NEAR EAST UNIVERSITY

Faculty Of Engineering

Department Of Computer Engineering

UART TRANSMITTER

Graduating Project
COM 400

Student: Nadeem Hassan
20034377

Supervisor : Mehmet Kadir Özakman
MSc., P.Eng.

Nicosia 2008

ACKNOWLEDGEMENTS

First, I feel very hounded proud to pay my special regards to my project supervisor

'Mehmet Kadir Ozkman',

Who always gave me courage me up to do something which could ever though about. He is the one gave me a hand in any all conditions of mine. He delivered me and did his best of efforts to make me able to complete my project.

One looks back with appreciation to the brilliant teachers, but with gratitude to those who touched our human feelings. The curriculum is so much necessary raw material, but warmth is the vital element for the growing plant and for the soul of the child.

Carl Jung
1875-1961, Swiss Psychiatrist

Secondly I want to pay special regards to my parents and especially humble affection teaching of ammi, and great support of abu whose enduring support just unforgettable and supporting me in all my life till today. I am nothing without their supports. They also encouraged me in nomatter what so ever the situation. I shall never forget their sacrifices for my education so that I can enjoy my successful life not only as well educated but also well mannered, resputcful human being . At the end once again I am thankful to all my siblings & friends of mine who helped me out and kept waiting for this moment of my life or even encouraged me to complete my graduation Computer Engineer.

ABSTRACT

All computer operating systems in use today support serial ports, because serial ports have been around for decades. Parallel ports are a more recent invention and are much faster than serial ports. USB ports are only a few years old, and will likely replace both serial and parallel ports completely over the next several years.

The name "serial" comes from the fact that a serial port "serializes" data. That is, it takes a byte of data and transmits the 8 bits in the byte one at a time. The advantage is that a serial port needs only one wire to transmit the 8 bits (while a parallel port needs 8). The disadvantage is that it takes 8 times longer to transmit the data than it would if there were 8 wires. Serial ports lower cable costs and make cables smaller.

Before each byte of data, a serial port sends a start bit, which is a single bit with a value of 0. After each byte of data, it sends a stop bit to signal that the byte is complete. It may also send a parity bit.

The synthesizer that generates the design taking the consideration of the Xilinx FPGA device that we specified at the top level design. Each programmable chips have own characteristic. The same VHDL code is portable and can be synthesized to different FPGA programmable devices. We used Xilinx development system and Xilinx programmable devices because each vendor's tools are developed its on programmable devices.

TABLE OF CONTENTS

List of Abbreviations.

HDL	Hardware description Language
VHSIC	Very High Speed Integrated Circuit
ASIC	Application Specific Integrated Circuits
RAM	Random Access Memory
RTL	Register transfer level
FPGAQ	Field Programmable Gate Arrays
CLB	configurable logic blocks
PLD	Programmable Logic Devices
IEEE	The Institute of Electrical and Electronics Engineers (read eye-triple-e)
IC	Integrated circuits
Ada	Name of Programming Language
UART	Universal Asynchronous Receiver/Transmitter

INTRODUCTION

The aim of this project is to design UART Transmitter. The project consists of two chapters with introduction in the beginning and conclusion at the end.

Chapter one presents is the design description of UART Transmitter and describing its main functions including ground realities in VHDL with its basic concepts.

Chapter two is about the Tool description of the Project in which detailed explanations about the Xilinx-ISE as well as description of the usage of project Navigator in VHDL and its basic concepts, function, design, syntaxes, language codes and different rule of statements which I used in my project.

Finally, the conclusion section presents the important results obtained within the project, references used in the whole task.

CHAPTER 1

UART Transmitter

DESIGN.

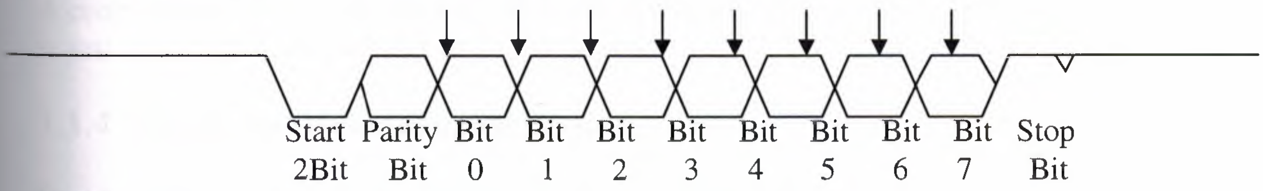
1. Defining the Function.

1.1 What it does:

UART stands for Universal Asynchronous Receiver and Transmitter. It is a protocol used in many places often including connecting terminals to computers. We'll look at the transmitter part of a UART.

1.1.1 Specifications:

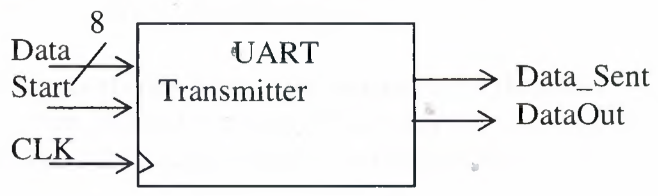
Data will be sent bit serially (one bit at a time) through a single line. The bit rate will be xx KHz. That means xx00 bits will arrive each second. (This is actually the rate for a MIDI UART communication. Computers generally run at 115.2 KHz). In each UART byte, there will be a start bit, the 8-bits of actually data, and a stop bit. This is shown below:



We do have parity bit present in here beside start bit.
Therefore totally number of bit we have at the end is 12.

Total No of Bit to send = 12bits

The transmitter's job is to get 8-bits of data (in parallel) and send each bit one at a time (DataOut) when it sees a Start signal. When our transmitter finishes transmitting, it tells us using a single bit output, Data_Sent. Here's the transmitter's block's I/O:



The clock we will use in our circuit is a 16MHz clock.

Now let's build the STD for the transmitter. Here's some pseudo-code for our STD:

```
1)  Wait for Start signal
2)  Grab data (and adding a START bit and END bit for a total of 10-bits.)
3)  For 10 bits:
    a. Put data bit to be sent on output (DataOut)
    b. Keep sending it until time to send next piece of data
```

c. Go to a.
4) Go to 1.

Here we see that there are states that wait time, and a state that counts how many times we've passed through it. Few other complexities are control word and parity check.

1.1.2 Parity bit.

A parity bit is a binary digit that is added to ensure that the number of bits with value of one in a given set of bits is always even or odd. Parity bits are used as the simplest error detecting code.
There are two variants of parity bits: even parity bit and odd parity bit. An even parity bit is set to 1 if the number of ones in a given set of bits is odd (making the *total* number of ones, including the parity bit, even). An odd parity bit is set to 1 if the number of ones in a given set of bits is even (making the *total* number of ones, including the parity bit, odd).

1.1.3 Control word.

A control word figures out the number of bits to be send if they are less than 8 bit or not , that is just to recognize the number of bits to be send.

1.1.4 Synchronous Serial Transmission.

Synchronous serial transmission requires that the sender and receiver share a clock with one another, or that the sender provide a strobe or other timing signal so that the receiver knows when to "read" the next bit of the data. In most forms of serial Synchronous communication, if there is no data available at a given instant to transmit, a fill character must be sent instead so that data is always being transmitted. Synchronous communication is usually more efficient because only data bits are transmitted between sender and receiver, and synchronous communication can be more costly if extra wiring and circuits are required to share a clock signal between the sender and receiver.

1.1.5 Asynchronous Serial Transmission.

Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver must agree on timing parameters in advance and special bits are added to each word which are used to synchronize the sending and receiving units.

When a word is given to the UART for Asynchronous transmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter.

1.1.6 Inputs & Outputs

In my project here I use the following design,

CTRL WORD=	Controls the number of bits
TBR =	Parallel Data
CRL=	Load the control word
MR=	Master Rest
SFD=	Controls the Output by TBRE
TBRL=	Parallel Data Load
TRC=	Transmit Clock
TBRE=	Output controlled by SFD input
TRE=	Transmit Error
TRO=	Transmit Output

I used a pakage in the design.

1.1.7 Package.

A package is the physical packaging of a chip, for example, PG84, VQ100, and PC48. Where we define the functions. In the project of I have been defining the fuction of parity which is a standard logic vector input carring 0-7 bits totally 8 and returns the output. Package is simply nothing but a fuction of parity.

Code.

```
package my_package is

   FUNCTION parity(inputs: std_logic_vector(7 downto 0)) RETURN std_logic;

end my_package;
```

1.1.8 Package Body.

The complete task for a package to be done in that particular package.

Code.

```
PACKAGE body my_package is

   FUNCTION parity(inputs: std_logic_vector(7 downto 0)) RETURN std_logic is
   variable temp: std_logic;
   begin
         temp:='0';
         for i in 7 downto 0 loop
            temp:=temp xor inputs(i);
         end loop;
            return temp;
         end parity;

end my_package;
```

Baud Rate.

Baud Rate represents the number of bits that are actually being sent over the media.

Data rate.

The amount of data that is actually moves from one medium to medium.

Modern high speed modems (2400, 9600, 14,400, and 19,200bps)

10^6/19200bps* = 52.083μsec

For each bit of data required to be sent= 16 clocks

So in 52.083μsec I have 16 clocks

Therefore,

$$52.083/16= 3.255125\ \mu sec$$

- Clock High Time: 1.6276 µsec.
- Clock Low Time: 1.6276 µsec.
- Input Setup Time: 0 µsec.
- Output V
    - Delay or Wait time: 10 µsec.
- Offset: 0 µsec
- Global Signals: GSR (FPGA)

Total clock Time = 3.255125µsec

To calculate the clock frequency,

$$f = 1/T$$
$$= 1/3.256 * 10^{-6}$$
$$= 10^{3} 10^{3} / 3.256$$

f=307.125kHz

3.255125µsec = 307.125kHz

VHDL DESIGN FLOW

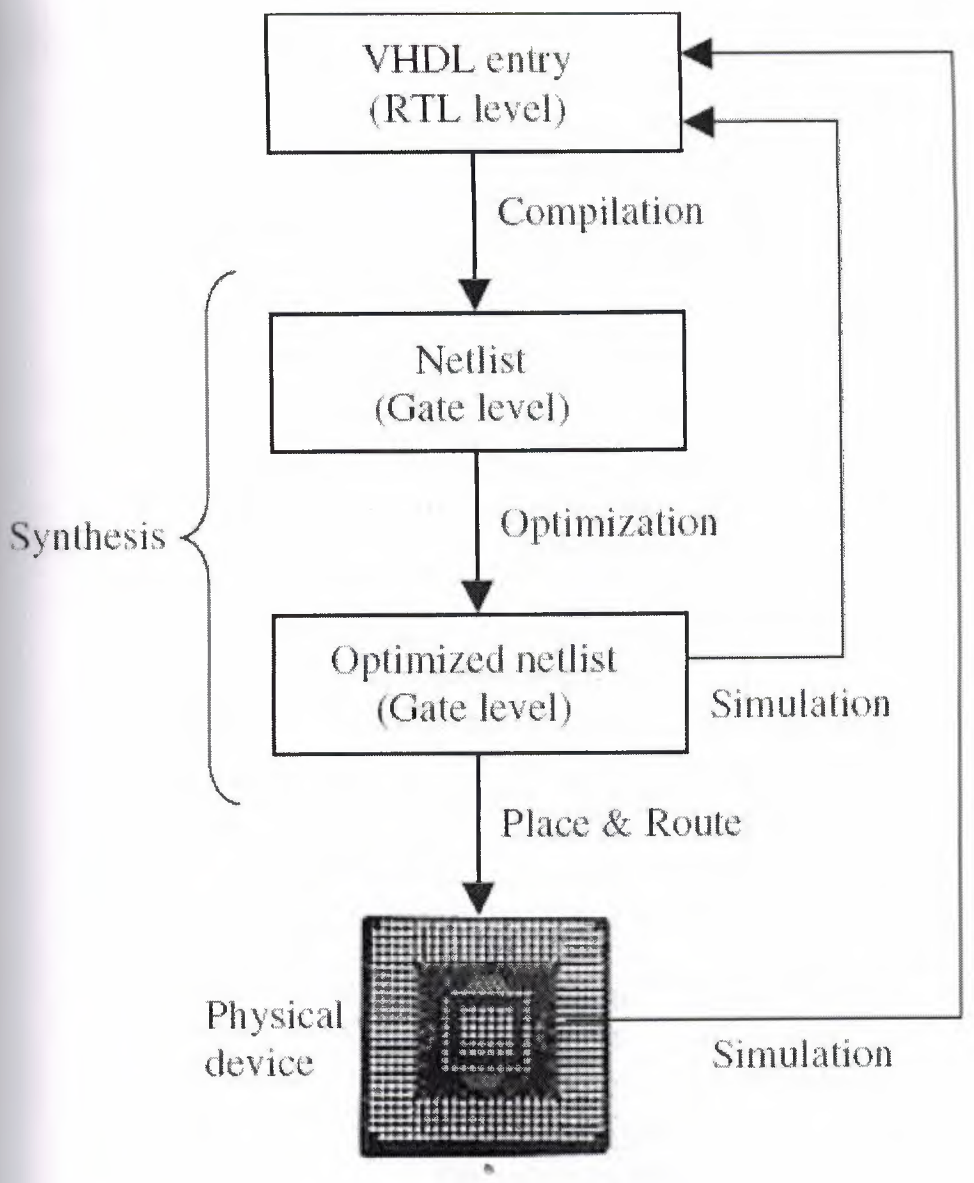
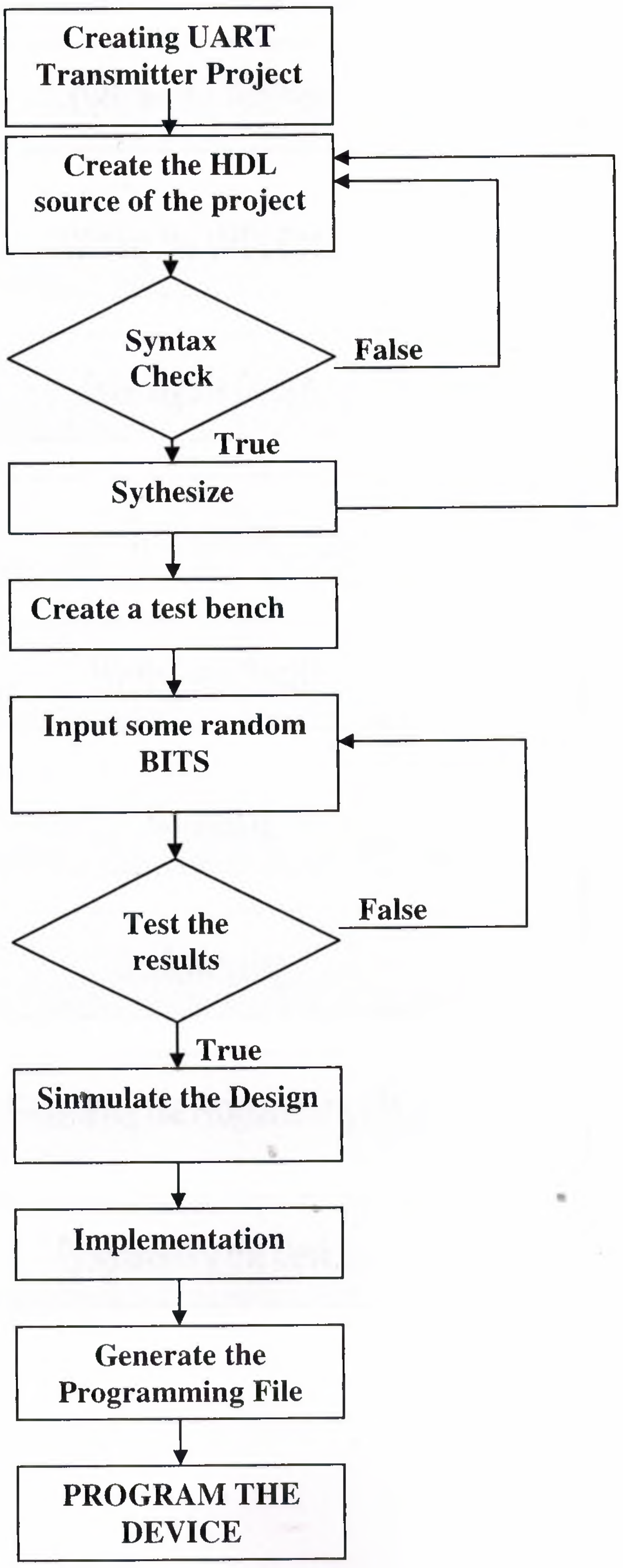
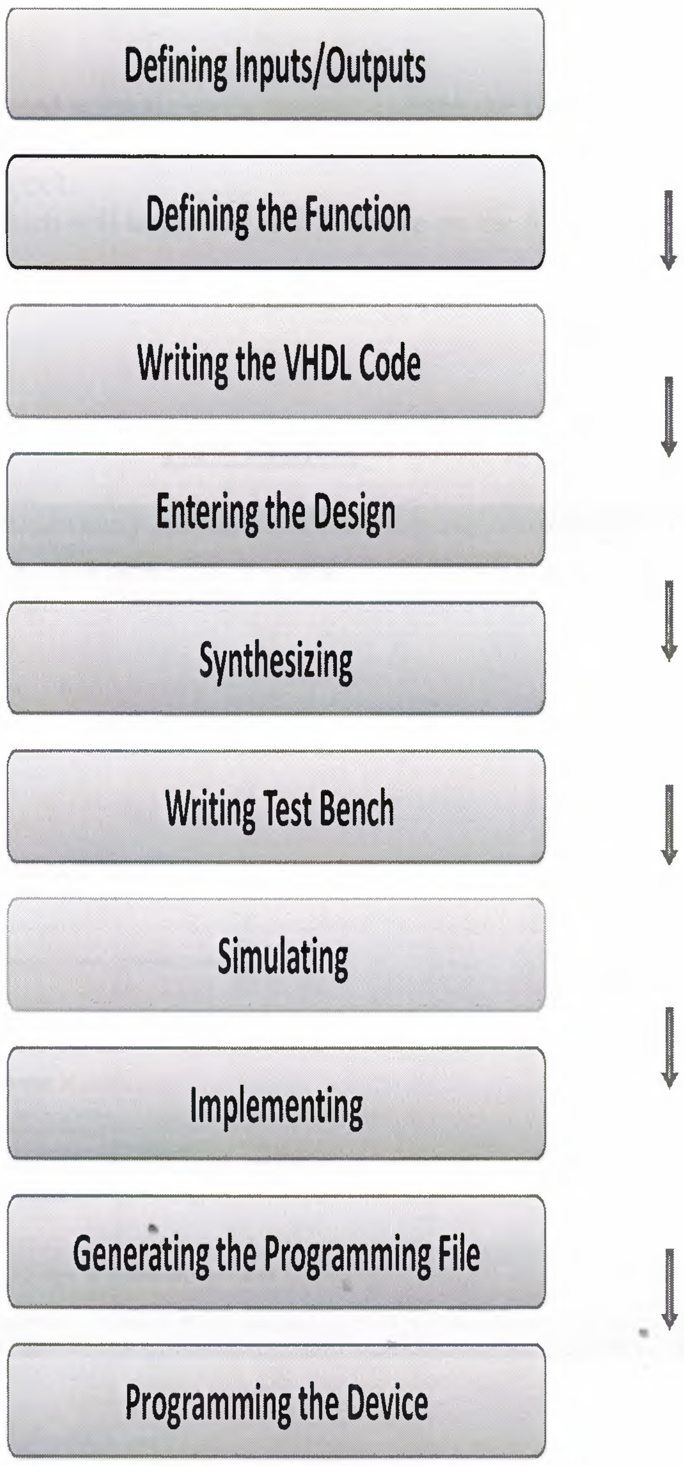


