

NEAR EAST UNIVERSITY



Faculty of Engineering

Department of Computer Engineering

SIXTY COUNTER

**Graduation Project
COM-400**

Student : Hakan Şahin(20033641)

Supervisor : Mehmet Kadir Özakman

Nicosia-2008

ACKNOWLEDGEMENTS

“First, I would like to thank my supervisor Mehmet Kadir Özakman for his invaluable advice and belief in my work and myself over the course of this Graduation Project..

Second, I would like to Express my gratitude to Near East University for the scholarship that made the work possible.

Third, I thank my family for their constant encouragement and support during the preparation of this project.

Finally, I would also like to thank all my friends for their advice and support.”

ABSTRACT

We will design sixty counter in this project . Sixty counter is an electronic desgin.We are using sixty counter in different areas as digital watch .

The xilinx ise 9.1i software will be used for to create the sixty counter design . We have selected this program because is very useful for to do this electronic design . We can design many things that we want using xilinx ise software.

We will explain briefly why xilinx ise software is useful and suitable for us.

Assume that you have a company you are working IT(information technology) sector . You can do many specific solutions , one day if you need a 32 bit processor for example.You are calling xilinx company and then you are saying 'we want a 32 bit processor'. Then they are sending a spesific FPGA (field programmable logic gate)chip kit(virtex,spartan etc..) as special for your request . Then you are taking the kit and connecting the internet at where you are . They are loading a 32 bit processor software your kit using the internet . So you have a 32 bit processor.

It's very useful for the companies because if you want you can change your processor to ram or rom etc.. You can convert so many things . If you have a kit on your hand you can create many things using this kit and xilinx software.

So we hope this technology will grow up and there will be so many vacany therefore we want to learn this technology and we used this software.

TABLE OF CONTENTS

ACKNOWLEDGMENT	i
ABSTRACT	ii
TABLE OF CONTENTS	iii
LIST OF ABBREVIATIONS	iv
INTRODUCTION	1
CHAPTER ONE : DESIGN	3
1.1)DESIGN DECRPTION	3
1.2) 1.2)ABOUT THE DESIGN	5
CHAPTER TWO : DESIGN STEPS	7
2.1)ENTER THE DESIGN	7
2.2)CREATING a VHDL SOURCE	8
2.3)GENERATING a CODE	11
2.4)SYNTHESIZE	17
2.5)WRITING a TEST BENCH	30
2.6)SIMULATING	34
2.7)EXPLANATION OF MY SIXTY COUNTER PROGRAM	36
CODES	
CHAPTER THREE :SPARTAN	40
3.1)OVERVIEW	40
3.2)CAPABILITIES	41
3.2.1)DUAL POWER MANAGEMENT	41
3.2.2)MULTIPLE LEVELS OF SECURITY	42
3.2.3)INTEGRATED FLASH MEMORY	42
3.2.4)XtremeDSP DSP 48A SLICE	43
3.2.5)EMBEDDED PROCESSING	43

3.2.6)FOUR LEVEL MEMORY ARCHITECTURE	44
3.2.7)LEADING CONNECTIVITY PLATFORM	45
3.2.8)CONFIGURABLE LOGIC BLOCKS	46
3.2.9)PRECISE CLOCK MANAGEMENT RESOURCES	46
3.2.10)COMPREHENSIVE CONFIGURATION	46
CAPABILITIES	
3.3)ADVANTAGES	47
CHAPTER FOUR:HISTORY	50
4.1)WHAT DOES XILINX MEAN ?	50
4.2) WHAT DOES XILINX NAME REPRESENT?	50
4.3)HOW XILINX BEGAN ?	53
4.3.1)NEV TECHNOLOGY	53
4.3.2)EFFECTIVE PARTNERSHIPS	54
4.3.3)INSPIRED EMPLOYEES	54
4.4)BUSINESS	55
4.4.1)LEARN ABOUT THE XILINX TECHNOLOGY,HOW and WHY TO PURPOSE IT ?	55
4.4.2)PROGRAMMABLE LOGIC IS XILINX's BUSINESS	55
4.4.3)USES FOR PROGRAMMABLE LOGIC	56

4.4.4)MULTIPLE PRODUCT LINES WITH SUPERLATIVE SOFTWARE SUPPORT	56
4.4.5)HIGH-PROFILE WORLDWIDE CUSTOMER and PARTNER BASE	56
4.4.6)XILINX'S VISION FOR THE FUTURE	57
4.5)GETTIN STARTED WITH FPGAs	57
4.5.1)WHAT ARE FPGAs?	57
4.5.2)COMMON FPGA FEATURES	59
4.5.3)FPGA SOLUTIONS, APPLICATIONS and END- MARKETS	60
4.6)XILINX's SUCCEES	62
4.6.1)THE SYNERGY OF TECHNOLOGY, PARTNERSHIP, and LEADERSHIP	62
4.6.2)XILINX's PARTNERS	62
4.6.3)XILINX's TECHNOLOGY	63
4.6.4)XILINX's EMPLOYEES	63
4.7)XILINX's VALUES	64
4.7.1)HOW XILINX WORK WITH ONE ANOTHER AND XILINX's PARTNERS, WHAT DO VALUES MEAN TO XILINX?	64

4.7.2)HOW DID XILINX CLEARLY DEFINE 64

IT's VALUES ?

4.7.3)HOW DO XILINX KEEP IT's VALUES 65

VISIBLE and VIABLE IN THE COMPANY ?

CONCLUSION 67

REFERENCES 69

LIST OF ABBREVIATIONS

ISE	Integrated Software Environment
FPGA	Field Programmable Gate Array
VHDL	Very high speed integrated circuit Hardware Description Language
DCM	Digital Clock Manager
RPM	Relationally Placed Macro
DUT	Design Under Test
UUT	Unit Under Test

INTRODUCTION

About the project , we will create a sixty counter . Sixty counter is a counter and when we run it , it counts up to sixty .

Block diagram of the Project looks like this .

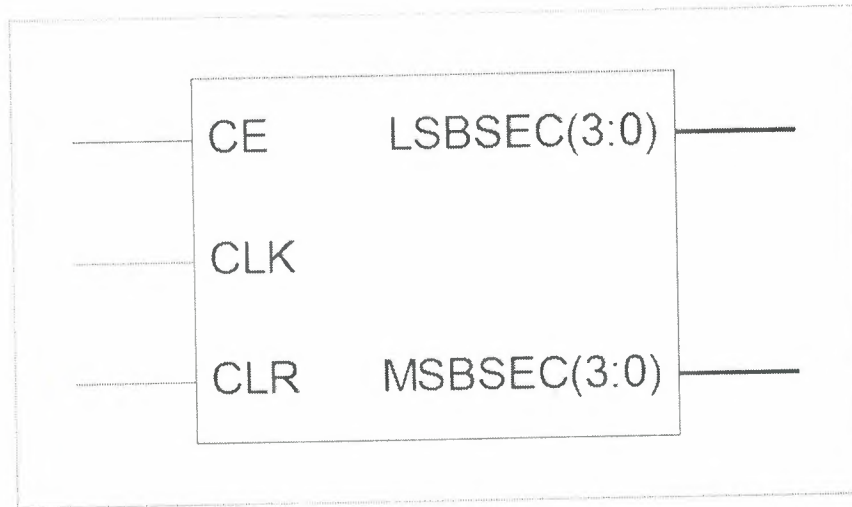


Figure1. block diagram

In the first chapter block diagram of the Project will be given and inputs and outputs from the block diagram will be defined . After that information about which language we used to create the project and what we used in software and hardware will be given . Then we will explain the Xilinx ISE 9.1 . In this chapter lastly we will show the design flow .

In the second chapter we will follow the steps which we gave at the end of chapter one . In the first step we will enter the design , the name of the project and directory of Project will be given by us . We will select a project device properties . Second step is creating a VHDL source . After creating a source we generate the codes . Then we check the generated codes from check syntax , to be sure if we have mistakes or not . Then we create a test bench . After creating test bench we get simulation to see the output of the project . In this chapter lastly we will give an information about the project .

In chapter three an information will be given by us about the spartan . We will give an overview and capabilities and advantages of spartan .

In chapter four we will talk about xilinx . We will explain the mean of xilinx , and what the name of xilinx represents . In business what xilinx represents .

As you see last two chapter is general information about the xilinx and spartan . But the first two chapter is about the project .

CHAPTER one DESIGN

1.1)DESIGN DECRPTION

The design is sixty counter . We are going to do a Sixty counter which counts up to sixty .

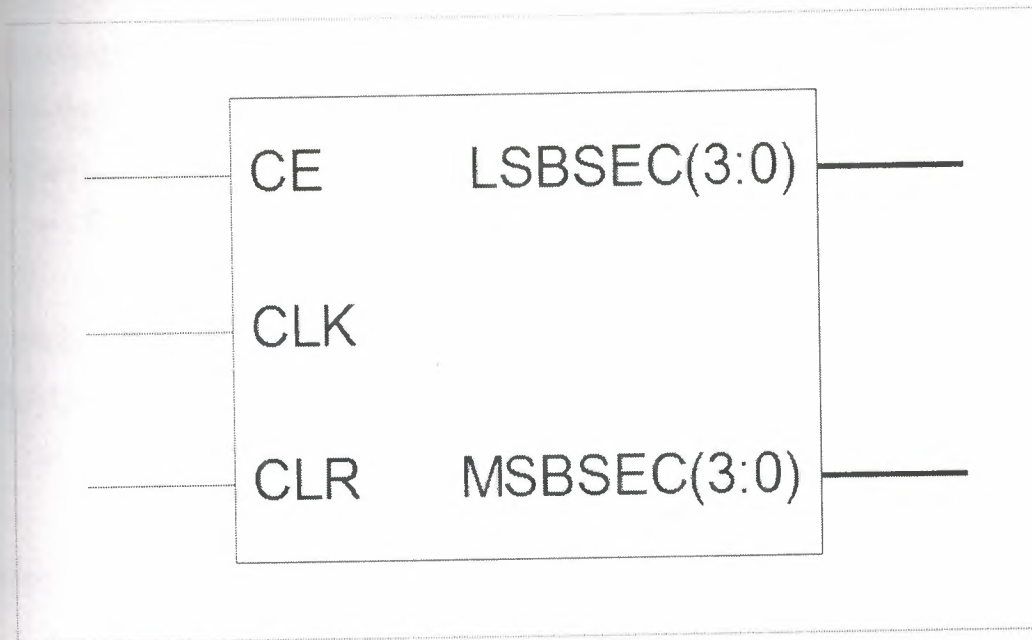


Figure2. Sixty counter block diagram

Inputs :

CLK

CLR

CE

The counter counts the clock cycles . If the clock cycle is 1second conter will count up to 1minute . If the clock cycle is 1minute counter will count up to 60minute . Clear is clears all the values . Clock enable is enables counter to count .

Outputs :

LSBSEC

MSBSEC

Lsbsec is counts from 0 to 9 . Msbsec is counts from 0-5 .

Used things : To create the project VHDL programming language will be used and there will be a some hardware and software requirements . We will use in software Xilinx ISE and hardware Spartan3E .

The briefly explanation of Xilinx ISE : The Integrated Software Environment (ISE™) is the Xilinx® design software suite that allows that , take the design from design entry through Xilinx device programming. The ISE Project Navigator manages and processes the designs through the following steps in the ISE design flow .

- **DESIGN ENTRY**

Design entry is the first step in the ISE design flow. During design entry, we can create our source files based on design objectives. We can create your top-level design file using a Hardware Description Language (HDL), such as VHDL, Verilog, or ABEL, or using a schematic. We can use multiple formats for the lower-level source files in the design.

- **SYNTHESIS**

After design entry and optional simulation, we run synthesis. During this step, VHDL, Verilog, or mixed language designs become netlist files that are accepted as input to the implementation step.

- **CREATE TEST BENCH**

The results of the works during all the program is displayed in this process . Also in this process some values is given to get the outputs .

- **IMPLEMENTATION**

After synthesis, we run design implementation, which converts the logical design into a physical file format that can be downloaded to the selected target device. From Project Navigator, we can run the implementation process in one step, or we can run each of the implementation processes separately. Implementation processes vary depending on whether we are targeting a Field Programmable Gate Array (FPGA) or a Complex Programmable Logic Device (CPLD).

- **VERIFICATION**

We can verify the functionality of our design at several points in the design flow. We can use simulator software to verify the functionality and timing of our design or a portion of our design. The simulator interprets VHDL or Verilog code into circuit functionality and displays logical results of the described HDL to determine correct circuit operation. Simulation allows us to create and verify complex functions in a relatively small amount of time. We can also run in-circuit verification after programming our device.

- **DEVICE CONFIGURATION**

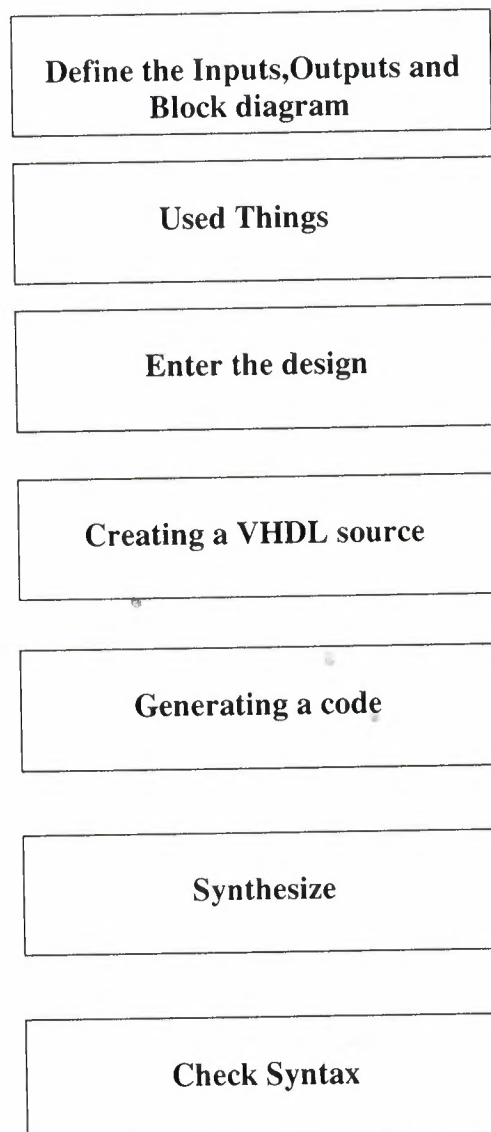
After generating a programming file, we configure our device. During configuration, we generate configuration files and download the programming files from a host computer to a Xilinx device.

1.2)ABOUT THE DESIGN

In this project we we have created a count sixty counter . This counter will count up to sixty . To create the counter we need two counter and the counters will work together . First counter will count from 0-9 and when the first counter comes nine it will add one to the second counter . By this way first counter will count the six times and the counter will reach the sixty .

Sixty counter can be used in digital watch .

The design flow lookslike this :



Creating a test bench

Simulating

Explaining project

Figure3. design flow

CHAPTER two DESIGN STEPS

2.1) ENTER THE DESIGN

In the first step we will create the Project basically . To do this firstly we must setup the XILINX ISE 9.1I . After the installation of the program we open the program and then we have some steps to create the program as follows :

1. Select File > New Project... The New Project Wizard appears.
2. Type SIXTY in the Project Name field.
3. Enter or browse to a location (directory path) for the new project. The works subdirectory is created automatically.

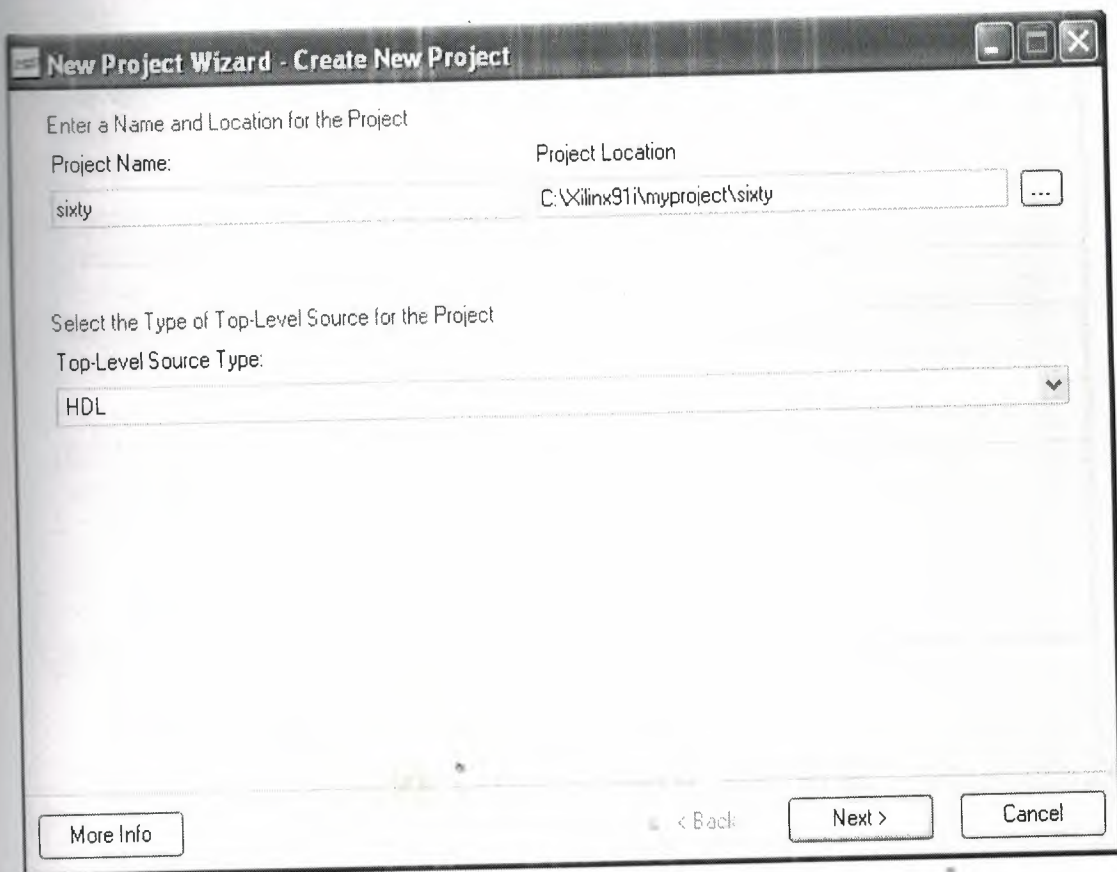


Figure4.Project name

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: Spartan3E

Device: XC3S100E

Package: VQ100

Speed Grade: -5

Top-Level Source Type: HDL

Synthesis Tool: XST (VHDL/Verilog)

Simulator: ISE Simulator (VHDL/Verilog)

Preferred Language: VHDL

Verify that Enable Enhanced Design Summary is selected.

