CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

IMPACTO DA METODOLOGIA DE IMPLEMENTAÇÃO NO DESEMPENHO EM FPGA DOS ALGORITMOS CRIPTOGRÁFICOS LEVES SIMON E SPECK

CLAUDIO ROBERTO COSTA

MARÍLIA
2016

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

IMPACTO DA METODOLOGIA DE IMPLEMENTAÇÃO NO DESEMPENHO EM FPGA DOS ALGORITMOS CRIPTOGRÁFICOS LEVES SIMON E SPECK

Monografia apresentada ao Centro Universitário Eurípides de Marília como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador
Prof.: Dr. Fábio Dacêncio Pereira

MARÍLIA
2016

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
MANTIDO PELA FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Cláudio Roberto Costa

IMPACTO DA METODOLOGIA DE IMPLEMENTAÇÃO NO DESEMPENHO EM FPGA DOS ALGORITMOS CRIPTOGRÁFICOS LEVES SIMON E SPECK.

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 9,0 (nove)

Orientador:Fábio Dacêncio Pereira _____

1°.Examinador:Rodolfo Barros Chiaramonte _____

2°.Examinador:Guilherme Rodrigues Bilar Guilherme R. Bilar

Marília, 05 de dezembro de 2016.

COSTA, Claudio Roberto

Impacto da Metodologia de Implementação no Desempenho em FPGA dos Algoritmos Criptográficos Leves Simon e Speck / Claudio Roberto Costa, orientador: Profº. MSc. Dr. Fábio Dacêncio Pereira, SP: [s.n.], 2016.

37 folhas

Monografia (Bacharelado em Ciência da Computação): Centro Universitário Eurípides de Marília.

Dedico este trabalha a minha família, que é o bem mais precioso que possuo, sem ela não teria alcançado este objetivo.

AGRADECIMENTOS

Agradeço a minha família, minha mãe, meu pai por estarem sempre ao meu lado em todos os momentos da minha vida, me dando o devido apoio para a realização dessa graduação, meu professor orientador Fabio Dacêncio Pereira por ter estimulado e fornecido as bases de minha pesquisa, tanto de iniciação científica quanto de trabalho de conclusão de curso, jamais teria chegado tão longe sem o seu apoio.

RESUMO

Ao longo dos anos a segurança da informação tem obtido muitos avanços, diversas técnicas têm sido desenvolvidas e/ou aperfeiçoadas para garantir que pessoas não autorizadas não tenham acesso às informações. Uma destas técnicas é a criptografia, que pode ser entendida como um conjunto de métodos e técnicas para cifrar ou codificar informações legíveis por meio de um algoritmo. Este trabalho consiste na pesquisa, implementação e comparação dos algoritmos criptográficos de cifras de bloco leve, Simon e Speck em relação a área, desempenho e consumo de energia quando projetados em FPGA. Foram propostas três arquiteturas para a implementação destes algoritmos, sendo duas baseadas em circuitos síncronos e uma em circuito assíncrono. Ao final é gerada uma relação vazão/área para avaliar o melhor custo benefício entre as arquiteturas e o seu posicionamento em relação a implementações encontradas na literatura.

Palavras-Chave: Cifras de bloco leve; Análise de Desempenho em FPGA; Simon e Speck

ABSTRACT

Over the years information security has made many advances, various techniques have been developed and / or improved to ensure that unauthorized persons do not have access to information. One such technique is encryption, which can be understood as a set of methods and techniques for encrypting or encoding readable information by means of an algorithm. This work consists of the research, implementation and comparison of the cryptographic algorithms of lightweight block ciphers, Simon and Speck in relation to area, performance and power consumption when projected in FPGA. Three architectures were proposed for the implementation of these algorithms, being two based on synchronous circuits and one on asynchronous circuit. At the end is generated a relation throughput/area to assess the best cost benefit among architectures and their positioning in relation to implementations found in the literature.

Keywords: Lightweight Block Ciphers; Performance Analysis in FPGA; Simon and Speck

LISTA DE ILUSTRAÇÕES

Figura 1 Classificação do Conteúdo Digital 13
Figura 2 - Processo de criptografia simétrica. 18
Figura 3 - Processo de criptografia assimétrica. 19
Figura 4 Estrutura da cifra de Feistel 21
Figura 5 Classificação por estrutura dos algoritmos considerados como cifras leves 22
Figura 6 Algoritmo Simon: Uma rodada de execução. 23
Figura 7 Algoritmo Speck: Uma rodada de execução. 24
Figura 8 Circuito Combinacional 29
Figura 9 Circuito Sequencial 30
Figura 10 Exemplo de Circuito Síncrono com uso de Variáveis 32
Figura 11 Exemplo de Circuito Síncrono com uso de Sinais 33
Figura 12 Representação da Implementação SSR. 34
Figura 13 Representação da Implementação SFR. 35
Figura 14 Representação da Implementação AFR. 37
Figura 15 Consumo de Energia Simon SFR e AFR 40
Figura 16 Consumo de Energia Simon SSR 41
Figura 17 Consumo de Energia Speck SFR e AFR 43
Figura 18 Consumo de Energia Speck SSR 43

LISTA DE TABELAS

Tabela 1 – Comparações de performance entre diferentes implementações em FPGA.....27
Tabela 2 – Comparações de área e vazão entre algoritmos encontrados no estado da arte em FPGA.....28
Tabela 3 – Estatística: Área e Desempenho de Simon.....39
Tabela 4 – Relação Área / Vazão Simon.....39
Tabela 5 – Consumo de Energia de Simon.....40
Tabela 6 – Análise Área e Desempenho de Speck.....41
Tabela 7 – Relação Área/ Vazão Speck.....42
Tabela 8 – Consumo de Energia de Speck.....42
Tabela 9 – Comparação da vazão do Simon e Speck.....43
Tabela 10 – Comparação da vazão do Simon e Speck com trabalhos correlatos.....44

LISTA DE ABREVIATURAS E SIGLAS

AES	Advances Encryption Standard
AFR	Async Full Rounds
ASICs	Application Specific Integrated Circuits
DES	Data Encryption Standard
FPGA	Field Programmable Gate Array
NBS	National Bureau of Standards
SFR	Sync Full Rounds
SSR	Sync Single Rounds
RFID	Radio Frequency Identification
XPA	XPower Analyzer

SUMÁRIO

1 INTRODUÇÃO 13

1.1 Objetivos 14

1.2 Metodologia 15

2 CONTEXTUALIZAÇÃO 16

2.1 O que é Criptografia? 16

2.2 Criptografia quanto ao Tipo das Operações 17

2.2.1 Substituição 17

2.2.2 Transposição ou Permutação 18

2.3 Criptografia quanto ao Número de Chaves 18

2.3.1 Criptografia Simétrica 18

2.3.2 Criptografia Assimétrica 19

2.3.3 Funções de Hash 19

2.3.4 Criptografia Homomórfica 19

2.4 Criptografia quanto ao Modo de Processamento 20

2.4.1 Cifras de Fluxo 20

2.4.2 Cifras de Blocos 20

2.4.3 Cifras Feistel (base Simon e Speck) 20

2.5 Cifras de Bloco Leve 21

2.6 Algoritmo Simon 22

2.7 Algoritmo Speck 23

2.8 Considerações finais do capítulo 24

3 TRABALHOS CORRELATOS 26

3.1 “Simon and Speck Block Ciphers for the Internet of Things” 26

3.2 “SpecTre: A Tiny Side-Channel Resistant Speck Core for FPGAs” 27

3.3 Considerações Finais 28

4 PROPOSTA DE ARQUITETURA PARA IMPLEMENTAÇÃO DOS ALGORITMOS SIMON E SPECK 29

4.1 Circuito Combinacional 29

4.2 Circuito Sequencial 29

4.3 VHDL (Very Hardware Description Language) 30

4.4 Sync Single Rounds (SSR) 34

4.5 Sync Full Rounds (SFR) 35

4.6 Async Full Rounds (AFR) 36
4.7 Considerações Finais 37
5 RESULTADOS E COMPARAÇÕES COM TRABALHOS CORRELATOS 38
5.1 Simon 32 38
5.2 Speck 32 41
5.3 Comparação com trabalhos correlatos. 43
6 CONCLUSÃO 45
REFÊRENCIAS 46

1 INTRODUÇÃO

Com os constantes avanços nos dispositivos computacionais a quantidade de conteúdo digital que é armazenada e transmitida é significatifamente crescente. Em cenários como: IoT (*Internet of Things*), HPC (*High-performance computing)* ou sistemas embarcados, o volume de informações pode ser ainda mais crítico. “O crescimento de aplicações de IoT é crescente, com necessidade de redes robustas, seguras, interoperáveis e escaláveis” (BEECHER, 2016, tradução nossa). Técnica de segurança como a criptografia, asseguram alguns serviços de proteção sobre esses conteúdos, como: confidencialidade, integridade e autenticidade. (MENEZES, OORSCHOT e VANSTONE, 1996)

No entanto, existem vários algoritmos de criptografia reconhecidos pela comunidade científica, com grau de complexidade diferente e exigências distintas relacionadas ao esforço computacional. Sendo assim, como adotar o método mais adequado para proteger as informações ?

Para isso é usual classificar e separar o conteúdo digital em informações ou dados, onde as informações são definidas como objetos, contendo valor ou significado para que sejam protegidos, assim, os demais objetos não necessitam de um grau de proteção por meio da criptografia. A figura 1 mostra a classificação do conteúdo digital.

