Fundação de Ensino Eurípides Soares da Rocha
Centro Universitário Eurípides de Marília – UNIVEM
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ToPSeC - UM PROCESSADOR COM SUPORTE A MUDANÇA DE TOPOLOGIA E PRIMITIVAS DE SEGURANÇA

BRUNO MOREIRA CESTARI

Marília
Novembro/2006

BRUNO MOREIRA CESTARI

ToPSeC - UM PROCESSADOR COM SUPORTE A MUDANÇA DE TOPOLOGIA E PRIMITIVAS DE SEGURANÇA

Dissertação apresentada ao Programa de Mestrado do Centro Universitário de Marília, mantido pela Fundação Eurípides Soares da Rocha, para obtenção do Título de Mestre em Ciência da Computação (Área de Concentração: Arquitetura de Sistemas Computacionais).

Orientador:
Prof. Dr. Edward David Moreno Ordonez

MARÍLIA
2006

CESTARI, Bruno Moreira
ToPSeC – Um processado com suporte a mudança de topologia e primitivas de segurança / Bruno Moreira Cestari; orientador: Edward David Moreno Ordonez.
Marília, SP: [s.n.], 2006.
126 f.

Dissertação (Mestrado em Ciência da Computação) – Centro Universitário Eurípides de Marília – Fundação Eurípides Soares da Rocha.

1. Arquitetura 2. VHDL 3. FPGA 4. Processadores de Aplicação Específica 5. Criptografia e Segurança.

CDD: 004.2

BRUNO MOREIRA CESTARI

ToPSeC - UM PROCESSADOR COM SUPORTE A MUDANÇA DE TOPOLOGIA E PRIMITIVAS DE SEGURANÇA

Banca examinadora da dissertação apresentada ao Programa de Mestrado da UNIVEM / F.E.E.S.R., para obtenção do Título de Mestre em Ciência da Computação. Área de Concentração: Arquitetura de Sistemas Computacionais.

Resultado: _____

ORIENTADOR: Prof. Dr. Edward David Moreno Ordonez

1º EXAMINADOR: _____

2º EXAMINADOR: _____

Marília, 14 de novembro de 2006.

AGRADECIMENTOS

Agradeço a Deus, por me dar forças e sabedoria para prosseguir na minha jornada.

Agradeço aos meus pais, por me proporcionarem uma boa educação para que um dia eu possa caminhar sozinho.

Agradeço a minha namorada Maira, por sempre me apoiar e me dar forças para seguir adiante, nunca me deixando desanimar com as adversidades que aparecem.

Agradeço a meus irmãos por me apoiarem.

Agradeço a todos os professores, em especial, o professor Dr. Edward por acreditar em mim.

Agradeço aos meus colegas de estudo, especialmente aos companheiros de hotel, por me proporcionarem momentos, alegria, descontração, consolo e companheirismo.

E também a todos que de maneira indireta me ajudaram neste objetivo.

CESTARI, Bruno Moreira. **ToPSeC - Um processador com suporte a mudança de topologia e primitivas de segurança.** 2006. 126 fl. Dissertação (Mestrado em Ciência da Computação) – Centro Universitário Eurípides Soares da Rocha, Marília, 2006-11-14

RESUMO

Atualmente, com a crescente utilização das redes de computadores em vários segmentos da sociedade, a transmissão dos dados e segurança da informação torna-se pontos fundamentais, exigindo soluções para que essas informações fluam de maneira eficiente e segura.

Este trabalho, baseando-se em trabalhos já existentes, tem por fim o estudo e a proposta de um processador capaz de realizar o roteamento de pacotes de dados específicos para este processador nas topologias Anel Unidirecional e Árvore Binária e a mudança entre essas duas topologias. Também permite a busca por rotas alternativas dentro dessas topologias, para que a transmissão dos dados ocorra de maneira eficiente.

Adicionalmente, os conceitos de criptografia foram incorporados nesse trabalho, sendo incluídos os algoritmos DES e AES, para que, juntamente com o roteamento nas topologias acima descritas, ofereça segurança na transmissão das informações.

Este processador foi descrito em VHDL e realizaram-se várias simulações de seu funcionamento, apresentando dados de sua implementação em alguns FPGAs. Os resultados obtidos permitem afirmar que possui um bom desempenho, já que ocupa pouco espaço nos FPGAs e a velocidade obtida foi de 40,035 e 34,902 MHz para os algoritmos DES e AES, respectivamente, no FPGA Virtex V600FG676.

Palavras-chave: Processador, Criptografia, VHDL, FPGA, Redes de Computadores, Roteamento, Anel Unidirecional, Árvore Binária.

CESTARI, Bruno Moreira. **ToPSeC - Um processador com suporte a mudança de topologia e primitivas de segurança.** 2006. 126 fl. Dissertação (Mestrado em Ciência da Computação) – Centro Universitário Eurípides Soares da Rocha, Marília, 2006-11-14

ABSTRACT

At present, with the growing utilization of the networks in several segments of the society, the data transmission and the information security becomes itself fundamental points, requiring solutions for this information flow of secure and efficient way.

This work, basing itself in a already existing work, has finally the study and proposal of a processor that is capable of making the routing of packages that are specifics for this processor in Unidirectional Ring and Binary Tree topologies, and the change between those two topologies, including seek alternatives routes inside those topologies, for that the data transmission occur of efficient and quick way.

Additionally, the cryptography concept was incorporated in this work, was included the DES and AES algorithms, for that jointly within the routing in topologies above described, offer the security in transmission of the information.

This network processor was described in VHDL and carried out several simulations of his operation, presenting data of its implementation in some FPGAs. The obtained results allow to affirm that it has a good performance, since occupies little space in the FPGAs and the obtained speed was 40,035 and 34,902 Mhz respectively for the DES and AES algoritms, in the FPGA Virtex V600FG676.

Keywords: Processor, Criptography, VHDL, FPGA, Networks, Routing, Unidirectional Ring, Binary Tree.

LISTA DE ILUSTRAÇÕES

Capítulo 1 – Introdução

Capítulo 2 – Processadores de Rede em FPGAs

Capítulo 3 – Conceitos de Segurança de Dados e Criptografia

Capítulo 4 – ToPSeC – Um processador com suporte a mudança de topologia

LISTA DE TABELAS

Capítulo 2 – Processadores de Rede em FPGAs

Capítulo 3 – Conceitos de Segurança de Dados e Criptografia

Capítulo 4 – ToPSeC – Um processador com suporte a mudança de topologia

LISTA DE ABREVIATURAS

AES	*Advanced Encryption Standard*
API	*Application Programing Interface*
ASI	*Agere System Interface*
ASIC	*Application Specific Integrated Circuit*
ATM	*Asynchronous Transfer Mode*
CBC	*Cipher Block Chaining Mode*
CFB	*Cipher FeedBack*
CISC	*Complex Instruction Set Computer*
DES	*Data Encryption Standard*
ECB	*Eletronic Code Block*
EPC	*Embedded Processor Complex*
FIFO	*First In First Out*
FPGA	*Field Programmable Gate Array*
FPL	*Functional Programming Language*
FPP	*Fast-Pattern Processor*
GPP	*General Purpose Processor*
HDL	*Hardware Description Language*
HTTP	*Hyper Text Transfer Protocol*
IDE	*Integrated Development Environment*
IP	*Internet Protocol*
IPSec	*Internet Protocol Security*
IPv6	*Internet Protocol version 6*
ISO	*International Organizations for Standardization*
MARS	*IBM's block cipher algoritm*
NIST	*National Institute of Standards and Technology*
NP	*Network Processor*
NPSim	*Network Processor Simulator*
NPSoC	*Network Processor System on Chip*
OFB	*Output FeedBack*
OSI	*Open System Interconnection*
P2M	*Peer to Mail*
P2P	*Peer to Peer*
PC	Controlador de Programa
PCI	*Peripheral Component Interconnect*
PF	*Permutação Final*
PI	*Permutação Inicial*
PMM	*Physical MAC Multiplexer*
QoS	*Quality of Service*

R2NP	*Reconfigurable RISC Network Processor*
RC6	Bloco criptográfico de chave simétrica derivada do algoritomo RC5
RCNP	*Reconfigurable CISC Network Processor*
RI	Registrador de Instruções
RISC	*Reduced Instruction Set Computer*
RSP	*Routing-Switching Processor*
SMTP	*Simple Mail Transfer Protocol*
SoC	*System on Chip*
SSL	*Secure Socket Layer*
TLS	*Transport Layer Security*
UC	Unidade de Controle
ULA	Unidade Lógica Aritmética
VHDL	*VHISC Hardware Description Language*
VLIW	*Very Large Instruction Word*
VoIP	*Voice over IP*
VPN	*Virtual Private Network*

SUMÁRIO

Capítulo 1

Introdução

1.1 A Sociedade e as Redes de Computadores

Com o passar dos anos, a tecnologia passa cada vez mais a fazer parte de nossas vidas, seja na utilização de um eletrodoméstico, na utilização de um caixa eletrônico ou na simples utilização de um e-mail. Essa realidade passa a fazer parte da vida das pessoas por proporcionar conforto, rapidez e agilidade nas ações realizadas no cotidiano.

Uma das principais tecnologias utilizadas pelas pessoas são as redes de computadores. Graças a elas podem-se ter sistemas distantes interligados, e assim dispor de informações instantaneamente. Um grande exemplo da utilização de redes de computadores, que se utiliza de longa data são as redes bancarias, onde um correntista tem acesso a informações tais como saldo e extrato em qualquer lugar que houver uma agência bancária dessa instituição.

Um pouco mais jovem, porém não menos importante, é o surgimento da Internet. Ela é definida como uma rede mundial de computadores, um meio de comunicação onde as pessoas podem interagir umas com as outras, trocar informações e utilizar serviços nela disponíveis. A Internet vem cada vez mais se popularizando e um fator determinante é a redução de custo dos computadores e equipamentos eletrônicos, e também a redução do custo de acesso e a facilidade do seu uso em instituições de ensino, bibliotecas, empresas e mesmo em casa.

Junto com o aumento da utilização da Internet aumenta o volume de dados que vão trafegar nela, como *downloads* e *uploads*, VoIP (*Voice over IP*), VPNs (*Virtual Private Networks*), transmissão de voz e imagem, P2P (*Peer to Peer*), P2M (*Peer to Mail*), etc. (SESHADRI, 2003).

Isso remete a um aumento, ano após ano, na utilização da Internet e, consequentemente, um aumento exponencial no tráfego de informações na rede contra um aumento não tão expressivo do poder de processamento (SHAH, 2001) conforme figura 1.1.

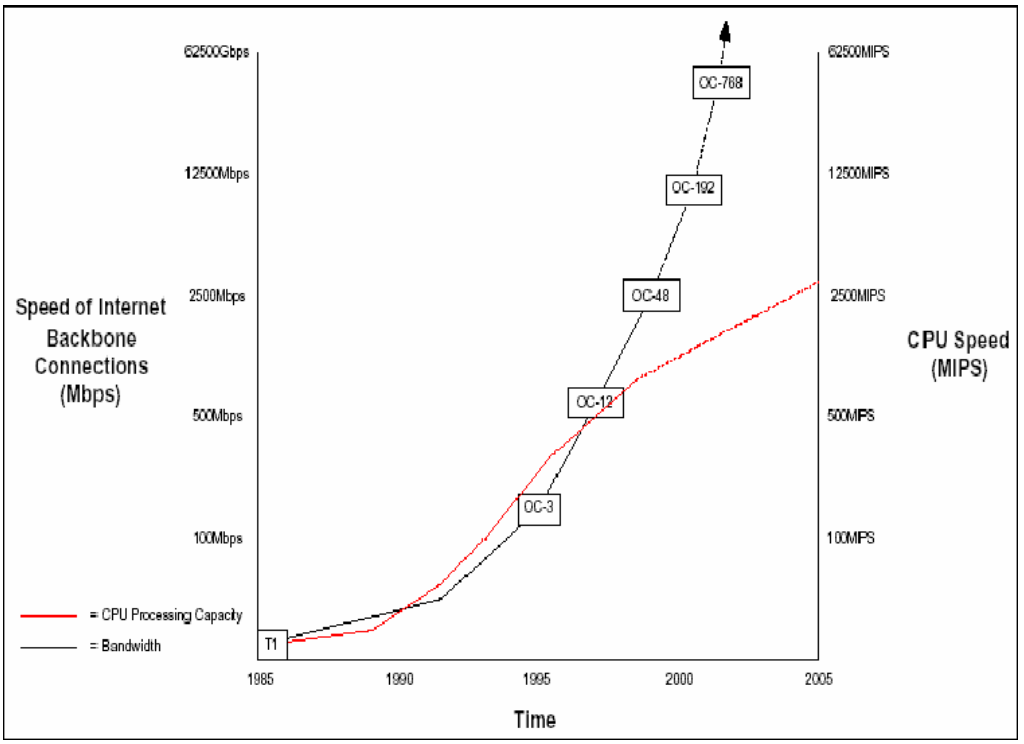


FIGURA 1.1 – Comparação do aumento do poder de processamento x largura de banda

Devido a esses e outros fatores tem crescido a necessidade de comunicações de rede que sejam robustas, estáveis e de alto desempenho.

Outro fator que contribui para a lentidão, não apenas na Internet, mas em qualquer rede é o congestionamento dos *links* de transmissão. O principal responsável por isso são os *gateways*, ou roteadores, que confinam o tráfego entre redes, se tornando verdadeiros gargalos na rede (TANEMBAUM, 1999).

Para que as redes tenham um uso efetivo, elas precisarão suportar novos protocolos que incluam diferentes serviços, segurança e várias outras funções de gerenciamento de redes. Enquanto as redes de computadores demandarem equipamentos que possuam uma grande

taxa de transmissão de dados, elas também precisarão de flexibilidade para suportar novos protocolos e aplicações (SHAH, 2001).

Uma solução viável para esse problema são os processadores de rede, onde o processamento dos pacotes de rede é mais rápido, possibilitando dessa forma o aumento da capacidade de processamento de pacotes na rede e consequentemente uma maior transmissão de dados (TANEMBAUM, 1999).

No entanto, não adiantaria ter um serviço de rede excelente se este não fosse seguro. A questão da segurança da informação em uma rede de computadores é um assunto a ser considerado, pois afeta todos os usuários de um sistema que tenha seu funcionamento em rede. Em uma rede, as informações que trafegam nela são suscetíveis a desvio ou cópia, de modo que a criptografia se torna uma alternativa atraente; caso ocorra desvio ou cópia de informações, essas serão ilegíveis, ou seja, não se conseguirá visualizar o que contiverem.

1.2 Justificativa e Importância dos Processadores de Rede e da Segurança com Criptografia em FPGA

O grande gargalo nas comunicações tem sido os equipamentos de rede, como roteadores, *hubs*, *switches*, etc. Eles centralizam o tráfego de pacotes e muitas vezes são os responsáveis pela falta de eficiência da rede. As tecnologias de transmissão de dados têm evoluído e esses equipamentos de rede têm que seguir essa evolução (MUZZI, 2005).

Para que o desenvolvimento desses equipamentos de rede ocorra de maneira rápida, existe a tecnologia FPGA (*Field Programmable Gate Array*), que possui como principal vantagem da fácil prototipação. Essa vantagem possibilita um *time-to-market* muito curto, permitindo uma rápida inserção do produto no mercado consumidor. O desenvolvimento e manutenção de um dispositivo em FPGA se torna muito simples e flexível, visto que o seu desenvolvimento é feito através de linguagem de programação, geralmente VHDL (*VHSIC Hardware Description Language*), tornando uma alternativa de baixo custo (SESHADRI, 2003).

Por se tratar do ToPSeC, um processador que será implementado em FPGA, sua execução será totalmente em hardware, possibilitando dessa forma, um processamento rápido.

Outro ponto fundamental é que os processadores de rede é um novo nicho mercadológico, onde ainda existem poucos produtos finais destinados aos consumidores.

Juntamente com todas as justificativas apresentadas acima, a questão da segurança em redes de computadores é um ponto importante a ser considerado. O desenvolvimento de um dispositivo que atuam em redes de comunicação geralmente visa apenas à otimização do processamento de pacotes de rede, o que o tornaria a comunicação mais rápida, mas não segura. Por esse motivo, a criptografia é um tema incisivo neste projeto, oferecendo segurança a informação trafegada.

Considerando processadores de rede, ainda há a questão da área de aplicação a que se destina. Normalmente, os processadores de rede não suportam mais de uma topologia, e nos casos em que há esse suporte, a configuração normalmente é única, no que tange a programação de processadores de rede em baixo nível. Poucas soluções existentes no mercado atendem os pontos apresentados acima.

Alguns roteadores solucionam parte desses problemas combinando um GPP (*General Purpose Processor*) com circuitos ASICs (*Application Specific Integrated Circuit*) que implementam algoritmos proprietários para prover funcionalidades adicionais. No entanto, esse tipo de projeto costuma ser grande, caro e demora cerca de 18 meses para ser desenvolvido. Um roteador que possua um ASIC, além de demorar a chegar ao mercado, teria um custo inviável para a maioria das pessoas. Outro ponto negativo de um ASIC é a dificuldade de se fazer modificações ou correções (SESHADRI, 2003).

Além da facilidade de modificações na estrutura de uma FPGA, já que, para que isto ocorra basta que a programação de seus circuitos seja alterada, o que pode ser feito em HDL (*Hardware Description Language*) como em diagrama esquemático, e posteriormente carregada nos seus blocos programáveis, a implantação de qualquer tipo de circuito em uma FPGA é mais rápido (MORENO, 2003).

A grande vantagem da implementação de algum circuito em FPGA é a fácil prototipação, facilitando a realização de manutenção. Para o desenvolvimento de um protótipo, basta que a programação dos circuitos seja feita e carregada no FPGA. Como vantagem, o FPGA oferece a possibilidade de reprogramação total e parcial em tempo de execução, o que garantem a flexibilidade desse tipo de tecnologia, permitindo a execução de diversos algoritmos durante a sua execução.

A figura 1.2 mostra uma comparação de flexibilidade x desempenho entre as tecnologias citadas para o desenvolvimento de *chips*.

No caso de uma implementação em ASIC, o mesmo é implementado em *chip* definitivo. Isso torna essa tecnologia cara e sem tolerância de falhas, pois caso haja um erro, tudo o que foi implementado se perde (MORENO, 2003).

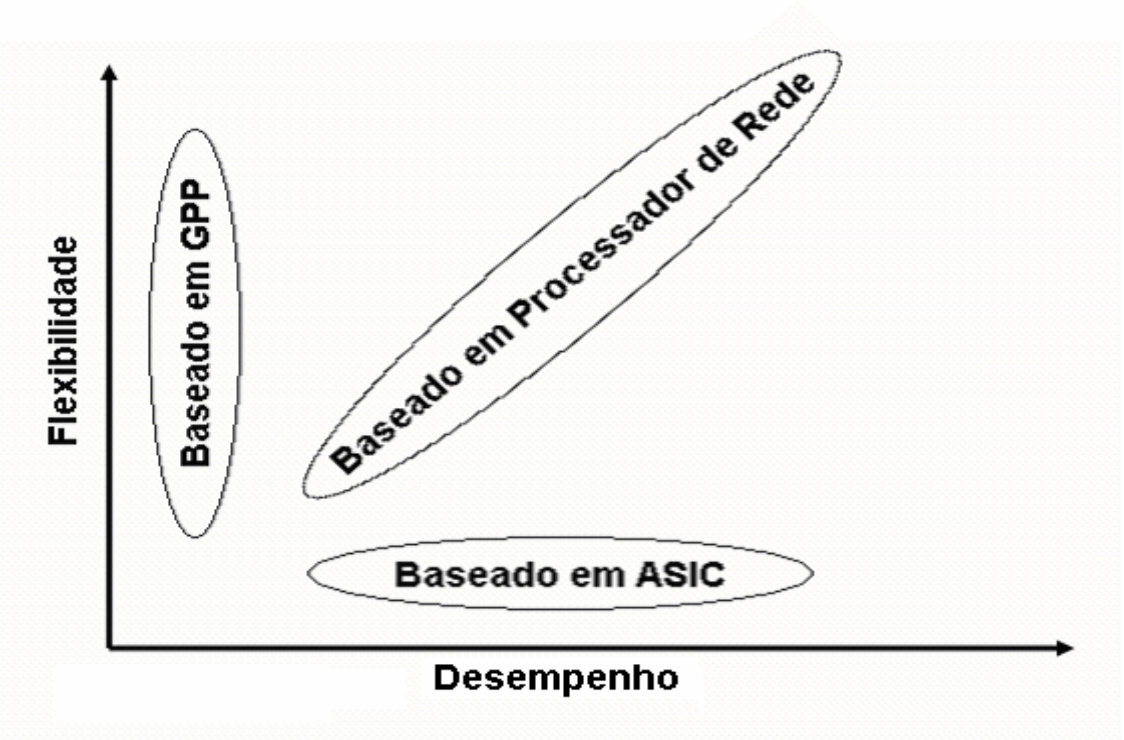


FIGURA 1.2 – Comparação dos eixos Flexibilidade e Desempenho de Chips ASICs, GPPs e NPs (PRADO, 2004)

O segmento de Processadores de Rede atualmente possui várias pesquisas, inclusive já existindo esse tipo de solução, mas a maior parte é realizada por empresas. É notório que empresas que possuem estrutura de desenvolvimento e montagem de *microchips* vêem no processador de rede um excelente investimento. O gráfico da figura 1.3 mostra esse dado comparando os ganhos do mercado de processadores de rede nos anos de 2000 a 2003. Mas essa tendência permanece, pois cada vez surgem novas pesquisas e equipamentos nesse segmento, mostrando-se uma área interessante para investimentos.

Com o advento da tecnologia FPGA, é possível implementar esse tipo de solução a um baixo custo e com um bom desempenho.

Os processadores de rede têm uma fundamental importância no desempenho das redes, possibilitando maior rapidez no processamento, já que ele exclusivamente irá executar o processamento de pacotes da rede, aumentando o desempenho e diminuindo o consumo de banda.

A implantação de um processador de rede em um equipamento de transmissão de rede como um sistema embutido é uma solução eficiente e viável, otimizando as funções de roteamento de pacotes, aumentando a velocidade, a eficiência e o desempenho de uma rede.

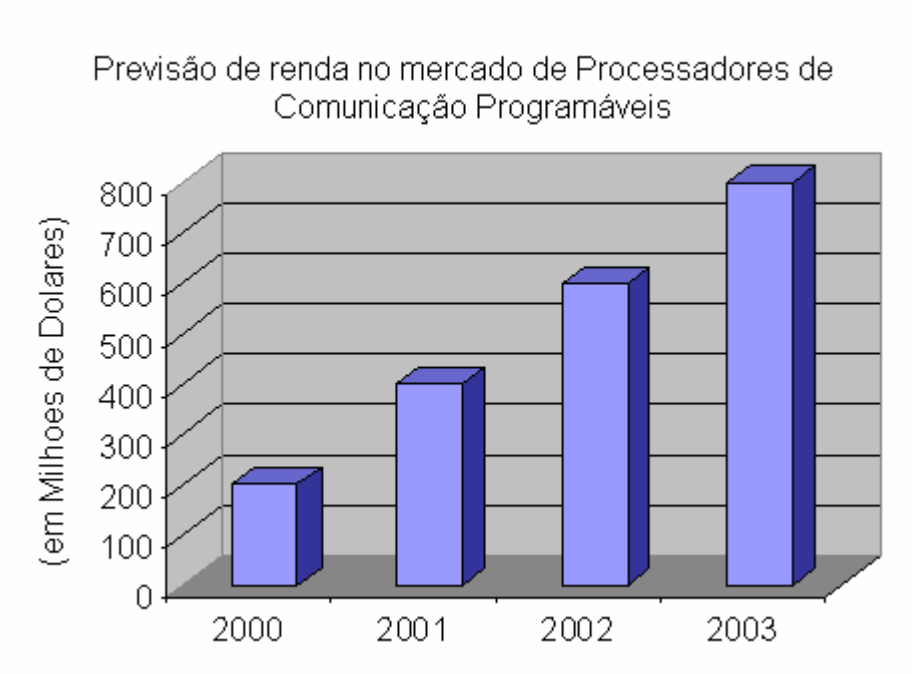


FIGURA 1.3 – Ganhos previstos no mercado de Processadores de Rede (CWYNAR, 2000).

Não menos importante, tem-se a questão da segurança na rede, onde a criptografia supre a necessidade de se manter a privacidade dos dados que trafegam na rede. Através dela, apenas quem possuir a senha (chave de criptografia) poderá decifrar a mensagem e ter acesso ao seu conteúdo.

Das soluções existentes no mercado de processadores de rede, existe desenvolvimento para fins acadêmicos ou científicos e para fins empresariais, mas poucas possuem suporte a segurança.

Atualmente existem poucos processadores de redes com primitivas de segurança. A implementação de um módulo de segurança através da inclusão de instruções criptográficas em um processador de rede irá prover um nível de segurança em uma rede, garantindo a confidencialidade na transmissão dos dados.

Também é interessante a esse processador de rede o suporte a várias topologias e também a possibilidade de alternância entre as topologias em tempo real. Isso permite que, por exemplo, em uma situação de ataque a um nó da rede, com a alteração da topologia utilizada no tráfego de informações naquele momento, haja uma alteração no fluxo da informação na rede de modo que o atacante perderia o sentido que a informação percorre dentro da topologia, permita o envio da informação ao seu destino.

1.3 Problemas Existentes nas Redes de Computadores

Atualmente, as redes de computadores são importantes devido à popularização da tecnologia existente no mercado, principalmente através da inclusão digital, e a integração dos recursos tecnológicos. Isso reflete na sua utilização, que aumenta tanto em volume de informação quanto no aumento da sua utilização (SHAH, 2001).

Isso causa um congestionamento nos *links* de transmissão, necessitando de equipamentos específicos que são capazes de realizar o processamento dos pacotes de uma rede de maneira mais eficiente (TANEMBAUM, 1999).

Para solucionar estes problemas foram criados os processadores de rede. Atualmente as principais soluções existentes no segmento de processadores de rede são proprietárias. Por possuírem estrutura e recursos, as empresas deste ramo acabam por desenvolverem produtos em ASIC por ser uma tecnologia que prove maior desempenho e sua fabricação ser em grande escala. No entanto, a inflexibilidade desse tipo de tecnologia, onde é muito difícil fazer alterações ou correções se torna sua principal desvantagem.

Também existem processadores de rede desenvolvidos com a tecnologia de GPP, que provê flexibilidade, para a necessidade de alterações necessárias sem maiores dificuldades, e lentidão, não sendo tão veloz quanto um dispositivo de rede deve ser (PRADO, 2004). A maioria dos dispositivos de rede existentes são simples e baratos, não satisfazendo essa necessidade.

Outro ponto fundamental em relação ao tráfego de informações em uma rede diz respeito à sua segurança. A maior parte de dispositivos de rede não provêem segurança alguma às informações por eles enviadas; muitas vezes, essa função é desempenhada por software, através de autenticações e criptografia. Mas, como já foi visto anteriormente, a execução em software é muito mais lenta do que a execução em hardware e, como há interesse em soluções para melhorar o desempenho da transmissão de dados em uma rede, esse fator também deve ser considerado um problema a ser solucionado.

1.4 Objetivos da Dissertação

Com a crescente demanda de largura de banda e necessidade por velocidade nas transmissões das redes atuais, os processadores de rede se inserem nesse contexto como uma alternativa, oferecendo soluções através da otimização do roteamento das redes, execução em

hardware, reprogramação de funções visando ganho em desempenho e suporte a vários protocolos e segurança, possibilitando dessa forma uma transmissão eficiente dos dados pela rede e uma grande inter-conectividade com outros tipos de redes.

Ainda nesse contexto, por mais veloz que seja uma rede, não há garantia da segurança das informações trafegadas nela, tendo a possibilidade da captura de pacotes. Insere-se também neste contexto, a criptografia como solução para a segurança da informação em uma rede, garantindo a integridade dos dados trafegados em uma rede.

A proposta desse projeto de pesquisa é a implementação de um NP (*Network Processor*) com suporte a mudança de topologia, que permita a busca de caminhos alternativos dentro da topologia escolhida, e ofereça primitivas de segurança. Este processador foi descrito em VHDL e posteriormente implementado em FPGA. O objetivo é o desenvolvimento de um processador de rede que permita o roteamento de pacotes de dados específicos para esse processador, com escolha das topologias do tipo Anel Unidiecional e Árvore Binária, com a possibilidade de mudança de rota. Adicionalmente, conter primitivas de segurança com criptografia através dos algoritmos DES (*Data Encryption Standard*) e AES (*Advanced Encryption Standard*).

O Processador ToPSeC é baseado no NPSoC (*Network Processor System on Chip*) (PRADO, 2004), que é um primeiro protótipo de um processador de rede desenvolvido pela equipe de Arquitetura de Computadores do UNIVEM. O NPSoC, por sua vez, teve por base o processador R2NP (FREITAS, 2002). Nesses trabalhos correlatos, houve uma preocupação com a inclusão de topologias como Anel Unidirecional, Árvore Binária e Hipercubo, mas não efetivamente a escolha entre elas e a busca por um caminho alternativo dentro de uma determinada topologia.

Nesse processador também foram adicionados algoritmos de criptografia para garantir a segurança dos dados por ele enviados. Esses algoritmos de criptografia foram baseados nas respectivas implementações descritas em (PEREIRA, 2004), (MORENO, 2005) e (OLIVEIRA, 2006).

1.5 Metodologia

Levando em consideração os problemas do dia-a-dia em redes e na Internet com relação à velocidade, ao desempenho, à flexibilidade e segurança, foi adotada como metodologia do projeto o estudo de NP. Foi verificada suas vantagens em relação a outros

tipos de soluções do mesmo segmento e o estudo de tecnologias que já são utilizadas para tal finalidade, como GPP e ASICs, suas vantagens e desvantagens e a escolha de uma terceira tecnologia, o FPGA.

Dessa forma, tendo como foco o desenvolvimento de um projeto similar, mas com vantagens em relação a outras tecnologias estudadas e atualmente utilizadas por grandes empresas, como flexibilidade, desempenho, baixo custo, facilidade de desenvolvimento e prototipação e, consequentemente, menor tempo de desenvolvimento e maior rapidez para o produto final chegar ao mercado.

Pelo fato da escolha da realização do projeto em FPGA, foi necessário o estudo para a implementação do projeto em VHDL, juntamente com algoritmos criptográficos, como o DES e o AES, com o intuito de desenvolver em VHDL um processador de rede com implementações de segurança com criptografia.

Também foi realizado um estudo do NPSoC (PRADO, 2004), já que este trabalho baseia-se nele, e adicionalmente, o estudo das topologias de rede no qual este processador de rede foi implementado, bem como suas funções de roteamento.

Após o desenvolvimento, foram realizadas simulações para verificar o funcionamento, permitindo assim, ter uma perspectiva para a implementação real em um FPGA.

1.6 Organização da Dissertação

A dissertação é composta de 6 capítulos, descritos a seguir:

O Capítulo 1 apresenta alguns conceitos introdutórios sobre o assunto, e mostra uma visão geral deste projeto, apresentando conceitos sobre redes de computadores, a importância na utilização dos processadores de rede e da criptografia, bem como os problemas nelas existentes, os objetivos do projeto e a metodologia adotados no projeto de um processador de rede com segurança.

O Capítulo 2 apresenta os processadores de redes, o modelo OSI e em quais camadas atuam, as diferentes tecnologias empregadas no desenvolvimento de processadores de redes e apresenta alguns processadores de rede já desenvolvidos.

O Capítulo 3 apresenta conceitos de criptografia e segurança da informação, um breve histórico acerca do assunto, os tipos de criptografia existentes e apresenta os algoritmos criptográficos utilizados neste trabalho.

O Capítulo 4 apresenta o Processador de Rede NPSoC, bem como sua arquitetura.

O Capítulo 5 apresenta o ToPSeC, mostrando sua arquitetura, modificações e implementações, as alterações sofridas a cada nova versão, as topologias utilizadas nesse processador, a possibilidade de escolha entre elas e a busca por caminhos alternativos, as instruções especiais para oferecer essas funcionalidades, as instruções para realizar criptografia e as simulações detalhadas de cada situação existente nesse processador.

Finalmente, o Capítulo 6 traz as conclusões do projeto e algumas idéias para trabalhos futuros.

Capítulo 2

Processadores de Rede em FPGAs

2.1 Introdução

Como já dito no capítulo 1, o crescente aumento da utilização de largura de banda nas redes, tanto coorporativas, particulares ou públicas, e a disponibilidade de acesso a uma rede, que a cada dia aumenta, geraram uma necessidade de produtos e serviços que melhore o uso de redes (PETRUZZA, 2004) (SHAH, 2001).

Nesse contexto que se insere a idéia dos Processadores de Rede, que são uma classe emergente de circuitos integrados programáveis baseados na tecnologia SoC, classificação essa por possuir vários blocos como memória, portas e interface de comunicação, que executam funções específicas de comunicação de forma mais eficiente que os GPPs.

Existe uma tendência em relação aos processadores de rede onde estes se tornem o principal componente para redes (SHAH, 2001). Com o avanço da tecnologia, empresas interessadas em conquistar essa fatia do mercado, estão desenvolvendo processadores de rede dedicados, capazes de serem mais velozes e apresentarem maior desempenho que os GPPs. Os processadores de rede serão uma das principais soluções dos gargalos das comunicações de roteadores, *switches*, etc.

Basicamente, as características de um processador de rede é processamento em tempo real, segurança, *store and forward*, manipulação de pacotes IP (*Internet Protocol*). Devem

ainda ser flexíveis, escaláveis, permitir troca de pacotes de tamanhos diferentes em redes com diferentes protocolos e topologias e, por fim, possuir capacidade de aprendizado (SHAH, 2001).

2.2 O Modelo OSI e os Processadore de Rede

O modelo OSI (*Open System Interconnection*) foi criado em 1977 pela ISO (*International Organization for Standardization*) com o objetivo de criar padrões de conectividade para interligar sistemas de computadores locais e remotos. Ele possui uma divisão muito clara das camadas de um sistema de comunicação. Isso é um importante auxílio para o entendimento dos diversos protocolos de comunicação do mercado. Os aspectos gerais da rede estão divididos em 7 camadas funcionais, facilitando desse modo, a compreensão de questões fundamentais sobre redes, apesar deste modelo não ter sido adotado para fins comerciais (TANEMBAUM, 1999).

A Tabela 2.1 mostra o modelo OSI e a atuação em cada uma das camadas desse modelo.

TABELA 2.1 – Modelo de 7 camadas ISO/OSI (TANEMBAUM, 1999).

