UNIVERSIDADE FEDERAL DA BAHIA

ESCOLA POLITÉCNICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

JOÃO CARLOS NUNES BITTENCOURT

IMPLEMENTAÇÃO EM HARDWARE RECONFIGURÁVEL DO ALGORITMO DE CRIPTOGRAFIA CLEFIA COM SUPORTE COMPLETO À EXPANSÃO DE CHAVES

SALVADOR

ABRIL, 2016

JOÃO CARLOS NUNES BITTENCOURT

IMPLEMENTAÇÃO EM HARDWARE RECONFIGURÁVEL DO ALGORITMO DE CRIPTOGRAFIA CLEFIA COM SUPORTE COMPLETO À EXPANSÃO DE CHAVES

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal da Bahia como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.
Orientador: Wagner Luiz Alves de Oliveira

Salvador

abril, 2016

B624 Bittencourt, João Carlos Nunes.
Implementação em hardware reconfigurável do algoritmo de criptografia CLEFIA com suporte completo à expansão de chaves/ João Carlos Nunes Bittencourt. – Salvador, 2016.
125 f. : il. color.

Orientador: Prof. Dr. Wagner Luiz Alves de Oliveira.

Dissertação (Mestrado) – Universidade Federal da Bahia. Escola Politécnica, 2016.

1. CLEFIA - Algoritmo. 2. Criptografia. 3. Hardware. I. Oliveira, Wagner Luiz Alves de. II. Universidade Federal da Bahia. III. Título.

CDD: 005.2

João Carlos Nunes Bittencourt

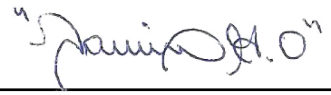
Implementação em Hardware Reconfigurável do Algoritmo de Criptografia CLEFIA com Suporte Completo à Expansão de Chaves

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal da Bahia como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Trabalho aprovado. Salvador, 29 de abril de 2016:

Professor Doutor
Wagner Luiz Alves de Oliveira (Orientador)

Professor Doutor
Jés de Jesus Fiais Cerqueira
Universidade Federal da Bahia



Professor Doutor
Edward David Moreno Ordonez
Universidade Federal de Sergipe

Professor Doutor
Edson Pinto Santana
Universidade Federal da Bahia

Salvador
abril, 2016

Este trabalho é dedicado à minha esposa Claudia
e aos meus filhos João Gabriel e João Miguel.

"O sábio nunca diz tudo o que pensa,
mas pensa sempre tudo o que diz."
(Aristóteles)

BITTENCOURT, J.C.N. Implementação em Hardware Reconfigurável do Algoritmo de Criptografia CLEFIA com Suporte Completo à Expansão de Chaves. 125 f. il. color. Dissertação (Mestrado) – Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal da Bahia. Salvador, 2016.

RESUMO

O CLEFIA é um algoritmo de criptografia simétrica proposto e desenvolvido pela Sony Corporation com ênfase em Gerenciamento de Direitos Autorais.A estrutura proposta para o CLEFIA suporta chaves compartilhadas de 128, 192 e 256 bits. O algoritmo aperfeiçoa a segurança da cifra a partir do uso de técnicas como *Diffusion Switch Mechanisms*, representada por múltiplas matrizes de difusão organizadas em uma ordem predeterminada, tendo em vista garantir imunidade contra ataques diferenciais e lineares. *Whitening Keys* também são utilizadas, a partir da combinação do dado com partes da chave de codificação, antes da primeira iteração e ao final do processo de codificação. Trabalhos recentes indicam que o CLEFIA mostra-se uma solução eficiente, particularmente quando implementada em hardware, com estrutura compatível a padrões populares como o AES. Estruturas compactas em hardware têm significativa importância no projeto de sistemas embutidos, tais como aplicações com RFID ou plataformas de Sistemas-em-Chip, como pequenos dispositivos de comunicação. Apesar de pesquisas recentes terem explorado implementações do CLEFIA baseadas em tecnologias ASIC, soluções em FPGA podem também ser exploradas devido a suas vantagens em termos de adaptatividade computacional, reduzido *time-to-market* e custos de projeto para soluções dedicadas.

Este trabalho propõe uma estrutura em hardware compacta e de alta taxa de transferência de dados, para o cômputo do algoritmo CLEFIA e seu respectivo mecanismo de expansão de chaves completo. Embora o presente estado da arte suporte apenas expansão de chaves de 128 bits, característica justificada pela necessidade de modificações na rede Feistel utilizada na cifra, neste trabalho demonstramos ser viável a implementação de uma estrutura com suporte completo à expansão de chaves do CLEFIA, ao custo de 200 SLICEs e baixo impacto em termos de desempenho, com frequência máxima de 369 MHz. Os resultados foram obtidos a partir da utilização de registradores de deslocamento endereçáveis, presentes em dispositivos FPGA modernos, e uma estrutura de controle de escalonamento adaptável, permitindo o cômputo das redes Feistel *4-branch* e *8-branch* sobre a mesma estrutura. Os resultados obtidos a partir de a implementação do circuito em um dispositivo Xilinx Virtex 5 sugerem que taxas de transferência superiores a 1 Gbps podem ser alcançadas com baixo custo em termos de área, com medidas de eficiência semelhantes àquelas apresentadas pelo estado da arte.

Palavras-chaves: Criptografia. CLEFIA. Hardware compacto. FPGA.

BITTENCOURT, J.C.N. CLEFIA FPGA Implementation with Full Key Support. 125 p. il. color. Thesis (Master) – Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal da Bahia. Salvador, 2016.

ABSTRACT

CLEFIA is a symmetrical block ciphering algorithm proposed and developed by SONY Corporation focused in Digital Rights Management (DRM) purposes. CLEFIA blockcipher supports shared key lengths of 128, 192, and 256 bits. The algorithm improves the security of encryption with the use of techniques such as Diffusion Switch Mechanisms, consisting of multiple diffusion matrices in a predetermined order, to ensure immunity against differential and linear attacks. Whitening Keys are also used, combining data with portions of the key before the first round and after the last round. Recent works had reported that CLEFIA is a highly efficient solution, particularly in hardware implementations, and its structure indicated its compatibility with popular algorithms such as AES. Compact hadware structures are very significant when developing small embedded systems, such as RFID or System-on-Chip platforms, such as comunication devices. Althought researchers have being exploring CLEFIA implementation over ASIC technologies, FPGAs solutions can also be explored due to their advantages in computation adaptability, time-to-market, development costs, and deployment time for dedicated solutions.

In this work a compact and high throughput hardware structure is proposed allowing for the computation of the novel 128-bit CLEFIA encryption algorithm and its associated full key expansion. While the existing state of the art only supports the 128-bit key schedule, given the needed modification to the CLEFIA Fiestel network, herein we show that with a small area and low performance impact, full key expansion can be deployed. The hardware synthesis of such a structure resulted in a ocupation of 200 SLICEs and a maximum operating frequency of about 369 MHz. This is achieved by using addressable shift registers available in modern FPGAs and with adaptable scheduling, allowing to compute the 4 and 8 branch CLEFIA Feistel network within the same structure. The obtained experimental results on a Xilinx Virtex 5 FPGA suggest that throughputs above 1 Gbps can be achieved, achieving efficiency metrics similar to limited state of the art.

Key-words: Blockcipher. CLEFIA. Compact hardware. FPGA.

LISTA DE ILUSTRAÇÕES

Figura 1 – Modelo conceitual do CLEFIA. 28
Figura 2 – Funções F do CLEFIA. 29
Figura 3 – Caminho de dados do CLEFIA ($GFN_{4,n}$). 32
Figura 4 – Estrutura Feistel 8-*branch* usada na expansão de chaves do CLEFIA. . 34
Figura 5 – Representação da função *DoubleSwap* Σ. 35
Figura 6 – Distribuição dos CLBs, apresentado no Guia do Usuário da Xilinx Virtex 5 (Xilinx, 2012). 40
Figura 7 – Arquitetura proposta por Proença e Chaves (2011). 44
Figura 8 – Arquitetura de expansão de chaves de 128 bits (CHAVES, 2013). . . . 46
Figura 9 – Arquitetura de cifra dupla (RESENDE; CHAVES, 2015). 46
Figura 10 – Proposta da estrutura híbrida $GFN_{4,n}/GFN_{8,n}$. 50
Figura 11 – Proposta de arquitetura para a estrutura 8/4-branch do CLEFIA. . . . 53
Figura 12 – Proposta de arquitetura para a estrutura 8/4-branch do CLEFIA com redução do caminho crítico. 56
Figura 13 – Estrutura proposta para o mecanismo de Expansão de Chave. 57
Figura 14 – Proposta de estrutura de expansão de chave completa com uso de registradores de deslocamento (SRL). 59
Figura 15 – Estrutura final da Feistel 8/4-branch do CLEFIA. 62
Figura 16 – Visão detalhada da organização das BRAMs para as *T-boxes*. 64
Figura 17 – Estrutura final do circuito de Expansão de Chave. 65
Figura 18 – Detalhamento da estrutura de execução da função Σ. 67
Figura 19 – Visão detalhada da organização da memória de chaves. 68
Figura 20 – Representação do Co-processador de cifra do CLEFIA. 79
Figura 21 – Fluxograma da Máquina de Estados para a Unidade de Controle do Co-processador de Cifra do CLEFIA. 81
Figura 22 – Representação do Circuito Final de Expansão de Chaves do CLEFIA. . 82
Figura 23 – Fluxograma da máquina de estados para a Unidade de Controle do mecanismo de Expansão de Chave do CLEFIA referente à introdução da chave de entrada. 83
Figura 24 – Fluxograma da máquina de estados para a Unidade de Controle do mecanismo de Expansão de Chave do CLEFIA referente à geração das *Round Keys*. 85
Figura 25 – Representação do co-processador de cifra do CLEFIA com Expansão de Chave. 86
Figura 26 – Esquemático do SLICEL presente no dispositivo Xilinx Virtex 5, como descrito no Xilinx's Virtex-5 FPGA User Guide 123

Figura 27 – Esquemático do SLICEM presente no dispositivo Xilinx Virtex 5, como descrito no Xilinx's Virtex-5 FPGA User Guide 124
Figura 28 – Esquemático do BRAM presente no dispositivo Xilinx Virtex 5, como descrito no Xilinx's Virtex-5 FPGA User Guide 125

LISTA DE TABELAS

Tabela 1 – Escalonamento da entrada da chave K, de 128 bits, no mecanismo de expansão de chave e introdução da chave no bloco GFN. 70
Tabela 2 – Escalonamento da entrada da chave K, de 192 bits, no mecanismo de expansão de chave, geração das *witening keys* e introdução da chave no bloco GFN. 70
Tabela 3 – Escalonamento da entrada da chave K, de 256 bits, no mecanismo de expansão de chave, geração das *witening keys* e introdução da chave no bloco GFN. 71
Tabela 4 – Escalonamento de operações da da cifra do CLEFIA, onde $a'i$ e $b'i$ são resultados oriundos da Função-F e n o total de iterações. 73
Tabela 5 – Escalonamento de operações do processo de obtenção da chave intermediária L no modo 8-*branch*, onde $a'i$, $b'i$, $c'i$ e $d'i$ são resultados da Função-F. 74
Tabela 6 – Escalonamento da permutação realizada pela função Σ. 76
Tabela 7 – Escalonamento de operações da expansão de chave de cifra do CLEFIA-128, onde Σ_j^i representa o bloco resultante da função Σ correspondente à iteração i. 77
Tabela 8 – Escalonamento de operações da expansão de chave de cifra do CLEFIA-256., onde Σ_y^k corresponde ao bloco resultante da função Σ correspondente à iteração k e n especifica o total de ciclos de clock. 78
Tabela 9 – Utilização de recursos do Co-processador de cifra. 90
Tabela 10 – Análise de desempenho do Co-processador de cifra. 91
Tabela 11 – Utilização de recursos do núcleo de Expansão de Chave. 92
Tabela 12 – Utilização de recursos das estruturas propostas. 93
Tabela 13 – Análise de desempenho do Co-processador com Expansão de Chave. . . 94
Tabela 14 – Comparação de performance em relação ao Estado da Arte. 96
Tabela 15 – Ocupação do Registrador de Deslocamento e endereçamento da saída na cifra do CLEFIA. 108
Tabela 16 – Escalonamento de operações da cifra do CLEFIA, onde a_i e b_i são resultados oriundos da Função-F e n o total de iterações. 109
Tabela 17 – Ocupação do Registrador de Deslocamento e endereçamento de saída na cifra do CLEFIA para chaves de 192 bits. 111
Tabela 18 – Escalonamento de operações do processo de obtenção da chave intermediária L no modo 8-*branch*, onde $a'i$, $b'i$, $c'i$ e $d'i$ são resultados da Função-F. 114

Tabela 19 – Escalonamento da entrada da chave K, de 128 bits, no mecanismo de expansão de chave e introdução da chave no bloco GFN. 115
Tabela 20 – Ocupação do Registrador de Deslocamento da Chave de Entrada (SRL16). 115
Tabela 21 – Escalonamento de operações da expansão de chave de cifra do CLEFIA-128, onde Σ_j^i corresponde ao bloco resultante da função Σ na iteração i. 116
Tabela 22 – Escalonamento da entrada da chave K, de 192 bits, no mecanismo de expansão de chave, geração das *witening keys* e introdução da chave no bloco GFN. 117
Tabela 23 – Escalonamento da entrada da chave K, de 256 bits, no mecanismo de expansão de chave, geração das *witening keys* e introdução da chave no bloco GFN. 118
Tabela 24 – Ocupação do Registrador de Deslocamente para Chaves de Entrada de 192 bits. Onde os elementos WK_i representam os blocos de 32 bits que compõem a Whitening Key. 119
Tabela 25 – Ocupação do Registrador de Deslocamente para Chaves de Entrada de 256 bits. Onde os elementos WK_i representam os blocos de 32 bits que compõem a Whitening Key. 120

LISTA DE ABREVIATURAS E SIGLAS

ASIC	Application Specific Integrated Circuit
AES	Advanced Encryption Standard
BRAM	Block RAM
CBC	Cipher Block Chaining
CLB	Cell Logic Blocks
CRIPTEC	Comitê Japonês de Pesquisa e Avaliação Criptográfica
DES	Data Encryption Standard
DSM	Diffusion Switch Mechanism
DSP	Digital Signal Processor
ECB	Electronic Code Block
EDA	Electronic Design Automation
FF	Flip-flop
FPGA	Fiel-Programable Gate Array
FSM	Finite State Machine
GFN	Generalized Feistel Network
GF	Galois Field
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
LUT	Lookup Tables
MUX	Multiplexador
RAM	Randon Access Memory
ROM	Read Only Memory
SRAM	Static Randon Access Memory
UC	Unidade de Controle

SUMÁRIO

1 INTRODUÇÃO 23
1.1 Objetivos e Metodologia 24
1.2 Organização do Trabalho 24

2 CLEFIA 27
2.1 Funções F 28
2.2 Implementações do DSM usando (*T-Boxes*) 30
2.3 Processo de Codificação 31
2.4 Expansão de Chaves 33
2.4.1 Expansão de Chaves de 128 bits 35
2.4.2 Expansão de Chaves de 192 bits 36
2.4.3 Expansão de Chaves de 256 bits 36

3 ESTADO DA ARTE 39
3.1 Tecnologia de Dispositivos FPGA 39
3.2 Algoritmos de Criptografia em FPGA 41
3.3 Estado da Arte do CLEFIA em FPGA 42
3.3.1 Pipeline Desdobrado de Kryjak e Gorgón (2009) 43
3.3.2 Arquitetura Compacta de Proença e Chaves (2011) 43
3.3.3 Arquiteturas Iterativa e Serial de Hanley e O'Neill (2012) 44
3.3.4 Expansão de Chaves de 128 bits de Chaves (2013) 45
3.3.5 Arquitetura de Cifra Dupla de Resende e Chaves (2015) 46
3.4 Conclusões sobre o Estado da Arte 47

4 PROJETO DE ARQUITETURA DO CLEFIA 49
4.1 Estrutura GFN para Cifra do CLEFIA 50
4.1.1 Mapeamento das T-Boxes em BRAMs 51
4.1.2 Otimizações da Arquitetura 52
4.1.2.1 Redução do Caminho Crítico 54
4.2 Expansão de Chave 55
4.2.1 Utilizando "Registradores de Deslocamento" na Expansão de Chave . . 58
4.3 Conclusões Gerais sobre o Projeto da Arquitetura 60

5 DETALHES DE IMPLEMENTAÇÃO E MELHORIAS 61
5.1 Gerenciamento das *Whitening Keys* e Chave de Codificação . . 61
5.2 Circuito Final e Detalhes de Implementação 62
5.2.1 Estrutura Feistel 8/4-branch 62

5.2.2 Mapeamento das T-boxes em BRAM 63
5.2.3 Estrutura de Expansão de Chaves Completa 64
5.2.4 Organização da Memória de Chaves 67
5.3 Descrição do Escalonamento de Operações 69
5.3.1 Obtenção da chave K e Cálculo das Whitening Keys 69
5.3.2 Circuito de Codificação/Decodificação $GFN_{4/8,n}$ 71
5.3.3 Obtenção das Round Keys 76
5.4 Unidades de Controle e Circuito Final 79
5.4.1 Estrutura Feistel 8/4-branch 79
5.4.2 Mecanismo de Expansão de Chaves 82
5.4.3 Co-processador de Cifra com Expansão de Chave 86
5.5 Considerações Finais sobre a Implementação da Arquitetura . 87

6 ANÁLISE DOS RESULTADOS 89
6.1 Análise do Núcleo de Cifra do CLEFIA 89
6.2 Análise de Desempenho para Expansão de Chave 91
6.3 Comparação com o Estado da Arte 94
6.4 Conclusões sobre a Análise dos Resultados 98

7 CONCLUSÕES 99
7.1 Trabalhos Futuros 100

REFERÊNCIAS 103

APÊNDICES 105

APÊNDICE A – DETALHAMENTO DA CIFRA DO CLEFIA EM PIPELINE 108

APÊNDICE B – DETALHAMENTO DA PRODUÇÃO DA CHAVE INTERMEDIÁRIA L DO CLEFIA EM PIPELINE 111

APÊNDICE C – DETALHAMENTO DA EXPANSÃO DE CHAVES DE 128 BITS 115

APÊNDICE D – DETALHAMENTO DA EXPANSÃO DE CHAVES DE 192 E 256 BITS 117

ANEXOS 121

ANEXO A – ESQUEMÁTICO DOS COMPONENTES DO DISPOSITIVO FPGA 123

1 INTRODUÇÃO

O mercado de sistemas embutidos tem crescido significativamente nas últimas décadas. Atualmente, o uso de sistemas móveis e embutidos, também conhecidos como "sistemas (embutidos) inteligentes" ou *smart devices*, já superam o uso de computadores pessoais. Grande parte destes novos sistemas intercedem no cotidiano das pessoas com o mundo que as cercam, servindo-as dos mais diversos meios de comunicação. Sistemas computacionais modernos têm que gerenciar cada vez mais informações confidenciais ou ditas "sensíveis", demandando por mecanismos de proteção cada dia mais sofisticados.

Na era da Informação, a confidencialidade não está mais limitada a governantes ou agências militares, fazendo-se necessária também para grandes e pequenas empresas ou até mesmo para pessoas comuns. A transmissão de dados digitais através de canais públicos, como redes *wireless* e de telefonia celular, tem aumentado constantemente. O uso destes sistemas para realização de transações bancárias e comerciais, como o fornecimento de números de cartões de crédito, já se tornou prática comum entre usuários de sistemas computacionais. Para proteger os dados em sistemas computacionais e de comunicação do acesso não autorizado, foi necessária a adoção de mecanismos de criptografia capazes de impedir a interceptação de informações sigilosas transmitidas através de meios públicos de acesso. Neste sentido, verifica-se a necessidade de implementações compactas destes mecanismos. Um destes mecanismos é o algoritmo CLEFIA, proposto e desenvolvido pela Sony Corporation (SHIRAI et al., 2007).

Em 2012, o CLEFIA foi declarado padrão internacional na ISO/IEC 29192-2 (ISO/IEC 29192-2, 2012). Mais recentemente, o Comitê Japonês de Pesquisa e Avaliação Criptográfica (CRYPTEC) incluiu o CLEFIA na sua lista de *Candidate Recommended Ciphers*, na revisão de 2013. Seu algoritmo suporta chaves de codificação de 128, 192 e 256 bits, proporcionando um ambiente criptográfico seguro, através do uso de mecanismos como chaveamento difuso e *whitening keys*, dentre outros, visando proporcionar imunidade contra ataques diferenciais e lineares (SHIRAI; KYOJI, 2006). Trabalhos recentes que envolvem o CLEFIA têm destacado seu desempenho, particularmente em implementações em hardware baseadas em ASIC ou em tecnologias FPGA. A maioria das abordagens busca estruturas compactas, ao mesmo tempo que tenta manter alto desempenho, otimizando os recursos computacionais e explorando o paralelismo entre operações.

Em virtude da complexidade associada a uma rede Feistel 8-*branch* na expansão de chaves de criptografia de 192 e 256 bits, na maioria das estruturas atuais que provêm um mecanismo de expansão de chave, adota-se apenas o uso de chaves de 128 bits (4-*branch*) (SUGAWARA et al., 2008; CHAVES, 2013; HANLEY; O'NEILL, 2012; AKISHITA; HIWATARI, 2012). Por outro lado, estruturas que consideram a o cômputo

offline da expansão de chaves, mesmo não dando suporte à expansão de chaves de todos os tamanhos, ou mesmo nenhuma delas, nenhum impacto é verificado no processo de criptografia, uma vez que a única diferença no núcleo de cifra advém do número de iterações necessárias (KRYJAK; GORGóN, 2009; PROENÇA; CHAVES, 2011). O principal fator que influencia na decisão de introduzir uma estrutura em hardware para expansão de chaves está associada ao aumento da ocupação em *chip* associadas à soluções que implementam um mecanismo completo. Este fator é sensivelmente agravado quando considerada a expansão de chaves de 192 e 256 bits.

1.1 OBJETIVOS E METODOLOGIA

O principal objetivo deste trabalho é demonstrar a viabilidade da implementação de uma estrutura de criptografia para o CLEFIA, com suporte tanto para a rede Feistel 4-*branch* quanto para a rede 8-*branch* e seu respectivo mecanismo de expansão para todos os tamanhos de chave, com baixo custo e atingindo *throughputs* semelhantes àqueles presentes na literatura.

Para tanto, no que se refere às arquiteturas de criptografia, foram consideradas as estruturas propostas por Kryjak e Gorgón (2009), Proença e Chaves (2011) e Resende e Chaves (2015). Tendo em vista as estruturas de expansão de chave, apenas o trabalho proposto por Chaves (2013) foi considerado, devido às semelhanças entre a sua proposta e a apresentada neste trabalho.

Como prova de conceito, o presente trabalho foi implementado em dispositivos com tecnologia *Field-Programmable Gate Array* (FPGA), dado a sua cada vez maior presença em sistemas embutidos, adaptabilidade e facilidade de prototipação. Os resultados experimentais sugerem que a partir do uso de blocos de memória embutidos nos dispositivos FPGA (BRAM), e com o uso de registradores de deslocamento baseados em células lógicas primitivas, uma arquitetura eficiente para implementação completamente compatível com o CLEFIA pode ser obtida. A um custo de 200 células lógicas (*Slices*) e 3 BRAMs, *throughputs* acima de 1.2 Gbps podem ser alcançados, com eficiência superior em relação àquelas encontradas na literatura.

1.2 ORGANIZAÇÃO DO TRABALHO

Além do capítulo introdutório, este trabalho se encontra estruturado da seguinte forma: no Capítulo 2 são apresentados os detalhes de implementação do algoritmo CLEFIA; o Capítulo 3 apresenta uma breve introdução aos trabalhos mais relevantes, no que diz respeito a implementações do CLEFIA e sua integração em dispositivos FPGA; o Capítulo 4 é dedicado a apresentar as soluções propostas para codificação/decodificação e expansão de chaves do CLEFIA, enquanto o Capítulo 5 continua apresentando os detalhes da

implementação; um estudo comparativo frente aos trabalhos relacionados é apresentado no Capítulo 6. Por fim, o Capítulo 7 conclui este trabalho com as considerações finais e sugestões de trabalhos futuros. Apêndices são referenciados ao longo do texto, tendo em vista acrescentar informações extras.

2 CLEFIA

Comumente denominados como cifras de bloco (*block ciphers*), grande parte dos algoritmos de criptografia modernos são construídos a partir de três processos principais: codificação, decodificação e expansão de chaves (*key scheduling*). De maneira geral, cifras de blocos operam sobre conjuntos de bits de tamanho fixo, denominados blocos. O tamanho dos blocos pode variar de acordo com as especificidades de cada cifra. Para cada bloco de dados, um conjunto de iterações sob a codificação/decodificação é realizado. Cada iteração implica no uso de uma chave de iteração (*round key*) submetida ao bloco de entrada. Estas chaves são derivadas da chave de codificação por meio de um processo conhecido como *key scheduling*, ou expansão de chave. Cada algoritmo define um conjunto de transformações para a devida obtenção das suas *round keys*. Em alguns casos, as chaves destinadas à codificação podem ser utilizadas na decodificação, invertendo-se a ordem de aplicação das mesmas.

CLEFIA é um algoritmo de criptografia simétrico, para blocos de 128 bits, com suporte a chaves de codificação de 128, 192 ou 256 bits. Assim como na maioria dos algoritmos de criptografia atuais, o CLEFIA é composto por um mecanismo de expansão de chave, seguido do processo de transformação da cadeia de caracteres de entrada em um criptograma. Um modelo conceitual associado aos processos que envolvem a execução do algoritmo é apresentado na Figura 1. Este conjunto de transformações é realizada a partir de uma sequência de iterações, empregando um algoritmo relativamente homogêneo (SHIRAI et al., 2007). Esta regularidade facilita o desenvolvimento de arquiteturas compactas em hardware, viabilizando a sua utilização em sistemas embutidos de recursos limitados.

Técnicas empregadas nos mais recentes algoritmos de criptografia também são encontradas no CLEFIA, tais como *Whitening Keys*, Redes de Feistel e Mecanismo de Chaveamento Difuso (do inglês *Diffusion Switch Mechanism* e abreviado como DSM). *Whitening Key* é uma técnica usada para aumentar a segurança em algoritmos iterativos, que consiste em operações que combinam partes do dado a ser criptografado com blocos de dados formados a partir da chave de codificação. No CLEFIA este processo de "branqueamento" é realizando antes da primeira e ao final da última iteração. Redes de Feistel são estruturas largamente utilizadas em projetos de algoritmos de criptografia. Propostas por H. Feistel, no início dos anos 1970, sua estrutura ganhou visibilidade ao ser adotada na implementação do *Data Encryption Standard* (DES), predecessor do atual *Advanced Encryption Standard* (AES). Por fim, o DSM consiste no uso de matrizes de difusão, organizadas em uma ordem pré-determinada, e um conjunto de *S-boxes*, tendo em vista garantir imunidade contra diversos tipos de ataques conhecidos (SHIRAI; KYOJI, 2006). O CLEFIA emprega, em sua estrutura, duas *S-boxes* e duas matrizes de difusão (Sony Corporation, 2007b).

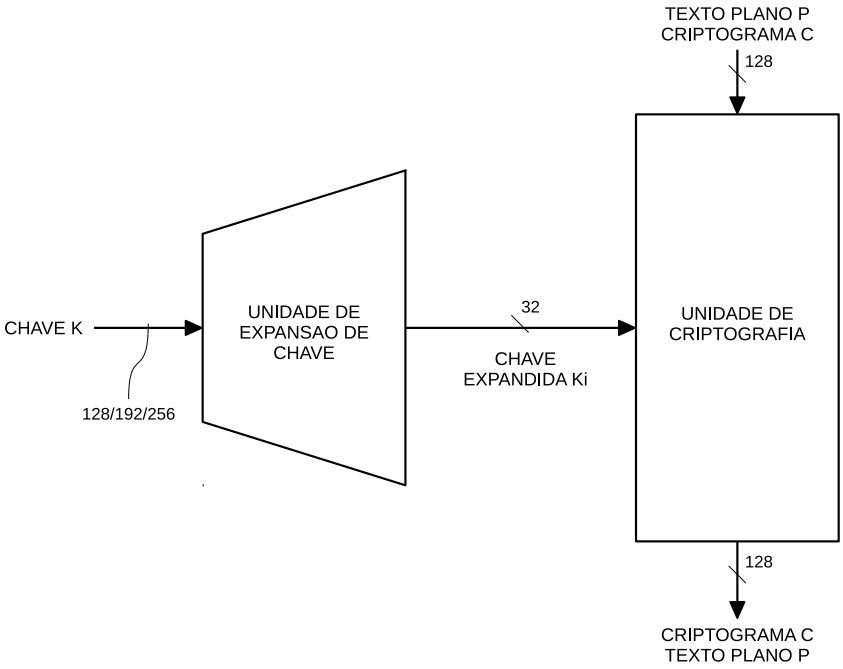
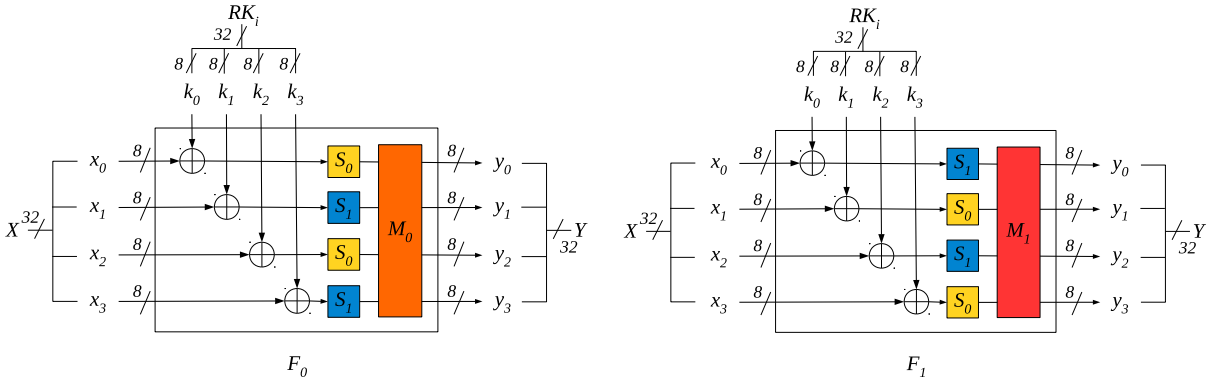


Figura 1 – Modelo conceitual do CLEFIA.

Assim como na maioria dos algoritmos de criptografia iterativos, o CLEFIA requer um conjunto de chaves pré-computadas, utilizadas em cada iteração. Estas chaves são obtidas por meio de um mecanismo de expansão de chave (*key scheduling*), a partir da chave de codificação. O mecanismo proposto para o CLEFIA utiliza as redes de Feistel 4-*branch* e 8-*branch* para tal expansão. Para chaves de 128 bits, a estrutura 4-*branch*, utilizada no processo de cifra, pode ser compartilhada, reduzindo o impacto da implementação da expansão de chave em hardware. Por outro lado, a expansão proposta para chaves de cifra de 192 e 256 bits requer uma rede de Feistel 8-*branch*. Esta característica implica na necessidade de estruturas adicionais, tornando-se um obstáculo para implementação em hardware. Adicionalmente, o mecanismo de expansão de chave emprega a função *DoubleSwap* (Σ). A função Σ é formada a partir de permutações bit a bit sob um bloco de 128 bits, implementadas de modo a aumentar a eficiência do processo de expansão de chaves do CLEFIA (SHIRAI et al., 2007).

2.1 FUNÇÕES F

Duas Funções-F (F_0 e F_1) são empregadas no CLEFIA, de forma independente, a cada iteração, de acordo com a Figura 2. Cada função recebe duas entradas de 32 bits, as quais são associadas ao bloco de dados (X) e a uma *round key* (RK_i). O resultado mapeado é emitido através de um sinal de 32 bits (Y). Cada saída é então combinada com dois blocos de 32 bits oriundos do bloco de entrada ou de uma etapa de permutação realizada à cada iteração do CLEFIA.



(a) Função F_0. (b) Função F_1.

Figura 2 – Funções F do CLEFIA.

Fonte: Adaptado de Sony Corporation (2007a).

Duas funções de substituição baseadas em *S-boxes*, representadas por S_0 e S_1 na Figura 2, são empregadas na composição das funções F_0 e F_1. O resultado da realização de cada *S-box* representa uma matriz 16×16. A saída de cada *S-box* é determinada em função da operação de substituição a partir dos 4 bytes oriundos da incorporação de uma uma *round key* RK_i ao dado X, por meio de uma operação XOR, como descrito na Figura 2. Os primeiros 4 bits, oriundos da operação XOR, são usados para mapear a linha (L), enquanto os 4 bits restantes indicam a coluna (C) a ser mapeada. O resultado da etapa de substituição representa o valor correspondente à intersecção entre as coordenadas $L \times C$, levando a uma nova cadeia de 4 bytes.

A função S_0 é definida a partir da divisão do byte de entrada em dois valores de 4 bits, seguido da sua substituição usando *look up tables* menores. Os valores resultantes das duas substituições são então multiplicados sob um campo finito $GF(2^4)$, descrito em sua forma polinomial como $z = x^4 + x + 1$ (Sony Corporation, 2007a).

Por outro lado, a função S_1 apresenta uma estrutura menos direta em relação a S_0, uma vez que evolve duas transformações afins e uma inversa sob $GF(2^8)$ (Sony Corporation, 2007a).

Como parte integrante do mecanismo de difusão do CLEFIA, duas matrizes de difusão M_0 e M_1 (4×4) são empregadas em F_0 e F_1, respectivamente. Dessa forma, os quatro bytes, oriundos da operação de substituição, são multiplicados por cada linha da matriz e somados sob um campo finito $GF(2^8)$. A expessão oriunda deste processo é definida em 2.1 e o desmembramento desta expressão é descrito em 2.2.

$$\begin{aligned} y &= M_0 \times S(X \oplus RK), \ para\ F_0 \\ y &= M_1 \times S(X \oplus RK), \ para\ F_1 \end{aligned} \tag{2.1}$$

Para F_0

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0x01 & 0x02 & 0x04 & 0x06 \\ 0x02 & 0x01 & 0x06 & 0x04 \\ 0x04 & 0x06 & 0x01 & 0x02 \\ 0x06 & 0x04 & 0x02 & 0x01 \end{pmatrix} \begin{pmatrix} S_0(x_0 \oplus k_0) \\ S_1(x_1 \oplus k_1) \\ S_0(x_2 \oplus k_2) \\ S_1(x_3 \oplus k_3) \end{pmatrix} \tag{2.2}$$

$$M_0(x) = 06x^3 + 04x^2 + 02x + 01$$

Para F_1

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0x01 & 0x08 & 0x02 & 0x0A \\ 0x08 & 0x01 & 0x0A & 0x02 \\ 0x02 & 0x0A & 0x01 & 0x08 \\ 0x0A & 0x02 & 0x08 & 0x01 \end{pmatrix} \begin{pmatrix} S_0(x_0 \oplus k_0) \\ S_1(x_1 \oplus k_1) \\ S_0(x_2 \oplus k_2) \\ S_1(x_3 \oplus k_3) \end{pmatrix} \tag{2.3}$$

$$M_1(x) = 0Ax^3 + 02x^2 + 08x + 01$$

Além de usar duas matrizes de difusão diferentes, as funções F_0 e F_1 diferem na sequência em que as *S-boxes* são mapeadas. Adicionalmente, as similaridades observadas nas matrizes M_0 e M_1 sugerem simplificações por meio do compartilhamento de recursos entre as duas matrizes e da substituição das *S-boxes* e matrizes de difusão por *T-boxes* (SUGAWARA et al., 2008).

2.2 IMPLEMENTAÇÕES DO DSM USANDO (*T-BOXES*)

A combinação das duas *S-boxes* e matrizes de difusão representam o Mecanismo de Chaveamento Difuso, presente no CLEFIA. A partir da identificação de similaridades entre os termos de M_0 e M_1 verifica-se a possibilidade de otimização das operações realizadas no DSM por meio do uso de um mecanismo conhecido como *T-box*. No CLEFIA, uma *T-box* pode ser utilizada para substituir uma *S-box* e a multiplicação matricial que a segue, visando reduzir a complexidade aritmética. Dessa forma, as operações entre as *S-boxes* (S_0 e S_1) e as matrizes de difusão (M_0 e M_1) podem ser substituídas por operações de busca, seguidas da operação XOR (SUGAWARA et al., 2008).

O uso de *T-boxes*, em circuitos ASIC, implica em um grande consumo de área em *chip* (SUGAWARA et al., 2008). Dispositivos FPGA são capazes de explorar esse tipo

de simplificação de forma eficaz, mantendo um baixo consumo de blocos lógicos e alta capacidade de processamento, através de implementações baseadas em memória. Uma *T-box* pode ser implementada de diferentes formas, de acordo com as demandas relativas a capacidade de processamento e custo de área. Um estudo realizado em Sony Corporation (2007b) apresenta quatro tipos de *T-box*. A Equação 2.4 descreve a *T-box Type-I* a partir de oito tabelas, com entradas de 8 bits e saídas de 32 bits.

