Análise Comparativa do Custo e do Desempenho de um Algoritmo de Criptografia para Sistemas Embarcados Explorando o Particionamento Hardware/Software

Robson Sopran[1], Douglas R. Melo[1,2], Cesar A. Zeferino[1], Eduardo A. Bezerra[2]

[1]Laboratório de Sistemas Embarcados e Distribuídos
Universidade do Vale do Itajaí (UNIVALI) – Itajaí, SC – Brasil

[2]Laboratório de Comunicações e Sistemas Embarcados
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

robson.sopran@edu.univali.br, drm@univali.br,
zeferino@univali.br, eduardo.bezerra@ufsc.br

Abstract. *Encryption techniques imply in an extra cost when they are applied to large communication flows or data structures. Therefore, this work presents the comparative analysis of costs and performance of SIMON cryptography technique. This technique was implemented in FPGA by addressing different platforms, using the same algorithm in software, in hardware, and in a hardware-software partitioned approach. The results show that the fully-hardware scenario needs more silicon resources than fully-software and hardware-software solution. However, it presents lower latency and better energy efficiency.*

Resumo. *Técnicas de criptografia implicam em um sobrecusto quando aplicadas em grandes fluxos de comunicação ou estruturas de dados. Dessa forma, este projeto apresenta o desenvolvimento e análise de custo e desempenho da técnica de criptografia SIMON. A técnica foi implementada em FPGA abordando diferentes plataformas. O mesmo algoritmo foi implementado em software, hardware e em uma abordagem hardware-software. Os resultados obtidos mostram que o cenário totalmente em hardware necessita de mais recursos em silício do que as soluções totalmente em software e particionada. No entanto, apresentou uma menor latência e melhor eficiência energética.*

1. Introdução

A criptografia é uma técnica baseada em formulações matemáticas que trabalha com parâmetros chamados chaves e pode ser classificada em simétrica ou assimétrica. Algoritmos de chave simétrica utilizam uma única chave para cifrar e para decifrar a informação (codificação e decodificação). Algoritmos de chave assimétrica utilizam duas chaves: uma chave pública para cifrar a informação e uma chave privada para decifrá-la. A chave pública pode ser disponibilizada para qualquer pessoa e é geralmente utilizada para confidencialidade (cifrar mensagens), já a chave privada é conhecida somente pelo transmissor e pelo receptor da mensagem [Almeida e Mendes 2005].

Em um sistema computacional, um algoritmo executando em hardware possui um desempenho superior à sua implementação em software, porém o custo de um sistema totalmente em hardware se torna elevado. Em contrapartida, uma implementação totalmente em software pode ser ineficaz em relação ao desempenho. Com isso, uma opção intermediária e balanceada é o particionamento de um algoritmo em estruturas de hardware e de software [Vahid 2008].

Com base nessa afirmação, este trabalho realizou a análise do custo e desempenho da técnica de criptografia simétrica SIMON, uma técnica de criptografia leve desenvolvida pela NSA [Ray et al. 2013], comumente empregada na comunicação de sistemas embarcados. A avaliação consiste na comparação de custo em silício, tempo de execução, potência dissipada e consumo de energia de três diferentes abordagens: software, hardware e com particionamento hardware/software. O mesmo algoritmo foi implementado em FPGA (Field-Programmable Gate Array) com o uso de um processador embarcado para executar as partes de software e de sistemas digitais de propósito específico para implementar as funções do algoritmo em hardware.

O restante deste artigo é organizado como segue. A Seção 2 apresenta os trabalhos relacionados. Na Seção 3, Projeto Arquitetural, são apresentados os critérios de seleção do algoritmo e também os diferentes cenários em que esse foi implementado. A Seção 4 apresenta detalhes da implementação e das tecnologias adotadas. Na Seção 5 são apresentados e comparados os resultados das diferentes implementações. Por fim, na Seção 6, são expostas as conclusões e considerações finais.

2. Trabalhos Relacionados

Visando identificar e analisar as contribuições já feitas sobre o tema, foram estudados algoritmos, linguagens, ferramentas e métricas de avaliação, suas técnicas de implementação e resultados obtidos de cinco trabalhos relacionados. Esses trabalhos avaliam a latência de um ou mais algoritmos de criptografia em diferentes abordagens.

Nos trabalhos de Lazanha (2005) e Nadeem e Javed (2005) é feita a análise da latência de algoritmos em software, sendo que em Lazanha (2005) é avaliado o desempenho um único algoritmo (SERPENT) em linguagens C e Java, enquanto em Nadeem e Javed (2005) são avaliados e comparados quatro algoritmos diferentes: (*i*) DES; (*ii*) 3DES; (*iii*) AES; e (*iv*) BLOWFISH.

