ARQUITETURA PARA UM DECODIFICADOR DE CÓDIGOS DE HERMITE PELA SOLUÇÃO DE UMA EQUAÇÃO CHAVE

Leocarlos B. S. Lima, Francisco M. Assis and Lirida A. B. Naviner

Resumo - Uma nova arquitetura de implementação em hardware é proposta para um algoritmo de decodificação de códigos algébrico-geométricos (AG) baseados em curvas de Hermite. O algoritmo de decodificação objeto deste trabalho busca iterativamente funções localizadoras e avaliadoras de erros que satisfaçam um critério de equação chave. São descritos operadores otimizados para implementar os cálculos mais freqüentes do decodificador. A descrição da arquitetura deste decodificador segue a descrição de arquiteturas para unidades aritméticas em corpos finitos de característica 2, necessárias à implementação em hardware de qualquer sistema de codificação / decodificação de canal usando códigos de bloco.

Palavras-chave: Códigos algébrico-geométricos, curvas de Hermite, decodificação, arquitetura.

Abstract - A new architecture is proposed for hardware implementation of a decoding algorithm of algebraic-geometric (AG) codes based on Hermitian curves. The decoding algorithm, object of this work, searches error locator and evaluator functions iteratively that satisfy a key equation criterion. Optimized operators to implement the most frequent calculations in the decoder are also proposed. The description of the architecture of this decoder follows the description of architectures for arithmetical units in finite fields of characteristic 2, necessary to implemente any channel coding / decoding system using block codes.

Keywords: Algebraic-geometric codes, Hermitian curves, decoding, architecture.

A classe de códigos de bloco conhecidos como *códigos algébrico-geométricos* (AG) foi proposta em 1982 por M. A. Tsfasman, S. G. Vlăduţ e T. Zink [43]. Eles associaram o trabalho do matemático russo V. D. Goppa, que propôs a construção de códigos pela avaliação de funções algébricas em pontos de curvas definidas sobre corpos finitos [14], com recentes resultados da geometria algébrica [4, 5, 10, 23, 26]. Os conhecidos códigos de Reed-Solomon (RS) são um caso particular dos códigos AG quando a curva algébrica adotada é uma reta. Curvas sobre corpos finitos podem ter muito mais pontos que uma simples reta, de forma que os códigos AG podem apresentar comprimento muito maior que os códigos RS.

Os códigos AG apresentam excelente comportamento assintótico no que se refere à relação entre a taxa de informação[1], que representa o comprometimento da taxa de transmissão do sistema devido à utilização do código (introdução de redundâncias), e a distância mínima relativa[2], que representa a capacidade de correção de erros do código. Ou seja, os códigos AG não comprometem a taxa de transmissão em detrimento da capacidade de correção, ou vice versa, para um comprimento de código $n \to \infty$. Os códigos AG apresentam parâmetros como comprimento, capacidade de correção e taxa de informação melhores que códigos já estabelecidos, como os de Reed-Solomon [9, 23].

Em 1989, Jørn Justesen *et al.* propuseram o primeiro algoritmo de decodificação para códigos AG, que consistia numa generalização do algoritmo PGZ para decodificação de códigos BCH [17, 24]. Desde então, diversos trabalhos, sob diferentes abordagens, vêm sendo apresentados em torno do tema da decodificação destes códigos [3, 6, 11, 12, 15, 16, 18, 22, 27, 30, 32–36, 38–40, 44]. Entretanto, poucos trabalhos são encontrados abordando o problema da implementação em hardware de decodificadores para estes códigos [1, 2, 19, 25, 29, 31, 37]. Nesta linha, merece destaque o algoritmo de estrutura regular e simples proposto em 1998 por Ralf Kötter [21]. A idéia central na proposta de Kötter foi utilizar uma configuração em paralelo de vários algoritmos BMS modificados, de modo a obter um esquema que apresentasse a mesma complexidade de tempo de um decodificador de Berlekamp-Massey para códigos RS.

Em [27], M. E. O'Sullivan apresentou um algoritmo de decodificação de códigos AG de um único ponto baseado na solução de uma equação chave. Este algoritmo busca iterativamente polinômios localizadores de erros, ou seja, polinômios que apresentam zeros nos pontos associados às posições em que ocorreram erros na transmissão. Uma arquitetura para este algoritmo foi relatada em [29]. Em [28], O'Sullivan estendeu esta abordagem para fornecer simultaneamente polinômios localizadores e avaliadores de erros. No presente artigo, propomos uma nova arquitetura para a implementação desta segunda abordagem do algoritmo de O'Sullivan para o caso de curvas de Hermite [7, 8]. Este trabalho incluiu ainda a descrição de unidades aritméticas em corpo finito responsáveis pelas operações aritméticas no de-

Este trabalho foi financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES (http://www.capes.gov.br), tendo sido inserido em acordo CAPES / COFECUB (Brasil / França).

Leocarlos B. S. Lima está ligado ao Centro Universitário de João Pessoa – UNIPÊ, BR 230, km 22, Água Fria, 58053-000, João Pessoa-PB, Brasil (contato@leocarloslima.com).

Francisco M. Assis está ligado à Universidade Federal de Campina Grande – UFCG / DEE, Caixa Postal 10053, 58109-970, Campina Grande, PB, Brasil (fmarcos@dee.ufcg.edu.br).

Lirida A. B. Naviner está ligada à École Nationale Supérieure des Télécommunications – ENST / COMELEC, 46, rue Barrault, 75634, Cedex 13, Paris, França (lirida.naviner@enst.fr).

[1] A taxa de informação de um código consiste na razão $R = \frac{k}{n}$ entre a dimensão k e o comprimento n do código.

[2] A distância mínima relativa de um código consiste na razão $\delta = \frac{d}{n}$ entre a distância mínima d e o comprimento n do código.

codificador.

A Seção 1 descreve os códigos AG baseados em curvas de Hermite, conhecidos como códigos de Hermite. A Seção 2 descreve sucintamente o algoritmo de decodificação de O'Sullivan. A arquitetura proposta para o decodificador é descrita na Seção 3. As conclusões são apresentadas na Seção 4.

1. CÓDIGOS DE HERMITE

Os códigos AG são construídos a partir de curvas algébricas definidas sobre corpos finitos [13, 17, 42]. Considere um corpo finito $\mathbb{F}_q$ de q elementos. Considere uma curva algébrica $\mathcal{X}$, que possui n pontos racionais $\{P_{n-1}, \ldots, P_0\}$ e um ponto Q no infinito. Considere os divisores $G = mQ$ e $D = P_0 + \cdots + P_{n-1}$ de $\mathcal{X}$, cujos suportes sejam disjuntos. Considere um espaço $L(G)$ de funções racionais f, tais que $(f) \succeq -mQ$. Um código AG de um único ponto denotado por $C(D, G)$ é o espaço vetorial n-dimensional definido por

$$\{(f(P_{n-1}), \ldots, f(P_1), f(P_0)) : f \in L(mQ)\}. \quad (1)$$

Um código de Hermite consiste num código AG baseado numa curva de Hermite $y^r + y = x^{r+1}$ definida sobre um corpo finito $\mathbb{F}_{r^2}$, em que r é um número inteiro [41]. Este tipo de curva possui gênero $g = \frac{r^2-r}{2}$ e apresenta r^3 pontos afins e um único ponto Q no infinito. Um código de Hermite de comprimento $n = r^3$ é gerado por (1). Uma base para o espaço de funções $L(mQ)$ é dado por

$$\left\{x^i y^j \mid ir + j(r+1) \leq m,\ 0 \leq i,\ 0 \leq j \leq r-1\right\}. \quad (2)$$

No caso dos códigos de Hermite, considera-se um conjunto $\Lambda = \{ir + j(r+1) : 0 \leq i \leq r, j \in \mathbb{N}\}$ de anti-lacunas em Q. Considera-se ainda um anel $R = \mathbb{F}_{r^2}[x, y]/(y^r + y = x^{r+1})$ de funções com pólos exclusivamente nos pontos da curva.

2. ALGORITMO DE DECODIFICAÇÃO

Um processo completo de decodificação utilizando o algoritmo de O'Sullivan inclui as seguintes tarefas (*cf.* figura 1):

a) Cálculo das síndromes (vetor $\vec{S}$) a partir do vetor recebido e da matriz de paridade do código;

b) Determinação dos polinômios avaliadores ϕ e localizadores f de erros (esta etapa constitui usualmente o algoritmo de decodificação propriamente dito);

c) Determinação do vetor de erros $\vec{e}$ a partir dos polinômios avaliadores e localizadores de erros;

d) Correção do vetor recebido, que consiste basicamente em subtrair deste vetor o vetor erro obtido das etapas anteriores;

e) Mapeamento do vetor corrigido $\vec{c}$ no vetor de informação $\vec{i}$ correspondente.

O algoritmo proposto por O'Sullivan em [28] concerne unicamente à tarefa (b) na figura 1. Em geral, os algoritmos de decodificação propostos são restritos à tarefa de determinação das posições de ocorrência dos erros, considerada o núcleo de um processo de decodificação de códigos de bloco. As demais tarefas são consideradas problemas de soluções conhecidas. Da mesma forma, a arquitetura para este decodificador apresentada aqui corresponde estritamente ao algoritmo de O'Sullivan e, portanto, à tarefa (b) da figura 1.

Considere um corpo finito $\mathbb{F}_q$, no qual $q = r^2$ e r é um inteiro positivo. Na prática, para uma implementação em hardware, considera-se sempre r como sendo uma potência de 2, de modo que $\mathbb{F}_q$ seja um corpo de característica 2. A necessidade por corpos finitos de característica 2 deve-se à natureza binária dos circuitos digitais.

