JOÃO DALLAMUTA
HENRIQUE AJUZ HOLZMANN
(ORGANIZADORES)

Collection:

APPLIED ELECTRICAL ENGINEERING

Atena Editora
Ano 2022

JOÃO DALLAMUTA
HENRIQUE AJUZ HOLZMANN
(ORGANIZADORES)

Collection:

APPLIED ELECTRICAL ENGINEERING

Atena Editora
Ano 2022

Editora chefe
Profª Drª Antonella Carvalho de Oliveira
Editora executiva
Natalia Oliveira
Assistente editorial
Flávia Roberta Barão
Bibliotecária
Janaina Ramos
Projeto gráfico
Camila Alves de Cremo
Daphynny Pamplona
Gabriel Motomu Teshima
Luiza Alves Batista
Natália Sandrini de Azevedo
Imagens da capa
iStock
Edição de arte
Luiza Alves Batista

Atena Editora
Ano 2022

Atena
Editora
Ano 2022

Collection: applied electrical engineering

Diagramação: Camila Alves de Cremo
Correção: Yaiddy Paola Martinez
Indexação: Amanda Kelly da Costa Veiga
Revisão: Os autores
Organizadores: João Dallamuta
Henrique Ajuz Holzmann.

Dados Internacionais de Catalogação na Publicação (CIP)

C697 Collection: applied electrical engineering / Organizadores João Dallamuta, Henrique Ajuz Holzmann. – Ponta Grossa - PR: Atena, 2022.

Formato: PDF
Requisitos de sistema: Adobe Acrobat Reader
Modo de acesso: World Wide Web
Inclui bibliografia
ISBN 978-65-5983-858-5
DOI: https://doi.org/10.22533/at.ed.585222801

1. Electrical engineering. I. Dallamuta, João (Organizador). II. Holzmann, Henrique Ajuz (Organizador). III. Título.

CDD 621.3

Elaborado por Bibliotecária Janaina Ramos – CRB-8/9166

Atena Editora
Ponta Grossa – Paraná – Brasil
Telefone: +55 (42) 3323-5493
www.atenaeditora.com.br
contato@atenaeditora.com.br

Atena Editora
Ano 2022

DECLARAÇÃO DOS AUTORES

Os autores desta obra: 1. Atestam não possuir qualquer interesse comercial que constitua um conflito de interesses em relação ao artigo científico publicado; 2. Declaram que participaram ativamente da construção dos respectivos manuscritos, preferencialmente na: a) Concepção do estudo, e/ou aquisição de dados, e/ou análise e interpretação de dados; b) Elaboração do artigo ou revisão com vistas a tornar o material intelectualmente relevante; c) Aprovação final do manuscrito para submissão.; 3. Certificam que os artigos científicos publicados estão completamente isentos de dados e/ou resultados fraudulentos; 4. Confirmam a citação e a referência correta de todos os dados e de interpretações de dados de outras pesquisas; 5. Reconhecem terem informado todas as fontes de financiamento recebidas para a consecução da pesquisa; 6. Autorizam a edição da obra, que incluem os registros de ficha catalográfica, ISBN, DOI e demais indexadores, projeto visual e criação de capa, diagramação de miolo, assim como lançamento e divulgação da mesma conforme critérios da Atena Editora.

DECLARAÇÃO DA EDITORA

A Atena Editora declara, para os devidos fins de direito, que: 1. A presente publicação constitui apenas transferência temporária dos direitos autorais, direito sobre a publicação, inclusive não constitui responsabilidade solidária na criação dos manuscritos publicados, nos termos previstos na Lei sobre direitos autorais (Lei 9610/98), no art. 184 do Código Penal e no art. 927 do Código Civil; 2. Autoriza e incentiva os autores a assinarem contratos com repositórios institucionais, com fins exclusivos de divulgação da obra, desde que com o devido reconhecimento de autoria e edição e sem qualquer finalidade comercial; 3. Todos os e-book são *open access, desta forma* não os comercializa em seu site, sites parceiros, plataformas de *e-commerce,* ou qualquer outro meio virtual ou físico, portanto, está isenta de repasses de direitos autorais aos autores; 4. Todos os membros do conselho editorial são doutores e vinculados a instituições de ensino superior públicas, conforme recomendação da CAPES para obtenção do Qualis livro; 5. Não cede, comercializa ou autoriza a utilização dos nomes e e-mails dos autores, bem como nenhum outro dado dos mesmos, para qualquer finalidade que não o escopo da divulgação desta obra.

Atena
Editora
Ano 2022

APRESENTAÇÃO

A engenharia elétrica tornou-se uma profissão há cerca de 130 anos, com o início da distribuição de eletricidade em caráter comercial e com a difusão acelerada do telégrafo em escala global no final do século XIX.

