ARTIGO CIENTÍFICO
ÁREA DE CONCENTRAÇÃO

CIÊNCIA E TECNOLOGIA

ARQUITETURA E VERIFICAÇÃO DE UM SISTEMA DIGITAL BASEADO NA CRIPTOGRAFIA AES 128 BITS

Leandro Barbosa Lima[1], Yuzo Iano[2]
Doutorando em Engenharia Elétrica e Computação[1], Doutor em Engenharia Elétrica[2]

RESUMO: Ao desenvolver um novo sistema, é importante verificar que o sistema está em conformidade com os requisitos documentados e fornece recursos específicos. Dessa forma o design e verificação, comumente conhecida como front-end no fluxo de projetos de circuitos integrados digitais, concentram o máximo de atenção nesta fase de projeto afim de garantir a funcionalidade do dispositivo de forma segura. Ao garantir de forma segura a funcionalidade do sistema, através da implementação de componentes de virificação funcional para a validação "verificação funcional" a metodologia eRM (Metodologia de Reuso com Linguagem-e), para gerar estimulos no dispositivo AES (Encryption Standard Advance). A verificação é realizada no dispositivo digital AES que implementado em HDL (Linguagem de Descrição de Hardware) Verilog. Tem como funcionalidade criptografar e descriptografar textos, podendo gerar chaves com tamanho de 128 bits. A fim de asegurar a funcionalidade do dispositivo, o recurso de cobertura de código, também foi utilizada. Há muitas vantagens em usar tal recurso no design, podendo ser utilizado na verificação functional através de assertions SVA, pois são nativamente integrados ao idioma na linguagem system verylog, podem ser verificados na simulação e na verificação formal e são convenientes para os designers usarem durante a codificação.

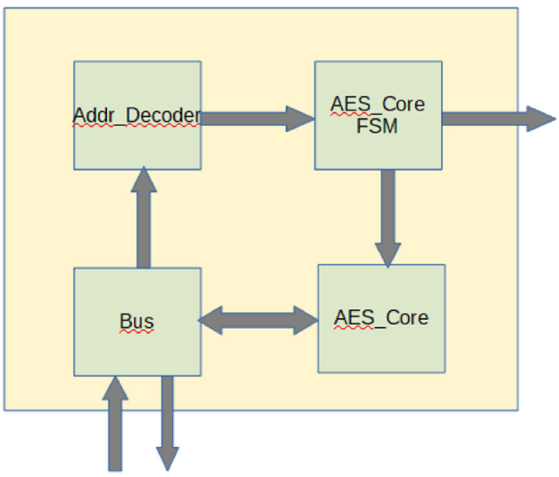
Palavras-chave: Aes, Criptografia, HDL, Segurança de Hardware, Verificação Funcional, Verilog.

INTRODUÇÃO

A implementação deste trabalho foi desenvolvida em HDL (Linguagem de Descrição de Hadware) utilizando como base, *Advanced Encryption Standard* (AES), que fornece um meio de proteção de dados utilizando o algoritmo Rijndael. Este algoritmo recebe como entrada 128 bits de dados do tipo texto e também a chave com tamanho de 128 bits. Após várias rodadas de cálculos, o algoritmo produz uma versão chiphered de 128 bits de dados de texto, como saída. Durante as rodadas de algoritmo, o bits de dados estão sujeitos a substituição, operações com mudança de dados, operações com dados misturados e operações de adição com uma versão de chave ampliada com cifra original. Todas estas operações são realizadas no Campo de Galois [1]. Na figura 1, o diagrama de blocos mostra os sub-blocos internos do sistema.

A criptografia/descriptografia do sistema AES é uma máquina de estado que controla as etapas para converter o conjunto de bytes para um número de bytes iniciais diferentes. O formulário usado para fazer isso utiliza o método ECB "*Electronic Codebook Mode*", que permite realizar etapas para geração no modo de descriptografar arquivo criptografado. Este bloco possui etapas para geração de dados que utiliza operação lógica XOR dos bytes, que para gerar elementos produzidos também com outros elementos, substituindo os bytes, utilizando uma tabela de valores predeterminados, através do deslocamento e troca de posições lógicas entre bytes [2].

FIGURA 1 Bloco D-IP CRIPT



Fonte: o autor

O algoritmo consiste em quatro camadas que manipula todos os 128 bits do caminho de dados. Assim, existem quatro tipos diferentes de camadas a cada rodada, com exceção da primeira e última camada, Essas camadas são:

1. Camada de Adição de Chave e Camada de Substituição de Bytes (S-Box)

2. Camada ShiftRows e Camada MixColumns

Além disso, a programação de teclas calcula teclas redondas ou subchaves da chave AES original.

