Synthesis of Convolution Encoder and Viterbi decoder of rate 2/3 using Xilinx ISE tool

Rakhi B. Menon¹, Dr. Gnana Sheela K.²
M.Tech Student¹, Professor², Department of ECE, TOCH Institute of Science & Technology

Abstract—Synthesis of Convolution Encoder and Viterbi Decoder is mainly focused in this paper. The code for encoder and decoder and their test benches are written using Verilog HDL and are simulated using Altera Model Sim. The Synthesis process is done for Encoder and Decoder using Xilinx ISE tool. Once the functional verification is done, the RTL model is taken to the synthesis process using the Xilinx ISE tool. In synthesis process, the RTL model will be converted to the gate level net list mapped to a specific technology library. In this step, any semantic and syntax errors are checked. The synthesis report is created which gives the details of errors and warning. The codes must be made synthesizable in order to implement the design on FPGA.

Index Terms— Convolution encoder, Synthesis, Viterbi decoder, Xilinx ISE tool

I. INTRODUCTION

Data transmissions over wireless channels are affected by attenuation, distortion, interference and noise, which affect the receiver’s ability to receive correct information. The use of re-transmission methods is not efficient and has large latency measure up to the rising speed and data rates of communication links, the need of new techniques arise here to be compatible with those systems. In the recent past, the error-correcting coding has become one integral part in nearly all the modern data communication and storage systems. With the continuously increasing demands for higher speed and lower power communication systems, enhanced VLSI implementations of those error-correcting codes that are currently used in practical applications have great current importance. Coding techniques are essential for a communication system to achieve high performance. One of the most important and direct applications of information theory is coding theory. The purpose of Forward Error Correction (FEC) is to improve the capacity of channel by adding some carefully designed redundant information to the data which is being transmitted through the channel. The process of adding this redundant information is known as channel coding.

II. CONVOLUTION ENCODER

Convolutional codes are frequently used to correct errors in noisy channels. They have rather good correcting capability and perform well even on very bad channels. Convolutional codes are extensively used in satellite communications. The modest complexity and good performance have made the Viterbi algorithm the preferred decoding method for convolutional codes to overcome transmission errors. Convolutional encoder with Viterbi decoder acts as powerful method for Forward Error Correction (FEC). An encoder includes extra information in the transmitted signal to reduce the probability of errors in the received signal that may be corrupted by noise. Every two bits of data stream are encoded into three bits for transmission. The ratio of input to output information in an encoder is the rate of the encoder; this is a rate 2/3 encoder. The following equations relate the three encoder output bits (Yn2, Yn1, and Yn0) to the two encoder input bits (Xn1 and Xn0).

\[ \begin{align*}
Yn2 &= Xn1 \quad (1.1) \\
Yn1 &= Xn0 \ xor \ Df2 \quad (1.2) \\
Yn0 &= Df1 \quad (1.3)
\end{align*} \]

![Convolution Encoder of rate 2/3](image1)

III. VITERBI DECODER

Encoding process is a simple task but decoding the encoded signals is quite a difficult task. There are different decoding techniques such as threshold decoding, sequential decoding and Viterbi decoding and the last one is the best of the decoding techniques. The main processes in a Viterbi decoder are branch metric calculation, path metric calculation and trace backing. There are two types of Viterbi decoder- Hard decision Viterbi decoder and Soft decision Viterbi decoder. Hard decision uses hamming distance whereas Soft decision decoder using Euclidean distance. The methodology used here is Soft decision Viterbi decoding technique. The internal modules inside Viterbi decoder are Euclidean distance calculation module, Subset decode module, Compute Metric module, Compare Select module, Path In module, Path Memory module, Metric module, Reduce module and Output decision module. The output of decoder is 2 bits. The block diagram of Viterbi decoder is shown below.
IV. SIMULATION

Codes of Convolution encoder and Viterbi decoder and their test benches is written in Verilog HDL and is simulated using Altera Model Sim. Simulator is a software program to verify functionality of a circuit. The functionality of code is checked. The inputs are applied and corresponding outputs are checked. If the expected outputs are obtained then the circuit design is correct. Simulation gives the output waveforms in form of zeros and ones. Here arises what is called RTL diagrams. The RTL-level (behavioural) simulation enables us to verify or simulate a description at the system or chip level. At this step, no timing information is provided, and simulation should be performed in unit-delay mode to avoid the possibility of a race condition. Although problems with the size or timing of the hardware may still crop up later, the designer can at least be sure that his logic is functionally correct before going on to the next stage of development. The Convolution encoder simulated using Model Sim is shown below.

