ARQUITETURA E DESEMPENHO DE UM CRIPTOPROCESSADOR VLIW

Fábio Dacêncio Pereira, Edward David Moreno Ordonez

UNIVEM-Centro Universitário Eurípides de Marília
Av Hygino Muzzi Filho 529, CEP 17525-901, Marilia, S.P.

{fabiopereira, edmoreno}@fundanet.br

ABSTRACT

Neste artigo apresenta-se a arquitetura e o desempenho de um processador específico para criptografia de algoritmos simétricos, nomeado como Criptoprocessador VLIW. No artigo são discutidos detalhes da arquitetura e especifica-se seu conjunto de instruções. Módulos especiais para a execução de operações próprias desses algoritmos simétricos foram projetados para aumentar a performance e simplificar o programa fonte.

É importante salientar que os módulos especiais, que diferenciam este criptoprocessador, não são específicos para um determinado algoritmo de criptografia. Estes foram projetados de forma a serem pré-configurados de acordo com as características do algoritmo que será executado.

O criptoprocessador foi descrito utilizando a linguagem VHDL e um protótipo foi sintetizado e implementado em um FPGA Virtex II Pro

1. INTRODUÇÃO

A segurança nas transmissões de informações torna-se cada vez mais importante, exigindo que novas técnicas e algoritmos de criptografia de informações sejam desenvolvidos para promover um ambiente de transmissão seguro e veloz.

A criptografia é uma das ferramentas para se obter um grau de segurança confiável na transmissão de informações. Existem, basicamente, dois tipos de criptografia: a criptografia simétrica e a assimétrica. A primeira utiliza a mesma chave tanto no processo de cifragem como no de decifragem. Na criptografia assimétrica tem-se um par de chaves, uma pública e outra privada, sendo que de forma geral todo texto cifrado com a chave publica só pode ser decifrado com a chave privada.

Neste artigo, apresenta-se a proposta e a implementação de um criptoprocessador VLIW dedicado para a execução de algoritmos criptográficos simétricos, utilizando a tecnologia FPGA. Apresenta-se também os resultados obtidos na implementação em FPGA, discutindo detalhes da arquitetura e especificando o seu conjunto de instruções.

Como ponto de partida para o projeto do criptoprocessador foi realizado um estudo detalhado dos algoritmos de criptografia RC6[10], Serpent[13], Cast-128[21], MARS[12], Twofish[14], Magenta[11], Frog[15], BlowFish[16] e IDEA[20] com destaque para os algoritmos DES[7], AES[8] e RC5[9].

O objetivo principal do estudo foi identificar e quantificar as operações realizadas com maior freqüência e operações específicas de cada algoritmo. Ao final deste levantamento, conseguiu-se visualizar os fatores que influenciam diretamente no projeto do criptoprocessador, definindo as características da arquitetura proposta como, por exemplo: número de registradores, operações da unidade lógica e aritmética, buffers, módulos especiais, entre outras. Além disso notou-se a ocorrência significativa de operações como, deslocamento, rotação, permutação, substituição de bits e a operação lógica XOR (ou-exclusivo).

Neste contexto, características importantes do criptoprocessador, como módulos e instruções especiais, foram projetadas e implementadas em FPGA. Sendo que o objetivo principal foi projetar uma arquitetura que suportasse o maior número de algoritmos simétricos e atingisse um bom desempenho geral.

Este criptoprocessador suporta uma série de algoritmos simétricos, inclusive algoritmos atuais, que utilizam chaves de 128 bits ou maior. É importante salientar que os módulos especiais propostos e implementados, que diferenciam este criptoprocessador, não são específicos para um determinado algoritmo de criptografia. Estes módulos foram projetados de forma a serem pré-configurados de acordo com as características do algoritmo que será executado. Esta característica ressalta uma das qualidades do criptoprocessador VLIW, a quantidade de algoritmos suportados.

2. ARQUITETURA VLIW

A arquitetura VLIW pode ser definida a partir de dois conceitos: (i) o micro-código horizontal e (ii) o processamento superescalar. Uma máquina VLIW típica tem palavras com centenas de bits. O Criptoprocessador VLIW proposto neste trabalho tem 160 bits por palavra. As operações a serem executadas simultaneamente são armazenadas nesta palavra VLIW. O micro-código horizontal de cada palavra VLIW é formado por opcodes e dados que especificam as operações a serem executadas em diferentes unidades funcionais.

A arquitetura VLIW busca uma palavra VLIW em um único endereço de memória e cada unidade funcional (UF) executa uma das diferentes operações contidas na palavra. As principais vantagens de uma arquitetura VLIW são:

- Arquitetura altamente regular e exposta ao compilador.
- O compilador tem conhecimento prévio de todos os efeitos das operações sobre a arquitetura. Capacidade de despacho de múltiplas operações;
- Mantém o hardware de controle simples, permitindo teoricamente um ciclo de clock menor.