Para descriptografia, todas as camadas devem ser invertidas, ou seja, a Camada de Substituição de Bytes se torna a Substituição de Bytes Invertidos, camada de aplicação e assim por diante. Além disso, a ordem das subchaves é revertida [3].

1 REGISTROS INTERNOS

Os registros podem ser acessados usando o barramento.

Bit 0: Máscara de Interrupção Criptográfica. Se definido, pode enviar sinal de interrupção;

Bit 1: Interrupção Encriptação/Desencriptação de forma completa.

1.1 DEFINIÇÃO DA MASCARA

a. Key 0, Key 1, Key 2 e Key 3, acesso RW: cada um contém quatro bytes da chave, cada uma delas é uma coluna da matriz de chave;

b. Bloco 0, Bloco 1, Bloco 2 e Bloco 3, acesso RW: cada um contém quatro bytes do bloco, cada um é uma coluna da matriz de bloco, de uma maneira semelhante como a matriz chave; e

c. O bloco contém os dados a serem processados, ou os dados quando o processamento for concluído.

1.2 CAMPO FINITO ARITIMÉTICO

O algoritmo funciona num campo finito aritmético que é diferente do padrão aritmético de números inteiros. Há um número limitado de elementos no campo finito, e todas as operações realizadas no campo finito resultando num elemento dentro do campo.

O nome do campo finito utilizado para esta operação é nomeado após os criadores do algoritmo serem chamados de Campo Finito de Rijndael. Neste campo finito, cada byte é interpretado como um polinômio. Por exemplo, o binário: {} é 01010011 interpretado como o polinômio: 0x7 + 1x6 + 0x5 + 1x4 + 0x3 + 0x2 + 1x1 + 1x0 = $x^6 + x^4 + x + 1$.

Ainda neste campo, as operações de soma e subtração entre A e B são simplesmente um XOR entre A e B. A operação de multiplicação é uma operação mais complicada, devido o resultado da operação estar em campo finito, então o resultado deve ser um polinômio com grau máximo 7. A multiplicação de dois polinômios com grau máximo 7, pode ser de no máximo 14 graus. A solução para este problema é resolvido, tendo o resto da divisão de polinómio resultante pelo polinômio $x^8 + x^4 + x^3 + x + 1$ (binário: 100010011 {}). O polinômio $x^8 + x^4 + x^3 + x + 1$, é chamado reduzindo o polinomial. Para calcular A * B. sendo:

a. Multiplicar A e B como polinômios.

b. Dividir o resultado pela redução polinômio $x^8 + x^4 + x^3 + x + 1$.

c. O resultado é o resto da divisão anterior.

Alternativamente, para calcular P=A*B.

a. P=0, AUX = 0;

b. Repetir por 8 vezes, uma vez para cada bit; e

1) se LSB de B=1, então P e A=XOR;

2) guarda o bit MSB de A em AUX;

3) um desvio para a esquerda;

4) se o bit armazenado em AUX é 1, XOR A com 8x00011011 (x^8 do polinômio redução está implícito); e

5) deslocamento para a direita B, Divisão A/B é uma multiplicação de pelo inverso multiplicativo de B (A*B = 1). O inverso multiplicativo de B está o número que multiplicado por B (descrito acima) resulta em o número 1.

1.3 CHAVE AES

As chaves são interpretadas como matrizes. Por exemplo, o bloco A00A01A02: A13A14A15 teria de ser interpretado como a matriz [4].

1.3.1 Passo AES SubBytes

O passo AES SubBytes é uma simples substituição de cada byte do bloco utilizando uma transformação a fim de o inverso multiplicativo do byte (neste caso, usa-se 0 como o inverso multiplicativo de 0). Normalmente, uma tabela de pesquisa é utilizada para determinar o valor transformado a partir do byte inicial, e uma outra para determinar o inverso.

O circuito que converte o byte original para o byte substituído é geralmente chamado S-Box. O circuito inverso é chamado Inversa S-Box.

Os circuitos S-Box Rijndael e Inversa S-Box Transforma, têm como operação encontrar o valor equivalente a um número S-Box. Dessa forma, um valor em hexadecimal (0x7A), localiza a linha que começa com a coluna 7, que começa com um valor desejado na intersecção em hexadecimal (0xDA).

