FEC - Reed Solomon - Aplicado em Verilog

Ausilon V. Souza, Erick Moreira, Lucas S. Ross , Luiz R. L. Rodriges, Nicolas F. Amaral, Rafael A. Reis, and Tarciso G. B. de Bello
Instituto Nacional de Telecomunicações (Inatel)

***Abstract*—Este artigo apresenta a implementação do código Reed-Solomon de Correção de Erros (FEC) em hardware utilizando Verilog. Os códigos Reed-Solomon são amplamente utilizados em sistemas de comunicação para garantir a integridade dos dados, detectando e corrigindo erros durante a transmissão. Descrevemos o projeto dos processos de codificação e decodificação do código Reed-Solomon, com foco na otimização da implementação em hardware para alta taxa de transferência e baixa latência. A implementação em Verilog visa plataformas FPGA, permitindo correção de erros em tempo real em sistemas de comunicação digital. Além disso, o artigo discute os desafios encontrados durante a implementação em hardware e as soluções adotadas .**

***Index Terms*—FEC- Forward Error Correction, Canal , Correção de Erro, Verilog, .**

I. Introdução

O CÓDIGO Reed-Solomon (RS) é um dos mais importantes códigos de correção de erros (FEC - Forward Error Correction), amplamente utilizado em sistemas de comunicação e armazenamento de dados. Desenvolvido por Irving Reed e Gustave Solomon em 1960, este código é baseado em operações matemáticas sobre corpos finitos, conhecidos como campos de Galois (GF), e é projetado para detectar e corrigir múltiplos erros de bits. Sua robustez e eficiência o tornaram uma escolha fundamental em áreas como transmissão digital, armazenamento de dados e sistemas de comunicação de alta performance, onde a preservação da integridade dos dados é crucial [1,2].

A importância da correção de erros em sistemas digitais não pode ser subestimada, especialmente em canais de comunicação sujeitos a ruídos e distúrbios [3]. Códigos como o Reed-Solomon desempenham um papel fundamental ao permitir que os dados corrompidos durante a transmissão sejam restaurados sem a necessidade de retransmissões, o que é essencial em sistemas com alta latência ou limitações de largura de banda. O código RS é particularmente eficaz na correção de erros em blocos de dados, permitindo a recuperação completa dos dados originais, mesmo com múltiplos erros .

Entre suas diversas aplicações, o código Reed-Solomon é encontrado em tecnologias que exigem alta confiabilidade, como CDs, DVDs, sistemas de modulação de comunicação (por exemplo, modems e satélites) e sistemas de armazenamento redundante, como RAID (Redundant Array of Independent Disks) [4]. No contexto dos sistemas de comunicação, o RS é utilizado em tecnologias que operam sob condições adversas, como a comunicação via satélite, onde a atenuação e interferências causadas por fenômenos atmosféricos podem gerar perdas de dados. A flexibilidade e a capacidade de adaptação do código tornam-no indispensável em sistemas que necessitam de correção eficiente, mesmo em ambientes ruidosos e com limitações de recursos .

Além disso, o código RS não é apenas relevante em sistemas de comunicação e armazenamento, mas também em sistemas embarcados e em hardware digital. Sua implementação em dispositivos como FPGAs e ASICs permite que a correção de erros seja realizada em tempo real, com alta taxa de processamento e baixo consumo de recursos [5]. Esses avanços possibilitaram que o código Reed-Solomon fosse aplicado em áreas como a detecção de erros em sinais digitais em tempo real, onde a latência e a confiabilidade são fatores críticos. Em sistemas de comunicação modernos, a combinação de alta capacidade de correção de erros e a implementação eficiente em hardware torna o RS um dos códigos de correção mais versáteis e confiáveis .

Sendo assim, à medida que a demanda por maior eficiência e velocidade nas comunicações cresce, os algoritmos de codificação e decodificação do código Reed-Solomon continuam sendo aprimorados. Novos algoritmos e técnicas de implementação em hardware têm sido propostos para reduzir a complexidade e aumentar a velocidade de processamento, mantendo a eficácia do código. A constante evolução desse campo é fundamental para acompanhar as necessidades emergentes de sistemas de comunicação de alta taxa de erro, como redes 5G e futuros sistemas 6G, onde a correção de erros desempenhará um papel central na confiabilidade das redes de comunicação [6].