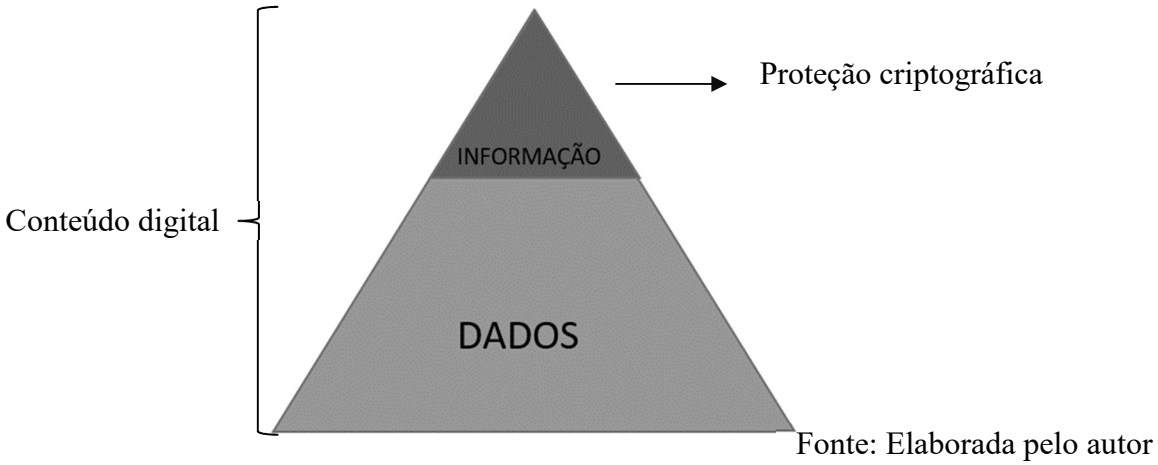


Figura 1 Classificação do Conteúdo Digital

Atualmente as informações são classificadas conforme os critérios de cada organização ou pessoa. O tipo de serviço de proteção mais utilizado para a segurança desses dados é a confidencialidade, resultando em um esquema de classificação por níveis críticos de proteção sendo elas: informação confidencial, restrita, uso interno e pública. (ISO/IEC-27001, 2005)

No esquema de classificação do conteúdo digital exposto, apenas uma parte do conteúdo digital classificado como informações são protegidas por algoritmos de criptografia. Contudo,

basta que no processo de identificação dos dados e informações ocorram erros, equívocos ou descuido para que estas sejam categorizadas erroneamente e consequentemente não recebam a proteção adequada por meio da criptografia.

Como proteger a grande parte do conteúdo digital que está em transmissão ou armazenado em sistemas computacionais sem degradar o desempenho computacional? (MORENO, 2013)

Ao longo dos anos surgiram vários algoritmos criptográficos para atender a necessidade de proteção dessa grande massa de dados, tais como Simon , Speck (BEAULIEU, SHORS, *et al.*, 2013), Tea (WHEELER e NEEDHAM, 1995), Twine (SUZAKI, *et al.*, 2011), LBlock (WU e ZHANG, 2011), LED (GUO, PEYRIN, *et al.*, 2011), entre outros.

Neste trabalho apresentam-se dois algoritmos de cifras de bloco leve, Simon e Speck (BEAULIEU, *et al.*, 2013), ambos se adéquam a uma grande variedade de aplicações como: Internet of Things – IoT (Internet das coisas), computação pervasiva e/ou ubíqua, sistemas embarcados ou em cenários onde o hardware disponível é limitado em relação a capacidade de processamento, memória e consumo de energia.

Os algoritmos Simon e Speck são explorados neste trabalho abordando diferentes metodologias de implementação em circuitos programáveis FPGA com o objetivo de analisar aspectos que circundam a relação entre desempenho, área e consumo de energia.

1.1 Objetivos

O objetivo principal é o estudo e implementação de criptografia de cifras de bloco leve com ênfase para os algoritmos Simon e Speck, utilizando diferentes metodologias de desenvolvimento, visando a comparação de área, desempenho e consumo de energia. Ao final foram comparados com resultados encontrados na literatura científica. Para isso foram definidos os seguintes objetivos específicos:

- Pesquisa de trabalho correlatos
- Definição de diferentes tipos de arquiteturas
- Estudo e implementação dos Algoritmos criptográficos de cifras de bloco leve Simon e Speck nas arquiteturas propostas
- Análise dos resultados das implementações e comparação com trabalhos correlatos

1.2 Metodologia

A metodologia de desenvolvimento deste trabalho foi dividida em 7 passos, sendo:

1. Pesquisa de trabalhos correlatos: Nesta etapa foi realizado pesquisa de trabalhos correlatos no estado da arte nos principais acervos e bibliotecas, tais como IEEE, ACM, Eprint, entre outros. Esta pesquisa foi realizada com intuito de averiguar a relevância do trabalho realizado e fazer comparações com os resultados.
2. Estudo dos Algoritmos Simon e Speck: O estudo e entendimento de cada etapa dos algoritmos (geração de chaves, cifração e decifração) foi de suma importância para a etapa de implementação.
3. Implementação dos algoritmos nas Arquiteturas: Foram propostas três arquiteturas *Sync Single Rounds* (SSR)*, Sync Full Rounds* (SFR) e *Async Full Rounds* (AFR), com finalidade comparar os resultados quanto ao desempenho, área consumida e consumo de energia. As arquiteturas serão definidas com maiores detalhes no capítulo 4.

   A seguir são expostos breves comentários sobre cada arquitetura proposta.

   - SSR: Uma proposta de arquitetura síncrona, que para cada fase de execução do algoritmo é necessário um ciclo de *clock*, podendo gerar uma menor área.
   - SFR: Uma proposta de arquitetura síncrona, onde toda as etapas do algoritmo são executadas em um único ciclo de *clock*, porém pode gerar uma maior área.
   - AFR: Uma proposta de arquitetura assíncrona, todas as etapas do algoritmo ocorrem de forma sequencial, dependendo apenas da logica de execução do circuito combinacional.
4. Análise e comparação em relação:
   - Área: Quantidade de *SlicesLuts* consumido.
   - Desempenho: Para o desempenho foram utilizadas duas métricas: a frequência máxima de trabalho e o período mínimo de execução.
   - Consumo de energia: O consumo de energia é a quantidade de *Watts* que é consumido pela execução do algoritmo.

2 CONTEXTUALIZAÇÃO

Neste capítulo, apresentam-se as definições de alguns autores sobre o que é a criptografia e qual o seu propósito, como os algoritmos criptográficos são geralmente classificados quanto a estrutura e tipo, são apresentados os algoritmos considerados de cifras de bloco leve com ênfase nos algoritmos Simon e Speck, enfoques deste trabalho.

2.1 O que é Criptografia?

O termo Criptografia surgiu da fusão das palavras gregas "*Kryptós*" e "*gráphein*", que significam "oculto" e "grafia", respectivamente (FERREIRA, 2003). Muitos autores definem a criptografia, seguem algumas definições.

Segundo (SIMON, 1999)

> A criptografia pode ser entendida como um conjunto de métodos e técnicas para cifrar ou codificar informações legíveis por meio de um algoritmo, convertendo um texto original em um texto ilegível, sendo possível por mediante um processo inverso recuperar as informações originais.

De acordo com (TERADA, 2011)

> Algoritmos criptográficos basicamente objetivam “esconder” informações sigilosas de qualquer pessoa desautorizada a lê-las, isto é, de qualquer pessoa que *não* conheça a *chave* secreta de criptografia.

Segundo (FERREIRA, 2003)

> É a arte ou ciência de escrever em cifra ou em códigos, de forma a permitir que somente o destinatário a decifre e compreenda, ou seja, criptografia transforma textos originais chamados plaintext ou texto claro, em uma informação transformada, chamada texto cifrado ou ciphertext, que usualmente tem aparência de um texto randômico ilegível.

Para (STALLINGS, 2012) é o estudo do projeto de técnicas para garantir o sigilo e/ou a autenticidade da informação.

A criptografia é uma ferramenta que permite evitar a interceptação, manipulação e a falsificação de dados. As diferentes técnicas ou combinações de técnicas abordadas neste capítulo visam garantir esses requisitos.

Ao longo dos anos quatro grupos de pessoas contribuíram para a arte da criptografia: os militares, os diplomatas, as pessoas que gostavam de guardar memórias e os amantes, entre eles os militares tiveram maior influência, tendo definido a base para a tecnologia (TANENBAUM, 2003).

A seguir são expostos alguns termos utilizados pela criptografia.

- Texto claro: mensagem original.
- Texto cifrado ou Cifra: é o texto claro após ser processado pelo algoritmo de criptografia. Ele varia de acordo com o texto claro ef chave, cada conjunto de chave e texto claro produz uma cifra diferente.
- Algoritmo de cifração: algoritmo que realiza transformações no texto claro.
- Algoritmo de decifração: é basicamente o algoritmo de criptografia executado de modo inverso, fazendo com que o texto cifrado volte a ser o texto claro.
- Chave: é uma entrada do algoritmo de criptografia, é independente do algoritmo e do texto claro, cada chave produz uma saída diferente no algoritmo.

Os algoritmos criptográficos são caracterizados em três dimensões diferentes: 1 - O tipo das operações usadas para transformar o texto claro em texto cifrado: *Substituição e Transposição ou Permutação*. 2 - O número de chaves usadas: criptografia simétrica ou de chave única e criptografia assimétrica ou de chave pública. 3 - O modo como o texto claro é processado: Cifra de bloco e Cifra de fluxo (STALLINGS, 2012). Esses termos estão mais detalhados a seguir.

2.2 Criptografia quanto ao Tipo das Operações

Na criptografia as operações devem obedecer ao requisito fundamental de que nenhuma informação deve ser perdida. Os algoritmos criptográficos baseiam-se em dois princípios: Substituição e Transposição ou Permutação (STALLINGS, 2012).

Para (TANENBAUM, 2003) os algoritmos criptográficos que utilizam técnicas de Substituição e Transposição são considerados como criptografia tradicional

2.2.1 Substituição

Uma técnica de substituição é aquela em que letras de texto claro são substituídas por outras letras ou por números ou símbolos. Se o texto claro for visto como uma sequência de bits, então a substituição envolve substituir padrões de bits de texto claro por padrões de bits de texto cifrado (STALLINGS, 2012).

2.2.2 Transposição ou Permutação

Nesta cifra a letra ou bloco de letras é substituído por uma letra ou bloco de letras diferentes, desta forma o texto fica ilegível (TERADA, 2011) .

Quando os algoritmos criptográficos utilizam dessas duas técnicas eles podem classificados como Cifras de Substituição e Permutação.

2.3 Criptografia quanto ao Número de Chaves

Os algoritmos criptográficos podem ser divididos em dois grandes grupos Criptografia Simétrica e Criptografia assimétrica. Este tipo de criptografia com uso de chaves é considerado como criptografia moderna, porém utilizam as mesmas técnicas (Substituição e Transposição ou Permutação) da criptografia tradicional. Tendo como objetivo tornar o algoritmo de criptografia complexo para que mesmo que um atacante (pessoa maliciosa) obtenha parte do texto cifrado, sem a posse da chave não consiga captar o sentido desta parte do texto (TANENBAUM, 2003).

Nesta seção também são expostos dois outros tipos de criptografia: Funções de Hash e Criptografia Homomórfica.

2.3.1 Criptografia Simétrica

No processo de cifrar e decifrar as mensagens é utilizado uma única chave, sendo assim um remetente deve primeiramente enviar a chave para o destinatário para que quando a mensagem for recebida, o destinatário possa fazer o processo de decifrar a mensagem. A figura 2 ilustra o processo de criptografia simétrica.

