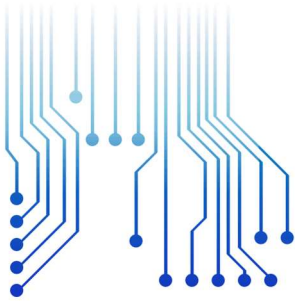
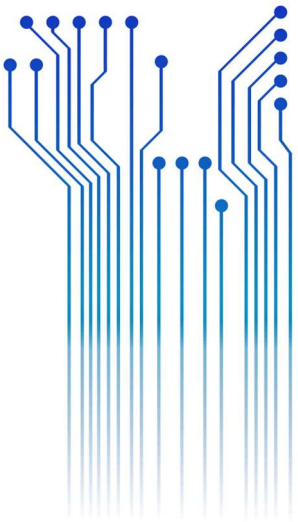


CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RUBENS FERNANDES ROUX ABRANTES

TRABALHO DE CONCLUSÃO DE CURSO

IMPLEMENTAÇÃO EM HARDWARE DE ENCRIPTAÇÃO E DECRIPTAÇÃO COM UTILIZAÇÃO DO *ADVANCED ENCRYPTION STANDARD* (AES)



Campina Grande
2018

RUBENS FERNANDES ROUX ABRANTES

IMPLEMENTAÇÃO EM HARDWARE DE ENCRIPTAÇÃO E DECRIPTAÇÃO COM UTILIZAÇÃO DO *ADVANCED ENCRYPTION STANDARD* (AES)

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Microeletrônica

Orientador:
Professor Gutemberg Gonçalves dos Santos Júnior, D.Sc.

Campina Grande
2018

RUBENS FERNANDES ROUX ABRANTES

IMPLEMENTAÇÃO EM HARDWARE DE ENCRIPTAÇÃO E DECRIPTAÇÃO COM UTILIZAÇÃO DO *ADVANCED ENCRYPTION STANDARD* (AES)

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Microeletrônica

Aprovado em 22 / 03 / 2018

Professor Marcos Ricardo Alcântara Morais, D.Sc.
Universidade Federal de Campina Grande
Avaliador

Professor Gutemberg Gonçalves dos Santos Júnior, D.Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Dedico este trabalho a todos que
me apoiaram ao longo do curso.

AGRADECIMENTOS

Agradeço a minha mãe, Maria Sabina, por ter se esforçado tanto para me proporcionar uma boa educação, por ter me alimentado com saúde, força e coragem, as quais que foram essenciais para superação de todas as adversidades ao longo desta caminhada.

Agradeço também ao meu padrasto, Luciano Alberto, que me indicou o curso e ajudou a formar o caráter que tenho hoje.

Agradeço também a minha namorada, Emilly Melo, que me apoia desde o início do curso.

Ao meu orientador Prof. Gutemberg Gonçalves, por todas as oportunidades e orientações concedidas ao longo desses últimos anos.

A todos os membros do projeto de excelência em microeletrônica, especialmente a Gabriel Villanova que coordenou a elaboração do AES, a Pedro Cavalcante que coordenou a verificação, a Lucas Eliseu por ter dado assistência na implementação em FPGA e também a Samuel Mendes, Dimas Germano e Cícero Freire.

Agradeço também a toda minha família, que com todo carinho e apoio, não mediu esforços para eu chegar a esta etapa da minha vida.

Agradeço também ao *StackOverFlow* pois sempre que tive dúvidas ele foi capaz de saná-las sem qualquer dificuldade.

Enfim, agradeço a todos que de alguma forma, passaram pela minha vida e contribuíram para a construção de quem sou hoje.

“The Universe is under no obligation to make sense to you”

Neil deGrasse Tyson.

RESUMO

A criptografia aplicada à segurança em transações de informações eletrônicas adquiriu uma grande relevância nos últimos anos. O *Advanced Encryption Standard* (AES) é utilizado para proteção de dados, desde dados governamentais até redes de WiFi, por exemplo. O tamanho de sua chave torna-o mais robusto contra ação de *hackers*. Este trabalho apresenta um estudo sobre o AES e a implementação do mesmo em *hardware*. Inicialmente, foi feito um estudo teórico acerca do tema, seguido da arquitetura proposta e finalmente, resultados de simulações e testes realizado em uma FPGA para validação do algoritmo.

Palavras-chave: AES, *design* de algoritmo e análise, FPGA, segurança de dados.

ABSTRACT

Encryption applied to security of electronic information transactions has become more relevant in past years. The Advanced Encryption Standard (AES) is used for data protection, from government data to WiFi networks, for example. The size of its key makes it more robust against hackers. This work presents a study about AES and its implementation in hardware. Initially, a theoretical study was done on the subject, followed by the proposed architecture and finally, results of simulations and tests performed in a FPGA for validation of the algorithm.

Keywords: AES, algorithm design and analysis, data security, FPGA.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A importância da criptografia aplicada à segurança em transações de informações eletrônicas adquiriu uma grande relevância nos últimos anos. Isso faz com que a proteção de dados exerça um papel cada vez mais importante na sociedade.

Segundo Deshpande et al (2009), o padrão de encriptação avançado (AES), também conhecido como *Rijndael*, é uma forma de encriptação adotada pelo governo do Estados Unidos, que especifica um algoritmo que seja capaz de proteger informações sensíveis.

Segundo FIPS 197 (2001), o algoritmo do AES é capaz de encriptar e decriptar informação. Encriptação converte dados para uma forma ininteligível, chamada de *cipher-text*. Decriptação do *cipher-text* converte a informação de volta para sua forma inicial, chamada *plain text*. O algoritmo do AES pode utilizar chaves de encriptação de 128, 192 e 256 bits para encriptar blocos de 128 bits.

Devido a fácil implementação e por ser utilizado universalmente de segurança de redes WiFi até transações de informação sensível pelo governo do Estados Unidos, o AES foi a opção escolhida para esta implementação.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Estudos sobre método de encriptação e decriptação AES, e implementação em um Arranjo de Portas Programáveis em Campo (FPGA) do mesmo para proteção de dados.

1.1.2 OBJETIVOS ESPECÍFICOS

- Revisão bibliográfica acerca do AES;
- Desenvolvimento do AES em linguagem de *hardware*;
- Implementação do AES em uma FPGA.

1.2 METODOLOGIA

A metodologia empregada neste trabalho envolveu a realização de pesquisa e atualização bibliográfica sobre o tema proposto.

A implementação do AES foi realizada na linguagem de programação de *hardware* SystemVerilog, primeiramente a criação do módulo de expansão da chave, seguido da encriptação e decriptação.

Posteriormente, foi criado um suporte para utilização de diversos modos de encriptação usados para maior segurança dos dados. Para realizar a comunicação com o *design*, uma interface AMBA AXI4-Lite foi usada.

Em seguida foi realizada a verificação do *design* utilizando métodos automáticos de verificação e por fim foi implementado em uma FPGA.

2 ALGORITMO AES

O AES é um algoritmo de criptografia destinada a compor sistemas de cifragem e decifragem simétrica, i.e., mesma chave para encriptar e decriptar. É uma cifra de bloco, ou seja, opera em blocos de tamanho fixo de 128 *bits* ou 16 *bytes*. Pode operar com chaves de 128, 192 ou 256 bits. Foi desenvolvido pelo governo dos Estados Unidos e anunciado pelo Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (NIST) como U.S. FIPS PUB (FIPS 197).

O objetivo da encriptação bem-sucedida é que seja impraticável de descobrir a mensagem original caso somente se possua a mensagem cifrada, mas não a chave de criptografia. Para isso, busca-se minimizar qualquer correlação visível entre a entrada e a saída, de modo que a mesma (e/ou a chave) possa ser deduzida simplesmente observando-se um número muito grande de cifras. Para isso, usa-se uma série de "rodadas" (*rounds*) em que os *bytes* sofrem transformações não lineares, porém reversíveis, i.e., para decifrar, simplesmente se executa o inverso das mesmas operações, em ordem inversa.