Camada	Nome	Função
Camada 7	Aplicação	É representada pelo usuário final no modelo OSI, selecionando serviços a serem fornecidos pelas camadas inferiores, entre eles, o correio eletrônico, transferência de arquivos, etc.
Camada 6	Apresentação	Transfere informações de um software de aplicação da camada de sessão para o sistema operacional. Criptografia, conversão entre caracteres ASCII e EBCDIC, compressão e descompressão de dados são algumas funções acumuladas nesta camada.
Camada 5	Sessão	Reconhece os nós da rede local LAN e configura a tabela de endereçamento entre fonte e destino, isto é, estabelece as sessões, no qual o usuário poderá acessar outras máquinas da rede.
Camada 4	Transporte	Controla a transferência de dados e transmissões.
Camada 3	Rede	Conexão lógica, cuidando do tráfego e roteamentos dos dados da rede.
Camada 2	Enlace	Acesso lógico ao ambiente físico, transmissão de dados e reconhecimento de erros
Camada 1	Física	Aspectos mecânicos, elétricos e físicos

Os processadores de rede podem atuar nas camadas de 2 a 7 do modelo OSI, e são projetados para aperfeiçoar tarefas específicas de rede, podendo variar das camadas 2 a 4, 2 a

5 e 2 a 7, tornando possível a sua utilização em uma ampla faixa de aplicações de rede, dependendo para isto, do foco inicial do projeto e dos serviços para o qual o processador de rede será desenvolvido (SHAH, 2001).

As camadas a serem implementadas nos processadores de rede dependem especificamente da natureza da aplicação para a qual se deseja criar soluções.

2.3 Análise de diferentes tecnologias utilizadas no desenvolvimento de NPs

Para que uma rede funcione de maneira eficiente, ela precisa suportar novos protocolos que incluem serviços diferenciados, segurança e várias outras funcionalidades de gerenciamento de rede. Enquanto as redes demandarem equipamentos com alta taxa de transmissão de dados elas também precisarão de flexibilidade para suportar protocolos e aplicações (SHAH, 2001).

Atualmente existe o desenvolvimento de uma nova metodologia para o desenvolvimento de soluções para as redes de computadores atuais. Essa solução são os Processadores de Rede, que são dispositivos que combinam a flexibilidade dos GPPs e o desempenho dos ASICs. Isso tem atraído atenção dos grandes fabricantes de *chips*, como Intel, MMC, Lucent, Agere, IBM, etc. (PRADO, 2004). No entanto, essas tecnologias que são normalmente utilizadas, possuem desvantagens que podem descaracterizar o projeto de um processador de rede, conforme visto no capítulo 1.

As implementações de processadores de rede existentes são baseadas nas seguintes tecnologias (SHAH, 2001):

- **ASIC (*Application Specific Integrated Circuit*) –** uma solução exclusivamente em nível de hardware, possuindo alto desempenho, mas baixa flexibilidade;
- **Co-Processor –** um hardware, possivelmente configurável com uma interface de programação limitada;
- **GPP (*General Purpose Processor*) –** um processador programável para computação de propósito geral, totalmente flexível, mas oferece pouco desempenho.
- **FPGA (*Field Programmable Gate Array*) –** um dispositivo que pode ser reprogramável a nível de blocos, situando-se intermediariamente entre os dispositivos ASICs e os GPPs. Oferece alta flexibilidade e um bom desempenho;

No entanto, nenhuma das soluções acima possui todas as características que o processamento de rede requer (PRADO, 2004). Porém, dentre as tecnologias apresentadas, as que possuem uma melhor relação flexibilidade x desempenho são as tecnologias FPGA e Co-processador, de acordo com a figura 2.1.

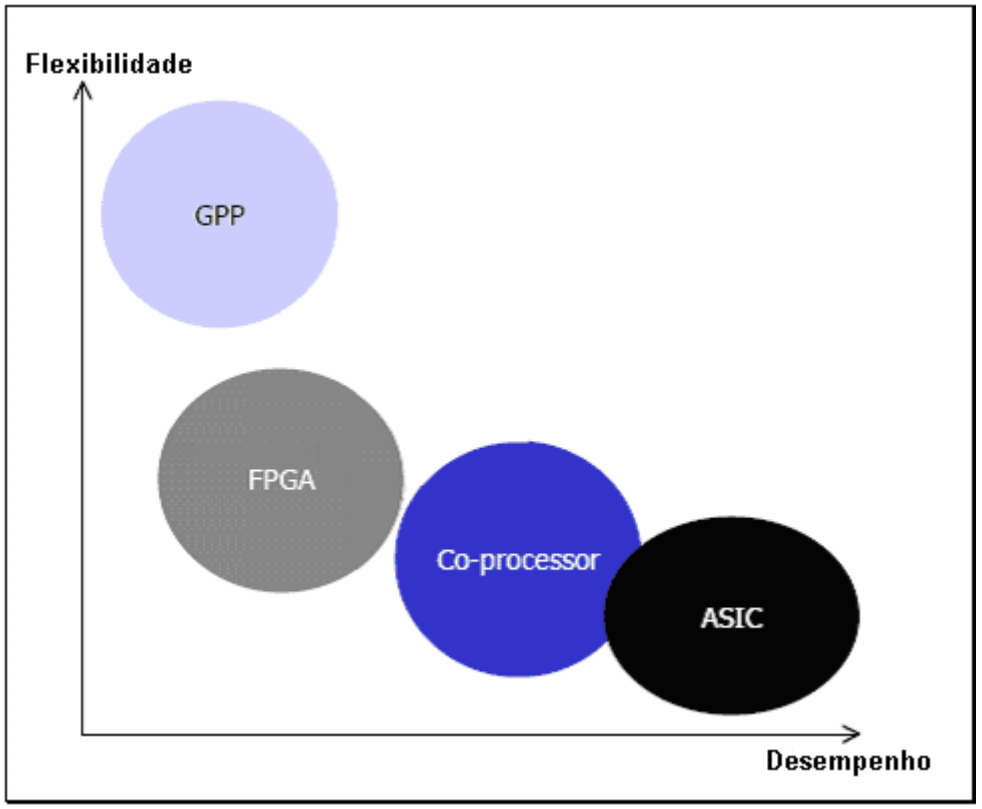


FIGURA 2.1 – Áreas de Implementação: Flexibilidade X Desempenho (SHAH, 2001).

As características da tecnologia FPGA, como execução em hardware, flexibilidade, fácil desenvolvimento, baixo custo e principalmente, facilidade na prototipagem do projeto, motivaram a escolha da tecnologia FPGA para o desenvolvimento desse projeto.

Segundo (PRADO, 2004), sistemas mais tradicionais geralmente utilizam componentes reconfiguráveis para emulação de componentes de hardware baseados em uma, duas ou mais FPGAs, e são usados para o desenvolvimento de pequenas interfaces, implementação de lógica padrão e uma pequena memória com dispositivos reconfiguráveis, o que permite desenvolver pequenos sistemas com flexibilidade superior a sistemas tradicionais. Outra vantagem clara é a possibilidade de reconfiguração do sistema, possibilitando que toda lógica do circuito possa ser facilmente modificada, reduzindo o tempo e custo de implantação.

Algumas das aplicações mais interessantes para sistema reprogramável são aquelas em que o é visto não como um elemento isolado, mas como um complemento, uma extensão de

um sistema de computação, e que irá otimizar as funções a qual se destina nesse sistema, operando como um co-processador ou um sistema embutido, conforme figura 2.2 (SHAH, 2001).

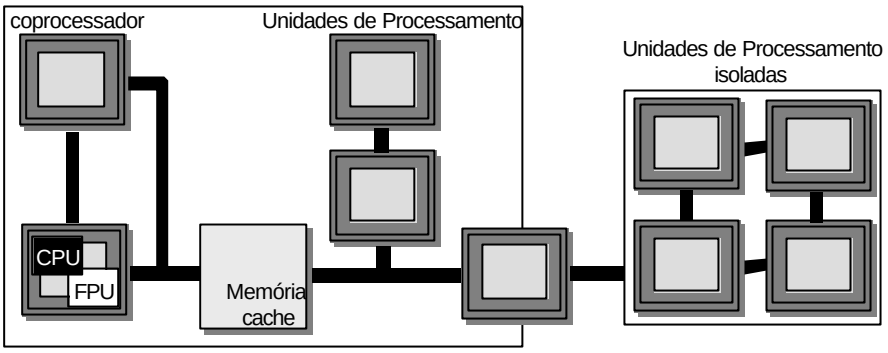


FIGURA 2.2 – Unidades de Processamento FPGAs (PRADO, 2004).

A figura 2.3 traz um comparativo entre as tecnologias citadas para o uso de processadores de rede, sendo usados os parâmetros de flexibilidade, desempenho, *time-to-market* (TTM), consumo de energia e custos de produção.

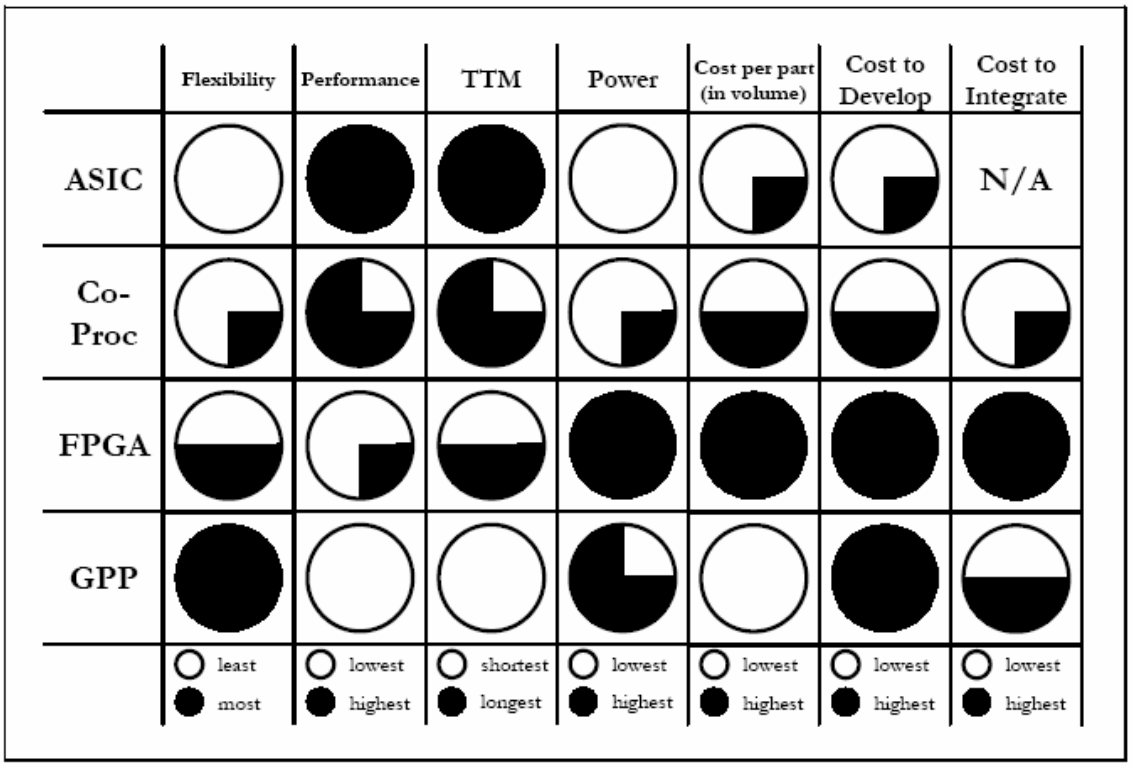


FIGURA 2.3 – Comparativo entre tecnologias utilizadas para NP (SHAH, 2001).

Segundo a figura 2.3, a tecnologia FPGA é a mais adequada para ser utilizada no desenvolvimento de processadores de rede, pois possui flexibilidade, desempenho e TTM satisfatórios.

Além das características próprias de cada tecnologia, um processador de rede deve fornecer também as seguintes funcionalidades (PRADO, 2004):

- **Análise de Pacote –** a classificação de pacotes e processamento do estado dos protocolos;
- **Pesquisa de Cabeçalho –** a identificação de um pacote particular ou fluxo para determinar o seu destino;
- **QoS (*Quality of Service* – Qualidade de Serviço) –** regras de política de segurança e exigências de processamento;
- **Modificação de Pacote –** baseado nos resultados da pesquisa, o pacote pode ser modificado antes de ser enviado aos *switches* ou mecanismos de Gerenciamento de Tráfego.

2.4 Estado da Arte dos Processadores de Rede

Os processadores de rede possuem um campo vasto e ativo, tanto para fins comerciais como acadêmicos. Processadores de rede comerciais são vistos como o próximo passo em direção ao processamento e transmissão de dados em alta velocidade e, atualmente gozam de uma segunda geração, com a fusão das companhias produtoras de processadores de rede em direção aos pontos chaves dessa nova metodologia. Ao mesmo tempo, pesquisas acadêmicas continuam desenvolvendo novos métodos para prover serviços de rede em hardware reconfiguráveis (HARPER, 2003).

Esta seção apresenta algumas das principais entidades engajadas nesse tipo de pesquisa, bem como suas soluções para redes e apresenta algumas pesquisas sobre processadores de rede acadêmicos atuais.

2.4.1 Agere PayloadPlus

Este processador é composto por três chips, FPP (*Fast-Pattern Processor* – Processador de Padrões Rápido), RSP (*Routing-Switching Processor* – Processador de Roteamento) e o ASI (*Agere System Interface* – Interface de Sistema da Agere). O FPP e o RSP são responsáveis pelo processamento de pacotes em alta velocidade enquanto o ASI provê a interface PCI (*Peripheral Component Interconnect*) para o controle do microprocessador para um processamento de controle planejado, estatísticas e monitoramento de fluxo dos dados entre o FPP e o RSP (AGERE, 2002).

O FPP recebe os dados da interface da camada física, executa reconhecimento de protocolo, classifica os dados e remonta-os antes de movê-los para o RSP. O RSP é responsável por enfileirar as modificações de pacotes, formar tráfego e rotular a QoS. A figura 2.4 ilustra como o PayloadPlus é usado em uma configuração típica de roteamento (AGERE, 2002).

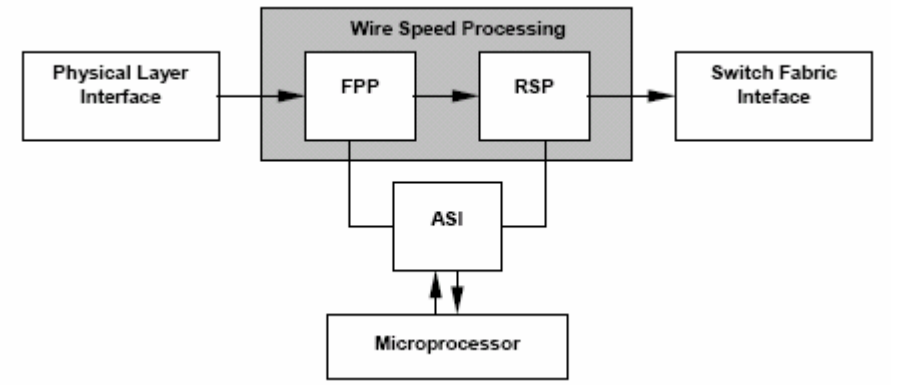


FIGURA 2.4 – Estrutura do Chip Agere PayloadPlus (STENSTRÖM, 2002).

O Processador Agere PayloadPlus possui três características principais de arquitetura que o distingue dos outros processadores de rede do mercado:

1. Ele é construído baseado em uma patente para a *"busca eficiente de uma base de conhecimento para determinar se um objeto combina qualquer pluralidade de conhecimento nas bases de entradas"*. Basicamente ele provê uma rota eficiente para determinar qual ação de roteamento de pacotes, e é usado para classificar células de redes ATM (*Asynchronous Transfer Mode*) a velocidades superiores a 2.4 Gbps (STENSTRÖM, 2002).

2. O RSP usa uma série de VLIW (*Very Large Instruction Word*) para processamento de pacotes em vez de RISC (*Reduced Instruction Set Computer*), comumente usados em outros processadores de rede (STENSTRÖM, 2002).
3. Diferente da maioria dos processadores de rede, o PayloadPlus não provê customização em C/C++ para o desenvolvimento de aplicações. A Agere começou a utilizar uma linguagem proprietária chamada FPL (*Functional Programming Language*) para programar o FPP. O FPL foi desenvolvido para ser intuitivo com o código de instruções, assim como um protocolo de definição de linguagem. De acordo com a Agere, o FPL oferece alta redução do código para aplicações, com cada linha equivalendo de 10 a 20 instruções em C. Ele é programado utilizando linguagem de script C-like e uma API (*Application Programing Interface*) que oferece baixo nível de controle do *chip set* (AGERE, 2002).

2.4.2 IBM PowerNP

O IBM PowerNP é uma solução com 16 processadores de protocolos, 7 co-processadores especializados e a arquitetura do PowerPC. Suporta pacotes sobre SONET (POS) e redes *Gigabit Ethernet* a 2.5 Gbps e é focado para atuar nas camadas de 2 a 5 (ALLEN, 2001) (SHAH, 2001).

Ele consiste em um EPC (*Embedded Processor Complex*), especial para hardware de processamento de quadros e interfaces periféricas. Cada par de processador de protocolo possui 3 estágios de *pipeline*. Dois dos processadores de protocolo são especializados, um para guiar os blocos e o outro para construir um levantamento de tabelas. Os sete co-processadores executam as seguintes funções (IBM, 1999):

- *Data store* – conecta o buffer de quadros para prover capacidade de DMA;
- *Checksum* – calcula os *checksums* iniciais;
- *Interface* – provê a todos os protocolos acesso aos registros internos, contadores e memória;
- *String Copy* – habilita movimentação eficiente de dados dentro do EPC;
- *Policy* – examina o fluxo de controle da informação e checa pela conformidade com a largura de banda pré-alocada.

Cada processador de protocolo tem uma memória de instrução de 8kb. Há várias memórias de controle interno espalhadas no chip variando de 8kb a 32kb. A figura 2.5 ilustra a macro arquitetura do PowerNP e a figura 2.6 mostra a arquitetura do EPC.

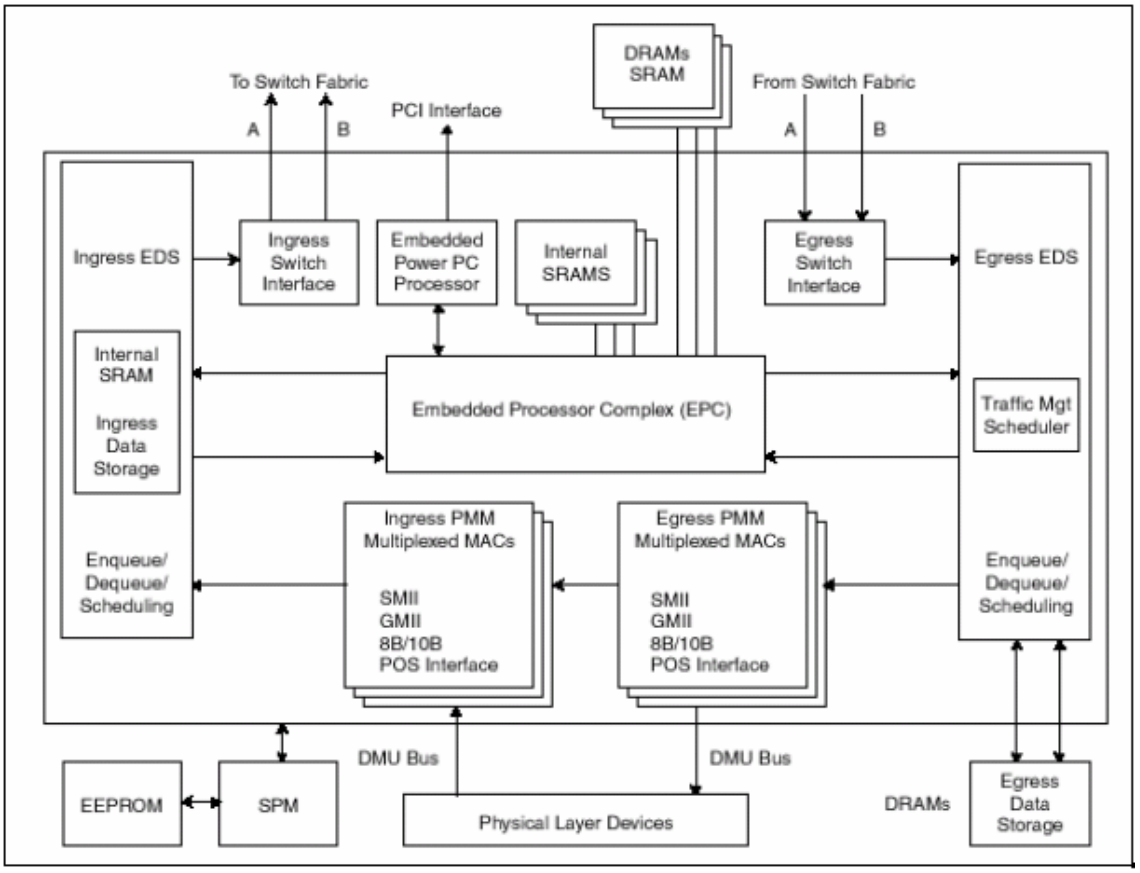


FIGURA 2.5 – IBM PowerNP (IBM, 1999).

Grande parte do processamento dos pacotes atualmente ocorre fora da EPC. Um quadro inicial primeiro vem para o PMM (*Physical MAC Multiplexer)*, que valida o quadro (CRC *check*) e o armazena em um *buffer*. Ele então passa parte do quadro para o Processador de Protocolo para um levantamento dos quadros. O assistente de classificação do hardware ajuda, identificando o formato do quadro, que é usado pelo Processador de Protocolo para executar um levantamento nas tabelas que residem no Controle da Memória. O árbitro do Controle de Memória é usado para gerenciar múltiplos levantamentos na memória para diferentes Processadores de Protocolo. Apenas o levantamento é finalizado, ingressando na Interface de Escolha para as alterações necessárias. A Unidade Completa assegura a correta ordem dos quadros antes que eles sejam colocados de volta na área de troca (STENSTRÖM, 2002).

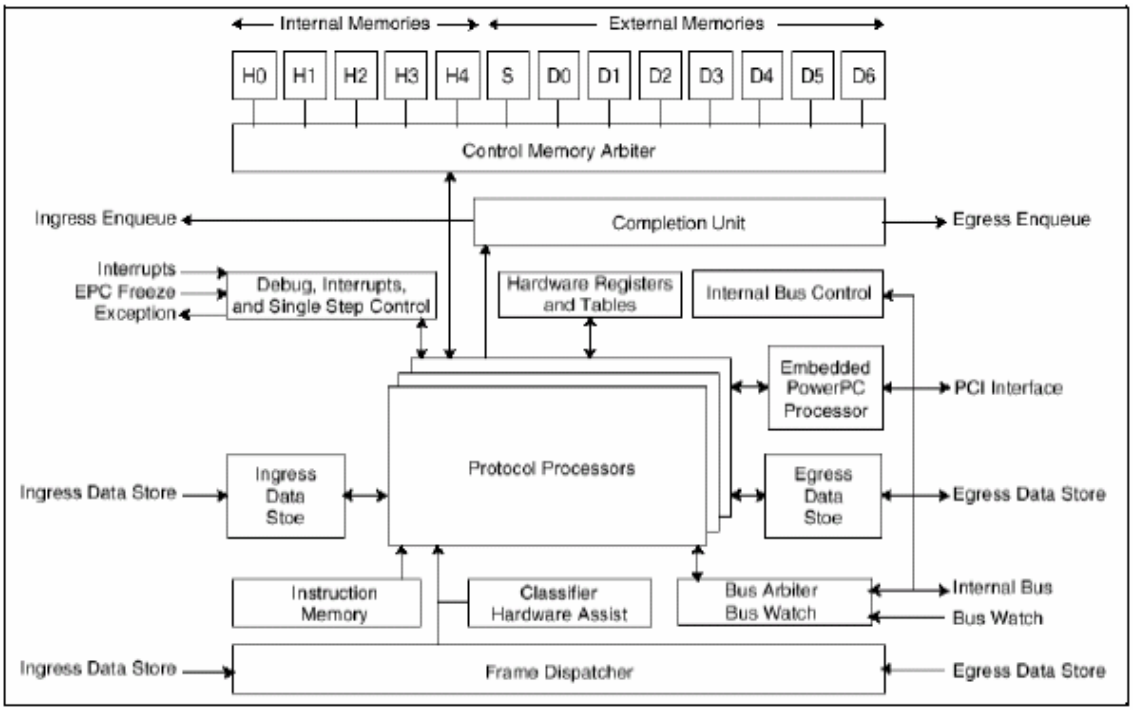


FIGURA 2.6 – Arquitetura do EPC (IBM, 1999).

2.4.3 Intel IXP1200

O Intel IXP1200 é o primeiro de uma série. Foi desenvolvido para atuar nas camadas de 2 a 4 e pode suportar uma taxa de pacotes de 2.5 Mpackets/s. Assim como outros dispositivos, camadas mais altas podem ser suportadas conectando processadores externos com a interface PCI. O IXP consiste em 6 "*micro-engines*" e um controlador StrongARM. As *micro-engines* possuem hardware para suportar 4 *threads* cada. Há também um hardware especial para funções de *hash*, enfileiramento e ciclos de mudança simples e giro (SHAH, 2001) (INTEL, 2004).

O IXP1200 é composto de 6 *micro-engines* programáveis e um StrongARM de 200 MHz que coordena as atividades do sistema. O barramento IX, de 64-bits, oferece alta conectividade de largura de banda para as *micro-engines*, StrongARM, memória e para dispositivos *off-chip* como um dispositivo MAC ou outro IXP1200. A interface do barramento PCI permite a integração com um processador de controle externo.

As *micro-engines* executam todas as tarefas de processamento de pacotes. Eles têm hardware para suportar quatro *threads* cada, para um total de 24 *threads* em um chip. Enquanto as quatro *threads* em uma *micro-engine* compartilham o arquivo de registro, há um software que policia as separações do arquivo de registro em quatro partes, uma para cada

thread. As *micro-engines* também têm instruções especiais para o processamento de pacotes, como achar o primeiro bit, mudança de contexto e extrair *byte* / palavra.

Ainda, para as *micro-engines*, o IXP tem algumas unidades de hardware especiais que ajudam no processamento do pacote. Há uma ferramenta programável de *hash* e filas especializadas compartilhadas por todas as *micro-engines* e o StrongARM. Receber/transmitir FIFOs (*First In First Out)* provê uma interface aos dispositivos da camada-MAC pela leitura/escrita de pacotes dentro/fora das filas *on-chip* que podem ser acessados via barramento IX. A figura 2.7 ilustra a arquitetura do IXP1200.

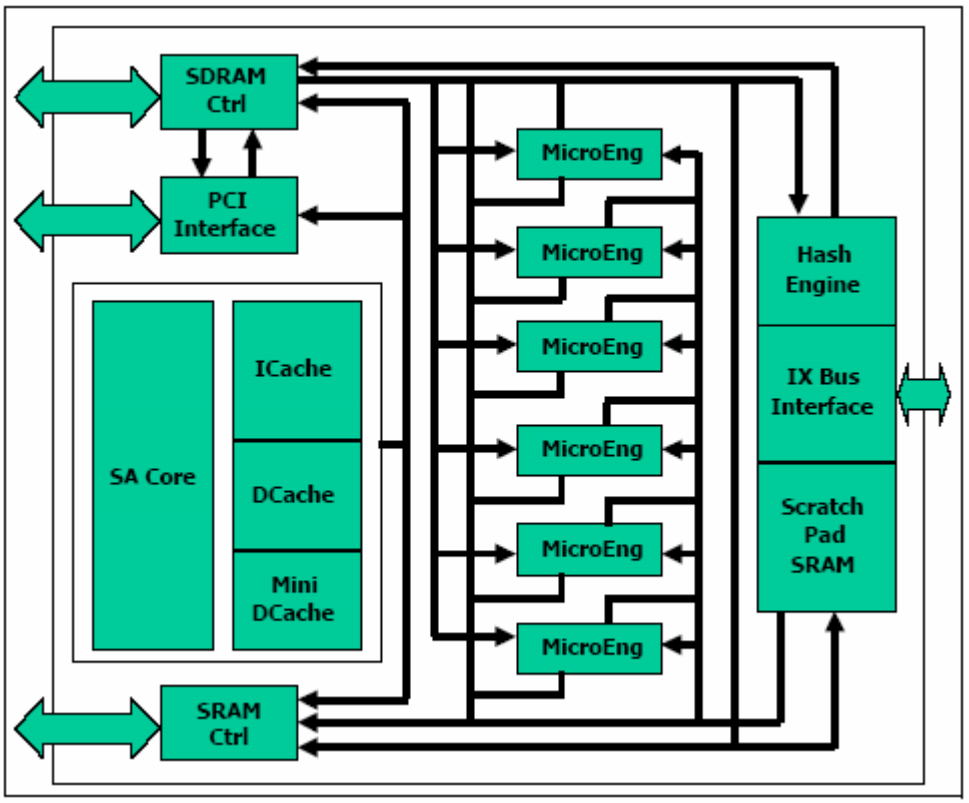
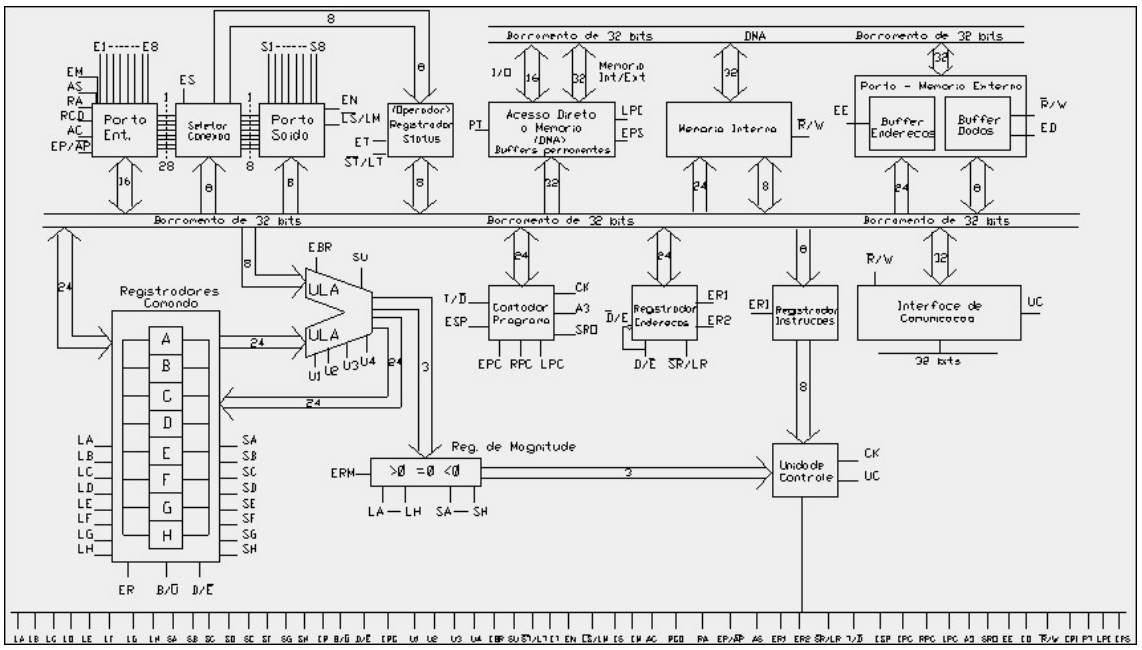


FIGURA 2.7 – Arquitetura do Intel IXP1200 (INTEL, 2002).

A programação no IXP1200 é totalmente feita em *macro-assembly*, através do compilador C. Dado que as 6 *micro-engines* executam em paralelo, programar no IXP se torna uma tarefa difícil. Este processo é acentuado pela linguagem *assembly* incluindo contexto de escolha e outras características únicas. No entanto, seu IDE (*Integrated Development Environment*) prove uma significativa ajuda na programação do dispositivo. Seu ambiente de simulação de configurações e claras visualizações mostram toda atividade no chip, executando o *debugging* muito mais facilmente (INTEL, 2002).

2.4.4 RCNP – Processador de Rede com suporte a Multi-protocolo e Topologias Dinâmicas

No RCNP (*Reconfigurable* CISC *Network Processor*) – Processador de Rede com suporte a Multi-protocolo e Topologias Dinâmicas, o projeto inicial foi o desenvolvimento de um processador usando tecnologia CISC (*Complex Instruction Set Computer*), com barramento de endereço de 24 bits, e barramento de dados de 8 bits, uma memória de 16 Mb, 8 portas de entrada com *buffers* reconfiguráveis (figura 2.8). Também possui 8 portas de saída que são ligadas às portas de entrada por um seletor de conexão do tipo *Crossbar* reconfigurável, que permitiria topologias diferentes (FREITAS, 2001).



FIFURA 2.8 – Arquitetura do RCNP (FREITAS, 2001)

Uma das principais características deste processador é a possibilidade de reconfiguração dos *buffers* de entrada. O funcionamento ocorre existindo fisicamente 8 entradas, portanto 8 *buffers* temporários. Quando um pacote chega por uma destas entradas, o processador analisa o cabeçalho e tenta redirecionar a saída desejada, caso esteja livre (*cut through*). Caso a saída esteja sendo utilizada, o processador guarda o pacote em memória ou libera o pacote para ser armazenado em *buffers* permanentes da Porta de Entrada. Uma vez o pacote liberado, é feita a verificação do seu tamanho (reconfiguração – análise feita pelo *hardware*) para alocação dos *buffers* permanentes. Cada *buffer* permanente possui um

pequeno registrador que guarda a informação da última Porta de Entrada, de onde veio o pacote. Esta informação é muito útil quando não se encontra o endereço de destino na tabela de roteamento e é necessário reenviar o pacote para sua origem.

A técnica *stored and forward* é utilizada quando há a necessidade de guardar o pacote na memória ou nos *buffers* permanentes. A técnica *cut through* é utilizada pelo processador, que tenta enviar o pacote sem precisar guardá-los em *buffers* temporários.

Foi também definido um conjunto de instruções composto de 54 instruções, entre elas as de operações lógicas, aritméticas, acesso às portas, saltos condicionais e incondicionais e de roteamento, e uma breve descrição das instruções de roteamento do RCNP citadas na Tabela 2.2, como no caso de acesso às portas de comunicação. É importante perceber que as instruções irão controlar o funcionamento dos *buffers* reconfiguráveis, aumentando a otimização do armazenamento dos pacotes. Irão controlar também o acesso ao barramento e ao Seletor de Conexão, aumentando, portanto, a flexibilidade de conexões entre portas e, por conseqüência, criando topologias novas sempre que necessário.