Figure 1.1
Summary of VHDL design flow.

UART DESIGN FLOW



PROCEDURE STEPS

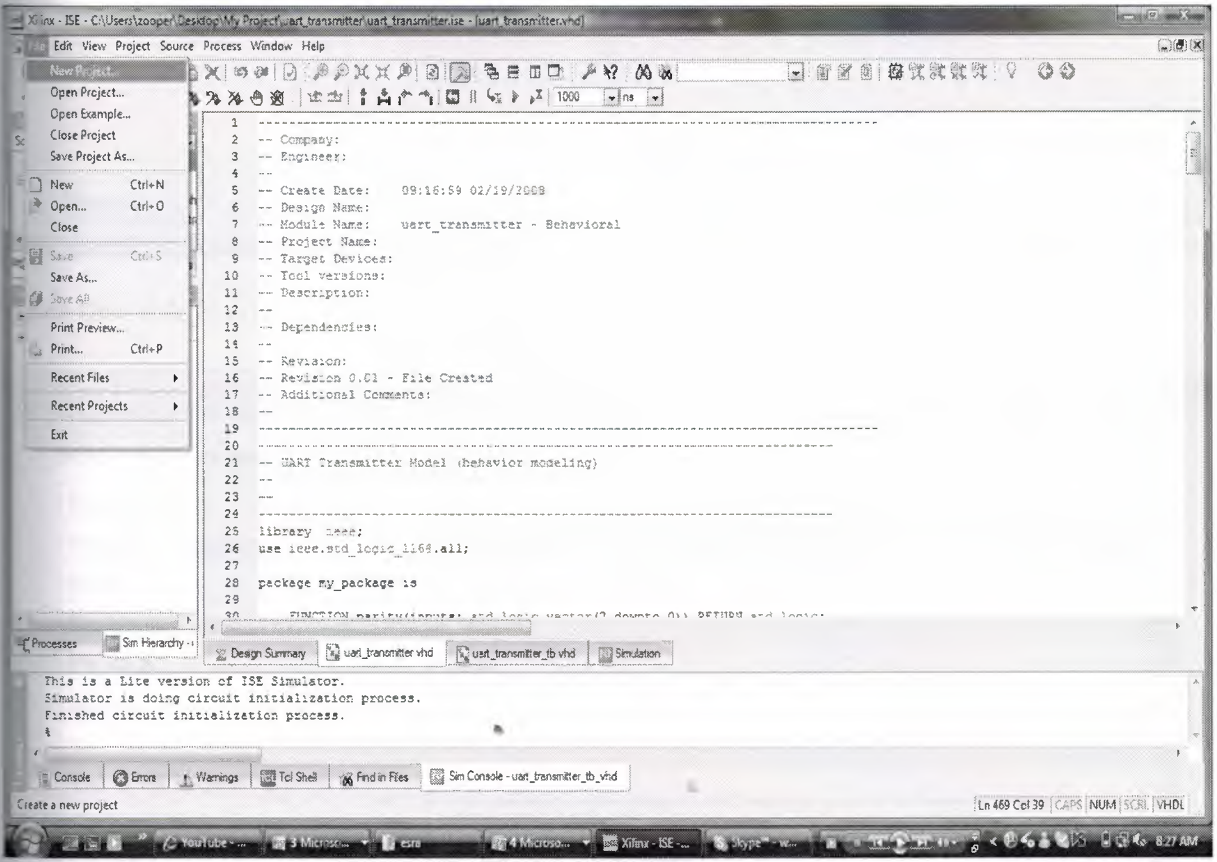


I use the xilinx – ISE integrated software environment to create the UART Transmitter Project and enter the VHDL code.

1.2.1 Create a New Project.

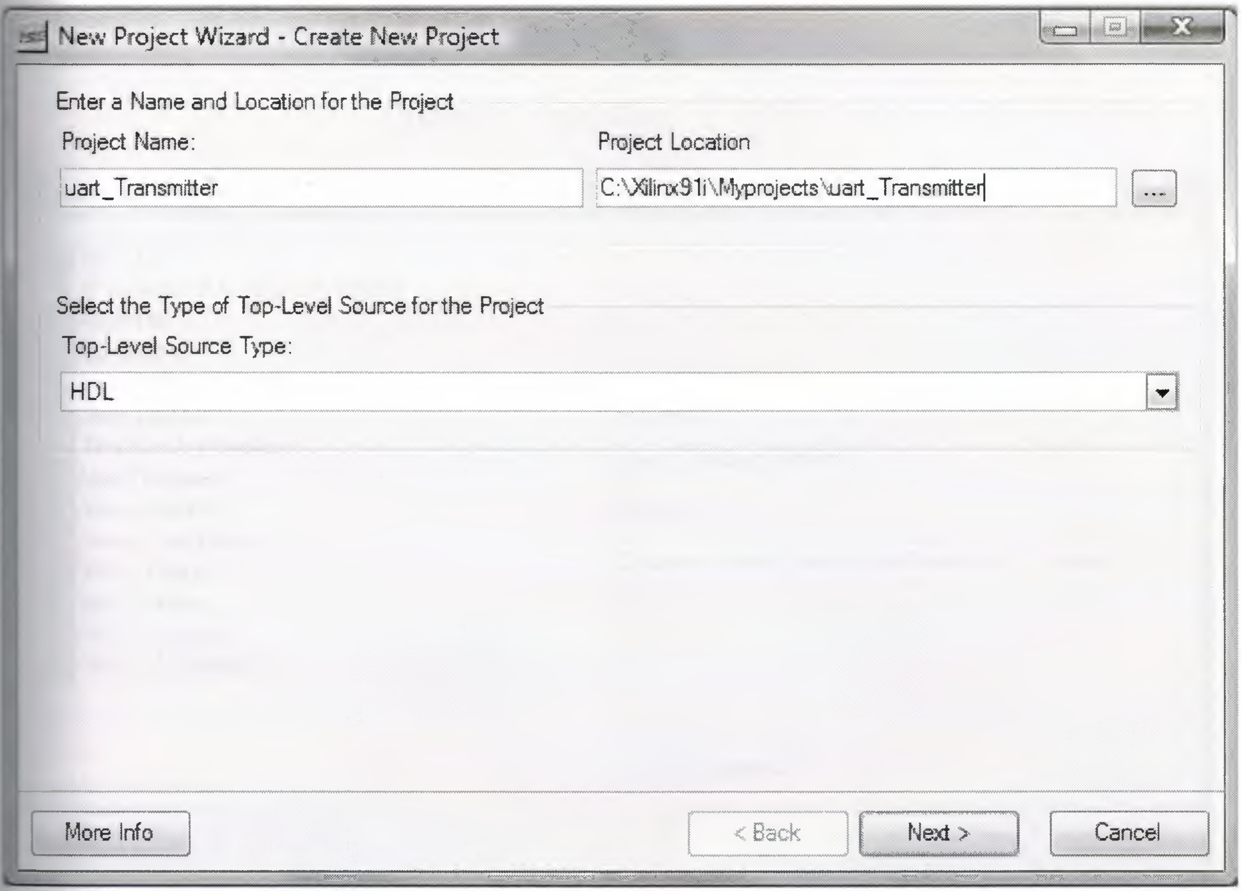
Create a new ISE project which will target the FPGA device on the SPARTAN-3E

New Project



```
--------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    09:16:59 02/19/2008
-- Design Name:
-- Module Name:    uart_transmitter - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
--------------------------------------------------------------------------------
--------------------------------------------------------------------------
-- UART Transmitter Model (behavior modeling)
--
--
--------------------------------------------------------------------------
library  ieee;
use ieee.std_logic_1164.all;

package my_package is

    FUNCTION parity(inputs: std_logic_vector(7 downto 0)) RETURN std_logic;
```

```
This is a Lite version of ISE Simulator.
Simulator is doing circuit initialization process.
Finished circuit initialization process.
%
```



Enter a Name and Location for the Project

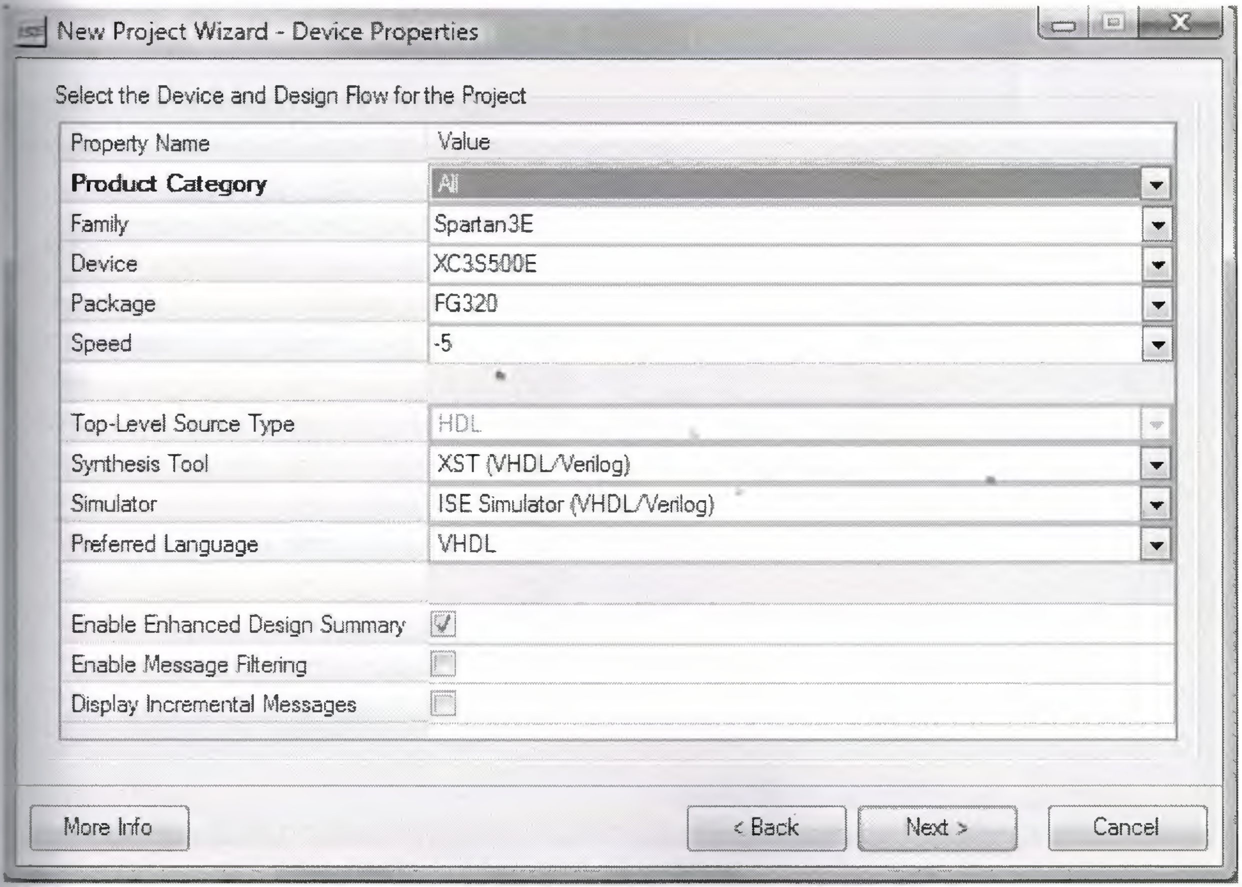
Project Name: uart_Transmitter

Project Location: C:\Xilinx91i\Myprojects\uart_Transmitter

Select the Type of Top-Level Source for the Project

Top-Level Source Type: HDL

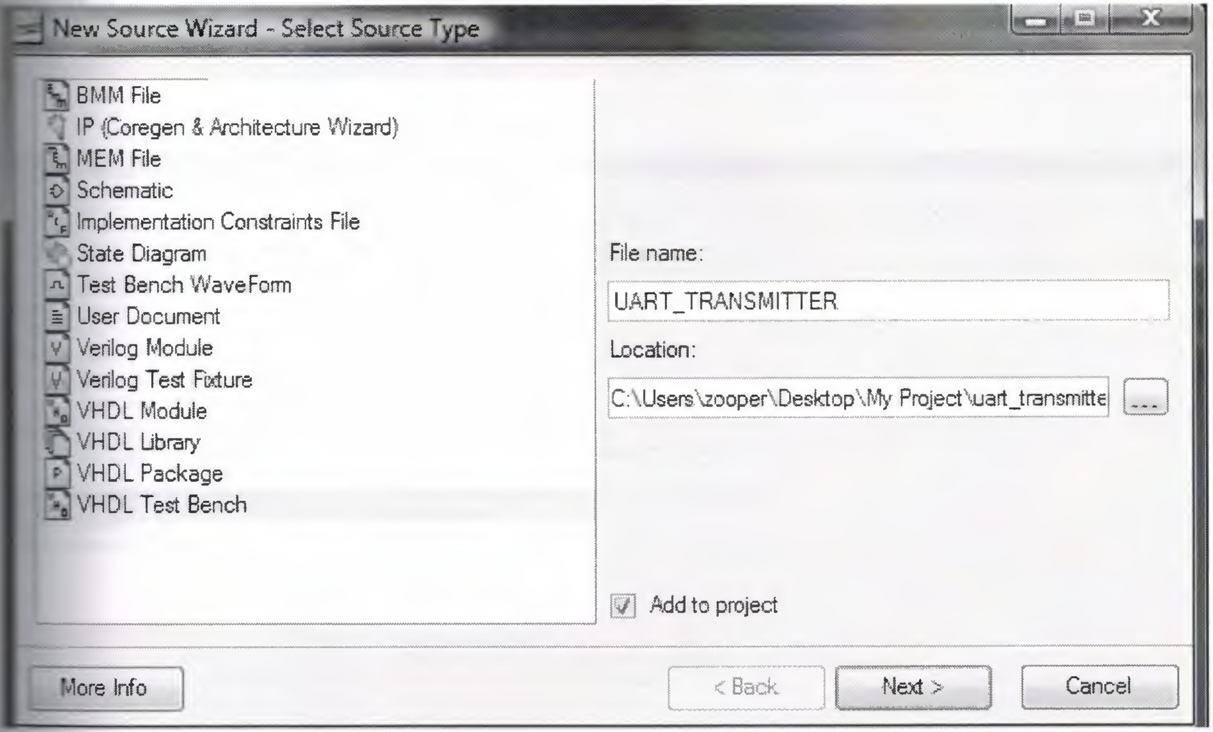
When the table is complete, your project properties will look like the following:



Select the Device and Design Flow for the Project

Property Name	Value
Product Category	All
Family	Spartan3E
Device	XC3S500E
Package	FG320
Speed	-5
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISE Simulator (VHDL/Verilog)
Preferred Language	VHDL
Enable Enhanced Design Summary	✓
Enable Message Filtering	☐
Display Incremental Messages	☐

Select Source Type.


