Roberto de Matos

CONVERSORES REVERSOS RNS-BINÁRIO EFICIENTES PARA APLICAÇÕES COM AMPLA FAIXA DINÂMICA

Tese submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina para obtenção do grau de Doutor em Engenharia Elétrica.

Orientador:Dr. Eduardo Augusto Bezerra
Co-orientador:Dr. Hector Pettenghi Roldan

Florianópolis
2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

de Matos, Roberto
CONVERSORES REVERSOS RNS-BINÁRIO EFICIENTES PARA APLICAÇÕES COM AMPLA FAIXA DINÂMICA / Roberto de Matos ; orientador, Eduardo Augusto Bezerra, coorientador, Héctor Pettenghi Roldán, 2018.
113 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Engenharia Elétrica, Florianópolis, 2018.

Inclui referências.

1. Engenharia Elétrica. 2. Sistema Numérico por Resíduos. 3. RNS. 4. Multiplicadores. 5. Teorema Chinês do Resto. I. Bezerra, Eduardo Augusto. II. Roldán, Héctor Pettenghi. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia Elétrica. IV. Título.

Roberto de Matos

CONVERSORES REVERSOS RNS-BINÁRIO EFICIENTES PARA APLICAÇÕES COM AMPLA FAIXA DINÂMICA

Esta Tese de Doutorado foi julgada adequada para obtenção do Título de " Doutor" em Engenharia Elétrica, Área de Concentração em Circuitos e Sistemas Integrados, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.

Florianópolis, 22 de Março de 2018

_____

Prof. Dr. Marcelo Lobo Heldwein
Coordenador do Curso

_____

Prof. Dr. Eduardo Augusto Bezerra
Orientador

_____

Prof. Dr. Hector Pettenghi Roldan
Co-orientador

Banca Examinadora:

_____

Prof. Dr. Fabian Luis Vargas
Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS

_____

Prof. Dr. Leonel Sousa
Universidade de Lisboa – IST

_____

Prof. Dr. Cesar Albenes Zeferino
Universidade do Vale do Itajaí – Univali

Dedico esse trabalho aos meus pais, Catarina e Anísio (in memorian), que sempre prezaram pela minha educação seja ela formal ou não, à minha esposa, Karin, e aos meus filhos, Miguel e Vinícius, que dão um sentido mais pleno à concretização de mais essa etapa na minha vida.

Agradecimentos

Muitas pessoas merecem agradecimentos por terem contribuído na realização desse trabalho. Em primeiro lugar meu orientador, professor Eduardo Augusto Bezerra, que desde o primeiro contato abriu a oportunidade para entrada no programa de pós-graduação e me apoiou de forma incondicional em todos os momentos do doutorado, além de ser uma inspiração como pessoa e profissional. Em seguida, meu co-orientador, professor Hector Pettenghi Roldan, que me apresentou uma nova área com dedicação e paciência, contribuiu de uma forma muito próxima no direcionamento do tema, implementações e análises. Ao colega, Rogério Paludo, que muito contribuiu com comentários, discussões, implementações e esclarecimentos sobre o tema. Finalmente, a todos os meus colegas do IFSC que apoiaram direta ou indiretamente o trabalho e ao IFSC como instituição que possibilitou o meu afastamento para conclusão do doutorado.

Resumo

Aritmética de resíduos, baseada em Sistema Numérico por Resíduos (RNS - *Residue Number Systems*), tem sido utilizada em sistemas digitais durante vários anos. RNS é uma abordagem aritmética livre de *carry* que oferece o potencial para alta velocidade e computação paralela. Operações aritméticas, como adição, subtração e multiplicação, podem ser realizadas de forma mais eficiente do que nos sistemas binários convencionais, de forma independente e simultânea, em vários canais de resíduos. A adoção de RNS tem proporcionado melhorias significativas na eficiência de diferentes tipos de aplicações de processamento digital de sinal. Uma unidade aritmética completa baseada em RNS possui quatro características principais relacionadas com a sua funcionalidade: a) conjunto de módulos, b) conversão direta, c) unidades aritméticas modulares e d) conversão reversa. Cada unidade completa é baseada em um conjunto de módulos, os quais são inteiros primos entre si. A faixa dinâmica é definida pelo produto de todos os módulos e define o intervalo de entrada. O conversor direto, também chamado de conversor binário para RNS, converte um número binário na representação RNS, baseada no conjunto de módulos. As unidades aritméticas modulares são os blocos instanciados paralelamente que de fato executam as operações livre de *carry* de soma, subtração e multiplicação. Por fim, o conversor reverso, também chamado de conversor RNS para binário, transforma os vários resíduos calculados pelos canais paralelos no resultado binário equivalente. Um grande número de canais pode melhorar o cálculo aritmético, entretanto pode haver grandes perdas de desempenho causado pelo custo da execução da conversão reversa. Com conversores reversos eficientes, capazes de suportar grandes conjuntos de módulos, é possível compensar este custo adicional, especialmente quando várias operações aritméticas precisam de ser realizadas. Este trabalho propõe conversores reversos eficientes para aplicações com uma faixa dinâmica ampla que foram implementados com um método de compactação lógica que supera o melhor conversor reverso do estado da arte, com um faixa dinâmica equivalente, apresentando uma aceleração de $2,77$ vezes, e uma redução média de $82,16\%$ e $88,32\%$ em área e potência, respectivamente.

Palavras-chave: Sistema Numérico por Resíduos, RNS, operadores aritméticos, multiplicadores, somadores, teorema chinês do resto, CRT.

Abstract

Residue Number System (RNS) has been used in digital processing realm for many years. RNS is a carry-free arithmetic system with modular characteristics offering the potential for high-speed and parallel computation. Arithmetic operations, such as addition, subtraction, and multiplication, can be carried out more efficiently than in the conventional binary systems. Because they are performed independently and concurrently in several residue channels. Each RNS unit is based on a moduli set, which are co-prime integers. The dynamic range is defined by the product of the moduli and defines the unit input range. The choice of the moduli set is of key importance in order to obtain an efficient RNS unit. A complete arithmetic unit based on RNS has three stages: a) forward conversion, b) modular arithmetic unit, and c) reverse conversion. Forward converter transforms a binary number into its specific RNS representation. The modular arithmetic unit is the processing blocks instantiated in parallel to actually perform the sum, subtraction, and multiplication. Finally, the reverse converter transforms an RNS represented number into its equivalent binary number. Several channels can improve the arithmetic computation at the cost of reverse conversion performance. With efficient reverse converters, capable of supporting large moduli sets, it is possible to compensate for this extra cost, especially when several arithmetic operations have to be performed. This work proposes an efficient reverse converter for applications with a large dynamic range. It has been implemented with a logical compression method and experimental results suggest that area reductions up to 82.16% and speed up of 2.77 can be obtained with our proposal in comparison with the best state-of-the-art reverse converter with an equivalent dynamic range. The proposed converter allows fewer bits per channel in comparison with the most efficient solutions with smaller dynamic ranges.

Keywords: Residue Number System, RNS, arithmetic operators, multipliers, adders, Chinese Remainder Theorem, CRT.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

Lista de Equações

Sumário

CAPÍTULO 1

Introdução

Com a disseminação dos sistemas computacionais móveis de comunicação, o Processamento Digital de Sinal (DSP - *Digital Signal Processing*) tem migrado de poderosos servidores para unidades especialistas dentro de processadores embarcados. A cada dia essas aplicações exigem maior velocidade na manipulação aritmética de operandos com grande comprimento de palavra. Uma das maneiras de melhorar o desempenho desses sistemas é usar uma representação numérica alternativa em vez de usar a representação convencional binária de complemento de dois.

O Sistema Numérico por Resíduos (RNS - *Residue Number Systems*) tem sido utilizado para para melhorar o desempenho de sistemas digitais durante muitos anos [1]. RNS é um sistema numérico não-posicional, no qual um grande operando é decomposto em um conjunto de operandos menores (resíduos), cada um processado em um canal paralelo independente. Assim RNS permite uma abordagem aritmética livre de *carry* com características modulares que oferece o potencial para alta velocidade e computação paralela. Operações aritméticas, como adição, subtração e multiplicação, podem ser realizadas de forma mais eficiente do que nos sistemas binários convencionais [1], de forma independente e simultânea, em vários canais de resíduo. A adoção de RNS tem proporcionado melhorias significativas de eficiência para os diferentes tipos de aplicação de processamento digital de sinal [1], além de permitir escalar facilmente para aplicações com requisitos de grande faixa dinâmica, como filtragens adaptativas e criptografia [2].

Uma das operações mais complexas e custosas de uma unidade aritmética baseada em RNS é a conversão reversa (RNS para binário). Os algoritmos mais utilizados para a conversão reversa são o Teorema Chinês do Resto (CRT - *Chinese Remainder Theorem*) [1], a Conversão de Raiz Mista (MRC - *Mixed-Radix Conversion*) [3] e o novo CRT-I [4]. A complexidade desses algoritmos depende principalmente do tamanho do módulo e do número de canais. Por isso, a escolha do conjunto de módulos é essencial para que se obtenha implementações RNS eficientes.

Os conjuntos de módulos com uma grande quantidade de canais podem melhorar a computação aritmética ao custo do desempenho da conversão reversa.

Com conversores reversos eficientes, capazes de suportar conjuntos de módulos grandes, é possível compensar esse custo extra, especialmente quando várias operações aritméticas devem ser realizadas em série. Nesses casos, o uso de múltiplos canais modulares aritméticos pode levar à melhoria do desempenho. Além disso, o conjunto de módulos também define o intervalo para representação exclusiva de inteiros, chamado de faixa dinâmica (*DR - Dynamic Range*).

Dessa forma, a maioria das pesquisas sobre RNS está focada em propor conjunto de módulos e projetar conversores reversos eficientes [5], uma vez que essa etapa possui um custo alto no hardware. Assim, um projeto sem critério pode neutralizar o ganho da paralelização nas unidades aritméticas modulares. Em outras palavras, para aumentar a aplicabilidade das unidades baseadas em RNS é necessário melhorar o desempenho global, no qual o conversor reverso tem um grande peso.

O conjunto de módulos mais tradicional é o conjunto de 3 módulos proposto por [1] $\{2^n, 2^n+1, 2^n-1\}$, com uma faixa dinâmica de aproximadamente $3n$-bit. No entanto, nem o nível de paralelismo nem a *DR* alcançado por esse conjunto são suficientes [6, 7]. Nesses casos, as extensões horizontais podem ser utilizadas para adicionar mais módulos ao conjunto de módulos, enquanto extensões verticais estendem um módulo, normalmente a potência de dois, aumentando a *DR*, mas sem aumentar o número de canais. Essas extensões tem um impacto na implementação e na eficiência da unidade completa em RNS. O importante é selecionar um conjunto de módulos equilibrado entre a faixa dinâmica requerida pela aplicação e o número de canais para paralelização das operações aritméticas.

1.1 Objetivos

O objetivo central desta tese é propor conjuntos de módulos e conversores reversos mais eficientes que os do estado da arte, levando em consideração aplicações de ampla faixa dinâmica e várias operações aritméticas por canal. Para atingir o objetivo principal, os seguintes objetivos específicos foram definidos:

- Implementar unidades aritméticas modulares, mais especificamente MAC (Multiplicador–ACumulador), para avaliação do atraso no caminho crítico de canais com diferentes larguras de *bits* e módulos.

- Propor conjunto de módulos de forma a diminuir o número de *bits* por canal, aumentando o nível de paralelismo e diminuindo o atraso no caminho crítico das unidades aritméticas dos canais modulares. Além de alcançar a maior faixa dinâmica possível para beneficiar aplicações com essas características.
- Desenvolver conversores reversos baseados nos conjuntos de módulos propostos.
- Otimizar unidades recorrentes dentro dos conversores para alcançar melhor eficiência.
- Avaliar e comparar os conjuntos e os conversores propostos com o estado da arte.

1.2 Contribuições

O escopo da tese limita-se à conversão RNS para binário, entretanto, além das contribuições na área dos conversores reversos, outros resultados foram alcançadas paralelamente ao objetivo principal. As contribuições desta tese são resumidamente as seguintes:

- Conversor reverso com maior *DR* para conjunto de módulos específico do estado da arte ($DR \simeq 9n$) [8]. Além disso, este conjunto de módulos específicos oferece operações modulares mais simples na conversão reversa, em comparação com as abordagens genéricas [9] e [10]. O conversor proposto permite menos *bits* por canal em comparação com as soluções mais eficientes com intervalos dinâmicos menores.
- Procedimento para otimizar as multiplicações modulares por constantes baseadas no módulo $\{2^n\}$, $\{2^n \pm 1\}$ e $\{2^n \pm k\}$. Tais operações demonstraram ser cruciais para o projeto conversores reversos eficientes.
- Proposta de arquitetura [11] para o conjunto modular genérico usando o Teorema Chinês do Resto Aproximado (CRTf - *Chinese Remainder Theorem with Fractional Values*) [12].

- Método composto de duas abordagens para projetar conversores reversos para conjunto de módulos genérico e escalável. A primeira abordagem é baseada em estágios iterativos usados para reduzir a complexidade da etapa final do conversor. A segunda abordagem minimiza o número de estágios iterativos necessários na conversão para apenas dois níveis, com custo mínimo de área em comparação com a solução de vários níveis e podendo escalar eficientemente com maiores conjunto de módulos e n [10].

1.3 Organização da Tese

Esta tese contém seis capítulos, incluindo esta introdução. Exceto pelo próximo capítulo, que fundamenta o assunto e sintetiza o estado da arte dos conversores reversos, e o último capítulo, que conclui a tese, os outros três capítulos tem uma relação direta com os artigos publicados/submetidos às revistas e conferências internacionais, o que facilita a compreensão de cada capítulo por si só. Embora esta abordagem de apresentação não convencional tenha sido adotada, os capítulos foram estruturados de forma consistente, a fim de garantir uma fácil leitura e compreensão de todas as idéias e soluções propostas como trabalho unificado. Isso traz para a tese duas características importantes: i) Clareza na apresentação de contribuições de pesquisa; ii) Uma compreensão mais fácil da análise entre as propostas de conjuntos de módulos/implementações/otimizações e os resultados experimentais, uma vez que são reunidos em cada capítulo. Esclarecido isso, abaixo é apresentada a descrição em detalhes do restante da tese:

- **Capítulo 2** – Este capítulo apresenta uma fundamentação básica sobre RNS; o processo de conversão binário para RNS (conversão direta); as unidades aritméticas modulares de multiplicação e soma; os algoritmos de conversão RNS para binário (conversão reversa). Em seguida apresenta e analisa os conjuntos de módulos usados nos conversores reversos mais eficientes do estado da arte.

- **Capítulo 3** – Este capítulo apresenta duas técnicas de otimização de unidades de multiplicação modular por constante. Tais operações são cruciais para o projeto de conversores reversos e unidades aritméticas eficientes. Ambos os métodos foram aplicadas em unidades MACs para diferentes módulos e larguras de operandos, com o objetivo de demonstrar seus ga-

nhos e limitações. Nos próximos capítulos, essas técnicas são aplicadas nas propostas e os resultados experimentais demostram que a otimização dos multiplicadores modulares tem um impacto na velocidade e na área dos conversores reversos com as maiores faixas dinâmicas do estado da arte.

- **Capítulo 4** – Neste capítulo são propostas extensões verticais e horizontais do conjunto tradicional de 3 módulos $\{2^n, 2^n-1, 2^n+1\}$, com a faixa dinâmica equivalente a $3n$-bit, para escalar a faixa dinâmica e melhorar o paralelismo de acordo com os requisitos. São apresentados dois métodos diferentes para projetar conversores reversos genéricos para conjuntos de módulos estendidos para os intervalos dinâmicos desejados. Os resultados experimentais demonstram que as abordagens propostas alcançam reduções significativas da área. Além disso, os resultados obtidos também validam a escalabilidade melhorada das abordagens propostas, permitindo melhores resultados com o aumento de n e da DR. Ainda é proposta uma arquitetura de hardware melhorada para o algoritmo de conversão reversa (CRTf), usando conjuntos de módulos genéricos balanceados.

- **Capítulo 5** – Este capítulo sintetiza uma das maiores contribuições da tese: Um método para projetar conversores reversos RNS eficientes, incluindo as técnicas citadas acima, usando um novo conjunto de módulos com a faixa dinâmica de $9n$ *bits*. Esse conjunto apresenta a maior faixa dinâmica do estado da arte, que até então restringia-se a implementações de conversores reversos eficientes até $DR \simeq 8n$. A abordagem proposta resulta em maior paralelismo com um número menor de *bits* por canal.

- **Capítulo 6** – Finalmente, este capítulo apresenta as conclusões e considerações sobre as principais contribuições deste trabalho, além de sugerir novos desdobramentos na forma de trabalhos futuros.

CAPÍTULO 2

Fundamentação e Estado da Arte

Uma sistema típico baseado em RNS, apresentado na Figura 2.1, possui quatro características principais relacionadas com a sua funcionalidade: *i*) conjunto de módulos, *ii*) conversão direta, *iii*) unidades aritméticas modulares e *iv*) conversão reversa. Cada unidade completa é baseada em um conjunto de módulos, os quais são inteiros primos entre si (co-primos). A faixa dinâmica é definida pelo produto de todos os módulos e define o intervalo de entrada. O conversor direto, também chamado de conversor binário para RNS, converte um número binário na para sua representação equivalente em RNS, baseada no conjunto de módulos. As unidades aritméticas modulares são os blocos que de fato executam paralelamente as operações de soma, subtração e multiplicação livres de *carry* . Por fim, o conversor reverso, também chamado de conversor RNS para binário, transforma os vários resíduos calculados pelos canais paralelos em um número binário equivalente. Esta seção apresentará de forma básica o sistema numérico residual, seleção do conjuntos de módulos, a etapa de conversão direta, as unidades aritméticas modulares de soma e multiplicação e, finalmente, a conversão reversa.

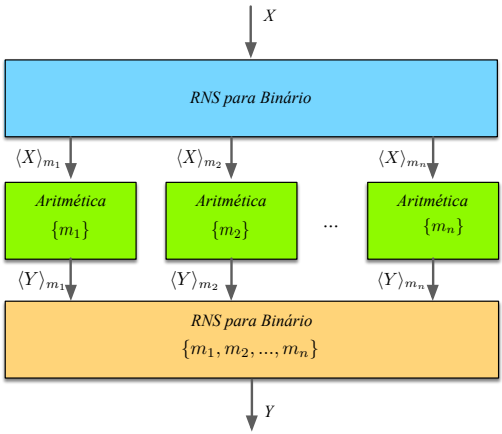


Figura 2.1: *Diagrama de blocos de uma unidade aritmética completa baseada em RNS.*

2.1 Introdução do Sistema Numérico de Resíduos

A representação de um valor em RNS se dá por uma sequência de valores menores para uma determinada base. Essa base é formada por um conjunto de módulos $\{m_1, m_2, ..., m_n\}$. Os módulos devem ser co-primos, ou seja, o MDC (Máximo Divisor Comum) deve ser igual a 1, i.e., $MDC(m_i, m_j) = 1$, onde $i \neq j$. Assim, um valor X representado em RNS tem a forma $X = (R_1, R_2, ..., R_n)$, onde:

$$R_i = X \ mod \ m_i = |X|_{m_i}, \qquad 0 \leq R_i < m_i. \tag{2.1}$$

Ou seja, o valor R_i é o resto da divisão inteira de X por m_i. Por exemplo, considerando um sistema RNS baseado no conjunto de módulos $\{5, 8, 11\}$. A representação dos valores 32 e 48 seria:

$$|32|_5 = 2, \qquad |32|_8 = 0, \qquad |32|_{11} = 10$$
$$|48|_5 = 3, \qquad |48|_8 = 0, \qquad |48|_{11} = 4.$$

Portanto, o valor 32 e 48 são representados em $RNS_{(5,8,11)}$ como $(2, 0, 10)$ e $(3, 0, 4)$, respectivamente. Podemos fazer a soma desses dois números utilizando a representação por resíduo:

$$|2+3|_5 = |0|_5, \qquad |0+0|_8 = |0|_8, \qquad |10+4|_{11} = |3|_{11}.$$

O valor da soma é 80, equivalente à representação $RNS_{(5,6,9)}$ $(0, 0, 3)$. A verificação nos confirma a operação:

$$|80|_5 = 0, \qquad |80|_8 = 0, \qquad |80|_{11} = 3.$$

Isso demonstra como as operações de soma podem ser executadas paralelamente em diferentes canais residuais sem propagação de *carry*. O resto das operações segue o mesmo princípio.

Como citado anteriormente, o intervalo para representação exclusiva de inteiros é dada pelo DR (*Dynamic Range*) de cada conjunto de módulos, ou

seja, cada X terá uma representação RNS única desde que esteja no intervalo de $[0, DR-1]$. O DR é calculado multiplicando todos os módulos do conjunto: $DR = m_1 \times m_2 \times ... \times m_n$. Para o conjunto de módulos usado como exemplo $\{5, 8, 11\}$ a faixa dinâmica é $[0, (5 \times 8 \times 11 - 1)] = [0, 439]$. Outra forma de representar o DR aproximado de um conjunto de módulos é utilizando o número de *bits* do produto.

2.2 Conjunto de Módulos

Duas características importantes em unidades RNS são: i) O nível de paralelismo, ou seja, o número de canais modulares em que o operando será dividido e processado em paralelo; ii) A faixa dinâmica (DR), que é o intervalo para representação exclusiva de inteiros. Essas duas características são definidas pelo conjunto de módulos, bem como a complexidade das conversões. Está demonstrado no estado da arte que os conjuntos de módulos que são potência de dois $\{2^n, 2^n \pm 1\}$ simplificam as operações aritméticas necessárias para as implementações RNS [5]. O conjunto de módulos mais tradicional é $\{2^n, 2^n - 1, 2^n + 1\}$, com três módulos e o DR por volta de $3n$ *bits* ($DR \simeq 3n$) [6, 7]. Entretanto, esse DR não é o suficiente para aplicações que exigem uma faixa dinâmica maior e mais paralelismo. Dessa forma, novos conjuntos de módulos vêm sendo introduzidos nos últimos anos. Abaixo é apresentada uma lista de alguns conjuntos que propõem o aumento da faixa dinâmica e na Tabela 2.1 eles são organizados em uma matriz de número de módulos por DR. Estes conjuntos de módulos serão revisitados em detalhes no estado da arte dos conversores reversos (Subseção 2.5.1).

Tabela 2.1: *Principais conjuntos de módulos do estado da arte organizados com relação ao número de módulos vs. DR.*

		Número de Módulos			
		3	4	5	6
DR	$4n$	[7]	[13] [14] [15]	–	–
	$5n$	–	[16] [17]	[18] [19] [20]	–
	$6n$	–	[16] [17]	[21] com ($\beta = n$)	[21] com ($\beta = 0n$)
	$7n$	–	–	[21] com ($\beta = 2n$)	[21] com ($\beta = n$)
	$8n+1$	–	–	–	[21] com ($\beta = 2n$)

- $\{2^n, 2^n \pm 1, \mathbf{2^{n+1} + 1}\}$ [13];
- $\{2^n, 2^n \pm 1, \mathbf{2^{n+1} - 1}\}$ [14];
- $\{2^n, 2^n \pm 1, \mathbf{2^{n-1} + 1}\}$ e $\{2^n, 2^n \pm 1, \mathbf{2^{n-1} - 1}\}$ [15].
- $\{2^{n+\beta}, 2^n \pm 1\}$ [7], onde $0 \leq \beta \leq n$ é usado para extender o *DR* verticalmente até 4*n-bits* com o conjunto de 3 módulos. A sobrecarga do canal 2^n até 2^{2n} pode ser feito sem afetar o atraso nos canais aritméticos.
- $\{2^n, 2^n \pm 1, \mathbf{2^{n\pm1} - 1}\}$ [18];
- $\{2^{n+1}, 2^n \pm 1, \mathbf{2^{n+1} \pm 1}\}$ [19], os módulos considerados nesse trabalho são números co-primos para n par, porém, multiplicativas inversas complexas são necessárias, resultando em uma estrutura de conversão reversa bastante custosa;
- $\{2^n, 2^n \pm 1, \mathbf{2^n \pm 2^{\frac{(n+1)}{2}} + 1}\}$ [20], o qual é composto de módulos co-primos para n ímpar e foi revisitado por Hiasat em [22].
- $\{\mathbf{2^{\overbrace{2n}^{n+\beta}}}, 2^n \pm 1, \mathbf{2^n \pm 2^{\frac{(n+1)}{2}} + 1}\}$. Extensão puramente vertical proposta por [21], com ($\beta = n$), do conjunto apresentando em [22].
- $\{\mathbf{2^{2n}}, 2^n \pm 1, \mathbf{2^{2n+1} - 1}\}$ [16], extensão vertical e horizontal ao custo de um conjunto de módulos desbalanceado.
- $\{\mathbf{2^{2n}}, 2^n \pm 1, \mathbf{2^{2n} + 1}\}$ [17], extensão vertical e horizontal ao custo de um conjunto de módulos desbalanceado também.
- $\{\mathbf{2^{\overbrace{3n}^{n+\beta}}}, 2^n \pm 1, \mathbf{2^n \pm 2^{\frac{(n+1)}{2}} + 1}\}$. Extensão puramente vertical proposta por [21], com ($\beta = 2n$), do conjunto apresentando em [22].
- $\{\mathbf{2^{\overbrace{3n}^{n+\beta}}}, 2^n \pm 1, \mathbf{2^n \pm 2^{\frac{(n+1)}{2}} + 1, 2^{n+1} + 1}\}$. Extensão vertical e horizontal proposta por [21], com ($\beta = 2n$). Conjunto se mantém balanceado entre os canais

2.3 Conversores Diretos

Apesar da complexidade dos conversores reversos ser maior e poder inviabilizar a aplicação de RNS na aceleração de algoritmos de processamento digitais, os conversores diretos, também chamados de conversores binários para RNS, tem um papel importante no objetivo de implementar uma unidade aritmética completa baseada em RNS. O conversor direto consiste em unidades independentes para cada módulo do conjunto de módulos definidos para transformar um número inteiro binário ponderado em sua representação equivalente RNS. A Figura 2.2 apresenta um esquema gráfico que representa a conversão de uma variável X para os módulos $2^n, 2^n-1, 2^n+1$. Considerando que a variável X tem m bits, ou seja, $X = (x_{m-1}x_{m-2}...x_1x_0)_2$, ela deve ser dividida em j blocos (b_i) de tamanho n. Para cálculo do valor $|X|_{2^n}$ somente os n bits menos significativos são considerados, o resto dos bits são desprezados. Esse módulo proporciona uma conversão direta sem a necessidade de hardware, sendo representada matematicamente por:

$$|X|_{2^n} = b_0 = (x_{n-1}...x_1x_0)_2. \tag{2.2}$$

No processo de conversão para os módulos $2^n \pm 1$ todos os blocos (b_i) são somados aplicando o módulo correspondente, com a diferença que para o módulo 2^n+1 os blocos ímpares, destacado na Figura 2.2, são negativos. A Eq. (2.3) e a Eq. (2.4) apresentam o processo de conversão para os módulos 2^n-1 e 2^n+1, respectivamente.

$$|X|_{2^n-1} = \left| \sum_{i=0}^{j-1} b_i \right|_{2^n-1} \tag{2.3}$$

$$|X|_{2^n+1} = \left| \sum_{i=0}^{j-1} (-1)^i b_i \right|_{2^n+1} \tag{2.4}$$

Abaixo a Eq. (2.5) converte um valor binário X de tamanho j *bits*, ou seja, $X = (x_{j-1}x_{j-2}...x_1x_0)_2$, em seu valor módulo m ($|X|_m$):

$$|X|_m = \left| \sum_{i=0}^{j-1} 2^i x_i \right|_m = \left| \sum_{i=0}^{j-1} |2^i|_m x_i \right|_m = \left| \sum_{i=0}^{j-1} b_i \right|_m \tag{2.5}$$

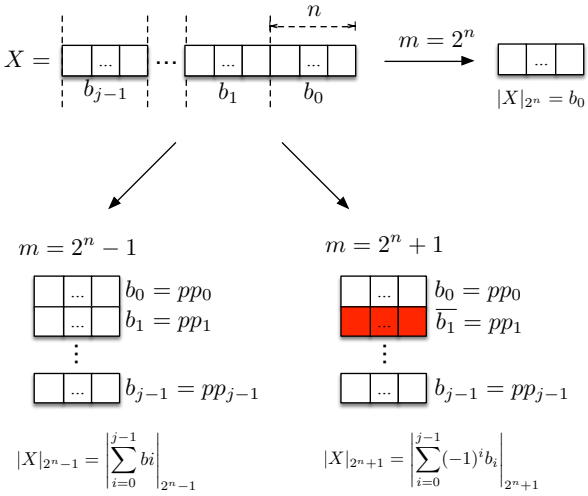


Figura 2.2: *Esquema de conversão direta para os módulos* $2^n, 2^n-1, 2^n+1$ *de um valor* X.

