Convolution Encoder Implementation using FPGA

S.S. Podutwar
Jagadambha College of Engineering & Technology, Yavatmal

H.M. Baradkar
Jagadambha College of Engineering & Technology, Yavatmal

V.R.Thakare
Jagadambha College of Engineering & Technology, Yavatmal

ABSTRACT

Convolution encoding is a Forward Error Correction (FEC) technique used in continuous one-way and real time communication links. It can provide substantial improvement in bit error rates so that small, low power, inexpensive transmitters can be used in such applications as satellites and hand-held communication devices. This thesis documents the development of a programmable convolution encoder implemented in a Field Programmable Gate Array (FPGA) from Xilinx, Inc., called the XC2S100. The encoder is capable of coding a digital data stream with any one of 39 convolution codes. The encoder is made up of the combinational and sequential logic circuits. The design is simplified so that FPGA implementation of this encoder is simpler. We have written the VHDL code for convolution encoder and results are tested on FPGA kit for simulation and synthesis.

Keyword:
FPGA, VHDL,Encoder

1. INTRODUCTION

VHDL is used to simulate the functionality of digital electronics circuit at levels of abstraction ranging from pure behavior down to gate level, and is also used to synthesis (i.e. automatically generate) gate level description from more abstract (RLT) descriptions [1]. VHDL is commonly used to support the high level design (or language based design) process, in which an electronic design is verified by means of through simulation at a high level of abstraction before proceeding to detailed design using automatic synthesized tool. VHDL becomes an IEEE Standard in 1987, and this version of language has been widely used in the electronics industry and academia. The standard was revised in 1993 to include a number of significant improvements. A hierarchical portion of a hardware design is described in VHDL by an Entity together with architecture. The entity defines the interface to the block of hardware (i.e. the inputs and outputs), while the architecture defines its internal structure or behavior.[1]. An entity may posses several alternative architectures. Hierarchy is defines in terms of components, which are analogous to chip socket. A component is instantiated within architecture to represent a copy of lower level hierarchical block. The association between the instance of the complete design hierarchy is assembled. The structure of an electronic circuit is described by, making instances of the components within architecture and instances together using signal. A signal represents an electrical connection, a wire or a bus. A port map is used to connect signals to the ports of a component instantiation, where a port represents a pin. Each signal has a type, as does every value in VHDL. The type defines both a set of values and the set of operations that can be performed on those values. A type is often define in a package, which a piece of VHDL containing definitions, which are common to several entities, architectures or other packages. Individual wires are often represented as signals of type Std_Logic, which are defined in the package Std_Logic_1164, another IEEE Standard. The behavior of an electronic circuit is described using processes (which represent the leaves in the hierarchy tree of the design). Each process executes concurrently with respect to all other processes, but the statements inside a process execute in sequential order

2.1 RTL Simulation:

The RTL simulation step is used to verify the correctness of the VHDL description. It gives clock-by-clock behavior of the design. The VHDL simulator reads the VHDL description, compiles it to remove any syntax errors. After syntax have been removed, the design. After simulation has completed, the stimulus should apply to the design and run to the design. After simulation has been run, the simulator will have generated output data that can be analyzed. Output data generated in the form of waveform and it takes tabulate form. Waveform display shows the values of the signal of the design over time. Tabular format shows signal transition form top to bottom. While output data is being analyzed, the user finds errors in design description he traced down the source of the errors in the code make change in it and recompile the code again. While designer will be happy with the behavior of the design the process of building real hardware will start.

2.2 VHDL Synthesis:

The goal of the VHDL synthesis step is to create a design that implements the required functionality and matches the designer's constraints in speed, area or power. The VHDL synthesis tools convert the VHDL description into a netlist in the target FPGA technology. To synthesis VHDL description the designer reads the verified VHDL description into the VHDL synthesis tool. It may report syntax errors and synthesis errors. Synthesis errors usually result from the designer using constructs that are not synthesizable. If there are no errors, the designers can synthesis the design and map the design to target technology. The designer looks at the synthesized output to determine whether or not the synthesis produced a good result. The synthesizer produces an output net list in the target technology and the number of report files.