Em Chiaramonte e Moreno (2002), Almeida e Costa (2010) e Hasamnis et al. (2012) é feita a análise da latência de um algoritmo criptográfico em software (linguagem C) e hardware (VHDL), sendo que Chiaramonte e Moreno (2002) utilizam o algoritmo POSICIONAL (de sua autoria), Almeida e Costa (2010) utilizam o AES e o RSA, e destaca-se a implementação de Hasamnis et al. (2012), que utiliza de um cripto-processador (AES) conectado ao *softcore* NIOS II pelo barramento Avalon, sendo essa arquitetura similar à adotada neste projeto.

3. Projeto Arquitetural

3.1 Seleção do algoritmo

Comitês de pesquisa e avaliação de algoritmos estabelecem quais os melhores algoritmos, baseados em propriedades definidas por eles. Muitos destes algoritmos são

implementados em hardware, pois mantêm características que ajudam neste tipo de implementação, dentre eles encontram-se: o IDEA [Lai 1991], o AES [Daemen 1999] e vários outros algoritmos encontrados na literatura.

Para a escolha do algoritmo a ser utilizado neste trabalho, foram avaliados 20 algoritmos de criptografia simétrica utilizando como critério o ano de publicação, especificação, disponibilidade de implementação em software e hardware, e suas aplicações em sistemas embarcados. Dentre esses, foram escolhidos os cinco mais atuais para um estudo mais detalhado de seu código e funções, sendo: (*i*) SMS4 [Su et al. 2010]; (*ii*) CLEFIA [Sony 2007]; (*iii*) PRESENT [Wang 2007]; (*iv*) SIMON; e (*v*) SPECK [Ray et al. 2013]. Para a escolha do algoritmo foram observadas características como atualidade, implementações disponíveis e características de criptografia leve que possibilite o uso em sistemas restritos, incluindo etiquetas de RFID, sensores, cartões inteligentes sem contato e dispositivos portáteis em geral. Dentre os cinco, aquele que mais se adequou aos requisitos do trabalho foi o algoritmo SIMON, pois possui características de criptografia leve e mantém o foco de suas implementações em sistemas embarcados, contexto no qual este trabalho está inserido. Além disso, segundo Ray et al. (2013), o algoritmo SIMON mantém aspectos de corretude, eficiência e propriedades unidirecionais.

O algoritmo SIMON trabalha com duas funções básicas, a função de geração de subchaves, onde são definidas subchaves menores utilizando como referência a chave principal, e a função de rodada, onde são utilizadas estas subchaves para realizar a criptografia ou decriptografia dos dados.

3.2 Cenários para implementação

Para implementação do algoritmo SIMON, foram definidos três cenários diferentes: (*i*) software; (*ii*) hardware; e (*iii*) particionamento hardware/software. Em todos os casos, utiliza-se um processador para a leitura e escrita das mensagens (blocos de tamanho fixo do texto claro), e um buffer para o armazenamento das mensagens cifradas. Utilizando uma distribuição de chave simples onde o emissor manda a chave ao receptor antes da mensagem cifrada ou clara. São medidas as latências do início do processo de criptografia à escrita no buffer e da leitura do buffer ao término da decriptografia.

O processador utilizado foi o LEON3, que se trata de um modelo VHDL sintetizável de um processador de 32-bits da arquitetura SPARC, altamente configurável e adequado para sistemas embarcados, ele é totalmente parametrizável através do uso de arquivos VHDL genéricos, e não depende de qualquer pacote de configuração global [GAISLER 2016].

No primeiro cenário (Figura 1), o algoritmo é implementado totalmente em software e executado no LEON3. O processador é responsável pelo armazenamento do texto claro, a criptografia (a) por meio da função `SIMON_SW_ENC`, a decriptografia (b) por meio da função `SIMON_SW_DEC` e a transmissão e recepção da mensagem.