Todos as operações no AES tratam os *bytes* de entrada como um corpo finito em 2^8. Isto significa que:

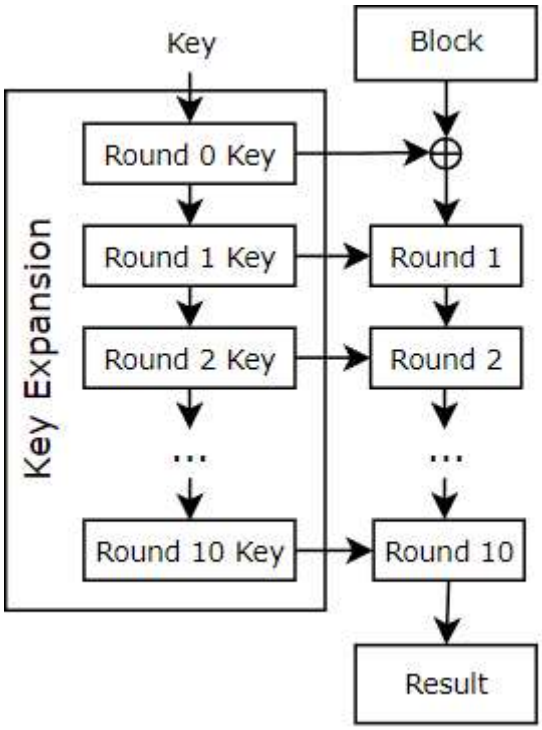
1. Há um conjunto $[0,255]$ formado por todos os valores possíveis para um *byte* e um desses elementos é chamado “zero” (neste caso, o próprio 0);
2. Há uma operação, chamada de “adição” definida como “OU exclusivo” (XOR), que se aplica a quaisquer dois elementos nesse conjunto e cujo resultado também é um elemento desse conjunto. Essa operação precisa ser associativa, comutativa, possuir elemento neutro, e cada elemento deve ter um inverso;
3. Há uma operação, chamada de "multiplicação", com características semelhantes à "adição". Exceto pelo elemento "zero", que não tem inverso (e o elemento neutro da multiplicação é chamado de "um"). A “multiplicação” também precisa ser distributiva em relação à adição.

Para realizar a “multiplicação” cada operando deve ser tratado como um polinômio com base na sua representação binária (Ex.: $6_d = 110_b = x^2 + x$). Posteriormente os polinômios devem ser multiplicados e divididos por um “agente redutor”. O resto da divisão será o resultado da “multiplicação” (devendo ser interpretado

novamente como um número). No caso do AES, o “agente redutor” escolhido é: $x^8 + x^4 + x^3 + x + 1 = 100011011_b = 283_d$.

O algoritmo funciona em rodadas, ou *rounds*, de modo que em cada uma delas realiza uma série de operações reversíveis em cima do estado. O objetivo é que cada *byte* da entrada seja "combinado" com diversos *bytes* da chave, de modo que pequenas alterações tanto na chave quanto na mensagem provoquem mudanças significativas na cifra. O algoritmo implementado utiliza chave de 128 *bits*, portanto possui 10 rodadas, como apresentado na Figura 1.

Figura 1 – Algoritmo simplificado do AES.



Fonte: O próprio autor.

Em cada rodada não é utilizada a chave original de criptografia, mas sim uma série de chaves derivadas da mesma. Essa derivação usa um algoritmo chamado *Rijndael key schedule* ou *key expansion* (expansão da chave) e seu pseudocódigo é apresentado na Figura 2. Esta função recebe a chave de 128 *bits* e retorna a expansão da chave, 11 matrizes de 128 *bits*, a ser utilizada em cada rodada da encriptação ou decriptação.

Figura 2 – Pseudocódigo para expansão da chave.

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

Fonte: (FIPS 197, 2001).

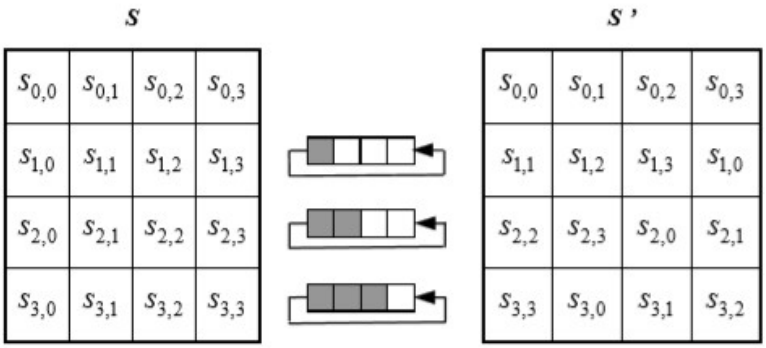
Para realização da encriptação, os seguintes passos são seguidos: A rodada inicial consiste de uma operação XOR da chave com o bloco de entrada, que resulta na entrada para a próxima. Nas rodadas 1 a 9 cada *byte* de estado é transformado de acordo com a matriz de substituição da Figura 3, em seguida uma rotação das linhas da esquerda para a direita, conforme apresentado na Figura 4, posteriormente, em cada coluna da matriz, são combinados os *bytes* com todos os outros da mesma coluna com “pesos” diferentes, ilustrado na Figura 5, e por fim é realizado uma operação XOR da chave expandida da rodada específica com o bloco. Na última rodada são realizadas as operações de substituição dos *bytes*, rotação das linhas e uma XOR com a chave da rodada, resultando o bloco encriptado.

Figura 3 – Matriz de substituição para encriptação do *byte* xy (em hexadecimal).

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fonte: (FIPS 197, 2001).

Figura 4 – Rotação das linhas para encriptação.



Fonte: (FIPS 197, 2001).

Figura 5 – Combinação dos *bytes* da coluna para encriptação.

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

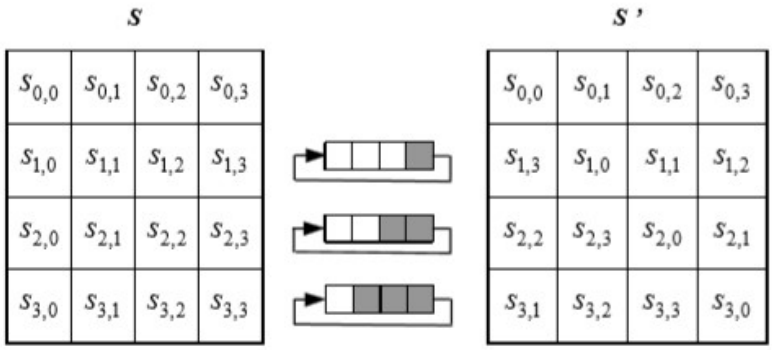
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

Fonte: (FIPS 197, 2001).

O processo de decriptação consiste em aplicar o inverso das mesmas operações com a mesma chave expandida. Na primeira rodada é realizada uma XOR da chave com

o bloco. Nas rodadas de 1 a 9 é realizada a rotação inversa das linhas da direita para esquerda, ilustrado na Figura 6, em seguida cada *byte* de estado é transformado de acordo com a matriz apresentada na Figura 7, posteriormente é realizada uma XOR com a chave expandida da rodada e por fim são combinados os *bytes* com todos os outros de mesma coluna com "pesos" diferentes, apresentado na Figura 8. Na última rodada a rotação inversa das linhas é realizada, a substituição de cada *byte* e por fim uma XOR com a chave expandida da rodada, resultado no bloco decriptado.