New Source Wizard - Select Source Type

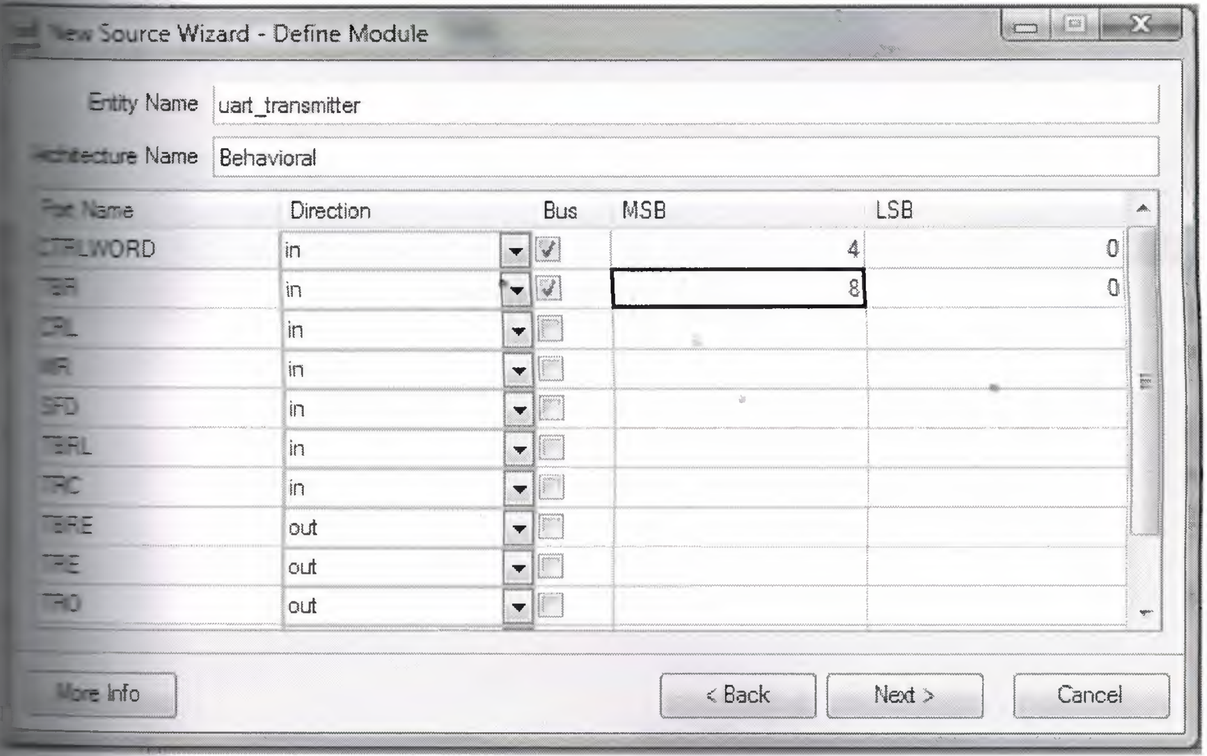
- BMM File
- IP (Coregen & Architecture Wizard)
- MEM File
- Schematic
- Implementation Constraints File
- State Diagram
- Test Bench WaveForm
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench

File name: UART_TRANSMITTER

Location: C:\Users\zooper\Desktop\My Project\uart_transmitte

☑ Add to project

Defining the Module.

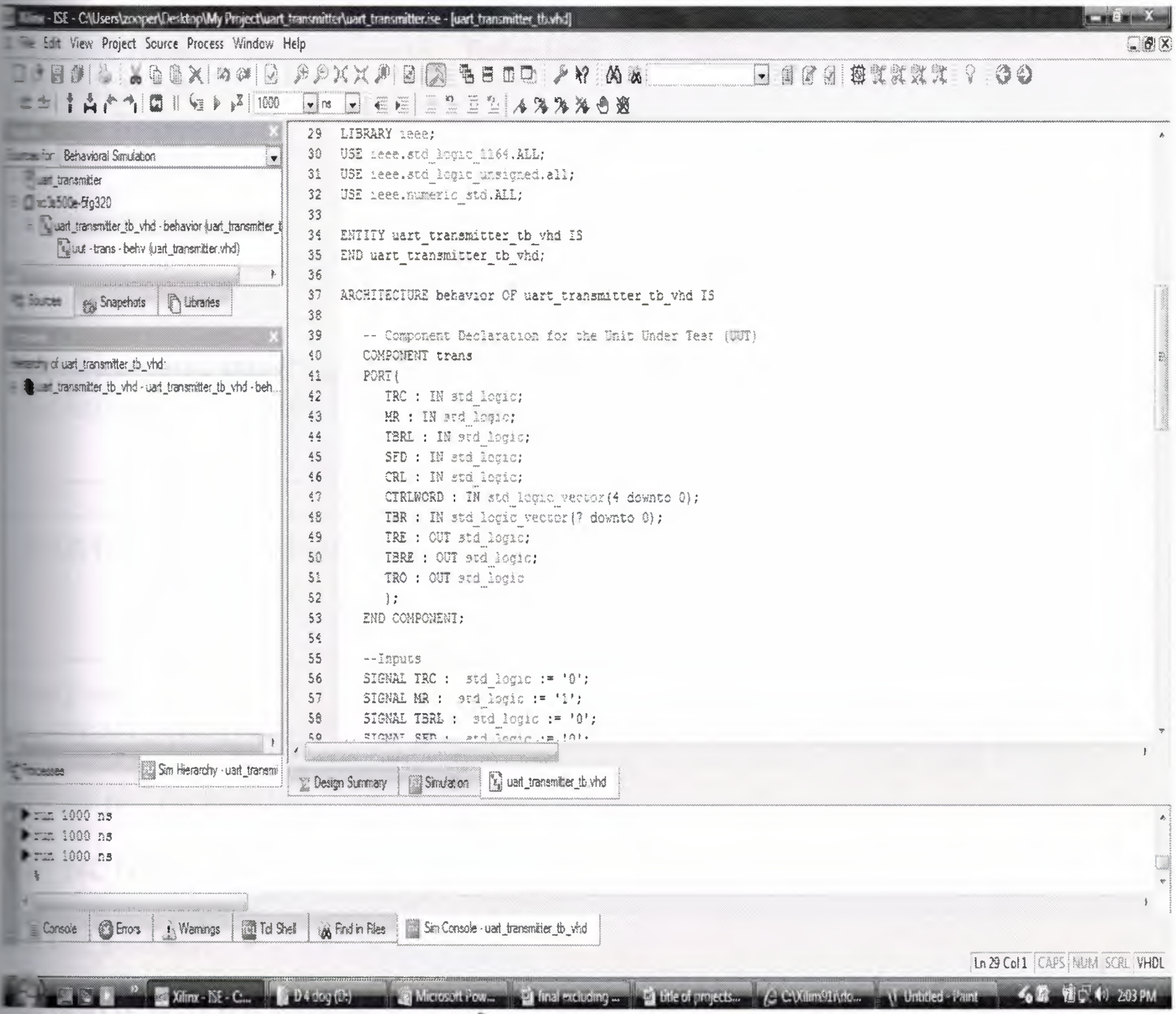

New Source Wizard - Define Module

Entity Name: uart_transmitter

Architecture Name: Behavioral

Port Name	Direction	Bus	MSB	LSB
CTRLWORD	in	✓	4	0
TBR	in	✓	8	0
CRL	in			
WR	in			
SFD	in			
TBRL	in			
TRC	in			
TBRE	out			
TRE	out			
TRO	out			

UART Transmitter Project in ISE.


Xilinx - ISE - C:\Users\zooper\Desktop\My Project\uart_transmitter\uart_transmitter.ise - [uart_transmitter_tb.vhd]
File Edit View Project Source Process Window Help
Sources for: Behavioral Simulation
uart_transmitter
xc3s500e-5fg320
uart_transmitter_tb_vhd - behavior (uart_transmitter_t
uut - trans - behv (uart_transmitter.vhd)
Sources
Snapshots
Libraries
Hierarchy of uart_transmitter_tb_vhd:
uart_transmitter_tb_vhd - uart_transmitter_tb_vhd - beh...
Processes
Sim Hierarchy - uart_transmi
Design Summary
Simulation
uart_transmitter_tb.vhd
Console
Errors
Warnings
Tcl Shell
Find in Files
Sim Console - uart_transmitter_tb_vhd
Ln 29 Col 1
CAPS
NUM
SCRL
VHDL
2:03 PM

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

ENTITY uart_transmitter_tb_vhd IS
END uart_transmitter_tb_vhd;

ARCHITECTURE behavior OF uart_transmitter_tb_vhd IS

   -- Component Declaration for the Unit Under Test (UUT)
   COMPONENT trans
   PORT(
      TRC : IN std_logic;
      MR : IN std_logic;
      TBRL : IN std_logic;
      SFD : IN std_logic;
      CRL : IN std_logic;
      CTRLWORD : IN std_logic_vector(4 downto 0);
      TBR : IN std_logic_vector(7 downto 0);
      TRE : OUT std_logic;
      TBRE : OUT std_logic;
      TRO : OUT std_logic
      );
   END COMPONENT;

   --Inputs
   SIGNAL TRC :  std_logic := '0';
   SIGNAL MR :  std_logic := '1';
   SIGNAL TBRL :  std_logic := '0';
   SIGNAL SFD :  std_logic := '0';
```

```
run 1000 ns
run 1000 ns
run 1000 ns
```

Project in ISE

1.2 VHDL CODE OF UART TRANSMITTER.

```
----------------------------------------------------------------------------------
-- Company:           Near East University
-- Engineer:          Nadeem Hassan
--
-- Create Date:    09:16:59 02/19/2008
-- Design Name:
-- Module Name:    uart_transmitter - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
---------------------------------------------------------------------------
-- UART Transmitter Model (behavior modeling)
--
--
---------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

package my_package is

  FUNCTION parity(inputs: std_logic_vector(7 downto 0)) RETURN std_logic;

end my_package;

PACKAGE body my_package is

  FUNCTION parity(inputs: std_logic_vector(7 downto 0)) RETURN std_logic is
  variable temp: std_logic;
  begin
        temp:='0';
        for i in 7 downto 0 loop
           temp:=temp xor inputs(i);
        end loop;
           return temp;
        end parity;

end my_package;

-------------------------------------------------------------------------
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use work.my_package.all;
---------------------------------------------------------------------
entity trans is
port(   TRC:            in std_logic;
        MR:             in std_logic;
        TBRL:           in std_logic;
        SFD:            in std_logic;
        CRL:            in std_logic;
        CTRLWORD:in std_logic_vector(4 downto 0);
        TBR:            in std_logic_vector(7 downto 0);
        TRE:            out std_logic;
        TBRE:           out std_logic;
        TRO:            out std_logic
);
end trans;
---------------------------------------------------------------------
architecture behv of trans is

  constant C0: unsigned(4 downto 0):="00001";
  constant CI: unsigned(4 downto 0):="01000";
  constant CM: unsigned(4 downto 0):="10000";
  constant MM: unsigned(3 downto 0):="1111";
  signal trans_reg: std_logic_vector(11 downto 0);
  signal trans_reg_temp: std_logic_vector(11 downto 0);
  signal TBR_sig: std_logic_vector(7 downto 0);
  signal old_tbr_sig: std_logic_vector(7 downto 0);
  signal delay: std_logic_vector(0 downto 0);
  signal go: std_logic;
  signal t_pari: std_logic;
  signal clk: std_logic;
  signal i: unsigned(3 downto 0);
  signal ctrl_word: std_logic_vector(4 downto 0);

begin

--P0: process(MR,TRC,i,TBR)                -- generate 16* clock
--begin
--      if MR='1' then
--              i<="00000";
--      elsif TRC='1' and TRC'event then
--              i<=i+C0;
--              if i=CI then
--                      clk<='1';
--              elsif i=CM then
--                      i<="00000";
--                      clk<='0';
--              end if;
--      end if;
--end process;

P1: process(CRL,CTRLWORD)                    -- load control word here
  begin
        if CRL='1' then
           ctrl_word<=CTRLWORD;
        end if;
  end process;

-- this process make out the transmit register according to control word.
P2: process(clk,ctrl_word,MR,TBR,TBRL,TBR_sig,t_pari)
  begin
        if (MR='1') then
           TBR_sig<="11111111";
           t_pari <= '0';
           trans_reg_temp <= "1111" & TBR_sig;
        else
           if TBRL='0' then
                 TBR_sig <= TBR;
                 t_pari <= parity(TBR);          -- call function and compute the parity
           end if;

        case ctrl_word is
           when "00000"|"00001" =>
                 if t_pari='0' then                              -- odd parity
                    trans_reg_temp <= "111" & "11" & '1' & TBR_sig(4 downto 0) & '0';

                 elsif t_pari='1' then                           -- even parity
                    trans_reg_temp <= "111" & "11" & '0' & TBR_sig(4 downto 0) & '0';
                 end if;
           when "00010"|"00011" =>
                 if t_pari='0' then                              -- odd parity
                    trans_reg_temp <= "111" & "11" & '0' & TBR_sig(4 downto 0) & '0';

                 elsif t_pari='1' then                           -- even parity
                    trans_reg_temp <= "111" & "11" & '1' & TBR_sig(4 downto 0) & '0';
                 end if;
           when "00100"|"00110"|"00101"|"00111" =>                -- no parity
                 trans_reg_temp <= "1111" & "11" & TBR_sig(4 downto 0) & '0';
           when "01000"|"01001" =>
                 if t_pari='0' then                              -- odd parity
                    trans_reg_temp <= "11" & "11" & '1' & TBR_sig(5 downto 0) & '0';

                 elsif t_pari='1' then                           -- even parity
                    trans_reg_temp <= "11" & "11" & '0' & TBR_sig(5 downto 0) & '0';
                 end if;
           when "01010"|"01011" =>
                 if t_pari='0' then                              -- odd parity
                    trans_reg_temp <= "11" & "11" & '0' & TBR_sig(5 downto 0) & '0';
```

```
          elsif t_pari='1' then                         -- even parity
             trans_reg_temp <= "11" & "11" & '1' & TBR_sig(5 downto 0) & '0';
          end if;
      when "01100"|"01110"|"01101"|"01111" =>                   -- no parity
          trans_reg_temp <= "111" & "11" & TBR_sig(5 downto 0) & '0';
      when "10000"|"10001" =>
          if t_pari='0' then                            -- odd parity
             trans_reg_temp <= '1' & "11" & '1' & TBR_sig(6 downto 0) & '0';

          elsif t_pari='1' then                         -- even parity
             trans_reg_temp <= '1' & "11" & '0' & TBR_sig(6 downto 0) & '0';
          end if;
      when "10010"|"10011" =>
          if t_pari='0' then                            -- odd parity
             trans_reg_temp <= '1' & "11" & '0' & TBR_sig(6 downto 0) & '0';

          elsif t_pari='1' then                         -- even parity
             trans_reg_temp <= '1' & "11" & '1' & TBR_sig(6 downto 0) & '0';
          end if;
      when "10100"|"10110"|"10101"|"10111" =>                   -- no parity
          trans_reg_temp <= "11" & "11" & TBR_sig(6 downto 0) & '0';
      when "11000"|"11001" =>
          if t_pari='0' then                            -- odd parity
             trans_reg_temp <= "11" & '1' & TBR_sig(7 downto 0) & '0';
          elsif t_pari='1' then                         -- even parity
             trans_reg_temp <= "11" & '0' & TBR_sig(7 downto 0) & '0';
          end if;
      when "11010"|"11011" =>
          if t_pari='0' then                            -- odd parity
             trans_reg_temp <= "11" & '0' & TBR_sig(7 downto 0) & '0';
          elsif t_pari='1' then                         -- even parity
             trans_reg_temp <= "11" & '1' & TBR_sig(7 downto 0) & '0';
          end if;
      when others =>                                    -- no parity
          trans_reg_temp <= '1' & "11" & TBR_sig(7 downto 0) & '0';
    end case;
  end if;
end process;