Fonte: (DEVMEDIA)

Figura 2 - Processo de criptografia simétrica.

2.3.2 Criptografia Assimétrica

Diferente da criptografia simétrica que utiliza apenas uma chave a criptografia assimétrica utiliza duas chaves, uma privada que deve ser mantida em segredo e outra pública que pode ser distribuída. A chave pública é utilizada na cifração e a chave privada é utilizada na decifração. O algoritmo RSA permite que com qualquer uma das duas chaves relacionadas pode ser usada para a criptografia, com a outra usada para a decriptografia (STALLINGS, 2012). A figura 3 ilustra o processo de criptografia assimétrica com uso da chave pública para a cifração e a chave privada para a decifração.

Fonte: (DEVMEDIA)

Figura 3 - Processo de criptografia assimétrica.

2.3.3 Funções de Hash

A função de hash resume um texto claro de tamanho variável (h) e produz como saída um texto resumido de tamanho fixo, conhecido como **código de hash** H(M). As funções de hash não utilizam chave, são apenas uma síntese da mensagem (texto claro). O código de hash também é conhecido como **síntese da mensagem** ou **valor de hash**. O código de hash permite a detecção de erros, uma vez que a alteração de qualquer bit (ou conjunto de bits) do texto claro produz um código hash diferente (STALLINGS, 2012).

2.3.4 Criptografia Homomórfica

É dita como criptografia homomórfica se ela permite realizar operações computacionais sobre os textos cifrados, de tal forma que o resultado seja um texto cifrado dessas mesmas operações computacionais executadas sobre o texto claro (SANTOS, 2014).

2.4 Criptografia quanto ao Modo de Processamento

Quanto ao modo de processamento os algoritmos criptográficos se dividem em dois tipos: Cifras de Fluxo e Cifras de Blocos.

2.4.1 Cifras de Fluxo

As mensagens são cifradas bit a bit, em um fluxo contínuo, sem espera por blocos completos. Também conhecida como criptografia em *stream* de dados, onde a criptografia se dá mediante uma operação (XOR, AND, entre outras) entre o bit de dados e o bit gerado pela chave (MORENO, PEREIRA e CHIARAMONTE, 2005). Um exemplo simples e clássico de cifras de fluxo, é a cifra de César (do Império Romano).

2.4.2 Cifras de Blocos

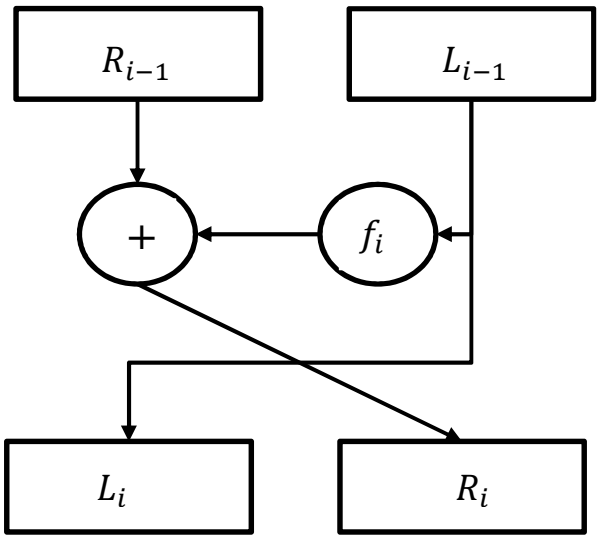
A cifra de bloco é uma função que mapeia blocos de *n*-bits de texto não cifrado para blocos de *n*-bits de texto cifrado, sendo *n* o comprimento do bloco. A função é parametrizada por *k*-bits da chave K contendo o mesmo tamanho do bloco, assim evitando a expansão dos dados (MENEZES, OORSCHOT e VANSTONE, 1996).

2.4.3 Cifras Feistel (base Simon e Speck)

A maioria dos algoritmos de cifras de blocos possuem uma estrutura descrita por Horst Feistel da IBM em 1973 (FEISTEL, 1973). Os passos do bloco de Feistel é descrito a seguir:

- As entradas dos algoritmos são divididas em duas partes iguais *Left* e *Right*;
- as duas partes sofrem *n* iterações de processamento para gerar o texto cifrado;
- cada iteração recebe *Left*-1 e *Right*-1 derivadas da iteração anterior e também uma sub-chave Ki derivada da chave inicial K;
- as sub-chaves são geradas por um algoritmo gerador de chaves, fazendo com que cada chave seja diferente;
- todas as iterações têm a mesma estrutura e são chamadas de ***round*** (rodada);

Para cada *round*, aplica-se uma função denominada ***round function*** e operação de XOR bit a bit, como demonstra a figura 4:



Fonte: Elaborada pelo autor

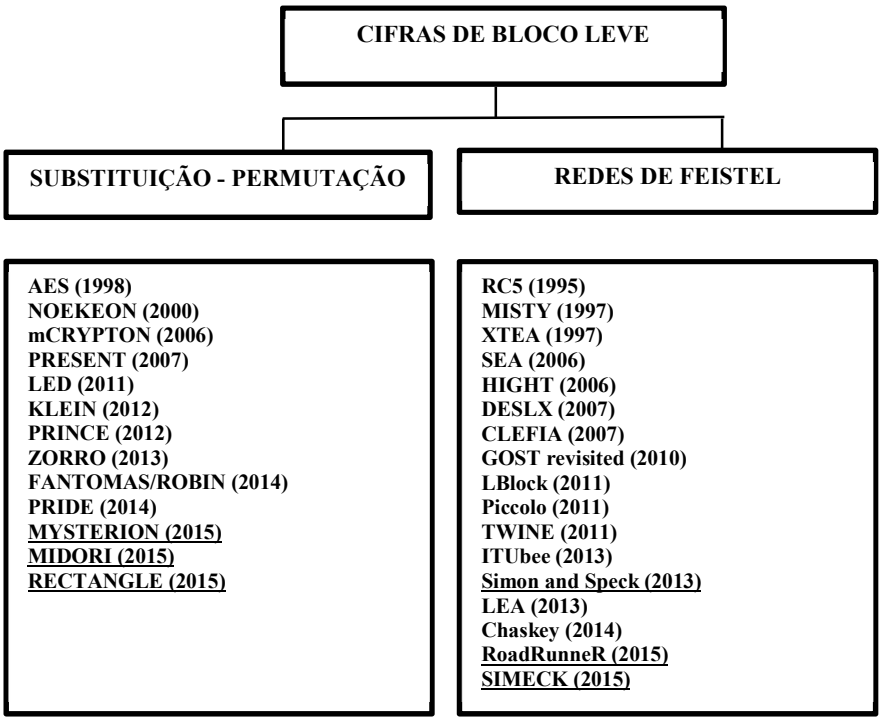
Figura 4 Estrutura da cifra de Feistel

Os algoritmos Simon e Speck enfoques deste trabalho, utilizam em suas rodadas as cifras de Feistel.

No ano 1977, o NBS (*National Bureau of Standards*) estabeleceu o primeiro algoritmo criptográfico padrão com base em cifras simétricas de blocos, o *Data Encryption Standard* – DES. Esse algoritmo possui características de cifra em bloco, baseados na função de Feistel e em redes de substituição-permutação, atualmente base de algoritmos criptográficos considerados como Cifras de bloco leve.

2.5 Cifras de Bloco Leve

As cifras de bloco leve operam em blocos de tamanhos fixos, que variam dependendo do algoritmo, geralmente de tamanho 64 e 128 bits. Pode-se classificar as cifras de bloco leve como “Redes de Substituição-Permutação” e “Redes de Feistel”, essa segunda utilizada nos algoritmos Simon e Speck, como pode ser observado na figura 5.



Fonte: Elaborada pelo autor

Figura 5 Classificação por estrutura dos algoritmos considerados como cifras leves.

Como pode ser observado (sublinhado) na figura 5, recentemente, novos algoritmos de cifras leve foram criados, como é o caso do Simon e Speck, e também outros como: Simeck, RoadRunnner, Rectangle, Midori, Mysterion. Isso realça a demanda deste tipo de proteção no cenário atual dos sistemas computacionais. Os algoritmos Simon e Speck são explicados com maiores detalhes nas seções seguintes.

2.6 Algoritmo Simon

O bloco de cifra do algoritmo Simon é composto por uma palavra de *n* bits (portanto um bloco 2*n*-bits), este é denotado como Simon2n, onde *n* pode ser 16, 24, 32, 48 ou 64. Simon2n possui m palavras chaves (*mn*-bit) que se refere a como Simon2n/mn. Por exemplo, Simon64/128 refere-se à versão de Simon em blocos de texto simples de 64 bits, usando uma chave de 128 bits. (BEAULIEU, *et al.*, 2013)

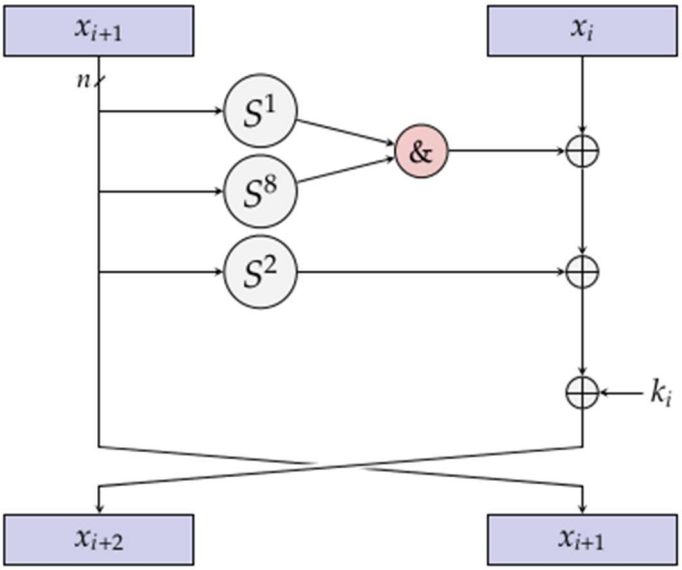
Esse algoritmo utiliza as operações XOR ($\oplus$) bit a bit, AND (&) bit a bit e descolamento circular de bits (S^n) nas suas rodadas de cifração (1) e decifração (2), onde *l* e *r* são as duas partes que compõem o texto claro (esquerda e direita respectivamente) e *k* é a chave da rodada. Cada rodada do algoritmo é expressada na equação 1:

$$R(l, r, k) = ((S1(l) \,\&\, S8(l)) \oplus S2(l) \oplus r \oplus k, l) \quad (1)$$

A rodada inversa do algoritmo é expressada na equação 2:

$$R^{-1}(l, r, k) = (r, (S1(r) \,\&\, S8(r)) \oplus S2(r) \oplus l \oplus k) \quad (2)$$

A figura 6 demonstra uma rodada (*round*) de Simon.



