvector(DIN);
      S_INT <= CONV_INTEGER(SHIFT_AMT);
      process(CLK)
      begin
       if(CLK' event and CLK='1') then
          if(ENB = '0') then
                DOUT <= DIN;
             else
                   DOUT_BIT <= DIN_BIT ror S_INT;
                   DOUT <= to_stdlogicvector(DOUT_BIT);
             end if;
       end if;
```

```
    end process;
end RTL;
```

1.2.6.Synthesize

Design use the xilinx 'synthesize tool' to synthesize the code. The synthesize result is shown below.

TABLE OF CONTENTS

--> Parameter TMPDIR set to ./xst/projnav.tmp
CPU : 0.00 / 2.70 s | Elapsed : 0.00 / 2.00 s

--> Parameter xsthdpdir set to ./xst

```
CPU : 0.00 / 2.72 s | Elapsed : 0.00 / 2.00 s

--> Reading design: BARREL_SHIFT.prj


=========================================================
====================
*                Synthesis Options Summary              *
=========================================================
====================
---- Source Parameters
Input File Name                    : "BARREL_SHIFT.prj"
Input Format                       : mixed
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "BARREL_SHIFT"
Output Format                      : NGC
Target Device                      : xc3s200-4-ft256

---- Source Options
Top Module Name                    : BARREL_SHIFT
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
Safe Implementation                : No
FSM Style                          : lut
RAM Extraction                     : Yes
RAM Style                          : Auto
ROM Extraction                     : Yes
Mux Style                          : Auto
Decoder Extraction                 : YES
Priority Encoder Extraction        : YES
Shift Register Extraction          : YES
Logical Shifter Extraction         : YES
```

XOR Collapsing : YES
ROM Style : Auto
Mux Extraction : YES
Resource Sharing : YES
Asynchronous To Synchronous : NO
Multiplier Style : auto
Automatic Register Balancing : No

---- Target Options
Add IO Buffers : YES
Global Maximum Fanout : 500
Add Generic Clock Buffer(BUFG) : 8
Register Duplication : YES
Slice Packing : YES
Optimize Instantiated Primitives : NO
Use Clock Enable : Yes
Use Synchronous Set : Yes
Use Synchronous Reset : Yes
Pack IO Registers into IOBs : auto
Equivalent register Removal : YES

---- General Options
Optimization Goal : Speed
Optimization Effort : 1
Library Search Order : BARREL_SHIFT.lso
Keep Hierarchy : NO
RTL Output : Yes
Global Optimization : AllClockNets
Read Cores : YES
Write Timing Constraints : NO
Cross Clock Analysis : NO
Hierarchy Separator : /
Bus Delimiter : <>
Case Specifier : maintain

```
Slice Utilization Ratio          : 100
BRAM Utilization Ratio           : 100
Verilog 2001                     : YES
Auto BRAM Packing                : NO
Slice Utilization Ratio Delta    : 5

=========================================================
====================


=========================================================
====================
*                  HDL Compilation                      *
=========================================================
====================
Compiling vhdl file
"E:/WebPACK_SFD_91i/xilinx/myprojects/BARREL_SHIFT/BARREL_S
HIFT.vhd" in Library work.
Entity <barrel_shift> compiled.
Entity <barrel_shift> (Architecture <rtl>) compiled.

=========================================================
====================
*                Design Hierarchy Analysis              *
=========================================================
====================
Analyzing hierarchy for entity <BARREL_SHIFT> in library <work>
(architecture <rtl>).


=========================================================
====================
*                  HDL Analysis                         *
```

```
=========================================================
====================
Analyzing Entity <BARREL_SHIFT> in library <work> (Architecture
<rtl>).
Entity <BARREL_SHIFT> analyzed. Unit <BARREL_SHIFT> generated.


=========================================================
====================
*                    HDL Synthesis                      *
=========================================================
====================

Performing bidirectional port resolution...

Synthesizing Unit <BARREL_SHIFT>.
    Related source file is
"E:/WebPACK_SFD_91i/xilinx/myprojects/BARREL_SHIFT/BARREL_S
HIFT.vhd".
WARNING:Xst:646 - Signal <S_INT<31:4>> is assigned but never used.
    Found 16-bit register for signal <DOUT>.
    Found 16-bit register for signal <DOUT_BIT>.
    Summary:
        inferred  32 D-type flip-flop(s).
Unit <BARREL_SHIFT> synthesized.


=========================================================
====================
HDL Synthesis Report

Macro Statistics
# Registers                          : 2
 16-bit register                     : 2
```

```
=========================================================
====================

=========================================================
====================
*                  Advanced HDL Synthesis                  *
=========================================================
====================

Loading device for application Rf_Device from file '3s200.nph' in
environment E:\WebPACK_SFD_91i.

=========================================================
====================
Advanced HDL Synthesis Report

Macro Statistics
# Registers                          : 32
 Flip-Flops                         : 32

=========================================================
====================

=========================================================
====================
*                  Low Level Synthesis                     *
=========================================================
====================

Optimizing unit <BARREL_SHIFT> ...

Mapping all equations...
Building and optimizing final netlist ...
```

Found area constraint ratio of 100 (+ 5) on block BARREL_SHIFT, actual ratio is 2.

Final Macro Processing ...

==========
==========
Final Register Report

Macro Statistics
Registers : 32
Flip-Flops : 32

==========
==========

==========
==========
* Partition Report *
==========
==========

Partition Implementation Status

No Partitions were found in this design.

==========
==========
* Final Report *
==========
==========

```
Final Results
RTL Top Level Output File Name     : BARREL_SHIFT.ngr
Top Level Output File Name         : BARREL_SHIFT
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : NO

Design Statistics
# IOs                              : 38

Cell Usage :
# BELS                             : 80
#      LUT3                        : 80
# FlipFlops/Latches                : 32
#      FD                          : 16
#      FDE                         : 16
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 37
#      IBUF                        : 21
#      OBUF                        : 16
=========================================================
====================

Device utilization summary:
---------------------------

Selected Device : 3s200ft256-4

 Number of Slices:                46  out of   1920     2%
 Number of Slice Flip Flops:      32  out of   3840     0%
 Number of 4 input LUTs:          80  out of   3840     2%
 Number of IOs:                   38
 Number of bonded IOBs:           38  out of    173    21%
```

Number of GCLKs: 1 out of 8 12%

Partition Resource Summary:

No Partitions were found in this design.

==========

==========

TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
GENERATED AFTER PLACE-and-ROUTE.

Clock Information:

Clock Signal	Clock buffer(FF name)	Load
CLK	BUFGP	32

Asynchronous Control Signals Information:

No asynchronous control signals found in this design

Timing Summary:

```
---------------
Speed Grade: -4

  Minimum period: 2.343ns (Maximum Frequency: 426.803MHz)
  Minimum input arrival time before clock: 8.020ns
  Maximum output required time after clock: 7.165ns
  Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================
====================
Timing constraint: Default period analysis for Clock 'CLK'
  Clock period: 2.343ns (frequency: 426.803MHz)
  Total number of paths / destination ports: 16 / 16
-------------------------------------------------------------------------
Delay:               2.343ns (Levels of Logic = 1)
  Source:            DOUT_BIT_0 (FF)
  Destination:       DOUT_0 (FF)
  Source Clock:      CLK rising
  Destination Clock: CLK rising

  Data Path: DOUT_BIT_0 to DOUT_0
                              Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDE:C->Q             1    0.720   0.869  DOUT_BIT_0 (DOUT_BIT_0)
     LUT3:I2->O           1    0.551   0.000  DOUT_mux0001<0>1
(DOUT_mux0001<0>)
     FD:D                      0.203          DOUT_0
    ----------------------------------------
    Total                      2.343ns (1.474ns logic, 0.869ns route)
```

```
                    (62.9% logic, 37.1% route)

=========================================================
==================
Timing constraint: Default OFFSET IN BEFORE for Clock 'CLK'
  Total number of paths / destination ports: 544 / 48
-------------------------------------------------------------------
Offset:              8.020ns (Levels of Logic = 5)
  Source:            SHIFT_AMT<1> (PAD)
  Destination:       DOUT_BIT_0 (FF)
  Destination Clock: CLK rising

  Data Path: SHIFT_AMT<1> to DOUT_BIT_0
                        Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            16   0.821   1.576  SHIFT_AMT_1_IBUF
(SHIFT_AMT_1_IBUF)
     LUT3:I0->O            2   0.551   1.072  DOUT_BIT_mux0001<10>81
(DOUT_BIT_mux0001<10>_bdd12)
     LUT3:I1->O            2   0.551   1.072  DOUT_BIT_mux0001<13>61
(DOUT_BIT_mux0001<13>_bdd8)
     LUT3:I1->O            2   0.551   1.072  DOUT_BIT_mux0001<13>51
(DOUT_BIT_mux0001<13>_bdd1)
     LUT3:I1->O            1   0.551   0.000  DOUT_BIT_mux0001<5>11
(DOUT_BIT_mux0001<5>)
     FDE:D                     0.203          DOUT_BIT_5
    ----------------------------------------
    Total                      8.020ns (3.228ns logic, 4.792ns route)
                                       (40.2% logic, 59.8% route)

=========================================================
==================
Timing constraint: Default OFFSET OUT AFTER for Clock 'CLK'
```

```
 Total number of paths / destination ports: 16 / 16
-------------------------------------------------------------------------
Offset:          7.165ns (Levels of Logic = 1)
  Source:        DOUT_15 (FF)
  Destination:   DOUT<15> (PAD)
  Source Clock:  CLK rising