Figura 6 – Rotação das linhas para decriptação.



Fonte: (FIPS 197, 2001).

Figura 7 – Matriz de substituição para decriptação do *byte* xy (em hexadecimal).

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Fonte: (FIPS 197, 2001).

Figura 8 – Combinação dos *bytes* da coluna para decriptação.

$$s'_{0,c} = (\{0e\} \bullet s_{0,c}) \oplus (\{0b\} \bullet s_{1,c}) \oplus (\{0d\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$

$$s'_{1,c} = (\{09\} \bullet s_{0,c}) \oplus (\{0e\} \bullet s_{1,c}) \oplus (\{0b\} \bullet s_{2,c}) \oplus (\{0d\} \bullet s_{3,c})$$

$$s'_{2,c} = (\{0d\} \bullet s_{0,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0e\} \bullet s_{2,c}) \oplus (\{0b\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{0b\} \bullet s_{0,c}) \oplus (\{0d\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0e\} \bullet s_{3,c})$$

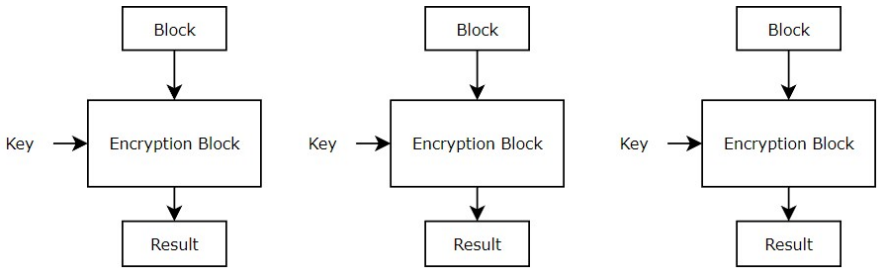
Fonte: (FIPS 197, 2001).

2.1 MODOS DE OPERAÇÃO

2.1.1 MODO ECB

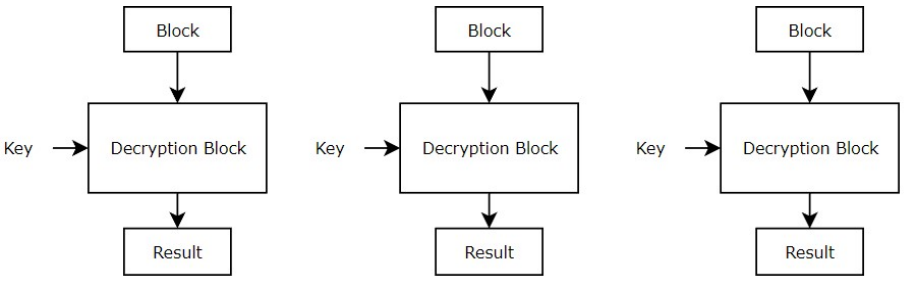
O método mais simples de encriptação é o *cookbook* eletrônico (ECB) que consiste na mensagem ser dividida em blocos, e cada bloco é encriptado ou decriptado separadamente, ilustrado na Figura 9 e na Figura 10.

Figura 9 – Modo de encriptação ECB.



Fonte: O próprio autor.

Figura 10 – Modo de decriptação ECB.

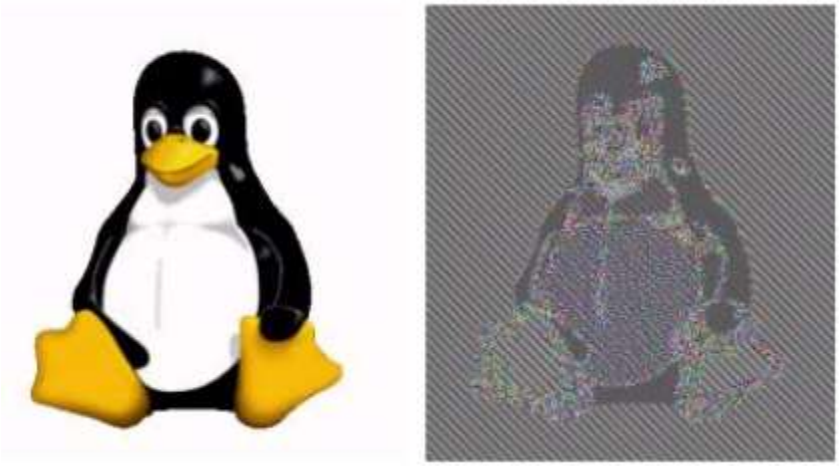


Fonte: O próprio autor.

Este método possuí como vantagem a possibilidade de paralelização tanto da encriptação, como da decriptação, todavia não é comumente utilizado pela falta de difusão, i.e., ele encripta blocos idênticos de informações em blocos idênticos

encriptados, sem realizar a proteção necessária de padrões. Um exemplo deste problema é apresentado na Figura 11, na qual uma imagem é encriptada e seu resultado é apresentado à direita da imagem original, no qual claramente é possível identificar padrões.

Figura 11 – Esquerda: Imagem original. Direita: Imagem encriptada com o modo ECB.

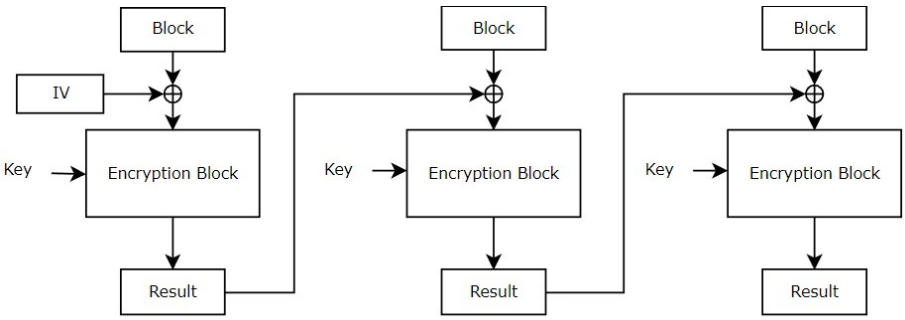


Fonte: (Huang et al, 2013).

2.1.2 MODO CBC

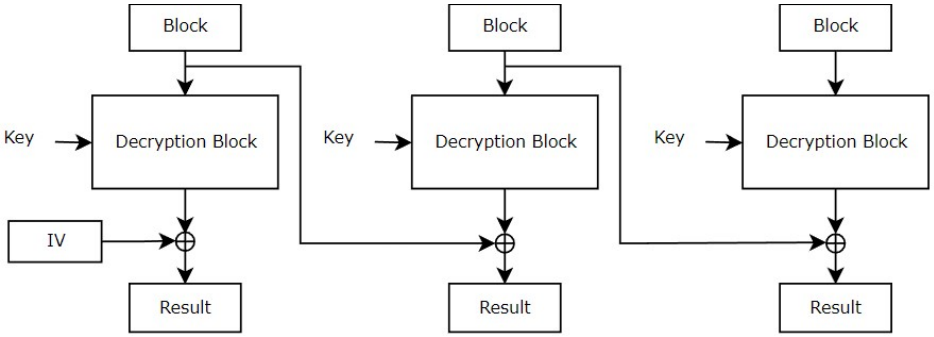
O modo de encadeamento de bloco de cifra (CBC) foi criado por Ehrsam et al (1976) com objetivo de superar os problemas de difusão do modo ECB. Neste modo, assim como nos próximos apresentados, utiliza-se de um vetor de inicialização (IV) de forma que a primeira encriptação é realizada uma XOR do IV com o bloco de entrada e o IV das próximas encriptações é o próprio bloco encriptado, ilustrado na Figura 12.

Figura 12 – Modo de encriptação CBC.



Fonte: O próprio autor.