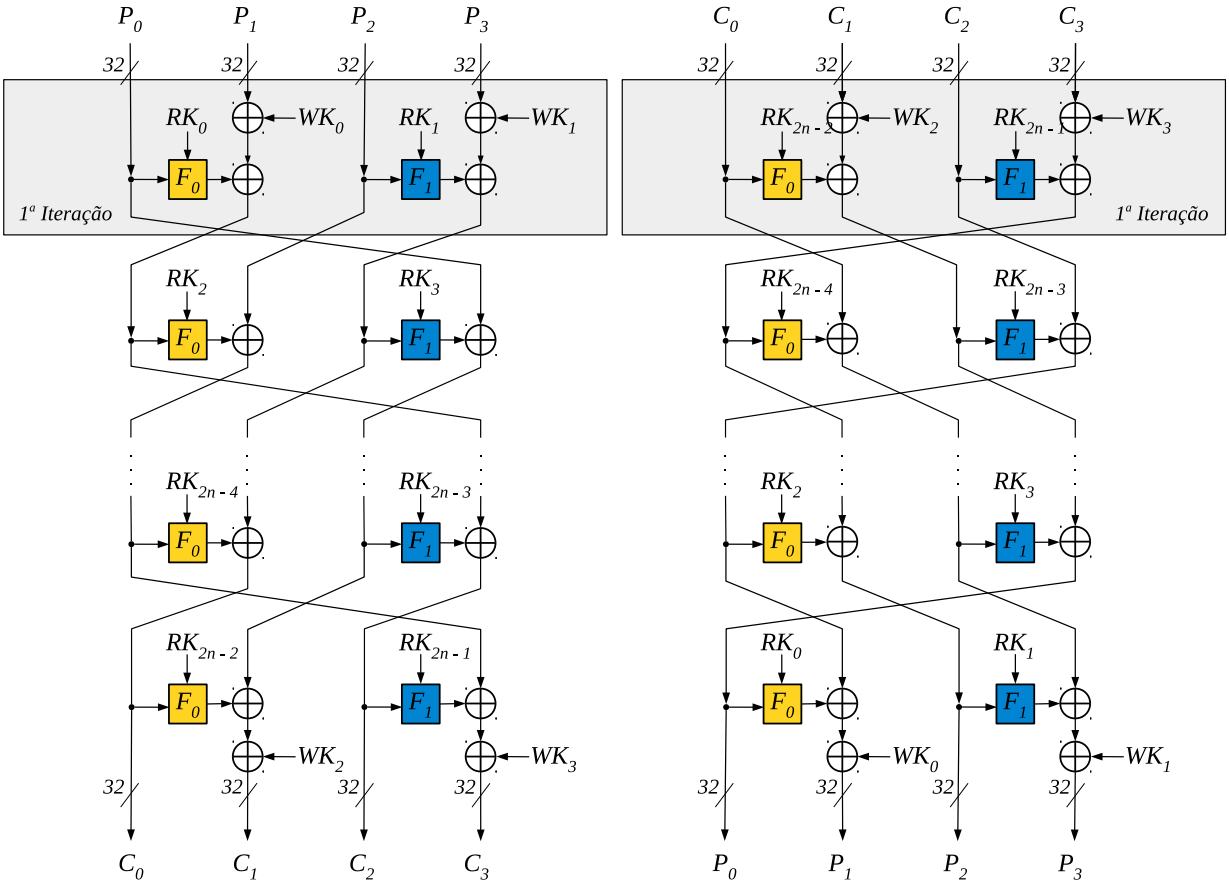
$$\begin{aligned}
T_{00} &= (S_0, 02{\times}S_0, 04{\times}S_0, 06{\times}S_0) \\
T_{01} &= (02{\times}S_1, S_1, 06{\times}S_1, 04{\times}S_1) \\
T_{02} &= (04{\times}S_0, 06{\times}S_0, S_0, 02{\times}S_0) \\
T_{03} &= (06{\times}S_1, 04{\times}S_1, 02{\times}S_1, S_1) \\
T_{10} &= (S_1, 08{\times}S_1, 02{\times}S_1, 0A{\times}S_1) \\
T_{11} &= (08{\times}S_0, S_0, 0A{\times}S_0, 02{\times}S_0) \\
T_{12} &= (02{\times}S_1, 0A{\times}S_1, S_1, 08{\times}S_1) \\
T_{13} &= (0A{\times}S_0, 02{\times}S_0, 08{\times}S_0, S_0)
\end{aligned} \tag{2.4}$$

2.3 PROCESSO DE CODIFICAÇÃO

O principal elemento de processamento do CLEFIA é uma rede generalizada iterativa de Feistel 4-*branch*, Tipo-2, definida como $GFN_{4,n}$, onde n representa a quantidade de iterações. O tamanho da chave de codificação influencia diretamente na quantidade de iterações: 18, 22 ou 26, para chaves de 128, 192 e 256 bits, respectivamente (SHIRAI; MIZUNO, 2007). A estrutura de Feistel, utilizada no CLEFIA, é composta por quatro linhas de dados de entrada/saída, formadas por blocos de 32 bits, distribuídas em pares em meio às Funções-F. Assim como na maioria dos blocos de cifra simétricos, o CLEFIA emprega apenas operações aritméticas simples e de troca/substituição de bytes. Além disso, as *S-boxes* e matrizes de difusão podem ser fundidas e transformadas em um processo de mapeamento usando *Look Up Table* (LUT).

O processo de codificação, descrito na Figura 3a, recebe um conjunto de dados de entrada representado por uma cadeia de caracteres simples (*plaintext*) de 128 bits $P = P_0|P_1|P_2|P_3$, quatro *whitening keys* de 32 bits $WK = WK_0|WK_1|WK_2|WK_3$, e um conjunto de *round keys* RK_i. O resultado deste processo, $C = C_0|C_1|C_2|C_3$, representa um criptograma de 128 bits.

O primeiro passo do processo de cifra é representado a partir da operação XOR entre a segunda e a quarta palavra da cadeia de caracteres de entrada P (P_1 e P_3) com o primeiro e o segundo bloco de 32 bits da *whitening key* (WK_0 e WK_1). Este processo é denominado de *whitening* e as sub-chaves são conhecidas como *whitening keys*. Esta etapa, destacada na Figura 3a, representa a primeira parte do processo de *whitening*. Após esta operação, as iterações são executadas.



(a) Codificação. (b) Decodificação.

Figura 3 – Caminho de dados do CLEFIA ($GFN_{4,n}$).

Fonte: Adaptado de Sony Corporation (2007a).

O cálculo de cada iteração é formado a partir das funções paralelas não-lineares F_0 e F_1. Suas entradas são compostas, respectivamente, por uma cópia da primeira e terceira palavras, e duas *round keys*. A saída de cada uma destas funções é processada através de uma operação XOR com a segunda e quarta palavras, respectivamente. A primeira e a terceira palavras não são submetidas a processamento algum, sendo apenas permutadas, juntamente com as outras duas palavras processadas. Este processo se dá a partir de um deslocamento circular à esquerda de todas as palavras de 32 bits, como ilustra a Figura 3a. Esta operação é definida como Feistel *word swap*.

Na última iteração, a Feistel *word swap* é substituída por uma operação XOR entre a segunda e a quarta palavra com as últimas duas *whitening keys* (WK_2 e WK_3). A cadeia de caracteres cifrados C é então obtida diretamente da saída do bloco de cifra. O Algoritmo 1 sintetiza as operações descritas para a obtenção do criptograma C a partir da cadeia P, do conjunto de *round keys* de 32 bits RK e da *whitening key* WK. Esta representação considera que operações são realizadas sobre blocos de dados 32 bits.

Algoritmo 1 Cifra de uma cadeia de caracteres P

Input: P, RK, WK

$T_0|T_1|T_2|T_3 \leftarrow P_0|P_1 \oplus WK_0|P_2|P_3 \oplus WK_1$

for $i = 0$ **to** $n - 1$ **do**

$T_1 = T_1 \oplus F_0(RK_{2i}, T_0), T_3 = T_3 \oplus F_1(RK_{2i+1}, T_2)$

$T_0|T_1|T_2|T_3 \leftarrow T_1|T_2|T_3|T_0$

end for

$C_0|C_1|C_2|C_3 \leftarrow T_0|T_1 \oplus WK_2|T_2|T_3 \oplus WK_3$

Output: C

Em virtude da simetria oriunda da implementação de uma estrutura baseada em rede Feistel, o processo de decodificação é idêntico ao de codificação. Dessa forma, as mesmas unidades computacionais podem ser utilizadas, diferindo apenas na ordem em que as operações são realizadas. Além disso, como descrito na 3b, as *round keys* e blocos de 32 bits da *whitening key* devem ser fornecidos na ordem inversa (SHIRAI et al., 2007).

Os processos de codificação e decodificação empregados no CLEFIA fazem uso de duas *round keys*, a cada iteração, além de uma *whitening key* de 128 bits distribuída em blocos sequenciais de 32 bits. Estas chaves são obtidas a partir da chave de cifra de 128, 192 ou 256 bits, por meio de um mecanismo chamado de expansão de chaves (SHIRAI et al., 2007), descrito na sequência.

2.4 EXPANSÃO DE CHAVES

Como descrito nas seções anteriores, a cada iteração duas *round-keys* de 32 bits são utilizadas. O CLEFIA emprega em sua estrutura 18, 22 ou 26 iterações, para chaves de 128, 192 e 256 bits, respectivamente. Dessa forma, um total de 36, 44 ou 52 *round keys* (dependendo da quantidade de iterações) são utilizadas, além de quatro *whitening keys* adicionais (SHIRAI et al., 2007). Para obter este conjunto de chaves, a chave de codificação de 128, 192 ou 256 bits precisa ser expandida. Esta expansão é realizada a partir de um algoritmo de expansão de chave (SHIRAI et al., 2007).

A expansão para uma chave de 128 bits utiliza a mesma rede GFN 4-*branch* originária do processo de criptografia do CLEFIA. A diferença na expansão é que o dado de entrada da GFN passa agora a corresponder à própria chave de criptografia. Além disso, as *round keys* são substituídas por um conjunto de constantes definido no documento de especificação do algoritmo (Sony Corporation, 2007a). Note ainda que o processo de *whitening* não é realizado na expansão de chaves, logo nenhuma *whitening key* é utilizada.

Ao considerar a expansão de chaves de 192 e 256 bits, a estrutura GFN se torna uma 8-*branch* ($GFN_{8,n}$), descrita na Figura 4. Neste caso a entrada corresponde à uma combinação de $K = K_L|K_R$. As *round keys* utilizadas na GFN são também substituídas por um conjunto diferente de valores constantes CON_i. A estrutura de Feistel 8-*branch* faz

uso de dois pares das funções F_0 e F_1, sendo capaz de processar oito palavras de entrada a cada iteração. Nesta rede, cópias da primeira e terceira palavras são fornecidas às funções F_0 e F_1, ocorrendo o mesmo com a quinta e sétima palavras, como descrito na Figura 4. O resultado de cada função é processado a partir do operador XOR com a segunda, quarta, sexta e oitava palavras. Em seguida, os valores resultantes são permutados. Na última iteração, as saídas L_L e L_R são obtidas diretamente, como descrito na Figura 4.

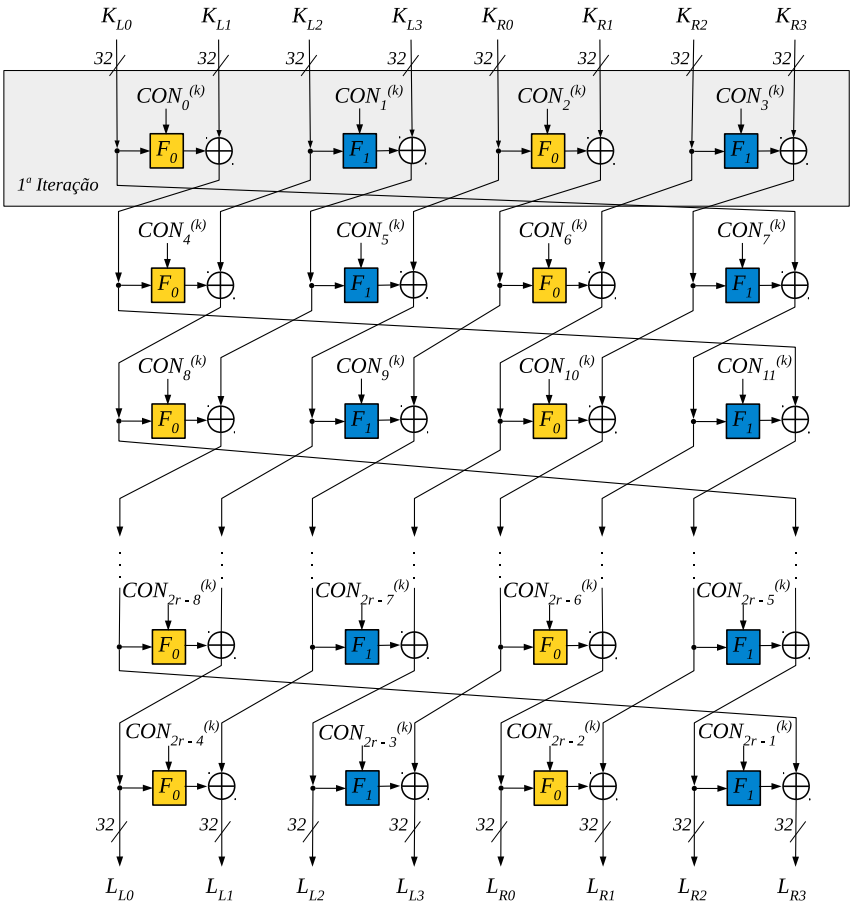


Figura 4 – Estrutura Feistel 8-*branch* usada na expansão de chaves do CLEFIA.

Fonte: Adaptado de Sony Corporation (2007a).

No lugar de uma conjunto de caracteres cifrados, o resultado da GFN corresponde à um bloco de dados de 128 bits $L = L_0|L_1|L_2|L_3$, para chaves de mesmo tamanho. Para chaves de 192 ou 256 bits, o resultado corresponde à uma cadeia binária de 256 bits, dividida em dois blocos de 128 bits $L_L = L_{L0}|L_{L1}|L_{L2}|L_{L3}$ e $L_R = L_{R0}|L_{R1}|L_{R2}|L_{R3}$.

Ao final do processamento da GFN, o resultado (L ou L_L e L_R) é expandido de forma iterativa, utilizando a função *double swap* (Σ). A expressão que corresponde à função Σ é definida na Equação 2.5, onde $X[a-b]$ corresponde a um conjunto de bits delimitado pelo intervalo entre os bits a e b da cadeia binária X, onde 0 corresponde ao índice do bit mais significativo (Figura 5) (Sony Corporation, 2007a).

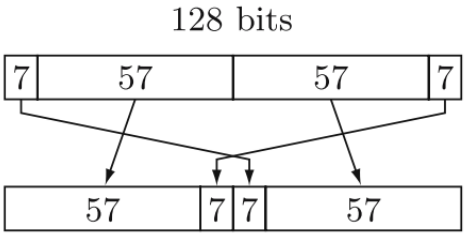


Figura 5 – Representação da função *DoubleSwap* Σ.

Durante a execução da função Σ, um conjunto definido de cadeias de bits de um bloco de 128 bits é permutado. Esta operação ($\Sigma(Y)$) resulta em outro bloco de mesmo tamanho.

$$\Sigma(X) = X[7-63]|X[121-127]|X[0-6]|X[64-120] \tag{2.5}$$

Com isso, as *round keys* de 32 bits são obtidas a partir da soma alternada entre blocos de 32 bits de L, K e $\Sigma(X)$, com um novo conjunto predefinido de constantes, também oriundas da especificação do algoritmo (Sony Corporation, 2007a).

A partir das informações acima, verifica-se que o mecanismo de expansão de chave do CLEFIA opera de forma diferente para cada tamanho de chave. As seções seguintes visam detalhar o funcionamento destes três possíveis modos de operação.

2.4.1 Expansão de Chaves de 128 bits

O processo de expansão de chave do CLEFIA é dividido em duas etapas: (i) geração da *whitening key* (WK), a qual é formada por quatro blocos de 32 bits (WK_i), a partir de uma chave de entrada K; e (ii) expansão da chave K em um conjunto de *round keys*. Para uma chave de codificação de 128 bits (K^{128}), as quatro *whitening keys* são extraídas diretamente a partir da chave de entrada:

$$WK_0|WK_1|WK_2|WK_3 \leftarrow K. \tag{2.6}$$

O processo de obtenção das *round keys* consiste em um mecanismo computacional complexo, dividido em duas etapas. No primeiro passo, a chave L é calculada sobre uma $GFN_{4,12}$. Na segunda etapa, o bloco L é manipulado de modo a produzir o conjunto de *round keys* necessárias.

O processo de obtenção do valor de L, expressado na Equação 2.7, utiliza um total de 24 constantes CON_i^{128} $(0 \leq i < 24)$ no lugar das *round keys*.

$$L = GFN_{4,12}(CON_{0:23}^{128}, K) \tag{2.7}$$

As *round keys* são obtidas a partir da execução do Algoritmo 2. Ao longo desta etapa são utilizadas 36 constantes CON_i^{128} $(24 \leq i < 60)$. Por fim, um total de 60 constantes pré-computadas são utilizadas (Sony Corporation, 2007a).

Algoritmo 2 Obtenção das *round keys* para K^{128}.

Input: K, L
  for $i = 0$ **to** 8 **do**
    $T = L \oplus (CON^{128}_{24+4i}|CON^{128}_{24+4i+1}|CON^{128}_{24+4i+2}|CON^{128}_{24+4i+3})$
    $L = \Sigma(L)$
    if i is odd **then** $T = T \oplus K$
    end if
    $RK_{4i}|RK_{4i+1}|RK_{4i+2}|RK_{4i+3} \leftarrow T$
  end for
Output: RK

2.4.2 Expansão de Chaves de 192 bits

Para uma chave de codificação de 192 bits (K^{192}), as quatro *whitening keys* de 32 bits já não são mais obtidas diretamente. Neste modo de operação, a chave de codificação K é inicialmente transformada em dois blocos de 128 bits K_L e K_R, com as *whitening keys* WK_i sendo obtidas a partir da operação lógica XOR entre K_L e K_R, como descrito na Equação 2.8.

$$\begin{aligned} K_L &\leftarrow K_0|K_1|K_2|K_3 \\ K_R &\leftarrow K_4|K_5|\overline{K_0}|\overline{K_1} \\ WK &= K_L \oplus K_R. \end{aligned} \tag{2.8}$$

Na expansão de chaves de 192 bits, a estrutura utilizada para obtenção dos blocos intermediários L_L e L_R é a $GFN_{8,10}$. Como descrito anteriormente, a entrada de uma $GFN_{8,n}$ corresponde à combinação de dois blocos de 128 bits ($K = K_L|K_R$). A saída representa a combinação dos blocos intermediários ($L = L_L|L_R$), como descrito na Equação 2.9.

$$L_L|L_R = GFN_{8,10}(CON^{192}_{0:40}, K_L|K_R) \tag{2.9}$$

Diferente da expansão de chaves de 128 bits, neste processo as quatro *round keys* são usadas a cada iteração. Para gerar as chaves a partir de K^{192} são necessárias 10 iterações. Dessa forma, 40 constantes (CON^{192}) de 32 bits são utilizadas no cálculo da GFN. O processo de obtenção das *round keys* é apresentado no Algoritmo 3.

2.4.3 Expansão de Chaves de 256 bits

Para uma chave de 256 bits, K_L and K_R podem ser obtidas de forma direta, como descrito na Equação 2.10.

$$\begin{aligned} K_L &\leftarrow K_0|K_1|K_2|K_3 \\ K_R &\leftarrow K_4|K_5|K_6|K_7 \\ WK &= K_L \oplus K_R. \end{aligned} \tag{2.10}$$

Apesar de apresentar o mesmo mecanismo, a expansão de chaves de 256 bits apresenta pequenas diferenças em relação à expansão de K^{192}. Além da nova forma de

obtenção dos valores de K_L e K_R, as constantes agora passam a ser denominadas CON^{256}. Além disso, o número de iterações do Algoritmo 3 passa de 11 para 13: ($i = 0$ *to* 12). O resultado corresponde a um conjunto de 52 *round keys* de 32 bits.

Algoritmo 3 Obtenção das *round keys* para K^{192}

Input: K, L
  for $i = 0$ **to** 10 **do**
    if i mod $4 = 0$ **ou** 1 **then**
      $T = L_L \oplus (CON^{192}_{40+4i}|CON^{192}_{40+4i+1}|CON^{192}_{40+4i+2}|CON^{192}_{40+4i+3})$
      $L_L = \Sigma(L_L)$
    else
      $T = L_R \oplus (CON^{192}_{40+4i}|CON^{192}_{40+4i+1}|CON^{192}_{40+4i+2}|CON^{192}_{40+4i+3})$
      $L_R = \Sigma(L_R)$
        if i is odd **then** $T = T \oplus K_L$
        end if
    end if
      $RK_{4i}|RK_{4i+1}|RK_{4i+2}|RK_{4i+3} \leftarrow T$
  end for
Output: RK

3 ESTADO DA ARTE

O CLEFIA foi oficialmente publicado em 2007 e em 2012 foi adotado como cifra de bloco compacta no padrão ISO/IEC 29192-2 (ISO/IEC 29192-2, 2012). Por ser relativamente novo, um conjunto restrito de trabalhos acadêmicos, propondo sua implementação em hardware, foi apresentado desde então: Sugawara et al. (2008) e Akishita e Hiwatari (2012) para ASIC; Kryjak e Gorgón (2009), Proença e Chaves (2011), Hanley e O'Neill (2012), Chaves (2013) e Resende e Chaves (2015) para FPGA.

Nesta dissertação, apenas os estudos referentes ao CLEFIA em tecnologias FPGA são considerados. Este capítulo apresenta uma breve introdução aos dispositivos FPGA e as técnicas mais comumente utilizadas em projetos de cifras de bloco, assim como um levantamento dos trabalhos acadêmicos relacionados de maior destaque.

3.1 TECNOLOGIA DE DISPOSITIVOS FPGA

FPGA (*Field-Programmable Gate Array*) é um dispositivo formado por um arranjo bidimensional de blocos lógicos (células lógicas) ou *Slices*, células de memória, eventuais estruturas especializadas e canais de roteamento programáveis. Diferente de um Circuito Integrado de Aplicação Específica (ASIC), a célula lógica de um dispositivo FPGA pode ser configurada de modo a executar uma função lógica arbitrária. Os canais de roteamento, dispostos na forma de uma matriz bidimensional, são utilizados para realizar a conexão entre os blocos. Cada célula é capaz de realizar qualquer função de lógica combinacional (AND, OR, XOR, etc.). Essa configuração permite desenvolver qualquer função a partir da associação de uma série de blocos lógicos, traduzida na forma de um circuito digital. Apesar de largamente utilizados como produtos intermediários, associados a protótipos, em projetos de ASIC, dispositivos FPGA vêm ocupando um espaço de destaque no desenvolvimento de plataformas de produto final.

No passado, dispositivos FPGA eram caracterizados por serem pouco eficientes em termos de ocupação do *chip* (uso de células lógicas), consumo de energia e desempenho (frequência de operação), quando comparados a soluções equivalentes em ASIC. Ao longo dos últimos anos, essa diferença vem sendo significativamente reduzida, graças aos avanços no processo de fabricação e modelo de projeto, trazendo uma série de melhorias em termos de consumo, velocidade, custo e opções de reconfiguração, tais como a reconfiguração parcial em tempo de execução. Ainda hoje, soluções ASIC, de um modo geral, apresentam melhores indicadores de eficiência. Entretanto, o alto nível de esforço associado a este tipo de projeto, além do reduzido *time-to-market*, têm elevado os índices de utilização de FPGAs, principalmente no mercado de sistemas embutidos. A escolha por estes dispositivos

leva em conta características como: desenvolvimento do projeto, desempenho, custo de produção e, mais recentemente, opções de reconfiguração.

Uma das principais características presentes em dispositivos FPGA está presente na forma com que as funções lógicas são implementadas. Blocos lógicos, que comportam as funções lógicas, são projetados como *Lookup Tables* (LUTs), implementadas sobre memórias de tecnologia SRAM, ou a partir de um conjunto multiplexador e memória. De modo geral, um bloco lógico é composto por uma ou mais LUTs e um conjunto de registradores que podem armazenar a saída de cada LUT. Esta composição de cada bloco lógico e as características de funcionamento das LUTs variam de acordo com o fabricante ou mesmo entre as famílias de dispositivos.

Este trabalho baseia-se na família de dispositivos Xilinx Virtex, haja visto a compatibilidade no que se refere à comparação dos resultados com os trabalhos relacionados. Entretanto, outros dispositivos compatíveis com esse tipo de aplicação coexistem no mercado, como a família Stratix da ALTERA ou a ProASIC da Microsemi.

Dispositivos FPGA Xilinx Virtex 5 são compostos a partir de um conjunto de Blocos Lógicos Configuráveis (CLBs), conectados à uma matriz de roteamento. Cada CLB contém dois *Slices* dispostos em colunas e possuem cadeias de *carry* independentes, como descrito na Figura 6.

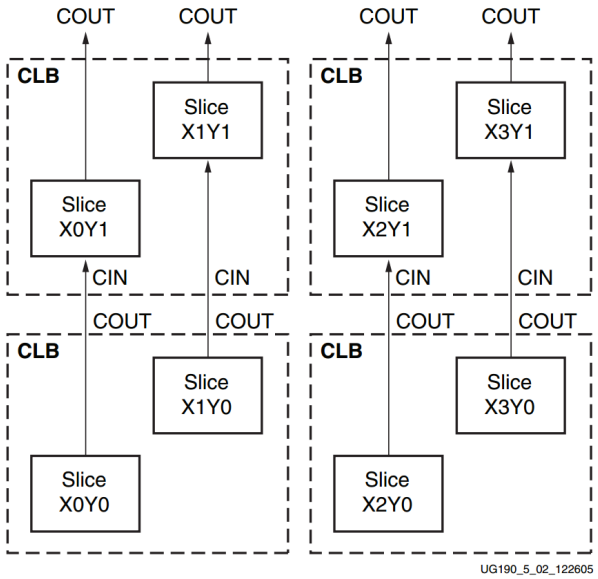


Figura 6 – Distribuição dos CLBs, apresentado no Guia do Usuário da Xilinx Virtex 5 (Xilinx, 2012).

Cada *Slice* contém quatro LUTs configuráveis, quatro registradores (*flip-flops*, FF) e uma chave seletora não controlável pelo programador. Uma LUT representa o elemento lógico básico do dispositivo FPGA, funcionando como uma *lookup table* capaz de reproduzir uma operação lógica.

Dispositivos FPGA modernos podem apresentar mais de um tipo de *Slice* em sua composição. Em maior número, o SLICEL é uma estrutura simples que se comporta de acordo com as características de configuração descritas até agora. Apresentadas em menor número, o SLICEM oferece um conjunto de opções de configuração adicionais ao SLICEL, dentre as quais: (i) configuração das LUTs na forma de uma RAM/ROM distribuída; ou (ii) como um registrador de deslocamento endereçável de profundidade 16 ou 32 bits (SRL16 ou SRL32). Esta última, em particular, apresenta um papel importante nas otimizações propostas no presente trabalho.

Por fim, um componente dedicado, largamente utilizado em dispositivos FPGA, é o bloco de memória RAM/ROM *dual-port* (BRAM) configurável. Este bloco de memória é uma das principais funcionalidades presentes atualmente nas tecnologias FPGA, uma vez que não há solução semelhante em ASIC. Dispositivos FPGA modernos são capazes de armazenar até 36 kb de dados em cada BRAM. Esta estrutura opera apenas no modo de escrita síncrono, contudo, a leitura pode ser configurada no modo síncrono ou assíncrono. Adicionalmente, o tamanho dos barramentos de entrada/saída e a profundidade da memória também podem ser configurados.

Aspectos detalhados e esquemáticos dos elementos SLICEL, SLICEM e BRAM podem ser encontrados no Anexo A, na forma como eles são apresentados no Guia do Usuário do Xilinx Virtex 5.

3.2 ALGORITMOS DE CRIPTOGRAFIA EM FPGA

Dois dos aspectos principais, considerados no projeto de cifras de bloco em hardware, são área ocupada e desempenho, determinados a partir da frequência de operação e a capacidade de processamento (*throughput*) do circuito. Na prática, os dois fatores tendem a apresentar efeitos análogos. Apesar de arquiteturas compactas ocuparem menos área, em geral, apresentam baixo desempenho. Por outro lado, arquiteturas paralelas apresentam alta capacidade de processamento, ao custo de uma estrutura formada a partir de módulos que operam simultaneamente, implicando em uma maior ocupação de área.

No projeto de coprocessadores para cifras de bloco iterativas, os avanços estão relacionados, principalmente, a técnicas de concentração de iterações em uma mesma estrutura (*round folding*), na forma de estruturas recursivas (*rolled*) ou expandidas (*unrolled*). Acrescenta-se a isso a utilização de componentes de hardware de lógica dedicada e o escalonamento de operações (GOOD; BENAISSA, 2005; RODRIGUEZ-HENRIQUEZ et al., 2007; PROENÇA; CHAVES, 2011). Por fim, simplificações matemáticas, fusão de dados e remoção de duplicidades permitem reduzir o caminho crítico associado a operações aritméticas, por meio de mecanismos de mapeamento direto, usando funções de *lookup* (SUGAWARA et al., 2008; PROENÇA; CHAVES, 2011). Particularmente em dispositivos FPGA, operações de busca têm sido largamente utilizadas, devido à presença da BRAM,

ou mesmo da configuração de LUTs para operarem como RAM/ROM distribuída.

Uma das formas mais diretas de se obter um comparativo entre área e *throughput*, em implementações de cifras de bloco, é a partir da concentração/expansão de iterações. Em estruturas expandidas, o cálculo de cada iteração é desmembrado na forma de uma arquitetura em *pipeline*. Com isso, várias iterações podem ser executadas de forma independente e em paralelo. Em geral, esta abordagem demanda mais área em *chip* mas, por outro lado, é capaz de operar em frequências maiores e, consequentemente, alcançar elevadas taxas de transferência, particularmente quando o algoritmo opera no modo ECB (*Electronic Code Book*). Entretanto, quando há uma dependência entre os blocos, como no modo CBC (*Cipher Block Chaining*), a característica de operação sob altas taxas de transferência não é alcançada. Em estruturas recursivas, cada iteração é realizada em um ou mais ciclos de clock, resultando em estruturas menores. Consequentemente, neste tipo de arquitetura, o *throughput* é tipicamente baixo.

Outra diferenciação utilizada nos trabalhos acadêmicos relacionados refere-se à operação de substituição. De forma geral, elas podem variar entre implementações mais detalhistas, baseadas em funções lógicas, às mais simples, usando *lookup tables* equivalentes, baseadas em memória. Em projetos que levam em conta apenas funções lógicas, as operações de substituição e multiplicação matricial são implementadas diretamente de suas expressões matemáticas, através do uso de componentes lógicos. Estas implementações são mais compactas, mas geralmente levam mais tempo para serem completadas. Em FPGA, a técnica que tem demonstrado maior eficiência na implementação das operações de substituição é o uso de *lookup tables* (RESENDE; CHAVES, 2015). Esta técnica visa transformar a operação de substituição em *lookup tables* equivalentes, implementadas em LUT ou BRAM. Adicionalmente, operações subsequentes, como a multiplicação por matrizes de difusão do CLEFIA, também podem ser computadas e armazenadas em uma estrutura chamada *T-box*. Arquiteturas baseadas no uso de memória podem resultar em circuitos mais rápidos, ao custo de blocos de memória embutidos em dispositivos FPGA.

3.3 ESTADO DA ARTE DO CLEFIA EM FPGA

Arquiteturas em hardware para a realização do CLEFIA, voltadas para ASIC ou FPGA, vêm sendo propostas e analisadas por uma série de trabalhos acadêmicos. O principal objetivo destes trabalhos é o de atender a requisitos de alto desempenho de execução ou o de apresentar modelos compactos, caracterizados a partir do baixo consumo de blocos/células lógicas.

A seguir são apresentadas as principais arquiteturas voltadas para dispositivos FPGA presentes na literatura acadêmica.

3.3.1 Pipeline Desdobrado de Kryjak e Gorgón (2009)

Uma das primeiras implementações do CLEFIA em FPGA foi proposta por Kryjak e Gorgón (2009), com ênfase nos dispositivos Xilinx das famílias Virtex 2-Pro, Virtex 4 e 5. Neste trabalho, os autores propõem uma estrutura de cifra expandida, na medida em que consideram quatro alternativas de implementação das funções S: (i) a partir da definição; (ii) S-*box*; (iii) T-*box*; e (iv) implementação mista. Esta última abordagem implementa a função S_0 a partir da definição, enquanto a S_1 é implementada com base em *lookup table* (S-*box*), devido a complexidade associada à transformação $GF(2^8)$ e consequente demanda por recursos lógicos adicionais. As quatro técnicas apresentadas utilizam configuração baseada em LUT, sem recorrer ao uso de BRAM.

Quanto submetidas aos testes na Virtex 2-Pro, Kryjak e Gorgón (2009) mostram que nenhuma implementação da função S prevalece sobre outra. Por outro lado, na mais recente família de dispositivos Virtex 5, os resultados apresentados indicam que a abordagem mista é mais eficiente em termos de recursos e desempenho, seguida da abordagem baseada em T-*box*.

Além disso, o trabalho apresenta duas configurações para o projeto do mecanismo de expansão de chaves de 128, 192 e 256 bits. A primeira, capaz de produzir as *round keys* e *whitening key* sob uma estrutura iterativa, armazenando as chaves em *flip-flops*. A segunda versão é paralelizada e não utiliza elementos de armazenamento, produzindo a sequência de chaves necessárias a cada iteração da cifra.

3.3.2 Arquitetura Compacta de Proença e Chaves (2011)

O primeiro trabalho focado em arquiteturas compactas para o CLEFIA em FPGA foi proposto por Proença e Chaves (2011). Duas arquiteturas recursivas, baseadas no mapeamento das T-*boxes* em BRAMs para o CLEFIA são apresentadas: a primeira baseada em um *data path* de 128 bits; a segunda, em um *data path* de 32 bits. As duas estruturas foram analisadas sob a perspectiva das famílias de dispositivos Xilinx Spartan 3, Virtex 4 e Virtex 5.

A arquitetura *Type-I* estabelece uma relação mais próxima com a sequência de iterações do CLEFIA, ao computar as funções F_0 e F_1 em paralelo. A principal otimização desta arquitetura está na implementação de T-*boxes* no lugar das funções S. Cada ramo das funções F contém um elemento de memória (BRAM) que armazena os 256 blocos de 32 bits (8 kbits). Nesta abordagem, quatro BRAMs de 18 kb ou 36 kb são utilizadas.

A arquitetura *Type-II* (Figura 7) merece maior destaque, uma vez que explora a simetria entre F_0 e F_1, de modo a integrar as duas T-*boxes* em um mesmo componente, reunindo o cálculo das funções S em uma única *lookup table*. Esta estrutura, por sua vez, é submetida à um controle de escalonamento em *pipeline*, no qual, a cada ciclo de clock,

apenas uma função F é computada. Isso implica que dois ciclos de clock são necessários para computar uma iteração do CLEFIA (F_0 e F_1). Sendo assim, uma única estrutura compartilhada pode ser utilizada para calcular as funções F_0 e F_1, mantendo o mesmo desempenho através do *forwarding* dos blocos de dados.

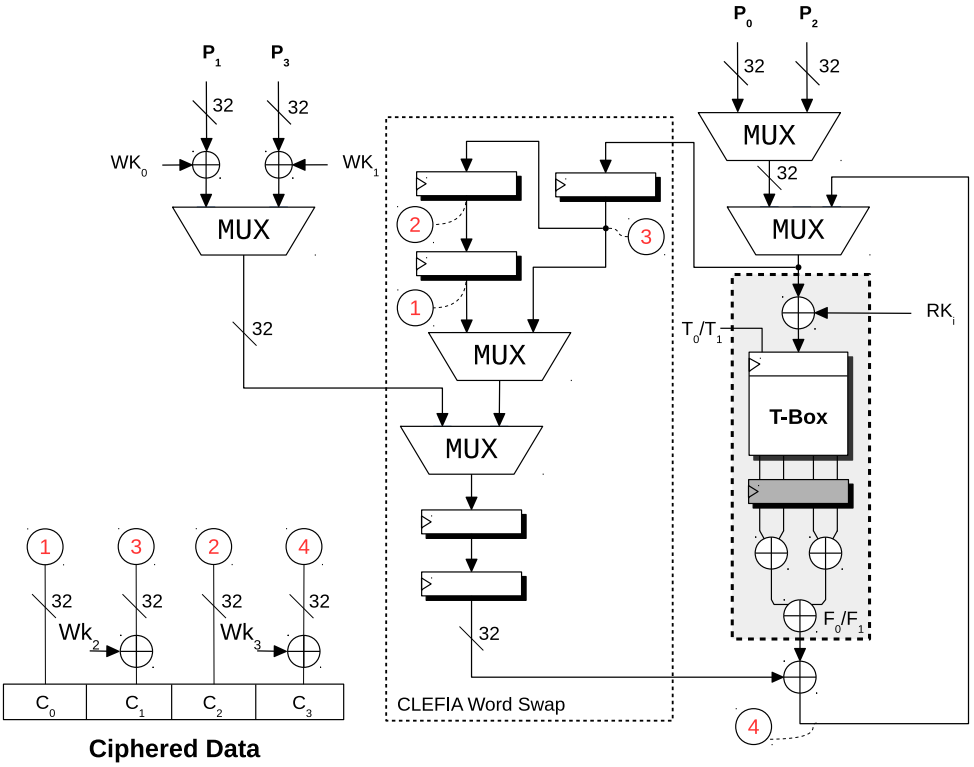


Figura 7 – Arquitetura proposta por Proença e Chaves (2011).