Considere um código de Hermite de comprimento n definido pela curva de Hermite dada pela equação $y^r + y = x^{r+1}$ de gênero g. Considere $P_1, \ldots, P_n$ os r^3 pontos racionais da curva e Q o único ponto no infinito no plano projetivo. Considere Λ o conjunto das anti-lacunas em Q e $\theta_k = x^{\mu(k)} y^{\nu(k)}$ um monômio associado a uma ordem de pólo $k = \mu(k)r + \nu(k)(r+1)$ em Q.

Se $R = \mathbb{F}_q[x, y] / \left(y^r + y = x^{r+1}\right)$ é um anel de polinômios com pólos unicamente nos pontos da curva, uma síndrome $S(f)$ de uma função $f = f_{pl-1}\theta_{pl-1} + \cdots + f_1\theta_1 + f_0 \in R$ qualquer é calculada da forma

$$\begin{aligned} S(f) &= \sum_{k=0}^{n-1} v_k f(P_k) \\ &= \sum_{k=0}^{n-1} v_k \sum_{i=0}^{pl-1} f_i \theta_i(P_k) \\ &= \sum_{i=0}^{pl-1} f_i \sum_{k=0}^{n-1} v_k \theta_i(P_k) \\ &= \sum_{i=0}^{pl-1} f_i S_i \end{aligned} \quad (3)$$

em que $\vec{v} = [v_{n-1} \ \cdots \ v_1 \ v_0]$ é o vetor recebido e $S_i = \sum_{k=1}^{n} v_k \theta_i(P_k)$ é uma síndrome do vetor de síndromes $\vec{S} = [S_{pl-1} \ \cdots \ S_1 \ S_0]$ calculado a partir do vetor recebido.

O cálculo iterativo dos polinômios localizadores e avaliadores de erros pelo algoritmo de O'Sullivan é baseado na solução de uma equação chave (*cf.* (4)) derivada de uma série de síndromes h_e definida por

$$h_e = \frac{x^r}{x} \sum_{i=0}^{r} \sum_{j=0}^{\infty} s_{ij} x^{-i} y^{-j},$$

em que $0 \leq i \leq r$, $j \geq 0$ e $s_{ij} = S(x^i y^j)$ é uma síndrome calculada por (3) considerando $f = x^i y^j$.

O núcleo do algoritmo de O'Sullivan é o Corolário 3.11 de [28] que determina que f é uma função localizadora de erros (função que possui zeros nos pontos correspondentes às posições do vetor recebido em que os erros aconteceram) se e só se $fh_e \in R$. Com base neste resultado, o algoritmo procura iterativamente pares de funções $f, \phi \in R$ (funções

localizadoras e avaliadoras de erros, respectivamente) que satisfazem a equação chave

$$fh_e = \phi. \tag{4}$$

A cada iteração, o algoritmo incrementa a dimensão do espaço de busca por funções e verifica se os pares de funções produzidos na última iteração $\dot{f}$ e $\dot{\phi}$ (um ponto sobre uma função indica que ela foi produzida na última iteração) satisfazem ainda a equação chave (4). Se um par $\dot{f}$ e $\dot{\phi}$ não mais satisfaz (4), um novo par de funções f e ϕ é calculado a partir de $\dot{f}$ e $\dot{\phi}$ e de funções auxiliares g e ψ. Estas funções auxiliares são na verdade funções f e ϕ, respectivamente, que não satisfizeram a equação chave numa iteração precedente. Elas são úteis porque representam o espaço ortogonal ao de busca das funções desejadas. A atualização das funções, de modo que elas passem a satisfazer (4), constitui o terceiro passo do algoritmo de O'Sullivan (algoritmo 1 descrito nos próximos parágrafos).

Da proposição 4.1 em [28], o vetor de erros $\vec{e}$ pode ser determinado pela equação

$$e_i = \frac{\phi}{f'}(P_i), \tag{5}$$

em que $f^{'}$ denota a derivada primeira de f em relação a x, ou seja, df/dx. Esta derivada é calculada seguindo as regras usuais de derivação de funções racionais em corpos finitos de característica 2 e considerando que $dx^k/dx = 0$, se k for par, e $dx^k/dx = x^{k-1}$, se k for ímpar. Além disso, da curva de Hermite tem-se que

$$d(y^r + y)/dx = d(x^{r+1})/dx \Rightarrow$$
$$dy/dx = x^r,$$

já que r como potência de 2 é sempre um número par.

Após um número suficiente de iterações (it_{max}), garante-se que entre as funções localizadoras fornecidas pelo algoritmo existe ao menos uma função f tal que $f^{'}$ não se anula num ponto associado a uma posição de um erro. Desta forma, a determinação do vetor erro $\vec{e}$ a partir das funções f e ϕ fornecidas pelo algoritmo (tarefa (c) na figura 1) é feita da seguinte forma:

1. Avaliação das funções f para determinar quais as possíveis posições de ocorrências dos erros (posições associadas aos zeros das funções);
2. Determinação das derivadas primeiras $f^{'}$ (materialmente, esta tarefa não apresenta qualquer complexidade para corpos finitos de característica 2);
3. Avaliação das funções $\frac{\phi}{f'}$, para cada par f e ϕ, nos pontos em que f se anula, mas $f^{'}$ não se anula;
4. Os coeficientes e_i do vetor erro $\vec{e}$ consistem na união dos resultados obtidos para cada par f e ϕ.

O algoritmo 1 descrito a seguir apresenta o algoritmo de decodificação de O'Sullivan. Numa iteração it, denota-se por α, $\beta \in \mathbb{F}_q$ valores de síndromes de funções (*cf.* (3)), por σ, $\delta \subset \Lambda$ conjuntos de parâmetros de controle (veja parágrafo seguinte), e por f, g, ϕ, $\psi \in R$ as funções localizadoras e avaliadoras de erros e suas respectivas funções auxiliares.

Considere aqui $f \in R$ uma função que não satisfaz a equação chave e, portanto, não localiza os erros ocorridos. Defina

$$\mathrm{span}(f) = \min\{-\nu_Q(g) : S(fg) \neq 0\}$$

como a menor ordem de pólo de uma função g qualquer, tal que $S(fg) \neq 0$. Defina ainda

$$\mathrm{fail}(f) = -\nu_Q(f) + \mathrm{span}(f)$$

como a menor ordem de pólo de uma função qualquer, maior ou igual que a ordem de pólo de f, cuja síndrome seja não nula.

Numa dada iteração it, define-se um conjunto Σ_{it} na forma

$$\Sigma_{it} = \{-\nu_Q(f) \in \Lambda : \mathrm{fail}(f) > it\}$$

como sendo uma partição do conjunto Λ cujas funções que possuem estas ordens de pólo em Q apresentam $\mathrm{fail}(f) > it$. Em outras palavras, Σ_{it} constitui um conjunto de ordens de pólo de funções f, tal que as menores ordens de pólo das funções múltiplas de f não localizadoras de erros são maiores que it.

Define-se ainda um conjunto Δ_{it} na forma

$$\Delta_{it} = \{\mathrm{span}(g) \in \Lambda : \mathrm{fail}(g) \leq it\}$$

como sendo outra partição de Λ cujas funções g que apresentam estes valores de $\mathrm{span}(g)$ apresentam $\mathrm{fail}(g) \leq it$. Em outras palavras, Δ_{it} constitui o conjunto dos $\mathrm{span}(g)$, tal que as menores ordens de pólo das funções múltiplas de g não localizadoras de erros são menores ou iguais a it.

Os conjuntos Σ_{it} e Δ_{it} são complementares sobre o conjunto de anti-lacunas Λ, ou seja,

$$\Delta_{it} = \Lambda \cap \Sigma_{it}.$$

Os conjuntos σ e δ consistem em mínimos e máximos de Σ e Δ, respectivamente, segundo uma ordenação parcial denotada por $\preceq$, em que $a \preceq b$ se $b - a \in \Lambda$ (*cf.* [28]). Em uma iteração it, a cada função localizadora de erros f está associado um elemento de σ, que representa a ordem de pólo desta função. Já os elementos de δ correspondem às ordens de pólo de cada uma das funções auxiliares g numa dada iteração it.

A cada iteração, síndromes $\alpha = S(\dot{f})$ são calculadas. Com base nos valores destas síndromes, os parâmetros de controle σ e δ são atualizados. Em seguida, utilizando os novos parâmetros de controle, as funções f, g, ϕ e ψ são atualizadas.

Denota-se por $\widetilde{f}$ a parte R de f, ou seja, a função f excluindo-se os termos θ_i nos quais i é uma lacuna. No algoritmo 1, as funções $\max\{\star\}$ e $\min\{\star\}$ correspondem aos máximos e mínimos com relação à ordenação parcial $\preceq$.

Algoritmo 1 (Decodificador de O'Sullivan)

Entradas:

- *O vetor de síndromes $\vec{S} = [S(\theta_{n-k}) \cdots S(\theta_0)]$ do vetor recebido $\vec{v}$, em que θ_i são as funções de base do espaço ortogonal ao código. Este vetor é obtido pelo produto de $\vec{v}$ pela matriz de paridade do código. $S(\theta_i) = 0$ se i é uma lacuna.*

Saídas:

- *As funções f e ϕ localizadoras e avaliadoras de erros, respectivamente.*

Inicialização:

- $it = -1$;
- $\sigma_{-1} = \{0\}$;
- $f(0) = 1$;
- $\phi(0) = 0$.