1.3.2 Passo AES ShiftRows

O passo ShiftRows é muito simples, cada linha terá suas linhas desviadas por um valor fixo. A primeira linha será mantida, a segunda será deslocada por 1, a terceira será deslocado por 2, e a quarta será deslocada por 3.

1.3.3 Passo AES MixColumns

Para os MixColumns, a matriz bloco deve ser multiplicada pela matriz de transformação utilizando o campo finito aritmético. Para simplificar, cada coluna pode ser multiplicada individualmente.

1.3.4 AES Rodada de Adição de Chave

Para o AddRoundKey, a chave gerada para aquela rodada usando campo finito aritmético deve ser adicionada à matriz bloco (soma é XOR).

O Inverso AddRoundKey é exatamente o mesmo que AddRoundKey, uma vez que uma operação XOR é simétrica. Por exemplo, tendo o bloco A = {01010011} e a chave B = {10100110}, o resultado de C = A XOR B é {11110101}.

Para recuperar um, faz-se A = C XOR B, resultando em A = {01010011}.

1.3.5 Programação de Chave

Agendamento de chave é o processo onde cada tecla rodada é usada para gerar a chave rodada seguinte.

Para isso, primeiro é necessário aplicar uma transformação para a última linha da chave. A última linha é rodada, em seguida o S-Box é aplicada a cada bit, em seguida, para cada rodada de um determinado valor é XOR, com o primeiro byte. Este valor é chamado de RCON (*Round Constant*) e calculado fazendo 2 (duas), rodadas e usando um campo finito aritmético, de modo que a primeira rodada seja: RCON HEX 0x01, 0x02, em seguida, 0x04, 0x08, 0x10, ..., 0x80, 0x1B, 0x36.

Depois disso, a primeira linha é XOR com este valor, a segunda linha é XOR com a nova primeira linha e assim por diante.

2 PROCESSO DE CRIPTOGRAFIA

O processo de criptografia é realizado executando as etapas seguintes no bloco e ar-

mazena os resultados de volta no bloco:

a. No passo inicial, o bloco é XORed com a chave original.

b. As próximas nove rodadas operam o mesmo, sendo:

1) realização de SubBytes;

2) realização de ShiftRows;

3) realização de MixColumns; e

4) geração do RoundKey e realização do AddRoundKey .

c. A última rodada, sendo:

1) realização de SubBytes;

2) realização de ShiftRows; e

3) geração do RoundKey e realização do AddRoundKey.

3 DECODIFICAÇÃO DO PROCESSO

Para o processo de decodificação, é preciso realizar as operações inversas da criptografia na ordem reversa. Sendo assim, são executas as etapas seguintes no bloco e os resultados são armazenados de volta no mesmo bloco:

a. Passo 1: gerar todas as chaves redondas para ser utilizado pela ordem inversa.

b. Passo 2: segue a ordem, sendo:

1) obtenção do próximo RoundKey em ordem inversa e realização do AddRoundKey;

2) realização de MixColumns inversas;

3) realização de ShiftRows inversas; e

4) realização de SubBytes inversas.

Quando ocorrer um erro, o sinal flag de erro deve ser ser definido como um sinal de Erro.

O dispositivo opera em dois modos: criptografia e descriptografia. O processo de criptografia é realizado executando as seguintes etapas no bloco D-IP CRIPT e armazenando os resultados novamente no bloco D-IP CRIPT.

4 MODOS DE OPERAÇÃO

O dispositivo opera em dois modos: criptografia e descriptografia. O processo de criptografia é realizado executando as seguintes etapas no bloco e armazenando os resultados novamente no bloco:

a. Na etapa inicial, o bloco é XORed com a chave original.

b. As próximas 9 rodadas funcionam da mesma maneira:

1) execuçao de SubBytes;

2) execução de ShiftRows;

3) execução de MixColumns;

4) geração de chave redonda e execução de addRoundKey;

c. Na última rodada:

1) execução de SubBytes;

2) execução de ShiftRows; e

3) geração de chave redonda e execução de addRoundKey.

O processo de descriptografia é realizado executando as seguintes etapas no dispositivo, armazenando os resultados no dispositivo:

a. A primeira execução é gerar todas as chaves redondas para serem usadas na ordem inversa.

b. Na primeira rodada:

1) obtenção do último RoundKey e execução do AddRoundKey;

2) execução de ShiftRows inversos; e

3) execução de SubBytes inversos.

5 RESULTADOS

Este dispositivo tem 54.576 Registros e 1195 Flip-Flops. A principal preocupação no desenvolvimento foi a otimização de recursos, sempre realizando um *trade-off* entre a criptografia e descriptografia, considerando o objetivo de uso em ASIC.