Na estrutura apresentada na Figura 7, a primeira e terceira palavras de cada iteração alternam-se na alimentação da entrada da T-*box*. Enquanto isso, uma cadeia de registradores, devidamente escalonados, é responsável por garantir o funcionamento da permutação presente no final de cada iteração. Para referência futura, nomearemos este circuito, daqui por diante, como "*CLEFIA Word Swap*".

Por fim, nenhuma estrutura de expansão de chaves é proposta em tal trabalho. Dessa forma, uma BRAM excedente é considerada para armazenar as *round keys* computadas previamente.

3.3.3 Arquiteturas Iterativa e Serial de Hanley e O'Neill (2012)

Um estudo comparativo entre duas novas cifras de bloco adotadas no padrão ISO/IEC 29192-2 (ISO/IEC 29192-2, 2012) (CLEFIA e PRESENT) é apresentado por Hanley e O'Neill (2012). Em seu trabalho, duas estruturas de cifra recursiva para o CLEFIA são analisadas, em dispostivos FPGA da família Virtex 2 e Virtex 5 da Xilinx, sem o uso de recursos adicionais embutidos, como BRAM.

As duas estruturas apresentadas pelos autores são compostas de barramentos de entrada de 8 bits, processados sob um caminho de dados de tamanho variado. Além disso, as duas estruturas foram implementadas a partir de LUTs e registradores. Destaca-se, ainda, que o estudo restringe-se à cifra e expansão de chave para chaves de codificação de 128 bits, de modo a compatibilizar as análises em função da cifra PRESENT.

A arquitetura iterativa propõe a implementação das *S-boxes* em LUTs configuradas na forma de ROMs distribuídas, com leitura assíncrona. As multiplicações, presentes no mecanismo de difusão, por sua vez, são implementadas de forma direta, usando operações lógicas XOR como descrito em (Sony Corporation, 2007a). O cálculo das duas funções F e a *CLEFIA Word Swap* são realizadas em um mesmo ciclo de clock. Na expansão de chaves, as constantes utilizadas são obtidas em tempo de execução, a partir da proposta sugerida em Sugawara et al. (2008). A chave intermediária L é computada e armazenada em um registrador de 128 bits, da mesma forma que a chave de codificação usada no processo de obtenção das *round keys*.

Na arquitetura serial, a diferença reside na execução da função F, onde um byte é calculado a cada ciclo de clock, implicando no aumento da latência e redução do *throughput*. Além disso, a estrutura opera sob um caminho de dados de 8 bits, com a região de expansão de chaves baseada em uma estrutura de 128 bits.

3.3.4 Expansão de Chaves de 128 bits de Chaves (2013)

O primeiro registro de destaque com ênfase em arquiteturas compactas para expansão de chave do CLEFIA foi apresentado por Chaves (2013). O trabalho apresenta uma arquitetura de 32 bits para cálculo das *round keys* e *whitening key*, a partir de chaves de codificação de 128 bits.

Nenhum hardware adicional é necessário para o cômputo da *whitening key*, sendo os blocos de 32 bits obtidos a partir da chave de cifra. A estrutura $GFN_{4,12}$ proposta por Proença e Chaves (2011) é utilizada aqui para o cômputo da chave intermediária L. A chave intermediária L, por sua vez, é conduzida através de dois barramentos de 32 bits, destacados em (2) e (3), nas Figuras 7 e 8. Nesta estrutura, um bloco de memória compartilhado (BRAM) de 36 kb é utilizado para armazenar as 24 *round keys* e as 60 constantes de 32 bits.

A referida arquitetura apresenta um caminho de dados misto de 32 e 128 bits. Apenas uma *round key* é calculada a cada ciclo de clock, através de um caminho de dados de 32 bits. Uma vez que a função Σ é realizada em um único ciclo de clock, um caminho de 128 bits é necessário para a realizar as permutações sobre L. Os blocos de dados da chave intermediária L, permutada de 128 bits, são selecionados a partir de um multiplexador 4×32 bits. Desse modo, a função Σ é realizada a cada quatro ciclos de clock.

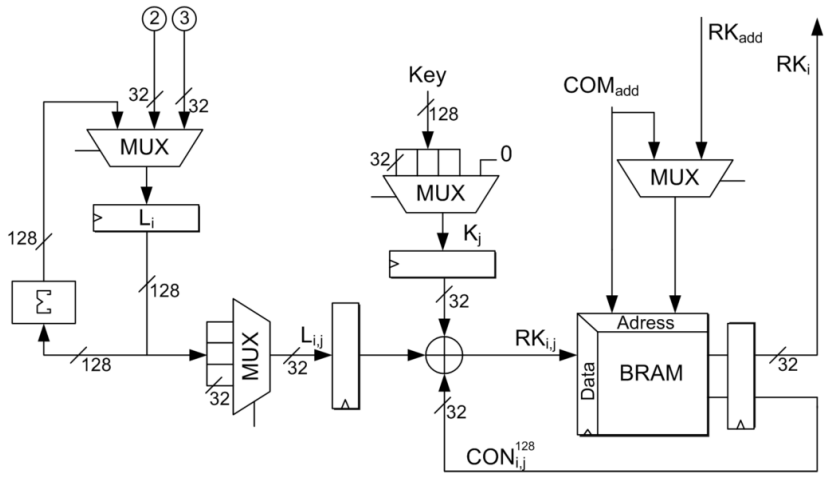


Figura 8 – Arquitetura de expansão de chaves de 128 bits (CHAVES, 2013).

3.3.5 Arquitetura de Cifra Dupla de Resende e Chaves (2015)

Mais recentemente, Resende e Chaves (2015) apresentaram um circuito compacto multi-criptográfico, considerando as cifras AES e CLEFIA. Neste trabalho, os autores sugerem otimizações a partir do uso de *T-box* para a realização da função de substituição. A arquitetura proposta é baseada em um caminho de dados de 32 bits de largura, em *pipeline* recursivo, como mostra a Figura 9.

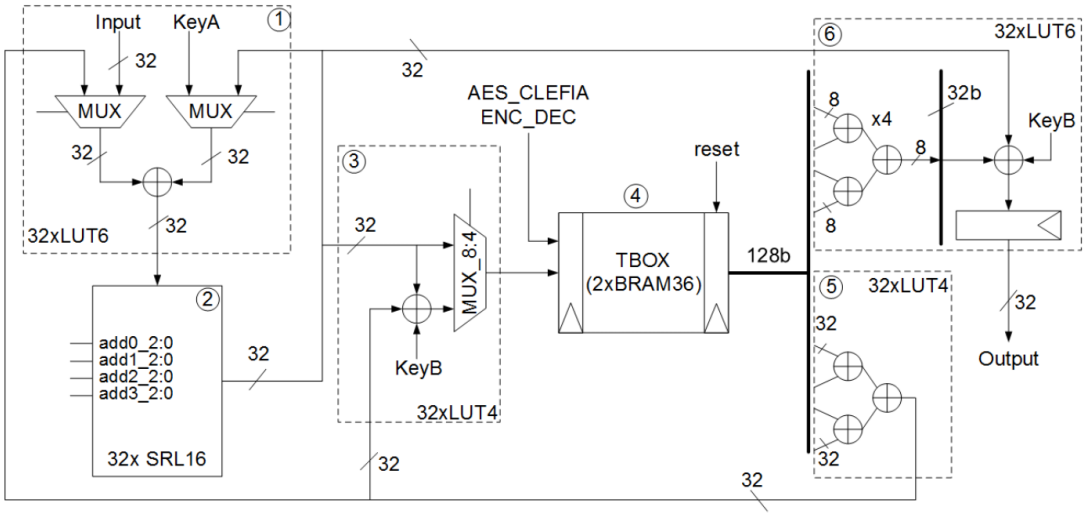


Figura 9 – Arquitetura de cifra dupla (RESENDE; CHAVES, 2015).

Em seu trabalho, os autores sugerem a fusão das operações de deslocamento do AES e o *CLEFIA Word Swap*, a partir de registradores de deslocamento multi-endereçáveis de 16 bits, implementados usando o modo LUT SRL16 dos dispositivos Xilinx. Cada LUT opera como um registrador de deslocamento endereçável de 16 bits de profundidade X 1 bit de largura, permitindo combinar 4 grupos de 8 LUTs. Nesta estrutura, uma iteração do CLEFIA é realizada em dois ciclos de clock. Por fim, tal estudo não apresenta solução

para expansão de chave e considera que as *round keys* são pré-computadas e armazenadas em uma BRAM interna. Dessa forma, um total de três BRAMs de 36 kb é requerido para sua implementação.

Apesar de residir em um contexto de estruturas de cifra múltipla, o referido trabalho introduz um conjunto de otimizações que influenciaram o projeto da arquitetura descrita no presente trabalho.

3.4 CONCLUSÕES SOBRE O ESTADO DA ARTE

As contribuições mais recentes sobre implementações do CLEFIA em FPGA apresentam uma visão das abordagens mais comumente utilizadas. Estruturas baseadas no desdobramento de iterações (*unrolled round*) são mais indicadas para cifras mais rápidas, limitando-se a estruturas sem realimentação, nas quais os blocos de entrada operam de modo independente. Por outro lado, estruturas de iteração recursiva (*rolled round*) apresentam arquiteturas compactas em cifras baseadas em cadeias de realimentação. É válido destacar ainda que esse tipo de implementação requer ajustes cuidadosos, para atender a requisitos de alto desempenho.

No que se refere ao projeto de cifras de bloco, dois métodos se destacam na implementação das transformações não lineares: (i) lógica *hard-wired*; e (ii) *lookup tables* endereçáveis, tais como *S-box* e *T-box*. Em dispositivos FPGA, é comum o uso de blocos de memória *on-chip* para armazenamento de tabelas. Além disso, o uso de registradores de deslocamento endereçáveis tem demonstrado eficácia na implementação da *CLEFIA Word Swap*.

Pouco explorada nos trabalhos relacionados à implementação do CLEFIA em hardware, a expansão de chave pode ser implementada de três formas: (i) pré-computada em software e armazenada em uma estrutura de armazenamento (LUT ou BRAM); (ii) computada em tempo de execução; e (iii) pré-computada em hardware e armazenada em LUT ou BRAM. Mais comumente utilizado, o cômputo prévio das *round keys* em software conduz à arquiteturas em hardware compactas a operar sobre altas taxas de transferência, face à ausência de um mecanismo de expansão de chave dedicado. A coexistência dos mecanismos de expansão de chave demandam por mais área em *chip*, principalmente quando consideradas chaves de 192 e 256 bits, exigindo um esforço de projeto maior para atingir o mesmo desempenho.

Os capítulos seguintes são dedicados à descrição do projeto do bloco de cifra CLEFIA completo, levando em consideração as técnicas apresentadas até aqui.

4 PROJETO DE ARQUITETURA DO CLEFIA

O principal objetivo deste trabalho é projetar uma estrutura compacta capaz de processar a cifra de bloco do CLEFIA e a expansão de chave completa. Neste sentido, o primeiro passo desta pesquisa consistiu em estabelecer um esboço funcional dos dois mecanismos, a partir da combinação de técnicas presentes em arquiteturas de blocos de cifra compactas nas publicações científicas observadas.

Como descrito no Capítulo 2, o CLEFIA pode ser dividido em duas etapas de processamento: (i) a expansão da chave de codificação e obtenção das *round keys*; e (ii) a codificação/decodificação de uma cadeia de caracteres. No processo de expansão, uma chave de codificação de 128, 192 ou 256 bits é inicialmente submetida à uma rede de Feistel. O resultado desta operação, representada por uma chave intermediária L, é então processado, de forma iterativa, através da função $\Sigma(L)$ e o resultado adicionado à um conjunto de constantes pré-definidas e, de forma alternada, também à chave de codificação.

Na Seção 2.4, foi definido que, na expansão de chaves de 128 bits, a obtenção da chave intermediária L pode ser realizada sobre a mesma estrutura $GFN_{4,n}$ usada no processo de cifra. Por outro lado, a expansão de chaves de 192 e 256 bits requer uma estrutura $GFN_{8,n}$. Uma vez que o processamento sob uma estrutura GFN é o primeiro passo no sentido de proporcionar um mecanismo de expansão de chave completo para o CLEFIA, este trabalho começa por descrever o esboço da arquitetura de cifra. A estrutura GFN, apresentada a seguir, baseia-se no projeto de cifras de bloco por iteração recursiva (*rolled round*), em particular a estrutura proposta por Proença e Chaves (2011). Esta estrutura considera uma implementação em *pipeline* baseada em T-*box* a partir de um *data path* de 32 bits. A adoção desta solução é baseada na hipótese de que projetos que utilizam esta estratégia têm resultado em estruturas compactas e eficientes, particularmente em pesquisas direcionadas a dispositivos FPGA (PROENÇA; CHAVES, 2011; RESENDE; CHAVES, 2015). Finalmente, escolha da estrutura proposta por Proença e Chaves (2011) como ponto de partida para esta pesquisa levou em consideração a simplicidade de adequação da estrutura face aos novos requisitos do projeto.

Este capítulo descreve a proposta de estrutura para o cálculo da cifra através de um circuito híbrido $GFN_{4/8,n}$ e apresenta a descrição do módulo de expansão de chave com suporte a todos os tamanhos de chave do CLEFIA. Adicionalmente, são discutidas otimizações em nível de arquitetura, considerando a família de dispositivos FPGA Xilinx Virtex.

4.1 ESTRUTURA *GFN* PARA CIFRA DO CLEFIA

A estrutura de cifra proposta por Proença e Chaves (2011) baseia-se na implementação da permutação, nomeadamente *CLEFIA Word Swap*, necessária para a realização da rede de Feistel, a partir de uma cadeia de registradores multiplexada. Esta estrutura é reponsável por armazenar os valores intermediários e selecioná-los à medida em que são necessários para o processamento. Uma estrutura semelhante para rede de Feistel 8-*branch* sugere a presença de um caminho de dados maior, devido à necessidade de armazenar e multiplexar valores intermediários adicionais. O armazenamento e multiplexação podem ser conduzidos através de registradores extras e multiplexadores maiores, resultando na estrutura ilustrada na Figura 10. As modificações, em relação à estrutura $GFN_{4,n}$ originalmente proposta por Proença e Chaves (2011) e reproduzidas na Figura 7, são destacadas em cinza.

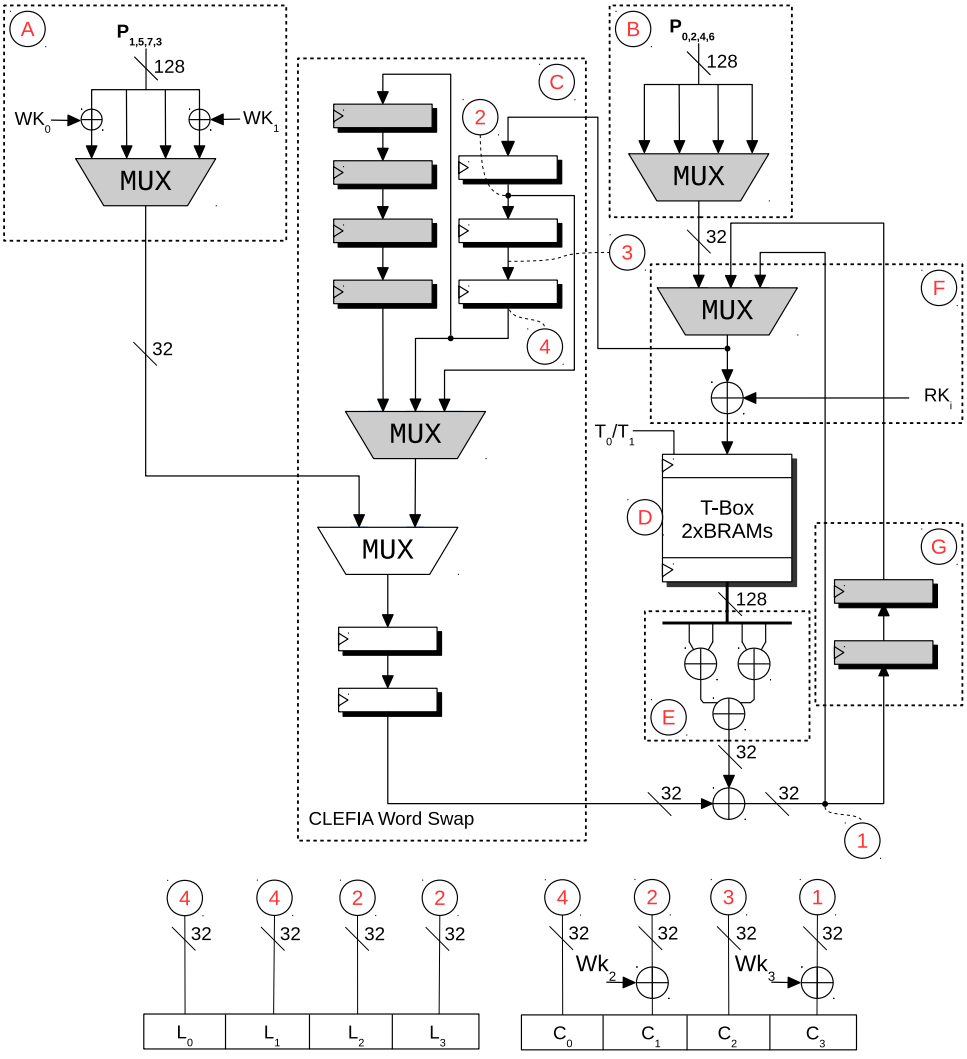


Figura 10 – Proposta da estrutura híbrida $GFN_{4,n}/GFN_{8,n}$.

Por considerar blocos de entrada de 256 bits, duas novas entradas foram adicionadas aos multiplexadores de entrada presentes nos blocos (A) e (B). Além disso, um novo

caminho de dados foi adicionado a partir da inserção dos quatro registradores destacados no bloco (C), implicando no acréscimo de uma nova entrada ao multiplexador de saída do bloco. Esta estrutura é utilizada para garantir a propagação dos valores oriundos da saída das funções F face ao processo de permutação do CLEFIA. Por fim, os dois registradores presentes no bloco (G) foram empregados, implicando na presença de uma nova entrada no multiplexador de entrada do bloco (F). Esta estrutura garante o escalonamento adequado da estrutura quando operando na forma de uma estrutura $GFN_{8,n}$.

Os dados de entrada (cadeia de caracteres simples ou chave de codificação) são fornecidos ao circuito através dos multiplexadores localizados nos estágios (A) e (B) da Figura 10, a partir da seleção adequadamente escalonada de blocos de 32 bits. Nesta estrutura são aceitos dados de 128 e 256 bits. No primeiro caso, os blocos P_i $(3 < i \leq 7)$ são ignorados. Adicionalmente, no estágio (A), as *whitening keys* são adicionadas aos blocos P_1 e P_3 da cadeia de caracteres de entrada. Na expansão de chave do CLEFIA esta operação XOR não é realizada. Dessa forma, incialmente, nós assumiremos que o sistema é capaz de zerar a entrada da *whitening key* externamente.

O mecanismo de substituição e difusão é apresentado no estágio (D). As demais multiplicações e as somas sobre $GF(2^8)$ são realizadas a partir de uma árvore de XOR 4-para-1 de 32 bits, presente no estágio (E). No estágio (F), o bloco de 32 bits resultante é realimentado e somado à sua respectiva *round key*. O estágio final consiste em adicionar as duas últimas *whitening keys* aos blocos de saída, obtidos diretamente dos nós enumerados na Figura 10. Os registradores, destacados nos estágios (C) e (G), são utilizados apenas no processamento da $GFN_{8,n}$, uma vez que o escalonamento recursivo da *CLEFIA Word Swap* requer quatro ciclos de clock para ser completado.

Uma vez que o cálculo original ainda pode ser realizado, a partir de ajustes no escalonamento dos multiplexadores dos estágios (C) e (F), esta estrutura é capaz de realizar as redes de Feistel $GFN_{4,n}$ e $GFN_{8,n}$. Quando configurado para cifra ou expansão de chave de 128 bits (modo 4-*branch*), a estrutura comporta-se como uma $GFN_{4,n}$, usando dois ciclos de clock por iteração. Por outro lado, quando configurado para expansão de chaves de 192 ou 256 bits (modo 8-*branch*), a estrutura processa dois pares de funções F, implicando na necessidade de quatro ciclos de clock para a realização de uma iteração.

4.1.1 Mapeamento das *T-Boxes* em BRAMs

A composição do mecanismo de difusão do CLEFIA consiste de operações baseadas em *S-boxes* seguidas de uma multiplicação matricial. Como descrito na Seção 2.2, em uma implementação baseada no uso de memórias, estas operações podem ser parcialmente condensadas em uma *T-box*. Com o uso das *T-boxes* é possível dividir a estrutura do DSM em uma tabela baseada em entradas de 8 bits e saída de 32 bits. Em dispositivos FPGA, a implementação desse tipo de estrutura pode ser realizada de forma eficiente

com o uso de BRAMs. Esse tipo de estratégia tem se mostrado eficiente, uma vez que conduz à obtenção de circuitos com alta capacidade de processamento e baixa demanda por LUTs (RODRIGUEZ-HENRIQUEZ et al., 2007; PROENÇA; CHAVES, 2011; RESENDE; CHAVES, 2015). A análise da Equação 2.4, apresentada na Seção 2.2, permite verificar uma série de similaridades entre as *T-boxes*.

O estudo apresentado por Sugawara et al. (2008) otimiza o uso das *T-boxes*, explorando as semelhanças entre as operações. Por exemplo, as tabelas T_{00} e T_{02} realizam a mesma operação, diferindo apenas em um deslocamento de 16 bits da saída. O mesmo acontece com os pares T_{01}/T_{03}, T_{10}/T_{12} e T_{11}/T_{13}. Devido à presença de duas *S-boxes* (S_0 e S_1) e duas diferentes matrizes de difusão (M_0 e M_1), uma memória capaz de comportar as *T-boxes* do CLEFIA deverá possuir no mínimo:

$$4_{T-boxes} \times 256_{combinações} \times 32_{bits} = 32kb \tag{4.1}$$

Devido à presença de estruturas de memória do tipo *dual-port* na maioria dos dispositivos FPGA, duas buscas podem ser realizadas em uma memória simultaneamente. Considerando FPGAs Xilinx Virtex 5 como tecnologia alvo deste estudo, blocos de memória (BRAMs) de 36 kb, embutidos no *chip*, são utilizados na implementação das duas *T-boxes*, em destaque no estágio (D), na Figura 10. Uma vez que a arquitetura proposta é formada a partir de um caminho de dados de 32 bits, duas memórias *dual-port* são necessárias. Um registrador embutido na saída das BRAMs é utilizado para controlar o fluxo do processo em *pipeline* e reduzir o caminho crítico. Desse modo, uma iteração do CLEFIA é executada em dois ou quatro ciclos de clock, dependendo do modo de operação.

4.1.2 Otimizações da Arquitetura

A principal otimização considerada neste trabalho, no sentido de minimizar os requisitos de área da estrutura $GFN_{8,n}$ proposta, está relacionada ao mecanismo de permutação *CLEFIA Word Swap*. A implementação da estrutura apresentada na Figura 10 implica no aumento da quantidade de registradores, além da extensão do tamanho dos multiplexadores. O reflexo disso é traduzido no uso excessivo de LUTs e FFs do dispositivo. Assumindo o uso de FPGAs da família Virtex 5 da Xilinx, compostas por LUTs de seis entradas, o bloco (A) pode ser implementado com, no mínimo, 64 LUTs. Além disso, o aumento do número de registradores, quando não proporcional ao acréscimo de LUTs, pode resultar na presença de *Slices* que ocupam apenas seus registradores, aumentando o índice de estruturas com LUTs inutilizadas.

Ao considerar tecnologias presentes em dispositivos FPGA modernos, estes registradores simples podem ser substituídos por registradores de deslocamento. Se estes registradores de deslocamento puderem ser endereçados, de modo a produzir na saída valores internos específicos, então a operação de permutação pode ser realizada a partir do

controle adequado destes endereços. Nos dispositivos FPGA da Xilinx, este registrador de deslocamento endereçável pode ser mapeado dentro de Lookup Tables (LUTs) operando nos modos SRL16 ou SRL32. Como descrito na Seção 3.1, cada LUT pode ser configurada na forma de um registrador de deslocamento endereçável de 1 bit, capaz de armazenar 16 ou 32 bits. Dessa forma, o valor completo a ser armazenado e a operação de troca de palavras podem ser implementados usando 32 LUTs, como descrito no estágio (2) da Figura 11. Um registrador adicional foi acrescentado fora do registrador do deslocamento, tendo em vista reduzir o caminho crítico.

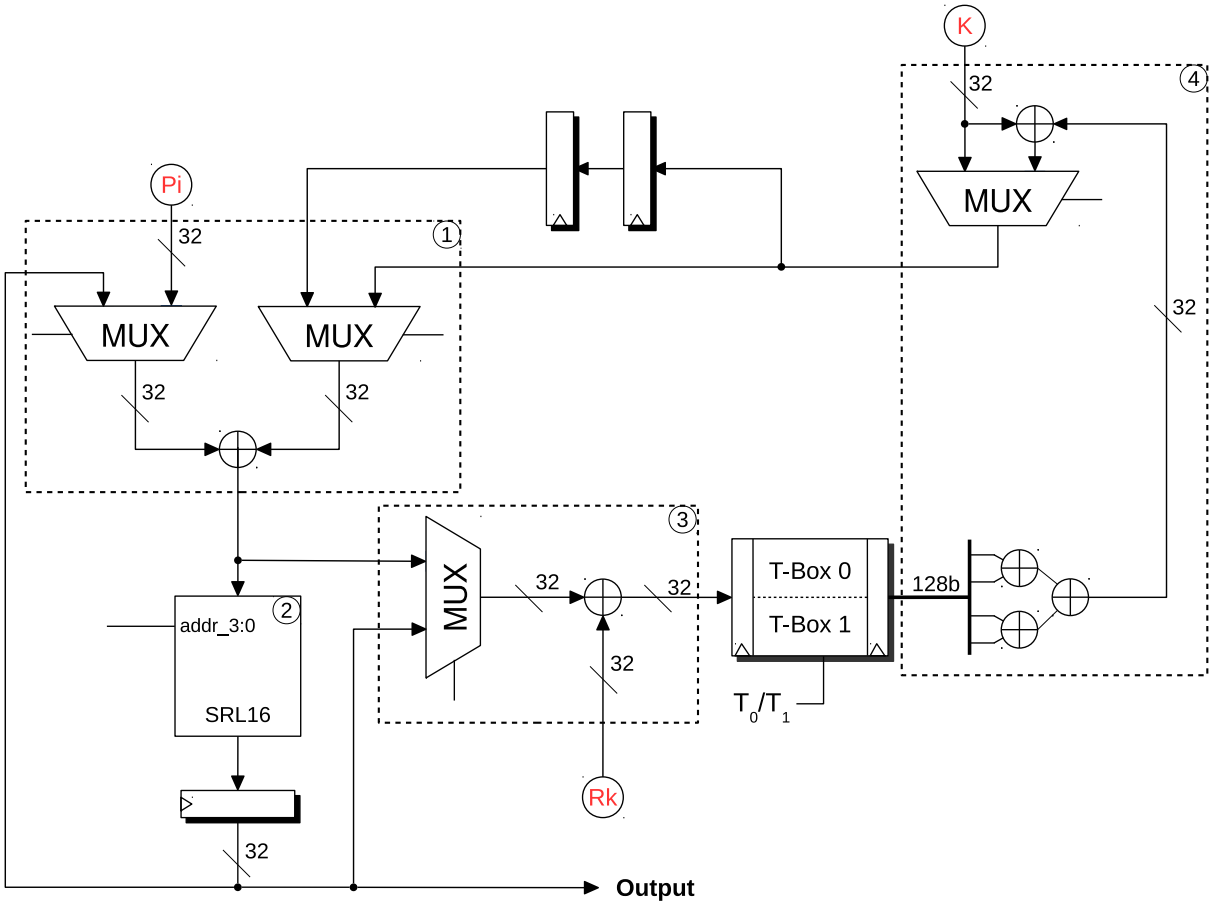


Figura 11 – Proposta de arquitetura para a estrutura 8/4-branch do CLEFIA.

A entrada de dados na estrutura também pode ser otimizada, levando em conta que barramentos externos em aplicações de sistemas embutidos são, normalmente, menores que 128 bits. Aqui nós consideramos que o dado é introduzido na estrutura usando um barramento de entrada (P_i), composto por 32 bits. Tal característica permite simplificar os multiplexadores de entrada, como descrito no estágio (1) da Figura 11. Note que a entrada do circuito considera dois barramentos de entrada, representados por (P_i) no estágio (1) e (K) no bloco (4). Este último, por sua vez, é utilizado para introdução da chave de cifra, no processo de obtenção da chave intermediária L e das *whitening keys*,

quando configurada para cifra. Em ambos os casos, inicialmente assume-se ser possível zerar as entradas externamente, durante o fluxo de processamento.

Ao final de cada ciclo de realização da função F dentro de uma iteração, o resultado da árvore de XOR 4-para-1 é imediatamente realimentado através do multiplexador localizado no estágio (4). Dessa forma, a estrutura garante que, ao longo do processamento, uma iteração estará sempre sendo executada.

A última operação realizada no algoritmo do CLEFIA é a XOR entre as duas *whitening keys* na iteração final, quando a permutação já não é mais necessária. O resultado desta operação é novamente introduzido no registrador de deslocamento. Dessa forma, os blocos resultantes podem ser extraídos, em qualquer ordem, a partir do registrador de deslocamento, diretamente para a saída do circuito, sem a necessidade de um estágio de saída.

Uma vez que o registrador de deslocamento passa a ser responsável por realizar a permutação e fornecer os blocos na saída na ordem correta, um novo bloco de dados não pode ser fornecido ao mesmo tempo em que o último é concluído. Isso implica na necessidade de 2 ciclos adicionais, para concluir o processamento sem produzir conflitos de dados. Os detalhes de implementação são discutidos na Seção 5.3.

Embora a descrição funcional apresentada até o momento opere adequadamente para o processo de cifra, a geração das chaves L de 128 e 256 bits da expansão de chave deve ser tratada com cautela. Durante o processo de expansão de uma chave de cifra de 128, 192 ou 256 bits, a chave intermediária L, descrita na Seção 2.4, é coletada diretamente da saída do registrador de deslocamento, sendo transferida para uma estrutura de expansão de chave diferente, descrita a seguir. No caso das chaves de 192 e 256 bits, 12 ciclos de clock adicionais são necessários após a execução de cada iteração. Essa latência extra permite que os blocos de dados de L sejam enviados em ordem, diretamente para o módulo de expansão. Sabendo que o processo de cifra só pode começar uma vez que as *round keys* tenham sido computadas, estes ciclos adicionais, necessários para a alimentação da chave L no bloco de expansão de chave, não produzem impacto geral no desempenho da cifra.

Assim como na estrutura apresentada na Figura 10, no modo 8-*branch*, o dado realimentado é proveniente da saída dos registradores adicionais, localizados na parte superior da Figura 11.

4.1.2.1 Redução do Caminho Crítico

O caminho crítico de um circuito pode ser definido como o caminho mais longo, em termos de temporização, dentro de um único estágio funcional. Em dispositivos FPGA, este caminho é determinado a partir dos estágios em nível de estruturas de sequenciamento de LUTs em um único estágio, delimitado entre dois registradores. Além disso, é possível

analisar o caminho crítico dentro de um estágio a partir do atraso associado à cada célula lógica ou característica da LUT, em dispositivos FPGA.

O diagrama exposto na Figura 11 apresenta um modelo funcional para implementação do circuito híbrido $GFN_{8/4}$ para o CLEFIA, sem considerar o desempenho geral do circuito. Na estrutura proposta, os registradores localizados nas saídas da *T-box* e do registrador de deslocamento (SRL16) delimitam os estágios de execução do sistema. Até então, é possível afirmar que o caminho crítico do sistema está localizado entre a saída de 128 bits das *T-boxes* e suas respectivas entradas de 32 bits, quando operando no modo 4-*branch*. Neste caminho, um total de doze variáveis diferentes podem ser contabilizadas, ao longo dos três níveis lógicos que representam o caminho crítico do sistema (④-①-③). No modo 8-*branch* a saída é extraída diretamente do registrador de deslocamento. Além disso, os dois registradores, usados para o escalonamento das permutações, reduzem o número de níveis no caminho e o seu impacto no desempenho geral do circuito. Ao retirar a capacidade de realizar o encaminhamento direto da entrada (P_i), o caminho crítico do circuito pode ser reduzido para apenas dois níveis lógicos. Desse modo, os valores provenientes da saída do estágio ④ são alimentados diretamente na entrada do estágio ③. Para isso, foi necessário adicionar um multiplexador ao bloco ③, o qual alimenta o estágio com o bloco de dado oriundo do estágio ④ ou do registrador localizado no topo da Figura 12 nos modos 4-*branch* e 8-*branch*, respectivamente.

Com esta proposta de otimização, apresentada na Figura 12, o caminho formado entre a saídas e entradas das *T-boxes* possui dois níveis lógicos, compostos por nove variáveis diferentes de 32 bits. Note a presença de um multiplexador, necessário para configuração do circuito no modo 8-*branch*. Apesar de se apresentar como um acréscimo à estrutura do bloco, sua inclusão não interfere no número de LUTs, haja vista que o estágio pode ser implementado a partir de LUT de 6 entradas. O mesmo acontece com o caminho delimitado entre os estágios ④ e a entrada do SRL (②). Uma vez que o estágio de saída está ligado diretamente à um registrador, o mesmo não interfere no desempenho geral do sistema.

4.2 EXPANSÃO DE CHAVE

A estrutura de expansão de chaves, apresentada na Figura 13, baseia-se na arquitetura de *data path* misto (128 e 32 bits), proposta por Chaves (2013).

Uma vez que a expansão de chaves do CLEFIA é realizada após o cálculo da $GFN_{4,n}$ ou $GFN_{8,n}$, o primeiro passo corresponde à obtenção da cadeia de caracteres de entrada da GFN. Este valor, por sua vez, pode ser a cópia de uma chave de codificação de 128 bits K ou formada por uma cadeia de 256 bits, proveniente da combinação entre KL e KR. Além disso, foi dito na Seção 2.4 que, para uma chave de codificação de 128 bits, a *whitening key* é obtida diretamente da chave de entrada. Por outro lado, a

whitening key derivada de chaves de 192 e 256 bits é obtida através da operação XOR entre os blocos da parte alta e da parte baixa da chave K, representados por (A) e (B) na Figura 13, respectivamente. O valor da *whitening key* não é utilizado durante a expansão de chave e sua saída é ligada diretamente ao circuito da GFN. Esta característica influencia diretamente a implementação, no que diz respeito à integração dos circuitos. Detalhes sobre a implementação e integração do módulos são discutidos no próximo Capítulo.