O presente artigo está estruturado da seguinte forma: a Seção 2 - Fundamentação Teórica aborda os trabalhos relevantes que discutiram sobre o código Reed-Solomon. A Seção 3 - Implementação descreve detalhadamente a implementação do código Reed-Solomon em *Verilog*, incluindo os processos de codificação e decodificação. Por fim, a Seção 4 - Conclusão apresenta uma análise dos resultados obtidos com a implementação, destacando os desafios enfrentados e as soluções adotadas, além de sugerir direções futuras para a pesquisa e melhorias no sistema. Todo o código utilizado está disponível em https://github.com/emenezes/ReedSolomonVerilog.

II. Fundamentação Teorica

O código Reed-Solomon (RS) é um código de correção de erros baseado em polinômios e operações em corpos finitos, especificamente no campo de Galois (GF). Seu principal objetivo é detectar e corrigir erros em blocos de dados durante a transmissão ou armazenamento, o que é fundamental em sistemas de comunicação digital, como CDs, DVDs, redes de comunicação via satélite, e sistemas de armazenamento de dados. O código RS é amplamente utilizado porque tem

a capacidade de corrigir múltiplos erros, tornando-o robusto contra ruídos e interferências durante a transmissão de dados.

O código RS é definido por três parâmetros principais: n, k e t. O parâmetro n representa o comprimento total do bloco de código, ou seja, o número total de símbolos transmitidos, enquanto k é o número de símbolos de dados, ou seja, a quantidade de informação útil que será transmitida. A diferença entre n e k é o número de símbolos de paridade ou redundância adicionados ao bloco, sendo que o código Reed-Solomon pode corrigir até t erros, onde t é dado por:

$$t = \frac{n-k}{2} \tag{1}$$

Segue abaixo alguns trabalhos relevantes sobre o RS e suas principais contribuições.

O artigo [1] apresenta os códigos de correção de erros Reed-Solomon, que são amplamente utilizados em sistemas de comunicação para garantir a integridade dos dados durante a transmissão. Quando uma fonte transmite n bits, parte desses bits, n-m, são redundantes ou bits de paridade. Esses bits redundantes têm a função essencial de permitir a recuperação da mensagem original, mesmo quando ruídos ou interferências corrompem os dados durante a transmissão, causando erros em alguns bits do código.

Além disso, esses códigos são fundamentais em sistemas que exigem alta confiabilidade, como nos protocolos de correção de erros em códigos QR e em sistemas de armazenamento óptico. Sua eficácia se deve à estrutura matemática robusta que permite a detecção e correção de múltiplos erros simultaneamente, aumentando a eficiência da comunicação. Em comparação com outros métodos de detecção e correção de múltiplos erros, os códigos Reed-Solomon se destacam pela sua flexibilidade e por serem capazes de operar de maneira eficaz mesmo em canais de comunicação ruidosos, tornando-os uma escolha preferencial em diversas tecnologias modernas

Assim como apresentado no [7] o código Reed-Solomon RS (204,188) é utilizado em três padrões: satélite (DVB-S), cabo (DVB-C) e terrestre (DVB-T). Os polinômios geradores de código para o padrão sem fio 802.16 são apresentados neste artigo, sendo pré-definidos no codificador. O decodificador Reed-Solomon utiliza o código (255,247) para verificar a funcionalidade do código RS no padrão IEEE 802.16. O codificador e o decodificador Reed-Solomon foram implementados no dispositivo "xc4vlxI5-12-sf363" como alvo. Essa implementação atende a todas as restrições de tempo.

No trabalho [8] os códigos Reed-Solomon foram utilizados para avaliar o desempenho do decodificador RS nos FPGAs Xilinx Virtex II Pro (xc2vp50-5-ff1148) e Xilinx Spartan 3e (xc3s500e-4-fg320). Um sistema de correção de erros para frente, baseado no padrão de rede 802.16, é empregado para melhorar o desempenho geral do sistema. O padrão de rede 802.16 recomenda o uso do código Reed-Solomon RS (255,239). O código foi implementado em linguagem de descrição de hardware, ou seja, VHDL. O decodificador Reed-Solomon, implementado no FPGA Virtex II Pro, resultou em uma significativa economia de área, o que representa uma redução no custo e na complexidade do hardware do sistema.