O fato de escalonar as instruções em nível de compilação torna esta arquitetura simples quando comparada com outras (superescalar e superpipeline), pois o paralelismo das instruções não é definido em tempo de execução. A simplificação da arquitetura, teoricamente, diminui o tempo do ciclo de clock, otimizando o processamento como um todo. Duas das principais desvantagens do modelo VLIW são:

- A previsão incorreta do caminho tomado em desvios condicionais pode afetar consideravelmente sua performance. Como a previsão é feita estaticamente, informações importantes disponíveis em tempo de execução são completamente negligenciadas.

Importante salientar que no caso deste trabalho, onde se destaca a execução de algoritmos criptográficos simétricos, esta desvantagem não afeta o desempenho, pois os algoritmos são códigos estáticos e não são alterados em tempo de execução.

- Ineficiência em programas onde a dependência de dados é grande. Neste caso, grande parte das instruções não poderá executar em paralelo, prejudicando o desempenho do sistema.
  Esta desvantagem influência no desempenho do nosso sistema, a dependência de dados varia de algoritmo para algoritmo.

A escolha do modelo de arquitetura VLIW justifica-se pela necessidade de atingir um desempenho diferenciado com um hardware composto por módulos independentes que, em conjunto, podem ser facilmente controlados e coordenados para a execução paralela de instruções. Diferente de outras arquiteturas que têm estas características, a arquitetura VLIW não necessita de um hardware sofisticado para o controle do fluxo de informações, gerando um hardware relativamente simplificado quando comparado com outras arquiteturas como superescalar e superpipeline.

O fato de os algoritmos criptográficos simétricos serem compostos por uma seqüência de operações estáticas, faz com que diminua de forma significativa a ocorrência de eventos aleatórios em tempo de execução que possam prejudicar o desempenho do criptoprocessador, que não está preparado para tratar este tipo de exceção. Isto não quer dizer que ocorrerá um erro, mas apenas prejudicará o desempenho da execução.

3. ARQUITETURA DO CRIPTOPROCESSADOR

Nesta seção apresenta-se uma visão geral do criptoprocessador VLIW. A seguir, destacam-se algumas das características do criptoprocessador, que serão discutidas neste artigo.

- Arquitetura VLIW composta modelo Harvard+Pipeline como um conjunto RISC de instruções.
- Palavra VLIW de 160 Bits e 25 Instruções
- Cache de dados e instrução (Harvard)
- Unidades Funcionais de 128 Bits (UFs).
- Até 4 instruções executando em um único ciclo
- 16 Permutações por ciclo
- Pipeline de 3 estágios globais
- 24 Registradores

A palavra VLIW de 160 bits armazena até 4 instruções de 40 bits cada, que serão executadas em paralelo. O paralelismo acima de 4 instruções torna o hardware mais complexo, aumentando o tempo do ciclo de máquina. Além disso, a ocorrência de mais de 4 instruções sendo executadas em paralelo não seria comum em algoritmos de criptografia simétricos, já que a dependência de dados inibiria esta possibilidade, prejudicando o desempenho geral do criptoprocessador, sub-utilizando suas unidades funcionais.

A ocorrência maior de algumas instruções tais como deslocamento, rotação, permutação e substituição de bits torna importante a criação de módulos dedicados e diferenciados para atingir uma melhor performance[22].

Os módulos de deslocamento e rotação de bits podem atuar em apenas um único bit como até em um bloco de 32 bits. Esta propriedade diminui o número de instruções necessárias para execução destas operações.

As unidades funcionais de permutação e substituição foram projetadas para atingir um desempenho satisfatório e suportar o maior número de algoritmos possíveis. A

criação de módulos independentes de LOAD/STORE e de movimentação entre registradores e desvios (MOV/DESVIOS) são importantes principalmente em processadores que adotam o modelo Harvard, que possui memórias distintas para instruções e dados.

A figura 1 ilustra a arquitetura top-level do criptoprocessador VLIW. O criptoprocessador é formado por quatro partes, (i) o despachador de instruções, (ii) as unidades funcionais, (iii) o banco de registradores e (iv) a unidade de controle.