TABELA 2.2 - Algumas instruções específicas de roteamento do RCNP. (FREITAS, 2001)

ISA – *Rede*	***Descrição do Funcionamento***
LRS	Reg. Status ← Reg. B
SRS	Reg. B ← Reg. Status
SEC	Buffer (Seletor de Conexão) ← Reg. B
SEL	Buffer (seleção PS) ← Reg. B
SAI	Buffer (mensagem PS) ← Reg. A
BRC	PS (Todas) ← Buffer de Entrada
FCX	OS ← Buffer temporário (Fecha conexão)
ENT BC	Reg. A ← Buffer (mensagem PE – posição BC)
ENT	Reg. A ← Buffer (mensagem PE – posição inicial)
PUT	Buffer (seleção PE) ← Algoritmo de escolha
PUT B	Buffer (seleção PE) ← Reg. B
SETDSC	Reg. Desconexão ← Reg. D (buffer) Reg. C (dado)
SET C,DE	Buffer C (tam. Pacote) ← Reg. D até E
TAM	Reg. C e D ← Buscar tamanho pacote (Buffer de trabalho, ecolhido por PUT)
TAM A	Reg. C e D ← Buscar tamanho pacote (Buffer escolhido por Reg. A)

2.4.4 A R2NP - Processador de Rede RISC Reconfigurável

O R2NP (*Reconfigurable RISC Network Processor*) – Processador de Rede RISC Reconfigurável, é na verdade um SoC, pois é composto por vários blocos lógicos que na

maioria das vezes são externos e trabalham em conjunto com processadores, como se pode ver na figura 2.9. A utilização de *micro-engines* e multiplexador no lugar de *buffers* de entrada temporários e estáticos se deve ao fato do *micro-engine* ser um hardware dedicado que tem a função de ler um espaço de memória, onde estão guardados dados referentes ao protocolo, para tomar decisões de roteamento preliminares, redirecionando os pacotes para os *buffers* reconfiguráveis ou para a saída, através da chave *crossbar* (FREITAS, 2002).

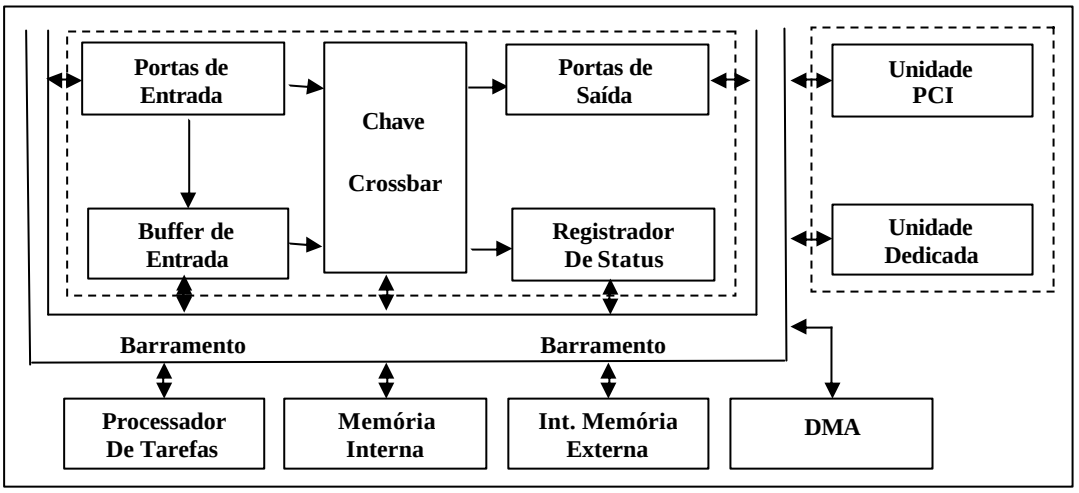


FIGURA 2.9 – Arquitetura do R2NP (FREITAS, 2002).

A função do multiplexador é possibilitar a utilização de uma porta de entrada, que estava ligada a uma *micro-engine*. Pode-se perder uma *micro-engine*, mas continuando com 8 portas.

No R2NP não existem mais as duas categorias de *buffers* (temporários e permanentes) (FREITAS, 2001). Os *buffers* temporários foram substituídos pelas *micro-engines* e os *buffers* permanentes continuam reconfiguráveis como no RCNP. O seletor de conexão passou a se chamar apenas Chave *Crossbar* Configurável.

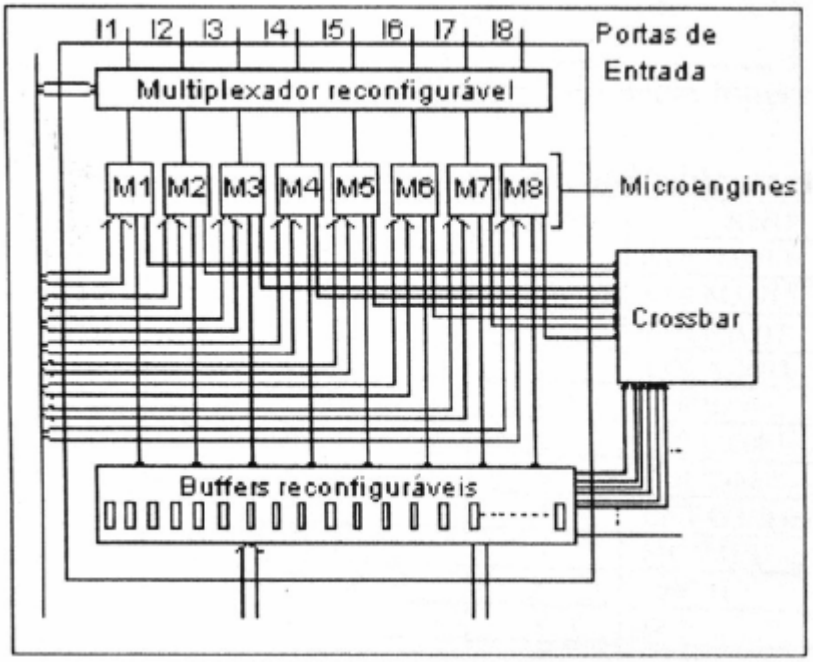


FIGURA 2.10 – Microarquitetura das portas de E/S (FREITAS, 2002).

Seu funcionamento é similar a uma matriz de conexões internas capaz de redirecionar pacotes das portas de entrada para as portas de saída, tal como no RCNP. A Figura 2.10 ilustra a microarquitetura das portas de entrada, *micro-engines*, chave *crossbar*, multiplexador e *buffers* reconfiguráveis.

2.5 NPSoC – Um novo Processador de Rede

O NPSoc é baseado no RCNP – Processador de Rede com Suporte a Multi-protocolo e Topologias Dinâmicas (FREITAS, 2000) e (FREITAS, 2001) e no R2NP – Processador de Rede RISC Reconfigurável (FREITAS, 2002). Estes processadores não foram implementados ainda, somente um módulo, o do *CrossBar*, foi simulado na ferramenta Xilinx. Espera-se uma futura integração do trabalho aqui realizado com o módulo Crossbar da PUC-MG. No NPSoC, foi definido um conjunto de instruções simples, porém que dá cobertura ao funcionamento do processador, como seus registradores, ULA, a UC, PC, RI e a maioria das instruções de roteamento, necessárias para a execução de alguns algoritmos propostos para a simulação do funcionamento de algumas redes. Portanto, para o desenvolvimento do NPSoC, deu-se especial atenção a estes dois processadores, RCNP e R2NP, os quais somente trabalharam em nível de simulação (PRADO, 2004).

2.5.1 Arquitetura do NPSoC

O NPSoC possui as características do RCNP, sendo modificado a proposta inicial devido à complexidade na implementação das *microengines*, as quais são verdadeiras caixas pretas, dificultando assim a visualização de seu funcionamento, mas não perdendo a característica RISC (PRADO, 2004), já que a quantidade de instruções ficou bastante reduzida pelo fato de que só foram implementados alguns algoritmos de roteamento, para redes tipo anel, estrela, árvore e hipercubo. Sua arquitetura ficou então definida como segue:

- 8 portas de saída nomeadas de B1 a B8;
- 8 registradores de 32 bits;
- ULA;
- Contador de Programa;
- Unidade de Controle;
- Registrador de Instruções;
- Registrador de Endereços.

Na figura 2.11, fica claro que o NPSoC é um processador com módulos especiais, instruções de roteamento, porém com todos os componentes necessários para o funcionamento de qualquer processador, como contador de programa, UC, ULA, registrador de instruções e etc. (PRADO, 2004).

O processador depende de uma Unidade de Controle para realizar suas operações de controle, o que especificamente neste processador, se resolveu fazer utilizando o método de FSM (*Finite State Machine*), ou seja, máquina de estados finita, onde seus estados são muito bem definidos e respeitam o ciclo de busca, decodificação e execução dos computadores Von Neumann. O ciclo se executa desta maneira:

1. É feita a busca da 1ª instrução;
2. Soma-se 1 para o contador de programa, ou seja, atualiza o contador;
3. Decodifica a instrução, através de seus *opcodes*;
4. Se essa instrução trouxer dados, serão armazenados em registradores internos;

5. Executa a instrução, normalmente isso é feito pela ULA;
6. Registra os resultados em local apropriado: memória, *buffer* e etc; Retornando ao passo 1, ou seja, irá buscar a próxima instrução, dando continuidade ao ciclo.

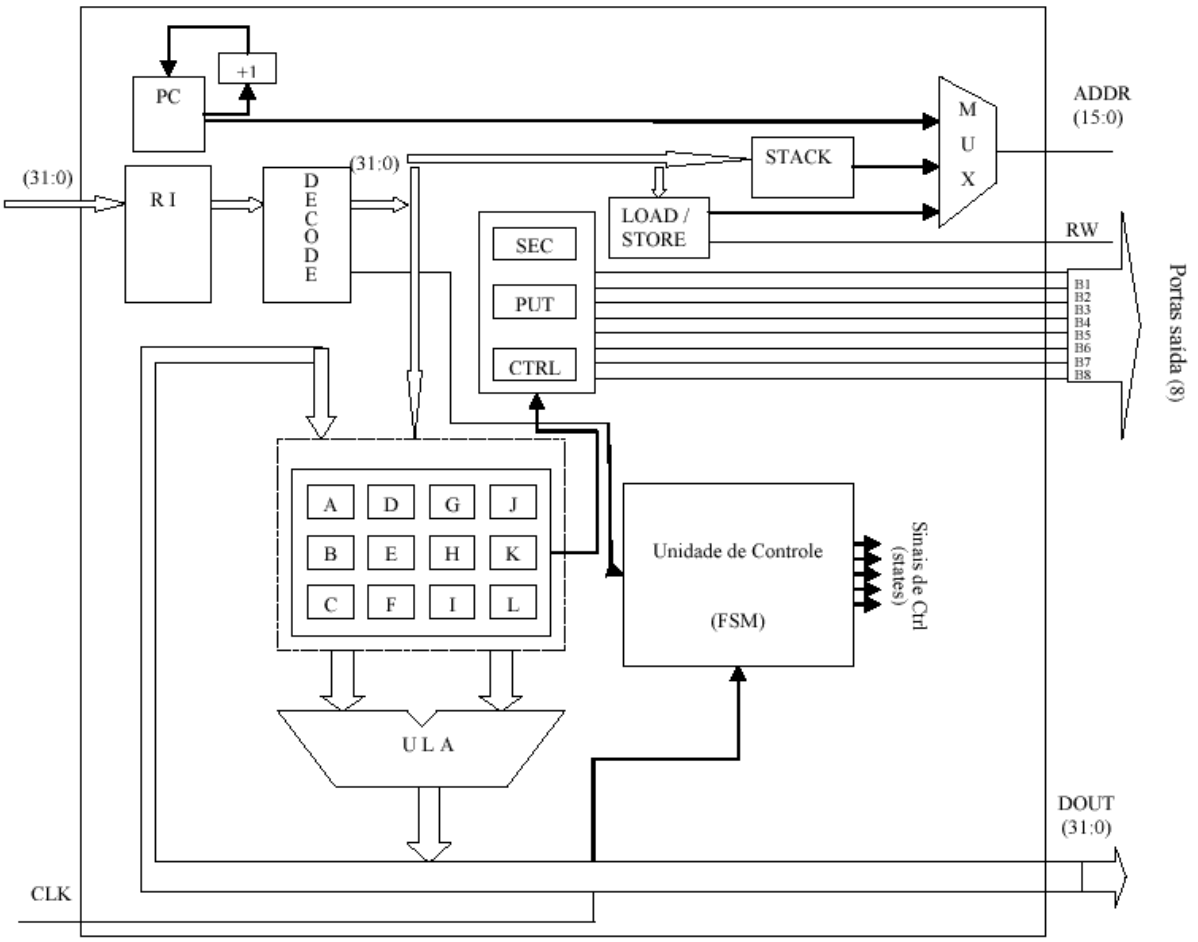


FIGURA 2.11 – Arquitetura detalhada do NPSoC (PRADO, 2004)

A comunicação com esse local apropriado, é feita por meio de um barramento cuja função é levar os dados da ULA para a memória, instruções para o RI, realimentar os registradores da ULA e etc. Para isso foi necessária a implementação de uma memória (uma pilha) de 512 Kbytes, como se pode observar na Figura 2.12.

No NPSoC, além das instruções comuns em qualquer processador, como instruções aritméticas e lógicas, de manipulação e de desvio, temos a implementação de um módulo especial com instruções específicas de roteamento (PRADO, 2004).

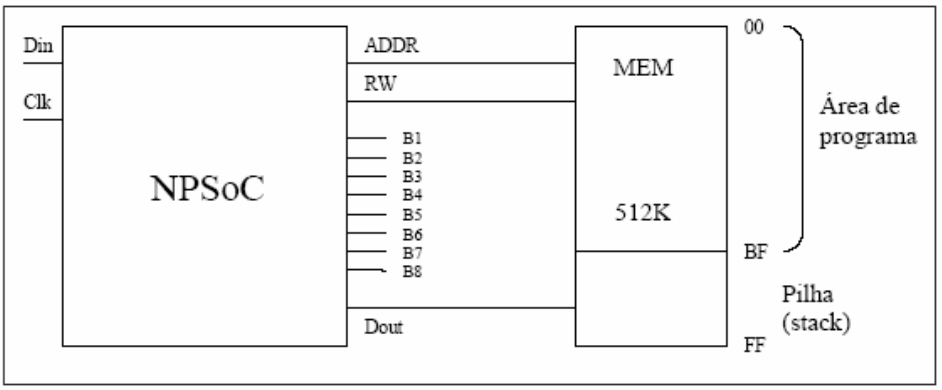


FIGURA 2.12 – *Top Level* da arquitetura NPSoC (PRADO, 2004)

4.3 Conjunto de instruções (ISA)

O conjunto de instruções do Processador de Rede NPSoC segue o modelo RISC, onde, com apenas 32 instruções, é possível descrever a maioria dos algoritmos de roteamento presentes nos NPs comerciais atuais. As instruções foram divididas em quatro classes (PRADO, 2004), conforme tabela 2.3:

TABELA 2.3 – Instruções utilizadas no NPSoC

Lógicas e Aritméticas

AND	Compara dois registros	**SUB**	Subtrai dois registros
OR	Compara dois registros	**INC**	Incrementa um registro
XOR	Idem OR, porém nega o resultado	**DEC**	Decrementa um registro
ADD	Soma dois registros	**NEG**	Nega o registro

Movimentação

LDA	Guarda os dados indo para a memória	**LID**	Atribui um valor para o registrador
MOV	Transfere os dados entre células de memória e registradores	**STR**	Guarda os dados vindo da memória
STX	Inverte os operandos	**SPUSH**	Coloca o conteúdo de um registro na pilha
ROD	Rotação à direita	**ROE**	Rotação à esquerda
SPOP	Retira um valor da pilha e coloca em um registro		

Desvio

JMP	Faz um desvio incondicional	**JZ**	Desvia se zero
JNZ	Desvia se não zero	**JMI**	Desvia se valor maior ou igual
JMZ	Desvia se valor maior que zero		

Especiais

BRC	Faz *broadcasting* para todas as portas de saída com o conteúdo do *buffer* de entrada	**PUT B**	Retira o valor do registrador B e atribui ao *buffer*
SEC	Atribui ao *buffer* o conteúdo do registrador B	**PUT**	Compara o maior vizinho ímpar
FCX	Fecha conexão entre um *buffer* temporário e a porta de saída	**SAI**	Envia dados do registrador para a saída escolhida
ENT	Atribui ao registrador A o conteúdo do *buffer* da porta de entrada	**SUI**	Recebe os dados do *buffer* para um registrador escolhido
SETDSC	Seta em registradores dos *buffers* temporários qual valor que, contido no pacote, desfaz a conexão feita por FCX	**SET C,D,E**	Configura os registradores com o local exato onde encontrar a informação do tamanho do pacote que será recebido

As instruções especiais se referem àquelas utilizadas para o roteamento e transição de dados entre os *buffers* reconfiguráveis e os registradores, bem como o seletor de conexões.

4.4 Resultados do NPSoC

Segundo (PRADO, 2004) o NPSoC, é o primeiro processador de rede, no Brasil, implementado em VHDL e prototipado em FPGAs.

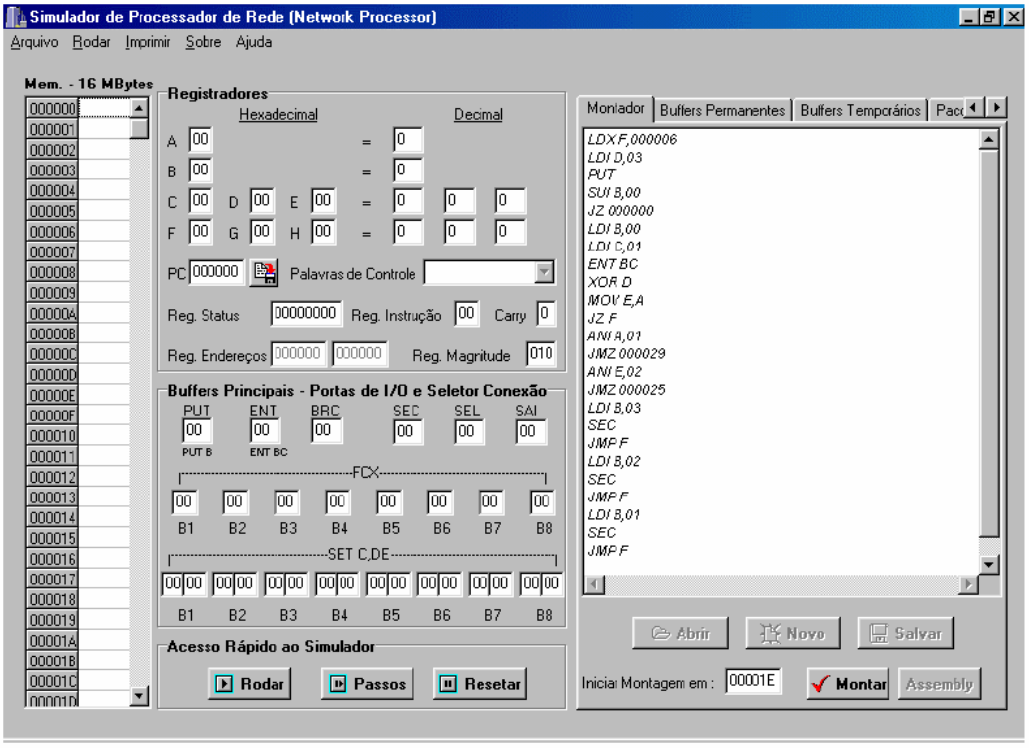


FIGURA 2.13 – Interface do montador do NPSim (FREITAS, 2000)

Como forma de validar a arquitetura e o conjunto de instruções proposto, utilizamos o Simulador NPSim (Network Processor Simulator) (FREITAS 2000), criado pelo grupo da PUC-MG, o que permitiu aferir o comportamento dos principais blocos da arquitetura e suas instruções. A figura 2.13 ilustra a interface do montador do NPSim.

No NPSoC, apenas foi implementado o suporte a topologia Anel Unidirecional, sendo que essa implementação o coloca como um concentrador de rede, havendo um NPSoC no centro dessa topologia e distribuindo a informação entre os outros nós da rede.

Com relação às estatísticas de implementação e desempenho do NPSoC, utilizando um FPGA Spartan2 XC2S200E, o tempo de propagação do circuito foi de 14,976 ns e a sua freqüência foi de 66, 773 Mhz, podendo ser observado que, pelo fato de oferecer suporte a operações básicas como AND e OR, e por oferecer suporte à topologia Anel Unidirecional, conclui-se que o NPSoC possui um desempenho satisfatório.

A seguir, a tabela 2.4 traz um comparativo de algumas características do processadores relatados neste capítulo:

TABELA 2.4 – Comparativo entre os Processadores de Rede

	Processador	Tipo de Processamento	Tecnologia de implementação	Suporte a Reconfiguração	Multi-Protocolos	Suporte a Segurança
RCNP	CISC	Pipeline	FPGA	SIM	SIM	NÃO
R2NP	RISC	Pipeline	FPGA	SIM	SIM	NÃO
NPSoC	RISC	Paralelo	FPGA	SIM	SIM	NÃO
PayloadPlus	RISC	Pipeline	ASIC	SIM	SIM	NÃO
PowerNP	RISC	Pipeline	ASIC/GPP	SIM	SIM	NÃO
IXP1200	RISC	Paralelo	ASIC/GPP	SIM	NÃO	SIM

Os processadores de rede é uma nova metodologia para solucionar os problemas de demanda por largura de banda e velocidade na transmissão dos dados.

Várias propostas de processadores de redes são disponíveis, tanto comerciais como acadêmicos. No entanto, é raro uma implementação que ofereça suporte a criptografia. O próximo capítulo trás conceitos sobre criptografia e algoritmos criptográficos.

Capítulo 3

Conceitos de Segurança de Dados e Criptografia

3.1 Segurança da Informação

A segurança eletrônica nunca foi tão importante e discutida como nos dias de hoje. Cada vez há mais casos de violações de contas bancárias, acesso a informações sigilosas, invasões e destruições de sistemas. Hoje em dia informações são transmitidas com mais eficiência e velocidade, mas nem sempre de maneira segura (MORENO, 2005).

O conceito de segurança da informação é muito mais amplo do que a simples proteção dos dados em *nível lógico*. Para proporcionar uma segurança real é preciso estar atento a vários fatores. Em primeiro lugar há a necessidade de se caracterizar o sistema que irá armazenar as informações para poder identificar as ameaças, e neste sentido, fazer a seguinte subdivisão (LOPEZ, 2004):

1. **Sistemas Isolados –** São aqueles que não estão conectados a nenhum tipo de rede;
2. **Sistemas Interconectados –** Atualmente, a maioria dos computadores existentes pertence a uma rede de computador, enviando e recebendo informações do exterior quase que constantemente. Neste tipo de sistema sempre há riscos potenciais.

E quanto às questões de segurança, o seu objetivo pode ser classificado em:

- **Segurança Física –** Esta categoria engloba todos os assuntos relacionados com a salvaguarda e o suporte físico da informação. Neste nível estão as medidas contra incêndio, sobrecargas elétricas, políticas de *backup*, etc;
- **Segurança da Informação –** Nesta categoria apresenta-se a preservação da informação frente a pessoas não autorizadas. É neste nível que entra a criptografia, simétrica e assimétrica;
- **Segurança do Canal de Comunicação –** Raramente os canais de comunicação são seguros. Devido ao fato de a maior parte das redes existentes pertencerem a terceiros, se torna muito improvável assegurar totalmente a segurança nela;
- **Problemas de Autenticação –** Devido aos problemas nos canais de comunicação, se torna necessário assegurar que a informação que se recebe em um computador vem realmente de quem efetivamente a enviou. Neste nível é conveniente a utilização da Assinatura Digital;
- **Problemas de Suplantação –** Nas redes existe a possibilidade de que qualquer usuário autorizado possa acessar as informações, mesmo de fora. Assim, tem-se que garantir que usuários não estão sendo suplantados por intrusos. Normalmente se utiliza mecanismos baseados em *password* para solucionar este problema;
- **"Descarte de Informação" ou Não Repúdio –** É necessário haver uma política de não descarte de informação, pois esta, se armazenada, pode revelar o autor de uma determinada ação. Ex: um usuário ter realizado um ato de comércio e posteriormente tentar negar esse mesmo ato.

A tabela 3.1 mostra a expansão dos objetivos principais dos sistemas criptográficos.

TABELA 3.1 – Objetivos dos Sistemas Criptográficos (MENEZES, 1996).

Características	Descrição
Privacidade ou confidencialidade	Manter informações secretas disponíveis para quem está autorizado
Integridade dos dados	Garantir que a informação não seja alterada por meios desconhecidos ou não autorizados
Autenticação ou identificação da entidade	Validação da identidade de uma entidade
Autenticação de mensagens	Confirmar a origem da informação
Assinatura	Maneira de associar a informação à entidade
Autorização	Comunicar, à outra entidade, da permissão oficial para fazer algo ou ser alguém
Validação	Maneira de fornecer tempo limite de autorização para usar ou manipular informações e recursos
Controle de acesso	Restringir o acesso aos recursos à entidades privilegiadas
Certificação	Confirmação de uma informação por uma entidade confiável
Selo de criação	Gravação da hora de criação ou tempo de existência da informação
Testemunhar	Verificação da criação ou existência da informação por uma entidade diferente da criadora
Recibo	Reconhecimento de que a informação foi recebida
Confirmação	Reconhecimento de que o serviço foi providenciado
Propriedade	Maneira de prover uma entidade com os direitos legais para usar ou transferir um recurso para outras entidades
Anonimato	Ocultar a identidade de uma entidade envolvida no processo
Não-repúdio	Impedir a recusa de compromissos ou ações anteriores
Anulação	Revogação de certificação ou autorização

3.2 Criptografia

A palavra criptografia vem do grego "*kryptós*" que significa oculto, e "*grápheim*" que significa escritura, e sua definição é "*Arte de escrever com chave secreta de modo enigmático*" (LOPEZ, 2004).

Pode-se definir a criptografia como sendo um conjunto de métodos e técnicas para cifrar ou codificar informações legíveis por meio de um algoritmo, convertendo um texto original em um texto ilegível, sendo possível, mediante processo inverso recuperar as informações originais (MORENO, 2005). A figura 3.1 mostra esse processo.

FIGURA 3.1 - Esquema geral para cifragem de um texto (PEREIRA, 2004).

Uma definição mais técnica para Criptografia é, o estudo de técnicas matemáticas que relatam os aspectos da informação, assim como a confidencialidade, integridade da informação, autenticação de entrada e autenticação de dados na origem (MENEZES, 1996).

A criptografia pode ser utilizada basicamente por meio de códigos ou de cifras. Os códigos funcionam trocando partes da informação inicial por códigos predefinidos. Para se conseguir visualizar essa informação que foi codificada, deverá se conhecer quais códigos foram utilizados. Já as cifras são técnicas nas quais a informação inicial é cifrada, ou seja, há a transposição e/ou substituição dos caracteres da informação inicial, gerando uma segunda informação não legível. Somente terá acesso ao significado da informação cifrada quem conhecer o processo de cifragem (MORENO, 2005). As cifras incluem o conceito de chaves, que dependendo do tipo de algoritmo utilizado, podem ser públicas ou privadas, apresentado na seção 3.5.

A criptografia é uma arte tão antiga quanto a própria escrita. Já estava presente no sistema de escrita hieroglífica dos egípcios; os romanos por sua vez, utilizavam códigos secretos para comunicar planos de batalha.

Um dado interessante é que a criptografia permaneceu estagnada até meados do século XX, tendo a partir deste momento, um desenvolvimento acelerado com a utilização do computador, onde a criptografia realmente cresceu, incorporando complexos algoritmos matemáticos. Isso ocorreu após a Segunda Guerra Mundial (PEREIRA, 2004).

3.1.1 Criptografia tradicional

Baseava-se em sistemas de substituições monoalfabéticas (troca dos caracteres que compõem a mensagem, caractere a caractere, numa lógica ou seqüência pré-estabelecida), polialfabéticas (substituições de grupos de caracteres numa seqüência pré-estabelecida), permutações ou transposições (alteração da ordem dos caracteres).

Outra técnica é conhecida como esteganografia, que consiste em ocultar a mensagem de forma que não seja percebida. Uma forma moderna de esteganografia são mensagens escondidas em imagens e que trafegam despercebidas pela Internet (SILVA, 2003).

Um dos algoritmos de transposição monoalfabéticas mais antigos que se tem notícia é a chamada cifra de César, cuja autoria é atribuída ao Imperador Romano Júlio César. Consistia basicamente em somar a cada letra do alfabeto um valor fixo, de forma que ficasse ilegível. Para ler a mensagem bastava subtrair esse valor fixo somado para achar o significado

da mensagem. Neste algoritmo já se pode notar o conceito de chave de cifragem, revolucionando a criptografia (PEREIRA, 2004).

A seguir temos a figura 3.2 demonstrando a cifra de substituição monoalfebética.

TP = CRIPTOGRAFIA

Chave = 13

TC = PEVCGBTPNSVN

FIGURA 3.2 – Cifra de substituição (SILVA, 2003).

Na figura 3.2 vê-se o exemplo de cifra de transposição com chave = SENHA, os números sob essa palavra representam a sua ordem alfabética e representam a ordem em que as colunas serão lidas.

S	E	N	H	A	Chave = SENHA
5	2	4	3	1	TP = TEXTONAOCIFRADO
T	E	X	T	O	TC = OIOEARTCDXOATNF
N	A	O	C	I	
F	R	A	D	O	

FIGURA 3.3 – Cifra de transposição (SILVA, 2003).

As substituições polialfabéticas correspondem a aplicar-se “n” cifras monoalfabéticas. Um exemplo desta técnica é a cifra de *Vigenère* que utiliza como chave um conjunto de letras ($l_1, l_2 .... l_n$). Primeiramente divide-se a mensagem “M” em grupos de “n” caracteres ($m_1, m_2 .... m_n$) e aplica-se a seguinte operação, utilizando o alfabeto de 26 letras:

$M' = (m_1 + l_1) \bmod 26, (m_2 + l_2) \bmod 26 \ ..... \ (m_n + l_n) \bmod 26$ (LOPEZ, 2004)

A figura 3.4 exemplifica a cifra de *Vigenère*. Os números (6,17,8,19,14) correspondem a posição que as letras da chave (GRITO) ocupam no alfabeto.

M = VAMOSATACARAMANHASEMFALTA

Chave = GRITO = (G,R,I,T,O) = (6,17,8,19,14)

Divide-se M em bloco de 05 letras = VAMOS.ATACA.RAMAN.HASEM.FALTA

Após a substituição M’ = BRUHG.GKIVO.XRUTB.NRAXA.LRTMO

FIGURA 3.4 – Cifra de Vigenère (CARVALHO, 2001).

A última forma de criptografia clássica utilizada antes da invenção dos computadores foram os sistemas rotores e que foram largamente utilizados durante a Segunda Guerra Mundial. A máquina ENIGMA, que é a mais famosa utilização desses sistemas, tinha a aparência de uma máquina de escrever comum, no entanto, em seu interior existiam 3 rotores que faziam a codificação. Quando se desejava mandar uma mensagem, digitava-se normalmente no teclado da máquina e em um painel luminoso as letras cifradas correspondentes ficavam visíveis, copiava-se então a mensagem cifrada e quando chegava ao destino digitava-se a mensagem cifrada e lia-se a mensagem original no painel luminoso (SILVA, 2003). A figura 3.5 apresenta um pequeno esquema do funcionamento da máquina ENIGMA.

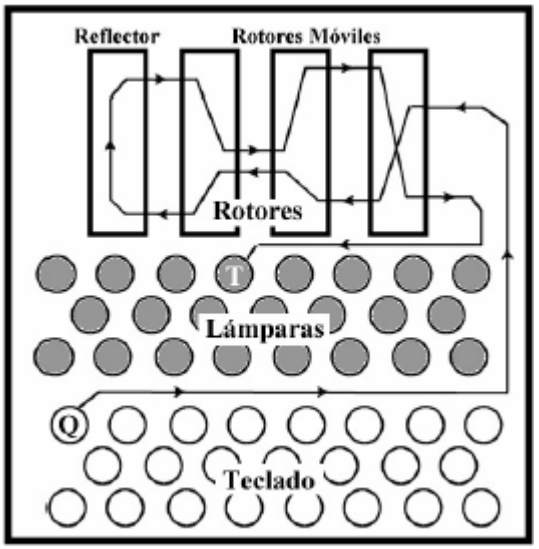


FIGURA 3.5 – Máquina ENIGMA – Diagrama Simbólico (LOPEZ, 2004)

Outras máquinas conhecidas que utilizavam sistema de rotores foram a SIGABA, utilizada pelo exército americano, e as máquinas PURPLE e RED, ambas invenções

japonesas, utilizadas na Segunda Guerra Mundial, e cujas características são secretas e desconhecidas (LOPES, 2004).

3.2.2 Criptografia moderna

Teve início com a invenção dos computadores, cuja invenção possibilitou a evolução dos algoritmos de criptografia, não trabalhando com caracteres e sim diretamente nos bits.

Além da evolução dos algoritmos de criptografia, a invenção dos computadores possibilitou a quebra das antigas técnicas em poucos segundos, devido a alta capacidade de processamento presente nos computadores. A criptografia teve seu grande desenvolvimento em épocas de guerra.

O primeiro algoritmo realmente aplicado em grande escala foi o *Lúcifer*, desenvolvido pela IBM por volta de 1973 e que serviu como base para o DES, o qual, em 1977, tornou-se padrão americano em criptografia (SILVA, 2003) (FIPS46-2, 1993).

3.3 Primitivas Criptográficas

As primitivas criptográficas são estruturas elementares utilizadas para a construção dos criptosistemas e dos protocolos criptográficos. O modelo de classificação mais completo das diversas primitivas criptográficas é o proposto por (MENEZES, 1996) conforme figura 3.6.

Além desta classificação, (MENEZES, 1996) sugere os seguintes critérios para avaliação de tais primitivas:

- **Nível de segurança –** Normalmente é calculado com base na quantidade de operações necessárias para alcançar o seu objetivo;
- **Funcionalidade –** As primitivas precisam ser combinadas para atender aos objetivos de segurança da informação. Quais delas atendem melhor estas características é determinado pelas propriedades básicas desta primitiva;
- **Método de Operação –** A utilização das primitivas depende dos modos de operação possíveis;
- **Desempenho –** Refere-se à eficiência com que a primitiva executa sua função;

- **Facilidade de Implementação –** Refere-se à complexidade exigida pela primitiva para sua implementação tanto em hardware quanto em software.

