Design de Blocos Construtivos de um Decodificador de Informação Baseado em Conjuntos de Informação

Jefferson Rodrigo Schuertz, Sibilla Batista da Luz França
Universidade Federal do Paraná (UFPR)
Curitiba – PR, Brasil
jeffersonschuertz.eng@gmail.com, sibilla@eletrica.ufpr.br

***Resumo*— A recente popularização dos dispositivos móveis provocou um grande aumento no volume de dados que passaram a circular nos sistemas de comunicação. Neste cenário, os códigos corretores de erros são usados frequentemente, permitindo identificar e corrigir erros nas mensagens recebidas ou armazenadas. Este trabalho apresenta a implementação em *hardware* de dois blocos construtivos de um decodificador, baseado em conjuntos de informação para códigos de blocos. O algoritmo de decodificação apresenta desempenho similar à decodificação por máxima verossimilhança (MLD), porém, com melhor desempenho. Para isso, são utilizados os símbolos mais confiáveis da mensagem recebida, reduzindo o número de comparações necessárias durante o processo de decodificação. A arquitetura proposta foi concebida para lidar com diversos códigos com uso eficiente dos recursos de *hardware*. O primeiro bloco foi concebido para realizar a demodulação da mensagem recebida e ordenar os símbolos mais confiáveis, enquanto o segundo bloco obtém uma matriz ($G_{nova,}$) utilizada para gerar as palavras-código candidatas. O circuito foi descrito em VHDL para a FPGA Virtex 5, além de ter sido sintetizado o esquemático para um circuito dedicado. Foram utilizados 574 LUTs (*Lookup Tables*) e 208 registradores da FPGA para um código de Hamming C (7, 4).**

I. INTRODUÇÃO

As mudanças ocorridas nas últimas décadas, promovidas por grandes avanços tecnológicos, impactaram profundamente o modo de se viver e de se comunicar, dispositivos móveis tornaram-se difundidos em grandes números, sendo que, no cenário atual, bilhões de pessoas estão conectadas diariamente pela internet.

Atualmente é imprescindível garantir que o grande volume de dados que circula diariamente seja armazenado, transmitido e recebido de maneira eficiente. Sendo assim, caso os dados sejam corrompidos em qualquer uma das etapas pelos efeitos de canal, tais como o ruído, interferência e desvanecimento, estes devem ser recuperados. Para isso, é recorrente o uso dos códigos corretores de erros [1], onde insere-se redundâncias em uma sequência de dados antes do armazenamento e transmissão para que, na decodificação, erros sejam identificados e corrigidos.

Na Figura 1, é exemplificado o funcionamento dos códigos corretores de erros, nela verifica-se uma mensagem binária **u** formada por *k* bits (*k* = 6) que ao ser codificada, isto é, acrescentar bits de redundância, resulta em uma palavra-código **c** de *n bits* (*n* = 8). Após a codificação, **c** é transmitida através de um canal ruidoso. A mensagem recebida **c*** pode estar corrompida, ou seja, ter sofrido uma alteração. Sendo assim, **c*** é submetido a um algoritmo de decodificação que retorna, enfim, a mensagem original **u**, desde que o número de erros da mensagem não ultrapasse a capacidade de correção do código.

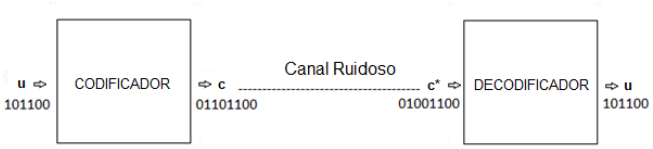

Fig. 1.*Representação clássica de um sistema*

Buscando-se reduzir o número de operações realizadas durante o processo de decodificação, são estudados algoritmos subótimos, com desempenhos similares ao MLD, método custoso, porém de máxima confiança [2]. O algoritmo de decodificação utilizado neste trabalho baseia-se em conjuntos de informação. De acordo com [1-3], é possível, durante o processo de decodificação de códigos de blocos, reduzir o número palavras-código candidatas. Para tal, a mensagem demodulada tem seus símbolos ordenados em uma escala de confiabilidade para, em seguida, realizar uma série de operações matriciais redutoras. Ao fim, reduz-se significativamente o número de operações de comparação necessárias para se determinar qual a palavra candidata corresponde a mensagem enviada originalmente.

