International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

FPGA Prototyping of 8-Bit Trellis Encoder & IC Design using Cadence

Jayesh Argade[1], Vaishnavi Avhad[2], Anisha Bhujbal[3], Mr. H. S. Thakar[4]

[1,2,3]Dept. of ENTC
[4]Guide, Assistant Professor, Department of Electronics and Telecommunication, Pune Institute of Computer Technology
jayeshargade07@gmail.com[1], vgavhad2001@gmail.com[2] , anishabhujbal2000@gmail.com[3] , hsthakar@pict.edu[4]

ABSTRACT:

This paper provides an overview of FPGA prototyping and 8-bit trellis encoding. Trellis coding is a method for enhancing the performance of error-prone channels in digital communication systems. The method entails employing a convolutional encoder to encode the transmitted data and a Viterbi decoder to decode the received data. In this study, we go over the fundamentals of trellis coding and talk about how an 8-bit trellis encoder is made. We also talk about the system's effectiveness and efficiency as well as the encoder's implementation on an FPGA. Our findings demonstrate that trellis encoding can be made a practical choice for real-time communication systems by greatly increasing its speed and effectiveness when an FPGA is used. The study's conclusion discusses possibilities.

Keywords— Convolution 8-bit Trellis Diagram, VERILOG, FPGA, Cadence, Xilinx 14.7 ISE

I. Introduction

Data must be transmitted through error-prone channel in digital communication networks. Trellis en- coding is one of the methods for enhancing the performance of these error-prone channels. Trellis encoding is method that entails employing a convolutional encoder to encode the transmitted data. The methods capacity to increase the accuracy of the transmitted data has led to its widespread adoption in a variety of communication systems.

This paper's objective is to provide an overview of FPGA prototyping of 8-bit trellis encoding. In this study, we go over the fundamental of 8-bit trellis encoder is made. We also do the system effectiveness and efficiency as well as the encoders implementation on an FPGA.

II. Background and motivation

1. Digital communication systems: Digital communication systems have become ubiquitous in modern society, and reliable transmission of data is essential for many applications. Forward error correction (FEC) techniques, such as trellis encoding, are commonly used to improve the reliability of data transmission over noisy channels.
2. FPGA technology: FPGA technology has advanced significantly in recent years, providing a flexible and cost-effective solution for implementing digital signal processing (DSP) functions, including trellis encoding and decoding.
3. Need for prototyping: FPGA prototyping allows for the rapid development and testing of digital communication systems, enabling engineers to evaluate the performance of different encoding and decoding algorithms and optimize system parameters.
4. IC design using Cadence: Integrated circuit (IC) design using tools such as Cadence is essential for developing high-performance and low-power digital communication systems. By designing custom ICs, engineers can optimize system performance and reduce system cost and power consumption.

5. Optimization of 8-bit trellis encoder: The design and implementation of an 8-bit trellis encoder on an FPGA board, as well as IC design using Cadence, can enable engineers to optimize the performance of the encoder, such as achieving high coding efficiency, low complexity, and low power consumption.

III. Research problem and objectives

1. Develop an optimized 8-bit trellis encoder algorithm that maximizes coding efficiency while minimizing complexity and power consumption.
2. Implement the 8-bit trellis encoder on an FPGA board and evaluate its performance in terms of coding efficiency, complexity, and power consumption.
3. Design and optimize custom integrated circuits (ICs) using Cadence that implement the 8-bit trellis encoder, with the goal of further improving performance and reducing power consumption.
4. Compare the performance of the FPGA implementation and custom IC designs with existing implementations of trellis encoding on FPGA boards and ICs, respectively.
5. Analyse the trade-offs between performance, complexity, and power consumption for the different implementations of trellis encoding and identify areas for further optimization.

IV. Scope and limitations Scope:

1. The research focuses on optimizing the performance of an 8-bit trellis encoder algorithm using FPGA prototyping and IC design using Cadence.
2. The research includes the implementation and evaluation of the 8-bit trellis encoder algorithm on an FPGA board, as well as the design and optimization of custom integrated circuits (ICs) using Cadence.
3. The research compares the performance of the FPGA implementation and custom IC designs with existing implementations of trellis encoding on FPGA boards and ICs, respectively.
4. The research analyses the trade-offs between performance, complexity, and power consumption for the different implementations of trellis encoding.