Figura 13 – Modo de decriptação CBC.



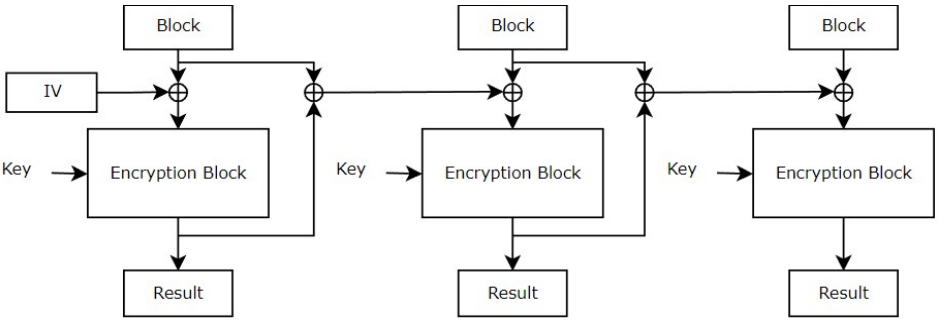
Fonte: O próprio autor.

Este método resolve o problema da falta de difusão do modo ECB e tem possibilidade de paralelização da decriptação, ilustrado na Figura 13, entretanto não é possível na encriptação. Também se observa que, na decriptação, com exceção do primeiro bloco de informação, todos os blocos posteriores podem ser recuperados sem o conhecimento do IV.

2.1.3 MODO PCBC

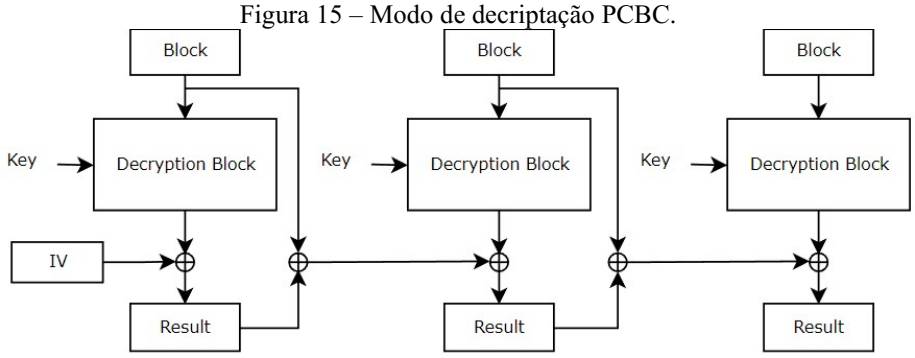
O modo de propagação de encadeamento de bloco de cifra (PCBC) foi criado para criar pequenas mudanças no bloco encriptado e que elas se propaguem indefinidamente tanto na encriptação como na decriptação. Neste modo, é realizada uma XOR entre o IV e o bloco de informações na primeira encriptação e o IV das próximas encriptações é dado pela XOR entre o bloco encriptado e o de informações.

Figura 14 – Modo de encriptação PCBC.



Fonte: O próprio autor.

Figura 15 – Modo de decriptação PCBC.



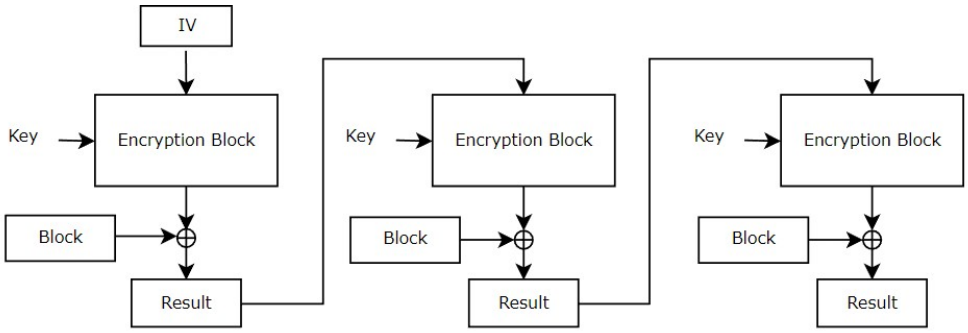
Fonte: O próprio autor.

Este método é utilizado nos protocolos de autenticação *Kerberos v4* e *WASTE*, todavia não é vastamente utilizado porque caso dois blocos encriptados trocarem de posição, não afeta a decriptação de blocos subsequentes. Além disso, a encriptação e decriptação não são paralelizáveis.

2.1.4 MODO CFB

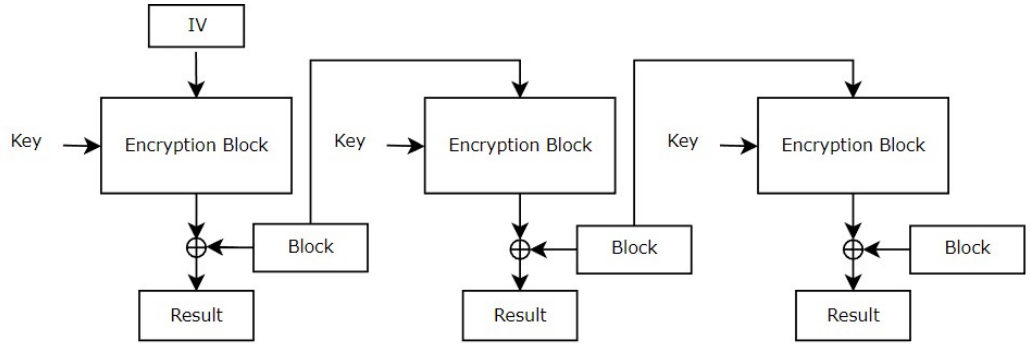
O modo de *feedback* de cifra (CFB) é similar ao modo CBC, em que a encriptação consiste em encriptar o IV, realiza-se uma XOR do resultado dessa encriptação com o bloco de informações, resultando no bloco encriptado, e o IV da próxima operação é o bloco encriptado. Vale destacar que, diferente dos métodos citados, a decriptação utiliza o bloco de encriptação do AES.

Figura 16 – Modo de encriptação CFB.



Fonte: O próprio autor.

Figura 17 – Modo de decriptação CFB.



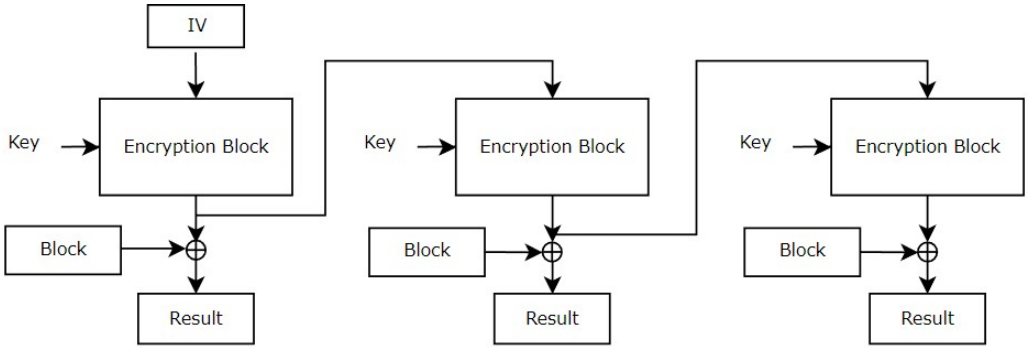
Fonte: O próprio autor.

Este método possibilita uma decriptação paralelizada, entretanto, assim como o modo CBC, o não conhecimento do IV só impede a decriptação do primeiro bloco encriptado.

2.1.5 MODO OFB

O modo de *feedback* de saída (OFB) opera da mesma forma tanto na encriptação como na decriptação: O IV é encriptado e o resultado é o IV da operação seguinte, e este realiza uma XOR com o bloco de informação para obter o bloco encriptado.