Este trabalho explora a implementação do algoritmo subótimo citado em [2] utilizando VHDL (*VHSCI Hardware Description Language*). No entanto, os dois

blocos construtivos desenvolvidos assemelham-se a arquitetura de um decodificador apresentado em [3], o qual fez uso da Teoria dos Conjunto de Informação, previamente descrito em [4-5]. Nos decodificadores baseados em conjuntos de informação são utilizados os níveis de confiabilidade dos símbolos recebidos, gerando um conjunto reduzido de palavras-código candidatas para escolher a que mais se assemelha com a mensagem recebida, o que os difere do algoritmo MLD, onde todas as 2^k palavras-códigos são comparadas. Posteriormente, foi realizado a síntese do esquemático em uma ferramenta de circuitos integrados, obtendo dados relevantes, referentes ao consumo de potência e área de superfície ocupada pelo mesmo.

II. DESCRIÇÃO DO ALGORITMO

O algoritmo obtido em [2] constrói matrizes parciais a partir dos elementos considerados mais confiáveis a fim de gerar outra matriz para enfim, gerar o conjunto de $k + 1$ palavras-código candidatas. Para exemplificar as tarefas realizadas no algoritmo, considere que uma mensagem **u** hipotética é codificada, utilizando a matriz geradora **G** característica de um código C(7, 4), onde as *k* primeiras colunas tratam-se de uma matriz identidade, enquanto, as demais são geradas através de operações entre os elementos das anteriores.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad (1)$$

Através do produto de **u** e **G**, gera-se a palavra-código **c** = [1 1 0 1 0 0 0] (*n* = 7), que é então transmitida através de um canal ruidoso. Após a transmissão, **c*** já tem seus símbolos representados por valores decimais que representam uma quantização do sinal analógico recebido de três *bits*. No exemplo, **c* =** [7, 5, 4, 3, 2, 1, 0].

A primeira tarefa do decodificador trata-se da decodificação abrupta dos dados de entrada, ao mesmo tempo em que se atribui níveis de confiança a cada símbolo demodulado em uma escala de 0 a 3, onde o maior valor desta é considerado de maior confiança e o menor como de menor confiança. É determinado que os valores 0 e 7 são de confiança 3, os valores 1 e 6 de confiança 2 e, por fim, os valores 2 e 5 de confiança 1 e os dois restantes de confiança 0. Para a decisão abrupta, os símbolos com valores iguais ou superiores a quatro são tratados como nível lógico alto, '1', e aqueles com valores inferiores a quatro são tratados como nível lógico baixo, '0'. Assim, para este exemplo, **c** = [1 1 0 0 0 1 0]. Além disso, um vetor chamado $\mathbf{r_{parcial}}$ deverá também ser formado apenas pelos *k* símbolos mais confiáveis de **u**. Sendo assim $\mathbf{r_{parcial}}$ = [0110]. Posteriormente, uma matriz, denominada $\mathbf{G_{parcial}}$, deve ser gerada, sendo constituída de *k* colunas de **G**, cujos índices correspondem aos dos *k* símbolos mais confiáveis de **c*** em ordem decrescente de confiabilidade. Portanto, neste caso, $\mathbf{G_{parcial}}$ corresponde a matriz:

$$\mathbf{G_{parcial}} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Em seguida, deve-se verificar se a matriz $\mathbf{G_{parcial}}$ obtida possui inversa, se sim, deve-se obtê-la, caso contrário, é necessário buscar outro arranjo de *k* colunas de **G**. Ao obter a matriz $\mathbf{G_{parcial}}\boldsymbol{l}^{-1}$, inicia-se o processo de obtenção da $\mathbf{G_{nova}}$.

O processo de obtenção de $\mathbf{G_{nova}}$ consiste em realizar o produto matricial entre **G** e $\mathbf{G_{parcial}}\boldsymbol{l}^{-1}$. Sendo assim:

$$\mathbf{G_{nova}} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (3)$$

Uma vez calculado $\mathbf{G_{nova}}$, já pode-se obter as $k + 1$ palavras candidatas. A primeira, $\mathbf{c_1}$, é extraída pelo produto de $\mathbf{r_{parcial}}$ e $\mathbf{G_{nova}}$. Logo:

$$\mathbf{c_1} = [0110] \cdot \mathbf{G_{nova}} \quad (4)$$

As outras *k* palavras candidata podem ser logradas por, também, *k* variações de $\mathbf{r_{parcial}}$ obtidas por inversões de algumas das posições. Então, consequentemente tem-se:

$$\mathbf{c_2} = [1110] \cdot \mathbf{G_{nova}} \quad (5)$$

$$\mathbf{c_3} = [0010] \cdot \mathbf{G_{nova}} \quad (6)$$

$$\mathbf{c_4} = [0100] \cdot \mathbf{G_{nova}} \quad (7)$$

$$\mathbf{c_5} = [0111] \cdot \mathbf{G_{nova}} \quad (8)$$

Cada uma das palavras-código candidatas é então comparada com a palavra-código recebida, qual for mais semelhante será considerada como sendo a palavra correta, encerrando assim a decodificação.