Fonte: (BEAULIEU, SHORS, *et al.*, 2013)

Figura 6 Algoritmo Simon: Uma rodada de execução.

Para as operações de cifração e decifração são necessárias T (T depende do tamanho de n) rodadas, e consequentemente a geração das chaves para cada rodada a partir da expansão da chave inicial. Exceto a chave k_i de cada rodada, todas as rodadas de Simon são exatamente as mesmas, e as operações são perfeitamente simétricas em relação ao mapa de deslocamento circular em n-bits palavras.

2.7 Algoritmo Speck

O algoritmo de Speck foi otimizado para desempenho em implementações em software. Representado usualmente pelo tamanho do bloco ($2n$) e o tamanho da chave (mn), onde n é o tamanho da palavra e m o número de palavras chaves. A notação para as diferentes variantes Speck é análoga a utilizada por Simon. Por exemplo, quando Speck96/144 refere-se à versão de Speck com blocos de texto claro de tamanho 96 bits e utilizando uma chave de 144 bits. (BEAULIEU, *et al.*, 2013)

Esse algoritmo utiliza em suas operações de criptografia (5) e geração de chaves as operações de XOR ($\oplus$) bit a bit, adição modular de $2^n(+)$ e descolamento circular de bits (S^n) nas suas rodadas. Na sua operação de decifração (6) ao invés de adição modular é utilizado a subtração modular. Cada rodada de Speck depende do mapa (4):

$$R_k : GF(2)^n \times GF(2)^n \rightarrow GF(2)^n \times GF(2)^n \quad (4)$$

Definido por:

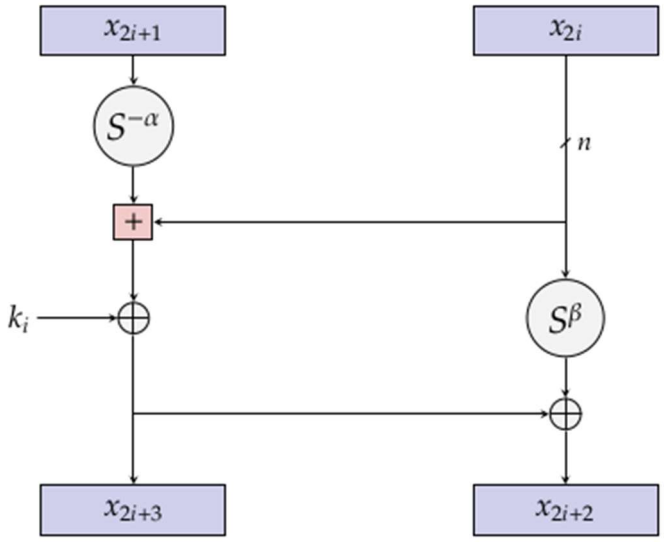
$$R_k(x,y) = ((S^{-\alpha}x + y) \oplus k, S^{\beta}y \oplus (S^{-\alpha}x + y) \oplus k) \quad (5)$$

A função inversa para a decifração expressada na equação 6:

$$R_k^{-1}(x,y) = (S^{\alpha}((x \oplus k) - S^{-\beta}(x \oplus y)), S^{-\beta}(x \oplus y)) \quad (6)$$

Para os blocos de tamanho 32, os valores de rotação são α=7 eβ=2, para os outros tamanhos de blocos α=8 e β= 3 (BEAULIEU, *et al.*, 2013).

A figura 7 demonstra cada função de rodada do algoritmo Speck.



Fonte: [Beaulieu et al. 2013]

Figura 7 Algoritmo Speck: Uma rodada de execução.

2.8 Considerações finais do capítulo

As definições de criptografia e as técnicas discutidas neste capítulo são importantes para o entendimento de algoritmos criptográficos, isto é importante para que as implementações dos algoritmos criptográficos sejam feitas corretamente e de forma eficiente. Este capítulo também

foi a base teórica a para posicionar as cifras de bloco leve entre os demais algoritmos de criptografia.

O próximo capítulo apresenta os trabalhos correlatos com ênfase no desempenho dos algoritmos Simon e Speck, que são abordados neste trabalho.

3 TRABALHOS CORRELATOS

Esta seção tem como objetivo abordar artigos encontrados no estado da arte, os quais tratam sobre implementações dos algoritmos de cifra leve Simon e Speck, ambos com ênfase na plataforma FPGA. Serão apresentados dois trabalhos, onde ambos apresentam resultados de suas implementações, onde são comparados o desempenho dos algoritmos por meio do impacto de área e vazão de cada um.

3.1 “Simon and Speck Block Ciphers for the Internet of Things”

No trabalho de (BEAULIEU, *et al.*, 2015) foram realizadas implementações dos algoritmos Simon e Speck em plataformas como circuitos integrados de aplicação especifica (ASICs), FPGAs e microcontroladores, com a finalidade de mensurar o desempenho destes em diversos cenários para aplicação deste tipo de segurança direcionado à Internet das coisas (IoT).

A implementação feita em ASICs do algoritmo Simon utiliza menor área quando comparadas com outras cifras de bloco leve contendo o mesmo tamanho do bloco e chave. A lógica necessária para calcular um bit por rodada é pequena, quando a escala da implementação é igual ou superior a dois bits, podem ser atualizadas em apenas um ciclo de relógio com impacto mínimo sobre a área.

Por conseguinte, a implementação do algoritmo Speck nesta mesma plataforma não traz tanta diferença relacionado ao desempenho de Simon. Uma das principais diferenças entre estes é que Speck substitui as portas lógicas AND por somador completo e adiciona multiplexadores para atualização dos estados.

Segundo (BEAULIEU et al. 2015) O artigo coloca em destaque os resultados obtidos nas implementações através de simulações com linguagem VHDL, relacionando-os a outros algoritmos encontrados no estado da arte. Com a velocidade do relógio à 100 kHz, foram comparados com outros algoritmos que correspondem com o mesmo tamanho de bloco, tamanho de chave, velocidade e testados em plataformas ASICs, os quais podem ser encontrados no estado da arte no cenário de cifras de bloco leve.

Nas implementações seguintes, realizadas nas plataformas FPGAs na linguagem VHDL e microcontroladores na linguagem Assembly e C, foram comparadas novamente com outros algoritmos criptográficos contidos no estado da arte, onde ao observar os resultados deixa evidente sua eficiência em relação aos outros. Apesar dos algoritmos serem implementados em diversas plataformas, tanto em hardware quanto em software, Simon e Speck não apresentaram

dificuldades em adaptação, diferentemente, como por exemplo, o algoritmo de cifra leve PRESENT.

Entre as implementações realizadas no trabalho de (BEAULIEU et al.2015), destacam-se as implementações utilizando Xilinx Spartan-6 FPGA, relacionadas na tabela 1, com o intuito de comparar com os resultados destacados no final deste artigo.

Tabela 1 – Comparações de performance entre diferentes implementações em FPGA

Tamanho	Algoritmo	Área (Slice LUTs)	Vazão (Mbits/s)
64/128	**Simon**	**24**	**9.6**
	Simon	**138**	**512**
	Speck	**34**	**7.0**
	Speck	**153**	**416**
128/128	**Simon**	**28**	**5.7**
	Simon	**197**	**567**
	Simon	**375**	**867**
	Speck	**36**	**5.0**
	Speck	**232**	**455**
	Speck	**401**	**920**

Logo os autores, concluem que os algoritmos Simon e Speck possuem vantagens não apenas comparado ao desempenho em plataformas diferentes, mas também, pela sua flexibilidade. Possuindo versatilidade, além de sua simplicidade, faz com que esses dois algoritmos sejam ideais para uso em redes heterogêneas, onde permitem ser otimizados para uma aplicação especifica.

3.2 "SpecTre: A Tiny Side-Channel Resistant Speck Core for FPGAs"

No trabalho de (CHEN, *et al.*, 2015) foram realizadas duas implementações do algoritmo de criptografia Speck 128/128 na plataforma Xilinx Spartan-3 FPGA series, utilizando a linguagem de descrição de hardware Verilog, para aplicação deste tipo de criptografia no cenário de Internet das coisas.

As duas implementações baseiam-se em estratégias onde os registradores são implementados na forma de serialização de bits na transmissão de dados para outros

componentes que compõe a arquitetura, sendo apenas a chave disposta paralelamente durante a execução das rodadas.

Porém, a diferença entre os dois algoritmos é que a segunda implementação, afim de aumentar sua confiabilidade, acrescentou a criptografia limiar. Esse criptosistema aplica os conceitos de partilha XOR-secreto baseado em computação multipartidária comprovadamente seguro contra-ataques de canal lateral. Foram feitos análise de consumo de energia e a resistência à ataques de canal lateral.

O termo criptografia de limiar denomina um conjunto de algoritmos tipicamente de tolerância a faltas/intrusões mas que surgiram no âmbito da segurança.

Os resultados das implementações na plataforma FPGA são relacionados na tabela 2, associados a resultados de desempenho de outros algoritmos encontrados no estado da arte, onde a implementação dos autores encontra-se em destaque.

Tabela 2 – Comparações de área e vazão entre algoritmos encontrados no estado da arte em FPGA

Cifra	Área (Slice LUTs)	Vazão (Mbits/s)	Plataforma
Criptografia limiar;			
TI-Speck128/128	**99**	**9.68**	**Xc3s50**
TI-Simon128/128	87	3.0	Xc3s50
Cifras de bloco desprotegido;			
Speck128/128	**43**	**10.05**	**Xc3s50**
Simon128/128	36	3.6	Xc3s50

3.3 Considerações Finais

Os trabalhos abordados neste capítulo utilizam diferentes métodos de implementação dos algoritmos Simon e Speck em diferentes tipos de FPGAs, entretanto os resultados serão utilizados para comparação com os resultados das implementações obtidos neste trabalho.

No capítulo seguinte são expostas as arquiteturas propostas para as implementações dos algoritmos criptográficos Simon e Speck, as arquiteturas utilizam diferentes formas de descrever o comportamento e estrutura do hardware, os resultados serão comparados com os resultados das arquiteturas dos trabalhos correlatos.