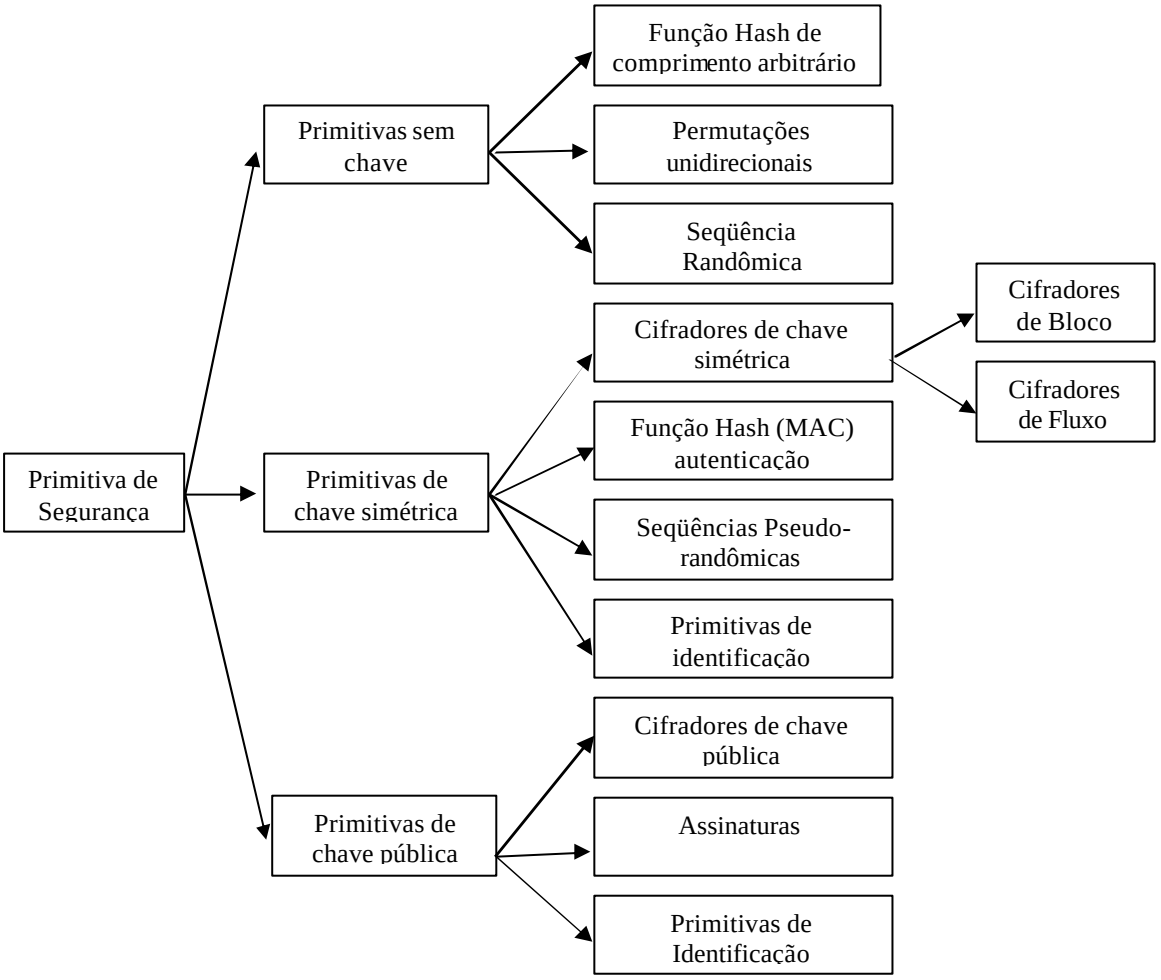


FIGURA 3.6 – Classificação das Primitivas Criptográficas (MENEZES, 1996).

3.4 Sistemas Critográficos ou Criptossistemas

Um sistema criptográfico é como se fosse uma quíntupla (M, C, K, E, D), onde M é o conjunto de todas as mensagens não cifradas, C é o conjunto de todas as mensagens cifradas, K é o conjunto de chaves possíveis de se utilizar no sistema, E é o conjunto de funções ou primitivas para cifrar M e obter C, e D é o conjunto de funções ou primitivas para decifrar C e obter novamente M (LOPEZ, 2004).

A notação desse sistema é Dk(Ek(m)) = m.

Existem duas correntes para a definição dos sistemas criptográficos. Para uma corrente, um sistema criptográfico ou criptossistema é representado pelos algoritmos e pelo conjunto de todos os possíveis textos puros, textos cifrados e chaves de cifragem. Já para uma segunda corrente, um criptossistema é um conjunto de primitivas utilizadas para fornecer serviços de segurança às informações (SILVA, 2003).

3.5 Classificação dos Sistemas Criptográficos ou Criptossistemas

Os criptossistemas dividem-se basicamente em dois tipos fundamentais:

- **Criptossistemas de chave privada ou simétrica –** utilizam a mesma chave tanto para o processo de cifragem quanto no de decifragem. Tem como ponto positivo a grande velocidade de processamento, e como ponto negativo o fato de exigir um meio seguro para a troca da chave (SILVA, 2003). A figura 3.7 traz um modelo de funcionamento do modelo simétrico. Como exemplo, no DES ocorre o problema de distribuição de chaves. A chave tem que ser enviada para todos os usuários autorizados antes que as mensagens possam ser trocadas. Essa ação resulta num atraso e possibilita que a chave chegue a pessoas não autorizadas (MORENO, 2005).

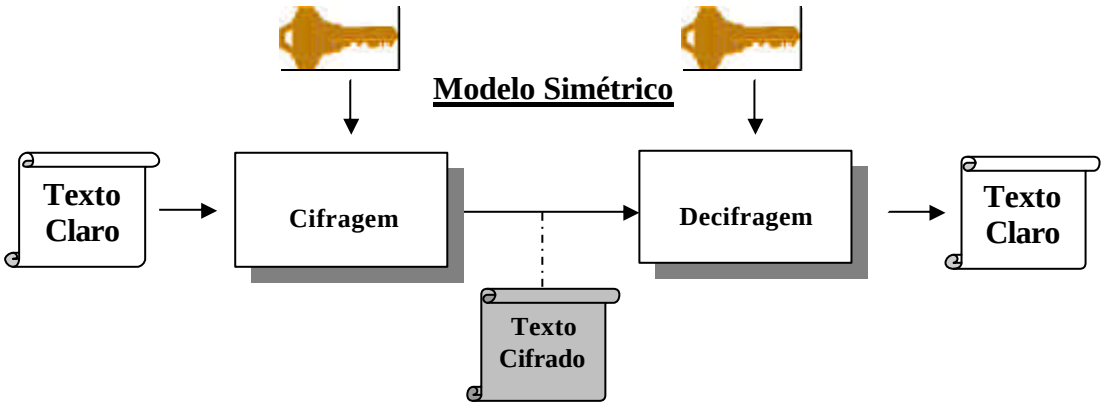


FIGURA 3.7 – Funcionamento do modelo simétrico de criptografia (PEREIRA, 2004).

- **Criptossistema de chave pública ou assimétrica –** utilizam um par de chaves, sendo uma delas pública e a outra privada. A criptografia de chaves públicas foi inventada em 1976 por Whitfield Diffie e Marin Hellman, a fim de resolver o problema de distribuição de chaves (LOPEZ, 2004) (MORENO, 2005). A figura 3.8 traz um

modelo de funcionamento assimétrico. Normalmente neste tipo, cifra-se com a chave pública e deficra-se com a chave privada.

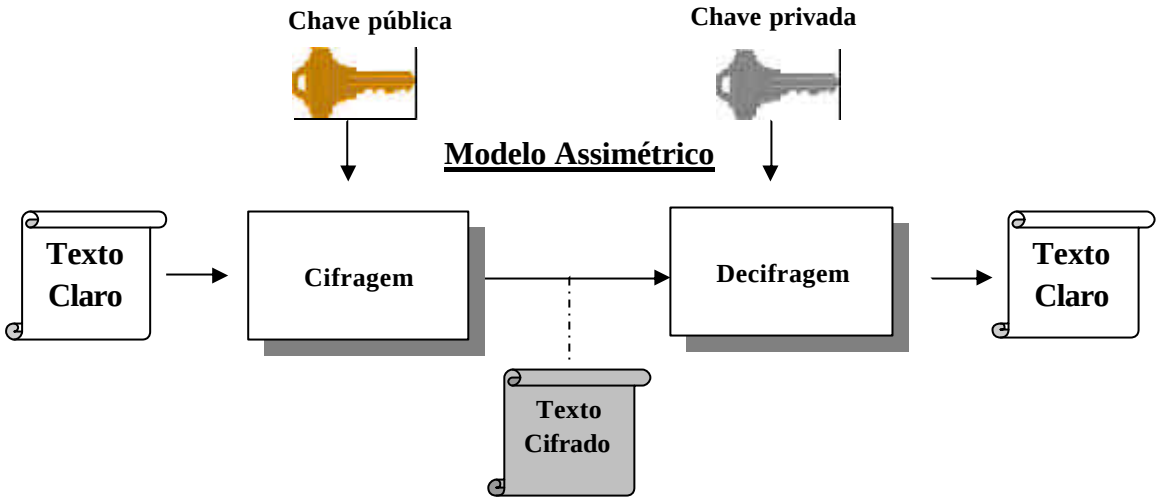


FIGURA 3.8 – Funcionamento do modelo assimétrico de criptografia (PEREIRA, 2004).

No entanto, as chaves são intercambiáveis, podendo-se cifrar com a chave privada e decifrar com a chave pública. Esta última técnica serve como "assinatura digital". Normalmente a chave possui mais de mil bits, ocasionando uma menor velocidade de processamento se comparado aos algoritmos de chave simétrica. A figura 3.9 mostra o modelo de geração de uma assinatura digital.

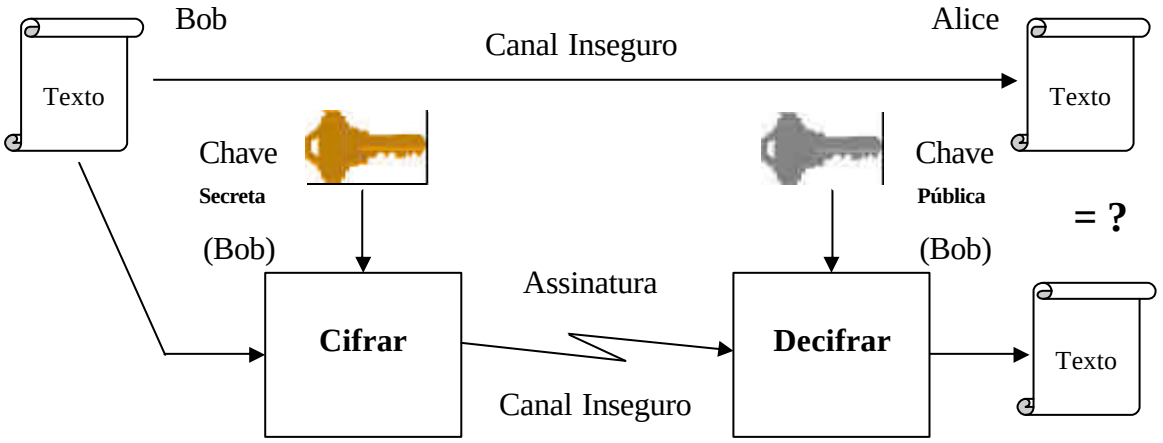


FIGURA 3.9 – Geração de assinatura digital de um documento (PEREIRA, 2004).

A assinatura digital é uma forma de identificação do emissor de determinadas informações. No processo normal da criptografia, o emissor criptografa as informações com a chave pública e envia ao destinatário, que com a sua chave privada decifra a informação. No

caso da assinatura digital, quem possui a chave privada, a usa para criptografar as informações e envia-las ao destinatário. O destinatário, por sua vez, utiliza a chave pública para decifrar as informações e ter acesso ao seu conteúdo. Isso possibilita, por parte do receptor, a identificação de quem criptografou essas informações, já que a chave utilizada para realizar a criptografia dessas informações é privada, sendo exclusivamente de uma pessoa.

Existe ainda a possibilidade de um modelo híbrido, que pode ser desenvolvido aproveitando-se as vantagens de cada tipo de algoritmo. O algoritmo simétrico, por ser mais rápido, é utilizado no ciframento da mensagem em si, enquanto o algoritmo assimétrico, embora lento, permite implementar a distribuição de chaves e é utilizado em aplicações de assinatura digital (MORENO, 2005). Como exemplo de modelos híbridos tem-se o IPSec (*Internet Protocol Security*), que é o padrão de protocolo criptográfico desenvolvido para o IPv6 (*Internet Protocol version 6*), o SSL (*Secure Socket Layer*) e TLS (*Transport Layer Security*), que oferecem suporte de segurança criptográfica para os protocolos HTTP (*Hyper Text Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*) e Telnet (MORENO, 2005).

3.6 Principais modos de operações dos criptossistemas simétricos

Os modos de operações referem-se à forma com que os algoritmos tratam as mensagens. Podem-se classificar os algoritmos de criptografia por meio do tratamento dado às informações que serão processadas, sendo em algoritmos de bloco e algoritmos de fluxo.

- Algoritmo de blocos: A cifra de blocos opera sobre blocos de dados. Esse tipo de algoritmo trata as mensagens que excedem o tamanho do bloco predefinido. O texto antes de ser cifrado é dividido em blocos que variam normalmente de 8 a 16 bytes. Quando o texto não completa o número de bytes de um bloco, este é preenchido com dados conhecidos até completar o número de bytes do bloco. A forma mais comum de preenchimento é determinar o número de bytes que deve ser preenchido e utilizar esse valor para preencher o bloco (MORENO, 2005).

Um problema na cifra de bloco é que se o mesmo bloco de texto simples aparecer mais de uma vez, a cifra gerada será a mesma, facilitando o ataque ao texto cifrado. Para resolver esse problema são utilizados os modos de realimentação, listados abaixo:

- **Modo ECB (*Eletronic Code Block*) –** cada bloco da mensagem é processado independentemente gerando uma seqüência de blocos cifrados. Se os blocos cifrados forem reordenados e se for efetuada a decifragem, obter-se-á a mensagem também reordenada. Este modo oferece menor nível de segurança e, tendo em vista que os blocos podem ser repetidos, pois blocos de texto idênticos geram blocos cifrados também idênticos, ou trocados, um atacante pode modificar a mensagem original apenas inserindo duplicatas dos blocos cifrados ou simplesmente alterando sua ordem (SILVA, 2003).
- **CBC (*Cipher Block Chaining Mode*) –** este modo incorpora um mecanismo de retroalimentação da cifra por blocos. A codificação dos blocos anteriores condiciona a codificação de bloco atual, para o que será impossível substituir um bloco individual na mesma mensagem cifrada. Isto é possível através de uma operação XOR entre o bloco da mensagem que será codificada e o último criptograma obtido (LOPEZ, 2004).
- **CFB (*Cipher FeedBack*) –** Neste modo também ocorre a realimentação através de uma operação XOR, mas agora somente sobre o grupo, e não mais sobre um bloco.
- **OFB (*Output FeedBack*) –** Similar ao CFB, incorpora mecanismos para manter a sincronia entre os geradores de seqüência origem e destino.

- Algoritmo de fluxo: Os algoritmos de fluxo criptografam a mensagem bit a bit, em fluxo contínuo, sem esperar que se tenha um bloco completo de bits. É conhecido como criptografia em stream de dados, sendo realizada com uma operação XOR entre bit de dado e o bit gerado pela chave (MORENO, 2005).

Comparando esses dois modos de operação dos criptossistemas simétricos, os algoritmos de blocos levam vantagem pois processam os dados como um conjunto de bits, sendo mais rápidos e seguros para a comunicação digital. Ainda, como vantagem, tem-se que os blocos podem ser codificados fora de ordem, o que é bom para o acesso aleatório, além de ser resistente a erros, uma vez que um bloco não depende de outro. Entretanto, como desvantagem, se a mensagem possuir padrões repetitivos nos blocos, o texto cifrado também o apresentará, o que facilitará o serviço do criptoanalista. Também existe a desvantagem da possibilidade da modificação da mensagem original substituindo os blocos por outros modificados (MORENO, 2005).

3.7 Algoritmo de Criptografia DES

O algoritmo de criptografia DES é composto de operações simples, como permutações, substituições, XOR e deslocamentos. O DES criptografa informações por meio do processo de cifra de bloco com tamanho de 64 bits e retorna blocos de texto cifrado do mesmo tamanho usando uma chave de 56 bits.

O DES possui 16 iterações e em cada iteração utiliza uma subchave diferente derivada da chave original. Para cifrar um determinado bloco de texto (64 bits), é necessário utilizar as subchaves na ordem crescente, isto é, na iteração 1 do processamento principal usa-se a subchave 1, na iteração 2, a subchave 2, na 3, a subchave 3 e, assim por diante. Para decifrar um bloco de texto, as subchaves são aplicadas na ordem inversa ou decrescente. Neste contexto: para decifrar a iteração 1, usa-se a subchave 16, na iteração 2, se utiliza a subchave 15, na iteração 3, a subchave 14 e assim por diante.

A figura 3.10 mostra a representação do processamento principal do algoritmo DES, onde é possível verificar que o processamento principal é dividido em três etapas executadas em seqüência. A primeira etapa é a PI (permutação inicial), a segunda corresponde às iterações e a terceira e última etapa corresponde à PF (permutação final). Maiores detalhes das operações do algoritmo e da implementação podem ser obtidas em (MORENO, 2005).

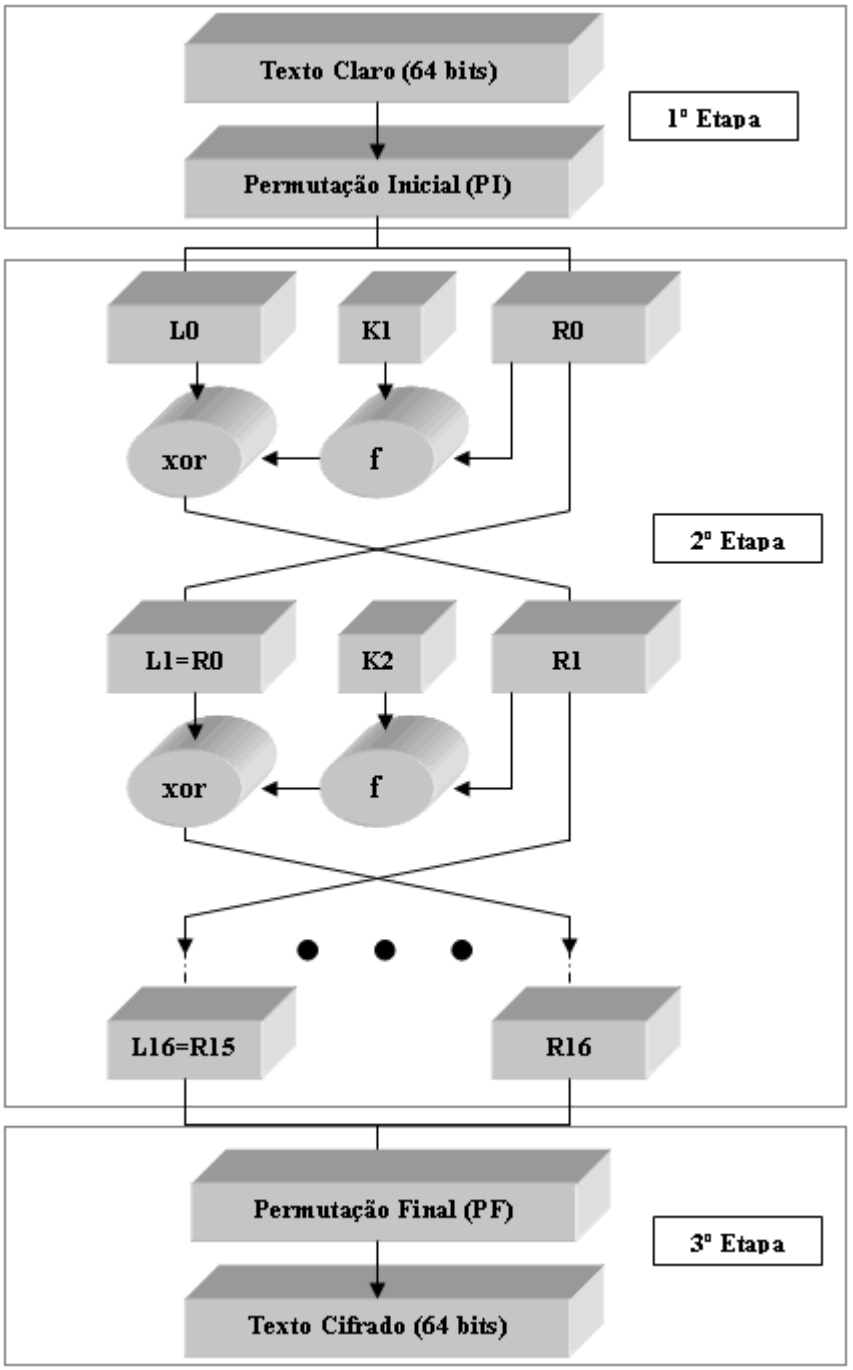


FIGURA 3.10 – Operações do Algoritmo DES (MORENO, 2005)

3.8 Algoritmo de Criptografia AES

Com a "evolução das máquinas", uma chave de 64 bits, que até então era o padrão de segurança, não poderia mais garantir a segurança necessária por muito tempo. Em 1997, o NIST (*National Institute for Standards and Technology*) iniciou um processo de escolha para substituir o algoritmo DES, que até então foi o padrão de criptografia. O NIST especificou

que os candidatos deveriam operar com tamanho de chave e bloco variável com tamanho máximo de 128 bits.

Também especificou como características desse algoritmo segurança forte, projeto simples, desempenho e não ser patenteado.

Na primeira etapa, o NIST recebeu vinte e uma submissões, das quais quinze atendiam as exigências. Esses algoritmos foram testados e avaliados pela comunidade científica e empresas ligadas a segurança. Para a segunda rodada foram classificados apenas cinco algoritmos, MARS, RC6, *Rijndael*, *Serpent* e *TwoFish*, os quais satisfaziam as principais condições, como cifrar e decifrar blocos de 128 bits, trabalhar com chaves de 128 a 256 bits e ser mais rápido que o 3DES (AESWINNER, 2000).

Como todos os requisitos básicos foram atendidos pelos concorrentes, a escolha foi tomada com base em certas características, como segurança, eficiência em hardware, e software, flexibilidade de implementação e modos de operação.

Em 2 de outubro de 2000 o NIST anunciava a vitória do algoritmo *Rijndael* como vencedor do concurso e anunciava oficialmente a sua adoção como sendo o novo AES. A palavra *Rijndael*, empregada para a denominação AES, é um acrônimo formado pelos nomes de seus autores, os belgas Joan Daemen e Vincent Rijmen (LOPEZ, 2004).

AES é um sistema de cifra por blocos, utilizado para manejar longitude de chaves e blocos variáveis, ambas compreendidas entre 128 e 256 bits. Divide-se em duas partes principais: a fase de tratamento da chave e fase de cifragem. Esta última consiste em *n-rodadas* que dependem diretamente do comprimento da chave e do comprimento do bloco. Considerando-se um tamanho de bloco fixo em 128 bits e a chave variando em 128, 192 e 256 têm-se o seguinte número de rodadas, conforme demonstrado na tabela 3.2 (SILVA, 2003) (DAEMEN, 1999).

TABELA 3.2 – Número de rodadas para AES em função dos tamanhos da chave e bloco (LOPEZ, 2004).

N*r*	**N*b* = 4 (128 bits)**	**N*b* = 6 (192 bits)**	**N*b* = 8 (256 bits)**
N*k* = 4 (128 bits)	10	12	14
N*k* = 6 (192 bits)	12	12	14
N*k* = 8 (256 bits)	14	14	14

O componente central do *Rijndael*, onde são efetuadas todas as operações, é uma matriz de bytes de duas dimensões chamada de *State.* O *Rijndael* possui um número fixo de 4 linhas e um número de colunas que varia de acordo com o tamanho do bloco de dados,

podendo ter 4, 6 ou 8 colunas. A figura 3.11 representa essa estrutura e a ordem em que os bytes do bloco são colocados nesta estrutura quando se inicia o processamento.

3.8.1 Estrutura do AES

O AES é um algoritmo de bloco com tamanho de chave variável de 128 a 256 bits, podendo ter tamanhos de blocos e tamanhos de chaves diferentes. Em função do tamanho do bloco e das chaves é determinada a quantidade de rodadas necessárias para cifrar e decifrar. Ele opera com um determinado número de blocos de 32 bits, os quais são ordenados por colunas de 4 bytes, as quais são chamadas de N*b*. Os possíveis valores de N*b* são 4, 6 e 8, equivalentes a blocos de 128, 192 e 256 bits. Desse modo, sempre que for referido N*b*, significa que tem N*b* x 32 bits de tamanho de dados. A chave é agrupada da mesma forma em colunas que o bloco de dados, mas com a sigla N*k*. Com base nos valores que N*b* e N*k* podem assumir é que se determina a quantidade de rodadas a serem executadas, identificada pela sigla N*r*. Na tabela 3.2 pode-se verificar as possíveis combinações e o número de rodadas a serem executadas (PEREIRA, 2004).

3.8.2 Funções de cifragem e decifragem

As operações que cada bloco está sujeito durante o processo de cifragem são:

- **SubByte –** os bytes de cada bloco são substituídos linearmente por seus equivalentes em uma tabela de substituição (S-BOX) (DAEMEN, 1999) (FIGURA 3.11).

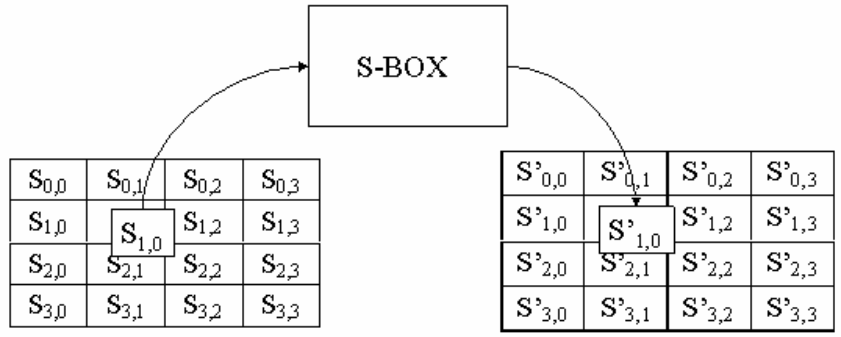


FIGURA 3.11 – Função SubByte (PEREIRA, 2004).

- **ShiftRow (Deslocamento de Linhas) –** nesta etapa, os bytes das últimas três linhas são submetidos a uma operação de rotação em diferentes valores que dependem do tamanho do bloco (DAEMEN, 1999) (FIGURA 3.12).

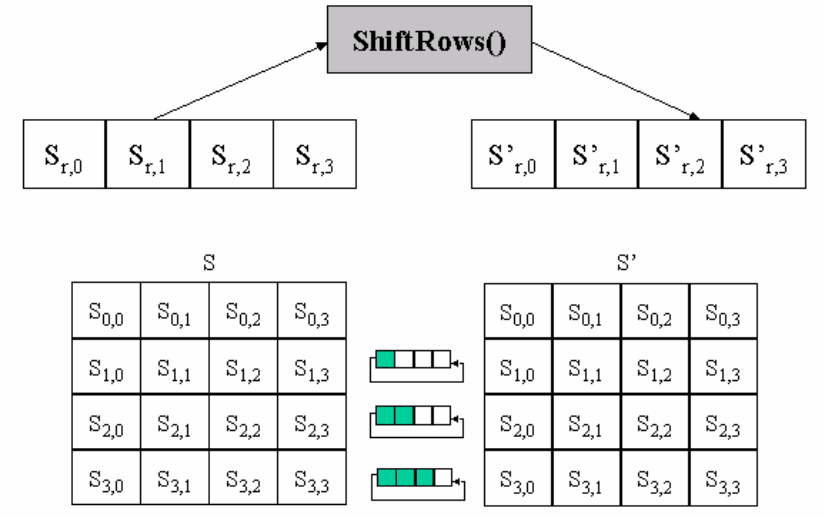


FIGURA 3.12 – Função ShiftRow (PEREIRA0 2004).

- **MixColumn –** Cada grupo de quatro bytes é sujeitado a uma multiplicação modular. Isto proporciona a cada byte do grupo influenciar em todos os outros bytes (DAEMEN, 1999) (FIGURA 3.13).

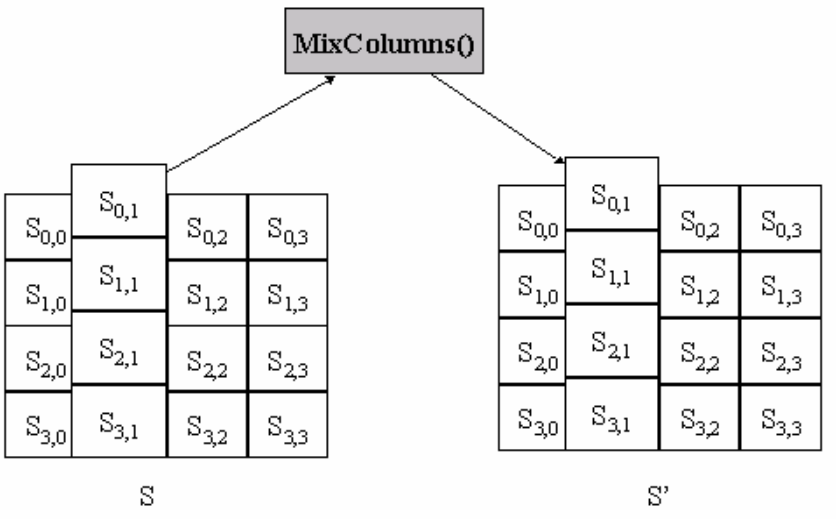


FIGURA 3.13 – Multiplicação MixColum (PEREIRA, 2004).

- **AddRoundKey (Adição da Chave de Rodada) –** A função AddRoundKey efetua uma operação XOR entre os dados do *state* e a *RoundKey*, o qual possui o mesmo tamanho do bloco (DAEMEN, 1999) (FIGURA 3.14).

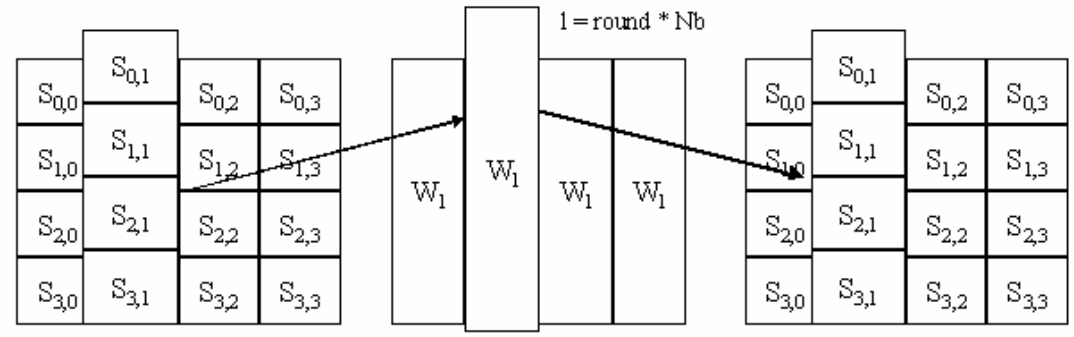


FIGURA 3.14 – Operação de AddAroundKey (PEREIRA, 2004).

Para criptografar um bloco de dados com o *Rijndael*, é executado primeiro um passo *AddRoundKey* (fazendo uma lógica XOR entre uma sub-chave e o bloco), depois são executadas as rodadas regulares, compostas pelas funções descritas acima (*SubByte, ShiftRow e MixColumn*) e ao final de cada rodada regular é executada novamente o passo *AddRoundKey* (SILVA, 2003).

Cada rodada regular envolve quatro passos, onde o primeiro passo é *SubByte*, onde cada byte do bloco é trocado por seu substituto em uma S-BOX, logo em seguida ele executa o *ShiftRow*, onde as linhas da matriz que representa o *state* sofrem um rotacionamento cíclico à esquerda, exceto a primeira linha. A última rodada é idêntica às rodadas regulares, excluindo-se a função *MixColumn*.

Executar o processo inverso, descriptografar, consiste em executar de maneira inversa o processo de criptografar, executando de trás pra frente o processo acima descrito; porém as funções *SubByte*, *ShitfRow* e *MixColumn* necessitam ser as suas inversas matemáticas para realizar o processo de decifrar (PEREIRA, 2004).

A figura 3.15 abaixo representa os processos de cifrar e decifrar do algoritmo AES.

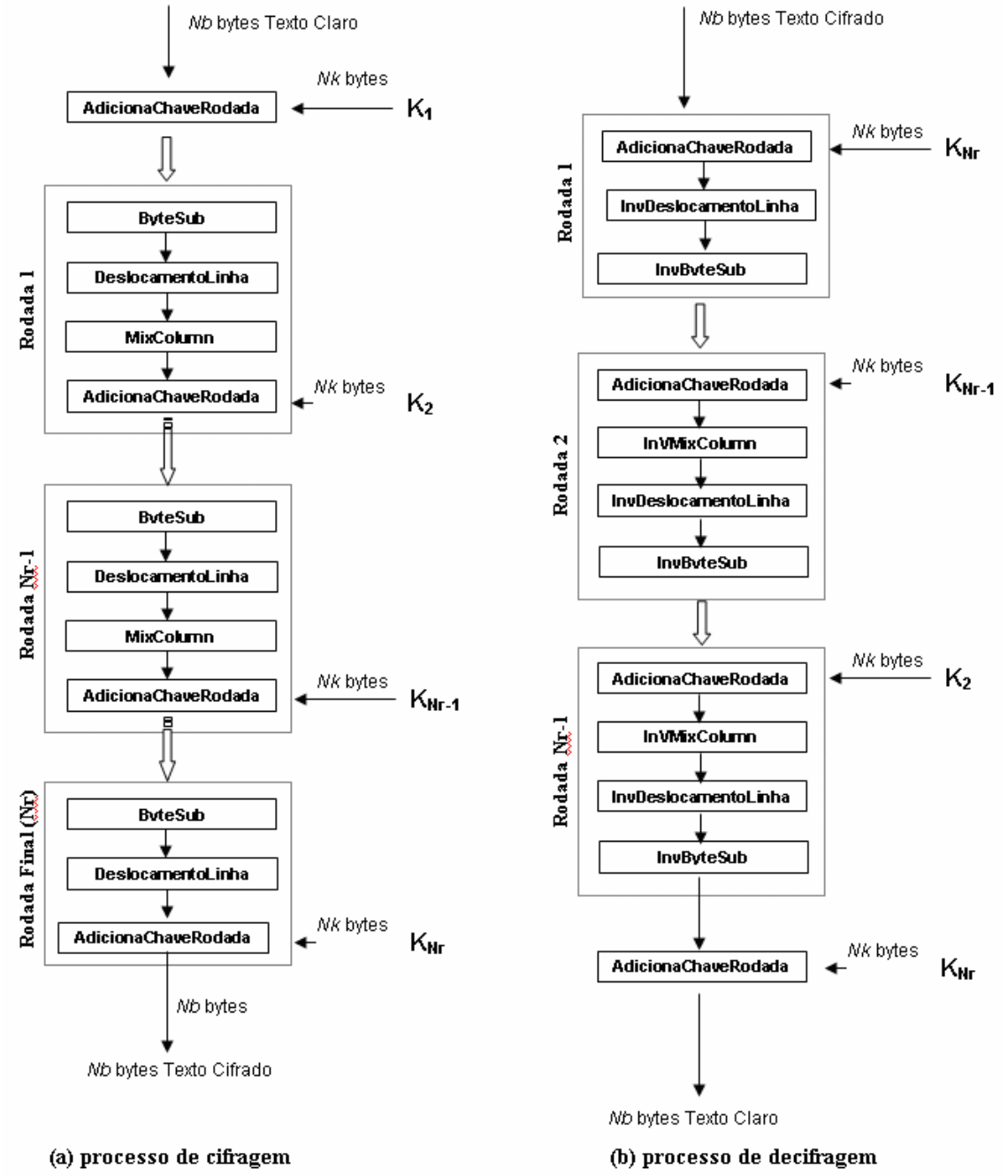


FIGURA 3.15 – Fluxo de dados no algoritmo AES (PEREIRA, 2004).