  Data Path: DOUT_15 to DOUT<15>
                        Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FD:C->Q              1   0.720   0.801  DOUT_15 (DOUT_15)
     OBUF:I->O                5.644          DOUT_15_OBUF (DOUT<15>)
    ----------------------------------------
    Total                    7.165ns (6.364ns logic, 0.801ns route)
                             (88.8% logic, 11.2% route)

=========================================================
====================
CPU : 15.67 / 18.84 s | Elapsed : 16.00 / 19.00 s

-->

Total memory usage is 141532 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    1 (   0 filtered)
Number of infos    :    0 (   0 filtered)
```

Xilinx sythesize tool created the following design. Top level block diagram.

1.2.7. Writing Test Bench

In the test bench generated 100 MHz. I wrote in the data in to the BARREL SHIFTER and read it back to verify data can be written and read correctly.

Create a VHDL TESTBENCH file for the project as follow

We select *project* and *new new source* as figure 13.

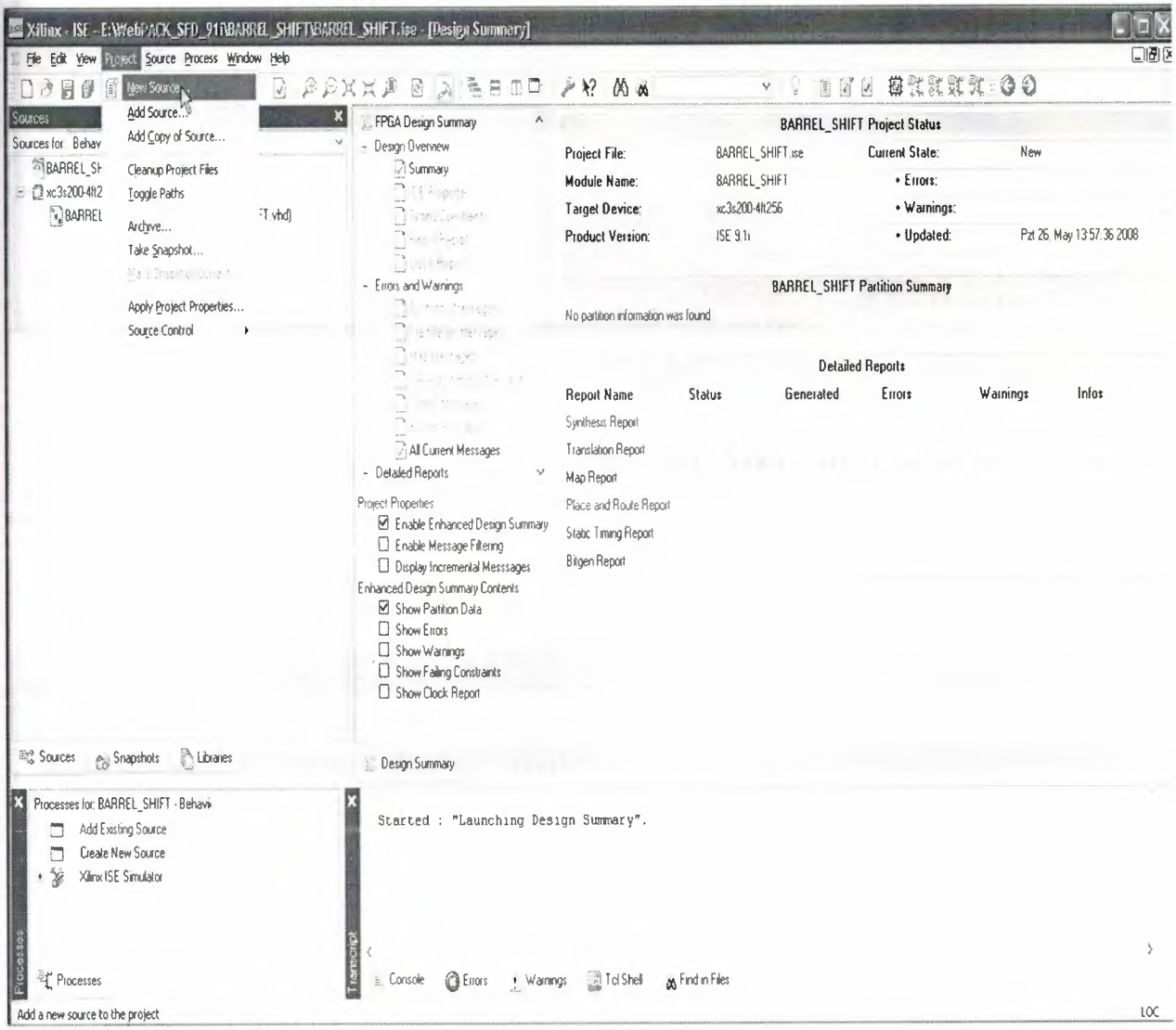

Figure.13 Entering new source as test bench

And we choose *VHDL Test Bench* as a source type.Then type *BARREL_SHIFT_TB* as a file name then click next as shown in figure 15.

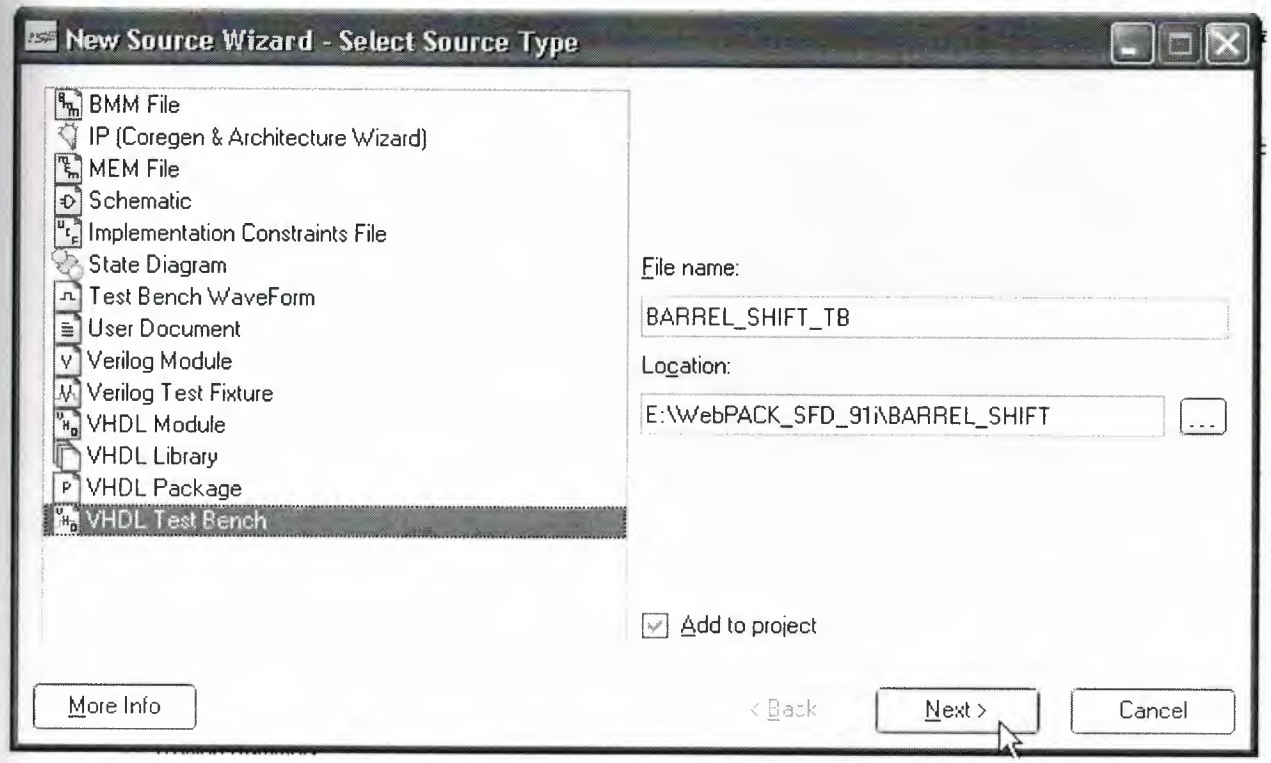

Figure.15 Selecting module

After the selecting Source type we declare Associate Source;so we click next as shown in figure 16.

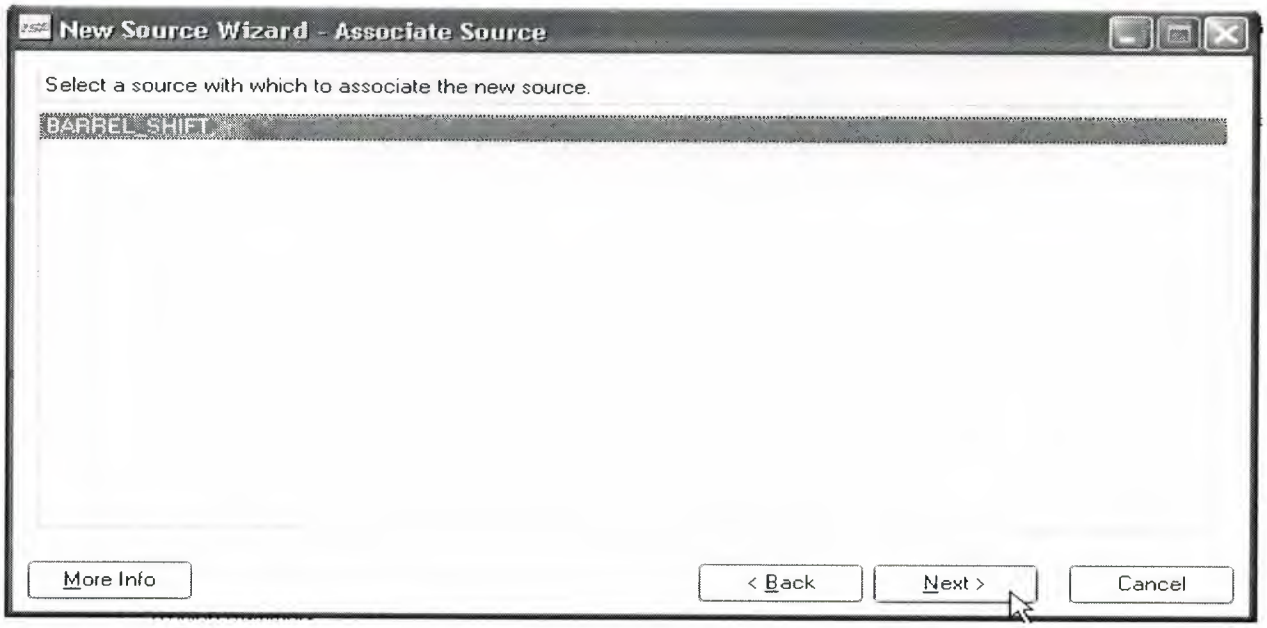

Figure.16 Association of source

And we click Finish for ending the creating TEST BENCH.

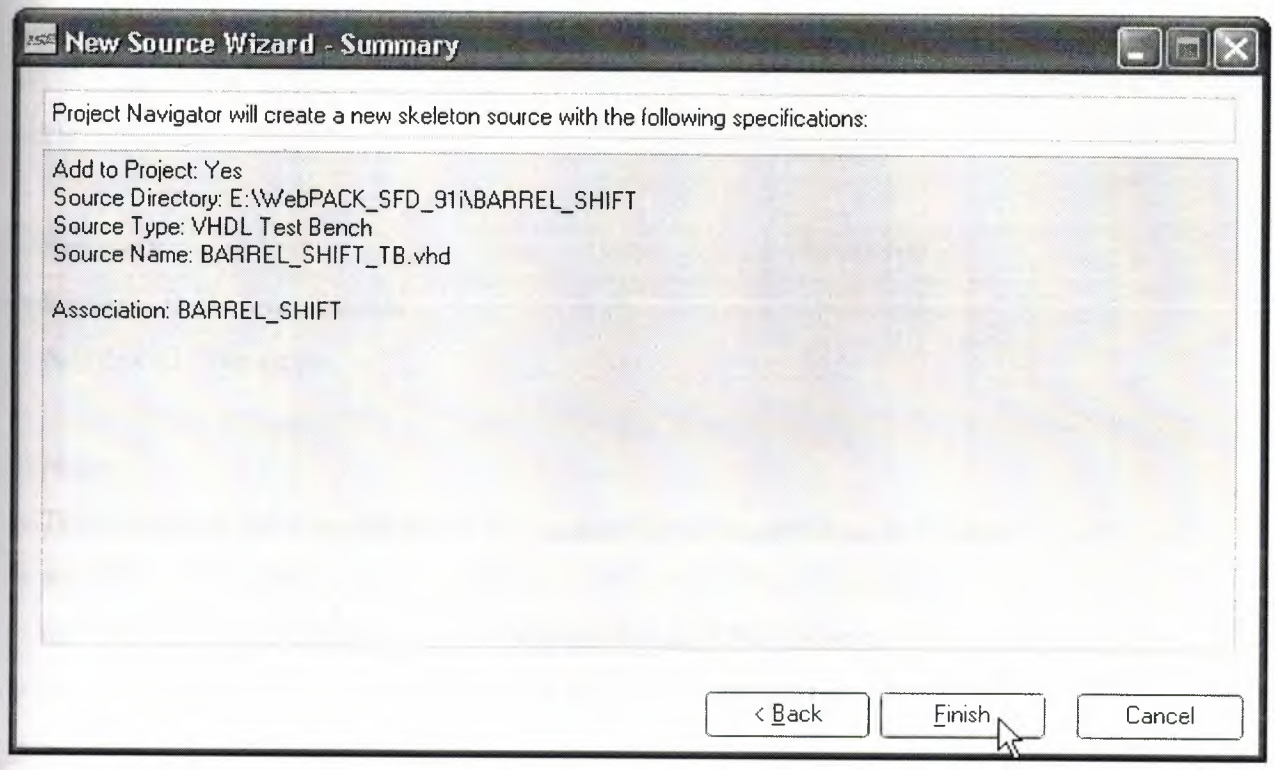

Figure.17 Design summary

And design has a TEST BENCH but program gives us the base codes as following;