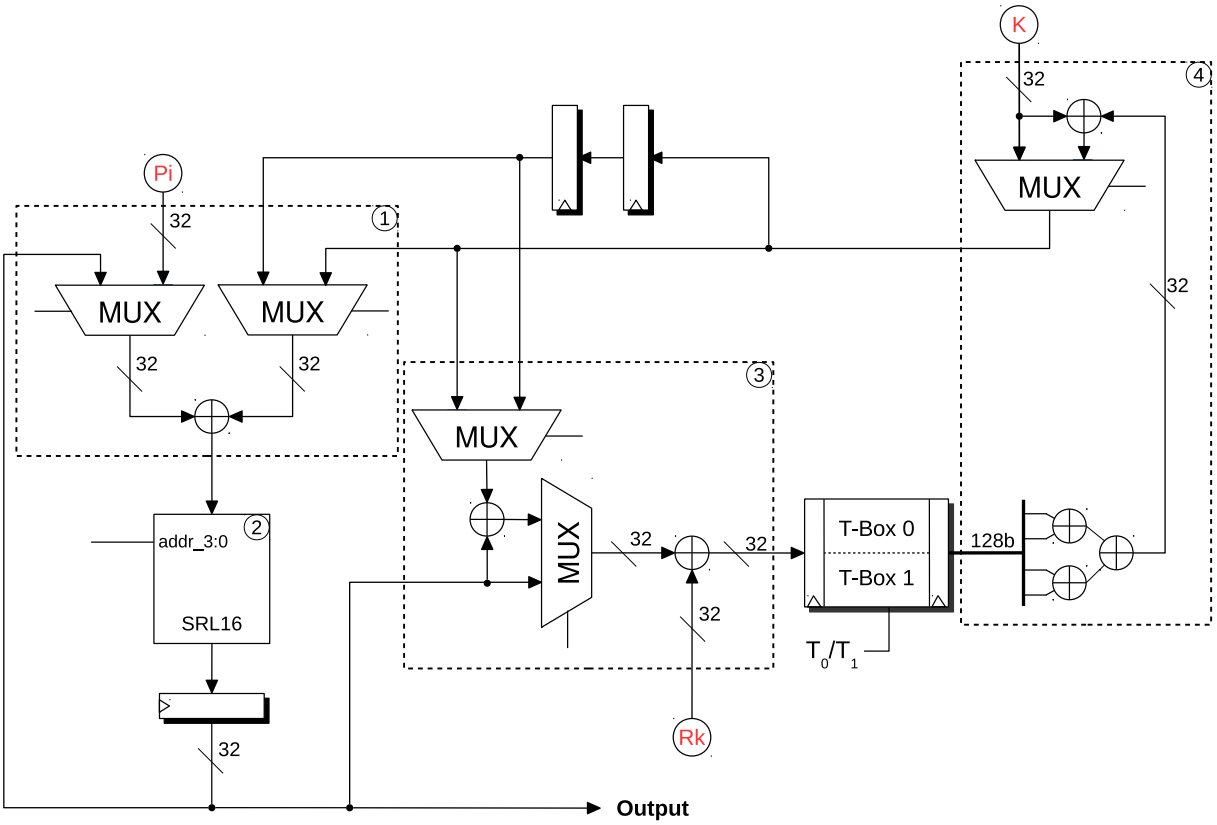


Figura 12 – Proposta de arquitetura para a estrutura 8/4-branch do CLEFIA com redução do caminho crítico.

Uma vez concluída a etapa de obtenção da chave intermediária L, a expansão de chave é processada a partir de três estágios. O estágio (5) representa a comutação entre as chaves (KL e KR) utilizadas no processamento da expansão. Como descrito na Seção 2.4, o cálculo alterna entre o uso das chaves de cifra K para 128 bits, KL e KR para 192 ou 256 bits e zeros. Uma vez que, para compor KR em chaves de 192 bits é necessário obter uma cópia negada dos 64 bits mais baixos de KL, um multiplexador é requerido para selecionar entre os blocos de KR ou $\overline{KL}$. Apesar de serem conduzidos por barramentos de 128 bits, internamente, o canal de saída do estágio (5) é dividido em quatro cadeias de 32 bits, por meio de multiplexadores específicos.

No estágio (6), composto por um *data path* de 128 bits, a função Σ é aplicada sobre L, seguida do cálculo das *round keys*, no estágio (7). A função Σ pode ser realizada de

duas formas: (i) para chaves de 128 bits, apenas o valor de LL é utilizado; e (ii) para chaves de codificação maiores, os valores de LL e LR são computados de forma iterativa. Os dois modos de operação utilizam um caminho de dados de 128 bits. A fim de pré-carregar os valores iniciais de L, um conjunto de blocos de 32 bits alimentam o circuito através do sinal (L) na Figura 13. Os valores obtidos a partir da GFN são carregados nos registradores LL e LR através de um conjunto de blocos de 32 bits em um canal de 128 bits, como descrito na Figura 13. A implementação em hardware desse pré-carregamento é feita através de operações de deslocamento de 32 bits. Além disso, tendo em vista não interferir no caminho crítico, um registrador de 128 bits foi implementado na saída do estágio (6).

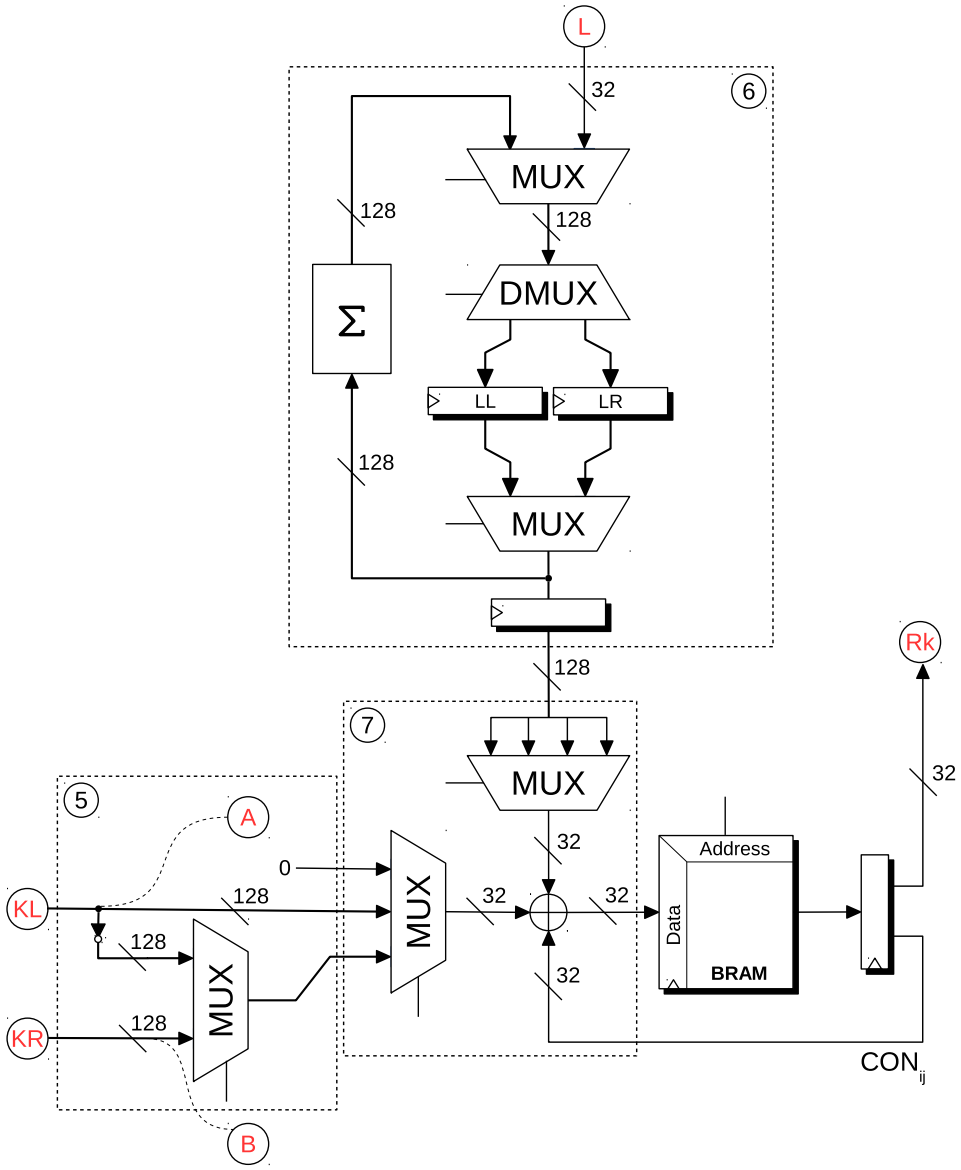


Figura 13 – Estrutura proposta para o mecanismo de Expansão de Chave.

Para cada tamanho de chave, um conjunto pré-definido de constantes é mapeado dentro de uma BRAM de 32 bits. Tanto as constantes (CON_i^j) quanto as *round keys* são armazenadas dentro da mesma BRAM.

A última etapa para obtenção de uma *round key* é conduzida a partir da operação XOR entre a constante correspondente à iteração CON_i^j, a chave de entrada K_i, KL/KR_i ou zeros e o valor de L_i. Este processo é realizado através de um caminho de dados de 32 bits, com o resultado sendo armazenado nos primeiros endereços da BRAM.

Uma atenção especial foi dada ao mapeamento da BRAM, no sentido de obter um mecanismo de endereçamento adequado e, ao mesmo tempo, de baixa complexidade. A BRAM de chaves foi organizada de modo que, o endereço inicial de cada conjunto de chaves ou constantes, seja uma potência de base dois. Considerando o uso de cadeias de 32 bits, um mínimo de 9 kb seria necessário para armazenar todas as chaves. Entretanto, com o uso do mecanismo de mapeamento proposto, o tamanho do bloco de memória necessário aumentou para aproximadamente 12 kb, possível de ser armazenado confortavelmente em BRAMs de 18 kb ou 36 kb.

Apesar de ser capaz de realizar a expansão de chaves de todos os tamanhos suportados pelo CLEFIA, a arquitetura proposta na Figura 13 resultou em uma estrutura pouco otimizada em termos de área ocupada. Devido ao aumento da estrutura do *data path*, associado às características inerentes à expansão de chaves de 192 e 256 bits, a arquitetura requer multiplexadores e registradores maiores. Em dispositivos FPGA, estruturas com estas características tendem a resultar em implementações pouco otimizadas, dada a limitação no tamanho das suas LUTs. Além disso, uma estrutura de expansão de chave pouco otimizada interfere também na frequência de operação do mecanismo de cifra e, consequentemente, no desempenho geral do sistema.

4.2.1 Utilizando "Registradores de Deslocamento" na Expansão de Chave

No sentido de obter um arquitetura compacta, capaz de atender aos requisitos de alto desempenho computacional, este trabalho propõe duas otimizações, a partir da substituição dos registradores e multiplexadores por LUTs no modo SRL. A primeira otimização é relativa ao estágio (5). Na estrutura apresentada na Figura 13, este estágio corresponde a um *data path* de 128 bits. Na estrutura descrita a partir da Figura 14, o estágio (5) opera sob um caminho de dados de 32 bits. No lugar de processar os blocos oriundos diretamente da entrada do circuito, esta nova estrutura considera o armazenamento da chave de codificação em um conjunto de LUTs operando no modo SRL16 (estágio (6)). Uma vez que a chave tenha sido armazenada dentro do registrador de deslocamento, ela pode ser enviada à estrutura $GFN_{4/8,n}$, para o cálculo da chave intermediária L, através da ligação (K), após o registro de saída do estágio (6) na Figura 14. Da mesma forma, esta saída é utilizada para fornecer as *whitening keys* à estrutura de cifra.

As *whitening keys* derivadas de chaves de 192 e 256 bits são obtidas através da operação XOR de 32 bits, representada na parte superior do estágio (5). Os dois

registradores, localizados à esquerda da Figura 14, são utilizados no escalonamento dos blocos de 32 bits necessários para a obtenção das *whitening keys*. Dessa forma, dois ciclos são necessários para o cômputo de cada bloco de 32 bits da *whitening key*. Uma vez que esta operação só é realizada uma vez a cada troca de chave, tal latência extra não tem influência no *throughput* geral do circuito. O valor resultante da XOR é então armazenado no bloco SRL16.

O estágio (6) da Figura 13 também foi otimizado de modo a operar sob o caminho de dados de 32 bits, transformando-se nos estágios (7) e (8), destacados na Figura 14. No intuito de atender os requisitos dessa etapa de processamento de forma compacta, os blocos de 32 bits da chave intermediária L, oriundos da GFN, são armazenadas em LUTs operando no modo SRL32 (estágio (8)). Dessa forma, a transformação Σ passa a atuar sobre blocos de 32 bits, a partir da combinação de partes da chave L devidamente escalonadas. Para tanto, um registrador intermediário, localizado na saída do estágio (8), é usado para armazenar parte do valor de L, a qual é combinada com o segundo segmento oriundo diretamente da saída do bloco SRL32. Detalhes sobre o escalonamento das operações relizadas no *data path* da Figura 14 são apresentados na Seção 5.3.

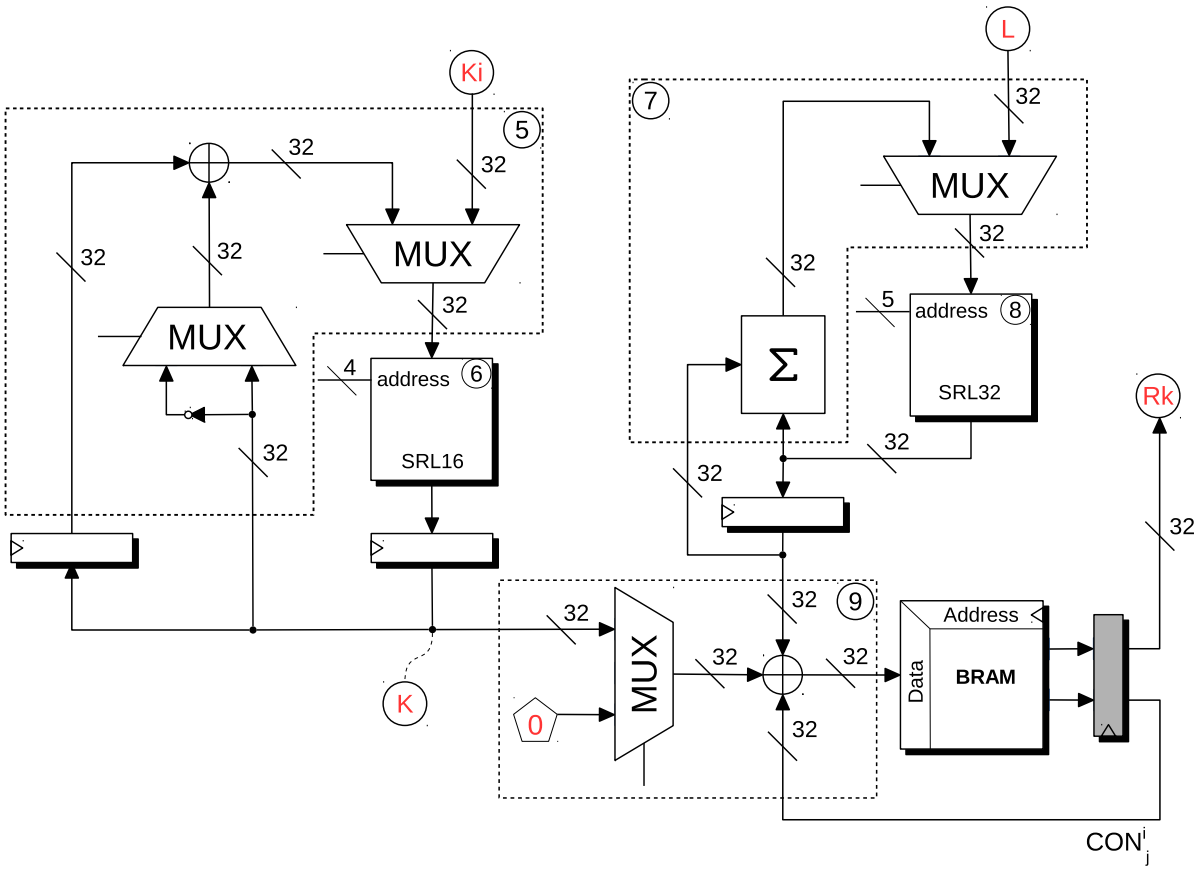


Figura 14 – Proposta de estrutura de expansão de chave completa com uso de registradores de deslocamento (SRL).

4.3 CONCLUSÕES GERAIS SOBRE O PROJETO DA ARQUITETURA

Este capítulo discutiu aspectos gerais da implementação das estruturas de cifra e expansão de chave do CLEFIA. O aspecto predominante, nas duas arquiteturas discutidas, está no uso de registradores de deslocamento endereçáveis para armazenar os valores intermediários da *CLEFIA Feistel Word Swap*. O projeto também superou a dificuldade de implementar a expansão de chave completa, ao propor o uso destes registradores de deslocamento para armazenar a chave de codificação, a *whitening key* e os valores intermediários de L e $\Sigma(L)$. Além disso, um cuidado especial foi tomado no que diz respeito à manutenção do caminho crítico interno às duas estruturas, assim como na ligação entre elas. Com isso, foi possível integrá-las de forma otimizada, sem influência no desempenho geral do circuito. O capítulo seguinte aborda detalhes da implementação física dos dois circuitos, assim como os detalhes de implementação e a integração dos dois componentes e os elementos de controle.

5 DETALHES DE IMPLEMENTAÇÃO E MELHORIAS

Projetos de estruturas compactas em FPGA demandam por análises detalhadas no que se refere ao uso de recursos, em especial as LUTs. Em dispositivos FPGA, a lógica combinacional, implementada na forma de LUTs, é fator determinante no sentido da eficiência do circuito final No capítulo anterior, as Figuras 12 e 14 apresentaram uma visão geral das estruturas funcionais para cifra e expansão de chaves, baseadas no compartilhamento de recursos e na exploração de estruturas internas dos dispositivos FPGA.

Este capítulo apresenta um detalhamento dos aspectos de implementação, assumindo como alvo tecnológico dispositivos da família Xilinx Virtex 5. Dessa forma, consideraremos a presença de LUTs de seis entradas (LUT6) configuráveis. Dispositivos modernos, tal qual os das famílias Virtex 7, Kintex e Artyx, não foram avaliados neste estudo, uma vez que o uso efetivo do tipo de tecnologia associada a estes dispositivos exigiria modificações adicionais ao circuito, extrapolando os objetivos deste trabalho.

5.1 GERENCIAMENTO DAS *WHITENING KEYS* E CHAVE DE CODIFICAÇÃO

Na Seção 4.1, nós assumimos que as *whitening keys* poderiam ser fornecidas ao circuito de codificação através de um barramento controlado por um circuito externo. Ao considerar apenas a implementação da estrutura GFN, a BRAM utilizada para armazenar as *round keys* pode armazenar as *whitening keys*. Neste caso, é possível usar espaços de endereçamento não alocados da memória de 18 kb ou 36 kb para fornecer as chaves, na ordem necessária, assumindo ainda que ambos os conjuntos de chaves são computados fora do *chip*. Por possuir endereços de memória não utilizados, é possível também extrair zeros desta memória, sem interferir diretamente no caminho de dados do circuito enquanto as *whitening keys* não são utilizadas.

A partir da introdução do circuito de expansão de chave, essa BRAM deixa de existir e as *whitening keys* também passam a ser computadas internamente. Na Seção 4.2, foi definido que ambos os valores correspondentes às *whitening keys* e à chave de codificação (K), armazenados no bloco SRL16, poderiam ser extraídos da mesma saída. Uma vez que, para armazenar a chave de codificação e a respectiva *whitening key*, são necessárias, no máximo, 12 posições de memória, é possível enviar zeros para a entrada (K) da Figura 12, apontando para um dos endereços não utilizados do registrador de deslocamento.

5.2 CIRCUITO FINAL E DETALHES DE IMPLEMENTAÇÃO

Neste ponto, as arquiteturas finais dos circuitos propostos para a cifra e expansão de chave do CLEFIA, são apresentadas nas Figuras 15 e 17, as quais são referenciadas à medida em que os detalhes de implementação forem apresentados. Além disso, ao longo das próximas seções, os modos de operação para expansão de chaves de 192, 256 e 128 bits são indicados como CLEFIA-256 e CLEFIA-128, respectivamente.

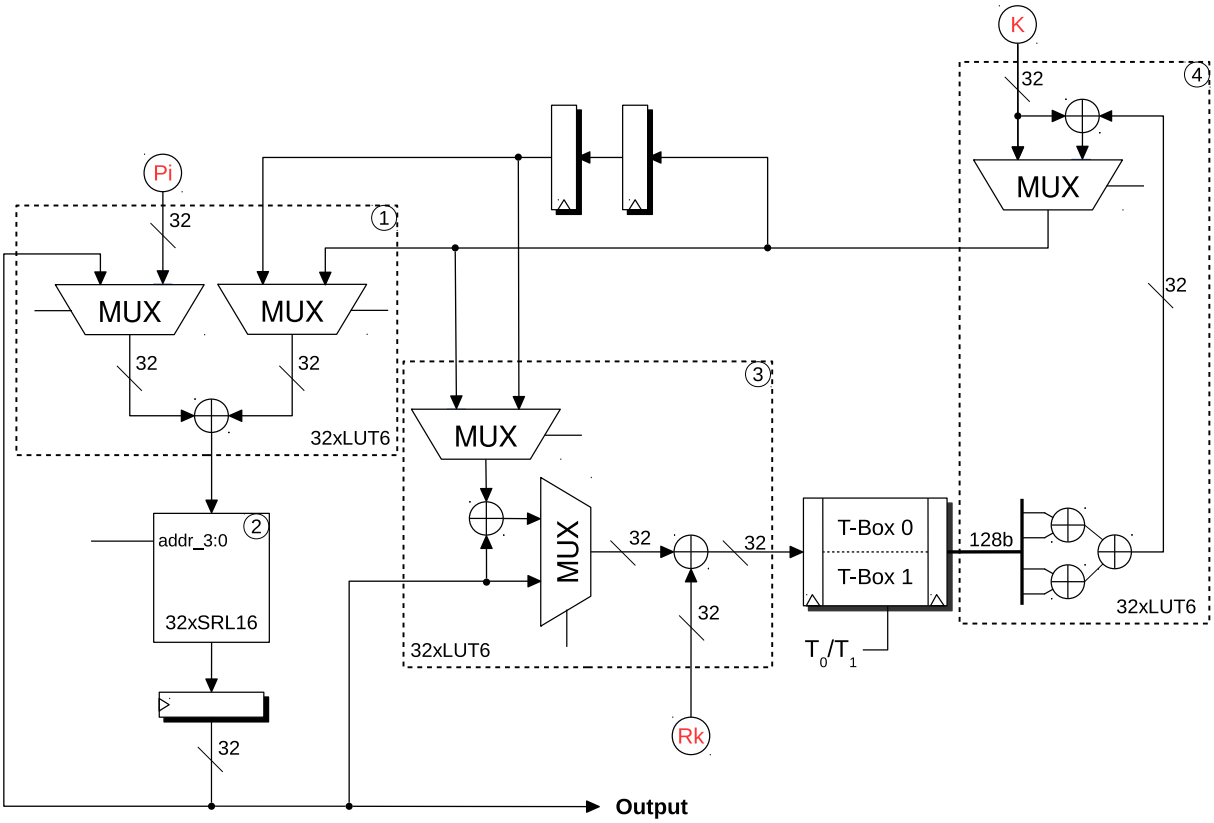


Figura 15 – Estrutura final da Feistel 8/4-branch do CLEFIA.

Na Seção 3.1, foi verificado que o elemento bloco lógico é essencialmente composto a partir de LUTs de tamanhos de entradas variáveis. A combinação dessas entradas, por meio de uma função lógica programada no dispositivo, produzirá um bit de saída. Por se tratar de um *data path* de 32 bits, a arquitetura aqui proposta usa, no mínimo, 32 LUTs para implementar cada estágio.

5.2.1 Estrutura Feistel 8/4-branch

Na estrutura representada na Figura 15, o bloco (1) é responsável por controlar o fluxo de entrada do circuito e do dado encaminhado pelo estágio (4), sendo composto por 32 LUT6s. O registrador de deslocamento (bloco (2)) é implementado a partir da combinação de 32 LUTs configuradas no modo SRL16. O estágio (3) é responsável por

determinar se a informação que alimentará as entradas das *T-boxes* é proveniente do estágio (4) (modo 4-*branch*) ou da saída do registrador de deslocamento (modo 8-*branch*). Note que a XOR na entrada do estágio (3) realiza a mesma operação a qual são submetidos os dados oriundos dos multiplexadores, presente no estágio (1). Todavia, a réplica desta operação permitiu a redução do caminho crítico do circuito - além disso, a XOR adicional pôde ser mapeada para a mesma função lógica das LUT4s do estágio (3).

Por fim, o estágio (4) é responsável por realizar a árvore de XORs nas saídas das *T-boxes*. Adicionalmente, este estágio é também utilizado como caminho para introdução das *whitening keys* no circuito. Assim, este estágio assume um papel significativo no sistema, no sentido de remover a lógica do estágio de saída, uma vez que a soma das *whitening keys* referentes à última iteração é realizada antes de o bloco ser realimentado na direção do registrador de deslocamento. Dessa forma, o bloco resultante pode ser extraído, em ordem, diretamente da saída do registrador de deslocamento.

5.2.2 Mapeamento das *T-boxes* em BRAM

A partir dos estudos apresentados neste trabalho, a estrutura responsável por computar a função F do CLEFIA foi substituída por *T-boxes* correspondentes, projetadas na forma de *look up tables* e implementadas em BRAMs nativas do dispositivo FPGA. Face aos dispositivos das famílias Virtex da Xilinx, este estudo considera BRAMs de 36 kb ou 18 kb para o projeto das *T-boxes*. Considerando uma estrutura de cifra com caminho de dados de 32 bits, duas BRAMs *dual-port* são necessárias, uma vez que a multiplicação entre o vetor de saída e a matriz de difusão ($GF(2^8)$) é realizada fora da BRAM. O endereço de cada *T-box* é derivado da operação XOR entre os bytes oriundos da *round key* e do valor encaminhado do estágio de (4), na Figura 15. Cada byte resultante desta operação lógica é responsável por mapear uma BRAM. No estágio (3), duas operações lógicas podem acontecer, de acordo com o escalonamento ou o modo de operação:

$$TBox_{input} = SRL16_{reg} \oplus RK_i; \tag{5.1}$$

$$= SRL16_{reg} \oplus RK_i \oplus Forward\_Word(4); \tag{5.2}$$

onde $SRL16_{reg}$ corresponde ao dado oriundo do registrador localizado na saída do registrador de deslocamento, RK_i o bloco de 32 bits correspondente à *round key* da iteração e $Forward\_Word(4)$ a palavra encaminhada a partir do estágio (4). Durante o processo de cômputo da chave intermediária apenas a Equação 5.1 é realizada, uma vez que o bloco de entrada do estágio (3) na Figura 15 será sempre oriundo do registrador de deslocamento. Por outro lado, no processo de cifra, a Equação 5.1 é usada somente durante a inicialização do circuito. Detalhes sobre o escalonamento das operações são discutidos nas seções a seguir.

Adicionalmente, um nono bit é utilizado para compor o endereço, com o objetivo de comutar entre as *T-boxes* associadas às funções F_0 e F_1, apresentadas na Figura 16. Dado

que uma BRAM é capaz de retornar blocos de 32 bits, cada bloco de memória armazena aproximadamente 16 kb. Uma vez que o projeto do mecanismo de seleção do estágio (3) utiliza dois bit, 32 LUT6s são necessárias para sua implementação. Na Figura 16, as entradas "*Fwd byte (4)*" e "*Fwd byte (FF)*" correspondem ao resultado encaminhado da saída do estágio (4) e do dado oriundo da saída do registrador localizado na região superior da Figura 15.

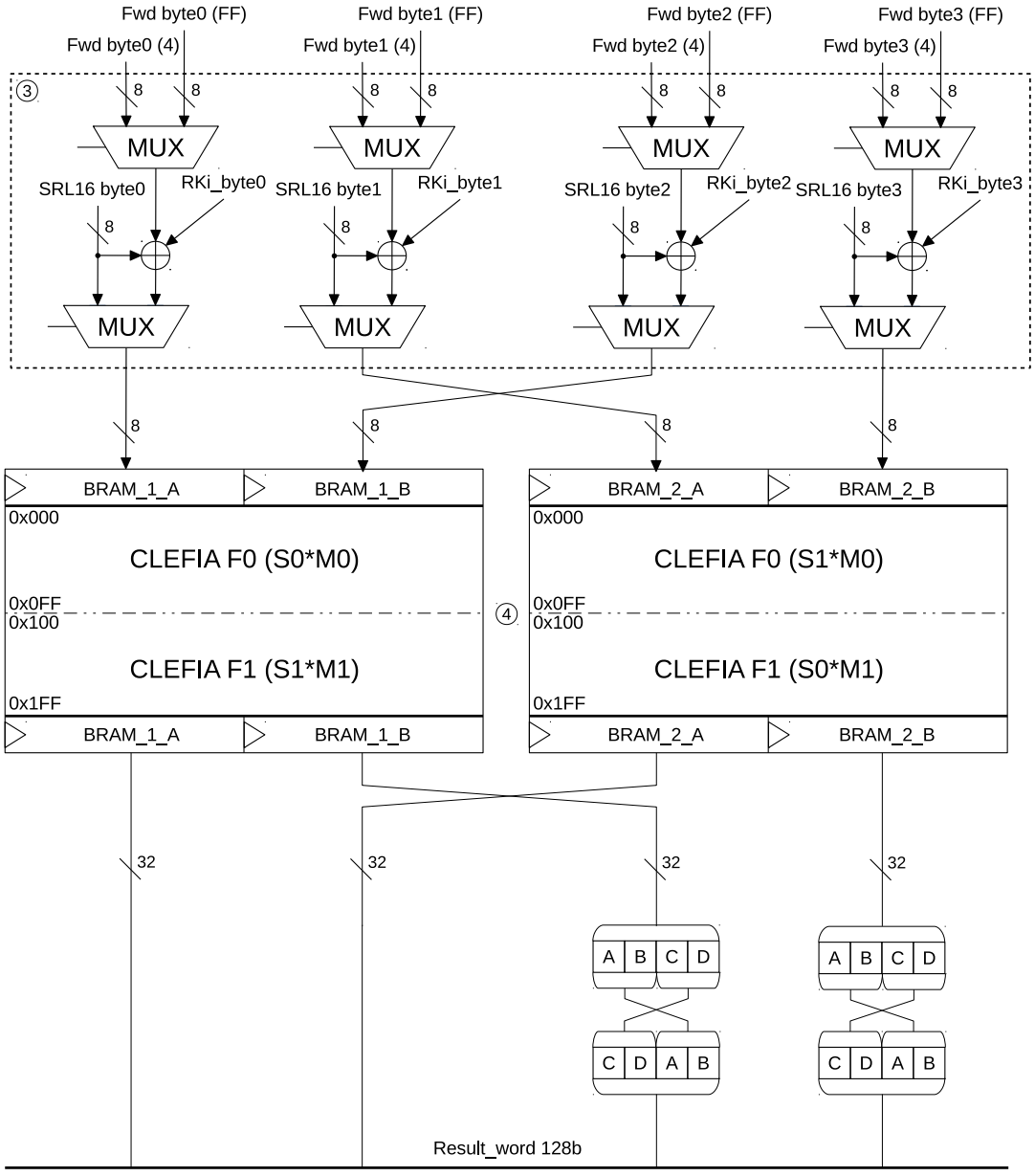


Figura 16 – Visão detalhada da organização das BRAMs para as *T-boxes*.

5.2.3 Estrutura de Expansão de Chaves Completa

Em nível de implementação, o circuito de expansão de chaves, apresentado na Figura 17, é dividido em três estágios funcionais. O primeiro, representado a partir dos blocos (5) e (6), compostos por 32 LUT5s e 32 SRL16s, respectivamente, corresponde

ao estágio de entrada do circuito, além do cálculo e armazenamento das *whitening keys*. Este estágio constitui a primeira etapa de processamento do CLEFIA, após a inicialização do circuito. Os blocos de 32 bits da chave de cifra K de 128 bits ou $KL \mid KR$ de 192 ou 256 bits, são introduzidos no circuito através da entrada (K_i). O bloco de 64 bits que compõe a parte baixa de K, em chaves de 192 bits (KR_2 e KR_3), são obtidos a partir do escalonamento dos endereços do registrador de deslocamento e controle das entradas de realimentação, localizadas na região inferior do estágio (5). Uma vez que há espaços vazios no SRL16, é possível compartilhar os recursos do estágio (5) para pré-computar estes valores, antes de serem requisitados. Caso contrário, uma lógica adicional seria necessária na saída (K) do circuito, implicando no acréscimo de 32 LUT2s.

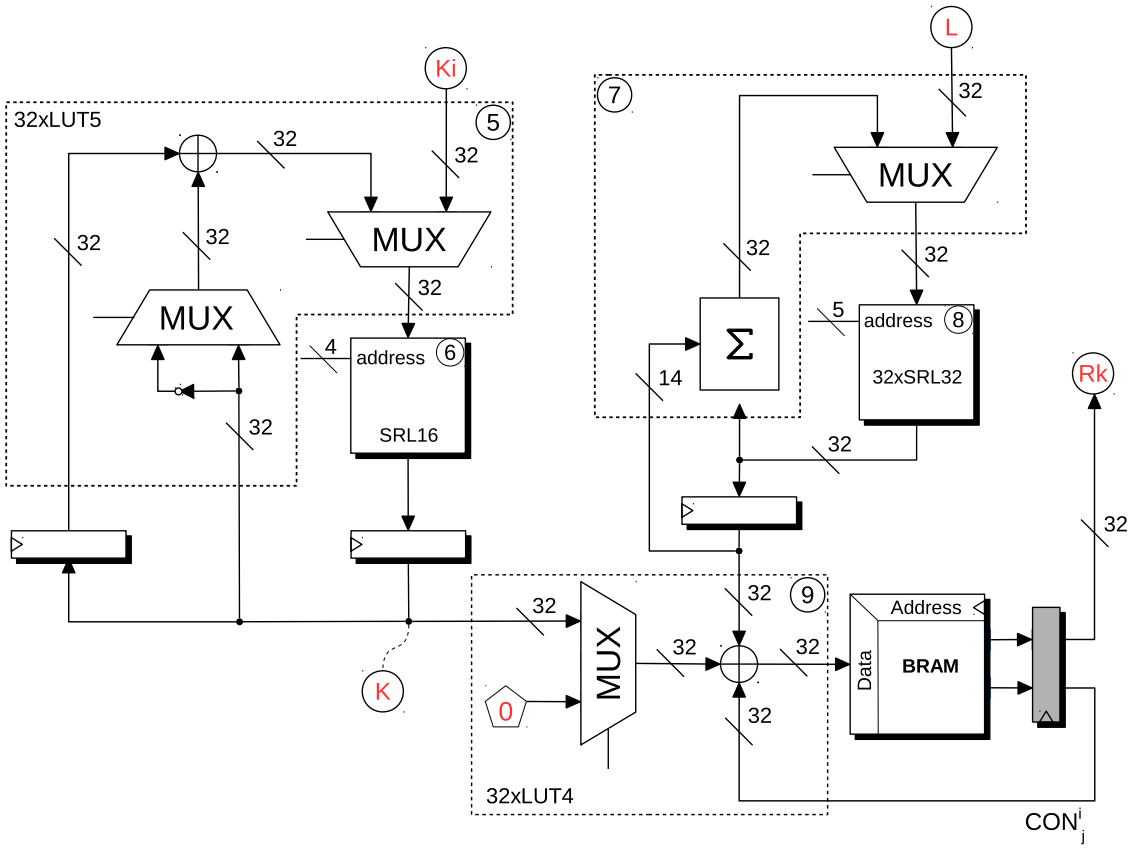


Figura 17 – Estrutura final do circuito de Expansão de Chave.

As *whitening keys* para chaves de 192 e 256 bits são obtidas a partir do controle da saída do registrador de deslocamento (6) da Figura 17. Uma das características da LUT implementada no modo SRL é a possibilidade de controlar o deslocamento dentro do registrador, permitindo assim fixar os endereços de cada chave e, consequentemente, reduzir a complexidade da unidade de controle. Deste modo, apenas 12 posições do registrador de deslocamento (6) são utilizadas ao longo do processo. Uma vez finalizado o cálculo e armazenamento da *whitening keys*, a chave formada a partir da combinação $KL \mid KR$ é submetida ao módulo de cifra $GFN_{8/4,n}$, através do barramento (K).

O estágio (7) corresponde ao processo de captura da chave intermediária L e cômputo da função Σ. Os blocos de 32 bits, oriundos do bloco $GFN_{8/4,n}$, são introduzidos através do barramento (L) e alimentam o registrador de deslocamento (8), na etapa de inicialização. Este registrador de deslocamento, por sua vez, devido à necessidade de armazenar chaves L e $\Sigma(L)$ de 256 bits, é implementado a partir da combinação de 32 LUTs, configuradas no modo SRL32.

Durante a descrição do escalonamento das operações realizadas neste estágio, será possível verificar que são necessários, no mínimo 20 bits, de cada SRL32, para garantir a realização da função Σ de forma iterativa - tal função faz parte do bloco (7) na Figura 17. Uma estrutura iterativa é proposta, no sentido de reduzir o *data path* necessário para realizar as permutações através de barramentos de 32 bits. Dessa forma, quatro ciclos de clock são necessários para realizar as permutações sob as chaves L, LL ou LR, de 128 bits, a partir da combinação de pares de blocos de 32 bits permutados. Cada iteração corresponde à combinação de partes de dois blocos de 32 bits, oriundos da saída do registrador de deslocamento (8) e o registrador na entrada do estágio (9). A partir do endereçamento adequado dos blocos de 32 bits oriundos do registrador de deslocamento, é possível implementar a função Σ com a introdução de apenas três *stalls* na realização de cada iteração do processo de expansão de chaves.