Na primeira metade do século XX a difusão da telefonia e da radiodifusão além do crescimento vigoroso dos sistemas elétricos de produção, transmissão e distribuição de eletricidade, deu os contornos definitivos para a carreira de engenheiro eletricista que na segunda metade do século, com a difusão dos semicondutores e da computação gerou variações de ênfase de formação como engenheiros eletrônicos, de telecomunicações, de controle e automação ou de computação.

Produzir conhecimento em engenharia elétrica é portando pesquisar em uma gama enorme de áreas, subáreas e abordagens de uma engenharia que é onipresente em praticamente todos os campos da ciência e tecnologia.

Neste livro temos uma diversidade de temas, níveis de profundidade e abordagens de pesquisa, envolvendo aspectos técnicos e científicos. Aos autores e editores, agradecemos pela confiança e espirito de parceria.

João Dallamuta

Henrique Ajuz Holzmann

SUMÁRIO

https://doi.org/10.22533/at.ed.58522280115

CAPÍTULO 3

OTIMIZAÇÃO DE OPERAÇÕES MODULARES ATRAVÉS DO USO DE PSEUDO-MÓDULOS

Data de aceite: 10/01/2022

Augusto Cezar Boldori Vassoler
Universidade Federal de Santa Catarina
CTC – Centro Tecnológico
Campus universitário – Trindade
Programa de Iniciação Científica - UFSC
Florianópolis - SC

RESUMO: Operações modulares são largamente empregadas em diversas áreas da computação e do processamento de dados, como criptografia[7], mineração de criptomoedas, processamento digital de sinais (DSP) [10], unidades de processamento gráfico (GPUs), entre muitas outras. Em geral essas aplicações costumam trabalhar com números muito elevados, podendo chegar a centenas e até milhares de casas decimais. Por conta disso, é natural esperar-se um tempo de processamento elevado, o que geralmente ocorre por conta da utilização de módulos não otimizados e considerados lentos. Com isso, pode-se dizer que é de extrema importância a escolha correta dos módulos para operação, de modo que possibilitem o menor tempo de operação possível. Um meio de mitigar essa influência negativa no tempo de realização é a substituição do módulo original por pseudo-módulos, que são obtidos através da fatoração e remodulação dos valores originais. Dessa forma, podem ser obtidos módulos em formatos considerados preferíveis, que apresentam menores tempos de operação. Além disso, as formas de implementação podem ser realizadas de formas diferentes em software e em hardware, o que impacta no desempenho. Dessa forma, o presente trabalha visa verificar a influência do tipo do módulo utilizado no desempenho temporal de operações modulares em hardware e software (a fim de possibilitar futuras aplicações), além da verificação do ganho de tempo de operação ocasionado pela substituição dos módulos originais por pseudo-módulos e também a realização da escolha mais otimizada possível para os mesmos.

PALAVRAS-CHAVE: Processamento digital de sinais, operações modulares, circuitos digitais.

1 I INTRODUÇÃO

A aritmética modular é um sistema de aritmética para inteiros, que foi inicialmente introduzido pelo matemático Suíço Euler, com a abordagem da congruência [11]. Apesar disso, a abordagem moderna da aritmética modular foi desenvolvida por Carl Friedrich Gauss [11]. A realização de operações modulares se dá através da recodificação dos operandos pelos restos da divisão de seus respectivos módulos. A operação desejada, seja ela adição, multiplicação, divisão ou outras é então realizada utilizando os restos obtidos, sendo que o resultado da operação será novamente remodulado, obtendo-se assim o resultado final.

Como exemplo se pode considerar um caso onde há duas constantes Y=69 e Z=12 e quer-se realizar as operações de soma (Y+Z) e multiplicação (Y*Z) entre ambas. Para tanto, deve-se modular Y e Z por 7, sendo:

$$Y = 69 \; mod \; 7 = 6$$

$$Z = 12 \; mod \; 7 = 5$$

Posteriormente soma-se e multiplica-se os resíduos e modula-se novamente o resultado, como é mostrado a seguir.

$$Y + Z = (6 + 5) \; mod \; 7 = 4$$

$$Y * Z = (6 * 5) \; mod \; 7 = 2$$

Realizando a modulação do resultado do cálculo realizado diretamente comprova-se que o método é correto, uma vez que o resultado é exatamente o mesmo:

- **Y+Z:**

$$(Y + Z) = (69 + 12) \; mod \; 7$$

$$(Y + Z) = 81 \; mod \; 7$$

$$(Y + Z) = 4$$

- **Y*Z**:

$$(Y * Z) = (69 * 12) \; mod \; 7$$

$$(Y * Z) = 828 \; mod \; 7$$

$$(Y + Z) = 2$$

O emprego desse tipo de operação apresenta grande importância para as mais diversas finalidades, como a implementação de algoritmos de processamento de sinais digitais e filtros digitais[10], aplicação de códigos de criptografia [7], como o algoritmo de Montgomery[9], codificação de vídeo em unidades gráficas de processamento, além de muitas outras que podem ser citadas. É comum que os módulos utilizados nesses contextos sejam de ordem muito elevada, contanto várias casas decimais em seu expoente, o que pode facilmente levar a um aumento considerável no tempo de processamento, aumento esse que é indesejável. A principal causa para esse problema é a utilização de módulos não ótimos, os quais inevitavelmente levam a um elevado tempo de processamento.