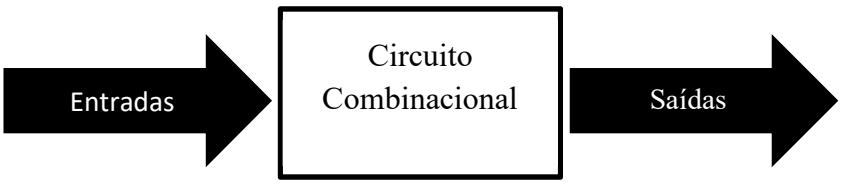
4 PROPOSTA DE ARQUITETURA PARA IMPLEMENTAÇÃO DOS ALGORITMOS SIMON E SPECK

Neste capítulo é discutido sobre as arquiteturas propostas, suas especificações utilizadas e o que é esperado nas criptografias leves Simon e Speck. As arquiteturas foram projetadas com *Field Programmable Gate Array* (FPGA), particularmente na família Xilinx Artix7 - XC7A100T, Xilinx ISE 14.2, utilizando a linguagem de descrição VHDL. As arquiteturas implementadas são categorizadas em: *Sync Single Rounds, Sync Full Rounds* e *Async Full Rounds*.

Inicialmente apresentam-se alguns conceitos sobre circuito combinacional, circuito sequencial e a linguagem de descrição de hardware VHDL, antes de explicar cada arquitetura.

4.1 Circuito Combinacional

Um circuito combinacional é composto por um conjunto de portas lógicas que determinam os valores das saídas a partir dos valores das entradas atuais do circuito.



Fonte: Elaborada pelo autor

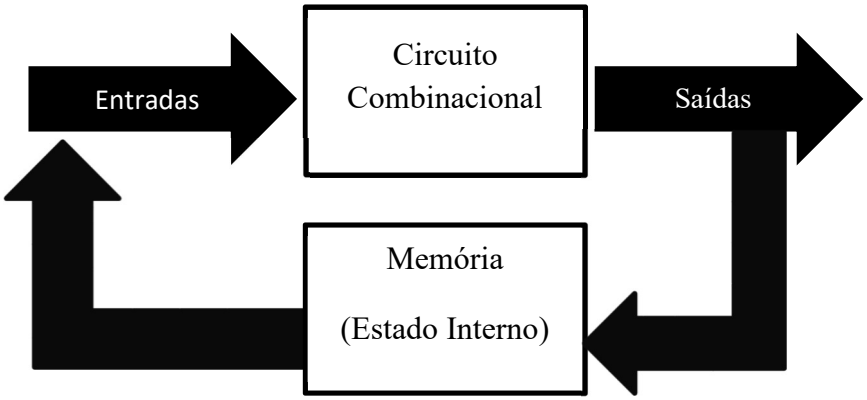
Figura 8 Circuito Combinacional

A figura 8 representa um modelo genérico de um circuito combinacional que segue a apenas a lógica combinacional e utiliza álgebra de Boole como ferramenta. As saídas dependem exclusivamente da combinação de valores presentes nas entradas (VAHID, 2010).

4.2 Circuito Sequencial

Um circuito sequencial pode ser representado por um circuito combinacional e elementos de memória. As partes do circuito combinacional tem entradas externas e internas e saídas externas e internas, as entradas externas são as entradas atuais e as entradas internas são provenientes da memória que fornecem o estado do circuito, as saídas externas são as saídas do

circuito, as saídas internas fornecem o cálculo do próximo estado; a memória tem como entrada o valor do próximo estado que será armazenado e realimenta o circuito combinacional. A figura 9 representa um modelo genérico de um circuito sequencial (GÜNTZEL e NASCIMENTO, 2001).



Fonte: Elaborada pelo autor

Figura 9 Circuito Sequencial

Os circuitos sequenciais podem ser divididos em dois tipos, conforme o comportamento temporal dos seus sinais: Assíncronos e Síncronos.

Assíncronos: O estado do circuito muda no momento em que suas entradas sofrem alterações, essas alterações ocorrem realmente ao final do circuito, este tempo é o somatório dos tempos de propagação das portas lógicas (SKAHILL, 1996).

Síncronos: Nos circuitos síncronos os valores das entradas são sincronizadas com um sinal especial denominado de relógio (*clock*, em inglês), as mudanças de estados é resultado da observação das entradas e do instante de tempo das transições do sinal de *clock* (SKAHILL, 1996).

4.3 VHDL (Very Hardware Description Language)

A linguagem VHDL é uma linguagem de descrição de hardware desenvolvida pelo departamento de defesa dos Estados Unidos no início da década de 1980, como uma forma concisa de documentar os projetos no programa de circuito integrado de velocidade muito alta

(VHSIC) usando uma linguagem de descrição de hardware (*Hardware Description Language - HDL*), posteriormente o nome foi abreviado para VHDL. Foram desenvolvidos programas que recebiam arquivos em VHDL e simulavam a operação dos circuitos. A linguagem foi padronizada pela IEEE, o que tornou universalmente atraente para engenheiros e para criadores de ferramentas. (TOCCI, WIDMER e MOSS, 2008)

Em VHDL é possível descrever o hardware de duas formas: a *comportamental* e a *estrutural*. A forma *estrutural* indica todos os componentes e interconexões que constituem o circuito. A forma *comportamental* descreve o hardware visualizando no seu comportamento e descrevendo o mesmo com instruções de alto nível de abstração (como por exemplo: *if* e *while*) (ORDONEZ, PEREIRA, *et al.*, 2003).

A linguagem VHDL contém três elementos que possuem valor específico: constantes (*constant*), variáveis (*variable*) e sinais (*signal*). Constante e variáveis tem conceitos similares ao de outras linguagens, o que não ocorre com os sinais (MORENO, PEREIRA e CHIARAMONTE, 2005).

- **Constate:** Elemento que é inicializado com um valor *x* e não pode ser alterado posteriormente.

  Ex.: CONSTANT dez : INTEGER := 10;
  CONSTANT dez : STD_LOVIC_VECTOR := “1010”;

- **Variável:** Segue os mesmos conceitos de outras linguagens. Devem ser colocadas dentro de processos (**PROCESS**) ou em subprogramas, tendo seu escopo local (dentro do processo ou subprograma). Quando um valor é atribuído a ela, a atualização é imediata, isto ocorre porque as variáveis são elementos abstratos da linguagem e possuem uma concordância física real imediata (MORENO, PEREIRA e CHIARAMONTE, 2005). A figura 10 exemplifica o uso de variável em um circuito síncrono.

```
1     process (clock)
2           variable cont: integer := 0;
3     begin
4       if rising_edge (clock) then
5           for i in 0 to 9
6           loop
7                cont := cont +1;
8           end loop;
9       end if;
10    end process;
```

Figura 10 Exemplo de Circuito Síncrono com uso de Variáveis

Na linha 1 é feita a declaração de um processo síncrono, referenciando a sincronização com o sinal *clock* (relógio do circuito) e na linha 10 finaliza o processo;

Na linha 2 é declarado a variável do tipo inteiro, inicializada com valor 0;

Entre as linhas 4 à 9 declara-se uma estrutura condicional, executada somente quando a borda de subida do *clock*;

Nas linhas 6 até 8 é feito um laço repetitivo que executa 10 vezes, no final da execução do laço a variável *cont* estará com o valor 10, isto é possível por que as variáveis são atualizadas no mesmo instante.

Observando que a declaração da variável é feita dentro do processo e seu escopo é apenas dentro do processo.

- Sinal: Diferente das variáveis os sinais não são atualizados instantaneamente, isto ocorre porque os sinais possuem um significado físico, representam conexões reais de um circuito. A declaração é feita da mesma maneira que as constantes e variáveis com a diferença que podem ser *normal*, *register* ou *bus* (MORENO, PEREIRA e CHIARAMONTE, 2005). A figura 11 exemplifica o uso de sinais em um circuito síncrono.

```
1       architecture Behavioral of exemplo is
2       signal cont : integer <= 0;
3       begin
4       process (clock)
5          begin
6             if rising_edge (clock) then
7                cont <= cont +1;
8             end if;
9          end process;
10      end behavioral;
```

Figura 11 Exemplo de Circuito Síncrono com uso de Sinais

Entre as linhas 1 e 10 é declarado todo o comportamento da arquitetura;

Na linha 2 declara-se um sinal *cont* do tipo inteiro, inicializado com valor 0;

Na linha 3 inicia a declaração do comportamento da arquitetura;

Na linha 4 é feita a declaração de um processo síncrono, referenciando a sincronização com o sinal *clock* (relógio do circuito) e na linha 9 finaliza o processo;

Entre as linhas 6 à 8 declara-se uma estrutura condicional, executada somente quando a borda de subida do *clock*,

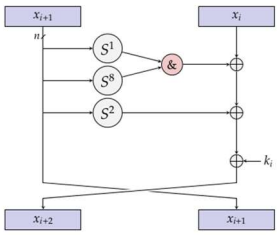
Observando que a atribuição do valor no sinal *cont* na linha 7 somente será atualizado no final do processo.

Desse modo conforme o que foi apresentado que vai desde circuitos combinacionais até linguagem VHDL nas seções seguintes é discutido as diferentes arquiteturas implementadas, utilizando as diversas formas de descrever o hardware proposto para esse trabalho.

4.4 Sync Single Rounds (SSR)

Arquitetura síncrona, dependente de um *clock* base, descrita utilizando sinais (*signal*) onde os valores dos cálculos são atualizados ao término de cada iteração (*clock*), não possibilitando o uso de laços repetitivos, resultando no aumento significativo de iterações, porém tem-se um ganho em relação à área utilizada dentro do circuito e a frequência de operação. A figura 12 faz uma analogia ao modelo da arquitetura SSR.

```
1   architecture Behavioral of SSR is
2       signal simon32:  STD_LOGIC;
3   begin
4       process (clock)
5       begin
6         if rising_edge (clock) then
7         end if;
8       end process;
9   end Behavioral;
```



Uma interação do algoritmo Speck ou Simon

Fonte: Elaborada pelo autor

Figura 12 Representação da Implementação SSR.