Figura 18 – Modo de encriptação e decriptação OFB.



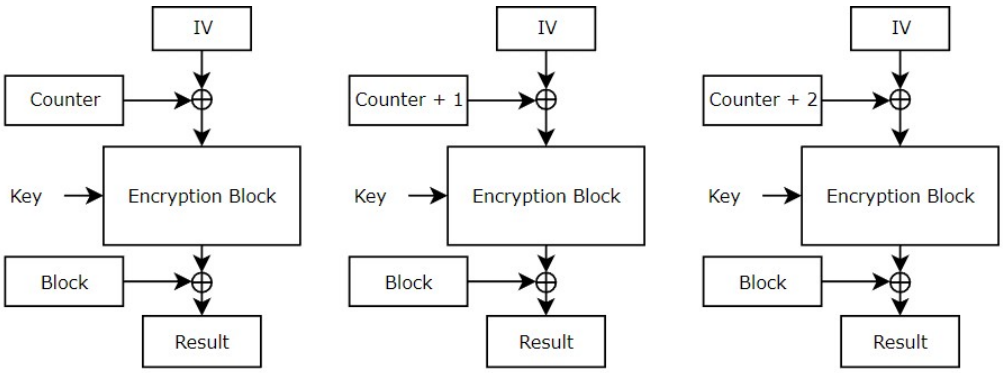
Fonte: O próprio autor.

Este método impossibilita decriptação ou encriptação paralelizável, porém todos os blocos de resultado só podem ser obtidos com o conhecimento do IV.

2.1.6 MODO CTR

O último modo de operação apresentado neste trabalho é o contador (CTR) que, assim como o OFB, efetua as mesmas operações tanto para a encriptação como a decriptação: O IV é constante, e é realizado uma XOR dele com um contador que varia a cada bloco obtido, o resultado é encriptado e realiza-se uma XOR dele com o bloco de informações, obtendo o resultado.

Figura 19 – Modo de encriptação e decriptação CTR.



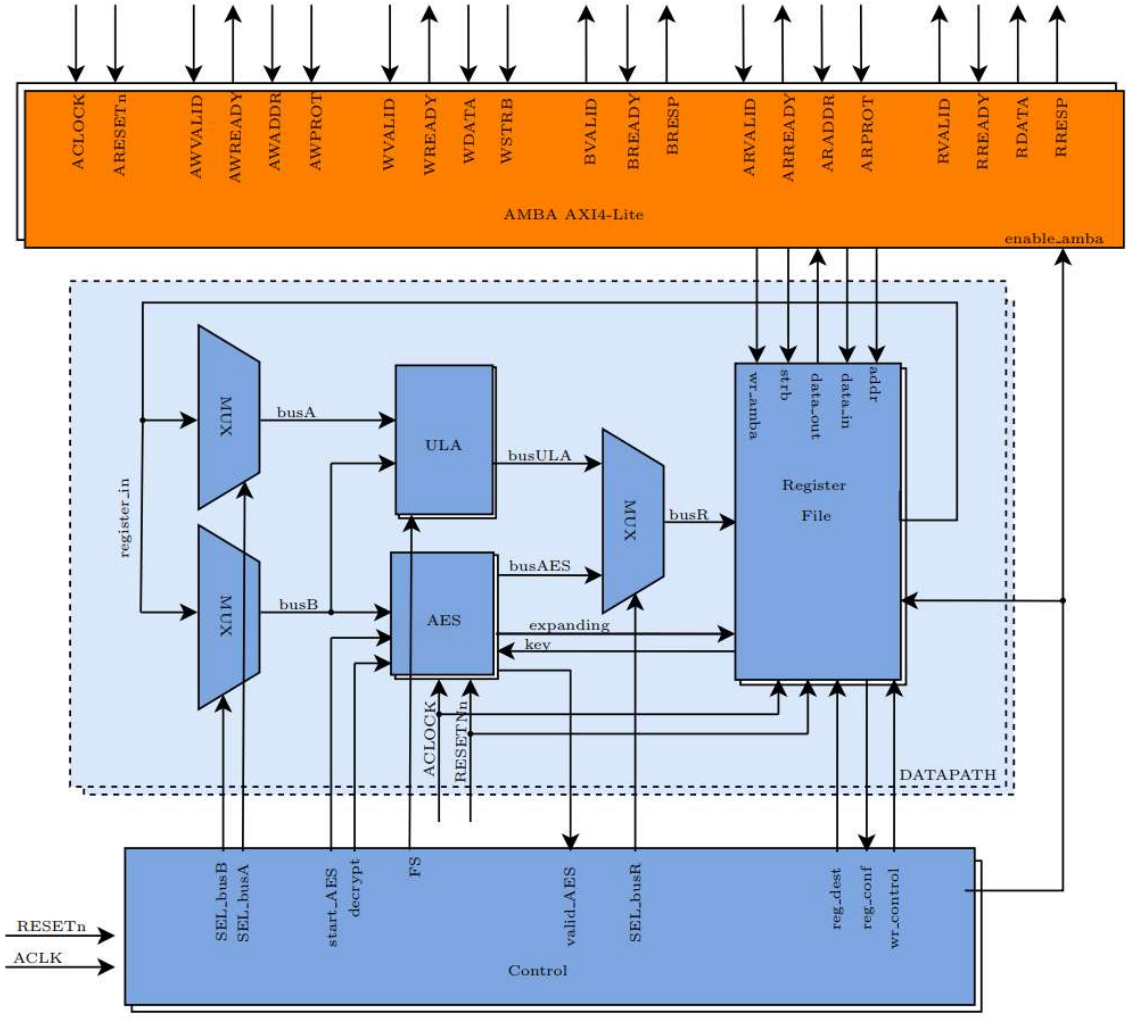
Fonte: O próprio autor.

Este método possuí uma vantagem que só o ECB disponibilizava: Tanto a encriptação como a decriptação são paralelizáveis. E, diferente do ECB, ocorre a difusão, de forma que sem o conhecimento do IV e do contador, é impossível decriptar a informação.

3 ARQUITETURA PROPOSTA

A propriedade intelectual (IP) aqui apresentada, implementa este algoritmo, oferecendo um hardware dedicado para transmitir e receber dados com segurança, permitindo chaves com tamanhos de 128 bits. Ainda pode ser configurado para processar todos os modos de operações citados. A comunicação é feita através de uma interface AMBA AXI4-Lite. A Figura 20 apresenta a arquitetura proposta.

Figura 20 – Arquitetura proposta do IP.



Fonte: O próprio autor.

A interface AMBA AXI4-Lite foi escolhida por, dentre outros motivos: Suportar *designs* de sistemas de alta performance e alta frequência, e possuir canais separados de leitura e escrita, promovendo acesso direto à memória (DMA) com baixo custo.

3.1 MAPA DE MEMÓRIA

Os registradores utilizados são apresentados na Tabela 1, todos tem o tamanho de 32 *bits* e as palavras são organizadas como *Little-Endian*, i.e., extremidade menor primeiro.

Tabela 1 – Mapa de memória do IP.