Através de testes e estudos previamente realizados sabe-se que módulos temporalmente eficientes possuem o formato $\{2^n \pm 1\}$ e também $\{2^n \pm 2^\alpha \pm 1\}$, onde α também é um número inteiro [5]. Por outro lado, á também os módulos do tipo $\{2^n \pm k\}$, onde K também é uma constante inteira, que não se encaixam em nenhum dos padrões anteriores, e que apresentarão uma menor eficiência, sendo sua utilização considerada indesejável. Para esse último caso, é sabido que o custo temporal de operação em hardware é inversamente proporcional ao número de 1's na representação binária de K.

Para casos em que seja necessário realizar uma determinada operação utilizando um módulo considerado ruim, uma solução prática é a sua substituição por pseudo-módulos que se encaixem em algum dos formatos eficientes. Com isso é fácil perceber que pseudo-módulos nada mais são do que valores derivados a partir de outros módulos, por meio do processo de remodulação, em geral com o objetivo de obter-se um K mais eficiente. Para facilitar esse processo, existe uma propriedade que permite obter-se um pseudo-módulo a partir de qualquer módulo, que é enunciada da seguinte forma: "sejam os módulos m_1 e m_2, o módulo m_2 de uma operação modular realizada em m_1 é igual ao módulo m_2 da própria operação se $m_1 = k * m_2$, sendo k uma constante inteira. Ou seja, m_1 deve ser múltiplo de m_2, então m_1 será pseudo-módulo de m_2"[1]. Matematicamente, a propriedade pode ser traduzida da seguinte forma:

$$mod(mod(operação, m_1), m_2) = mod(operação, m_2), \text{ para } m_1 \text{ múltiplo de } m_2$$

Para exemplificação pode ser usada a operação $(a*b*c)$ *mod* m_0, onde a=65536, b=256, c=4 e m_0 =341. O resultado da operação direta será 64, e o módulo 341 pode ser reescrito para que se obtenha seu formato:

$$341 = 512 - 171 = 2^9 - 171$$

Então nesse caso tem-se k = 171, cuja representação binária será k=10101011, possuindo cinco 1's na mesma. É possível então encontrar o pseudo-módulo de 341 aplicando a propriedade acima definida. Considerando uma constante de 10 bits (n=10) tem-se

$$mod(2^{10}, 341) = 1$$

Logo o pseudo-módulo deverá ser 2^{10}-1=1023, o que leva a um formato ótimo do tipo 2^n-1. Tem-se então que 1023 = 3*341, logo sendo múltiplo e verificando a condição requerida e confirmando ser um pseudo-módulo.

É importante ressaltar que para a realização desse processo consiste na conversão inicial do valor de entrada para o pseudo-módulo desejado, em sequência realizando as operações aritméticas necessárias na base convertida e, apenas posteriormente a realização de todas elas é feita a reconversão do resultado final para o módulo original [1]. Desse modo, matematicamente a reconversão é feita na forma *mod*(*mod*(*RESULTADO*, m_1),m_2), da mesma forma que é mostrada no esquemático.

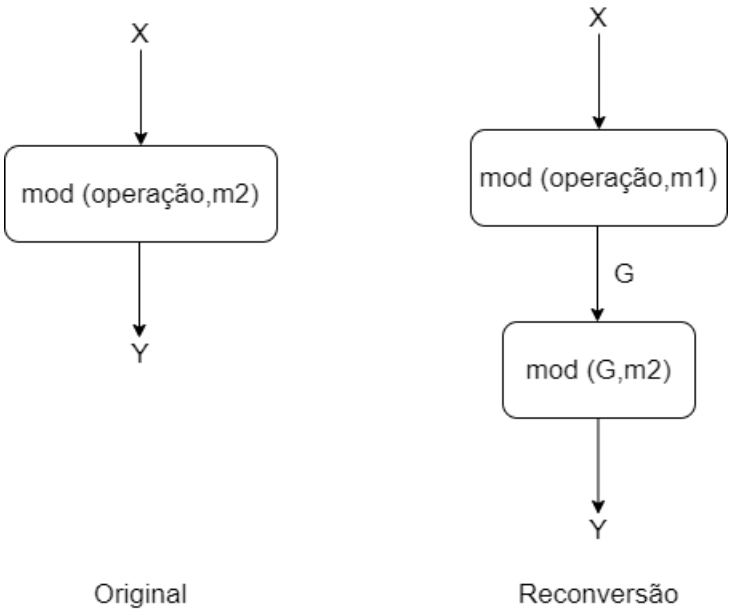

Figura 1 - Conversão e reconversão para pseudo-módulo

A fim de verificar o real impacto temporal dessa mudança em operações realizadas em hardware, foram implementadas multiplicações utilizando aritmética modular, verificando o impacto temporal de diferentes tipos de módulos. Também foram realizados testes utilizando algoritmos em software, a fim de verificar quais parâmetros impactam em seu desempenho.