-- P3 describes the whole transmission procedure
P3: process
        variable cnt: integer range 0 to 12;
        variable cnt_limit: integer range 0 to 12;
  begin

        wait until TRC'event and TRC='1';

        if MR='1' then
           old_tbr_sig <= "11111111";
           delay<="0";
           go<='0';
  i <= "0000";
else
  if (i=MM) then
       i<="0000";
       if(go='0') then
          delay<="0";
          cnt := 12;
          TRE <= '1';
          if (old_tbr_sig=TBR_sig) then
               go <= '0';
          elsif (old_tbr_sig/=TBR_sig) then
               go <='1';
          end if;
          trans_reg <= "111111111111";
       elsif (go='1' and delay="0") then
          go<='1';
          cnt:=0;
          delay<=delay+1;
          TRE <= '1';
          trans_reg <= "111111111111";
       elsif (go='1' and delay="1" and cnt=0) then
          go <= '1';
          cnt:=cnt+1;
          delay<=delay+0;
          old_tbr_sig<=TBR_sig;
          TRE<='0';
          trans_reg <= trans_reg_temp;
       elsif (go='1' and delay="1" and cnt/=0) then
          trans_reg <= '1' & trans_reg(11 downto 1);
          case ctrl_word(4 downto 2) is
               when "000" =>          if ctrl_word(0)='0' then
                                         cnt_limit := 8;
                                      elsif ctrl_word(0)='1' then
                                         cnt_limit :=9;
                                      end if;
               when "001" =>          if ctrl_word(0)='0' then
                                         cnt_limit := 7;
                                      elsif ctrl_word(0)='1' then
                                         cnt_limit :=8;
                                      end if;
               when "010" =>          if ctrl_word(0)='0' then
                                         cnt_limit := 9;
                                      elsif ctrl_word(0)='1' then
                                         cnt_limit :=10;
                                      end if;
               when "011" =>          if ctrl_word(0)='0' then
                                         cnt_limit := 8;
                                      elsif ctrl_word(0)='1' then
                                         cnt_limit :=9;
                                      end if;
               when "100" =>          if ctrl_word(0)='0' then
```

```
                        cnt_limit := 10;
                    elsif ctrl_word(0)='1' then
                        cnt_limit :=11;
                    end if;
            when "101" =>       if ctrl_word(0)='0' then
                        cnt_limit := 9;
                    elsif ctrl_word(0)='1' then
                        cnt_limit :=10;
                    end if;
            when "110" =>       if ctrl_word(0)='0' then
                        cnt_limit :=11;
                    elsif ctrl_word(0)='1' then
                        cnt_limit :=12;
                    end if;
            when "111" =>       if ctrl_word(0)='0' then
                        cnt_limit :=10;
                    elsif ctrl_word(0)='1' then
                        cnt_limit :=11;
                    end if;
            when others =>
        end case;

        if cnt/=cnt_limit then
            go <= '1';
            delay<=delay+0;
            cnt:=cnt+1;
            TRE<='0';
        elsif cnt=cnt_limit then
            go <= '0';
            delay<="0";
            TRE<='1';
        end if;
      end if;

      if SFD='1' then
        TBRE <= 'Z';
      elsif SFD='0' then
        if (cnt=0 or cnt=1) then
            TBRE <= '0';
        else
            TBRE <= '1';
        end if;
      end if;
    else
      i<=i+1;
    end if;
  end if;
end process;
```

```
    -- this cocurrent statement handles the serial output of Transmitter
    tro <= trans_reg(0);

end behv;
```

<< ....>>

1.3 SYNTHESIS

I use the Xilinx-ISE- 'synthesis tool' to synthesize the code.

The synthesis Report goes as follows

- Synthesis Options Summary
- HDL Compilation
- Design Hierarchy Analysis
- HDL Analysis
- HDL Synthesis
- Advanced HDL Synthesis
- Low Level Synthesis
- Partition Report
- Final Report

The synthesize result is shown below:

```
Release 9.1.02i - xst J.32
Copyright (c) 1995-2007 Xilinx, Inc.  All rights reserved.
--> Parameter TMPDIR set to ./xst/projnav.tmp
CPU : 0.00 / 1.73 s | Elapsed : 0.00 / 2.00 s

--> Parameter xsthdpdir set to ./xst
CPU : 0.00 / 1.73 s | Elapsed : 0.00 / 2.00 s

--> Reading design: trans.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                    : "trans.prj"
Input Format                       : mixed
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "trans"
Output Format                      : NGC
Target Device                      : xc3s500e-5-fg320

---- Source Options
Top Module Name                    : trans
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               : trans.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/xilinx/myprojects/uart_transmitter/uart_transmitter.vhd" in
Library work.
Package <my_package> compiled.
Package body <my_package> compiled.
Entity <trans> compiled.
Entity <trans> (Architecture <behv>) compiled.

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


=========================================================================
======
*                    HDL Analysis                    *
=========================================================================
======
Analyzing Entity <trans> in library <work> (Architecture <behv>).
Entity <trans> analyzed. Unit <trans> generated.


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

Performing bidirectional port resolution...

Synthesizing Unit <trans>.
   Related source file is
"C:/Xilinx91i/xilinx/myprojects/uart_transmitter/uart_transmitter.vhd".
WARNING:Xst:1780 - Signal <clk> is never used or assigned.
WARNING:Xst:737 - Found 8-bit latch for signal <TBR_sig>.
WARNING:Xst:737 - Found 1-bit latch for signal <t_pari>.
WARNING:Xst:737 - Found 5-bit latch for signal <ctrl_word>.
WARNING:Xst - Property "use_dsp48" is not applicable for this technology.
   Found 1-bit register for signal <TRE>.
   Found 1-bit tristate buffer for signal <TBRE>.
   Found 4-bit register for signal <cnt>.
   Found 4-bit adder for signal <cnt$add0000> created at line 272.
   Found 4-bit comparator equal for signal <cnt$cmp_eq0001> created at line 269.
   Found 4-bit comparator not equal for signal <cnt$cmp_ne0000> created at line 269.
   Found 4-bit 8-to-1 multiplexer for signal <cnt_limit$mux0001> created at line 225.
   Found 1-bit register for signal <delay<0>>.
   Found 1-bit register for signal <go>.
   Found 8-bit comparator equal for signal <go$cmp_eq0001> created at line 204.
   Found 4-bit up counter for signal <i>.
   Found 1-bit register for signal <Mtridata_TBRE> created at line 282.
   Found 1-bit register for signal <Mtrien_TBRE> created at line 282.
   Found 8-bit register for signal <old_tbr_sig>.
   Found 1-bit xor8 for signal <t_pari$xor0001> created at line 41.
   Found 12-bit register for signal <trans_reg>.
   Summary:
        inferred   1 Counter(s).
        inferred  29 D-type flip-flop(s).
        inferred   1 Adder/Subtractor(s).
        inferred   3 Comparator(s).
        inferred   4 Multiplexer(s).
        inferred   1 Xor(s).
```

```
    inferred   1 Tristate(s).
Unit <trans> synthesized.

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

Macro Statistics
# Adders/Subtractors                   : 1
 4-bit adder                           : 1
# Counters                             : 1
 4-bit up counter                      : 1
# Registers                            : 8
 1-bit register                        : 5
 12-bit register                       : 1
 4-bit register                        : 1
 8-bit register                        : 1
# Latches                              : 3
 1-bit latch                           : 1
 5-bit latch                           : 1
 8-bit latch                           : 1
# Comparators                          : 3
 4-bit comparator equal                : 1
 4-bit comparator not equal            : 1
 8-bit comparator equal                : 1
# Multiplexers                         : 1
 4-bit 8-to-1 multiplexer              : 1
# Tristates                            : 1
 1-bit tristate buffer                 : 1
# Xors                                 : 1
 1-bit xor8                            : 1

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

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

Loading device for application Rf_Device from file '3s500e.nph' in environment C:\Xilinx91i.

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

Macro Statistics
# Adders/Subtractors                   : 1
 4-bit adder                           : 1
```

```
# Counters                               : 1
 4-bit up counter                          : 1
# Registers                              : 29
 Flip-Flops                              : 29
# Latches                                : 3
 1-bit latch                            : 1
 5-bit latch                            : 1
 8-bit latch                            : 1
# Comparators                               : 3
 4-bit comparator equal                       : 1
 4-bit comparator not equal                     : 1
 8-bit comparator equal                       : 1
# Multiplexers                            : 1
 4-bit 8-to-1 multiplexer                    : 1
# Xors                                  : 1
 1-bit xor8                              : 1

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

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

Optimizing unit <trans> ...
WARNING:Xst:1710 - FF/Latch  <trans_reg_11> (without init value) has a constant value of
1 in block <trans>.
WARNING:Xst:1710 - FF/Latch  <trans_reg_10> (without init value) has a constant value of
1 in block <trans>.
WARNING:Xst:1710 - FF/Latch  <trans_reg_10> (without init value) has a constant value of
1 in block <trans>.
WARNING:Xst:1710 - FF/Latch  <trans_reg_10> (without init value) has a constant value of
1 in block <trans>.
WARNING:Xst:1710 - FF/Latch  <trans_reg_10> (without init value) has a constant value of
1 in block <trans>.

Mapping all equations...
Building and optimizing final netlist ...
Found area constraint ratio of 100 (+ 5) on block trans, actual ratio is 1.

Final Macro Processing ...

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

Macro Statistics
# Registers                              : 31
 Flip-Flops                              : 31
```

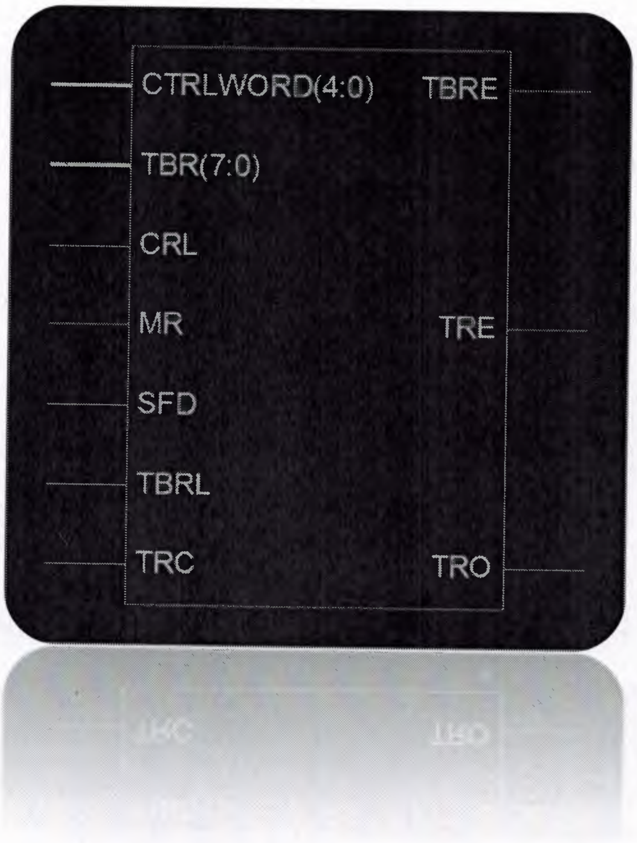
```
=========================================================================
======



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

Partition Implementation Status
-------------------------------

 No Partitions were found in this design.

-------------------------------
=========================================================================
======
*                      Final Report                        *
=========================================================================
======
Final Results
RTL Top Level Output File Name     : trans.ngr
Top Level Output File Name         : trans
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : NO

Design Statistics
# IOs                              : 21

Cell Usage :
# BELS                             : 96
#      INV                         : 1
#      LUT2                        : 8
#      LUT2_D                      : 1
#      LUT3                        : 15
#      LUT3_D                      : 1
#      LUT4                        : 51
#      LUT4_D                      : 5
#      LUT4_L                      : 8
#      MUXF5                       : 6
# FlipFlops/Latches                : 45
#      FDE                         : 13
#      FDR                         : 4
#      FDRE                        : 6
#      FDSE                        : 8
#      LD                    : 5
#      LDC_1                 : 1
#      LDP_1                 : 8
# Clock Buffers              : 3
#      BUFGP                 : 3
# IO Buffers                 : 18
#      IBUF                  : 15
#      OBUF                  : 2
#      OBUFT                 : 1
=========================================================================
======


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

Selected Device : 3s500efg320-5

 Number of Slices:                      49  out of   4656     1%
 Number of Slice Flip Flops:            40  out of   9312     0%
 Number of 4 input LUTs:                90  out of   9312     0%
 Number of IOs:                         21
 Number of bonded IOBs:                 21  out of    232     9%
    IOB Flip Flops:                      5
 Number of GCLKs:                        3  out of     24    12%


---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

---------------------------


=========================================================================
======
TIMING REPORT

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

Clock Information:
------------------
-----------------------------------+------------------------+-------+
Clock Signal                       | Clock buffer(FF name)  | Load  |
-----------------------------------+------------------------+-------+
TRC                                | BUFGP                  | 31    |
TBRL                               | BUFGP                  | 9     |
CRL                             | BUFGP                 | 5     |
-------------------------------+-----------------------+-------+

Asynchronous Control Signals Information:
--------------------------------------
-------------------------------+-----------------------+-------+
Control Signal                  | Buffer(FF name)       | Load  |
-------------------------------+-----------------------+-------+
MR                              | IBUF                  | 9     |
-------------------------------+-----------------------+-------+


Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 5.686ns (Maximum Frequency: 175.858MHz)
   Minimum input arrival time before clock: 5.787ns
   Maximum output required time after clock: 4.040ns
   Maximum combinational path delay: No path found

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

=========================================================================
Timing constraint: Default period analysis for Clock 'TRC'
  Clock period: 5.686ns (frequency: 175.858MHz)
  Total number of paths / destination ports: 480 / 57
-------------------------------------------------------------------------
Delay:               5.686ns (Levels of Logic = 3)
  Source:            i_1 (FF)
  Destination:       cnt_0 (FF)
  Source Clock:      TRC rising
  Destination Clock: TRC rising

  Data Path: i_1 to cnt_0
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDR:C->Q              4   0.514   0.568  i_1 (i_1)
     LUT4:I1->O            5   0.612   0.607  old_tbr_sig_not000111 (go_cmp_eq0000)
     LUT2:I1->O           15   0.612   0.867  cnt_and000011 (trans_reg_not0001)
     LUT4:I3->O            4   0.612   0.499  cnt_and00001 (cnt_and0000)
     FDRE:R                    0.795          cnt_0
    ----------------------------------------
    Total                      5.686ns (3.145ns logic, 2.541ns route)
                                       (55.3% logic, 44.7% route)
```

```
================================================================
======
Timing constraint: Default OFFSET IN BEFORE for Clock 'TRC'
 Total number of paths / destination ports: 67 / 50
-------------------------------------------------------------------------
Offset:              5.787ns (Levels of Logic = 5)
 Source:             SFD (PAD)
 Destination:        cnt_2 (FF)
 Destination Clock: TRC rising

 Data Path: SFD to cnt_2
                       Gate    Net
  Cell:in->out     fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O          11   1.106   0.796  SFD_IBUF (SFD_IBUF)
   LUT4:I3->O          1   0.612   0.426  cnt_cmp_ne0000199_SW0_SW0 (N626)
   LUT4:I1->O          2   0.612   0.383  cnt_cmp_ne0000199_SW0 (N606)
   LUT4:I3->O          1   0.612   0.360  cnt_mux0001<1>29 (cnt_mux0001<1>_map10)
   LUT4:I3->O          1   0.612   0.000  cnt_mux0001<1>65 (cnt_mux0001<1>)
   FDRE:D                  0.268          cnt_2
  ----------------------------------------
  Total                 5.787ns (3.822ns logic, 1.965ns route)
                           (66.0% logic, 34.0% route)

================================================================
======
Timing constraint: Default OFFSET IN BEFORE for Clock 'TBRL'
 Total number of paths / destination ports: 16 / 9
-------------------------------------------------------------------------
Offset:              3.639ns (Levels of Logic = 3)
 Source:             TBR<3> (PAD)
 Destination:        t_pari (LATCH)
 Destination Clock: TBRL rising

 Data Path: TBR<3> to t_pari
                       Gate    Net
  Cell:in->out     fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O           2   1.106   0.532  TBR_3_IBUF (TBR_3_IBUF)
   LUT4:I0->O          1   0.612   0.509  t_pari_xor000112 (t_pari_xor0001_map6)
   LUT2:I0->O          1   0.612   0.000  t_pari_xor000126 (t_pari_xor0001)
   LDC_1:D                 0.268          t_pari
  ----------------------------------------
  Total                 3.639ns (2.598ns logic, 1.041ns route)
                           (71.4% logic, 28.6% route)