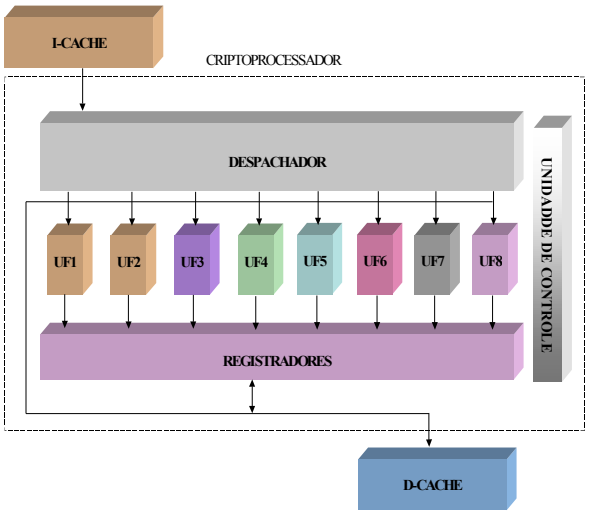


Figura 1 – Arquitetura *top-level* do Criptoprocessador VLIW

Observa-se na figura 1 a presença de duas memórias Caches, uma de instruções (I-CACHE) e outra de dados (D-CACHE). Estas memórias têm endereçamento individual, o que permite a leitura e a escrita simultâneas em memórias distintas, diminuindo o número de ciclos nos estágios de busca e escrita de resultados. Na cache de instruções devem estar armazenados os algoritmos de criptografia descritos em assembly do criptoprocessador. Na cache de dados são armazenadas informações como o conteúdo das S-BOXES de um determinado algoritmo, o texto claro e os resultados de operações realizadas pelo criptoprocessador.

O tamanho da I-CACHE é de 16x160, armazenando um pouco mais de 1 Mbyte de informações, possibilitando o armazenamento de vários algoritmos criptográficos. A D-CACHE é de 16x128 armazenando um total de 1 Mbyte. Na figura 1 as memórias são consideradas como parte externa do criptoprocessador, isto porque, neste protótipo, as memórias não serão implementadas em FPGA. Visto que seria pouco viável a descrição de memórias destas dimensões nestes dispositivos.

O módulo despachador de instruções é responsável pelo direcionamento de cada instrução contida em uma palavra VLIW para a execução. Em princípio, esta função exigiria um hardware sofisticado para tomar decisões, mas na arquitetura VLIW o despachador não engloba funções de decisão em tempo de execução. Isto é, a definição de qual instrução será executada e qual será a unidade funcional selecionada é de responsabilidade do compilador, o que simplifica o hardware e aumenta a velocidade de execução.

Na figura 1, observa-se que há oito unidades funcionais, as quais podem executar um conjunto de até quatro operações selecionadas pelo opcode da instrução. As unidades funcionais são:

- UF1 e UF2 - Unidade Lógica e Aritmética.
- UF3 - Deslocador de 1,2,4, 8 e 32 Bits
- UF4 - Rotacionador de 1,2,4,8 e 32 Bits
- UF5 - Permutação (P-BOX)
- UF6 - Substituição (S-BOX)
- UF7 - Load/Store,
- UF8 - MOV/DESVIOS

As unidades funcionais (UFs) são responsáveis pela execução das instruções direcionadas pelo módulo despachador. Cada UF possui um registrador de instruções (RI), que armazena a instrução que será executada. Assim, tem-se um total de oito registradores de instruções.

Cada UF pode decodificar a instrução que irá executar. A execução entre as unidades funcionais é paralela. Em um determinado ciclo de execução, até quatro unidades funcionais poderão estar em execução, já que o número máximo de instruções em uma palavra VLIW é de 4 instruções. A tabela 1 apresenta as unidades funcionais e os principais registradores que estas utilizam.

Tabela 1 – Principais Registradores utilizados pelas unidades funcionais

UNIDADE FUNCIONAL	PRINCIPAIS REGISTRADORES
ULA 1	A1 e B1
ULA 2	A2 e B2
Deslocador	A3
Rotacionador	A4
Permutação (P-BOX)	A5, B5, X
Substituição (S-BOX)	A6, B6, SPC
Load/Store	DPC, IPC
MOV/DESVIOS	JPC

Os registradores A, B e X são de 128 bits, e armazenam operandos e resultados das unidades funcionais. Registradores e operações de 128 bits otimizam a descrição dos algoritmos simétricos atuais, que operam com chaves e blocos de texto de 128 bits ou mais. Desta forma, é possível realizar operações de 128 bits em um único ciclo e armazena-las nos registradores apropriados otimizando a execução do algoritmo implementado.

Os registradores contadores (SPC, DPC, IPC e JPC) são de 16 bits, responsáveis por endereçar as memórias D-

CACHE (16x128) e I-CACHE (16x160). Para todas as operações, o resultado será armazenado no registrador A. A tabela 2 mostra todos os registradores organizados por função. Pode-se ver que o banco de registradores totaliza 24 registradores, cada unidade funcional utiliza um ou mais registradores distintos.

Tabela 2 – Registradores organizados por função

FUNÇÃO	REGISTRADORES
Registradores de operandos e resultados	X, A1, B1, A2, B2, A3, A4, A5, B5, A6, B6
Registradores contadores	PERAC, AC1, AC2, SPC, DPC, IPC, JPC
Registradores de configuração	SBOXEND, SBOXCOL, SBOXQ, TBO, TBD, B

Toda a arquitetura é gerenciada pela unidade de controle, que organiza o fluxo de execução, inclusive o pipeline de três estágios. Detalhes de cada registrador e da arquitetura em geral são apresentados nas seções seguintes.