2 | MATERIAIS E MÉTODOS

O método utilizado para a caracterização dos testes em hardware consistiu na implementação de multiplicadores em VHDL, os quais operam com os valores modulares de $\{2^n\}, \{2^n \pm 1\}, \{2^n \pm k\}, \{2^n + 2^\alpha + 1\} e \{2^n - 2^\alpha - 1\}$. A arquitetura de multiplicadores utilizados para implementar as operações com $\{2^n \pm k\}$ está mostrada na figura 1, enquanto que a usada para $2^n \pm 1$ é apresentada na figura 2.

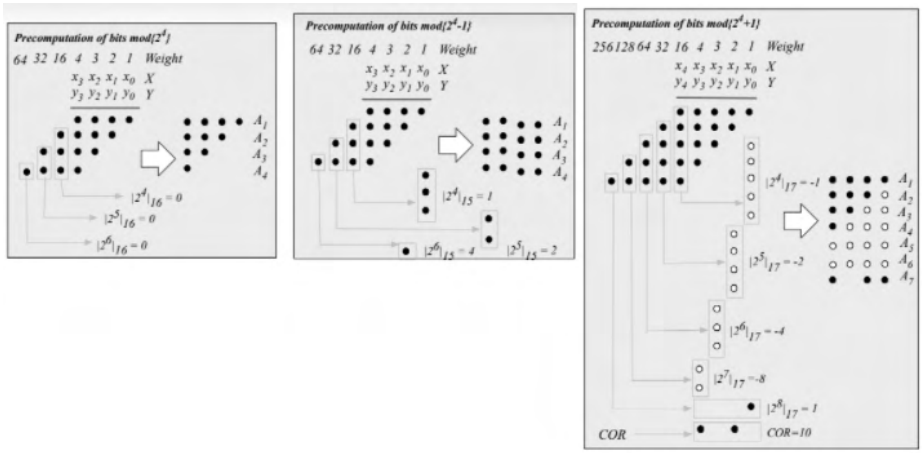

Figura 2 - Pré-computação de módulos 2^4 e 2^4 +/- 1

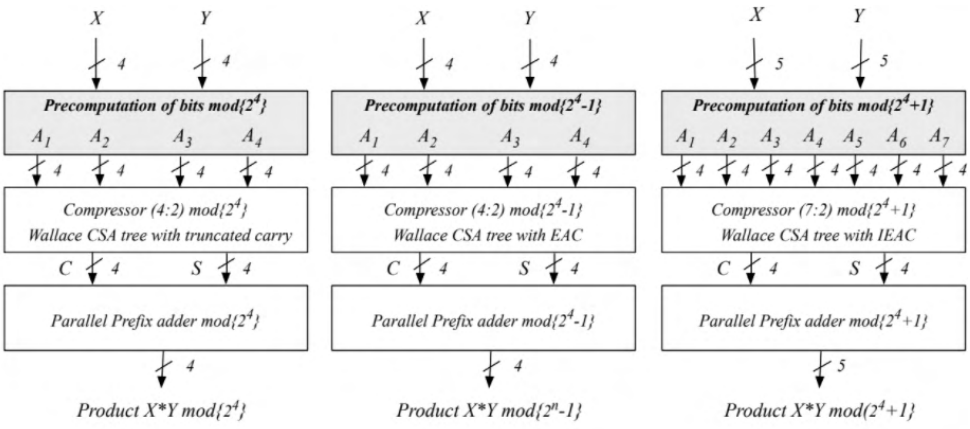

Figura 3 - Multiplicadores para módulos 2^4 e 2^4 +/- 1

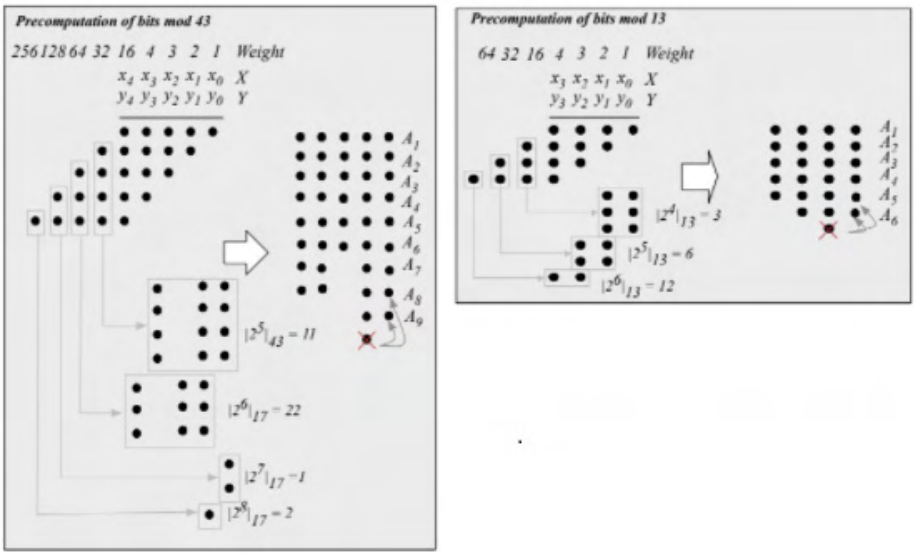

Figura 4 - Pré-computação do módulo 43 (2^5 +11) e módulo 13 (2^4 – 3)