================================================================
======
Timing constraint: Default OFFSET IN BEFORE for Clock 'CRL'
 Total number of paths / destination ports: 5 / 5
-------------------------------------------------------------------------
```

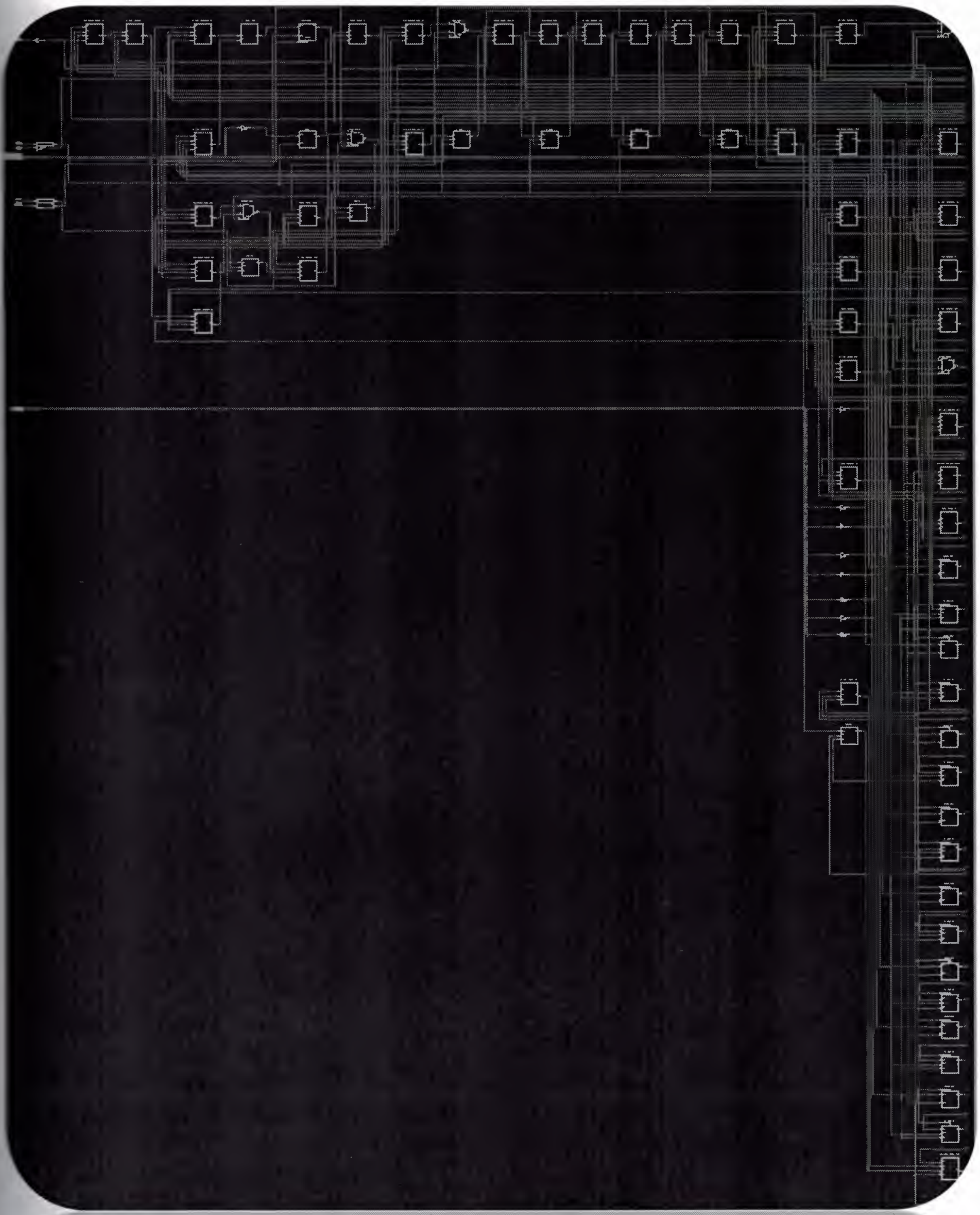
```
Offset:            1.731ns (Levels of Logic = 1)
 Source:            CTRLWORD<0> (PAD)
 Destination:        ctrl_word_0 (LATCH)
 Destination Clock: CRL falling

 Data Path: CTRLWORD<0> to ctrl_word_0
                          Gate     Net
   Cell:in->out     fanout   Delay   Delay  Logical Name (Net Name)
   ----------------------------------------  ------------
    IBUF:I->O            1   1.106   0.357  CTRLWORD_0_IBUF (CTRLWORD_0_IBUF)
    LD:D                 0.268          ctrl_word_0
   ----------------------------------------
   Total                  1.731ns (1.374ns logic, 0.357ns route)
                            (79.4% logic, 20.6% route)

=========================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'TRC'
 Total number of paths / destination ports: 4 / 3
-------------------------------------------------------------------------
Offset:            4.040ns (Levels of Logic = 1)
 Source:            TRE (FF)
 Destination:        TRE (PAD)
 Source Clock:      TRC rising

 Data Path: TRE to TRE
                          Gate     Net
   Cell:in->out     fanout   Delay   Delay  Logical Name (Net Name)
   ----------------------------------------  ------------
    FDE:C->Q             1   0.514   0.357  TRE (TRE_OBUF)
    OBUF:I->O                3.169          TRE_OBUF (TRE)
   ----------------------------------------
   Total                  4.040ns (3.683ns logic, 0.357ns route)
                            (91.2% logic, 8.8% route)

=========================================================================
CPU : 17.72 / 19.72 s | Elapsed : 18.00 / 20.00 s

-->

Total memory usage is 160540 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :   10 (   0 filtered)
Number of infos    :    0 (   0 filtered)
```

<< ....>>

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

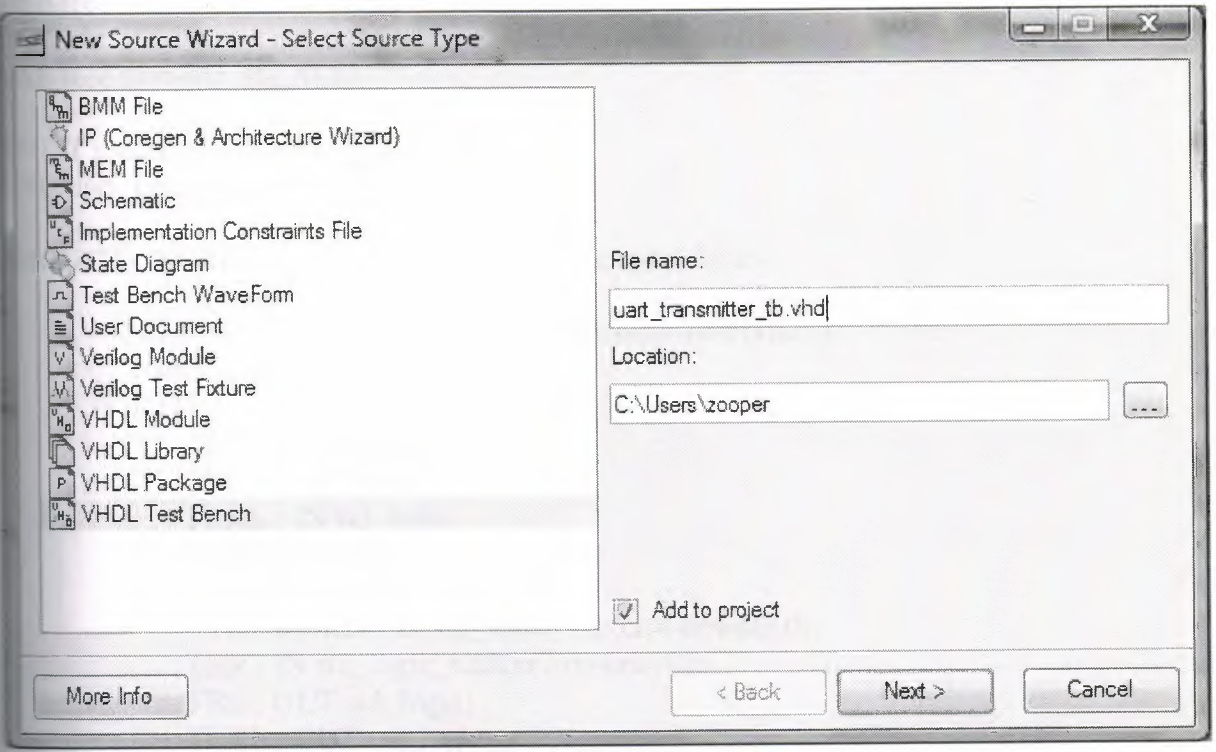


Block Diagram


Detailed Magnified Circuit

1.4 Writing Test Bench.

In the test bench generated 160 MHz. I wrote in the data in to the UART Transmitter and read it back to verify data can be written and read correctly. The simulation result is shown below.



File name: uart_transmitter_tb.vhd

Location: C:\Users\zooper

☑ Add to project

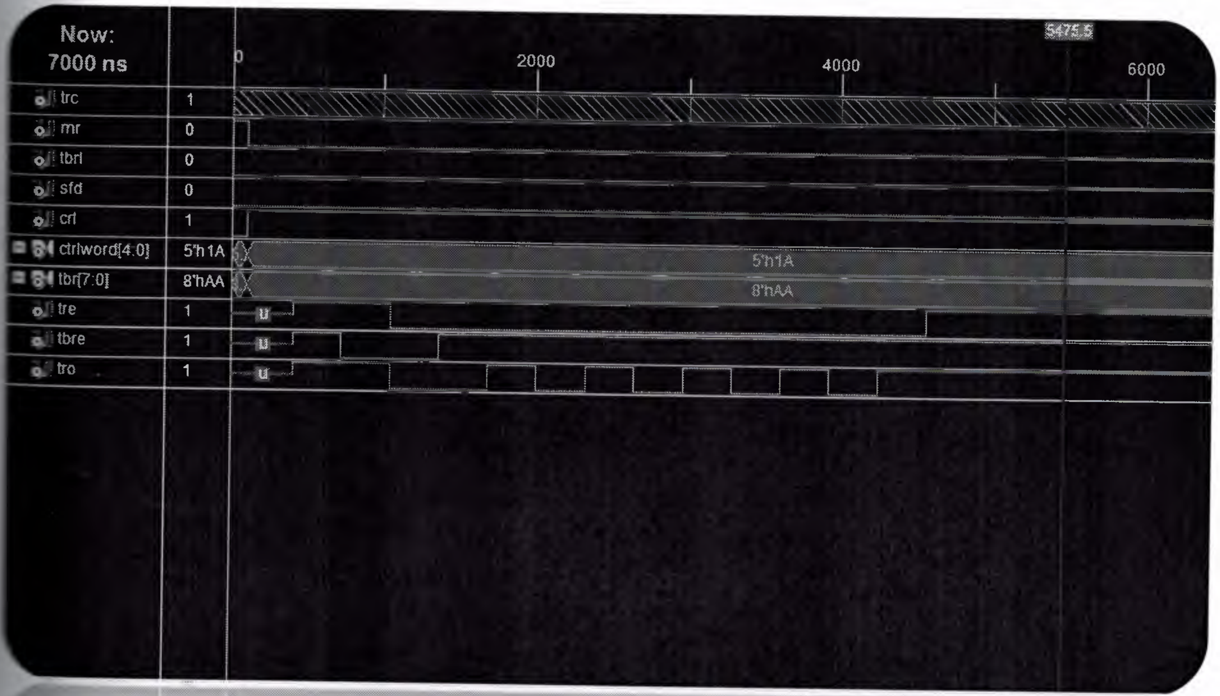
```
-------------------------------------------------------------------------
– Company:    Near East University
– Engineer:    Nadeem Hassan
–
– Create Date:   11:07:59 03/26/2008
– Design Name:   trans
– Module Name:   C:/Xilinx91i/xilinx/myprojects/uart_transmitter/uart_transmitter_tb.vhd
– Project Name:  uart_transmitter
– Target Device:
– Tool versions:
– Description:
–
– VHDL Test Bench Created by ISE for module: trans
–
– 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 uart_transmitter_tb_vhd IS
END uart_transmitter_tb_vhd;

ARCHITECTURE behavior OF uart_transmitter_tb_vhd IS

	-- Component Declaration for the Unit Under Test (UUT)
	COMPONENT trans
	PORT(
		TRC : IN std_logic;
		MR : IN std_logic;
		TBRL : IN std_logic;
		SFD : IN std_logic;
		CRL : IN std_logic;
		CTRLWORD : IN std_logic_vector(4 downto 0);
		TBR : IN std_logic_vector(7 downto 0);
		TRE : OUT std_logic;
		TBRE : OUT std_logic;
		TRO : OUT std_logic
		);
	END COMPONENT;

	--Inputs
	SIGNAL TRC :  std_logic := '0';
	SIGNAL MR :  std_logic := '1';
	SIGNAL TBRL :  std_logic := '0';
	SIGNAL SFD :  std_logic := '0';
	SIGNAL CRL :  std_logic := '0';
	SIGNAL CTRLWORD :  std_logic_vector(4 downto 0) := (others=>'0');
	SIGNAL TBR :  std_logic_vector(7 downto 0) := (others=>'0');

	--Outputs
	SIGNAL TRE :  std_logic;
	SIGNAL TBRE :  std_logic;
	SIGNAL TRO :  std_logic;

BEGIN

	-- Instantiate the Unit Under Test (UUT)
	uut: trans PORT MAP(
		TRC => TRC,
                MR => MR,
                TBRL => TBRL,
                SFD => SFD,
                CRL => CRL,
                CTRLWORD => CTRLWORD,
                TBR => TBR,
                TRE => TRE,
                TBRE => TBRE,
                TRO => TRO
        );


-- Please ensure that the constant PERIOD is defined prior to the
-- begin statement in the architecture. Refer to the PERIOD Constant Template
-- for more info.

TRC <= not TRC after 10 ns;


        tb : PROCESS
        BEGIN

                -- Wait 100 ns for global reset to finish
                wait for 100 ns;
MR<='0';
TBRL<='0';
CRL<='1';
CTRLWORD<="11010";
TBR<="10101010";
--SFD<='1';
wait for 500 ns;
--CRL<='0';
--MR<='0';

wait for 500 ns;
TBR<="10101010";
                -- Place stimulus here

                wait; -- will wait forever
        END PROCESS;