Se substituirmos os módulos m por $2^n \pm k$ de forma genérica, com k ímpar, já que só números ímpares podem ser co-primos, temos:

$$|X|_{2^n-k} = \left|\sum_{i=0}^{j-1} k^i b_i\right|_{2^n-k} = \left|\sum_{i=0}^{j-1} \left|k^i\right|_{2^n-k} b_i\right|_{2^n-k} \tag{2.6}$$

$$|X|_{2^n+k} = \left|\sum_{i=0}^{j-1} (-k)^i b_i\right|_{2^n+k} = \left|\sum_{i=0}^{j-1} \left|-k^i\right|_{2^n+k} b_i\right|_{2^n+k} \tag{2.7}$$

Por exemplo, considerando $n = 3$ para $\{2^n\}$, $\{2^n-1\}$ e $\{2^n+1\}$, temos o seguinte conjunto de módulos $8, 7, 9$. Para calcular os resíduos do valor binário $X = 119$ temos:

$$r_1 = |X|_{2^n} = |119|_8 = |(01110111)_2|_8 = (111)_2 = 7$$

$$r_2 = |X|_{2^n-1} = |119|_7 = |(01110111)_2|_7 =$$
$$= |(01)_2 + (110)_2 + (111)_2|_7 = |1+6+7|_7 = 0$$

$$r_3 = |X|_{2^n+1} = |119|_9 = |(01110111)_2|_9 =$$
$$= |(01)_2 - (110)_2 + (111)_2|_9 = |1-6+7|_9 = 2$$

Dessa forma, a representação residual de 119 para o conjunto de módulos $(8,7,9)$ é $(7,0,2)$.

Existem vários trabalhos que propõem conversores diretos eficientes [23, 24, 25, 26], entretanto, as arquiteturas não se diferenciam muito da estrutura genérica apresentada na Figura 2.3. A pré-computação é uma etapa sem *hardware* que se encarrega de obter os produtos parciais $\left|2^i\right|_m x_i$, $0 \leq i \leq j-1$. Depois os "g" produtos parciais são computados por somadores multi-operando, como árvores de somadores *CSA* (*Carry-Save Adders*) do módulo correspondente [23] ou compressores modulares, executando a operação $\left|\sum_{i=0}^{j-1} \left|2^i\right|_m x_i\right|_m$ e obtendo dois vetores C_o e S_o, os quais são somados modularmente em um conversor final (FC - *Final Converter*). O FC é composto por somadores completos que executam uma soma modular e é detalhado na próxima seção.

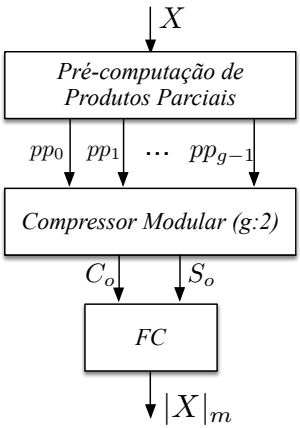


Figura 2.3: *Arquitetura genérica de conversor binário para RNS.*

2.4 Unidades Aritméticas Modulares

2.4.1 Somadores

Para implementação dos conversores reversos e unidades aritméticas de soma modular eficientes é utilizado basicamente duas classes de somadores: *i*) Os somadores multi-operandos que não realizam a soma completa, evitando a propagação de *carry* e gerando ao final dois vetores, *carry* (C_o) e soma (S_o); *ii*) Os somadores de dois operandos que é o estágio final da soma, conhecido como conversor final (FC). Existem diversas propostas para essas duas classes, essa seção aborda rapidamente as estruturas utilizadas pelas propostas desta tese, ou seja, árvores *CSA* e compressores para somadores multi-operandos e conversor final paralelo para somadores de dois operandos.

É possível escolher várias topologias de árvore *CSA*. Por exemplo, a árvore de Wallace é conhecida pelo seu ótimo tempo de computação, ao adicionar vários operandos. É a topologia com menor atraso entre as topologias, mas com a maior complexidade de roteamento das linhas de conexão. Já a árvore balanceada tem o menor número de linhas de interconexão, mas tem maior atraso. A Figura 2.4 mostra um exemplo de árvore de Wallace com *CSA+EAC* para soma modular $\{2^n-1\}$ de 10 operandos.

Já os compressores utilizam contadores paralelos que reduzem *bits* da mesma coluna. Um compressor maior pode ser gerados a partir de árvores de compressores menores e ainda tem uma estrutura mais regular com relação às árvores *CSA* porque os operandos são somados na forma de uma árvore binária. Conforme apresentado na Figura 2.5

A soma modular de dois operandos $|A+B|_m$ é calculada como $A+B-m$ se $A+B \geq m$ ou $A+B$ caso contrário. A estrutura paralela apresentada na Figura 2.6 processa as duas operações juntas e é usado um multiplexador $2:1$ para selecionar um dos resultados. O multiplexador é controlado pelo sinal da operação $A+B-m$.

A arquitetura geral pode ser especializada facilmente para $\{2^n\}$ simplesmente implementando um somador binário (*CPA - Carry-Propagate Adder*) e ignorando as saídas de *carry*. Já para o módulo $\{2^n+1\}$ pode ser usado um *CPA* com EAC (*End-Around Carry*). Para aumentar a eficiência, os *CPAs* são substituídos por somadores mais eficientes na computação dos *carrys*; por exemplo, o

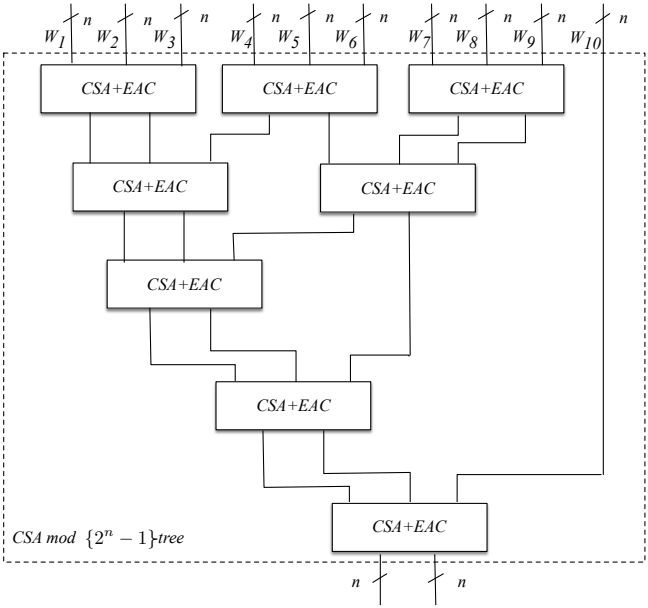


Figura 2.4: *Árvore* CSA *para soma modular* $\{2^n - 1\}$ *de* 10 *operandos.*

somardor de prefixo paralelo KSA (*Kogge-Stone Adder*).

Para o módulo $\{2^n + 1\}$ as arquiteturas são mais complexas porque são necessários circuitos adicionais para tratar os *bits* com peso $(n+1)$. Entretanto, o emprego do sistema numérico *diminished-one* é usado para evitar que o circuito de soma precise tratar esse bit adicional. O bit adicional somente é "1" para representar o valor 0, o que é facilmente alcançando subtraindo em 1 o valor binário normal. A vantagem desta representação é que o valor zero, para o qual todas as operações aritméticas são inibidas, é identificado de forma exclusiva com o valor 1 no *bit* mais significativo (MSB - *Most Significant Bit*). Por isso, para reduzir a complexidade e melhorar a eficiência, o sistema numérico *diminished-one* é usado juntamente com o *IEAC* (*Inverte End-Around Carry*) no projeto de somadores de módulo$\{2^n + 1\}$.

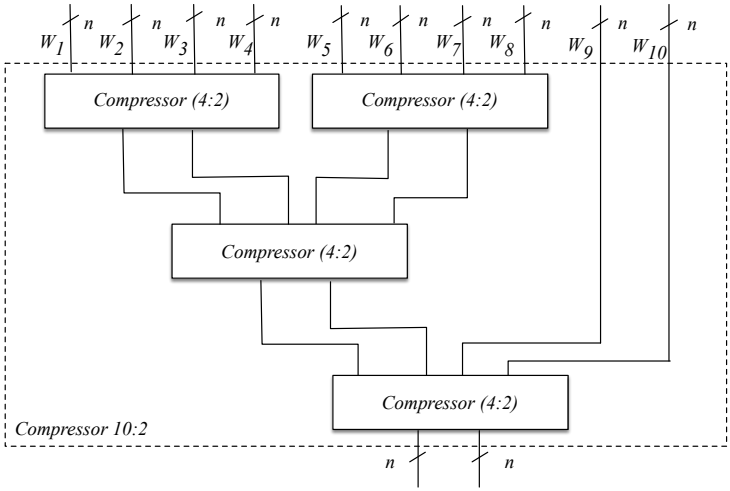


Figura 2.5: *Compressor* $(10:2)$*, a partir de compressores* $(4:2)$*, para* 10 *operandos.*

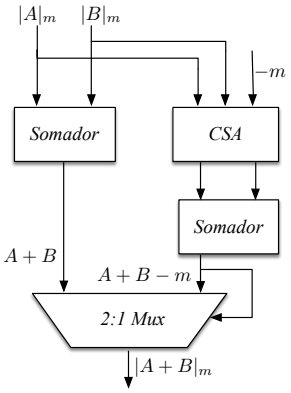


Figura 2.6: *Estrutura Paralela do Conversor Final (FC)*

2.4.2 Multiplicadores

2.4.2.1 Multiplicação $\{2^n \pm 1\}$

A multiplicação modular pode ser implementada colocando um circuito de redução modular na saída de um multiplicador binário comum. Entretanto, projetos mais eficientes são possíveis se a redução modular é combinada com a acumulação dos produtos parciais utilizando árvores multiníveis de *CSA* e uma soma final executada por *CPA* com *EAC*.

A Figura 2.7 mostra a estrutura da soma para um multiplicador módulo $\{2^n - 1\}$ com $n = 8$ comparada com a implementação binária. Os pesos maiores do que 2^n (lado esquerdo da linha vertical laranja na multiplicação binária) são divididos por 2^n, ou seja, os *bits carry* de saída com peso $\{2^{2n-2}, ..., 2^n\}$ são inseridos na mesma linha com pesos $\{|2^{2n-2}|_{2^n-1}, ..., |2^n|_{2^n-1}\} = \{2^{n-1}, ..., 2^0\}$. Portanto, a árvore *CSA* resultante é da forma $n \times n$ (comprimento $\times$ profundidade).

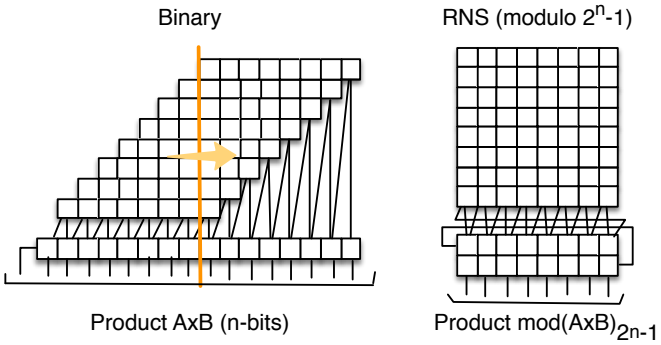


Figura 2.7: *Multiplicação modular* $\{2^n - 1\}$ *com* $n = 8$ vs. *multiplicação binária. Fonte: [27]*

O produto do multiplicador $\{2^n + 1\}$ é calculado seguindo o mesmo método. As principais diferenças são que os *bits* com peso $\{2^{2n-1}, ..., 2^n\}$ são complementados e inseridos na próxima linha com pesos $\{|2^{2n-1}|_{2^n+1}, ..., |2^n|_{2^n+1}\} = \{-2^{n-1}, ..., -2^0\}$, indicados como valores complementados pelos quadrados es-

curecidos na Figura 2.8. Dessa forma, um estágio adicional é necessário na árvore *CSA* para soma do produto parcial complementado. Finalmente é adicionado um termo corretor (COR) para ajustar os *bits* complementados inseridos no ajuste de peso. A soma final também é executada por *CPA* com *IEAC* de módulo equivalente.

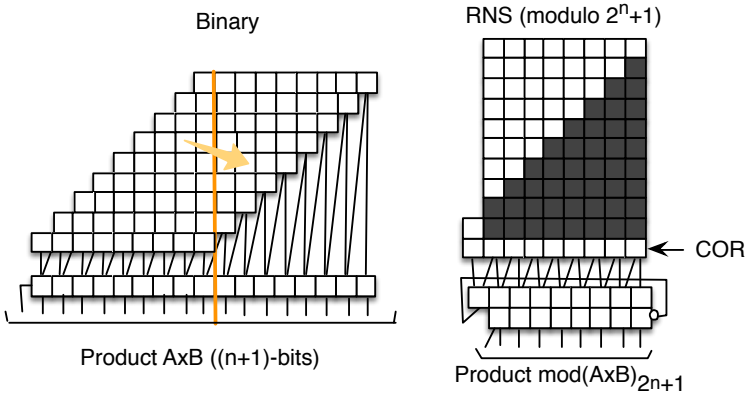


Figura 2.8: *Multiplicação modular* $\{2^n+1\}$ *com* $n=8$ vs. *multiplicação binária.*
Fonte: [27]

Essa técnica gera um multiplicador modular bastante otimizado para duas variáveis. Por exemplo, para o multiplicador módulo $\{2^n-1\}$ com $n=8$ é necessária uma árvore de quatro níveis empregando seis módulos *CSA* ao todo. Entretanto, se um dos operandos é conhecido e constante, os produtos parciais podem ser compactados de forma a garantir uma diminuição na altura da árvore *CSA* e, consequentemente, no atraso causado pela operação.

2.4.2.2 Multiplicação $\{2^n \pm k\}$

A multiplicação modular $2^n \pm k$ também pode ser executada com a reinjeção dos *bits* para diminuir a complexidade da soma dos produtos parciais. O cálculo dos pesos de cada coluna para os módulos 2^n-k é dado por $W_i = |2^i|_{2^n+k}$, com $n \leq i < 2n$. A Figura 2.9 apresenta o processo para $n=8$ e $k=-3$. É pos-

sível verificar os pesos calculados na primeira linha. Cada coluna à esquerda da linha vertical laranja tem seu peso decomposto em pesos equivalentes das colunas da direita e os *bits* equivalentes são reinseridos nos produtos parciais; por exemplo, os *bits* da coluna 3 são reinseridos na coluna com peso 2 e 1. Os quadrados que representam os *bits* com mesmo peso possuem o mesmo preenchimento para facilitar a compreensão do padrão de reinserção.

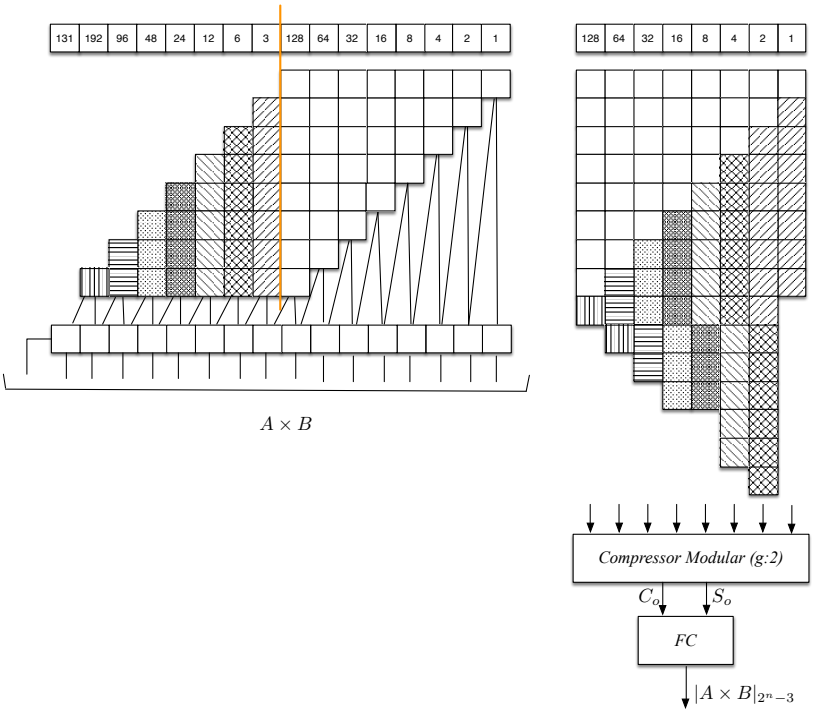


Figura 2.9: *Multiplicação modular* $\{2^n - 3\}$ *com* $n = 8$.

Para a multiplicação modular $2^n + k$, o cálculo dos pesos é mais complicado e segue a Eq. (2.8). A Figura 2.10 apresenta o processo para $n = 8$ e $k = -3$. É possível perceber que existem colunas com pesos negativos. Os *bits* dessas colunas precisam ser complementados, representados em vermelho, e reinseridos à direita. Além disso, é preciso calcular um fator corretor (*COR*) para equalizar as

inversões dos *bits*.

$$W_i = \begin{cases} |2^i|_{2^n+k} - 2^n + k, & |2^i|_{2^n+k} > \frac{2^n+k-1}{2} \\ |2^i|_{2^n+k}, & |2^i|_{2^n+k} \leq \frac{2^n+k-1}{2} \end{cases}, n \leq i \leq 2n \tag{2.8}$$

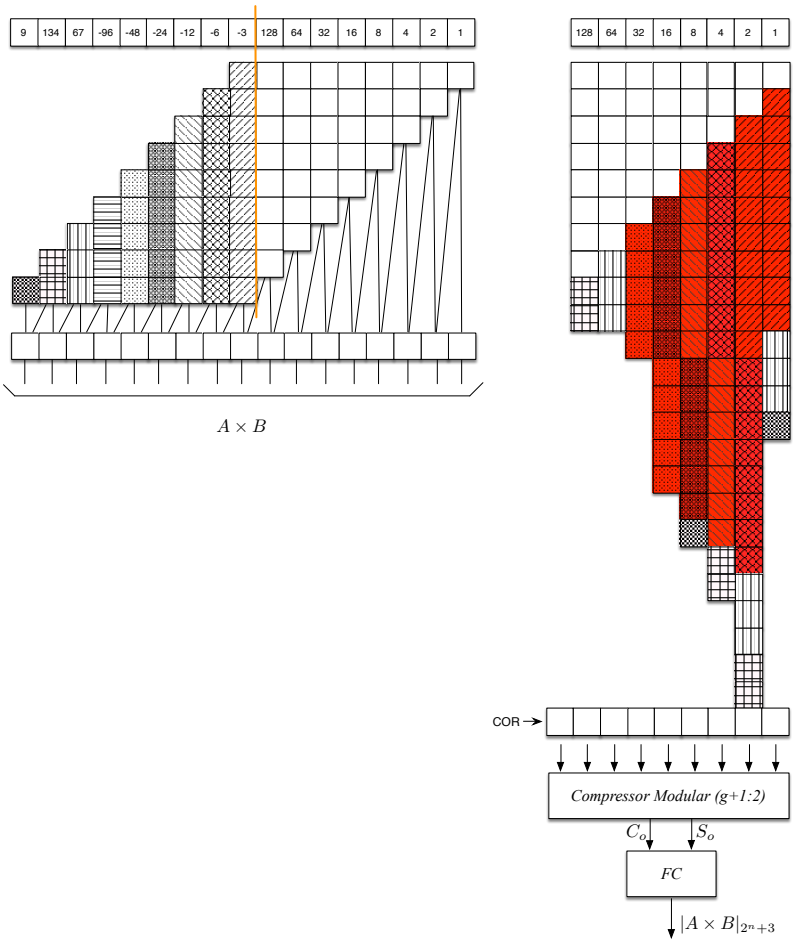


Figura 2.10: *Multiplicação modular* $\{2^n - 3\}$ *com* $n = 8$.

EAC

2.5 Conversores Reversos

O conversor reverso transforma os vários resíduos calculados pelos canais paralelos em um número binário equivalente. Para tal, vamos analisar os três algoritmos mais empregados na etapa de conversão CRT [1], MRC [3] e o novo CRT-I [4]. Por exemplo, considerando uma representação RNS baseado no conjunto de módulos $(m_1, m_2, ..., m_n)$, um valor representado $(R_1, R_2, ..., R_n)$ pode ser convertido para binário (X) das seguintes forma:

A. *Usando o CRT*:

$$X = \left| \sum_{i=1}^{n} \hat{m}_i \left| \hat{m}_i^{-1} \right|_{m_i} \times R_i \right|_M . \tag{2.9}$$

A faixa dinâmica é igual ao produto de todos os módulos do conjunto $(M = \prod_{i=1}^{n} m_i)$, $\hat{m}_i = M/m_i$, e $\left| \hat{m}_i^{-1} \right|_{m_i}$ representa a multiplicativa inversa do $\hat{m}_i$ com o respectivo módulo m_i.

B. *Usando o MRC*:

$$X = v_n \prod_{i=1}^{n-1} m_i + ... + v_3 m_2 m_1 + v_2 m_1 + v_1 \tag{2.10}$$

Os coeficientes $v_1, v_2, ..., v_n$ podem ser obtidos dos resíduos por:

$$\begin{aligned} v_n = \Big| \big(\big((R_n - v_1) \times |m_1^{-1}|_{m_n} - v_2 \big) \times \\ \times |m_2^{-1}|_{m_n} - ... - v_{n-1} \big) \times |m_{n-1}^{-1}|_{m_n} \Big|_{m_n} \end{aligned} \tag{2.11}$$

C. *Usando o Novo CRT-I [4]*:

$$X = \left| \sum_{i=1}^{n} |V_i R_i|_{\hat{m}_1} \right|_{\hat{m}_1} m_1 + R_1 \tag{2.12}$$

$$V_1 = \frac{\hat{m}_1 |\hat{m}_1^{-1}|_{m_1} - 1}{m_1} \tag{2.13}$$

$$V_i = \left|\hat{m}_i^{-1}\right|_{m_i} \frac{\hat{m}_i}{m_1}, \qquad 2 \leq i \leq n \tag{2.14}$$

Abaixo temos um exemplo numérico onde o valor representado em RNS (conjunto de módulos $m_1, m_2, m_3 = 5, 8, 11$) como $(R_1, R_2, R_3) = (3, 0, 4)$ é convertido com as Equações 2.9, 2.10 e 2.12, respectivamente.

