Implementação VLSI e em FPGA de um decodificador baseado em conjuntos de infomação

Jefferson Rodrigo Schuertz[1], Sibilla Batista da Luz França[2]
Universidade Federal do Paraná (UFPR)
Curitiba – PR, Brasil
jeffersonschuertz@ufpr.br[1], sibilla@eletrica.ufpr.br[2]

Resumo— Visando manter a integridade do grande volume de dados em circulação nos sistemas de comunicação, códigos corretores de erros são frequentemente utilizados, permitindo identificar e corrigir erros nas mensagens transmitidas ou armazenadas. Diversos algoritmos para decodificação de dados têm sido abordados na literatura nos últimos anos. Este artigo apresenta a implementação em hardware de um decodificador baseado em conjuntos de informação para códigos de blocos. O método escolhido tem desempenho similar ao método de decodificação por máxima verossimilhança (MLD). A técnica consiste em extrair os símbolos mais confiáveis para, então, gerar um conjunto reduzido de palavras-código candidatas, diminuindo assim o número de comparações para recuperar a mensagem. Sua implementação apresenta menor complexidade, exigindo menos recursos de hardware. O circuito foi inicialmente descrito em VHDL e implementado na FPGA Virtex 5. Posteriormente, fez-se a síntese e o layout de alguns blocos do decodificador na tecnologia 130 nm.

I. INTRODUÇÃO

Atualmente, para algumas aplicações, é imperativo garantir que os dados sejam transmitidos, recebidos e armazenados de modo confiável e íntegro. No entanto, os efeitos de canal podem afetar a integridade dos dados em algum ponto, seja durante o armazenamento ou transmissão [1]. Nesse cenário, é comum usar códigos de correção de erros (ECCs - *Error Correcting Codes*). Esses códigos foram criados para melhorar o desempenho da comunicação, corrigindo erros decorrentes de eventos como a interferência e o ruído. Para recuperar a informação, eventualmente corrompida, bits de redundância são inseridos na mensagem durante o processo de codificação, permitindo, no processo de decodificação, a sua recuperação, desde que o número de erros não exceda o limite de correção do código [2-3].

Os ECCs podem ser divididos em duas categorias principais: códigos de bloco e códigos convolucionais [3]. Tradicionalmente, os códigos de bloco, abordados neste trabalho, são representados como **C**(*n*, *k*), onde *n* é o número total de bits da palavra-código e *k* o número de bits de informação, sendo *n-k* o número de bits de redundância.

Uma representação da utilização de um ECC é apresentada na figura 1. Nela observa-se que na fonte de mensagem uma mensagem **u** é gerada e, então, codificada na palavra-código **c**. Posteriormente, a palavra-código é transmitida em um canal ruidoso. A palavra-código recebida, possivelmente corrompida, é denominada **c***. Devido aos bits de redundância e ao uso de um algoritmo de decodificação, a mensagem pode ser restaurada pelo decodificador, disponibilizando assim **u**' ao receptor.

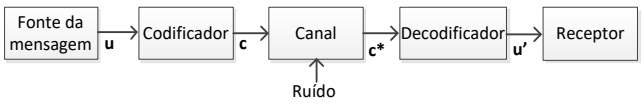

Fig. 1. *Uso de códigos corretores de erros.*

A decodificação por máxima verossimilhança (MLD - *Maximum Likelihood Decoding*) é um algoritmo de decodificação ótimo. Entretanto, é custoso, devido ao elevado número de comparações necessárias. Sendo assim, algoritmos alternativos, com desempenhos próximos ao MLD, têm sido extensivamente estudados na literatura [3-11]. Alguns desses algoritmos são baseados em conjuntos de informação (IS - *Information-Set*) [4-11]. Um conjunto de informações é qualquer conjunto de *k* colunas linearmente independentes de uma matriz geradora **G** [4].

Em decodificadores baseados em IS os símbolos mais confiáveis da mensagem são usados para obter um conjunto de informações e, em seguida, gerar um conjunto de palavras-código candidatas mais prováveis. Esse conjunto reduzido permite diminuir o número de comparações para escolher a melhor candidata, enquanto o método MLD exige 2^k comparações.