Devido ao escalonamento utilizado para realização das permutações na função Σ, a estrutura implementada utiliza blocos de 14 bits na entrada do circuito que realiza a permutação e combinação dos bits de L_i. Nesta configuração, o bloco de entrada oriundo do registrador de entrada do estágio (9) foi divido em dois blocos de 7 bits, (A) e (B), na Figura 18. Estes barramentos, por sua vez, correspondem às partes alta e baixa da saída do registrador, respectivamente. Diante desta característica, a estimativa de implementação deste estágio em nível de LUTs requer uma análise individualizada, uma vez que o *data path* não pode mais ser analisado como essencialmente de 32 bits. A estrutura apresentada na Figura 18 representa a configuração interna do bloco (7), detalhando implementação da função Σ. O elemento responsável pela permutação da função Σ corresponde à uma operação de combinação de partes dos dados oriundos de (8) e do registrador de entrada do estágio (8), de acordo com um critério de seleção pré-determinado. Dessa forma, o bloco (7), composto por 7 LUT3s, é utilizado para selecionar entre os blocos de entrada (A) e (B), e 32 LUT5s, nas quais são realizadas as operações de seleção, combinação e permutação.

Neste momento, é possível questionar a presença de bloco adicional, utilizado apenas para seleção dos blocos oriundos do registrador de entrada do estágio (9). Em nível de projeto, é possível descrever este estágio usando apenas 32 LUT6s, adotando um único canal de entrada, assim como implementado para o barramento (C). Teoricamente, ao utilizar LUTs com menor número de entradas, é possível acelerar o processo de roteamento

do circuito. Por outro lado, na prática, isso depende tanto das características presentes nos algorítimos de síntese lógica e física das ferramentas de EDA, como da experiência do projetista em proceder com o particionamento e roteamento manual dos componentes lógicos de um circuito.

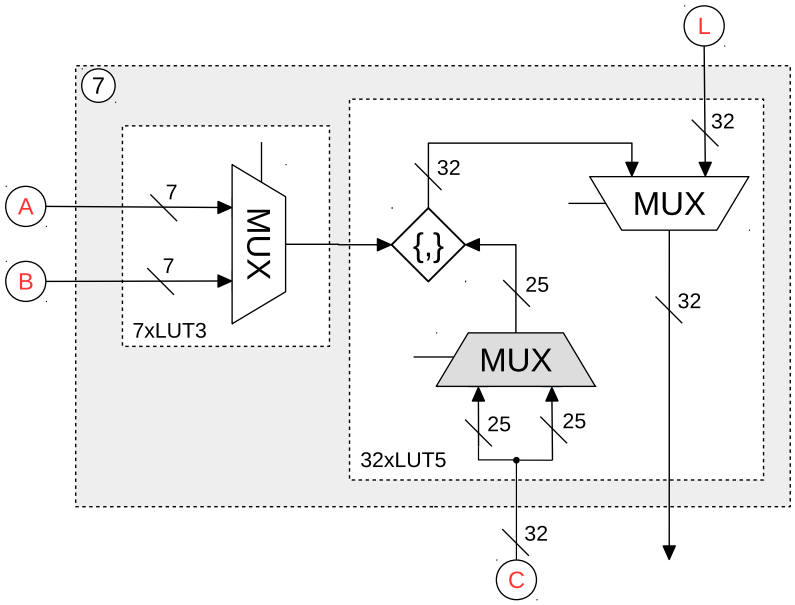


Figura 18 – Detalhamento da estrutura de execução da função Σ.

Os resultados destacados até aqui foram confirmados por meio de verificação do *floorplanning* do circuito. Os detalhes acerca da análise de área ocupada e desempenho do circuito são apresentados no capítulo seguinte.

Por fim, o estágio (9) consiste do cômputo da *round key* a partir de blocos de 32 bits oriundos de L ou $\Sigma(L)$, da chave de cifra K ou de um das constantes formadas a partir dos conjuntos CON^{128}, CON^{192} ou CON^{256}. Uma característica de operação deste estágio é que a entrada correspondente ao bloco da chave K pode conter zeros, a partir do controle do endereçamento do bloco (6). Com isso, o estágio (9) poderia ser implementado usando 32 LUT3s. Entretanto, resultados experimentais sugerem que a implementação proposta para o bloco (9) na Figura 17, baseada apenas em LUTs de 4 entradas, apresenta melhores resultados em termos de eficiência. Além disso, considerando a implementação do circuito em dispostivos FPGA, ambas as soluções ocupam a mesma quantidade de LUTs.

5.2.4 Organização da Memória de Chaves

A fim de conduzir a expansão de chave de forma compacta, as constantes CON_i^j e as *round keys* RK_i^j geradas, onde j determina o conjunto de chaves (128, 192 ou 256 bits), são armazenadas em uma BRAM. Adicionalmente, as "chaves" usadas na obtenção da chave intermediária L também são armazenadas nessa mesma estrutura. A BRAM descrita na Figura 19 opera tanto como RAM, armazenando e lendo as *round keys* geradas, quanto

como ROM, onde as constantes e "chaves" são armazenadas. Para isso, uma *dual-port* RAM foi utilizada, para mapeamento das *round keys* e constantes utilizadas nos processos de cômputo da cifra e expansão da chave de cifra. A proposta apresentada na Figura 19 difere daquela apresentada em Chaves (2013) na medida em que considera o armazenamento de todas as constantes necessárias para expansão de chaves de 128, 192 e 256 bits.

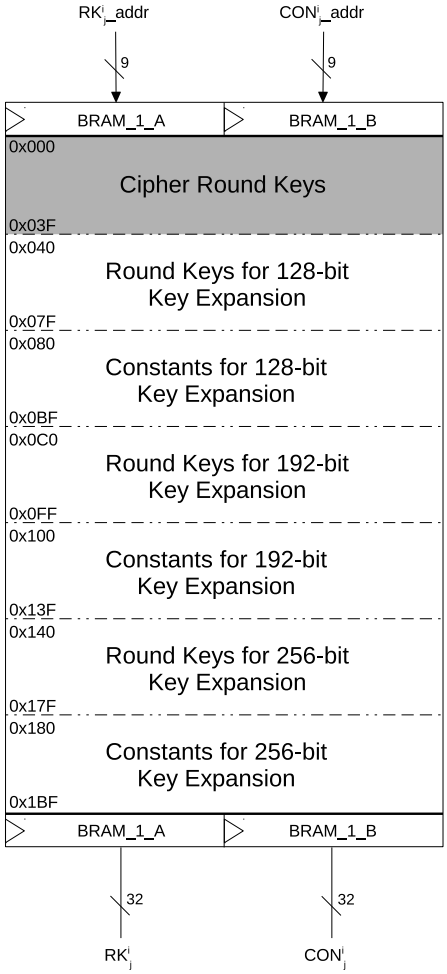


Figura 19 – Visão detalhada da organização da memória de chaves.

O controle do endereço de entrada da memória é feito dentro da unidade de controle, por meio de uma porta compartilhada. Esta porta, endereçada a partir do sinal $RK^i_j\_addr$, pode apontar tanto para as *round keys*, quando uma cadeia de caracteres está sendo codificada, quanto para as "chaves" usadas no cálculo do valor de L. As constantes, usadas apenas durante a expansão de chave, são acessadas através de uma porta exclusiva.

No sentido de reduzir a complexidade do controle associado ao endereçamento dos dados, o endereço inicial de cada conjunto de constantes ou "chaves" corresponde à uma potência de base dois. Este valor é mapeado a partir da combinação entre os três bits mais significativos da cadeia de bits de endereço. Como pode ser observado na porção destacada da Figura 19, a parte mais alta da BRAM é usada para armazenar as *round keys* computadas durante a execução do mecanismo de expansão. Dado que esta

memória armazena cadeias de dados de 32 bits, aproximadamente 14 kb são necessários para armazenar as constantes, chaves pré-computadas e as *round keys* geradas, os quais podem ser alocados nas memórias de 18 kb ou 36 kb do dispositivo.

5.3 DESCRIÇÃO DO ESCALONAMENTO DE OPERAÇÕES

Na seção anterior, as arquiteturas para os módulos de cifra e expansão de chave foram detalhados. No circuito $GFN_{8/4,n}$, apresentado na Figura 12, três estágios de pipeline podem ser identificados: o estágio de entrada/*retorno* que determina a entrada do registrador de deslocamento; o estágio de encaminhamento que define a entrada das BRAMs; e o estágio de mapeamento das T-*boxes*. Por outro lado, o circuito de expansão de chave é formado a partir de duas etapas de processamento: (i) obtenção da chave K e cálculo das *whitening keys* e; (ii) a expansão de chave, propriamente dita. Esta última, por sua vez, é realizada de forma iterativa em dois estágios de execução: o estágio de entrada e execução da função Σ e; o estágio de cálculo e armazenamento das respectivas *round keys*. O processo de obtenção das *whitening keys* é realizado antes do cálculo chave intermediária, e por isso não interfere no fluxo de dados da expansão da chave de cifra. Detalhes da implementação destes estágios são apresentados nas próximas seções.

5.3.1 Obtenção da chave K e Cálculo das *Whitening Keys*

A primeira etapa a ser considerada durante a execução completa do algoritmo CLEFIA, na estrutura proposta, consiste em introduzir as chaves K ou KL e KR e calcular as *whitening keys*, no caso de chaves maiores que 128 bits. Este processo é detalhado nas Tabelas 1, 2 e 3, considerando chaves de 128, 192 e 256 bits, respectivamente. Nas Tabelas a seguir, a coluna **Delay Slot** representa a saída do registrador destacado à direita na Figura 17. Nesta etapa, as chaves de entrada são armazenadas em blocos de 32 bits, à medida em que são alimentadas dentro do circuito. Uma vez que as *whitening keys* para chaves de 128 bits são obtidas diretamente, apenas 4 ciclos de clock são necessários para esta operação. Adicionalmente, quatro ciclos de clock são utilizados para introduzir a chave K na estrutura GFN.

Chaves de cifra de 192 bits, por sua vez, utilizam 6 ciclos de clock para introduzir os blocos de 32 bits no SRL16. Além disso, 64 bits adicionais são produzidos diretamente, a partir da cadeia binária composta pelos bits da parte alta do bloco de entrada. Neste sentido, dois ciclos de clock adicionais são necessários para computar $KR_2^{192} = \sim KL_0^{192}$ e $KR_3^{192} = \sim KL_1^{192}$. Uma vez que, o cômputo de cada *whitening key* requer dois ciclos de clock, um total de 16 ciclos são necessários para completar o pré-processamento da chave de entrada. Para introduzir a chave $K = KL \mid KR$ de 256 bits, são necessários oito ciclos de clock, totalizando assim 24 ciclos para o início do processo de obtenção da chave intermediária L.

Tabela 1 – Escalonamento da entrada da chave K, de 128 bits, no mecanismo de expansão de chave e introdução da chave no bloco GFN.

Ciclo	**Entrada do SRL**	**Saída do SRL**	**Delay Slot**	**Saída** (K)
1	$K_0 = WK_0$	x	x	x
2	$K_1 = WK_1$	x	x	x
3	$K_2 = WK_2$	x	x	x
4	$K_3 = WK_3$	K_0	x	x
5	–	K_1	x	K_0
6	–	K_2	x	K_1
7	–	K_3	x	K_2
8	–	x	x	K_3

Tabela 2 – Escalonamento da entrada da chave K, de 192 bits, no mecanismo de expansão de chave, geração das *witening keys* e introdução da chave no bloco GFN.

Ciclo	**Entrada do SRL**	**Saída do SRL**	**Delay Slot**	**Saída** (K)
1	KL_0	x	x	x
2	KL_1	x	x	x
3	KL_2	x	x	x
4	KL_3	x	x	x
5	KR_0	x	x	x
6	KR_1	KL_0	x	x
7	$\sim KL_0$	KL_1	0	x
8	$\sim KL_1$	KL_0	0	x
9	–	KR_0	x	x
10	$WK_0 = KL_0 \oplus KR_0$	KL_1	KL_0	x
11	–	KR_1	KR_0	x
12	$WK_1 = KL_1 \oplus KR_1$	KL_2	KL_1	x
13	–	$\sim KL_0$	KR_1	x
14	$WK_2 = KL_2 \oplus \sim KL_0$	KL_3	KL_2	x
15	–	$\sim KL_1$	$\sim KL_0$	x
16	$WK_0 = KL_3 \oplus \sim KL_1$	KL_0	KL_3	x
17	–	KL_1	$\sim KL_1$	KL_0
18	–	KL_2	x	KL_1
19	–	KL_3	x	KL_2
20	–	KR_0	x	KL_3
21	–	KR_1	x	KR_0
22	–	$\sim KL_0$	x	KR_1
23	–	$\sim KL_1$	x	$\sim KL_0 = KR_2$
24		x	x	$\sim KL_1 = KR_3$

O pré-processamento de chaves de cifra de 256 bits é semelhante ao de chaves de 192 bits, diferindo que, no primeiro caso, a obtenção do bloco $K = KL \mid KR$ de 256 bits é realizada de forma direta. Uma vez finalizada a obtenção das *whitening keys*, o processo segue com a introdução dos blocos de 32 bits correspondentes à chave K, através do barramento (K). Este processo novamente requer oito ciclos de clock adicionais para ser concluído. Ambas as chaves de 192 e 256 bits requerem 24 ciclos para conclusão da etapa de pré-processamento e introdução da chave K no módulo GFN.

Tabela 3 – Escalonamento da entrada da chave K, de 256 bits, no mecanismo de expansão de chave, geração das *witening keys* e introdução da chave no bloco GFN.

Ciclo	**Entrada do SRL**	**Saída do SRL**	**Delay Slot**	**Saída (K)**
1	KL_0	x	x	x
2	KL_1	x	x	x
3	KL_2	x	x	x
4	KL_3	x	x	x
5	KR_0	x	x	x
6	KR_1	x	x	x
7	KR_2	x	x	x
8	KR_3	KL_0	x	x
9	–	KR_0	x	x
10	$WK_0 = KL_0 \oplus KR_0$	KL_1	KL_0	x
11	–	KR_1	KR_0	x
12	$WK_1 = KL_1 \oplus KR_1$	KL_2	KL_1	x
13	–	KR_2	KR_1	x
14	$WK_2 = KL_2 \oplus KR_2$	KL_3	KL_2	x
15	–	KR_3	KR_2	x
16	$WK_3 = KL_3 \oplus KR_3$	KL_0	KL_3	x
17	–	KL_1	x	KL_0
18	–	KL_2	x	KL_1
19	–	KL_3	x	KL_2
20	–	KR_0	x	KL_3
21	–	KR_1	x	KR_0
22	–	KR_2	x	KR_1
23	–	KR_3	x	KR_2
24		x	x	KR_3

5.3.2 Circuito de Codificação/Decodificação $GFN_{4/8,n}$

O escalonamento de operações para o CLEFIA, nos modos 4-*branch* e 8-*branch* estão presentes nas Tabelas 4 e 5, considerando todos os tamanhos de chave. Em ambas as Tabelas, a coluna 'Saída da T-Box' representa o resultado do mapeamento das *T-Boxes* e o resultado da operação $GF(2^8)$, derivado pela árvore de XOR (Figura 15 (4)). Além disso,

este processo pertence ao mesmo estágio *pipeline* que 'Entrada do SRL16' e 'Entrada da T-Box'. O estágio 'T-Box Process' (dentro da BRAM) representa o processo de mapeamento das funções F nas *Lookup Tables* do CLEFIA.

No processo de cifra, cada iteração começa quando o primeiro bloco de 32 bits da iteração correspondente é mapeado no registrador do deslocamento SRL16, ou encaminhado do estágio (4) (Figura 15) até a entrada da BRAM. Cada bloco de 32 bits, alimentando a entrada da BRAM, leva três ciclos de clock para ser processado, com o resultado armazenado dentro do registrador de deslocamento. Cada iteração do CLEFIA parte do processamento de duas palavras de 32 bits dentro das *T-Boxes*, representando o cômputo das respectivas funções F_0 e F_1. Dessa forma, fazendo uso do encaminhamento da palavra de saída das *T-Boxes*, uma iteração pode ser realizada em dois ciclos de clock.

Durante a realização da cifra, quatro pulsos de clock são necessários para inicializar a primeira iteração. O primeiro ciclo é utilizado para garantir a ausência de conflitos de dados durante a extração do bloco de saída. Os pulsos restantes correspondem ao atraso necessário para a introdução da 1ª e 3ª palavras de entrada (P_0 e P_2). Após o primeiro ciclo da primeira iteração garante-se que a cadeia a ser codificada, devidamente "*whitened*", foi introduzida dentro registrador de deslocamento. Neste mesmo instante, o CLEFIA já pode ser inicializado a partir do endereçamento sequencial dos blocos P_0 e P_2. Ainda na primeira iteração (*round*), verifica-se a presença de um *stall* em função do atraso inicial - efeito do registrador de saída do SRL16. Veremos a seguir que este atraso, por sua vez, não implica em perdas significativas associadas à latência final do circuito.

Durante a execução do CLEFIA no modo 4-*branch*, o registrador de deslocamento endereça palavras diretamente para a entrada da BRAM apenas na primeira iteração. A partir da segunda iteração, todas as entradas da BRAM são provenientes da XOR entre os blocos encaminhados do estágio (4) e resultante da operação *Feistel Word Swap*, realizada através do registrador de deslocamento.

Para a correta execução do CLEFIA, cada iteração deve consultar duas *T-Boxes* diferentes. Dessa forma, para suprir a necessidade de endereçamento das funções-F, a cada ciclo de clock uma BRAM deve ser acessada alternadamente. Na codificação, esta alternância pode ser verificada na Tabela 4. Devido à diferença presente na realização das permutações durante a decodificação, esta sequência de alternância é repetida na forma: $F_0,\ F_1,\ F_1,\ F_0$.

Ao final da execução da última iteração do CLEFIA ($n = \{18, 22, 26\}$), os elementos do pipeline precisam ser esvaziados (*flush round*), assim como as duas últimas palavras armazenadas no registrador de deslocamento. Neste momento são também introduzidas as duas últimas *whitening keys*. Após esta etapa, o registrador de deslocamento ignora a última permutação, ao endereçar para a saída do circuito os blocos finais computados com as respectivas *whitening keys*, já devidamente ordenados.

Tabela 4 – Escalonamento de operações da da cifra do CLEFIA, onde $a'i$ e $b'i$ são resultados oriundos da Função-F e n o total de iterações.

Iteração	**Entrada do SRL16**	**Entrada da T-Box**	**Processamento na T-Box**	**Saída da T-Box**	**Estágio de Saída**
Inicialização	X	X	X	0	X
	P_0	X	X	0	X
	$P_1 + WK_0$	X	X	0	X
	P_2	X	X	0	X
1ª	$P_3 + WK_1$	$P_0 + RK_0$	X	0	X
	X	$P_2 + RK_1$	$F_0\{P_0 + RK_0\}$	0	X
2ª	$a_1 + P_1 + WK_0 = A_1$	$A_1 + RK_2$	$F_1\{P_2 + RK_1\}$	a_1	X
	$b_1 + P_3 + WK_1 = B_1$	$B_1 + RK_3$	$F_0\{A_1 + RK_2\}$	b_1	X
3ª	$a_2 + P_2 = A_2$	$A_2 + RK_4$	$F_1\{B_1 + RK_4\}$	a_2	X
	$b_2 + P_0 = B_2$	$B_2 + RK_5$	$F_0\{A_2 + RK_5\}$	b_2	X
4ª	$a_3 + B_1 = A_3$	$A_3 + RK_6$	$F_1\{B_2 + RK_6\}$	a_3	X
	$b_3 + A_1 = B_3$	$B_3 + RK_7$	$F_0\{A_3 + RK_7\}$	b_3	X
5ª	$a_4 + B_2 = A_4$	$A_4 + RK_8$	$F_1\{B_3 + RK_8\}$	a_4	X
	$b_4 + A_2 = B_4$	$B_4 + RK_9$	$F_0\{A_4 + RK_9\}$	b_4	X
6ª	$a_5 + B_3 = A_5$	$A_5 + RK_{10}$	$F_1\{A_5 + RK_{10}\}$	a_5	X
	$b_5 + A_3 = B_5$	$B_5 + RK_{11}$	$F_0\{B_5 + RK_{11}\}$	b_5	X
...	...				
n	$a_{n-1} + B_{n-2} = A_{n-1}$	$A_{n-1} + RK_{2n-2}$	$F_1\{B_{n-2} + RK_{2n-3}\}$	a_{n-1}	X
	$b_{n-1} + A_{n-2} = B_{n-1}$	$B_{n-1} + RK_{2n-1}$	$F_0\{A_{n-1} + RK_{2n-2}\}$	b_{n-1}	X
Flush	$a_n + B_{n-1} + WK_2 = A_n$	X	$F_1\{B_{n-1} + RK_{2n-1}\}$	a_n	X
	$b_n + A_{n-1} + WK_3 = B_n$	X	X	b_n	X
Unswap	X	X	X	0	$A_{n-1} = C_0$
	X	X	X	0	$A_n = C_1$
	P'_0	X	X	0	$B_{n-1} = C_2$
	$P'_1 + WK_0$	X	X	0	$B_n = C_3$
Nova 1ª	P'_2	$P'_0 + RK_0$	X	0	X
	$P'_3 + WK_1$	$P'_1 + RK_1$	$F_0\{P'_0 + RK_0\}$	0	X

Tabela 5 – Escalonamento de operações do processo de obtenção da chave intermediária L no modo 8-*branch*, onde $a'i$, $b'i$, $c'i$ e $d'i$ são resultados da Função-F.

Round	**Entrada do SRL16**	**Entrada da T-Box**	**Processamento na T-Box**	**Saída da T-Box**	**Estágio de Saída**
Init	X	X	X	0	X
	KL_0	X	X	0	X
	KL_1	X	X	0	X
	KL_2	X	X	0	X
	KL_3	X	X	0	X
	KR_0	X	X	0	X
	KR_1	X	X	0	X
1	KR_2	$KL_0 + RK_0$	X	0	X
	KR_3	$KL_2 + RK_1$	$F_0\{KL_0 + RK_0\}$	0	X
	X	$KR_0 + RK_2$	$F_1\{KL_2 + RK_1\}$	a_1	X
	X	$KR_2 + RK_3$	$F_0\{KR_0 + RK_2\}$	c_1	X
2	$a_1 + KL_1 = A_1$	$A_1 + RK_4$	$F_1(KR_2 + RK_3\}$	b_1	X
	$c_1 + KL_3 = C_1$	$C_1 + RK_5$	$F_0\{A_1 + RK_4\}$	d_1	X
	$b_1 + KR_1 = B_1$	$B_1 + RK_6$	$F_1\{C_1 + RK_5\}$	a_2	X
	$d_1 + KR_3 = D_1$	$D_1 + RK_7$	$F0_{\{}B_1 + RK_7\}$	c_2	X
3	$a_2 + KL_2 = A_2$	$A_2 + RK_8$	$F_1\{D_1 + RK_7\}$	b_2	X
	$c_2 + KR_0 = C_2$	$C_2 + RK_9$	$F_0\{A_2 + RK_8\}$	d_2	X
	$b_2 + KR_2 = B_2$	$B_2 + RK_{10}$	$F_1\{C_2 + RK_9\}$	a_3	X
	$d_2 + KL_0 = D_2$	$D_2 + RK_{11}$	$F_0\{B_2 + RK_{10}\}$	c_3	X
4	$a_3 + C_1 = A_3$	$A_3 + RK_{12}$	$F_1\{D_2 + RK_{11}\}$	b_3	X
	$c_3 + B_1 = C_3$	$C_3 + RK_{13}$	$F_0\{A_3 + RK_{12}\}$	d_3	X
	$b_3 + D_1 = B_3$	$B_3 + RK_{14}$	$F_1\{C_3 + RK_{13}\}$	a_4	X
	$d_3 + A_1 = D_3$	$D_3 + RK_{15}$	$F_0\{B_3 + RK_{14}\}$	c_4	X
5	$a_4 + C_2 = A_4$	$A_4 + RK_{12}$	$F_1\{D_3 + RK_{11}\}$	b4	X
	$c_4 + B_2 = C_4$	$C_4 + RK_{13}$	$F_0\{A_4 + RK_{12}\}$	d_4	X
	$b_4 + D_2 = B_4$	$B_4 + RK_{14}$	$F_1\{C_4 + RK_{13}\}$	a_5	X
	$d_4 + A_2 = D_4$	$D_4 + RK_{15}$	$F_0\{B_4 + RK_{14}\}$	c_5	X
...	...				
10	$a_9 + C_7 = A_9$	$A_9 + RK_{32}$	$F_1\{D_8 + RK_{31}\}$	b_9	X
	$c_9 + B_7 = C_9$	$C_9 + RK_{33}$	$F_0\{A_9 + RK_{32}\}$	d_9	X
	$b_9 + D_7 = B_9$	$B_9 + RK_{34}$	$F_1\{C_9 + RK_{33}\}$	a_{10}	X
	$d_9 + A_7 = D_9$	$D_9 + RK_{35}$	$F_0\{B_9 + RK_{34}\}$	c_{10}	X
Flush	$a_{10} + C_8 = A_{10}$	X	$F_1\{D_9 + RK_{35}\}$	b_{10}	X
	$c_{10} + B_8 = C_{10}$	X	X	d_{10}	X
	$b_{10} + D_8 = B_{10}$	X	X	X	X
	$d_{10} + A_8 = D_{10}$	X	X	X	X
Unswap	X	X	X	X	$A_9 = LL_0$
	X	X	X	X	$A_{10} = LL_1$
	X	X	X	X	$C_9 = LL_2$
	X	X	X	X	$C_{10} = LL_3$
	X	X	X	X	$B_9 = LR_0$
	X	X	X	X	$B_{10} = LR_1$
	X	X	X	X	$D_9 = LR_2$
	X	X	X	X	$D_{10} = LR_3$

Durante os últimos ciclos de clock do CLEFIA (*unswap round*), verifica-se que o estágio de 'Entrada do SRL16' não é utilizado. Se nenhuma reconfiguração for necessária, tal como mudança de chave ou modo de operação, um novo bloco de dados pode ser introduzido no registrador de deslocamento. Esta característica reduz o *overhead* de ciclos de clock entre operações, ao permitir que um novo bloco de cifra seja processado imediatamente após a finalização do anterior.

O escalonamento descrito na Tabela 4 representa também a sequência de obtenção da chave intermediária L. Sendo assim, quando a chave de cifra é de 128 bits, temos $n = 12$, $P = K_{128}$ e $C = L_{128}$. Além disso, o processo não introduz as *whitening keys* na entrada e na saída do circuito.

Quando configurado no modo 8-*branch* do CLEFIA (Tabela 5), cada iteração também é iniciada com o mapeamento do primeiro bloco de 32 bits da iteração correspondente. Nesse caso, o bloco encaminhado para a entrada da BRAM pode ser oriundo do registrador de deslocamento ou proveniente do estágio (4), através do registrador localizado na parte superior da Figura 15. Neste modo de operação, cada iteração parte do processamento de quatro cadeias de 32 bits dentro das T-*boxes*, correspondendo à realização da sequência alternada: $F_0,\ F_1,\ F_0,\ F_1$. Diferente da operação no modo de cifra, quando configurado no modo 8-*branch*, a estrutura proposta para o CLEFIA $GFN_{8,n}$ utiliza quatro ciclos por iteração.

Para garantir que não haja conflito de dados na entrada do registrador de deslocamento, seis pulsos de clock são utilizados para inicialização do circuito. Os quatro pulsos seguintes representam a primeira iteração, na qual os blocos iniciais ($KL_0,\ KL_2,\ KR_0$ e KR_2) são introduzidos na entrada da BRAM. Ainda na primeira iteração, dois *stalls* ocorrem, necessários ao atraso provocado pelo introdução dos registradores localizados na parte superior da Figura 15, utilizados no escalonamento das operações no modo 8-*branch*. Tomando como ponto de partida a segunda iteração, os blocos que mapeiam a BRAM passam a ser obtidos indiretamente, através do encaminhamento direto do estágio (4) operado com o bloco permutado, oriundo do registrador de deslocamento.

Chaves de codificação de 192 e 256 bits possuem a mesma latência, no que se refere ao cômputo sobre $GFN_{8,n}$. Ao todo, um conjunto formado por 40 "*round keys*" ($RK_i, 0 \leq i < 40$) diferentes é utilizado, em ambos os casos.

Na produção da chave intermediária L de 256 bits, também é necessário esvaziar o *data path*, antes de emitir os dados para a saída do circuito. Este processo requer quatro ciclos de clock adicionais, para introduzir as quatro últimas palavras ($a_{10},\ c_{10},\ b_{10},\ d_{10}$) no registrador de deslocamento. Por fim, os blocos são enviados, em ordem, para a saída do circuito. As Tabelas 4 e 5 estão replicadas, em sua integridade, nos Apêndices A e 17, juntamente com o escalonamento de endereços dos registradores de deslocamento.

5.3.3 Obtenção das *Round Keys*

Uma vez computada, a chave intermediária L precisa ser processada através de uma estrutura iterativa que encaminha o resultado para o registrador de deslocamento. A estrutura de escalonamento apresentada na Tabela 6 descreve o ciclo de permutação utilizado para reduzir o tamanho do *data path*, assim como o número de *stalls* necessários à sua realização. Na descrição das operações, os valores de B e C correspondem aos barramentos de entrada, destacados na Figura 18. A expressão C_{high} indica que estão sendo manipulados os 25 bits da parte alta de C - mesma analogia para C_{low}, considerando a parte baixa do mesmo bloco.

Tabela 6 – Escalonamento da permutação realizada pela função Σ.

#	**Operação**
1	$\Sigma_3 = B_{low} \mid C_{high}$
2	$\Sigma_1 = C_{low} \mid B_{low}$
3	$\Sigma_0 = C_{low} \mid B_{high}$
4	$\Sigma_2 = B_{high} \mid C_{high}$

A cada ciclo de clock, a função Σ realiza uma combinação da cadeia de bits de entrada, no sentido de formar um bloco de 32 bits na saída do circuito. Cada bloco corresponde à um valor Σ_i correspondente ($0 \leq i < 4$). Dessa forma, quatro ciclos de clock são necessários para realizar a função Σ. Veremos adiante que, com o escalonamento apropriado, é possível reduzir o *overhead* de ciclos de clock imposto por esta combinação de operações.

O escalonamento de operações para expansão de chaves do CLEFIA-128 e CLEFIA-256 podem ser encontrados nas Tabelas 7 e 8. Nesta estrutura, o processo é iniciado a partir da introdução dos blocos de 32 bits da chave intermediária L, oriundos da estrutura $GFN_{8/4,n}$. Na expansão de chaves, consideramos uma iteração o processo de obtenção de uma sequência de quatro *round keys*, oriundas do processamento da chave L ou LL e LR e, alternadamente, de K ou KL e KR. Cada iteração tem início ou a partir do endereçamento dos blocos de 32 bits da chave intermediária L, ou da realimentação do valor de $\Sigma(L)$, ambos através do registrador de deslocamento. Uma vez introduzido no registrador de saída do estágio de entrada, no ciclo seguinte, a *round key* é computada e armazenada na memória de chaves.

O cálculo de cada *round key* requer apenas um ciclo de clock, tanto no modo CLEFIA-128 quanto no CLEFIA-256. No CLEFIA-128, a primeira iteração começa três ciclos após a inicialização do circuito, tendo em vista o atraso inicial associado à alimentação das estruturas de armazenamento do *data path*. Ao final de cada iteração, três ciclos adicionais são necessários para finalização do cômputo de $\Sigma(L^j)$. Estes ciclos resultam na introdução de três *stalls* na execução de cada iteração, com excessão da primeira, que necessita apenas de dois *stalls*.

Tabela 7 – Escalonamento de operações da expansão de chave de cifra do CLEFIA-128, onde Σ_j^i representa o bloco resultante da função Σ correspondente à iteração i.

Ciclo	**Entrada do SRL32**	**Saída do SRL32**	**Entrada da BRAM**
1	L_0	X	X
2	L_1	L_0	X
3	L_2	L_1	$RK_0 = L_0 + CON_0$
4	L_3	L_2	$RK_1 = L_1 + CON_1$
5	Σ_3^1	L_3	$RK_2 = L_2 + CON_2$
6	Σ_1^1	L_1	$RK_3 = L_3 + CON_3$
7	Σ_0^1	L_0	X
8	Σ_2^1	L_2	X
9	X	Σ_0^1	$RK_4 = K_0 + \Sigma_0^1 + CON_4$
10	X	Σ_1^1	$RK_5 = K_1 + \Sigma_1^1 + CON_5$
11	X	Σ_2^1	$RK_6 = K_2 + \Sigma_2^1 + CON_6$
12	Σ_0^2	Σ_3^1	$RK_7 = K3 + \Sigma_3^1 + CON_7$
13	Σ_2^2	Σ_1^1	X
14	Σ_3^2	Σ_0^1	X
15	Σ_1^2	Σ_2^1	X
...	...		
51	X	Σ_0^7	$RK_{28} = K_0 + \Sigma_0^7 + CON_{28}$
52	X	Σ_1^7	$RK_{29} = K_1 + \Sigma_1^7 + CON_{29}$
53	X	Σ_2^7	$RK_{30} = K_2 + \Sigma_2^7 + CON_{30}$
54	Σ_0^8	Σ_3^7	$RK_{31} = K3 + \Sigma_3^7 + CON_{31}$
55	Σ_2^8	Σ_1^7	X
56	Σ_3^8	Σ_0^7	X
57	Σ_1^8	Σ_2^7	X
58	X	Σ_0^8	$RK_{32} = \Sigma_0^8 + CON_{32}$
59	X	Σ_1^8	$RK_{33} = \Sigma_1^8 + CON_{33}$
60	X	Σ_2^8	$RK_{34} = \Sigma_2^8 + CON_{34}$
61	X	Σ_3^8	$RK_{35} = \Sigma_3^8 + CON_{35}$

No CLEFIA-256 (Tabela 8), a primeira iteração inicia a partir do 7º ciclo de clock. Estes ciclos adicionais são utilizados para alimentação do circuito e escalonamento da função $\Sigma(LL^1)$. Poderia-se iniciar a primeira iteração já no 3º ciclo de clock - contudo, seria necessário aguardar a finalização da alimentação do circuito e a execução da função Σ sobre LL^1 para retomar o processo. Verifica-se, ainda na Tabela 8, que não há *stalls* entre as duas últimas iterações, uma vez que não é necessário realizar Σ sobre ΣR^{k-1}.

A diferença no que se refere ao tamanho da chave, no modo de operação CLEFIA-256, reside no número de iterações e total de constantes utilizadas. Na descrição da Tabela 8, $j = 44$ para chaves de 192 bits, ao passo que, para chaves de 256 bits, $j = 52$. A realização do processo de expansão de chaves leva $n = 77$ e $n = 90$ ciclos de clock para chaves de 192 e 256 bits, respectivamente. Ambas as Tabelas 7 e 8 estão replicadas nos Apêndices C e D, juntamente com os detalhes do escalonamento dos registradores de deslocamento.

Tabela 8 – Escalonamento de operações da expansão de chave de cifra do CLEFIA-256., onde Σ_y^k corresponde ao bloco resultante da função Σ correspondente à iteração k e n especifica o total de ciclos de clock.