Limitations:

1. The research focuses on optimizing the performance of an 8-bit trellis encoder algorithm and does not explore other encoding techniques or modulation schemes.
2. The research only considers the implementation of the 8-bit trellis encoder on an FPGA board and custom IC designs and does not investigate the implementation on other platforms such as application-specific integrated circuits (ASICs) or software-defined radios.
3. The research evaluates the performance of the 8- bit trellis encoder using simulations and does not include experimental validation.
4. The research does not consider the impact of channel fading, multipath propagation, or other real- world channel impairments on the performance of the 8-bit trellis encoder.

V. Literature Review

1. "FPGA Implementation of Convolutional Encoder using VHDL" by Dhananjay Kumar and Shubhra- jyoti Roychoudhury (2016): This paper presents the design and FPGA implementation of a convolutional encoder using VHDL. The authors discuss the architecture of the encoder, the VHDL code used for implementation, and the experimental results obtained from the implementation. The paper also discusses the advantages of using FPGAs for implementing convolutional encoders and the challenges involved in FPGA-based implementation.
2. "Design and Implementation of an FPGA-Based Convolutional Encoder" by Cheng-Chih Chen and Ying-Yu Chen (2017): This paper describes the de- sign and implementation of a convolutional encoder on an FPGA board. The authors discuss the architecture of the

encoder, the VHDL code used for implementation, and the experimental results obtained from the implementation. The paper also compares the performance of the FPGA-based implementation with a software-based implementation and highlights the advantages of using FPGAs for real-time signal processing.

3. "VHDL Implementation of a Convolutional Encoder and Decoder for Wireless Communication Systems" by Xiong Li and Xiaofeng Liu (2019): This paper presents the VHDL implementation of a convolutional encoder and decoder for wireless communication systems. The authors describe the design of the encoder and decoder, the VHDL code used for implementation, and the experimental results obtained from the implementation. The paper also discusses the performance of the implementation in terms of bit error rate (BER) and throughput and compares it with other implementations in the literature.
4. "Design and Implementation of a Viterbi Decoder using FPGA" by Tae Hoon Kim and Jae Won Lee (2017): This paper describes the design and FPGA implementation of a Viterbi decoder for wireless communication systems. The authors discuss the architecture of the decoder, the VHDL code used for implementation, and the experimental results obtained from the implementation. The paper also compares the performance of the FPGA-based implementation with a software-based implementation and highlights the advantages of using FPGAs for real-time signal processing.
5. "Design and Implementation of Convolutional En- coder and Viterbi Decoder using VHDL" by S. Rajalakshmi and S. Sathishkumar (2015): This paper presents the design and VHDL implementation of a convolutional encoder and Viterbi decoder for wireless communication systems. The authors de- scribe the architecture of the encoder and decoder, the VHDL code used for implementation, and the experimental results obtained from the implemen- tation. The paper also discusses the performance of the implementation in terms of BER and throughput and compares it with other implementations in the literature.
6. "Design and Implementation of an FPGA-Based Trellis Encoder" by H. K. Lee and K. M. Yoo (2018): This paper describes the design and FPGA implementation of a trellis encoder for wireless communication systems. The authors discuss the architecture of the encoder, the VHDL code used for implementation, and the experimental results obtained from the implementation. The paper also compares the performance of the FPGA-based implementation with a software-based implementation and high- lights the advantages of using FPGAs for real-time signal processing.
7. "FPGA-Based Design of a High-Speed Trellis En- coder and Decoder for 10 Gbit/s Optical Communication Systems" by Y. Liu, L. Zhang, and Y. Zhang (2017): This paper presents the FPGA-based design of a high-speed trellis encoder and decoder for 10 Gbit/s optical communication systems. The authors describe the design of the encoder and decoder, the VHDL code used for implementation, and the experimental results obtained from the implementation. The paper also discusses the challenges in- volved in implementing high-speed trellis encoders and decoder.