Este trabalho apresenta a implementação em hardware do algoritmo de decodificação proposto em [6], contudo, sem estabelecer um critério de parada. O decodificador projetado foi inteiramente implementado em FPGA para diferentes tamanhos de código. Posteriormente, três dos

quatro blocos foram projetados como um circuito integrado dedicado (ASIC).

O artigo está estruturado da seguinte forma. Na seção II apresenta-se uma descrição do algoritmo escolhido. A seção III descreve a arquitetura dos blocos construtivos do decodificador. Em seguida, a seção IV apresenta o desenvolvimento e os resultados obtidos. Finalmente, a seção V apresenta a conclusão.

II. ALGORITMO DE DECODIFICAÇAO

O algoritmo proposto em [6] faz uso dos símbolos mais confiáveis da palavra-código recebida e, através de sucessivas operações algébricas, produz um conjunto de k+1 palavras-código candidatas, selecionando, por fim, a mais provável. Este algoritmo é descrito a seguir. Considera-se que a palavra-código **c**, gerada no codificador, foi modulada em BPSK e transmitida através de um canal ruidoso. Os símbolos que compõe a palavra-código recebida **c*** são quantizados em três bits, portanto, representando-os como um valor inteiro, cada símbolo de **c*** pode assumir o valor entre 0 e 7. Os valores dos símbolos mais confiáveis são 0 e 7, enquanto os valores dos símbolos menos confiáveis são 3 e 4. O funcionamento do algoritmo, para um código **C**(n, k) pode ser descrito em 9 passos:

1. Construir o vetor **r**, que corresponde à decodificação por decisão abrupta de **c***. Os símbolos com os valores entre 0 e 3 são considerados valor lógico '0' e, os valores entre 4 e 7, como valor lógico '1'.
2. Construir o vetor **s**, o qual deve armazenar os símbolos de **c*** em ordem decrescente de confiabilidade.
3. Selecionar os k símbolos mais confiáveis de **r**, segundo o vetor de confiabilidade **s**, formando o vetor $\mathbf{r}_{parcial}$.
4. Formar a matriz denominada $\mathbf{G}_{parcial}$, de tamanho $k \times k$, que é composta por k colunas da matriz gerador **G**, de tamanho $k \times n$. As colunas escolhidas de **G** são indicadas pelo vetor de confiabilidade **s**.
5. Checar se $\mathbf{G}_{parcial}$ possui uma matriz inversa. Caso não possua, outra matriz parcial deve ser formada utilizando um conjunto diferente de símbolos mais confiáveis.
6. Realizar a inversão da matriz $\mathbf{G}_{parcial}$.
7. Gerar a matriz denominada $\mathbf{G}_{nova}$, de tamanho $k \times n$, através do produto matricial entre $\mathbf{G}_{parcial}^{-1}$ e **G**.
8. Produzir o conjunto de k+1 palavras-código candidatas. A primeira é obtida através do produto entre $\mathbf{G}_{nova}$ e $\mathbf{r}_{parcial}$ e as demais pelo produto entre $\mathbf{G}_{nova}$ e o bit-flipping de cada posição de $\mathbf{r}_{parcial}$.
9. Selecionar a palavra-código vencedora, isto é, a que apresentar a menor distância suave em relação a **c***.

III. DESENVOLVIMENTO E RESULTADOS

A arquitetura proposta para o decodificador é composta por quatro blocos, conforme apresentado na figura 2. O Bloco I, tendo como entrada **c***, realiza os passos 1 e 2 do algoritmo, disponibilizando em sua saída os sinais **r** e **s**. O Bloco II, tem seus sinais de entrada provenientes da saída do Bloco I e realiza os passos de 3 até 7, disponibilizando em sua saída os sinais $\mathbf{r}_{parcial}$ e $\mathbf{G}_{new}$. O Bloco III, recebe os sinais do Bloco II e efetua o passo 8. Finalmente, o Bloco IV recebe a saída do Bloco III e realiza o passo 9 do algoritmo.