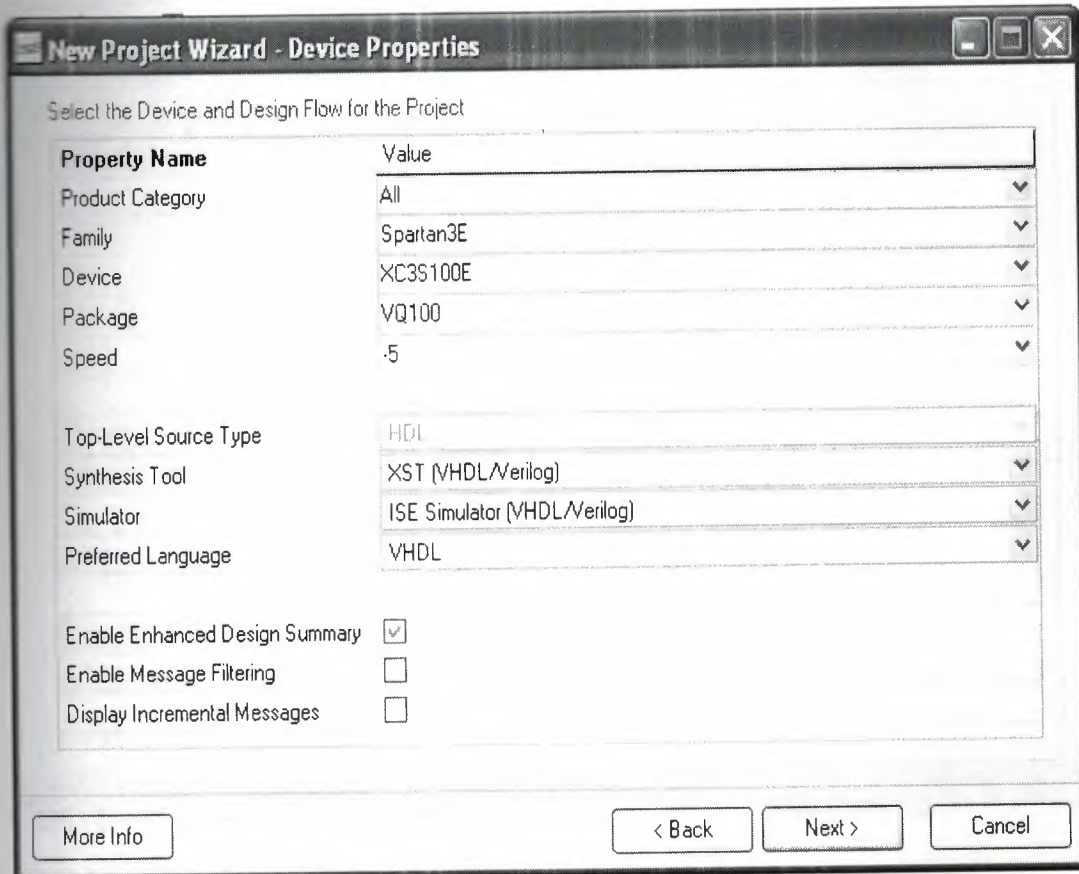


Figure5. Project Device Properties

Leave the default values in the remaining fields.

2.2) CREATING a VHDL SOURCE

Create a VHDL source file for the project as follows:

1. Click the New Source button in the New Project Wizard.
2. Select VHDL Module as the source type.
3. Type in the file name CNT60.

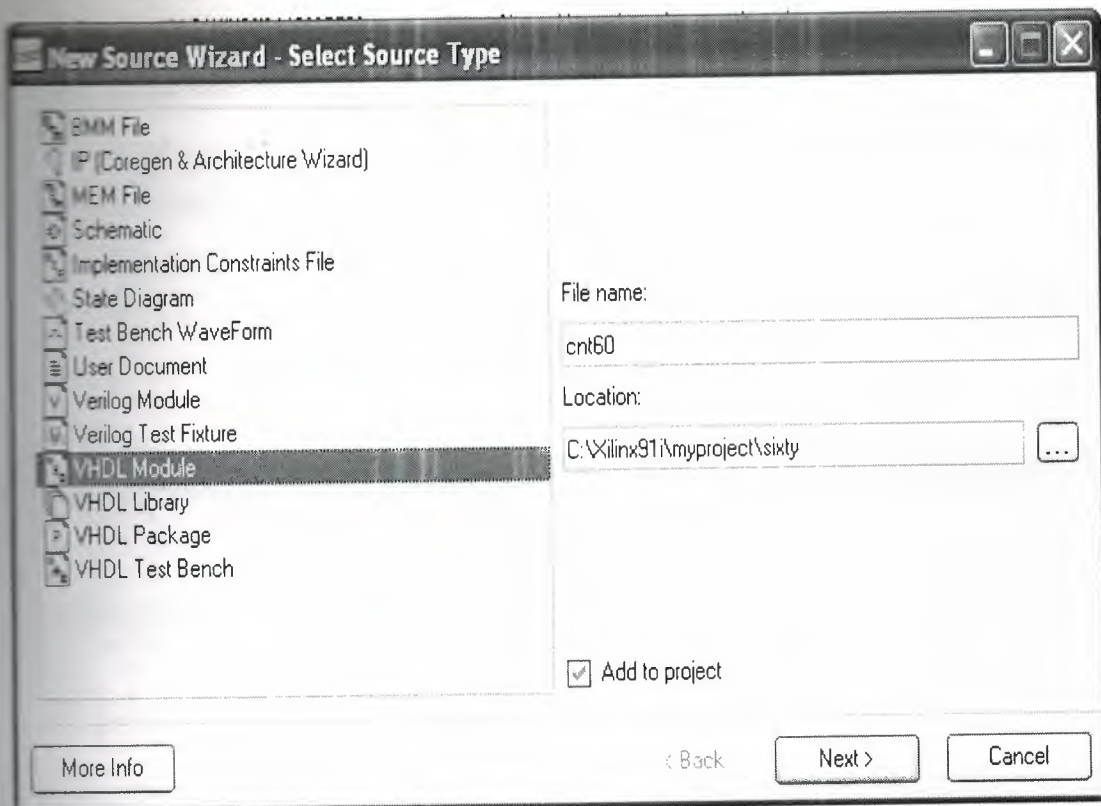


Figure6. Adding source

4. Verify that the Add to project checkbox is selected.
5. Click Next.
6. Declare the ports for the counter design by filling in the port information as shown below:

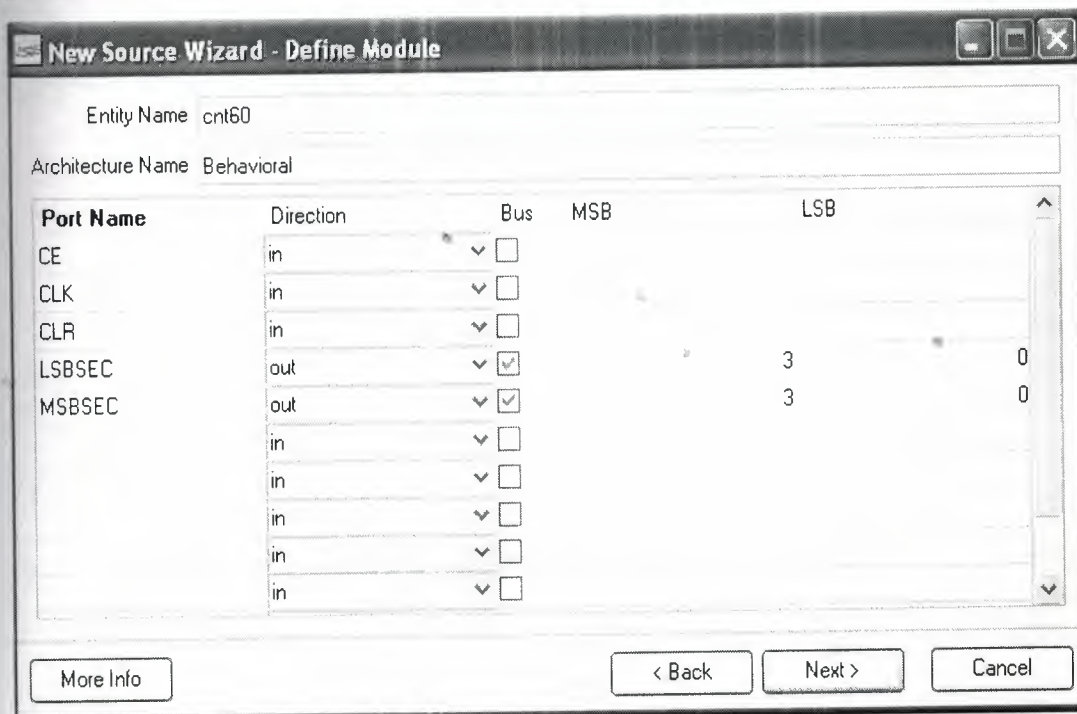


Figure7. Define Module

7. Click Next, then Finish in the New Source Wizard - Summary dialog box to complete the new source file template.

8. Click Next, then Next, then Finish.

The source file containing the entity/architecture pair displays in the Workspace, and the counter displays in the Source tab, as shown below:



Figure8.Program

2.3) GENERATING a CODE

In next step we add some necessary signals , process and some codes to the program . After that the program will be as shown :

```
-- Company:
-- Engineer:
--
-- Create Date: 12:38:22 05/08/2008
-- Design Name:
-- Module Name: cnt60 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity cnt60 is
```

```

Port ( CE : in STD_LOGIC;
      CLK : in STD_LOGIC;
      CLR : in STD_LOGIC;
      LSBSEC : out STD_LOGIC_VECTOR (3 downto 0);
      MSBSEC : out STD_LOGIC_VECTOR (3 downto 0));
end cnt60;
architecture Behavioral of cnt60 is
component scntnr
port ( CE : in STD_LOGIC ;
      CLK : in STD_LOGIC ;
      CLR : in STD_LOGIC ;
      QOUT : out STD_LOGIC_VECTOR ( 3 downto 0 ));
end component;
signal lsbout : STD_LOGIC_VECTOR (3 downto 0) ;
signal msbout : STD_LOGIC_VECTOR (3 downto 0) ;
signal msbce : STD_LOGIC ;
signal lsbtc : STD_LOGIC ;
signal msbclr : STD_LOGIC ;
signal msbtc : STD_LOGIC ;
begin
lsbcount : scntnr port map (CE => CE , CLK => CLK , CLR => CLR , QOUT =>
lsbout);
msbcount : scntnr port map (CE => msbce ,CLK => CLK,CLR=> msbclr,
QOUT=>msbout);
process (lsbout)
begin
if (lsbout="1001") then
lsbtc<='1';
else
lsbtc<='0';
end if;
end process;
process (msbout)
begin

```

```

if (msbout="0110")then
msbtc<='1';
%
msbtc<='0';
end if ;
end process;

msbce<=CE and lsbtc;
msbclr <= CLR or msbtc;
LSBSEC <= lsbout ;
MSBSEC <= msbout ;
end Behavioral;

```

Now one more process must be created . Because we have two counter in this program so second counter we are adding a other source as following :

1. Click the New Source button in the New Project Wizard.
2. Select VHDL Module as the source type.
3. Type in the file name SCNTR.

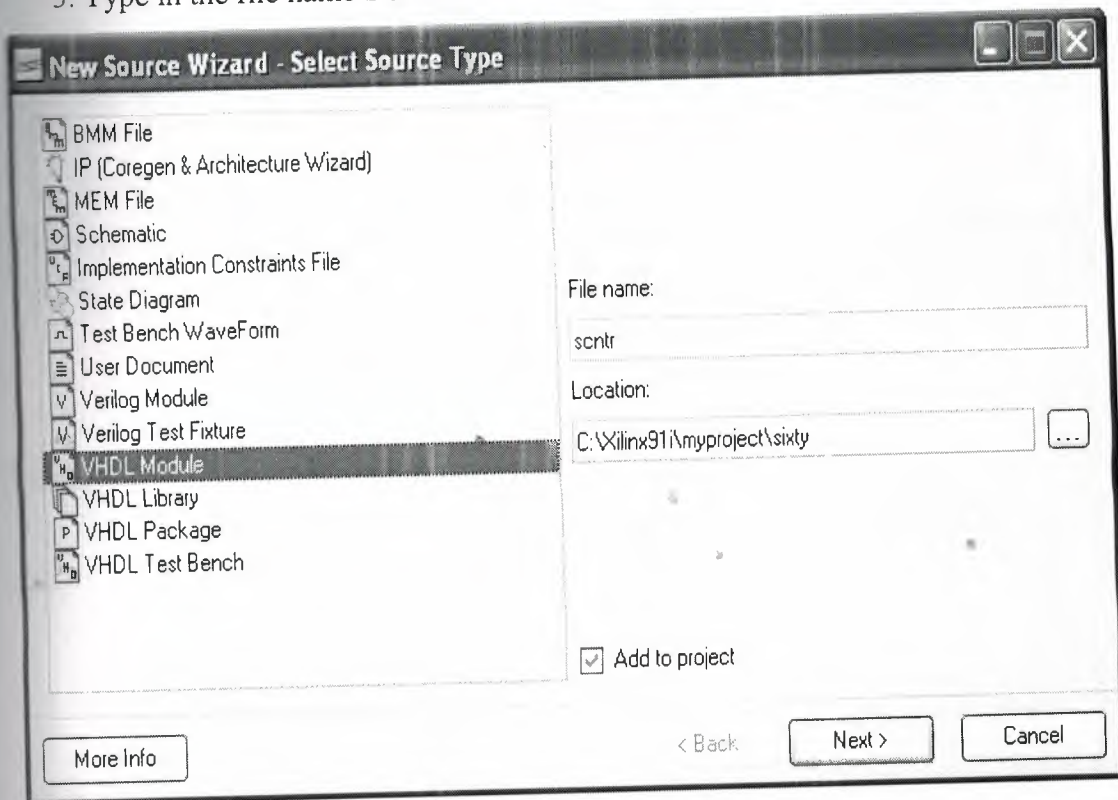


Figure9.Adding source

4. Verify that the Add to project checkbox is selected.
5. Click Next.

6. Declare the ports for the counter design by filling in the port information as shown below:

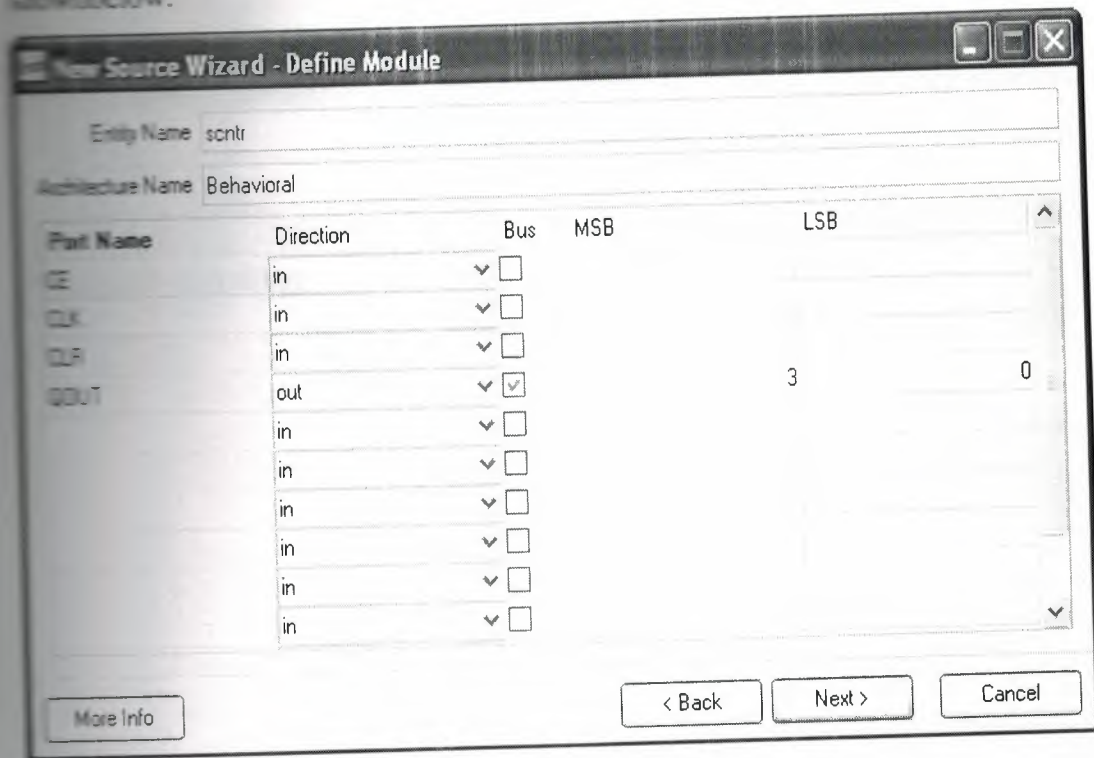


Figure10. Define Module

7. Click Next, then Finish in the New Source Wizard - Summary dialog box to complete the new source file template.

8. Click Next, then Next, then Finish.

The source file containing the entity/architecture pair displays in the Workspace, and the counter displays in the Source tab, as shown below:

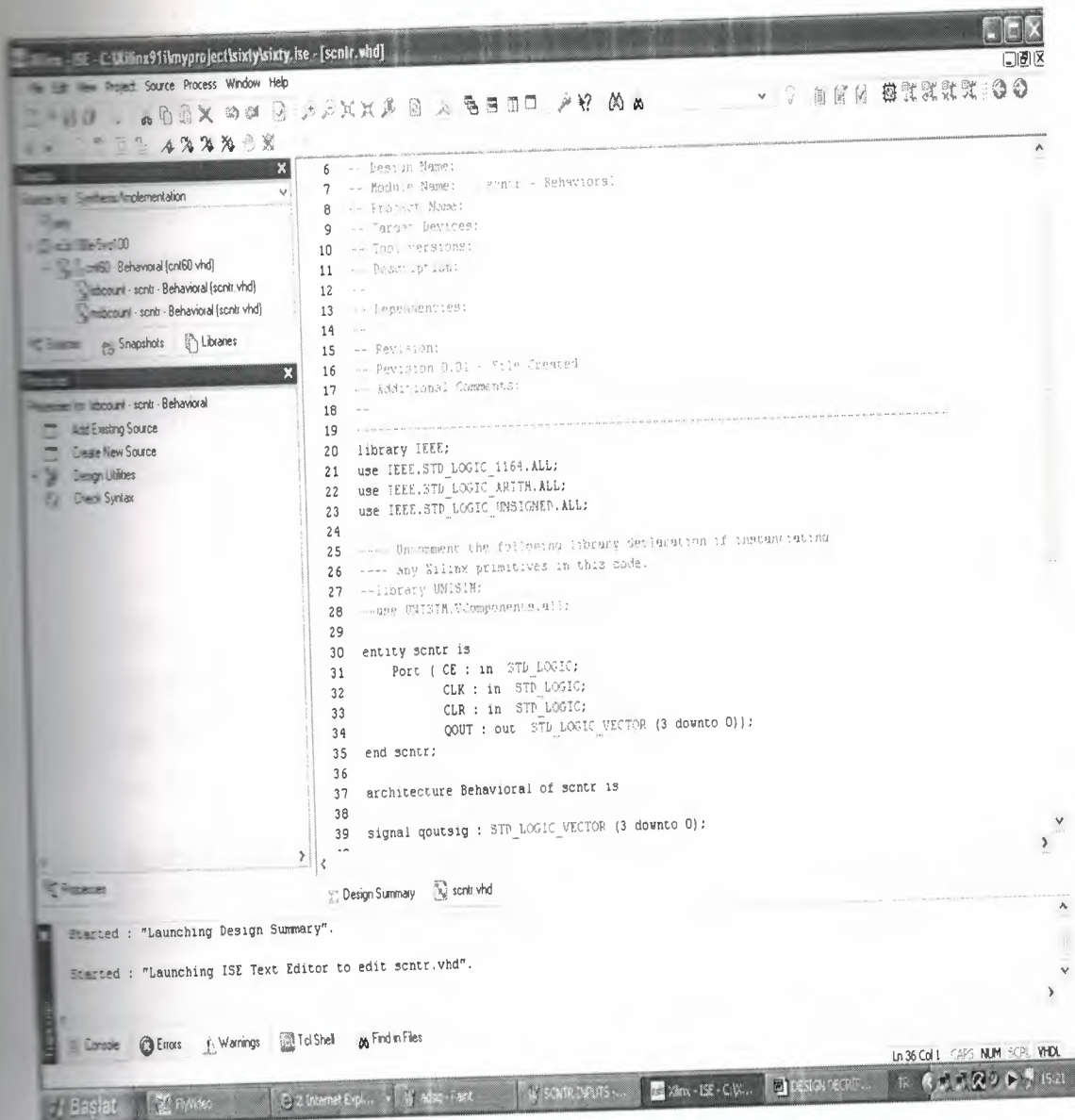


Figure 11. Program

In next step we add some necessary signals , process and some codes to the program . After that the program will be as shown :

```

-----
-- Company:
-- Engineer:
--
-- Create Date: 13:01:54 05/08/2008
-- Design Name:
-- Module Name: scntr - Behavioral
-- Project Name:

```

```
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
```

```
-----
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity scnr is
    Port ( CE : in STD_LOGIC;
          CLK : in STD_LOGIC;
          CLR : in STD_LOGIC;
          QOUT : out STD_LOGIC_VECTOR (3 downto 0));
end scnr;
architecture Behavioral of scnr is
    signal qoutsig : STD_LOGIC_VECTOR (3 downto 0);
    begin
    process (CE,CLK,CLR)
        begin
        if (CLR='1') then
            qoutsig<="0000";
        elsif (CE='1')then
            if (CLK'event and CLK ='1') then
```




```

if (qoutsig="1001") then
qoutsig<="0000";
else
qoutsig<= qoutsig+"0001";
end if;
end if;
end process;
QOUT<=qoutsig;
end Behavioral;

```

2.4)SYNTHESIZE

After design entry and optional simulation, we run synthesis. In the Sources tab, select Synthesis/Implementation from the Design View drop-down list, and select the top module . In the Processes tab, double-click Synthesize.

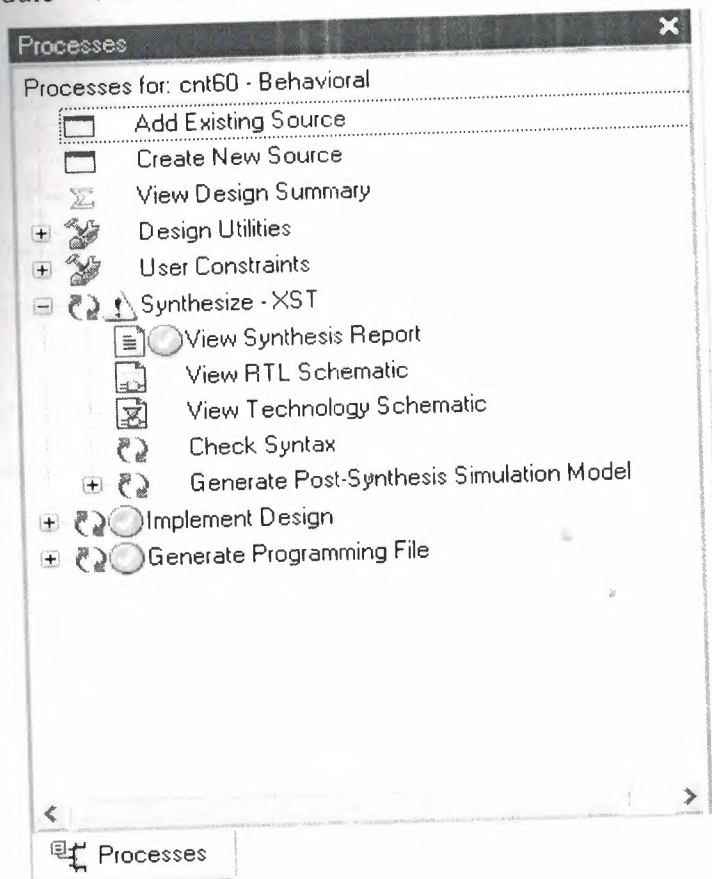


Figure12.Select synthesize

As you can see from the figure.12 we can get the synthesis report , we can look RTL schematic of the our design , we can look the technology schematic of the design and we can check the syntax to be sure that the if we have mistakes or not .

Synthesis Report : This report contains the results from the synthesis run, including area and timing estimation.

Release 9.1i - xst J.30

Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.

-> Parameter TMPDIR set to ./xst/projnav.tmp

CPU : 0.00 / 0.36 s | Elapsed : 0.00 / 0.00 s

-> Parameter xsthdpdir set to ./xst

CPU : 0.00 / 0.36 s | Elapsed : 0.00 / 0.00 s

-> Reading design: cnt60.prj

TABLE OF CONTENTS

- 1) Synthesis Options Summary
- 2) HDL Compilation
- 3) Design Hierarchy Analysis
- 4) HDL Analysis
- 5) HDL Synthesis
 - 5.1) HDL Synthesis Report
- 6) Advanced HDL Synthesis
 - 6.1) Advanced HDL Synthesis Report
- 7) Low Level Synthesis
- 8) Partition Report
- 9) Final Report
 - 9.1) Device utilization summary
 - 9.2) Partition Resource Summary
 - 9.3) TIMING REPORT

=====
=====

*	Synthesis Options Summary	*
---	---------------------------	---

=====
=====

---- Source Parameters

Input File Name : "cnt60.prj"

NEAR EAST UNIVERSITY



Faculty of Engineering

Department of Computer Engineering

SIXTY COUNTER

**Graduation Project
COM-400**

Student : Hakan Şahin(20033641)

Supervisor : Mehmet Kadir Özakman

Nicosia-2008

ACKNOWLEDGEMENTS

“First, I would like to thank my supervisor Mehmet Kadir Özakman for his invaluable advice and belief in my work and myself over the course of this Graduation Project..

Second, I would like to Express my gratitude to Near East University for the scholarship that made the work possible.

Third, I thank my family for their constant encouragement and support during the preparation of this project.

Finally, I would also like to thank all my friends for their advice and support.”

ABSTRACT

We will design sixty counter in this project . Sixty counter is an electronic desgin.We are using sixty counter in different areas as digital watch .

The xilinx ise 9.1i software will be used for to create the sixty counter design . We have selected this program because is very useful for to do this electronic design . We can design many things that we want using xilinx ise software.

We will explain briefly why xilinx ise software is useful and suitable for us.

Assume that you have a company you are working IT(information technology) sector . You can do many specific solutions , one day if you need a 32 bit processor for example.You are calling xilinx company and then you are saying 'we want a 32 bit processor'. Then they are sending a spesific FPGA (field programmable logic gate)chip kit(virtex,spartan etc..) as special for your request . Then you are taking the kit and connecting the internet at where you are . They are loading a 32 bit processor software your kit using the internet . So you have a 32 bit processor.

It's very useful for the companies because if you want you can change your processor to ram or rom etc.. You can convert so many things . If you have a kit on your hand you can create many things using this kit and xilinx software.

So we hope this technology will grow up and there will be so many vacany therefore we want to learn this technology and we used this software.

TABLE OF CONTENTS

ACKNOWLEDGMENT	i
ABSTRACT	ii
TABLE OF CONTENTS	iii
LIST OF ABBREVIATIONS	iv
INTRODUCTION	1
CHAPTER ONE : DESIGN	3
1.1)DESIGN DECRPTION	3
1.2) 1.2)ABOUT THE DESIGN	5
CHAPTER TWO : DESIGN STEPS	7
2.1)ENTER THE DESIGN	7
2.2)CREATING a VHDL SOURCE	8
2.3)GENERATING a CODE	11
2.4)SYNTHESIZE	17
2.5)WRITING a TEST BENCH	30
2.6)SIMULATING	34
2.7)EXPLANATION OF MY SIXTY COUNTER PROGRAM	36
CODES	
CHAPTER THREE :SPARTAN	40
3.1)OVERVIEW	40
3.2)CAPABILITIES	41
3.2.1)DUAL POWER MANAGEMENT	41
3.2.2)MULTIPLE LEVELS OF SECURITY	42
3.2.3)INTEGRATED FLASH MEMORY	42
3.2.4)XtremeDSP DSP 48A SLICE	43
3.2.5)EMBEDDED PROCESSING	43

3.2.6)FOUR LEVEL MEMORY ARCHITECTURE	44
3.2.7)LEADING CONNECTIVITY PLATFORM	45
3.2.8)CONFIGURABLE LOGIC BLOCKS	46
3.2.9)PRECISE CLOCK MANAGEMENT RESOURCES	46
3.2.10)COMPREHENSIVE CONFIGURATION	46
CAPABILITIES	
3.3)ADVANTAGES	47
CHAPTER FOUR:HISTORY	50
4.1)WHAT DOES XILINX MEAN ?	50
4.2) WHAT DOES XILINX NAME REPRESENT?	50
4.3)HOW XILINX BEGAN ?	53
4.3.1)NEV TECHNOLOGY	53
4.3.2)EFFECTIVE PARTNERSHIPS	54
4.3.3)INSPIRED EMPLOYEES	54
4.4)BUSINESS	55
4.4.1)LEARN ABOUT THE XILINX TECHNOLOGY,HOW and WHY TO PURPOSE IT ?	55
4.4.2)PROGRAMMABLE LOGIC IS XILINX's BUSINESS	55
4.4.3)USES FOR PROGRAMMABLE LOGIC	56

4.4.4)MULTIPLE PRODUCT LINES WITH SUPERLATIVE SOFTWARE SUPPORT	56
4.4.5)HIGH-PROFILE WORLDWIDE CUSTOMER and PARTNER BASE	56
4.4.6)XILINX'S VISION FOR THE FUTURE	57
4.5)GETTIN STARTED WITH FPGAs	57
4.5.1)WHAT ARE FPGAs?	57
4.5.2)COMMON FPGA FEATURES	59
4.5.3)FPGA SOLUTIONS, APPLICATIONS and END- MARKETS	60
4.6)XILINX's SUCCEES	62
4.6.1)THE SYNERGY OF TECHNOLOGY, PARTNERSHIP, and LEADERSHIP	62
4.6.2)XILINX's PARTNERS	62
4.6.3)XILINX's TECHNOLOGY	63
4.6.4)XILINX's EMPLOYEES	63
4.7)XILINX's VALUES	64
4.7.1)HOW XILINX WORK WITH ONE ANOTHER AND XILINX's PARTNERS, WHAT DO VALUES MEAN TO XILINX?	64

4.7.2)HOW DID XILINX CLEARLY DEFINE 64

IT's VALUES ?

4.7.3)HOW DO XILINX KEEP IT's VALUES 65

VISIBLE and VIABLE IN THE COMPANY ?

CONCLUSION 67

REFERENCES 69

LIST OF ABBREVIATIONS

ISE	Integrated Software Environment
FPGA	Field Programmable Gate Array
VHDL	Very high speed integrated circuit Hardware Description Language
DCM	Digital Clock Manager
RPM	Relationally Placed Macro
DUT	Design Under Test
UUT	Unit Under Test

INTRODUCTION

About the project , we will create a sixty counter . Sixty counter is a counter and when we run it , it counts up to sixty .

Block diagram of the Project looks like this .

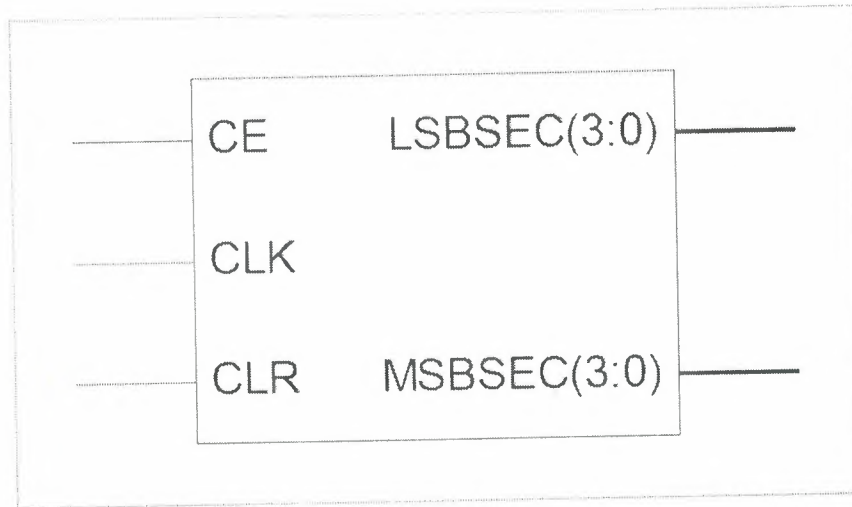


Figure1. block diagram

In the first chapter block diagram of the Project will be given and inputs and outputs from the block diagram will be defined . After that information about which language we used to create the project and what we used in software and hardware will be given . Then we will explain the Xilinx ISE 9.1 . In this chapter lastly we will show the design flow .

In the second chapter we will follow the steps which we gave at the end of chapter one . In the first step we will enter the design , the name of the project and directory of Project will be given by us . We will select a project device properties . Second step is creating a VHDL source . After creating a source we generate the codes . Then we check the generated codes from check syntax , to be sure if we have mistakes or not . Then we create a test bench . After creating test bench we get simulation to see the output of the project . In this chapter lastly we will give an information about the project .

In chapter three an information will be given by us about the spartan . We will give an overview and capabilities and advantages of spartan .

In chapter four we will talk about xilinx . We will explain the mean of xilinx , and what the name of xilinx represents . In business what xilinx represents .

As you see last two chapter is general information about the xilinx and spartan . But the first two chapter is about the project .

CHAPTER one DESIGN

1.1)DESIGN DECRPTION

The design is sixty counter . We are going to do a Sixty counter which counts up to sixty .

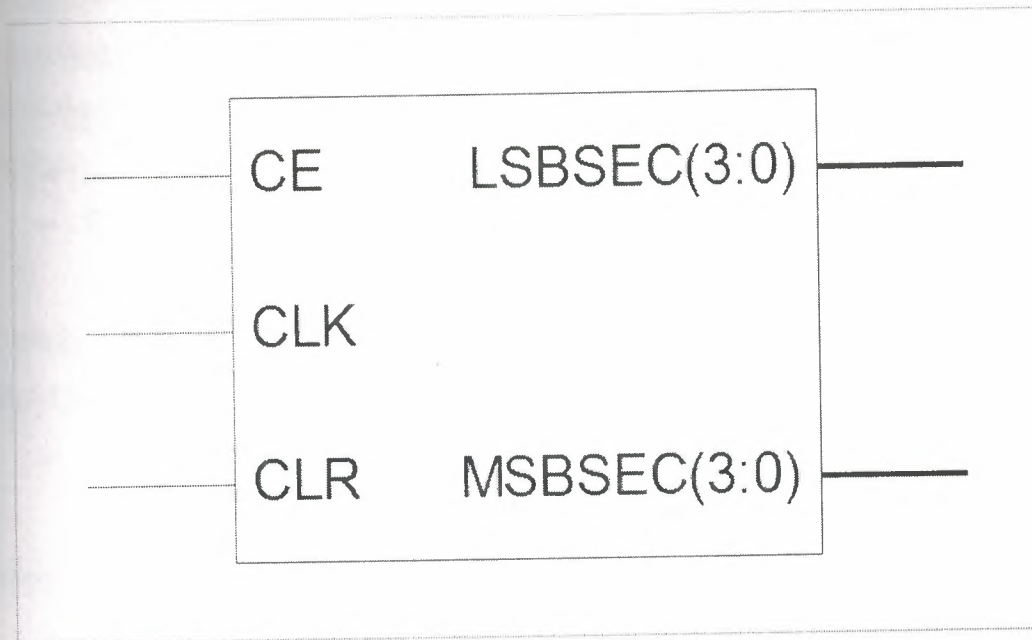


Figure2. Sixty counter block diagram

Inputs :

CLK

CLR

CE

The counter counts the clock cycles . If the clock cycle is 1second conter will count up to 1minute . If the clock cycle is 1minute counter will count up to 60minute . Clear is clears all the values . Clock enable is enables counter to count .