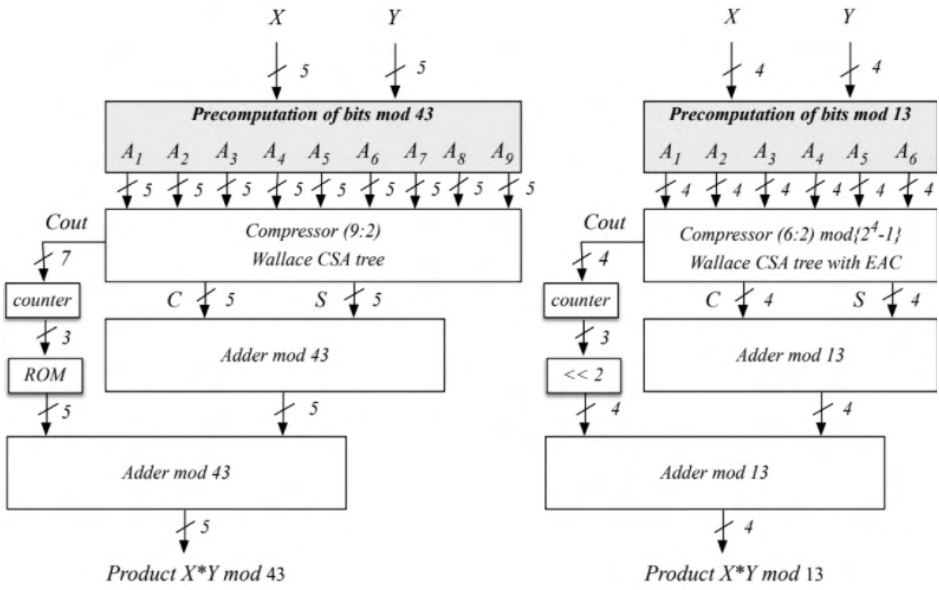

Figura 5 - Multiplicadores para os módulos 2^4 -3 e 2^5 + 11

A unidade multiplicadora para $\{2^n \pm k\}$ utiliza uma memória ROM para armazenamento das contribuições dos carrys somados. São claramente menos eficientes em comparação com as unidades de $\{2^n \pm 1\}$ por exemplo, que não necessitam de memória, realizando toda operação de forma combinacional.

Para a obtenção dos resultados de delay dos circuitos, foi realizada a síntese

dos mesmos em tecnologia ASICs Taiwan Semiconductors de 65 nm, sendo os testes realizados para entradas de até 35 bits. Tendo em mãos os referidos valores, foi realizado um processo iterativo com os dados, a fim de obter as curvas de regressão do delay total em função do número de bits da entrada, para cada um dos conjuntos de módulos testados.