VI. FPGA Prototyping of Trellis Encoder

The design and implementation of an 8-bit trellis encoder on an FPGA board involves several steps. The following is an overview of the process:

1. Design the 8-bit trellis encoder algorithm: The first step is to design the 8-bit trellis encoder algorithm, which involves selecting the appropriate convolutional code and designing the trellis diagram. The goal is to maximize coding efficiency while minimizing complexity and power consumption.
2. Convert the trellis diagram to code: Once the trellis diagram is designed, it needs to be translated into code that can be implemented on the FPGA board. This typically involves using a hardware description language (HDL) such as Verilog or VHDL.
3. Simulate the design: Before implementing the design on the FPGA board, it is important to simulate the design to ensure that it functions as expected. This involves using a software tool such as Model Sim to simulate the design and verify its performance.
4. Implement the design on the FPGA board: Once the de- sign has been simulated and verified, it can be implemented on the FPGA board. This involves using a development board and programming the FPGA with the code generated in step 2.
5. Evaluate the performance of the implementation: After the design has been implemented on the FPGA board, it is important to evaluate its performance in terms of coding efficiency, complexity, and power consumption. This can be done using software tools such as Signal Tap or Chip scope, which allow for real-time monitoring of the FPGA’s performance.

VII. IC Design of Trellis Encoder using Cadence

The design flow for implementing an 8-bit trellis encoder using Cadence typically involves the following steps:

1. Specification and modelling: The first step in the design flow is to specify the functional requirements of the 8-bit trellis encoder and to develop a high- level model of the system. This typically involves using a hardware description language (HDL) such as Verilog or VHDL to describe the functionality of the system.
2. Synthesis and optimization: Once the high-level model is developed, it is synthesized into a gate- level netlist, which can be implemented using physical components such as transistors and capacitors. During this process, the netlist is optimized for performance, power consumption, and area.
3. Physical design: The physical design step involves placing and routing the components on the chip, and optimizing the layout for performance, power consumption, and reliability. This step is critical to ensure that the design meets its performance targets and is manufacturable.
4. Verification: Verification is a critical step in the de- sign flow, as it ensures that the design functions correctly under a range of operating conditions. This involves simulating the design at the gate- level and verifying that it meets its functional and performance requirements.
5. Design for testability: Design for testability involves adding test circuits to the design that enable the chip to be tested for correct operation once it is manufactured. This is an important step to ensure that the chip can be validated and debugged in the event of errors or defects.
6. Manufacturing: Once the design is verified and tested, it can be manufactured using standard semi- conductor manufacturing processes. This typically involves using lithography to transfer the design onto a silicon wafer, and then processing the wafer to create individual chips.

In the context of implementing an 8-bit trellis encoder using Cadence, the above design flow can be adapted as follows:

1. Specification and modelling: This step involves defining the functional requirements of the 8-bit trellis encoder and developing a high-level model of the system using a hardware description language such as Verilog or VHDL.
2. Synthesis and optimization: Once the high-level model is developed, it is synthesized into a gate- level netlist using Cadence tools such as RTL Com- piler or Encounter RTL Compiler. During this pro- cess, the netlist is optimized for performance, power consumption, and area.
3. Physical design: The physical design step involves using Cadence tools such as Encounter Digital Implementation System (EDI) or Innovus to place and route the components on the chip, and optimize the layout for performance, power consumption, and reliability.
4. Verification: Verification is a critical step in the de- sign flow, and involves simulating the design at the gate-level using Cadence tools such as Incisive or Xcelium, and verifying that it meets its functional and performance requirements.
5. Design for testability: This step involves adding test circuits to the design using Cadence tools such as Encounter Test or Test Kompress, which enable the chip to be tested for correct operation once it is manufactured.
6. Manufacturing: Once the design is verified and tested, it can be manufactured using standard semi- conductor manufacturing processes.

VIII. Results and Discussion

There are several factors to consider when comparing the FPGA and IC implementations of an 8-bit trellis encoder.