4. CONJUNTO E FORMATO DAS INSTRUÇÕES

O conjunto de instruções do nosso criptoprocessador segue o modelo RISC, onde, com apenas 25 instruções, é possível descrever a maioria dos algoritmos de criptografia simétrica. As instruções podem ser divididas em quatro classes:

- Lógicas e Aritméticas: AND,OR, XOR, ADD, SUB, SHL, SHR, ROL, ROR, INC, DEC, NOT, CLR, NOP.
- Movimentação: LOAD, STORE, MOV.
- Desvios: JMP, JZ, JL, JG.
- Especiais: PERINIC, PERBIT, SBOXINIC, SBOX.

As instruções especiais fazem acesso aos módulos especiais do criptoprocessador. Cada grupo de instruções é executado por uma unidade funcional especifica. Importante salientar que, das 8 unidades funcionais, duas são para execução de instruções especiais, tais como a permutação e a substituição. Também existem unidades funcionais próprias para instruções que merecem destaque do projeto do criptoprocessador como o deslocamento e a rotação de bits, comuns em algoritmos de criptografia simétricos. A relação unidade funcional e instruções está descrita na tabela 3

Tabela 3 – Instruções por unidades funcionais

UNIDADE FUNCIONAL	INSTRUÇÕES
ULA 1 e 2	AND,OR, XOR, ADD, SUB, INC, DEC, NOT, CLR E NOP
Deslocador	SHL E SHR
Rotacionador	ROR E ROL
Permutação (P-BOX)	PERINIC E PERBIT
Substituição (S-BOX)	SBOXINIC, SBOX
Load/Store	LOAD, STORE
MOV/DESVIO	MOV, JMP, JZ , JG, JL

De forma geral, cada palavra VLIW é armazenada em uma única posição de memória, isto é, para acessar uma palavra é necessário apenas um endereço de memória. Neste criptoprocessador cada palavra tem 160 bits e contém quatro instruções, de 40 bits cada, que podem ser executadas simultaneamente, ver figura 2. Algumas instruções ocupam toda palavra VLIW, como é o caso da SBOXINIC e a PERBIT. Desta forma, poderá ser executada apenas uma destas instruções por ciclo.

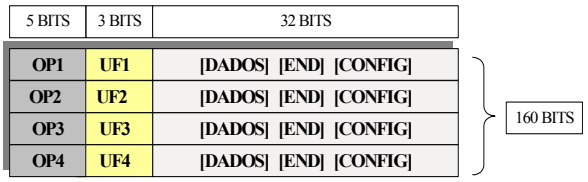


Figura 2 – Formato da palavra VLIW

Algumas instruções são exclusivas, isto é, não podem ocorrer na mesma palavra VLIW. A seguir, mais detalhes sobre as instruções exclusivas.

4.1. Instruções Exclusivas

As instruções exclusivas acessam a Cache de dados e registradores de dados. Para manter a coerência e integridade do acesso a esses dispositivos, não é possível a execução de mais de uma instrução exclusiva na mesma palavra VLIW. Estas instruções são:

- LOAD – Carrega dado de um determinado endereço de memória (D-CACHE)
- STORE - Armazena dados na memória (D-CACHE)
- SBOX - Substitui dados (D-CACHE)

Estas instruções são exclusivas, pois neste caso, fisicamente não é possível o acesso simultâneo à mesma memória (D-CACHE).

Caso especial: a instrução MOV movimenta dados entre registradores em um único ciclo. Dependendo do contexto, esta instrução pode gerar uma incoerência, prejudicando a integridade dos dados e do fluxo de execução. Assim, a instrução MOV pode ser considerada exclusiva em alguns casos. A seguir, um exemplo:

Exemplo: Palavra VLIW [ADD A1, B1 | INC A2 | MOV A1,A3 | NOP]

Esta palavra contém quatro instruções (ADD, INC, MOV e NOP) que são executadas simultaneamente em um único ciclo. Neste caso, existe uma incoerência, já que o valor de A1, após a execução, pode não ser o desejado. Isso acontece porque, tanto a instrução de ADD como a instrução de MOV, modificam o conteúdo de A1. Sendo

que a predominância neste caso é da instrução de movimentação (MOV). Portanto A1, no final da execução, armazenará o valor contido em A3 (MOV A1, A3).

Neste contexto, a instrução MOV pode ser considerada exclusiva. Sendo indicado sua execução em um ciclo anterior ou posterior à instrução de ADD, conforme o resultado que se deseja obter.

4.2. Instruções especiais

As instruções especiais atendem as características próprias de algoritmos de criptografia simétricos como a permutação e substituição de bits. Estas são especiais por estarem presentes na maioria dos algoritmos simétricos e possuir unidades funcionais dedicadas para executá-las no criptoprocessador VLIW. Esta classe de instruções não está presente em processadores de propósito geral.

Cada instrução especial pode ser considerada como uma macro-instrução, ou seja, elas são simples de serem utilizadas, mas executam uma tarefa complexa, composta por várias operações, tudo em um único ciclo de máquina.