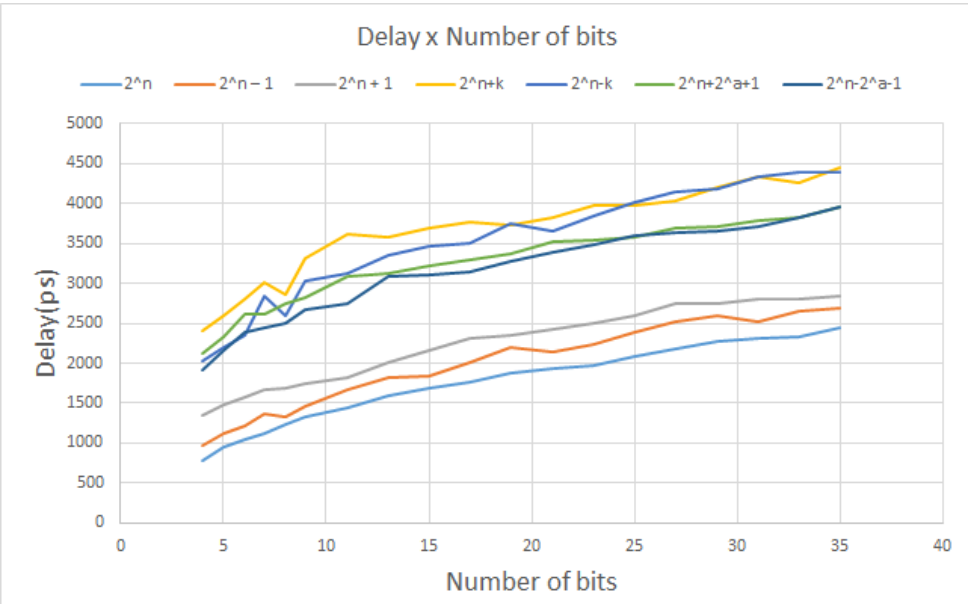

Figura 6 - Curvas de regressão obtidas para o delay

Para a realização dos testes foi escolhido um conjunto de módulos específicos, com seus respectivos pseudo-módulos, para diferentes números de bits (n) e números de 1's na representação binária de k. Os tipos escolhidos foram especificamente $\{2^n\text{-}1\}$, $\{2^n\text{-}k\}$ com dois 1's em k e $\{2^n\text{+}k\}$ com três ou mais 1's em k.

Pseudomódulo	Valor	1's em K	Módulo	Valor	1's em K
2^26 - 1	67108863	1	2^24 + k	22369621	12
2^30 -1	1073741823	1	2^28 + k	357913941	14
2^25 - 5	33554427	2	2^23 + k	11184809	11
2^28 - 65	268435391	2	2^25 + k	38347913	8

Tabela 1 - Módulos e pseudo-módulos utilizados para os testes em hardware

Os testes em software foram realizados utilizando a linguagem Python, através de um script que realiza o módulo de uma multiplicação de constantes e contabiliza o tempo total de operação. Foram utilizados os mesmos módulos da Tabela 1.

3 I RESULTADOS E DISCUSSÃO

Os valores obtidos obtidos estão sobrepostos às suas respectivas curvas nos gráficos abaixo.