Entre as linhas 1 e 9 é feita toda a declaração dos componentes estruturais e como a arquitetura se comporta;

Na linha 2 é declarado o "*signal*" (sinal) *simon* do tipo "*STD_LOGIC*" (lógico);

Obs. Os valores dos sinais não são atualizados instantaneamente, mas somente ao termino do processo;

Na linha 3 inicia a descrição do comportamento da arquitetura;

Na linha 4 é declarado o processo ("*clock*");

Na linha 5 inicia o processo;

Entre as linhas 6 e 7 se houver borda de subida do "*clock*" (ativação), então entra no bloco de instruções e executas as instruções ali descritas;

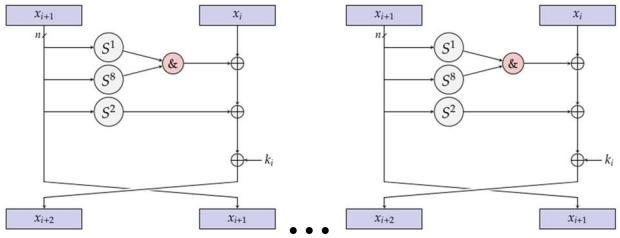
Na linha 8 finaliza o processo;

Na linha 9 finaliza a descrição do comportamento da arquitetura.

4.5 Sync Full Rounds (SFR)

Arquitetura síncrona, dependente de um *clock* base, descrita utilizando variáveis (*variable*) onde os valores dos cálculos necessários para executar o algoritmo são atualizados instantaneamente, isso possibilita a utilização de laços repetitivos (*for*), fazendo com que em apenas uma única iteração do circuito (*clock*) sejam realizadas a geração das chaves e as operações de cifrar ou decifrar. No entanto, esse tipo de metodologia pode consumir uma maior área do circuito digital. A figura 13 faz uma analogia ao modelo da arquitetura SFR.

```
1    architecture Behavioral of SSR is
2    begin
3       process (clock)
4          variable simon32:  STD_LOGIC;
5       begin
6         if rising_edge (clock) then
7            for i in 0 to n
8            Loop
```




```
9            end loop;
10         end if;
11       end process;
12   end Behavioral;
```

Fonte: Elaborada pelo autor

Figura 13 Representação da Implementação SFR.

Entre as linhas 1 e 12 é feita toda a declaração dos componentes estruturais e como a arquitetura se comporta;

Na linha 2 inicia a descrição do comportamento da arquitetura;

Na linha 3 é declarado o processo (“*clock*”);

Na linha 4 declarados a “*variable*” (variável) do tipo “*STD_LOGIC*” (lógica);

Obs. Os valores das variáveis são atualizados instantaneamente, permitindo a utilização de laço repetitivos;

Na linha 5 inicia o processo;

Entre as linhas 6 e 10 se houver borda de subida do "*clock*" (ativação), então entra no bloco de instruções e executas as instruções ali descritas;

Na linha 7 é declarado o laço repetitivo "for" indo de i = 0 até i = n (quantidade de iterações);

Na linha 8 inicia o "loop", e realiza as instruções contidas dentro do bloco;

Na linha 9 finaliza o bloco de instruções;

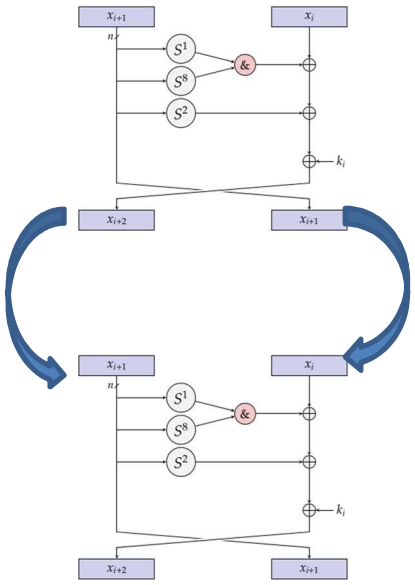
Na linha 11 finaliza o processo;

Na linha 12 finaliza a descrição do comportamento da arquitetura.

4.6 Async Full Rounds (AFR)

Arquitetura assíncrona, composta de circuitos combinacionais onde ocorre a associação de portas lógicas e suas operações podem ser especificadas por meio de um conjunto de equações Booleanas, esta é utilizada para realizar as operações do *round function* e da geração das chaves. Esta arquitetura também contém uma memória onde são armazenadas as chaves geradas e os resultados das operações realizadas por cada round realimentando os rounds seguintes. O tempo de iteração de cara round está associado diretamente ao tempo de atraso de propagação dos valores entre as portas lógicas, das entradas até a saída. Este tipo de arquitetura pode atingir taxas altas de desempenho, porém pode consumir uma maior área do circuito digital implementado. A figura 14 faz uma analogia ao modelo da arquitetura AFR.

```
1    architecture Behavioral of SSR is
2        signal simon32_iteração_1:  STD_LOGIC;
3        signal simon32_iteração_n:  STD_LOGIC;
4    begin
```



Ausência de *clock*, lógica combinacional

```
5    end Behavioral;
```

Fonte: Elaborada pelo autor

Figura 14 Representação da Implementação AFR.

Entre as linhas 1 e 5 é feita toda a declaração dos componentes estruturais e como a arquitetura se comporta;

Nas linhas 2 e 3 são declarados os sinais do tipo "*STD_LOGIC*" (lógicos);

Na linha 4 inicia a descrição do comportamento da arquitetura;

Na linha 5 finaliza a descrição do comportamento da arquitetura.

4.7 Considerações Finais

As arquiteturas expostas neste capítulo utilizam diferentes especificações, portanto as implementações geradas por elas utilizam diferentes métodos e podem gerar diferentes resultados quanto a desempenho, área consumida e consumo de energia. Os resultados obtidos com a implementação dos algoritmos Simon e Speck utilizando as arquiteturas descritas são expostas no capítulo seguinte.

5 RESULTADOS E COMPARAÇÕES COM TRABALHOS CORRELATOS

Implementando os algoritmos criptográficos Simon e Speck nas arquiteturas propostas na seção acima e utilizando à ferramenta ISE Design Suite 14.2 da Xilinx, é fornecido o período (ns) mínimo de execução do circuito, a frequência máxima de execução (Mhz) e a quantidade Slice Luts (área). Com posse desses dados foi calculado a vazão e relação área/vazão.

Para a avaliação da vazão de cada arquitetura em ambos os algoritmos, é realizado o cálculo a partir da divisão do tempo base (1 segundo), pelo período de execução que é o tempo de propagação da lógica do circuito implementado, multiplicado pela quantidade iterações necessárias para cada implementação e por fim multiplicado pelo tamanho da palavra (32 bits). Esse cálculo é representado na equação 7.

$$Vazão = \frac{1}{Período * Qtde\ Iterações} * 32 \quad (7) \qquad VA = \frac{Área}{vazão} \quad (8)$$

Obtido a vazão, uma relação entre vazão e área (equação 8) é realizada para comparar o desempenho entre os algoritmos. A variável área é situada pela a quantidade de SliceLUTs consumida por cada implementação.

Para a coleta dos dados referente ao consumo de energia foi utilizado a ferramenta da Xilinx XPower Analyzer, que fornece uma estimativa de consumo de energia pós-implementação. Segundo a Xilinx a XPA é a ferramenta mais precisa, uma vez que pode ler a partir do banco de dados do projeto implementado os exatos recursos lógicos e de roteamento utilizados. A XPA apresenta um relatório de consumo de energia, podendo navegar em diferentes visões do projeto e também permite ajustar as configurações de ambiente (temperatura ambiente) para que se possa avaliar como reduzir o consumo de energia. (XILINX, 2011)

Nas seções 5.1 e 5.2 são apresentados e especificados os algoritmos Simon e Speck implementados utilizando as arquiteturas propostas, assim como a análise de desempenho de área, vazão e consumo de energia.

5.1 Simon 32

Nesta seção são apresentados os resultados de desempenho, área, vazão e consumo de energia das implementações do algoritmo Simon32/64, com T = 32 (quantidade de rodadas),

projetadas no FPGA da Xilinx Artix7 – XC7A100T utilizando a linguagem VHDL, utilizando as arquiteturas SFR, SSR e AFR.

Na tabela 3 tem-se informações sobre desempenho, área e a quantidade de iteração das arquiteturas descritas em VHDL e implementadas em FPGA, destacando-se frequência máxima do circuito e seu consumo de área medido em SliceLUTs.

Tabela 3 – Estatística: Área e Desempenho de Simon

SIMON	**Desempenho**		**Área**	**N° de Iterações**
	Período (ns)	Frequência (MHz)	Slice LUTs	Cifração
SFR	41,499	24,096	1504	1
SSR	2,719	367,796	973	65
AFR	26,087	38,333	2465	1

Em relação a área, destaca-se a arquitetura SSR, que obteve o menor consumo, com 973 Slices LUTs. Quando comparado com as arquiteturas SFR e AFR tem-se um ganho de 54,5% e 153,3%, respectivamente.

Com relação a frequência, novamente a arquitetura SSR obteve o melhor resultado, sendo 1526% e 959% superior as arquiteturas SFR e AFR. No entanto deve-se enfatizar que a arquitetura SSR necessita de 65 interações (ciclos de *clock*) para a execução do algoritmo Simon, enquanto as demais apenas 1 iteração.

Para gerar uma comparação entre as arquiteturas é proposta a análise da relação área/vazão, apresentada na tabela 4.

Tabela 4 – Relação Área / Vazão Simon

SIMON	**Vazão**	**Área/Vazão**
	Mbits/s	Slice LUTs/(Mbits/s)
SFR	771	1,95
SSR	181	5,37
AFR	1.226	2,01

Vazão denota a arquitetura com melhor desempenho em relação ao volume de informações cifradas por segundo. A relação área/vazão indica o melhor custo benéfico entre área e vazão. Com relação a vazão, a arquitetura que obteve que destacou se foi a AFR com

1.226 Mbits/s, sendo superior 59% e 700% quando comparada com as arquiteturas SFR e SSR respectivamente.

Em relação a área/vazão, a arquitetura com melhor resultado é a SFR, sendo 3% e 185% mais vantajosa que as arquiteturas AFR e SSR respectivamente.

A tabela 5 expõe o consumo de energia das arquiteturas em um ambiente com temperatura de 25º Celsius, podendo observar que para as arquiteturas SFR e AFR o consumo bem inferior quando comparados com a arquitetura SSR, isso ocorre devido a quantidade de iterações necessárias para completar a encriptação de um bloco na arquitetura SSR. Ainda pode-se notar que a arquitetura que consume menos energia é a arquitetura AFR.

Tabela 5 – Consumo de Energia de Simon

SIMON	**(W)**
SFR	0,042
SSR	2,73
AFR	0,041

Foram coletados dados referentes ao consumo de energia das arquiteturas com temperaturas entre 0º à 85º Celsius. Onde pode-se observar que com o aumento de temperatura o consumo de energia é maior. Esses dados estão expostos em formato de gráfico nas figuras 15 e 16.