2.3 Functional Gate Simulation:

Some designers might want to do a quick check on the output of the synthesis tool to make sure that the synthesis tool produced a design that is functionally correct. If proper design rules are followed for the input, the synthesis tool should never an output that is functionally different from RTL VHDL input, unless tool has bug. To check the results of synthesis tool, the designer runs a functional gate level verification. The designer reads the output VHDL input, unless tool has bug. To check the results of synthesis tool, the designer runs a functional gate level verification. The designer reads the output VHDL netlist from the synthesis tool plus a library of the synthesis primitives into the VHDL simulator and runs the simulation.

If design matches, then the synthesis tool did not produce logic mismatches, if it does not match; the designer needs to debug the RTL description. Mapping: Tool partitions the design into the logic blocks available on the device. Good partitioning results in better performance.

System Partitioning-

Dividing a large system into smaller / standard modules.

2.4 Place and Route:

Place and route tools are used to take the design netlist and implement the design in the target technology device. The place and route tools place each primitive from the netlist into an appropriate location on the target device and then route the signals between the primitives to connect the devices according to netlist. Inputs to this are device information, netlist, timing constraints, which gives the place and route the tools and indication about which the signals have critical timing associated with them and route these in the most timing efficient manner. And placement constraints specify the placement of large parts of the design. After all the cells are placed and routed, the output consists of data files that can be used to implement the chip.

2.5 Basics of Programmable Logic Technologies:

There are five types of programmable logic devices (PLDs):Read-only memory (ROM), Programmable logic array(PLA), Programmable array logic(PAL), Complex programmable gate array (CPLD),Field programmable gate array (FPGA).In PLDs, programming technologies are established or break interconnections, build look-up tables, and control transistor switching [4].

The oldest of programming technologies is the use of fuses, mask programming, antifuse.

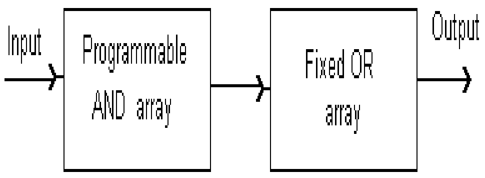


Figure 1: PAL Device

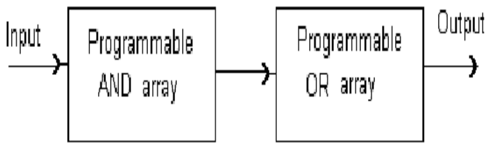


Figure 2 : PLA Device

In the case of FPGAs (fig.3) these files describe all of the connections needed to make FPGA macro cell implement. The other output from the place and route software is a file used to generate the timing file.

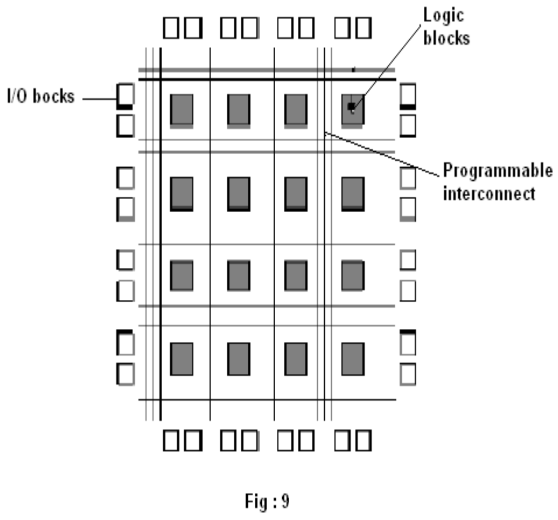


Fig.3 FPGA

FPGAs are arrays of logic blocks, which can be linked together to form complex logic implementations. They can be used for both combinational and sequential logic. They are more flexible and complex than CPLDs. CPLD, as shown in fig.10, is a device made of smaller common macro cells, each containing a PAL and a flip-flop, which can be interconnected with other macro cells using a programmable switch matrix. Macrocells are integrated to form a logic block. It's anon-volatile device, retaining its setting even after power off [4].

2.6 Post Layout Timing Simulation:

After the place and route process has completed, the designer will want to verify the results of the place and route process. This simulation combines the netlist used for place and route process into a simulation that checks both the functionality and timing of the design. The designer can run t he simulation and generate accurate output waveforms that show whether or not the device is operating properly and if the timing is being met.