Outputs :

LSBSEC

MSBSEC

Lsbsec is counts from 0 to 9 . Msbsec is counts from 0-5 .

Used things : To create the project VHDL programming language will be used and there will be a some hardware and software requirements . We will use in software Xilinx ISE and hardware Spartan3E .

The briefly explanation of Xilinx ISE : The Integrated Software Environment (ISE™) is the Xilinx® design software suite that allows that , take the design from design entry through Xilinx device programming. The ISE Project Navigator manages and processes the designs through the following steps in the ISE design flow .

- **DESIGN ENTRY**

Design entry is the first step in the ISE design flow. During design entry, we can create our source files based on design objectives. We can create your top-level design file using a Hardware Description Language (HDL), such as VHDL, Verilog, or ABEL, or using a schematic. We can use multiple formats for the lower-level source files in the design.

- **SYNTHESIS**

After design entry and optional simulation, we run synthesis. During this step, VHDL, Verilog, or mixed language designs become netlist files that are accepted as input to the implementation step.

- **CREATE TEST BENCH**

The results of the works during all the program is displayed in this process . Also in this process some values is given to get the outputs .

- **IMPLEMENTATION**

After synthesis, we run design implementation, which converts the logical design into a physical file format that can be downloaded to the selected target device. From Project Navigator, we can run the implementation process in one step, or we can run each of the implementation processes separately. Implementation processes vary depending on whether we are targeting a Field Programmable Gate Array (FPGA) or a Complex Programmable Logic Device (CPLD).

- **VERIFICATION**

We can verify the functionality of our design at several points in the design flow. We can use simulator software to verify the functionality and timing of our design or a portion of our design. The simulator interprets VHDL or Verilog code into circuit functionality and displays logical results of the described HDL to determine correct circuit operation. Simulation allows us to create and verify complex functions in a relatively small amount of time. We can also run in-circuit verification after programming our device.

- **DEVICE CONFIGURATION**

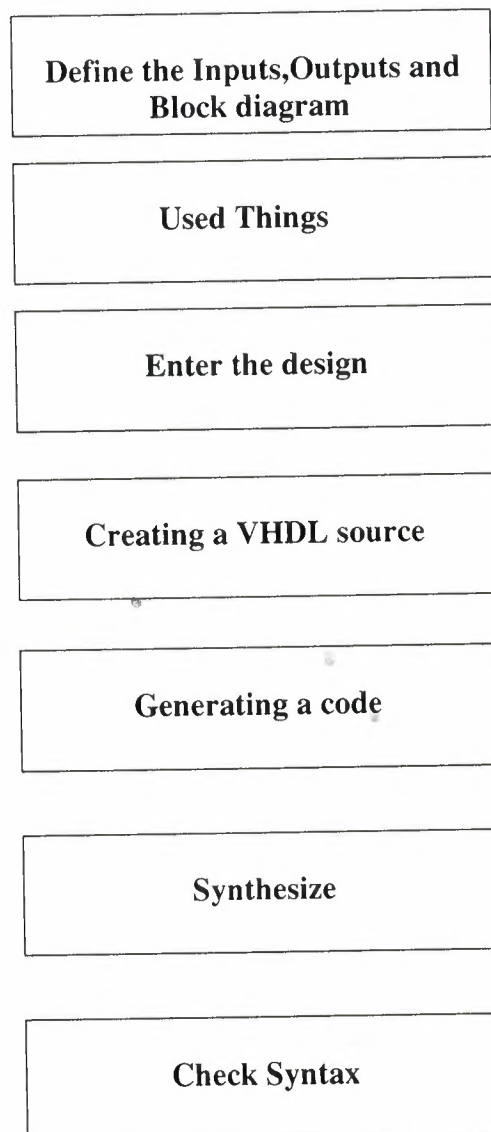
After generating a programming file, we configure our device. During configuration, we generate configuration files and download the programming files from a host computer to a Xilinx device.

1.2)ABOUT THE DESIGN

In this project we we have created a count sixty counter . This counter will count up to sixty . To create the counter we need two counter and the counters will work together . First counter will count from 0-9 and when the first counter comes nine it will add one to the second counter . By this way first counter will count the six times and the counter will reach the sixty .

Sixty counter can be used in digital watch .

The design flow lookslike this :



Creating a test bench

Simulating

Explaining project

Figure3. design flow

CHAPTER two DESIGN STEPS

2.1) ENTER THE DESIGN

In the first step we will create the Project basically . To do this firstly we must setup the XILINX ISE 9.1I . After the installation of the program we open the program and then we have some steps to create the program as follows :

1. Select File > New Project... The New Project Wizard appears.
2. Type SIXTY in the Project Name field.
3. Enter or browse to a location (directory path) for the new project. The works subdirectory is created automatically.

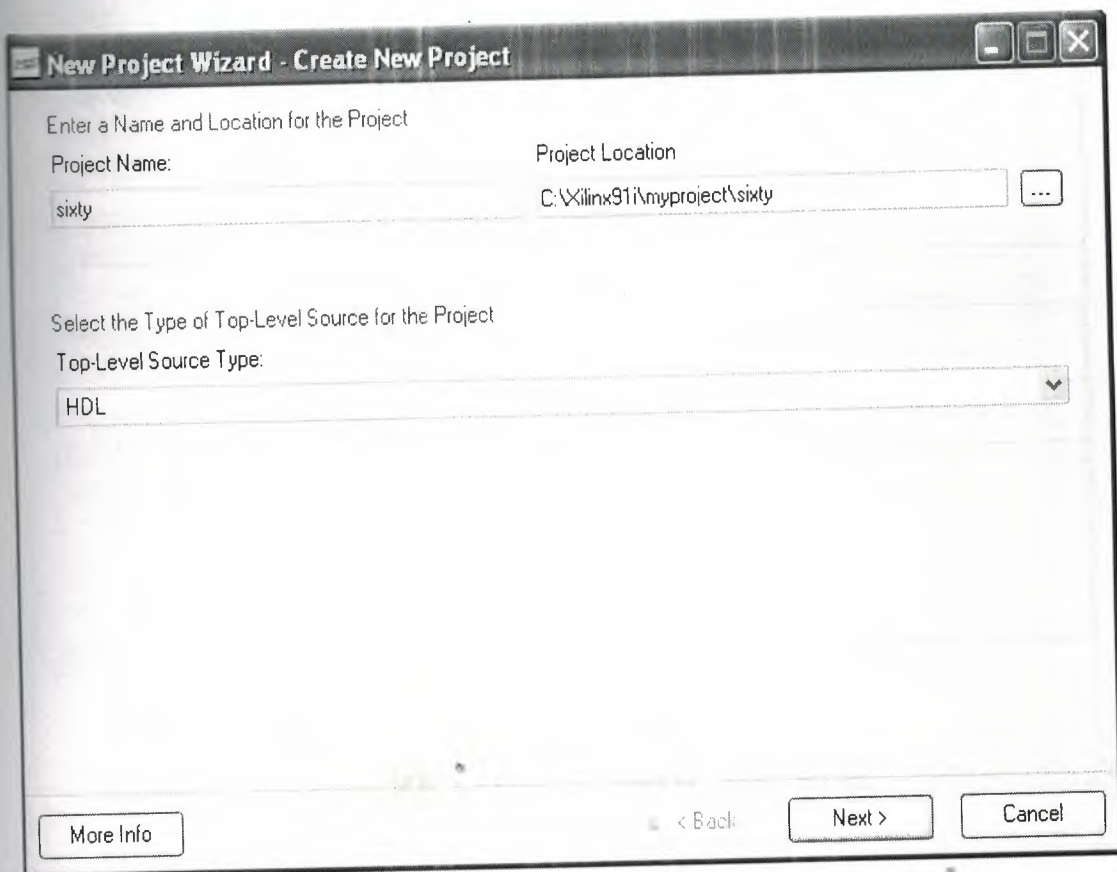


Figure4.Project name

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: Spartan3E

Device: XC3S100E

Package: VQ100

Speed Grade: -5

Top-Level Source Type: HDL

Synthesis Tool: XST (VHDL/Verilog)

Simulator: ISE Simulator (VHDL/Verilog)

Preferred Language: VHDL

Verify that Enable Enhanced Design Summary is selected.

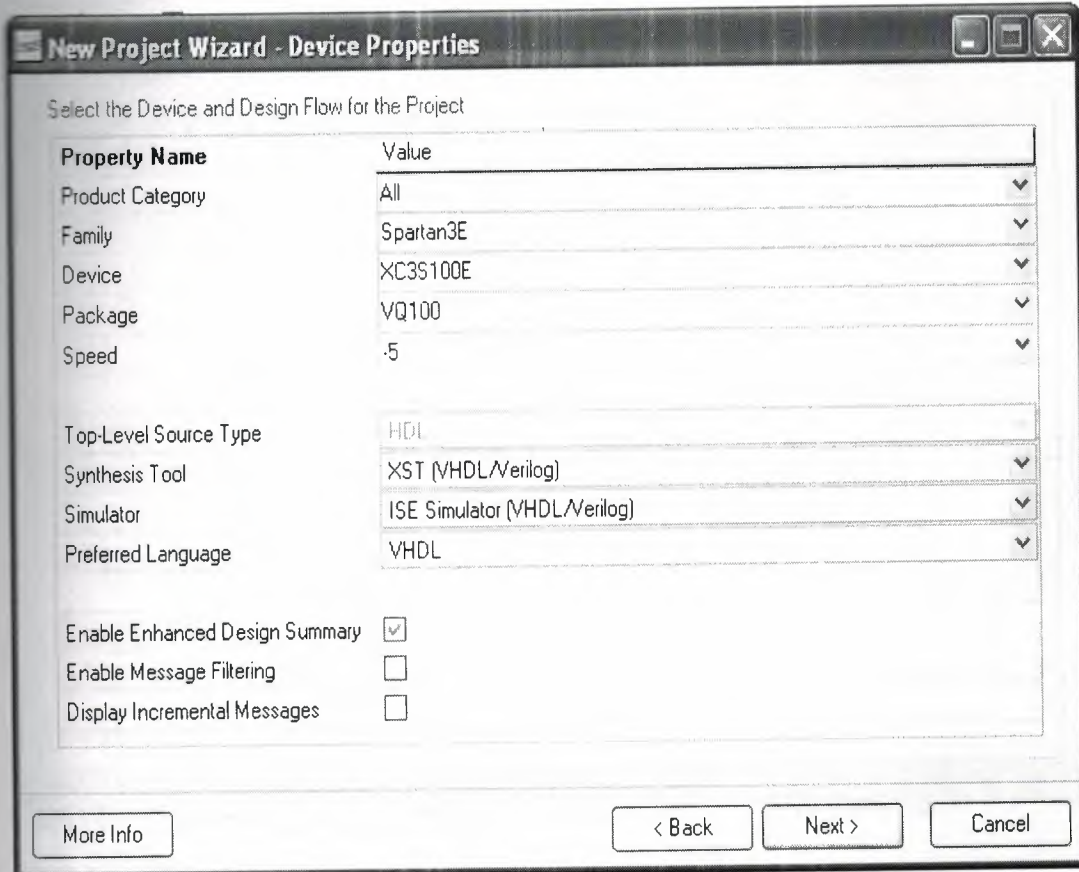


Figure5. Project Device Properties

Leave the default values in the remaining fields.

2.2) CREATING a VHDL SOURCE

Create a VHDL source file for the project as follows:

1. Click the New Source button in the New Project Wizard.
2. Select VHDL Module as the source type.
3. Type in the file name CNT60.

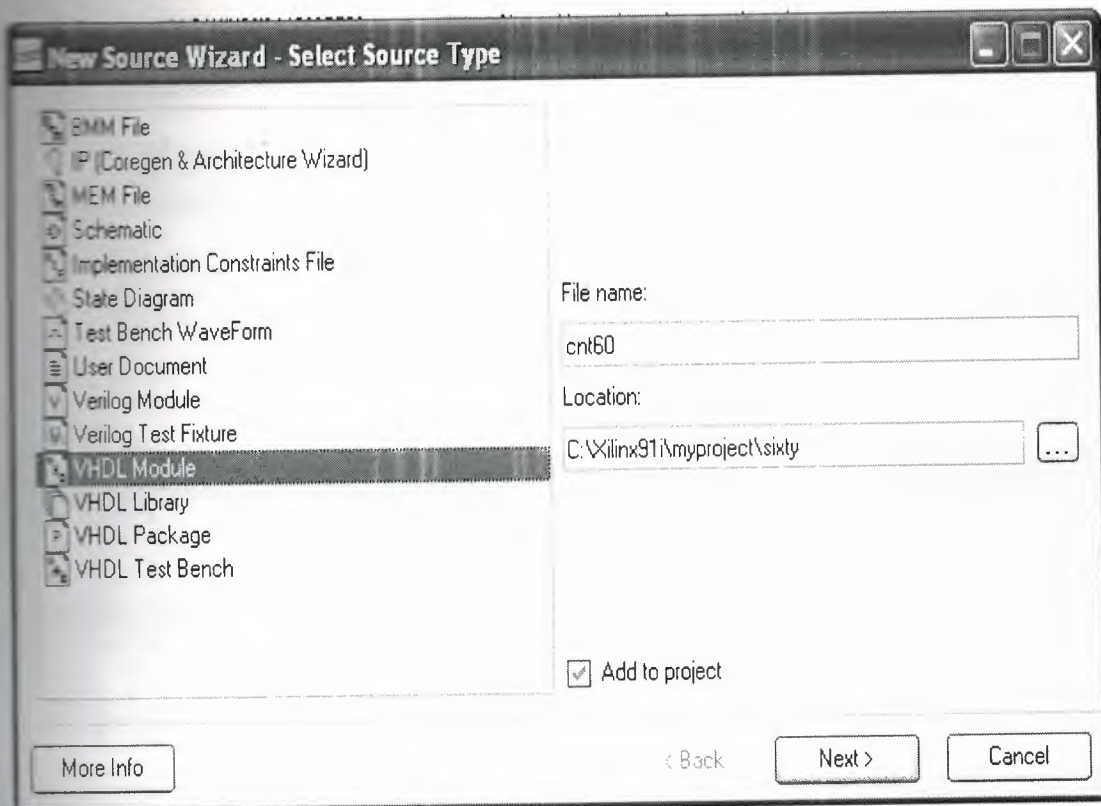


Figure6. Adding source

4. Verify that the Add to project checkbox is selected.
5. Click Next.
6. Declare the ports for the counter design by filling in the port information as shown below:

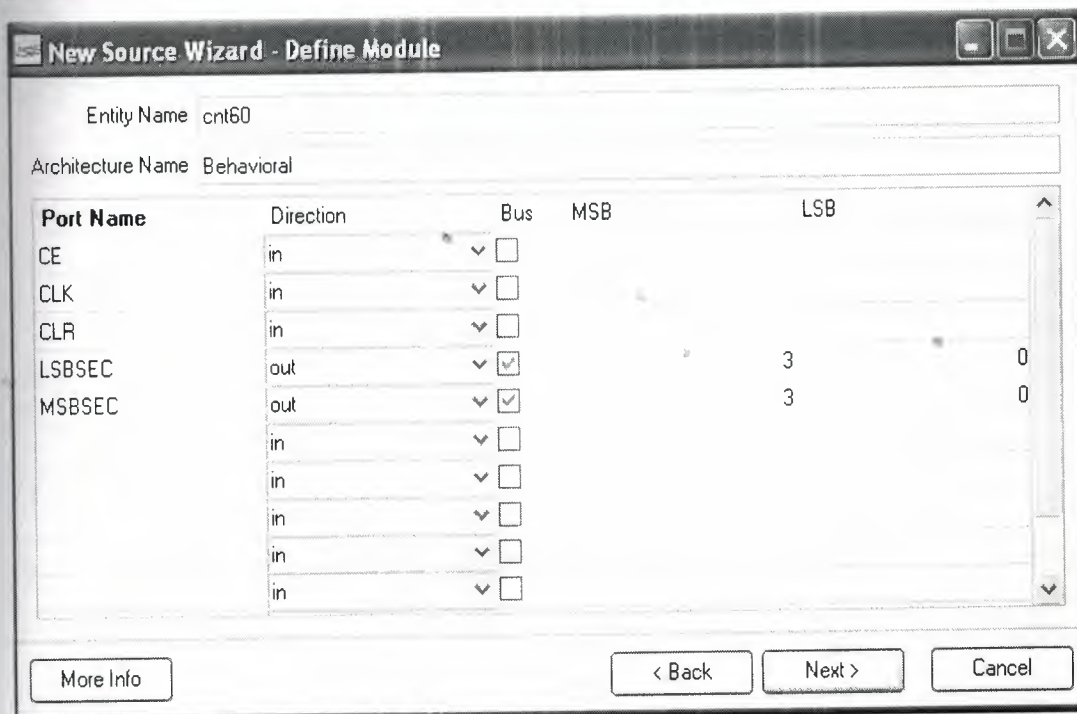


Figure7. Define Module

7. Click Next, then Finish in the New Source Wizard - Summary dialog box to complete the new source file template.

8. Click Next, then Next, then Finish.

The source file containing the entity/architecture pair displays in the Workspace, and the counter displays in the Source tab, as shown below:



Figure8.Program

2.3) GENERATING a CODE

In next step we add some necessary signals , process and some codes to the program . After that the program will be as shown :

```
-- Company:
-- Engineer:
--
-- Create Date: 12:38:22 05/08/2008
-- Design Name:
-- Module Name: cnt60 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity cnt60 is
```

```

Port ( CE : in STD_LOGIC;
      CLK : in STD_LOGIC;
      CLR : in STD_LOGIC;
      LSBSEC : out STD_LOGIC_VECTOR (3 downto 0);
      MSBSEC : out STD_LOGIC_VECTOR (3 downto 0));
end cnt60;
architecture Behavioral of cnt60 is
component scntnr
port ( CE : in STD_LOGIC ;
      CLK : in STD_LOGIC ;
      CLR : in STD_LOGIC ;
      QOUT : out STD_LOGIC_VECTOR ( 3 downto 0 ));
end component;
signal lsbout : STD_LOGIC_VECTOR (3 downto 0) ;
signal msbout : STD_LOGIC_VECTOR (3 downto 0) ;
signal msbce : STD_LOGIC ;
signal lsbtc : STD_LOGIC ;
signal msbclr : STD_LOGIC ;
signal msbtc : STD_LOGIC ;
begin
lsbcount : scntnr port map (CE => CE , CLK => CLK , CLR => CLR , QOUT =>
lsbout);
msbcount : scntnr port map (CE => msbce ,CLK => CLK,CLR=> msbclr,
QOUT=>msbout);
process (lsbout)
begin
if (lsbout="1001") then
lsbtc<='1';
else
lsbtc<='0';
end if;
end process;
process (msbout)
begin

```

```

if (msbout="0110")then
msbtc<='1';
%
msbtc<='0';
end if ;
end process;

msbce<=CE and lsbtc;
msbclr <= CLR or msbtc;
LSBSEC <= lsbout ;
MSBSEC <= msbout ;
end Behavioral;

```

Now one more process must be created . Because we have two counter in this program so second counter we are adding a other source as following :

1. Click the New Source button in the New Project Wizard.
2. Select VHDL Module as the source type.
3. Type in the file name SCNTR.