```
--------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:   14:01:16 05/26/2008
-- Design Name:   BARREL_SHIFT
-- Module Name:
E:/WebPACK_SFD_91i/BARREL_SHIFT/BARREL_SHIFT_TB.vhd
-- Project Name:  BARREL_SHIFT
-- Target Device:
-- Tool versions:
```

```
-- Description:
--
-- VHDL Test Bench Created by ISE for module: BARREL_SHIFT
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY BARREL_SHIFT_TB_vhd IS
END BARREL_SHIFT_TB_vhd;

ARCHITECTURE behavior OF BARREL_SHIFT_TB_vhd IS

        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT BARREL_SHIFT
        PORT(
                DIN : IN std_logic_vector(15 downto 0);
                SHIFT_AMT : IN std_logic_vector(3 downto 0);
                CLK : IN std_logic;
```

```
        DOUT : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;

    --Inputs
    SIGNAL CLK :  std_logic := '0';
    SIGNAL DIN :  std_logic_vector(15 downto 0) := (others=>'0');
    SIGNAL SHIFT_AMT :  std_logic_vector(3 downto 0) := (others=>'0');

    --Outputs
    SIGNAL DOUT :  std_logic_vector(15 downto 0);

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: BARREL_SHIFT PORT MAP(
        DIN => DIN,
        SHIFT_AMT => SHIFT_AMT,
        CLK => CLK,
        DOUT => DOUT
    );

    tb : PROCESS
    BEGIN

        -- Wait 100 ns for global reset to finish
        wait for 100 ns;

        -- Place stimulus here

        wait; -- will wait forever
    END PROCESS;

END;
```

Now, I write my program in it as following ;

```
--------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:   14:34:51 04/16/2008
-- Design Name:   BARREL_SHIFT
-- Module Name:
D:/WebPACK_SFD_91i/xilinx/myprojects/BARREL_SHIFT/BARREL_SHIFT_TB.vh
d
-- Project Name:  BARREL_SHIFT
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: BARREL_SHIFT
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY BARREL_SHIFT_TB_vhd IS
END BARREL_SHIFT_TB_vhd;

ARCHITECTURE behavior OF BARREL_SHIFT_TB_vhd IS

        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT BARREL_SHIFT
        PORT(
                DIN : IN std_logic_vector(15 downto 0);
                SHIFT_AMT : IN std_logic_vector(3 downto 0);
                CLK : IN std_logic;
                ENB : IN std_logic;
                DOUT : OUT std_logic_vector(15 downto 0)
                );
        END COMPONENT;

        --Inputs
        SIGNAL CLK :  std_logic := '0';
        SIGNAL ENB :  std_logic := '0';
        SIGNAL DIN :  std_logic_vector(15 downto 0) := (others=>'0');
        SIGNAL SHIFT_AMT :  std_logic_vector(3 downto 0) := (others=>'0');

        --Outputs
        SIGNAL DOUT :  std_logic_vector(15 downto 0);

BEGIN
```

```
-- Instantiate the Unit Under Test (UUT)
uut: BARREL_SHIFT PORT MAP(
        DIN => DIN,
        SHIFT_AMT => SHIFT_AMT,
        CLK => CLK,
        ENB => ENB,
        DOUT => DOUT
);

CLK_PR : process
begin
CLK<='0';
wait for 10ns;
CLK<='1';
wait for 10 ns;
end process;


tb : PROCESS
BEGIN

        -- Wait 100 ns for global reset to finish
        wait for 100 ns;

        -- Test that data input (DIN) 0000000000000001 is rotated right by the
        -- shift amount (SHIFT_AMT) 1.
        ENB<='1';
        DIN<="0000000000000001";
        SHIFT_AMT<="0001";

  wait for 100 ns;
        -- Test that data input (DIN) 0000000000000011 is rotated right by the
        -- shift amount (SHIFT_AMT) 3.
        ENB<='1';
```

```
        DIN<="0000000000000011";
        SHIFT_AMT<="0011";


        wait for 100 ns;
        -- Test that data input ,0000000000000011, (DIN) is loaded correctly to
the output
        ENB<='0';
        DIN<="0000000000000011";

            wait for 100 ns;
        -- Test that data input "1111000000000011" (DIN) is loaded  correctly to
the output.
        ENB<='0';
        DIN<="1111000000000011";

        -- Place stimulus here
        wait; -- will wait forever
    END PROCESS;

END;
```

Then click Xilinx ISE Simulator then check syntax ,it shown as below:

Figure.18 Checking syntax

```
Running Fuse ...
Compiling vhdl file
"E:/WebPACK_SFD_91i/xilinx/myprojects/BARREL_SHIFT/BARREL_S
HIFT.vhd" in Library work.
Entity < BARREL_SHIFT > compiled.
Entity < BARREL_SHIFT > (Architecture <rtl>) compiled.
Compiling vhdl file
"E:/WebPACK_SFD_91i/xilinx/myprojects/BARREL_SHIFT/BARREL_S
HIFT_TB.vhd" in Library work.
Entity <BARREL_SHIFT_TB_vhd> compiled.
Entity < BARREL_SHIFT _TB_vhd> (Architecture <behavior>)
compiled.
Parsing "BARREL_SHIFT_TB_vhd_beh.prj": 0.34
Codegen   work/BARREL_SHIFT: 0.03
Codegen   work/BARREL_SHIFT/RTL: 2.05
Codegen   work/BARREL_SHIFT_TB_vhd: 0.00
Codegen   work/BARREL_SHIFT_TB_vhd/behavior: 0.69
Building BARREL_SHIFT_TB_vhd_isim_beh.exe
Running ISim simulation engine ...
This is a Lite version of ISE Simulator.