Endereço	Pseudônimo	Uso	Tipo de Acesso
0x00	key[0]	Primeira parte da chave	Escrita/Leitura
0x04	key[1]	Segunda parte da chave	Escrita/Leitura
0x08	key[2]	Terceira parte da chave	Escrita/Leitura
0x0C	key[3]	Quarta parte da chave	Escrita/Leitura
0x10	r0[0]	Primeira parte do bloco	Escrita/Leitura
0x14	r0[1]	Segunda parte do bloco	Escrita/Leitura
0x18	r0[2]	Terceira parte do bloco	Escrita/Leitura
0x1C	r0[3]	Quarta parte do bloco	Escrita/Leitura
0x20	r1[0]	Primeira parte do IV	Escrita/Leitura
0x24	r1[1]	Segunda parte do IV	Escrita/Leitura
0x28	r1[2]	Terceira parte do IV	Escrita/Leitura
0x2C	r1[3]	Quarta parte do IV	Escrita/Leitura
0x30	r2[0]	Primeira parte do resultado	Escrita/Leitura
0x34	r2[1]	Segunda parte do resultado	Escrita/Leitura
0x38	r2[2]	Terceira parte do resultado	Escrita/Leitura
0x3C	r2[3]	Quarta parte do resultado	Escrita/Leitura
0x40	r3[0]	Primeira parte do contador	Escrita/Leitura
0x44	r3[1]	Segunda parte do contador	Escrita/Leitura
0x48	reg_cmd	Registrador de comando	Escrita/Leitura
0x4C	reg_status	Registrador de status	Leitura

Fonte: O próprio autor.

3.1.1 REGISTRADOR DE COMANDO

A Tabela 2 apresenta os *bits* registrador de comando, que possui a função de definir o modo de operação e iniciar as operações no IP. A Tabela 3 detalha o significado de cada um destes.

Tabela 2 – Registrador de comando.

Bit	**[31]**	**[30:5]**	**[4]**	**[3]**	**[2:0]**
---	começo	Reservado	zerar contador	decriptar	modo

Fonte: O próprio autor.

Tabela 3 – Descrição dos *bits* de comando.

Comando	**Modos de configuração**	**Operação**
[31]: começo	0 1	IP em modo de espera IP realiza operações
[4]: zerar contador	0 1	Mantem o valor do registrador r3 Zera o valor do registrador r3
[3]: decriptar	0 1	Modo de encriptação Modo de decriptação
[2:0]: modo	000 001 010 011 100 101 110 111	Modo ECB Modo CBC Modo PCBC Modo CFB Modo OFB Modo CTR Modo PCBC Modo CFB

Fonte: O próprio autor.

3.1.2 REGISTRADOR DE STATUS

A Tabela 4 apresenta os *bits* registrador de status, que retorna informações sobre o processamento do IP, sendo de somente leitura. A Tabela 5 detalha o significado de cada um destes.

Tabela 4 – Registrador de status.

Bit	**[31]**	**[30:6]**	**[5:3]**	**[2]**	**[1]**	**[0]**
---	concluído	reservado	modo	expansão da chave	decriptação	encriptação

Fonte: O próprio autor.

Tabela 5 – Descrição dos *bits* de status.

Comando	Valores lidos	Operação
[31]: concluído	0 1	O IP está realizando operações O IP está esperando o comando para começo
[5:3]: modo	000 001 010 011 100 101 110 111	Modo ECB Modo CBC Modo PCBC Modo CFB Modo OFB Modo CTR Modo PCBC Modo CFB
[2]: expansão da chave	0 1	Expansão da chave foi concluída Expansão da chave está sendo realizada
[1]: decriptação	0 1	- Decriptação em progresso
[0]: encriptação	0 1	- Encriptação em progresso

Fonte: O próprio autor.

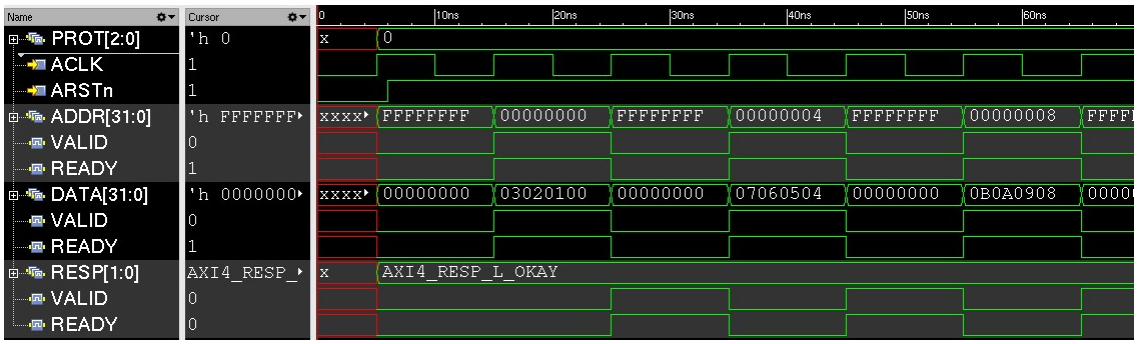
4 RESULTADOS

4.1 SIMULAÇÕES

As formas de ondas apresentadas neste tópico são geradas pelo *software* SimVision™ da Cadence® Incisive®, usado para debugar *designs* digitais, analógicos ou de sinal misto escritos nas linguagens Verilog, SystemVerilog, VHDL e SystemC®, ou uma combinação das mesmas. Também possibilita a realização de múltiplas simulações e a análise do comportamento do *design* e do *testbench* em qualquer ponto do processo de verificação.

Primeiramente foi implementada a interface AMBA AXI4-Lite para realizar a comunicação com o IP, a Figura 21 apresenta as formas de onda para o canal de escrita, onde observa-se que o *handshake* da interface é realizado de forma combinacional, a fim de otimizar o desempenho do IP, e tanto o endereço como o dado a ser escrito está sendo validado ao mesmo tempo, gerando uma resposta válida para a transação.

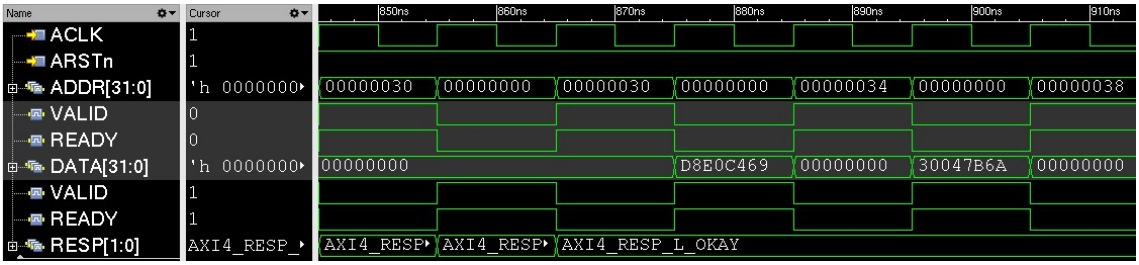
Figura 21 – Formas de onda para o canal de escrita.



Fonte: O próprio autor.

A Figura 22 apresenta as formas de onda do canal de leitura do IP. Da mesma forma do canal de escrita, os sinais de *handshake* da interface é realizado de forma combinacional e esta forma de onda é apresentada no momento em que o IP está finalizando uma encriptação, levando a respostas de erro antes de terminar de realizar a operação, e quando a operação termina a leitura e feita normalmente, gerando um dado válido para a transação.

Figura 22 – Formas de onda para o canal de leitura.



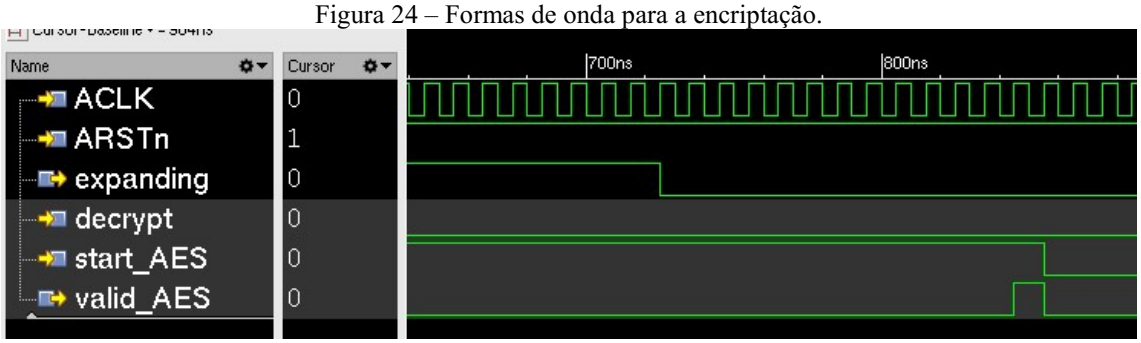
Fonte: O próprio autor.