<<< **Passo 1: Cálculo das síndromes** $\alpha = S(f)$ >>>

- *Incrementar* $it = it + 1$. *Se* $it > it_{max}$, *encerrar o algoritmo;*
- *Salvar os valores obtidos na última iteração* $\dot{f} = f$, $\dot{\phi} = \phi$, $\dot{g} = g$, $\dot{\psi} = \psi$, $\dot{\beta} = \beta$, $\dot{\delta} = \delta$, $\dot{\sigma} = \sigma$;
- *Para cada* $s \in \dot{\sigma}$, *calcular* $\alpha(s) = S(\widetilde{\dot{f}\theta_{it-s}})$. *Se* $S(\theta_{it})$ *(necessário a cada iteração) não está disponível, calcular* $S(\dot{f}\theta_{it-s} - \theta_{it})$ *e empregar o algoritmo de decisão por maioria para determinar* $S(\theta_{it})$. *Em seguida, calcular* $\alpha(s) = S(\widetilde{\dot{f}\theta_{it-s} - \theta_{it}}) + S(\theta_{it})$.
- *O algoritmo de decisão por maioria consiste em se calcular o conjunto* $\Gamma = \dot{\Sigma} \cap \quad it - \dot{\Sigma}$ *e, para cada* $a \in \Gamma$, *encontrar um* $s \in \dot{\sigma}$ *tal que* $s \preceq a$. *Em seguida, escolher* $\zeta \in \mathbb{F}_{r^2}$ *tal que* $g = \theta_{it} + \zeta \dot{f}(s)\theta_{it-s}$ *apresenta grau menor que* it. *O valor de* $S(\theta_{it})$ *consiste na maioria dos valores* $S(g)$ *calculados por cada* s *escolhido.*

<<< **Passo 2 : Determinação dos parâmetros de controle** >>>

- *Calcular* $\delta' = \{it - s : s \in \dot{\sigma},\ it - s \in \Lambda$ *e* $\alpha(s) \neq 0\}$;
- *Calcular* $\delta = \max\{\delta' \cup \dot{\delta}\}$;
- *Calcular* $\sigma = \min\{t : \nexists c \in \delta$ *com* $t \prec c\}$.

<<< **Passo 3 : Atualização das funções** >>>

- *Para cada* $c \in \delta$, *fazer*
  - $g(c) = \begin{cases} \dot{g}(c), & \text{se } c \in \dot{\delta}, \\ \dot{f}(it - c), & \text{caso contrário}; \end{cases}$
  - $\psi(c) = \begin{cases} \dot{\psi}(c), & \text{se } c \in \dot{\delta}, \\ \dot{\phi}(it - c), & \text{caso contrário}; \end{cases}$
  - $\beta(c) = \begin{cases} \dot{\beta}(c), & \text{se } c \in \dot{\delta}, \\ \alpha(it - c), & \text{caso contrário}; \end{cases}$
- *Para cada* $t \in \sigma$, *encontrar* $s \in \dot{\sigma}$ *e* $a \in \Lambda$ *tais que* $s + a = t$. *Denota-se por* $\overline{\phi}$ *a função constituída pelos termos de* ϕ *com valorização discreta máxima* $it - t - 2g + 1$.
  - *Se* $it - t$ *é uma lacuna, calcular*
    - $f(t) = \dot{f}(s)\theta_a$;
    - $\phi(t) = \overline{\dot{\phi}(s)\theta_a} + \alpha(s)\theta_{2g-1-(it-t)}$;
  - *Se* $it - t$ *é uma anti-lacuna e* $\alpha(s) \neq 0$, *encontrar* $c \in \dot{\delta}$ *e* $b \in \Lambda$ *tais que* $c - s = it - t$. *Se* $it - t$ *é uma anti-lacuna e* $\alpha(s) = 0$, *então* $t = s$ *e pode-se escolher* b *e* c *quaisquer. Considerando* $\gamma = \alpha(s)/\dot{\beta}(c)$, *calcular*
    - $f(t) = \dot{f}(s)\theta_a - \gamma\dot{g}(c)\theta_b$;
    - $\phi(t) = \overline{\dot{\phi}(s)\theta_a} - \gamma\overline{\dot{\psi}(c)\theta_b}$;

2.1 EXEMPLO

Considere como exemplo um código de Hermite $(64, 54)$ corretor de 2 erros, definido por uma curva de Hermite $y^4 + y = x^5$ de gênero $g = 6$ sobre $\mathbb{F}_{16}$, portanto de parâmetros $r = 4$ e $m = 59$. Esta curva apresenta 64 pontos racionais mais um ponto Q no infinito do plano projetivo, razão pela qual o código apresenta um comprimento $n = 64$. Este código $(64, 54)$ é definido por um espaço de funções $L(59Q)$. O espaço ortogonal a este código, utilizado na decodificação, é o espaço $L((r^3+r^2-r-2-m)Q) = L(15Q)$. Este código apresenta um conjunto de lacunas $\{1, 2, 3, 6, 7, 11\}$ de 6 elementos. Ele apresenta uma matriz de paridade de 10 linhas e, portanto, possui 10 síndromes a calcular a partir do vetor recebido.

Considere um vetor recebido $\vec{v}$ ao qual foram adicionados erros de valor:

- α^5 na posição associada ao ponto (α^8, α^3);
- e α^{13} na posição associada ao ponto (α^7, α^{13}).

Desta forma, é obtido o vetor de síndromes

$$\vec{S} = [\alpha\ 1\ \alpha^{13}\ \alpha^9\ 0\ \alpha^2\ \alpha^9\ \alpha^4\ 0\ 0\ \alpha^7\ \alpha^7\ 0\ 0\ 0\ \alpha^7]$$

que apresenta 10 síndromes calculadas a partir de $\vec{v}$ nas posições das anti-lacunas e zeros nas posições das 6 lacunas.

A tabela 1 descreve os valores resultados da decodificação a cada iteração do algoritmo 1.

Ao fim da iteração $it = 21$, o algoritmo produz dois pares de funções f e ϕ:

- $f_1 = x^2 + x + \alpha^{11}$;
- $\phi_1 = \alpha^7 xy^3 + \alpha^4 y^3 + \alpha^7 xy^2 + \alpha y^2 \alpha^2 xy + \alpha^6 y + \alpha^{14} x + \alpha^{11}$;
- $f_2 = y + \alpha^{12} + \alpha^{11}$;
- $\phi_2 = \alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2 + \alpha^{14} y + \alpha^9 x + \alpha^7$.

A função f_1 apresenta zeros nas posições associadas aos pontos (α^8, α^3), (α^8, α^{14}), (α^7, α^9), (α^7, α^7), (α^7, α^6), (α^7, α^{13}), (α^8, α^{11}) e (α^8, α^{12}). Aplicando (5), obtém-se um valor de erro α^5 para o ponto (α^8, α^3), α^{13} para o ponto (α^7, α^{13}) e 0 para os demais pontos.

Já a função f_2 apresenta zeros nas posições associadas aos pontos (α^9, α), (α^{12}, α^2), (α^8, α^3), (α^4, α^6) e (α^7, α^{13}). Aplicando (5), obtém-se novamente um valor de erro α^5 para o ponto (α^8, α^3), α^{13} para o ponto (α^7, α^{13}) e 0 para os demais pontos.

3. ARQUITETURA

Observa-se que o decodificador de O'Sullivan apresenta uma complexidade mais importante concentrada na parte de controle do algoritmo (passo 2 e determinação das anti-lacunas a e b no passo 3 do algoritmo 1). Esta característica difere do que se observa em outros algoritmos de decodificação para códigos AG. Em geral, estes algoritmos apresentam uma complexidade concentrada sobretudo no cálculo das síndromes e na atualização das funções. Foi esta característica deste algoritmo que nos motivou e conduziu em direção à sua exploração.

Em [28], O'Sullivan não apresenta uma discussão sobre a complexidade de seu algoritmo. Ele não fornece sequer certas informações necessárias à determinação desta complexidade, como o comprimento dos registradores das funções. Nós deduzimos todos os parâmetros que faltavam para a determinação da complexidade e para a implementação deste decodificador.

3.1 UNIDADES ARITMÉTICAS

No decodificador de O'Sullivan, todas as operações aritméticas sobre valores de síndromes e coeficientes de funções em R, símbolos do corpo finito $\mathbb{F}_q$, são realizadas utilizando unidades aritméticas desenvolvidas para este corpo. As figuras 2, 3 e 4 apresentam as arquiteturas dos operadores de adição, multiplicação e inversão que foram implementadas para o corpo finito $\mathbb{F}_{16}$.

Na arquitetura a ser apresentada para o decodificador do algoritmo 1, um símbolo de $\mathbb{F}_{r^2}$ é representado por $\log_2 r^2$ bits. Uma função de R consiste num vetor de pl símbolos, em que pl é o número máximo de iterações do algoritmo necessárias à obtenção das funções desejadas mais 1 (observa-se que numa iteração it a ordem de pólo de qualquer função no algoritmo jamais é maior que it), ou seja,

$$pl = it_{max} + 1.$$

Todos os números inteiros no circuito são representados por il bits.

3.2 CÁLCULOS MAIS FREQÜENTES

Dentre os cálculos efetuados no algoritmo, pode-se destacar alguns mais utilizados e com maior impacto sobre o desempenho da implementação e para os quais são propostas arquiteturas particulares:

3.2.1 OPERADOR PARA O PRODUTO $F.\theta_C$

Considere f uma função e θ_c um monômio em R. A operação $f.\theta_c$ é sistematicamente efetuada nos cálculos de síndrome e nas atualizações das funções (etapas 1 e 3 do algoritmo). Este produto de funções em R é feito módulo a equação da curva de Hermite e, portanto, implica a substituição $x^{r+1} = y^r + y$. Portanto, este produto de funções não se traduz numa mera convolução de vetores, no caso um simples deslocamento de um vetor, mas exige elementos adicionais que implementem a substituição descrita.