Simulator is doing circuit initialization process.
Finished circuit initialization process.
Check syntax succesfully.
```

After the checking syntax successfully,it is time to simulate our design as shown;

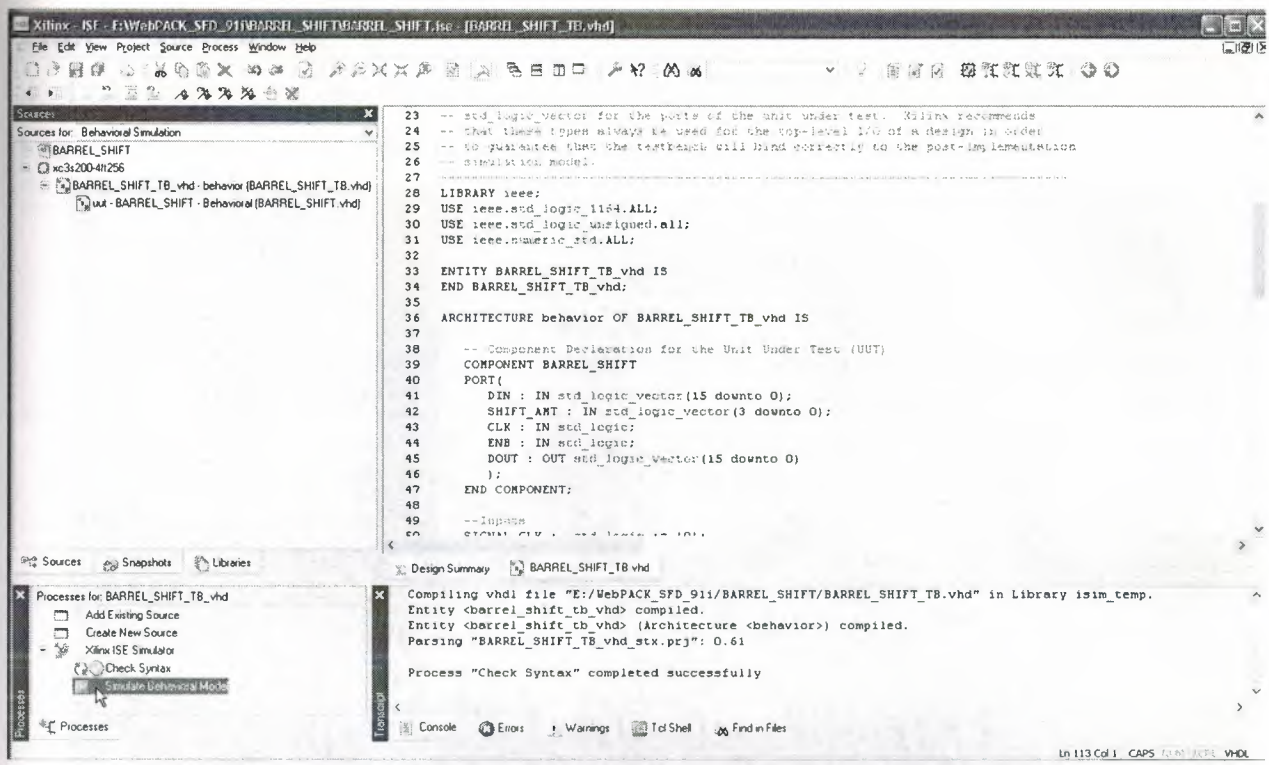

Figure.19 Simulate behavioral

1.2.8. Simulating

I have created a test bench to the BARREL SHIFTER.In this test bench I wrote in to the BARREL_SHIFT and read the data I wrote.

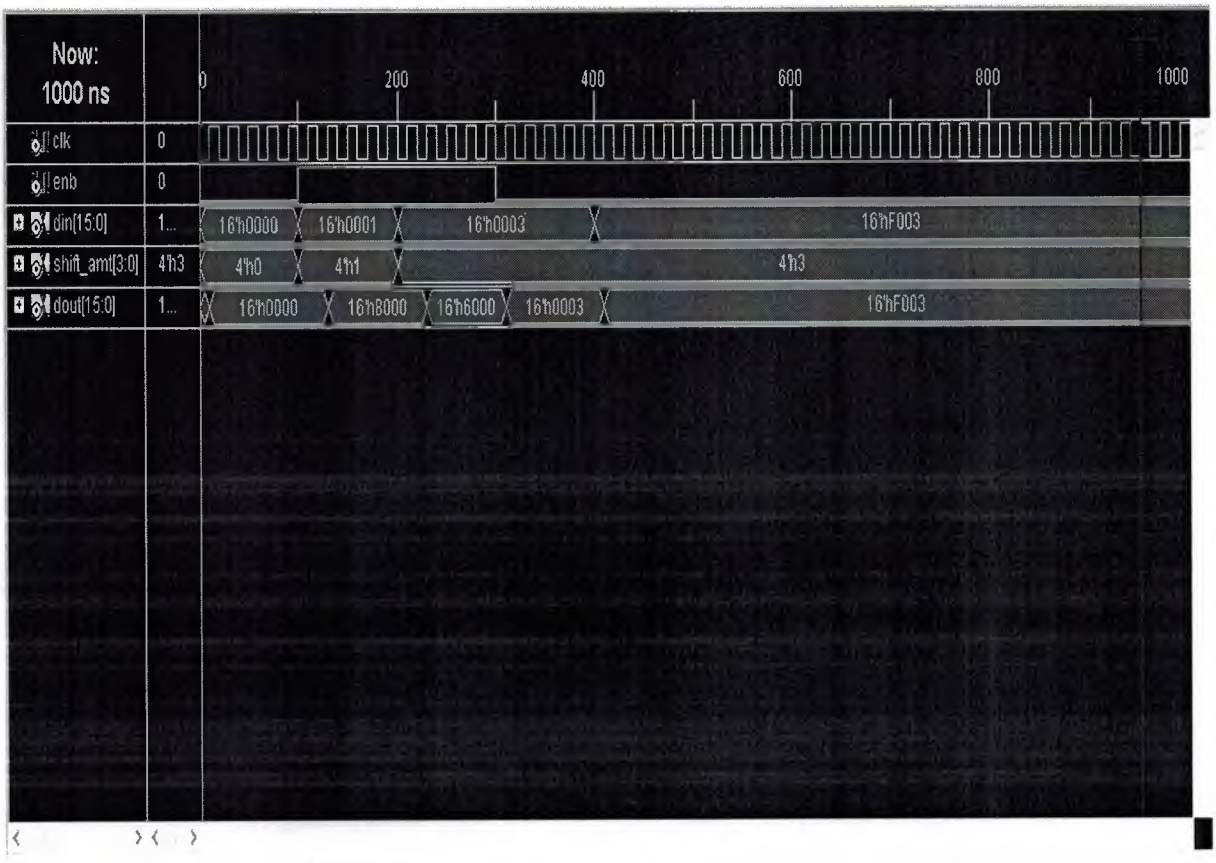

Figure.20 Design simulation

CHAPTER TWO: ABOUT XILINX

2.1.What Does Xilinx Mean?

How, many ask, did Xilinx (pronounced "Zylinks") get its unusual name? In 1984, when Xilinx was just forming, the new company tried to register several "sensible" names, but they were all taken. This became an expensive proposition and the founders, (being very frugal), decided to create an unusual name that wasn't taken. Thus, two of the founders came up with "Xilinx."

2.2. What Does the Xilinx Name Represent?

Xilinx Fellow Bill Carter, who was at Xilinx from the start, explains. "The 'X's' at each end represent programmable logic blocks. The "linx" represents programmable links that connect the logic blocks together, a key innovation embodied in FPGAs."

While Xilinx doesn't follow all the branding and phonetically-correct rules for naming a company, a Xilinx by any other name would not be as sweet.

2.3. History Of Xilinx

2.3.1. How Xilinx Began

Two brilliant engineers and a marketing guru working in Silicon Valley in 1984 had a dream. Bernie Vonderschmitt, Ross Freeman, and Jim Barnett dreamed of starting a different kind of company.

They wanted to create a company that would develop and launch state-of-the-art technology in an entirely new field. And they wanted to lead it in such a way that the

people who worked there loved their jobs, enjoyed working together, and were fascinated with their work.

The technology that propelled Xilinx into being was considered an off-the-wall concept in 1984. Invented by Xilinx co-founder Ross Freeman, the new semiconductor, now known as the field programmable gate array, was a completely new form of programmable logic.

These chips could be personalized by customers to perform a variety of functions by programming them with the help of software. "The concept," says Xilinx Fellow Bill Carter, who was the eighth employee to be hired in the new company in 1984, "required lots of transistors and, at that time, transistors were considered extremely precious. People thought that Ross's idea was pretty far out."

Ross postulated that transistors, because of Moore's Law (the doubling of transistor density every 18 months) would be getting less expensive and, therefore, less precious every year. In the years to come, a multi-billion dollar market for field programmable gate arrays (FPGAs) emerged, creating the foundation for the successful enterprise that Xilinx is today. Sadly, Ross Freeman passed away in 1989. The technology he invented is thriving and continues to delight more and more customers in an ever-widening breadth of industries.

Effective Partnerships:

Bernie Vonderschmitt, an engineer and an MBA graduate, came up with a powerful business model for the young company. When he was General Manager of the Solid State Division of RCA, he became convinced, working at the time with three in-house foundries making semiconductors, that semiconductor factories (or fabs) were expensive and burdensome. "If I ever start a semiconductor company, it will be fabless," he vowed. "We'll find partners who can do our manufacturing for us."

And that is exactly what Xilinx did in 1984. Since then, the idea has become so compelling and popular that today there are approximately 700 fabless semiconductor companies around the world.

Inspired Employees:

However, the three founders wanted to not only revolutionize technology but the way companies are managed as well. Ross Freeman put it best. He hoped to start a company that had solid, ethical values, invited employee loyalty, made a good and useful product, helped make employees feel like owners, and encouraged people to enjoy their work.

The co-founders called this set of values and people objectives their "philosophy" and looked for employees who felt comfortable in this environment. And their theory - which has proven correct - was that if you created this kind of community atmosphere for clever and inventive people, they would stay, keeping their innovation and expertise in the company.

These original values regarding the treatment of employees and the way they interact with each other provided the basis for how Xilinx operates today. They help make Xilinx a great place to work.

And the technology that the three men introduced to the world is more popular than ever. It has become pervasive and mainstream, thanks to the technology and cost benefits that have come about because of Moore's Law.

The dream that Bernie, Ross, and Jim talked about in 1984 is a reality today, proving that dreams do come true.

2.3.2. Learn about our technology, and how and why we pursue it

There are three types of electronic devices: memory, processors, and logic. Memory devices store random information (contents of a spreadsheet or database); processors execute software instructions to perform a wide variety of tasks (running a data processing program or video game); and logic provides specific functions (communications between devices, and every other function a system must perform). There are two categories of logic devices: fixed or custom, and programmable or changeable. We are in the programmable logic business.

2.3.3. Programmable Logic is Our Business

Xilinx leads the Programmable Logic Device (PLD) market - one of the fastest growing segments of the semiconductor industry. This market features a revolutionary technology called the field programmable gate array (FPGA) that our company pioneered in 1984.

Xilinx is the world's leading supplier of programmable logic solutions. We supply customers with "off-the-shelf" logic devices that customers can program to perform specific functions using the development tools we provide.

This programmability provides a revolutionary alternative to fixed or custom logic devices that typically require many months to design, test, and manufacture. Xilinx customers enjoy the benefit of faster time-to-market and increased product design flexibility as a result.

2.3.4. Uses for Programmable Logic

Our company's business is drawn from a variety of industry segments. In recent years, a large portion of our revenues came from the communications marketplace. However, we have become increasingly more diversified to include the consumer, industrial, and automotive sectors.

You can find Xilinx chips in a wide variety of digital electronic applications ranging from wireless base stations to HDTV to portable handsets.

2.3.5. Multiple Product LInes with Superlative Software Support

Our extensive product line includes silicon solutions like the Virtex™ series FPGAs (high performance FPGAs for networking, communications, and video/imaging applications); Spartan™ FPGAs (ideal for high volume applications); and CoolRunner™ CPLD families (Complex Programmable Logic Devices that offer ultra low cost and low power). We also offer a powerful suite of high performance software design tools.

2.3.6. High-profile Worldwide Customer and Partner Base

Xilinx has over 21,000 customers around the globe, including Alcatel, Cisco Systems, EMC, Ericsson, Fujitsu, Hewlett-Packard, IBM, Lucent Technologies, and Motorola. Most of our sales are handled by outside partners: both large distributors and independent sales representatives.