CRT:

$$\begin{aligned}
&M = 5 \times 8 \times 11 = 440 \\
&\hat{m}_1 = 440/5 = 88 \quad \rightarrow \quad |\hat{m}_1^{-1}|_5 = 2 \\
&\hat{m}_2 = 440/8 = 55 \quad \rightarrow \quad |\hat{m}_2^{-1}|_8 = 7 \\
&\hat{m}_3 = 440/11 = 40 \quad \rightarrow \quad |\hat{m}_3^{-1}|_{11} = 8 \\
&X = |88 \times 2 \times 3 + 55 \times 7 \times 0 + 40 \times 8 \times 4|_{440} = 48
\end{aligned}$$

MRC:

$$\begin{aligned}
&v_1 = R_1 = 3 \\
&v_2 = |(0-3) \times 5|_8 = 1 \\
&v_3 = |((4-3) \times 9 - 1) \times 7|_{11} = 1 \\
&X = 1 \times 5 \times 8 + 1 \times 5 + 3 = 48
\end{aligned}$$

Novo CRT-I:

$$\begin{aligned}
V_1 &= \frac{\hat{m}_1 |\hat{m}_1^{-1}|_{m_1} - 1}{m_1} = \frac{88 \times 2 - 1}{5} = 35 \\
V_2 &= \left|\hat{m}_2^{-1}\right|_{m_2} \frac{\hat{m}_2}{m_1} = 7 \times \frac{55}{5} = 77 \\
V_3 &= \left|\hat{m}_3^{-1}\right|_{m_3} \frac{\hat{m}_3}{m_1} = 8 \times \frac{40}{5} = 64 \\
X &= |35 \times 3 + 77 \times 0 + 64 \times 4|_{88} \times 5 + 3 = 48
\end{aligned}$$

■

Apesar do algoritmo MRC ser bastante difundido e usado na área de conversores reversos, ele não apresenta boa eficiência para conjuntos modulares focados na obtenção de grandes faixas dinâmicas e aumento da paralelização devido à sua natureza recursiva e serial [10]. Baseados no CRT, ainda existe os algoritmos Novo CRT II e o CRTf, que são detalhados na Seção 4.2.

Para criação de fato do conversor reverso, é necessário levar em consideração as equações de conversão dos algoritmos, após serem simplificadas com a substituição dos valores do conjunto de módulos escolhido, e utilizar componentes de *hardware*, como somadores, multiplicadores, entre outros. Na Figura 2.11 são apresentadas arquiteturas genéricas para implementação do algoritmos CRT e o Novo CRT-I. A fase de pré-computação é executada em tempo de projeto e tem como objetivo gerar o menor número de produtos parciais da multiplicação modular das constantes geradas pelos algoritmos pelas variáveis (R_i). A quantidade de produtos parciais é o maior impacto na eficiência dos conversores, uma vez que eles determinam a profundidade da árvore *CSA* ou do compressor que aparece como a segunda etapa. Depois disso, o conversor final faz a soma com propagação de *carry* e gera o valor binário X para o CRT. Enquanto que para completar o Novo CRT-I é necessária a concatenação com R_1, denotada pelo símbolo $*$, conforme em [10, 22], sem custo em área ou atraso.

2.5.1 Estado da Arte de Conversores Reversos

Este estado da arte se refere aos conversores reversos em função dos conjuntos modulares, uma vez que as estruturas reversas de conversão geralmente são apresentadas sempre que um novo conjunto de módulos é proposto.

Como comentado anteriormente, é possível usar extensões horizontais para adicionar mais módulos ao conjunto, enquanto as extensões verticais dos canais são usadas para aumentar o *DR*. Esta abordagem tem sido considerada e proposta no estado da arte, como os conjuntos de quatro módulos com um *DR* de cerca de $4n$-bit: $\{2^n, 2^n \pm 1, \mathbf{2^{n+1} + 1}\}$ e $\{2^n, 2^n \pm 1, \mathbf{2^{n+1} - 1}\}$ [13], [14], $\{2^n, 2^n \pm 1, \mathbf{2^{n-1} + 1}\}$ e $\{2^n, 2^n \pm 1, \mathbf{2^{n-1} - 1}\}$ [15]. As extensões horizontais de conjuntos de cinco módulos com um *DR* de aproximadamente de $5n$ *bits* também foram propostas: $\{2^n, 2^n \pm 1, \mathbf{2^{n\pm1} - 1}\}$ [18], $\{2^{n+1}, 2^n \pm 1, \mathbf{2^{n+1} \pm 1}\}$ [19], e $\{2^n, 2^n \pm 1, \mathbf{2^n \pm 2^{\frac{(n+1)}{2}} + 1}\}$ [20] que é composto por módulos co-primos para n ímpar e foi revisitado pela Hiasat em [22]. Os módulos considerados

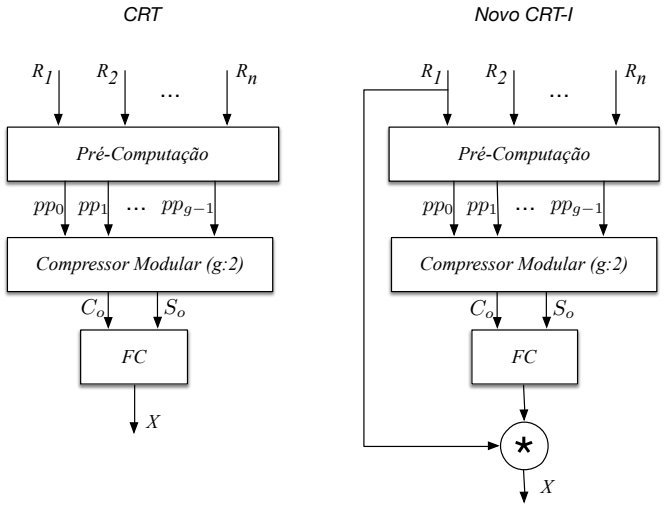


Figura 2.11: *Diagrama de blocos genéricos dos conversores reversos baseados no CRT e Novo CRT-I.*

em [19] são números co-primos por n iguais, no entanto, são necessárias multiplicativas inversas complexas, resultando em estruturas de conversão reversa dispendiosas. Em [28], os autores propuseram um conjunto de oito módulos $\{2^{n-5}-1, 2^{n-3}-1, 2^{n-3}+1, 2^{n-2}+1, 2^{n-1}-1, 2^{n-1}+1, 2^n, 2^n+1\}$. O conjunto de módulo proposto não é regular, apresentando canais com n até $n-5$ *bits* com módulos não co-primos, resultando em um baixo *DR*. Como em [19], multiplicativa inversas complexas são necessárias, resultando em estruturas de conversão inversa dispendiosas e hierarquicamente complicadas. Adicionalmente, extensões verticais têm sido propostas, como em [7] ($\{\mathbf{2}^{n+\beta}, 2^n \pm 1\}$, $0 \leq \beta \leq n$), no qual a variação de β é usado para aumentar o *DR* até 4*n-bits* com um conjunto de três módulos. Isso é alcançado em direção a um conjunto de módulos mais balanceado, uma vez que há diferença de desempenho entre as unidades 2^n e as unidades aritméticas $2^n \pm k$. Portanto, a sobrecarga do canal $\{2^n\}$ até $\{2^{2n}\}$ pode ser feita sem afetar o atraso nos canais aritméticos.

Recentemente, foram propostos conjunto de módulos com ambas as extensões: $\{\mathbf{2}^{2n}, 2^n \pm 1, \mathbf{2}^{2n+1} \mathbf{-1}\}$ [16], $\{\mathbf{2}^{2n}, 2^n \pm 1, \mathbf{2}^{2n} \mathbf{+1}\}$ [17], e $\{\mathbf{2}^{n+\beta}, 2^n \pm$

$1, \mathbf{2^n \pm 2^{\frac{n+1}{2}} + 1}, \mathbf{2^{n+1} + 1}\}$ e)$\{\mathbf{2^{n+\beta}}, 2^n \pm 1, \mathbf{2^n \pm 2^{\frac{n+1}{2}} + 1}, \mathbf{2^{n-1} + 1}\}$ com $-\frac{(n-1)}{2} \leq \beta \leq 3n$ [21]. As propostas [16] e [17] fornecem um *DR* de $\simeq 6n$*-bits*. A proposta [21] fornece um conjunto de módulos mais balanceado com um *DR* de $(8n+1)$ *bits*.

O artigo [9] apresentou um método baseado no Novo CRT-I para projetar conversores reversos que usam conjunto de módulos estendidos, híbridos e genéricos na forma de $\{\mathbf{2^{n+\beta}}, 2^n \pm 1, \mathbf{2^n \pm k_1}, \mathbf{2^n \pm k_2}, ..., \mathbf{2^n \pm k_f}\}$, com k_j sendo ímpar e $0 \leq \beta \leq n$. No entanto, esse método impõe uma seleção de peso modular complexa dos termos multiplicativos associada às entradas dos resíduos, que é uma desvantagem substancial. Além disso, a adição modular dessas entradas ponderadas requer um grande número de comparações e, consequentemente, um circuito dedicado é usado na arquitetura para reduzir a complexidade do passo de conversão final.

Por outro lado, os conversores reversos baseados no Teorema Chinês do Resto Aproximado (CRTf - *Chinese Remainder Theorem with Fractional Values*) apresentados por [12] oferecem melhorias em relação à implementação clássica do CRT. Essa abordagem propõe um novo formato para o CRTf, que permite uma conversão reversa sem erros. Comparado com o CRT clássico, que utiliza operações modulares mais complexas, o CRTf permite uma melhor implementação de *hardware* porque ele usa operações modulares mais simples. O gargalo dessa abordagem são as multiplicações por constantes, necessárias no processo de conversão.

Apesar das diferentes arquitetura apresentadas, os conversores reversos podem ser generalizados conforme a Figura 2.11 e analisadas com relação ao número de produtos parciais (g) na saída da pré-computação, como exemplo, na proposta de [7] temos $g = 4$, somados por dois *CSA* e um somador completo, ambos módulo $2^{2n} - 1$. Já [18], [22], [21], [9] e [17] propõem uma solução com $g = 5$, $g = 8$, $g = 10$, $g = 9$ e $g = 5$, respectivamente.

2.6 Conclusão

Esse capítulo apresentou as principais propostas no estado da arte para conjuntos de módulos e conversores. Existem diversas abordagens, mas o eixo central das soluções propostas baseadas no CRT, no Novo CRT-I e no CRTf bus-

cam conjuntos balanceados que permitam operações modulares mais simples no conversor reverso com a finalidade de diminuir a complexidade dessa etapa. Ao final, tudo indica que, apesar das diversas abordagens discutidas, a maior influencia está no número de produtos parciais (g). Existe um forte indício que, além da proposta de novos conjuntos balanceados, implementações focadas na otimização dos produtos parciais ("Pré-computação") e, consequentemente, na diminuição da complexidade da etapa de "compressão modular (g:2)", trazem avanço para área de conversão de RNS para binário.

CAPÍTULO 3

Otimização de Multiplicadores Modulares por Constante

No Capítulo 2, foram apresentados os algoritmos mais empregados na conversão da representação RNS para binário. Analisando as equações desses algoritmos, é possível perceber a repetição sistemática do multiplicador modular por constante na Eq. (2.9) do algoritmo CRT $\left| \overbrace{\hat{m}_i \left| \hat{m}_i^{-1} \right|_{m_i}}^{Constante} \times R_i \right|_M$ e na Eq. (2.12) do Novo CRT-I $\left| \overbrace{V_i}^{Constante} \times R_i \right|_{\hat{m}_1}$. Além disso, no mesmo capítulo foi apresentada a arquitetura de hardware genérica para implementação dos conversores reversos (Figura 2.11), baseados no CRT e Novo CRT-I. Destacou-se a etapa de compressão modular, que possui grande influência na eficiência da implementações dos conversores. Basicamente, essa etapa executa a soma dos produtos parciais gerados da multiplicação das várias constantes, dadas pelas equações dos algoritmos, com as variáveis, valores dos resíduos, para executar a conversão. Ao final da análise do estado da arte dos conversores reversos, percebe-se que, além da proposta de novos conjuntos de módulos, os trabalhos da área sempre visam diminuir a quantidade de produtos parciais, que entram no compressor, com as mais diversas estratégias.

Dessa forma, o emprego de técnicas de otimização focadas em diminuir o número de produtos parciais para determinadas constantes extraídas do conjunto de módulos implicará em uma implementação de conversor reverso mais eficiente e podendo ser aplicada em diversas proposta da área. Nesse capítulo será apresentado um método, publicado em parte em [11], que propõe e organiza algumas técnicas de otimização de multiplicação modular por constante focada nos conversores reversos baseados no CRT e no Novo CRT-I, os quais tem como características a acumulação modular dos resultados de várias multiplicações de grandes constantes por variáveis. Depois é apresentado um segundo método de otimização lógica focado na multiplicação modular na forma $\{2^n \pm k\}$ de uma constante por uma variável. Ao final, as técnicas são aplicadas a uma unidade aritmética modular variando o módulo e a largura de *bits*.

3.1 Método de Otimização para Conversores Reversos

Como visto na seção anterior, as técnicas mais comuns de conversão reversa dependem de multiplicação modular por constante. Em geral, são feitas várias multiplicações de pequenas variáveis com grandes constantes e depois os produtos resultantes são todos somados. Diminuir a complexidade dessa soma é essencial para ter conversores cada vez mais rápidos. Dessa forma, para aproveitar essa particularidade dos algoritmos de conversão reversa, foi criado um método que em três etapas reune diversas técnicas de otimização lógica e usa a melhor combinação para uma determinada constante.

Para ilustrar as técnicas utilizadas, considere a soma dos produtos gerados entre as constantes 32767 $(111111111111111)_2$, 249 $(11111001)_2$, 193 $(11000001)_2$ e as variáveis P $(p_3p_2p_1p_0)_2$, Q $(q_1q_0)_2$ e R $(r_2r_1r_0)_2$. A operação modular implementada é dada por $|32767 \times p_3p_2p_1p_0 + 249 \times q_1q_0 + 193 \times r_1r_0|_{255}$. É importante notar que o estudo de caso escolhido usa modulo $2^n - 1$ ($n = 8$), entretanto essas técnicas podem ser estendidas para qualquer módulo ($2^n + 1$, $2^n \pm k$ ou 2^n).

3.1.1 Pré-computação das Constantes

A primeira etapa é a pré-computação das constantes. Nessa etapa a constante é analisada e pode ser modificada para gerar menos produtos parciais. São usados três métodos conforme apresentado na Figura 3.1: complemento, recodificação e a combinação das duas. Cada método é testado para verificar qual gera o menor número de produtos parciais.

Uma técnica simples, embora muito útil para reduzir o número de produtos parciais, consiste em complementar o valor de uma determinada constante baseado no módulo usado. Por exemplo, $|32767 \times P|_{255} = |127 \times P|_{255} = |-128 \times P|_{255}$. Para tratar o valor negativo, é aplicado o complemento de um na variável P ($\overline{P}$) e aplicado um fator corretor adicional (COR_P). A expressão resultante é $|128 \times \overline{P} + COR_P|$, onde COR_P é o menor valor que satisfaz: $|2^7 \times \overline{P} + COR_P|_{255} = 0$, quando $P = 0$. Dessa forma, a solução é $COR_P = 120$. O valor de COR_P é escolhido para se ajustar exatamente na matriz, como apresentado na Figura 3.1.

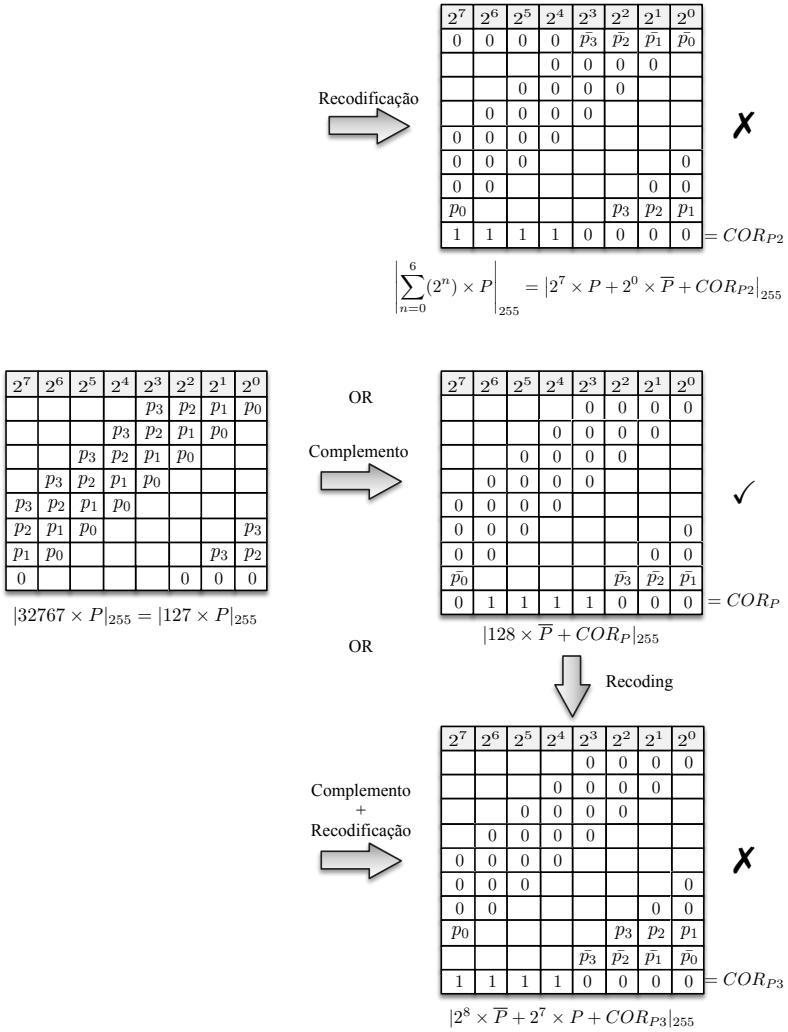


Figura 3.1: *Representação gráfica da etapa de pré-computação.*

Outra técnica poderosa é a recodificação de "uns" sucessivos na representação binária de uma constante. Por exemplo, $|127 \times P|_{255}$ pode ser representado como $|\sum_{n=0}^{6}(2^n) \times P|_{255}$, o qual é equivalente a $|(2^7 - 2^0) \times P|_{255}$. O termo negativo também pode ser alcançado complementando a variável e adicionando um fator corretor ($COR_{P2} = 240$) da mesma forma que apresentado anteriormente. Resultando em $|128 \times \overline{P} + COR_{P2}|_{255}$. Nessa técnica, cada bloco de "uns" gera dois produtos parciais e um fator corretor. O algoritmo da pré-computação acumula todos os corretores em apenas um produto parcial. Assim, quando há mais de dois blocos de pelo menos três "uns" em uma dada constante, a constante recodificada gerará menos produtos parciais que a original.

A última técnica testada é a recodificação do complemento. Essa otimização traz resultados positivos quando a constante possui "uns" esparsos e blocos de "zeros". Após o complemento, esses valores são invertidos criando blocos de "uns" que podem ser otimizados usando a recodificação. No exemplo apresentado após o complemento temos: $|-128 \times P|_{255}$. A recodificação é aplicado à constante 128 $(10000000)_2$, gerando o seguinte $|-(2^8 - 2^7) \times P|_{255} = |-2^8 \times P + 2^7 \times P|_{255}$. Finalmente, seguindo os passos discutidos anteriormente para eliminar o termo negativo, temos: $|2^8 \times \overline{P} + 2^7 \times P + COR_{P3}|_{255}$, onde $COR_{P3} = 240$.

Para $|32767 \times P|_{255}$ (Figura 3.1), o algoritmo escolherá a técnica do complemento porque gera dois produtos parciais ao invés de três das outras técnicas discutidas. Para completar o exemplo proposto anteriormente, as constantes 249 e 193 são submetidas ao método de pré-computação e os métodos que geram o menor número de produtos parciais são os métodos complemento e o complemento mais recodificação, respectivamente. O resultado pode ser visto nas Figuras 3.2 e 3.3.

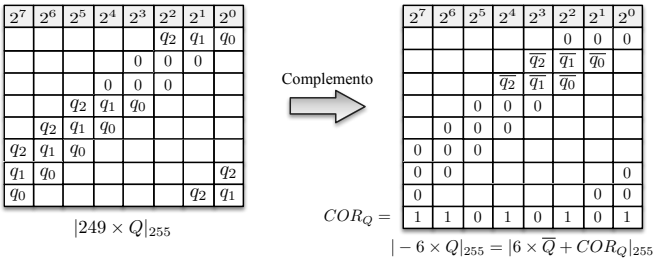


Figura 3.2: *Representação gráfica do resultado da pré-computação para* $|249 \times Q|_{255}$ *complementado.*

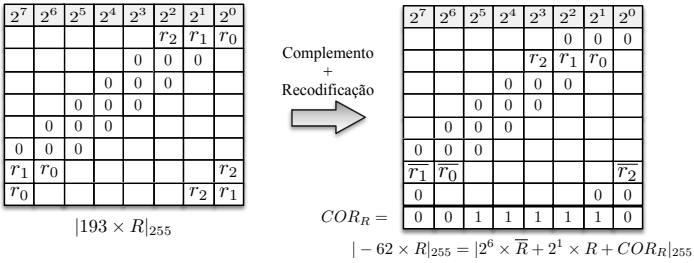


Figura 3.3: *Representação gráfica do resultado da pré-computação para* $|193 \times R|_{255}$ *complementado e recodificado.*

3.1.2 Compactação Vertical

Esta etapa tem o objetivo de eliminar toda a informação inútil nos produtos parciais. Ela é usada em vários pontos do processo e reune três técnicas simples: i) Compactação dos fatores corretores (*CORs*); ii) Compactação vertical; e iii) Eliminação.

A primeira técnica consiste em aproveitar a característica da soma modular e unificar todos os *CORs* no menor número de produtos parciais possíveis. No nosso exemplo, conseguimos resumir os *CORs*, gerados a partir da aplicação das técnicas de pré-computação, em apenas um produto parcial somando todos os valores e aplicando o módulo correspondente. O resultado seria: $|COR_P + COR_Q + COR_R|_{255} = |120 + 213 + 62|_{255} = 140 = 10001100_2$. A segunda técnica visa eliminar as lacunas (zeros) existentes entre os produtos parciais. A idéia é eliminar qualquer constante sem valor, diminuindo consideravelmente a quantidade de produtos parciais e complexidade do somador. Não é necessária nenhuma adaptação, uma vez que a mudança vertical do *bit* não muda o seu peso. A última técnica é a eliminação dos produtos parciais sem informação.

Apesar dessa etapa reunir técnicas simples, elas tem um impacto considerável na acumulação de produtos modulares. No exemplo proposto anteriormente, ela reduz de 27 para 5 produtos parciais, o que pode ser visto graficamente na Figura 3.4.

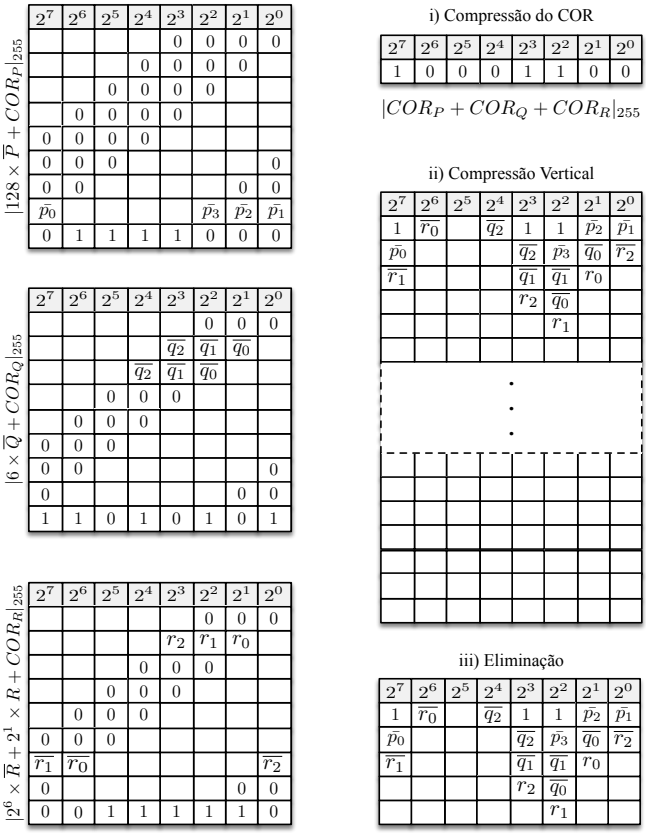
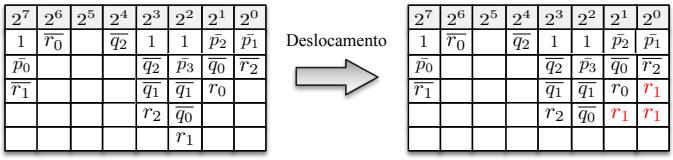


Figura 3.4: *As três técnicas da etapa de compactação vertical para o exemplo proposto.*

3.1.3 Deslocamento Horizontal

O último passo é o computacionalmente mais complexo dentre as três etapas. Além da compactação vertical, ainda é possível mudar os *bits* horizontalmente, adicionando novos *bits* com pesos equivalentes nas lacunas. Por exemplo, o último produto parcial depois da compactação vertical consiste em apenas um *bit* relevante (r_1). No entanto, exigiria um somador extra antes da adição

final. Dessa forma, é desejável mover o *bit* r_1 para lacunas em outros produtos parciais aproveitando ao máximo os somadores instanciados. Mesmo que isso signifique replicar a mesma informação em diversas posições para manter o peso original do *bit* deslocado. A Figura 3.5 apresenta a operação: $r_1 \times 2^2 = r_1 \times 2^1 + r_1 \times 2^0 + r_1 \times 2^0$.



$$r_1 \times 2^2 = r_1 \times 2^1 + r_1 \times 2^0 + r_1 \times 2^0$$

Figura 3.5: *Etapa de Deslocamento Horizontal.*

3.2 Otimização lógica para multiplicação modular $\{2^n \pm k\}$ por constante.

O método desta seção pode ser utilizado em conjunto com o método apresentado na seção anterior. Ele deve ser executado depois da fase de pré-computação e aceita apenas a otimização de uma variável por vez. Em geral, os módulos nas formas $2^n \pm 1$ e 2^n não se beneficiam desse método devido à característica intrínseca da reinjeção de *bits*, apresentada na Seção 2.4.2, ou seja, são poucos os sinais iguais e/ou complementados na mesma coluna para esses módulos.