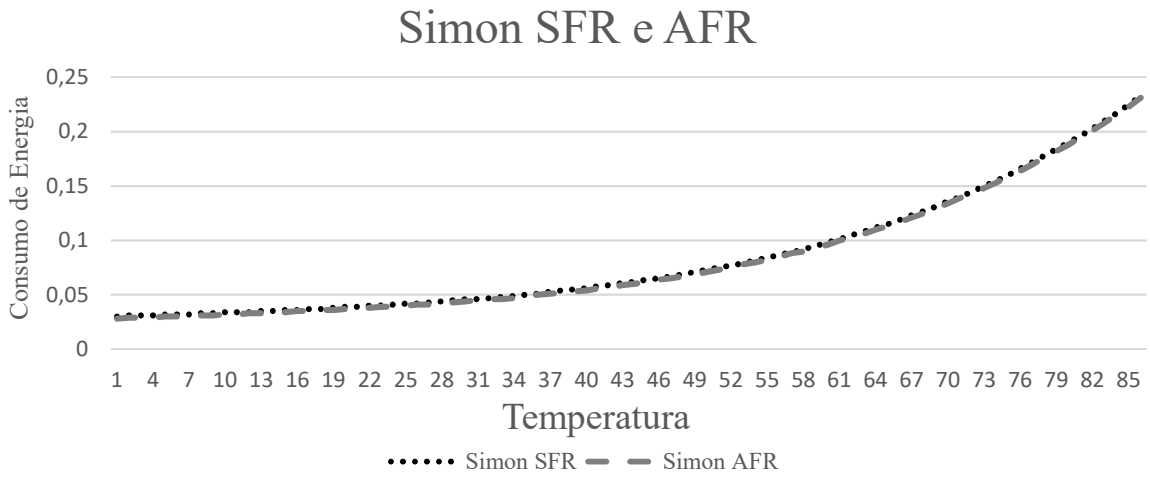


Figura 15 Consumo de Energia Simon SFR e AFR

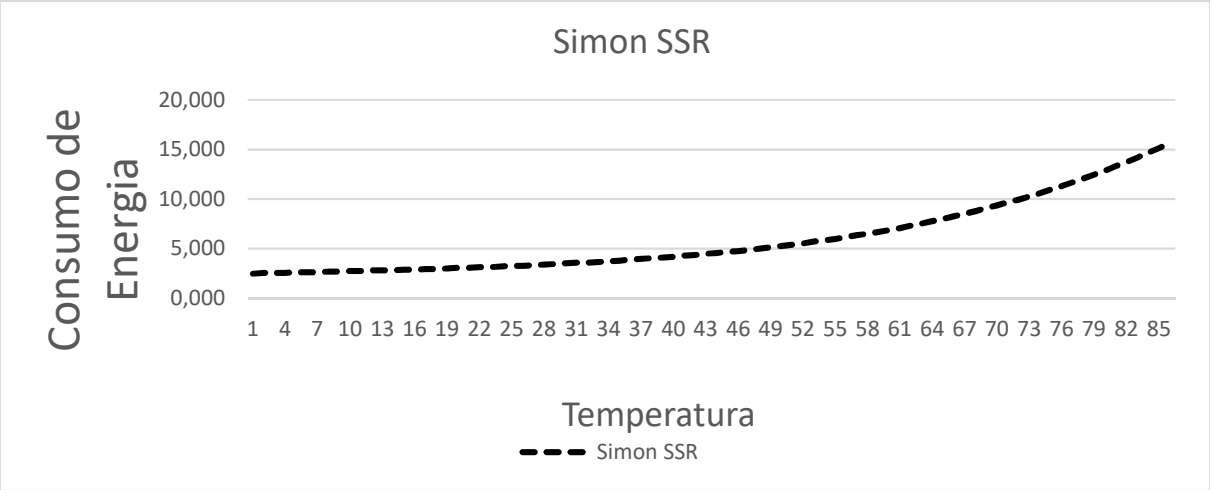


Figura 16 Consumo de Energia Simon SSR

5.2 Speck 32

Nesta seção são apresentados os resultados de desempenho, área, vazão e consumo de energia das implementações do algoritmo Speck32/64, com T = 22 (quantidade de rodadas), utilizando o mesmo cenário em que ocorreu a implementação do algoritmo Simon, foram coletados os dados referentes ao número de iterações, a quantidade de SliceLUTs, o período mínimo e a frequência máxima, para gerar uma comparação entre as implementações, é proposta a análise da relação área/vazão. Estas informações foram relacionadas na tabela 6.

Tabela 6 – Análise Área e Desempenho de Speck

SPECK	**Desempenho**		**Área**	**Nº de Iterações**
	Período (ns)	Frequência (MHz)	SliceLUTs	Cifração
SFR	75,746	13,202	2238	1
SSR	4,012	249,258	752	45
AFR	36,522	27,38	485	1

Em relação a área, destaca-se a arquitetura AFR, que obteve o menor consumo, com 485 Slices LUTs. Quando comparado com as arquiteturas SFR e SSR tem-se um ganho de 55% e 361%, respectivamente.

Com relação a frequência, a arquitetura SSR obteve o melhor resultado, sendo 1.888% e 910% superior as arquiteturas SFR e AFR. No entanto deve-se enfatizar que a arquitetura SSR

necessita de 45 interações (ciclos de clock) para a execução do algoritmo Speck, enquanto as demais apenas 1 iteração.

A tabela 7 demostra a relação entre vazão e área, para comparar o desempenho dos algoritmos. Para a área é utilizado a quantidade de SliceLUTs.

Tabela 7 – Relação Área/ Vazão Speck

SPECK	**Vazão**	**Área/Vazão**
	Mbits/s	Slice LUTs/(Mbits/s)
SFR	422	5,3
SSR	177	4,24
AFR	876	0,55

Com relação a vazão, a arquitetura que obteve que destacou se foi a AFR com 876 Mbits/s, sendo superior 200% e 494% quando comparada com as arquiteturas SFR e SSR respectivamente. Em relação a área/vazão, a arquitetura com melhor resultado é novamente a AFR, sendo 963% e 770% mais vantajosa que as arquiteturas SFR e SSR respectivamente.

A tabela 8 expõe o consumo de energia das arquiteturas em um ambiente com temperatura de 25º Celsius, podendo observar que para as arquiteturas SFR e AFR o consumo é bem inferior quando comparados com a arquitetura SSR, isso ocorre devido a quantidade de iterações necessárias para completar a encriptação de um bloco na arquitetura SSR. Ainda pode-se notar que a arquitetura que consume menos energia é a arquitetura AFR.

Tabela 8 – Consumo de Energia de Speck

SPECK	**(W)**
SFR	0,110
SSR	3,575
AFR	0,041

Foram coletados dados referentes ao consumo de energia das arquiteturas com temperaturas indo de 0º à 85º Celsius. Onde pode-se observar que com o aumento de temperatura o consumo de energia é maior. Esses dados estão expostos em formato de gráfico nas figuras 17 e 18.

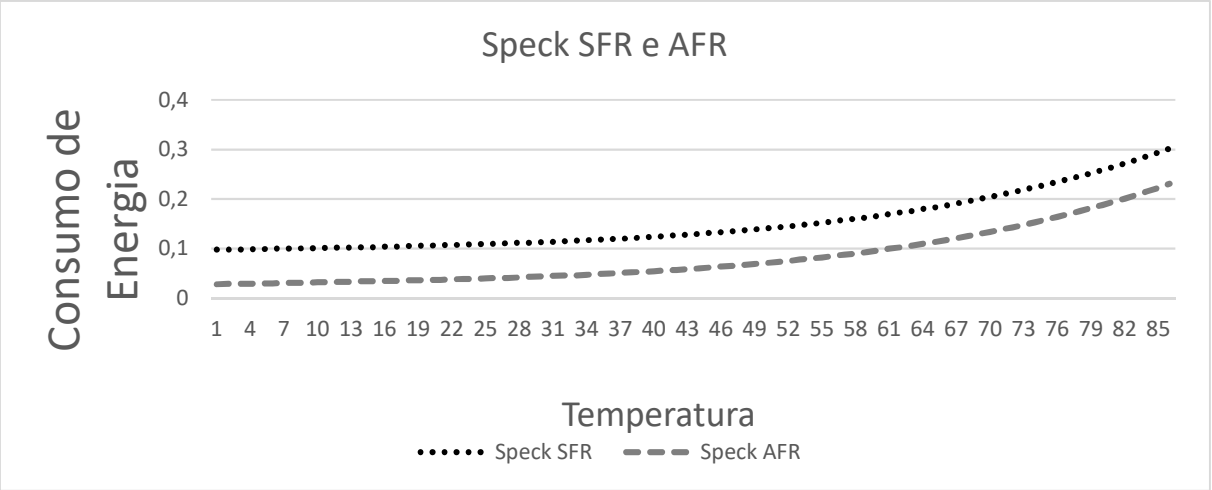


Figura 17 Consumo de Energia Speck SFR e AFR

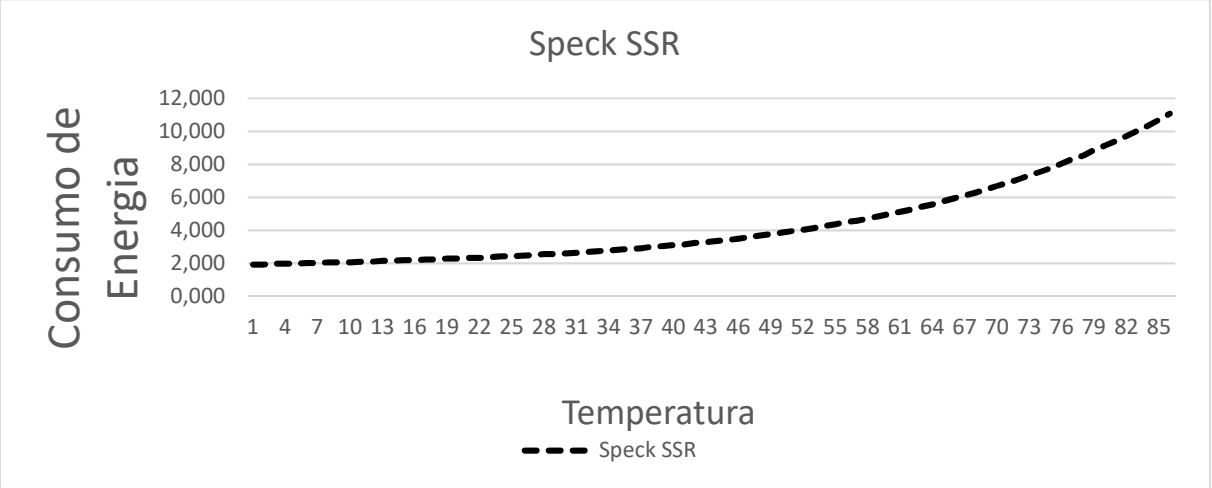


Figura 18 Consumo de Energia Speck SSR

5.3 Comparação com trabalhos correlatos.

Na comparação entres as implementações realizadas neste trabalho o algoritmo Simon atingiu melhor desempenho em relação ao Speck, destacando a arquitetura AFR, que obteve a melhor vazão, porém Speck obteve uma melhor relação área/vazão. A tabela 9 demostra os resultados da arquitetura AFR que será adotada para comparação a seguir com os trabalhos correlatos

Tabela 9 – Comparação da vazão do Simon e Speck

Arquitetura	Algoritmo	Vazão (Mbits/s)	Área/Vazão
AFR	Simon	1.226	2,01
AFR	Speck	876	0,55

Apesar de os artigos correlatos adotarem famílias de FPGAs diferentes e também tamanhos de blocos e chaves é possível fazer a comparação resguardando esta observação, e adotando apenas a vazão final alcançada por essas implementações. A tabela 10 apresenta os melhores resultados atingidos pelos trabalhos correlatos e também por este trabalho (AFR).