3.9 Justificativa de escolha dos algoritmos DES e AES

Os Algoritmos adotados neste trabalho são o DES e o AES, pois são algoritmos flexíveis, seguros e com desempenho satisfatório.

Como características, pode-se destacar:

- **Segurança Forte –** Cada um em seu tempo, o DES e o AES foram projetados para suportar ataques futuros;
- **Projeto Simples –** possuem fácil criptoanálise;
- **Bom desempenho –** seu desempenho é satisfatório em várias plataformas;
- **Sem patente –** é de domínio público.

Influenciado pela tendência do software livre e pelo fato de ser um trabalho acadêmico, a escolha na utilização desses algoritmos de criptografia neste trabalho foi motivada por se tratar de um algoritmo que é padrão internacional e referência em criptografia, além da sua livre utilização (AESWINNER, 2000) e (FIPS46-2, 1993).

Assim, este capítulo traz uma visão geral da segurança da informação, especificamente a utilização da criptografia, mostrando sua origem, tipos e funcionamento e traz uma visão geral dos algoritmos de criptografia DES e AES, que foram adotados como padrão de criptografia pelo NIST, bem como seu funcionamento e estrutura.

Este capítulo tem fundamental importância nessa dissertação por tratar de segurança de dados e criptografia. Como definido no capitulo 1, este projeto se propõe a desenvolver um processador de rede com segurança, que será oferecida através de criptografia, especificamente utilizando-se os algoritmos DES e AES.

O próximo capítulo trata da proposta do ToPSeC, um processador que oferece suporte as topologias Anel Unidirecional e Árvore Binária. Também trata do impacto causado pela inclusão de primitivas de segurança com criptografia. É mostrando sua arquitetura, conjunto de instruções, seu estágio atual e seus resultados a cada implementação.

Capítulo 4

ToPSeC – Um processador com suporte a mudança de topologia

4.1 Processador de Rede para mudança de topologias

O objetivo deste capítulo é apresentar o ToPSeC, um processador com suporte a algumas topologias de rede e com implementações de criptografia para prover segurança.

O processador proposto neste trabalho é baseado no NPSoC (PRADO, 2004), que por sua vez, teve como base os processadores de rede RCNP (FREITAS, 2001) e no R2NP (FREITAS, 2002).

Nesse novo processador foram feitas algumas alterações do projeto original do NPSoC. As instruções básicas foram mantidas para uma eventual alteração ou inclusão de uma nova funcionalidade ao mesmo, adicionalmente novas instruções foram criadas e algumas já existentes foram adaptadas para dar suporte às implementações que são apresentadas neste capítulo.

O ToPSeC é um processador especial simples que tem por finalidade a transmissão de informações a outros processadores ToPSeC de maneira segura, que são dispostos nas topologias propostas neste trabalho (Anel Unidirecional e Árvore Binária).

Neste processador há uma dependência da UC para realizar as operações de controle, como busca, decodificação e execução. Neste caso, foi utilizada a metodologia de Máquina de

Estado Finito (FSM), onde seus estados são bem definidos e respeitam o ciclo de execução dos computadores Von Neumann.

Neste processador, os ciclos se executam da seguinte maneira:

1 º. É feita a busca da primeira instrução na memória e soma-se 1 ao PC;

2 º. Decodifica a instrução;

3 º. Se a instrução trouxer informações, estas são armazenadas em registradores internos;

4 º. Execução da instrução pela ULA;

Armazena o resultado nos registradores internos e "seta" o processador ao estado de busca, dando continuidade ao ciclo.

Este capítulo apresenta as implementações utilizadas no desenvolvimento do ToPSeC, bem como a arquitetura proposta para determinada implementação, a descrição do seu comportamento, as definições estabelecidas, um diagrama da topologia, as simulações realizadas com a ferramenta Xilinx Fondation Series 3.1i comprovando o funcionamento da mesma e as estatísticas de implementação e desempenho em FPGA.

4.2 ToPSeC – Topologia Anel Unidirecional

A primeira topologia estudada é a Anel Unidirecional. Nesta topologia, cada nó é um ToPSeC, formando um círculo por onde os dados trafegam em apenas uma direção.

O seu comportamento se resume em um dos nós enviar um dado a um outro nó qualquer que estaria nessa rede. Nessa topologia, os nós estariam conectados um a um, como um barramento circular. Esse dado trafegaria na rede em apenas uma direção. No caminho até o destino da informação, haveria vários outros nós, que devem verificar se são o destino dessa informação e, em caso negativo, repassá-la ao próximo nó. Esse comportamento faz com que a informação enviada chegue ao seu destino.

Para o funcionamento dessa topologia, não importa o número de nós que essa rede possa ter, pois o tratamento da informação que chega é feito no nó local, mas por definição de arquitetura, foi estipulado 15 nós.

Para que a implementação dessa topologia fosse possível da maneira que foi descrita, algumas definições foram estabelecidas:

- Não existe *broadcast;*
- O endereço de cada nó já é sabido e é declarado no início da rotina;
- O número de nós é fixo em 15 nós, assim o endereço de cada nó se faz em 4 bits;
- A informação trafegada possui 16 bits.

4.2.1 Arquitetura do ToPSeC – Topologia Anel Unidirecional

Baseada na arquitetura do NPSoC, mas com algumas modificações na arquitetura, devido às alterações propostas, a arquitetura do ToPSeC ficou definida da seguinte forma:

- 4 portas de entrada, nomeadas de E1 a E4 de 16 bits;
- 4 portas de saída, nomeadas de S1 a S4 de 16 bits;
- 4 registradores de propósito geral de 8 bits (A, B, C e D);
- 1 registrador específico para roteamento de 4 bits (ID);
- 1 registrador específico para roteamento de 8 bits (OD);
- Unidade Lógica e Aritmética (ULA);
- Contador de Programa (PC);
- Unidade de Controle (UC);
- Registrador de Instruções (RI).

Com relação à arquitetura proposta e levando em conta que este projeto tomou por base um trabalho já existente, nem todos os componentes correntes na arquitetura são utilizados nessa versão. A decisão de manter na arquitetura componentes como as portas de entrada E2, E3 e E4, as portas de saída S2, S3 e S4 e os registradores de propósito geral B, C e D foi motivada visando oferecer maior flexibilidade para futuras versões.

A figura 4.1 traz uma ilustração da arquitetura do ToPSeC. Nela é possível verificar as principais diferenças de arquitetura em relação ao NPSoC mostrada na figura 4.1.

As principais diferenças entre as duas arquiteturas são a redução do número de portas de saída e a diminuição do seu tamanho, a inclusão de 4 portas de entrada, a diminuição dos registradores de propósito geral em número e tamanho.

As demais alterações do projeto original nesta versão são a inclusão de registradores específicos para roteamento, como ID (Identificador) e OD (Origem/Destino), e a criação de novas instruções, para oferecer suporte à topologia Anel Unidirecional.

O ToPSeC, além das alterações sofridas no NPSoC, continua oferecendo suporte a operações simples, como AND, OR, ADD, SUB, etc.

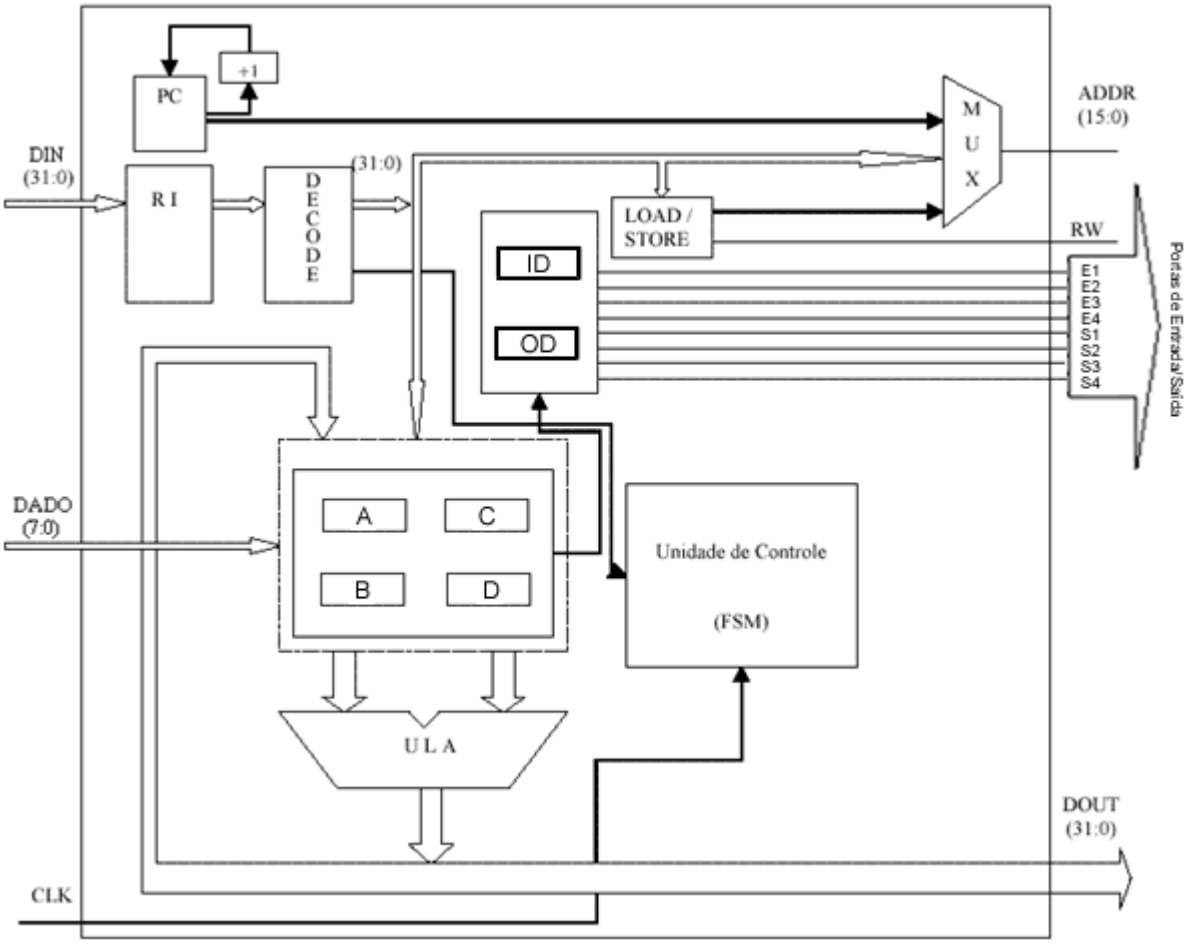


FIGURA 4.1. Arquitetura do ToPSeC – Versão Inicial

Um último ponto a ser observado sobre a arquitetura desse processador são as portas de entrada e saída e as informações que trafegam nelas.

Cada uma dessas portas possui o tamanho de 16 bits. Esses bits são divididos conforme a figura 4.2.

Quando uma informação é enviada de um nó a outro, a informação que chega através da porta de entrada é formada pelo endereço do nó que originou a informação aquele nó, o destino da informação e o seu conteúdo.

FIGURA 4.2. Formato da mensagem – Versão Inicial

O tamanho de 4 bits destinado ao endereçamento da origem e destino foi definido visando o limite de até 16 nós. Os 8 bits restantes são referentes ao dado que é enviado através da rede.

4.2.2 Conjunto de instruções (ISA) – Topologia Anel Unidirecional

Nesta versão inicial, no qual foi implementada a topologia Anel Unidirecional, o conjunto de instruções do Processador de Rede ToPSeC segue o modelo RISC, onde foi definido um reduzido conjunto de instruções, contendo apenas 14 instruções que dão suporte a essa topologia.

A tabela 4.1 traz as instruções utilizadas nesse processador, classificando-as quanto a sua finalidade e destacando aquelas que foram criadas e alteradas para esse projeto.

TABELA 4.1 – Instruções utilizadas no ToPSeC – Versão Inicial

Lógicas e Aritméticas			
AND	Realiza a operação E entre três registros	**SUB**	Subtrai três registros
OR	Realiza a operação OU entre três reistros	**INC**	Incrementa um registro em +1
XOR	Idem OR, porém nega o resultado	**DEC**	Decrementa um registro em -1
ADD	Soma três registros	**NEG**	Nega um registro
Movimentação			
ENT	Armazena a origem, o destino e o dado	**LID**	Carrega o endereço e a posição do nó atual
Desvio			
JUMP	Desvio incondicional	**JDEST**	Desvia se o dado é para o nó atual
Envio			
SEND	Envia a informação através da porta de saída ao nó verificado	**SAD**	Envia a informação ao próximo nó

Das 14 instruções apresentadas na tabela acima, três novas instruções foram incluídas para dar suporte à topologia anel unidirecional, sendo uma de movimentação (LID), uma de desvio (JDEST) e uma de envio (SAD). Das instruções de movimentação, a instrução ENT foi modificada do projeto original para atender as necessidades no novo processador na topologia anel unidirecional, pois tinha um comportamento diferente do necessário para o funcionamento dessa topologia.

4.2.3 Programa Teste e Funcionamento – Topologia Anel Unidirecional

Para a versão inicial, a comparação utilizada foi o programa implementado para a topologia Anel Unidirecional no NPSoC. A principal alteração é quanto ao seu funcionamento.

No NPSoC, a implementação dessa topologia foi feita pensando em apenas um processador de rede no centro de uma rede em anel, gerenciando e enviando a informação para o seu destino correto em apenas um sentido, como se fosse um dispositivo concentrador de rede.

Na versão inicial do ToPSeC, a implementação da topologia Anel Unidirecional segue a idéia de que cada nó seja independente, como se cada um fosse um processador de rede distinto, executando o mesmo algoritmo para a topologia Anel Unidirecional. Quando o pacote chegar, haverá uma verificação se o dado recebido é para o nó atual ou não, tomando a decisão de enviá-lo ao próximo nó, em caso negativo, ou simplesmente iniciar o tratamento da informação, em caso afirmativo. A figura 4.3 mostra a diferença de funcionamento da topologia anel unidirecional no NPSoC e no ToPSeC.

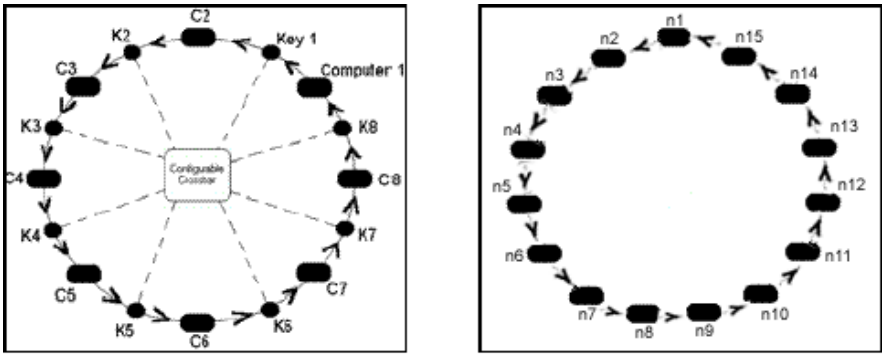


FIGURA 4.3 – Funcionamento da Topologia Anel Unidirecional no NPSoC e ToPSeC

Analisando a figura 4.3, fica claro que o anel unidirecional do NPSoC tem um comportamento de enviar a todos os nós da rede o dado, possuindo inclusive em sua arquitetura 8 portas de saída, numeradas de B1 a B8 (figura 4.1), onde em cada porta de saída, estaria conectado um nó da rede. Todos os nós receberiam a informação, e ficaria a cargo de cada um analisar se o dado é para ele ou não. Seu funcionamento é semelhante ao de um *hub*. Isso se reflete no código *Assembly* (tabela 4.2 a), onde é possível notar, na arquitetura do NPSoC, que o registrador B funciona como um contador, que vai de 8 até 1, dando o destino das informações enviadas a cada salto.

Já no Anel Unidirecional do ToPSeC, o número de nós que essa topologia poderia ter nessa implementação, não se limita ao número de portas de saída que o processador de rede tiver. A única porta de saída que será utilizada é a S1, e cada nó, ao receber a informação de outro nó contendo o dado e seu destino, verifica se este pertence a ele ou não. Em caso afirmativo, o programa termina, pois a partir deste ponto, seria o tratamento com a informação recebida, e este não é o foco deste trabalho. Em caso negativo, é enviado para o próximo nó que fará o mesmo tratamento, até que a informação chegue ao seu destino (tabela 4.2 b).

TABELA 4.2 – Comparação das instruções *Assembly* do NPSoC e do ToPSeC para a topologia Anel Unidirecional

a)

	NPSoC	Ciclos
0	LDI A,07	5
1	LDI B,01	5
2	PUT B	4
3	LDI B,08	5
4	SEC	4
5	FCX	4
6	PUT B	4
7	DCR B	4 * 8
8	SEC	4 * 7
9	FCX	4 * 7
10	DCR A	4 * 7
11	JMZ 000009	5 * 8
12	HTL	4
Total		191

b)

	ToPSeC	Ciclos
0	LID 03	2
1	ENT	2
2	JDEST 4	2
3	SAD	2
4	HALT	2
Total		10

A principal diferença que se observa na tabela 4.2 a) e b), além da grande diferença de ciclos, é o comportamento do processador na topologia Anel Unidirecional.

Pelo fato do diferente funcionamento dos dois processadores na mesma topologia é que se têm programas tão diferente e com ciclos tão diferentes. No entendimento desse

projeto, essa nova implementação é melhor e mais eficiente pois torna cada nó independente, não limita a quantidades de nós à quantidade de portas de entrada e saída e tem uma execução mais rápida, pois na pior situação, onde o nó 1 enviaria uma informação ao nó 15, passando por todos os outros nós, o número de ciclos seria de 150 contra 191 ciclos do NPSoC.

4.2.4 Simulações da Topologia Anel Unidirecional

Neste tópico a simulação é referente ao funcionamento do programa que faz o envio da informação através de uma rede cuja topologia seja anel unidirecional, conforme figura 4.4.

O seu funcionamento tem início com o armazenamento do número do nó corrente através de DIN (1) no registrador ID (2) através da instrução LID.

Após o nó ter sido identificado, as informações que vêm pela rede, através da porta de entrada S1, são armazenadas em A e OD (3), através da instrução ENT, que armazenam respectivamente nesta rotina o dado e o seu destino.

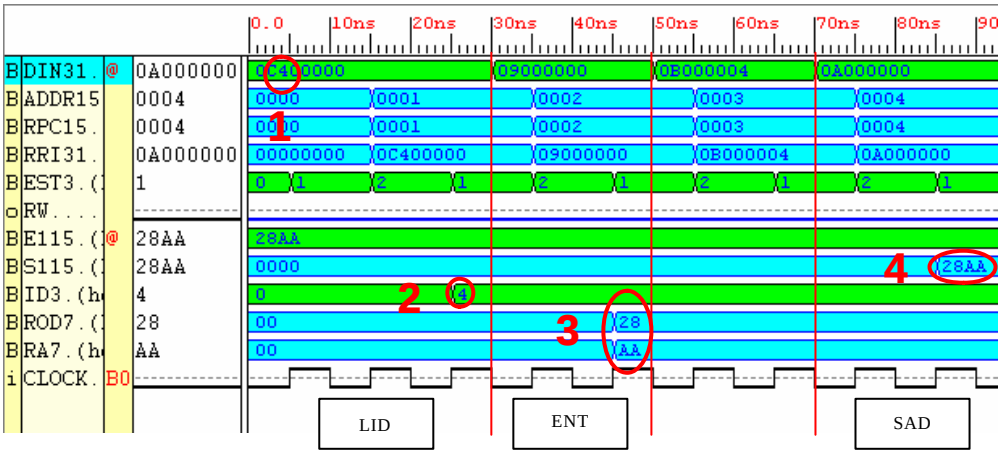


FIGURA 4.4 – Simulação da topologia Anel Unidirecional

Após o nó ter sido identificado e a informação armazenada, é feita uma verificação, identificando se o dado é para o nó atual, o que é feito com a instrução JDEST. Em caso afirmativo, o programa é finalizado. Em caso negativo, o dado e o seu destino são enviados para o próximo nó, através da porta de saída S1 (4), através da instrução SAD.

A simulação mostrada na figura 4.4 corresponde às instruções LID, ENT e SAD, descritas em VHDL na seguinte estrutura:

```
WHEN INSTR_LID =>
      ID <= RI(23 DOWNTO 20);
      EST <= BUSCA;

WHEN INSTR_ENT =>
      A <= E1(7 DOWNTO 0);
      OD(7 DOWNTO 0) <= E1(15 DOWNTO 8);
      EST <= BUSCA;

WHEN INSTR_SAD =>
      S1(15 DOWNTO 0) <= OD(7 DOWNTO 0) & A;
      EST <= BUSCA;
```

O próximo nó terá o mesmo comportamento, até que a comparação do destino for igual ao identificador do nó corrente. A simulação da figura 4.4 mostra o comportamento caso a informação não seja para o nó atual.

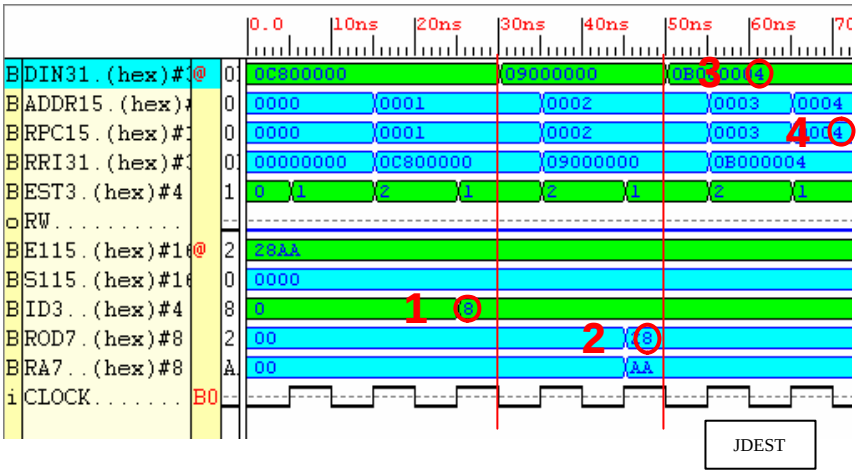


FIGURA 4.5 – Simulação da instrução JDEST

A simulação mostrada na figura 4.5 mostra a situação caso a informação for para o nó atual. Após armazenar o identificador do nó atual no registrador ID (1) e armazenar o destino que vem através da porta de entrada S1 no registrador OD (2), é feito um teste para verificar se o destino desse dado é para esse nó atual. Esse teste é feito pela instrução JDEST, que aponta para onde o programa *assembly* deverá desviar (3) caso a verificação entre o identificador do nó atual (1) e o destino (2) sejam iguais, e neste caso, o desvio no programa é feito (4) para o final das instruções para o programa ser finalizado.

A simulação mostrada na figura 4.5 corresponde à instrução JDEST, descrita em VHDL na seguinte estrutura:

```
WHEN INSTR_JDEST =>
      IF ID(3 DOWNTO 0) = OD(3 DOWNTO 0) THEN
      PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
      RW <='0';
```

```
END IF;
EST <= BUSCA;
```

4.2.5 Estatísticas de Implementação e Desempenho – Topologia Anel Unidirecional

Após a finalização das simulações da versão 1, o ToPSeC foi implementado em dois modelos de FPGAs para que fosse feito um comparativo de implementação e desempenho. A tabela 4.3 traz as estatísticas de dois modelos que foram implementados através da ferramenta Xilinx 3.1 (BOURT, 1998).

TABELA 4.3 – Estatísticas com parâmetros espaciais e temporais

Spartan2 2S50FG256										
Slices (CLBs)			**Flip-Flop**			**Luts de 4 entradas**			**Tempo de Propagação**	
Total	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Tp (ns)**	**Freq. Mhz**
61	768	7	87	1.536	5	76	1.536	4	10.983	91.050
Virtex V600FG676										
Slices (CLBs)			**Flip-Flop**			**Luts de 4 entradas**			**Tempo de Propagação**	
Total	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Tp (ns)**	**Freq. Mhz**
61	6.912	1	87	3.824	1	76	13.824	1	14.852	67.331

Olhando a tabela 4.3 é possível verificar que a versão inicial utiliza uma porcentagem muito pequena tanto no FPGA Spartan2 quanto no Virtex, permitindo a inclusão de novos recursos, como suporte a uma nova topologia por exemplo, sem a necessidade de trocar de modelo ou inclusão de recursos físicos adicionais.

As medidas de propagação e freqüência no circuito no FPGA Spartan2 são respectivamente 10,983 ns e 91,050 Mhz, e no FPGA Virtex são respectivamente 14,812 ns e 67,331 Mhz.

Isso mostra que apesar do FPGA Spartan2 possuir menos recursos e menor espaço para mapeamento interno no *chip* do que no Virtex, a versão deste projeto fica mais veloz no Spartan2 pelo fato do tempo de propagação ser menor, e aumenta a freqüência do processador.

Também se pode perceber essa diferença analisando os números de Flip-Flops e Luts de 4 entradas, onde as quantidades nos dois modelos de FPGA são iguais, mas a ocupação é

maior no Virtex, principalmente em relação aos Luts de 4 entradas, influenciando diretamente no tempo de propagação e freqüência do processador.

Os *Floor Plans*, que são as plantas baixas das duas implementações do processador são mostradas abaixo nas figuras 4.6 e 4.7, que mostram a alocação dos componentes do código em VHDL respectivamente nos *chips*.Spartan2 2S50FG256 e Virtex V600FG676, ajudando a visualizar os parâmetros espaciais e temporais apresentados na tabela 4.3, indicando a ocupação do chip nos modelos utilizados.

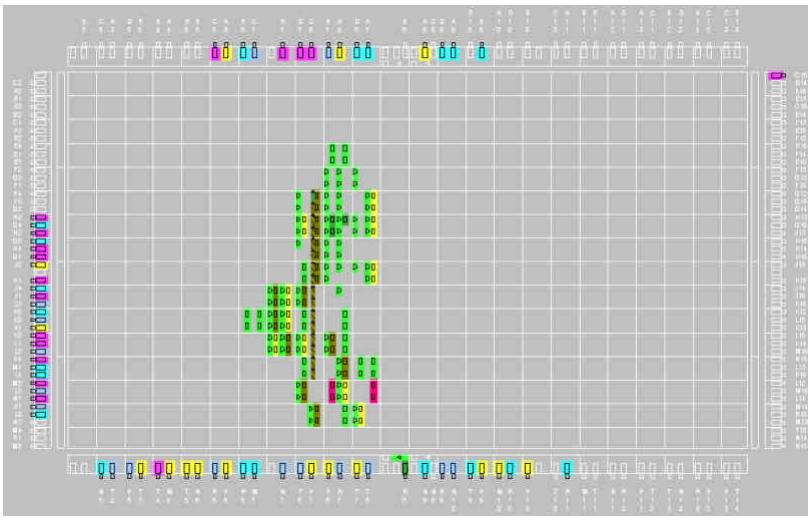


FIGURA 4.6 – *Floor Plan* Spartan2 S50FG256 – Topologia Anel Unidirecional

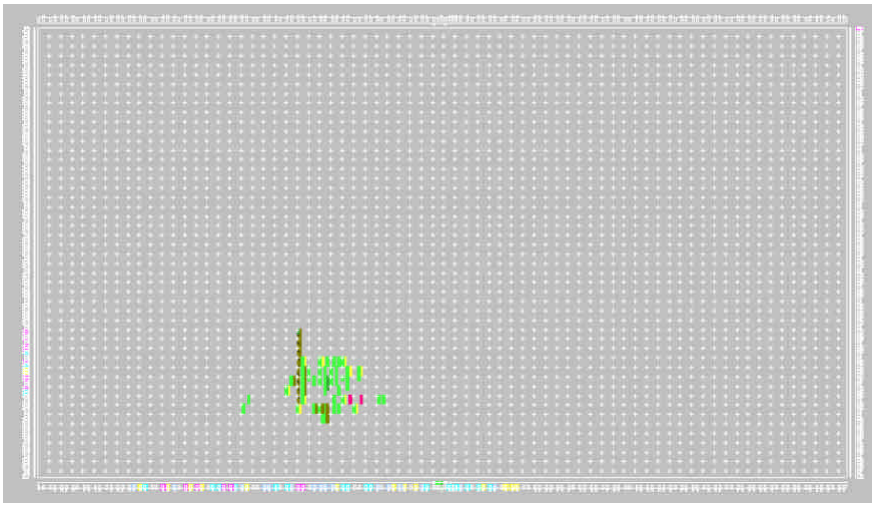


FIGURA 4.7 – *Floor Plan* Virtex V600FG676 – Topologia Anel Unidirecional

4.3 ToPSeC – Topologia Árvore Binária

A segunda topologia estudada é a Árvore Binária. Nesta topologia, cada nó também é um ToPSeC, mas a estrutura dessa topologia é uma árvore binária, isto é, com tres ramificações a partir de cada nó, exceto no nó inicial dessa topologia. O primeiro nó que possui apenas dois nós vizinhos e os demais nós vizinho possuem três ramificações.

O seu comportamento se resume em um dos nós enviar um dado a um outro nó qualquer que estaria nessa rede. Nessa topologia, os nós estariam conectados um para três, onde cada nó possui 3 nós vizinhos. Esse dado trafegaria na rede sempre fazendo comparações de lado e nível, verificando essas informações e comparando-as com o do nó atual, para que dessa forma, ele decida a melhor direção a seguir. Essas comparações são feitas através de uma tabela de mapeamento dessa topologia que cada nó possui.

No caminho até o destino da informação, haveria vários outros nós, que devem verificar se são o destino dessa informação e, em caso negativo, refazer as comparações necessárias para novamente decidir o melhor caminho e repassar a informação até que esta chegue ao seu destino.

Para o funcionamento dessa topologia também não importa o número de nós que essa rede possa ter, pois o tratamento da informação que chega é feito no nó local, mas por definição de limite de arquitetura, foi estipulado em 15 nós.

A implementação dessa topologia segue as mesmas definições da topologia Anel Unidirecional, como declaração do identificador do nó, o número de nós na rede e o tamanho da informação trafegada.

4.3.1 Arquitetura do ToPSeC – Topologia Árvore Binária

Baseada na arquitetura do NPSoC, mas também com algumas modificações na arquitetura, devido às alterações propostas, a arquitetura do ToPSeC para também oferecer suporte a esta topologia, ficou definida da seguinte forma:

- 4 portas de entrada, nomeadas de E1 a E4 de 16 bits;
- 4 portas de saída, nomeadas de S1 a S4 de 16 bits;
- 4 registradores de propósito geral de 8 bits (A, B, C e D);

- 1 registrador específico para roteamento de 4 bits (ID);
- 3 registradores específicos para roteamento de 8 bits (OD, CMP e POSIC);
- Unidade Lógica e Aritmética (ULA);
- Contador de Programa (PC);
- Unidade de Controle (UC);
- Registrador de Instruções (RI).

A figura 4.8 traz uma ilustração da arquitetura do ToPSeC neste estágio. Nela é possível verificar as principais diferenças de arquitetura desta nova versão em relação à versão anterior descritos na figura 4.1.

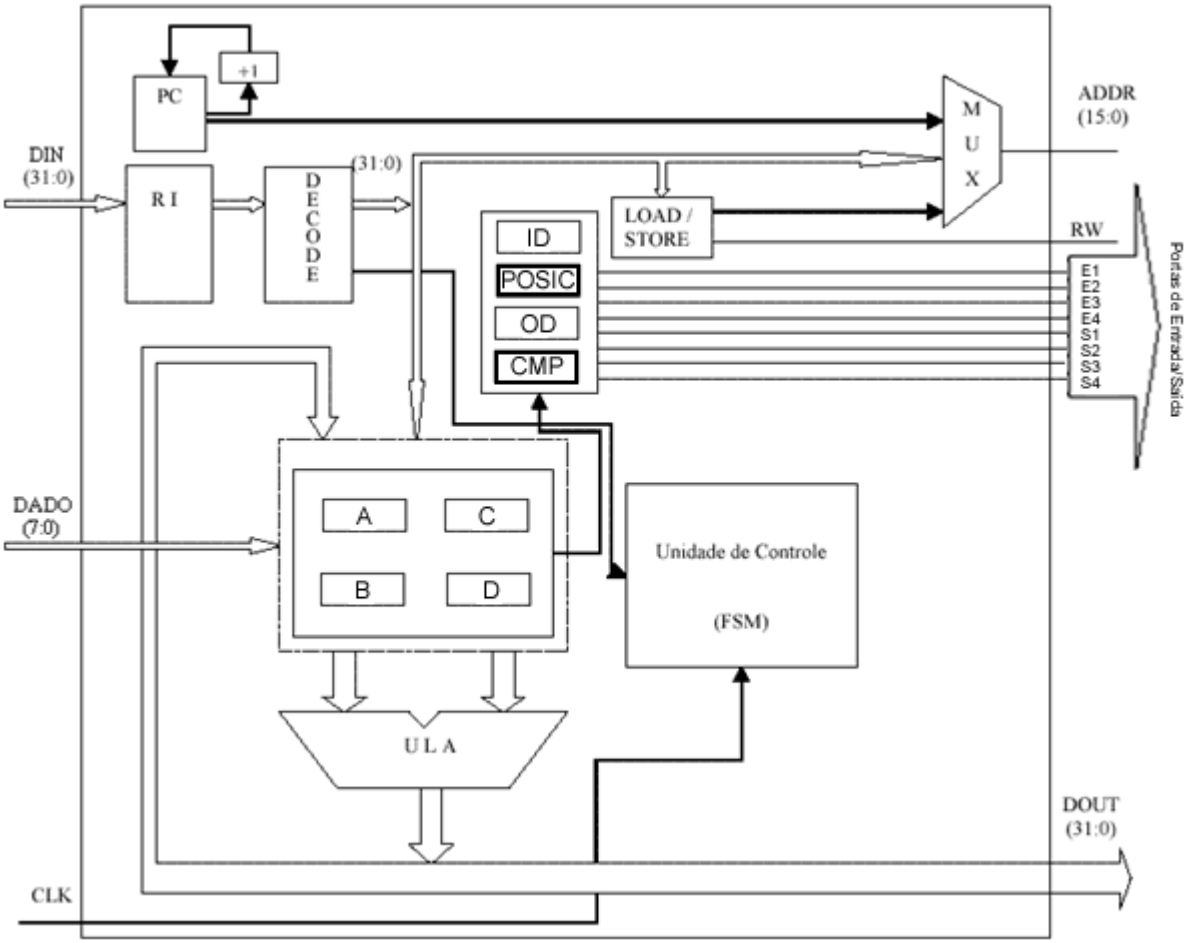


FIGURA 4.8 – Arquitetura do ToPSeC – Versão 2

As principais diferenças de arquitetura em relação à implementação da versão anterior é a inclusão de dois novos registradores de propósito específico, como POSIC (Posição) e CMP (Comparador), que são utilizados na decisão do melhor caminho dentre os três nós vizinhos para o envio do dado e a utilização de uma memória com tamanho de palavra de 24 bits e com 15 posições, que correspondem à quantidade de nós na rede. Essa memória contém o mapeamento da topologia Árvore Binária, conforme é mostrado na figura 4.9.