1. Flexibility: FPGAs are inherently more flexible than ICs since they can be reconfigured to implement different functions or algorithms. Therefore, if there is a need to change the functionality of the system or to implement additional features, it may be easier and more cost-effective to do so on an FPGA. ICs, on the other hand, are designed to implement specific functions and cannot be reconfigured once they are manufactured.
2. Performance: ICs can achieve higher levels of performance compared to FPGAs since they are optimized for specific functions and do not incur the overhead of reconfigurability. ICs can also be optimized for specific manufacturing processes, which can lead to higher speeds and lower power consumption. However, FPGAs can also achieve high levels of performance, especially for applications that require parallel processing or high levels of customization.
3. Cost: ICs are generally more expensive to manufacture than FPGAs since they require a custom manufacturing process. However, once an IC is manufactured in high volumes, the cost per unit can be lower than that of an FPGA. FPGAs, on the other hand, are generally less expensive to prototype and manufacture in low volumes, but the cost per unit can be higher than that of an IC in high volumes.

4. Design time: FPGAs can have shorter design times compared to ICs since they can be programmed and tested quickly. In contrast, ICs require a longer design time since they need to be designed, manufactured, and tested. However, once an IC design is finalized, it can be manufactured in large volumes, leading to a shorter time to market compared to FPGAs.

Advantages of FPGA prototyping:

1. Flexibility: FPGAs are highly flexible and can be programmed to implement different digital circuits, making them suitable for rapid prototyping and testing of various designs.
2. Speed: FPGA-based designs can operate at high speeds since FPGAs can be programmed to perform parallel pro- cessing.
3. Ease of use: FPGAs have a low learning curve, and their programmable nature makes them relatively easy to use.

Advantages of IC design using Cadence:

1. High performance: ICs can achieve high levels of performance since they are optimized for specific applications and can be designed using specialized tools like Cadence.
2. Low power consumption: ICs can consume less power than FPGAs, making them more suitable for power- sensitive applications.
3. Scalability: ICs can be designed to scale up to large-scale applications, making them more suitable for high-volume manufacturing.

IX. Future Research

Suggestions for further optimization of trellis encoder performance: There are several ways to optimize the performance of a trellis encoder. Here are a few suggestions:

1. Use a larger constellation: A larger constellation, such as 16-QAM or 64-QAM, can provide better performance than smaller constellations like BPSK or QPSK. However, larger constellations can be more susceptible to noise and interference, so appropriate error-correction techniques should be used.
2. Increase the constraint length: Increasing the constraint length of the encoder can improve the error-correction capability of the system. However, increasing the constraint length also increases the complexity of the encoder.
3. Use adaptive modulation: Adaptive modulation can ad- just the modulation scheme and coding rate based on the channel conditions. This can provide better performance in situations where the channel conditions vary over time.
4. Use soft decision decoding: Soft decision decoding can provide better performance than hard decision decoding by considering the likelihood of each possible bit value based on the received signal.
5. Use channel coding: Channel coding, such as Reed- Solomon coding or Turbo coding, can improve the error- correction capability of the system. However, channel coding also increases the complexity of the system.
6. Optimize the trellis encoder implementation: The implementation of the trellis encoder can be optimized by using efficient algorithms and hardware design techniques. For example, pipelining and parallel processing can reduce the latency and increase the throughput of the encoder.

Investigating the use of advanced FPGA architectures or IC manufacturing processes:

Investigating the use of advanced FPGA architectures or IC manufacturing processes can offer several benefits for optimizing the performance of a trellis encoder.

Advanced FPGA architectures can offer higher performance and more efficient use of resources compared to traditional FPGAs. For example, some advanced FPGA architectures include dedicated hardware for DSP and memory access, which can significantly improve the performance of digital signal processing applications like trellis encoding. Additionally, advanced FPGA architectures can provide support for high-speed interfaces like PCI Express or Ethernet, which can increase the throughput of the encoder.

On the other hand, IC manufacturing processes can provide higher levels of integration and performance than FPGAs. For example, using advanced IC manufacturing processes like FinFET or 3D stacking can increase the performance and reduce the power consumption of the trellis encoder. Additionally, using IC manufacturing processes can provide more control over the physical layout of the circuit, which can lead to better optimization of performance and power consumption. However, both advanced FPGA architectures and IC manufacturing processes can be more expensive and complex than traditional FPGA or IC design techniques. Additionally, these advanced techniques may require specialized expertise and tools, which can add to the development cost and time.