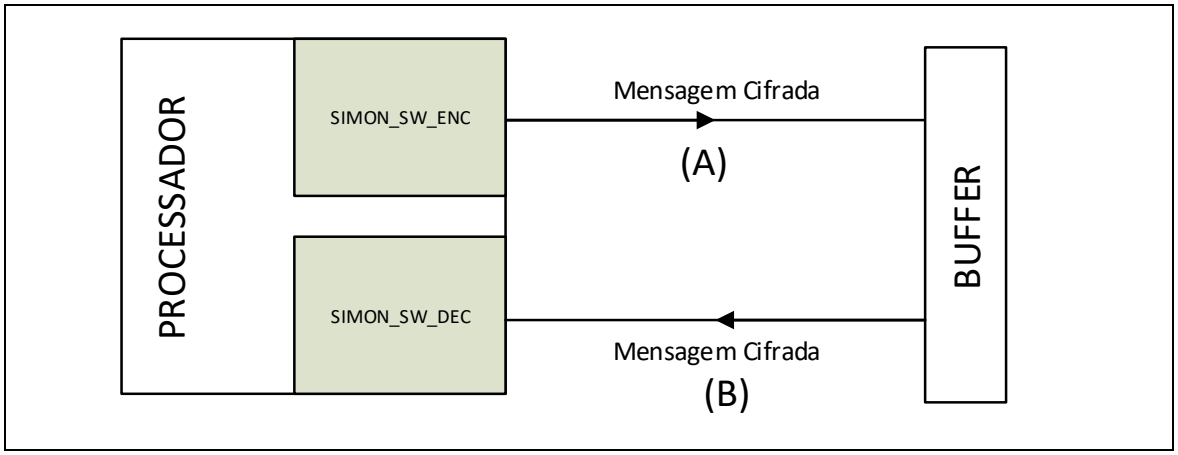

Figura 1. Solução em software: (a) criptografia; (b) decriptografia

No segundo cenário (Figura 2), a criptografia e a decriptografia são feitas por blocos de hardware que realizam todas as funções do algoritmo. O LEON3 é responsável pelo armazenamento do texto claro, bem como pela transmissão e recepção de mensagens claras. A criptografia (a) é realizada pelo bloco de hardware `SIMON_HW_ENC` e a decriptografia (b) pelo bloco `SIMON_HW_DEC`.

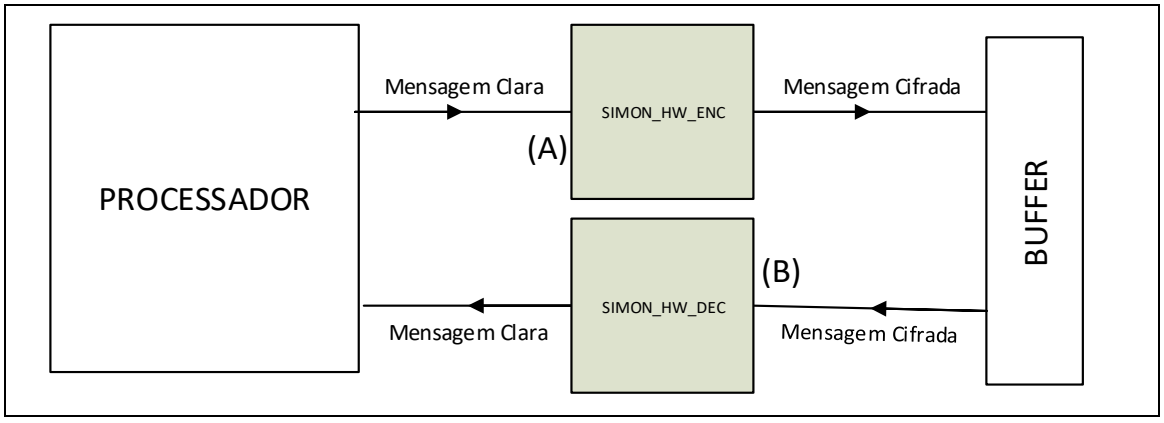

Figura 2. Solução em hardware: (a) criptografia; (b) decriptografia

No terceiro cenário (Figura 3), avalia-se o algoritmo utilizando o particionamento hardware/software, com as funções mais custosas do algoritmo executadas em hardware. O LEON3 é responsável pelo armazenamento do texto claro, a criptografia parcial (a) por meio da função `SIMON_SW_HALF_ENC`, a decriptografia parcial (b) por meio da função `SIMON_SW_HALF_DEC` e a transmissão e recepção da mensagem intermediária. O hardware é composto de dois blocos: (a) `SIMON_HW_HALF_ENC`, para a recepção da mensagem intermediária e envio da mensagem cifrada; e (b) `SIMON_HW_HALF_DEC`, para a recepção da mensagem cifrada e envio da mensagem intermediária. Nos dois casos (A e B), o software realiza a função de geração de subchaves e o hardware realiza a função de rodada, pois colocar a função de geração de subchaves em hardware resulta em um sobrecusto maior, uma vez que necessita armazenar mais variáveis intermediárias.