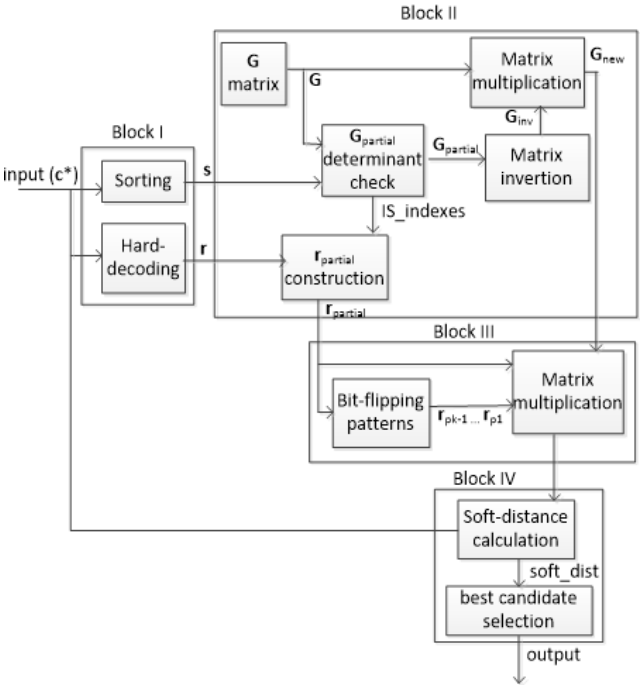

Fig. 2.*Arquitetura do decodificador.*

O decodificador mostrado na figura 2 foi implementado em VHDL e sintetizado para uma FPGA Virtex 5. Após sua validação realizou-se o design VLSI dos Blocos I, II e III a partir mesmo código VHDL. Estas implementações e seus resultados são apresentados a seguir.

A. Implementação em FPGA

O Bloco I realiza as primeiras etapas do algoritmo. Este bloco faz a decodificação por decisão abrupta da palavra-código **c*** e a atribuição de níveis de confiança a cada símbolo. Para isso, utilizou-se uma escala entre 0 e 3, onde o nível 0 é o menos confiável e 3 o mais confiável. Para ordenar os símbolos mais confiáveis de modo decrescente, o circuito de classificação clássico baseado no algoritmo *Insertion Sort* foi usado [13]. O circuito é apresentado na figura 3, onde *d* é o símbolo a ser inserido e **s** corresponde ao vetor em processo de ordenação.

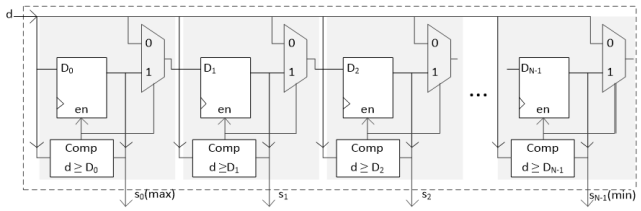

Fig. 3.*Método de ordenação Insertion Sort.*

O vetor **s** foi implementado como uma coleção de dados, contendo duas informações: o nível de confiabilidade de cada símbolo e seu índice. O número de ciclos necessários na ordenação é igual ao número de símbolos da palavra-código **c***. O consumo de recursos lógicos, a frequência de operação e a latência podem ser verificados na Tabela 1.

TABELA 1. IMPLEMENTAÇÃO EM FPGA DO BLOCO I

Código	LUTs	Registradores	Fmax (MHz)	Latência (ciclos)
C(7,4)	84	40	336.86	7
C(15,7)	232	66	245.5	15

O Bloco II realiza a construção da matriz $\mathbf{G}_{\text{parcial}}$ utilizando as *k* colunas de **G**, cujos índices correspondem aos primeiros *k* elementos do vetor **s**, consumindo apenas um ciclo de *clock*. Em seguida, verifica se $\mathbf{G}_{\text{parcial}}$ possui uma matriz inversa. Sendo assim, mais *k*-1 ciclos de *clock* são necessários. O processo consiste na redução da matriz pelo método de eliminação de Gauss Jordan para, então, obter o determinante. Se a matriz for reversível (determinante igual a '1') avança-se para a etapa posterior, caso contrário, mais *k* ciclos são necessários para se construir e validar outra matriz parcial.

O processo para verificar se $\mathbf{G}_{\text{parcial}}$ é reversível foi subdividido em três passos:

1. Verificar se o pivô, elemento Xi,j é igual a '1'. Caso não seja, deve-se trocar por outra linha que tenha '1' na posição na coluna do pivô
2. Verificar se há elementos diferentes de '0' nas posições abaixo e acima do pivô.
3. Realizar a operação lógica XOR entre a linha do pivô e as linhas identificadas na etapa anterior.