Exploring the use of other design tools and methodologies for FPGA prototyping and IC design:

Exploring the use of other design tools and methodologies for FPGA prototyping and IC design can offer several benefits for optimizing the performance and reducing the development time of a trellis encoder.

One potential tool for FPGA prototyping is High-Level Synthesis (HLS) tools. These tools allow designers to write code in high-level languages like C or Matlab and automatically convert it to FPGA-compatible code. This can significantly reduce the development time and increase the productivity of the design process. Additionally, some HLS tools can provide more efficient use of resources compared to traditional FPGA design flows, which can lead to higher performance and reduced power consumption.

Another potential tool for IC design is Virtual Prototyping. Virtual Prototyping tools allow designers to create a virtual model of the system and simulate it under various conditions. This can provide a more accurate and comprehensive evaluation of the system performance compared to traditional simulation tools. Additionally, Virtual Prototyping can enable early identification of potential design issues and reduce the risk of costly design errors.

Another methodology that can be explored is System-Level Design. System-Level Design allows designers to design and evaluate the entire system, rather than individual components. This can lead to a more optimized and efficient system design, as the interactions between components can be accounted for in the design process. Additionally, System-Level Design can enable early identification of potential bottlenecks and help optimize the system for the specific requirements of the application.

X. Conclusion

Summary of key findings and contributions:

The primary objective of this study was to investigate the implementation of an 8-bit trellis encoder using FPGA prototyping and IC design methodologies. The study compared the performance and efficiency of these two approaches and explored potential areas for optimization.

The results of the study indicate that both FPGA prototyping and IC design can be effective for implementing a trellis encoder. FPGA prototyping offers flexibility, reusability, and ease of design iteration, while IC design provides higher performance, reduced power consumption, and more optimized layouts.

The study also identified several potential areas for optimization, including exploring the use of advanced FPGA architectures or IC manufacturing processes, investigating other design tools and methodologies, and optimizing the encoder parameters for the specific application requirements.

Overall, this study makes a significant contribution to the field of trellis encoding by providing insights into the implementation of the encoder using FPGA prototyping and IC design methodologies.

Implications and recommendations for future research:

The findings of this study have several implications for future research. One potential area for future research is the investigation of advanced FPGA architectures and IC manufacturing processes for further optimization of the trellis encoder performance. Another area for future research is the exploration of other design tools and methodologies, such as High-Level Synthesis (HLS) tools and System-Level Design.

Additionally, future research could focus on optimizing the encoder parameters for specific applications, including exploring different constraint lengths and code rates. The study also highlights the importance of considering the trade-offs between performance, power consumption, and cost in the design process.

Finally, future research could also investigate the use of trellis encoding in other applications, such as wireless communication systems or image and video compression. Overall, this study provides a foundation for future research in the field of trellis encoding and FPGA prototyping and IC design methodologies.

XI. References:

[1] H. Su and S. S. Bhattacharyya, "An FPGA-Based Implementation of a High-Speed Convolutional Encoder," in IEEE Transactions on VLSI Systems, vol. 21, no. 4, pp. 623-627, April 2013.

[2] Y. Zheng, Y. Xu, H. Yang and H. Wang, "Implementation of a high-speed trellis encoder on FPGA," in Electronics Letters, vol. 52, no. 19, pp. 1604-1606, 15 Sept. 2016.

[3] M. Li, H. Xu and Y. Li, "Design of Trellis Encoder and Decoder Based on FPGA," 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics, Hangzhou, 2015, pp. 76-79.

[4] D. Chen and K. Takeuchi, "Low power and area-efficient trellis-coded modulation with power-scalable code-rate scheme for FPGA-based wireless communication systems," in International Journal of Electronics and Communications, vol. 92, pp. 34-42, May 2018.

[5] A. Nayak and S. Dash, "Design of a High-Speed Trellis Encoder for WLAN," 2015 International Conference on Computing, Communication and Security (ICCCS), Pamplemousses, 2015, pp. 1-6

[6] S. S. Bhattacharyya, "Digital Signal Processing with Field Programmable Gate Arrays," 3rd ed., Springer, 2017.