The company has a very flexible business model that has contributed to our success as an employer and a competitor. We are a "fabless" supplier and do not manufacture our logic devices. Instead, we have formed close strategic alliances with chip manufacturers like UMC and Toshiba. This strategy, along with outsourcing most of sales, allows us to focus more of our energies on R&D, marketing and technical support. The resulting flexibility gives us the ability to rearrange business priorities quickly as we respond to the cyclical nature of the semiconductor market. It also has made it easier for the company to avoid layoffs in periods of downturn.

2.3.7. Our Vision for the Future

What essentially defines Xilinx is vision. Our long-term business goal is to put a PLD in every piece of electronic equipment within the next 10 years. Our long-term management goal is to create a company that sets the standard for managing high technology companies. While these are ambitious undertakings, at Xilinx, visions have a way of turning into reality.

2.4. Success Of Xilinx

2.4.1. The synergy of technology, partnership, and leadership

While the rest of the industry continues the practices of layoffs and shaking-off excessive inventory, Xilinx is busy innovating, collaborating, and introducing new products to market. Unlike many of our counterparts, Xilinx views downturns as an opportunity to focus on research and development, streamline operations, and deliver new products that change the FPGA landscape.

For the past few years, Xilinx has asserted a considerable market leadership position. We secured over 50% of the PLD market share: larger than all other public PLD companies combined. By creatively avoiding layoffs and empowering employees, we rose to become the fourth best company to work for in America (Forbes magazine).

2.4.2. Our Partners

Through the power of innovation and partnerships, Xilinx takes the FPGA-based value chain to a new level. By teaming with technology leaders in silicon fabrication, design automation, system level tools, IP, and design services, we deliver a complete value chain and strengthen our position as a strategic partner for our customers. Delivering this complete value chain enables the fastest innovation while reducing total development and system costs for our customers. It also reduces time to market and increases time in market for our customer's products.

In March 2002, through partnering with industry leaders IBM, WindRiver Systems, and Conexant, Xilinx delivered the Virtex™-II Pro programmable system solution. The solution was the first of its kind and is the most flexible tool ever invented for a designer. The Virtex-II Pro FPGA includes programmable logic fabric with high-speed embedded PowerPC processors and integrated 3.125 gigabit RocketIO™ serial transceivers supported by leading design tools. Recent additions to the family and lower price points have now made the Virtex-II Pro solution the de-facto standard for all programmable logic users.

The Virtex-II Pro solution responds to the issues facing design teams and their corporations. By delivering both high-performance processing and high bandwidth connectivity on a single device, many design challenges associated with integration, high-speed interfacing, high performance processing, and new design methodologies are effectively solved. The rapid rate of change in technology and standards demands a solution that is completely flexible and reduces inventory risks and NRE costs - the Virtex-II Pro solution delivers.

2.4.3. Our Technology

Xilinx is a company built on delivering maximum customer value and ongoing innovation throughout all of our product lines. Xilinx recently revamped all of its products from the new Spartan™-IIE cost-optimized FPGA solution to the CoolRunner™-II RealDigital CPLD solution, the Virtex-II Pro platform for programmable systems, and the Virtex-II EasyPath solution for cost management. We also introduced the world's fastest and most productive software tool suite with our ISE 4.2i software release, numerous intellectual property cores, and the technical training necessary to decrease time-to-knowledge for the rapid assimilation of this new technology. We continue to focus on raising the bar by adding more value in every category of the value chain.

Through the years, Xilinx has evolved into a solutions company rather than remaining just a chip company. We can only be better tomorrow than we are today by working closely with our customers and anticipating their needs. Xilinx's job is to continue to expand our capabilities and our partnerships, so we can continue to be a strategic partner for our client companies.

2.4.4. Our Employees

Xilinx is an innovation engine and our employees are the keys to our innovation. Such innovation requires personnel policies that allow employees to make their own decisions and take risks. Our company values and corporate culture promote teamwork and very open communication. We know that keeping employees satisfied leads directly to innovation, customer satisfaction, and ultimately, increased profits. Our employees are inspired and know they make a real difference.

This unique work environment has resulted in breakthrough technology, marketing and community achievements. For example, Xilinx continues to support local schools through our Stock for Students program and made a $1 million donation to the American Red Cross. Also, Xilinx was the first semiconductor company to simulcast training in North America and Europe through industry events like Programmable World 2002.

With a combination of innovative products, world-class partners, inspired employees and the recognition of the balance between business and community, our clients have taken Xilinx solutions, management, and employees to heart. This is a reminder that good people ultimately do come in first when they are inspired and empowered to be leaders

2.5. Values Of Xilinx

2.5.1. How we work with one another and our partners

2.5.1.1.What do values mean to Xilinx?

Our values have helped set the character of our company. They are more than a set of lofty ideals put down on paper and left to yellow in conference rooms. Values are very much alive and well in Xilinx. We don't just talk about them, we try and live them every day.

Our values also provide the backdrop for the dialogue we have with colleagues. They help us make business decisions. They provide the framework for interacting with each other. What's especially impressive about our values is that they are accepted and acceptable around the world. The practices may be different in different places, but the values are relevant everywhere.

While a whirlwind of business change constantly surrounds us - and we accept change as the reality of today's high-tech industry - it's important to know that the values we depend upon are constant and unchanging. They are, in a very real sense, our permanent anchors.

2.5.1.2.How did we clearly define our values?

The set of values we believe in started with our company's founders. The three men who followed their dream of starting a new enterprise were just as concerned about the work environment as they were about the new, innovative technology they were pioneering. Respect for the dignity of the individual was the cornerstone of the philosophy upon which Xilinx was founded.

In 1996, our company started a grassroots process to articulate our values. We looked very carefully at our business and made certain the values were connected with what would move us forward and foster our growth in the marketplace. Most importantly, we wanted to capture in words what we liked so much about working at Xilinx and the ideals our founders had set in motion.

The result was a description of the the eight Xilinx CREATIVE values.