2.7 Static Timing:

For designs of 10,000 gates to 100,000 gates, post route timing simulation can be a good method of verifying design functionality and timing. However, as designs get larger, or if the designer does not have test vector, the designer can use static timing analysis to make sure the design meet the timing requirements advantages of static timing are 1)The language supports flexible design methodologies, bottom up, top down or mixed.2) It supports synchronous and asynchronous timing model 3) It is an IEEE & ANSI std therefore models using this language are portable.4) It supports four basic description styles.a)Structural b) Behavioral c) Dataflow d) Mixed

3.1 Design flow

1. Creating the initial description of the design. The foundation series suit supports two design methodologies: based on schematic (Schematic flow) and based on hardware description language (HDL Flow). We can also describe certain part of the design as graphical state diagram.

2. Depending on the chosen design methodology:

a. For HDL flow: selecting the top-level design and synthesizing the design.

b. For schematic flow: Synthesizing the HDL macros if any exist, generate the net list.

3. Functional simulation of the design.

4. Editing design constraints and controls.

5. Optimizing the design for certain programmable device.

6. Running the Place and Route program a bit stream file and timing netlist.

7. Timing simulation of the design, based on the post layout timing net list. The timing netlist contains timing information extracted from the file generated during Place and Route processing. Simulations based on such netlist describe the circuit behavior far more accurately than functional simulation.

3.2 Creating Project

Create a new ISE project which will target the FPGA device on the Spartan-2 Startup Kit.

To create a new project:

1. Select File New Project... The New Project Wizard appears.

2. Type tutorial in the Project Name field.

3. Enter or browse to a location (directory path) for the new project. A tutorial sub directory is created automatically.

4. Verify that HDL is selected from the Top-Level Source Type list.

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

♦ Product Category:All

♦ Family:Spartan2

♦ Device:XC2s100

♦ Package:TQ144

♦ Speed Grade:-6

♦ Top-Level Module Type: HDL

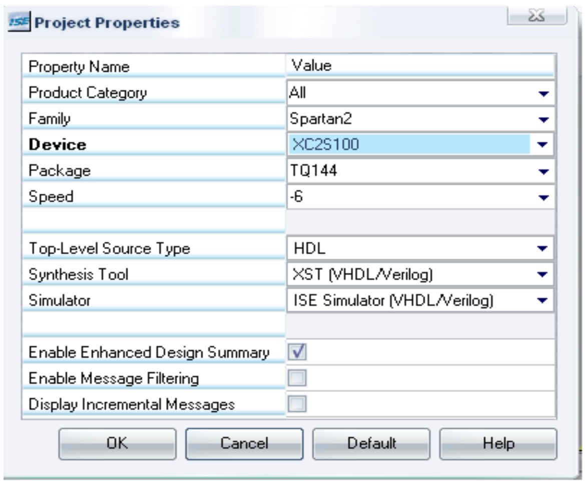
♦ Synthesis Tool:XST (VHDL/ Verilog)

♦ Simulator:ISE Simulator (VHDL/Verilog)

♦ Verify that Enable Enhanced Design Summary is selected.

Leave the default values in the remaining fields.

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



Property Name	Value
Product Category	All
Family	Spartan2
Device	XC2S100
Package	TQ144
Speed	-6
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISE Simulator (VHDL/Verilog)
Enable Enhanced Design Summary	✓
Enable Message Filtering	☐
Display Incremental Messages	☐

Fig.4 Project Device Properties

6. Click Next to proceed to the Create New Source window in the New Project Wizard. At the end of the next section, your new project will be complete.

3.3 Create an HDL Source

In this section, you will create the top-level HDL file for your design. Determine the language that you wish to use for the tutorial. Then, continue either to the "Creating a VHDL Source" section below.