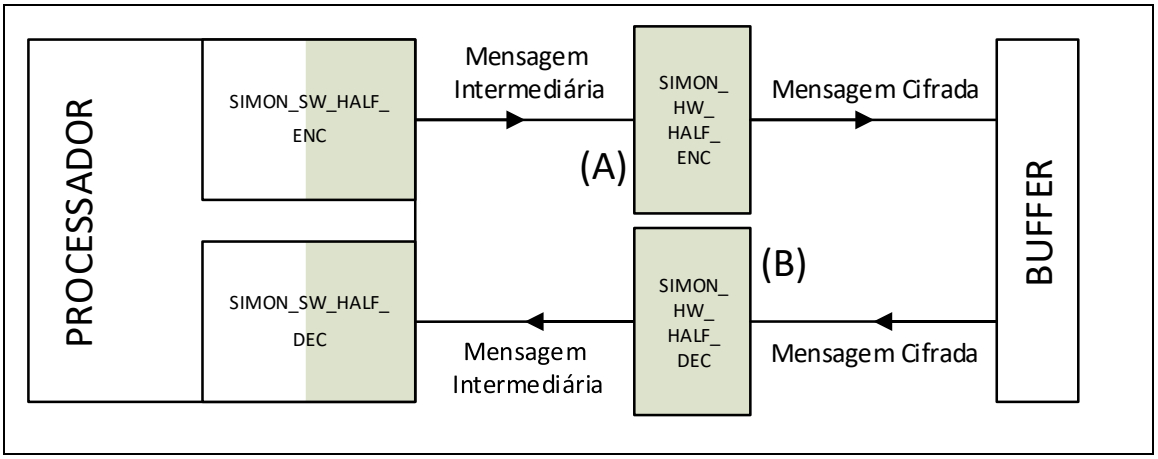

Figura 3. Solução particionada: (a) criptografia; (b) decriptografia

4. Implementação

O desenvolvimento do projeto seguiu os cenários de software, hardware e particionamento citados. O software é implementado utilizando a linguagem C, e executado em uma versão sintetizada do processador LEON3 em FPGA. Os blocos em hardware são descritos em VHDL e conectados ao barramento do processador.

Os três cenários seguem as etapas da máquina de estados ilustrada na Figura 4, explicada a seguir.

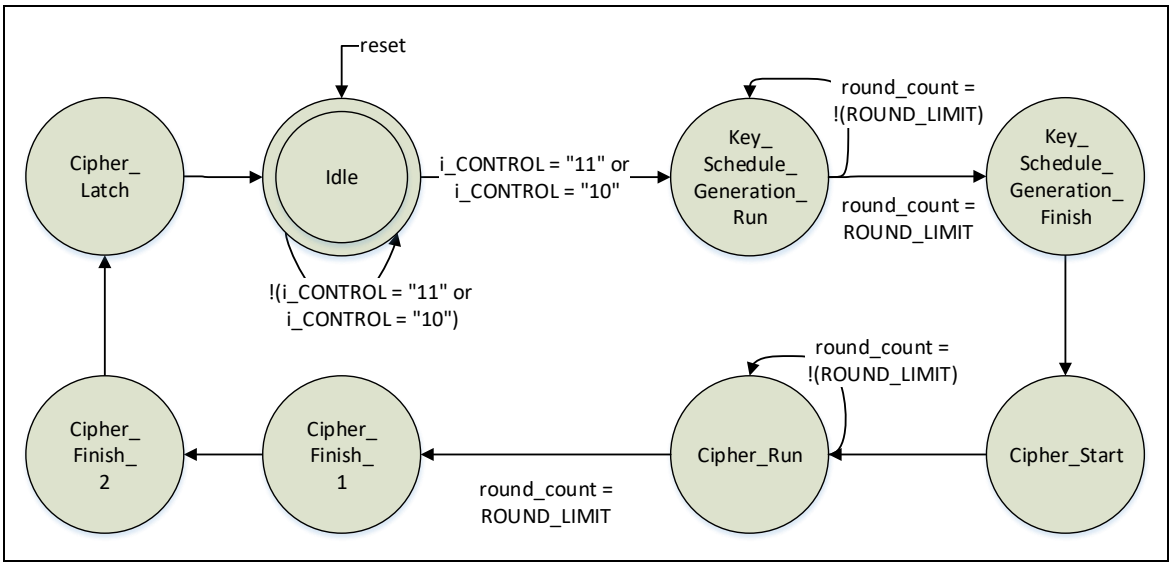

Figura 4. Máquina de estados do cripto-processador

A geração de subchaves é feita nos estados `Key_Schedule_Generation_Run` e `Key_Schedule_Generation_Finish`. Já a função rodada percorre cinco estados: (*i*) `Cipher_Start`; (*ii*) `Cipher_Run`; (*iii*) `Cipher_Finish_1`; (*iv*) `Cipher_Finish_2`; e (*v*) `Cipher_Latch`. Os estados `Key_Schedule_Generation_Run` e `Cipher_Run` ficam em laço até atingir o número de rodadas limite (32), realizando a criptografia ou decriptografia dos dados. Ao término do processo, a máquina retorna ao estado `Idle`.