Posteriormente foi observado o funcionamento da expansão da chave e os procedimentos de encriptação e decriptação do IP e os valores foram comparados com os apresentados nos apêndices A e B de (FIPS 197, 2001). As Figuras 23, 24 e 25 apresentam, respectivamente, as formas de onda para a expansão da chave, encriptação e decriptação.

Figura 23 – Formas de onda para a expansão da chave.

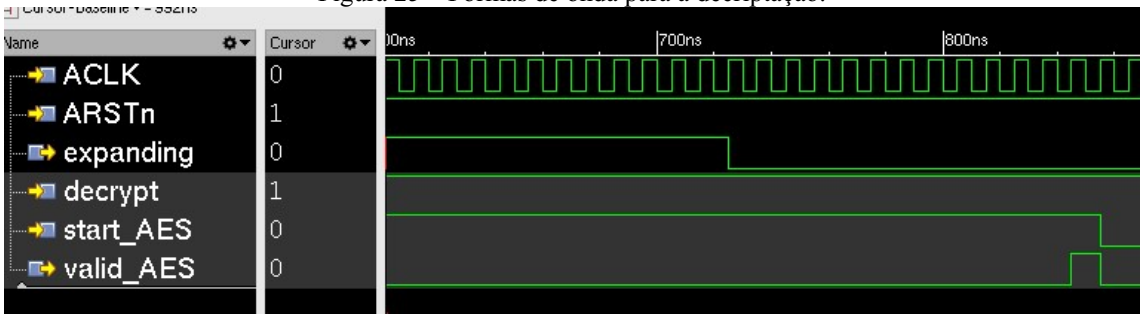
Fonte: O próprio autor.

Figura 24 – Formas de onda para a encriptação.



Fonte: O próprio autor.

Figura 25 – Formas de onda para a decriptação.



Fonte: O próprio autor.

Verifica-se que a expansão da chave é realizada e forma sequencial, com a obtenção de quatro palavras de 32 *bits* da chave expandida por *clock*, esta operação com duração de 42 pulsos. Vale destacar que a chave expandida é armazenada internamente, de forma que se a chave não for alterada, esta operação só necessita ser realizada uma vez. Na encriptação e decriptação observa-se que cada rodada é calculada em um pulso de *clock*, com um total de 12 pulsos até a validação do resultado.

Em seguida foi realizado testes para verificar se os valores das encriptações e decriptações são válidos para todos os modos de operação implementados. Os dados, em hexadecimal, em (1) foram escritos nos registradores dos IP e a Tabela 6 apresenta os resultados obtidos para cada um dos modos.

$$\begin{gathered} Chave = 000102030405060708090A0B0C0D0E0F \\ Bloco1 = 00112233445566778899AABBCCDDEEFF \\ Bloco2 = FFEEDDCCBBAA99887766554433221100 \\ IV = CCCCCCCCAAAAAAAAFFFFFFFFEEEEEEEE \\ contador = 0 \end{gathered} \tag{1}$$

Tabela 6 – Resultados para cada modo de encriptação.

Modo	Primeiro bloco de encriptação	Segundo bloco de encriptação
ECB	69C4E0D86A7B0430D8CDB78070B4C55A	1B872378795F4FFD772855FC87CA964D
CBC	F6217F61B2D50A6AE6791f8C384B1E07	00E6F7C3F089B33AF8D701BEB170AF82
PCBC	F6217F61B2D50A6AE6791f8C384B1E07	1766EA65883AE0BE5DE323B1431CBD54
CFB	79C571F93E6E91590576009881A3AB8E	3F778AF1BA19E580C751137195BCFF23
OFB	79C571F93E6E91590576009881A3AB8E	03219503dd5973187FADC74474E3F047
CTR	79C571F93E6E91590576009881A3AB8E	833441D22AD2BAAEEfAB5EA586AC0DF2

Fonte: O próprio autor.

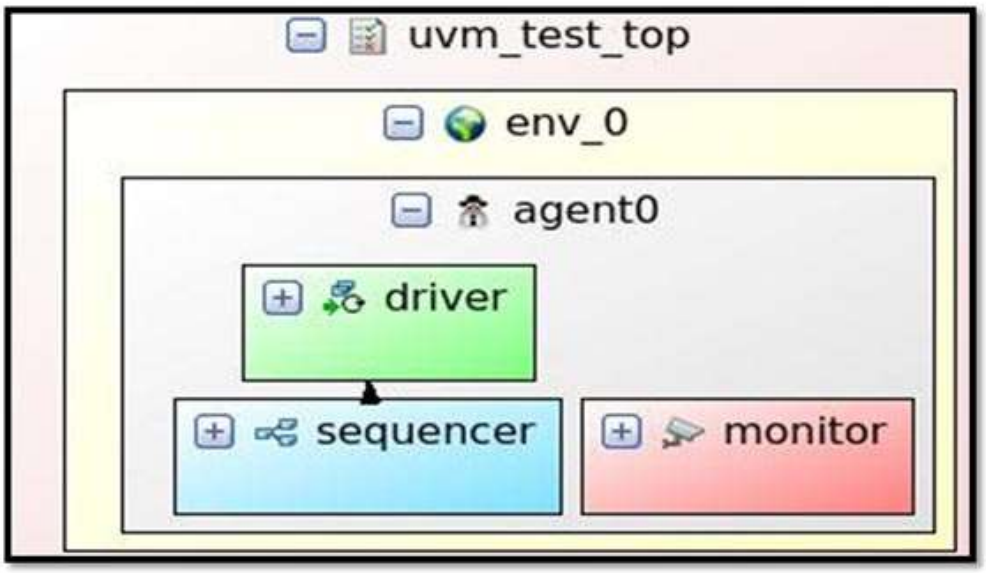
Os resultados obtidos foram validados manualmente e prosseguiu para a realização da verificação automática do IP.

4.2 VERIFICAÇÃO DO IP

Sistemas digitais vêm crescendo em complexidade tornando essencial a utilização de metodologias automáticas de verificação. Os métodos utilizados nos primeiros *designs* digitais, verificar formas de ondas e checagens manuais, não são mais viáveis para os mais recentes.

Para validar o código criado, foi utilizado a metodologia de verificação universal (UVM), que é padronizada para verificar *designs* de circuitos integrados. UVM é uma biblioteca de *system verilog* que é utilizada para facilitar a automatização da verificação. A Figura 26 apresenta os componentes da classe, que utiliza um modelo de referência escrito em linguagem de *software* para comparar os dados e sinais retornados pelo IP com o esperado.

Figura 26 – Ambiente simples de UVM.