3.4 Creating a VHDL Source:

Create a VHDL source file for the project as follows:

1. Click the New Source button in the New Project Wizard.

2. Select VHDL Module as the source type.

3. Type in the file name ce.

4. Verify that the Add to project checkbox is selected.

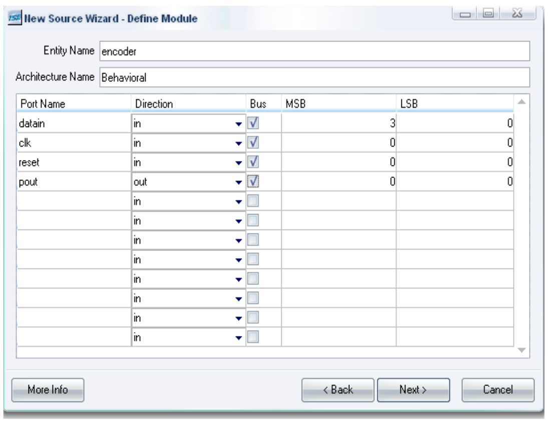
5. Click Next.

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

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

8. Click Next, then Next, then Finish.

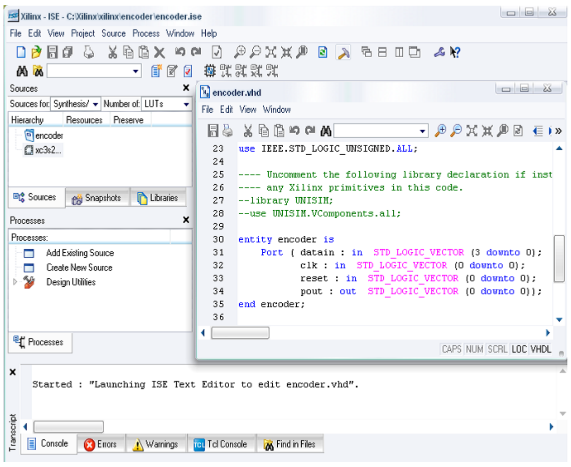
The source file containing the entity/architecture pair displays in the Workspace, and the convolution encoder displays in the Source tab, as shown below:



Entity Name: encoder
Architecture Name: Behavioral

Port Name	Direction	Bus	MSB	LSB
datain	in	✓	3	0
clk	in	✓	0	0
reset	in	✓	0	0
pout	out	✓	0	0
	in			
	in			
	in			
	in			
	in			
	in			
	in			
	in			

Fig5 : Define Module



```
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if inst
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity encoder is
    Port ( datain : in  STD_LOGIC_VECTOR (3 downto 0);
           clk : in  STD_LOGIC_VECTOR (0 downto 0);
           reset : in  STD_LOGIC_VECTOR (0 downto 0);
           pout : out  STD_LOGIC_VECTOR (0 downto 0));
end encoder;
```

```
Started : "Launching ISE Text Editor to edit encoder.vhd".
```

Fig 6 New Project in ISE

3.5 Using Language Templates (VHDL) :

The next step in creating the new source is to add the behavioral description for the Convolution encoder. To do this you will use a Convolution encoder code example from the ISE Language

Templates and customize it for the convolutional encoder design.

1. Place the cursor just below the begin statement within the Convolution encoder architecture.

2. Open the Language Templates by selecting Edit →Language Templates…

Note: You can tile the Language Templates and the convolutional encoder file by selecting Window →Tile Vertically to make them both visible.

3. Using the "+" symbol, browse to the following code example:

VHDL →Synthesis Constructs →Coding Examples → Convolution encoder

4. With Convolution encoder selected, select Edit →Use in File, or select the Use Template in File toolbar button. This step copies the template into the Convolution encoder source file.

5. Close the Language Templates.

4. DESIGN SIMULATION AND RESULT:

4.1 Verifying Functionality using Behavioral Simulation:

Create a test bench waveform containing input stimulus you can use to verify the functionality of the convolution encoder. The test bench waveform is a graphical view of a test bench.

Create the test bench waveform as follows:

1. Select the convolution encoder HDL file in the Sources window.

2. Create a new test bench source by selecting Project →New Source.

3. In the New Source Wizard, select Test Bench Waveform as the source type, and type convolutional encoder in the File Name field.

4. Click Next.

5. The Associated Source page shows that you are associating the test bench waveform with the source file. Click Next.

6. The Summary page shows that the source will be added to the project, and it displays the source directory, type and name. Click Finish.

7. You need to set the clock frequency, setup time and output delay times in the Initialize Timing dialog box before the test bench waveform editing window opens.

The requirements for this design are the following:

The convolution encoder must operate correctly with an input clock frequency = 25 MHz.

The DIRECTION input will be valid 10 ns before the rising edge of CLOCK.