Por exemplo, vamos considerar uma multiplicação modular $2^n - k$, sendo $n = 8$ e $k = 3$, da constante 222 por uma variável de 8 *bits* (x). Isso gera os produtos parciais mostrados na Figura 3.6. Devido à natureza modular da multiplicação, vários sinais são reinjetados e algumas colunas podem ser reduzidas usando técnicas simples de otimização lógica. Por exemplo, a coluna com peso 2^3 contém duas vezes x_4 e x_7 e uma vez $\overline{x_7}$. Nesse caso, seria possível deslocar o x_4 para esquerda, pois $(x_4 \times 2^3) \times 2 = x_4 \times 2^4$. Já para o sinal x_7 poderia ser realizada a mesma operação ou, ainda melhor, aproveitar o seu complemento para substituir dois sinais pelo valor "um" que, posteriormente, pode ser comprimido com outras constates. Sendo assim, temos: $((x_7 + x_7 + \overline{x_7}) \times 2^3 = (x_7 + 1) \times 2^3$.

'x(6)'	'x(5)'	'x(4)'	'x(3)'	'x(2)'	'x(1)'	'x(0)'	'x(7)'
'x(5)'	'x(4)'	'x(3)'	'x(2)'	'x(1)'	'x(0)'	'x(7)'	'x(6)'
'x(4)'	'x(3)'	'x(2)'	'x(1)'	'x(0)'	'x(7)'	'x(6)'	'x(5)'
'x(3)'	'x(2)'	'x(1)'	'x(0)'	'x(7)'	'x(6)'	'x(5)'	'x(4)'
'x(1)'	'x(0)'	'x(6)'	'x(7)'	'x(6)'	'x(5)'	'x(4)'	'x(2)'
'x(0)'	'x(7)'	'x(5)'	'x(5)'	'x(4)'	'x(3)'	'x(2)'	'x(1)'
[1]	'x(6)'	'x(7)'	'x(4)'	'x(3)'	'x(2)'	'x(1)'	'not x(7)'
[0]	[1]	'x(6)'	'x(6)'	'x(7)'	'x(7)'	'x(7)'	[0]
[0]	[0]	'not x(7)'	'x(5)'	'x(5)'	'x(6)'	'x(6)'	[0]
[0]	[0]	[0]	'not x(7)'	'x(4)'	'x(4)'	'x(5)'	[0]
[0]	[0]	[0]	[0]	'not x(7)'	'x(3)'	'x(3)'	[0]
[0]	[0]	[0]	[0]	[0]	'not x(7)'	'x(2)'	[0]

Figura 3.6: *Produtos Parciais da operação $|222 \times x|_{253}$ antes da otimização.*

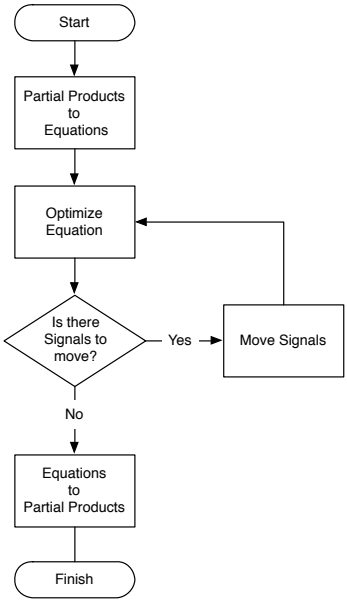


Figura 3.7: *Algoritmo da etapa de otimização lógica.*

Para automatizar essa etapa, foi criado um algoritmo (Figura 3.7) que converte cada coluna em uma equação, a simplifica, move os sinais duplicados

para a coluna à esquerda e, ao encontrar a condição de parada, converte as equações em produtos parciais novamente. A primeira etapa é converter cada coluna em uma equação. Por exemplo, a coluna com peso 2^3 é convertida em $col_{2^3} = x_2 + x_1 + x_0 + x_7 + x_6 + x_4 + x_3 + x_7 + x_5 + x_4 + 1 - x_7$. Os valores complementados como $\overline{x_7}$ são representados como $1 - x_7$. O processo de otimização retorna a equação já simplificada ($col_{2^3} = x_0 + x_1 + x_2 + x_3 + 2 \times x_4 + x_5 + x_6 + x_7 + 1$), possibilitando que a próxima etapa, de análise da equação otimizada, verifique os sinais passíveis de deslocamento para uma coluna à esquerda (ex.: $2 \times x_4$). O deslocamento é feito de forma iterativa, ou seja, o sinal é deslocado apenas uma coluna por vez para compor a próxima equação e sofrer o mesmo processo novamente. Toda essas etapas se repetem enquanto for possível otimizar as equações. O resultado final da etapa de otimização do exemplo é mostrado na Figura 3.8.

```
'x(0)'   'x(0)'   'x(1)'   'x(0)'   'x(0)'   'x(0)'   'x(0)'   'x(1)'
'x(1)'   'x(7)'   'x(6)'   'x(1)'   'x(1)'   'x(1)'   'x(1)'   'x(2)'
'x(2)'   [   0]   [   0]   'x(5)'   'x(4)'   'x(3)'   'x(4)'   'x(3)'
[   0]   [   0]   [   0]   'x(7)'   'x(6)'   'x(5)'   [   0]   [   0]
```

Figura 3.8: *Produtos Parciais da operação* $|222 \times x|_{253}$ *depois da otimização.*

O exemplo mostra uma redução de $66,67\%$ no número de produtos parciais, o que implica em melhoria na potência, área e no atraso do circuito de soma modular. A redução de 12 para 4 produtos parciais permite implementar uma árvore de apenas *2* níveis com *2* blocos *CSA* ao invés de uma árvore de *5* níveis com *10* blocos *CSA*.

3.3 Resultados Experimentais

Para mostrar os aspectos práticos e as vantagens de ambos os métodos de otimização, foram implementadas variações de uma unidade aritmética modular bastante utilizada na área de processamento digital de sinais, o *MAC* (Multiplicador-ACumulador). Cada unidade foi descrita em VHDL e sintetizada. A ferramenta utilizada foi a *Synopsys Design Vision* (versão E-2010.12-SP4) para gerar um *netlist* otimizado mapeado para tecnologia *ASIC standard cells* de 90*n*m da empresa UMC [29]. Os parâmetros de compilação foram *"-exact_map -map_effort medium -area_effort medium -power_effort medium"*.

A arquitetura interna das unidades MAC é baseada em [30], exceto que a unidade é implementada sem os registradores de entrada e saída, ou seja, como um circuito combinacional puro. Foi implementada uma unidade para cada largura de *n bits* no intervalo $[8..65]$ e para cada módulo $2^n \pm k$, sendo $k = [0, 1, 3, 15, 21]$, ou seja, os módulos mais comuns na área dos conversores reversos. Além disso, consideramos a multiplicação por constantes arbitrárias uniformemente distribuídas com a mesma quantidade de informação, 50% de "uns" na representação binária, garantindo uma comparação justa, independentemente do número de *bits*.

A Figura 3.9a apresenta as unidades sem qualquer otimização e a Figura 3.9b apresenta as mesmas unidades com todas as técnicas de otimizações apresentadas nas Seções 3.1 e 3.2. Ambas as figuras possuem a mesma proporção e a mesma escala nos eixos para possibilitar uma comparação direta. É possível perceber nos dois casos que um número menor de *bits* é particularmente significativo para o atraso ao considerar operações aritméticas modulares em cada canal, o que indica que o projeto de conversores reversos com ampla faixa dinâmica deve seguir na direção do aumento do número de canais e, consequentemente, no aumento da paralelização e diminuição do número de *bits* por canal para tirar proveito de unidades aritméticas mais eficientes.

Com relação às técnicas de otimização, os resultados demonstram uma melhora para todos os módulos na média. Os módulos mais afetados pela otimização são os compostos por k com mais "uns" na representação, em sequência decrescente $\pm 15, \pm 21, \pm 3, \pm 1, \pm 0$. Tendo em média uma otimização de $28,21\%$, $22,44\%$, $12,66\%$, $6,95\%$ e $3,70\%$, respectivamente. Entre os módulos com k positivo e negativo os mais beneficiados pelas técnicas são os positivos (Figura 3.10), exceto para $k = \pm 3$, que registrou o mesmo ganho na otimização independente do sinal.

Para alguns casos, a unidade otimizada retornou um atraso maior que a unidade original. Isso acontece mais frequentemente para $k = 0$, porque para operações do módulo 2^n os *bits* são truncados, e para $k = \pm 1$, porque a reinjeção de *bits* é simétrica. Dessa forma, para esses três módulos, as técnicas de otimização da Seção 3.2 não são eficazes.

(a) *Sem Otimizar* **(b)** *Otimizados*

Figura 3.9: *Resultados experimentais para uma unidade modular MAC por constante, variando o módulo* $\{2^n \pm k\}$ *e a largura de* bits *(n).*

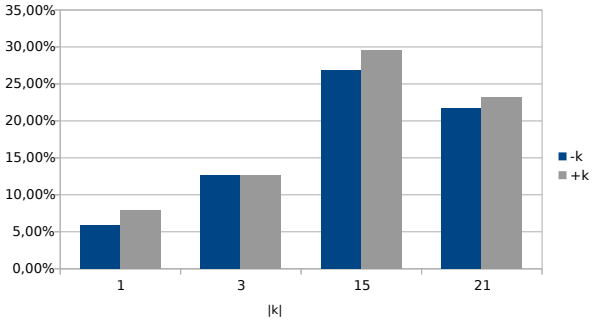


Figura 3.10: *Percentual de otimização para* $\pm k$.

Uma análise mais aprofundada pode ser feita para o pior ponto nos experimentos ($k = 0$, $n = 13$), no qual a unidade original é $21,15\%$ melhor. Nesse caso específico, para variável escolhida aleatoriamente (1101010101100_2), a aplicação das técnicas de otimização da Seção 3.1, que foram concebidas para otimizar as multiplicações multi-operando dos conversores reversos, acabam sendo prejudiciais, uma vez que não há diferença dos produtos parciais pré-computados ($6+COR$) e originais (7). Dessa forma, as portas inversoras e o fator corretor tem um impacto maior que a redução dos produtos parciais. Isso pode ser contornado de diversas formas na etapa de pré-computação quando aplicado em apenas um multiplicador e deve ser abordado como trabalho futuro.

3.3.1 Comparação entre Implementações Binária e RNS

Para avaliar o crescimento do atraso em relação ao número de operações em série binário *vs.* RNS, foram utilizados os resultados experimentais dos MACs modulares otimizados apresentados na Tabela 3.9b e foi implementada uma operação binária equivalente ($S = const \times Var_1 + Var_2$). O $DR = 110$ *bits* foi selecionado para ambas as implementações. O conjunto de módulos selecionado para implementação RNS foi $\{2^{2n}, 2^{\pm}1, 2^n \pm 3, 2^n \pm 9, 2^n \pm 15\}$, com $DR = 10n$ e $n = 11$. Dessa forma, o canal com maior atraso (caminho crítico) é o 2^{2n} para a implementação otimizada, ou seja, o atraso da operação é igual a $0,68ns$. Entretanto, é necessário somar o atraso da conversão reversa, que pode ser obtido em [10] e é igual a $4,80ns$. Totalizando para uma operação o atraso total de

$5,48ns$. Já a implementação binária tem um atraso total para uma operação de $4,43ns$.

Apesar da implementação binária ser mais eficiente para uma operação, a partir de duas operações em série, a implementação RNS tem um melhor desempenho, conforme apresentado na Tabela 3.1. Isso porque o custo da conversão reversa permanece constante e a taxa do aumento do atraso por operação adicionada é muito menor para RNS.

Tabela 3.1: *MACs Binário vs. RNS para DR* $= 110$ *bits*

Nº Op.	1	2	3	4	5	6
Binário (*ns*)	4,43	8,86	13,29	17,72	22,15	26,58
RNS (*ns*)	5,48	6,16	6,84	7,52	8,20	8,88
Ganho	−23,70%	30,47%	48,53%	57,56%	62,98%	66,59%

3.4 Conclusão

Neste capítulo, foram apresentados dois métodos de otimização, o primeiro focado nos conversores reversos que são em sua maioria construídos a partir de múltiplas multiplicações entre diferentes constantes e variáveis. O método procura organizar e otimizar as lacunas nos produtos parciais calculados em tempo de projeto de forma a diminuir a complexidade do somador que finalizará o cálculo. O outro método é focado na natureza da reinjeção de *bits* dos módulos $\{2^n \pm k\}$, que possibilita otimizações lógicas consideráveis coluna a coluna.

Apesar dos métodos terem sido concebidos para aplicações específicas, nada impede de aplicar os dois métodos em qualquer multiplicação por constante. Dessa forma, ambos os métodos foram aplicadas em unidades MACs para diferentes módulos e larguras de operandos, com o objetivo de demonstrar seus ganhos e as limitações. Os resultados demonstram um ganho em média no intervalo testado para unidades modulares $\{2^n \pm k\}$, $\{2^n \pm 1\}$ e $\{2^n\}$ de $21,10\%$, $6,95\%$ e $3,70\%$, respectivamente. Nos próximos capítulos, os métodos serão aplicados nos conversores reversos propostos.

CAPÍTULO 4

Propostas dos Conversores Reversos Genéricos

Este capítulo apresenta duas contribuições da tese relacionadas com as melhorias dos conversores reversos baseados em abordagens genéricas com capacidade de prover grande faixa dinâmica e publicadas em [10] e [11]. São elas: *i*) Um novo método para projetar conversores reversos para conjuntos de módulos estendidos apresentado em [9] e uma técnica adicional que minimiza o número de níveis requeridos; *ii*) Uma arquitetura de hardware melhorada para CRTf [12] usando conjuntos de módulos genéricos balanceados. Ao final é apresentada uma seção de resultados que faz a comparação das propostas com a eficiência dos conversores reversos com grande *DR* do estado da arte. Além disso, também são apresentadas as propostas otimizadas com as técnicas apresentadas no capítulo anterior.

4.1 Método para Projetar Conversores Reversos de Dois Níveis para Ampla Faixa Dinâmica

4.1.1 Extensões Híbridas *Multi-Level* para o Conjunto de Três Módulos $\{2^n, 2^n \pm 1\}$

Nesta propostas são considerados ambas extensões (vertical e horizontal). Para a extensão vertical, a potência de dois é estendida, alcançando um conjunto de módulos mais equilibrado, uma vez que os módulos com potência de dois geralmente permitem operações aritméticas mais eficientes do que os conjuntos de módulos restantes para o mesmo comprimento de palavra [21]. Isso leva ao conjunto de módulos $\{2^{n+\beta}, 2^n \pm 1\}$, com $0 \leq \beta \leq n$, cobrindo *DRs* de até $4n$ *bits* [7].

Com o objetivo de conseguir conjuntos de módulos mais amplos usando módulos genéricos, as extensões horizontais do conjunto de módulos são aqui propostas pela adição de pares de módulos conjugados ao conjunto de módulos acima, da mesma forma que [9]. Esses pares de módulos conjugados são da forma $2^n \pm k_j$, $0 \leq j \leq f$, com k_j um valor ímpar no intervalo $1 \leq k_j < 2^{n-1}$ [31]

escolhido de tal forma que todos os módulos são co-primos. Com isso, conjunto de módulos da forma $\{\mathbf{2^{n+\beta}}, \mathbf{2^n \pm k_f}, ..., \mathbf{2^n \pm k_1}, \mathbf{2^n \pm k_0}\}$ são obtidos com um *DR* em torno de $(1+\frac{\beta}{n}+2\times(f+1))\times n$-bit para qualquer número inteiro n.

Os valores de k_j podem ser escolhidos para obter o maior *DR* possível, no entanto, uma função de custo pode ser usada para obter os conjuntos de módulos mais equilibrados e minimizar o número de "1"s na representação de k_j para derivar aritmética eficientes. Já o k_c com menor número de "1"s deriva conversores reversos mais eficientes, onde $\prod_{i=1}^{j-1} m_i = 2^m - k_c$ [9]. Deve-s notar que o método proposto também pode ser usado para derivar conversores reversos para conjuntos de módulos com pares de módulos não conjugados. Aqui, nós apenas detalhamos conjuntos de módulos com pares de módulos conjugados para simplificar a explicação.

Para ilustrar as extensões de conjunto de módulos propostos, primeiro derivamos a extensão para o conjunto de módulos com $f = 1$ e $k_0 = 1$, resultando no conjunto de módulos $\{\mathbf{2^{n+\beta}}, \mathbf{2^n \pm k_1}, 2^n \pm 1, \}$. Também é apresentada a derivação e discussão das limitações de extensão do conjunto de módulos com pares de módulos conjugados para diferentes valores de f.

4.1.1.1 Conjunto de Módulos $\{2^{n+\beta}, 2^n \pm k_1, 2^n \pm 1\}$

Como apresentado acima, vamos considerar o valor $0 \leq \beta \leq n$. Daqui para frente, os valores dos conjuntos de módulos serão ordenados em ordem decrescente, excluindo 2^n+1 e 2^n-1, resultando em: $m_1 = 2^{n+\beta}$, $m_2 = 2^n + k_1$, $m_3 = 2^n - k_1$, $m_4 = 2^n + 1$, $m_5 = 2^n - 1$, enquanto que $\hat{m}_1 = 2^{4n} - 2^{2n}(k_1^2+1) + k_1^2$, $\hat{m}_2 = 2^{n+\beta}(2^{2n}-1)(2^n - k_1)$, $\hat{m}_3 = 2^{n+\beta}(2^{2n}-1)(2^n + k_1)$, $\hat{m}_4 = 2^{n+\beta}(2^n - 1)(2^{2n} - k_1^2)$, $\hat{m}_5 = 2^{n+\beta}(2^n+1)(2^{2n} - k_1^2)$.

Para a extensão proposta, a seguinte expressão $\hat{\hat{m}}_i$ é usada:

$$\hat{\hat{m}}_i = \frac{M}{\prod_{j=1}^{i} m_j} \quad with \;\; 1 \leq i \leq 5.$$

A escolha do k_1 deve satisfazer que o conjunto de módulos resultante $\{\mathbf{2^{n+\beta}}, \mathbf{2^n \pm k_1}, 2^n \pm 1, \}$ seja composto apenas por números co-primos.

Os valores das multiplicativas inversas são números inteiros, os quais podem ser obtidos aplicando a condição $\left|(\hat{m}_i)(\hat{m}_i^{-1})\right|_{m_i} = 1$, $1 \leq i \leq 5$ [22].

É importante perceber que a multiplicativa inversa $\left|\hat{m}_1^{-1}\right|_{m_1}$ satisfaz a seguinte equação quando $0 \leq \beta \leq n$:

$$\left|\left|\hat{m}_1^{-1}\right|_{m_1}\hat{m}_1\right|_{m_1} = \left|\left|\hat{m}_1^{-1}\right|_{m_1}(\overbrace{\left|2^{4n}\right|_{m_1}}^{=0} - \overbrace{\left|2^{2n}\right|_{m_1}}^{=0}(k_1^2+1)+k_1^2)\right|_{m_1} =$$
$$= \left|\left|\hat{m}_1^{-1}\right|_{m_1}k_1^2\right|_{m_1} = 1.$$

Dado que $|(\psi m_1+1)|_{m_1} = 1$, sendo ψ um inteiro positivo, a multiplicativa inversa $\left|\hat{m}_1^{-1}\right|_{m_1}$ pode ser expressa como:

$$\left|\left|\hat{m}_1^{-1}\right|_{m_1}k_1^2\right|_{m_1} = |(\psi m_1+1)|_{m_1} = 1 \Rightarrow \left|\hat{m}_1^{-1}\right|_{m_1} = \frac{\psi m_1+1}{k_1^2}.$$

- Portanto, é possível reduzir o cálculo do módulo de M para $\hat{\hat{m}}_1 = \hat{m}_1$ na equação do CRT (Eq. (2.9)), apresentada na Seção 2.5, do seguinte modo:

$$X = \left|\sum_{i=1}^{5}\overbrace{\hat{m}_i\left|\hat{m}_i^{-1}\right|_{m_i}}^{V_{0i}}R_i\right|_M = \left|\left|\left|\hat{m}_1^{-1}\right|_{m_1}\hat{m}_1R_1\right|_M + \sum_{i=2}^{5}\overbrace{\left|\hat{m}_i^{-1}\right|_{m_i}\hat{m}_i}^{V'_{0i}}R_i\right|_M\Bigg|_M =$$
$$= \left|\left|\hat{m}_1^{-1}\right|_{m_1}(2^{4n}-2^{2n}(k_1^2+1)+k_1^2)R_1 + \sum_{i=2}^{5}V'_{0i}R_i\right|_M =$$
$$= \left|\left|\hat{m}_1^{-1}\right|_{m_1}R_1\left[2^{4n}-2^{2n}(k_1^2+1)\right] + (m_1\psi+1)R_1 + \sum_{i=2}^{5}V'_{0i}R_i\right|_M =$$
$$= \left|\overbrace{\left[\left|\hat{m}_1^{-1}\right|_{m_1}\left[2^{4n}-2^{2n}(k_1^2+1)\right]+m_1\psi\right]}^{V'_{01}}R_1 + R_1 + \sum_{i=2}^{5}V'_{0i}R_i\right|_M =$$
$$= \left|\sum_{i=1}^{5}V'_{0i}R_i + R_1\right|_M, \tag{4.1}$$

onde Eq. (4.1) pode ser reescrita como:

$$X = \sum_{i=1}^{5} V_{0i}^{'} R_i - MA(X) + R_1 = \sum_{i=1}^{5} \left(\frac{V_{0i}^{'}}{m_1} R_i - \frac{\overbrace{M}^{\hat{\hat{m}}_1}}{m_1} A(X) \right) m_1 + R_1 =$$

$$= \left| \sum_{i=1}^{5} \frac{\overbrace{V_{0i}^{'}}^{V_{1i}}}{m_1} R_i \right|_{\hat{\hat{m}}_1} m_1 + R_1 = \overbrace{\left| \sum_{i=1}^{5} V_{1i} R_i \right|_{\hat{\hat{m}}_1}}^{X_1} m_1 + R_1. \tag{4.2}$$

Como $X_1 m_1$ é uma operação de rotação à esquerda de $(n+\beta)$-bits, o X_1 pode ser concatenado com R_1 para derivar X, onde R_1 se torna o bit menos significativo $(n+\beta)$ *bits* de X.