III. IMPLEMENTAÇÃO DO ALGORITMO EM VHDL

Durante o estudo do algoritmo proposto em [2] percebeu-se que seria adequado apresentar uma arquitetura composta de quatro blocos. No entanto, apenas o Bloco I e II foram sintetizados e simulados neste trabalho, sendo que os Blocos III e IV foram apenas teorizados e corresponderiam as etapas posteriores a obtenção da matriz $\mathbf{G_{nova}}$.

A. Bloco I

Este bloco realiza as primeiras etapas do algoritmo apresentado na seção II. A primeira delas consiste em tratar os sinais de entrada, todos representados através de um vetor de números inteiros não sinalizados, de três bits. Em seguida, realiza-se uma série de operações, cria-se um vetor de um tipo personalizado de dado, cujo papel consiste em armazenar os valores de confiança de cada símbolo, respeitando a escala mencionada anteriormente, além de armazenar, também, a posição no vetor de entrada que cada símbolo ocupa. Determinou-se, também, um

vetor formado pela decisão abrupta e é, imediatamente, enviado ao Bloco II.

A tarefa seguinte trata-se da ordenação dos níveis de confiança designados a cada símbolo, para este fim, optou-se pelo uso de um circuito ordenador baseado no algoritmo de ordenação *Insertion Sort* (Figura 2*).*

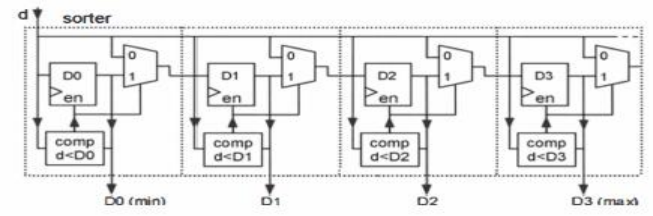

Fig. 2.*Método Insertion Sort em Hardware [2]*

São necessários *n* ciclos de *clock* para a conclusão, o vetor que foi ordenado manterá em ordem decrescente de confiança dos índices das posições dos bits de entrada. Ao fim, este vetor é enviado para a entrado do Bloco II

B. Bloco II

O Bloco II tem como primeira operação a geração da matriz $\mathbf{G}_{\text{parcial}}$, uma matriz constituída pela extração das *k* colunas mais confiáveis da matriz geradora **G**. Após esta etapa, verifica-se se, $\mathbf{G}_{\text{parcial}}$ possui inversa. A verificação ocorre ao escalonar a matriz e, então, ao analisar a diagonal principal aplicando a operação booleana *AND* e examinando se o resultado corresponde ao nível lógico alto, se sim, a matriz possui inversa. Caso a matriz não tenha inversa seleciona-se uma outra matriz com as *k* colunas mais confiáveis.

Ao operar sobre a matriz $\mathbf{G}_{\mathbf{parcial}}$ exibida acima, deve-se realizar outros *k* ciclos de *clock*, O processo para encontrar o determinante custa *k* ciclos de *clock* e foi subdivido em cinco etapas, sendo elas:

1. Verificar se o pivô, ou seja, o elemento $X_{i,j}$ da matriz mantém o *bit '1'* ou *'0'*. Os índices *i* e *j* são ambos numericamente iguais ao atual ciclo dedicado ao processo;
2. Adequar o pivô, isto é, na ocasião deste manter o *bit '0'*, deve-se, imediatamente, buscar alguma linha inferior, cujo primeiro elemento seja diferente de '*0',* e realizar uma troca com a linha do pivô. Feito isso, o pivô estará adequado;
3. Observar se os elementos de cada linha abaixo e acima na coluna do pivô apresentam o *bit '1'* ou '0';
4. Se constatado na etapa anterior um bit '1', a linha deste deve sofrer uma operação XOR com a linha do pivô. Esta consiste em neutralizar qualquer *bit '1'* presente fora da diagonal superior principal.