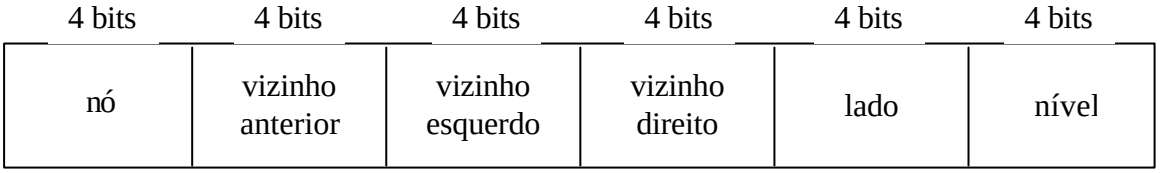


FIGURA 4.9 – Formato da memória de mapeamento da topologia Árvore Binária

Essa topologia foi implementada mantendo a versão inicial e adicionando recursos para que dessa forma, continue oferecendo suporte à topologia Anel Unidirecional e também ofereça suporte à topologia Árvore Binária.

Nestas duas novas versões, o tamanho das portas e entrada, de saída e da informação que trafega na rede continua idêntico à versão inicial (figura 4.2).

Com relação às versões 2 e 3, não existem diferenças de arquitetura entre elas. As diferenças são apenas quanto às instruções, que foram otimizadas de uma para outra.

4.3.2 Conjunto de instruções (ISA) – Topologia Árvore Binária

Na versão 2 que implementou a topologia Árvore Binária, o conjunto de instruções do Processador de Rede ToPSeC também segue o modelo RISC, onde foi definido um reduzido conjunto de instruções, contendo apenas 24 instruções que dão suporte a essa topologia, cumulativamente com a topologia Anel Unidirecional.

A tabela 4.4 traz apenas as instruções que foram adicionadas ou modificadas em relação à versão inicial, não relacionando aquelas que não sofreram alteração.

Para a versão 2, dez novas instruções foram incluídas, sendo seis instruções de desvio (JMPDSI, JMPDB, JMPDIA, JMPDDA, JMPNP, JMPNI) e quatro instruções de comparação (CMPSN, CMPBN, CMPNP, CMPNI).

A instrução de movimentação ENT, e as instruções de desvio JUMP e JDEST foram aproveitadas por oferecer as funcionalidades necessárias para a topologia Árvore Binária. As instruções de movimentação LID e a instrução de envio SEND foram alteradas para atender as necessidades de envio na topologia implementada nesta versão.

TABELA 4.4 – Instruções utilizadas no ToPSeC – Versão 2

Movimentação

ENT	Armazena a origem, o destino e o dado	**LID**	Carrega o endereço e a posição do nó atual
Desvio			
JMPDSI	Desvia se nível do destino for menor ou igual ao nó atual	**JMPDIA**	Desvia se o lado do destino for igual ao nó atual
JMPDB	Desvia se nível do destino for maior que o nó atual	**JMPDDA**	Desvia se o lado do destino for diferente do nó atual
JMPNP	Desvia se for para o maior vizinho par	**JMPNI**	Desvia se for para o maior vizinho ímpar

Comparação

CMPSN	Compara o menor vizinho	**CMPNP**	Compara o maior vizinho par
CMPBN	Compara os maiores vizinhos	**CMPNI**	Compara o maior vizinho ímpar

Envio

SEND	Envia a informação através da porta de saída ao nó verificado

Com relação à otimização dessa versão, houve a eliminação das instruções de comparação CMPNP e CMPNI e das instruções de salto JMPNP e JMPNI que foram incorporadas à nova instrução JMPN. Dessa forma, o programa *assembly* se torna mais enxuto e rápido, conforme comparação dos programas de teste constante na tabela 4.5.

4.3.3 Programa Teste e Funcionamento – Topologia Árvore Binária

Para a topologia Árvore Binária, houve algumas definições chaves que foram feitas para que fosse possível fazer a implementação das instruções da maneira como elas foram realizadas. Inicialmente, para a topologia Anel Unidirecional, foram criados dois registradores específicos, o ID, que representa o identificador do nó e o OD, que representa o identificador do nó de origem e de destino. Aproveitando esses dois registradores, mas não suficientes para a topologia Árvore Binária, também foram criados dois novos registradores específicos, o POSIC e o CMP. Levando-se em conta que essa topologia necessita de novos elementos,

foram definidos lados e níveis. Dessa forma, todo nó além do seu identificador, também possui um lado e nível, que são armazenados pelo registrador POSIC, e o registrador CMP, que armazena o nó correto para o envio da informação, de forma que seja o caminho correto. Também foi implementada uma memória ROM contendo informações de todos os nós dessa rede. Posteriormente, essa memória deverá ser uma memória RAM que será preenchida por uma ação de *broadcast*. Isso tudo para que cada nó seja capaz de, através de comparações do nó atual e do nó destino, verificar qual o nó vizinho que melhor se encaixa para que dessa forma, o caminho siga a melhor direção. A figura 4.10 mostra a o desenho da topologia Árvore Binária e os seus lados e níveis estabelecidos neste trabalho.

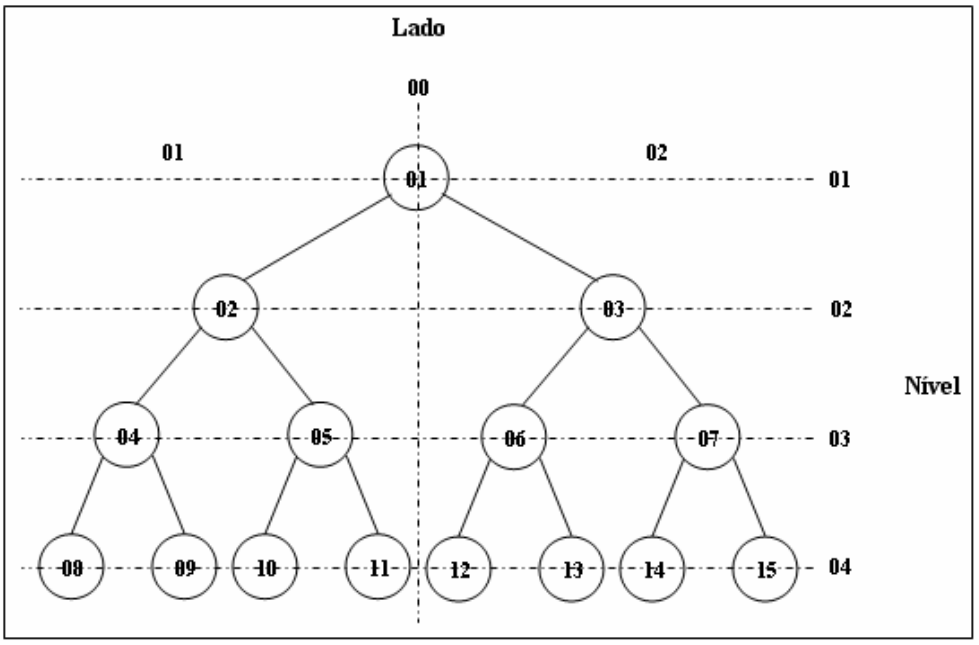


FIGURA 4.10 – Funcionamento da Topologia Árvore Binária

Analisando a figura 4.10, há a possibilidade de se identificar os nós, mas principalmente os lados e os níveis estabelecidos, que são utilizados como parâmetros de comparação na decisão de qual nó vizinho do nó em questão será o próximo a receber a informação. Nesta topologia, o ToPSeC possui 3 comportamentos diferentes em relação ao recebimento da informação:

1 º. A informação é para o nó atual;

2 º. O destino está em um nível menor ou igual ao nível do nó atual;

3 º. O destino está em um nível maior ao nível do nó atual:

   A. O destino e o nó atual estão do mesmo lado;

   B. O destino e o nó atual estão em lados diferentes;

Ainda em relação ao terceiro comportamento, cada situação apresentada (A e B) poderá ocorrer para casos onde o identificador do próximo nó a receber a informação seja par ou ímpar, expandindo as duas situações do terceiro comportamento para 4 possibilidades.

Dependendo da situação que se configurar, o ToPSeC, baseado no seu código *Assembly*, descrito na tabela 4.5 a), decidirá para qual vizinho enviar a informação recebida, de modo que ela se movimente pelo caminho mais curto.

Como pode ser observado na tabela 4.5 b), no código *assembly* da versão otimizada, houve uma diminuição de código, conforme já descrito no item 4.3.2. Com a eliminação das instruções CMPNP, CMPNI, JMPNP e JMPNI e suas funcionalidades incorporadas à nova instrução JMPN, houve uma diminuição do código assembly e consequentemente, uma diminuição do número de ciclos, tornando o programa *assembly* da otimização da versão 2 mais rápido.

TABELA 4.5 – Comparação das instruções *Assembly* do ToPSeC Versões 02

a)

	ToPSeC – V2	Ciclos
0	LID 01,12	2
1	ENT	2
2	JDEST 20	2
3	JMPDSI 5	3
4	JMPDB 7	3
5	CMPSN	3
6	JUMP 19	2
7	JMPDIA 9	3
8	JMPDDA 14	3
9	CMPBN	3
10	CMPNP	2
11	JMPNP 19	2
12	CMPNI	2
13	JMPNI 19	2
14	CMPBN	3
15	CMPNP	2
16	JMPNP 19	2
17	CMPNI	2
18	JMPNI 19	2
19	SEND	2
20	HALT	2
Total		49

b)

	V2 - Otimizado	Ciclos
0	LID 01,12	2
1	ENT	2
2	JDEST 14	2
3	JMPDSI 5	3
4	JMPDB 7	3
5	CMPSN	3
6	JMPN 13	2
7	JMPDIA 9	3
8	JMPDDA 11	3
9	CMPBN	3
10	JMPN 13	2
11	CMPBN	3
12	JMPN 13	2
13	SEND	2
14	HALT	2
Total		37

Com a otimização da versão 2, o número de instruções passou de 24 para 20, o que representa uma melhora de 24% em relação à quantidade de ciclos que eram de 49 e agora são 37.

Com relação aos 3 comportamentos do ToPSeC descritos anteriormente, no primeiro e segundo comportamentos, o desempenho medido em ciclos, apesar da diferença de instruções, são idênticos. Já no terceiro comportamento, a versão otimizada foram 2 ciclos mais rápida em relação à versão 2 em todos os casos, representando um ganho de 11% em cada caso.

4.3.4 Simulações da Topologia Árvore Binária

Neste tópico a simulação é referente ao funcionamento do programa que faz o envio da informação através de uma rede cuja topologia seja Árvore Binária (figura 4.11).

Para esta topologia existem 3 comportamentos diferentes já expostos anteriormente. A seguir, é apresentado cada um desses comportamentos, devidamente explicados e ilustrados:

1º Comportamento – A informação é para o nó atual:

Nesse primeiro comportamento demonstrado na figura 4.11, a operação é semelhante à topologia Anel Unidirecional, a única diferença é que na topologia Árvore Binária, na instrução LID é informado apenas o identificador, e na topologia Árvore Binária juntamente com o identificador, também é informado a posição do nó na topologia.

Inicialmente é informado o número do nó atual, seu lado e nível (1) que são armazenados respectivamente nos registradores ID e POSIC (2), através da entrada de dados DIN pela instrução LID.

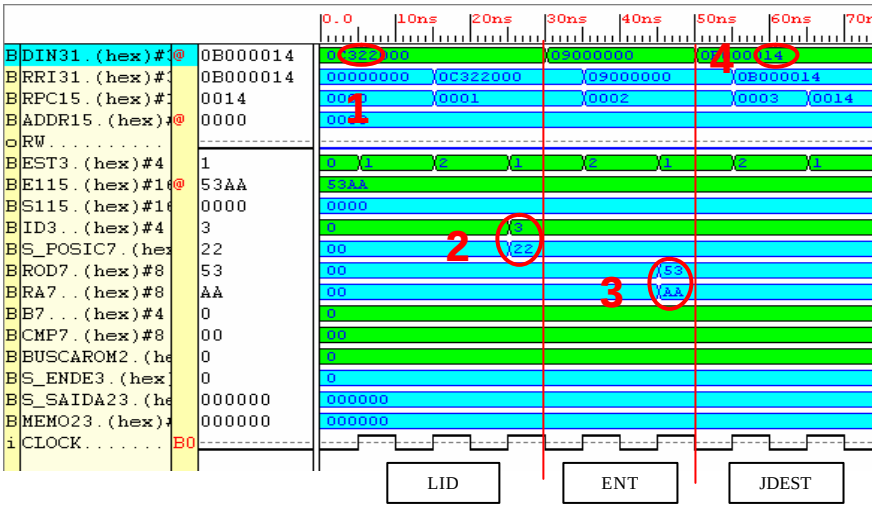


FIGURA 4.11 – Simulação 1º comportamento – Topologia Árvore Binária

Na seqüência, a instrução ENT armazena os endereços do nó que enviou anteriormente e o endereço do destino no registrador ID e o dado no registrador A (3) vindos pela porta de entrada E1.

Em seguida é feita uma comparação do identificador do nó atual com o destino. Se essa comparação for igual, significa que a informação chegou ao destino e o programa sofre um desvio para o final (4).

A simulação mostrada na figura 4.11 corresponde às instruções LID, ENT e JDEST, descritas em VHDL na seguinte estrutura:

```
WHEN INSTR_LID =>
      ID <= RI(23 DOWNTO 20);
      POSIC <= RI(19 DOWNTO 12)
      EST <= BUSCA;

WHEN INSTR_ENT =>
      A <= E1(7 DOWNTO 0);
      OD(7 DOWNTO 0)  <= E1(15 DOWNTO 8);
      EST  <= BUSCA;

WHEN INSTR_JDEST =>
      IF ID(3 DOWNTO 0) = OD(3 DOWNTO 0) THEN
      PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
      RW <='0';
      END IF;
```

2º Comportamento – O destino está em um nível menor ou igual ao nível do nó atual:

Este segundo comportamento, igualmente ao primeiro, tem início com a entrada dos dados (1) com a instrução LID, que armazena o número do nó corrente no registrador ID e os valores do lado e do nível deste mesmo nó no registrador POSIC e logo após a identificação do nó, a informação que vem da rede através da porta de entrada E1 é armazenada nos registradores A e OD, que armazenam o dado e a origem/destino respectivamente, através da instrução ENT.

O próximo passo é verificar com a instrução JMPDSI se o nível do destino é menor ou igual ao nível do nó atual. Essa comparação é feita consultando uma memória que cada nó possui e que contém o mapeamento da topologia corrente (1 e 2) . Caso o nível do destino for menor ou igual ao nível do nó atual, o programa é desviado (3 e 4) para então é feita uma outra comparação, agora com a instrução CMPSN para descobrir o menor vizinho ou vizinho anterior do nó atual (5) e armazená-lo no registrador CMP (6). Nesta topologia o menor

vizinho significa o vizinho acima. Neste comportamento, quando o nível do destino for menor que o nível do nó atual o vizinho que figura como o melhor caminho para receber a informação é o vizinho anterior ou menor. Após essa comparação um salto incondicional através da instrução JUMP (7) é feito para a instrução SEND, pois o próximo vizinho a receber a informação já é conhecido. No envio, o conjunto de dados que é enviado para o próximo nó é composto pelo identificador do nó que está enviando a informação neste momento, o identificador do destino e o dado (8). O envio é realizado através da porta de saída S1. Após isso, o programa é finalizado. Os passos dessa simulação são ilustrados na figura 4.12.

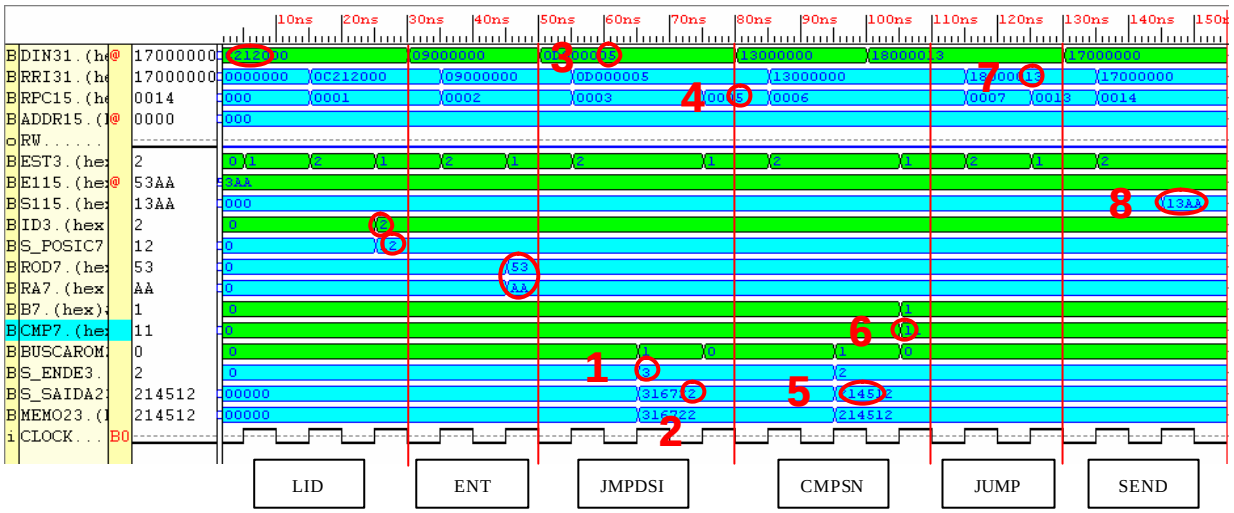


FIGURA 4.12 – Simulação 2º comportamento – Topologia Árvore Binária

A simulação mostrada na figura 4.12 corresponde principalmente às instruções JMPDSI, CMPSN, JUMP e SEND, descritas em VHDL na seguinte estrutura:

```
      WHEN INSTR_JMPDSI =>
            IF BUSCAROM = 0 THEN
                  ENDE <= CONV_INTEGER(OD(3 DOWNTO 0));
                  BUSCAROM <= 1;
            ELSIF SAIDA(3 DOWNTO 0) < POSIC(3 DOWNTO 0) OR SAIDA(3 DOWNTO
0) = POSIC(3 DOWNTO 0) THEN
                  PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
                  RW <='0';
                  EST <= BUSCA;
                  BUSCAROM <= 0;
            END IF;

      WHEN INSTR_CMPSN =>
            IF BUSCAROM = 0 THEN
                  ENDE <= CONV_INTEGER(ID(3 DOWNTO 0));
                  BUSCAROM <= 1;
```

```
        ELSE
              CMP(7 DOWNTO 4) <= SAIDA(11 DOWNTO 8);
              CMP(3 DOWNTO 0) <= SAIDA(11 DOWNTO 8);
              B(7 DOWNTO 4) <= SAIDA(11 DOWNTO 8);
        END IF;
        EST <= BUSCA;
        BUSCAROM <= 0;

  WHEN INSTR_JUMP =>
        PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
        RW <='0';
        EST <= BUSCA;
        BUSCAROM <= 0;

  WHEN INSTR_SEND =>
        S1 (15 DOWNTO 0) <= B (7 DOWNTO 4) & OD (3 DOWNTO 0) & A;
```

3º Comportamento – O destino está em um nível maior do que o nível do nó atual:

Neste terceiro comportamento, para a versão 2 e sua otimização, apresenta duas simulações, para o caso do destino e o nó atual estarem do mesmo lado e para o caso de estarem em lados diferentes.

Apenas lembrando que para todas as situações da topologia Árvore Binária, a seqüência inicial do programa *assembly*, que consiste na identificação do nó e sua posição com a instrução LID, e a entrada da informação com a instrução ENT são iguais. O próximo passo é a verificação se o nível do destino é maior que o do nó atual através da instrução JMPDB. Essa instrução realiza essa comparação verificando se o nível do destino é maior que o destino do nó atual através de uma consulta ao mapeamento do destino na memória (1 e 2).

Destino e nó atual em lados iguais:

Após essa verificação da instrução JMPDB, o programa sofre um desvio (3 e 4), para a instrução JMPDIA, onde é feita mais uma verificação para comparar se o lado do destino é igual ao lado do nó atual. Essa comparação é feita pesquisando na memória o mapeamento referente ao nó destino e comparando o seu lado como o lado do nó atual. Se essa comparação for verdadeira, o programa sofre um desvio (5) e é feita uma nova comparação dos maiores vizinhos ou vizinho esquerdo e direito, através da instrução CMPBN, que realiza uma pesquisa no mapeamento do nó atual na memória para verificar quais são os endereços dos vizinhos do nó atual (6).

Assim são armazenados no registrador CMP os dois maiores vizinhos do nó atual (7), de modo que haja a possibilidade de testar para qual será enviado a informação. Nesta topologia, os dois maiores vizinhos significa o vizinho par e o ímpar.

Depois da definição dos maiores vizinhos, existem dois caminhos, pois sempre que o envio da informação direciona os vizinhos abaixo do nó, existem duas possibilidades. Essa possibilidade se resume ao próximo nó a receber a informação ser par ou ímpar. Essa verificação é feita através das instruções CMPNP e CMPNI.

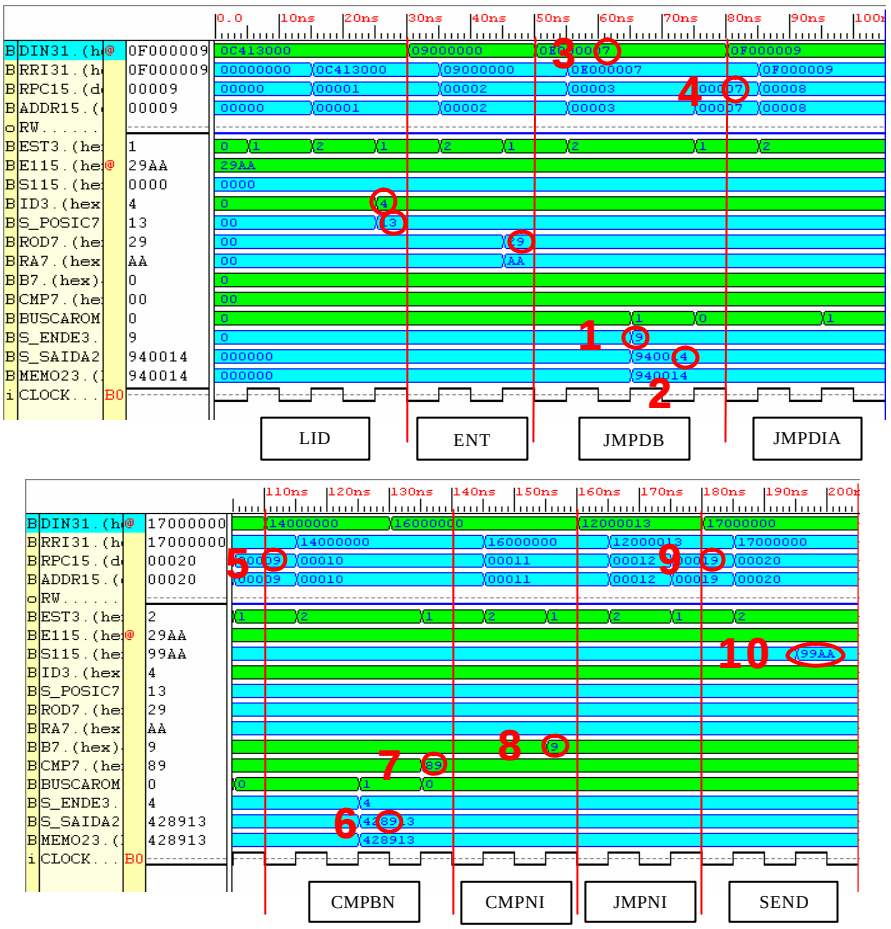


FIGURA 4.13 – Simulação 3º comportamento, mesmo lado do destino

Na simulação da figura 4.13, a situação mostrada é o envio para o vizinho ímpar ou vizinho esquerdo. A instrução CMPNI realiza essa verificação fazendo uma verificação do último *bit* do endereço do nó atual e do destino. Essa verificação do *bit* mais significativo mostra se o número verificado é divisível por 2, então ele é par e caso contrário, ele é ímpar. Nessa instrução essa verificação consiste em testar se o último *bit* do endereço do nó atual e do destino é diferente de zero. A instrução CMPNP faz exatamente o contrário, ou seja, verifica se o último bit do endereço do nó atual e do destino é igual à zero.

Com a comparação as instruções CMPNI ou CMPNP, o número do próximo nó é então armazenado no registrador B (8). Sequencialmente às instruções anteriores, as instruções JMPNI e JMPNP respectivamente irão acompanhar as instruções CMPNI e CMPNP. Essas duas instruções realizam um desvio caso a verificação da instrução anterior seja verdadeira (9). No caso da figura 4.13, o próximo nó a receber a informação é ímpar.

E, por fim, a instrução SEND monta a informação com o maior vizinho, que neste caso é ímpar, o destino da informação e o dado, e atribui a porta de saída S1 para que seja enviado ao próximo nó (10).

A simulação mostrada na figura 4.13 corresponde principalmente às instruções JMPDB, JMPDIA, CMPBN, CMPNI, JMPNI, descritas em VHDL na seguinte estrutura:

```
WHEN INSTR_JMPDB =>
      IF BUSCAROM = 0 THEN
            ENDE <= CONV_INTEGER(OD(3 DOWNTO 0));
            BUSCAROM <= 1;
      ELSIF SAIDA(3 DOWNTO 0) > POSIC(3 DOWNTO 0) THEN
            PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
            RW <='0';
            EST <= BUSCA;
            BUSCAROM <= 0;
      END IF;


WHEN INSTR_JMPDIA =>
      IF BUSCAROM = 0 THEN
            ENDE <= CONV_INTEGER(OD(3 DOWNTO 0));
            BUSCAROM <= 1;
      ELSIF SAIDA(7 DOWNTO 4) = POSIC(7 DOWNTO 4) THEN
            PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
            EST <= BUSCA;
            BUSCAROM <= 0;
      END IF;

WHEN INSTR_CMPBN =>
      IF BUSCAROM = 0 THEN
            ENDE <= CONV_INTEGER(ID(3 DOWNTO 0));
            BUSCAROM <= 1;
      ELSE
            CMP(7 DOWNTO 4) <= SAIDA(15 DOWNTO 12);
            CMP(3 DOWNTO 0) <= SAIDA(11 DOWNTO 8);
            EST <= BUSCA;
            BUSCAROM <= 0;
      END IF;

WHEN INSTR_CMPNI =>
      IF OD (0 DOWNTO 0) /= "0" AND CMP (0 DOWNTO 0) /= "0" THEN
            B (7 DOWNTO 4) <= CMP (3 DOWNTO 0);
      END IF;
      EST <= BUSCA;
      BUSCAROM <= 0;

WHEN INSTR_JMPNI =>
```

```
PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
RW <='0';
EST <= BUSCA;
```

Em relação à versão 2, a sua otimização (figura 4.14) segue praticamente o mesmo curso da simulação anterior, mas com a otimização das instruções, as instruções CMPNI, CMPNP, JMPNI e JMPNP foram substituídas apenas por JMPN, reduzindo dessa forma o programa e o seu tempo de execução em 2 clocks por instrução que deixa de ser utilizada, mas a funcionalidade do programa *assembly* dessas instruções continua o mesmo.

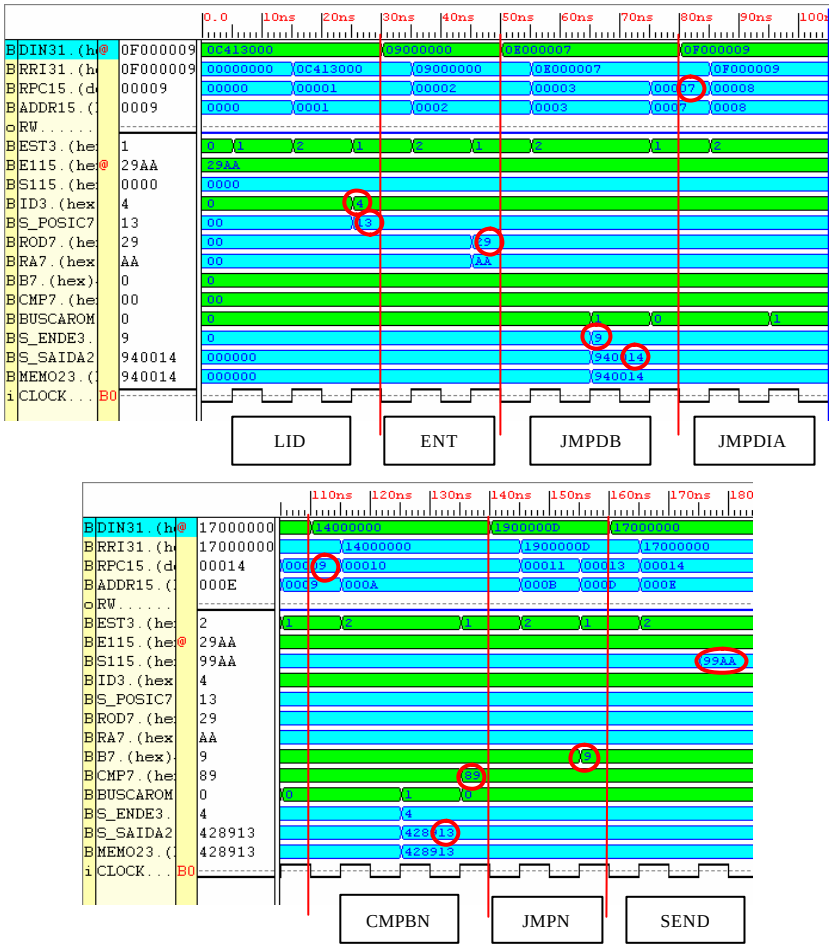


FIGURA 4.14 – Simulação 3º comportamento, mesmo lado do destino - otimizado

A simulação da figura 4.14 segue as mesmas instruções da simulação anterior, diferenciando-se a partir da instrução CMPBN, onde eram anteriormente executadas as instruções CMPNI, CMPNP, JMPNI e JMPNP, substituídas pela instrução JMPN que agora assumiu o papel de verificar se o último *bit* do destino e do nó atual é igual ou diferente de zero. De acordo com o resultado dessa comparação, define para qual dos dois maiores

vizinhos (vizinho ímpar/esquerdo ou par/direito) deverá ser enviada a informação e desvia para a instrução SEND que enviará o endereço do nó atual, o endereço do destino e o dado.

A simulação mostrada na figura 4.14 corresponde principalmente à instrução JMPN, descrita em VHDL na seguinte estrutura:

```
WHEN INSTR_JMPN =>
      IF OD (0) = '0'  AND CMP (4)  = '0' THEN
            B (7 DOWNTO 4) <= CMP (7 DOWNTO 4);
            PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
            RW <='0';
            EST <= BUSCA;
      ELSIF OD (0) /= '0' AND CMP (0) /= '0' THEN
            B (7 DOWNTO 4) <= CMP (3 DOWNTO 0);
            PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
            RW <='0';
            EST <= BUSCA;
      END IF;
```

Destino e nó atual em lados diferentes:

Nesta situação, a única instrução que diferencia da situação ilustrada na figura 4.13 é a instrução JMPDIA, que é substituída pela JMPDDA. Essa instrução faz justamente a verificação se o destino e o nó atual estão em lados diferentes. As outras instruções permanecem as mesmas, inclusive nas situações em que o vizinho a receber a informação for ímpar (CMPNI e JMPNI) ou par (CMPNP e JMPNP).

Em relação à otimização da situação anterior (figura 4.14) em nada diferencia nesta situação, já que com a substituição das instruções CMPNI, CMPNP, JMPNI e JMPNP por JMPN, a seqüência de instruções segue de maneira idêntica.

4.3.5 Estatísticas de Implementação e Desempenho – Topologia Árvore Binária

Após a finalização das simulações da versão 2, o ToPSeC foi implementado em dois modelos de FPGAs para que fosse feito um comparativo de implementação e desempenho. A tabela 4.6 traz as estatísticas de dois modelos que foram implementados através da ferramenta Xilinx 3.1 (BOURT, 1998).

Olhando a tabela 4.6 e comparando com as estatísticas de implementação e desempenho da versão inicial correntes na tabela 4.3, é possível verificar que a versão 2

continua utilizando uma porcentagem pequena tanto no FPGA Spartan2 quanto no Virtex, mostrando que ainda é possível inclusão de novos recursos pois ainda existe espaço.

TABELA 4.6 – Estatísticas com parâmetros espaciais e temporais da Versão 02

Spartan2 S50FG256										
Slices (CLBs)			Flip-Flop			Luts de 4 entradas			Tempo de Propagação	
Total	Ocupado	%	Total	Ocupado	%	Total	Ocupado	%	Tp (ns)	Freq. Mhz
118	768	15	108	1.536	7	180	1.536	11	15,663	63,845
Virtex V600FG676										
Slices (CLBs)			Flip-Flop			Luts de 4 entradas			Tempo de Propagação	
Total	Ocupado	%	Total	Ocupado	%	Total	Ocupado	%	Tp (ns)	Freq. Mhz
118	6.912	1	108	13.824	1	180	13.824	1	15,224	65,686

Na versão 2, as medidas de propagação e freqüência no circuito no FPGA Spartan2 são respectivamente 15,663 ns e 63,845 Mhz, e no FPGA Virtex são respectivamente 15,224 ns e 65,686 Mhz.