No trabalho [9], é apresentada a implementação do código RS(204,188) para o sistema DVB e do código RS(255,239) para o sistema IEEE 802.16 em FPGA Virtex4. Foi desenvolvido um sistema de codificação concatenada baseado no código RS(204,188), utilizando a linguagem de descrição de hardware VHDL, de alta velocidade, e sintetizado para o chip FPGA. A implementação do CODEC Reed-Solomon demonstrou resultados superiores, com desempenho aprimorado em relação a outras soluções semelhantes. A frequência máxima atingida pelo codificador foi de 308,5 MHz, enquanto o decodificador operou a 145,6 MHz, proporcionando uma significativa eficiência no processamento de dados. Esses resultados evidenciam o potencial do FPGA Virtex4 para otimizar sistemas de codificação, sendo uma solução robusta para aplicações em sistemas de comunicação de alta performance. Além disso, a utilização de VHDL garantiu maior flexibilidade e velocidade na implementação, atendendo aos requisitos de tempo real exigidos por esses sistemas complexos.

III. Implementação

Para implementar o Reed-Solomon em *Verilog*, foram adotados os seguintes elementos: Divisor de Clock, Gerador de Bit Aleatório, Codificador de Reed Solomon, Conversor Serial/Paralelo, Canal, Conversor Paralelo/Serial, Decodificador de Reed Solomon, Comparador e a Testbench Geral. A figura 1 ilustra os modúlos informados anteriomente.

A. Divisor de Clock

Divisor de Clock: Tem a função de reduzir a frequência do sinal de clock principal, gerando sinais de temporização apropriados para os demais blocos do sistema implementado em *Verilog*. Isso garante que cada componente opere de forma sincronizada, evitando falhas de temporização. Seu uso é fundamental em projetos digitais, especialmente quando diferentes módulos exigem clocks com taxas distintas. A principal vantagem é permitir um controle preciso do tempo de operação de cada estágio. Além disso, facilita testes e depuração em hardware reconfigurável como FPGAs.

B. Gerador de Bit Aleatório

Gerador de Bit Aleatório: Responsável por produzir uma sequência de bits pseudoaleatórios, simulando dados de entrada para o sistema de codificação Reed-Solomon. Seu objetivo é validar o funcionamento do codificador e decodificador sob condições reais de uso, com entradas variáveis. É essencial para testar a robustez do sistema frente a diferentes padrões de entrada. A aleatoriedade introduzida permite observar o comportamento do sistema em situações imprevisíveis. Isso ajuda na verificação da confiabilidade da implementação em *Verilog*.

C. Codificador Reed-Solomon

Codificador Reed-Solomon: Implementado em *Verilog*, recebe dados em formato paralelo e adiciona símbolos de paridade, gerando um bloco de código capaz de detectar e corrigir múltiplos erros. Seu objetivo é aumentar a confiabilidade

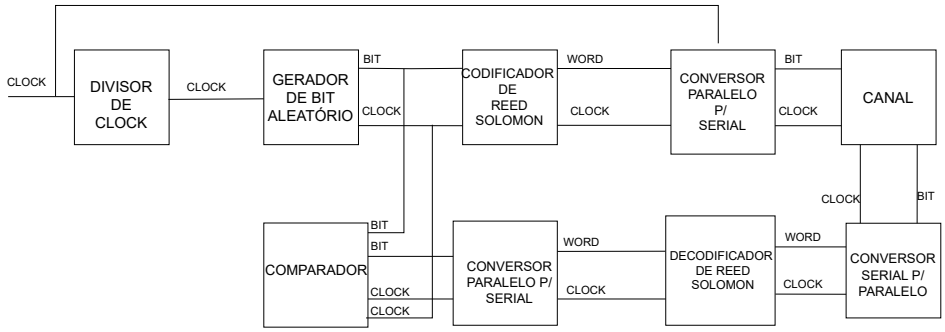

Figure 1: Módulos do FEC Reed Solomon

da transmissão de dados em ambientes ruidosos. É um dos códigos mais utilizados em sistemas digitais, como DVB, CDs e redes sem fio, devido à sua alta eficiência na correção de erros. A codificação é feita por operações polinomiais em campos de Galois. Sua principal vantagem é permitir a correção sem necessidade de retransmissão. Além disso o codificador Reed-Solomon recebe um bloco de k símbolos de dados e gera $n - k$ símbolos de paridade. O processo de codificação pode ser descrito de forma simplificada: a entrada de dados é representada como um polinômio $P(x)$, e o polinômio gerador $g(x)$ é utilizado para gerar os símbolos de paridade. O resultado da divisão do polinômio $P(x)$ por $g(x)$ gera os coeficientes do polinômio de paridade. Esse processo de codificação envolve a multiplicação polinomial no campo de Galois, onde cada coeficiente do polinômio é um símbolo e, portanto, cada símbolo gerado no processo de codificação corresponde a um valor no campo finito, assegurando a integridade dos dados e possibilitando a correção de erros na decodificação.