Figure 3. Simulation of Convolution encoder

V. SYNTHESIS

An intermediate representation of the hardware design is produced. This step is called synthesis and the result is a representation called a net list. Once the functional verification is done, the RTL model is taken to the synthesis process using the Xilinx ISE tool. In synthesis process, the RTL model will be converted to the gate level netlist mapped to a specific technology library. In this step, any semantic and syntax errors are checked. The synthesis report is created which gives the details of errors and warning if any. The net list is device independent, so its contents do not depend on the particulars of the FPGA; it is usually stored in a standard format called the Electronic Design Interchange Format (EDIF). The RTL (Register Transfer Logic) can be viewed as black box after synthesis of design is made. It shows the inputs and outputs of the system. By double-clicking on the diagram we can see gates, flip-flops and MUX.

A. Synthesis of Convolution Encoder

The RTL (Register Transfer Logic) of Convolution encoder is shown below, where an input of two bits is provided to the encoder along with clock and reset. Three bit values are obtained as output.

Figure 4. RTL view of Convolution encoder

The device utilization includes the following.

- Logic Utilization
- Logic Distribution
- Total Gate count for the Design

Figure 5. Schematic diagram of Convolution encoder

Figure 6. Device Utilization of Convolution Encoder

The device utilization summary is shown above in which its gives the details of number of devices used from the available devices and also represented in %. Hence as the result of the synthesis process, the device utilization in the used device and package is shown.

Number of Slices: 2 out of 960 0%
Number of Slice Flip Flops: 2 out of 1920 0%
Number of 4 input LUTs: 1 out of 1920 0%
Number of bonded IOBs: 7 out of 108 6%
Number of GCLKs: 1 out of 24 4%

B. Synthesis of Viterbi decoder

The RTL (Register Transfer Logic) of Viterbi decoder is shown below, where an input of three bits is provided to the encoder along with clock and reset. Two bit values are obtained as output along with an error bit. If the error bit is 0,
there are no errors occurred. When the error bit is 1, it indicates the presence of errors.

The device utilization summery is shown above in which it gives the details of number of devices used from the available devices and also represented in %. Hence as the result of the synthesis process, the device utilization in the used device and package is shown

<table>
<thead>
<tr>
<th>Device Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops:</td>
<td>137 out of 1920</td>
<td>7%</td>
<td></td>
</tr>
<tr>
<td>Number of 4 input LUTs:</td>
<td>477 out of 1920</td>
<td>24%</td>
<td></td>
</tr>
<tr>
<td>Logic Distribution:</td>
<td>240 out of 960</td>
<td>25%</td>
<td></td>
</tr>
<tr>
<td>Number of Slices containing only related logic:</td>
<td>245 out of 960</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Number of Slices containing unrelated logic:</td>
<td>5 out of 960</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td>Total Number of 4 input LUTs:</td>
<td>477 out of 1920</td>
<td>24%</td>
<td></td>
</tr>
<tr>
<td>I/O Flip Flops:</td>
<td>1 out of 108</td>
<td>7%</td>
<td></td>
</tr>
<tr>
<td>Number of GCLKs:</td>
<td>1 out of 24</td>
<td>4%</td>
<td></td>
</tr>
<tr>
<td>Total equivalent gate count for design:</td>
<td>2,191</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional FPGA gate counter for IOBs:</td>
<td>384</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**C. Synthesis of Viterbi distance module**

The first module is Viterbi distance calculation module and this module calculates the Euclidean distance which is required for performing soft decision Viterbi decoding. Input to this module is the three bit output of Convolution encoder along with clock and reset. Eight Euclidean distances are obtained as the output.

The result of the synthesis process, the device utilization in the used device and package is shown

<table>
<thead>
<tr>
<th>Device Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices:</td>
<td>14 out of 960</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>Number of 4 input LUTs:</td>
<td>24 out of 1920</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>Number of bonded IOBs:</td>
<td>27 out of 108</td>
<td>25%</td>
<td></td>
</tr>
</tbody>
</table>

**D. Synthesis of Subset Decode module**

The second module is subset decode module and it groups 0th and 4th distance as a subset. Similarly (1,5) subset, (2,6) subset and (3,7) subsets are made. The Euclidean distances obtained from the previous module are provided as the input to subset decode along with clock and reset. Four subsets along with control signals are generated as output.
The third module is the compute metric module where branch metrics and path metrics are added together. Eight paths are obtained as the output.