Esta substituição se opera da seguinte forma. Considere $\theta_c = x^{\mu(c)}y^{\nu(c)}$. Multiplicando-se um monômio qualquer $f_k x^{\mu(k)}y^{\nu(k)}$ por θ_c, obtém-se $f_k x^{\mu(k)+\mu(c)}y^{\nu(k)+\nu(c)}$ de ordem de pólo $k + c$ em Q. No entanto, se $\mu(k) + \mu(c) > r$, tem-se que

$$x^{\mu(k)+\mu(c)}y^{\nu(k)+\nu(c)}\frac{(y^r + y)}{x^{r+1}} = x^{\mu(k)+\mu(c)-r-1}(y^{\nu(k)+\nu(c)+1} + y^{\nu(k)+\nu(c)+r}). \quad (6)$$

Observe que $x^{\mu(k)+\mu(c)-r-1}y^{\nu(k)+\nu(c)+r}$ apresenta uma ordem de pólo em Q igual a

$$(\mu(k) + \mu(c) - r - 1)r + (\nu(k) + \nu(c) + r)(r + 1) = k + c$$

e $x^{\mu(k)+\mu(c)-r-1}y^{\nu(k)+\nu(c)+1}$ apresenta uma ordem de pólo em Q igual a

$$(\mu(k) + \mu(c) - r - 1)r + (\nu(k) + \nu(c) + 1)(r + 1) = k + c - r^2 + 1. \quad (7)$$

Portanto, o produto de uma função f por um monômio θ_c em R corresponde na verdade ao deslocamento de f em c posições, sendo que às posições de ordem k da função em que $\mu(k) + \mu(c) > r$, adiciona-se o coeficiente da posição de ordem $k + c - r^2 + 1$.

O'Sullivan em [28] observou este fato e propôs a implementação deste produto pelo uso de dois registradores, em que o primeiro armazena os coeficientes de f e o segundo armazena os coeficientes de f em que $\mu(k) + \mu(c) > r$. O primeiro registrador é deslocado em c posições (a direção do deslocamento depende do sinal de c) e o segundo registrador é deslocado em $c - r^2 + 1$ posições. Em seguida, os dois registradores são somados para gerar a função resultado do produto $f.\theta_c$.

Uma arquitetura menos complexa e mais eficiente para este operador é apresentada na figura 5. Ela consiste num único registrador de deslocamento modificado para incluir elementos de decisão e operadores de adição entre células de coeficientes da função. Este módulo recebe uma função f e valores associados a um inteiro c (sinais de controle determinados a partir de c no módulo controlador) e fornece uma função, resultado do produto $f.\theta_c$.

Toda a operação deste módulo é determinada pelo sinal de controle D, gerado pelo módulo controlador do decodificador. Um sinal D é constituído por 3 bits, que determinam os 5 diferentes estados de operação em um módulo $f.\theta_c$, a saber ($\star$ indica indiferença quanto ao bit ser '0' ou '1'):

Estado $\star$00 Indica que o módulo deve carregar-se com a função de entrada f';

Estado $\star$01 Indica que o módulo deve efetuar a adição dos componentes $f_k + f_{k-r^2+1}$, o que corresponde à operação módulo equação da curva de Hermite;

Estado 010 Indica a operação de deslocamento do registrador à esquerda;

Estado 110 Indica a operação de deslocamento do registrador à direita;

Estado $\star$11 Indica o estado de suspensão, no qual os valores do registrador são preservados.

3.2.2 OPERADOR PARA A FUNÇÃO $\mu(C) = I$

Considere uma função $\mu(c) = i$, tal que $ir + j(r+1) = c$. Esta função é utilizada como sub-módulo do operador $f.\theta_c$ para determinar se o coeficiente f_k do k-ésimo monômio do registrador modificado de $f.\theta_c$ deve ser adicionado ao coeficiente f_{k-r^2+1} (*cf.* figura 5). Isto equivale à realização do módulo equação da curva de Hermite.

Observa-se que esta função $\mu(c)$ pode ser realizada por um operador de módulo $(r+1)$. A operação módulo $(r+1)$ pode ser implementada por uma cadeia de até

$$il^2 - il - \lceil \log(r+1) \rceil^2 + \lceil \log(r+1) \rceil$$

somadores completos (na análise de cada caso particular, esta complexidade pode ser bastante reduzida).

Um exemplo de uma arquitetura deste tipo para uma operação módulo 5 sobre um inteiro de 7 bits, utilizada na implementação do decodificador para o código $(64, 54)$, é apresentada na figura 6.

3.2.3 OPERADOR PARA O TESTE DE PERTENÇA A Λ

Quando de quase todas as decisões no algoritmo de decodificação, evidencia-se a necessidade de realizar um teste do tipo $c \in \Lambda$, para um dado inteiro c.

Este tipo de teste pode ser realizado por uma busca em tabela verificando-se sua negativa, ou seja, se c pertence ao conjunto das anti-lacunas, que em geral é menor, ou se é negativo. Além disso, este teste pode ser realizado pelo uso da função $\mu(c)$. Observe que $c \in \Lambda$ se e só se $\mu(c)r \leq c$.

Sobre o espaço de funções relativamente pequeno do código $(64, 54)$ (existem apenas seis lacunas: 1, 2, 3, 6, 7 e 11), optou-se por empregar o método de busca em tabela.

3.3 MÓDULOS OPERACIONAIS

A figura 7 ilustra a arquitetura geral para o decodificador que emprega o algoritmo de O'Sullivan. Ela inclui r blocos MP (processadores principais), r blocos AP (processadores auxiliares), um buffer para as síndromes de entrada e um módulo controlador que fornece os sinais de controle para as seqüências de operação. Os blocos MP e AP são interconectados por intermédio de um barramento bidirecional.

Dependendo do código sobre o qual se trabalha (comprimento e capacidade de correção), o número de blocos MP e AP necessários pode ser menor que r. Não há uma regra geral para determinar o número de blocos MP e AP necessários, caso possa ser menor que r, mas se o código é capaz de corrigir $\tau < r$ erros, então τ blocos MP e $\tau - 1$ blocos AP são suficientes. Isto se deve ao fato de que o máximo elemento do conjunto δ não pode ser maior que a τ-ésima anti-lacuna. Por exemplo, no caso do código AG $(64, 54)$ de nosso exemplo, corretor de 2 erros, apenas são necessários 2 módulos MP e 1 módulo AP.

O decodificador recebe um vetor de $n - k$ síndromes $\vec{S}$ e fornece um conjunto de r funções localizadoras f e r funções avaliadoras de erros ϕ.

Uma arquitetura para a unidade MP é apresentada na figura 8. Cada módulo MP opera sobre uma função localizadora de erros f e uma função avaliadora de erros ϕ. Ele implementa o cálculo de uma síndrome α a partir de uma função f e todas as operações de atualização das funções f e ϕ.

Este módulo emprega dois operadores $f.\theta_c$:

- um para o cálculo de α e para a atualização de f,
- e outro para a atualização de ϕ.

Ele emprega ainda:

- um registrador para memorizar α;
- dois registradores para memorizar f e ϕ;
- e dois registradores para memorizar as funções $g\theta_b$ e $\psi\theta_b$ recebidas dos módulos AP (os produtos $g\theta_b$ e $\psi\theta_b$ são realizados pelos módulos AP).

Nos módulos MP, um módulo decodificador chamado DEC1 gera um vetor de bits na forma $0 \ldots 010 \ldots 0$ a partir de um inteiro tr, no qual o único bit '1' ocupa a tr-ésima posição do vetor de bits. Este vetor é utilizado para implementar a operação de adição de uma função por um monômio θ_c na atualização de funções ϕ. Um módulo decodificador chamado DEC2 opera de modo análogo, fornecendo um vetor de bits na forma $1 \ldots 10 \ldots 0$, com bits '1' até a tr-ésima posição do vetor. Este vetor é utilizado para truncar uma função.

Os detalhes da arquitetura para unidades AP são apresentados na figura 9. Este módulo emprega:

- dois operadores $f.\theta_c$ para calcular os produtos $g.\theta_b$ e $psi.\theta_b$ utilizados na atualização das funções f e ϕ de alguma unidade MP;
- registradores para memorizar uma síndrome β (ela guarda uma síndrome α relativa à função f associada a g), assim como $\dot{\beta}$, $\dot{g}$ e $\dot{\psi}$ (valores produzidos na última iteração).

3.4 MÓDULO CONTROLADOR

O circuito controlador do decodificador gera todos os sinais de controle para os módulos MP e AP, assim como para o buffer das síndromes de entrada, e para a saída do decodificador. Ele é também responsável por gerar internamente os conjuntos σ e δ e, por conseguinte, tomar todas as decisões relativas ao funcionamento do decodificador. É ele que determina, por exemplo, se uma função deve ser atualizada, por que tipo de operação e utilizando quais outras funções.

O controlador implementa ainda o procedimento de decisão por maioria. Numa iteração na qual $S(\theta_{it})$ é desconhecida, os módulos MP calculam os valores estimados de α, uma vez que $S(\theta_{it}) = 0$ no registrador de síndromes localizado no módulo buffer das síndromes de entrada. O módulo de decisão por maioria interno ao controlador calcula o conjunto Γ e determina os valores de σ associados. Em seguida, ele escolhe dentre os valores de α estimados qual deles corresponde a $S(\theta_{it})$. Este valor é, então, copiado no registrador de síndromes e adicionado aos valores de α em todos os módulos MP.