Em seguida, o Bloco II realiza a inversão de $\mathbf{G}_{\text{parcial}}$, tal operação, ocorre por meio de um algoritmo aprimorado do método de eliminação de Gauss Jordan [12]. São necessários *k* ciclos de *clock* para realizar a inversão. O método consiste em gerar uma matriz de maior ordem ($k \times 2k$), onde as primeiras *k* colunas são idênticas as de $\mathbf{G}_{\text{parcial}}$, já as demais equivalem as colunas de uma matriz identidade de ordem *k*. Cada passo do processo de inversão é apresentado a seguir.

1. Ajustar o pivô, nesse algoritmo o pivô sempre será o elemento $X_{1,1}$.
2. Verificar se há elementos diferentes de '0' nas linhas abaixo do pivô e executar a operação lógica XOR entre a linha do pivô e as linhas que apresentarem elementos iguais a '1'.
3. Executar a operação denominada *shift left-up*, ou seja, cada elemento da matriz deve ser movido uma posição para a esquerda e uma para cima.

Obtida a inversa, é realizado o produto matricial entre $\mathbf{G}_{\text{partial}}^{-1}$ e **G**, originando a matriz $\mathbf{G}_{\text{nova}}$. Os resultados da implementação podem ser verificados na Tabela 2.

TABELA 2. IMPLEMENTAÇÃO EM FPGA DO BLOCO II.

Código	LUTs	Registradores	Fmax (MHz)	Latência (ciclos)
C(7,4)	497	55	168.0	≥ 8
C(15,7)	1546	169	105.6	≥ 14

O Bloco III gera o conjunto reduzido de *k*+1 palavras-código candidatas. A primeira palavra-código candidata, $\mathbf{c}_0$, é obtida pelo produto entre $\mathbf{G}_{\text{nova}}$ e $\mathbf{r}_{\text{parcial}}$. As demais *k* palavras-código candidatas são obtidas pelo produto entre $\mathbf{G}_{\text{nova}}$ e as variações de $\mathbf{r}_{\text{parcial}}$ através da técnica de *bit-flipping*. Os resultados da síntese são apresentados na Tabela 3.

TABELA 3. IMPLEMENTAÇÃO EM FPGA DO BLOCO III.

Código	LUTs	Registradores	Fmax (MHz)	Latência (ciclos)
C(7,4)	106	46	565.0	1
C(15,7)	302	136	312.0	1

O bloco IV compara cada uma das palavras-código candidatas com **c***, para então determinar a mais semelhante, isto é, menor distância suave. Esta tarefa consome apenas um ciclo de *clock*. Os resultados da síntese são apresentados na Tabela 4.

TABELA 4. IMPLEMENTAÇÃO EM FPGA DO BLOCO IV.

Código	LUTs	Registradores	Fmax (MHz)	Latência (ciclos)
C(7,4)	43	13	598.1	1
C(15,7)	149	32	321.5	1

Finalmente, na Tabela 5, são apresentados os resultados para a implementação em FPGA do decodificador completo para os códigos **C**(7, 4) e **C**(15, 7).

TABELA 5. IMPLEMENTAÇÃO DO DECODIFICADOR.

Código	LUTs	Registradores	Fmax (MHz)	Latência
C(7,4)	568	161	168.0	≥ 17
C(15,7)	2080	352	105.6	≥ 29

B. Síntese lógica e layout dos blocos para um circuito integrado

As sínteses lógicas dos Blocos I, II e III foram realizadas para o código **C**(7, 4) utilizando a ferramenta *Genus* da *Cadence Systems* na tecnologia GF BiCMOS 8HP 130 nm. Na Tabela 6 são apresentados os resultados de área,consumo de potência e o número de células para cada bloco. A netlist gerada pelo *Genus* foi simulada no

ModelSim e comparada com os resultados de simulação em FPGA.

TABELA 6. RESULTADOS DA SÍNTESE LÓGICA PARA BICMOS 130 NM.

Recursos	Bloco I	Bloco II	Bloco III
Número de células	187	698	232
Área consumida (μm^2)	3007	7143	3142
Potência consumida (mW)	1.46	5.70	3.83

Os layouts dos blocos I, II e III para o código **C**(7, 4) foram realizados através da ferramenta *Innovus*. A Tabela 7 mostra os resultados da síntese física. O layout de cada bloco foi validado por meio de verificações de geometria, conectividade e de DRC. As *netlists* pós-layout geradas foram simuladas e validadas usando a ferramenta ModelSim.