The output (COUNT_OUT) must be valid 10 ns after the rising edge of CLOCK.

The design requirements correspond with the values below.

Fill in the fields in the Initialize Timing dialog box with the following information:

♦ Clock Time High:20 ns.

♦ Clock Time Low:20 ns.

♦ Input Setup Time:10 ns.

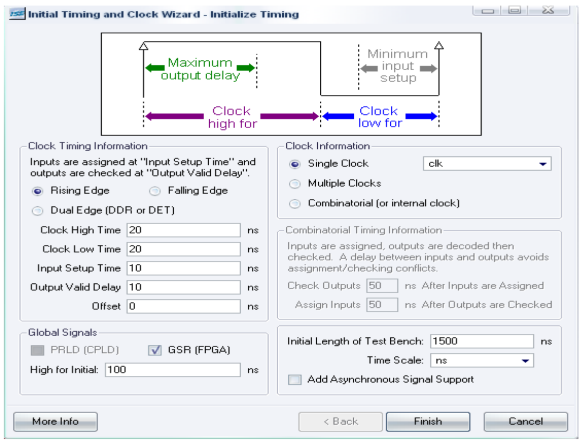
♦ Output Valid Delay:10 ns.

♦ Offset:0 ns.

♦ Global Signals:GSR (FPGA)

Leave the default values in the remaining fields.

8. Click Finish to complete the timing initialization.



Initial Timing and Clock Wizard - Initialize Timing

Clock Timing Information
Inputs are assigned at "Input Setup Time" and outputs are checked at "Output Valid Delay".
[✓ Rising Edge] Falling Edge
Dual Edge (DDR or DET)

Clock High Time	20	ns
Clock Low Time	20	ns
Input Setup Time	10	ns
Output Valid Delay	10	ns
Offset	0	ns

Clock Information
[✓ Single Clock] clk
Multiple Clocks
Combinatorial (or internal clock)

Combinatorial Timing Information
Inputs are assigned, outputs are decoded then checked. A delay between inputs and outputs avoids assignment/checking conflicts.
Check Outputs 50 ns After Inputs are Assigned
Assign Inputs 50 ns After Outputs are Checked

Global Signals
PRLD (CPLD) [✓ GSR (FPGA)]
High for Initial: 100 ns

Initial Length of Test Bench: 1500 ns
Time Scale: ns
Add Asynchronous Signal Support

More Info | < Back | Finish | Cancel

Fig 7. Initialize Timing

The blue shaded areas that precede the rising edge of the CLOCK correspond to the Input Setup Time in the Initialize Timing dialog box. Toggle the DIRECTION port to define the input stimulus for the convolutional encoder design as follows:

♦ Click on the blue cell at approximately the 300 ns to assert DIRECTION high so that the convolutional encoder will count up.

♦ Click on the blue cell at approximately the 900 ns to assert DIRECTION high so that the convolutional encoder will count down.

10. Save the waveform

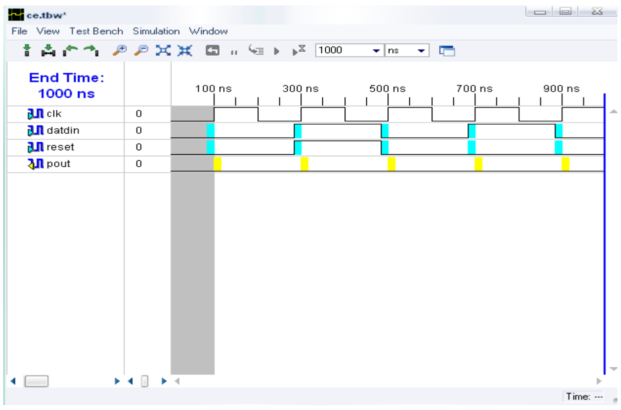


Fig 8. Test Bench Waveform

11. In the Sources window, select the Behavioral Simulation view to see that the test bench waveform file is automatically added to your project.
12. Close the test bench waveform.