Para suportar o maior número possível de algoritmos foi necessário definir parâmetros de inicialização que configuram o modo de execução das instruções especiais. A seguir, as instruções especiais:

Instruções de permutação	**Instruções de substituição**
PERINIC - Inicialização da instrução de permutação	SBOXINIC - Inicialização da instrução de substituição
PERBIT - Permutação de bits. (16 permutações por ciclo)	SBOX – Substituição de bits.

* as instruções de SBOXINIC e PERBIT são de 160 bits.

As instruções de permutação realizam a troca de bits predefinidos em uma tabela de permutação estática. Esta operação é realizada por uma unidade funcional específica (UF5). O funcionamento detalhado das instruções de substituição é apresentado na seção seguinte.

4.3. SBOX - Operação de Substituição

A S-BOX ou caixa de substituição é um conjunto de dados que forma um vetor ou uma matriz bidimensional. Toda S-BOX é composta por uma entrada de bits (Input S-BOX), que determina a localização do dado de substituição, e uma saída de bits (Output S-BOX), que retorna o valor do dado de substituição. O dado de substituição irá sobrescrever os bits utilizados na entrada da S-BOX.

A instrução SBOX realiza a substituição de bits predeterminados. Através de uma busca por linha e coluna, identifica-se o endereço onde está armazenado o dado de substituição. Este dado é recuperado e armazenado na posição adequada dentro do registrador resultante, realizando assim a substituição.

Este processo é executado com base em alguns parâmetros configuráveis, os quais são necessários para calcular o endereço de memória onde está localizado o dado de substituição.

Alguns parâmetros podem variar conforme o algoritmo criptográfico que está sendo descrito, por isso a necessidade de parâmetros configuráveis. Na tabela 4 tem-se o resultado de um estudo realizado neste projeto que demonstra como as S-BOXs podem variar de algoritmo para algoritmo.

Tabela 4 – Comparação das SBOXES de diferentes algoritmos de criptografia

Algoritmo	**NSBOX**	**Entrada**	**Saída**	**LIN**	**COL**
DES	8	6	4	1 e 6	2,3,4,5
AES	1	8	8	1,2,3,4	5,6,7,8
Serpent	8	4	4	--	1,2,3,4
Cast-128	4	8	32	--	1 a 8
MARS	2	8	32	--	1 a 8
Twofish	8	4	4	--	1 a 4
Magenta	1	8	8	--	1 a 8
Frog	1	8	8	--	1 a 8
BlowFish	4	8	32	--	1 a 8
LOK97	2	14	8	--	1 a 14
RC5	--	--	--	--	--
RC6	--	--	--	--	--
IDEA	--	--	--	--	--

*NSBOX – Número de Sboxes
*Entrada – Número de bits de entrada da S-BOX.
*Saída – Número de bits de sáida da S-BOX.
*LIN – Bits que determinam a linha *COL – Bits que determinam a coluna

Após um estudo preliminar realizado, os seguintes parâmetros foram considerados necessários para implementar qualquer S-BOX de algoritmos criptográficos simétricos:

- Número de SBOXES
- Número de bits de entrada da S-BOX
- Número de bits de saída da S-BOX
- Os bits que determinam a linha e bits que determinam a coluna da SBOX.

Observando a tabela 4, nota-se que há uma diversificação quanto ao tamanho das SBOXES, dos bits de entrada e saída e da quantidade de S-BOXES tornam o projeto do hardware relativamente complexo. Um hardware dedicado para cada S-BOX descrita na tabela 4 pode tornar o circuito veloz, mas necessita de um elevado número de componentes para implementá-lo.

Desenvolver uma única S-BOX genérica que seja veloz e que atenda a maioria dos algoritmos simétricos pode ser considerado um desafio. Neste projeto é proposta uma solução através de parâmetros de configuração por software, isto é, através de uma instrução, o programador poderá configurar o tamanho da S-BOX, bits de entrada e

saída, bits que determinam linha e coluna e outros parâmetros necessários para configurar a S-BOX desejada.

Uma solução futura poderia ser através de reconfiguração, onde o hardware decide a necessidade de uma nova S-BOX e esta é reconfigurada em tempo de execução (RTR – Run Time Reconfiguration) [5].

Os parâmetros de configuração são implementados através da instrução SBOXINIC. A seguir os parâmetros necessários e suas descrições:

- SBOXEND - Endereço de memória onde está localizada a S-BOX. Os dados da S-BOX estão armazenados na memória D-CACHE. Deve-se apenas indicar a partir de qual endereço inicia a S-BOX, independente do número de S-BOX que o algoritmo utiliza.
- SBOXCOL - Número de colunas de uma SBOX.
- SBOXQ – Quantidade de elementos de uma S-BOX.
- TBO - Tamanho do bloco origem, representa a quantidade em bits do bloco que será substituído. O bloco origem alimenta a entrada SBOX gerando o endereço do dado de substituição.
- TBD - Tamanho do bloco destino, representa a quantidade em bits do bloco de substituição. Ao final, o bloco origem será substituído pelo bloco destino, mesmo que forem de tamanhos diferentes. O bloco destino armazenará o resultado da substituição.
- LIN – Bits ou Bytes do bloco origem que determinam a linha de endereçamento da SBOX. Cada algoritmo tem uma forma específica para determinar quais os bits ou bytes que determinaram a linha e a coluna de uma S-BOX, onde está localizado o dado de substituição. Este parâmetro determina estes bits ou bytes para a linha.
- COL - Bits ou Bytes do bloco destino que determinam a coluna de endereçamento da SBOX, de forma similar ao LIN.
- B – Tipo Bit ou Byte, quando B=0 (zero), os valores de LIN e COL são considerados bits. Quando B=1, os valores de LIN e COL são considerados Bytes.

Todos os parâmetros são armazenados em registradores dedicados, assim há um limite físico de bits para a respectiva representação de cada parâmetro, os quais determinam os algoritmos suportados pelo criptoprocessador. Na tabela 5, tem-se a representação máxima de cada parâmetro.

Alguns parâmetros podem representar um limite maior do que os encontrados em algoritmos de criptografia simétricos atuais. Mas com o objetivo de adaptação a novos algoritmos, estes limites foram propostos para suportar futuros algoritmos com S-BOXES diferenciadas.

Tabela 5 – Limites de representação dos parâmetros da operação de substituição

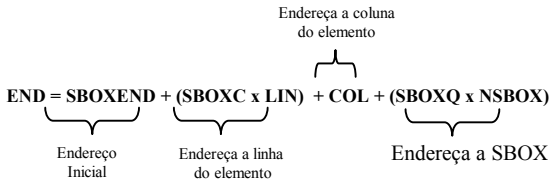
Parâmetro	Bits	Descrição
SBOXEND	16	Endereça até 65535 posições de memória (D-CACHE)
SBOXCOL	16	Configura S-BOXES até 65535 colunas
SBOXQ	16	Configura S-BOXES até 65535 elementos
TBO	6	Configura blocos origem com até 64 elementos
TBD	6	Configura blocos destino com até 64 elementos
LIN	32	Configura 8 bits para linha em modo bit (B=0); Configura até 2 bytes em modo byte (B=1);
COL	32	Configura 8 bits para coluna em modo bit (B=0); Configura até 2 bytes em modo byte (B=1);

A definição do número de bits de cada parâmetro foi baseada na tabela 4, onde se pode identificar e quantificar os bits necessários para representar o número máximo de colunas de uma S-BOX, o tamanho máximo do bloco origem e destino e outros parâmetros. A sintaxe da instrução de inicialização e configuração da S-BOX é:

SBOXINIC SBOXEND, SBOXCOL, SBOXQ, TBO, TBD, LIN, COL, B

Após a inicialização e configuração através da instrução SBOXINIC, pode ser realizada a operação de substituição propriamente dita. A instrução SBOX realiza os cálculos necessários para localizar o dado de substituição armazenado na memória D-CACHE, baseando-se nos parâmetros configuráveis.

A instrução SBOX utiliza o parâmetro NSBOX que identifica qual S-BOX deve ser utilizada na substituição. Quando o algoritmo de criptografia utiliza apenas uma S-BOX, o valor de NSBOX é zero. Para identificar um elemento dentro da memória, foi desenvolvida uma fórmula matemática que através de alguns dos parâmetros configuráveis, pode identificar o endereço desejado. Ver fórmula 1.



Fórmula 1 – Endereçamento do dado de substituição.

A memória é um vetor de N posições. Em alguns algoritmos, as S-BOXES são representadas por matrizes, como no DES e no AES. Em outros algoritmos, por vetores (como no Serpent, Magenta e outros). Para identificar a posição de memória correspondente, independente do tipo de S-BOX, deve-se utilizar a fórmula 1.

O valor contido em END após execução representa o endereço de memória onde está armazenado o dado de substituição. A seguir, a seqüência de passos necessários para executar a instrução S-BOX. Esses passos são:

1º Passo – determinar os bits de entrada da S-BOX.

2º Passo – determinar o valor da linha (LIN) e da coluna (COL) do dado de substituição desejado.
3º passo – Utilizando a fórmula 1, gerar o endereço físico do dado de substituição, (D-CACHE).
4º passo – Determinar o bloco de destino e realizar a substituição de bits.

A operação de substituição SBOX utiliza os seguintes registradores:

- A6 – armazena resultado final da substituição
- B6 – armazena bits que serão substituídos
- Registradores dos parâmetros configuráveis: SBOXEND, SBOXCOL, SBOXQ, TBO,TBD,LIN,COL.
- AC1 – Acumulador 1, aponta para o bloco origem que será substituído
- AC2 - Acumulador 2, aponta para o bloco destino de substituição

A cada substituição, o AC1 é incrementado com o valor do registrador TBO (tamanho do bloco origem), apontando assim para o próximo bloco origem que será substituído. O mesmo acontece com AC2, mas este será incrementado com o valor de TBD (tamanho do bloco destino), apontando para o próximo bloco destino de substituição. A instrução de inicialização e configuração da SBOX zera o valor de AC1 e AC2.