The device utilization summery is shown above in which its gives the details of number of devices used from the available devices and also represented in %.

- Number of Slices: 14 out of 960 1%
- Number of 4 input LUTs: 24 out of 1920 1%
- Number of bonded IOBs: 42 out of 108 38%
- IOB Flip Flops: 24
- Number of GCLKs: 1 out of 24 4%

The output of Compute metric is given as the input to the Compare Select Module. The branch metrics and path
metrics are added together and minimum of them is compared and selected by this module.

**Figure 19. RTL view of Compute Select module**

**Figure 20. Schematic diagram of Compare Select module**

**Figure 21. Device Utilization of Compare Select module**

The device utilization summary is shown above in which it gives the details of number of devices used from the available devices.

- Number of Slices: 25 out of 960 (2%)
- Number of 4 input LUTs: 44 out of 1920 (2%)
- Number of bonded IOBs: 64 out of 108 (59%)

**G. Synthesis of Path In module**

The output of Compare Select module along with clock, reset and output of subset decode module act as input of this module and produces a Path of 12 bit value.

**Figure 22. RTL view of Path In module**

**Figure 23. Schematic diagram of Path In module**

**Figure 24. Device Utilization of Path In module**

The result of the synthesis process, the device utilization in the used device and package is shown.

- Number of Slices: 6 out of 960 (0%)
- Number of 4 input LUTs: 4 out of 1920 (0%)
- Number of bonded IOBs: 22 out of 108 (20%)

IOB Flip Flops: 10

All Rights Reserved © 2016 IJARECE
H. Synthesis of Path Memory module
The output of Path In module, output of Compare Select module along with clock and reset forms the input of the Path Memory module and produces four path of three bits.

Figure 25. RTL view of Path Memory module

Figure 26. Schematic view of Path Memory module

Figure 27. Device Utilization of Path Memory module
The device utilization summary is shown above in which it gives the details of number of devices used from the available devices and also represented in %.
Number of Slices: 75 out of 960 7%
Number of Slice Flip Flops: 132 out of 1920 6%
Number of 4 input LUTs: 132 out of 1920 6%
Number of bonded IOBs: 30 out of 108 27%
Number of GCLKs: 1 out of 24 4%

I. Synthesis of Metric module
Output from Reduce module, clock and reset are the input of Metric Module and its output is again re circulated to metric compute module. Thus present branch metric is added to path metric of the previous stage of the trellis.

Figure 28. RTL view of Metric module

Figure 29. Schematic view of Metric module

Figure 30. Device Utilization of Metric module
The result of the synthesis process, the device utilization in the used device and package is shown
Number of Slices: 12 out of 960 1%
Number of bonded IOBs: 42 out of 108 38%
IOB Flip Flops: 20
Number of GCLKs: 1 out of 24 4%

J. Synthesis of Reduce module
The four paths selected by Compare Select module forms the input of Reduce module and its output is given as the input to metric module as path metrics and branch metrics have to be continuously added together for the completion of decoding process.
The device utilization summary is shown above in which it gives the details of number of devices used from the available devices and also represented in %. Hence as the result of the synthesis process, the device utilization in the used device.

Number of Slices: 36 out of 960 3%
Number of 4 input LUTs: 62 out of 1920 3%
Number of bonded IOBs: 42 out of 108 38%

K. Synthesis of Output Decision module
The output from path memory module along with a control signal, which was an output of reduce module is fed into the output decision module as input. This is the final block of the Viterbi decoder that produces a two bit output.

VI. CONCLUSION
The Convolution Encoder and Viterbi decoder are simulated using Altera Model Sim and Synthesized using Xilinx ISE tool. RTL view and Schematic diagram of encoder, decoder and their internal modules makes the internal structures and connections easily understandable. Device Utilization Summary reveals the amount of memory used by each module.

REFERENCES
on Computer Networks and Information Technology (ICCNIT), pp. 93-98, 2011


AUTHOR PROFILE

Rakhi B. Menon completed her B.Tech in Electronics & Communication Engineering under Mahatma Gandhi University. Currently she is pursuing M.Tech in the specialization of VLSI and Embedded System of CUSAT.

Dr. Gnana Sheela K received her Ph.D in Electronics & Communication from Anna University, Chennai. She is working as a Professor, Department of ECE, TOC-H Institute of Science and Technology. She has published 25 international journal papers. She is life member of ISTE