Customer Focused
Respect
Excellence
Accountability
Teamwork
Integrity
Very Open Communication
Enjoying Our Work

It was the creativity of the founders and their innovative ideas that launched a new category of products for customers around the world. And it is the creativity of our products and patents that has propelled us to market leadership.

How do we keep our values visible and viable in the company?

At Xilinx, we believe that making decisions based on our values translates directly to the bottom line, helps us be more successful in our business, and brings us closer to realizing our company vision of setting a new standard for managing a high-tech company. Customers are eager to do business with a company whose values are as excellent as their products.

Another way the values are kept alive is through a variety of appreciation programs. The most popular one is the Values Medallion award. Employees nominate someone who has exhibited a teamwork value, and, each quarter, several winners are randomly selected from the nominees. These individuals are awarded 10 shares of stock and receive public recognition from Wim.

How do we "enforce" the values? We don't. We leave it up to each individual to act in accordance with them. The values are there to direct our actions, to guide us professionally and personally, and to inspire us in the worst and the best of times.

2.6. Spartan Series

Lowest Total Cost. Period.

- Up to 50% lower system cost than competing FPGAs
- Industry's largest selection of device/package options
- Industry's most comprehensive IP library
- Leading embedded and DSP solutions
- Efficient, cost-effective board designs
- Allows use of fewer standard components
- Increased system reliability by eliminating external components

Multiple Domain-Optimized Platforms in the 90nm Spartan-3 Generation

Digital Signal Processing

Spartan-3A DSP : FPGA – DSP optimized for applications where integrated DSP MACs and expanded memory are required.Ideal for designs requiring low cost FPGAs for signal processing applications such as military radio, surveillance cameras, medical imaging, etc.

Spartan-3AN : FPGA– Non-volatile for applications where non-volatile, system integration, security, large user flash are required.Ideal for space-critical or secure applications as well as low cost embedded controllers.

Spartan-3A : FPGA – I/O optimized for applications where I/O count and capabilities matter more than logic density. Ideal for bridging, differential signaling and memory interfacing applications, requiring wide or multiple interfaces and modest processing.

Spartan-3E : FPGA – Logic optimized for applications where logic densities matter more than I/O count.Ideal for logic integration, DSP co-processing and embedded control, requiring significant processing and narrow or few interfaces.

Spartan-3 : FPGA – For highest density and pin-count applications for applications where both high logic density and high I/O count are important.Ideal for highly-integrated data-processing applications.

2.6.1. Overwiev

Multiple Domain-Optimized Platforms

Spartan®-3 generation FPGAs offer multiple platforms ranging from extremely low cost packaging to high performance DSP solutions while maintaining the lowest total cost possible.

Several platforms are available, each suited to your specific application need:

DSP
Non-volatile
I/O optimized
Logic optimized
Highest density and pin-count

Spartan-3 Generation FPGAS offers five platforms for your specific application needs

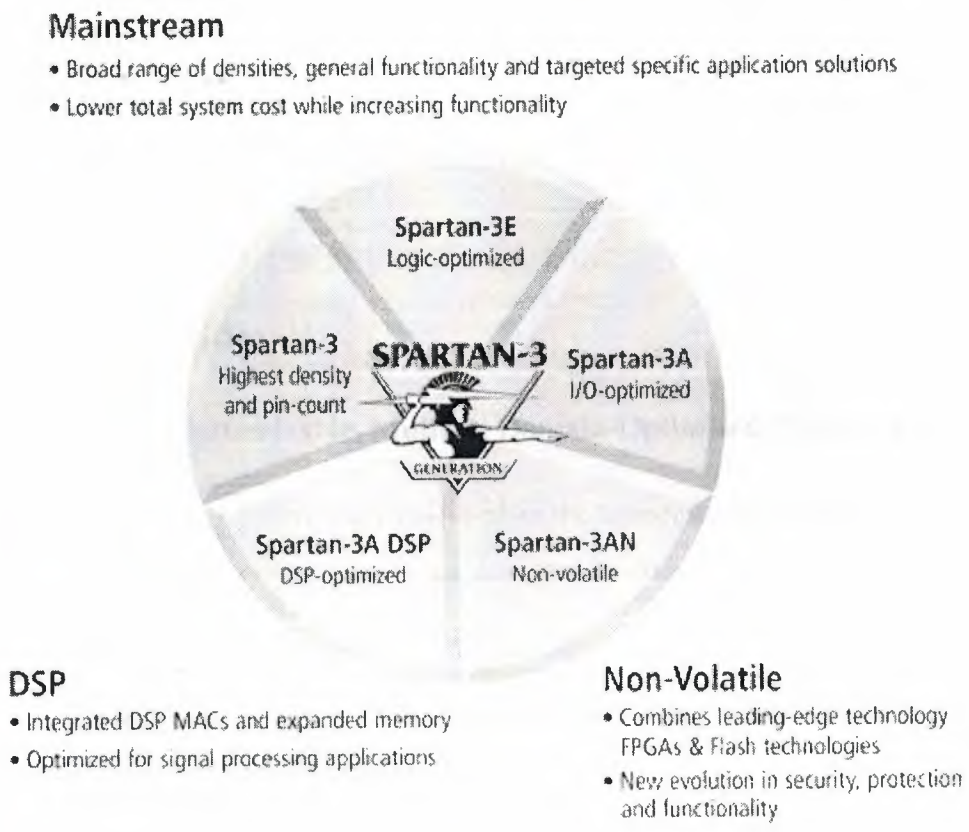

Figure.21 Generation of Spartan 3

Spartan-3 Generation FPGA platform for my design

Spartan Platform*	Gates	Integrated Flash	I/Os	Logic Cells	Block RAM	Embedded Multipliers	DCMs	Voltage
Spartan-3A DSP	3.4M	–	519	53, 712	2268 Kb	126 18x18†	8	3.3V - 1.2V**
Spartan-3AN	1.4M	16Mb	502	25,344	576 Kb	32 18x18	8	3.3V - 1.2V**
Spartan-3A	1.4M	–	502	25,344	576 Kb	32 18x18	8	3.3V - 1.2V**
Spartan-3E	1.6M	–	376	33,192	648 Kb	36 18x18	8	3.3V - 1.2V**
Spartan-3	5M	–	633	74,880	1872 Kb	104 18x18	4	3.3V - 1.2V**

* Maximum values listed for each Spartan platform.

** Spartan-3/3A/3E/3AN/3A DSP platforms offer multi-standard, multi-voltage SelectIO™ interface pins.

Integrated in the 126 DSP48A slices (Advanced Multiply Accumulate Element).

2.6.2. Capabilities

Easily Customizable, Multiple Domain-Optimized Platforms

Spartan®-3 generation FPGAs offer the broadest selection of platforms allowing the lowest possible system cost to the designer by choosing the perfect FPGA for the application.

In addition, unique features and capabilities are available to provide the ultimate low cost, high volume system designs:

- Dual power management
- Multiple levels of security
- Integrated Flash memory
- XtremeDSP DSP 48A Slice
- Embedded Processing
- Four level memory architecture
- Leading connectivity platform
- Configurable logic blocks
- Precise clock management resources
- Comprehensive configuration capabilities

Dual power management

The integrated power management in Spartan-3 generation FPGAs can reduce power consumption by up to 99%. A single pin activated, hardware feature is built-in unlike other external implementations which requires additional components.

Suspend mode

- Over 40% static power reduction
- All states saved in memory
- Scale down voltage (VCCAUX) and shut off non-essential functions (e.g., FPGA inputs, interconnects)
- System synchronization for fast wake-up

Hibernate mode

- Up to 99% static power reduction
- Shut off all power
- Wake up time
- Ultimate battery life extension

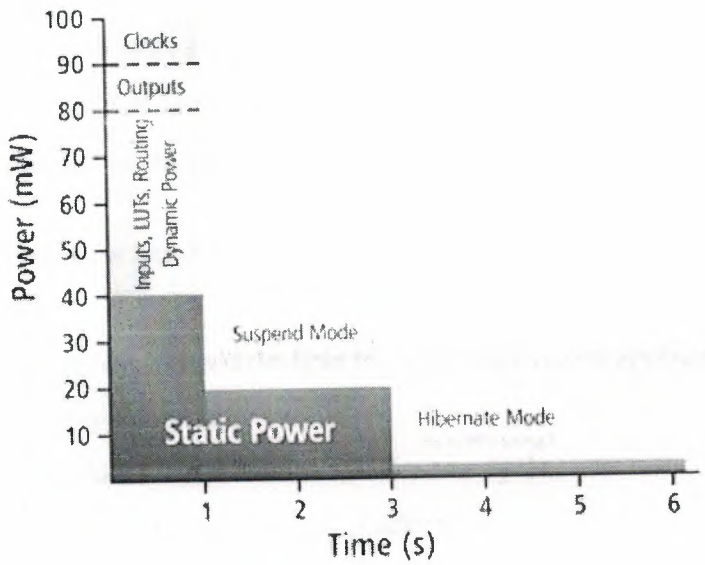

Figure.22 Power management

Dual power management modes allow the device to go to an extremely low power state which can reduce power consumption significantly.Available only in Spartan-3A/ 3AN/ 3A DSP FPGAs.

Multiple levels of security

Spartan-3 generation FPGAs offer the ultimate flexibility in customizing security solutions for high volume, low-cost systems.

- Customizable security algorithms utilizing unique Device DNA
- Monitor JTAG access and take action
- Monitor for tampering and alert for a bitstream alteration
- Ability to increase algorithm complexity
- Hidden bitstream deters bitstream snooping
- Tamper resistant packaging
- JTAG lockdown prevents "backdoor" access
- Readback disable to prevent configuration readback via JTAG or ICAP

Available only in Spartan-3A/ 3AN/ 3A DSP FPGAs.

Integrated Flash memory

This integrated memory found in Spartan-3AN FPGAs can be used for both device configuration as well as a valuable system resource for the user. It provides simple and secure embedded application storage while enabling advanced real-time control with fine-grained protection, lockdown and erase features.

- Simple and secure embedded application storage with up to 11Mb of integrated user Flash
- Enables single-chip board designs for space-conscious applications
- Worry-free configuration
- Twenty year data retention with 100K write cycles
- Pin compatible to the Spartan-3A platform

Available only in Spartan-3AN FPGAs.

XtremeDSP DSP 48A Slice

The Spartan-3A DSP platform's optimized DSP48A slice achieves 250 MHz operation in the slowest speed grade and enables advanced DSP functions to derive over 30 GMACS.

- XtremeDSP™ slices providing advanced MACC functionality
- Configurable logic blocks to store data and implement logic functions
- Precise clock management resources
- Advanced I/O structure
- 18-bit by 18-bit, two's complement multiplier with full precision 36-bit result, sign extended to 48 bits
- Pre-adder saves 9 logic slices per DSP48A used
- Two input, flexible 48-bit adder/subtracter with optional registered accumulation feedback