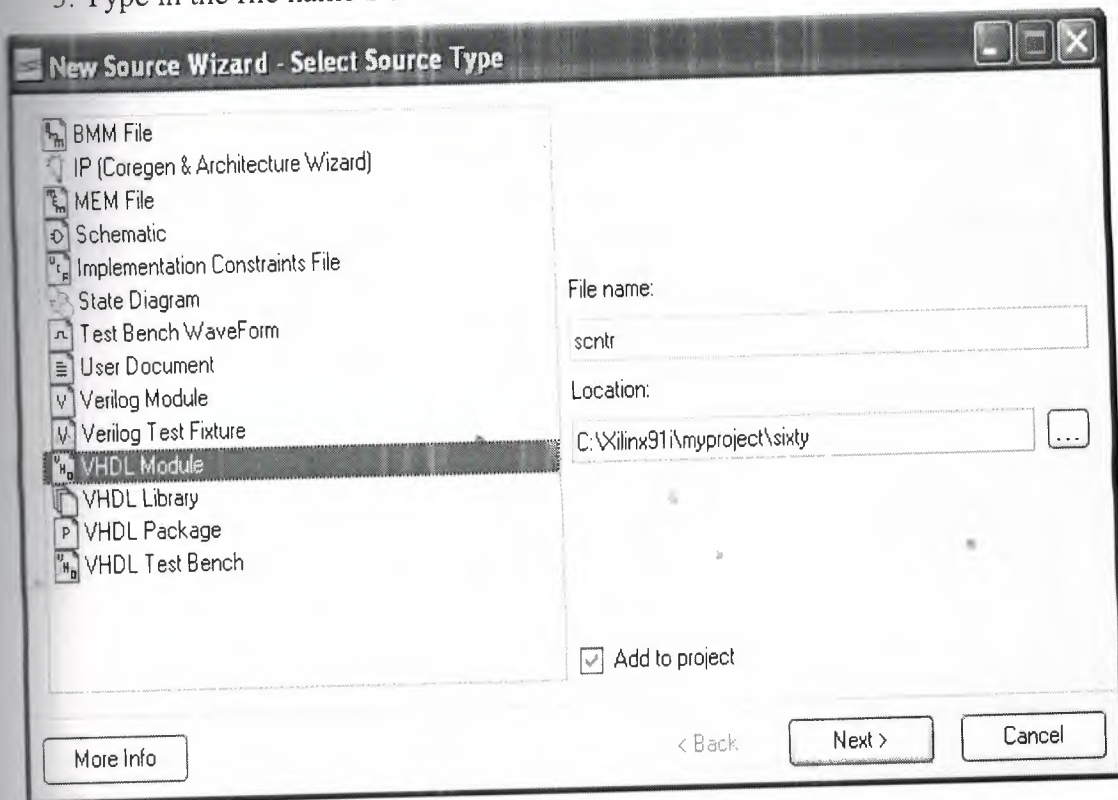


Figure9.Adding source

4. Verify that the Add to project checkbox is selected.
5. Click Next.

6. Declare the ports for the counter design by filling in the port information as shown below:

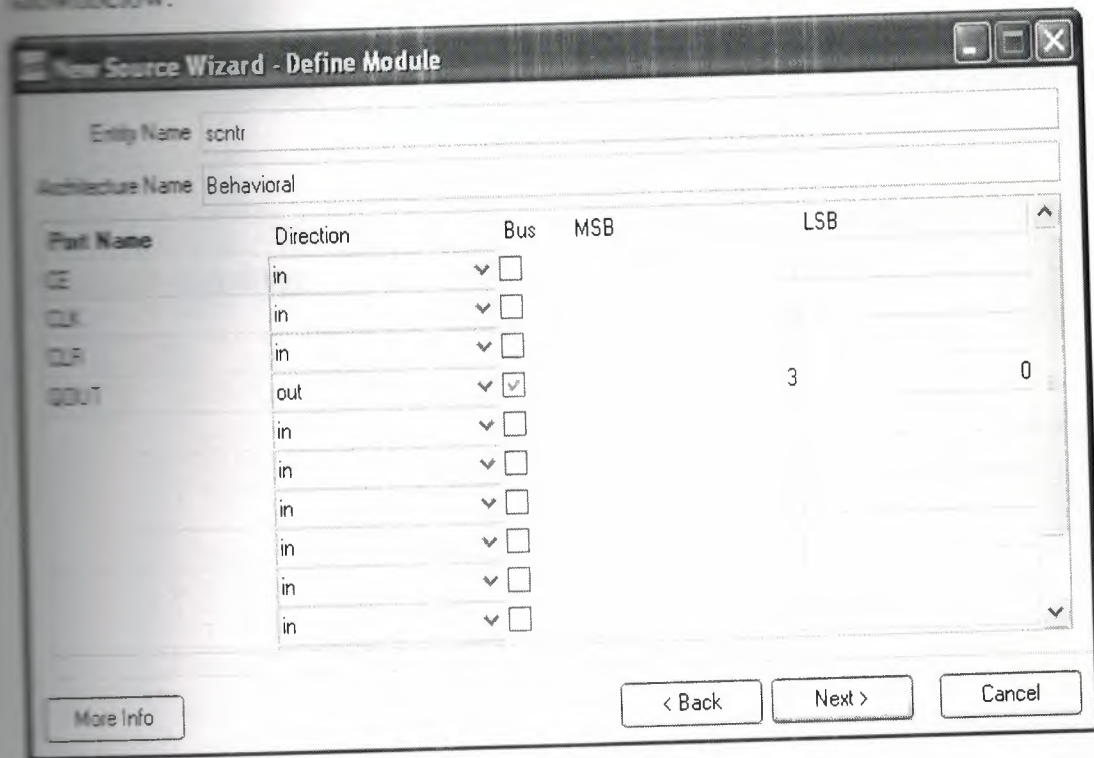


Figure10. Define Module

7. Click Next, then Finish in the New Source Wizard - Summary dialog box to complete the new source file template.

8. Click Next, then Next, then Finish.

The source file containing the entity/architecture pair displays in the Workspace, and the counter displays in the Source tab, as shown below:

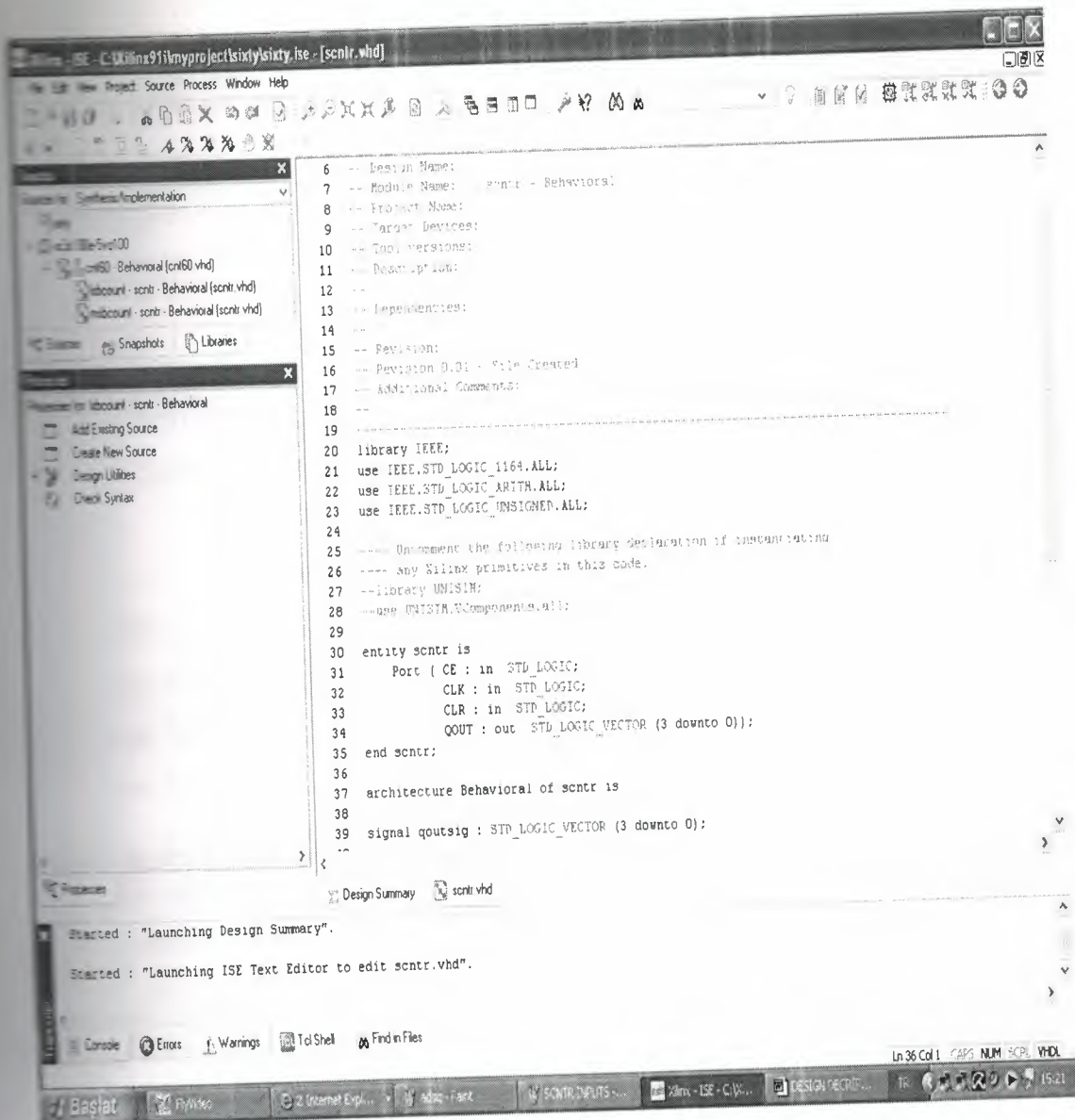


Figure 11. Program

In next step we add some necessary signals , process and some codes to the program . After that the program will be as shown :

```

-----
-- Company:
-- Engineer:
--
-- Create Date: 13:01:54 05/08/2008
-- Design Name:
-- Module Name: scntr - Behavioral
-- Project Name:

```

```
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
```


```
-----
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity scnr is
    Port ( CE : in STD_LOGIC;
          CLK : in STD_LOGIC;
          CLR : in STD_LOGIC;
          QOUT : out STD_LOGIC_VECTOR (3 downto 0));
end scnr;
architecture Behavioral of scnr is
    signal qoutsig : STD_LOGIC_VECTOR (3 downto 0);
    begin
    process (CE,CLK,CLR)
        begin
        if (CLR='1') then
            qoutsig<="0000";
        elsif (CE='1')then
            if (CLK'event and CLK = '1') then
```

```

if (qoutsig="1001") then
qoutsig<="0000";
else
qoutsig<= qoutsig+"0001";
end if;
end if;
end process;
QOUT<=qoutsig;
end Behavioral;

```

2.4)SYNTHESIZE

After design entry and optional simulation, we run synthesis. In the Sources tab, select Synthesis/Implementation from the Design View drop-down list, and select the top module . In the Processes tab, double-click Synthesize.

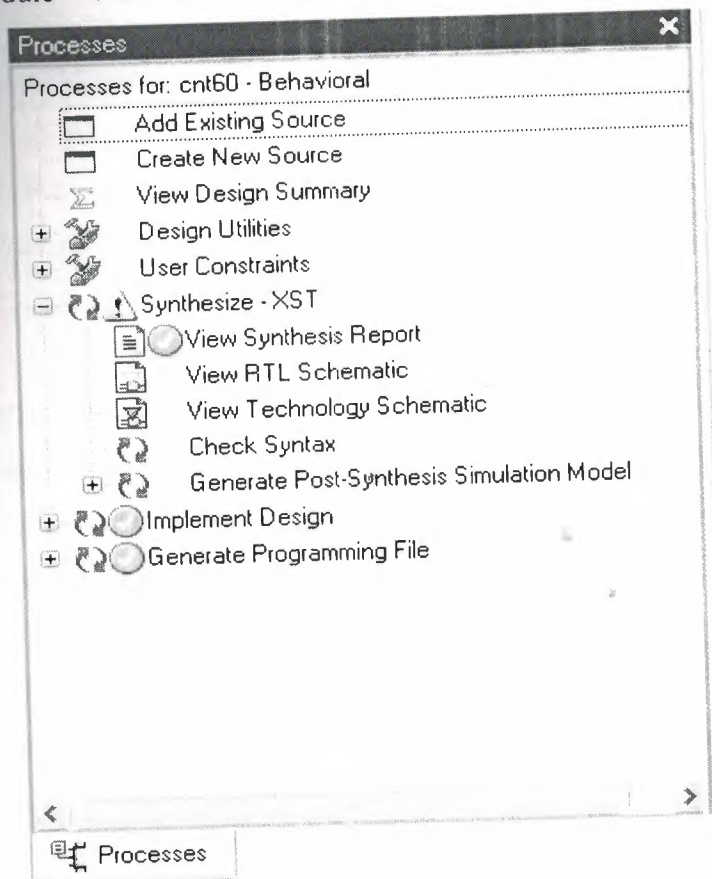


Figure12.Select synthesize

As you can see from the figure.12 we can get the synthesis report , we can look RTL schematic of the our design , we can look the technology schematic of the design and we can check the syntax to be sure that the if we have mistakes or not .

Synthesis Report : This report contains the results from the synthesis run, including area and timing estimation.

Release 9.1i - xst J.30

Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.

-> Parameter TMPDIR set to ./xst/projnav.tmp

CPU : 0.00 / 0.36 s | Elapsed : 0.00 / 0.00 s

-> Parameter xsthdpdir set to ./xst

CPU : 0.00 / 0.36 s | Elapsed : 0.00 / 0.00 s

-> Reading design: cnt60.prj

TABLE OF CONTENTS

- 1) Synthesis Options Summary
- 2) HDL Compilation
- 3) Design Hierarchy Analysis
- 4) HDL Analysis
- 5) HDL Synthesis
 - 5.1) HDL Synthesis Report
- 6) Advanced HDL Synthesis
 - 6.1) Advanced HDL Synthesis Report
- 7) Low Level Synthesis
- 8) Partition Report
- 9) Final Report
 - 9.1) Device utilization summary
 - 9.2) Partition Resource Summary
 - 9.3) TIMING REPORT

=====

=====

*

Synthesis Options Summary

*

=====

=====

---- Source Parameters

Input File Name : "cnt60.prj"

Input Format : mixed
Ignore Synthesis Constraint File : NO

--- Target Parameters

Output File Name : "cnt60"
Output Format : NGC
Target Device : xc3s100e-5-vq100

--- Source Options

Top Module Name : cnt60
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) : 24
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 : cnt60.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 "C:/Xilinx91i/myproject/sixty/sctr.vhd" in Library work.

Architecture behavioral of Entity sctr is up to date.

Compiling vhdl file "C:/Xilinx91i/myproject/sixty/cnt60.vhd" in Library work.

Entity <cnt60> compiled.

Entity <cnt60> (Architecture <behavioral>) compiled.

=====

=====

* Design Hierarchy Analysis *

=====

=====

Analyzing hierarchy for entity <cnt60> in library <work> (architecture <behavioral>).

Analyzing hierarchy for entity <sctr> in library <work> (architecture <behavioral>).

=====

=====

* HDL Analysis *

=====

=====

Analyzing Entity <cnt60> in library <work> (Architecture <behavioral>).

Entity <cnt60> analyzed. Unit <cnt60> generated.

Analyzing Entity <sctr> in library <work> (Architecture <behavioral>).

Entity <sctr> analyzed. Unit <sctr> generated.

=====

=====

* HDL Synthesis *

=====

=====

Performing bidirectional port resolution...

Synthesizing Unit <scntr>.

Related source file is "C:/Xilinx91i/myproject/sixty/scntr.vhd".

Found 4-bit up counter for signal <qoutsig>.

Summary:

inferred 1 Counter(s).

Unit <scntr> synthesized.

Synthesizing Unit <cnt60>.

Related source file is "C:/Xilinx91i/myproject/sixty/cnt60.vhd".

Unit <cnt60> synthesized.

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

Macro Statistics

# Counters	: 2
4-bit up counter	: 2

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

=====
=====
Loading device for application Rf_Device from file '3s100e.nph' in environment
C:/Xilinx91i.

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

Macro Statistics

Counters : 2
4-bit up counter : 2

=====
=====

=====
=====

* Low Level Synthesis *

=====
=====

Optimizing unit <cnt60> ...

Mapping all equations...

Building and optimizing final netlist ...

Found area constraint ratio of 100 (+ 5) on block cnt60, actual ratio is 0.

Final Macro Processing ...

=====
=====

Final Register Report

Macro Statistics

Registers : 8
Flip-Flops : 8

=====
=====

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

=====
=====
Partition Implementation Status

No Partitions were found in this design.