Ciclo	**Entrada do SRL32**	**Saída do SRL32**	**Entrada da BRAM**
1	LL_0	X	X
2	LL_1	X	X
3	LL_2	X	X
4	LL_3	X	X
5	LR_0	X	X
6	LR_1	LL_0	X
7	LR_2	LL_1	$RK_0 = LL_0 + CON_0^i$
8	LR_3	LL_2	$RK_1 = LL_1 + CON_1^i$
9	ΣL_3^1	LL_3	$RK_2 = LL_2 + CON_2^i$
10	ΣL_1^1	LL_1	$RK_3 = LL_3 + CON_3^i$
11	ΣL_0^1	LL_0	X
12	ΣL_2^1	LL_2	X
13	X	ΣL_0^1	X
14	X	ΣL_1^1	$RK_4 = KR_0 + \Sigma L_0^1 + CON_4^i$
15	X	ΣL_2^1	$RK_5 = KR_1 + \Sigma L_1^1 + CON_5^i$
16	ΣL_3^2	ΣL_3^1	$RK_6 = KR_2 + \Sigma L_2^1 + CON_6^i$
17	ΣL_1^2	ΣL_1^1	$RK_7 = KR_3 + \Sigma L_3^1 + CON_7^i$
18	ΣL_0^2	ΣL_0^1	X
19	ΣL_2^2	ΣL_2^1	X
20	X	LR_0	X
21	X	LR_1	$RK_8 = LR_0 + CON_8^i$
22	X	LR_2	$RK_9 = LR_1 + CON_9^i$
23	ΣR_3^1	LR_3	$RK_{10} = LR_2 + CON_{10}^i$
24	ΣR_1^1	LR_1	$RK_{11} = LR_3 + CON_{11}^i$
25	ΣR_0^1	LR_0	X
26	ΣR_2^1	LR_2	X
27	X	ΣR_0^1	X
28	X	ΣR_1^1	$RK_{12} = KL_0 + \Sigma R_0^1 + CON_{12}^i$
29	X	ΣR_2^1	$RK_{13} = KL_1 + \Sigma R_1^1 + CON_{13}^i$
30	ΣR_3^2	ΣR_3^1	$RK_{14} = KL_2 + \Sigma R_2^1 + CON_{14}^i$
31	ΣR_1^2	ΣR_1^1	$RK_{15} = KL_3 + \Sigma R_3^1 + CON_{15}^i$
...	...		...
$n-7$	X	ΣL_1^k	$RK_{36} = KR_0 + \Sigma L_0^k + CON_{j-8}^i$
$n-6$	X	ΣL_2^k	$RK_{37} = KR_1 + \Sigma L_1^k + CON_{j-7}^i$
$n-5$	X	ΣL_3^k	$RK_{38} = KR_2 + \Sigma L_2^k + CON_{j-6}^i$
$n-4$	X	ΣR_0^{k-1}	$RK_{39} = KR_3 + \Sigma L_3^k + CON_{j-5}^i$
$n-3$	X	ΣR_1^{k-1}	$RK_{40} = \Sigma R_0^{k-1} + CON_{j-4}^i$
$n-2$	X	ΣR_2^{k-1}	$RK_{41} = \Sigma R_1^{k-1} + CON_{j-3}^i$
$n-1$	X	ΣR_3^{k-1}	$RK_{42} = \Sigma R_2^{k-1} + CON_{j-2}^i$
n	X	X	$RK_{43} = \Sigma R_3^{k-1} + CON_{j-1}^i$

5.4 UNIDADES DE CONTROLE E CIRCUITO FINAL

Uma vez concluída a descrição do *data path* para ambas as arquiteturas de cifra e expansão de chave do CLEFIA, resta agora descrever as suas respectivas Unidades de Controle (UC). A arquitetura proposta prevê Unidades de Controle distribuídas, tendo em vista que o mecanismo de cifra pode ser utilizado independente da expansão de chaves, quando considerado o pré-cômputo das *round keys* e *whitening key* fora do *chip*. Esta seção apresenta uma visão em nível de interface dos respectivos circuitos, com ênfase na demonstração dos sinais externos e na distribuição das conexões internas em cada unidade.

5.4.1 Estrutura Feistel 8/4-branch

O circuito completo para a estrutura $GFN_{4/8,n}$ é apresentado na Figura 20, na qual os terminais destacados com linhas tracejadas são implementados apenas quando em conjunto com a implementação do circuito de expansão de chave. A Unidade de Controle do circuito foi projetada como uma FSM (*Finite State Machine*) de cinco estados. O processo baseia-se no uso de dois contadores: o primeiro para contar a quantidade de iterações computadas; e o segundo para registrar o número de ciclos de clock ocorridos dentro de um estado. O processo é realizado através do fluxograma descrito na Figura 21.

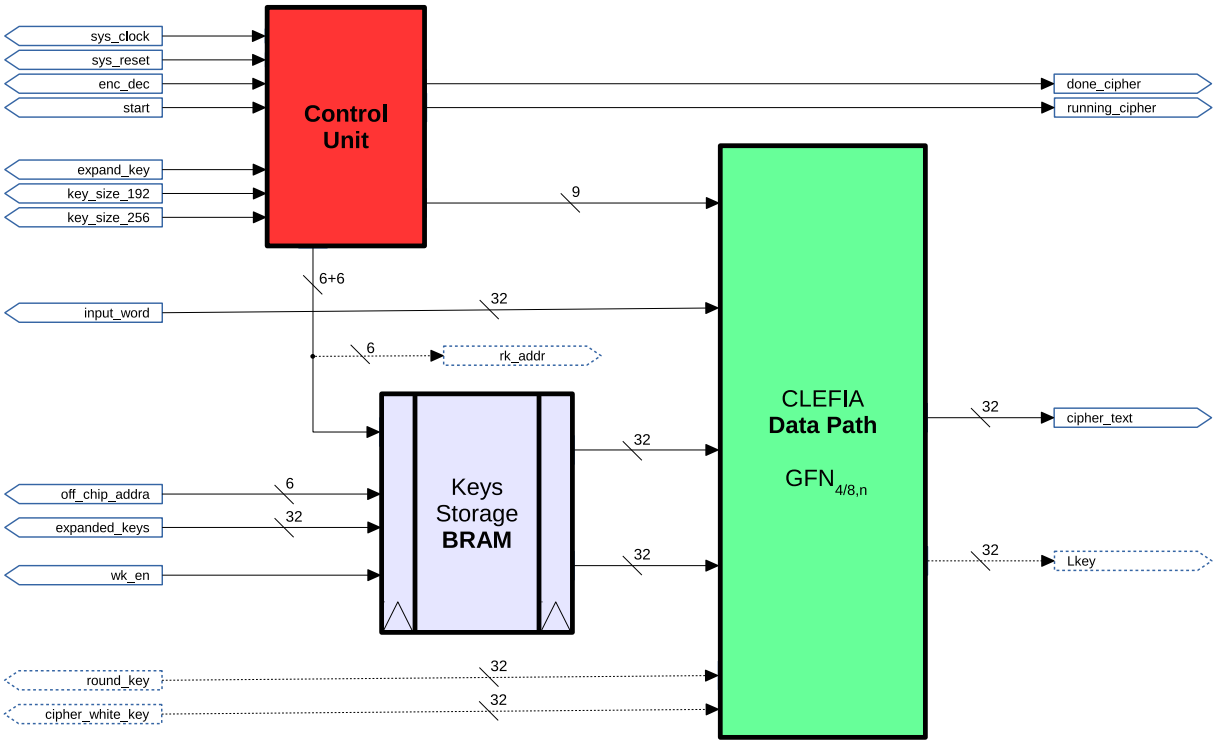


Figura 20 – Representação do Co-processador de cifra do CLEFIA.

Ao iniciar o processo de cifra, pela primeira vez, a UC deve ser reinicializada para seu estado inicial (‘idle’), através do sinal ‘sys_reset’, no qual é esperado que o

usuário introduza os parâmetros de configuração. Os parâmetros de configuração devem ser introduzidos apenas no estado inicial. Ao todo, existem quatro parâmetros de configuração, oriundos de sinais externos e definidos pelo usuário, que determinam o modo ('enc_dec' e 'expand_key') e o tamanho da chave. Para este último, os sinais 'key_size_192' e 'key_size_256' determinam o tamanho da chave, de modo que o último tem prioridade sobre o primeiro, caso os dois estejam ativos simultaneamente. Quando ambos os sinais estiverem inativos, o circuito assume a configuração padrão para chaves de 128 bits. A combinação dos sinais que determina o tamanho de chave, associada à ativação da entrada 'expand_key' configura a execução no sentido da geração da chave intermediária L.

O diagrama apresentado na Figura 20 considera, ainda, a presença de uma memória *dual-port*, a qual armazena ambos os conjuntos de *round keys* e blocos de 32 bits da *whitening key* pré-computados. Neste caso, antes de inicializar o circuito, as chaves já devem ter sido introduzidas na memória e nenhuma reconfiguração deve ocorrer a partir de então. Este modo de operação só é considerado na ausência do mecanismo de expansão de chave. Por não se tratar do foco deste trabalho, o processo de configuração e carga desta memória não será discutido.

O primeiro estado da UC representa o estágio de preparação/extração dos blocos de entrada e execução da primeira iteração. Quando configurado para expansão de chave, a transição para o próximo estado está associada à carga da chave de cifra e geração das *whitening keys*. O primeiro ciclo de clock é utilizado para inicializar os registradores de saída da Unidade de Controle.

Durante o processo de cifra, os quatro primeiros ciclos de clock são usados para carregar o registrador de deslocamento, a partir da extração dos blocos de 32 bits do texto plano ou da cadeia cifrada. Apesar disso, o processo de codificação pode ser inicializado a partir do segundo ciclo de clock. Os dois ciclos seguintes são usados para execução da primeira iteração, uma vez que, neste caso, o endereçamento do registrador de deslocamento é exclusivamente diferente (ver Apêndice A). Na expansão de chaves de 192 ou 256 bits, este estado leva 17 ciclos de clock para ser completado. Os nove primeiros são utilizados para extração dos 8 blocos de 32 bits da entrada de chave 'white_cipher_key' e o cômputo da primeira iteração. Os 8 ciclos adicionais deste estado são utilizados para computar a segunda e terceira iterações. Assim como na cifra do CLEFIA, este modo de operação não requer que toda a cadeia de entrada seja introduzida para inicializar. Dessa forma, no modo 8-*branch*, o processamento pode ser inicializado a partir do quarto ciclo de clock.

Uma vez finalizada a primeira iteração, a UC inicia o processamento das iterações intermediárias sequencialmente, através dos estados 'loop0' e 'loop1', ambos realizados em um ciclo de clock. Ao final do 'loop0' o contador de iterações é incrementado. Se a iteração final for atingida, a UC interrompe a execução dos estados recursivos e transita para o estado final ('flush_unswap').

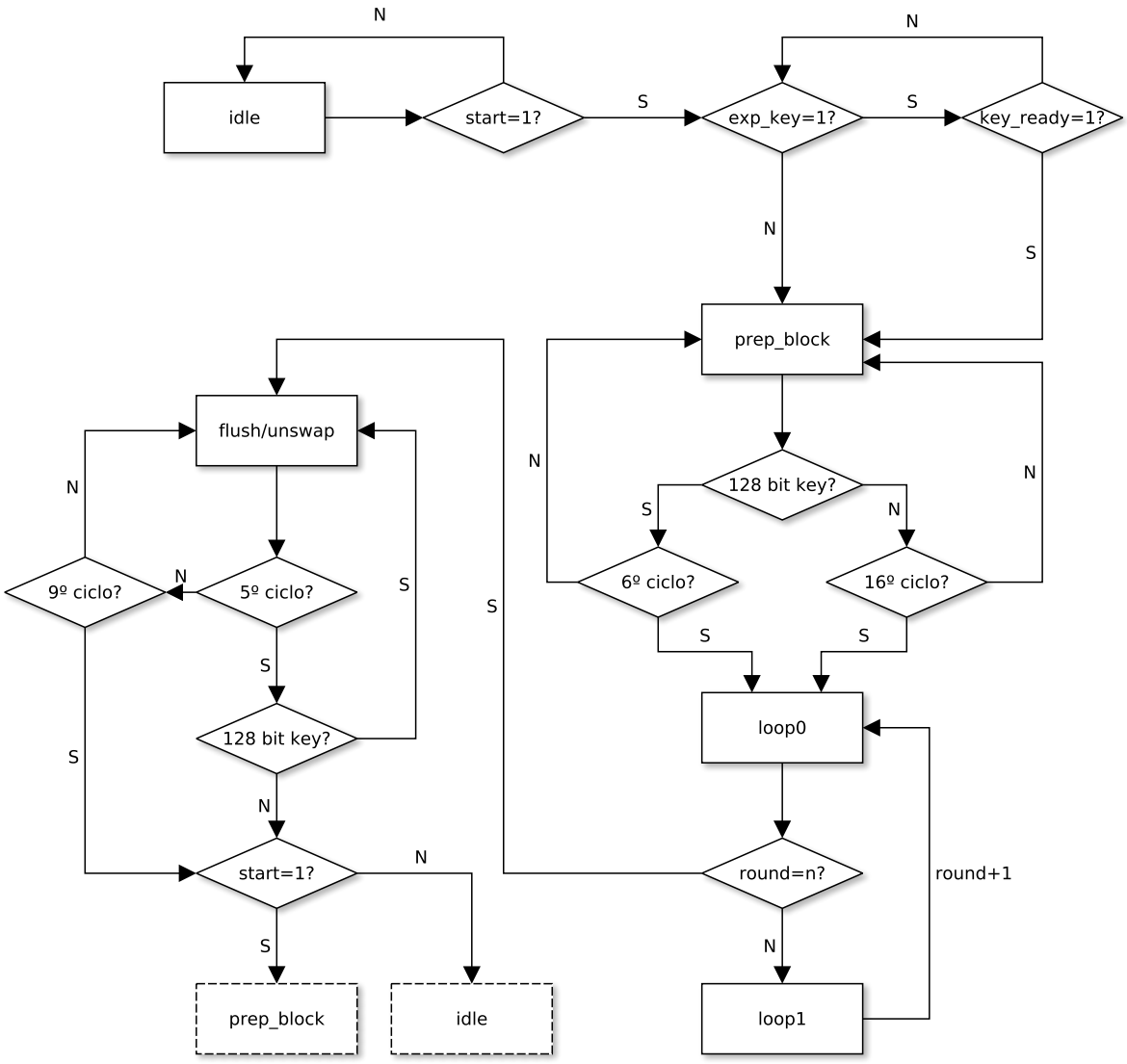


Figura 21 – Fluxograma da Máquina de Estados para a Unidade de Controle do Co-processador de Cifra do CLEFIA.

O estado 'flush_swap' é realizado durante 5 ou 9 ciclos de clock para os modos 4-*branch* e 8-*branch*, respectivamente. Neste estado, o pipeline é esvaziado à medida em que as duas últimas *whitening keys* são introduzidas e somadas às palavras de saída. Para indicar que a última palavra foi transmitida para o registrador de saída, o sinal 'done' é ativado no último ciclo de clock.

Uma vez finalizada a execução do estado 'flush_unswap' na codificação/decodificação, o circuito está pronto para receber um novo bloco de dados (4×32 bits). Entretanto, para reduzir o número de ciclos durante esta transição, sem implicar em conflitos de dados, uma nova cadeia de bits pode ser introduzida no terceiro ciclo do estado final. Este recurso só pode ser utilizado se os parâmetros de configuração não forem alterados. Se nenhum outro bloco estiver pronto para ser introduzido no circuito, ou se houver a necessidade

de reconfiguração dos parâmetros, ao final da execução o circuito retorna para o estado 'idle'. O circuito saltará para este estado também, a qualquer momento, se a execução for interrompida.

5.4.2 Mecanismo de Expansão de Chaves

O circuito que descreve a interface dos componentes associados ao mecanismo de expansão de chaves é apresentado na Figura 22. Como descrito anteriormente, a estrutura projetada para o co-processador do CLEFIA é composta por duas Unidades de Controle descentralizadas. No sentido de estabelecer a sincronia entre os circuitos, as UCs compartilham dos mesmos parâmetros de configuração. Assim como no núcleo de processamento do CLEFIA, a Unidade de Controle da expansão de chaves foi implementada sob a perspectiva de uma máquina de estados. A estrutura compacta para implementação do mecanismo de expansão de chave do CLEFIA conduz ao aumento da complexidade do seu elemento de controle. Diante disso, a Unidade de Controle deste módulo foi projetada de acordo com os dois estágios de implementação da expansão de chaves descritos anteriormente: (i) extração e preparação da chave de entrada; e (ii) geração das *round keys* correspondentes.

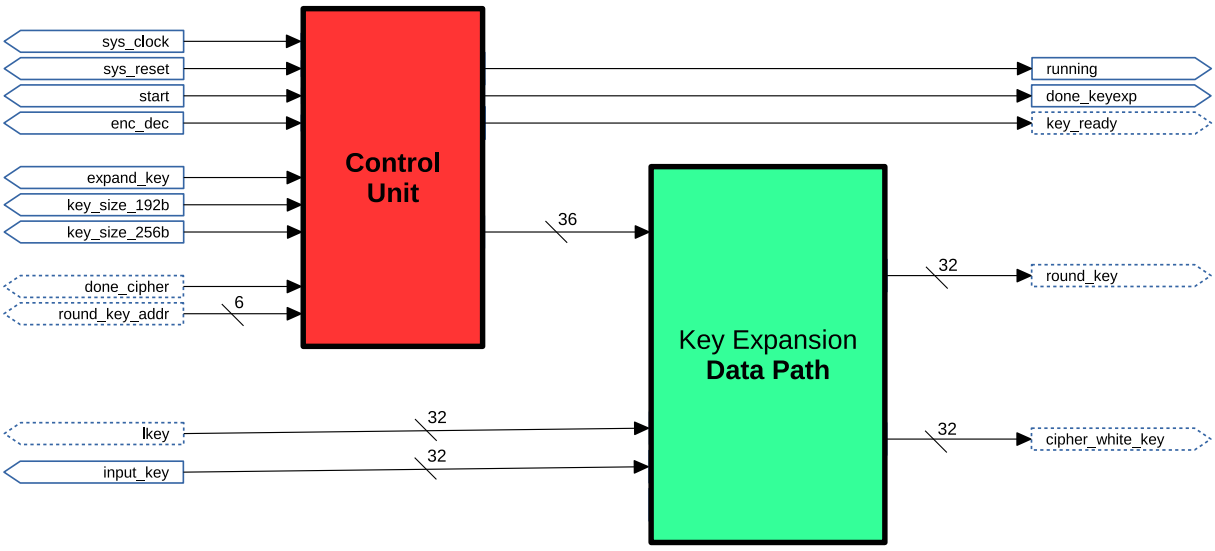


Figura 22 – Representação do Circuito Final de Expansão de Chaves do CLEFIA.

A máquina de estados é composta por 14 estados, dos quais cinco estão associados exclusivamente à extração da cadeia de entrada, obtenção da *whitening key* e transmissão da chave para o módulo de cifra. Os demais estados referem-se aos processos associados ao cômputo do conjunto de *round keys*. Tendo em vista facilitar a compreensão do desdobramento de operações deste circuito, cada estágio de implementação foi representado na forma de um fluxograma e apresentados nas Figuras 23 e 24.

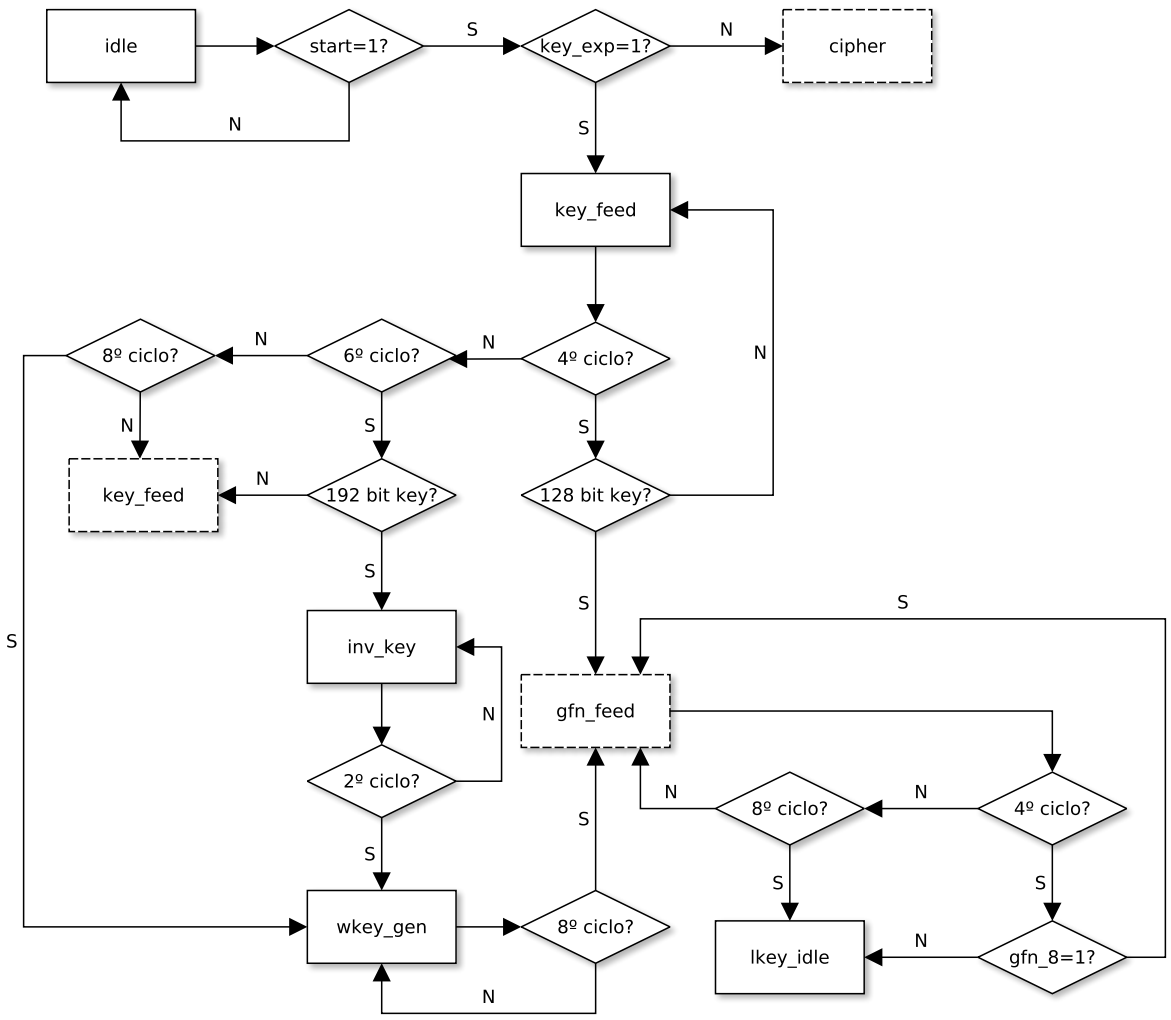


Figura 23 – Fluxograma da máquina de estados para a Unidade de Controle do mecanismo de Expansão de Chave do CLEFIA referente à introdução da chave de entrada.

Antes de iniciar o processo de configuração, a Unidade de Controle deve ser reinicializada, por meio da ativação do sinal de *reset*. Isso fará com que a máquina de estados retorne ao seu estado inicial ('idle'). Este estado também é utilizado para sincronizar as Unidades de Controle de cifra e expansão, quando operando somente no modo de cifra. Os parâmetros de configuração da expansão de chave devem ser definidos antes de inicializar o circuito.

A transição entre os estados da UC é controlada a partir de dois contadores. O primeiro contabiliza o número de ciclos de clock ocorridos dentro de um estado. O segundo é utilizado para registrar a quantidade de iterações realizadas. Na implementação deste último, uma iteração representa a obtenção de uma *round key*, correspondente a cada bloco de 32 bits das chaves de L ou LL e LR de 128 bits permutadas. Dessa forma, o contador de iterações pôde ser utilizado na composição do endereço do segmento de memória de

chaves. Adicionalmente, um terceiro contador é responsável por mapear o segmento de endereços associado às constantes correspondentes.

O endereço das chaves utilizadas no processo de obtenção da chave L utiliza o mesmo barramento responsável por requisitar as *round keys* da cifra. A definição do espaço de endereços é estebalecida de acordo com os parâmetros de configuração, novamente, com 'key_size_256' prioritário à 'key_size_192', se ambos os sinais estiverem ativos. Dois bits adicionais são acrescentados ao endereço de chave, com o objetivo de selecionar entre os quatro segmentos de chave presentes na memória. O mesmo acontece para o contador de endereços de constantes. A organização da memória de chaves foi explorada na Seção 5.2.3, e representada na Figura 19.

O estado 'key_feed', representado na Figura 23, é reponsável pelo processo de extração dos blocos de 32 bits que compõem a chave de codificação. Este estado pode ocupar 4, 6 ou 8 ciclos de clock, de acordo com o tamanho da chave de codificação. Durante a expansão de chaves de 192 bits, um estado adicional é acionado. No estado 'inv_key', as duas primeiras palavras da chave de cifra (KL_0 e KL_1) são negadas, compondo assim os 64 bits da parte baixa da chave (KR_2 e KR_3). Além de compor a chave que alimenta a entrada do circuito GFN, estas palavras são necessárias para o cômputo da *whitening key*.

O estado 'wkey_gen' e destinado à obtenção dos blocos de 32 bits da *whitening key*. Para computar os quatro blocos que compõem a chave, este estado é realizado ao longo de oito ciclos de clock. Quando configurado para expandir chaves de 128 bits, este processo não é necessário e a UC transita direto para o estado 'gfn_feed'. Este estado pode ser implementado em 4 ou 8 ciclos de clock, de acordo com o tamanho da chave. Uma vez finalizado o envio da chave K^{128} ou K^{256} para a saída do circuito, a UC permanece em um estado de espera ('lkey_idle'), até a conclusão do cômputo da chave L.

O fluxograma apresentado na Figura 24 corresponde ao fluxo de operações para obtenção das *round keys*. Nos estados a seguir, o primeiro ciclo é utilizado para inicializar o registrador localizado na saída do SRL32. O controle de inicialização do circuito está vinculado ao sinal de conclusão do processo de cifra. Este sinal interno indica que o primeiro bloco de 32 bits da chave L foi produzido, estando pronto para aquisição pelo registrador de deslocamento no ciclo seguinte. O controle do fluxo de entrada é realizado no estado 'lkey_init', levando 8 ou 12 ciclos de clock para chaves L de 128 e 256 bits, respectivamente. Além de controlar a entrada de dados, neste estado é realizado o primeiro bloco de quatro iterações, as quais são iniciadas no segundo ou sétimo ciclos. Os ciclos adicionais são utilizados para a realização da função Σ e a realimentação do registrador de deslocamento.

As operações recursivas, que envolvem o cálculo das *round keys* subsequentes e a operação sobre a função Σ, são iniciadas no estado 'loop0_sigma'. Quando consideradas chaves de 128 bits, a UC permanece neste estado até que a quantidade correspondente ao

número de *round keys* ($n = 36$) seja atingido. Por outro lado, na presença de cadeias de entrada de 256 bits, um passo adicional é necessário. Após o cálculo das oito primeiras chaves, obtidas em função de LL e $\Sigma(LL)$, um estado adicional é ativado, no sentido de adequar o escalonamento de endereços do registrador de deslocamento. Dessa forma, no estado 'init_lr_sigma', são produzidas as *round keys* oriundas da operação sobre a chave LR não permutada. Uma vez finalizada a operação de permutação, a UC retorna para o ciclo recursivo formado a partir dos estados 'loop0_sigma' e 'loop1_sigma', os quais operam alternadamente sobre LL e LR. Detalhes sobre o escalonamento realizado dentro destes estados são fornecidos no Apêndice D.

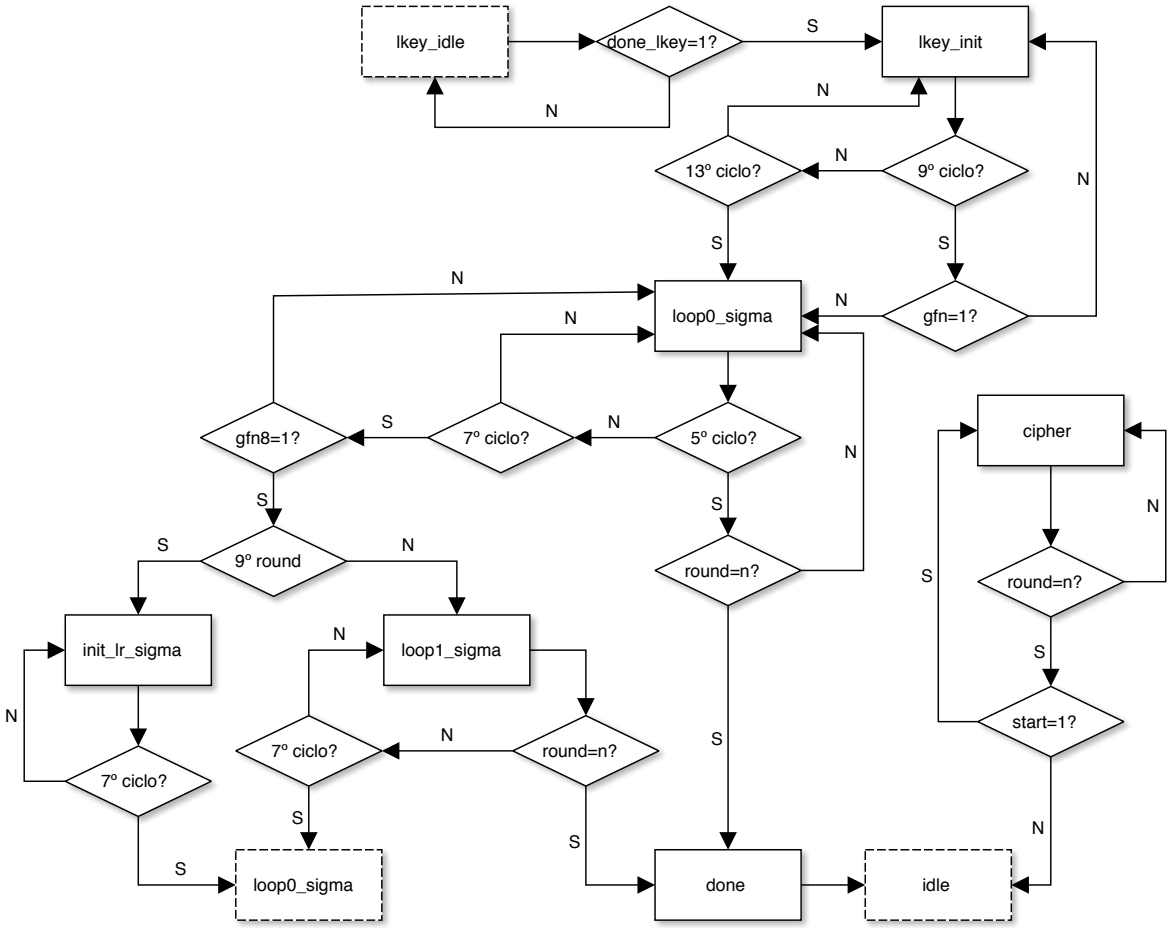


Figura 24 – Fluxograma da máquina de estados para a Unidade de Controle do mecanismo de Expansão de Chave do CLEFIA referente à geração das *Round Keys*.

Uma vez que o valor de n corresponda ao total de *round keys* esperado (44 ou 52), a UC transita para o estado de finalização ('done'). Este estado é responsável por transmitir um sinal para a saída do circuito, por um período igual a um ciclo de clock, indicando que o processo de expansão foi finalizado e que uma nova cifra pode ser inicializada. Por fim, a máquina de estados retorna para o estado 'idle', onde aguardará por uma nova configuração.

Como dito anteriormente, o estado inicial da UC é também utilizado para sincronizar os elementos de controle. A sincronização é estabelecida tendo em vista fornecer as *whitening keys* no ciclo correspondente. Este processo é conduzido durante o estado 'cipher', o qual replica os parâmetros referentes às iterações do CLEFIA e utiliza o mesmo contador de iterações da expansão de chave, para identificar quando mapear uma *whitening key*. A saída deste estado também depende do sinal 'enc_dec' que, quando ativo, indica que uma decodificação está a ser computada.

5.4.3 Co-processador de Cifra com Expansão de Chave

Uma vez finalizada a descrição dos circuitos de cifra e de expansão de chaves, esta seção descreve como ambas as estruturas estão associadas. A interface entre os circuitos que compõem o co-processador de cifra com expansão de chave é dada na Figura 25. Tal interface foi projetada de forma a não apresentar lógica intermediária, sendo utilizada apenas para interligar os sinais provenientes das saídas de ambos os circuitos. Esta característica propicia o desacoplamento das unidades, principalmente quando considerada a implementação do co-processador de cifra do CLEFIA, com expansão de chave fora do *chip*.

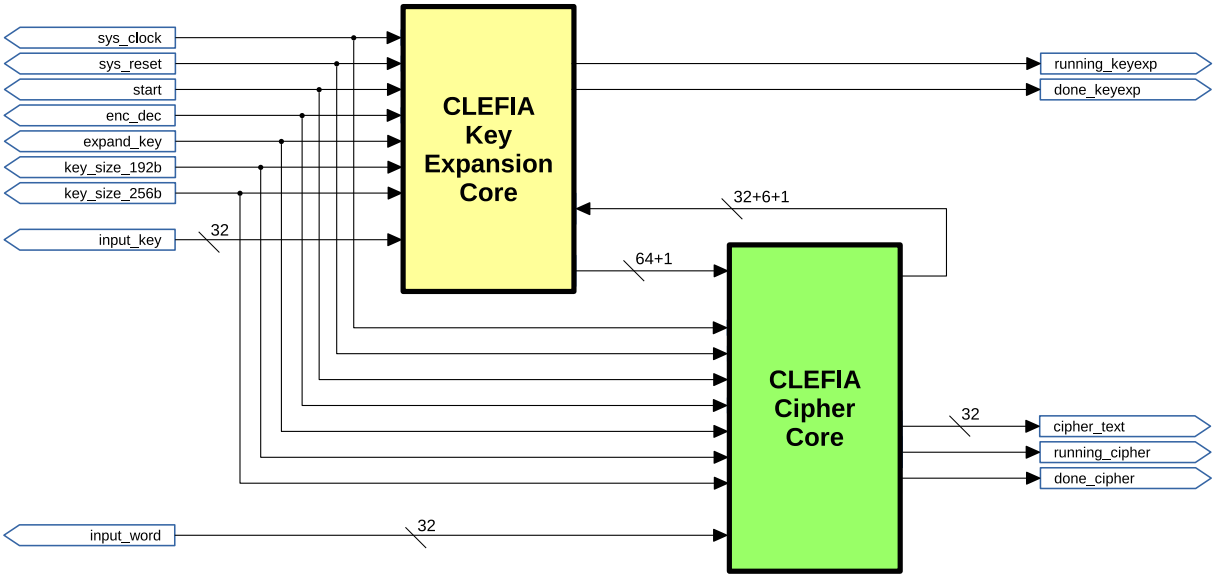


Figura 25 – Representação do co-processador de cifra do CLEFIA com Expansão de Chave.

A partir da análise das Figuras 20 e 22, verifica-se que apenas três sinais de controle são interfaceados entre os blocos. Na direção do bloco de cifra, apenas o sinal que indica a conclusão do cômputo das *whitening keys* é necessário. Adicionalmente, 64 bits de dados, oriundos no módulo de expansão de chaves, são transmitidos para o núcleo de cifra, referentes a: *round key*; e a *whitening key* ou chave de codificação. Considerando o sentido inverso, dois sinais de controle são necessários para a condução do processo de expansão.

O primeiro corresponde à uma cadeia de 6 bits que irá compor o endereço da memória de chaves. O segundo é oriundo do sinal de saída 'done_cipher'. Este sinal é utilizado para indicar a conclusão da obtenção da chave L, transmitida para o módulo de expansão através de um barramento interno de 32 bits.

5.5 CONSIDERAÇÕES FINAIS SOBRE A IMPLEMENTAÇÃO DA ARQUITETURA

Este capítulo tratou sobre a arquitetura e implementação do co-processador de cifra com expansão de chaves completa para o CLEFIA. O projeto da arquitetura foi cuidadosamente adaptado às limitações dos *chips* FPGA Xilinx Virtex 5, minimizando assim a utilização de blocos lógicos por meio do aproveitamento dos recursos internos à arquitetura deste dispositivo. Ambas as estruturas foram implementadas com base em uma arquitetura de 32 bits em pipeline de dois estágios. Dessa forma, na cifra, uma iteração é realizada em dois ciclos de clock. O processo de expansão de chaves é mais complexo e opera com latência igual a sete ciclos de clock por iteração, com a introdução de três *stalls* no caminho de dados.

O núcleo de cifra é formado por uma Unidade de Controle composta de cinco estados. Adicionalmente, este estudo sugere o desacoplamento do núcleo de cifra, a partir do uso de uma memória *dual-port* (incluindo a respectiva lógica de interface), usada para armazenar e prover *round keys* e *whitening key* pré-computadas fora do *chip*. Por possuir uma interface independente, a unidade de codificação proposta pode ser implementada em um ambiente onde a expansão de chave pode ser definida externamente.

O módulo de expansão de chave de cifra é controlado por uma máquina de estados projetada para operar com doze estados. Neste caso, a memória sugerida para o núcleo de cifra é descartada e, em seu lugar, utilizada uma memória compartilhada para prover todas as chaves e constantes. Adicionalmente, as *whitening keys* passam a ser provenientes de um registrador de deslocamento.

A estrutura completa descrita até aqui apresenta uma interface de entrada e saída simplificada no sentido de estabelecer uma interface com outras unidades de procesamento. Neste sentido, além de poder ser implementada na forma de um circuito específico, de baixo custo, a estrutura proposta pode ser configurada como um co-processador de cifra ou acelerador em um ambiente de computação de alto desempenho. Por fim, por ser voltada para arquiteturas reconfiguráveis, este co-processador pode facilmente ser integrado à um ambiente de co-projeto hardware software baseado em processadores embutidos nos *chips* FPGA, como é o caso do MicroBlaze em dispositivos da Xilinx.

O próximo capítulo apresenta uma análise de eficiência de cada um dos componentes do co-processador proposto neste trabalho, relacionando-o com as principais implementações presentes no estado da arte.

6 ANÁLISE DOS RESULTADOS

Nesta seção são apresentados os resultados experimentais, em nível de síntese física para dispositivos FPGA, das estruturas propostas e uma análise em relação aos principais trabalhos presentes no Estado da Arte. Os resultados foram obtidos utilizando a ferramenta Xilinx ISE Design Suite (v14.7), com o circuito descrito em VHDL. Os valores apresentados foram obtidos após o processo de *Place & Route*, utilizando os parâmetros padrão da ferramenta, a saber: síntese configurada com *Normal Speed Optimization Effort*; *High Optmization Effort (Area Targeted)* no processo de *Mapping*; e *No Extra Effort* na ferramenta de *Place & Route*.