TABELA 7. RESULTADOS DA SÍNTESE FÍSICA PARA BICMOS 130 NM.

	Dimensões (μm)	Área (μm^2)	Fmax (MHz)	Densidade (%)
Bloco I	148 × 136	19,856	70	78.16
Bloco II	152 × 182	28,756	50	92.0
Bloco III	135 × 123	16,680	75	69.54

IV. CONCLUSÃO

Este trabalho apresentou a implementação completa em FPGA e o design VLSI parcial de um decodificador baseado em conjuntos de informação. Uma arquitetura composta por quatro blocos foi proposta, na qual o circuito realiza a decodificação abrupta da mensagem recebida, extrai a confiabilidade dos símbolos, constrói uma matriz baseada nos símbolos mais confiáveis, gera um conjunto de palavras-código candidatas mais prováveis e finalmente determina a melhor candidata. Os resultados mostraram que o decodificador foi implementado de forma eficiente em FPGA e ASIC, consumindo um número reduzido de recursos lógicos. Como trabalho futuro, pretende-se finalizar o design VLSI do bloco IV e unir todos os blocos para obter o decodificador completo como um ASIC.

REFERÊNCIAS

[1] B. Sklar, "Digital Communications. Fundamentals and Applications", 2nd Prentice Hall, 2017.

[2] S. Lin e D.J. Costello, "Error Control Coding: Fundamentals and applications", 2nd ed. Prentice Hall, Junho, 2004.

[3] M. Fossorier, S. Lin, "Soft-decision decoding of linear block codes based on order statistics" Transactions on Information Theory, Vol. IT – 4I, No. 5, pp. 13791396, Sep. 1995.

[4] E. Prange, "The use of information sets in decoding cyclic codes" IEEE Transactions on information Theory, Vol. IT-8, pp. 5-9, Sep. 1962.

[5] Dorsch, B. "A decoding algorithm for binary block codes and J-ary output channels (Corresp.)," IEEE Transactions on Information Theory, vol. 20, no. 3, pp. 391-394, May 1974, doi: 10.1109/TIT.1974.1055217. Trans, Vol. IT-20, No. 3, Maio 1974, pp. 391–4.

[6] Brante, G. G. de O, Godoy W, Muniz, D.; "Information Set Based Soft-Decoding Algorithm for Block Codes". IEEE Latin America Transactions, Latin-American Conference on Communications, Bogota, 2010.

[7] Gortan, A.; Godoy, W. Jr.; Jasinski, R. P.; Pedroni, V. A. "Achieving Near-MLD Performance with Soft Information-Set Decoders Implemented in FPGAs". IEEE Asia Pacifc Conference on Circuits and Systems (APCCAS), Kuala Lumpur, 2010.

[8] Jasinski, R. P.; Pedroni, V. A. "Achieving Near-MLD Performance with Soft Information-Set Decoders Implemented in FPGAs". IEEE Asia Pacifc Conference on Circuits and SystemsCircuits and Systems (APCCAS), Kuala Lumpur, 2010.

[9] Scholl S. e N. When, "Hardware implementation of a Reed-Solomon soft decoder based on information set decoding". Design Automation and Test in Europe, Dezembro 2013.

[10] Karthik G.R. "Modified error insertion technique for information set based decoders" Conference: Advances in Computing, Communications, and Informatics (ICACCI), 2013 International Conference, Julho 2013.

[11] Versfeld J., Y. Genga e O. Overinde. "Improved Iterative Convergence Rate for Soft-Decision Bit-Level Reed-Solomon Decoders using Information Set Decoding". IEEE Wireless Africa Conference (WAC), Pretoria, Agosto 2019.

[12] Pedroni, V. A, Gortan A., Godoy W. Jr. "An improved GF(2) Matrix Inverter with Linear Time Complexity" Internaticional Conference on Reconfigurable Computing, 2010.

[13] Ribas, L., Castellis, D., Carrabina, J. "A linear sorter core based on programmable register file". Conference on Design of Circuits and Integrated Systemas (DCIS'04), 2004.