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

=====
=====
Final Results

RTL Top Level Output File Name : cnt60.ngr

Top Level Output File Name : cnt60

Output Format : NGC

Optimization Goal : Speed

Keep Hierarchy : NO

Design Statistics

IOs : 11

Cell Usage :

BELS : 13

GND : 1

INV : 2

LUT3 : 2

LUT4 : 6

MUXF5 : 2

FlipFlops/Latches : 8

FDCE : 8

Clock Buffers : 1

BUFGP : 1
IO Buffers : 10
IBUF : 2
OBUF : 8

Device utilization summary:

Selected Device : 3s100evq100-5

Number of Slices:	7 out of	960	0%
Number of Slice Flip Flops:	8 out of	1920	0%
Number of 4 input LUTs:	10 out of	1920	0%
Number of IOs:	11		
Number of bonded IOBs:	11 out of	66	16%
Number of GCLKs:	1 out of	24	4%

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	8

Asynchronous Control Signals Information:

Control Signal	Buffer(FF name)	Load
msbclr(msbclr_f5:O)	NONE(msbcount/qoutsig_2)	4
CLR	IBUF	4

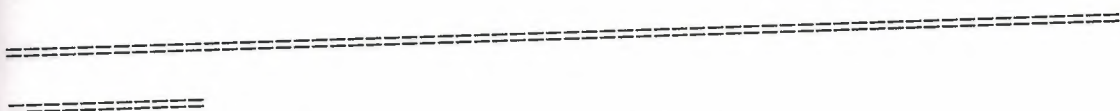
Timing Summary:

Speed Grade: -5

- Minimum period: 3.076ns (Maximum Frequency: 325.098MHz)
- Minimum input arrival time before clock: 3.519ns
- Maximum output required time after clock: 4.252ns
- 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: 3.076ns (frequency: 325.098MHz)

Total number of paths / destination ports: 40 / 12

Delay: 3.076ns (Levels of Logic = 2)

Source: lsbcount/qoutsig_2 (FF)

Destination: msbcount/qoutsig_0 (FF)

Source Clock: CLK rising

Destination Clock: CLK rising

Data Path: lsbcount/qoutsig_2 to msbcount/qoutsig_0

	Gate	Net			
Cell:in->out	fanout	Delay	Delay	Logical Name (Net Name)	
FDCE:C->Q	5	0.514	0.690	lsbcount/qoutsig_2 (lsbcount/qoutsig_2)	
LUT4:I0->O	1	0.612	0.000	msbce1 (N22)	
MUXF5:I1->O	4	0.278	0.499	msbce_f5 (msbce)	
FDCE:CE		0.483		msbcount/qoutsig_0	
Total		3.076ns (1.887ns logic, 1.189ns route)			
		(61.3% logic, 38.7% route)			

Timing constraint: Default OFFSET IN BEFORE for Clock 'CLK'

Total number of paths / destination ports: 8 / 8

Offset: 3.519ns (Levels of Logic = 3)

Source: CE (PAD)

Destination: msbcount/qoutsig_0 (FF)

Destination Clock: CLK rising

Data Path: CE to msbcount/qoutsig_0

	Gate	Net			
Cell:in->out	fanout	Delay	Delay	Logical Name (Net Name)	

IBUF:I->O	5	1.106	0.541	CE_IBUF (CE_IBUF)
LUT4:I3->O	1	0.612	0.000	msbce1 (N22)
MUXF5:I1->O	4	0.278	0.499	msbce_f5 (msbce)
FDCE:CE		0.483		msbcount/qoutsig_0

Total		3.519ns	(2.479ns logic, 1.040ns route)
			(70.4% logic, 29.6% route)

=====
 =====

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

Total number of paths / destination ports: 8 / 8

Offset: 4.252ns (Levels of Logic = 1)

Source: msbcount/qoutsig_0 (FF)

Destination: MSBSEC<0> (PAD)

Source Clock: CLK rising

Data Path: msbcount/qoutsig_0 to MSBSEC<0>

Cell:in->out	Gate	Net	fanout	Delay	Delay	Logical Name (Net Name)
FDCE:C->Q	6	0.514	0.569	msbcount/qoutsig_0	(msbcount/qoutsig_0)	
OBUF:I->O	3.169			MSBSEC_0_OBUF	(MSBSEC<0>)	

Total		4.252ns	(3.683ns logic, 0.569ns route)
			(86.6% logic, 13.4% route)

=====
 =====

CPU : 8.75 / 9.20 s | Elapsed : 9.00 / 9.00 s

-->

Total memory usage is 146204 kilobytes

Number of errors : 0 (0 filtered)

Number of warnings : 0 (0 filtered)

Number of infos : 0 (0 filtered)

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

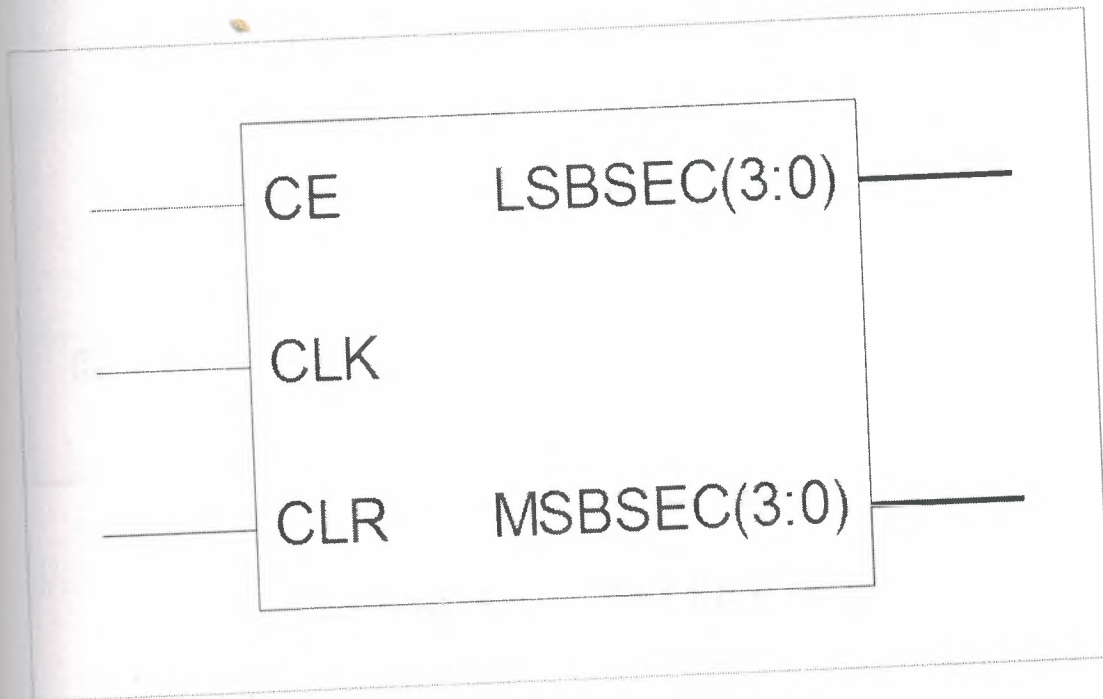


Figure13. Block diagram

Detailed block diagram.

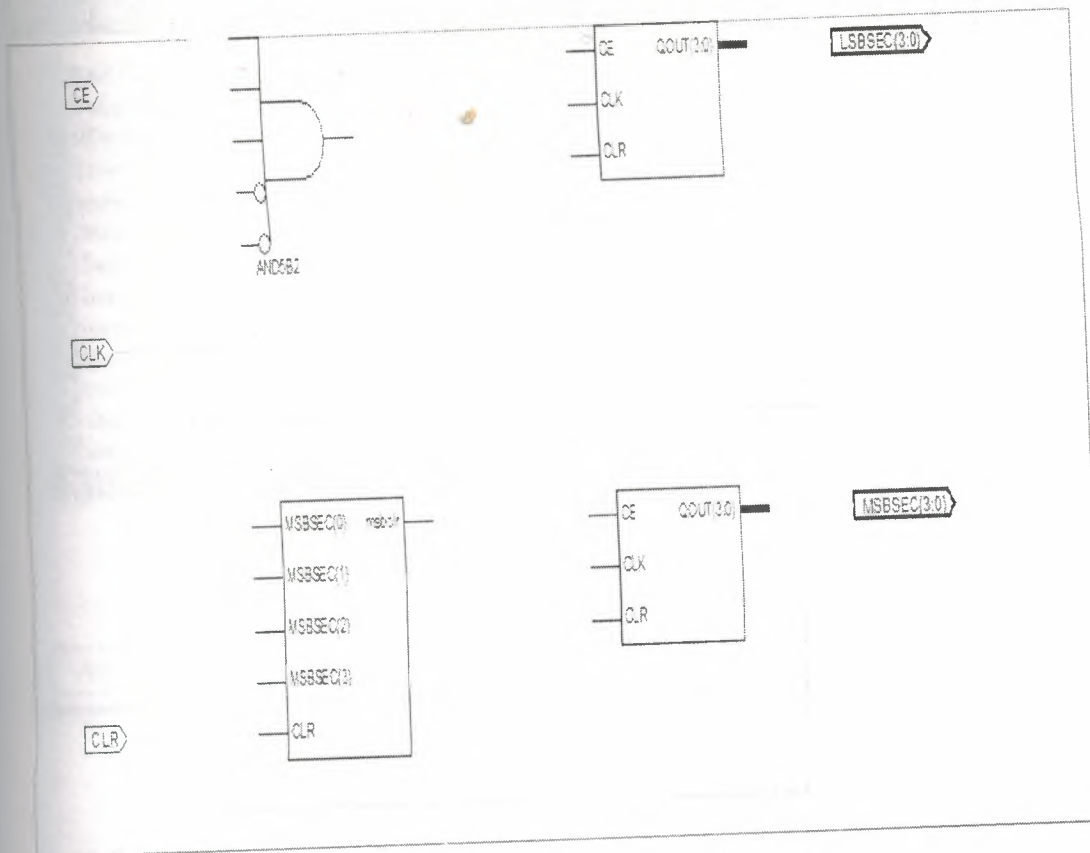


Figure14. Detailed block diagram

2.5) WRITING a TEST BENCH

To simulate the design, we need both the design under test (DUT) or unit under test (UUT) and the stimulus provided by the test bench. A test bench is HDL code that allows us to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing.

Create a test bench containing input stimulus we can use to verify the functionality of the CNT60 module.

1. Select the CNT60 HDL file in the Sources window.
2. Create a new test bench source by selecting Project → New Source.
3. In the New Source Wizard, select Test Bench as the source type, and type cnt60_tb in the File Name field.

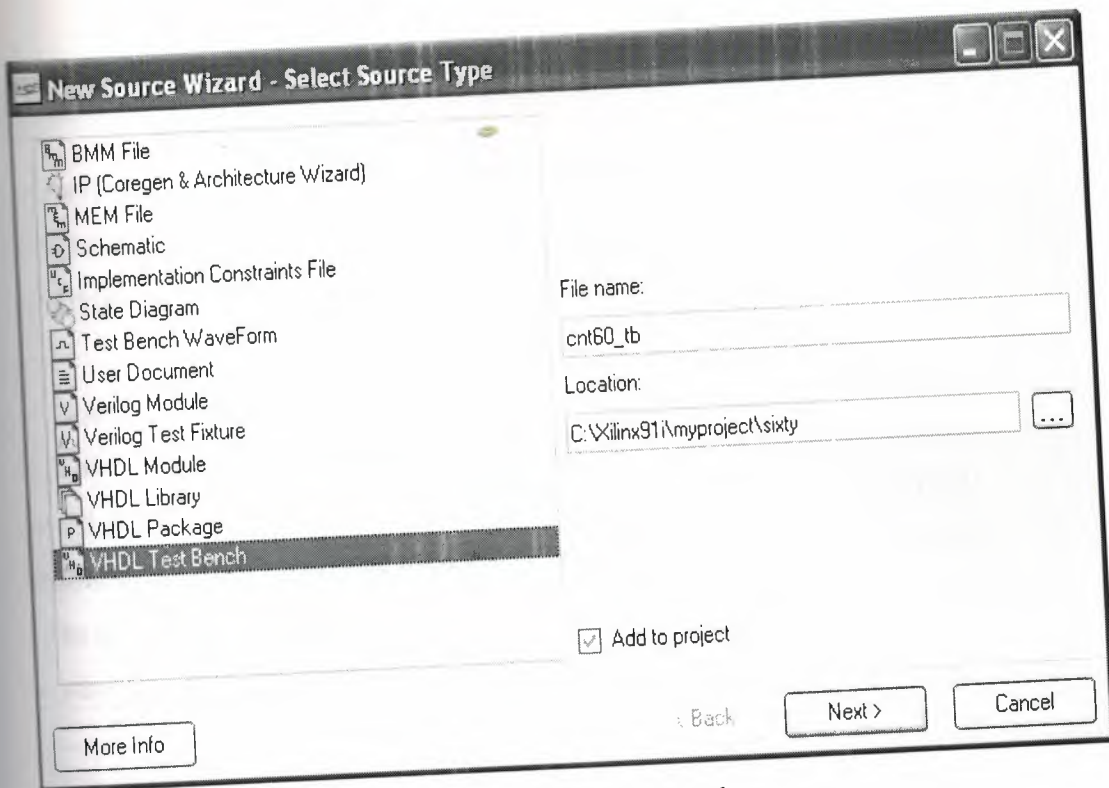


Figure15. Test bench

4. Click Next.
5. The Summary page shows that the source will be added to the project, and it displays the source directory, type and name. Click Finish.
6. We need some values to get output so we must add the following to the process :

```

-----
-- Company:
-- Engineer:
--
-- Create Date: 12:59:31 05/08/2008
-- Design Name: cnt60
-- Module Name: C:/Xilinx91i/myproject/sixty/sixty_tb.vhd
-- Project Name: sixty
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: cnt60
--

```

```
-- 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 sixty_tb_vhd IS
END sixty_tb_vhd;
ARCHITECTURE behavior OF sixty_tb_vhd IS
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT cnt60
    PORT(
        CE : IN std_logic;
        CLK : IN std_logic;
        CLR : IN std_logic;
        LSBSEC : OUT std_logic_vector(3 downto 0);
        MSBSEC : OUT std_logic_vector(3 downto 0)
    );
END COMPONENT;
--Inputs
SIGNAL CE : std_logic := '0';
SIGNAL CLK : std_logic := '0';
SIGNAL CLR : std_logic := '0';
```

```

--Outputs
SIGNAL LSBSEC : std_logic_vector(3 downto 0);
SIGNAL MSBSEC : std_logic_vector(3 downto 0);
BEGIN
  -- Instantiate the Unit Under Test (UUT)
  uut: cnt60 PORT MAP(
    CE => CE,
    CLK => CLK,
    CLR => CLR,
    LSBSEC => LSBSEC,
    MSBSEC => MSBSEC
  );
  clk_pr: process
BEGIN
  CLK<='0';
  WAIT FOR 10 nS;
  CLK<='1';
  WAIT FOR 10 nS;
  END PROCESS;
  tb : PROCESS
  BEGIN
  CLR<='1';
  -- Wait 100 ns for global reset to finish
    wait for 100 ns;
  CLR<='0';
  CE<='1';
  -- Place stimulus here
  wait; -- will wait forever
  END PROCESS;
END;

```

6. Click Finish to complete the timing initialization .

2.6)SIMULATING

During HDL simulation, the simulator software verifies the functionality and timing of the design or portion of the design. The simulator interprets VHDL or Verilog code into circuit functionality and displays logical results of the described HDL to determine correct circuit operation. Simulation allows you to create and verify complex functions in a relatively small amount of time.

We have created a test bench to the count sixty and it looks like this :

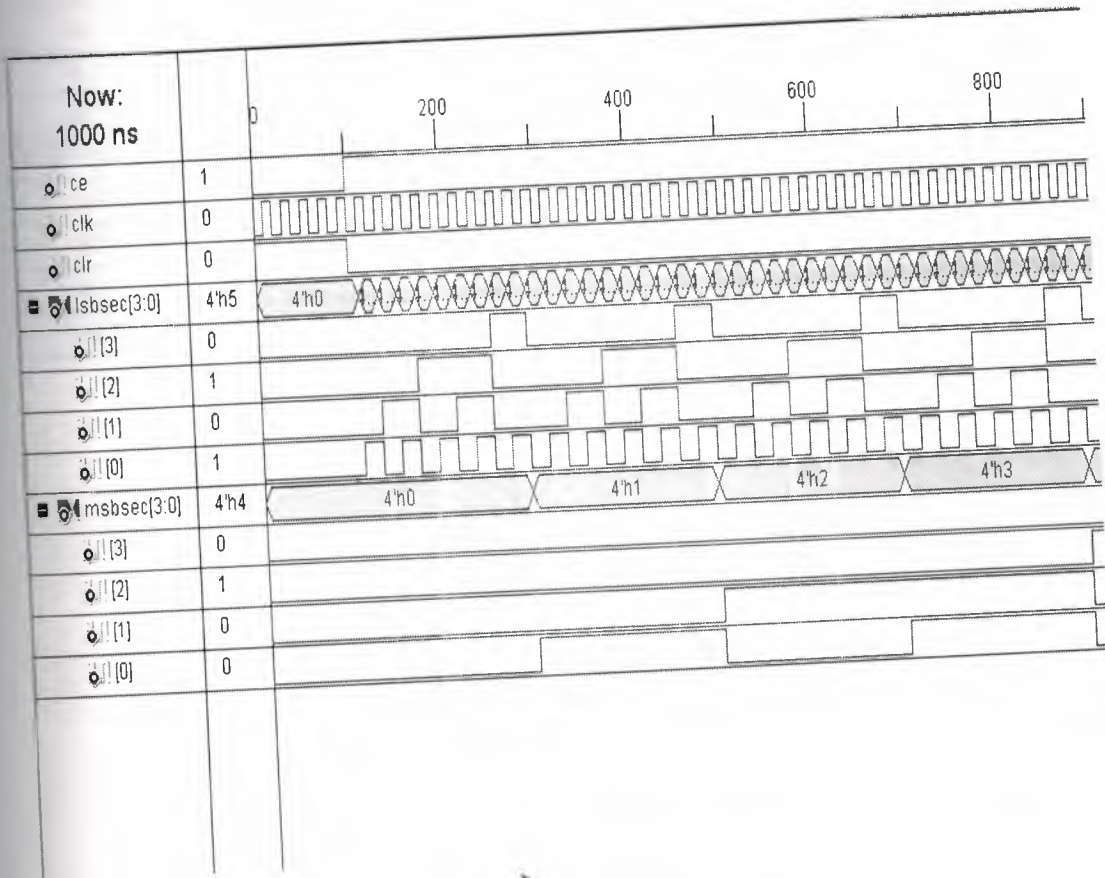


Figure16.Simulation

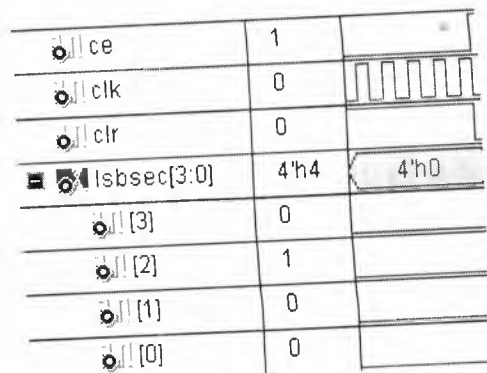


Figure17.Reset

This figure17 shows the reset part of Project . It resets the program for 100ns .