Uma vez obtido uma matriz $\mathbf{G}_{\text{parcial}}$ adequada obtém-se a inversa através de um algoritmo aprimorado do método de Eliminação de Gauss Jordan obtido em [6]. O processo consiste em operações matriciais que consomem *k* ciclos de *clock*. Inicialmente forma-se uma nova matriz a partir de $\mathbf{G}_{\text{parcial}}$, mas com ordem $k \times 2{\cdot}k$. As *k* primeiras colunas consistem na matriz original $\mathbf{G}_{\text{parcial}}$, enquanto às outras consistem nas colunas de uma matriz identidade de ordem *k*.

Após construir esta nova matriz, deve-se realizar 4 operações em *k* ciclos de *clock*. A primeira e segunda consistem em adequar o pivô, ou seja, garantir que a posição da primeira coluna e da primeira linha seja o *bit* '1', trocando linhas se necessário. Em seguida, todas as linhas abaixo da linha do pivô devem sofrer operações XOR com a própria linha do pivô. A última iteração consiste em realizar operação *shift left-up* em todas as colunas. Ao realizar esta série de operações em exatos *k* ciclos de *clock*, as *k* primeiras colunas da matriz aumentada consistem na matriz inversa de $\mathbf{G}_{\text{parcial}}$. processo foi subdividido em quatro etapas listadas abaixo.

1. Verificar se o pivô é adequado, ou seja, corresponde ao *bit '1'*. Nesta série de operações com as matrizes, o pivô sempre consistirá no primeiro elemento da primeira linha e coluna;
2. Adequar o pivô, isto é, na ocasião deste manter o *bit '0'*, deve-se, imediatamente, buscar alguma linha inferior, cujo primeiro elemento seja diferente de '*0',* e realizar uma troca com a linha do pivô. Feito isso, o pivô estará adequado;
3. Observar se há *bits* '1' nas linhas inferiores entre os elementos da coluna do pivô. Se constatado um bit '1', a linha deste deve sofrer uma operação XOR com a linha do pivô.
4. Realizar a operação denominada *shift left-up*, ou seja, cada elemento da matriz deve ser movido uma posição a esquerda e uma para cima.

Uma vez obtido a matriz inversa de $\mathbf{G}_{\text{parcial}}$ realiza-se o produto desta pela matriz geradora, onde cada elemento de posição *i,j* da nova matriz consistirá no somatório dos produtos dos elementos da linha *i* pelos elementos da linha *j* . Obtendo-se então a matriz $\mathbf{G}_{\text{nova.}}$

IV. RESULTADOS

Esta seção destina-se a apresentar os resultados obtidos após a síntese e simulação dos Blocos I e II. Utilizou-se a ferramenta *ISE Design Suite*. Os parâmetros de simulação foram um código C(7, 4) e uma mensagem $\mathbf{c^*}$ = [7, 5, 4, 3, 2, 1, 0].

A. Bloco I

Tratando-se da síntese deste bloco, obtiveram-se resultados através de duas ferramentas, a primeira delas consistiu no relatório disponibilizado pela ferramenta ISE para um FPGA Virtex 5, cujos dados estão compilados na Tabela 1.

TABELA 1. CONSUMO DE RECURSOS LÓGICOS DO BLOCO I

Consumo de Recursos Lógicos da FPGA Virtex 5	
Recursos	Quantidade consumida
LUT	95
Flip Flops	40

A outra ferramenta, também, utilizada foi o RTL Compiller, que permitiu a interpretação do código do Bloco, sintetizando um esquemático de um circuito dedicado. Os resultados obtidos por esta ferramenta foram preliminares e podem ser observados na Tabela 2.

TABELA 2. RELATÓRIO DE SÍNTESE DO BLOCO I

Consumo de Recursos Lógicos da FPGA Virtex 5	
Área Utilizada (μm^2)	353
Potência Total consumida (mW)	1,03

A simulação realizada pelo ISIM para o Bloco I, sendo visível nas figuras 3 e 4, foi modelada para um sinal de clock com período de 100 ns, ao passo que a entrada consistia em **c*** = [7, 5, 4, 3, 2, 1, 0]. Percebe-se a saída combinacional do vetor *output*, o qual, imediatamente, após o início da simulação tem seu valor atribuído. Entretanto, o vetor de confiança ordenado só está disponível ao fim do enésimo ciclo de clock.