Isso mostra que neste caso, apesar do FPGA Spartan2 ser um modelo de FPGA mais simples e a sua ocupação interna ser menor em relação ao no *chip* Virtex, a lógica envolvida para essa implementação se mostra mais favorável no FPGA Virtex, possuindo este um desempenho melhor se comparado com o *chip* Spartan2.

Em relação a estas estatísticas da versão inicial, conclui-se que houve uma diminuição de desempenho em relação ao FPGA Spartan2, mas no Virtex não houve muita diferença de desempenho, mostrando que esse modelo de FPGA possui um suporte bastante robusto para vários novos recursos que venham a serem incluídos.

Os *Floor Plans*, que são as plantas baixas das duas implementações do processador são mostrados abaixo nas figuras 4.15 e 4.16. Eles mostram a alocação dos componentes empregados na programação em VHDL respectivamente nos *chips* Spartan2 2S50FG256 e Virtex V600FG676, permitindo visualizar os parâmetros espaciais e temporais apresentados na tabela 4.6, indicando a ocupação do chip nos modelos utilizados.

Perceber-se o tamanho dos *chips* e quanto dentro deles é ocupado pela implementação atual. O modelo de FPGA Spartan2 2S50FG256 possui um espaço menor e por isso a área de ocupação é maior, já o modelo Virtex V600FG676 por possuir um espaço consideravelmente maior, a área de ocupação é menor. Mas isso não altera a quantidade de alguns recursos utilizados, como *Flip-Flops* e *Luts* de 4 entradas, que são iguais.

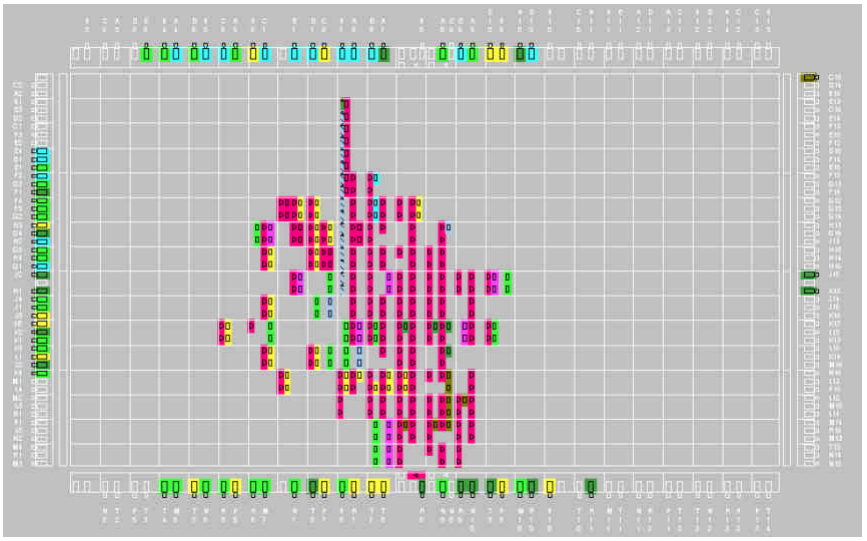


FIGURA 4.15 – *Floor Plan* Spartan2 S50FG256 – Topologia Árvore Binária

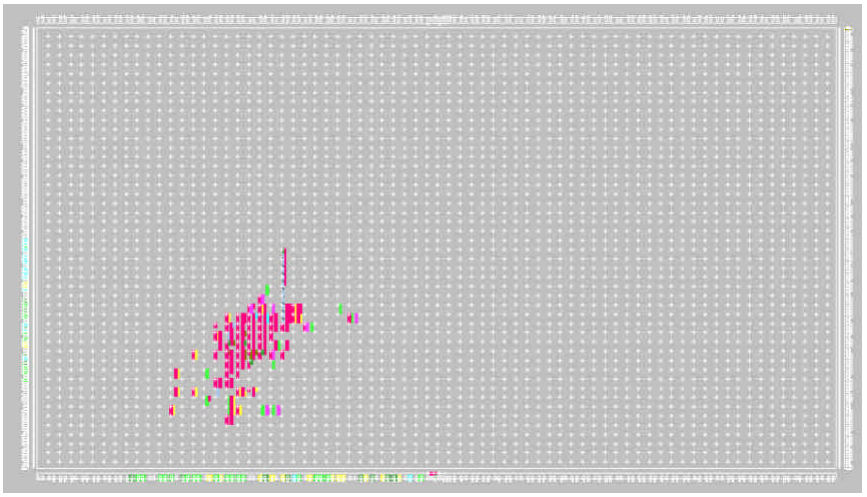


FIGURA 4.16 – *Floor Plan* Virtex V600FG676 – Topologia Árvore Binária

Com relação à otimização da versão 2 e olhando a tabela 4.7 é possível verificar que em relação à versão 2, agora a porcentagem de ocupação é menor ainda.

Nesta otimização de versão, as medidas de propagação e freqüência no circuito no FPGA Spartan2 são respectivamente 13,763 ns e 72,659 Mhz, e no FPGA Virtex são respectivamente 14,004 ns e 71,205 Mhz

Com esses números, pode-se perceber que a freqüência do processador aumentou consideravelmente e o tempo de propagação no *chip* diminuiu. Isso mostra que a otimização na quantidade de instruções reduz a lógica e a quantidade de recursos do FPGA, refletindo diretamente no desempenho do processador nos dois modelos de FPGA utilizados.

Em relação apenas à otimização feita, o modelo do FPGA Spartan2 se mostrou mais favorável, pois apresentou um desempenho melhor, mostrando resultados contrários aos da versão 2, onde o modelo de FPGA Virtex apresentou melhores resultados. Isso provavelmente se deve à lógica que deixou de ser empregada devido a otimização, que antes demandava recursos que agora não são mais utilizados.

TABELA 4.7 – Estatísticas com parâmetros espaciais e temporais da Versão 02 otimizada

Spartan2 S50FG256										
Slices (CLBs)			**Flip-Flop**			**Luts de 4 entradas**			**Tempo de Propagação**	
Total	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Tp (ns)**	**Freq. Mhz**
100	768	13	108	1.536	7	146	1.536	9	13,763	72,659
Virtex V600FG676										
Slices (CLBs)			**Flip-Flop**			**Luts de 4 entradas**			**Tempo de Propagação**	
Total	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Tp (ns)**	**Freq. Mhz**
100	6.912	1	108	13.824	1	146	13.824	1	14,044	71,205

Os *Floor Plans*, mostradas abaixo nas figuras 4.17 e 4.18, mostram a alocação dos componentes do código em VHDL respectivamente nos *chips* Spartan2 2S50FG256 e Virtex V600FG676 e permitem visualizar os parâmetros espaciais e temporais apresentados na tabela 4.7, indicando a ocupação do chip nos modelos utilizados.

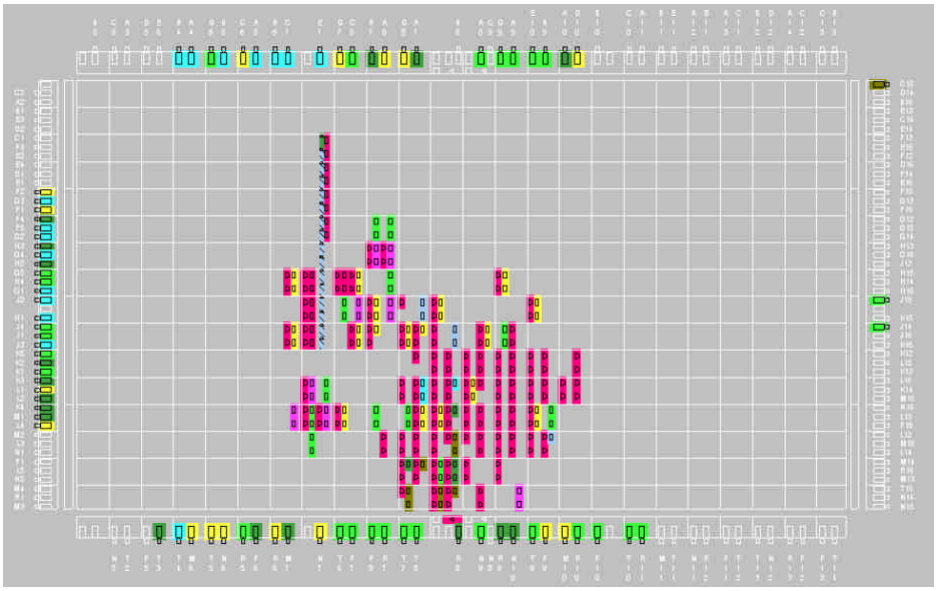


FIGURA 4.17 – *Floor Plan* Spartan2 S50FG256 – Topologia Árvore Binária otimizada

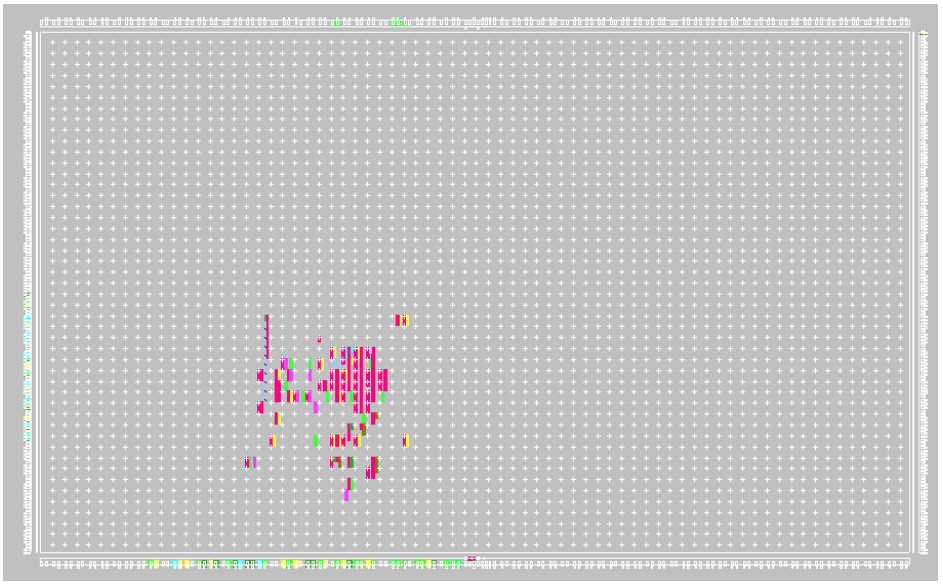


FIGURA 4.18 – *Floor Plan* Virtex V600FG676 – Topologia Árvore Binária otimizada

Observando os *Floor Plans* da versão 2 e da sua otimização é possível perceber uma diminuição de componentes utilizados entre os dois modelos de FPGA, mostrando que a otimização realmente teve impacto na implementação do processador.

4.4 ToPSeC – Escolha de topologia

Nos tópicos anteriores foi abordado o estudo e funcionamento das topologias Anel Unidirecional e Árvore Binária. Este tópico abordará a mudança dessas duas topologias neste processador.

Até o momento, as duas topologias propostas eram vistas separadamente. A partir deste ponto, há a possibilidade de escolha e mudança de topologia.

Para que a realização dessa implementação fosse possível, foi necessário adaptar a topologia Anel Unidirecional para que esta também possuísse o seu mapeamento da rede disposto em uma memória, assim como é feito na topologia Árvore Binária. Desta forma, se mantém a idéia de reprogramação e uma brecha para que futuramente isso seja feito neste processador em tempo real e com opções de reconfiguração dinâmica.

4.4.1 Arquitetura do ToPSeC – Escolha de topologia

Nesta versão, a arquitetura permanece a mesma que a da versão 2, inclusive pelo fato da quantidade de nós que cada topologia possui ser igual, e pelo tamanho das memórias que contem os mapeamentos das duas topologias também serem iguais.

As alterações sofridas foram os sinais do processador que são utilizados para o acesso a memória que contém o mapeamento da rede. Antes eram utilizados apenas dois sinais para acesso a memória, um para informar o nó a ser verificado e o outro para receber informações de mapeamento vindas da memória. Devido à inclusão de mais um mapeamento no processador, referente a outra topologia, neste caso Anel Unidirecional, esses sinais tiveram de ser individualizados por topologia, funcionando como uma máquina de estados que neste caso contém 3 estados, sendo um estado para informar a qual mapeamento de topologia o processador deverá seguir, e um para informar que a busca no mapeamento da topologia escolhida já foi feita.

4.4.2 Conjunto de instruções (ISA) – Escolha de topologia

Na versão 3, além das instruções para dar suporte às duas topologias, também foram criadas instruções para permitir a escolha da topologia a ser utilizada a cada envio de informação.

A tabela 4.8 traz as instruções utilizadas nesse processador, classificando-as quanto a sua finalidade e destacando, após a tabela, aquelas que foram criadas e alteradas para essa versão.

TABELA 4.8 – Instruções utilizadas no ToPSeC – Versão 3

Movimentação

TOPO	Informa a topologia a ser utilizada

Desvio

JMPANUN	Desvia se a topologia informada for Anel Unidirecional	**JMPARBN**	Desvia se a topologia informada for Árvore Binária

Comparação

CMPLN	Compara o vizinho anterior

Em relação à versão 2, quatro novas instruções foram incluídas, sendo uma de movimentação (TOPO), duas de desvio (JMPANUN e JMPARBN) e uma de comparação (CMPLN).

Nessa nova versão, com a criação dessas novas instruções e com a utilização de uma nova memória que contém o mapeamento da topologia Anel Unidirecional, a instrução de envio SAD se tornou desnecessária e por esse motivo foi excluída do conjunto de instruções.

4.4.3 Programa teste e Funcionamento – Escolha de topologia

Para esta versão, em que há a escolha de topologia, como já visto anteriormente, nenhuma alteração substancial de arquitetura foi feita, apenas foi inserida uma nova memória com o mapeamento da topologia Anel Unidirecional. Os outros elementos do processador serão utilizados conforme a topologia que for escolhida.

A tabela 4.9 traz as instruções *assembly* da versão 3 juntamente com as instruções para escolha de topologia.

A função principal desse programa nesta versão diz respeito à escolha de topologia, onde na instrução TOPO é informada a topologia a ser utilizada; as instruções JMPANUN e JMPARBN realizam um desvio no código caso a topologia escolhida seja respectivamente Anel Unidirecional e Árvore Binária, e a instrução CMPLN busca o vizinho que se situa à esquerda, ou seja, na topologia Anel Unidirecional o vizinho à esquerda é o próximo vizinho nessa topologia a receber a informação.

TABELA 4.9 – Instruções *Assembly* do ToPSeC Versões 03

	ToPSeC – V3	Ciclos
0	TOPO 1/2	2
1	LID 2,12	2
2	ENT	2
3	JDEST 18	2
4	JMPARBN 7	2
5	CMPLN	3
6	JMPANUN 17	2
7	JMPDSI 9	3
8	JMPDB 11	3
9	CMPSN	3

continuação		Ciclos
10	JMPN 17	2
11	JMPDIA 13	3
12	JMPDDA 15	3
13	CMPBN	3
14	JMPN 17	2
15	CMPBN	3
16	JMPN 17	2
17	SEND	2
18	HALT	2
Total		46

4.4.4 Simulações da Escolha de Topologia

Neste tópico, a simulação é referente ao funcionamento do programa que faz a escolha de topologia e, dependendo da escolha, a execução do bloco de código referente à topologia escolhida.

As duas figuras seguintes (figura 4.19 e 4.20) mostram respectivamente a simulação da escolha nas topologias Anel Unidirecional e Árvore Binária.

Estas simulações têm início com a instrução TOPO, onde é informada a topologia desejada (1), que será controlada pelo componente BUSCAROM, que controla em qual topologia o programa irá pesquisar o mapeamento existente (2).

Após ser informado o número do nó atual (3) pela instrução LID, receber os endereços do nó emissor da informação, o destino e o dado (4) pela instrução ENT, e de verificar se a informação é para o nó atual através da instrução JDEST, na seqüência é executada a instrução JMPARBN, que verifica se a topologia escolhida é Árvore Binária. Na situação mostrada na figura 4.19, como a topologia escolhida é a Anel Unidirecional, essa instrução não satisfaz a condição e é executada na seqüência a instrução CMPLN, que compara o vizinho da esquerda (5), que no caso da topologia Anel Unidirecional (6) e então o próximo nó a receber a informação é armazenado no registrador CMP (7). Na seqüência, é verificada se a topologia escolhida é Anel Unidirecional através da instrução JMPANUN, que é o caso desta simulação. Ainda nesta instrução, caso a topologia escolhida realmente for Anel Unidirecional, o próximo nó que está armazenado no registrador CMP é armazenado também no registrador B (8) e o programa *assemly* sofre um desvio (9 e 10) para a instrução SEND para enviar a informação ao próximo nó, já que o próximo nó já é conhecido na instrução CMPLN.

FIGURA 4.19 – Simulação completa da escolha da topologia Anel Unidrecional

Já a figura 4.20 mostra a simulação da situação de escolha para a topologia Árvore Binária. Após seguir a mesma rotina da simulação anterior, é informada a topologia a ser utilizada (1 e 2), informar o número do nó atual, sua posição (3), receber o número do nó que enviou a informação, o seu destino e o dado (4) e verificar se o dado é para esse nó, a próxima instrução a ser executada é a instrução JMPARBN.

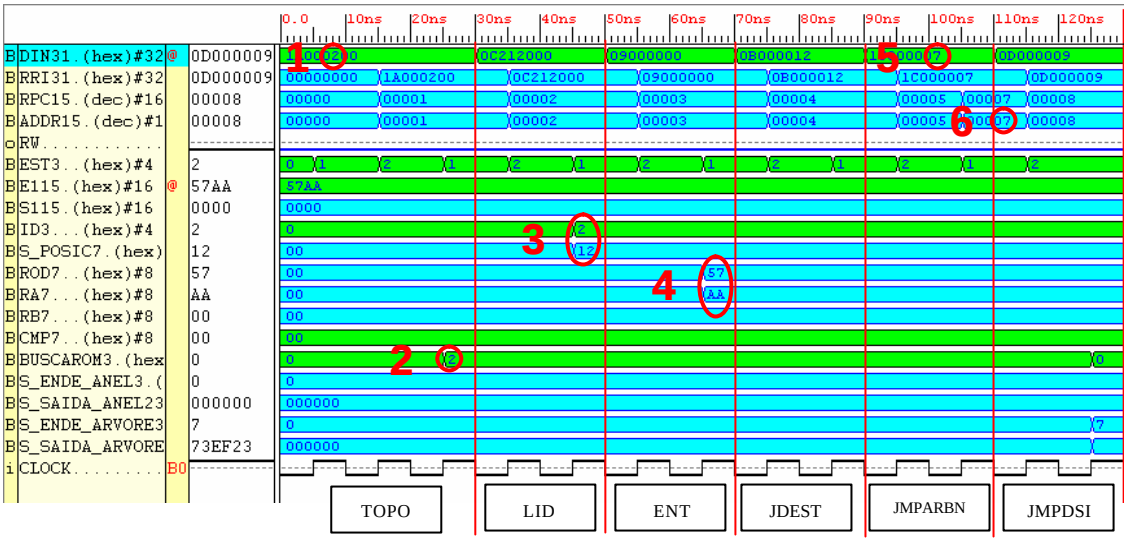


FIGURA 4.20 – Simulação parcial da escolha da topologia Árvore Binária

Essa instrução verifica se a topologia escolhida é a Árvore Binária. Como neste caso essa é a topologia escolhida, o código *assembly* sofre um desvio (5 e 6) para a instrução JMPDSI e é dada seqüência nas instruções para definir o melhor vizinho para enviar a informação para a topologia Árvore Binária.

A simulação mostrada na figura 4.19 e na figura 4.20 corresponde principalmente às instruções TOPO, JMPARBN, CMPLN, JMPANUN e JMPARBN, descritas em VHDL usando a seguinte estrutura:

```
WHEN INSTR_TOPO =>
      BUSCAROM(3 DOWNTO 0) <= RI(11 DOWNTO 8);
      RW <='0';
      EST <= BUSCA;

WHEN INSTR_JMPARBN =>
      IF BUSCAROM(3 DOWNTO 0) = "0010" THEN
            BUSCAROM(3 DOWNTO 0) <= "0010";
            PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
            RW <='0';
      END IF;
      EST <= BUSCA;

WHEN INSTR_CMPLN =>
      IF BUSCAROM(3 DOWNTO 0) = "0001" THEN
            ENDE_ANEL <= CONV_INTEGER(ID(3 DOWNTO 0));
            BUSCAROM(3 DOWNTO 0) <= "0000";
      ELSE
            CMP(7 DOWNTO 4) <= SAIDA_ANEL(15 DOWNTO 12);
            CMP(3 DOWNTO 0) <= SAIDA_ANEL(15 DOWNTO 12);
            EST <= BUSCA;
            BUSCAROM (3 DOWNTO 0) <= "0001";
      END IF;

WHEN INSTR_JMPANUN =>
```

```
        IF BUSCAROM(3 DOWNTO 0) = "0001" THEN
              B(7 DOWNTO 4) <= CMP(7 DOWNTO 4);
              BUSCAROM (3 DOWNTO 0) <= "0001";
              PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
              RW <='0';
        END IF;
        EST <= BUSCA;

WHEN INSTR_JMPARBN =>
        IF BUSCAROM (3 DOWNTO 0) = "0010" THEN
              PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
              RW <='0';
        END IF;
        EST <= BUSCA;
```

4.4.5 Estatísticas de Implementação e Desempenho – Escolha de topologia

Após a finalização das simulações da versão 3, o ToPSeC foi sintetizado em dois modelos de FPGAs para que fosse feito um comparativo de implementação e desempenho. A tabela 4.10 traz as estatísticas desses modelos de FPGAs através da ferramenta Xilinx 3.1 (BOURT, 1998).

TABELA 4.10 – Estatísticas com parâmetros espaciais e temporais da Versão 3

Spartan2 S50FG256										
Slices (CLBs)			**Flip-Flop**			**Luts de 4 entradas**			**Tempo de Propagação**	
Total	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Tp (ns)**	**Freq. Mhz**
118	768	15	118	1.536	7	117	1.536	11	15,487	64,570
Virtex V600FG676										
Slices (CLBs)			**Flip-Flop**			**Luts de 4 entradas**			**Tempo de Propagação**	
Total	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Tp (ns)**	**Freq. Mhz**
118	6.912	1	118	13.824	1	177	13.824	1	15,522	64,425

Olhando a tabela 4.6 é possível verificar que a versão 3, em relação à versão anterior ainda continua utilizando uma porcentagem pequena tanto no FPGA Spartan2 quanto no Virtex, mostrando que nesta versão, também é possível a inclusão de novos recursos pois ainda existe espaço.

Na versão 3, as medidas de propagação e freqüência no circuito no FPGA Spartan2 são respectivamente 15,487 ns e 64,570 Mhz, e no FPGA Virtex são respectivamente 15,522 ns e 64,425 Mhz.

Em relação a estas estatísticas da versão 2, conclui-se que houve uma pequena diminuição de desempenho, mas não uma mudança muito significativa pois as alterações nesta versão em relação à anterior foram pequenas e a lógica envolvida não diferencia da anterior, apenas aumenta de tamanho.

Os *Floor Plans* são mostradas abaixo nas figuras 4.21 e 4.22, que mostram a alocação dos componentes do código em VHDL respectivamente nos *chips* Spartan2 2S50FG256 e Virtex V600FG676, permitem visualizar os parâmetros espaciais e temporais apresentados na tabela 4.10, indicando a ocupação do chip nos modelos utilizados.

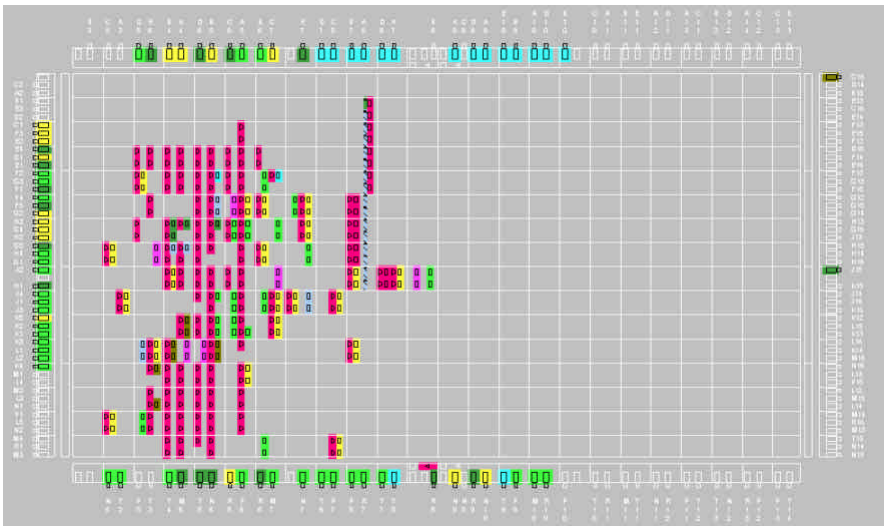


FIGURA 4.21 – *Floor Plan* Spartan2 S50FG256 – Escolha de topologia

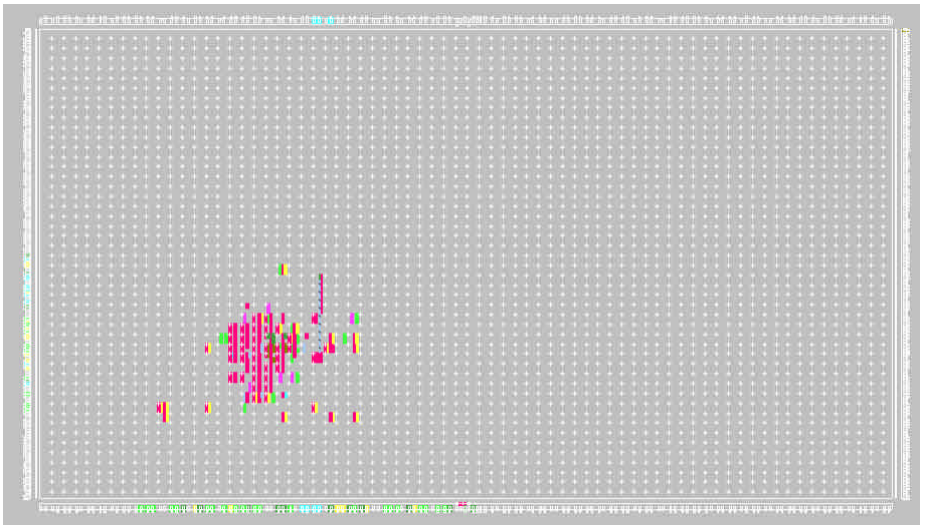


FIGURA 4.22 – *Floor Plan* Virtex V600FG676 – Escolha de topologia

4.5 ToPSeC – Caminho alternativo

Até o momento, este trabalho descrevia o funcionamento do processador ToPSeC inicialmente apenas na topologia Anel Unidirecional, depois passou a suportar a topologia Árvore Binária e por fim passou a suportar a escolha da utilização de uma dessas topologias.

Mas esse funcionamento se baseava na suposição de que todos os nós dessas duas topologias estivessem funcionando.

A partir deste tópico, será tratada a procura de um caminho alternativo dentro da topologia escolhida, de modo que o nó que estiver enviando a informação irá requisitar ao próximo nó o seu estado. Por questões de definição para que seja possível a simulação, para esta situação de busca de um caminho alternativo, cada nó que for requisitado responderá apenas se seu "Tipo" é de **Envio** ou **Recebimento**. Esse tipo visa definir o estado que o nó assume perante uma topologia para que seja possível o seu funcionamento.

Esse novo comportamento visa a busca de um caminho alternativo no caso em que o nó que for definido pelo nó atual para receber a informação não estiver com o seu "Tipo" em Recebimento. Atualmente essa busca por um caminho alternativo em caso de impossibilidade de envio da informação ao próximo nó está implementado apenas na topologia Anel Unidirecional. Futuramente também será implementado na topologia Árvore Binária.

4.5.1 Arquitetura do ToPSeC – Caminho alternativo

Baseada na arquitetura da última versão, mas com algumas modificações na arquitetura, devido às alterações propostas para essa nova implementação, como a inclusão do registrador TIPO e o aumento da porta de entrada E1 e porta de saída S1, a arquitetura do ToPSeC para esta topologia ficou definida da seguinte forma:

- 1 porta de entrada, nomeada de E1 de 20 bits;
- 3 portas de entrada, nomeadas de E2 a E4 de 16 bits;
- 1 portas de saída, nomeada de S1 de 20 bits;
- 3 portas de saída, nomeadas de S2 a S4 de 16 bits;
- 4 registradores de propósito geral de 8 bits (A, B, C e D);

- 2 registradores específicos para roteamento de 4 bits (ID e TIPO);
- 3 registradores específicos para roteamento de 8 bits (OD, CMP e POSIC);
- Unidade Lógica e Aritmética (ULA);
- Contador de Programa (PC);
- Unidade de Controle (UC);
- Registrador de Instruções (RI).

A figura 4.23 traz uma ilustração da arquitetura do ToPSeC neste estágio. Nela, é possível verificar as principais diferenças de arquitetura desta nova versão em relação à anterior, conforme a figura 4.8.

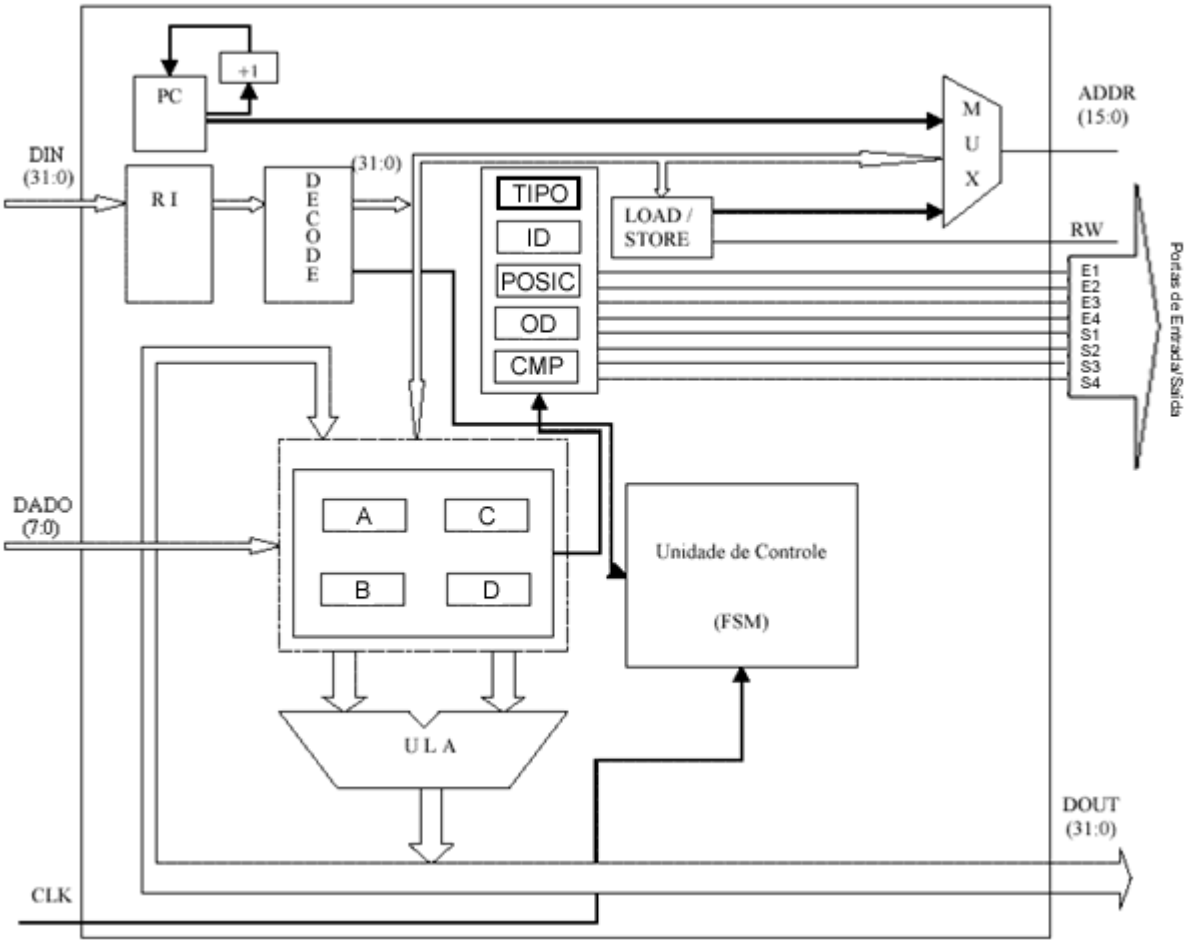


FIGURA 4.23 – Arquitetura do ToPSeC – Versão 4

A principal diferença de arquitetura em relação à implementação da versão anterior é a inclusão de um novo registrador de propósito específico, o TIPO, que irá armazenar qual o tipo desse processador. Se nesse registrador contiver o valor zero, significa que o seu tipo é para receber informações, e se conter o valor 1, o seu tipo é para enviar informações. Também sofreu alteração a porta de entrada E1 e a porta de saída S1, que antes possuíam 16 *bits* de tamanho e agora possuem 20 *bits* pelo fato do TIPO a partir de agora, passar a integrar a informação que trafega de nó em nó. Isso visa a informar qual o tipo que o nó que acabou de receber a informação deve assumir. Desse modo, o tipo desse nó receptor passa a ser de emissor, dando continuidade ao ciclo de envio da informação.

Neste tópico, inicialmente essa implementação foi realizada na topologia Anel Unidirecional, de modo que nela haja a verificação se o próximo nó poderá receber a informação ou não.

Com relação ao volume ao tipo de informação que irá trafegar, agora existem dois formatos de informações diferentes. A primeira é a que já vem sendo utilizada, que é o principal “pacote” de informação, que além de conter o endereço do próximo nó a receber a informação, o seu destino e o dado, passa a também ter o tipo que o processador deverá assumir ao receber a informação, para que dessa forma, cada processador mude o seu tipo, de recebimento para envio, conforme ilustra a figura 4.24.

FIGURA 4.24 – Formato da mensagem – Versão 04

O segundo formato de informação diz respeito à requisição que cada nó que estiver prestes a enviar a informação realiza ao próximo, para verificar se o mesmo poderá receber a informação. Essa requisição possui o seguinte formato, conforme a figura 4.25.