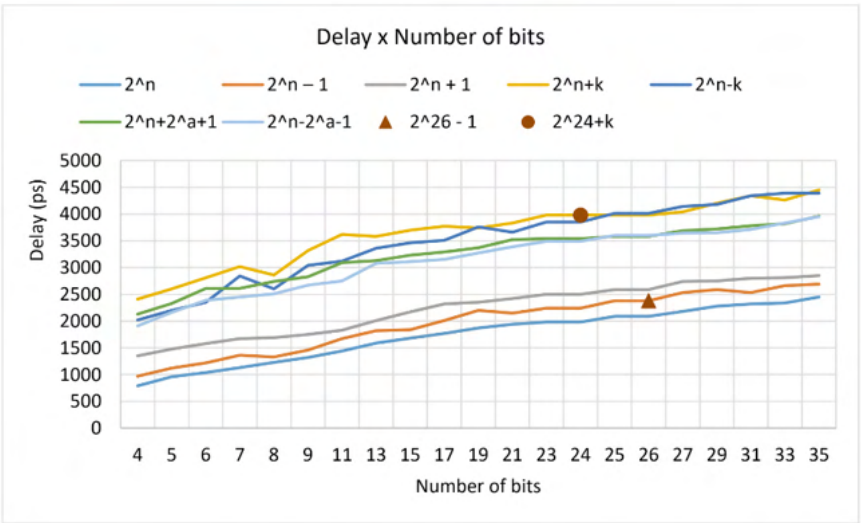

Figura 7 - Delay em hardware para módulo 2^24 +k e pseudo-módulo 2^26 -1

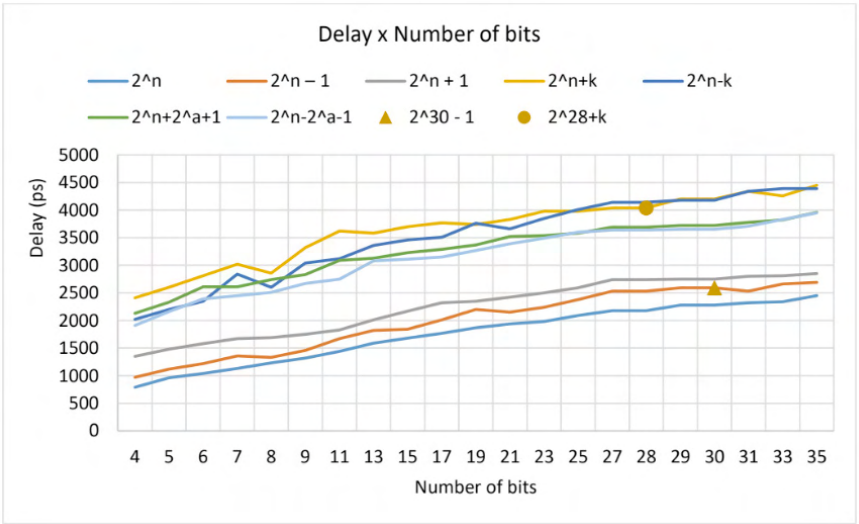

Figura 8 - Delay em hardware para módulo 2^28 +k e pseudo-módulo 2^30 -1

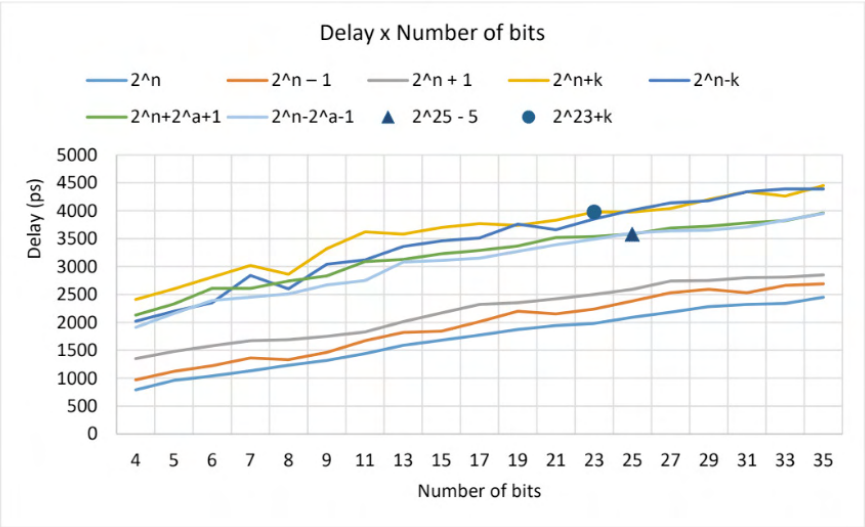

Figura 9 - Delay em hardware para módulo 2^23 +k e pseudo-módulo 2^25 -1

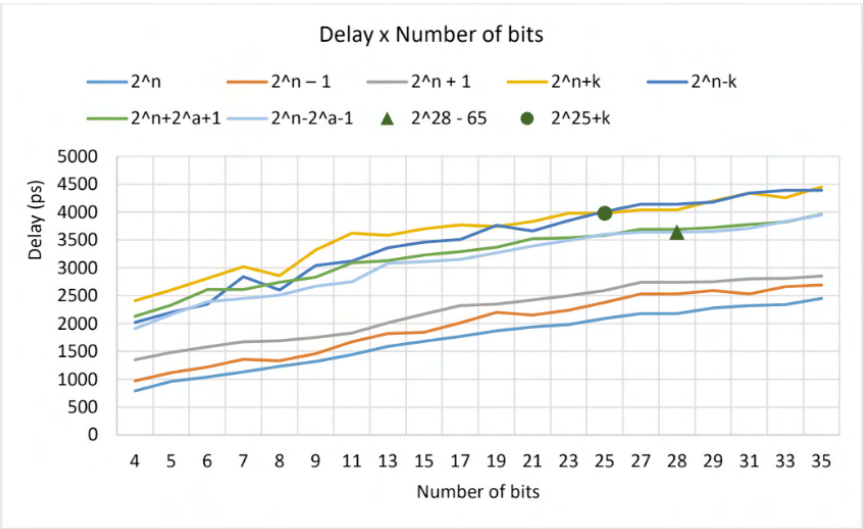

Figura 10 - Delay em hardware para módulo 2^25 +k e pseudo-módulo 2^28 - 65

Os valores dos pseudo-módulos estão na legenda representados pelos triângulos, enquanto os círculos mostram os delas referentes aos módulos. Observando os resultados obtidos, nota-se que em todos os casos os pseudo-módulos da forma $\{2^n \pm 1\}$, embora maiores (maior número n de bits) apresentam um menor delay em relação aos da forma $\{2^n \pm k\}$ e até mesmo aos do formato $\{2^n + 2^\infty + 1\}$ e $\{2^n - 2^\infty - 1\}$ (mais eficientes que os anteriores), o que já era previsto. Já para aqueles que possuem três ou mais 1's, o número de uns da representação binária não terá mais impacto no atraso. Isso se deve ao uso da memória ROM anteriormente evidenciada na arquitetura do multiplicador, a qual eliminará essa

influência. Por esse motivo os tipos $2^n \pm 1$ e aqueles que possuem dois 1's apresentam um melhor desempenho, uma vez que não usam componentes de armazenamento como memórias, que inevitavelmente inserem atraso.Apesar disso, é importante ressaltar a existência de outras implementações que processam a contribuição dos carrys de forma puramente combinacional, sendo que nesses casos a quantidade de números uns terá impacto no tempo de operação.

Pseudomódulo	Delay (us)	1's em K	Módulo	Delay(us)	1's em K
2^26 - 1	2.2745	1	2^24 + k	3.5702	12
2^30 -1	1.972	1	2^28 + k	3.1465	14
2^25 - 5	2.13	2	2^23 + k	3.475	11
2^28 - 65	1.88175	2	2^25 + k	2.9412	8

Tabela 2 - Tempos de operação em software

Na tabela 2 estão mostrados os resultados de atraso obtidos rodando o script em Python para cada um dos módulos. Analisando o padrão que se apresenta, é possível notar que o número de 1's acaba por impactar o desempenho em software, pois todos os módulos originais com mais de três 1's em k possuem um maior delay que seus pseudo-módulos em formato mais simples. Apesar disso, a relação de tempo em função do número de bits (n) não parece seguir uma tendência previsível, visto que o tempo de operação de $\{2^{25}-5\}$ é maior que o de $\{2^{28}-65\}$ por exemplo.