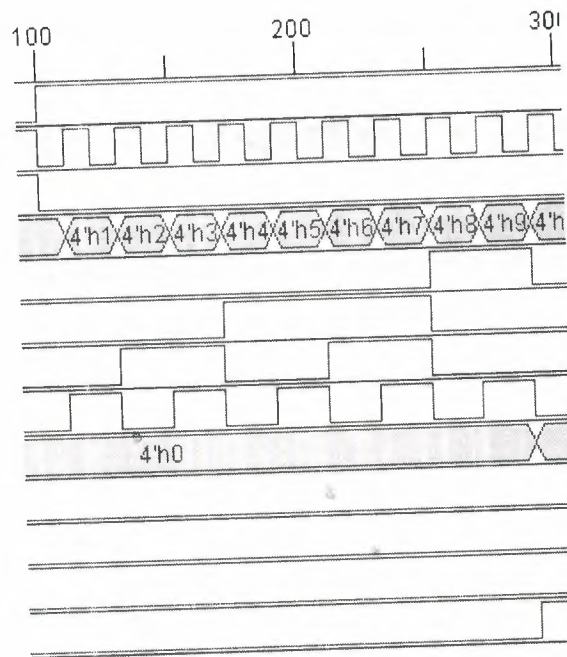
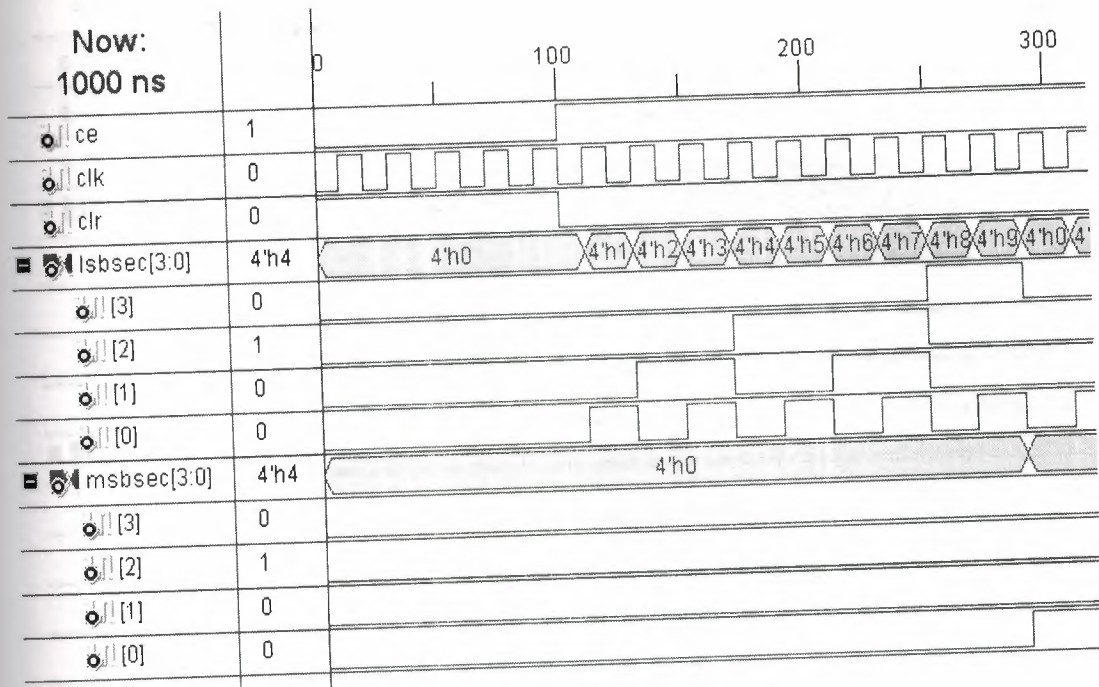


Figure18.Counting process

Here it shows the after reset part how program starts the counting . It starts counting after end of reset , when lsbsec comes 9 it adds to msbsec 1 as shown in the figures .

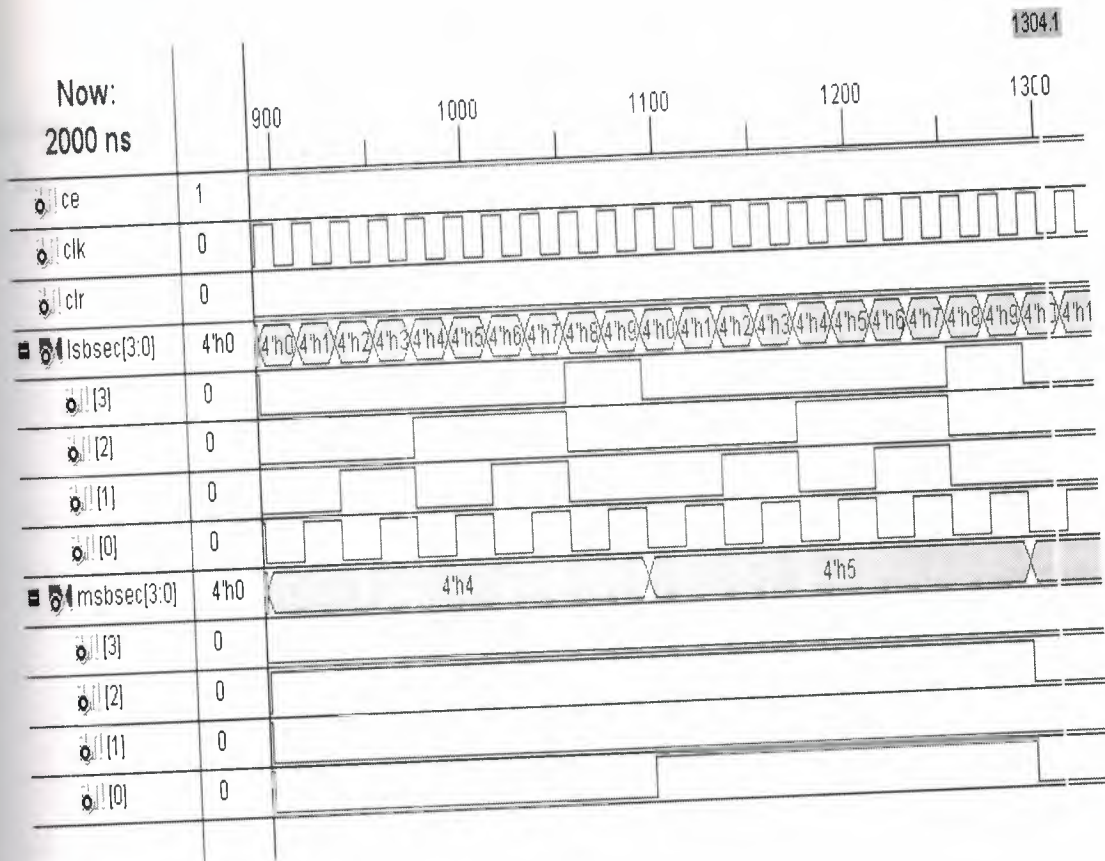


Figure19.Return zero

We can see in this figure when msbsec becomes 6 program starting the count from 0 again

2.7)EXPLANATION OF MY SIXTY COUNTER PROGRAM CODES

Now we finished the works . So now when we run the the program is counting sixty times . If we try to explain the codes that we used in the sources :

We have entered the input and output values while we were adding a source and they look like this :

entity cnt60 is

Port (CE : in STD_LOGIC;

CLK : in STD_LOGIC;

CLR : in STD_LOGIC;

LSBSEC : out STD_LOGIC_VECTOR (3 downto 0);

MSBSEC : out STD_LOGIC_VECTOR (3 downto 0));

End cnt60;

These inputs and outputs are general input and outputs of the counter .We must declare the components of other source inputs and outputs because we have two counter in the program .

```
component scntn
port ( CE : in STD_LOGIC ;
      CLK : in STD_LOGIC ;
      CLR : in STD_LOGIC ;
      QOUT : out STD_LOGIC_VECTOR ( 3 downto 0 ));
end component;
```

This shows the relation of the inputs and outputs , with the other source .

Now we declared some signals . These signals are necessary to not get the error

```
signal lsbout : STD_LOGIC_VECTOR (3 downto 0) ;
signal msbout : STD_LOGIC_VECTOR (3 downto 0) ;
signal msbce : STD_LOGIC ;
signal lsbtc : STD_LOGIC ;
signal msbclr : STD_LOGIC ;
signal msbtc : STD_LOGIC ;
```

Now we showed the port map of lsbcnt and msbcnt . We declared the port map because program will continue to process by looking these maps .

```
lsbcnt : scntn port map (CE => CE , CLK => CLK , CLR => CLR , QOUT =>
lsbout);
msbcnt : scntn port map (CE => msbce ,CLK => CLK,CLR=> msbclr,
QOUT=>msbout);
```

Now we declared the processes of lsbout and msbout .

```
process (lsbout)
begin
    if (lsbout="1001") then
        lsbtc<='1';
    else
        lsbtc<='0';
    end if;
end process;
process (msbout)
```

```

begin
    if (msbout="0110")then
msbtc<='1';
else
msbtc<='0';
end if ;
end process;

```

Now we passed to second source which is scntnr . We have entered the input and output values of scntnr while we were adding a source and they look like this entity scntnr is

```

Port ( CE : in STD_LOGIC;
      CLK : in STD_LOGIC;
      CLR : in STD_LOGIC;
      QOUT : out STD_LOGIC_VECTOR (3 downto 0));
end scntnr;

```

We will get an output signal to this place so I declared the 4bit output signal like this :

```

signal qoutsig : STD_LOGIC_VECTOR (3 downto 0);

```

Now we are defining a process

```

process (CE,CLK,CLR)

```

```

begin
    if (CLR='1') then
        qoutsig<="0000";
    elsif (CE='1')then
        if (CLK'event and CLK ='1') then
            if (qoutsig="1001") then
                qoutsig<="0000";
            else
                qoutsig<= qoutsig+"0001";
            end if;
        end if;
    end if;
end process;
QOUT<=qoutsig;

```


Here these codes wants to tell us the if CLR is not zero and if CE is one and if CLK event and CLK is one and if qoutsig is "1001" change qoutsig to "0000" if it's not add qoutsig+"0001" . By the other way clear musn't be 0 , clock enable must be 1 , so if output signal is 9 return 0 to count again . If output signal is not 9 add output signal to 1 .