END;
```

<< ....>>

1.5 Simulating.

I have created a test bench to the UART Transmitter. In this test bench I wrote in to the UART Transmitter and read the data for final proof reading.



I am transmitting AA which is 1010 1010
So according to my simulated result i have

The code is like;

00101010101

Where,

Start 1	Start2	Parity	A	A	End
1	0	0	1010	1010	1

CHAPTER 2

Xilinx-ISE-

2.1. ISE General Information

2.1.1. Xilinx ISE Overview

The Integrated Software Environment (ISE™) is the Xilinx® design software suite that allows us to take our design from design entry through Xilinx device programming. The ISE Project Navigator manages and processes our design through the following steps in the ISE design flow.

2.1.2. Design Entry

Design entry is the first step in the ISE design flow. During design entry, we create our source files based on our design objectives. We can create our 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 our design. If we are working with a synthesized EDIF or NGC/NGO file, we can skip design entry and synthesis and start with the implementation process.

2.1.3. 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.

2.1.4. 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).

2.1.5. 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 the device.

2.1.6. 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. Xilinx ISE Overview Architecture Support.

2.1.7. Architecture Support.

The ISE™ software supports the following device families.

FPGAs	CPLDs
Spartan™-II	CoolRunner™ XPLA3
Spartan-IIE	CoolRunner-II
Spartan-3	XC9500™
Spartan-3A	XC9500XL
Spartan-3E	XC9500XV
Spartan-3L	
Virtex™	
Virtex-E	
Virtex-II	
Virtex-II Pro	
Virtex-II Pro X	
Virtex-4	
Virtex-5	

Operating System Support.

The ISE™ software is supported on the following operating systems.

Operating System	Versions
Windows®	• Windows XP® Professional 32-bit/64-bit • Windows 2000® Professional
Solaris®	• Solaris 9 • Solaris 10
Linux	• Red Hat® Enterprise WS 3.0 32-bit/64-bit • Red Hat Enterprise WS 4.0 32-bit/64-bit

2.2. Using Project Navigator

2.2.1. Project Navigator Overview

Project Navigator organizes our design files and runs processes to move the design from design entry through implementation to programming the targeted Xilinx® device. Project Navigator is the high-level manager for our Xilinx FPGA and CPLD designs, which allows us to do the following:

Add and create design source files, which appear in the Sources window

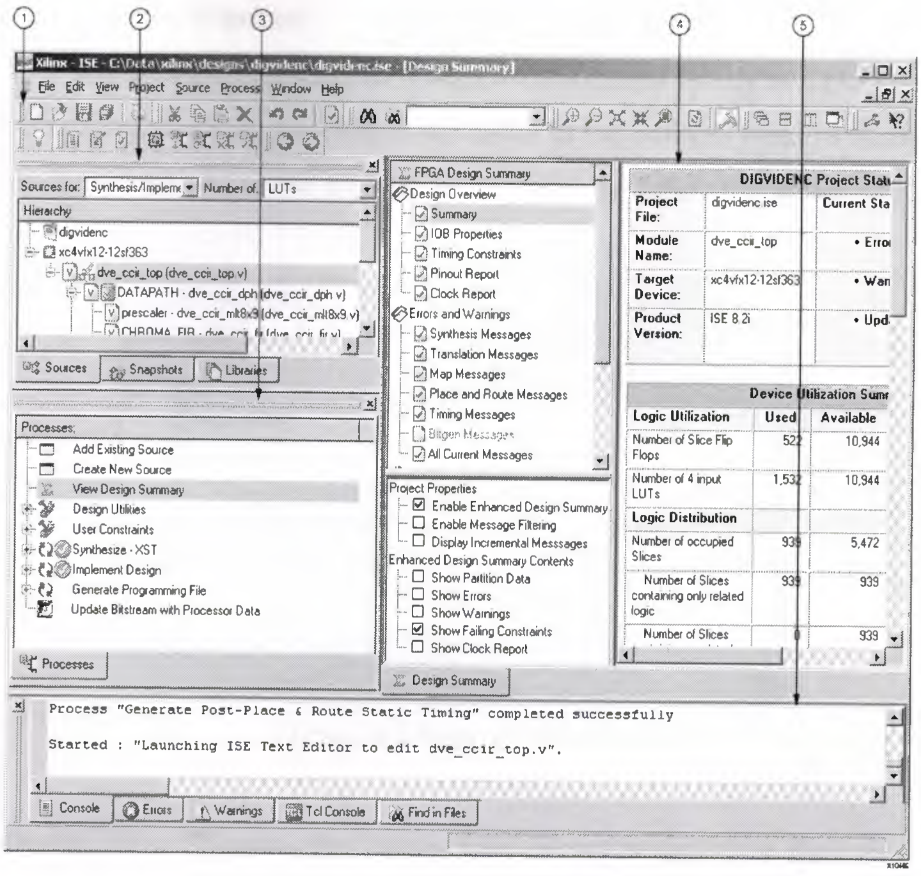
Modify the source files in the Workspace

Run processes on the source files in the Processes window

View output from the processes in the Transcript window

2.2.2. Project Navigator Main Window

The following figure shows the Project Navigator main window, which allows to manage our design starting with design entry through device configuration.


1
2
3
4
5
Xilinx - ISE - C:\Data\xilinx\designs\digvidenc\digvidenc.ise - [Design Summary]
File Edit View Project Source Process Window Help
Sources for: Synthesis/Impleme
Number of: LUTs
Hierarchy
Sources
Snapshots
Libraries
Processes
FPGA Design Summary
Design Summary
Console
Errors
Warnings
Tcl Console
Find in Files

Hierarchy:
- digvidenc
- xc4vfx12-12sf363
  - dve_ccir_top (dve_ccir_top.v)
    - DATAPATH - dve_ccir_dph (dve_ccir_dph.v)
      - prescaler - dve_ccir_mlt8x9 (dve_ccir_mlt8x9.v)
      - CHROMA_FIR - dve_ccir_fir (dve_ccir_fir.v)

Processes:
- Add Existing Source
- Create New Source
- View Design Summary
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Update Bitstream with Processor Data

FPGA Design Summary
- Design Overview
  - Summary
  - IOB Properties
  - Timing Constraints
  - Pinout Report
  - Clock Report
- Errors and Warnings
  - Synthesis Messages
  - Translation Messages
  - Map Messages
  - Place and Route Messages
  - Timing Messages
  - Bitgen Messages
  - All Current Messages

Project Properties
- [✓] Enable Enhanced Design Summary
- [ ] Enable Message Filtering
- [ ] Display Incremental Messsages

Enhanced Design Summary Contents
- [ ] Show Partition Data
- [ ] Show Errors
- [ ] Show Warnings
- [✓] Show Failing Constraints
- [ ] Show Clock Report

DIGVIDENC Project Stat		
Project File:	digvidenc.ise	Current Sta
Module Name:	dve_ccir_top	• Erro
Target Device:	xc4vfx12-12sf363	• Warn
Product Version:	ISE 8.2i	• Upd

Device Utilization Sum		
Logic Utilization	**Used**	**Available**
Number of Slice Flip Flops	522	10,944
Number of 4 input LUTs	1,532	10,944
Logic Distribution		
Number of occupied Slices	939	5,472
Number of Slices containing only related logic	939	939
Number of Slices	0	939