Os testes funcionais foram realizados a partir de um ambiente de verificação dotado de um conjunto de vetores de teste extraídos do modelo de referência disponível no sítio oficial do algoritmo CLEFIA. Na análise funcional dos blocos de cifra e de expansão de chaves, foram verificadas as saídas dos circuitos em função de um conjunto de vetores de entrada e os barramentos de dados internos. No que diz respeito à validação do co-processador integrado, a ênfase dos testes foi direcionada para os sinais de controle que viabilizam a integração no sentido de garantir a ausência de conflitos de dados e confirmar o escalonamento proposto. Como forma de validação final, um conjunto de testes foi aplicado à estrutura pós *Place & Route* do circuito, em um processo também chamado de *gate level simulation*.

O trabalho aqui apresentado foi proposto com vistas a explorar características e recursos presentes nos dispositivos da família Virtex 5 da Xilinx. No sentido de apresentar uma comparação confiável, as estruturas propostas em (PROENÇA; CHAVES, 2011; RESENDE; CHAVES, 2015; CHAVES, 2013) foram remapeadas para o dispositivo Virtex 5 “xc5vlx30”, utilizando exatamente as mesmas restrições. Os dados referentes às propostas apresentadas por Kryjak e Gorgón (2009) e Hanley e O’Neill (2012) foram extraídos diretamente dos resultados apresentados pelos respectivos autores.

Para avaliar a viabilidade da implementação das arquiteturas propostas em FPGAs de baixo custo, o dispositivo Xilinx Spartan 3E foi escolhido. Adicionalmente, a análise das arquiteturas foi estendida também para os dispositivos Xilinx Virtex 6. Na primeira parte desta análise, apenas a estrutura principal do CLEFIA é considerada, sem a presença do mecanismo de expansão de chaves.

6.1 ANÁLISE DO NÚCLEO DE CIFRA DO CLEFIA

A Tabela 9 apresenta um levantamento de requisitos em termos de recursos do dispositivo FPGA, considerando separadamente a Unidade de Controle e o *Data Path*,

assim como a implementação considerando apenas o núcleo de cifra. Os modelos com iniciais 'xc3s', 'xc5v' e 'xc6v' correspondem aos dispositivos FPGA Xilinx Spartan 3E, Virtex 5 e Virtex 6, respectivamente.

Tabela 9 – Utilização de recursos do Co-processador de cifra.

	Dispositivo	**SLICEs**	**LUTs**	**FFs**	**RAMBs**	**Freq$_{max}$ (MHz)**
Unidade de Controle	xc3s1200e-4	67	131	26	0	157,2
	xc5vlx30-3	26	75	29	0	460,2
	xc6vlx75t-3	41	77	30	0	538,5
Data Path	xc3s1200e-4	224	256	96	2	156,6
	xc5vlx30-3	71	128	96	2	342,7
	xc6vlx75t-3	168	416	96	2	328,5
Núcleo de Cifra	xc3s1200e-4	303	358	132	2+1	157,7
	xc5vlx30-3	80	206	126	2+1	386,1
	xc6vlx75t-3	69	191	124	2+1	343,4

Como já era esperado, a Unidade de Controle pode ser projetada de modo a requisitar menos elementos lógicos em relação ao *Data Path*. Os resultados experimentais sugerem ainda que, mesmo com a lógica adicional necessária para operar no modo 8-*branch*, a UC não influencia diretamente no tamanho do circuito final. O *Data path* apresentou as mesmas 128 LUTs identificadas no Capítulo 5.

No sentido de prover um Co-processador de cifra, além da Unidade de Controle e o *Data path*, o Núcleo de Cifra implementa uma pequena lógica extra para a interface de armazenamento na BRAM que armazena as *round* e *whitening keys*. Durante a análise do circuito final, esta lógica e a memória adicional são desconsideradas.

No que se refere ao Núcleo de Cifra, a análise dos dados presentes na Tabela 9 indica que a combinação de circuitos implementados em FPGA não significa, necessariamente, a soma de suas partes, uma vez que as ferramentas de síntese possuem grande liberdade de configuração. Essa característica, por vezes, implica em diferenças na distribuição da lógica em termos de SLICEs e LUTs. Além disso, estas variações na síntese lógica têm impacto significativo no processo de *Place & Route*. Uma das consequências desta característica está no aumento da frequência máxima de operação do Núcleo de Cifra, quando comparado ao *Data path*. Considerando a estrutura integrada, o caminho crítico do *Data path* foi alterado após o processo de mapeamento. Especificamente, o caminho crítico geral, formado a partir do atraso provocado pela combinação de 4 níveis lógicos, encontra-se na propagação do sinal que determina a transição entre os estados 'loop0' e 'flush_unswap' da máquina de estados.

Os valores apresentados na Tabela 10 referem-se à análise de taxa de transferência de dados (*throughput*). Esta taxa determina o período com o qual o circuito processa as

cadeias de bits de entrada. Os valores de *throughput* foram calculados de acordo com a expressão:

$$T\ (bits/segundo) = \frac{Tamanho\ (bits) \times Frequência\ (MHz)}{Latência\ (ciclos)}$$

Os valores identificados sugerem que taxas de 1,2 Gbps podem ser atingidas a um custo total de 80 SLICEs e 3 BRAMs em um dispositivo Virtex 5. Ao considerar o processamento de uma única sequência de bits no modo recursivo, a análise de desempenho considera 8 ciclos adicionais até o início da próxima cifra. Isso implica que o cômputo da cadeia binária i só deve ser iniciada após a finalização do bloco $i - 1$. Esta estratégia é considerada atualmente como o modo mais seguro de execução de cifras de bloco.

Tabela 10 – Análise de desempenho do Co-processador de cifra.

Dispositivo	**Tamanho da Chave (bits)**	**Ciclos Iniciais**	**Latência (ciclos)**	**Througput (Gbps)**	**Eficiência (Mbps/SLICE)**
xc3s1200e-4	128	4	42	0,481	1,6
	192	4	50	0,404	1,3
	256	4	58	0,348	1,1
xc5vlx30-3	128	4	42	1,208	14,7
	192	4	50	1,015	12,4
	256	4	58	0,875	10,7
xc6vlx75t-3	128	4	42	1,047	15,2
	192	4	50	0,879	12,7
	256	4	58	0,758	11,0

No estudo paralelo realizado para o dispositivo Spartan 3E, verifica-se que *throughputs* de aproximadamente 400 Mbps podem ser alcançados ao custo de 303 SLICEs e as mesmas 3 BRAMs. Essa alta taxa de ocupação reflete o fato de dispositivos da família Spartan 3E apresentarem apenas LUTs de 4 entradas. Por outro lado, os resultados para a Virtex 6 sugerem que a implementação do circuito neste dispositivo parece ter causado variações significativas durante o processo de otimização. Apesar de apresentar um maior número de LUTs para o *Data Path*, o Núcleo de Cifra foi implementado em uma quantidade menor de SLICEs. Uma análise dos registros de síntese sugere a necessidade de explorar melhor as características e novas funcionalidades agregadas à esta família de dispositivos. Entretanto, sem um estudo aprofundado das suas características físicas, o qual diverge dos objetivos deste trabalho, é difícil determinar as causas com precisão.

6.2 ANÁLISE DE DESEMPENHO PARA EXPANSÃO DE CHAVE

Até então, os resultados apresentam uma análise que considera somente a implementação do núcleo de criptografia do CLEFIA, desconsiderando também o cômputo das

chaves L de 256 bits, no modo 8-*branch*. Entretanto, na ausência de um elemento auxiliar de processamento ou quando o processo de expansão de chave precisa ser acelerado, um núcleo de criptografia com capacidade de expansão de chave deve ser considerado. Esta seção sumariza os resultados da implementação do Núcleo de Expansão de Chaves e do Co-processador para o CLEFIA, considerando a expansão de chaves de 128, 192 e 256 bits.

O levantamento da utilização de recursos do Núcleo de Expansão de Chave é apresentado na Tabela 11. Devido às características de funcionamento do controlador do Núcleo de Expansão de Chaves, descrito no Capítulo 5, era esperado que este circuito apresentasse um impacto significativo na implementação final. De acordo com os dados da Tabela 11, a Unidade de Controle ocupa aproximadamente 50% da área do Núcleo de Expansão de Chave. Ao considerar somente o *Data Path*, o circuito foi implementado a partir das mesmas 167 LUTs já indicadas no Capítulo 5.

Tabela 11 – Utilização de recursos do núcleo de Expansão de Chave.

	Dispositivo	**SLICEs**	**LUTs**	**FFs**	**RAMBs**	**Freq$_{max}$ (MHz)**
Unidade de Controle	xc3s1200e-4	111	199	61	0	147,3
	xc5vlx30-3	57	135	56	0	430,1
	xc6vlx75t-3	80	149	51	0	387,1
Data Path	xc3s1200e-4	247	302	192	0	191,3
	xc5vlx30-3	99	167	96	1	416,8
	xc6vlx75t-3	89	164	96	1	392,3
Núcleo de Expansão de Chave	xc3s1200e-4	357	501	253	1	138,6
	xc5vlx30-3	116	304	152	1	400,0
	xc6vlx75t-3	169	314	149	1	451,3

Apesar da ausência de atrasos consideráveis no caminho de dados, após a integração, o circuito apresentou uma degradação de cerca de 4% em relação ao *Data Path*. Esta degradação é devido à maior demanda no processo de roteamento, em função do tamanho da Unidade de Controle e sua integração para formar o circuito completo. Considerando a implementação do Núcleo de Expansão de Chave, o caminho crítico geral do circuito está localizado entre a UC e o *Data path*, correspondendo ao sinal de endereço propagado para o registrador de deslocamento (SRL16).

Nenhuma lógica adicional está associada à integração da Unidade de Controle e o *Data Path* do Núcleo de Expansão de Chave. Dessa forma, o circuito do Co-processador de cifra com expansão de chave do CLEFIA apresentou um padrão de integração coerente, em termos de alocação de recursos. A Tabela 12 sintetiza os resultados já analisados, em conjunto com a implementação do núcleo do CLEFIA com Expansão de Chave.

A análise da implementação do Co-processador com Expansão de Chave aponta para um acréscimo de 150% na área requerida, em relação ao Núcleo de Cifra independente.

Observa-se ainda uma degradação de cerca de 4% da frequência máxima de operação, em relação a este mesmo circuito. Novamente, esta degradação pode ser justificada em decorrência da lógica adicional associada à integração do circuito de expansão de chave. O elevado número de LUTs associado à esta implementação demanda maior esforço e agrega mais componentes, considerando-se os processos de mapeamento e roteamento do circuito completo.

Tabela 12 – Utilização de recursos das estruturas propostas.

	Dispositivo	**SLICEs**	**LUTs**	**FFs**	**RAMBs**	**Freq$_{max}$ (MHz)**
CLEFIA sem Expansão de Chave	xc3s1200e-4	303	358	132	2+1	157,7
	xc5vlx30-3	80	206	126	2+1	386,1
	xc6vlx75t-3	69	191	124	2+1	343,4
Expansão de Chave	xc3s1200e-4	357	501	253	1	138,6
	xc5vlx30-3	116	304	152	1	400,0
	xc6vlx75t-3	169	314	149	1	451,3
CLEFIA com Expansão de Chave	xc3s1200e-4	612	824	512	2+1	134,7
	xc5vlx30-3	200	505	273	2+1	369,1
	xc6vlx75t-3	210	507	265	2+1	335,3

Uma vez que o processo de expansão é realizado apenas uma vez a cada troca de chave, nenhuma análise de *throughput* foi realizada para este circuito. A Tabela 13 apresenta uma análise de desempenho do Co-processador com Expansão de Chave. O resultados sugerem *througputs* de aproximadamente 1,1 Gbps, ao custo de 200 SLICEs e 3 BRAMs, em um dispositivo FPGA Xilinx Virtex 5. Em termos de desempenho, verifica-se uma degradação de apenas 4,4% em relação ao núcleo de processamento do CLEFIA sem Expansão de Chave. Adicionalmente, a estrutura proposta para expansão de chave é capaz de processar até 4348 chaves de entrada de 256 bits por segundo, em um dispositivo Virtex 5. Nota-se ainda que, a Unidade de Controle do bloco de Expansão de Chave influenciou diretamente para a degradação da frequência de operação do circuito. Apesar de atingir frequência de operação maior, a integração dos circuitos refletiu na degradação de 8% da frequência de operação geral do sistema.

A implementação do Co-processador do CLEFIA com Expansão de Chave em um dispositivo Spartan 3E manteve *thoughputs* próximos de 400 Mbps, ao custo de 612 SLICEs. Uma vez que as LUTs presentes nos dispositivos da família Spartan 3E não implementam o modo SRL32, este registrador de deslocamento foi substituído por 64 LUTs configuradas no modo SRL16. Adicionalmente, uma lógica de seleção da saída dos registradores de deslocamento foi adicionada. Em um dispositivo FPGA Xilinx Virtex 6, o circuito de Expansão de Chave foi capaz de atingir frequência máxima de 451 MHz. Apesar do apresentar aproximadamente os mesmos requisitos em termos de LUTs e FFs, a

implementação no dispositivo Virtex 6 ocupou 46% mais área em relação aos resultados obtido para o *chip* Virtex 5.

Tabela 13 – Análise de desempenho do Co-processador com Expansão de Chave.

Dispositivo	**Tamanho da Chave (bits)**	**Ciclos Iniciais**	**Latência (ciclos)**	**Throughput (Gbps)**	**Eficiência (Mbps/SLICE)**
xc3s1200e-4	128	4	42	0,411	0,7
	192	4	50	0,345	0,6
	256	4	58	0,297	0,5
xc5vlx30-3	128	4	42	1,125	5,6
	192	4	50	0,945	4,7
	256	4	58	0,815	4,1
xc6vlx75t-3	128	4	42	1,022	4,9
	192	4	50	0,858	4,1
	256	4	58	0,740	3,5

6.3 COMPARAÇÃO COM O ESTADO DA ARTE

Os resultados obtidos para a estrutura proposta e aqueles presentes no Estado da Arte são apresentados na Tabela 14. Uma vez que é improvável que o núcleo do sistema opere em sua frequência máxima, os resultados de *throughput* são também apresentados para uma frequência de 13,56 MHz, padrão ISO para *smart cards* (ISO/IEC 14443-2, 2010). A seguir, uma sistematização das análises e algumas particularidades serão consideradas.

De acordo com os resultados até então discutidos, frequências de operação da ordem de 380 MHz podem ser alcançadas para a arquitetura proposta para o CLEFIA, sem a expansão de chave. Com isso é possível alcançar taxas de transferência de aproximadamente 1,2 Gbps. Para chegar a este resultado, 80 SLICEs e 3 BRAMs são necessárias. Uma BRAM adicional é considerada para armazenar as *round keys*. Considerando o *throughput* por SLICE como métrica de eficiência, uma eficiência de 14,7 Mbps/SLICE pode ser alcançada. Esta métrica pode ser contestada como uma medida tendenciosa, uma vez que não leva em consideração outros módulos do FPGA, como as BRAMs ou DSPs. Entretanto, dada a dificuldade em extrair valores equivalentes, esta estratégia é utilizada neste trabalho como a medida de comparação de eficiência. O núcleo de cifra do CLEFIA, integrado à estrutura de expansão de chaves de até 256 bits, é capaz de operar à uma frequência máxima de 369 MHz, ao custo de 200 SLICEs e as mesmas 3 BRAMs. Esta característica de operação resultou em um *throughput* máximo de aproximadamente 1,1 Gbps, resultando em uma eficiência igual a 5,6 Mbps/SLICE.

No que diz respeito ao presente Estado da Arte, este trabalho considera as abordagens voltadas para estruturas iterativas (*folded*) e desdobradas (*unfolded*). Ao considerar

uma abordagem iterativa (consequentemente a mais compacta das duas), com uma degradação de 18%, a estrutura proposta não apresentou a melhor taxa de transferência. Por outro lado, o núcleo de cifra apresentou a melhor área e frequência de operação, em relação aos demais trabalhos. Em Proença e Chaves (2011), os autores consideram apenas o processo de cifra, deixando de lado a expansão de chave. A menor latência em ciclos de clock resultou em *throughputs* de 1,3 Gbps. Quando comparado com a estrutura proposta, desconsiderando o módulo de expansão de chave, as otimizações propostas permitiram atingir *throughputs* semelhantes, demandando 2,5 vezes menos recursos.

Uma proposta de estrutura levemente diferente é apresentada em Resende e Chaves (2015). Neste trabalho, as estruturas de cifra de bloco CLEFIA e AES são combinadas em uma arquitetura unificada, considerando o uso de registradores de deslocamento endereçáveis. Esta característica os conduziu a uma estrutura compacta, capaz de computar os algoritmos de cifra de blocos CLEFIA e AES de forma eficiente. Os recursos extras, necessários ao suporte de ambos os codificadores, resultou em *throughput* inferiores, atingindo um máximo de 1,07 Gbps, ao custo de 123 SLICEs e 3 BRAMs. Note que nenhuma das estruturas destacadas acima são capazes de realizar a expansão de chave, demandando a realização deste processo fora do *chip*.

Quando o circuito precisa processar múltiplas sequências de dados, estruturas desdobradas podem ser utilizadas para atingir maiores taxas de transferência. A estrutura proposta em Kryjak e Gorgón (2009) resultou em uma frequência de operação reduzida, contudo o circuito é capaz de atingir *throughput* máximo de 21,376 Gbps. Esta taxa é alcançada ao custo de 2479 SLICEs, resultando em uma eficiência de 8,6 Mbps/SLICE, aproximadamente 60% daquela obtida pela estrutura proposta neste trabalho. Por outro lado, se o processo de cifra é dependente de diferentes blocos de dados, como nos modo *feedback*, esta estrutura desdobrada se torna altamente ineficiente, uma vez que a mesma requer que apenas um bloco por vez seja processado. Com isso, o *throughput* de dados é reduzido para 1,18 Gbps, resultando em uma eficiência igual a 0,48 Mbps/SLICE.

Tendo em vista os resultados alcançados, no que se refere ao *throughput* e a demanda por recursos, pode-se concluir que as modificação propostas para o núcleo de cifra do CLEFIA resultaram em uma estrutura eficiente, quando comparada à outras implementações. Esta afirmação é particularmente válida, ao considerar que o mesmo permite realizar as operações para $GFN_{4,n}$ e $GFN_{8,n}$, descartando a necessidade de reconfiguração e sem que isso resulte em um impacto significativo, em termos de área ou *throughput*.

Tabela 14 – Comparação de performance em relação ao Estado da Arte.

		Dispositivo	**SLICEs**	**BRAMs**	**Freq$_{max}$ (MHz)**	**Tp (Gbps)**	**Efficiencia (Mbps/S)**	**Tp (Mbps)**[1]	**Eficiência (kbps/S)**[1]
CLEFIA sem Expansão de Chave	Kryjak e Gorgón (2009)	xc5vlx30	2479	0	167	21,376	8,6	1735,68	700,2
	Proença e Chaves (2011)	xc3s1200e	270	2 + 1	185	0,658	2,4	48,21	178,6
		xc5vlx30	205	2 + 1	374	1,329	6,5	48,21	235,2
	Resende e Chaves (2015)[2]	xc5vlx30	123	2 + 1	352	1,073	8,7	40,36	328,2
	Proposto	xc3s1200e	303	2 + 1	157,7	0,481	1,6	41,33	136,4
		xc5vlx30	**80**	2 + 1	**386,1**	1,177	**14,7**	41,33	**516,6**
Expansão de Chave	Kryjak e Gorgón (2009)	xc2vp100	823	0	118	1,510	n.a.	173,57	n.a.
		xc2vp100	1462	0	74	1,579	n.a.	289,28	n.a.
	Chaves (2013)	xc3s1200e	218	1	193	0,412	n.a.	43,39	n.a.
		xc5vlx30	100	2 + 1	374	1,329	n.a.	43,39	n.a.
	Proposto	xc3s1200e	357	1	138,6	0,439	n.a.	37,73	n.a.
		xc5vlx30	116	1	**400,0**	1,113	n.a.	37,73	n.a.
CLEFIA com Expansão de Chave	Hanley e ONeill (2012)[3]	xc5vlx50	267	0	267	0,743	2,8	37,73	141,32
	Chaves (2013)[3]	xc3s1200e	574	2 + 1	184	654	1,1	48,21	84,0
		xc5vlx30	295	2 +1	374	1,33	4,5	48,21	163,4
	Proposto	xc3s1200e	612	2 + 1	134,7	0,411	0,7	41,33	67,5
		xc5vlx30	**200**	2 + 1	369,1	1,125	**5,6**	41,33	**206,6**

[1] Usando clock de 13,56 MHz, padrão ISO para *smart cards* (ISO/IEC 14443-2, 2010).
[2] Cifra dupla AES/CLEFIA.
[3] Suporta expansão de chaves de 128 bits, exclusivamente.

No tocante ao processo de expansão de chave, duas abordagens são consideradas: (i) obtê-las localmente, com um hardware dedicado; ou (ii) realizar o cálculo para obtenção das *round keys* fora do *chip* e armazená-las em uma memória durante o processo de inicialização do circuito. Apesar de levar a estruturas mais compactas e eficientes, a realização do processo de expansão de chave fora do *chip* não é indicada para sistemas como dispositivos embutidos pequenos (incapazes de suportar unidades de processamento de software), ou sistemas que demandem por mudanças de chave rápidas e frequentes. Nestes casos, uma estrutura dedicada para expansão de chave é requerida, demandando por mais recursos do dispositivo.

Em Kryjak e Gorgón (2009), os autores propõem uma estrutura iterativa para o cálculo de cada um dos três tipos de chave. Estas estruturas foram também adaptadas aos núcleos de cifra propostos por eles. O resultado apresentado na Tabela 14 descreve as métricas obtidas para expansão de chaves de 128 e 256 bits. A estrutura mais simples, para expansão de chaves de 128 bits, é capaz de atingir uma eficiência 80% maior em relação à mesma estrutura para chaves de 256 bits. Para o caso mais extremo, *throughputs* de 1,6 Gbps podem ser registrados. De forma geral, quando considerada a expansão de chaves de 256 bits, a estrutura proposta neste trabalho é capaz de atingir *throughput* semelhante, com custo de aproximadamente 14 vezes menos área.

O trabalho apresentado por Hanley e O'Neill (2012) propõe duas arquiteturas para o CLEFIA (serial e iterativa), com suporte à expansão de chaves de 128 bits. A Tabela 14 considera apenas os resultados para a implementação iterativa em um dispositivo Virtex 5. A principal característica desta arquitetura está na implementação das *S-Boxes* em LUTs. Com isso, 267 SLICEs são necessários para sua implementação, a operar em uma frequência máxima de 267 MHz, o que resulta em *throughput* máximo de 743 Mbps, 44% menor em relação ao da arquitetura aqui proposta. Em termos de eficiência, um ganho de 50% pode ser observado, quando considerada a estrutura aqui proposta.

Em Chaves (2013), uma arquitetura compacta para expansão de chave de 128 bits foi proposta. A implementação desta estrutura resultou em uma frequência máxima de 374 MHz, ao custo de 100 SLICEs. A estrutura de expansão de chave proposta neste trabalho foi projetada para dar suporte a todos os três tamanhos de chaves, ao custo de 116 SLICEs. Considerando que a estrutura proposta é capaz de expandir chaves dos três tamanhos possíveis no CLEFIA, o aumento de 16% em área não representa um acréscimo significativo na área final do circuito. Este resultado foi alcançado a partir da utilização de registradores de deslocamento endereçáveis, cuidadosamente escalonados.

A estrutura de expansão de chave aqui proposta requer 2 ou 6 ciclos de clock para obtenção parcial da chave intermediária oriunda da estrutura $GFN_{4/8,n}$. Além disso, 58, 72 ou 86 ciclos de clock são necessários para computar as *round keys* correspondentes às chaves de 128, 192 ou 256 bits. Ciclos adicionais foram introduzidos no sentido de permitir

a realização da função Σ sobre um caminho de dados de 32 bits. Entretanto, uma vez que a expansão de chave é realizada apenas uma vez, para a mesma chave de entrada, e esta foi capaz de operar à uma frequência superior ao núcleo principal, sua estrutura não produziu impacto significativo no desempenho do sistema.

A estrutura de cifra do CLEFIA com expansão de chave apresentada neste trabalho se mostrou capaz de operar à uma frequência máxima de 369 MHz, custando cerca de 47% menos área em relação à estrutura proposta por Chaves (2013). Mesmo com um *throughput* máximo 15% menor, a estrutura proposta atingiu índice de eficiência 20% maior.

6.4 CONCLUSÕES SOBRE A ANÁLISE DOS RESULTADOS

De modo geral, a estrutura proposta, capaz de computar o algoritmo CLEFIA completo, incluindo a expansão de chaves de todos os tamanhos, foi capaz de atingir *throughput* acima de 1 Gbps, ao custo de 200 SLICEs e 3 BRAMs. Quando comparada à estruturas compactas do CLEFIA com expansão de chave, presentes na literatura, a arquitetura apresentada neste trabalho alcançou a maior eficiência, com índice de 5,6 Mbps/SLICE. Adicionalmente, os resultados sugerem uma demanda 47% menor em relação à área em *chip*, quando comparada à melhor implementação presente no Estado da Arte. Com um *throughput* máximo de aproximadamente 1,2 Gbps e ao custo de 80 SLICEs e 3 BRAMs, a estrutrura proposta para o núcleo de cifra do CLEFIA apresentou eficiência 70% maior e área 50% menor, quando comparada à melhor implementação do CLEFIA até o presente.

7 CONCLUSÕES

O presente trabalho aqui concluído apresentou uma estrutura em hardware capaz de computar o algoritmo de cifra em bloco CLEFIA, incluindo a expansão de chave para todos os tamanhos definidos no padrão. A estrutura proposta apresenta evidências de que, apesar do custo em termos de área em *chip*, a implementação de um hardware dedicado para expansão de chave completo é viável.

Uma das principais dificuldades enfrentadas no projeto de um circuito de expansão de chaves completo do CLEFIA está na necessidade de estruturas diferentes para cada tipo de chave. Além da estrutura 4-*branch* para cifra e expansão de chaves de 128 bits, o processamento de chaves de 192 e 256 bits demanda uma rede Feistel 8-*branch*. Além disso, a alta complexidade estrutural associada ao processo de cômputo das *round keys* para chaves de 192 e 256 bits é considerada um obstáculo à sua implementação de forma eficiente e compacta.

Trabalhos relacionados apresentam uma perspectiva realista em torno das abordagens de implementação mais comumente utilizadas. Estruturas desdobradas são mais indicadas para codificação no modo de cifra *non-feedback*, enquanto arquiteturas iterativas apresentam menores taxas de ocupação e índices de eficiência no modo *feedback*. Em relação à cifra do CLEFIA, dois métodos foram destacados para implementação das transformações não lineares, as quais podem ser baseadas em implementação *hard-wired* ou mapeadas em *lookup tables*. Com a crescente evolução tecnológica, é comum encontrar estruturas de memória avançadas nos dispositivos FPGA modernos. De forma geral, estes elementos são indicados para implementação destas tabelas. Além disso, a utilização de registradores de deslocamento endereçáveis podem auxiliar na redução do tamanho do circuito.

As estruturas propostas neste trabalho para cifra e expansão de chave foram cuidadosamente projetadas, com vistas para dispositivos Xilinx da família Virtex 5 e seus respectivos componentes. O núcleo de codificação é composto por um *data path* de 32 bits, baseado no modelo de iteração recursiva (*rolled round*). A estrutura de expansão de chave foi também adaptada de modo a operar sobre um *data path* de 32 bits, sendo executada na forma de iterações sucessivas. Blocos de memória (BRAMs) foram utilizados para implementar as T-$Boxes$ e LUTs configuradas como registradores de deslocamento endereçáveis, para armazenamento de valores temporários. Com isso, esta pesquisa demonstrou que, a partir da exploração dos recursos presentes nos dispositivos FPGA, uma estrutura *dual-branch* Feistel pode ser implementada de forma eficiente.

Uma das principais caracterísitcas associadas à utilização de dispositivos FPGA refere-se à sua flexibilidade. Entratanto, é importante analisar o caráter de portabilidade

entre famílias de dispositivos, principalmente quando tratam-se de fabricantes diferentes. A solução proposta neste trabalho leva esta característica em consideração ao analisar as estruturas sob o escopo de *chips* Xilinx das famílias Spartan 3E e Spartan 6. Por serem do mesmo fabricante, este trabalho evidenciou que em dispositivos de baixo custo, poucas modificações precisam ser realizadas no sentido de adequar a estrutura proposta. Por outro lado, dispositivos de fabricantes diferentes, apesar de apresentarem funcionalidades semelhantes, diferem na forma de implementação. No que diz respeito aos FPGAs ALTERA, a estrutura proposta pode ser implementada a partir da substituição das BRAMs por modelos compatíveis à arquitetura do dispositivo. Os registradores de deslocamento (SRL), por sua vez podem ser implementados na forma de RAM-based Shift Registers. A diferença, neste caso, reside no fato de este último utilizar blocos de memória embutidas no *chip*.

Considerando apenas o núcleo de criptografia do CLEFIA, a arquitetura proposta requer um total de 80 SLICEs, para implementação do *data path* e da sua respectiva unidade de controle, dividida em cinco estágios, em um dispositivo Virtex 5. Este valor é reduzido para 69, quando a implementação é voltada para dispositivos Virtex 6. Utilizando uma estrutura iterativa, o circuito é capaz de operar à uma frequência máxima de 386 MHz. Considerando o modo *feedback*, um *throughput* de 1,2 Gbps pode ser alcançado, resultando em um fator de eficiência de aproximadamente 15 Mbps/SLICE. O núcleo de expansão de chave, quando implementado separadamente, requer 116 SLICEs, sendo capaz de operar à uma frequência máxima de 400 MHz em um dispositivo Virtex 5. Quando extrapolado para dispositivos Virtex 6, este valor sobe para 451 MHz. Apesar de considerar a hipótese de ajustes na arquitetura, ambas as estruturas foram implementadas de forma satisfatória também em um dispositivo da família Spartan 3E.

No que diz respeito ao co-processador de cifra com expansão de chave, a combinação das estruturas resultou em um circuito implementado em 200 SLICEs, operando à uma frequência máxima de 369 MHz. Com a lógica adicional associada ao núcleo de expansão de chave, o Co-processador do CLEFIA apresentou índice de 5,6 Mbps/SLICE, eficiência 20% maior que a melhor alternativa presente no Estado da Arte, sendo capaz de atingir um *throughput* geral de 1,12 Gbps.

7.1 TRABALHOS FUTUROS

Sob o espectro de trabalhos relacionados, a estrutura proposta para a rede *dual-branch* Feistel ($GFN_{4/8,n}$) foi a primeira capaz de realizar a cifra e expansão de chaves de 128, 192 e 256 bits, compartilhando da mesma estrutura e sem necessidade de reconfiguração. Esta característica abre portas para melhorias e adaptações futuras.

O trabalho proposto por Resende e Chaves (2015) mostrou ser viável a implementação de algoritmos de cifra múltipla, ao integrar o CLEFIA ao AES em uma estrutura compacta. Ao explorar características semelhantes deste e outros algoritmos, tal qual o

Twofish, novas estruturas de expansão de chave de múltiplas cifras podem ser propostas. O desafio, neste caso, está presente na característica do CLEFIA de possuir estruturas diferentes para expansão de chaves de diferentes tamanhos. Uma vez que o *data path* para cifra, que propomos neste trabalho, é capaz de realizar ambas as redes de Feistel implementadas no CLEFIA, este expansor múltiplo poderá compartilhar desta estrutura.

Com o aumento da demanda por aplicações baseadas em co-projeto hardware software, vislumbra-se a possibilidade de implementar esta estrutura em um ambiente integrado de Sistema em Chip (SoC). Atualmente os dois principais fabricantes de chips FPGA dispõem de soluções específicas para co-projeto usando unidades microprocessadas específicas implementadas na forma de *Intelectual Property* (IP) *cores*. Para tanto, esta unidade pode ser integrada à barramentos específicos como o Fast Simplex Link (FSL) para processadores MicroBlaze da Xilinx ou Avalon em dispositivos ALTERA. Destarte, ambos os fabricantes já possuem soluções dotadas de *Hard Processor Systems* (HPS), integrando ao chip FPGA processadores ARM. Neste sentido, com as devidas adaptações, o co-processador do CLEFIA pode ser integrado ao barramento AMBA/AXI presente neste tipo de estrutura.

Com o advento de novas características e tecnologias de fabricação de componentes associados aos dispositivos FPGA, novas opções de configuração são também postas à prova, desafiando projetistas e pesquisadores a explorar, ao máximo, tais dispositivos. Assim como os modos de configuração de LUTs SRL16 e SRL32, presentes na Virtex 5, outros aprimoramentos podem ser encontrados em dispositivos como Virtex 6 e 7, extrapolando até mesmo para as edições Ultra Scale da Xilinx. Juntamente com estes dispositivos em hardware, novas ferramentas de software, como a Xilinx Vivado Design Suite, surgem com o intuito de explorar de forma eficiente as novas tecnologias.

Explorar novas formas de implementação compacta das estruturas propostas, com vistas a tecnologias ASIC, também é uma extensão de pesquisa viável a partir deste trabalho. Neste sentido, o principal desafio está centrado na implementação do mecanismo de chaveamento difuso e a análise do *tradeoff* entre uma implementação direta ou com o uso de *T-boxes* Finalmente, pesquisadores habituados a projetos mais detalhados podem explorar as estruturas aqui apresentadas, no sentido de otimizar o roteamento e propagação dos sinais internos, além de produzir unidades de controle mais compactas.

REFERÊNCIAS

AKISHITA, T.; HIWATARI, H. *Very compact hardware implementations of the blockcipher CLEFIA*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. v. 7118. 278–292 p. (Lecture Notes in Computer Science, v. 7118). ISBN 978-3-642-28495-3. Disponível em: <http://dl.acm.org/citation.cfm?id=2186839.2186864>. Citado 2 vezes nas páginas 23 e 39.

CHAVES, R. Compact Clefia Implementation on FPGAs. In: ATHANAS, P.; PNEVMATIKATOS, D.; SKLAVOS, N. (Ed.). *Embedded Systems Design with FPGAs*. New York, NY: Springer New York, 2013. p. 225–243. ISBN 978-1-4614-1361-5. Disponível em: <http://www.springerlink.com/index/10.1007/978-1-4614-1362-2>. Citado 13 vezes nas páginas 13, 19, 23, 24, 39, 45, 46, 55, 68, 89, 96, 97 e 98.

GOOD, T.; BENAISSA, M. AES on FPGA from the Fastest to the Smallest. In: *Lecture Notes in Computer Science: Advances in Cryptology - Cryptographic Hardware and Embedded Systems - CHES 2005*. [S.l.: s.n.], 2005. p. 427 – 440. ISBN 978-3-540-28474-1. Citado na página 41.

HANLEY, N.; O'NEILL, M. Hardware Comparison of the ISO/IEC 29192-2 Block Ciphers. In: *2012 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2012. p. 57–62. ISBN 978-1-4673-2234-8. ISSN 2159-3469. Disponível em: <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6296448>. Citado 6 vezes nas páginas 19, 23, 39, 44, 89 e 97.

HANLEY, N.; ONEILL, M. Hardware Comparison of the ISO/IEC 29192-2 Block Ciphers. In: *2012 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2012. p. 57–62. ISBN 978-1-4673-2234-8. ISSN 2159-3469. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6296448>. Citado na página 96.

ISO/IEC 14443-2. *Identification cards - Contactless integrated circuit(s) cards - Proximity cards - Part 2: Radio frequency power and signal interface*. [S.l.], 2010. Citado 2 vezes nas páginas 94 e 96.

ISO/IEC 29192-2. *Information technology – Security techniques – Lightweight cryptography – Part 2: Block ciphers*. [S.l.], 2012. 41 p. Disponível em: <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=56552>. Citado 3 vezes nas páginas 23, 39 e 44.

KRYJAK, T.; GORGóN, M. Pipeline implementation of the 128-bit block cipher CLEFIA in FPGA. In: *International Conference on Field Programmable Logic and Applications, 2009.* Prague: [s.n.], 2009. p. 373–378. ISBN 9781424438921. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5272264>. Citado 8 vezes nas páginas 19, 24, 39, 43, 89, 95, 96 e 97.

PROENÇA, P.; CHAVES, R. Compact CLEFIA Implementation on FPGAS. In: *2011 21st International Conference on Field Programmable Logic and Applications*. IEEE, 2011. p. 512–517. ISBN 978-1-4577-1484-9. Disponível em:

<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6044874>. Citado 14 vezes nas páginas 13, 19, 24, 39, 41, 43, 44, 45, 49, 50, 52, 89, 95 e 96.

RESENDE, J. C.; CHAVES, R. Dual CLEFIA/AES Cipher Core on FPGA. In: *Applied Reconfigurable Computing (ARC)*. Bochum, Germany: [s.n.], 2015. p. 12. Citado 12 vezes nas páginas 13, 19, 24, 39, 42, 46, 49, 52, 89, 95, 96 e 100.