Tabela 10 – Comparação da vazão do Simon e Speck com trabalhos correlatos

Autor	**Algoritmo**	**Vazão (Mbits/s)**
BEAULIEU et al (2015)	Simon	867
BEAULIEU et al (2015)	Speck	920
CHEN (2015)	Speck	10,05
AFR	Simon	1.226
AFR	Speck	876

BEAULIEU, et al (2013), são autores dos algoritmos Simon e Speck, como pode-se observar a implementação de AFR/Simon (nossa) foi a melhor em relação a vazão entres os trabalhos correlatos apontados. Já para o Speck a implementação de referência BEAULIEU, et al (2013) atingiu o melhor resultado.

No entanto, dependendo do cenário onde será realizada a aplicação da solução, pode-se escolher entre a melhor implementação conforme os requisitos relacionadas ao desempenho ou área. Por ter uma vazão relativamente alta é possível cifrar dados e informações em sistemas computacionais, assim mesmo que alguma informação seja classificada como dados, ela estará protegida por criptografia.

6 CONCLUSÃO

Nos últimos anos, a quantidade de dispositivos embarcados tem sido crescente nas mais variadas aplicações tais como: *Radio Frequency Identification* (RFID) *Wireless Sensor Networks* (WSNs). Fornecimento de soluções de segurança para esses dispositivos amplamente utilizados tem atraído muita atenção por parte dos pesquisadores de criptografia. Esses tipos de dispositivos têm consumo energia muito limitado, limitações de memória e capacidade de computação, e aplicando, assim, soluções de segurança tradicionais, nesses contextos é muitas vezes impraticável. Assim, a criptografia de cifras de bloco leve foi desenvolvida para proporcionar algoritmos compactos e protocolos que se encaixam em ambientes com recursos limitados (YANG, ZHU, *et al.*, 2015).

As cifras de blocos leves recentemente propostas, Simon e Speck, levaram a documentos relativos à sua segurança. Isto é parcialmente devido ao fato de que estas cifras são reconhecidas como as cifras de blocos menores em cada um dos blocos e categorias de chave quando utilizado em ambientes de recursos limitados (YANG, ZHU, *et al.*, 2015).

Este trabalho teve como objetivo o estudo dos algoritmos criptográficos leves em especial os algoritmos Simon e Speck, propor arquiteturas utilizando diferentes metodologias para a implementação dos algoritmos em FPGA, realizar a análise dos resultados das implementações quanto ao desempenho, área e consumo de energia e comparar com os trabalhos encontrados no estado da arte. Os objetivos deste trabalho foram alcançados, sendo os resultados discutidos e comparados com trabalhos encontrados no estado da arte.

Para trabalhos futuros melhorias na implementação AFR, utilizando técnicas de *pipelines* de execução ou paralelização das arquiteturas com o intuito de aumentar ainda mais a vazão da arquitetura proposta. As técnicas de pipeline visam basicamente em dividir uma atividade em várias partes, desta forma, processa-se vários conjuntos de dados simultaneamente. A paralelização das arquiteturas com intuito de aproveitar o que cada arquitetura tem de melhor, fazendo uma implementação hibrida podendo provocar um ganho razoável no desempenho.

REFÊRENCIAS

BEAULIEU, R. et al. The Simon and Speck Families of Lightweight Block Ciphers. **Cryptology ePrint Archive, Report 2013/404 (2013)**, Fort Mead, p. 45, 19 jun. 2013. Disponivel em: <https://eprint.iacr.org>. Acesso em: 01 ago. 2015.

BEAULIEU, R. et al. Simon and Speck Block Ciphers for the Internet of Things*. **NIST Lightweight Cryptography Workshop**, Fort Meade, jun. 2015. Disponivel em: <https://eprint.iacr.org/2015/585.pdf>. Acesso em: 15 jan. 2016.

BEECHER, P. Interoperable Wireless Communications Networks for IoT. **Fórum Brasileiro de Internet das Coisas**, São Paulo, 01 set. 2016. Disponivel em: <http://iotbrasil.org.br/confira-a-programacao-i-congresso-brasileiro-e-latino-americano-de-iot/>.

CHEN, C. et al. SpecTre: A Tiny Side-Channel Resistant Speck Core for FPGAs. **IACR Cryptology ePrint Archive**, v. 2015, p. 691.

DEVMEDIA. Criptografia: Conceito e aplicações - Revista Easy Net Magazine 27. **DevMedia**. Disponivel em: <http://www.devmedia.com.br/criptografia-conceito-e-aplicacoes-revista-easy-net-magazine-27/26761>. Acesso em: 03 nov. 2016.

FEISTEL, H. **Cryptography and Computer Privacy**. [S.l.]: Scientific American, v. 228, 1973.

FERREIRA, F. N. F. **Segurança da Informação**. Rio de Janeiro: Ciência Moderna Ltda, 2003. 162 p. ISBN: 85-7393-290-2.

GÜNTZEL, J. ; NASCIMENTO, F. A. D. Introdução aos Sistemas Digitais. **Departamento de Informática e Estatítica - UFSC**, 2001. Disponivel em: <https://www.inf.ufsc.br/~j.guntzel/isd/isd.html>. Acesso em: 01 nov. 2016.

GUO, J. et al. The LED Block Cipher*. **Cryptographic Hardware and Embedded Systems–CHES 2011**, Springer Berlin Heidelberg, p. 326-341, 2011. Disponivel em: <http://eprint.iacr.org/2012/600.pdf>. Acesso em: 12 ago. 2015.

ISO/IEC-27001. **Information Security Management System (ISMS)**. [S.l.]. 2005.

MENEZES, A. J.; OORSCHOT, P. C. V.; VANSTONE, S. A. **Handbook of Applied Cryptography**. 5. ed. [S.l.]: CRC Press, v. 1, 1996. 816 p. Disponivel em: <http://cacr.uwaterloo.ca/hac/>. ISBN: 0-8493-8523-7.

MORENO, E. D. Platforms and Applications in Hardware Security Trends and Challenges. **IJSIA - International Joournal of Security and Its Applications**, vol 7, out. 2013. 101-115. ISSN: 1738-9976 IJSIA.

MORENO, E. D.; PEREIRA, F. D.; CHIARAMONTE, R. B. **Criptografia em Software e Hardwarwe**. São Paulo: Novatec, 2005.

ORDONEZ, E. D. M. et al. **Projeto, Desempenho e Aplicações de Sistemas Digitais em Circuitos Programáveis (FPGAs)**. Pompéia: Bless, 2003. ISBN: 85-87244-13-2.

SANTOS, L. C. D. **Implementação do esquema totalmente homomórfico sobre inteiros com chave reduzida**. Marília: [s.n.], 2014. 21 p. Disponivel em: <http://aberto.univem.edu.br/bitstream/handle/11077/1021/Luan%20Cardoso%20dos%20Santos.pdf?sequence=1>. Acesso em: 01 ago. 2016.

SKAHILL, K. **VHDL for PROGRAMMABLE LOGIC**. 1. ed. Menlo Park: Addison-Wesley Publishing Company, Inc, v. 1, 1996. 594 p.

STALLINGS, W. **Criptografia e Segurança de redes**. Tradução de Daniel Vieira. 4. ed. São Paulo: Person, 2012. 492 p.

SUZAKI, T. et al. Twine: A lightweight, versatile block cipher. **ECRYPT Workshop on Lightweight Cryptography**, p. 146-149, 2011. Disponivel em: <www.nec.co.jp/rd/media/code/research/images/twine LC11.pdf>. Acesso em: 02 ago. 2015.

TANENBAUM, A. S. **Redes de computadores**. Tradução de Vandenberg D. de Souza. 4º. ed. Amsterdam: CampuS, 2003.

TERADA, R. **Segurança de Dados - Criptografia em Redes de Computadores**. 2º. ed. [S.l.]: Edgar Bluncher, 2011.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **SISTEMAS DIGITAIS princípios e aplicações**. Tradução de Cláudia Martins. 10. ed. São Paulo: Pearson, 2008. 803 p.

VAHID, F. **Sistemas DIgitais Projeto, Otimização e Hdls**. 1. ed. Porto Alegre: BOOKMAN, v. 1, 2010. 600 p.

WHEELER, D. J.; NEEDHAM, R. M. TEA, a Tiny Encryption Algorithm. **Springer**, Preneel, v. 1008, p. 363-366, 1995.

WU, W.; ZHANG,. LBlock: A Lightweight Block Cipher*. **Applied Cryptography and Network Security**, Springer Berlin Heidelberg, jan. 2011. Disponivel em: <http://eprint.iacr.org/2011/345.pdf>. Acesso em: 03 ago. 2015.

XILINX. Power Methodology Guide. **UG786**, 13.1, 1 mar. 2011. Disponivel em: <http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/ug786_PowerMethodology.pdf>. Acesso em: 10 ago. 2016.

YANG, G. et al. The Simeck Family of Lightweight Block Ciphers. **IACR-CHES-2015**, 21 jun. 2015. Disponivel em: <https://eprint.iacr.org/2015/612>.