D. Conversor Serial/Paralelo

Conversor Serial/Paralelo: Converte a sequência de bits recebida serialmente em palavras de dados paralelas, adequadas para o processamento pelo codificador. Sua função é adaptar o formato dos dados de entrada para a estrutura de palavras exigida pelo Reed-Solomon. Esse bloco é crucial em sistemas onde os dados chegam em série, como na maioria dos canais de comunicação. A vantagem dessa conversão é a otimização da codificação em termos de velocidade e organização. Facilita também a integração com barramentos internos do sistema.

E. Canal

Canal: Representa o meio físico ou lógico por onde os dados codificados são transmitidos, sendo suscetível a ruídos e interferências que causam erros nos bits. No contexto da simulação em *Verilog*, o canal pode ser modelado para inserir erros intencionais e testar a capacidade de correção do sistema. Sua função principal é simular as condições reais de transmissão. A análise do comportamento do sistema diante de erros no canal é essencial para validar sua robustez. Assim, o canal é parte fundamental na verificação funcional do codec.

F. Conversor Paralelo/Serial

Conversor Paralelo/Serial: Tem a função de converter os dados codificados, originalmente em paralelo, em uma sequência serial de bits para transmissão eficiente pelo canal. Esse bloco é essencial em sistemas que utilizam transmissão serial para economizar pinos de I/O ou largura de banda. Sua implementação correta garante que os dados cheguem ao destino no formato esperado. A vantagem está na simplicidade do cabeamento e compatibilidade com a maioria dos canais físicos. Também reduz o custo e a complexidade do hardware de transmissão.

G. Decodificador Reed-Solomon

Decodificador Reed-Solomon: Reverte o processo de codificação, corrigindo erros e recuperando os dados originais a partir dos símbolos recebidos. Implementado em *Verilog*, realiza operações matemáticas complexas sobre campos finitos para identificar e corrigir erros. Sua função é fundamental para garantir a integridade dos dados recebidos. Ele permite a correção de múltiplos erros sem retransmissão, o que é uma grande vantagem em sistemas com baixa tolerância a perdas. Sua eficiência torna o sistema mais confiável e robusto.

H. Comparador

Comparador: Compara os dados decodificados com os dados originais gerados pelo gerador de bits aleatórios, verificando se o sistema corrigiu corretamente os erros inseridos no canal. Essa etapa é crucial para validar o funcionamento do codificador e decodificador Reed-Solomon implementados em *Verilog*. O objetivo do comparador é fornecer uma métrica objetiva de desempenho do sistema. Sua presença permite detectar falhas na lógica de correção ou na temporização. É uma ferramenta essencial na fase de testes e validação do projeto.

I. Testbench Geral

Um testbench é um ambiente de simulação usado para verificar se um projeto digital descrito em Verilog (HDL) está funcionando corretamente. A ideia é testar o projeto antes de fabricar o hardware físico, economizando tempo e dinheiro. O testbench simula o comportamento do projeto em diferentes situações e com diferentes entradas, ajudando

a identificar erros (bugs), validar funcionalidades e ajustar o design conforme necessário.O módulo que queremos testar é geralmente chamado de DUT (Design Under Test), e ele é instanciado dentro do testbench. O próprio testbench é um módulo Verilog separado, de nível superior, responsável por gerar os sinais de entrada, observar as saídas e compará-las com os valores esperados.Para isso, o testbench cria diferentes padrões de entrada, como sinais de clock, reset e outros sinais de controle, para simular várias condições e situações-limite. Ele pode usar funções e tarefas Verilog para organizar esses testes de forma mais clara e reutilizável.

Os sinais conectados à entrada do projeto podem ser chamados de "sinais de acionamento", enquanto os sinais conectados à saída do projeto podem ser chamados de "sinais de monitoramento". O sinal de acionamento deve ser do tipo reg, pois pode conter um valor e é atribuído principalmente ao bloco procedural (initials e always). Os sinais de monitoramento devem ser do tipo wire que recebem o valor direcionado pelo DUT.Os sinais do testbench são conectados às portas do DUT, e a resposta do projeto é monitorada ao longo do tempo para garantir que ele esteja se comportando como o esperado. Em projetos simples, como uma trava digital, esse processo pode ser feito em uma única função chamada logo após a aplicação dos sinais de entrada.