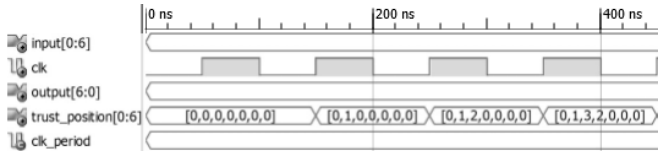

Fig. 3.*Simulação do Bloco I no período 0 ns - 450 ns*

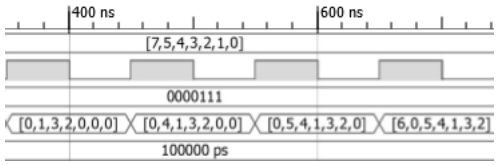

Fig. 4.*Simulação do Bloco I no período 350 ns - 750 ns*

B. Bloco II

Novamente, assim como para o bloco anterior, obtiveram-se resultados através de duas ferramentas, o ISE para uma FPGA Virtex 5 e o RTL Compiler. Os resultados estão dispostos na Tabela 3 e 4, respectivamente.

TABELA 3. CONSUMO DE RECURSOS LÓGICOS DO BLOCO II

Consumo de Recursos Lógicos da FPGA Virtex 5	
Recursos	Quantidade consumida
LUT	479
Flip Flops	183

TABELA 4. RELATÓRIO DE SÍNTESE DO BLOCO II

Consumo de Recursos Lógicos da FPGA Virtex 5	
Área Utilizada (μm^2)	4748
Potência Total consumida (mW)	2,26

A simulação realizada pelo ISIM para o Bloco II, visível nas figuras 5, foi modelada para um sinal de clock com período de 80 ns. A entrada consistia no vetor de confiança ordenado, proveniente do Bloco, *trust_position* = [6, 0, 5, 4, 1, 3, 2].

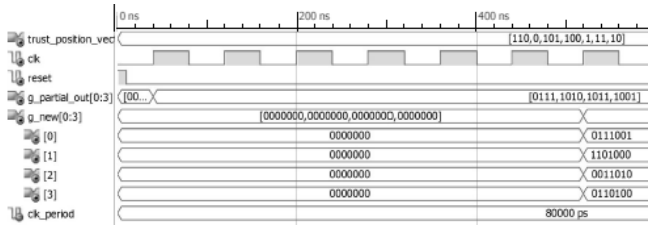

Fig. 5.Simulação do Bloco I no período 0 ns - 600 ns

Evidencia-se que a matriz $\mathbf{G_{parcial}}$ é gerada após a primeira borda de subida do clock. Entretanto, foram necessários outros $2 \cdot k - 1$ ciclos para obter a inversa de $\mathbf{G_{parcial}}$ e, então formar $\mathbf{G_{nova}}$. Destaca-se que uma análise mais profunda da simulação tornará evidente a progressão dos sinais que representam os processos de escalonamento e inversão descritos no algoritmo.

V. CONCLUSÃO

O primeiro e o segundo bloco foram sintetizados e simulados com êxito. Esses blocos representam a parte principal de uma arquitetura para decodificadores baseados em conjuntos de informação. Destaca-se que dispositivos com esta característica apresentam grande vantagem em termos de eficiência. Como salientado, são necessários $k + 1$ comparações entre as palavras candidatas, ao invés de $2^n - 1$ comparações, tal como no método MLD.

A implementação deste algoritmo em *hardware*, significou, primordialmente, descrever uma série de operações algébricas, sintetizá-las, simula-las e, por fim, transcreve-las para uma ferramenta de síntese de circuitos integrados. Dado o êxito e a simplicidade dos blocos construtivos desenvolvidos, o trabalho seguinte consistirá na construção de um terceiro e de um quarto bloco para, enfim, obter um decodificador completo.

REFERÊNCIAS

[1] PEDRONI, V.A. Digital Electronics and Design with VHDL. Boston: Elsevier Morgan Kaufmann Publishers, 2008.

[2] Brante, G. G. de O, A.; GODOY, W. Jr.; Muniz, D. N.; "Information Set Based Soft-Decoding Algorithm for Block Codes". IEEE IEEE Latin America Transactions, Latin-American Conference on Communications, Bogotá, 2010.

[3] 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 SystemsCircuits and Systems(APCCAS), Kuala Lumpur, 2010.

[4] 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.

[5] DORSCH, B. G. "A Decoding Algorithm for Binary Block Codes and J-ary Output". Channels," IEEE Trans. on Information Theory, vol. IT-20, pp. 391-394, May 1974

[6] JASINSKI V. P, PEDRONI, V. A, Gortan A., Godoy W. Jr. "An Improved GF(2) Matrix Inverter with Linear Time Complexity," International Conference on Reconfigurable Computing, 2010.