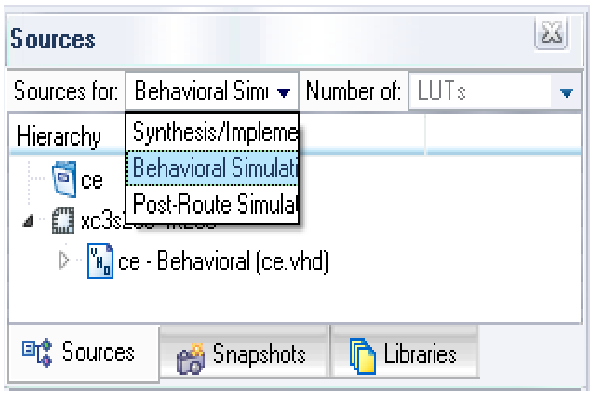


Fig9. Behavior Simulation Selection

4.2 Create a Self-Checking Test Bench Waveform:

Add the expected output values to finish creating the test bench waveform. This transforms the test bench waveform into a self-checking test bench waveform. The key benefit to a self-checking test bench waveform is that it compares the desired and actual output values and flags errors in your design as it goes through the various transformations, from behavioral HDL to the device specific representation.To create a self-checking test bench, edit output values manually, or run the Generate Expected Results process to create them automatically. If you run the Generate Expected Results process, visually inspect the output values to see if they are the ones you expected for the given set of input values. To create the self-checking test bench waveform automatically, do the following:

1. Verify that Behavioral Simulation is selected from the drop-down list in the Sources window.

2. Select the convolutional encoder_tbw file in the Sources window.

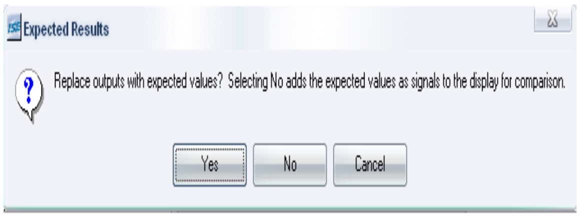
3. In the Processes tab, click the "+" to expand the Xilinx ISE Simulator process and double-click the Generate Expected Simulation Results process. This process simulates the design in a background process.

4. The Expected Results dialog box opens. Select Yes to annotate the results to the test bench.

5. Click the "+" to expand the COUNT_OUT bus and view the transitions that correspond to the Output Delay value (yellow cells) specified in the Initialize Timing dialog box.

6. Save the test bench waveform and close it.

You have now created a self-checking test bench waveform.



Expected Results

Replace outputs with expected values? Selecting No adds the expected values as signals to the display for comparison.

Fig 10. Expected Results Dialog Box

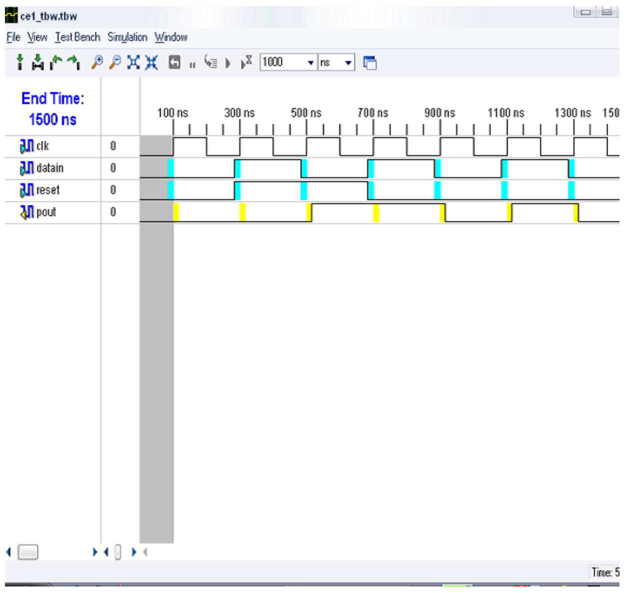


Fig 11.Test Bench Waveform with Results

4.3 Simulating Design Functionality:

Verify that the convolutional encoder design functions as you expect by performing behavior simulation as follows:

1. Verify that Behavioral Simulation and convolutional encoder_tbw are selected in the Sources

window.

2. In the Processes tab, click the "+" to expand the Xilinx ISE Simulator process and double-click the Simulate Behavioral Model process. The ISE Simulator opens and runs the simulation to the end of the test bench.