RODRIGUEZ-HENRIQUEZ, F. et al. *Cryptographic Algorithms on Reconfigurable Hardware)*. [s.n.], 2007. ISBN 0387366822. Disponível em: <http://books.google.com/books?hl=pt-BR&lr=&id=qIY9RTzemr8C&pgis=1>. Citado 2 vezes nas páginas 41 e 52.

SHIRAI, T.; KYOJI, S. On Feistel Structures Using a Diffusion Switching Mechanism. In: ROBSHAW, M. (Ed.). *Fast Software Encryption, 13th International Workshop, FSE 2006, Luxembourg, Luxembourg, March 15-17, 2006, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 4047). cap. Lecture No, p. pp 41–56. ISBN 978-3-540-36597-6. Disponível em: <http://www.springerlink.com/index/10.1007/11799313>. Citado 2 vezes nas páginas 23 e 27.

SHIRAI, T.; MIZUNO, A. A Compact and High Speed Cipher Suitable for Limited Resource Environment. *3rd ETSI security workshop presentation*, 2007. Disponível em: <http://docbox.etsi.org/workshop/2008/2008_securityworkshop/s8_2_taizo_shirai.pdf>. Citado na página 31.

SHIRAI, T. et al. The 128-Bit Blockcipher CLEFIA (Extended Abstract). In: *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*. [S.l.: s.n.], 2007. v. 4593, p. 181–195. ISBN 978-3-540-74617-1. ISSN 03029743. Citado 4 vezes nas páginas 23, 27, 28 e 33.

Sony Corporation. *The 128-bit Blockcipher CLEFIA Algorithm Specification*. [S.l.], 2007. 1–41 p. Citado 6 vezes nas páginas 29, 32, 33, 34, 35 e 45.

Sony Corporation. *The 128-bit Blockcipher CLEFIA Security and Performance Evaluations*. [S.l.], 2007. 1–52 p. Citado 2 vezes nas páginas 27 e 31.

SUGAWARA, T. et al. High-performance ASIC implementations of the 128-bit block cipher CLEFIA. In: *2008 IEEE International Symposium on Circuits and Systems*. Ieee, 2008. p. 2925–2928. ISBN 978-1-4244-1683-7. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4542070>. Citado 6 vezes nas páginas 23, 30, 39, 41, 45 e 52.

Xilinx. *Virtex-5 FPGA User Guide*. [S.l.], 2012. versão 5.4, 385 p. Disponível em: <http://www.xilinx.com/support/documentation/user_guides/ug190.pdf>. Citado 2 vezes nas páginas 13 e 40.

Apêndices

APÊNDICE A – DETALHAMENTO DA CIFRA DO CLEFIA EM PIPELINE

Tabela 15 – Ocupação do Registrador de Deslocamento e endereçamento da saída na cifra do CLEFIA.

	Posições no Registrador de Deslocamento							
Iteração	0	1	2	3	4	5	6	7
Inicialização	X	X						
	X	X	X					
	P_0	X	X	X				
	$P_1 + WK_0$	P_0	X	X	X			
1ª	P_2	$P_1 + WK_0$	P_0	X	X	X		
	$P_3 + WK_1$	P_2	$P_1 + WK_0$	P_0	X	X	X	
2ª	X	$P_3 + WK_1$	P_2	$P_1 + WK_0$	P_0	X	X	X
	A_1	X	$P_3 + WK_1$	P_2	$P_1 + WK_0$	P_0	X	X
3ª	B_1	A_1	X	$P_3 + WK_1$	P_2	$P_1 + WK_0$	P_0	X
	A_2	B_1	A_1	X	$P_3 + WK_1$	P_2	$P_1 + WK_0$	P_0
4ª	B_2	A_2	B_1	A_1	X	$P_3 + WK_1$	P_2	$P_1 + WK_0$
	A_3	B_2	A_2	B_1	A_1	X	$P_3 + WK_1$	P_2
5ª	B_3	A_3	B_2	A_2	B_1	A_1	X	$P_3 + WK_1$
	A_4	B_3	A_3	B_2	A_2	B_1	A_1	X
...	...							
n	B_{n-2}	A_{n-2}	B_{n-3}	A_{n-3}	B_{n-4}	A_{n-4}	B_{n-5}	A_{n-5}
	A_{n-1}	B_{n-2}	A_{n-2}	B_{n-3}	A_{n-3}	B_{n-4}	A_{n-4}	B_{n-5}
Flush	B_{n-1}	A_{n-1}	B_{n-2}	A_{n-2}	B_{n-3}	A_{n-3}	B_{n-4}	A_{n-4}
	A_n	B_{n-1}	A_{n-1}	B_{n-2}	A_{n-2}	B_{n-3}	A_{n-3}	B_{n-4}
Unswap	B_n	A_n	B_{n-1}	A_{n-1}	B_{n-2}	A_{n-2}	B_{n-3}	A_{n-3}
	X	B_n	A_n	B_{n-1}	A_{n-1}	B_{n-2}	A_{n-2}	B_{n-3}
	X	X	B_n	A_n	B_{n-1}	A_{n-1}	B_{n-2}	A_{n-2}
	P'_0	X	X	B_n	A_n	B_{n-1}	A_{n-1}	B_{n-2}
Nova 1ª	$P'_1 + WK_0$	P'_0	X	X	B_n	A_n	B_{n-1}	A_{n-1}
	P'_2	$P'_1 + WK_0$	P'_0	X	X	B_n	A_n	B_{n-1}

Tabela 16 – Escalonamento de operações da cifra do CLEFIA, onde a_i e b_i são resultados oriundos da Função-F e n o total de iterações.

Iteração	**Entrada do SRLn-2**	**Entrada da T-Box**	**Processamento na T-Box**	**Saída da T-Box**	**Estágio de Saída**
Inicialização	X	X	X	0	X
	P_0	X	X	0	X
	$P_1 + WK_0$	X	X	0	X
	P_2	X	X	0	X
1ª	$P_3 + WK_1$	$P_0 + RK_0$	X	0	X
	X	$P_2 + RK_1$	$F_0\{P_0 + RK_0\}$	0	X
2ª	$a_1 + P_1 + WK_0 = A_1$	$A_1 + RK_2$	$F_1\{P_2 + RK_1\}$	a_1	X
	$b_1 + P_3 + WK_1 = B_1$	$B_1 + RK_3$	$F_0\{A_1 + RK_2\}$	b_1	X
3ª	$a_2 + P_2 = A_2$	$A_2 + RK_4$	$F_1\{B_1 + RK_4\}$	a_2	X
	$b_2 + P_0 = B_2$	$B_2 + RK_5$	$F_0\{A_2 + RK_5\}$	b_2	X
4ª	$a_3 + B_1 = A_3$	$A_3 + RK_6$	$F_1\{B_2 + RK_6\}$	a_3	X
	$b_3 + A_1 = B_3$	$B_3 + RK_7$	$F_0\{A_3 + RK_7\}$	b_3	X
5ª	$a_4 + B_2 = A_4$	$A_4 + RK_8$	$F_1\{B_3 + RK_8\}$	a_4	X
	$b_4 + A_2 = B_4$	$B_4 + RK_9$	$F_0\{A_4 + RK_9\}$	b_4	X
6ª	$a_5 + B_3 = A_5$	$A_5 + RK_{10}$	$F_1\{A_5 + RK_{10}\}$	a_5	X
	$b_5 + A_3 = B_5$	$B_5 + RK_{11}$	$F_0\{B_5 + RK_{11}\}$	b_5	X
7ª	$a_6 + B_4 = A_6$	$A_6 + RK_{12}$	$F_1\{A_6 + RK_{12}\}$	a_6	X
	$b_6 + A_4 = B_6$	$B_6 + RK_{13}$	$F_0\{B_6 + RK_{13}\}$	b_6	X
...	...				
n	$a_{n-1} + B_{n-2} = A_{n-1}$	$A_{n-1} + RK_{2n-2}$	$F_1\{B_{n-2} + RK_{2n-3}\}$	a_{n-1}	X
	$b_{n-1} + A_{n-2} = B_{n-1}$	$B_{n-1} + RK_{2n-1}$	$F_0\{A_{n-1} + RK_{2n-2}\}$	b_{n-1}	X
Flush Round	$a_n + B_{n-1} + WK_2 = A_n$	X	$F_1\{B_{n-1} + RK_{2n-1}\}$	a_n	X
	$b_n + A_{n-1} + WK_3 = B_n$	X	X	b_n	X
Unswap Round	X	X	X	0	$A_{n-1} = C_0$
	X	X	X	0	$A_n = C_1$
	P'_0	X	X	0	$B_{n-1} = C_2$
	$P'_1 + WK_0$	X	X	0	$B_n = C_3$
Nova 1ª	P'_2	$P'_0 + RK_0$	X	0	X
	$P'_3 + WK_1$	$P'_1 + RK_1$	$F_0\{P'_0 + RK_0\}$	0	X

APÊNDICE B – DETALHAMENTO DA PRODUÇÃO DA CHAVE INTERMEDIÁRIA L DO CLEFIA EM PIPELINE

Tabela 17 – Ocupação do Registrador de Deslocamento e endereçamento de saída na cifra do CLEFIA para chaves de 192 bits.

	Posições no Registrador de Deslocamento															
Iteração	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Inicialização	X															
	X	X														
	KL_0	X	X													
	KL_1	KL_0	X	X												
	KL_2	KL_1	KL_0	X	X											
	KL_3	KL_2	KL_1	KL_0	X	X										
	KR_0	KL_3	KL_2	KL_1	KL_0	X	X									
1ª	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X	X								
	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X	X							
	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X	X						
	X	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X	X					
2ª	X	X	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X	X				
	A_1	X	X	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X	X			
	C_1	A_1	X	X	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X	X		
	B_1	C_1	A_1	X	X	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X	X	
3ª	D_1	B_1	C_1	A_1	X	X	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X	X
	A_2	D_1	B_1	C_1	A_1	X	X	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X

Continua na próxima página

Continuação da página anterior

	Posições no Registrador de Deslocamento															
Iteração	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	C_2	A_2	D_1	B_1	C_1	A_1	X	X	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0
	B_2	C_2	A_2	D_1	B_1	C_1	A_1	X	X	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1
4ª	D_2	B_2	C_2	A_2	D_1	B_1	C_1	A_1	X	X	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2
	A_3	D_2	B_2	C_2	A_2	D_1	B_1	C_1	A_1	X	X	KR_3	KR_2	KR_1	KR_0	KL_3
	C_3	A_3	D_2	B_2	C_2	A_2	D_1	B_1	C_1	A_1	X	X	KR_3	KR_2	KR_1	KR_0
	B_3	C_3	A_3	D_2	B_2	C_2	A_2	D_1	B_1	C_1	A_1	X	X	KR_3	KR_2	KR_1
5ª	D_3	B_3	C_3	A_3	D_2	B_2	C_2	A_2	D_1	B_1	C_1	A_1	X	X	KR_3	KR_2
	A_4	D_3	B_3	C_3	A_3	D_2	B_2	C_2	A_2	D_1	B_1	C_1	A_1	X	X	KR_3
	C_4	A_4	D_3	B_3	C_3	A_3	D_2	B_2	C_2	A_2	D_1	B_1	C_1	A_1	X	X
	B_4	C_4	A_4	D_3	B_3	C_3	A_3	D_2	B_2	C_2	A_2	D_1	B_1	C_1	A_1	X
...	...															
10ª	D_8	B_8	C_8	A_8	D_7	B_7	C_7	A_7	D_6	B_6	C_6	A_6	D_5	B_5	C_5	A_5
	A_9	D_8	B_8	C_8	A_8	D_7	B_7	C_7	A_7	D_6	B_6	C_6	A_6	D_5	B_5	C_5
	C_9	A_9	D_8	B_8	C_8	A_8	D_7	B_7	C_7	A_7	D_6	B_6	C_6	A_6	D_5	B_5
	B_9	C_9	A_9	D_8	B_8	C_8	A_8	D_7	B_7	C_7	A_7	D_6	B_6	C_6	A_6	D_5
Flush	D_9	B_9	C_9	A_9	D_8	B_8	C_8	A_8	D_7	B_7	C_7	A_7	D_6	B_6	C_6	A_6
	A_{10}	D_9	B_9	C_9	A_9	D_8	B_8	C_8	A_8	D_7	B_7	C_7	A_7	D_6	B_6	C_6
	C_{10}	A_{10}	D_9	B_9	C_9	A_9	D_8	B_8	C_8	A_8	D_7	B_7	C_7	A_7	D_6	B_6
	B_{10}	C_{10}	A_{10}	D_9	B_9	C_9	A_9	D_8	B_8	C_8	A_8	D_7	B_7	C_7	A_7	D_6
Unswap	D_{10}	B_{10}	C_{10}	A_{10}	D_9	B_9	C_9	A_9	D_8	B_8	C_8	A_8	D_7	B_7	C_7	A_7
	X	D_{10}	B_{10}	C_{10}	A_{10}	D_9	B_9	C_9	A_9	D_8	B_8	C_8	A_8	D_7	B_7	C_7
	X	X	D_{10}	B_{10}	C_{10}	A_{10}	D_9	B_9	C_9	A_9	D_8	B_8	C_8	A_8	D_7	B_7
	X	X	X	D_{10}	B_{10}	C_{10}	A_{10}	D_9	B_9	C_9	A_9	D_8	B_8	C_8	A_8	D_7

Continua na próxima página

	Continuação da página anterior															
	Posições no Registrador de Deslocamento															
Iteração	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	X	X	X	X	D_{10}	B_{10}	C_{10}	A_{10}	D_9	B_9	C_9	A_9	D_8	B_8	C_8	A_8
	X	X	X	X	X	D_{10}	B_{10}	C_{10}	A_{10}	D_9	B_9	C_9	A_9	D_8	B_8	C_8
	X	X	X	X	X	X	D_{10}	B_{10}	C_{10}	A_{10}	D_9	B_9	C_9	A_9	D_8	B_8
	X	X	X	X	X	X	X	D_{10}	B_{10}	C_{10}	A_{10}	D_9	B_9	C_9	A_9	D_8

Tabela 18 – Escalonamento de operações do processo de obtenção da chave intermediária L no modo 8-*branch*, onde $a'i$, $b'i$, $c'i$ e $d'i$ são resultados da Função-F.

Iteração	**Entrada do SRL16**	**Entrada da T-Box**	**Processamento na T-Box**	**Saída da T-Box**	**Estágio de Saída**
Inicialização	X	X	X	0	X
	KL_0	X	X	0	X
	KL_1	X	X	0	X
	KL_2	X	X	0	X
	KL_3	X	X	0	X
	KR_0	X	X	0	X
	KR_1	X	X	0	X
1ª	KR_2	$KL_0 + RK_0$	X	0	X
	KR_3	$KL_2 + RK_1$	$F_0\{KL_0 + RK_0\}$	0	X
	X	$KR_0 + RK_2$	$F_1\{KL_2 + RK_1\}$	a_1	X
	X	$KR_2 + RK_3$	$F_0\{KR_0 + RK_2\}$	c_1	X
2ª	$a_1 + KL_1 = A_1$	$A_1 + RK_4$	$F_1(KR_2 + RK_3\}$	b_1	X
	$c_1 + KL_3 = C_1$	$C_1 + RK_5$	$F_0\{A_1 + RK_4\}$	d_1	X
	$b_1 + KR_1 = B_1$	$B_1 + RK_6$	$F_1\{C_1 + RK_5\}$	a_2	X
	$d_1 + KR_3 = D_1$	$D_1 + RK_7$	$F0_{\{}B_1 + RK_7\}$	c_2	X
3ª	$a_2 + KL_2 = A_2$	$A_2 + RK_8$	$F_1\{D_1 + RK_7\}$	b_2	X
	$c_2 + KR_0 = C_2$	$C_2 + RK_9$	$F_0\{A_2 + RK_8\}$	d_2	X
	$b_2 + KR_2 = B_2$	$B_2 + RK_{10}$	$F_1\{C_2 + RK_9\}$	a_3	X
	$d_2 + KL_0 = D_2$	$D_2 + RK_{11}$	$F_0\{B_2 + RK_{10}\}$	c_3	X
4ª	$a_3 + C_1 = A_3$	$A_3 + RK_{12}$	$F_1\{D_2 + RK_{11}\}$	b_3	X
	$c_3 + B_1 = C_3$	$C_3 + RK_{13}$	$F_0\{A_3 + RK_{12}\}$	d_3	X
	$b_3 + D_1 = B_3$	$B_3 + RK_{14}$	$F_1\{C_3 + RK_{13}\}$	a_4	X
	$d_3 + A_1 = D_3$	$D_3 + RK_{15}$	$F_0\{B_3 + RK_{14}\}$	c_4	X
5ª	$a_4 + C_2 = A_4$	$A_4 + RK_{12}$	$F_1\{D_3 + RK_{11}\}$	b4	X
	$c_4 + B_2 = C_4$	$C_4 + RK_{13}$	$F_0\{A_4 + RK_{12}\}$	d_4	X
	$b_4 + D_2 = B_4$	$B_4 + RK_{14}$	$F_1\{C_4 + RK_{13}\}$	a_5	X
	$d_4 + A_2 = D_4$	$D_4 + RK_{15}$	$F_0\{B_4 + RK_{14}\}$	c_5	X
...	...				
10ª	$a_9 + C_7 = A_9$	$A_9 + RK_{32}$	$F_1\{D_8 + RK_{31}\}$	b_9	X
	$c_9 + B_7 = C_9$	$C_9 + RK_{33}$	$F_0\{A_9 + RK_{32}\}$	d_9	X
	$b_9 + D_7 = B_9$	$B_9 + RK_{34}$	$F_1\{C_9 + RK_{33}\}$	a_{10}	X
	$d_9 + A_7 = D_9$	$D_9 + RK_{35}$	$F_0\{B_9 + RK_{34}\}$	c_{10}	X
Flush	$a_{10} + C_8 = A_{10}$	X	$F_1\{D_9 + RK_{35}\}$	b_{10}	X
	$c_{10} + B_8 = C_{10}$	X	X	d_{10}	X
	$b_{10} + D_8 = B_{10}$	X	X	X	X
	$d_{10} + A_8 = D_{10}$	X	X	X	X
Unswap	X	X	X	X	$A_9 = LL_0$
	X	X	X	X	$A_{10} = LL_1$
	X	X	X	X	$C_9 = LL_2$
	X	X	X	X	$C_{10} = LL_3$
	X	X	X	X	$B_9 = LR_0$
	X	X	X	X	$B_{10} = LR_1$
	X	X	X	X	$D_9 = LR_2$
	X	X	X	X	$D_{10} = LR_3$

APÊNDICE C – DETALHAMENTO DA EXPANSÃO DE CHAVES DE 128 BITS

Tabela 19 – Escalonamento da entrada da chave K, de 128 bits, no mecanismo de expansão de chave e introdução da chave no bloco GFN.

Ciclo	**Entrada do SRL**	**Saída do SRL**	**Delay Slot**	**Saída** (K)
1	$K_0 = WK_0$	x	x	x
2	$K_1 = WK_1$	x	x	x
3	$K_2 = WK_2$	x	x	x
4	$K_3 = WK_3$	K_0	x	x
5	–	K_1	x	K_0
6	–	K_2	x	K_1
7	–	K_3	x	K_2
8	–	x	x	K_3

Tabela 20 – Ocupação do Registrador de Deslocamento da Chave de Entrada (SRL16).

	Ocupação do SRL16			
Ciclo	0	1	2	3
1	X			
2	$K_0 = WK_0$	X		
3	$K_1 = WK_1$	$K_0 = WK_0$	X	
4	$K_2 = WK_2$	$K_1 = WK_1$	$K_0 = WK_0$	X
5	$K_3 = WK_3$	$K_2 = WK_2$	$K_1 = WK_1$	$K_0 = WK_0$

Tabela 21 – Escalonamento de operações da expansão de chave de cifra do CLEFIA-128, onde Σ_j^i corresponde ao bloco resultante da função Σ na iteração i.

Ciclo	**Entrada do SRL32**	**Saída do SRL32**	**Entrada da BRAM**
1	L_0	X	X
2	L_1	L_0	X
3	L_2	L_1	$RK_0 = L_0 + CON_0$
4	L_3	L_2	$RK_1 = L_1 + CON_1$
5	Σ_3^1	L_3	$RK_2 = L_2 + CON_2$
6	Σ_1^1	L_1	$RK_3 = L_3 + CON_3$
7	Σ_0^1	L_0	X
8	Σ_2^1	L_2	X
9	X	Σ_0^1	$RK_4 = K_0 + \Sigma_0^1 + CON_4$
10	X	Σ_1^1	$RK_5 = K_1 + \Sigma_1^1 + CON_5$
11	X	Σ_2^1	$RK_6 = K_2 + \Sigma_2^1 + CON_6$
12	Σ_0^2	Σ_3^1	$RK_7 = K3 + \Sigma_3^1 + CON_7$
13	Σ_2^2	Σ_1^1	X
14	Σ_3^2	Σ_0^1	X
15	Σ_1^2	Σ_2^1	X
...	...		
51	X	Σ_0^7	$RK_28 = K_0 + \Sigma_0^7 + CON_{28}$
52	X	Σ_1^7	$RK_29 = K_1 + \Sigma_1^7 + CON_{29}$
53	X	Σ_2^7	$RK_30 = K_2 + \Sigma_2^7 + CON_{30}$
54	Σ_0^8	Σ_3^7	$RK_31 = K3 + \Sigma_3^7 + CON_{31}$
55	Σ_2^8	Σ_1^7	X
56	Σ_3^8	Σ_0^7	X
57	Σ_1^8	Σ_2^7	X
58	X	Σ_0^8	$RK_32 = \Sigma_0^8 + CON_{32}$
59	X	Σ_1^8	$RK_33 = \Sigma_1^8 + CON_{33}$
60	X	Σ_2^8	$RK_34 = \Sigma_2^8 + CON_{34}$
61	X	Σ_3^8	$RK_35 = \Sigma_3^8 + CON_{35}$

APÊNDICE D – DETALHAMENTO DA EXPANSÃO DE CHAVES DE 192 E 256 BITS

Tabela 22 – Escalonamento da entrada da chave K, de 192 bits, no mecanismo de expansão de chave, geração das *witening keys* e introdução da chave no bloco GFN.

Ciclo	**Entrada do SRL**	**Saída do SRL**	**Delay Slot**	**Saída** (K)
1	KL_0	x	x	x
2	KL_1	x	x	x
3	KL_2	x	x	x
4	KL_3	x	x	x
5	KR_0	x	x	x
6	KR_1	KL_0	x	x
7	$\sim KL_0$	KL_1	0	x
8	$\sim KL_1$	KL_0	0	x
9	–	KR_0	x	x
10	$WK_0 = KL_0 \oplus KR_0$	KL_1	KL_0	x
11	–	KR_1	KR_0	x
12	$WK_1 = KL_1 \oplus KR_1$	KL_2	KL_1	x
13	–	$\sim KL_0$	KR_1	x
14	$WK_2 = KL_2 \oplus \sim KL_0$	KL_3	KL_2	x
15	–	$\sim KL_1$	$\sim KL_0$	x
16	$WK_0 = KL_3 \oplus \sim KL_1$	KL_0	KL_3	x
17	–	KL_1	$\sim KL_1$	KL_0
18	–	KL_2	x	KL_1
19	–	KL_3	x	KL_2
20	–	KR_0	x	KL_3
21	–	KR_1	x	KR_0
22	–	$\sim KL_0$	x	KR_1
23	–	$\sim KL_1$	x	$\sim KL_0 = KR_2$
24		x	x	$\sim KL_1 = KR_3$

Tabela 23 – Escalonamento da entrada da chave K, de 256 bits, no mecanismo de expansão de chave, geração das *witening keys* e introdução da chave no bloco GFN.

Ciclo	**Entrada do SRL**	**Saída do SRL**	**Delay Slot**	**Saída** (K)
1	KL_0	x	x	x
2	KL_1	x	x	x
3	KL_2	x	x	x
4	KL_3	x	x	x
5	KR_0	x	x	x
6	KR_1	x	x	x
7	KR_2	x	x	x
8	KR_3	KL_0	x	x
9	–	KR_0	x	x
10	$WK_0 = KL_0 \oplus KR_0$	KL_1	KL_0	x
11	–	KR_1	KR_0	x
12	$WK_1 = KL_1 \oplus KR_1$	KL_2	KL_1	x
13	–	KR_2	KR_1	x
14	$WK_2 = KL_2 \oplus KR_2$	KL_3	KL_2	x
15	–	KR_3	KR_2	x
16	$WK_3 = KL_3 \oplus KR_3$	KL_0	KL_3	x
17	–	KL_1	x	KL_0
18	–	KL_2	x	KL_1
19	–	KL_3	x	KL_2
20	–	KR_0	x	KL_3
21	–	KR_1	x	KR_0
22	–	KR_2	x	KR_1
23	–	KR_3	x	KR_2
24		x	x	KR_3

Tabela 24 – Ocupação do Registrador de Deslocamente para Chaves de Entrada de 192 bits. Onde os elementos WK_i representam os blocos de 32 bits que compõem a Whitening Key.

	Ocupação do SRL16 para chaves de 192 bits											
Ciclo	0	1	2	3	4	5	6	7	8	9	10	11
1	X											
2	KL_0	X										
3	KL_1	KL_0	X									
4	KL_2	KL_1	KL_0	X								
5	KL_3	KL_2	KL_1	KL_0	X							
6	KR_0	KL_3	KL_2	KL_1	KL_0	X						
7	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X					
8	$\sim KL_0$	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X				
9	$\sim KL_1$	$\sim KL_0$	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0				
10	$\sim KL_1$	$\sim KL_0$	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X			
11	WK_0	$\sim KL_1$	$\sim KL_0$	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0			
12	WK_0	$\sim KL_1$	$\sim KL_0$	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X		
13	WK_1	WK_0	$\sim KL_1$	$\sim KL_0$	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0		
14	WK_1	WK_0	$\sim KL_1$	$\sim KL_0$	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X	
15	WK_2	WK_1	WK_0	$\sim KL_1$	$\sim KL_0$	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	
16	WK_2	WK_1	WK_0	$\sim KL_1$	$\sim KL_0$	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X
17	WK_3	WK_2	WK_1	WK_0	$\sim KL_1$	$\sim KL_0$	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0
...	...											

Tabela 25 – Ocupação do Registrador de Deslocamente para Chaves de Entrada de 256 bits. Onde os elementos WK_i representam os blocos de 32 bits que compõem a Whitening Key.

	Ocupação do SRL16 para chaves de 192 bits											
#	0	1	2	3	4	5	6	7	8	9	10	11
1	X											
2	KL_0	X										
3	KL_1	KL_0	X									
4	KL_2	KL_1	KL_0	X								
5	KL_3	KL_2	KL_1	KL_0	X							
6	KR_0	KL_3	KL_2	KL_1	KL_0	X						
7	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X					
8	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X				
9	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0				
10	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X			
11	WK_0	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0			
12	WK_0	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X		
13	WK_1	WK_0	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0		
14	WK_1	WK_1	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X	
15	WK_2	WK_1	WK_0	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	
16	WK_2	WK_2	WK_1	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0	X
17	WK_3	WK_2	WK_1	WK_0	KR_3	KR_2	KR_1	KR_0	KL_3	KL_2	KL_1	KL_0
...	...											

Anexos

ANEXO A – ESQUEMÁTICO DOS COMPONENTES DO DISPOSITIVO FPGA

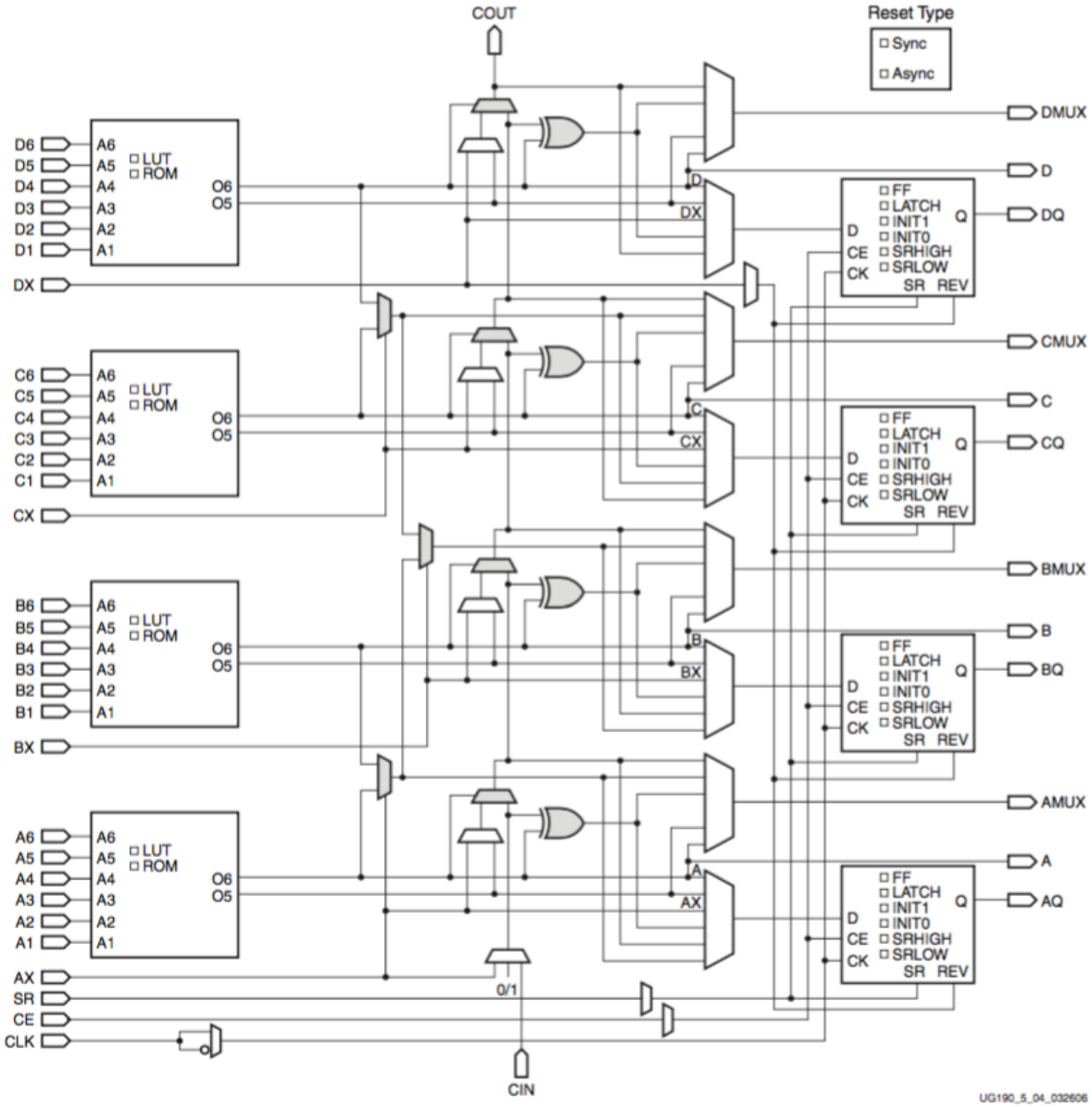


Figura 26 – Esquemático do SLICEL presente no dispositivo Xilinx Virtex 5, como descrito no Xilinx's Virtex-5 FPGA User Guide

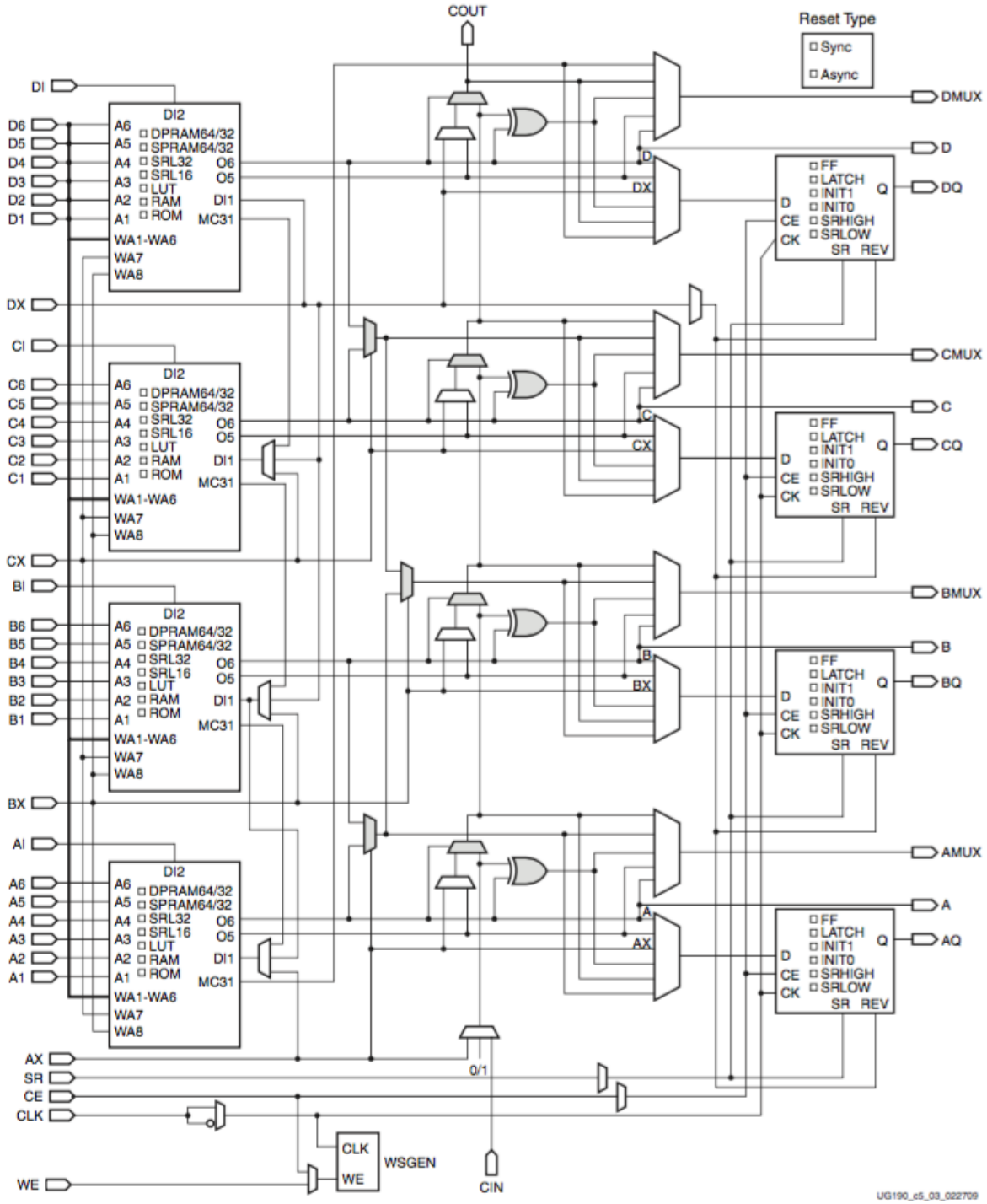


Figura 27 – Esquemático do SLICEM presente no dispositivo Xilinx Virtex 5, como descrito no Xilinx's Virtex-5 FPGA User Guide

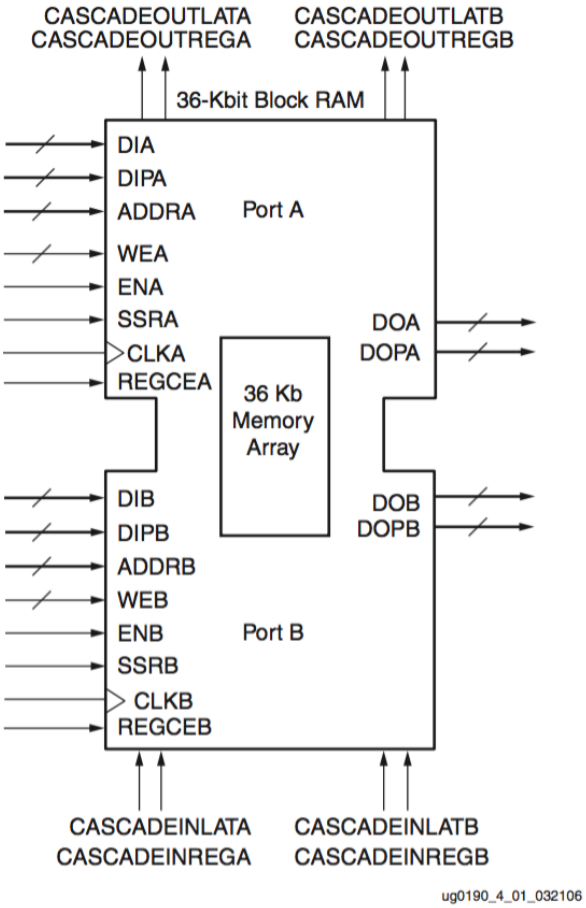


Figura 28 – Esquemático do BRAM presente no dispositivo Xilinx Virtex 5, como descrito no Xilinx's Virtex-5 FPGA User Guide