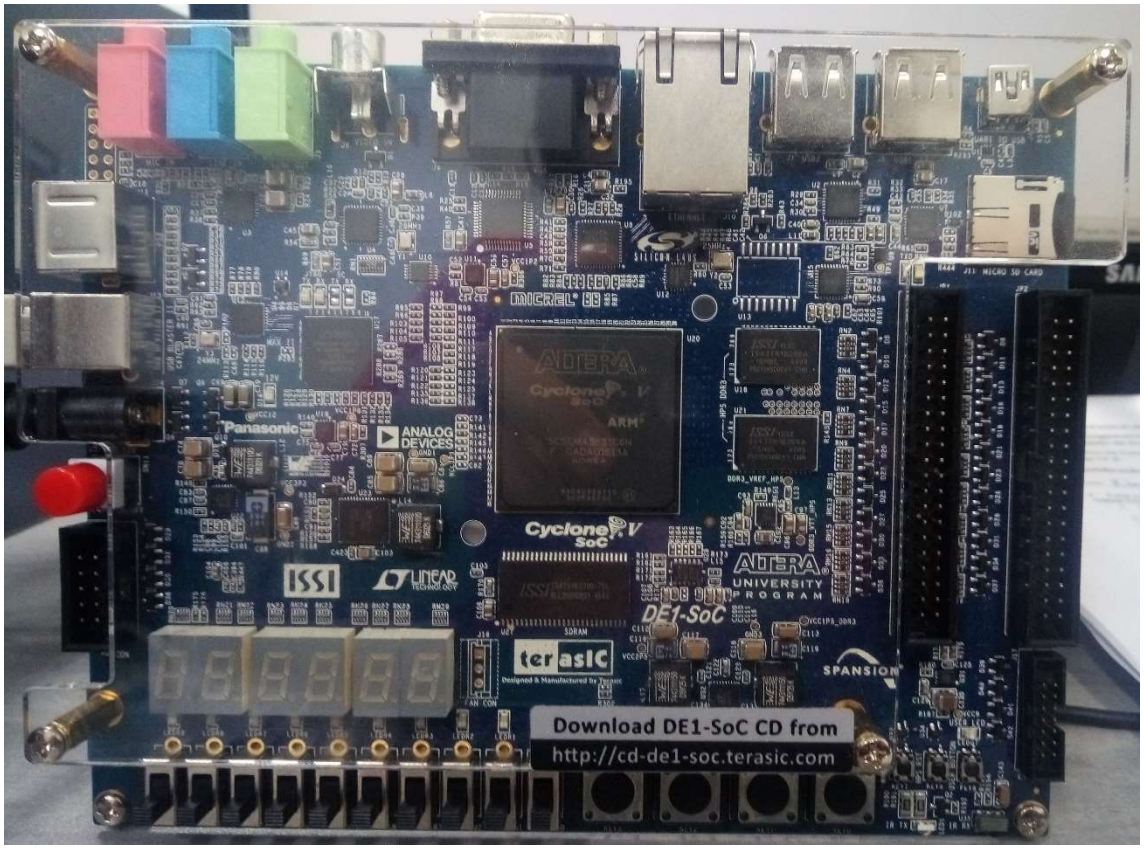
Fonte: (Riviera-PRO 2014.02).

A fim de validar a verificação, estímulos aleatórios foram utilizados para cobrir o maior número de casos de teste possíveis. Entretanto, estes não são o bastante por não ter uma métrica de problemas reais são definidos nestes. Para isso, foi definido um plano de cobertura e os resultados obtidos foram satisfatórios dado que 100% do que era desejado foi coberto.

4.3 IMPLEMENTAÇÃO EM UMA FPGA

A placa escolhida para implementação foi a Altera DE1, ilustrada na Figura 27, que oferece diversos recursos essenciais para uso em laboratórios na UFCG e para o desenvolvimento de sistemas digitais mais sofisticados.

Figura 27 – Placa de desenvolvimento Altera DE1.



O *software* Intel® Quartus® Prime foi utilizado para programar a FPGA. O *clock*, disponível na placa, de 50 MHz foi usado e uma chave foi ligada ao *reset* do *design*. Foi realizada uma comparação de cada valor apresentado na Tabela 6 com o resultado obtido na FPGA e designado para LEDs a veracidade destes. Todos os LEDs designados foram acesos, indicando que a implementação obteve os resultados esperados.

Por fim, foi estudada a implementação da encriptação de um cartão de memória. Inicialmente foram avaliados projetos existentes para realizar o acesso direto. Tanto o acesso por *Serial Peripheral Interface* (SPI) quanto por SD de 1 a 4 *bits* apresentou-se viável devido a impossibilidade de designar pinos ao cartão. A realização somente é possível a partir de um acesso ao *Hard Processing System* (HPS) disponibilizado pela empresa desenvolvedora da FPGA.

5 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho foi discutido a funcionalidade do AES, aplicações para acréscimo da segurança com a utilização de diversos modos de realimentação, a criação de um IP com a funcionalidade de encriptar ou decriptar dados com a utilização dos métodos citados, além da implementação do mesmo em uma FPGA.

A arquitetura proposta consiste em uma interface de comunicação, um banco de registradores, unidades lógicas aritméticas, o AES, e um controle, que determina o modo de operação do IP.

Os valores obtidos nas simulações condizem com os esperados pelos modelos de referência, e a implementação na FPGA com o *clock* máximo disponível na placa mostraram que o IP criado é viável para aplicações futuras.

Como sugestão de trabalhos futuros nesta área, tem-se a encriptação de dados de um sistema de armazenamento ou de sinais enviados para USB ou VGA da placa, bem como a implementação, junto a um processador e outros IPs, para criação de um *system on chip*. Dessa forma, percebe-se como o presente estudo reitera a relevância da ferramenta e a abrangência de suas possibilidades de aplicação.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] GOLDREICH, O. “Foundations of Cryptography Volume II Basic Applications”, Cambridge University Press, 2004.

[2] Federal Information “Advanced Encryption Standard (AES)”, Processing Standards Publication 197, Novembro 2001. Disponível em: <https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf>.

[3] DESHPANDE, A.M.; DESHPANDE, M.S.; KATATANAVAR, D.N. “FPGA Implementation of AES Encryption and Decryption”, IEEE, 2009.

[4] MANO, M.M. “Computer System Architecture Third Edition”, Pearson, 1992.

[5] HUANG, K.T.; CHIU, J.H.; SHEN, S.S. “A Novel Structure with Dynamic Operation Mode for Symmetric-Key Block Ciphers”, Internation Journal of Network Security & Its Applications, 2013. Disponível em: <http://airccse.org/journal/nsa/0113nsa02.pdf>.

[6] Computer Security Division’s Security Technology Group “Block cipher modes”, Criptographic toolkit, NIST, 2013.

[7] Computer Security Division’s Security Technology Group “Proposed modes”, Criptographic toolkit, NIST, 2013.

[8] FERGUSON, N.; SCHNEIER, B.; KOHNO, T. “Cryptography Engineering: Design Principles and Practical Applications”, Wiley Publishing, 2010.

[9] “Stream Cipher Reuse: A Graphic Example”, Criptosmith, 2015. Disponível em: <https://cryptosmith.com/2008/05/31/stream-reuse/>.

[10] MOELLER, B. “Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures”, 2004. Disponível em: <http://www.openssl.org/~bodo/tls-cbc.txt>.

[11] EHRSAM, W.F.; MEYER, C.H.W.; SMITH, J.L.; TUCHMAN, W.L. “Message verification and transmission error detection by block chaining”, US Patent 4074066, 1976.

[12] KOHL, J. “The Use of Encryption in Kerberos for Network Authentication”, ISBN 0387973176, 1990. Disponível em: <https://dsns.cs.nctu.edu.tw/research/crypto/HTML/PDF/C89/35.PDF>.

[13] LIPMAA, H.; ROGAWAY, P.; WAGNER, D. “Comments to NIST concerning AES modos of operation: CTR-mode encryption”, 2000.

[14] “AMBA® AXI™ and ACE™ Protocol Specification”, 2011. Disponível em: <http://www.gstitt.ece.ufl.edu/courses/fall15/eel4720_5721/labs/refs/AXI4_specification.pdf>