4.1. Implementação em Software

A implementação em software consiste na descrição do processador e de seus elementos básicos junto ao algoritmo em alto nível do SIMON. Utiliza-se um código em linguagem C, pois este contém uma função que realiza a criptografia e decriptografia, intitulada `crypt_decrypt`. Essa função segue todos os passos da máquina de estados da Figura 4 e realiza a geração de subchaves e o procedimento de rodada.

O procedimento de geração de subchaves realiza operações de *XOR* e *AND* bit a bit e rotação à direita para gerar 32 subchaves de 16 bits, as quais são utilizadas na função rodada. O procedimento de rodada realiza um *XOR* bit a bit com pedaços da mensagem e subchaves para, em seguida, realizar uma rotação bit-a-bit à esquerda, cifrando ou decriptografando a mensagem.

4.2. Implementação em Hardware

A implementação em hardware é feita em um único bloco (`SIMON_HW`). Este mantém dois blocos em sua estrutura: (*i*) o cripto-processador `SIMON_CIPHER` [Mccoy 2016], composto de módulos que fazem a função rodada e a geração de subchaves; e (*ii*) a interface, que recebe e mantém os dados adquiridos do barramento do processador LEON3 e os encaminha ao cripto-processador, bem como o controle do mesmo. Na Figura 5 é apresentado o diagrama de blocos da implementação em hardware, nesta implementação o cripto-processador de Mccoy (2016) foi modificado para se adequar a este trabalho.

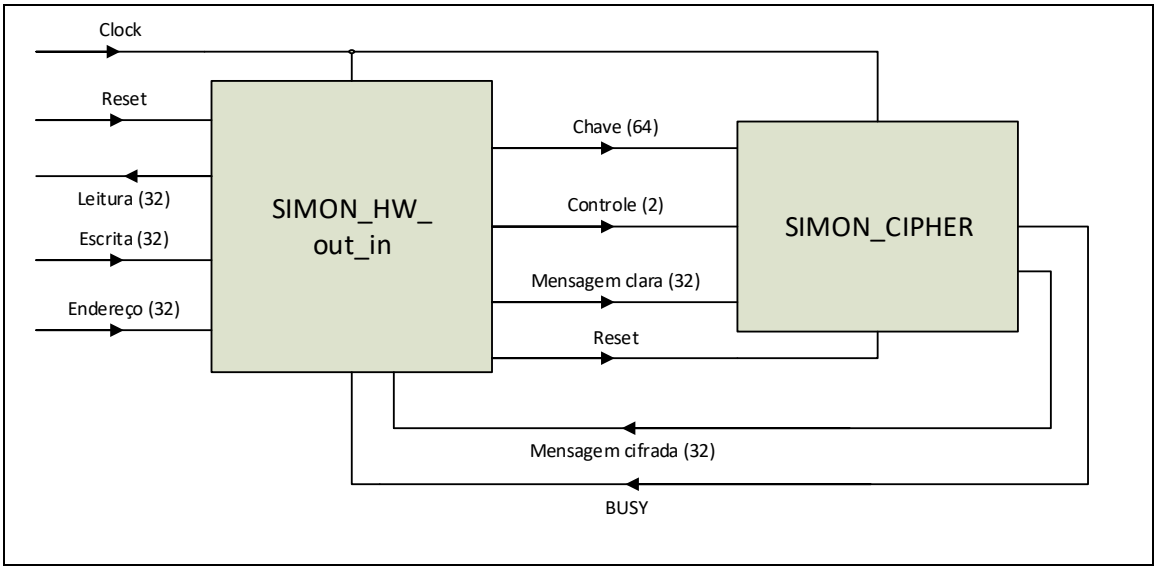

Figura 5. Diagrama de blocos do SIMON_HW

4.3. Implementação Particionada

Para realizar a implementação particionada foi escolhido o procedimento de geração de subchaves para ser executado em software e o procedimento de rodada para ser executado em hardware, em decorrência da estrutura sequencial do algoritmo. Dessa forma, as operações realizadas nos estados `Key_Schedule_Generation_Run` e `Key_Schedule_Generation_Finish` são implementadas em software e os demais estados em hardware.

A implementação particionada é feita em um único bloco que possui três outros blocos em sua estrutura: (*i*) cripto-processador, que nesta implementação realiza apenas a função de rodada; (*ii*) registradores, nos quais ficam armazenadas as subchaves adquiridas no procedimento de geração executado em software; e (*iii*) interface, que recebe e mantém os dados adquiridos do barramento do processador LEON3 e realiza o controle dos outros dois blocos. Na Figura 6 é apresentado o diagrama de blocos da implementação particionada.