- Na segunda redução, para o cálculo do módulo $\hat{\hat{m}}_2$, o valores V_{1i} são divididos em dois termos, os termos divisíveis e os não divisíveis da operação $\frac{V_{1i}}{\hat{\hat{m}}_2}$. Esses termos são denotados como $V_{1i}^{'}$ e $V_{1i}^{''}$, respectivamente, como apresentado na Eq. (4.3) e seus valores podem ser obtidos a partir da equação de divisão $\sum_{i=1}^{5} V_{1i}$ by $\hat{\hat{m}}_2$, $\sum_{i=1}^{5} V_{1i} = \sum_{i=1}^{5} \overbrace{\left\lfloor \frac{V_{1i}}{\hat{\hat{m}}_2} \right\rfloor}^{V_{1i}^{'}} \hat{\hat{m}}_2 + \sum_{i=1}^{2} V_{1i}^{''}$. É importante notar que $\hat{m}_j$ é divisível por $\hat{\hat{m}}_2$ for $3 \leq j \leq 5$ e consequentemente $V_{1j}^{''} = 0$ nesses casos.

$$
\begin{aligned}
X_1 &= \left|\sum_{i=1}^{5} V_{1i}R_i\right|_{\hat{\hat{m}}_1} = \left|\left|\sum_{i=1}^{5} V'_{1i}R_i\right|_{\hat{\hat{m}}_1} + \left|\sum_{i=1}^{2} V''_{1i}R_i\right|_{\hat{\hat{m}}_1}\right|_{\hat{\hat{m}}_1} = \\
&= \left|\sum_{i=1}^{5} V'_{1i}R_i - \hat{\hat{m}}_1 A(X_1) + \left|\sum_{i=1}^{2} V''_{1i}R_i\right|_{\hat{\hat{m}}_1}\right|_{\hat{\hat{m}}_1} = \\
&= \left|\sum_{i=1}^{5}\left(\frac{\overbrace{V'_{1i}}^{V_{2i}}}{m_2}R_i - \frac{\overbrace{\hat{\hat{m}}_1}^{\hat{\hat{m}}_2}}{m_2}A(X_1)\right)m_2 + \left|\sum_{i=1}^{2}\overbrace{V''_{1i}}^{\phi_{2i}}R_i\right|_{\hat{\hat{m}}_1}\right|_{\hat{\hat{m}}_1} = \\
&= \left|\overbrace{\left|\sum_{i=1}^{5} V_{2i}R_i\right|_{\hat{\hat{m}}_2}}^{X_2} m_2 + \left|\sum_{i=1}^{2}\phi_{2i}R_i\right|_{\hat{\hat{m}}_1}\right|_{\hat{\hat{m}}_1}.
\end{aligned}
\tag{4.3}
$$

- Na terceira redução para o cálculo do módulo $\hat{\hat{m}}_3$, os valores V_{2i} são divididos em $\sum_{i=1}^{5} V_{2i} = \sum_{i=1}^{5}(V'_{2i} + V''_{2i})$ usando a mesma abordagem acima. Observando que nesses casos $V''_{2i} = 0$ for $4 \leq j \leq 5$:

$$
\begin{aligned}
X_2 &= \left|\sum_{i=1}^{5} V_{2i}R_i\right|_{\hat{\hat{m}}_2} = \left|\left|\sum_{i=1}^{5} V'_{2i}R_i\right|_{\hat{\hat{m}}_2} + \left|\sum_{i=1}^{3} V''_{2i}R_i\right|_{\hat{\hat{m}}_2}\right|_{\hat{\hat{m}}_2} = \\
&= \left|\sum_{i=1}^{5} V'_{2i}R_i - \hat{\hat{m}}_2 A(X_2) + \left|\sum_{i=1}^{3} V''_{2i}R_i\right|_{\hat{\hat{m}}_2}\right|_{\hat{\hat{m}}_2} = \\
&= \left|\sum_{i=1}^{5}\left(\frac{\overbrace{V'_{2i}}^{V_{3i}}}{m_3}R_i - \frac{\overbrace{\hat{\hat{m}}_2}^{\hat{\hat{m}}_3}}{m_3}A(X_2)\right)m_3 + \left|\sum_{i=1}^{3}\overbrace{V''_{2i}}^{\phi_{3i}}R_i\right|_{\hat{\hat{m}}_2}\right|_{\hat{\hat{m}}_2} = \\
&= \left|\overbrace{\left|\sum_{i=1}^{5} V_{3i}R_i\right|_{\hat{\hat{m}}_3}}^{X_3} m_3 + \left|\sum_{i=1}^{3}\phi_{3i}R_i\right|_{\hat{\hat{m}}_2}\right|_{\hat{\hat{m}}_2}.
\end{aligned}
\tag{4.4}
$$

No fim:

$$X = X_1 m_1 + R_1; \qquad X_1 = \left| X_2 m_2 + \left| \sum_{i=1}^{2} \phi_{2i} R_i \right|_{\hat{\hat{m}}_1} \right|_{\hat{\hat{m}}_1};$$

$$X_2 = \left| X_3 m_3 + \left| \sum_{i=1}^{3} \phi_{3i} R_i \right|_{\hat{\hat{m}}_2} \right|_{\hat{\hat{m}}_2}; \qquad X_3 = \left| \sum_{i=1}^{5} V_{3i} R_i \right|_{\hat{\hat{m}}_3}. \tag{4.5}$$

É importante observar que $\phi_{3i} < m_3$ e $\phi_{2i} < m_2$, e consequentemente, as multiplicações dessas constantes pelos R_i correspondente não são operações modulares. As restrições para $\phi_{3i}R_i$ e $\phi_{2i}R_i$ são:

$$\begin{aligned} max(\phi_{31}R_1) &= (m_3-1)(m_1-1) < m_3 m_4 m_5; \\ max(\phi_{32}R_2) &= (m_3-1)(m_2-1) < m_3 m_4 m_5; \\ max(\phi_{33}R_3) &= (m_3-1)(m_3-1) < m_3 m_4 m_5; \\ max(\phi_{21}R_1) &= (m_2-1)(m_1-1) < m_2 m_3 m_4 m_5; \\ max(\phi_{22}R_2) &= (m_2-1)(m_2-1) < m_2 m_3 m_4 m_5, \end{aligned} \tag{4.6}$$

onde o intervalo $0 \leq \beta \leq n$ garante a Eq. (4.6).

Para obter uma adição não modular dos termos $\phi_{3i}R_i$ e $\phi_{2i}R_i$, $X = M-1$ é definido como entrada para fornecer os valores máximos de resíduos em $R_i = m_i - 1$, $1 \leq i \leq 3$:

$$R_i = |M-1|_{m_i} = \left| \overbrace{|M|_{m_i}}^{=0} - 1 \right|_{m_i} = |-1|_{m_i} = m_i - 1. \tag{4.7}$$

Assim sendo:

$$\begin{aligned} \sum_{i=1}^{3} \phi_{3i} R_i &= \overbrace{\phi_{31}}^{<m_3}(m_1-1) + \overbrace{\phi_{32}}^{<m_3}(m_2-1) + \overbrace{\phi_{33}}^{<m_3}(m_3-1) < \hat{\hat{m}}_2; \\ \sum_{i=1}^{2} \phi_{2i} R_i &= \overbrace{\phi_{21}}^{<m_2}(m_1-1) + \overbrace{\phi_{22}}^{<m_2}(m_2-1) < \hat{\hat{m}}_1, \end{aligned} \tag{4.8}$$

o que está satisfeito por $\beta = 0$ devido ao valor de $m_1 - 1 = m_5$, consequentemente, $\sum_{i=1}^{j+1} \phi_{(j+1)i} R_i < \hat{\hat{m}}_j$, por $j = 1$ e $j = 2$. No entanto, quando $\beta = n$ o valor

de $m_1 - 1$ é $m_4 m_5$ e, consequentemente, a Eq. (4.8) não pode ser satisfeita para alguns casos particulares. Se o módulo selecionado não satisfizer a Eq. (4.8), então a solução consiste na redução de β para $\beta = 0$. Uma vez que os valores de ϕ_{3i} e ϕ_{2i} são obtidos e é assegurado que Eq. (4.8) esteja satisfeita, é possível reescrever a Eq. (4.5) como:

$$X = X_1 m_1 + R_1; \qquad X_1 = \left| X_2 m_2 + \sum_{i=1}^{2} \phi_{2i} R_i \right|_{\hat{m}_1};$$
$$X_2 = \left| X_3 m_3 + \sum_{i=1}^{3} \phi_{3i} R_i \right|_{\hat{m}_2}; \qquad X_3 = \left| \sum_{i=1}^{5} V_{3i} R_i \right|_{\hat{m}_3}. \tag{4.9}$$

É importante notar que outra redução para $\hat{m}_4$ é possível. No entanto, o cálculo do módulo $\hat{m}_3$ pode ser facilmente implementada com uma operação binária de rotação à esquerda *ROL* (*Rotation Left*), com base em uma árvore *CSA* e um *EAC*, como explicado em [7], desde que $\hat{m}_3 = m_4 m_5 = 2^{2n} - 1$. As multiplicações pelos termos ϕ_{ji} e m_i, $1 \leq i \leq 3$ não são operações modulares, portanto, são implementadas por multiplicadores binários convencionais. O passo de conversão final para derivar X_3 é implementado usando um *KSA* e um *EAC*. O *KSA* é aqui considerado, uma vez que é a estrutura de somador com o melhor desempenho [32].

O passo final da conversão para derivar X_1 e X_2 requer apenas uma comparação, uma vez que $max\left(X_3 m_3 + \sum_{i=1}^{3} \phi_{3i} R_i\right) < 2 \times \hat{m}_2$ e $max(X_2 m_2 + \sum_{i=1}^{2} \phi_{2i} R_i) < 2 \times \hat{m}_1$. A arquitetura do conversor $\{\mathbf{2^{2n}}, \mathbf{2^n \pm k_1}, 2^n \pm 1\}$ resultante usando essa abordagem é mostrado na Figura 4.1.

4.1.1.2 Conjunto de Módulos $\{2^{n+\beta}, 2^n \pm k_f, \ldots, 2^n \pm k_2, 2^n \pm k_1, 2^n \pm 1\}$

Para uma extensão geral, a qual corresponde o conjunto de módulos $\{\mathbf{2^{n+\beta}}, \mathbf{2^n \pm k_f}, \ldots, \mathbf{2^n \pm k_2}, \mathbf{2^n \pm k_1}, 2^n \pm 1\}$, as multiplicativas inversas devem satisfazer a condição $\left|(\hat{m}_i)(\hat{m}_i^{-1})\right|_{m_i} = 1$, para $1 \leq i \leq (3 + 2 \times f)$. O número de reduções iterativas e módulos nos conjuntos são $t = 2 \times f + 1$ e $N = 2 \times f + 3$, respectivamente.

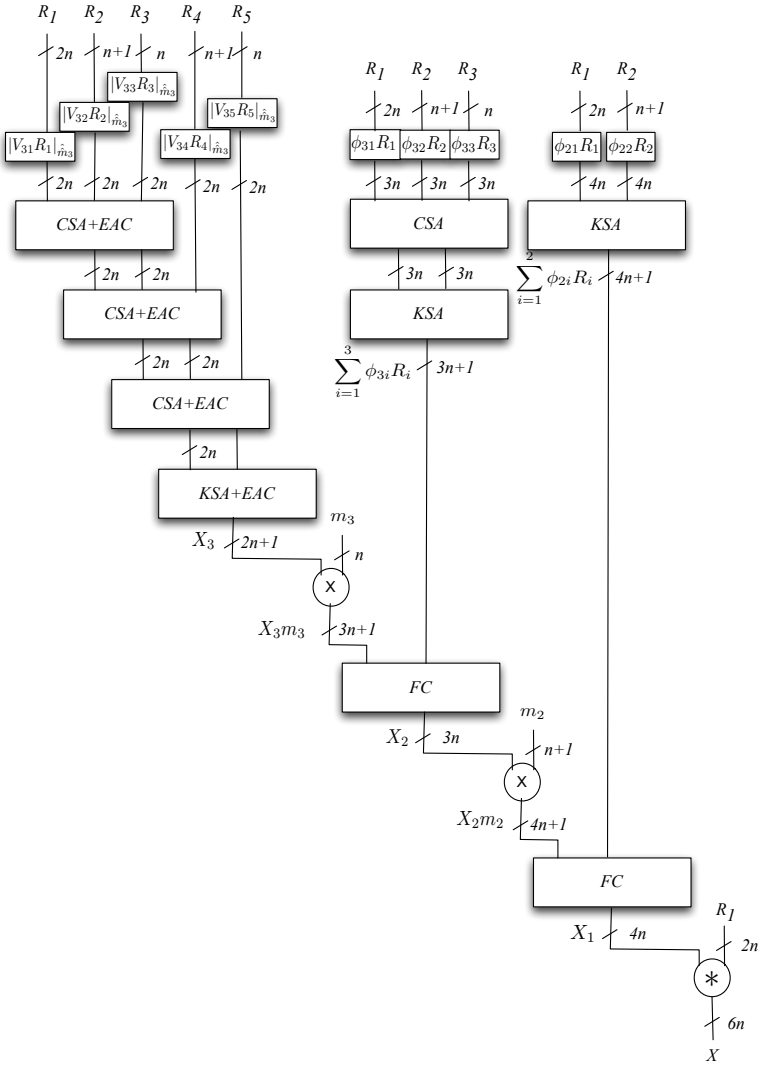


Figura 4.1: *Diagrama de bloco do conversor reverso* $\{\mathbf{2^{2n}}, \mathbf{2^n \pm k_1}, 2^n \pm 1\}$ *para abordagem* Multi-level.

Para obter o valor binário de X, Eq. (4.9) pode ser estendida:

$$
\begin{aligned}
X &= \overbrace{\left|\sum_{i=1}^{N} V_{1i}R_i\right|_{\hat{\hat{m}}_1}}^{X_1} m_1 + R_1 = \left|\sum_{i=1}^{N} V_{0i}R_i\right|_{\hat{\hat{m}}_0}; \\
X_1 &= \left|\overbrace{\left|\sum_{i=1}^{N} V_{2i}R_i\right|_{\hat{\hat{m}}_2}}^{X_2} m_2 + \sum_{i=1}^{2} \phi_{2i}R_i\right|_{\hat{\hat{m}}_1} = \left|\sum_{i=1}^{N} V_{1i}R_i\right|_{\hat{\hat{m}}_1}; \\
&\cdots \\
X_{j-1} &= \left|\overbrace{\left|\sum_{i=1}^{N} V_{ji}R_i\right|_{\hat{\hat{m}}_j}}^{X_j} m_j + \sum_{i=1}^{j} \phi_{ji}R_i\right|_{\hat{\hat{m}}_{j-1}} = \left|\sum_{i=1}^{N} V_{(j-1)i}R_i\right|_{\hat{\hat{m}}_{j-1}}; \\
&\cdots \\
X_{t-1} &= \left|X_t m_t + \sum_{i=1}^{t} \phi_{(t)i}R_i\right|_{\hat{\hat{m}}_{(t-1)}} = \left|\sum_{i=1}^{N} V_{(t-1)i}R_i\right|_{\hat{\hat{m}}_{(t-1)}}; \\
X_t &= \left|\sum_{i=1}^{N} V_{ti}R_i\right|_{\hat{\hat{m}}_t}.
\end{aligned}
\tag{4.10}
$$

Os parâmetros da Eq. (4.10) para a redução $(j+1)$, $1 \leq j \leq t-1$, podem ser derivados da seguinte forma.

$$
\begin{aligned}
\left|\sum_{i=1}^{N} V_{ji}R_i\right|_{\hat{\hat{m}}_j} &= \left|\overbrace{\sum_{i=1}^{N} \left\lfloor \frac{V_{ji}}{\hat{\hat{m}}_{j+1}} \right\rfloor \hat{\hat{m}}_{j+1} R_i}^{V'_{ji}} + \left|\sum_{i=1}^{j+1} V''_{ji}R_i\right|_{\hat{\hat{m}}_j}\right|_{\hat{\hat{m}}_j} = \\
&= \left|\overbrace{\sum_{i=1}^{N} \frac{V'_{ji}}{m_{j+1}} R_i}^{V_{(j+1)i}} m_{j+1} + \sum_{i=1}^{j+1} \overbrace{V''_{ji}}^{\phi_{(j+1)i}} R_i\right|_{\hat{\hat{m}}_{j+1}}\Bigg|_{\hat{\hat{m}}_j},
\end{aligned}
\tag{4.11}
$$

o mesmo método pode ser usado com a Eq. (4.3) e Eq. (4.4).

Para garantir que ao calcular $\sum_{i=1}^{j+1} \phi_{(j+1)i} R_i$ não seja necessária uma soma modular $\left|\sum_{i=1}^{j+1} \phi_{(j+1)i} R_i\right|_{\hat{m}_j}$, é necessário satisfazer a extensão da Eq. (4.8) para a redução $(j+1)$ $1 \leq j \leq t-1$. Nesse caso, também é usado $X = M-1$ como entrada para alcançar os valores máximos residuais em $R_i = m_i - 1$, $1 \leq i \leq N$, como apresentado na Eq. (4.7). Assim sendo:

$$\sum_{i=1}^{j+1} \phi_{(j+1)i} R_i = \sum_{i=1}^{j+1} \phi_{(j+1)i}(m_i - 1) < \hat{\hat{m}}_j.$$

que é satisfeito por $\beta = 0$ e na maioria dos casos por $\beta = n$.

4.1.2 Extensões híbridas de Dois Níveis do Conjunto $\{2^n, 2^n \pm 1\}$

É possível simplificar o número de reduções iterativas para duas iterações usando o *Lemma 1* e *Lemma 2* [33]:

Lemma 1:

$$|A|_p \times k = |k \times A|_{k \times p} .\%labellemma1_generic$$

Lemma 2:

$$\left||A|_q\right|_p = |A - k \times q|_p = |A|_p ; q = k \times p .\%labellemma2_generic$$

Para o conjunto de cinco módulos $\{\mathbf{2^{n+\beta}}, \mathbf{2^n \pm k_1}, 2^n \pm 1\}$, é possível evitar uma redução iterativa se o *Lemma 1* é aplicado para $X_2 m_2$ na Eq. (4.9):

$$X_2 m_2 = \left| X_3 m_3 + \sum_{i=1}^{3} \phi_{3i} R_i \right|_{\hat{\hat{m}}_2} \times m_2 = \left| X_3 m_3 m_2 + \sum_{i=1}^{3} \phi_{3i} m_2 R_i \right|_{\hat{\hat{m}}_1} .$$

Assim sendo, a Eq. (4.9) pode ser reduzida aplicando o *Lemma 2* para:

$$\begin{aligned} X &= X_A m_1 + R_1; \\ X_A &= \left| X_3 m_2 m_3 + \sum_{i=1}^{3} \beta_{Ai} R_i \right|_{\hat{\hat{m}}_1} ; \\ X_3 &= \left| \sum_{i=1}^{5} V_{3i} R_i \right|_{\hat{\hat{m}}_3} , \end{aligned} \tag{4.12}$$

com $\sum_{i=1}^{3} \beta_{Ai} R_i = \sum_{i=1}^{3} \phi_{3i} m_2 R_i + \sum_{i=1}^{2} \phi_{2i} R_i$.

É importante notar que os termos $\beta_{Ai} R_i$ são multiplicações não modulares, uma vez que $0 \leq \beta_{Ai} < m_2 m_3 - 1$:

$$\begin{aligned} max(\beta_{A1} R_1) &= [(m_3 - 1)m_2 + (m_2 - 1)](m_1 - 1) < m_3 m_4 m_5; \\ max(\beta_{A2} R_2) &= [(m_3 - 1)m_2 + (m_2 - 1)](m_2 - 1) < m_3 m_4 m_5; \\ max(\beta_{A3} R_3) &= [(m_3 - 1)m_2 + (m_2 - 1)](m_3 - 1) < m_3 m_4 m_5, \end{aligned} \tag{4.13}$$

onde o intervalo $0 \leq \beta \leq n$ garante as condições expressa na Eq. (4.13).

Para obter uma adição não modular na soma dos termos $\sum_{i=1}^{3} \beta_{Ai} R_i$, $X = M - 1$ é definido como entrada para atingir os valores máximos de resíduos em $R_i = m_i - 1$, $1 \leq i \leq 3$ como apresentado na Eq. (4.7), portanto:

$$\beta_{A1}(m_1 - 1) + \beta_{A2}(m_2 - 1) + \beta_{A3}(m_3 - 1) < m_2 m_3 m_4 m_5, \tag{4.14}$$

o qual é sempre ajustado para $\beta = 0$. Quando os valores de β_{Ai} para $\beta = n$ são obtidos e a condição da Eq. (4.14) é satisfeita, pode-se garantir uma adição não modular $\sum_{i=1}^{3} \beta_{Ai} R_i$ como apresentado na Eq. (4.12). Se o módulo selecionado não satisfaz a Eq. (4.14), então a solução consiste na redução de $\beta = n$ para $\beta = 0$.

A arquitetura de $\{\mathbf{2}^{2n}, \mathbf{2}^n \pm \boldsymbol{k}_1, 2^n \pm 1\}$ genérica usando essa aproximação é apresentada na Figura 4.2. A principal diferença em comparação com a abordagem *Multi-level* é o uso de apenas um multiplicador e um FC.

Para o conjunto de módulos genérico $\{\mathbf{2}^{2n}, \mathbf{2}^n \pm \boldsymbol{k}_f, ..., \mathbf{2}^n \pm \boldsymbol{k}_2, \mathbf{2}^n \pm \boldsymbol{k}_1, 2^n \pm 1\}$, com $t = 2 \times f + 1$ e $N = 2 \times f + 3$, pode-se aplicar o Lemma 1 $2(t-3)$ vezes e consequentemente a Eq. (4.12) pode ser expressa como:

$$\begin{aligned} X &= X_A m_1 + R_1; \\ X_A &= \left| X_3 \prod_{i=2}^{t} m_i + \sum_{i=1}^{t} \beta_{Ai} R_i \right|_{\hat{m}_1}; \\ X_3 &= \left| \sum_{i=1}^{N} V_{3i} R_i \right|_{\hat{m}_3}. \end{aligned}$$

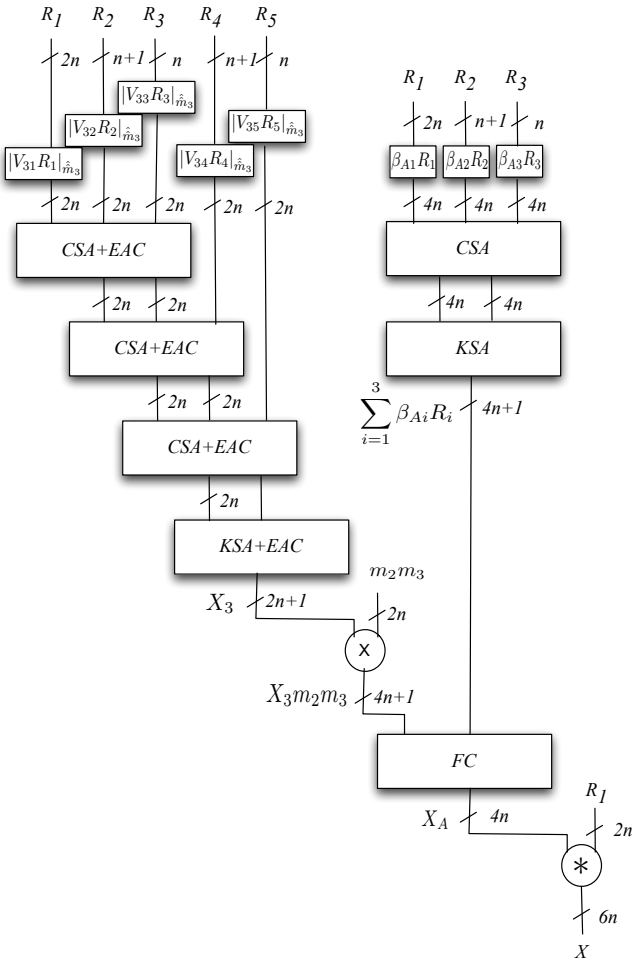


Figura 4.2: *Diagrama de bloco do conversor reverso* $\{\mathbf{2^{2n}}, \mathbf{2^n \pm k_1}, 2^n \pm 1\}$ *para abordagem* Two-level.

Nesse caso:

$$\sum_{i=1}^{t-1}\beta_{Ai}R_i = \sum_{i=1}^{2}\phi_{2i}R_i + \sum_{j=1}^{2}\sum_{i=1}^{t-2}\phi_{j(t+1-i)}\left(\prod_{i-1}^{t-1}m_{(t+1-i)}\right)R_j + \\ + \sum_{j=3}^{t-1}\sum_{i=3}^{t+3-j}\phi_{j(t+3-i)}\left(\prod_{i-1}^{t-1}m_{(t+1-i)}\right)R_j + \phi_{(t)t}\left(\prod_{i=2}^{t-1}m_i\right)R_t,$$

onde os valores de ϕ_{ji} podem ser derivados da Eq. (4.11).

A fim de garantir que a adição $\sum_{i=1}^{t}\beta_{Ai}R_i$ possa ser realizada por somadores não modulares, os valores de β_{Ai} obtidos multiplicando pelos valores máximos dos resíduos, $R_i = m_i - 1$, $1 \leq i \leq N$, precisam satisfazer:

$$\sum_{i=1}^{t}\beta_{Ai}R_i = \sum_{i=1}^{t}\beta_{Ai}(m_i-1) < \hat{\hat{m}}_1,$$

o que é satisfeito por $\beta = 0$ e na maioria dos casos por $\beta = n$.

4.1.3 Estimativa de Desempenho: Um Estudo de Caso

Para melhor ilustrar a metodologia proposta, um estudo de caso particular para o conjunto de módulos $\{\mathbf{2}^{2n}, 2^n \pm \mathbf{3}, 2^n \pm 1\}$ with $n = 4$, $\{m_1, m_2, m_3, m_4, m_5\} = \{256, 19, 13, 17, 15\}$, é aqui considerado. A seguir é apresentado os parâmetros resultantes para cada abordagem de conversão. Aplicando CRT [1]:

$$X = \Big| |\overbrace{3590145}^{V_{01}}R_1|_{16124160} + |\overbrace{3394560}^{V_{02}}R_2|_{16124160} + \\ + |\overbrace{11162880}^{V_{03}}R_3|_{16124160} + |\overbrace{15175680}^{V_{04}}R_4|_{16124160} + \\ + |\overbrace{15049216}^{V_{05}}R_5|_{16124160}\Big|_{16124160}. \tag{4.15}$$

Aplicando MRC [3]:

$$
\begin{aligned}
X = {} & \overbrace{|8(|7(|4(|1(R_5 - V_1)|_{15} - V_2)|_{15} - V_3)|_{15} - V_4)|_{15}}^{V_5} \times \overbrace{1074944}^{=m_1m_2m_3m_4} + \\
& + \overbrace{|4(|9(|1(R_4 - V_1)|_{17} - V_2)|_{17} - V_3)|_{17}}^{V_4} \times \overbrace{63232}^{=m_1m_2m_3} + \\
& + \overbrace{|11(|3(R_3 - V_1)|_{13} - V_2)|_{13}}^{V_3} \times \overbrace{4864}^{=m_1m_2} + \\
& + \overbrace{|17(R_2 - V_1)|_{19}}^{V_2} \times \overbrace{256}^{=m_1} + \overbrace{R1}^{=V_1},
\end{aligned}
\tag{4.16}
$$

onde os valores das constantes podem ser extraídos de [34].

Aplicando a metodologia descrita em [9], baseada no Novo CRT-I:

$$
\begin{aligned}
X = {} & \Big| |\overbrace{14024}^{V_{11}} R_1|_{62985} + |\overbrace{13260}^{V_{12}} R_2|_{62985} + |\overbrace{43605}^{V_{13}} R_3|_{62985} + \\
& + |\overbrace{59280}^{V_{14}} R_4|_{62985} + |\overbrace{58786}^{V_{15}} R_5|_{62985} \Big|_{62985} \times 256 + R_1.
\end{aligned}
\tag{4.17}
$$

Aplicando a metodologia descrita na Subseção 4.1.1:

$$
\begin{aligned}
X = {} & \Big|\Big|\Big|\Big| |\overbrace{56}^{V_{31}} R_1|_{255} + |\overbrace{53}^{V_{32}} R_2|_{255} + |\overbrace{176}^{V_{33}} R_3|_{255} + |\overbrace{240}^{V_{34}} R_4|_{255} + \\
& + |\overbrace{238}^{V_{35}} R_5|_{255} \Big|_{255} \times 13 + \overbrace{10}^{\phi_{31}} R_1 + \overbrace{8}^{\phi_{32}} R_2 + \\
& + \overbrace{7}^{\phi_{33}} R_3 \Big|_{3315} \times 19 + \overbrace{2}^{\phi_{21}} R_1 + \overbrace{17}^{\phi_{22}} R_2 \Big|_{62985} \times 256 + R_1.
\end{aligned}
\tag{4.18}
$$

Aplicando a metodologia descrita na Subseção 4.1.2:

$$X = \Big\| |\overbrace{56}^{V_{31}} R_1|_{255} + |\overbrace{53}^{V_{32}} R_2|_{255} + |\overbrace{176}^{V_{33}} R_3|_{255} + |\overbrace{240}^{V_{34}} R_4|_{255} + \\ + |\overbrace{238}^{V_{35}} R_5|_{255} \Big|_{255} \times 247 + \overbrace{192}^{\beta_{A1}} R_1 + \overbrace{169}^{\beta_{A2}} R_2 + \\ + \overbrace{133}^{\beta_{A3}} R_3 \Big|_{62985} \times 256 + R_1. \tag{4.19}$$

A Tabela 4.1 apresenta a comparação das operações de diferentes abordagens de conversão reversa para esse caso de estudo.