```
Process "Generate Post-Place & Route Static Timing" completed successfully

Started : "Launching ISE Text Editor to edit dve_ccir_top.v".
```

Figure 1.1 Main Window of Project Navigator

1 Toolbar

2 Sources window

3 Processes window

4 Workspace

5 Transcript window

2.2.3. Using the Sources Window

The first step in implementing our design for a Xilinx® FPGA or CPLD is to assemble the design source files into a project. The Sources tab in the Sources window shows the source files us create and add to our project, as shown in the following figure.

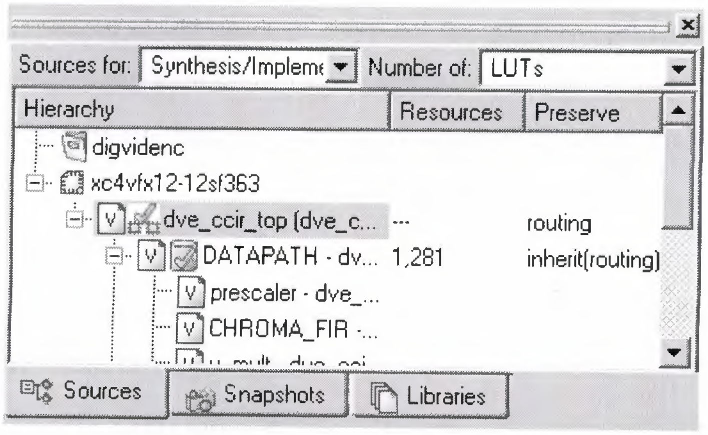


Figure 1.2 Sources Window

The Design View ("Sources for") drop-down list at the top of the Sources tab allows us to view only those source files associated with the selected Design View (for instance, Synthesis/Implementation). The "Number of" drop-down list, Resources column, and Preserve column are available for designs that use Partitions.

The Sources tab shows the hierarchy of our design. We can collapse and expand the levels by clicking the plus (+) or minus (-) icons. Each source file appears next to an icon that shows its file type. The file we select determines the processes available in the Processes window. We can double-click a source file to open it for editing in the Workspace. For information on the different file types, you can change the project properties, such as the device family to target, the top-level module type, the synthesis tool, the simulator, and the generated simulation language.

Depending on the source file and tool we are working with, additional tabs are available in the Sources window:

- Always available: Sources tab, Snapshots tab, Libraries tab
- Constraints Editor: Timing Constraints tab
- Floorplan Editor: Translated Netlist tab, Implemented Objects tab
- Schematic Editor: Symbols tab
- Technology Viewer: Design tab
- Timing Analyzer: Timing tab

2.2.4. Using the Processes Window

The Processes tab in the Processes window allows us to run actions or "processes" on the source file we select in the Sources tab of the Sources window. The processes change according to the source file we select. The Process tab shows the available processes in a hierarchical view. We can collapse and expand the levels by clicking the plus (+) or minus (-) icons. Processes are arranged in the order of a typical design flow: project creation, design entry, constraints management, synthesis, implementation, and programming file creation.

Depending on the source file and tool we are working with, additional tabs are available in the Processes window:

- Always available: Processes tab
- Floorplan Editor: Design Objects tab, Implemented - Selection tab
- ISE Simulator: Hierarchy Browser tab
- Schematic Editor: Options tab
- Timing Analyzer: Timing Objects tab

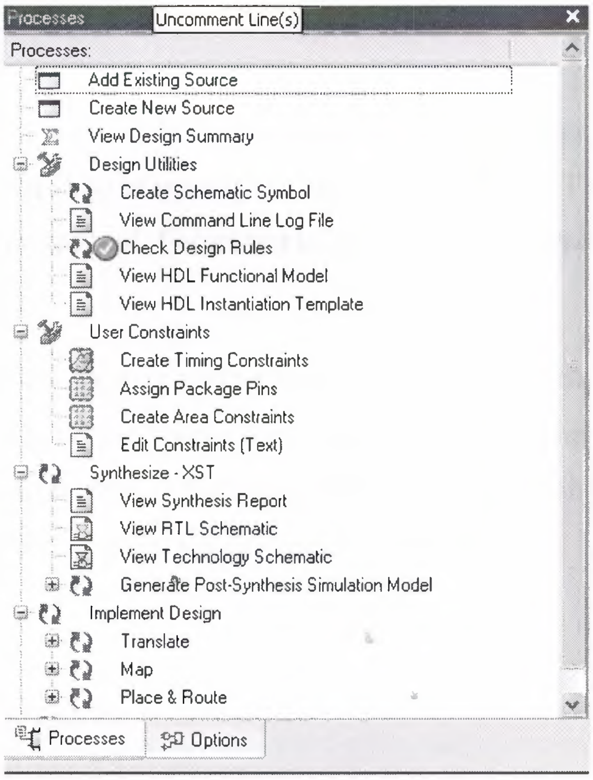


Figure 1.3 Process Window

2.2.5. Process Types

The following types of processes are available as we work on our design:

- Tasks

When we run a task process, the ISE software runs in "batch mode," that is, the software processes our source file but does not open any additional software tools in the Workspace. Output from the processes appears in the Transcript window.

- Reports

Most tasks include report sub-processes, which generate a summary or status report, for instance, the Synthesis Report or Map Report. When we run a report process, the report appears in the Workspace.

- Tools

When we run a tools process, the related tool launches in standalone mode or appears in the Workspace where we can view or modify our design source files. The icons for tools processes vary depending on the tool. For example, the Timing Analyzer icon is shown above.

2.2.6. Process Status

As we work on our design, we may make changes that require some or all of the processes to be rerun. For example, if we edit a source file, it may require that the Synthesis process and all subsequent process be rerun. Project Navigator keeps track of the changes we make and shows the status of each process with the following status icons:

Running

This icon shows that the process is running.

- Up-to-date

This icon shows that the process ran successfully with no errors or warnings and does not need to be rerun. If the icon is next to a report process, the report is up-to-date; however, associated tasks may have warnings or errors. If this occurs, we can read the report to determine the cause of the warnings or errors.

Warnings reported ⚠

This icon shows that the process ran successfully but that warnings were encountered.

Errors reported ⊗

This icon shows that the process ran but encountered an error.

- Out-of-Date ?

This icon shows that we made design changes, which require that the process be rerun. If this icon is next to a report process, we can rerun the associated task process to create an up-to-date version of the report.

No icon

If there is no icon, this shows that the process was never run.

2.2.7. Running Processes

To run a process, we can do any of the following:

Double-click the process, right-click while positioned over the process, and select Run from the popup menu, as shown in the following figure.

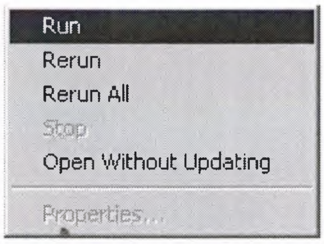


Figure 1.3 Pop-Up Menu for Run

Select the process, and then click the Run toolbar button.

- To run the Implement Design process and all preceding processes on the top module for the design, select Process > Implement Top Module, or click the Implement Top Module toolbar button.

When we run a process, Project Navigator automatically processes our design as follows: Automatically runs lower-level processes.

When we run a high-level process, Project Navigator runs associated lower-level processes or sub-processes. For example, if we run Implement Design for our FPGA design, all of the following sub-processes run: Translate Map, and Place & Route.

- Automatically runs preceding processes

When we run a process, Project Navigator runs any preceding processes that are required, thereby "pulling" our design through the design flow. For example, to pull our design through the entire flow, double-click Generate Programming File.

Automatically runs related processes for out-of-date processes.

If we run an out-of-date process, Project Navigator runs that process and any related processes required to bring that process up to date. It does not necessarily run all preceding processes. For example if we change our UCF file, the Synthesize process remains up to date, but the Translate process becomes out of date. If we run the Map process, Project Navigator runs Translate but does not run Synthesize. For more information on running processes, including additional Process menu commands.

2.2.8. Setting Process Properties

Most processes have a set of properties associated with them. Properties control specific options, which correspond to command line options. When properties are available for a process, we can right-click while positioned over the process and select Properties from the popup menu, as shown in the following figure.

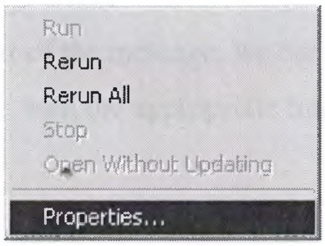


Figure 1.4 Pop-Up Menu for Properties

When we select Properties, a Process Properties dialog box appears, with standard properties that we can set. The Process Properties dialog box differs depending on the process we select.

After we become familiar with the standard properties, we can set additional, advanced properties in the Process Properties dialog box; however, setting these options is *not* recommended if we are just getting started with using the ISE software. When we enable the advanced properties, both standard and advanced properties appear in the Process Properties dialog box.

2.2.9. Using the Workspace

When we open a project source file, we open the Language Templates, or run certain processes, such as viewing reports or logs, the corresponding file or view appears in the Workspace. We can open multiple files or views at one time. Tabs at the bottom of the Workspace show the names for each file or view. A tab is clicked to bring it to the front.

To open a file or view in a standalone window outside of the Project Navigator Workspace, the Float toolbar button is used. To dock a floating window, the Dock toolbar button is used.

Float

Dock

The Dock toolbar button is only available from the floating window.

2.2.10. Using the Transcript Window

The Console tab of the Transcript window shows output messages from the processes we run. If a line number appears as part of the message, we can right-click the message and select Goto Source to open the source file with the appropriate line number highlighted.

Warning ⚠

- Error ⊗

Depending on the source file and tool we are working with, additional tabs are available in the Transcript window:

- Always available: Console tab, Errors tab, Warnings tab, Tcl Console tab, Find in Files tab.

- ISE Simulator: Simulation Console tab.

RTL and Technology Viewers: View by Name tab, View by Category tab.

2.2.11. Using the Toolbars

Toolbars provide convenient access to frequently used commands. To execute a command a toolbar button click once on. To see a short popup description of a toolbar button, the mouse pointer is holding over the button for about two seconds. A longer description appears in the status bar at the bottom of the main window.

For Help on a toolbar button, the Help toolbar button is clicked, and then the toolbar button is clicked for which we want Help. For more information on getting Help, we should see Using Xilinx Help.

2.3. Working with Projects

2.3.1. Creating a Project

Project Navigator allows us to manage our FPGA and CPLD designs using an ISE™ project, which contains all the files related to our design. First, we must create a project and then add source files. With our project open in Project Navigator, we can view and run processes on *all the files in our design. Project Navigator provides a wizard to help us create a* new project, as follows.

To Create a Project

Select File > New Project.

2. In the New Project Wizard Create New Project page, steps are as follows:
   - In the Project Name field, we enter a name for the project. It follows the naming conventions described in File Naming Conventions.
   - *In the Project Location field, we enter the directory name or browse to the* directory.
   - In the Top-Level Source Type drop-down list, we select one of the following:

HDL

We select this option if our top-level design file is a VHDL, Verilog, or ABEL (for CPLDs) file. An HDL Project can include lower-level modules of different file types, such as other HDL files, schematics, and "black boxes," such as IP cores and EDIF files.

Schematic

We select this option if our top-level design file is a schematic file. A schematic project can include lower-level modules of different file types, such as HDL files, other schematics, and "black boxes," such as IP cores and EDIF files. Project Navigator automatically converts any schematic files in our design to structural HDL before implementation; therefore, we *must* specify a synthesis tool when working with schematic projects, as described in step 5.

EDIF

Select this option if you converted your design to this file type, for example, using a synthesis tool. Using this file type allows you to skip the Project Navigator synthesis process and to start with the implementation processes.

NGC/NGO

This option is selected if the design is converted to this file type, for example, using a synthesis tool. Using this file type allows us to skip the Project Navigator synthesis process and start with the implementation processes.

3. Click Next.
4. If we are creating an HDL or schematic project, we skip to the next step. If we are creating an EDIF or NGC/NGO project, we should do the following in the Import EDIF/NGC Project page:

   In the Input Design field, we enter the name of the input design file, or browse to the file and it is selected.

   Select Copy the input design to the project directory to copy our file to the project directory. If we do not select this option, our file is accessed from the remote location.

   In the Constraint File field, we should enter the name of the constraints file, or browse to the file and select it.

Select Copy the constraints file to the project directory to copy our file to the project directory. If we do not select this option, our file is accessed from the remote location.

Click Next.

5. In the Device Properties page, we should set the following options. These settings affect other project options, such as the types of processes that are available for our design.

Product Category

Family

To target a Spartan-3L™ device, Spartan-3™ should be selected as the family. When creating an EDIF project, the device family information is read from our EDIF project file, and changing the device family is *not* recommended.

Device

To target a Spartan-3L device, device that ends in l should be selected, such as xc3s2000l.

Package

Speed

Top-Level Source Type

This is automatically set.

Synthesis Tool

We select one of the following synthesis tools and the HDL language for our project. VHDL/Verilog is a mixed language flow. If we plan to run behavioral simulation, our simulator must support multiple language simulation.

XST (Xilinx® Synthesis Technology)

XST is available with ISE Foundation™ software installations. It supports projects that include schematic design files and projects that include mixed language source files, such as VHDL and Verilog sources files in the same project. For more information, see.

Synplify and Synplify Pro (Synplicity®, Inc.)

The Synplify® software does *not* support projects that include mixed language source files. The Synplify Pro® software supports projects that include mixed language source files, such as VHDL and Verilog sources' files in the same project. The Synplify and Synplify Pro software do *not* support projects that include schematic design files.

LeonardoSpectrum (Mentor Graphics®, Inc.)

The LeonardoSpectrum™ software supports projects that include schematic design files. It does *not* support projects that include mixed language source files, such as VHDL and Verilog sources files in the same project.

Precision (Mentor Graphics®, Inc.)

The Precision® software supports projects that include schematic design files and projects that include mixed language source files, such as VHDL and Verilog sources files in the same project.

Simulator

We select one of the following simulators and the HDL language for simulation. The language we select determines the default language in which to generate simulation netlists and other generated files that affect simulation. We can also select the language in which to generate files by setting process properties as described in Setting Process Properties.

ISE Simulator (Xilinx®, Inc.)

This simulator allows us to run integrated simulation processes as part of our ISE design flow.

ModelSim (Mentor Graphics®, Inc.)

We can run integrated simulation processes as part of our ISE design flow using any of the following ModelSim® editions: ModelSim Xilinx Edition (MXE), ModelSim MXE Starter, ModelSim PE, or ModelSim SE™.

NC-Sim (Cadence®, Inc.)

The NC-Sim simulator is not integrated with ISE and must be run standalone.

VCS (Synopsys®, Inc.)

The VCS® simulator is not integrated with ISE and must be run standalone.

Others

We select this option if we do not have ISE Simulator or ModelSim installed or if we want to run simulation outside of Project Navigator. This instructs Project Navigator to disable the integrated simulation processes for our project.

Enable Enhanced Design Summary

We select this option to show the number of errors and warnings for each of the Detailed Reports in the Design Summary.

Enable Message Filtering

We select this option to show the number of messages we filtered in the Design Summary. We must enable this option, filter messages, and then run the software to show the number of filtered messages.

Display Incremental Messages

We select this option to show the number of new messages for the most recent software run in the Design Summary. We must enable this option and then run the software to show the number of new messages.

6. If we are creating an EDIF or NGC/NGO project, we should skip to step 8. If we are creating an HDL or schematic project, we click Next, and optionally, we create a new source file for our project in the Create New Source page. We can only create one new source file while creating a new project. We can create additional new sources after our project is created.

7. Click Next, and optionally, add existing source files to our project in the Add Existing Sources page.

8. Click Next to display the Project Summary page.

9. Click Finish to create the project.

If we prefer, we can create a project using the New Project dialog box instead of the New Project Wizard, as described above. To use the New Project dialog box, we should deselect the Use new project wizard option in the ISE General Options page of the Preferences dialog box.

What to Expect

Project Navigator creates the project file, project_name.ise, in the directory we specified. All source files related to the project appear in the Project Navigator Sources tab. Project Navigator manages our project based on the project properties (top-level module type, device type, synthesis tool, and language) we selected when we created the project. It organizes all the parts of our design and keeps track of the processes necessary to move the design from design entry through implementation to programming the targeted Xilinx device. For information on changing project properties, we see Changing Project, Source, and Snapshot Properties.

What to Do Next

We can perform any of the following:

To create and add source files to our project.

To add existing source files to our project.

To run processes on our source files.

2.3.2. Working with Snapshots

A snapshot is a read-only copy of the current project that allows us to do the following:

- To save different versions of our project for comparison
- Revert to a previously saved version of our project

A snapshot includes all of the files and directories in the project directory, such as source files, implementation files, and process files. Archiving is similar to using snapshots. However, archives are stored in ZIP files and cannot be opened within Project Navigator, without first being unzipped outside of Project Navigator.

What to Do First

We open a project in Project Navigator.

To Create a Snapshot

- Select Project > Take Snapshot.
- In the Take a Snapshot of the Project dialog box, we enter a name for our snapshot in the Snapshot Name field.
- In the Comment field, we add any notes related to this version of our project. Comments are optional.
- Click OK.

If our project includes source files that resides outside of the project directory, these remote source files are copied to a remote sources directory in the snapshot.

To View a Snapshot

- In the Sources window, we click the Snapshots tab.
- In the Snapshots tab, we select the desired snapshot.
- Select Source > Open to view the hierarchy of our design, or click the plus (+) or minus (-) icons to collapse and expand the levels. We can double-click a source file to open it for viewing in the Project Navigator Workspace.

To Rename a Snapshot or to Add Comments

- In the Snapshots tab, we select the top-level directory of the snapshot.
- Select Source > Properties.
- In the Snapshot Properties dialog box, we change the snapshot name or add comments.
- Click OK.

To Remove a Snapshot

Caution! Snapshots are deleted from the disk when removed from a project.

- In the Snapshots tab, we select the top-level directory of the snapshot to delete.
- Select Source > we remove, or press the Delete key on the keyboard.
- Click Yes to delete the snapshot directory from our disk.

To Replace the Project with a Snapshot

Caution! This procedure replaces the project with a saved snapshot version. Our project is *overwritten* unless we take a snapshot of it before we replace it.
The following procedure describes how to revert to a previously saved version of our project:

- In the Snapshots tab, we select the top-level directory of the snapshot.
- *Select Project > Make Snapshot Current.*
- Because this procedure overwrites our current project, we are prompted to save our current project as a snapshot. Click Yes to save the current project as a snapshot, or No to overwrite the current project without saving it as a snapshot.

The current project in the Sources tab is replaced by the selected snapshot. In addition, the original snapshot also remains intact in the snapshots directory. If our snapshot included remote sources in a remote sources directory, the remote sources directory is automatically copied to the project directory. However, the restored project maintains the links to the remote location of the source files, not to the files in the remote sources directory. If we want to work with the snapshot version of the remote source files, we must manually copy them from the remote sources directory to the remote location.

What to Expect

Each snapshot is stored in a separate directory named with the Snapshot Name we specified. These directories are located in the snapshots directory under the top-level directory for the associated project. In the Project Navigator, each snapshot is shown in the Snapshots tab.

2.4. Working with Projects Source File

2.4.1. Creating a Source File

A source file is any file that contains information about a design. Project Navigator provides a wizard to help us create new source files for our project. If we are targeting a Spartan-3A or Virtex-5 device, we can use the New Source Wizard to pre-assign package pins for an empty project. For details, Pre-Assigning Package Pins in the New Source Wizard.

What to Do First

Open a project in Project Navigator.

To Create a Source File

- Select Project > New Source.
- In the New Source Wizard, we select the type of source we want to create. Different source types are available depending on our project properties (top-level module type, device type, synthesis tool, and language). Some source types launch additional tools to help us create the file, as described in Source File Types.
- We enter a name for the new source file in the File Name field. Then we follow the naming conventions described in File Naming Conventions.

- In the Location field, we enter the directory name or browse to the directory.
- We select Add to Project to automatically add this source to the project. State machines created with StateCAD cannot be automatically added to the project. We must add them manually.
- Click Next.
- If we are creating a source file that needs to be associated with an existing source file, we should select the appropriate source file, and click Next. If this does not apply, skip to the next step.
- In the New Source Information window, we can read the summary information for the new source, and we click Finish.

After we click Finish, the New Source wizard closes. In some cases, a related tool is launched in which we can finish creating our file. After the source file is created, it appears in the Project Navigator Sources tab. If we selected Add to Project when creating the source file, the file is automatically added to the project.

2.4.2. Source File Types

The following table shows the source file types that appear in the Project Navigator Sources tab. Available source types vary depending on our project properties (top-level module type, device type, synthesis tool, and language). The last column describes what to expect when creating the file with the New Source wizard and, if applicable, includes the tool launched when using the New Source wizard or when editing the file from Project Navigator.

Table 1.3 Source File Types

File Type	Extension	Icon	Description	New Source Wizard Behavior/Tool Launched
ABEL Test Vector	.abv	📄	Describes input stimulus and expected outputs for logic simulation of ABEL design code.	Associates the file with the top-level module and opens the empty vector file in the text editor we specify in the Editor Options page of the Preferences dialog box.
ABEL-HDL	.abl	📄	Contains ABEL	Allows you to specify your pin

Module			design code.	names and opens the file in the text editor we specify in the Editor Options page of the Preferences dialog box.
Block RAM Memory Map (BMM File)	.bmm		Used in PowerPC™ and MicroBlaze™ processor designs to describe the organization of Block RAM memory. **Note** Only one BMM Module is allowed per project.	Opens the file in the text editor we specify in the Editor Options page of the Preferences dialog box. The CPU executable code is automatically inserted in the configuration file during design implementation.
Chipscope Definition and Connection (CDC File)	.cdc		Contains generic information about the trigger and data ports of the ChipScope™ core.	Adds the file to the project. Double-click the CDC file in the Sources tab to run the implementation process and launch the ChipScope Pro™ Core Inserter. For details, see the ChipScope Pro Debugging Overview. ChipScope Pro must be installed for this source type to be available.
Electronic Data Interchange Format (EDIF)	.edn, .edf, .edif, .sedif		Specifies the design netlist in an industry standard file format.	N/A Must be generated by a third-party design entry tool and added to the project. We can only add an EDIF file as a top-level module, not as a lower-level module. If you are using hierarchical EDIF files, lower-level EDIF files are

				automatically processed during the implementation process.
ELF	.elf		Contains an executable CPU code image. Only one ELF file is allowed per project.	N/A Must be generated by the Data2MEM command line tool and added to the project.
Embedded Processor	.xmp		Contains predefined logic functions.	Launches the Xilinx Platform Studio in which we can define the embedded processor system portion of our design. For details, see the Embedded Development Kit Documentation.
I/O Pin Assignments	.ucf and .v or .vhd		Allows us to create and add a UCF file with I/O pin data and add a template HDL file to an empty project.	The I/O pin assignment data displays in the Floorplan View and Package View in the Workspace. Pin assignments are saved in the UCF, which is added to the project and associated with the template HDL file. The output HDL file displays in the ISE Text Editor in the Workspace. This feature is only supported for Spartan™-3A and Virtex™-5 devices. For details, Pre-Assigning Package Pins in the New Source Wizard.
Implementation Constraints File also known as User Constraints File (UCF)	.ucf		Contains user-specified logical constraints.	Adds the file to the project. Double-click the UCF file in the Sources tab, or double-click a Constraints Entry process in the Processes tab to open the

				file. For details, see Constraints Entry Methods.
IP (Architecture Wizard)	.xaw		Contains predefined logic functions that configure architecture features or modules.	Launches one of the Xilinx Architecture Wizards in which we can define our IP. For details, see Working with Architecture Wizard IP.
IP (CoreGen)	.xco		Contains predefined logic functions.	Launches the Xilinx CORE Generator™ software in which we can define your IP. For details, see Working with CORE Generator IP.
Memory Definition (MEM File)	.mem		Used to define the contents of memory (RAMB4 and RAMB16). Only one MEM file is allowed per project.	Opens the file in the text editor we specify in the Editor Options page of the Preferences dialog box. The CPU executable code is automatically inserted in the configuration file during design implementation.
Project	.ise		Contains process property settings, status, and information for managing the ISE™ project.	N/A
Schematic	.sch		Contains a schematic design.	Opens the schematic file in the Project Navigator Workspace. For details, see the Schematic Overview.
State diagram	.dia		Contains a state diagram file.	Launches StateCAD in which we can define your state diagram. For details, see Working with State Machines.

Targeted device, package, and speed grade	N/A		Shows the targeted device, package, and speed grade.	N/A
Test Bench Waveform	.tbw		Contains a graphical representation of a test bench that can be converted to an HDL test bench or test fixture.	Prompts you to associate the file with a source and opens the Test Bench Waveform Editor in the Project Navigator Workspace with the signals populated. For details, see the ISE Simulator Help. This file is for use with the Xilinx® Test Bench Waveform Editor only.
Undefined	N/A		Contains an instantiated module that has not been added to the ISE project but is referenced by a source file in the ISE project.	N/A
User Document	.doc, .txt, .wri		Contains user information that is not implemented with the project, for example, supporting documentation.	N/A Must be added to the project.
Verilog Module	.v		Contains Verilog design code.	Opens the file in the text editor we specify in the Editor Options page of the Preferences dialog box.
Verilog Test Fixture	.v		Defines the stimulus to the ports of an HDL file.	Prompts you to associate the file with a Verilog source module and then opens a skeleton test bench file in the

				text editor you specify in the Editor Options page of the Preferences dialog box.
VHDL Library	.vhd		Contains a collection of VHDL packages.	Adds a new directory to the `vhdl` library directory in the Libraries tab.
VHDL Module	.vhd		Contains VHDL design code.	Opens the file in the text editor we specify in the Editor Options page of the Preferences dialog box.
VHDL Package	.vhd		Contains definitions, macros, sub-routines, supplemental types, subtypes, constants, functions, and other files.	Opens the file in the text editor we specify in the Editor Options page of the Preferences dialog box.
VHDL Test Bench	.vhd		Defines the stimulus to the ports of an HDL file.	Prompts we to associate the file with a VHDL source and then opens a skeleton test bench file in the text editor we specify in the Editor Options page of the Preferences dialog box.

2.4.3. Adding a Source File to a Project

Project Navigator allows us to add an existing source file to a project. The source file can reside in the project directory or in a remote directory. If we generated our source file using the New Source wizard and selected Add to Project, we do not need to add the source file to our project; it is automatically part of our project. The only exception is state diagrams, which must be added manually.

If we want to copy a source file from a remote directory to our project directory and add it to our project, use the Add Copy of Source command instead, as described in Adding a Copy of a Source File to a Project. If we are working with CORE Generator™ or Architecture Wizard IP, we must use the Add Copy of Source command to copy the IP core and associated files that reside in a remote directory to our local project directory. The files will not simulate or implement correctly if we add them as remote source files.

What to Do First

Open a project in Project Navigator.

To Add a Source File to a Project

- Select Project > Add Source.

Alternatively, we can double-click Add Existing Source in the Processes tab.

- In the Add Existing Sources dialog box, we browse to the source file and we select it.
- Click Open.
- In the Adding Source Files dialog box, we select the Design View in which we want the source file to appear.

If we want to change the Design View association after the source file has been added, select the source file in the Sources tab, and then select Source > Properties.

- Click OK.

What to Expect

The source file is added to your project, and the file appears as part of the design hierarchy in the Sources tab. If you added a remote source, the directory path appears with the file name.

If the source file you added refers to files that have *not* been added to the project, the file names appear in the design hierarchy as undefined files ?. You must add the referenced files to the project for the ISE software to track changes to the files.

We cannot add fixed netlist IP to an ISE project. However, you must include the IP cores in your project directory to use them in your project.

2.4.4. Adding a Copy of a Source File to a Project

Project Navigator allows us to copy a source file from a remote directory to our project directory and then, add it to our project as follows. If we want to leave the source file in the remote directory and add it to our project, see Adding a Source File to a Project.

What to Do First

Open a project in Project Navigator.

To Add a Copy of a Source File to a Project

Select Project > Add Copy of Source.

In the Add Existing Sources dialog box, browse to the source file and select it.

Click Open.

In the Adding Source Files dialog box, select the Design View in which we want the source file to appear.

If we want to change the Design View association after the source file has been added, select the source file in the Sources tab, and then select Source > Properties.

Click OK.

What to Expect

The copy of the source file is placed in the project directory and is added to our project. The file appears as part of the design hierarchy in the Sources tab.

If the source file we added refers to files that have not been added to the project, the file names appear in the design hierarchy as undefined files. We must add the referenced files to the project for the ISE software to track changes to the files.

Note: Some support files, such as CORE Generator wrapper and symbol files, are automatically copied to the project directory when we copy the source file.

2.4.5. Editing a Source File

After we create a source file, we can edit it using the ISE™ software.

What to Do First

Open a project in Project Navigator.

To Edit a Source File

In the Sources tab, select a Design View from the drop-down list. Double-click the source file. Edit the file in the tool that appears. Each source type launches a different tool to help us to edit the file, as described in Source File Types. See the Help provided with each tool for detailed information.

Step3- What to Expect

After we edit the source file, you may need to rerun certain processes to bring the project up-to-date, as described in Running and Stopping Processes.

2.4.6. Removing Files from a Project

We can remove files from a project that we no longer need. The file is removed from the project, but is not deleted from our disk.

Caution! When we remove snapshots, the snapshot directory *is* deleted from the disk. For details, see Working with Snapshots.

What to Do First

Open a project in Project Navigator.

To Remove a Source File from a Project.

In the Sources tab, select a Design View from the drop-down list.

Select the file to remove.

Select Source > Remove, or press the Delete key on the keyboard.

Click yes to remove the file from our project.

What to Expect

The file is removed from our project. Removing a source file may alter the status for certain processes in the Processes tab. We may need to rerun certain processes to bring the project up-to-date, as described in Running and Stopping Processes.

2.4.7. Working with VHDL Libraries

VHDL library files allow us to store commonly used packages and entities that we can use in our VHDL files. A VHDL package file contains common design elements that we can use in the VHDL file source files that make up our design. We use the following procedures to create VHDL libraries and package files and to move package files from one library to another.

What to Do First

Open a project in Project Navigator.

To Reference a VHDL Library or Package File.

To reference a VHDL library or package file in our VHDL file, we do the following:

- To reference a package file, include a use statement.
- To reference a library file, include a use and a library statement. When referencing a library, you must declare both the library and package name.

To Create a VHDL Library

The ISE software provides a work library. However, we can create our own libraries as described in Creating a Source File. We select VHDL Library as our source type.

After we create the VHDL library, the new library appears in the vhdl library. Click the Libraries tab to view the vhdl library.

Note: The Libraries tab also contains a verilog library in which you can store Verilog files for our reference. However, if we want to store commonly used modules and components that we can use in our Verilog files, see Working with Verilog Header Files.

To Create a VHDL Package File

We can create a VHDL package file as described in Creating a Source File. Select VHDL Package as our source type.

After we create the VHDL package file, the new file appears at the top of the Sources tab. By default, the new VHDL package file is added to the work library. Click the Libraries tab to view the work library. To move the file to a different library, see the following procedure:

To Move a VHDL Package File to a Library

To move a VHDL package file from one library to another, for example, from the default work library to a library that we created, the following should be done:
In the Sources window, click the Libraries tab.
Select the VHDL package file to move.

Note: We can only move package files with the .vhd extension. We cannot move any .vhd file. By default, new VHDL package files display at the bottom of the work library hierarchy.
Select Source > Move to Library.
In the Choose Library dialog box, select the library to move the file to.
Click OK.

2.5. Working with Processes

2.5.1. Running and Stopping Processes

In the Processes tab, we can run processes on our selected source file. We can run a task, generate a report, or launch a tool. We can also stop a process while it is running.

What to Do First

Open a project in Project Navigator.

To Run a Process

In the Sources tab, select a Design View from the drop-down list.
Select the source file to process.

The source file we select affects the processes that appear in the Processes tab; only the processes that apply to the selected source are shown.
In the Processes tab, we select a process.
From the Process menu, we select one of the following commands:
Run to run the selected process and any preceding processes that are out of date.
Alternatively, we can double-click the process to run it.
Rerun to force a run on the selected up-to-date process.

Rerun all to force a run on the selected up-to-date process and all processes that precede the selected process.

Open without Updating to open a file for an out-of-date task or to open an out-of-date report for investigative purposes.

We can also right-click a process and select one of these commands.

To Stop a Process

To stop the currently running process, select Process > Stop.

Stopping a process is not always immediate; some processes may proceed until a suitable stopping point is reached.

What to Expect

One of the following status icons appears next to the process in the Processes tab:

Running ◔

This icon shows that the process is running.

Up-to-date ✓

This icon shows that the process ran successfully with no errors or warnings and does not need to be rerun.

Warnings reported ⚠

This icon shows that the process ran successfully but that warnings were encountered.

Errors reported ✖

This icon shows that the process ran but encountered an error.

Out-of-Date ?

This icon shows that you made design changes, which require that the process be rerun. If this icon is next to a report process, we can rerun the process to create an up-to-date version of the report.

No icon

If there is no icon, this shows that the process was never run.

If a process is marked as up-to-date, warnings reported, or errors reported, we cannot use the Run command.

2.5.2. Setting Process Properties

We can set process properties that enable us to control the way our design is implemented. Properties differ based on the device family we are targeting, our top-level module type, and our synthesis tool. In addition, there are both standard and advanced properties. By default, only the standard properties display in the Process Properties dialog box.

Setting advanced options is not recommended if we are just getting started with using the ISE™ software. When we enable the advanced properties, both standard and advanced properties appear in the Process Properties dialog box.

What to Do First

Open a project in Project Navigator. To Set Process Properties

In the Sources tab, select a Design View from the drop-down list.

Select the source file to process.

The source file we select affects the processes that appear in the Processes tab; only the processes that apply to the selected source are shown.

In the Processes tab, select the process for which we want to set properties.

Select Process > Properties.

We can also right-click the process and select Properties. If there are no properties for a process, the Properties menu item is grayed out.

In the Process Properties dialog box, change the property options.

For detailed information on each of the options, click the Help button in the dialog box.

Select OK.

What to Expect

The new property settings are used the next time we run the selected process. Some properties are dependent on other properties. The values of dependent properties may change based on other property values that you set.

2.5.3. Setting Command Line Options using Process Properties

In some cases, the Process Properties dialog box may not include an option for the command line option we want to set. In this case, we can enter the command line option using advanced properties in the Process Properties dialog box.

Note For details on most command line options, see the *Development System Reference Guide*. For details on XST command line options, see the *XST User Guide*. For details on CompXlib command line options, see the *Synthesis and Simulation Design Guide*. For details on ModelSim® command line options, see the documentation provided with the ModelSim software.

What to Do First

Set your Process Settings preference to Advanced.

To Set Command Line Options; in the Sources tab, select a Design View from the drop-down list. Select the source file to process.

Note: The source file we select affects the processes that appear in the Processes tab. For example, we must select the top module for the Implement Design process to appear.

In the Processes tab, select the process for which we want to enter command line options.

Select Process > Properties.

Note: We can also right-click the process and select **Properties**. If there are no properties for a process, the Properties menu item is grayed out.

In the Process Properties dialog box, enter command line options in the Other Application name Command Line Options field. Separate multiple options with a space.

Note: To view the command line used for a process, see the command line log file as described in viewing a Command Line Log File. Partner processes, such as synthesis commands for the Synplify® tools, are not recorded in the command line log file.

Click OK.

What to Expect

The new property settings are used the next time we run the selected process. The following table shows how certain Project Navigator processes correspond to command Line tools.

Table 1.5 Project Navigator Corresponds to Line Tools

Process	Process Properties Tab	Command Line Tool
Compile HDL Simulation Libraries	Simulation Library Compiler Properties	CompXLib
Synthesize – XST	Synthesis Options	XST
Translate	Translate Properties (FPGA)	NGDBuild
	Design Properties (CPLD)	
Generate Post-Translate Simulation Model	Simulation Model Properties	NetGen
Map	Map Properties	MAP
Generate Post-Map Simulation Model	Simulation Model Properties	NetGen
Place and Route	Place & Route Properties	PAR
Generate Post-Place and Route Simulation Model	Simulation Model Properties	NetGen
Fit (CPLD)	Fitting Properties	CPLDFit
	Reports Properties	TAEngine
Generate Programming File	General Options (FPGA)	BitGen
	Programming Properties (CPLD)	Hprep6
Generate Timing (CPLD)	Timing Report Properties	TAEngine
Analyze Power	XPower Properties	Xpower
Generate Post-Fit Simulation Model (CPLD)	Simulation Model Properties	NetGen

2.6. Working with Reported Data

2.6.1. Reviewing Reports

When we run a task process, such as Synthesize or Map, a report is generated, such as the Synthesis Report or Map Report. Tasks and reports are denoted by the following icons in the Project Navigator Processes tab:

- Tasks
- Reports

What to Do First

We run a task process in Project Navigator.

To Review a Report

We can review reports using any of the following methods:

- In the Project Navigator Processes tab; we select the report and do one of the following:
  - We double-click a report to open it.
  - We select Process > Open Without Updating to open an out-of-date report for investigative purposes.
- In the Project Navigator Workspace, we click the Design Summary tab. We click a report in the upper left pane of the Design Summary to display it in the right pane. We click the links in the lower left pane to navigate to the different sections in the report. Only reports that include summary information are available.
- From the command line, we browse to the directory in which our output files reside and open the report. We can open most reports using a text editor or using the xreport command line tool, as described in Using the Design Summary for FPGAs.

What to Expect

Project Navigator opens the report in the Workspace and indicates the status of the report with one of the following icons in the Processes tab:

- Up-to-date ✓

  This icon shows that the report matches the associated task process and does not need to be rerun.

- Out-of-Date ?

  This icon shows that we made design changes, which require that the associated task process be rerun.

- No icon

  If there is no icon, the report was never created.

2.6.2. Using the Design Summary for FPGAs

The Design Summary allows us to quickly access design overview information, messages, and reports. The Design Summary is supported for FPGA designs only.

What to Do First

We can open the Design Summary using either of the following methods:

- Project Navigator: By default, the Design Summary appears in the Workspace when we open a project. If we close the Design Summary, we can reopen it by double-clicking View Design Summary in the Processes tab. If we do not want the Design Summary to appear in the Workspace when we open a project, deselect Open Design Summary when project is opened in the ISE General Options page of the Preferences dialog box.
- Command line: Type `xreport`. If we do not have an ISE file, we select File > New Project to create one. When we run command line tools, we must use the -ise option and specify the ISE project file we created.

Viewing Design Overview Information;

The Design Overview section in the upper left pane of the Design Summary allows us to view the following information about our design.

To View Summary Information;

We click the Summary link to view the following information in the right pane:

- Project status information, including our project name, targeted device, ISE version, the state of our design, the number of errors and warnings, and the date and time the summary was generated.
- Device utilization summary, which shows the estimated utilization after XST synthesis and the actual utilization after mapping.
- Performance data that summarizes place and route results.
- Links to detailed reports with high-level information about each report.
- Links to secondary reports, such as the Xplorer Report.

Working with Messages;

The Errors and Warnings section in the upper left pane of the Design Summary allows us to view and manipulate messages. We click the links under the Errors and Warnings section to show messages in the right pane. We must select the Enable Enhanced Design Summary option in the lower left pane to view and manipulate messages. We can manipulate the messages as follows.

To Organize Messages;

Select or deselect the following options in the lower left pane of the Design Summary to organize the messages in the right pane:

- Organize Messages options
    - Flat shows all messages in chronological order.
    - Collate Consecutive collapses messages with the same message number that appears next to one another in chronological order.
    - Collate All collapses all identical messages.

- View Messages options show or hide errors, warnings, or information messages.

- Show Columns options show or hide table columns.

To Filter Messages;

Many of the processes we run using the ISE™ software generate messages. In some cases, we may want to suppress a particular message from appearing. For example, we may get a warning message about an unconnected pin that we intend to be unconnected.

To suppress the message from subsequent runs of the ISE software, we do the following:

- In the upper left pane, we click Summary.
- In the lower left pane, we make sure Enable Message Filtering is selected.
- In the upper left pane, we click a link under the Errors and Warnings section.
- In the right pane, we right-click the message to filter and select one of the following commands:
    - Filter All Instances of This Message to filter out messages with the same library name and message number, regardless of the message text.
    - Filter This Instance Only to filter out messages with the same library name, message number, and text.

To remove a filter we created or to use more advanced message filtering, we can use the Message Filters tool, as described in Using Message Filters. When we suppress a message, it does *not* mean the issue is fixed. We do *not* filter messages for issues that must be fixed.

To Highlight Messages;

We right-click a message and select one of the following commands:

- Highlight All Instances highlights messages with the same library name and message number, regardless of the message text.

- Highlight This Instance highlights messages with the same library name, message number, and text.

- We remove Highlighting restores the message to its original color.

To Tag Messages;

In some cases, we may want to tag messages with our own categories, for example "Severity." To create our own categories and tag messages, we do the following:

- To create a tag as follows:
    - We right-click in the right pane, and we select Create Tags.
    - In the Create Tags dialog box, we click Add Tag Category.
    - In the Add Tag Category dialog box, we type a name for the category (for example, "Severity"), and we click OK.
    - In the Create Tags dialog box, we click our new category and we select Add Tag Value.
    - In the Add Tag Value dialog box, we type a name for the value (for example, "High"). We create as many tag values as we need, and we click OK.
    - In the Create Tags dialog box, we click OK.

- Right-click a message and select one of the following commands:
    - Tag All Instances of This Message tags messages with the same library name and message number, regardless of the message text.
    - Tag This Instance Only tags messages with the same library name, message number, and text.

- In the Tag Message dialog box, select the tag to apply.

- Click OK.

To Show New Messages;

To show the number of new messages for the most recent software run, we do the following:

- In the upper left pane, we click Summary.
- In the lower left pane, we make sure the Enable Enhanced Design Summary and Display Incremental Messages options are selected.

- We run the software.
- Do either of the following in the upper left pane:
    - We click Summary to view summary information in the right pane, which includes the number of new messages.
    - We click the links under the Errors and Warnings section to show messages in the right pane, with new messages marked as New.

2.6.3. Using Message Filters

Many of the processes we run using the ISE™ software generate messages. In some cases, we may want to suppress a particular message from appearing. For example, we may get a warning message about an unconnected pin that we intend to be unconnected. To suppress the message from subsequent runs of the ISE software, we can use Message Filters. Filtered messages are suppressed from reports, from the Project Navigator Console tab, and from command line output, but are shown in the Message Filters tool.

We can filter most messages that begin with "WARNING" or "INFO" and are followed by a library name and message number. For example, the following message can be filtered:

```
WARNING:Xst:454:message text
```

Where `Xst` is the library name and `454` is the message number. In this case, the library name is the internal Xilinx® library that stores the message. Messages for some processes *cannot* be filtered, for example, messages generated by third-party software, such as the Synplicity® software.

When we suppress a message, it does not mean the issue is fixed. We should not filter messages for issues that must be fixed.

Message Filtering Basic Steps;

Message filtering comprises the following basic steps. For *detailed* procedural information, we see Filtering Messages from Project Navigator *or* Filtering Messages from the Command Line, depending on how we are running the ISE software.

- We launch the Message Filters tool.
- In the Message Filters tool, we enable message filtering and we save our settings.
- We run the ISE software.
- In the Message Filters tool, all the messages generated by the ISE software that can be filtered appear in the Messages List.
- In the Filters List, create filters for the messages that we want to filter out of subsequent runs of the ISE software.
- We save the message filters we created.
- We run the ISE software. The messages for which we created filters no longer appear.

Message Filters Main Window;

The Message Filters tool allows you to create filters for most warning and informational messages. This tool contains the following parts.

Files Window

This window shows the project for which we are applying filters and allows us to enable or disable filtering. It also shows the Message Sources, which are the programs that produced messages.

Filters List

This is the window in which we create and store our message filters.

Messages List

This window shows all the messages generated by the most recent ISE software run. We can drag the messages from this list to the Filters List to create new filters.

Filters List and Messages List Tables

The tables in the Filters List and Messages List include the following parts:

- Filters List icons

Disabled filter ✗

This icon indicates that you temporarily stopped the filter using the Disable command available from the right-click menu.

Invalid filter

This icon indicates that the filter we created cannot be applied. Messages that are missing a necessary entry, such as the Message number or Library entry cannot be filtered.

Note Invalid filters are not saved, that is, when we close and then reopen the Message Filters tool, the invalid filters do not appear.

New filter

In the Filters List, this icon indicates that the filter has not yet been applied.

- Source

This column shows the program that generated the message.

- Type

This column shows the icons for the following message types:

Warning message

Warnings indicate problems with our design. When filtering out warning messages, make sure that we are not filtering out a message that requires fixing the problem instead.

Informational message

Informational messages are less severe than warnings and provide helpful suggestions for improving our design. For example, informational messages may suggest that we improve your coding style.

- Library

This column shows the internal Xilinx® library that stores the message. The library name appears as part of each message. For example, in the following message, `Xst` is the library name:

```
WARNING: Xst:454:message text
```

In some cases, the Library field matches the Source field.

- Msg #

  This column shows the message number used by the software program.

- Filter Text

  In the Filters List, this column shows the message text to filter. If we delete the variable text in this field, all messages with the same message number are filtered, regardless of the message text. If we keep the variable text in this field, only messages that have the same number *and* exact message text are filtered. We can also click the variable text and enter wildcards to create filters. Variable text is blue and underlined.

- Count

  In the Filters List, this column shows the number of times the filter was applied in our most recent software run. We double-click this cell to get additional information about the filter count.

- Message Text

  In the Messages List, this column shows the text for the captured messages.

- Filtered ✓

  In the Messages List, this icon indicates that the message was filtered.

CONCLUSION.

By using vhdl language i have proof that it is very possible to create & realize the way a UART transmitter works and the way it get the data in a parallel form and transfer it out in a form of serial data so easily.

Now at the end its very easy to put this program in a electronic programmable chip and therefore in this way I can have the UART tansmitter physically.

The benefit that we gained by using a vhdl xilinx is just to get rid of the hardware and bunddle of cables and resistors, connections after all if only a single cable or a connection is missing or misplace the whole design can be easily destroyed and moreover its expensive and requires red alert ATTENTIONS to go through such a process.

To sumup, extraordinary convinence and a brilliant results can only be achieved by using VHDL the language of today and tommrow.

REFERENCES

[1] Xilinx-ISE- Software Help.

[2] Xilinx-ISE- quick Start

[3] Xilinx-ISE- Software Manual

[4] www.xilinx.com