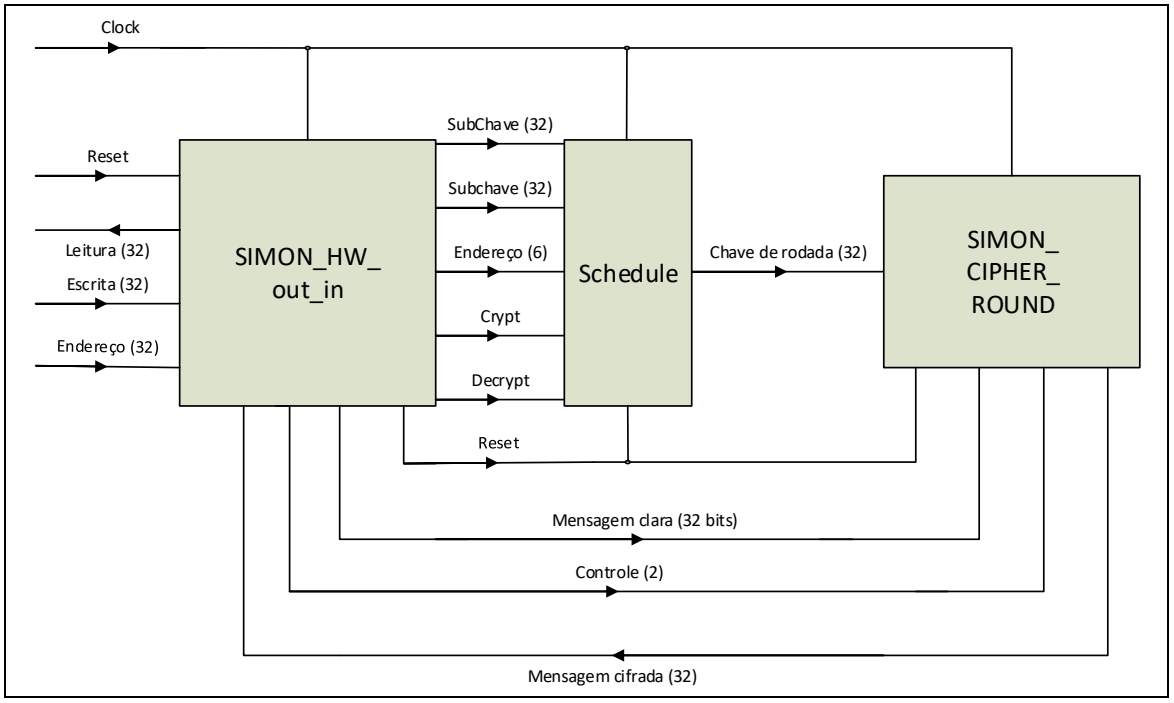

Figura 6. Diagrama de blocos do SIMON_HW_HALF

5. Resultados

Nesta seção, são apresentadas as métricas avaliadas nos três cenários implementados, sendo essas o custo em silício, o desempenho e a potência dissipada. Utilizou-se o relatório de síntese da ferramenta Quartus II e a ferramenta de *debugging* SignalTap II.

5.1. Custo de Área de Silício

Para avaliar o custo de silício do sistema, foram coletados dados do relatório de síntese e de compilação da ferramenta Quartus II. O sistema foi sintetizado para o dispositivo FPGA Cyclone II EP2C35F672C6.

Todos os cenários utilizam como base um sistema contendo o processador LEON3, memória e uma interface de gravação. A aplicação completamente em software não acrescenta nenhum custo adicional ao sistema mínimo adotado, enquanto as soluções em hardware e particionada ocupam LUTs (Look-up Tables) e FFs (Flip-Flops) adicionais.

Na Tabela1 é apresentado o custo em LUTs e FFs das três implementações. As LUTs representam a área de silício correspondente aos circuitos combinacionais, enquanto os FFs expressam o custo com a implementação de circuitos sequenciais.

Tabela 1. Custo em silício das implementações

Cenário	LUTs	FFs
Software	5506	2122
Software / Hardware	6329 (+15%)	2872 (+35,3%)
Hardware	6862 (+24,3%)	3090 (+45,6%)

Fazendo uma relação de sobrecustos de implementação em relação à realização em software, observa-se que o cenário com particionamento resulta em um aumento de 15% e de 35,3% nas quantidades de LUTs e FFs, respectivamente. Já o cenário hardware apresenta sobrecustos de 24,6% em LUTs e de 45,6% em FFs.