Dentre os sinais gerados pelo módulo controlador, destacam-se os sinais D, que controlam todo o funcionamento dos módulos $f.\theta_c$ (*cf.* figura 5). Uma descrição destes sinais D é apresentada na Seção 3.2.1.

Dentre as decisões tomadas pelo controlador, destaca-se a determinação das anti-lacunas a e b utilizadas no cálculo dos produtos $f.\theta_a$, $\phi.\theta_a$, $g.\theta_b$ e $\psi.\theta_b$, assim como para o endereçamento dos valores armazenados nos módulos MP para os módulos AP, e vice-versa. Dois sub-módulos do controlador são responsáveis por estes cálculos e estas tomadas de decisão: os módulos NONGAPMP e NONGAPAP (*cf.* figuras 10 e 11).

3.5 COMPLEXIDADE

Como já foi afirmado, em [28] O'Sullivan não trata da questão da complexidade de seu algoritmo. Ele afirma que a relação entre a capacidade de correção e o número de iterações necessárias para decodificar uma palavra recebida não é trivial. Ele também não determina o número de funções f e ϕ calculadas pelo algoritmo.

De modo a possibilitar a implementação deste decodificador, fez-se necessário determinar invariavelmente estes parâmetros. No que diz respeito ao número de iterações necessárias para corrigir até metade da distância mínima do código, considerou-se aqui o trabalho apresentado por Feng e Rao em [11]. Considerando que $m = n - k$ é o número de linhas da matriz de paridade do código (número de síndromes conhecidas a partir do vetor recebido) e g é o gênero da curva, Feng e Rao afirmam que em $it_{max} = m + g$ iterações a decodificação utilizando o esquema de decisão por maioria por eles proposto corrige até metade da distância mínima do código.

Pode-se verificar que numa dada iteração it uma operação $f.\theta_c$ fornece uma função que apresenta valorização discreta em Q máxima $-it$. Conseqüentemente, a valorização discreta em Q máxima de uma função qualquer do decodificador numa iteração it é sempre $-it$. Portanto, necessita-se implementar módulos $f.\theta_c$ e registradores para funções de comprimento $pl = it_{max} + 1$.

Em termos de uso de recursos materiais, um módulo $f.\theta_c$ emprega:

- pl operadores de adição em $\mathbb{F}_q$;
- $pl.r$ flip-flops;
- $pl.r$ demultiplexadores 2×1;
- $pl.r$ demultiplexadores 4×1;
- $pl - r^2 + 1$ comparadores de inteiros de rl bits (número de bits necessários para representar o valor de r);
- $pl - r^2 + 1$ portas AND;
- $pl - r^2 + 1$ portas OR.

Um módulo AP emprega:

- 2 módulos $f.\theta_c$;
- $(4pl + 2).r$ flip-flops.

Já um módulo MP, responsável pela quase totalidade dos cálculos da decodificação, emprega:

- 2 módulos $f.\theta_c$;
- $(4pl + 2).r$ flip-flops;
- $(4pl + 1).r$ demultiplexadores 2×1;
- $3pl$ operadores de multiplicação em $\mathbb{F}_q$;
- $3pl - 1$ operadores de adição em $\mathbb{F}_q$;
- $(pl + 1).r$ portas AND;
- 1 decodificador DEC1;
- 1 decodificador DEC2.

4. CONCLUSÕES

Uma nova arquitetura para decodificação hardware de códigos de Hermite foi apresentada. Esta arquitetura implementa o algoritmo de decodificação de O'Sullivan proposto em [28]. Parâmetros necessários à implementação do decodificador, que não haviam sido determinados por O'Sullivan, como a ordem de pólo máxima das funções e o número de funções calculadas pelo algoritmo, foram determinados.

A arquitetura proposta consiste em r unidades MP paralelas e r unidades AP paralelas responsáveis pela atualização iterativa de funções localizadoras e avaliadoras de erros. Estas unidades são interligadas por um barramento bidirecional. Uma unidade MP emprega 2 módulos $f.\theta_c$, $(4pl + 2).r$ flip-flops, $(4pl + 1).r$ demultiplexadores 2×1, $3pl$ operadores de multiplicação em $\mathbb{F}_q$, $3pl - 1$ operadores de adição em $\mathbb{F}_q$, $(pl + 1).r$ portas AND e 2 sub-módulos decodificadores de pl bits. Uma unidade AP emprega $(4pl + 2).r$ flip-flops e 2 módulos $f.\theta$. Cada módulo $f.\theta$ emprega pl operadores de adição em $\mathbb{F}_q$, $pl.r$ flip-flops, $pl.r$ demultiplexadores 2×1, $pl.r$ demultiplexadores 4×1, $pl - r^2 + 1$ comparadores de inteiros de rl bits (número de bits necessários para representar o valor de r), $pl - r^2 + 1$ portas AND e $pl - r^2 + 1$ portas OR. A arquitetura apresenta ainda um buffer para as síndromes de entrada e um circuito controlador responsável por gerar todos os sinais de controle para o circuito.

Foram ainda propostos operadores otimizados para implementar os cálculos mais freqüentes, tais como os produtos $f.\theta_c$.

A complexidade de um módulo MP assemelha-se à de uma arquitetura de Berlekamp-Massey para decodificação de códigos RS (ou a um dos r módulos do decodificador proposto por Kötter em [20]). O que se pode concluir a partir desta arquitetura descrita sobre a complexidade de decodificadores para códigos de Hermite em comparação com códigos RS é que os decodificadores para códigos de Hermite apresentam complexidade maior que decodificadores para códigos RS numa ordem de r vezes. No entanto, códigos RS necessitam utilizar corpos finitos muito maiores, portanto uma aritmética muito mais complexa, para corrigir vetores de mesmo comprimento. Desta forma, estima-se que a complexidade geral para comprimentos de código iguais (em número de bits) seja equivalente para ambos os códigos de Hermite e RS.

REFERÊNCIAS

[1] ASHBROOK, J. B., SHANBHAG, N. R., KÖTTER, R., AND BLAHUT, R. E. Implementation of a Hermitian decoder IC in 0.35μm CMOS. In *IEEE Custom Integrated Circuits Conference Proceedings* (2001), pp. 297–300.

[2] BELKOURA, Z. M., AND DE BARROS NAVINER, L. A. Hardware implementation issues of a BMS decoding approach for AG based codes. In *Proceedings of the IEEE Wireless Communications and Networking Conference* (2003), pp. 448–453.

[3] BERLEKAMP, E. R. Bounded distance + 1 soft-decision Reed-Solomon decoding. *IEEE Transactions on Information Theory 42*, 3 (1996), 704–720.

[4] BLAHUT, R. E. Encoding of codes on curves. In *IEEE International Symposium on Information Theory* (1994).

[5] CUNSHENG, D., NIEDERREITER, H., AND CHAOPING, X. Some new codes from algebraic curves. *IEEE Transactions on Information Theory 46* (2000), 2638–2642.

[6] DA SILVA LIMA, L. B., DE ASSIS, F. M., AND DE BARROS NAVINER, L. A. Decodificação de códigos algébrico-geométricos (in english, decoding of algebraic-geometric codes). *Revista da Sociedade Brasileira de Telecomunicações 18*, 3 (Dec. 2003), in press.

[7] DA SILVA LIMA, L. B., DE BARROS NAVINER, L. A., JAUBERT, J., AND DE ASSIS, F. M. Architecture for a decoder of algebraic-geometric codes based on Hermitian curves. In *MWSCAS Proceedings* (2003).

[8] DA SILVA LIMA, L. B., DE BARROS NAVINER, L. A., JAUBERT, J., AND DE ASSIS, F. M. Architecture pour un décodeur de codes algébriques basés sur les courbes d'Hermite. In *JSF Proceedings* (2003), pp. 226–230.

[9] DE ASSIS, F. M. Hit probability between frequency hopping sequences generated by Reed-Solomon and Hermitian codes. *Electronics Letters 32*, 11 (May 1996), 962–963.

[10] ELIA, M., VITERBO, E., AND BERTINETTI, G. Decoding of binary separable Goppa codes using Berlekamp-Massey algorithm. *Electronics Letters 35* (1999), 1720–1721.

[11] FENG, G.-L., AND RAO, T. R. N. Decoding of algebraic geometric codes up to the designed minimum distance. *IEEE Transactions on Information Theory 39*, 1 (Jan. 1993), 37–45.

[12] FENG, G.-L., WEI, V. K., RAO, T. R. N., AND TZENG, K. K. Simplified understanding and efficient decoding of a class of algebraic-geometric codes. *IEEE Transactions on Information Theory 40*, 4 (July 1994), 981–1002.

[13] FULTON, W. *Algebraic Curves: An Introduction to Algebraic Geometry*. Reading, MA: W. A. Benjamin, 1969.

[14] GOPPA, V. D. A new class of linear error-correcting codes. *Problems of Information Theory 6* (1970), 207–212.

[15] GURUSWAMI, V., AND SUDAN, M. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory 45* (Mar. 1999), 432–437.

[16] JENSEN, C. D. Fast decoding of codes from algebraic geometry. *IEEE Transactions on Information Theory 40* (Jan. 1994), 223–230.

[17] JUSTESEN, J., LARSEN, K. J., JENSEN, H. E., HAVEMOSE, A., AND HØHOLDT, T. Construction and decoding of a class of algebraic geometric codes. *IEEE Transactions on Information Theory 35*, 4 (July 1989), 811–821.