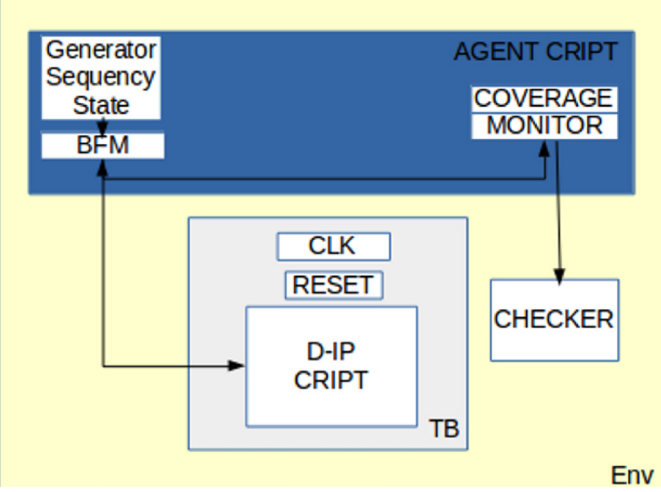
O processo de verificação funcional do dispositivo IP CRIPT (AES) utilizou a metodologia ERM, validando a funcionalidade lógica do circuito digital descrito em Verilog [5].

Os componentes do ambiente são feitos usando o idioma em uma linguagem de verificação de hardware chamada de Linguagem “e”.

O ambiente possui um gerador de estímulos, chamado gerador de sequência, que envia sinais para o modelo funcional de barramento e converte os geradores de entrada de um nível alto para um formato de estímulo compreensível pelo dispositivo.

A figura 2 mostra a arquitetura do Ambiente de Verificação do dispositivo IP CRIPT AES, contendo os componentes de verificação funcional.

FIGURA 2 Ambiente de verificação do dispositivo CRIPT (AES)



Fonte: o autor

A simulação produzida verificou e testou a funcionalidade do dispositivo após todas as instruções contidas na ROM serem executadas na DUT (Dispositivo Sob Teste). Isso implica gerar em um código-fonte simples para testar cada estado da interface de barramento além do bloco funcional de criptografia/descriptografia para garantir a funcionalidade de acordo com as especificações. Os cenerios verificados foram os seguintes:

a. Verificação de Reset: verifica se o dispositivo AES é reiniciado adequadamente e se os registros e máquinas de estado se encontram no estado inicial e durante a execução verifica se é obtido as redefinições para ver se os sinais também foram alterados para o estado inicial [6].

b. Verificação de interrupção: verifica se foi enviado um sinal com a finalidade de bloquear o processo obedecendo o protocolo, quatro palavras são esperadas até o bloco enviar uma interrupção para o dispositivo AES ao final do sinal criptografado ou lido no bit de configuração de registro, deve-se aguardar uma possível resposta, sendo:

1) chave criptografada corretamente.

2) texto criptografado corretamente.

c. Verificação de chave descriptografada: verifica se recebe do bloco obedecendo ao protocolo quatro palavras ao dispositivo AES, espera até que o bloco envie o resultado final do sinal de interrupção descriptografado ou usando um registrador para sinalizar interrupção no bit 1. A descriptografia possui dois modos onde um pode descriptografar usando a chave original e outro usando uma chave já criptografada. Para este caso de verificação da chave não verificada, trabalhou-se apenas com o texto descriptografado.

d. Verificação de geração de chaves: verifica a utlização de registrador de chaves para executar uma fase de criptografia gerando na chave do primeiro momento de rodadas a serem usadas em outro modo para criptografar os dados.

e. Verificação em todos os tipos de registros de configuração: possibilidade dos dados serem processados, afim de garantir que não haja erros usando muitas combinações possíveis.

f. Verificação de erro: verifica-se ao ler ou gravar novos dados na chave ou no texto dos registros.

7. Verificação de descriptografia com chave criptografada: verifica se foi enviado bloqueio dos dados em mais de uma chave para descriptografar. O modo utilizado é de 2 bits, sendo que, para a geração de chaves necessárias para descriptografar utiliza 4 operações de 32 bits.

CONCLUSÃO

A metodologia de reuso utilizada no dispositivo de criptografia baseado no AES mostrou-se muito eficaz para a produção de circuitos integrados digitais.