- DSP co-processing functions such as MAC engines, distributed algorithms and fully parallel FIR filters

Available only in Spartan-3A DSP FPGAs.

Embedded Processing

- Industry's most versatile, low-cost Embedded Processing platform
- Integrate processor into FPGA and reduce BOM
- Reduce obsolescence risks with soft processors
- Reduce inventory cost by using common flexible Embedded Processing architecture across multiple products.

Four level memory architecture

Four level memory architecture provides the optimal granularity and efficient area utilization.

- Up to 520 Kb distributed SelectRAM™+ memory
    - Each LUT works as a single-port or dual-port RAM/ROM

- LUTs can be cascaded to build larger memories
- Flexible memory for FIFOs, and buffers

- Up to 1.87 Mb embedded block RAM
  - Up to 104 blocks of synchronous 18 Kb block RAM can be cascaded
  - Each 18 Kb block can be configured as a single/dual-port RAM
  - Supports multiple aspect ratios, data-width conversion and parity
- Up to 16 Mb of integrated Flash memory
  - System flexibility with up to 11Mb of on-chip user Flash
  - Single-chip solution for failsafe field upgradeability using MultiBoot feature
  - New benchmark in non-volatile FPGA market for retention and cycling
- Popular low cost external memory
  - Connects low cost memories via interfaces such as HSTL and SSTL
  - Large system memory requirements

Memory Device	Electrical Interface
DDR SDRAM	SSTL 1.8V
DDR II SDRAM	SSTL 1.8V
DIMM DDR SDRAM	SSTL 2.5V
DIMM DDR II SDRAM	SSTL 1.8V
Network FCRAM	SSTL 2.5V
Network FCRAM II	SSTL 1.8V, HSTL 1.8V
RLDRAM	HSTL 1.8V
RLDRAM II	HSTL 1.8V
DDR SRAM	HSTL 2.5V, 1.8V
DDR II SRAM	HSTL 1.8V
QDR SRAM	HSTL 2.5V
QDR II SRAM	HSTL 1.8V
SyncBurst / ZBT SRAM	HSTL 2.5V

Leading connectivity platform

Implement multiple bridging, differential signaling, and memory interfaces with SelectIO™ technology.Supports most popular and emerging single-ended and differential signaling standards including TMDS, PPDS, SSTL3 Class I and II

Pre-engineered interface IP solutions including PCI™, PCI Express®, USB, Firewire, CAN, SPI, and I2C

Advanced interfacing supports up to 26 different single-ended and differential I/O standards ;

◦ Full hot-swap compliance and 3.3V support 622+ Mb/s data transfer rate per I/O

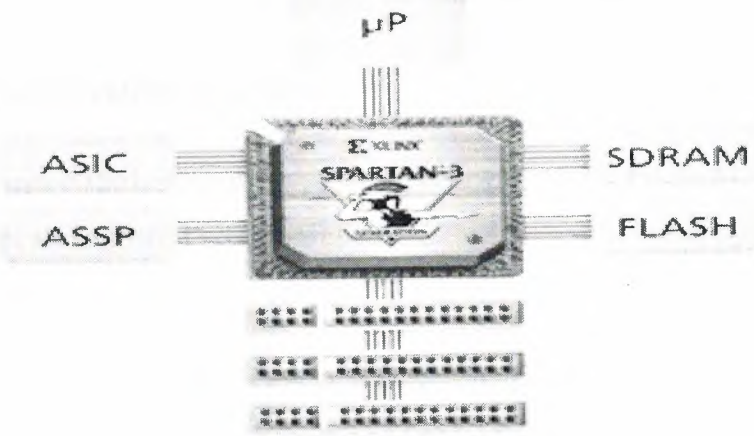

Figure.23 Connectivity platform

CLB architecture provides wider functionality and less logic levels resulting in higher performance.

- Four slices per CLB - two each for memory and logic functions.
- Wide-input functions - 16:1 mux in one CLB

- Fast arithmetic functions - two look-ahead carry chains per CLB column
- Four cascadable 16-bit addressable shift registers
- Two slices can be configured as distributed memory.

Precise clock management resources

A self-calibrating, fully digital solution for distributing, delaying, multiplying, dividing and phase-shifting clock signals.

- Up to 8 digital clock managers (DCMs)
- Flexible frequency generation from 5 MHz to 333 MHz
- Precision phase shift control for 0 - 360 degrees
- Fine grain control (1/256 clock period) for clock data synchronization
- Precise 50/50 duty cycle generation
- Up to 9 external outputs available for internal or external usage

Comprehensive configuration capabilities

The broadest flash memory support including Platform Flash, SPI and parallel Flash memories allow lowest cost configuration.

- Proprietary Platform Flash
    - Convenience of a single source supplier for FPGA and flash memory
    - Advanced features such as JTAG, bitsteam compression, design revisiton tracking
    - High speed programming
- MultiBoot capability allow multiple configurations
    - Failsafe upgrading
    - Application control of the bitstream selection replacement of large ASIC or multiple FPGAs with a single.

Spartan-3 generation FPGA

Third-party support for parallel, high-speed BPI configuration mode ;

- Fast parallel configuration speeds
- Easily procured through standard channels

- Commonly found on low-cost systems
- For larger densities, memory can be used for configuration and system functions.

2.6.3. Advantages

Spartan®-3 generation FPGAs offer the industry's leading multiple, value-centric platforms and features for low cost systems in high volume applications.

Lowest Total Cost. Period.

- Delivers lowest total cost
- Industry's largest selection of device/package options
- Industry's most comprehensive IP library
- Leading embedded and DSP solutions
- Efficient, cost-effective board designs
- Allows use of fewer standard components
- Increased system reliability by eliminating external components

Industry's only Dual Power Management Modes

- Instant power savings
- External component power reduction
- Built-in power savings features
- Advanced software tools to optimize low power design

Industry's Largest On-chip User Flash

- Superior system flexibility with up to 11Mb of on-chip user Flash
- Space-conscious applications

- Single-chip solution for failsafe field upgradeability using MultiBoot feature
- Worry free configuration
- New benchmark in non-volatile FPGA market for retention and cycling

Industry's Most Versatile, Low-Cost Embedded Processing Platform

Integrate MicroBlaze™ soft processor into FPGA and reduce BOM

Reduce obsolescence risks with soft processors.

Reduce inventory cost by using common flexible Embedded Processing architecture across multiple products.

Industry's only High Performance, Low Cost DSP Solution

Fill the DSP performance gap between traditional DSP processors and high-end ASIC and Virtex®-type solutions

Cost-optimized DSP architecture delivers superior results in performance and power consumption

Enables new applications in more cost-sensitive applications such as customer-premises wireless access, portable ultrasound, digital displays, surveillance, video processing.

2.6.4. SPARTAN-3 FPGA

- Get up to 50% lower total cost than competing FPGAs
- Achieve power savings instantly
- Enable low-cost security using Device DNA

2.6.4.1. High Logic and I/O Count

Spartan®-3 FPGAs offer platform capabilities with a wide range of I/O and density options. The Spartan-3 platform solution targets high logic/ pin count designs and applications.

2.6.4.2. What's in Spartan-3 FPGA?

Eight devices ranging from 50 K system gates to 5 M system gates Up to 104 Dual Port Block RAM - configurations from 1 to 36 bits wide Up to 104 embedded 18 x 18 multipliers that can be used with or without the BRAMs for DSP and complex math.Digital Clock Managers (DCM) for precision clocking and clock synthesis.XCITE technology which eliminates external termination resistors and simplifies PCB layout while lowering the BOM.Configuration using Platform Flash with bitstream compression.

The leading connectivity platform

Allows the ability to migrate to a larger or smaller device within the same package, without changing the pin out as the design changes Supports most popular and emerging single-ended and differential signaling standards including PCI and LVDS Up to 633 I/O pins Staggered I/O technology which increases the number of I/O per device compared to the competition

Low cost DSP solutions

1.8 billion multiply and accumulates (MACs) per second Up to 104 18x18 embedded multipliers for implementing compact DSP structures such as MAC engines, and adaptive and fully parallel FIR filters

SRL16 shift register logic and distributed memory for building compact DSP structures such as filters Block RAM for storing partial products and coefficients Complex DSP algorithms, such as Forward Error Correction (FEC) codecs, filters, for digital communications and imaging applications common functions such as a single channel, 64-tap FIR filter running at 8.1 MSPS that can be implemented for an effective cost.