4.4. Pipeline do Criptoprocessador VLIW

O criptoprocessador VLIW possui um pipeline de 3 estágios globais. Cada estágio é executado em um único ciclo de máquina, inclusive o estágio de execução de instruções. Os estágios são gerenciados por uma máquina finita de estados (unidade de controle) sincronizada pelo clock global. Os estágios são:

- **Estágio 1:** Busca e despacho de instruções. Busca uma palavra VLIW de 160 bits. Esta palavra é recebida pelo despachador de instruções, que realiza o direcionamento para as unidades funcionais predefinidas. Este estágio é realizado em um único ciclo de máquina.
- **Estágio 2**: Decodificação e execução. Neste estágio, as unidades funcionais decodificam as instruções e as executam.
- **Estágio 3**: Escrita em memória ou registradores. Os resultados são armazenados nos devidos registradores ou na memória D-CACHE. A arquitetura Harvard permite a leitura e escrita simultânea em memórias distintas.

Desta maneira, o pipeline está preenchido em três ciclos de máquina. Assim, a partir do terceiro ciclo podem ser executadas até 4 instruções por ciclo.O pipeline nunca é interrompido para o tratamento de instruções especiais ou de qualquer outra instrução. O fato de todas as instruções serem executadas em um único ciclo facilita a descrição do hardware que controla a transição entre estágios globais.

5. DESEMPENHO DO CRIPTOPROCESSADOR

Nesta seção apresenta-se de forma sucinta as estatísticas de desempenho do Criptoprocessador VLIW implementado em um FPGA Virtex II Pro. Estas estatísticas podem ser visualizadas na tabela 6.

Tabela 6 - Estatísticas da implementação em FPGA

Dispositivo	Slices	LUTs	FF	TP(ns)	Freq.(MHz)
Virtex II Pro	1315	2484	669	8,618	116,036

Como esperado o processador obteve um bom desempenho temporal, o tempo de propagação do circuito foi de 8,667ns atingindo uma freqüência máxima de 116 Mhz. O criptoprocessador mapeado no dispositivo FPGA, utilizou 1315 Slices e 669 Flip-Flops.

5.1. Comparação com processadores de propósito geral

Na tabela 7 tem-se a comparação com processadores de propósito geral (P3 e P4, Pentium III e IV), quando se executa o algoritmo DES.

Tabela 7 - Estatísticas da implementação em FPGA

Proc.	Freqüência	Memória	Texto claro	Tempo
P4	1,6 Ghz	128 MBytes	1 MByte	3,250s
P3	1.0 Ghz	256 Mbytes	1 MByte	4,256s
P3	800 Mhz	128 MBytes	1 Mbyte	5,307s
P3	500 Mhz	512 Mbytes	1 MByte	5,875s
VLIW	116 Mhz	1 Mbyte	1 MByte	0,36 s

Estas estatísticas de desempenho mostram que o criptoprocessador VLIW pode ser até 16 vezes mais rápido que um Pentium III 500Mhz.

5.2. Algoritmo DES no Criptoprocessador

Diferente da programação seqüencial, em processadores VLIW as instruções são representadas através de micro-códigos horizontais. A seguir, apresenta-se um trecho do micro-código horizontal do algoritmo DES, segundo o assembly do criptoprocessador

```
// permutação de expansão
PERBIT  31,00,01,02,03,04,03,04,05,06,07,08,07,08,09,10
PERBIT  11,12,11,12,13,14,15,16,15,16,17,18,19,20,19,20
PERBIT  21,22,23,24,23,24,25,26,27,28,27,28,29,30,31,00
// Xor com a chave Ki
MOV  B1,A3 LOAD A1,[K1] XOR  A1,B1 NOP
// inicio da operação de substituição
SBOXINIC  SBOXEND,SBOXCOL,SBOXQ,TBO,TBD,LIN,COL,B
SBOX 0 NOP NOP NOP
```

Um dado muito importante e interessante é o aproveitamento da capacidade de processamento do Criptoprocessador VLIW, que índica qual a porcentagem de utilização de todo recurso de hardware disponível por um determinado programa.

Para calcular este valor, é necessário quantificar qual a porcentagem de aproveitamento de cada palavra VLIW do algoritmo descrito. Isto é, se em uma palavra VLIW pode-se armazenar até 4 instruções que serão executadas em paralelo, o cálculo consiste em quantificar quantas instrução por palavra VLIW estão sendo efetivamente executadas.

Neste caso, o algoritmo DES teve um aproveitamento de 61,18% da capacidade total do criptoprocessador, que pode ser considerado aceitável, mas não ideal. O Ideal hipotético seria, de 100%. Mas sabe-se que não é comum esta taxa de aproveitamento. Cada algoritmo de criptografia simétrico tem um aproveitamento diferente, assim como diferentes descrições para o mesmo algoritmo também podem ter.

6. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou uma nova proposta de arquitetura, específica para algoritmos criptográficos simétricos. A arquitetura baseia-se no modelo VLIW e a sua implementação em FPGA obteve um bom desempenho temporal. Uma das características de maior relevância neste projeto é o número de algoritmos suportados pelo criptoprocessador. Todos os algoritmos criptográficos da tabela 4 podem ser implementados nesta arquitetura. As próximas atividades do projeto seriam:

- Otimizar a arquitetura, adicionado ou removendo elementos, como registradores e unidades funcionais.
- Estudar a possibilidade de reconfiguração da arquitetura.
- Implementar um simulador e montador assembly para o criptoprocessador.
- Implementar um compilador de alto nível para o criptoprocessador.

7. REFERÊNCIAS

[1] Lisa Wu, Chris Weaver, Todd Austin. CryptoManiac: A Fast Flexible Architecture for Secure Communication. Advanced Computer Architecture Laboratory, Univ. of Michigan, U.S.A, In Proc. of ISCA-2001, Goteborg, Sweden.

[2] Jerome Burke, John McDonald, Todd Austin. Architectural Support for Fast Symmetric-Key Cryptography..Advanced Computer Architecture Laboratory, Univ. of Michigan, U.S.A, In Proc. of ASPLOS IX, Cambridge, 2000.

[3] Christof Paar, Brebdon Chetwynd, Thomas Connor, Sheng Yung Deng, Steve Marchant, An Algorithm-Agile Cryptographic Coprocessor Based on FPGAs. Worcester Polytechnic Institute, U.S.A, The SPIE´s Symposium on Voice and Data Communications, Sep. 1999, Boston.

[4] Haiyong Xie, Li Zhou, Laxmi Bhuyan. An Architectural Analysis of Cryptographic Applications for Network Processors. Department of Computer Science and Engineering, Univ. of California, Riverside. IEEE First Workshop on Network Processors, with HPCA-8, Boston, U.S.A., Feb., 2002.

[5] Prof. C. Paar, Reconfigurable Hardware in Modern Cryptography. Technical Report, Cryptography and Information Security Group, Electrical and Computer Engineering Department. Worcester Polytechnic Institute,2000.

[6] Dong Bok Yeom, Jong Su Lee, Jong Sou Park, Sang Tae Kim The Simulation and Implementation of Next Generation Encryption Algorithm RIJNDAEL. In Proc. of MIC-2002.

[7] Federal Information Processing Standards Publication46-2, DATA ENCRYPTION STANDARD, Dezembro, 1993

[8] Federal Information Processing Standards Publication 197, ADVANCED ENCRYPTION STANDARD (AES), Novembro, 2001.

[9] Rivest Ron, The RC5 encryption algorithm, Fast Software Encryption, 2nd. International Workshop, Lec. Note in Comp. Sci. 1008, pp 86-96, Springer-Verlag, 1995).

[10] Ronald L. Rivest, RC6TM Block Cipher, RSA Security Inc, USA, 2000

[11] M.J. Jacobson, Jr._and K. Hubery, The MAGENTA Block Cipher Algorithm,GERMANY, 1998.

[12] Carolynn Burwick, Don Coppersmith, et al, MARS - a candidate cipher for AES, IBM Corporation, USA, 1999

[13] Ross Anderson, Eli Biham, Lars Knudsen, Serpent: A Proposal for the Advanced Encryption Standard, England, 1998.

[14] Bruce Schneier, John Kelsey, et al Twofish: A 128-Bit Block Cipher,USA, 1998.

[15] Dianelos Georgoudis, Damian Leroux, et al, Specification of the Algorithm THE "FROG" ENCRYPTION ALGORITHM,USA, 1998.

[16] B. Schneier, Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish), USA, 1994.

[17] Ordonez E. D. M, Pereira F. D. et al. Algoritmos de criptografia em hardware software. III Escola Regional de Informática, RJ/ES,2003.

[18] Ordonez E. D. M, Pereira F. D. et al. Projeto, Desempenho e Aplicações de Sistemas Digitais em FPGAs, Bless Gráfica e Editora, Marília/SP, 2003.

[19] Ordonez E. D. M, Pereira F. D. Otimização em VHDL e Desempenho em FPGAs do Algoritmo de Criptografia DES. IV Workshop em Sistemas Computacionais de Alto Desempenho (WSCAD), São Paulo, Novembro de 2003.

[20] Schneier, B, "The IDEA Encryption Algorithm", Dr. Dobb's Journal, December 1993.

[21] Carlisle Adams, "The CAST-128 Encryption Algorithm,", RFC 2144, May 1997.

[22] Fábio Dacêncio Pereira, Criptoprocessador VLIW para algoritmos Simétricos, Qualificação de Mestrado, Ciência da Computação, UNIVEM, Marília - Brasil, 2004.