Falhas de simulação foram encontradas e melhorias foram executadas com a cobertura de código. Quando utilizadas técnicas de verificação funcional, com criação de componentes de checagem de dados, houve um ganho ainda maior na simulação demonstrando as falhas com rapidez e facilidade [7].

Este projeto foi apoiado pelo Departamento de Comunicação da Universidade de-Campinas, Faculdade de Engenharia Elétrica e Computação e também pelo Laboratório de Comunicação Visual, que forneceu as ferramentas e os softwares usados para desenvolver este projeto.

ARCHITECTURE AND FUNCTIONAL VERIFICATION OF A DIGITAL SYSTEM BASED ON AES 128 BITS ENCRYPTION

ABSTRACT: When developing a new system, it is important to verify that the system conforms to documented requirements and provides specific resources. In this way design and verification, commonly recognized as a front end in the digital integrated circuit design flow, focus the utmost attention on this design phase in order to ensure device functionality in a safe manner. By securely ensuring system functionality by implementing functional verification components to validate "functional verification" the eRM (e-Language Reuse Methodology) methodology to generate stimuli in the Encryption Standard Advance (AES) device. Verification is performed on the AES digital device that is implemented

in Verilog HDL (Hardware Description Language). Its function is to encrypt and decrypt texts, and can generate keys with a size of 128 bits. In order to ensure device functionality, the code coverage feature was also used. There are many advantages to using such a feature in design and can be used for functional verification through SVA assertions, as they are natively integrated into the system verylog language, can be verified in simulation and formal verification, and are convenient for designers to use during coding.

Keywords: AES, Encryption, Functional Verification, Hardware Security, HDL, Verilog.

REFERÊNCIAS

[1] Daemen, Joan, Rijmen, Vincent “The Design of Rijndael: AES - The Advanced Encryption Standard” Springer, 2002.

[2] Ian Elliott, Peter Minns “FSM-based Digital Design using Verilog HDL” Wiley, 2008.

[3] Tim Good and Mohammed Benaissa. AES on FPGA from the fastest to the smallest. In Cryptographic Hardware and Embedded Systems–CHES 2005, Springer, 2005.

[4] NIST FIPS Pub. 197: Advanced encryption standard (AES). Federal Information Processing Standards Publication, 197:441–0311, 2001. Disponível em http://csrc. nist.gov/publications/fips/fips197/fips-197.pdf. 15

[5] A. S. M. L.Swarna Jyothi, Harish R, “Reusable Verification Environment for verification of Ethernet packet in Ethernet IP core, a verification strategy- an analysis,” IJCSNS Int. J. Comput. Sci. Netw. Secur., vol. 8, n. 11, pp. 226–236, 2008.

[6] J. M. F. Costa, Elton B, Victor A. Perone, Thayse L. A. Barbosa, Elmar U. K. Melcher, “Verificação Funcional para Sistemas Digitais utilizando System Verilog,” Natal, 2012.

[7] Laurence S. Bisht, Dmitry Korchemny, Erik Seligman “SystemVerilog Assertion Linting: Closing Potentially Critical Verification Holes”

Leandro Barbosa Lima é doutorando em Engenharia Elétrica e Computação pela Universidade Estadual de Campinas – UNICAMP, Mestre em Engenharia Elétrica e Computação pela Universidade Estadual de Campinas – UNICAMP, Especialização em Cybercrime e Cybersecurity Prevenção e Investigação de Crimes Digitais pela Unyleya. Possui cursos na área circuitos integrados digitais, segurança e proteção cibernética. Atualmente, exerce a função de chefe da seção técnica de informática e da subseção de segurança da informação do Hospital de Guarnição de Porto Velho. Já foi instrutor nos cursos de projetista de circuito integrado digital nos centros de treinamento: CTI Renato Archer e CTSP/USP. Pode ser contatado pelo e-mail: lbarbosalima@gmail.com

Yuzo Iano é doutor em Engenharia Elétrica pela Universidade Estadual de Campinas/SP – UNICAMP. Atualmente é professor titular da Unicamp. Tem experiência na área de Engenharia Elétrica, com ênfase em Telecomunicações, Eletrônica e Tecnologia da Informação. Ele trabalha nos seguintes assuntos: transmissão digital e processamento de imagens/áudio/vídeo/dados, hdtv, televisão digital, redes 4G/5G, middleware, transmissão, canalização, transmissão de sinais de televisão, reconhecimento de padrões, codificação digital de sinais, transmissão e armazenamento de dados e cidades inteligentes /digitais. Pode ser contatado pelo e-mail: yuzo@decom.fee.unicamp.br