[18] JUSTESEN, J., LARSEN, K. J., JENSEN, H. E., AND HØHOLDT, T. Fast decoding of codes from algebraic plane curves. *IEEE Transactions on Information Theory 38*, 1 (Jan. 1992), 111–119.

[19] KURIHATA, M., AND SAKATA, S. A fast parallel decoding algorithm for general one-point AG codes with a systolic array architecture. In *ISIT Proceedings* (1995), p. 99.

[20] KÖTTER, R. Fast generalized minimum-distance decoding of algebraic-geometry and Reed-Solomon codes. *IEEE Transactions on Information Theory 42*, 3 (May 1996), 721–737.

[21] KÖTTER, R. A fast parallel implementation of a Berlekamp-Massey algorithm for algebraic-geometric codes. *IEEE Transactions on Information Theory 44* (July 1998), 1353–1368.

[22] KÖTTER, R., AND VARDY, A. Algebraic soft-decision decoding of Reed-Solomon codes. In *ISIT Proceedings* (2000).

[23] LING, C. X. S. A class of linear codes with good parameters from algebraic curves. *IEEE Transactions on Information Theory 46* (2000), 1527–1532.

[24] LINT, J. H. V. Algebraic geometric codes. In *Coding Theory and Design Theory (Part I)*, vol. 21 of *IMA Volumes Math. Appl.* Berlin: Springer-Verlag, 1990, pp. 137–162.

[25] LIU, C.-W., HUANG, K.-T., AND LU, C.-C. A fast parallel implementation of Feng-Rao algorithm with systolic array structure. In *ISIT Proceedings* (1997).

[26] VAN LINT, J. H., AND SPRINGER, T. A. Generalized Reed-Solomon codes from algebraic geometry. *IEEE Transactions on Information Theory 33*, 3 (May 1987), 305–309.

[27] O'SULLIVAN, M. E. Decoding of codes defined by a single point on a curve. *IEEE Transactions on Information Theory 41*, 6 (Nov. 1995), 1709–1719.

[28] O'SULLIVAN, M. E. Decoding of Hermitian codes: The key equation and efficient error evaluation. *IEEE Transactions on Information Theory 46*, 2 (Mar. 2000), 512–523.

[29] O'SULLIVAN, M. E., AND POPE, S. P. VLSI architecture for a decoder for Hermitian codes. In *ISIT Proceedings* (1997), p. 376.

[30] PONNAMPALAM, V., AND VUCETIC, B. Soft decision decoding of Reed-Solomon codes. In *ISIT Proceedings* (2000).

[31] POPOVICI, E. M., O'SULLIVAN, M. E., FITZPATRICK, P., AND KÖTTER, R. Implementation of a Hermitian decoder. In *ISIT Proceedings* (2001), p. 311.

[32] PORTER, S. C., SHEN, B.-Z., AND PELLIKAAN, R. Decoding geometric Goppa codes using an extra place. *IEEE Transactions on Information Theory 38*, 6 (Nov. 1992), 1663–1676.

[33] SAINTS, K., AND HEEGARD, C. Algebraic-geometric codes and multidimensional cyclic codes: A unified theory and algorithms for decoding using Gröbner bases. *IEEE Transactions on Information Theory 41*, 6 (Nov. 1995), 1733–1751.

[34] SAKATA, S. Extension of the Berlekamp-Massey algorithm to N dimensions. *Informat. Comput. 84* (Feb. 1990), 207–239.

[35] SAKATA, S., JENSEN, H. E., AND HØHOLDT, T. Generalized Berlekamp-Massey decoding of algebraic geometric codes up to half the Feng-Rao bound. *IEEE Transactions on Information Theory 41* (Nov. 1995), 1762–1768.

[36] SAKATA, S., JUSTESEN, J., MADELUNG, Y., JENSEN, H. E., AND HØHOLDT, T. Fast decoding of algebraic-geometric codes up to the designed minimum distance. *IEEE Transactions on Information Theory 41*, 5 (Sept. 1995), 1672–1677.

[37] SAKATA, S., AND KURIHATA, M. A systolic array architecture for implementing a fast parallel decoding algorithm for one-point AG codes. In *ISIT Proceedings* (1997), p. 378.

[38] SAKATA, S., LEONARD, D. A., JENSEN, H. E., AND HØHOLDT, T. Fast erasure-and-error decoding of algebraic geometry codes up to the Feng-Rao bound. *IEEE Transactions on Information Theory 44* (July 1998), 1558–1565.

[39] SAKATA, S., NUMAKANI, Y., AND FUJISAWA, M. A fast interpolation method for list decoding of RS and algebraic-geometric codes. In *ISIT Proceedings* (2000), p. 479.

[40] SKOROBOGATOV, A. N., AND VLĂDUŢ, S. G. On the decoding of algebraic-geometric codes. *IEEE Transactions on Information Theory 36*, 5 (Sept. 1990), 1051–1060.

[41] STICHTENOTH, H. A note on Hermitian codes over $GF\ \ q^2$. *IEEE Transactions on Information Theory 34*, 5 (Sept. 1988), 1345–1348.

[42] STICHTENOTH, H. *Algebraic Function Fields and Codes*. Berlin: Springer-Verlag, 1993.

[43] TSFASMAN, M. A., VLĂDUŢ, S. G., AND ZINK, T. Modular curves, Shimura curves and Goppa codes, better than Varshamov-Gilbert bound. *Math. Nachr. 104* (1982), 13–28.

[44] WU, X.-W., AND SIEGEL, P. H. Efficient root-finding algorithm with application to list decoding of algebraic-geometric codes. *IEEE Transactions on Information Theory 47*, 6 (Sept. 2001), 2579–2587.

Leocarlos B. S. Lima formou-se Engenheiro Eletricista em 1997 pela então Universidade Federal da Paraíba – UFPB, Campus II, hoje Universidade Federal de Campina Grande – UFCG, em Campina Grande-PB, Brasil. Concluiu mestrado na área de Telecomunicações em 1999 pela mesma instituição. Entre 1999 e 2000, trabalhou pela Netcon Ltda. de Recife-PE, Brasil, como supervisor de obras de instalação de cabos e terminações ópticas da rede nacional de telefonia da Intelig. Em 2004, concluiu doutorado na área de Telecomunicações pela UFCG e pela École Nationale Supérieure des Télécommunications – ENST, em Paris, França. Atualmente, é professor pelo Centro Universitário de João Pessoa – UNIPÊ, em João Pessoa-PB. Suas áreas de interesse são codificação, criptografia e desenvolvimento algoritmo-arquitetural.

Francisco M. Assis possui a seguinte formação acadêmica: Arma de Comunicações pela Academia Militar das Agulhas Negras (AMAN/RJ), 1978, Engenharia Elétrica (Telecomunicações) pelo Instituto Militar de Engenharia (IME/RJ), 1985, Mestrado em Sistemas de Comunicações, pelo IME/RJ, 1991, Doutorado em Sistemas de Telecomunicações pela Pontifícia Universidade Católica do Rio de Janeiro (PUC/Rio), 1994. É professor adjunto do Departamento de Engenharia Elétrica da Universidade Federal de Campina Grande (DEE-UFCG) desde 1993. Suas áreas de interesse são teoria da informação, codificação e complexidade.

Lirida A. B. Naviner concluiu os cursos de Engenharia Elétrica e Mestrado em Processamento da Informação pela Universidade Federal da Paraíba – UFPB, respectivamente em 1988 e 1990. De 1994 a 1997, após conclusão do Doutorado em Eletrônica e Comunicações pela École Nationale Supérieure des Télécommunications – ENST, foi professora pesquisadora junto aos departamentos de Engenharia Elétrica e Sistemas de Computação da UFPB. Desde 1998, faz parte do quadro permanente de professores da ENST em Paris, França, onde exerce atividades de ensino, pesquisa e extensão. Membro do Centre National de Recherche Scientifique – CNRS, seus temas de interesse atuais são codificação de fonte/canal, filtragem para sistemas de comunicação multi-padrões, hw/sw codesign e arquiteturas reconfiguráveis e evolutivas.

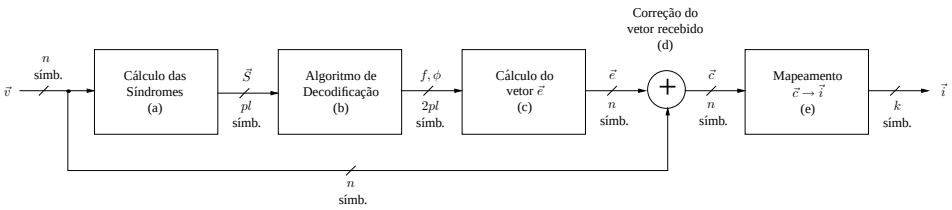

Figura 1. Esquema completo de decodificação baseado na abordagem proposta por O'Sullivan [28]. Considera-se um código de bloco (n, k). O comprimento pl de um polinômio é igual ao número máximo de iterações necessárias para processar a decodificação mais 1.