5.2. Desempenho

Nos três cenários, foi medido o intervalo de tempo do processo de criptografia e também do processo de decriptografia. Essa latência é medida utilizando um contador descrito em hardware, que é iniciado a partir do momento da detecção de uma transação pelo barramento e cessa quando a mensagem cifrada ou decriptografada passa por esse mesmo barramento.

A Figura 7 apresenta as latências em ciclos de relógio para a execução dos diferentes cenários de implementação do algoritmo. Quando comparados os ciclos de relógio necessários para execução das implementações observa-se que, tanto para a criptografia como para a decriptografia, a solução particionada é aproximadamente 2 vezes mais rápida que a solução em software. A solução totalmente em hardware é 1.147 vezes mais rápida que a solução particionada e 2.326 vezes mais rápida que a totalmente em software.

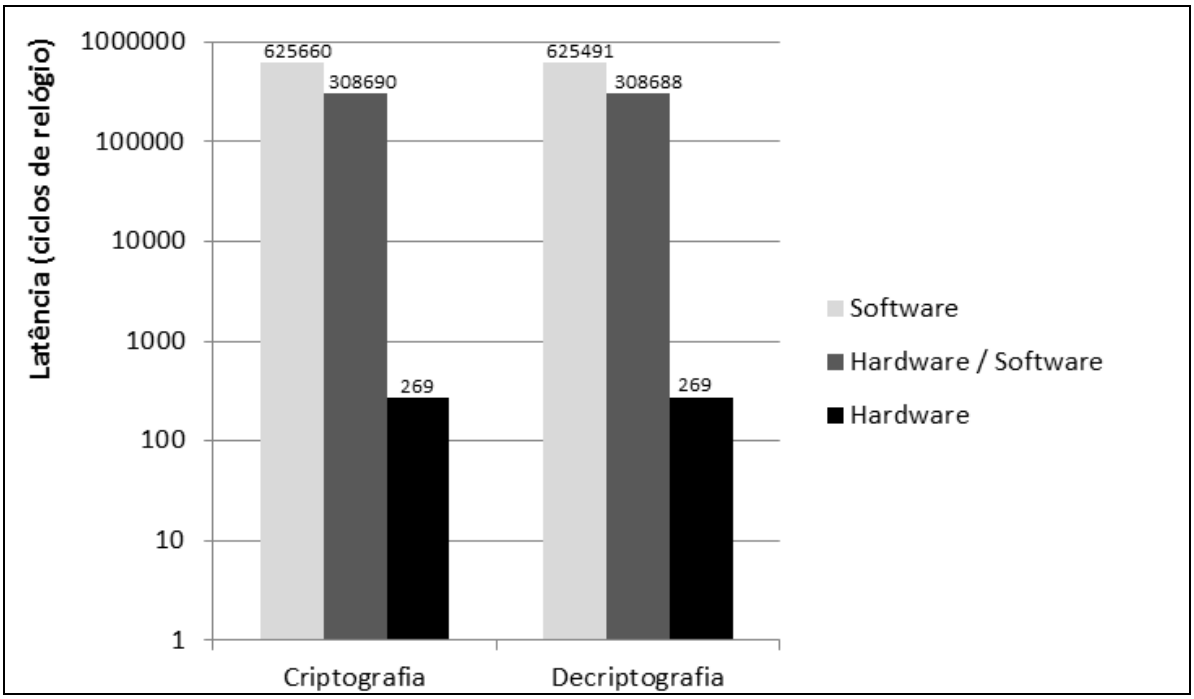

Figura 7. Desempenho nos três cenários

A frequência máxima de operação foi mantida nos três cenários, correspondendo a aproximadamente 65 MHz na plataforma sintetizada. Essa frequência é limitada pelo caminho crítico do processador LEON3, assinalando que os blocos em hardware não degradam o desempenho do sistema.

5.3. Potência dissipada

A potência dissipada (mW) pode ser descrita como a energia que se transformou em calor no decorrer de uma unidade de tempo, enquanto a energia consumida (mJ) é a potência dissipada pelo tempo em que o respectivo aparelho ficou ligado.

Para avaliar a potência dissipada, foram utilizados relatórios da ferramenta PowerPlay Power Analyzer Tool, disponível no Quartus II. Na Tabela 2, é apresentada a potência dissipada e a energia consumida nos três cenários.

Tabela 2. Potência dissipada e energia consumida (Fclk = 65 MHz)

Cenário	Potência Dissipada (mW)	Latência (ms)	Energia (mJ)
Software	273,76	19,248	5,269
Hardware/Software	294,59 (+7,6%)	9,498 (-50,75%)	2,798 (-46,9%)
Hardware	296,01 (+8,12%)	0,008 (-99,96%)	0,002 (-99,96%)

Utilizando como base a implementação em software, observa-se que o sobrecusto de potência é baixo: 7,6% para a implementação particionada e 8,12% para a implementação em hardware. Levando-se em conta o consumo de energia, a qual é função da potência dissipada e do tempo para a execução do algoritmo, observa-se que as soluções com parte ou todo o algoritmo implementado em hardware possuem maior eficiência energética que a solução em software.