Tabela 4.1: *Comparação de operações entre as abordagens dos algoritmos de conversão para* $\{\mathbf{2^{2n}}, \mathbf{2^n \pm 3}, 2^n \pm 1\}$, $n = 4$

Operação		CRT [1]	MRC [3]	Novo CRT [9]	*Multi-level* híbrido	2-níveis híbrido
Modular $\left\|V_{ji}R_i\right\|_{\hat{m}_j}$	#	5	10	5	5	5
	$\hat{m}_j$	16124160	*	62985	255	255
Adição	#	1	11	1	3	2
	Compressor (bits)	$5:1$ $(6n)$	$6:1$; $(2:1)\times 10$ $(6n)$;$(2n)\times 4$;$(3n)\times 3$; $(4n)\times 2$; $(5n)$	$5:1$ $(4n)$	$5:1$; $3:1$; $2:1$ $(2n)$; $(3n)$; $(4n)$	$5:1$; $3:1$ $(2n)$; $(4n)$
Multiplicação	#	–	4	–	7	4
	Comprimento da Mult.	–	$(n+1)\times 2n$ $n\times(3n+1)$ $(n+1)\times 4n$ $n\times(5n+1)$	–	$(2n+1)\times n$; $n\times n$; $2n\times n$; $(n+1)\times n$; $3n\times(n+1)$; $2n\times 2n$; $2n\times(n+1)$	$(2n+1)\times 2n$; $2n\times(n+1)$; $2n\times 2n$; $2n\times n$
FC Modular	#	1	–	1	2	1
	Comparisons	1	–	1	1	1

* A multiplicação modular dada não está na forma $\left\|V_{ji}R_i\right\|_{\hat{m}_j}$. Cada multiplicação modular da implementação do MRC é mostrada na Eq. (4.16).

Os conversores baseados no CRT [1] e no Novo CRT-I [9] são implementados usando as operações modulares $\left|V_{ji}R_i\right|_{\hat{m}_j}$ para obter uma única comparação no FC. No entanto, as adições e multiplicações modulares necessárias são mais complexas do que o resto das abordagens existentes e aqui propostas. Quando os termos V_{ji} têm uma grande quantidade "1s" em suas representações binárias, a abordagem usada em [21], que usa operações *ROL* para derivar as operações modulares na forma $\left|V_{ji}R_i\right|_{\hat{m}_j}$, não é eficiente.

As abordagens apresentadas em [9] requerem operações modulares complexas $\left|V_{ji}R_i\right|_{\hat{m}_j}$, uma vez que $\hat{m}_j$ não são da forma $\{2^n \pm 1\}$. No entanto, as propostas híbridas e *Two-level* usam o módulo $\hat{m}_j$ e $\hat{m}_3$, respectivamente, igual a $2^{2n}-1$, que permite a simplificação das operações modulares.

É importante notar que a adição dos cinco termos com $2n$ *bits* de comprimento, na abordagem proposta, é realizada por um *CSA+EAC* como explicado na Subsecção 4.1.1.

Dado que o uso de $\hat{m}_t$ e $\hat{m}_3$ é igual a $2^{2n}-1$ para as propostas híbrida e *Two-level*, respectivamente, as adições ao final das árvores *CSA+EAC* podem ser implementadas com um *KSA+EAC*. Além disso, os estágios de conversão final restantes requerem uma única comparação. É importante notar que, as implementações baseadas em CRT [1] e Novo CRT-I [9] requerem hardware extra na árvore *CSA* para calcular o passo de conversão final com uma única comparação.

4.2 Conversão Reversa CRTf Usando Conjuntos de Módulos Genéricos Balanceados

Para esta seção vamos considerar que um valor binário X é representado em RNS baseado no conjunto de módulos $\{m_1, m_2, \ldots, m_L\}$ como sendo $(x_1, x_2, \ldots, x_L)$ e que a faixa dinâmica é dada por $M = \prod_{i=1}^{L} m_i$.

A modificação do CRT usando valores fracionários (CRTf) foi introduzida pela primeira vez em [35] para realizar detecção de sinais e divisão em RNS. Todos os números em RNS de um determinado conjunto de módulos estão localizados no intervalo $[0, M)$. Se ambos os lados da Eq. (2.9) do CRT, apresentada no capítulo anterior, forem divididos por M, temos:

$$\tilde{X} = \frac{X}{M} = \left| \sum_{i=1}^{L} \frac{\left|M_i^{-1}\right|_{m_i}}{m_i} x_i \right|_1 = \left| \sum_{i=1}^{L} c_i x_i \right|_1 , \tag{4.20}$$

onde $c_i = \frac{\left|M_i^{-1}\right|_{m_i}}{m_i}$, $i = 1, 2, \ldots, L$, são constantes e podem ser pre-computadas. O valor $\tilde{X}$ pode ser considerado com uma característica posicional do número X.

A principal vantagem deste método é evitar o uso do módulo complexo M. No entanto, operar em números fracionários é uma computação complexa. [12] mostra que, aplicando a Eq. (4.20) de forma semelhante, a conversão reversa de X pode ser expressa como:

$$X = \left\lfloor \frac{\left|\sum_{i=1}^{L} c_i' x_i\right|_{2^N} \cdot M}{2^N} \right\rfloor = \left\lfloor \frac{X' \cdot M}{2^N} \right\rfloor . \tag{4.21}$$

Constantes c_i' apresentadas na Eq. (4.21) podem ser obtidas de:

$$c_i' = \left\lceil \frac{\left|M_i^{-1}\right|_{m_i}}{m_i} \cdot 2^N \right\rceil , i = 1, 2, \cdots, L, \tag{4.22}$$

e os valores de N representa a quantidade de *bits* da divisão final:

$$N = \left\lceil \log_2 M + \log_2 \sum_{i=1}^{L} (m_i - 1) \right\rceil - 1. \tag{4.23}$$

Embora o algoritmo CRTf possa ser aplicado para conjuntos de módulos genéricos, seria desejável escolher valores de módulos balanceados para o conjunto de módulos. Assim, esta seção propõe o uso de um conjunto de módulos genéricos com pares conjugados $m_i = \{2^{n+\beta}, 2^n \pm k_0, 2^n \pm k_1, \ldots, 2^n \pm k_f\}$, com k_f ímpar e $0 \leq \beta \leq 2n$, que foi amplamente utilizado no estado da arte como um conjunto de módulos balanceados [9]. O número de módulos neste conjunto de módulos é $L = 2f + 1$. O valor de M é expresso para o módulo genérico como:

$$M = 2^{n+\beta} M' = 2^{n+\beta} \prod_{i=2}^{L} m_i. \tag{4.24}$$

O valor de N na Eq. (4.23) tem dois termos logarítmicos. O primeiro termo logarítmico tem uma largura igual a DR. O segundo termo logarítmico tem uma largura que pode ser reescrita como $\sum_{i=1}^{L}(m_i-1) = \sum_{i=1}^{L}|m_i-1|_{m_i} = \sum_{i=1}^{L}|-1|_{m_i} = \sum_{i=1}^{L}|M-1|_{m_i}$. Para o conjunto de módulos genéricos considerado, o segundo termo pode ser escrito como $\log_2(2^{n+\beta}+(L-1)2^n-L)$ e resulta em $n+\beta$ quando $\beta \gg 1$. Assim, o valor final é $N \approx \mathrm{DR}+(n+\beta)$. Substituindo esta expressão e a Eq. (4.24) na Eq. (4.21) resulta em:

$$X = \left\lfloor \frac{X' \cdot 2^{n+\beta} M'}{2^{DR+(n+\beta)}} \right\rfloor = \left\lfloor \frac{X' \cdot M'}{2^{DR}} \right\rfloor. \tag{4.25}$$

A Eq. (4.25) pode ser expressa na forma binária para qualquer valor de f e β como:

$$X = \left\lfloor \frac{\sum_{i=0}^{N-1} 2^i x'_i \cdot \sum_{j=0}^{DR-1} 2^j m'_j}{2^{DR}} \right\rfloor =$$
$$= \left\lfloor \frac{\sum\limits_{i+j \geq DR} 2^{(i+j-DR)} x'_i m'_j \cdot \sum\limits_{i+j<DR} 2^{(i+j)} x'_i m'_j}{2^{DR}} \right\rfloor, \tag{4.26}$$

onde x'_i e m'_j são os *bits* na representação binária. Os limites das somas representam apenas as regiões onde há informações relevantes, ou seja, *bits* diferentes de zero.

A Figura 4.3 apresenta a arquitetura de hardware derivada para o conjunto de módulos genéricos. Como pode ser notado, a multiplicação binária seguida pela divisão na Eq. (4.26), permite que o produto final seja realizado em dois compressores paralelos. A informação relevante reside nos *bits* mais significativos (MSBs), que são computados usando um Compressor paralelo e um somador *KSA* (*MSBs Compressor* + KSA). Para determinar a soma final dos MSBs, é necessário calcular a contribuição do *carry* da parte LSB (*Cout* de *LSBs Compressor*), usado como entrada (*Cin*) no Compressor paralelo dos *MSBs*. A implementação original [12] propôs a execução da adição final usando $2DR+\beta+n$ *bits*, que é reduzido para DR usando a arquitetura proposta e tem vantagens em atraso e área.

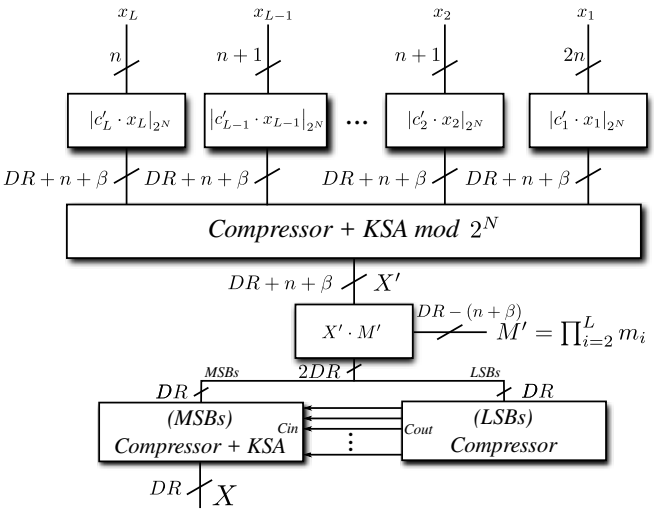


Figura 4.3: *Arquitetura de hardware proposta para o algoritmo CRTf.*

4.3 Resultados Experimentais

Os resultados experimentais foram separados em duas partes. A primeira parte avalia o desempenho e o custo da abordagem genérica de conversão reversa aqui proposto, comparando as estruturas de conversão *Two-level* e *Multi-Level* com as do estado da arte. A segunda parte avalia o CRTf com conjunto de módulos genéricos balanceados e aplicação das técnicas de compactação apresentadas no Capítulo 3. Elas foram aplicadas ao conversor reverso *Two-level* e ao próprio CRTf proposto. Além disso, também é realizada uma comparação com os conversores reversos do estado da arte para $DR = 6n$, $DR = 8n$ e $DR = 10n$. Para tal, todas as implementações foram descritas em VHDL e sintetizadas. A ferramenta utilizada foi a *Synopsys Design Vision* (versão E-2010.12-SP4) para gerar um *netlist* otimizado mapeado para tecnologia *ASIC standard cells* de 90*n*m da empresa UMC [29]. Os parâmetros de compilação foram *"-exact_map -map_effort medium -area_effort medium -power_effort medium"*.

4.3.1 Resultados e Discussão das Propostas *Two-level* e *Multi-Level*

Para avaliar a escalabilidade dos conversores reversos propostos (*Two-level* e *Multi-Level*), resultados experimentais para $DR = 6n$ ($4 \leq n < 22$) foram obtidos para os melhores conversores do estado da arte. São eles: *i*) conjunto de módulos $\{\mathbf{2^{2n}}, 2^n \pm 1, \mathbf{2^{2n} + 1}\}$ [17], o qual é a estrutura de conversão mais eficiente com relação a área; *ii*) conjunto de módulos $\{\mathbf{2^{n+\beta}}, 2^n \pm 1, \mathbf{2^n \pm 2^{\frac{n+1}{2}} + 1}\}$ [21], para $\beta = 0$ e $\beta = n$, sendo a configuração com $\beta = n$ a estrutura de conversão com o melhor desempenho para $DR = 6n$. Resultados experimentais foram obtidos para as poucas soluções extensíveis existentes usando CRT [1] e MRC [3] implementando um conversor reverso para o conjunto de módulos $\{\mathbf{2^{2n}}, \mathbf{2^n \pm 3}, 2^n \pm 1\}$. Além disso, as arquiteturas apresentadas em [9], que usam a nova técnica CRT-I, também são implementadas para o conjunto de módulos $\{\mathbf{2^{n+\beta}}, \mathbf{2^n \pm 3}, 2^n \pm 1\}$, para $\beta = 0$ e $\beta = n$. Embora exista no estado da arte mais conversores reversos dedicados para conjuntos de módulos com $DR = 6n$, esses não são considerados porque eles apresentam piores resultados do que os conversores apresentados em [17] e [21], como mostrado em [36].

A Figura 4.4 apresenta os resultados obtidos para potência, a área e o atraso do circuito, bem como o produto da área e do atraso, conhecido como ADP (*Area-Delay-Product*). Como esperado, as estruturas de conversão dedicadas [17] e [21] apresentam atrasos e área menores. No entanto, é importante notar que elas não podem ser estendidas a conjuntos de módulos com *DRs* maiores que $6n$ e $8n+1$ *bits*, respectivamente, que é o principal objetivo desse trabalho.

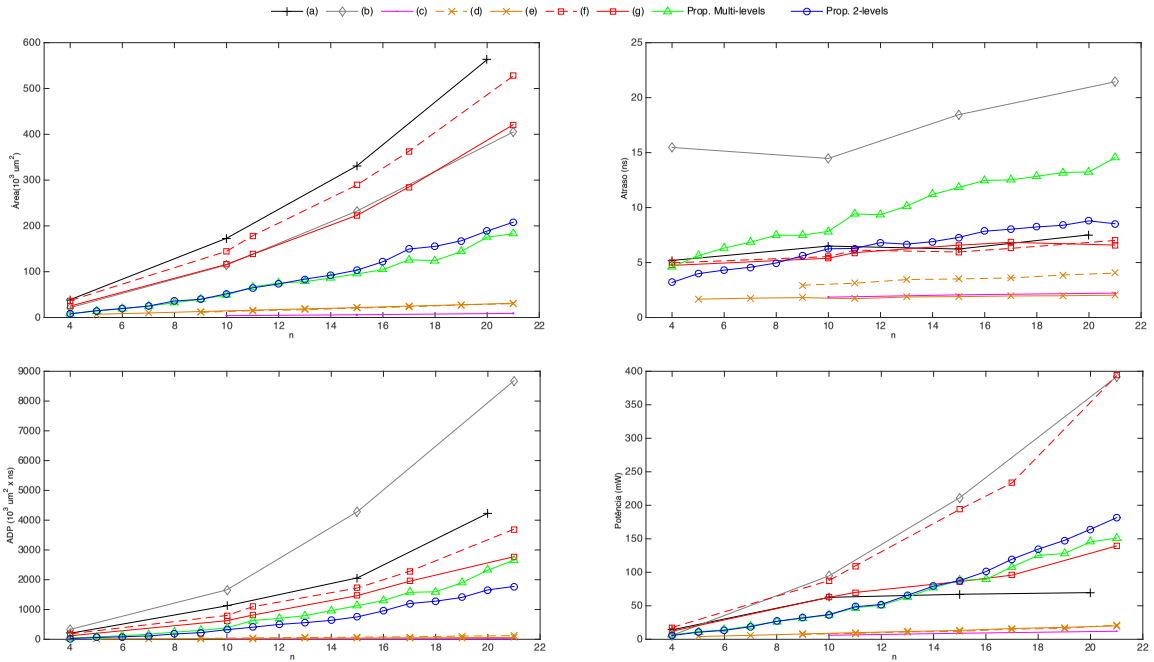


Figura 4.4: *Resultados Experimentais dos conversores reversos para o conjunto de módulo com* $DR = 6n$ bits*:* $\{2^{2n}, 2^n \pm 3, 2^n \pm 1\}$ *baseados em* ***(a)*** *CRT [1] e* ***(b)*** *MRC [3];* ***(c)*** $\{2^{2n}, 2^n \pm 1, 2^{2n} + 1\}$ *[17];* ***(d)*** $\{2^n, 2^n \pm 1, 2^n \pm 2^{\frac{n+1}{2}} + 1, 2^{n+1} + 1\}$ *e* $\{2^n, 2^n \pm 1, 2^n \pm 2^{\frac{n+1}{2}} + 1, 2^{n-1} + 1\}$ *[21] com* $\beta = 0$*;* ***(e)*** $\{2^{2n}, 2^n \pm 1, 2^n \pm 2^{\frac{n+1}{2}} + 1\}$ *[21] com* $\beta = n$*;* ***(f)*** $\{2^n, 2^n \pm 3, 2^n \pm 1\}$ *[9] e* ***(g)*** $\{2^{2n}, 2^n \pm 3, 2^n \pm 1\}$ *[9], ambos baseados no Novo CRT-I;* $\{2^{2n}, 2^n \pm 3, 2^n \pm 1\}$ *para as duas propostas* **Two-level** *e* **Multi-Level**.

Os resultados de área obtidos considerando as abordagens de conversão tradicionais baseadas no CRT [1] demonstram baixa eficiência nos valores de ADP. A solução MRC [3] mostra uma pequena melhoria em atraso e, consequentemente, altos valores de ADP. Quando comparadas com as abordagens reversas genéricas e escaláveis apresentadas em [9] ($\beta = 0$ e $\beta = n$), as abordagens aqui propostas sugerem uma redução significativa da área, na ordem de 54% quando $\beta = n$. Com relação ao atraso, as soluções multi-nível propostas são claramente mais lentas do que as arquiteturas apresentadas em [9], sendo em média 79% mais lentas. Por outro lado, na abordagem *Two-level*, o aumento de atraso é apenas de 16% em relação a [9]. Este aumento de atraso menos significativo é alcançado dado o paralelismo adicional explorado pela abordagem *Two-level*. Apesar deste aumento de atraso, a redução da área é mais do que suficiente para compensar o aumento de atraso se o valor de ADP for considerado como métrica de eficiência. Para o ADP, a abordagem *Multi-level* permite uma melhoria de eficiência de 16%, enquanto a abordagem *Two-level* permite uma melhoria de 41% em relação a [9] com $\beta = n$ e 71% em comparação com a técnica CRT [1]. Entre as duas abordagens propostas, a solução *Two-level* é claramente mais rápida que a *Multi-level*, considerando o intervalo analisado. Embora, a abordagem *Multi-level* exiba área menor para alguns casos particulares, não é suficiente para compensar a degradação do atraso quando o ADP é considerado. Em termos de potência, as propostas possuem valores semelhantes. Em comparação com as arquiteturas escaláveis [9], para $\beta = 0$, e o MRC, as abordagens propostas têm menor potência.

Dado os resultados acima apresentados, a análise a seguir só considera: (*i*) A arquitetura apresentada em [9], que mostrou ser a única implementação eficiente no estado da arte para grandes *DRs* e valores de *n*; e (*ii*) a abordagem *Two-level* aqui proposta, dado os melhores resultados em geral quando comparados com a abordagem *Multi-level*.

Para avaliar melhor a escalabilidade das abordagens de conversão propostas, é considerado um conjunto de módulos com $DR = 10n$ *bits*. Até a publicação de [10], nenhuma estrutura de conversão dedicada tinha sido proposta para conjuntos de módulos acima de $8n$ *bits*. A arquitetura escalável apresentada em [9] para $\beta = n$ foi escolhida para esta comparação porque tem os melhores resultados de ADP conforme apresentado na Figura 4.4. A Tabela 4.2 apresenta os resultados obtidos dos conversores para conjuntos de módulos com $DR = 10n$ *bits* na forma de $\{\mathbf{2^{2n}}, \mathbf{2^n \pm k_3}, \mathbf{2^n \pm k_2}, \mathbf{2^n \pm k_1}, 2^n \pm 1\}$, $7 \leq n \leq 13$. Os k_j são valores ímpares escolhidos para definir o conjunto mais balanceado da arquitetura apre-

Tabela 4.2: *Desempenho dos conversores reversos para DR* $= 10n$ bits

	$n=7$	$n=9$	$n=11$	$n=13$
Área ($10^3 \mu m^2$) [9]	74	331	475	610
Área ($10^3 \mu m^2$) 2-levels	91	152	197	240
Área reduction	-23%	**54%**	**59%**	**61%**
Atraso (ns) [9]	5.36	6.28	6.67	7.40
Atraso (ns) 2-levels	5.23	5.97	6.41	6.82
Speedup	**1.02**	**1.05**	**1.04**	**1.08**
ADP ($10^3 \mu m^2$) [9]	397	2079	3168	4514
ADP ($10^3 \mu m^2$) 2-levels	476	907	1263	1637
ADP reduction	-20%	**56%**	**60%**	**64%**
Potência (mW) [9]	58	141	168	206
Potência (mW) 2-levels	64	106	131	159
Potência reduction	-11%	**25%**	**22%**	**23%**

sentada em [9] com $\beta = n$ e a abordagem *Two-level* aqui proposta: $k_1 = 3$, $k_2 = 9$, $k_3 = 15$ para $n = 7, 11, 13$, e $k_1 = 3$, $k_2 = 9$, $k_3 = 21$ para $n = 9$. Esses resultados sugerem que para valores menores de n, a solução apresentada em [9] com $\beta = n$ exibe melhores resultados para as métricas de área, ADP e potência. No entanto, a solução aqui proposta melhora com o aumento de n, apresentando reduções de área e potência de até 61% e 25%, respectivamente, e uma aceleração de $1,08$, resultando em melhorias no valor de ADP de até $2,7$ vezes, no intervalo analisado. Mais importante ainda, a abordagem de conversão *Two-level* proposta pode escalar adequadamente em termos de *DR* e na largura do canal (2^n).

4.3.2 Resultados do CRTf Proposto e Análise do Impacto dos Métodos de Otimização

A arquitetura *Multi-level* foi descartada para essa comparação porque, como demonstrado na subseção anterior, ela teve pior desempenho que a proposta *Two-level*. As informações de desempenho da arquitetura *Two-level* da seção anterior são aqui repetidos como "*Two-level* Original" para facilitar a avaliação do impacto das técnicas de otimização. Para essa avaliação as arquiteturas $DR = 6n$ comparadas foram:

i) $\{2^{2n}; 2^n \pm 1; 2^n \pm 3\}$ para o CRT [1], *Two-level* Original, *Two-level* otimi-

zado, CRTf Original [12], e o CRTf otimizado;

ii) $\{2^{2n}; 2^n \pm 1; 2^n \pm 2^{\frac{n+1}{2}} + 1\}$ de [21];

iii) $\{2^{2n}; 2^n \pm 1; 2^{2n} + 1\}$ apresentado em [17], $n = 4, ..., 21$.

Como pode ser notado na Figura 4.5, a implementação original [12] do CRTf já apresenta atraso melhor que o CRT [1] e o *Two-level* Original. A técnica de otimização das multiplicações por constante introduz melhorias significativas em atraso e área para ambas as implementações analisadas. Comparado com *Two-level* Original, a implementação *Two-level* otimizada teve uma redução de área média de 37% e aceleração máxima de $1,81$ vezes para $n = 20$. Com a melhoria dos multiplicadores a implementação CRTf otimizada teve redução média da área de 88% e uma aceleração máxima de $1,41$ vezes para $n = 15$, se comparado ao CRTf original [12]. Comparando as propostas, o CRTf otimizado tem vantagem em atraso apenas para $n = 15$ se comparado com o *Two-level* otimizado.

Para avaliar melhor as arquiteturas otimizadas para maior faixa dinâmica, também comparou-se as implementações relevantes para $DR = 8n$, apresentadas na Tabela 4.3. Foram excluídas [1] e o CRTf original [12] porque eles mostraram os piores resultados ADP para $DR = 6n$. A arquitetura apresentada em [17] também está excluída neste momento porque não pode ser estendido para $DR > 6n$. Assim para $DR = 8n$ foram comparadas:

i) os conversores reversos específicos para $\{2^{3n}, 2^n - 1, 2^n + 1, 2^n - 2^{\frac{n+1}{2}} + 1, 2^n + 2^{\frac{n+1}{2}} + 1, 2^{n\pm1} + 1\}$ [21];

ii) o conjunto de módulos $\{2^{2n}, 2^n \pm 1, 2^n \pm 3, 2^n \pm 9\}$ para o *Two-level* Original, e as implementações otimizadas do *Two-level* e CRTf.

As melhores abordagens para $DR = 8n$ são o conjunto de módulo específico [21] em relação à área de circuito e a implementação da *Two-level* otimizada em relação ao atraso.