it	σ e δ	β	f e g	ϕ e ψ
-1	$\sigma : 0$		1	0
	$\delta : \emptyset$			
0	$\sigma : 4$		x	$\alpha^7 y^3$
	$\sigma : 5$		y	$\alpha^7 x^4$
	$\delta : 0$	α^7	1	0
4	$\sigma : 4$		$x+1$	$\alpha^7 y^3$
	$\sigma : 5$		y	$\alpha^7 x^4 + \alpha^7 x^3$
	$\delta : 0$	α^7	1	0
5	$\sigma : 4$		$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
	$\sigma : 5$		$y+1$	$\alpha^7 x^4 + \alpha^7 x^3$
	$\delta : 0$	α^7	1	0
8	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^7 xy^2$
	$\sigma : 5$		$y+1$	$\alpha^7 x^4 + \alpha^7 x^3 + \alpha^4 x^2$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
9	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^7 xy^2 + y^2$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
10	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^7 xy^2 + y^2 \alpha^2 xy$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
11	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^7 xy^2 + y^2 \alpha^2 xy$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2 + \alpha^{14} y$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
12	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^4 y^3 + \alpha^7 xy^2 + \alpha y^2 \alpha^2 xy$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2 + \alpha^{14} y + \alpha^9 x$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
14	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^4 y^3 + \alpha^7 xy^2 + \alpha y^2 \alpha^2 xy + \alpha^6 y$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2 + \alpha^{14} y + \alpha^9 x$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
15	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^4 y^3 + \alpha^7 xy^2 + \alpha y^2 \alpha^2 xy + \alpha^6 y + \alpha^{14} x$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2 + \alpha^{14} y + \alpha^9 x$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
16	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^4 y^3 + \alpha^7 xy^2 + \alpha y^2 \alpha^2 xy + \alpha^6 y + \alpha^{14} x$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2 + \alpha^{14} y + \alpha^9 x + \alpha^7$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
19	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^4 y^3 + \alpha^7 xy^2 + \alpha y^2 \alpha^2 xy + \alpha^6 y + \alpha^{14} x + \alpha^{11}$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2 + \alpha^{14} y + \alpha^9 x + \alpha^7$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$

Tabela 1. Resultados da decodificação do exemplo descrito na Seção 2.1. São mostrados os resultados apenas das iterações em que ocorreram mudanças.

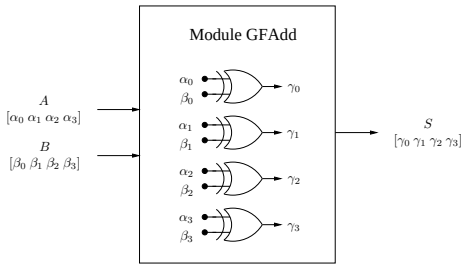

Figura 2. Arquitetura para o módulo GFADD, operador de adição de dois elementos A e B para o caso particular de $\mathbb{F}_{16}$.

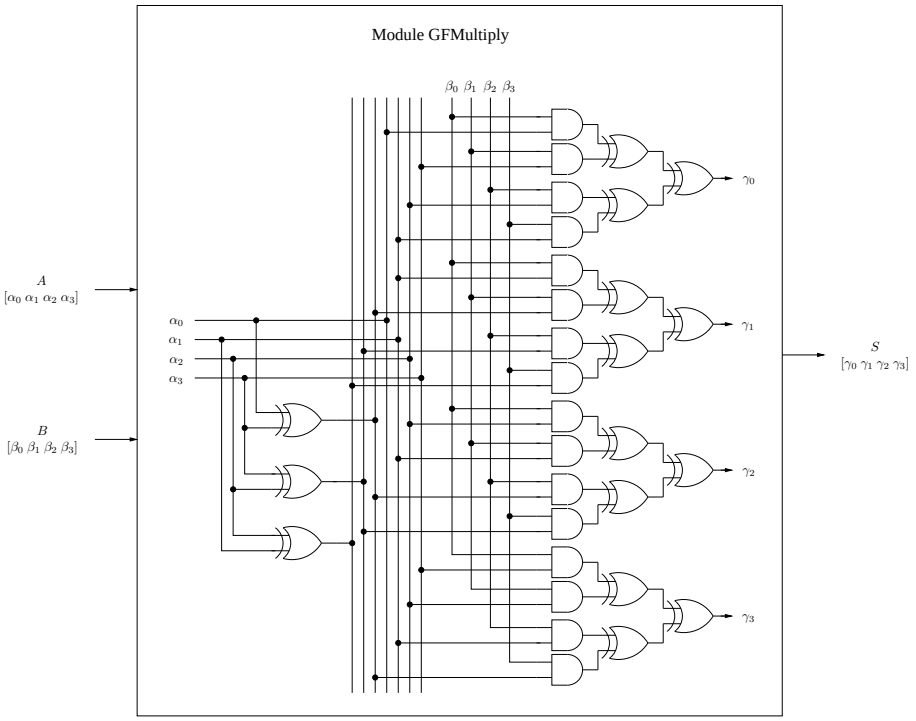

Figura 3. Arquitetura para o módulo GFMULTIPLY, operador de multiplicação de dois elementos A e B para o caso particular de $\mathbb{F}_{16}$.

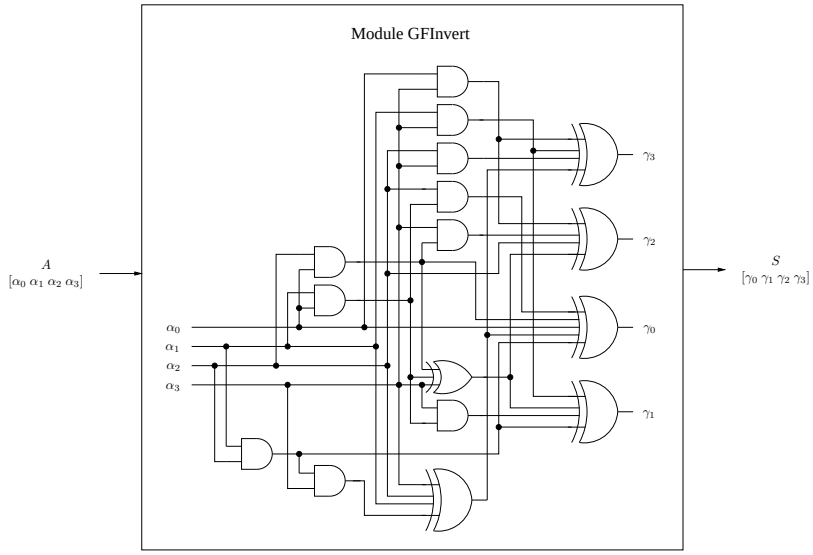

Figura 4. Arquitetura para o módulo GFINVERT, operador de inversão direta de um elemento A para o caso de $\mathbb{F}_{16}$.

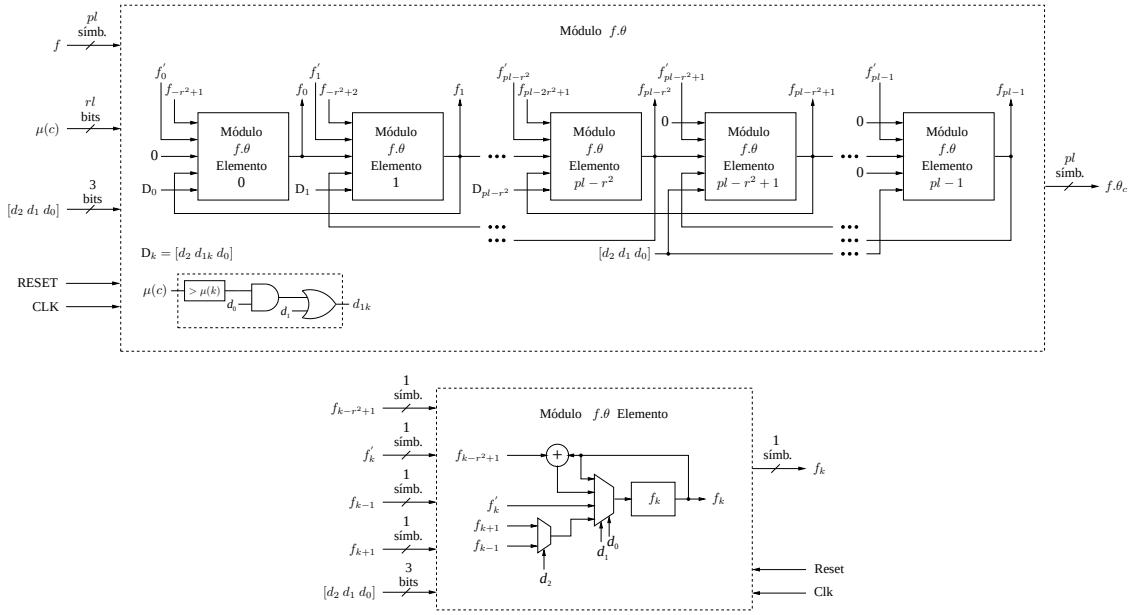

Figura 5. Arquitetura para a implementação da operação $f.\theta_c$. Num sub-módulo k do operador, considera-se como entrada da célula do registrador: $f_k^{'}$ se $d_{k1} = d_{k2} = 0$ (carregando a função de entrada $f^{'}$), $f_k + f_{k-r^2+1}$ se $d_{k1} = \overline{d_{k2}}$ e $\mu(c) \geq r - \mu(k)$, f_k se $d_{k1} = \overline{d_{k2}}$ e $\mu(c) < r - \mu(k)$, e f_{k+1} ou f_{k-1} se $d_{k1} = d_{k2} = 1$ e segundo o bit sinal de c (operação de deslocamento).