3. To view your simulation results, select the Simulation tab and zoom in on the transitions. The simulation waveform results will look like the following:

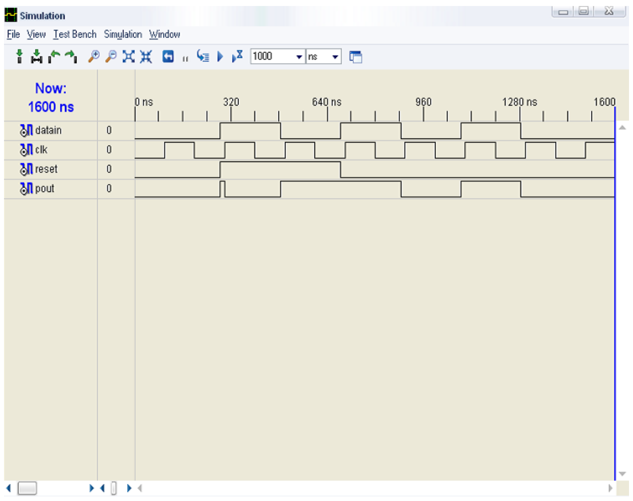


Fig 12.Simulation Results

4. Verify that the convolutional encoder is counting up and down as expected.

5. Close the simulation view. If you are prompted with the following message, "You have an active simulation open. Are you sure you want to close it?", click Yes to continue. You have now completed simulation of your design using the ISE Simulator.

4.4 Assigning Pin Location Constraints:

Specify the pin location for the port of the design so that they are connected correctly on the Spartan-2 Startup kit demo board.

1. Verify that convolution encoder is selected in the sources window.
2. Click on the user constrain.
3. Double click on the Edit constrain.
4. Then assign pins to the device is FPGA.
5. Then save the assign pin configuration.

4.5 Download design to Spartan 2 demo board:

This is the last step in the design verification process. This section provides simple instructions for downloading the counter design to the Spartan-3 Starter Kit demo board.

1. Connect the 5V DC power cable to the power input on the demo board (J4).

2. Connect the download cable between the PC and demo board (J7).

3. Select Synthesis/Implementation from the drop-down list in the Sources window.

4. Select counter in the Sources window.

5. In the Processes window, click the "+" sign to expand the Generate Programming File processes.

6. Double-click the Configure Device (iMPACT) process.

7. The Xilinx WebTalk Dialog box may open during this process. Click Decline.

8. Select Disable the collection of device usage statistics for this project only and

click OK.

9. In the Welcome dialog box, select Configure devices using Boundary-Scan (JTAG).

10. Verify that Automatically connect to a cable and identify Boundary-Scan chain is selected.

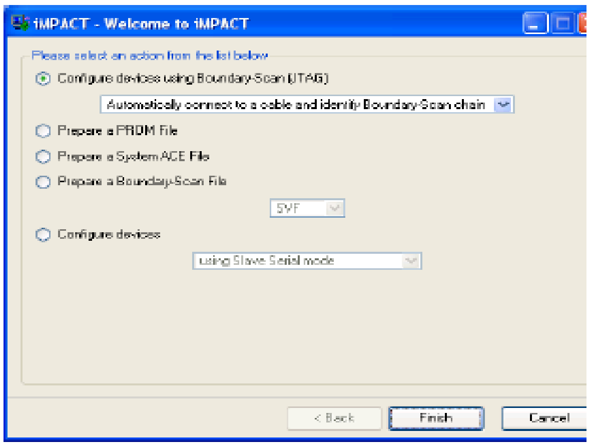
11. Click Finish.

12. If you get a message saying that there are two devices found, click OK to continue.

The devices connected to the JTAG chain on the board will be detected and displayed

in the iMPACT window.

13. The Assign New Configuration File dialog box appears. To assign a configuration file to the xc2s100 device in the JTAG chain, select the counter. bit file and click Open.


iMPACT - Welcome to iMPACT

Please select an action from the list below

(●) Configure devices using Boundary-Scan (JTAG)