After that we generate the test bench . In the test bench we entered only the input values to get the output and we declared the process like this .

```
clk_pr: process
BEGIN
  CLK<='0';
  WAIT FOR 10 nS;
  CLK<='1';
  WAIT FOR 10 nS;
END PROCESS;

tb : PROCESS
  BEGIN
    CLR<='1';
    -- Wait 100 ns for global reset to finish
    wait for 100 ns;

    CLR<='0';
    CE<='1';

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

CHAPTER three SPARTAN 3

3.1) OVERVIEW

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

See the complete Spartan-3 Generation product table .

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**

Figure 20. Spartan-3 Generation product table

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).

3.2) CAPABILITIES

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

3.2.1) DUAL POWER MANAGEMENT

The integrated power management (PDF) 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

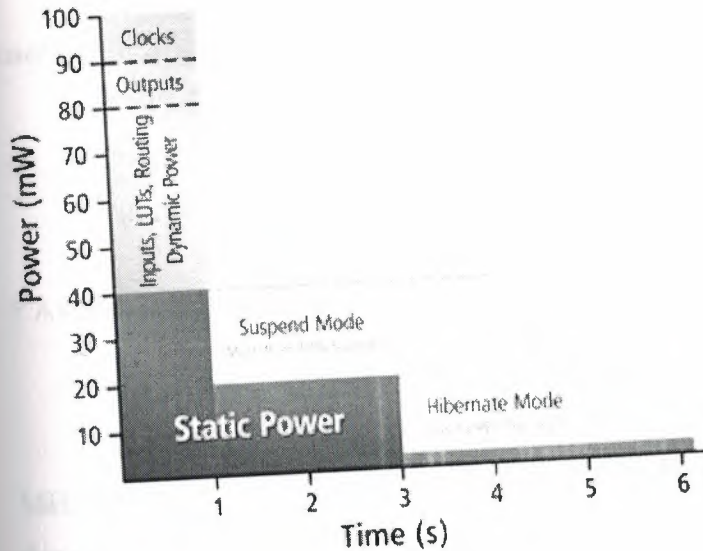


Figure 21. Hibernate

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.

3.2.2) 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.

3.2.3) 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.

3.2.4) XtremeDSP DSP 48A SLICE

The Spartan-3A DSP platform's optimized DSP48A slice (PDF) 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/subtractor 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.

3.2.5) 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

3.2.6) FOUR LEVEL MEMORY ARCHITECTURE

Four level memory architecture (PDF) 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

Figure 22. Memory requirements

3.2.7) 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

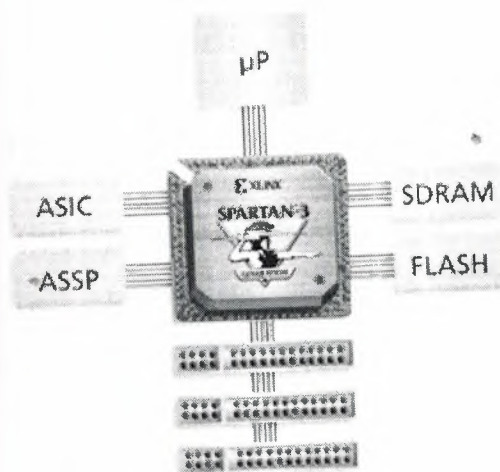


Figure 23. Spartan-3 generation FPGAs support multiple platform standards.

3.2.8) CONFIGURABLE LOGIC BLOCKS

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

3.2.9) 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

3.2.10) 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, bitstream compression, design revision 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

3.3)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 only Low-cost Security

Unique Device DNA serial number (57-bits)
helps prevent design cloning, unauthorized
overbuilding and reverse engineering

Secure mechanism to deliver IP

Customizable algorithms for security
as well as responses to failures



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



The Industry's Most Complete Design Solution for Optimal Results

ISE® Foundation™ Software

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

ISE WebPACK™ Software

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

ISE Classics software

A free collection of previously released ISE software tools

CHAPTER four HISTORY

4.1) 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."

4.2) WHAT DOES 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.

Read about the origin of Xilinx and the Xilinx name
Timeline of Significant Events in Xilinx History

1984

Ross Freeman, Bernie Vonderschmitt, and Jim Barnett found Xilinx. The company's business and management mission and philosophy are created, a new kind of company is born. The concept for a new type of product, the Field Programmable Gate Array, takes shape. Read more about how Xilinx began.

1985

Xilinx introduces its first product, the XC2064™.
It's the first-ever FPGA, a radical new form of programmable logic.

1987

Sales office established in Weybridge, England.
This is the first sales office outside North America, targeted to serve the European market.

1988

The company opens its first overseas office in Tokyo.
Xilinx K.K. is born. Initial focus is to serve Seiko, our first wafer supplier/partner.

1989

Xilinx founder, Ross Freeman, passes away.
His dream for the team and the technology lives on.

1990

Xilinx goes public at \$10 per share, after reaching several quarters of profitability.
Shares are \$0.83 when adjusted for splits. This is a major milestone along the path to realizing the company vision.

1991

The XC4000™ family of FPGAs is introduced.
This is the first broadly adopted FPGA and will become the primary Xilinx revenue driver for the 90's.

1993

Xilinx Scotland is established in Edinburgh.
This team's focus on IP solutions core and software development brings Xilinx a considerable competitive advantage.

1995

Xilinx Ireland officially opens in Dublin.
This is our first major site in Europe, establishing manufacturing and engineering capability outside the U.S.

Xilinx Colorado is established in Boulder.
Boulder employees significantly increase software development capability in the company. This is the first major North American site outside of Silicon Valley. It has since moved to Longmont, Colorado.

1996

Wim Roelandts joins as CEO and President.
He brings 30 years of Hewlett-Packard experience to his new assignment.



1997

CREATIVE Values dialogue is created throughout the company.

All employees participate in a process to articulate the values of Xilinx. These are: customer focus, respect, excellence, accountability, teamwork, integrity, very open communications and enjoying our work. The first letter of each value forms the acronym: CREATIVE.

1998

Virtex®™ FPGA family is introduced.

This is a major step in FPGA architecture and opens up new markets for the company. The Virtex family becomes the primary revenue driver to date.

1999

Xilinx Albuquerque opens with acquisition of CoolRunner™ team and technology. This product line offers new low power and lower cost CPLD products to customers.

2000

Xilinx revenue exceeds \$1B.

The company reaches the billion-dollar milestone only 10 years after going public. Employees take out a full-page ad saying "Thanks a Billion for Your Leadership" as a tribute to CEO Wim Roelandts at a pivotal point for the company.

2003

Spartan®™-3 family of products is introduced.

This very low-cost product is the world's first 90nm FPGA. The Spartan-3 technology puts us considerably ahead of our competition and in the company of premier advanced semiconductor manufacturers.

2003

Wim Roelandts becomes Chairman of the Board.

Bernie Vonderschmitt retires; the last company founder leaves an impressive legacy.

2004

Xilinx celebrates its 20th anniversary.

The company observes its first 20 years of life by honoring employees, customers, shareholders, partners, and local communities.

2008

Moshe Gavrielov is named President and CEO.

Wim Roelandts remains Chairman of the Board.

4.3)HOW XILINX BEGAN ?

4.3.1)NEV TECHNOLOGY

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.

Ross Freeman, Xilinx co-founder, invented the "field programmable gate array" (FPGA), a new form of programmable logic.	Bernie Vonderschmitt, Xilinx co-founder, pioneered the revolutionary concept of a fabless semiconductor.
---	--

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.

4.3.2) 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.

4.3.3) 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.

4.4) BUSINESS

4.4.1) LEARN ABOUT THE XILINX TECHNOLOGY, HOW and WHY TO PURPOSE 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.

4.4.2) PROGRAMMABLE LOGIC IS XILINX'S 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.

4.4.3)USES FOR PROGRAMMABLE LOGIC

Xilinx's 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.

4.4.4)MULTIPLE PRODUCT LINES WITH SUPERLATIVE SOFTWARE SUPPORT

Xilinx's 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.

4.4.5)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.

4.4.6) XILINX'S 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.

4.5) GETTING STARTED WITH FPGAs

4.5.1) WHAT ARE FPGAs?

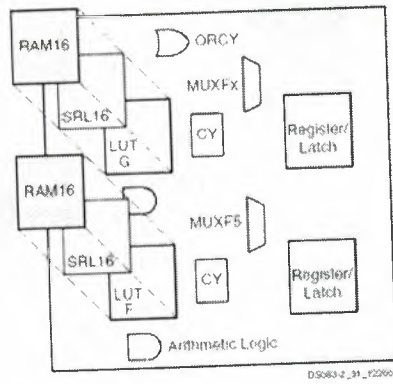
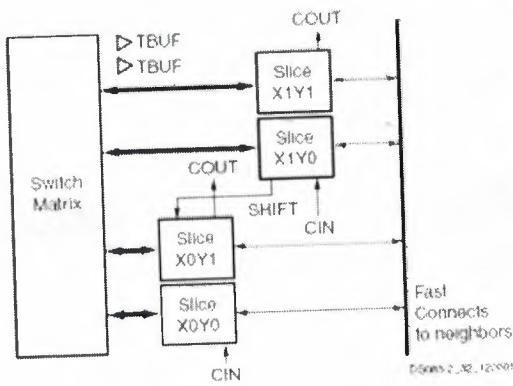
Field Programmable Gate Arrays (FPGAs) are programmable semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. As opposed to Application Specific Integrated Circuits (ASICs) where the device is custom built for the particular design, FPGAs can be programmed to the desired application or functionality requirements.

Although one-time programmable (OTP) FPGAs are available, the dominant type are SRAM based which can be reprogrammed as the design evolves.

Roll cursor over blue highlighted sections of the figure below to see more details.

CLB Details

The Configurable Logic Block is the basic logic unit in an FPGA. Exact numbers and features vary from device to device, but every CLB consists of a configurable switch matrix with 4 or 6 inputs, some selection circuitry (MUX, etc.), and flip-flops. The switch matrix is highly flexible and can be configured to handle combinatorial logic, shift registers or RAM. A high level CLB overview is shown here. More architectural details can be found in the applicable device's data sheet.

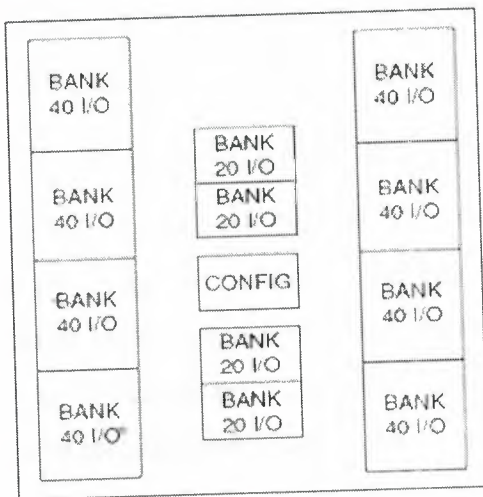


DCM Details

Digital clock management is provided by most FPGAs in the industry (all Xilinx FPGAs have this feature), and has nearly eliminated the skew and other issues that designers had to face with in designing global signals into FPGAs in the past.

IOB Details

Today's FPGAs provide support for dozens of I/O standards thus providing the ideal interface bridge in your system. I/O in FPGAs is grouped in banks (see figure below) with each bank independently able to support different I/O standards. Today's leading FPGAs provide over a dozen I/O banks, thus allowing flexibility in I/O support.



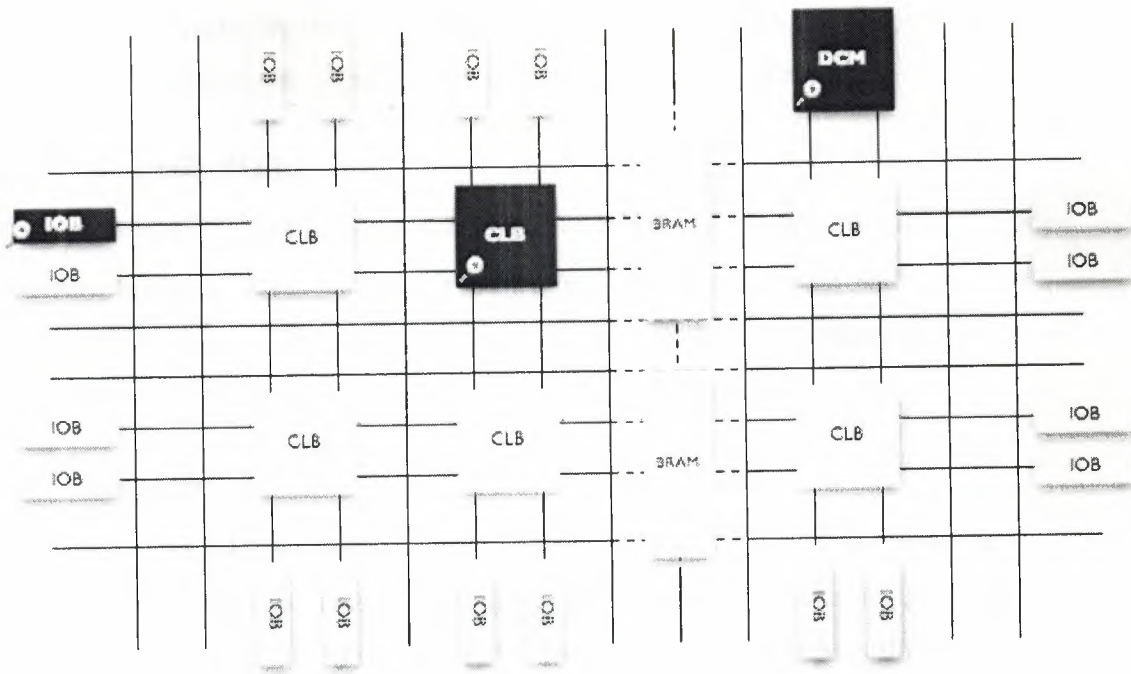


Figure 24.FPGA Block Structure

4.5.2)COMMON FPGA FEATURES

Today's FPGAs have evolved far beyond the basic capabilities present in their predecessors, and incorporate hard (ASIC type) blocks of commonly used functionality such as RAM, clock management, and DSP. Following are the basic components in an FPGA.

Configurable Logic Block (CLBs)

The CLB is the basic logic unit in an FPGA. Exact numbers and features vary from device to device, but every CLB consists of a configurable switch matrix with 4 or 6 inputs, some selection circuitry (MUX, etc), and flip-flops. The switch matrix is highly flexible and can be configured to handle combinatorial logic, shift registers, or RAM. More architectural details can be found in the applicable device's data sheet.

Interconnect

While the CLB provides the logic capability, flexible interconnect routing routes the signals between CLBs and to and from I/Os. Routing comes in several flavors, from that designed to interconnect between CLBs to fast horizontal and vertical long lines spanning the device to global low-skew routing for Clocking and other global signals.

The design software makes the interconnect routing task hidden to the user unless specified otherwise, thus significantly reducing design complexity.

SelectIO (IOBs)

Today's FPGAs provide support for dozens of I/O standards thus providing the ideal interface bridge in your system. I/O in FPGAs is grouped in banks with each bank independently able to support different I/O standards. Today's leading FPGAs provide over a dozen I/O banks, thus allowing flexibility in I/O support.

Memory

Embedded Block RAM memory is available in most FPGAs, which allows for on-chip memory in your design. These allow for on-chip memory for your design. Xilinx FPGAs provide up to 10 Mbits of on-chip memory in 36 kbit blocks that can support true dual-port operation.

Complete Clock Management

Digital clock management is provided by most FPGAs in the industry (all Xilinx FPGAs have this feature). The most advanced FPGAs from Xilinx offer both digital clock management and phase-looped locking that provide precision clock synthesis combined with jitter reduction and filtering.

4.5.3)FPGA SOLUTIONS, APPLICATIONS and END-MARKETS

Due to their programmable nature, FPGAs are an ideal fit for many different markets. As the industry leader, Xilinx provides comprehensive solutions consisting of FPGA devices, advanced software, and configurable, ready-to-use IP cores for markets and applications such as

End Markets

Aerospace & Defense

Radiation-tolerant FPGAs along with intellectual property for image processing, waveform generation, and partial reconfiguration for SDRs.

Automotive

Automotive silicon and IP solutions for gateway and driver assistance systems, comfort, convenience, and in-vehicle infotainment.

Broadcast

Solutions enabling a vast array of broadcast chain tasks as video and audio finds its way from the studio to production and transmission and then to the consumer.

Consumer

Cost-effective solutions enabling next generation, full-featured consumer applications, such as converged handsets, digital flat panel displays, information appliances, home networking, and residential set top boxes.

Industrial/Scientific/Medical

Industry-compliant solutions addressing market-specific needs and challenges in industrial automation, motor control, and high-end medical imaging.

Storage & Server

Data processing solutions for Network Attached Storage (NAS), Storage Area Network (SAN), servers, storage appliances, and more.

Wireless Communications

RF, base band, connectivity, transport and networking solutions for wireless equipment, addressing standards such as WCDMA, HSDPA, WiMAX and others.

Wired Communications

End-to-end solutions for the Reprogrammable Networking Linecard Packet Processing, Framers/MAC, serial backplanes, and more

Technology Solutions

DSP

The Xilinx XtremeDSP™ initiative helps you develop tailored high performance DSP solutions for aerospace and defense, digital communications, multimedia, video, and imaging industries.

Embedded Processing

Xilinx delivers an innovative and flexible range of processing solutions for your unique embedded applications.

4.6)XILINX's SUCCEES

4.6.1)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).

4.6.2)XILINX's 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.

4.6.3)XILINX's 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.

4.6.4)XILINX's EMPLOYEES

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

4.7)XILINX's VALUES

4.7.1)HOW XILINX WORK WITH ONE ANOTHER AND XILINX's PARTNERS,WHAT DO VALUES MEAN TO XILINX?

Xilinx's values have helped set the character of their 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.

Their values also provide the backdrop for the dialogue they have with colleagues. They help them make business decisions. They provide the framework for interacting with each other. What's especially impressive about their 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 them - and they accept change as the reality of today's high-tech industry - it's important to know that the values they depend upon are constant and unchanging. They are, in a very real sense, their permanent anchors.

4.7.2)HOW DID XILINX CLEARLY DEFINE IT's VALUES ?

The set of values xilinx believe in started with xilinx's 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, xilinx company started a grassroots process to articulate xilinx's values. They looked very carefully at their business and made certain the values were connected with what would move them forward and foster their growth in the marketplace. Most importantly, they wanted to capture in words what they 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.

4.7.3)HOW DO XILINX KEEP IT's VALUES VISIBLE and VIABLE IN THE COMPANY ?

At Xilinx, they believe that making decisions based on their values translates directly to the bottom line, helps them be more successful in their business, and brings them closer to realizing their 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 they "enforce" the values? They don't. They leave it up to each individual to act in accordance with them. The values are there to direct their actions, to guide them professionally and personally, and to inspire them in the worst and the best of times.

CONCLUSION

Finally I can say my project which named sixty counter is working correctly .

When I simulate the project lsbsec is counting from 0-9 and when it comes to 9 , lsbsec is adding 1 to msbsec . This process is counting until msbsec becomes 6 .

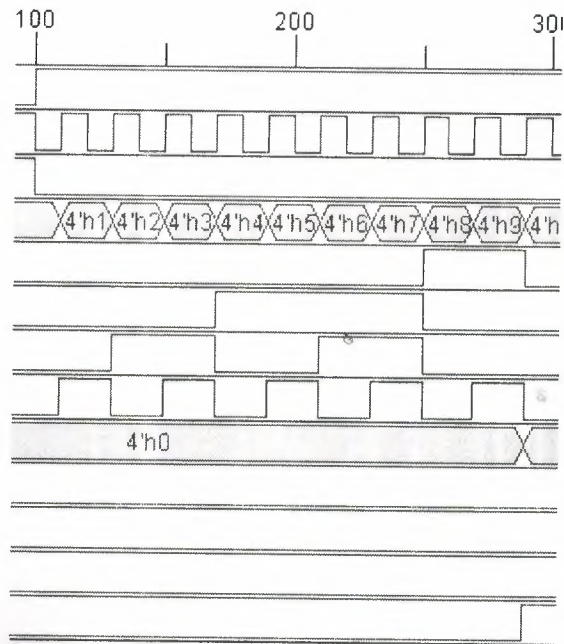
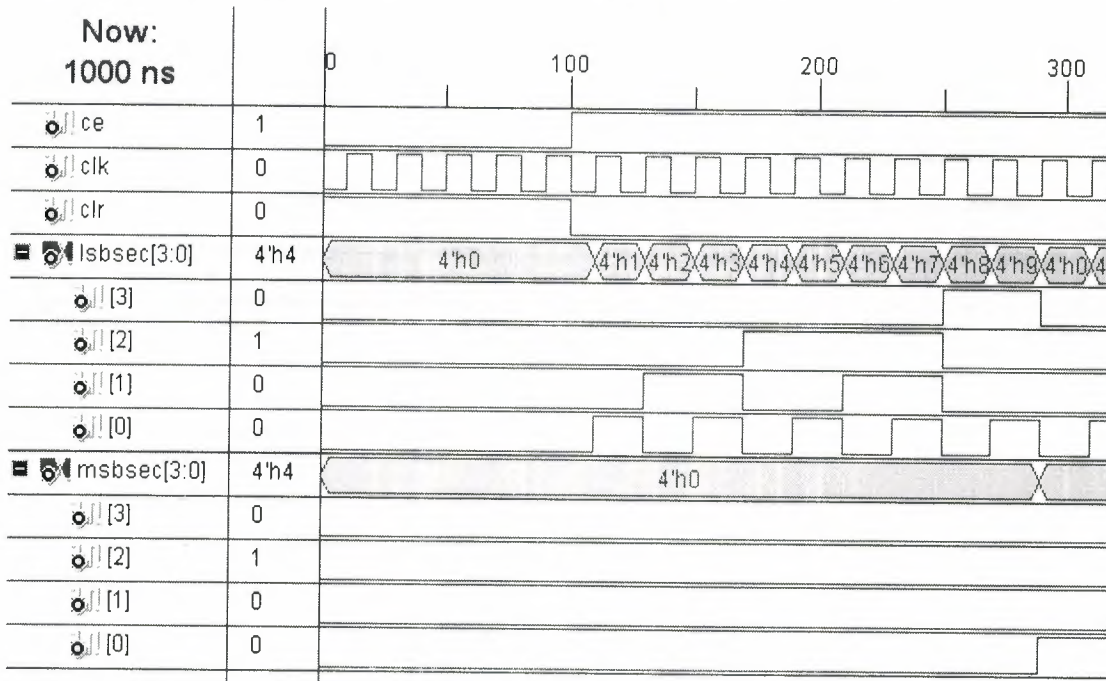


Figure25.Counting

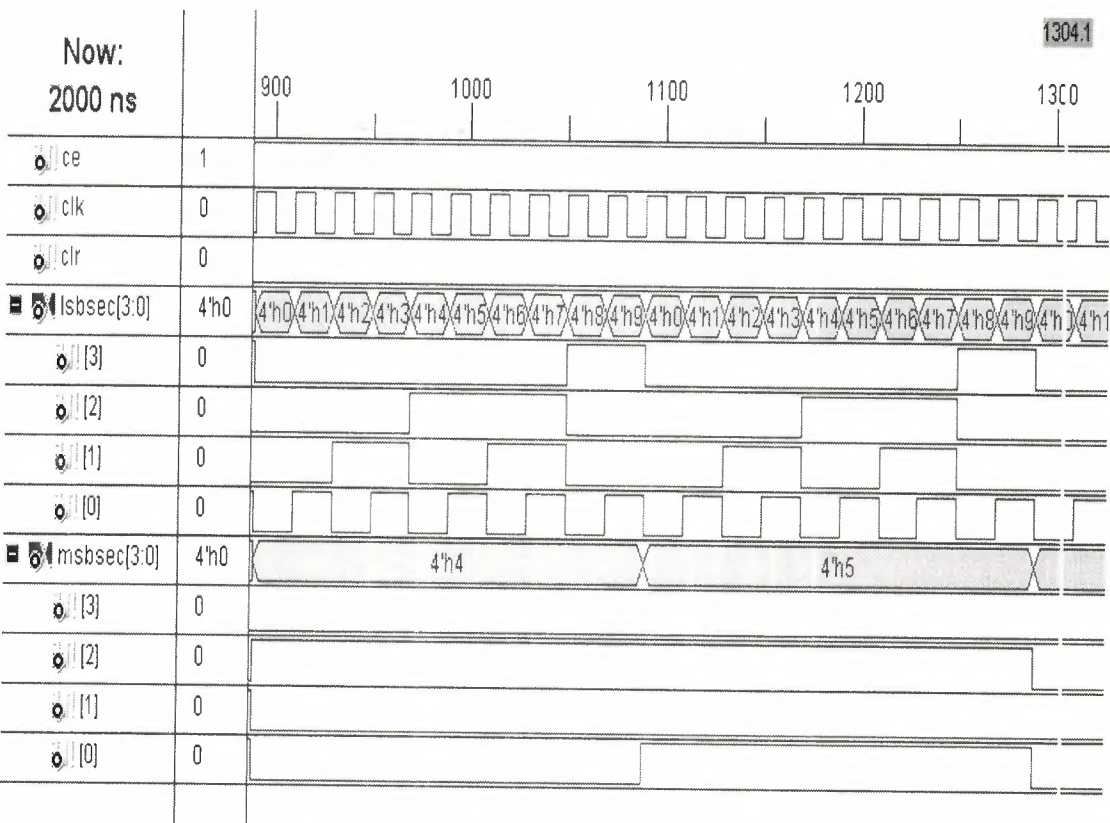


Figure26.Restarting to count

We can see in this figure when msbsec becomes 6 program starting the count from 0 again .

REFERENCES

[http://www.members.shaw.ca/kadirm/VHDL Course notes.htm](http://www.members.shaw.ca/kadirm/VHDL_Course_notes.htm)

<http://www.xilinx.com/company/index.htm>

[http://www.xilinx.com/products/silicon solutions](http://www.xilinx.com/products/silicon_solutions)