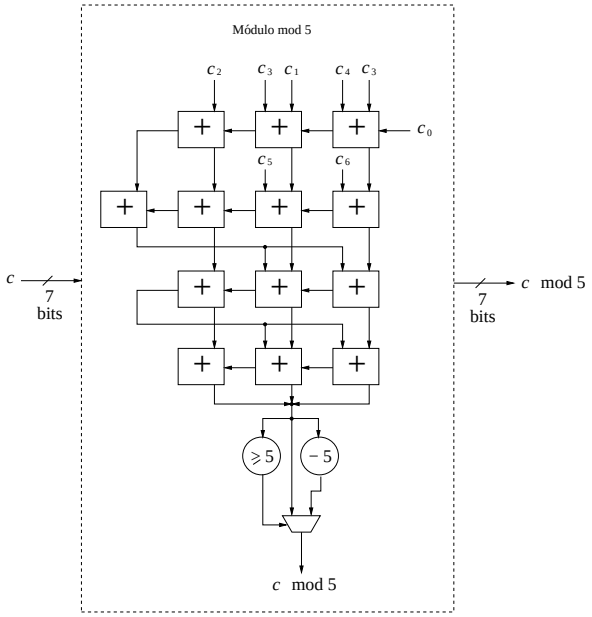

Figura 6. Arquitetura para um operador módulo 5 para um inteiro c de 7 bits de entrada (bit de sinal incluído).

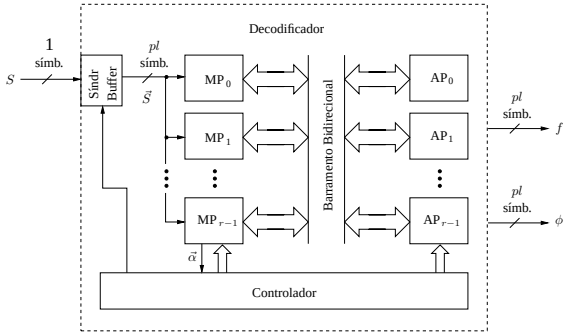

Figura 7. Arquitetura geral proposta para o decodificador de O'Sullivan [28].

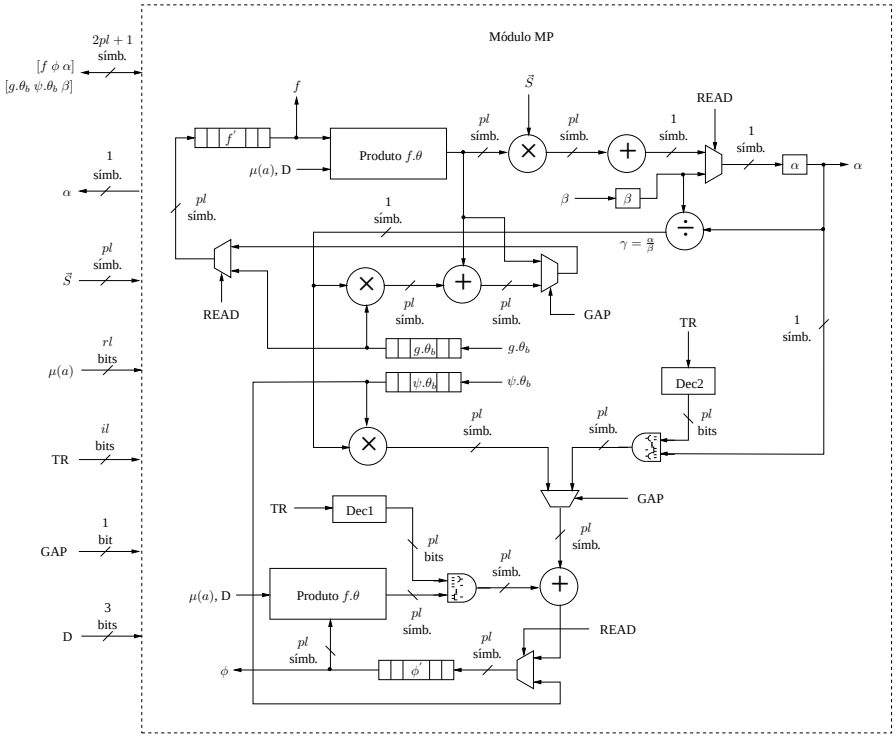

Figura 8. Arquitetura para blocos MP do decodificador, responsáveis pelo cálculo das síndromes α e pela atualização das funções localizadoras e avaliadoras de erros f e ϕ.

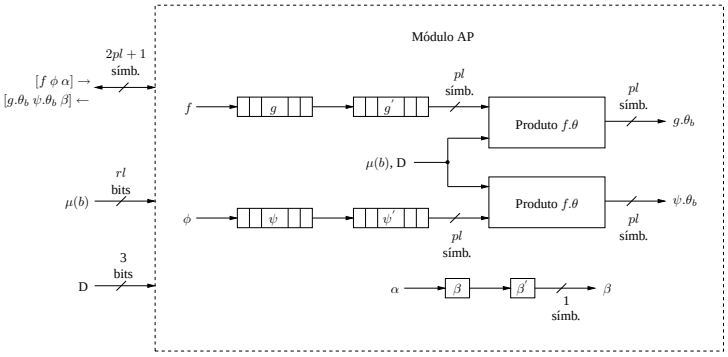

Figura 9. Arquitetura para blocos AP do decodificador, responsáveis pela atualização das funções auxiliares e pelo cálculo dos produtos $g.\theta_b$ e $\psi.\theta_b$.

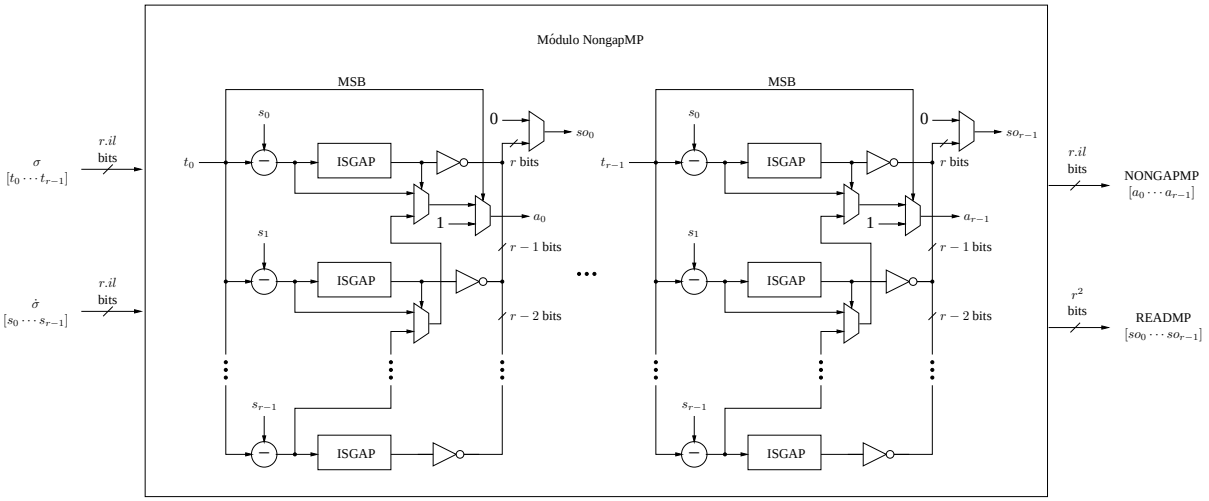

Figura 10. Arquitetura para um módulo NONGAPMP responsável pela determinação das anti-lacunas a utilizadas na atualização de funções, assim como pelo endereçamento dos valores armazenados nos módulos MP para os módulos AP específicos.

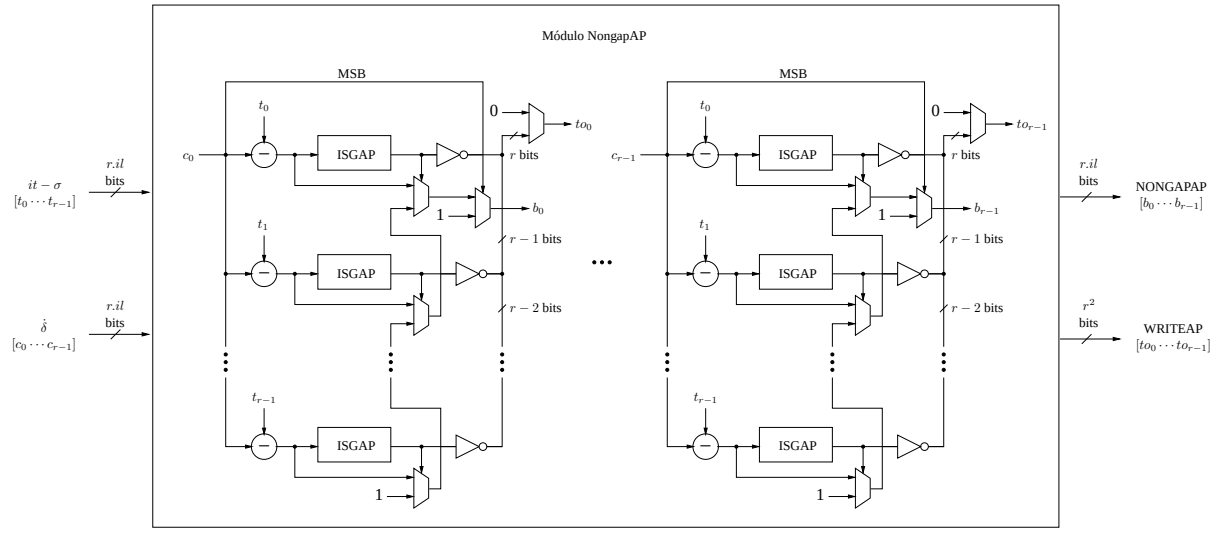

Figura 11. Arquitetura para um módulo NONGAPAP responsável pela determinação das anti-lacunas b utilizadas na atualização de funções, assim como pelo endereçamento dos valores armazenados nos módulos AP para os módulos MP específicos.