Para $DR = 10n$, foi considerado o conjunto de módulos $\{2^{2n}, 2^n \pm 1, 2^n \pm 3, 2^n \pm 9, 2^n \pm k_3\}$, com $k_3 = 21$ para $n = 9$, e $k_3 = 15$ para $n = 11, 13$. As únicas abordagens comparadas foram aquelas capazes de alcançar $DR = 10n$. A Tabela 4.3 mostra que para $DR = 10n$, a melhor proposta com relação à área e ao

atraso é a implementação *Two-Level* otimizada, que teve uma redução média da área de 27% e uma aceleração máxima de 1,40 vezes para $n = 13$ em comparação com o *Two-Level* Original.

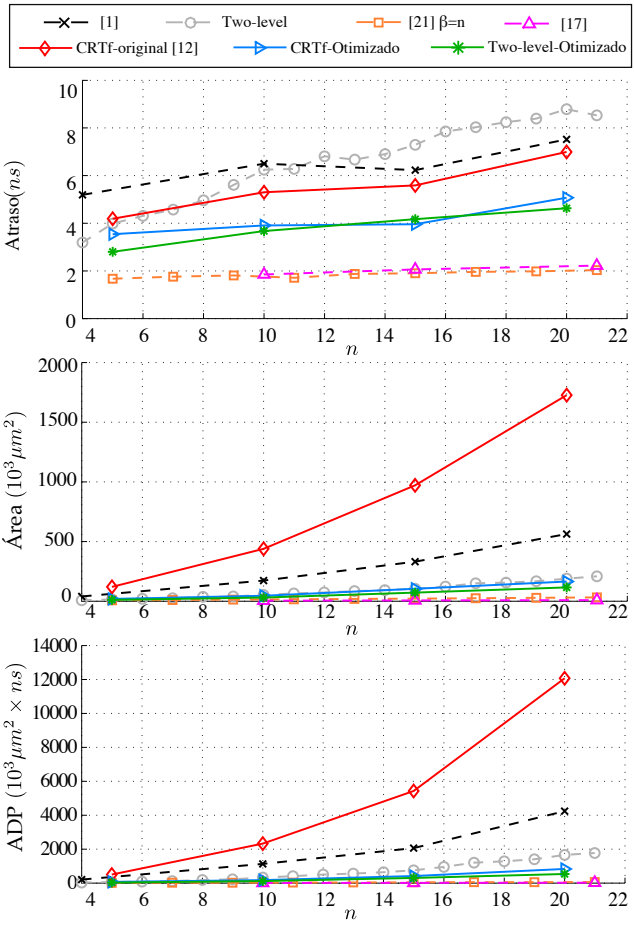


Figura 4.5: *Resultados Experimentais obtidos para DR* $= 6n$.

Tabela 4.3: *Resultados Experimentais para DR* $= 8n$.

	$(8n \pm 1)$-bits [21]			$(8n)$ *bits* Two-level Original			$(8n)$ *bits* Two-level Otimizado			$(8n)$ *bits* CRTf Otimizado		
n	9	11	13	9	11	13	9	11	13	9	11	13
DR	51	89	103	72	88	104	72	88	104	72	88	104
Atraso	4.11	4.87	4.51	5.93	6.31	6.79	**3.87**	**4.39**	**4.40**	5.14	4.98	5.03
Área $(10^3 \mu m^2)$	**16.0**	**12.1**	**24.1**	84	119	180	66.51	95.38	130.02	123.07	165.79	240.03

Tabela 4.4: *Resultados Experimentais para DR* $= 10n$.

	$(10n)$ *bits* Two-level Original			$(10n)$ *bits* Two-level Otimizado			$(10n)$ *bits* CRTf Otimizado		
n	9	11	13	9	11	13	9	11	13
DR	90	110	130	90	110	130	90	110	130
Atraso	5.97	6.41	6.82	**4.60**	**4.80**	**4.89**	5.35	5.86	5.77
Área $(10^3 \mu m^2)$	152	197	240	**105.54**	**141.29**	**189.74**	204.63	290.12	381.11

4.4 Conclusões

Neste capítulo, foi proposto um novo método para projetar conversores reversos RNS para conjuntos de módulos longos. Foram propostas duas abordagens que reduzem a seleção de peso modular dos termos multiplicativos associados às entradas. A primeira abordagem é baseada em estágios iterativos usados para reduzir a complexidade da etapa final do conversor. A segunda abordagem minimiza o número de estágios iterativos necessários na conversão para apenas dois níveis, com custo mínimo de área em comparação com a solução de vários níveis. Os resultados experimentais sugerem que as abordagens propostas permitem reduções de área significativas em comparação com o estado da arte, para estruturas de conversão reversa com *DR* genérico. Dadas as métricas de atraso semelhantes entre o estado da arte e a abordagem proposta *Two-level*, podem ser alcançadas melhorias de até $2,7$ vezes no valor de ADP, considerando um conjunto de módulos com uma faixa dinâmica de $10n$ *bits*. Mais importante ainda, os resultados obtidos sugerem que as abordagens propostas, em particular a abordagem *Two-level*, podem escalar eficientemente com conjuntos de módulos e n maiores.

Além disso, ainda foi avaliado o impacto do método de otimização de multiplicadores modulares por constantes apresentado no capítulo anterior aplicado à proposta do *Two-Level* (módulo $2^n - 1$) e CRTf usando conjuntos de módulos ge-

néricos balanceados (módulo 2^n).Os resultados experimentais do *Two-Level* com as multiplicações otimizadas para $DR = 6n$, $DR = 8n$, e $DR = 10n$ apresentaram, respectivamente, aceleração de $1,81$, $1,54$, e $1,40$ vezes e redução média de área de 37%, 23%, e 27% comparado com a implementação do *Two-Level* sem otimizações. Já o CRTf apresentou redução média de área de 88% e aceleração de $1,41$ vezes para $DR = 6n$ comparado com a implementação original do CRTf no estado de arte.

Apesar dos ganhos significativos com as novas arquiteturas apresentadas neste capítulo e a aplicação das técnicas de otimização propostas no Capítulo 3, ainda assim é preciso procurar outras alternativas para melhorar mais os conversores reversos com ampla faixa dinâmica. Dessa forma, o próximo capítulo foca esforços na definição de um conversor reverso dedicado e otimizado com ampla faixa dinâmica.

CAPÍTULO 5

Proposta do Conversor Reverso Dedicado com Faixa Dinâmica de $9n$ *bits*

Apesar dos avanços nos conversores reversos dinâmicos apresentado no Capítulo 4 e da possibilidade de estender para elevadas faixas dinâmicas, os conjuntos de módulos dedicados apresentam vantagens porque oferecem em geral operações modulares mais simples na conversão reversa, em comparação com as abordagens genéricas [9, 10, 12]. Isso permite a implementação de conversores reversos mais eficientes e o aumento da aplicabilidade de unidades RNS na solução de processamento digital de sinais. Além disso, as técnicas de otimização propostas no Capítulo 3 apoiam na análise dos novos conjuntos e na otimização final do sistema.

Dessa forma, neste capítulo é apresentada a principal contribuição da tese na área de conjuntos de módulos e conversores reversos dedicados, que é a proposta de um novo conjunto de módulos dedicados baseado na idéia apresentada em [21]. Esse conjunto de módulos permite conversões de RNS para binário usando unidades modulares na forma de $\{2^{6n}-1\}$, alcançando o conversor reverso mais eficiente para $DR \simeq 9n$ e com a maior faixa dinâmica do estado da arte.

5.1 Conversor Reverso RNS para conjunto de módulos com faixa dinâmica de $9n$ *bits*

Como apresentado no Capítulo 2, no trabalho [21], foi apresentado o conversor reverso usando o algoritmo CRT para o conjunto de módulo $\{2^{n+\beta}, 2^n - 1, 2^n + 1, 2^n - 2^{\frac{n+1}{2}} + 1, 2^n + 2^{\frac{n+1}{2}} + 1\}$, alcançando $DR = 5n + \beta$. O valor de β pode ser estendido de 0 até $2n$ mantendo o desempenho de todo o sistema RNS como demonstrado em [21]. Além disso, os autores afirmam que um módulo adicional $\{2^{n+1}+1\}$ ou $\{2^{n-1}+1\}$ também pode ser incluído no conjunto de módulos obtendo um DR até $(8n+1)$ ao custo do aumento do atraso. No entanto, os DRs alcançados demonstram o melhor desempenho no estado da arte para conversores reversos dedicados com $DR > 5n$.

Para o conjunto de módulos com $DR \simeq 7n$, eficiente com relação ao atraso, $\{m_1, m_2, m_3, m_4, m_5\} = \{2^{3n}, 2^n - 1, 2^n + 1, 2^n - 2^{\frac{n+1}{2}} + 1, 2^n + 2^{\frac{n+1}{2}} + 1\}$, a faixa dinâmica é igual ao produto dos cinco módulos do conjunto definido $\left(M = \prod_{i=1}^{5} m_i\right)$, $\hat{m}_i = M/m_i$, e $\left|\hat{m}_i^{-1}\right|_{m_i}$ representa a multiplicativa inversa para $\hat{m}_i$ no que diz respeito ao módulo m_i. O valor representado em RNS pelos cinco valores residuais pode ser convertido de volta para binário (X) utilizando o CRT:

$$X = \left| \sum_{i=1}^{5} \hat{m}_i \left|\hat{m}_i^{-1}\right|_{m_i} R_i \right|_M, \tag{5.1}$$

onde R_i, $0 \leq i \leq 5$, denota o resíduo para m_i, para o qual a representação em vetor de n-bit é $r_{i(n-1)}$, ..., r_{i0}. Eq. (5.1) pode ser reescrita como novo CRT em [21]:

$$X = \left| \sum_{i=1}^{5} v_i \right|_{\hat{m}_1} m_1 + R_1 = \left| \sum_{i=1}^{5} |V_i R_i|_{\hat{m}_1} \right|_{\hat{m}_1} m_1 + R_1, \tag{5.2}$$

onde $\hat{m}_1 = 2^{4n} - 1$, $V_1 = \left|\hat{m}_1^{-1}\right|_{m_1} \frac{(\hat{m}_1+1)}{m_1}$ e $V_i = \left|\hat{m}_i^{-1}\right|_{m_i} \frac{\hat{m}_i}{m_1}$ para $2 \leq i \leq 5$. Duas propriedades são usadas para implementar a arquitetura do conversor reverso: *i)* os multiplicadores modulares $|V_i R_i|_{\hat{m}_1}$ podem ser implementados usando operações *ROL*; *ii)* o módulo $2^{bn} - 1$ de um número negativo é realizado por um fator de correção. Portanto, a Eq. (5.2) é rescrita como em [21]:

$$X = \left| \sum_{i=1}^{7} W_{i(g_i)} + Z \right|_{\hat{m}_1} m_1 + R_1, \tag{5.3}$$

onde $W_{i(g_i)}$ são os vetores de $4n$-bit resultando de uma multiplicação modular depois das operações *ROL*, e Z o fator corretor. O parâmetro g_i representa o número de vetores de $4n$ *bits* por canal que depende dos padrões binários dos "uns" associados ao V_i correspondente. A soma de todos as contribuições dos vetores serão denotadas a partir de agora como $g = \sum_{i=1}^{5} g_i$, que para [21] é $g = 9$.

A principal contribuição do trabalho apresentado em [21] é baseado na separação de $\hat{m}_1 = \prod_{i=2}^{5} m_i = 2^{4n} - 1$ em um conjunto de módulos $\{\overbrace{m_2}, \overbrace{m_3}, m_4, m_5\}$ com um cumprimento máximo de *bits* de n. Dessa forma, $\hat{m}_1 = \overbrace{(2^n - 1)}^{m_2}\overbrace{(2^n + 1)}^{m_3}$ $\overbrace{(2^n - 2^{\frac{(n+1)}{2}} + 1)}^{m_4}\overbrace{(2^n + 2^{\frac{(n+1)}{2}} + 1)}^{m_5}$, alcança para $m_1 = 2^{3n}$ uma faixa dinâmica de

$DR \simeq 7n$. Portanto, o $\hat{m}_1$ permite o uso de um compressor módulo $2^{4n}-1$ como é apresentado na Figura 5.1, que é mais eficiente do que as unidades modulares genéricas. Por exemplo, a ideia de um conjunto de módulos genéricos para amplos *DR*s é apresentada em [9] e usa operações modulares complexas que não estão na forma $2^{b\times n}-1$ para derivar o valor binário final. Porém, os conjuntos com $\hat{m}_1 = 2^{b\times n}-1$, $b > 4$ não tem sido explorados até o momento.

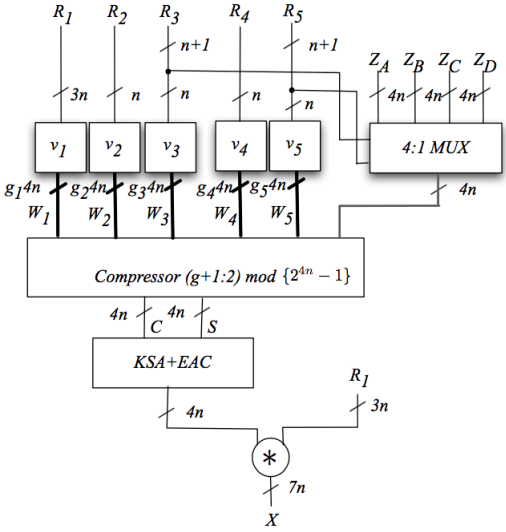


Figura 5.1: *Conversor reverso com DR = 7n.*
Fonte: [21]

5.2 Proposta do conjunto de módulos com $DR \simeq 9n$

O conjunto de módulos com $DR \simeq 9n$ aqui proposto é baseado na extensão do conjunto de módulos para aplicações com ampla faixa dinâmica usando operações modulares no formato $2^{b\times n}-1$, quando $b > 4$. Análises preliminares têm demonstrado que $\hat{m}_1 = 2^{b\times n}-1$, com $b > 7$, não pode ser dividido em módulos com tamanho máximo de *n bits* usando a mesma abordagem apresentada

em [21], resultando em conjuntos de módulos desequilibrados ou não co-primos. Entretanto, é possível obter um método similar como apresentado em [21], derivando conversores reversos eficientes com amplo *DR*, quando $\hat{m}_1 = 2^{6n} - 1$.

A extensão do conjunto de módulos proposto é $\{m_1, m_2, \ldots, m_9\}$, onde $m_1 = 2^{n+\beta}$ com $\hat{m}_1 = \prod_{i=2}^{9} m_i = 2^{6n} - 1$. Com a finalidade de obter o comprimento dos módulos até *n bits*, o valor $\hat{m}_1 = 2^{6n} - 1$ é dividido em oito módulos como mostrado na Figura 5.2, válido para $n = 4k + 6$ com $k \in \mathbb{N}$ (onde $\mathbb{N}$ é o conjunto dos números naturais incluindo o zero), ou seja, válido para $n = 6, 10, 14, \ldots$. Similar ao trabalho [21], pode-se considerar o valor $\beta = 2n$ para estender verticalmente o conjunto de módulos 2^n para 2^{3n} resultando em um *DR* igual a $M = 2^{3n} \times (2^{6n} - 1)$, em outras palavras, $DR \simeq 9n$-bit. Aplicando o conjunto de módulos propostos à Eq.(5.2), temos:

$$X = \left| \sum_{i=1}^{9} v_i \right|_{\hat{m}_1} m_1 + R_1 = \left| \sum_{i=1}^{9} |V_i R_i|_{\hat{m}_1} \right|_{\hat{m}_1} m_1 + R_1, \tag{5.4}$$

onde os valores de V_i que dependem de $\hat{m}_i$ e a multiplicativa inversa são requeridos. Os valores de $\hat{m}_i$ são obtidos pela aplicação de $\hat{m}_i = \frac{(2^{9n} - 2^{3n})}{m_i}$.

$$\hat{m}_1 = 2^{6n} - 1 = \begin{cases} 2^{3n} - 1 & = \begin{cases} 2^{\frac{3n}{2}} - 1 & = \begin{cases} m_2 = 2^{\frac{n}{2}} - 1, \\ m_3 = 2^n + 2^{\frac{n}{2}} + 1, \end{cases} \\ 2^{\frac{3n}{2}} + 1 & = \begin{cases} m_4 = 2^{\frac{(n+2)}{2}} + 2^{\frac{n}{2}} + 2^1 + 1, \\ m_5 = \frac{1}{3}(2^{\frac{n}{2}} - 2)^2 + 2^{\frac{n}{2}} - 1, \end{cases} \end{cases} \\ 2^{3n} + 1 & = \begin{cases} 2^{\frac{3n}{2}} - 2^{\frac{3n+2}{4}} + 1 & = \begin{cases} m_6 = 2^{\frac{n}{2}} + 2^{\frac{(n+2)}{4}} + 1, \\ m_7 = 2^n - 2^{\frac{(3n+2)}{4}} + 2^{\frac{n}{2}} - 2^{\frac{n+2}{4}} + 1, \end{cases} \\ 2^{\frac{3n}{2}} + 2^{\frac{3n+2}{4}} + 1 & = \begin{cases} m_8 = 2^{\frac{n}{2}} - 2^{\frac{(n+2)}{4}} + 1, \\ m_9 = 2^n + 2^{\frac{(3n+2)}{4}} + 2^{\frac{n}{2}} + 2^{\frac{(n+2)}{4}} + 1, \end{cases} \end{cases} \end{cases}$$

Figura 5.2: *Conjunto de módulos derivado de* $\hat{m}_1 = 2^{6n} - 1$.

O Algoritmo de Euclides pode ser aplicado para obter as multiplicativas inversas $\left|\hat{m}_i^{-1}\right|_{m_i}$ para o conjunto de módulos $\{m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9\}$, considerando a condição $\left|(\hat{m}_i) \times (\hat{m}_i^{-1})\right|_{m_i} = 1$ com $(i = 1, 2 \ldots 9)$. Provas de $\left|\hat{m}_1^{-1}\right|_{m_1} = |-1|_{2^{3n}}$ e $\left|\hat{m}_2^{-1}\right|_{m_2} = \frac{1}{6}(2^{\frac{n}{2}} - 2) + 2^{\frac{n-4}{2}}$ são mostradas na Eq. (5.5) e (5.6), respectivamente. Em seguida, é apresentado um valor numérico para o

conjunto de módulos proposto e $n = 6$.

$$\left| (\overbrace{2^{6n}}^{=0} - 1)(-1) \right|_{2^{3n}} = 1. \tag{5.5}$$

$$\left| \overbrace{(\left|2^{3n}\right|_{2^{\frac{n}{2}}-1})}^{=1} \overbrace{(\left|2^n + 2^{\frac{n}{2}} + 1\right|_{2^{\frac{n}{2}}-1})}^{=3} \overbrace{(\left|2^{\frac{3n}{2}} + 1\right|_{2^{\frac{n}{2}}-1})}^{=2} \right.$$

$$\left. \overbrace{(\left|2^{3n} + 1\right|_{2^{\frac{n}{2}}-1})}^{=2} \left(\frac{1}{6} (\overbrace{\left|2^{\frac{n}{2}} - 2\right|_{2^{\frac{n}{2}}-1}}^{=-1}) + 2^{\frac{n-4}{2}} \right) \right|_{2^{\frac{n}{2}}-1} =$$

$$= \left| 12(\frac{-1}{6} + 2^{\frac{n-4}{2}}) \right|_{2^{\frac{n}{2}}-1} = \left| -2 + 3\overbrace{(4)(2^{\frac{n-4}{2}})}^{=1} \right|_{2^{\frac{n}{2}}-1} = 1. \tag{5.6}$$

Exemplo: O conjunto modular para o caso particular de $n = 6$ é $\{2^{18}, 7, 73, 27, 19, 13, 37, 5, 109\}$, os valores de $\frac{\hat{m}_i}{m_1}$, multiplicativas inversas $\left|\hat{m}_i^{-1}\right|_{m_i}$, e V_i são apresentadas na Tabela 5.1. Para o valor binário $X = 786680833$, os valores residuais $\{R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9\}$ são $\{248833, 1, 56, 4, 7, 3, 5, 3, 38\}$. Aplicando a Eq. (5.4):

$$X = \left| \sum_{i=1}^{9} V_i R_i \right|_{\hat{m}_1} m_1 + R_1 = |\overbrace{3489398783}^{|V_1 R_1|_{\hat{m}_1}} + \overbrace{29451204315}^{|V_2 R_2|_{\hat{m}_1}} +$$

$$+ \overbrace{1882725390}^{|V_3 R_3|_{\hat{m}_1}} + \overbrace{43267818685}^{|V_4 R_4|_{\hat{m}_1}} + \overbrace{61485847605}^{|V_5 R_5|_{\hat{m}_1}} + \overbrace{21144454380}^{|V_6 R_6|_{\hat{m}_1}} +$$

$$+ \overbrace{59433060960}^{|V_7 R_7|_{\hat{m}_1}} + \overbrace{13743895347}^{|V_8 R_8|_{\hat{m}_1}} + \overbrace{40979504475}^{|V_9 R_9|_{\hat{m}_1}} |_{\hat{m}_1} m_1 + R_1 =$$

$$= |\overbrace{274877909940}^{3000}|_{68719476735} * 262144 + 248833 = 786680833. \tag{5.7}$$

As multiplicações $v_i = |V_i R_i|_{\hat{m}_1}$, $1 \leq i \leq 9$, são obtidas usando operações *ROL* da mesma forma como em [21], apresentado na Figura 5.3. Para esse caso de

Tabela 5.1: *Exemplo numérico para $n = 6$, com a extensão proposta, $\beta = 2n$, (DR de 54-bits).*

i	m_i	$\frac{\hat{m}_i}{m_1}$	$\left\|\hat{m}_i^{-1}\right\|_{m_i}$	V_i
1	262144	–	262143	68719214591
2	7	9817068105	3	29451204315
3	73	941362695	60	56481761700
4	27	2545165805	11	27996823855
5	19	3616814565	16	57869033040
6	13	5286113595	10	52861135950
7	37	1857283155	36	66862193580
8	5	13743895347	2	27487790694
9	109	630453915	39	24587702685

estudo, o número final de vetores de $6n$-bit, $W_{i(g_i)}$, é $g = \sum_{i=1}^{5} g_i = 36$. Um vetor adicional de $6n$-bit, Z, também é incluído. Sendo um valor selecionado pelas combinações r_{3n} e r_{9n}. Assim, a adição modular de $g+1$ termos é obtida por meio de um árvore *CSA* de $\{2^{6n}-1\}$ (ou um compressor $(g+1):2$) e uma soma final, que é implementada usando um *KSA* e *EAC* da mesma forma que em [21]. O *KSA* é aqui considerado uma vez que é uma das estruturas de somadores que implicam em um melhor desempenho [32]. Finalmente, o termo $\left|\sum_{i=1}^{9} V_i R_i\right|_{\hat{m}_1}$ é concatenado com R_1 para derivar o X.

Observe que o valor de Z é obtido considerando que nem todos os resíduos R_i têm o mesmo comprimento de *bits*. Assim, quando R_i tem menos de n *bits*, ele é completado com zeros para gerar o produto por V_i com $6n$ *bits*. Caso contrário, quando R_i tem mais de n *bits* (caso de R_3 e R_9), o bit extra é usado para determinar o fator Z no multiplexador, o qual será inserido como outro termo no compressor, como mostrado na Figura 5.3.

5.3 Análise Teórica do Conjunto de Módulos Proposto

O conjunto de módulos aqui proposto foi o primeiro módulo de RNS específico na literatura que inclui nove módulos, juntamente com um intervalo dinâmico de $9n$ *bits*. A característica mais interessante desse conjunto de módulos é que o produto de todos os módulos, exceto 2^{3n}, está na forma $2^{6n}-1$ que resulta apenas em uma árvore *CSA* seguida de um somador de módulo *EAC* como mostrado na Figura 5.3. Em comparação com o conjunto de seis módulos com

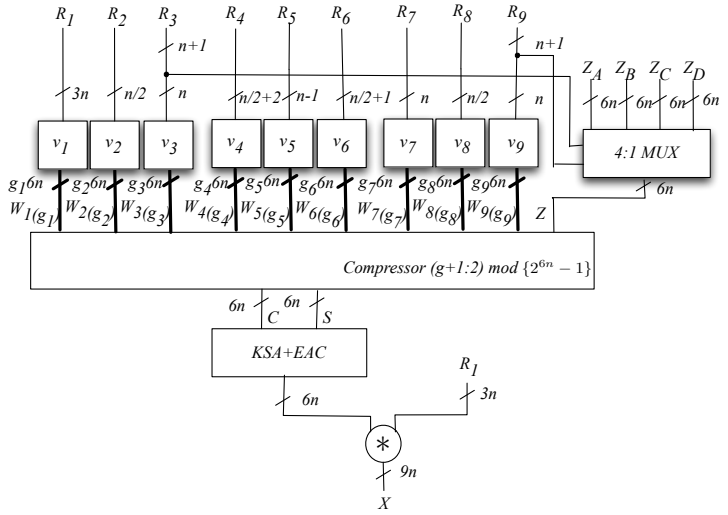


Figura 5.3: *Diagrama de bloco do conversor reverso proposto com DR* $\simeq 9n$.

$DR = 8n \pm 1$ *bits* introduzido em [21], o conjunto de módulos proposto pode fornecer melhores unidades RNS, uma vez que, o conversor reverso para o conjunto proposto é tão mais simples do que o conversor reverso para o conjunto de seis módulos de [21], além do intervalo dinâmico ser distribuído entre 9 módulos em vez de 6, resultando em canais com menor largura de *bits*. Portanto, o módulo proposto com nove módulos é mais rápido que o conjunto de seis módulos de [21] para o mesmo requisito de faixa dinâmica.