.Configurable logic blocks

- Two slices per CLB – Four LUT / registers per CLB plus extra carry logic for math and logic functions
- Wide-input functions – 8:1 mux in one CLB
- Fast arithmetic functions – Two look-ahead carry chains per CLB column

- Four cascadable 16-bit addressable shift registers

Multi-level memory architecture

- Up to 520 Kb Distributed SelectRAM™ Memory
- Up to 1.87 Mb Embedded Block RAM
- Popular external memory Interfaces

Precise clock management resources

All digital delay-locked loop (DLL) in each DCM Up to 4 Digital Clock Managers (DCMs) per device Flexible frequency generation from 5 MHz to 333 MHz Precision phase shift control for 0, 90, 180 or 270 degrees Fine grain control (1/256 clock period) for clock data synchronization Precise 50/50 duty cycle generation.

Why Spartan-3 FPGA?

Lowest total cost. Period.

- Delivers lowest total cost
- Industry's largest selection of device/package options
- Industry's most comprehensive IP library
- Leading embedded and DSP solutions
- Efficient, cost-effective board designs
- Allows use of fewer standard components
- Increased system reliability by eliminating external components

Low-cost Embedded Processing platform

- Integrate MicroBlaze™ soft processor into FPGA and reduce BOM
- Reduce obsolescence risks with soft processors
- Reduce inventory cost by using common flexible Embedded Processing architecture across multiple products.

Complete design solution for optimal results

The industry's most complete programmable logic design solution for optimal performance, power management, cost reduction, and productivity

Our free, easy-to-use logic design solution for your Xilinx CPLD or medium-density FPGA, with all the tools included in ISE Foundation, on both Windows and Linux

Spartan-3 Generation Significantly Lowers Power :

Publications

- Spartan-3 Generation Brochure
- Spartan®-3 generation FPGAs significantly lowers power system designs. This means longer battery life for portable products as well as power efficiency.

Key factors that enable low power designs include:

- Dual power management modes
- Built-in power saving features
- I SE software low power options
- External component power reduction
- Partners for power savings

Dual power management modes

Significantly reduce power consumption with extremely low power states.

- Suspend mode

• A single pin activated mode that saves power on all internal states as well as switching the I/O pins to a predetermined state. Also all input signaling is halted with all internal circuits shut off.

• Hibernate mode

• A single pin activated mode excellent for extended sleep modes by tri-stating all I/Os. In this mode, it is safe to shut off all power.

Built-in power savings features; Save device power automatically with innovative features and techniques in ISE design tools.

Automatic block power savings Completely shuts down any unused circuits, routing and blocks within the device implementation. Clock resources power savings

Disables unused clock nets on regions, columns and CLBs by organizing and refining into the minimum number of independent clock domains.

Performs further optimization by clocking each domain at the lowest possible frequency and use register enable on non-changing clock transitions to reduce power.

Additional power saving techniques can be employed:

- Clock gating / multiplexing
- Digital clock management (DCM)
- Implement DCM external to logic
- Distributed memory
- Block RAM
- Smart architecting block memory optimizing for power vs. speed

Optimize low power with ISE software.Performance or utilization driven parameters can be chosen when compiling a design.

Power optimization mode reduces routing capacitance to a minimum without altering timing constraints.External component power reduction Integrate external components as well as eliminate them.

Fewer power rail requirements reduce voltage regulator counts and well as remove decoupling circuitry due to digital DCM implementation.

Power draw of buffers and line drivers are saved due to quality design of output drive.

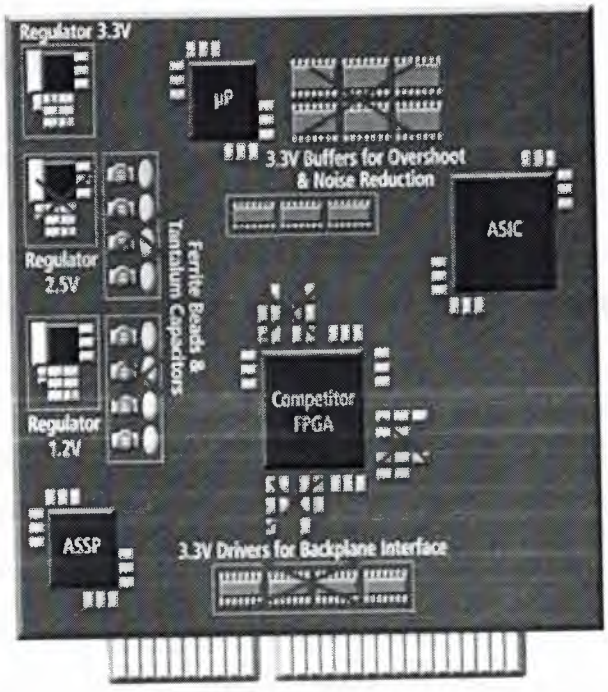

Figure.24 *Spartan-3 generation FPGAs eliminate the need for additional external components.*

Design Security Solutions

Protect Your Brand with Spartan®-3A/3AN/3A DSP FPGAs

Companies lose over $500B in lost sales every year because of counterfeit products. This threat grows by more than 12% per year, tarnishing the reputation.

Spartan-3A/3AN/3A DSP devices enable designers to implement a low cost, highly robust security solution to deter reverse-engineering, cloning and overbuilding.

Our Device DNA demo highlights how easily Device DNA technology can be used to implement a low-cost security solution for high volume applications. It requires either the Spartan-3A FPGA Starter Kit or the Spartan-3AN FPGA Starter Kit.

How Device DNA Works

The Device DNA security mechanism is similar to an ATM transaction.

In an ATM transaction, the active value generated from the card + pin combination is compared to a number stored in a bank computer that authorizes or rejects the transaction.

Similarly, the unique 57-bit Device DNA number is used with a customer-defined security algorithm to generate an active value. The active value is compared to a pre-stored check value to determine whether design functionality can proceed.

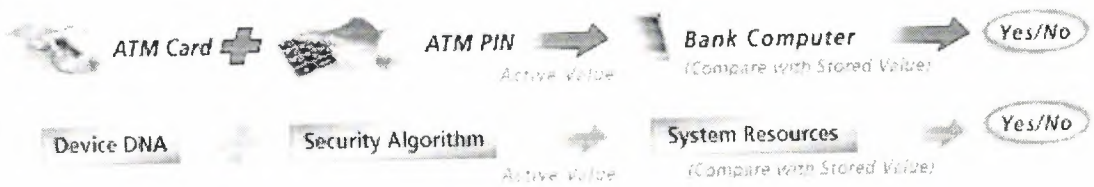

Figure.25 Similarities between Device DNA security and ATM transactions.

You can also enhance your design security with advanced solutions such as:

- Active defense.
- Bitstream validation.
- Advanced data manipulation.

Spartan-3 Compatibility

Within the Spartan-3 family, all devices are pin-compatible by package. When the need for future logic resources outgrows the capacity of the Spartan-3 device in current use, a larger device in the same package can serve as a direct replacement. Larger devices may add extra VREF and VCCOlines to support a greater number of

I/Os. In the larger device, more pins can convert from user I/Os to VREF lines.Also, additional VCCO lines are bonded out to pins that were "not connected" in the smaller device. Thus, it is important to plan for future upgrades at the time of the board's initial design by laying out connections to the extra pins.

The Spartan-3 family is not pin-compatible with any previous Xilinx FPGA family or with other platforms among the Spartan-3 Generation FPGAs.

CONCLUSION

The Design that has telling is Barrel Shifter,It shifts a data word by shift amount.It is Synchronous with the clock.It uses VHDL as a language, XILINX ISE 9.1i as software.

In this project, designed the 16 bit Barrel Shifter using VHDL language and the Xilinx ISE tools to program the Spartan 3 FPGA. First writing the requirements, specification, define the Barrel Shifter inputs and outputs, define the Barrel Shifter function,the VHDL code. Then using the Xilinx ISE tools. To complete the project.

In a Barrel Shifter the bits are shifted the desired number of bit positions in a single clock cycle that can attainment time by using a VHDL Xilinx.The Barrel Shifter is faster then the other shift registers.Because the data can shift as wanting to shift in a one clock cycle.And user do not neccessary to busy with any complex electronic circuits, just writing a program and then connecting the device that users want to regulate.

To sum up the programming in Xilinx has become so functional that required results can be easily achieved with faultless results.

REFERENCES

1. Instractor MEHMET KADİR ÖZAKMAN's course notes(www.members.shaw/kadirm)
2. www.xilinx.com
3. Xilinx-ISE9.1i Software Manuals
4. Xilinx-ISE9.1i Quick Start Manuals