6. Conclusões

Por meio dos resultados obtidos, apresentou-se a viabilidade de sistemas criptografados em hardware em relação ao seu desempenho, em software em relação ao custo e, por fim, o equilíbrio de custo e desempenho utilizando uma solução particionada.

Em relação ao custo de silício, observou-se um aumento significativo do consumo de recursos lógicos e do desempenho em hardware quando comparado com a implementação em software. Em relação à potência dissipada, ocorreu aumento do consumo na implementação particionada e na implementação em hardware em relação à implementação em software, das quais hardware e particionamento mantiveram resultados próximos. Porém, com relação à energia consumida, observou-se uma maior eficiência na solução particionada e em hardware, o que contribui para a escolha desses cenários na construção de sistemas embarcados com requisitos de baixo consumo ou alimentados por baterias. Nos três casos têm-se que a implementação particionada manteve o equilíbrio entre custo, desempenho e consumo, assim como fora previsto.

As contribuições deste trabalho são constituídas das análises comparativas das implementações (software, hardware e particionada) de forma que as mesmas podem ser utilizadas em projetos futuros e trabalhos correlatos. Como sugestão de trabalhos futuros, fica a integração desta técnica de criptografia em uma interface de rede para a comunicação de sistemas integrados baseados em Redes-em-Chip e também o

aprimoramento do processo de distribuição de chaves, uma vez que a mesma pode ser interceptada ou falsificada.

Referências

Almeida, A. A; Costa, V.G. (2010) “Criptografia em hardware com VHDL usando circuitos FPGA× criptografia em software”, http://www.enacomp.com.br/2010/cd/artigos/resumidos/enacomp2010_24.pdf, Maio.

Almeida, J. R. C; Mendes, M. D. N. C. (2016) “Criptografia em sistemas distribuídos”, http://www3.iesam-pa.edu.br/ojs/index.php/sistemas/article/view/501/401, Maio.

Chiaramonte, R. B; Moreno, E. D. (2002) “Desempenho de um algoritmo posicional em software e hardware”, In: SSI-02 (IV Simpósio Segurança em Informática), São José dos Campos.

Daemen, J.; Rijmen, V. (1999) “AES proposal: Rijndael”, http://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/rijndael_doc_V2.pdf, Novembro.

Gaisler, C. (2016) “LEON3 Processor”, http://www.gaisler.com/index.php/products/processors/leon3, Outubro.

Hasamnis, M. et al. (2012) “Implementation of aes as a custom hardware using Nios II processor”, http://airccse.org/journal/acij/papers/0712acij10.pdf, Maio.

Lai X., Massey J.L. (1991) “A Proposal for a New Block Encryption Standard”. In: Damgård I.B. (eds) Advances in Cryptology — EUROCRYPT ’90. EUROCRYPT 1990. Lecture Notes in Computer Science, vol 473. Springer, Berlin, Heidelberg

Lazanha, F. (2005) “Implementação e desempenho do algoritmo criptográfico “Serpent”“, http://aberto.univem.edu.br/handle/11077/379, Maio.

Mccoy, C. (2016) “Simon_Speck_Ciphers”, https://github.com/inmcm/Simon_Speck_Ciphers/tree/master/VHDL, Novembro.

Nadeem, A; Javed, M. Y. (2005) “A performance comparison of data encryption algorithms”, In: 2005 International Conference on Information and Communication Technologies. IEEE, p. 84-89.

Ray, B et al. (2013) “The SIMON and SPECK Families of Lightweight Block Ciphers”, https://eprint.iacr.org/2013/404.pdf, Julho.

Sony. (2007) “ Sony develops "CLEFIA": new block cipher algorithm based on state-of-the-art design technologies”, http://www.sony.net/SonyInfo/News/Press/200703/07-028E/index.html, Maio.

Su, B; Wu, W; Zhang, W. (2010) “Differential cryptanalysis of SMS4 block cipher”, http://eprint.iacr.org/2010/062.pdf, Maio.

Vahid, F. (2008) “Sistemas digitais: projeto, otimização e HDLs”, Tradução Anatólio Laschuk – Porto Alegre: Artmed.

Wang, C; Heys, H. M. (2009) “An ultra compact block cipher for serialized architecture implementations”, In: Canadian Conference on Electrical and Computer Engineering - CCECE 2009, IEEE 1085–1090.