4 | CONCLUSÃO

A partir dos resultados obtidos é possível notar que a utilização de pseudo-módulos para a substituição de módulos de baixo desempenho para melhorar o tempo de operação é realmente uma boa estratégia, uma vez que o atraso de processamento mostrou-se consideravelmente menor para os testes feitos em hardware, e também houve diminuição para as modulações feitas em software.

Apesar dessas semelhanças, o maior impacto observado com o uso dos multiplicadores deu-se em função da quantidade de bits do módulo, visto que o delay aumenta proporcionalmente ao mesmo, não havendo influência do valor de k, que é mitigada pela arquitetura utilizada. Já para o caso do script em Python, o número de 1's de k apresenta um impacto muito maior para os módulos testados, dado o pior desempenho dos módulos originais. Por outro lado, a influência do tamanho em bits do módulo e o seu tipo/formato não parece seguir nenhuma tendência, logo esses parâmetros podem não

apresentar nenhum influxo sobre o resultado.

É evidente que os valores absolutos obtidos nas simulações realizadas certamente sofrerão alterações se implementadas através de outras ferramentas, como por exemplo se for realizada a síntese do hardware em outra tecnologia, como 45 nm e 90 nm, ou se utilizada outra ferramenta para compilar o código, tal qual MATLAB, C, entre outros. No entanto, é esperado que correlação entre os tempos obtidos mantenha um padrão semelhante, de modo que as análises realizadas nesse trabalho ainda permaneçam válidas. Dessa forma evidencia-se que a presente técnica possui grande potencial para a aceleração de processes em Digital Signal Processors (DSPs) por exemplo, ou outros sistemas que possam realizar as operações requeridas diretamente em chip. Essa possibilidade de melhora também pode ser explorada em aplicações realizadas totalmente em software, como normalmente é o caso de criptografia e mineração de bitcoins, embora os resultados ainda não sejam conclusivos o suficiente para uma análise mais detalhada para esses casos.

REFERÊNCIAS

[1] B. Parhami; *Computer Arithmetic,* Vol. 1, Oxford University, 2007.

[2] N. Szabo, *Residue arithmetic and its applications to computer technology.* New York: McGraw-Hill, 1967.

[3] J. Bajard, and L. Imbert; *A Full RNS Implementation of RSA:* IEE Transactions on Computers; Vol. 53; pp. 769-774; (2004).

[4] R. Zimmermann; *Efficient VLSI implementation of modulo (2n _ 1) addition and multiplication*: Proceedings of the 14th IEEE Symposium on Computer Architecture; pp.158-167; (1999).

[5] H. Pettenghi, S. Cotofana and L. Sousa*; Efficient Method for Designing modulo multipliers*. Journal of Circuits, Systems, and Computers, v. 23, pp. 1450001(1)-1450001(20), (2013).

[6] R. Zimmermann; *Efficient VLSI implementation of modulo addition*

and multiplication: Proceedings of the 14th IEEE Symposium on

Computer Architecture; pp. 158-167; (1999).

[7] J. Bajard, and L. Imbert; *A Full RNS Implementation of RSA*: IEEE Transactions on Computers; Vol. 53; pp. 769-774; (2004).

[8] Mehrabi, M.A. *Improved Sum of Residues Modular Multiplication Algorithm. Cryptography* 2019, *3*, 14. https://doi.org/10.3390/cryptography3020014

[9] Kawamura, S., Komano, Y., Shimizu, H., & Yonemura, T. (2018). *RNS Montgomery reduction algorithms using quadratic residuosity*. *Journal of Cryptographic Engineering*, 1-19.

[10] G. Cardarilli, A. Nannarelli, M. Re; *Reducing power dissipation in FIR filters using the residue number system*: IEEE Proceedings of Midwest Symposium on Circuits and Systems; Vol. 1; pp. 320-323, (2000).

[11] C. Friedrish Gauss; *Disquisitiones arithmeticae:* New York, Springer-Verlag (1986)

ÍNDICE REMISSIVO

I

L

M

N

O

P

Q

R

S

T

V

www.atenaeditora.com.br
contato@atenaeditora.com.br
@atenaeditora
www.facebook.com/atenaeditora.com.br

Collection:

APPLIED ELECTRICAL ENGINEERING

Atena Editora
Ano 2022

www.atenaeditora.com.br
contato@atenaeditora.com.br
@atenaeditora
www.facebook.com/atenaeditora.com.br

Collection:

APPLIED ELECTRICAL ENGINEERING

Atena Editora
Ano 2022