O conjunto proposto não está totalmente balanceado, e um de seus módulos é bem maior do que outros. No entanto, esse desequilíbrio não conduzirá à degradação do desempenho da unidade aritmética RNS se os somadores de prefixo paralelo (PPA - *Paralel-Prefix Adder*) forem usados nos canais aritméticos RNS. Como o atraso dos PPAs cresce logaritmicamente com o aumento da largura de *bits*, resultando em uma pequena adição de atraso para grandes operandos. Para investigar este problema, semelhante ao método usado em [17] para comparar o desempenho aritmético de diferentes conjuntos de módulos especiais, os atrasos de adição em diferentes módulos do conjunto proposto são calculados e mostrados na Tabela 5.2. Um *KSA* é considerado para o módulo 2^{3n} uma vez

Tabela 5.2: *Atraso "unit-gate" para soma nos canais modulares do conjunto proposto.*

Módulo	Largura (*bits*)	Atraso (*unit gate*)
m_1	$3n$	$\simeq 2\log_2 n + 6.17$
m_2	$\frac{n}{2}$	$2\log_2 \frac{n}{2} + 3$
m_3	$n+1$	$2\log_2 n + 7$
m_4	$\frac{n}{2}+2$	$2\log_2 \frac{n+2}{2} + 7$
m_5	$n-2$	$2\log_2(n-2) + 7$
m_6	$\frac{n}{2}+1$	$2\log_2 \frac{n}{2} + 7$
m_7	n	$2\log_2(n-1) + 7$
m_8	$\frac{n}{2}$	$2\log_2 \frac{n-2}{2} + 7$
m_9	$n+1$	$2\log_2 n + 7$

que a adição para módulos na forma 2^k só precisa de um somador regular, ignorando os *carrys*. O atraso baseado no método "unit-gate" para um somador Kogge-Stone é $2\log_2(k)+3$ para operados de *k bits* [37]. Além disso, somadores baseados em prefixo rápido para módulos 2^k-1 de [38] são considerados para módulo $2^{n/2}-1$ com o atraso de $2\log_2(k)+3$. Finalmente, para realizar a adição em outro módulo, o somador prefixo paralelo para modulo genérico de [39] é considerado atraso de $2\log_2(k-1)+7$ em que k é a largura de *bits* do módulo. Pode-se ver que, embora o módulo 2^{3n} seja o módulo maior e não balanceado no conjunto, não é o caso dele aumentar o atraso da unidade aritmética RNS total, e os outros módulos com largura de $(n+1)$ *bits* tem atraso semelhantes, conforme explicado em [21].

5.4 Análise Teórica dos Conversores Reversos

Para melhor entender o atraso e a área do conversor reverso proposto, um modelo *unit-gate* é derivado para estimar o custo do circuito e o atraso do caminho crítico. Neste modelo, cada porta monotônica de duas entradas (por exemplo, "E" ou "OU") é considerada como uma unidade de área e atraso. Portanto, uma porta XOR/XNOR conta como duas unidades de área e atraso e, para simplificar, os requisitos de tempo e área do inversor (porta NOT) não são considerados [40]. É importante notar que o modelo teórico explica as características arquitetônicas dos conversores reversos usando uma avaliação agnóstica tecnológica. No entanto, a abordagem apresentada no Capítulo 3 tem um impacto significativo na

redução da profundidade dos compressores, diminuindo ainda mais o custo e o atraso no nível de implementação. Assim, a estimativa resultante será validada usando implementações VLSI.

Com base no modelo *unit-gate*, um multiplexador 2:1 tem 2 unidades de atraso e 3 unidades de área, enquanto um multiplexador 4:1, tem 4 unidades de atraso e 11 unidades de área. Um somador completo (FA - *Full-Adder*) contribui com 4 unidades de atraso e custo de 7 unidades em área. O somador multi-operando é organizado como um compressor usando conexões *EAC*. Este circuito requer aproximadamente $\log_{1.5} k \approx 1.7 \log_2 k$ estágios, onde k é o número de operandos a serem adicionados, com um FA de um 1 bit de atraso por estágio e área de $(k-2)$ FAs. Para ser justo em todas as comparações, os *CPAs* do estado da arte são substituídos por *KSAs* e o modelo para a arquitetura apresentada em [10] é adaptado para considerar o parâmetro g. O convencional *KSA* de n *bits* tem uma contribuição de $4+2\log_2 n$ em atraso e aproximadamente $3n+3n\log_2 n+3$ em área. O *KSA* com *EAC* para soma modular $2^n \pm 1$ é considerado organizado como em [41] com $5+2\log_2 n$ unidades em atraso e aproximadamente $7n+1.5n\log_2 n$ em área.

Por exemplo, é obtida a estimativa para o conversor reverso com $DR = 9n$ *bits*, apresentado na Figura 5.3. Este conversor é composto por um multiplexador $6n$-bit 4:1, um compressor de $6n$ *bits*, organizado como um compressor $((g+1):2)$, e para a adição final um *KSA* com *EAC* de $6n$ *bits*. O atraso do multiplexador não é considerado, uma vez que sua saída está conectada ao último nível do compressor. A área total obtida em unidades de portas é $9n\log_2 n+42ng+89.3n$, com contribuições de: um multiplexador $6n$-bit 4:1 resultando em $6n \times 11 = 66n$; um compressor com $(g+1-2) \times 6n \times 7 = 42ng-42n$; e, finalmente, um *KSA* com *EAC* e área de $7 \times 6n + 1.5 \times 6n\log_2(6n) \approx 65.3n+9n\log_2 n$. As contribuições de atraso, também em unidades de porta, são de $10.2+2\log_2 n$ para o *KSA* com *EAC*, e $4 \times 1.7\log_2(g+1) = 6.8\log_2(g+1)$ para o compressor, resultando em $2\log_2 n+6.8\log_2(g+1)+10.2$. A Tabela 5.3 apresenta a generalização dos resultados acima. Além disso, ela também mostra o atraso e a área de conversores reversos relacionados do estado da arte.

Tabela 5.3: *Atraso e Área do conversor reverso para os vários conjuntos de módulos.*

DR (*bit*)	Atraso (*unit gate*)	Área (*unit gate*)
$5n$ [22]	$2\log_2 n+25$	$6n\log_2 n+252n$
$6n$ [17]	$2\log_2 n+21$	$6n\log_2 n+112n+24$
$6n$ [21]	$2\log_2 n+29$	$6n\log_2 n+308n$
$7n$ [21]	$2\log_2 n+25$	$6n\log_2 n+252n$
$8n-1$ [21]	$6.8\log_2(n+8)+4\log_2 n+42.2$	$3.5n^2+7.5n\log_2 n+325.5n-1.5\log_2 n-77$
$8n+1$ [21]	$6.8\log_2(n+8)+4\log_2 n+38.2$	$3.5n^2+7.5n\log_2 n+325.5n+1.5\log_2 n+70$
$9n$ Proposta	$2\log_2 n+6.8\log_2(g+1)+10.2$	$9n\log_2 n+42ng+89.3n$

5.5 Discussão e Resultados Experimentais

Para comparar as diferentes implementações, cada arquitetura foi descrita em VHDL e sintetizada. A ferramenta utilizada foi a *Synopsys Design Vision* (versão E-2010.12-SP4) para gerar um *netlist* otimizado mapeado para tecnologia *ASIC standard cells* de 90*n*m da empresa UMC [29]. Os parâmetros de compilação foram *"-exact_map -map_effort medium -area_effort medium -power_effort medium"*.

As arquiteturas comparadas com a proposta para $n = 6$, $n = 10$ são:

i) o conjunto de módulos $DR \simeq 6n$ [17];

ii) o conjunto de módulos em [21] com $DR \simeq 6n$, $DR \simeq 7n$ e $DR \simeq 8n+1$;

iii) o conjunto modular $\{2^{2n}, 2^n \pm 1, 2^n \pm 3, 2^n - 5, 2^n + 7, 2^n - 11\}$ com $DR = 9n$ obtido da solução genérica apresentada por [9];

As arquiteturas acima foram escolhidas por possuirem os conjuntos de módulo mais balanceados no estado da arte.

A Tabela 5.4 apresenta os resultados experimentais coletados da síntese. Os conversores reversos foram implementados para $n = 6$ ($DR \approx 54$) e $n = 10$ ($DR \approx 90$). Alguns conjuntos de módulos não permitem determinados valores de n. Assim, para uma comparação justa, o valor de n é escolhido de forma a chegar o mais próximo possível dos *DRs* definidos para comparação (54 e 90). Para $DR = 7n$ [21] não são possíveis valores pares para n, porém, $n = 8$ é apresentado na Tabela 5.4 aplicando uma abordagem linear entre os resultados $n = 7$ e $n = 9$ a fim de esclarecer a comparação.

Com relação aos $DR \approx 9n$, conversor proposto otimizado apresenta uma redução média de área de $81,20\%$ e aceleração de $2,77$ podem ser obtidas com a nossa proposta otimizada em comparação com as melhores soluções existentes no estado da arte para $DR \simeq 9n$ [9]. Apesar de outros conversores reversos aqui analisados exibirem melhor desempenho, a principal contribuição da nossa proposta reside no fato de que é a abordagem de conversor reverso mais eficiente até o momento com o menor número de *bits* por canal resultando em operações aritméticas RNS mais rápidas. Comparando a proposta com e sem otimização é possível verificar uma redução média de área de $47,79\%$ e aceleração média foi de 1.16 vezes.

Como pode ser notado a partir dos resultados na Tabela 5.4, os conversores reversos com menor atraso e área são as topologias apresentadas por [17] e [21]. No entanto, nossa proposta resulta em menor número de *bits* por canal. Por exemplo, para alcançar $DR \approx 90$ a implementação requer $n = 10$, que é o menor em comparação com soluções do estado da arte para $DR = 6n$, $DR = 7n$ e $DR = 8n - 1$, exceto pelos valores apresentada por [9]. No entanto, os resultados mostram melhorias na área do circuito, atraso e potência.

Tabela 5.4: *Resultados de Área, Atraso e Potência obtidos a partir da síntese.*

DR	*n*	*bits*	Área (μm^2)	Atraso (ns)	Potência (mW)
$6n$ [17]	9	54	**5372**	**1,05**	**3,47**
$6n$ [21]	9	54	13035	1,81	7,98
$7n$ [21]	8	56	7875	1,64	5,17
$(8n+1)$ [21]	7	57	7086	3,80	25,38
$9n$ [9]	**6**	54	105394	5,53	134,39
$9n$ Proposto	**6**	54	33726	2,46	19,55
$9n$ Proposto Otimizado	**6**	54	19294	2,06	13,45
$6n$ [17]	15	90	**9022**	**1,07**	**5,74**
$6n$ [21]	15	90	21636	1,90	13,25
$7n$ [21]	13	91	13007	1,72	8,79
$(8n-1)$ [21]	11	87	12487	4,89	14,82
$9n$ [9]	**10**	90	252050	6,94	287,52
$9n$ Proposto	**10**	90	92704	2,72	110,91
$9n$ Proposto Otimizado	**10**	90	43772	2,43	38,37

5.6 Conclusões

Neste capítulo, foi apresentado um método para projetar conversores reversos RNS eficientes usando um novo conjunto de módulos específicos com a maior faixa dinâmica do estado da arte ($9n$ *bits*). Até a publicação original da proposta em [8], implementações eficientes de conversores reversos RNS para conjuntos de módulos específicos eram restritas até $DR \simeq 8n$. O uso de *DR*s maiores é útil para aplicações DSP com grande operandos devido à redução do número de *bits* por canal para executar a aritmética RNS. Os resultados experimentais demonstram que o conversor proposto otimizado exige em média de $82,16\%$ menos de área e é $2,77$ vezes mais rápido em comparação com uma solução existente no estado da arte que usa conjuntos de módulos arbitrários para derivar $DR = 9n$.

CAPÍTULO 6

Conclusões

Essa tese propôs um conjunto de módulos específicos para aplicações com faixa dinâmica de até $9n$, bem como a arquitetura para implementar o conversor reverso associado. Sendo que, até o momento, implementações eficientes de conversores reversos RNS para conjuntos de módulos específicos eram restritas a faixas dinâmicas de até no máximo $8n$. Os resultados experimentais demostraram que a arquitetura proposta supera as melhores soluções no estado da arte, que utilizam conjunto de módulos arbitrários para derivar faixa dinâmica de $9n$. O aumento do número de canais também é desejável para aplicações DSP com grande número de *bits* devido à redução do número de *bits* por canal.

Além do conjunto de módulos específico, esse trabalho também propôs dois métodos para projetar conversores reversos gerais para conjunto de módulos estendidos que fornecem faixas dinâmicas elevadas. Os conversores publicados previamente requeriam uma seleção de peso complexa das entradas ou etapas complexas de conversão final. A proposta possibilitou que a seleção de peso dos termos multiplicativos associados às entradas fosse reduzida a adições de $2n$ *bits* e a conversão final requer apenas uma comparação. Resultados experimentais demonstraram que as abordagens propostas alcançaram ganhos significativos na redução de área em comparação com o estado da arte, para estruturas de conversão reversa com faixa dinâmica genérica. Além disso, ainda foi proposta uma melhoria na arquitetura de hardware do CRTf usando conjuntos de módulos genéricos equilibrados.

Um procedimento que reúne diversas técnicas de otimização lógica para multiplicação modular por constante também foi apresentado. Apesar das técnicas empregadas serem simples, os resultados demonstraram um ganho substancial na implementação dos conversores reversos propostos, uma vez que é possível perceber a repetição sistemática do multiplicador modular por constante nas equações dos algoritmos de conversão.

Tanto a proposta do conjunto de módulos específico quanto os métodos de implementação das arquiteturas genéricas trazem avanços no estado da arte na área de RNS, mais especificamente na melhoria dos conversores reversos. Alcançando o conversor mais eficiente já apresentado para $DR \simeq 9n$, além de melhorias consideráveis nas implementações de estruturas de conversão reversas com DR genérico do estado da arte.

6.1 Lista de Publicações

Essa tese gerou dois artigos em conferências internacionais, dois artigos em revistas internacionais (um publicado e outro submetido, estando na segunda rodada de revisão) e um relatório técnico. Em ordem cronológica:

- **Matos, R.** and H. Pettenghi. *Mixed-radix conversion (MRC) equations for the moduli set* $\{2^{2n}, 2^n \pm 3, 2^n \pm 1\}$. Technical Report, DEEL-UFSC, 2015. URL: http://tele.sj.ifsc.edu.br/~roberto.matos/RNS/MRC-equations_v1.pdf.
- H. Pettenghi, R. Chaves, **Matos, R.**, and L. Sousa. *Method for designing two levels RNS reverse converters for large dynamic ranges*. Integration, the VLSI Journal, pages 1–9, 2016.
- H. Pettenghi, **Matos, R.**, and A. Molahosseini. *RNS reverse converters for moduli sets with dynamic ranges of 9n-bit*. In Circuits and Systems (LASCAS), 2016 IEEE Third Latin American Symposium on, pages 1–4, Feb 2016.
- **Matos, R.**, R. Paludo, N. Chervyakov, P. A. Lyakhov, and H. Pettenghi. *Efficient implementation of modular multiplication by constants applied to RNS reverse converters*. In 2017 IEEE International Symposium on Circuits and Systems (ISCAS), pages 1–4, May 2017.
- Em Revisão: H. Pettenghi, R. Paludo, **Matos, R.**, and P. A. Lyakhov. *Efficient RNS reverse converters for moduli sets with dynamic ranges up to* $(10n+1)$*-bit*. Circuits, Systems, and Signal Processing, 2018.

6.2 Trabalhos Futuros

No Capítulo 3, foi apresentado um estudo com unidades modulares MACs variando o módulo e a largura dos operandos. Ficou claro que o número de *bits* tem um impacto no atraso das unidades e a possibilidade de ter mais canais com menos *bits* pode acelerar a computação de grandes operandos, mesmo que a conversão reversa seja mais custosa. Dessa forma, com o aumento da diversidade de aplicações sendo migradas para RNS, é necessária uma ferramenta que analise

os requisitos da aplicação e gere um sistema RNS sob medida. A fase de pré-computação com as técnicas de otimização apresentadas no Capítulo 3 já é um embrião dessa iniciativa. Entretanto existem melhorias e formalizações que precisam ser analisadas ainda da fase de pré-computação, como a análise do ponto de vista computacional do problema da otimização. As várias técnicas com os diversos operandos explodem rapidamente o número de combinações possíveis, o que impossibilita um teste de cada opção para determinar qual seria a melhor combinação para chegar no menor número de produtos parciais. Estamos lidando com um problema NP-completo?

Além disso, as técnicas de otimização podem ser estendidas para multiplicação de duas variáveis, onde é possível abrir novos caminhos para otimização lógica multi-variável. Mesmo na otimização da multiplicação por constante, é possível direcionar o trabalho para algo com mais formalização matemática. Por exemplo, é possível transformar todas as colunas dos vetores em uma única equação e analisar quais as ferramentas algébricas poderiam ser utilizadas para minimizar o número de vetores finais e qual seria o limite dessa otimização.

Ainda com relação à ferramenta, seria interessante a geração automatizada de descrições de hardware para conversores diretos e reversos parametrizáveis, compressores eficientes e unidades aritméticas, entre outros. Isso propiciaria testes rápidos e comparações das diversas abordagens, facilitando o emprego de unidades RNS e a pesquisa na área.

No campo da arquitetura de conversores reversos genéricos, parece ser possível explorar as técnicas de otimização do número de vetores das multiplicações por constante para reorganização da arquitetura dos conversores e minimizar as somas completas no caminho crítico. Nessa linha de trabalho, é possível avaliar se existem regras que direcionam a tomada de decisão para geração automática de novos conversores.

Referências Bibliográficas

[1] N.S. Szabó and R.I. Tanaka. *Residue arithmetic and its applications to computer technology*. McGraw-Hill series in information processing and computers. McGraw-Hill, 1967.

[2] L. Sousa. 2^n RNS scalers for extended 4-moduli sets. *Computers, IEEE Transactions on*, 64(12):3322–3334, Dec 2015.

[3] B. Parhami. *Computer Arithmetic: Algorithms and Hardware Designs, Second Edition*. Oxford University Press, Incorporated, 2010.

[4] Yuke Wang. Residue-to-binary converters based on new chinese remainder theorems. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 47(3):197–205, Mar 2000.

[5] K. Navi, A.S. Molahosseini, and M. Esmaeildoust. How to teach residue number system to computer scientists and engineers. *Education, IEEE Transactions on*, 54(1):156–163, Feb 2011.

[6] Yuke Wang, Xiaoyu Song, M. Aboulhamid, and Hong Shen. Adder based residue to binary number converters for $(2^n-1, 2^n, 2^n+1)$. *Signal Processing, IEEE Transactions on*, 50(7):1772–1779, Jul 2002.

[7] R. Chaves and L. Sousa. $\{2^n+1, 2^{n+k}, 2^n-1\}$: a new RNS moduli set extension. In *Euromicro Symposium on Digital System Design, 2004. DSD 2004.*, pages 210–217, Aug 2004.

[8] H. Pettenghi, **Matos, R.**, and A. Molahosseini. Rns reverse converters for moduli sets with dynamic ranges of 9*n*-bit. In *Circuits and Systems (LASCAS), 2016 IEEE Third Latin American Symposium on*, pages 1–4, Feb 2016.

[9] H. Pettenghi, R. Chaves, and L. Sousa. Method to design general RNS reverse converters for extended moduli sets. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 60(12):877–881, Dec 2013.

[10] H. Pettenghi, R. Chaves, **Matos, R.**, and L. Sousa. Method for designing two levels RNS reverse converters for large dynamic ranges. *Integration, the VLSI Journal*, pages 1 – 9, 2016.

[11] **Matos, R.**, R. Paludo, N. Chervyakov, P. A. Lyakhov, and H. Pettenghi. Efficient implementation of modular multiplication by constants applied to RNS reverse converters. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, May 2017.

[12] N.I. Chervyakov, A. S. Molahosseini, P. A. Lyakhov, M. G. Babenko, and M. A. Deryabin. Residue-to-binary conversion for general moduli sets based on approximate chinese remainder theorem. *International Journal of Computer Mathematics*, 94(9):1833–1849, 2017.

[13] P.V. Ananda Mohan and A.B. Premkumar. RNS-to-binary converters for two four-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 54(6):1245–1254, June 2007.

[14] M. Hosseinzadeh, A.S. Molahosseini, and K. Navi. An improved reverse converter for the moduli set $\{2^n-1, 2^n, 2^n+1, 2n+1-1\}$. *Trans. IEICE Electronics Express*, 5(17):672–677, Sep. 2008.

[15] B. Cao, T. Srikanthan, and C.H. Chang. Efficient reverse converters for four-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1\}$. *Computers and Digital Techniques, IEE Proceedings* -, 152(5):687–696, Sept 2005.

[16] L. Sousa and S. Antao. MRC-based RNS reverse converters for the four-moduli sets $\{2^n+1, 2^n-1, 2^n, 2^{2n+1}-1\}$ and $\{2^n+1, 2^n-1, 2^{2n}, 2^{2n+1}-1\}$. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 59(4):244–248, April 2012.

[17] A.S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei, and S. Timarchi. Efficient reverse converter designs for the new 4-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ and $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$ based on new crts. *Circuits*

and Systems I: Regular Papers, IEEE Transactions on, 57(4):823–835, April 2010.

[18] Bin Cao, Chip-Hong Chang, and T. Srikanthan. A. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 54(5):1041–1049, May 2007.

[19] A. Skavantzos and T. Stouraitis. Grouped-moduli residue number systems for fast signal processing. In *Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on*, volume 3, pages 478–483 vol.3, Jul 1999.

[20] A. Skavantzos. An efficient residue to weighted converter for a new residue number system. In *VLSI, 1998. Proceedings of the 8th Great Lakes Symposium on*, pages 185–191, Feb 1998.

[21] H. Pettenghi, R. Chaves, and L. Sousa. RNS reverse converters for moduli sets with dynamic ranges up to $(8n+1)$-bit. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 60(6):1487–1500, June 2013.

[22] A.A. Hiasat. Vlsi implementation of new arithmetic residue to binary decoders. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 13(1):153–158, Jan 2005.

[23] S.J. Piestrak. Design of residue generators and multioperand modular adders using carry-save adders. *Computers, IEEE Transactions on*, 43(1):68–77, Jan 1994.

[24] A.B. Premkumar, E.L. Ang, and E.M. Lai. Improved memoryless RNS forward converter based on the periodicity of residues. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 53(2):133–137, Feb 2006.

[25] Amir Sabbagh Molahosseini, Keivan Navi, Omid Hashemipour, and Ali Jalali. An efficient architecture for designing reverse converters based on a general three-moduli set. *Journal of Systems Architecture*, 54(10):929 – 934, 2008.

[26] A.S. Molahosseini, K. Navi, and M.K. Rafsanjani. Efficient forward and reverse converters for a new high-radix moduli set. In *Information Technology, 2008. ITSim 2008. International Symposium on*, volume 4, pages 1–4, Aug 2008.

[27] H. Pettenghi, F. Pratas, and L. Sousa. Method for designing efficient mixed radix multipliers. *Circuits, Systems, and Signal Processing*, 33(10):3165–3193, 2014.

[28] A. Skavantzos, M. Abdallah, T. Stouraitis, and D. Schinianakis. Design of a balanced 8-modulus RNS. In *Electronics, Circuits, and Systems, 2009. ICECS 2009. 16th IEEE International Conference on*, pages 61–64, Dec 2009.

[29] Inc. Virtual Silicon Technology. Umc high density standards cells library 90*n*m cmos process, 2010.

[30] S.J. Piestrak and K.S. Berezowski. Design of residue multipliers-accumulators using periodicity. In *Signals and Systems Conference, 208. (ISSC 2008). IET Irish*, pages 380–385, June 2008.

[31] S. Ma, J. H. Hu, and C. H. Wang. A novel modulo $2^n - 2^k - 1$ adder for residue number system. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(11):2962–2972, Nov 2013.

[32] G. Dimitrakopoulos and D. Nikolos. High-speed parallel-prefix vlsi ling adders. *Computers, IEEE Transactions on*, 54(2):225–231, Feb 2005.

[33] Yuke Wang. New chinese remainder theorems. In *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers (Cat. No.98CH36284)*, volume 1, pages 165–171 vol.1, Nov 1998.

[34] **Matos, R.** and H. Pettenghi. Mixed-radix conversion (MRC) equations for the moduli set $\{2^{2n}, 2^n \pm 3, 2^n \pm 1\}$. Technical report, DEEL-UFSC, 2015.

[35] C.Y. Hung and B. Parhami. An approximate sign detection method for residue numbers and its application to RNS division. *Computers & Mathematics with Applications*, 27(4):23 – 35, 1994.

[36] H. Ahmadifar and G. Jaberipur. A new residue number system with 5-moduli set: $22q, 2q \pm 3, 2q \pm 1$. *The Computer Journal*, 58(7):1548–1565, 2015.

[37] Reto Zimmermann. Binary Adder Architectures for Cell-Based VLSI and Their Synthesis. Master's thesis, Swiss Federal Institute of Technology, Zurich, 1997.

[38] Lampros Kalampoukas, Dimitris Nikolos, Costas Efstathiou, Haridimos T. Vergos, and John Kalamatianos. High-speed parallel-prefix modulo 2n - 1 adders. *IEEE Trans. Comput.*, 49(7):673–680, July 2000.

[39] R. A. Patel, M. Benaissa, N. Powell, and S. Boussakta. Novel power-delay-area-efficient approach to generic modular addition. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(6):1279–1292, June 2007.

[40] A. Tyagi. A reduced-area scheme for carry-select adders. *Computers, IEEE Transactions on*, 42(10):1163–1170, Oct 1993.

[41] R. Zimmermann. Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication. In *Proceedings 14th IEEE Symposium on Computer Arithmetic*, pages 158–167, 1999.