Automatically connect to a cable and identify Boundary-Scan chain

( ) Prepare a PROM File

( ) Prepare a System ACE File

( ) Prepare a Boundary-Scan File

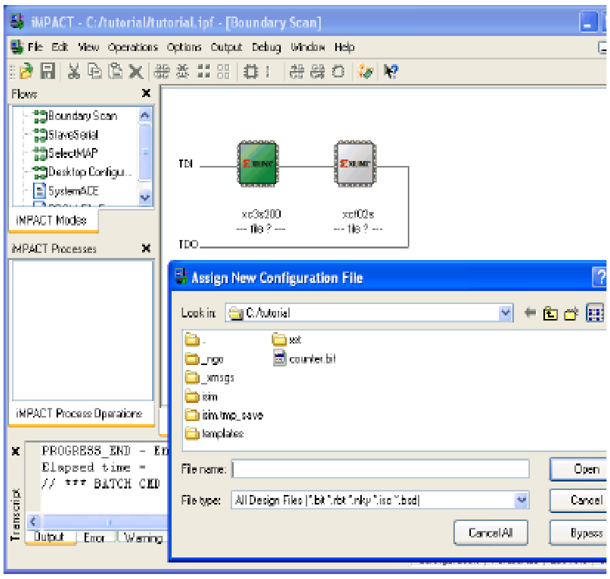
SVF

( ) Configure devices

using Slave Serial mode

< Back | Finish | Cancel

Fig 13. Post-Place & Route Simulation



```
PROGRESS_END - En
Elapsed time =
// *** BATCH CMD
```

Assign New Configuration File

Look in: C:/tutorial

.. | xst
_ngo | counter.bit
_xmsgs
isim
isim.tmp_save
templates

File name:

File type: All Design Files (*.bit *.rbt *.nky *.isc *.bsd)

Open | Cancel | Cancel All | Bypass

Fig 14. iMPACT Welcome Dialog Box

14. If we get a Warning message, click OK.
15. Select Bypass to skip any remaining devices.
16. Right-click on the xc2s100 device image, and select Program...
The Programming Properties dialog box opens.
17. Click OK to program the device. When programming is complete, the Program Succeeded message is displayed in the board, LEDs 0, 1, 2, and 3 are lit, indicating that the convolution encoder is running.
18. Close iMPACT without saving. You have completed the ISE Quick Start Tutorial. For an in-depth explanation of the ISE design tools, see the ISE In-Depth Tutorial on the Xilinx® web site at: www.xilinx.com

5.APPLICATIO OFCONVOLUTION ENCODER:

Convolution codes are capable of allowing a communication system to reach a Shannon limit, which is theoretical limit of SNR error free communication over a noisy channel .Image Processing: It is used for removing the noise in the image. It also used for digital Image compression and expansion. [5] Telecommunication has a vital use of convolution encoder in the process of data transmission and reception In military terrestrial telephony is a good example of use of convolution encoder for noise reduction. It is used in deep space communication. It finds its application in long haul terrestrial wireless. Satellite communication also employs convolution encoder for digital image processing, high resolution video graphic images.G.S.M speech and channel coding: In order to send our voice across a radio network, we have to turn our voice into a digital signal. GSM uses a method called RPE-LPC (Regular Pulse Excited - Linear Predictive Coder with a Long Term Predictor Loop) to turn our analog voice into a compressed digital equivalent. Once we have a digital signal we have to add some sort of redundancy so that we can recover from errors when we trams our digital voice over the radio channel. GSM uses a convolution codes to encode digital speech representations.. Convolution encoder is used in forward error correction to improve data communication. Digital communication system: The convolution encoder finds a verity of application in the digital communication system like in DSP, Mobile communication, medical science, computer aided design (CAD).

6. CONCLUSION:

Research follows concept & invention follow research. With the advent of FPGA is supposed to be the best approach towards gaming , control , new digital & high definition broad cast TV transmission , multimedia LCD, plasma display technology & the automotive technology .In both market segments, designers are turning to FPGAs for the unique advantages they offer in terms of rapid development & programmability & their growing availability at lower price points. Value FPGAs are also finding their way into many data & programmers Concept of FPGA will require some time to be develop as a technique to be used so we are not far from the era of Field Programmable Gate Array which will cover the whole electronics ERA with its highly versatile and flexible features.

REFERENCE:

[1]. VHDL Primer , J.Bhaskar

[2]. Logic and computer Design by M. Morris Mano, C.R. Kime

[3]. Circuit Design with VHDL by Douglas Perry.

[4]. VHDL by Brown & Vranesic

[5] Digital Communication by Simon Haykin