Com a simulação feita por um testbench, é possível validar o projeto, identificar possíveis problemas e fazer ajustes antes de passar para a fase de implementação física. Por isso, os testbenches são uma parte essencial no desenvolvimento de circuitos digitais, garantindo que tudo funcione corretamente desde o início.

IV. Resultados e Conclusões

A implementação dos blocos funcionais em *Verilog*, incluindo o Divisor de Clock, Gerador de Bit Aleatório, Conversores Serial/Paralelo, Codificador e Decodificador Reed-Solomon, mostrou-se eficaz e coerente com os objetivos do projeto. Cada módulo foi testado isoladamente e em conjunto, evidenciando sincronização adequada dos sinais e correto fluxo de dados. A arquitetura modular permitiu uma fácil integração dos blocos e validação progressiva do sistema. A geração controlada de erros no canal simulou com sucesso condições reais de transmissão. A abordagem de testes funcionais via simulação demonstrou boa robustez e confiabilidade. O ambiente de verificação ModelSim e testbench facilitou a análise detalhada do comportamento dos sinais.

Os resultados obtidos indicam que o Codificador(7,3) Reed-Solomon, mesmo com sua complexidade algébrica, pode ser eficientemente sintetizado em linguagem de hardware, como é o caso do *Verilog*. O sistema corrigiu com sucesso erros inseridos no canal, e o Decodificador recuperou corretamente os dados originais em todos os casos de teste simulados. O Comparador confirmou a eficácia do processo, demonstrando que o erro foi corrigido dentro da capacidade de correção prevista pelo código. Além disso, os conversores de dados atuaram conforme o esperado, mantendo a integridade dos dados entre as conversões serial e paralela. O sistema mostrou potencial para aplicações reais como FPGAs com ambientes ruidosos e de comunicação crítica.

Conclui-se que a implementação do sistema de codificação e decodificação Reed-Solomon em *Verilog* é viável, eficaz e apresenta excelente desempenho em termos de correção de erros. A modularidade dos blocos facilita a reutilização e expansão para sistemas maiores. Como trabalhos futuros, propõe-se a inclusão de uma interface de controle via UART ou SPI, além de otimizações para reduzir o consumo de recursos lógicos em FPGAs. A aplicação prática em sistemas de comunicação digital, armazenamento e transmissão de dados pode se beneficiar dessa abordagem, agregando confiabilidade e robustez ao sistema final.

References

[1] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[2] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*. Elsevier, 1977, vol. 16.

[3] X. Xiao, B. Vasić, S. Lin, K. Abdel-Ghaffar, and W. E. Ryan, "Reed-solomon based quasi-cyclic ldpc codes: Designs, girth, cycle structure, and reduction of short cycles," *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5275–5286, 2019.

[4] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," in *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, 1988, pp. 109–116.

[5] G. M. Almeida, E. A. Bezerra, L. V. Cargnini, R. D. R. Fagundes, and D. G. Mesquita, "A reed-solomon algorithm for fpga area optimization in space applications," in *Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*. IEEE, 2007, pp. 243–249.

[6] R. Ferrara, L. Torres-Figueroa, H. Boche, C. Deppe, W. Labidi, J. U. Moenich, and V.-C. Andrei, "Implementation and experimental evaluation of reed-solomon identification," in *European Wireless 2022; 27th European Wireless Conference*. VDE, 2022, pp. 1–6.

[7] L. Chaari, M. Fourati, N. Masmoudi, and L. Kamoun, "A reconfigurable fec system based on reed-solomon codec for dvb and 802.16 network," *WSEAS transactions on circuits and systems*, vol. 8, no. 8, pp. 729–744, 2009.

[8] B. Tiwari and R. Mehra, "Design and implementation of reed solomon decoder for 802.16 network using fpga," in *2012 IEEE International Conference on Signal Processing, Computing and Control*. IEEE, 2012, pp. 1–5.

[9] B. Tiwari and M. Rajesh, "Fpga implementation of rs codec for digital video broadcasting," *VSRD-IJEECE Journal*, vol. 2, no. 2, pp. 68–77, 2012.