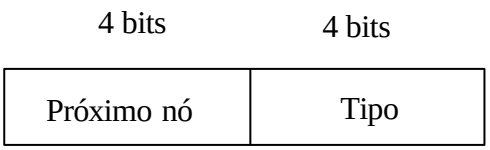


FIGURA 4.25 – Formato da requisição

4.5.2 Conjunto de instruções (ISA) – Caminho alternativo

Nesta versão, para dar suporte a essa nova implementação que visa à busca de um caminho alternativo no caso da impossibilidade de enviar um dado a um determinado nó, foram criadas instruções específicas para a realização dessa tarefa.

A tabela 4.11 traz as instruções utilizadas nesse processador, classificando-as quanto a sua finalidade e destacando após a tabela, aquelas que foram criadas e alteradas para essa versão.

TABELA 4.11 – Instruções utilizadas no ToPSeC– Versão 04

Desvio

JMPNXT	Desvia se o próximo nó estiver pronto para receber a informação

Comparação

CMPALT	Compara o vizinho alternativo caso o próximo não puder receber a informação

Envio

SNDNXT	Envia requisição ao próximo nó	**AWRBFR**	Responde a requisição enviada

Igualmente à versão 3, nessa nova versão foram criadas quatro novas instruções, sendo uma de desvio (JMPNXT), uma de comparação (CMPALT) e duas de envio (SNDNXT e AWRBFR).

Primeiramente, a instrução AWRBFR figura no processador que assume o papel de receptor, respondendo as requisições enviadas para verificar se este nó poderá receber a informação. No processador que irá enviar a informação, a instrução SNDNXT envia a requisição ao nó que ele for enviar a informação. A resposta desse nó é verificada pela instrução JMPNXT, que caso a resposta do próximo nó, que foi requisitado a receber a informação seja afirmativa, desvia o envio da mesma. Caso a resposta seja negativa, é executada a instrução CMPALT, que busca um nó alternativo. No caso da topologia Anel Unidirecional, é o próximo nó a esquerda, pela disposição dessa arquitetura.

4.5.3 Programa teste e Funcionamento – Caminho alternativo

Esta versão se resume pela busca de um caminho alternativo para o envio da informação caso o próximo nó não esteja disponível. A escolha por um novo vizinho para que seja feito o envio da informação acontece após uma verificação de disponibilidade do mesmo. A figura 4.26 ilustra o funcionamento dessa busca.

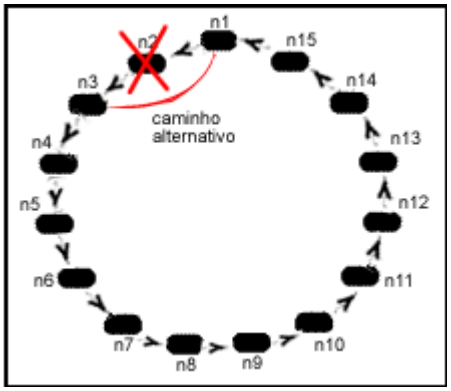


FIGURA 4.26 – Busca pelo caminho alternativo na topologia Anel Unidirecional

Para esta versão, em que há a busca de um caminho alternativo na topologia Anel Unidirecional, houve alterações de arquitetura, como a inclusão de mais um registrador específico, com a função de informar o "tipo" do processador e foi aumentada uma porta de entrada e uma de saída, visando justamente o tráfego dos dados já descritos na sessão 4.4.1.

O dado "tipo", visa especificar a função do nó atual, que poderá ser emissor ou receptor, e sempre é iniciado como receptor. Esse "tipo" é informado ao nó atual pelo nó anterior que enviou a informação. Assim, ele passaria de receptor a emissor, dando seqüência no envio da informação até o seu destino.

A tabela 4.12 traz as instruções *assembly* da versão 4 juntamente com as instruções para escolha de topologia.

TABELA 4.12 – Instruções *Assembly* do ToPSeC Versões 4

	ToPSeC – V4	Ciclos
0	TOPO 1/2	2
1	LID 2,12	2
2	AWRBFR	2
3	ENT	2
4	JDEST 23	2
5	JMPARBN 8	2
6	CMPLN	3

continuação		Ciclos
7	JMPANUN 18	2
8	JMPDSI 10	3
9	JMPDB 12	3
10	CMPSN	3
11	JMPN 18	2
12	JMPDIA 14	3
13	JMPDDA 16	3

continuação		Ciclos
14	CMPBN	3
15	JMPN 18	2
16	CMPBN	3
17	JMPN 18	2
18	SNDNXT	2
19	JMPNXT 22	2

continuação		Ciclos
20	CMPALT	3
21	JUMP 18	2
22	SEND	2
23	HALT	2
Total		57

Essa funcionalidade para escolher um caminho alternativo ocorre no momento antes do envio da informação. Na topologia Anel Unidirecional, o nó que irá enviar a informação, após ter localizado o endereço do próximo nó, envia uma requisição através da instrução SNDNXT para verificar se ele poderá receber a informação. O nó requisitado irá responder através da instrução AWRBFR. Caso a resposta for afirmativa, então há um desvio para a instrução SEND para realizar o envio. Caso contrário, a instrução CMPALT localiza o próximo nó após o nó requisitado e novamente envia uma requisição, mantendo o ciclo da verificação se ele poderá receber ou não a informação.

4.5.4 Simulações do Caminho Alternativo

Neste tópico, a simulação é referente ao funcionamento do programa que faz a verificação da disponibilidade do próximo nó a receber a informação. Este poderá ter dois rumos: estar disponível, onde então seria enviada a informação normalmente, ou estar indisponível, havendo a necessidade do envio a um outro nó, dando seqüência no funcionamento da topologia escolhida.

Primeiramente, a simulação apresentada mostra a requisição ao próximo nó para verificar se o mesmo poderá receber a informação, que é feita através da instrução SNDNXT, sendo para isso, utilizada a porta de saída S2.

A requisição é composta do endereço do nó que receberá a requisição e o tipo do emissor (1). Com desse envio, o nó requisitado irá verificar se a informação realmente é para ele e se o nó que o está requisitando está com o seu tipo como “emitir”. A instrução SNDNXT da figura 4.27 ilustra o envio da requisição.

A resposta do nó requisitado chega pela porta de entrada E2 (2), e então é verificado se ele poderá receber a informação. Essa verificação é feita analisando a resposta, que é composta do endereço do nó requisitado e o seu tipo. Se o tipo for zero, ele é um receptor e poderá receber a informação. Então é feito um desvio (3 e 4) para a instrução SEND, que enviará a informação pela porta de saída S1 (5), que a partir desta versão, passa a ter o

tamanho de 20 *bits* e possuir também o tipo do nó emissor, que deverá ser assumido pelo próximo nó a receber a informação.

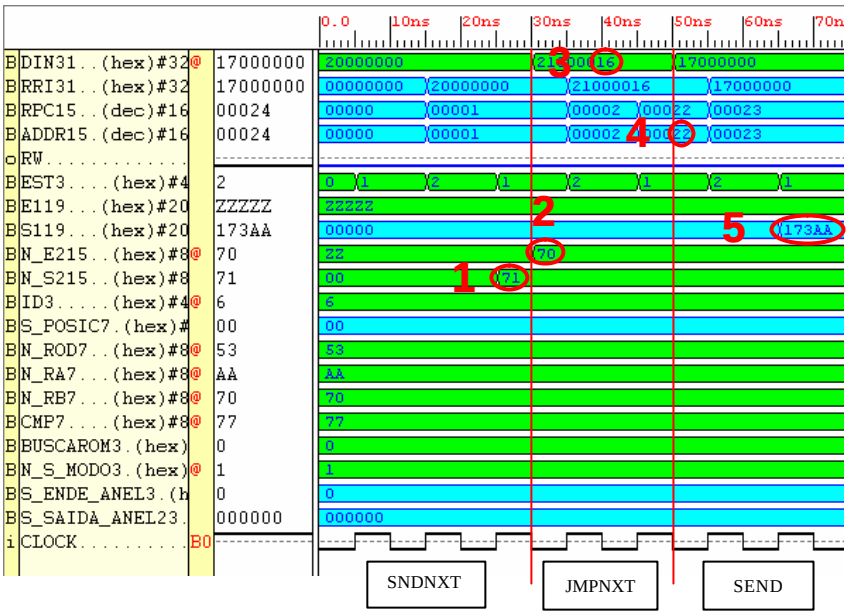


FIGURA 4.27 – Simulação de envio de requisição e resposta

A simulação mostrada na figura 4.27 corresponde principalmente às instruções SNDNXT e JMPNXT, descritas em VHDL na seguinte estrutura:

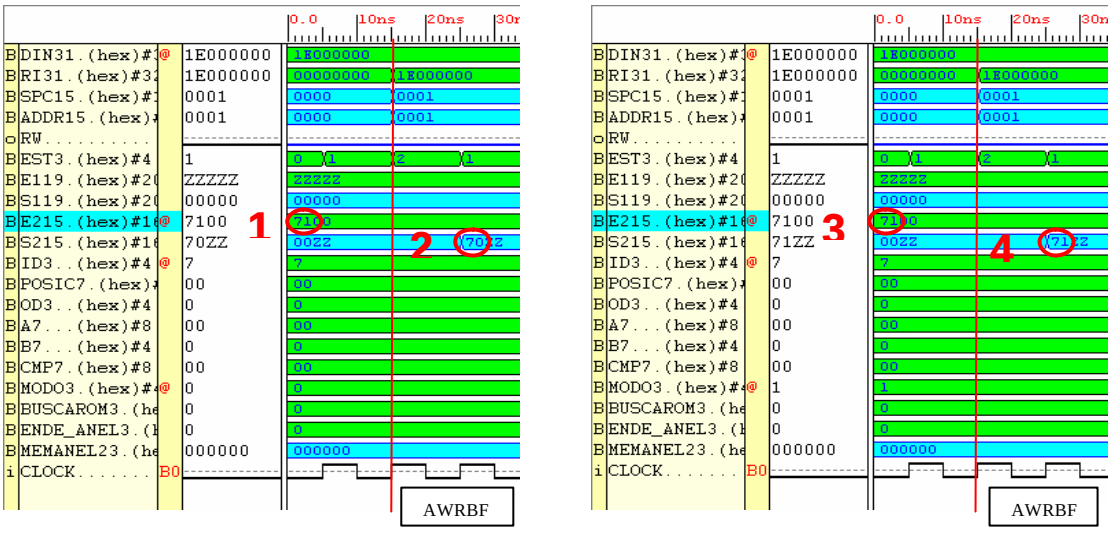
```
WHEN INSTR_SNDNXT =>
      S2(15 DOWNTO 12) <= B(7 DOWNTO 4);
      S2(11 DOWNTO 8) <= TIPO(3 DOWNTO 0);
      EST <= BUSCA;

WHEN INSTR_JMPNXT =>
      IF E2(11 DOWNTO 8) = "0000" THEN
            PC(7 DOWNTO 0) <= RI(7 DOWNTO 0);
            RW <= '0';
      END IF;
      EST <= BUSCA;
```

O próximo nó, ao receber a requisição pela porta de entrada E1 (1 e 3), irá responder se está disponível ou não para receber a informação através da instrução AWRBFR, conforme a figura 4.28.

Caso o tipo do processador for receptor, a resposta pela porta de saída S2 será o seu número identificador e o seu tipo que será zero, ou seja, receptor (2). Caso contrário, isso significa que seu tipo é emissor e ele não poderá receber a informação, respondendo pela

porta de saída S2 o seu número identificador e o seu tipo, que neste caso é um, ou seja, emissor (4).



Resposta afirmativa **Resposta negativa**

FIGURA 4.28 – Simulação da resposta no próximo nó

A simulação mostrada na figura 4.28 corresponde principalmente à instrução AWRBFR, descritas em VHDL na seguinte estrutura:

```
      WHEN INSTR_AWRBFR =>
            IF ID(3 DOWNTO 0) = E2(15 DOWNTO 12) AND E2 (11 DOWNTO 8) /=
TIPO (3 DOWNTO 0) THEN
                  S2(15 DOWNTO 12) <= ID(3 DOWNTO 0);
                  S2 (11 DOWNTO 8) <= TIPO (3 DOWNTO 0);
            ELSIF ID (3 DOWNTO 0) = E2 (15 DOWNTO 12) AND E2 (11 DOWNTO 8)
= TIPO (3 DOWNTO 0) THEN
                  S2 (15 DOWNTO 12) <= ID (3 DOWNTO 0);
                  S2 (11 DOWNTO 8) <= "0001";
            END IF;
            EST <= BUSCA;
```

No caso da resposta do nó requisitado ser de que o seu tipo é de emissor, significa que ele não poderá receber a informação. Então o nó atual irá procurar o próximo nó perante a topologia atual, que neste caso é a Anel Unidirecional, e então será o próximo nó à esquerda (2 e 3) após o nó requisitado (1). Essa verificação é feita através da instrução CMPALT em caso negativo, conforme ilustra a figura 4.29.

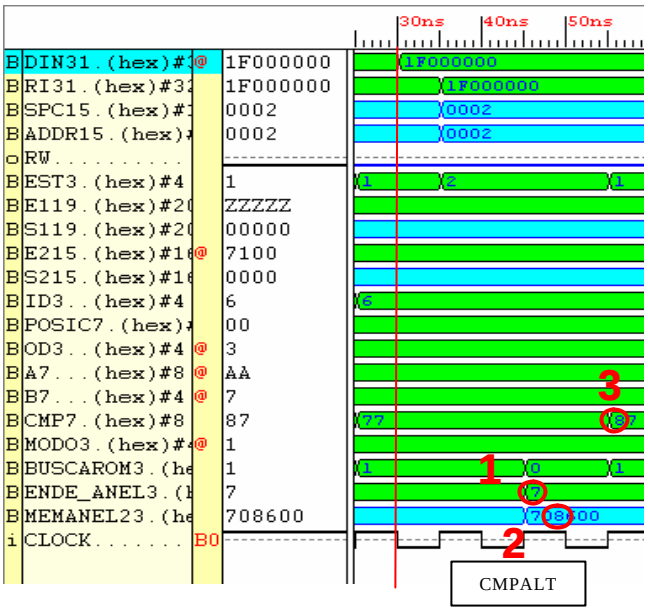


FIGURA 4.29 – Simulação do caminho alternativo – Topologia Anel Unidirecional

A simulação mostrada na figura 4.29 corresponde principalmente à instrução CMPALT, descritas em VHDL na seguinte estrutura:

```
WHEN INSTR_CMPALT =>
      IF BUSCAROM(3 DOWNTO 0) = "0001" THEN
            ENDE_ANEL <= CONV_INTEGER(ID(3 DOWNTO 0) + '1');
            BUSCAROM(3 DOWNTO 0) <= "0000";
      ELSE
            CMP(7 DOWNTO 4) <= SAIDA_ANEL(15 DOWNTO 12);
            CMP(3 DOWNTO 0) <= CMP(7 DOWNTO 4);
            EST <= BUSCA;
            BUSCAROM(3 DOWNTO 0) <= "0001";
      END IF;
```

A implementação dessa versão visa o funcionamento do ToPSeC, realizando uma verificação se o próximo nó a receber a informação irá recebe-la. Desse modo, o funcionamento da rede não sofre interrupção, buscando outro caminho. Essa implementação foi motivada pela necessidade de se buscar caminhos alternativos nos casos em que um nó da rede se tornar indisponível. Um exemplo real em que um nó poderia se tornar indisponível é nos casos de ataques. Nesta situação, o nó que fosse atacado é isolado do mapeamento da topologia, deixando de fazer parte da rede. Isso poderia assegurar que o atacante não tenha acesso às informações trafegadas na rede, pois esse nó não receberá nenhuma informação pois estará fora do mapeamento, isolando o problema e não interrompendo a comunicação dos dados pelos outros nós da rede.

4.5.5 Estatísticas de Implementação e Desempenho – Caminho alternativo

A tabela 4.13 mostra as estatísticas desses modelos de FPGAs através da ferramenta Xilinx 3.1 (BOURT, 1998).

TABELA 4.13 – Estatísticas com parâmetros espaciais e temporais da Versão 4

Spartan2 S50FG256										
Slices (CLBs)			**Flip-Flop**			**Luts de 4 entradas**			**Tempo de Propagação**	
Total	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Tp (ns)**	**Freq. Mhz**
135	768	17	122	1.536	7	216	1.536	14	16,391	61,009
Virtex V600FG676										
Slices (CLBs)			**Flip-Flop**			**Luts de 4 entradas**			**Tempo de Propagação**	
Total	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Tp (ns)**	**Freq. Mhz**
135	6.912	1	122	13.824	1	216	13.824	1	17,536	57,026

Observando a tabela 4.13 é possível verificar que em relação à versão 3, a utilização do *chip* continua em uma porcentagem pequena no modelos de FPGA utilizados. Isso permite a inclusão de novos recursos sem a necessidade da troca de modelo ou inclusão de recursos físicos adicionais.

As medidas de propagação e freqüência no circuito no FPGA Spartan2 são respectivamente 16,391 ns e 61,009 Mhz, e no FPGA Virtex são respectivamente 17,536 ns e 57,026 Mhz.

Igualmente à maioria das comparações das implementações das versões anteriores, percebe-se que apesar do FPGA Spartan2 possuir menos recursos e menor espaço para mapeamento interno no *chip* do que no Virtex, o projeto neste ponto fica mais veloz no Spartan2 pelo fato do tempo de propagação ser menor; com isso, a freqüência do processador aumenta em relação a Virtex.

Também se percebe essa diferença analisando os números de Flip-Flops e Luts de 4 entradas, onde as quantidades nos dois modelos de FPGA são iguais, mas a ocupação é maior no Virtex, principalmente em relação aos Luts de 4 entradas, influenciando diretamente no tempo de propagação e freqüência do processador. Essa comparação é semelhante a da versão inicial, onde a simplicidade e o tamanho do projeto são mais bem adaptados no modelo mais simples dentre os dois utilizados.

Os *Floor Plans* são mostradas abaixo nas figuras 4.30 e 4.31, que mostram a alocação dos componentes do código em VHDL respectivamente nos *chips* Spartan2 2S50FG256 e Virtex V600FG676, ajudam a mostrar os parâmetros espaciais e temporais apresentados na tabela 4.13, indicando a ocupação do chip nos modelos utilizados.

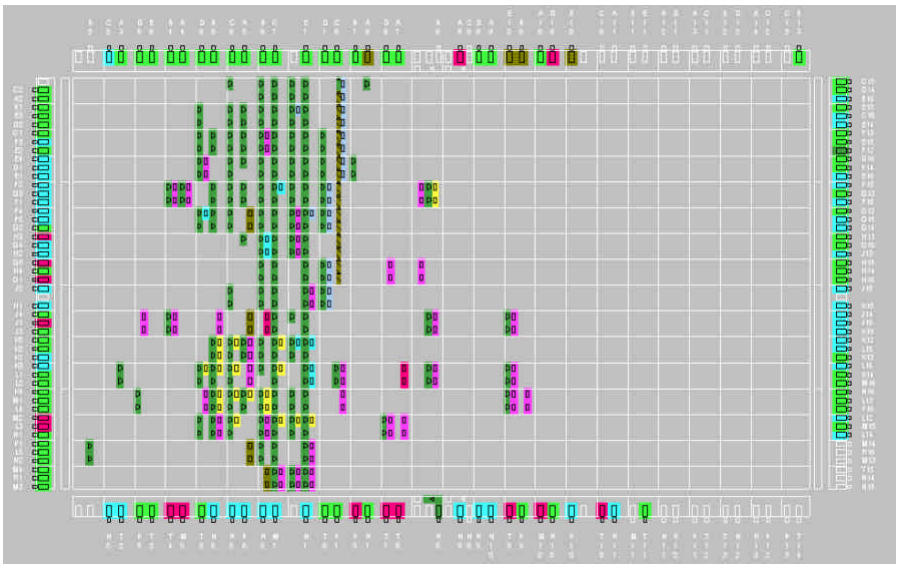


FIGURA 4.30 – *Floor Plan* Spartan2 S50FG256 – Caminho alternativo na Topologia Anel Unidirecional

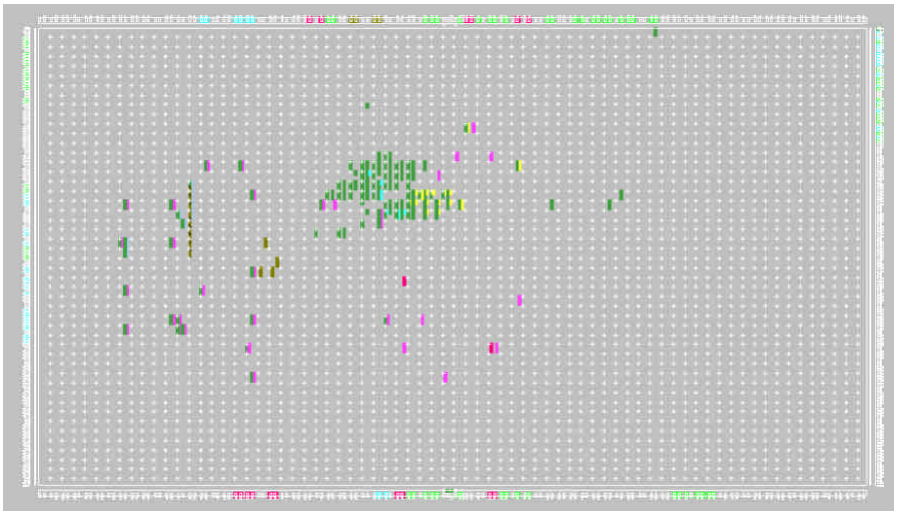


FIGURA 4.31 – *Floor Plan* Virtex V600FG676 – Caminho alternativo na Topologia Anel Unidirecional

Conforme já descrito em análises de *Floor Plans* anteriores, as figuras 4.30 e 4.31 demonstram por ser uma implementação simples, o fator do tamanho interno do chip influencia diretamente no tempo de propagação e, conseqüentemente, na velocidade do processador, que nesta versão também é mais veloz no modelo mais simples de FPGA escolhido.

4.6 ToPSeC – Primitivas de segurança

Neste tópico, será abordada a questão da segurança através da criptografia. Também será tratada a sua implementação nesta última versão implementada.

Existem vários algoritmos que poderiam ser utilizados, mas por questões abordadas no capítulo 3 como segurança forte, projeto simples, bom desempenho e sem patente, os algoritmos escolhidos são o DES e AES.

Com a adição desses dois algoritmos de criptografia visando oferecer segurança na informação trafegada na rede, algumas alterações no projeto foram feitas para suportar a inserção de tais algoritmos.

Dessa forma, três novas instruções foram criadas. As instruções CRIPTA e DECRIPTA foram adicionadas para realizar a cifragem e decifragem da informação. Essas duas instruções foram feitas seguindo a implementação realizada em (OLIVEIRA, 2006), onde também houve a inclusão dos algoritmos de criptografia DES e AES para que fosse comparado o desempenho do PERS com e sem criptografia.

A instrução INF também foi adicionada para simular o momento do envio inicial pelo nó origem, sendo utilizada logo após a instrução CRIPTA e no momento do recebimento da informação pelo nó destino, após a verificação da instrução JDEST, que se utiliza a instrução DECRIPTA.

Neste tópico é abordado a inclusão da funcionalidade de criptografia no ToPSeC. Com essa nova funcionalidade, o ToPSeC será capaz de mascarar o dado que irá trafegar na rede, e apenas o destino conseguirá ler esse dado.

Essa alteração ocasionou mudanças na arquitetura do ToPSeC e também foi necessária a utilização de um novo programa para teste e a sua simulação.

4.6.1 Arquitetura do ToPSeC – Primitivas de segurança

Primeiramente, em relação à mudança de arquitetura, as alterações sofridas pela implementação do algoritmo DES, foram o aumento de tamanho da porta de entrada E1 e da porta de saída S1, de 20 bits para 76 bits. O registrador A também sofreu uma alteração no seu tamanho, passando de 8 para 64 bits e foi incluído um módulo para realizar a criptografia, e um novo estado, que refere a este módulo quando a cifragem ou decifragem estiver em execução.

Já com relação à mudança de arquitetura referente às alterações sofridas pela implementação do algoritmo AES, foi o aumento de tamanho da porta de entrada E1 e da porta de saída S1, de 20 bits para 140 bits. O registrador A também sofreu uma alteração no seu tamanho, passando de 8 para 128 bits e, nesta implementação, foi alterado o módulo para realizar a criptografia com o algoritmo AES e permanece o estado, que refere a este módulo quando estiver em execução de cifragem ou decifragem.

Quanto ao tamanho da informação que a partir de agora irá trafegar na rede, esta também sofreu alteração. A figura 4.32 mostra os formatos da informação trafegada nos dois algoritmos de criptografia propostos.

Apenas lembrando, a inclusão dos algoritmos de criptografia DES e AES foi feita separadamente. As instruções de cifragem e decifragem foram feitas de maneira que fossem genéricas, mas o seu conteúdo, assim como o tamanho do dado cifrado muda para cada implementação.

FIGURA 4.32 – Formato da mensagem cifrada

Desse modo, o conjunto de instruções e o programa *assembly* são os mesmos para ambos, mas pode-se verificar nas simulações, nas estatísticas de implementação e desempenho e no *floor plan* que são diferentes.

4.6.2 Conjunto de instruções (ISA) – Primitivas de segurança

Neste ponto, o conjunto de instruções também sofreu alterações. Inicialmente, devido à inclusão de um novo módulo para criptografia, duas novas instruções para realizar a

cifragem e decifragem do dado foram criadas, sendo elas as instruções CRIPTA e DECRIPTA.

Mas como a cifragem só ocorre no momento do envio da informação, também foi criada a instrução INF para informar o identificador do nó atual, o identificador do nó destino e a informação, que a partir de agora, passa a ser criptografada.

A tabela 4.14 traz o conjunto de instruções atualizado.

TABELA 4.14 – Instruções utilizadas no ToPSeC – Versão 05

Movimentação

INF	Armazena o nó atual, o destino e o dado

Criptografia

CRIPTA	Cifra o dado	**DECRIPTA**	Decifra o dado

Nota-se que as instruções CRIPTA e DECRIPTA são genéricas, não sendo informada para qual algoritmo de criptografia corresponde cada uma. Apenas uma ressalva, em que a instrução DECRIPTA para o algoritmo AES não está implementada.

4.6.3 Programa Teste e Funcionamento – Primitivas de Segurança

Para esta versão, são utilizados dois programas em *assembly*. O programa de recebimento da informação já existente nas outras versões recebeu a instrução DECRIPTA, para realizar a decifragem no caso de a informação ser para o nó atual, e também foi criado um programa para o envio inicial da informação a partir de um nó qualquer. Esse programa em *assembly* foi criado utilizando as instruções já existentes no processador juntamente com a instrução INF, para informar a origem, o destino e o dado a ser enviado, e também a instrução CRIPTA, para prover uma correta escolha inicial de nó para o envio da informação. Neste ponto, o conjunto caso específico de criptografia, a execução da rotina para cifragem do dado, é idêntico nas duas topologias propostas, Anel Unidirecional e Árvore Binária.

TABELA 4.15 – Instruções *Assembly* do ToPSeC para o recebimento e envio dos dados cifrados nas topologias Anel Unidirecional e Árvore Binária

	Recebimento do dado	Ciclos
0	TOPO 1/2	2
1	LID 2,12	2
2	AWRBFR	2
3	ENT	2
4	JDEST 24	2
5	JMPARBN 8	2
6	CMPLN	3
7	JMPANUN 18	2
8	JMPDSI 10	3
9	JMPDB 11	3
10	CMPSN	3
11	JMPN 18	2
12	JMPDIA 14	3
13	JMPDDA 16	3
14	CMPBN	3
15	JMPN 18	2
16	CMPBN	3
17	JMPN 18	2
18	SNDNXT	2
19	JMPNXT 22	2
20	CMPALT	3
21	JUMP 18	2
22	SEND	2
23	HALT	2
24	DECRIPTA	16 / 14
25	JUMP 23	2
Total		75 / 73

	Envio do dado	Ciclos
0	TOPO 1/2	2
1	INF 5AAA	2
2	CRIPTA	16 / 14
3	AWRBFR	2
4	JMPARBN 7	2
5	CMPLN	3
6	JMPANUN 17	2
7	JMPDSI 9	3
8	JMPDB 10	3
9	CMPSN	3
10	JMPN 17	2
11	JMPDIA 13	3
12	JMPDDA 15	3
13	CMPBN	3
14	JMPN 17	2
15	CMPBN	3
16	JMPN 17	2
17	SNDNXT	2
18	JMPNXT 21	2
19	CMPALT	3
20	JUMP 17	2
21	SEND	2
22	HALT	2
Total		69 / 67

4.6.4 Simulações das Primitivas de segurança

Antes de demonstrar as simulações é importante ressaltar que, para que a criptografia acontecesse, foi necessária uma inclusão de mais um estado, desse modo. A inclusão desse estado se fez necessária para que os algoritmos DES e AES executassem as 16 e 10 iterações do processo de cifragem do dado.

Os trechos de código seguintes são respectivamente os estados incluídos dos algoritmos DES e AES neste processador.

```
WHEN CRIPTO => -- ESTADO DE CRIPTOGRAFIA (DES)
RODADA <= RODADA + 1;
IF RODADA = "1111" THEN
      RODADA <= "0000";
      IF MODO = '1' THEN
            A(63 DOWNTO 0) <= CIFRA;
      ELSIF MODO='0' THEN
```

```
            A <= CIFRA(63 DOWNTO 0);
      END IF;
      EST <= BUSCA;
END IF;

WHEN CRIPTO => -- ESTADO DE CRIPTOGRAFIA (AES)
RODADA <= RODADA + 1;
IF RODADA = "1101" THEN
      RODADA <= "0000";
      IF MODO='1' THEN
            A(127 DOWNTO 0) <= CIFRA;
      ELSIF MODO='0' THEN
            A <= CIFRA(127 DOWNTO 0);
      END IF;
      EST <= BUSCA;
END IF;
```

Na figura 4.33 observa-se a cifragem do dado através do algoritmo de criptografia DES. Primeiramente é informado o dado a ser cifrado (1 e 2).

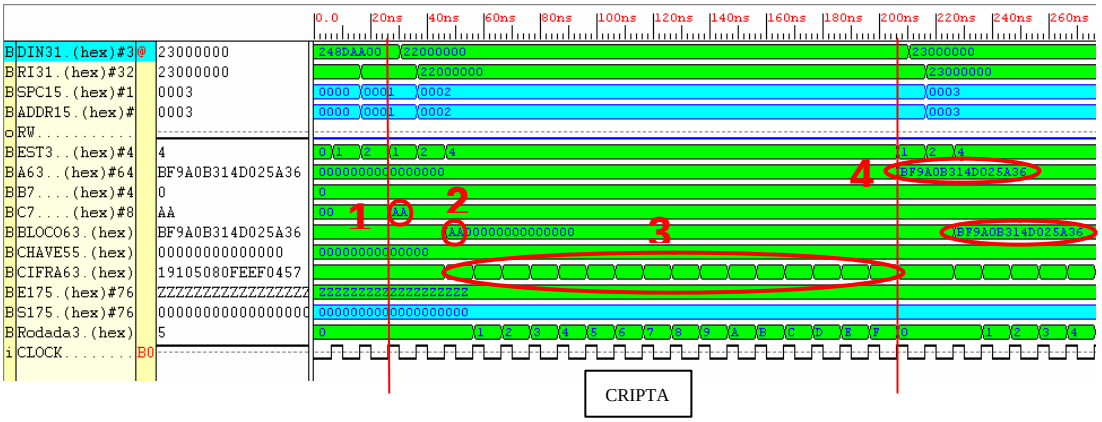


FIGURA 4.33 – Cifragem do dado (DES)

Após ser informado o destino do dado, a instrução CRIPTA realiza 16 iterações (3) para cifrar o dado contido no registrador C (1), transformando em informação de 64 *bits*, que é atribuída ao registrador A (4).

A simulação mostrada na figura 4.33 corresponde principalmente à instrução CRIPTA, implementadas em VHDL na seguinte estrutura:

```
WHEN INSTR_CRIPTA =>
      CHAVE <= "00000000000000000000000000000000000000000000000
                  00000000000000";
      MODO <='1';
      BLOCO <= C & PAD;
      EST <= CRIPTO;
```

Sequencialmente, a figura 4.34 mostra a execução da instrução DECRIPTA da versão implementada com o algoritmo de criptografia DES.

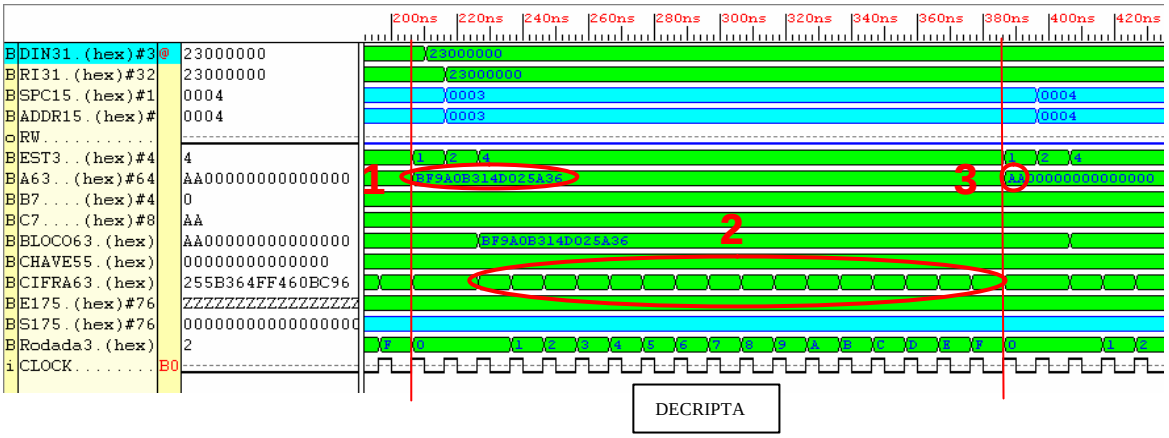


FIGURA 4.34 – Decifragem do dado (DES)

O processo de decifragem do dado criptografado de 64 *bits* contido no registrador A (1), após 16 iterações (2) pode ser observado novamente no registrador A (3).

A simulação mostrada na figura 4.34 corresponde principalmente à instrução DECRIPTA, implementadas em VHDL na seguinte estrutura:

```
WHEN INSTR_DECRIPTA =>
      MODO <='0';
      BLOCO <= A(63 downto 0);
      EST <= CRIPTO;
```

Na figura 4.35 também se observa a cifragem do dado, mas agora com o algoritmo de criptografia AES. Após ser informado o destino o dado, a instrução CRIPTA realiza 10 iterações (3) para cifrar o dado contido no registrador C para o registrador A (1 e 2), que após esse processo, aparece cifrado em um bloco de 128 *bits* (4).

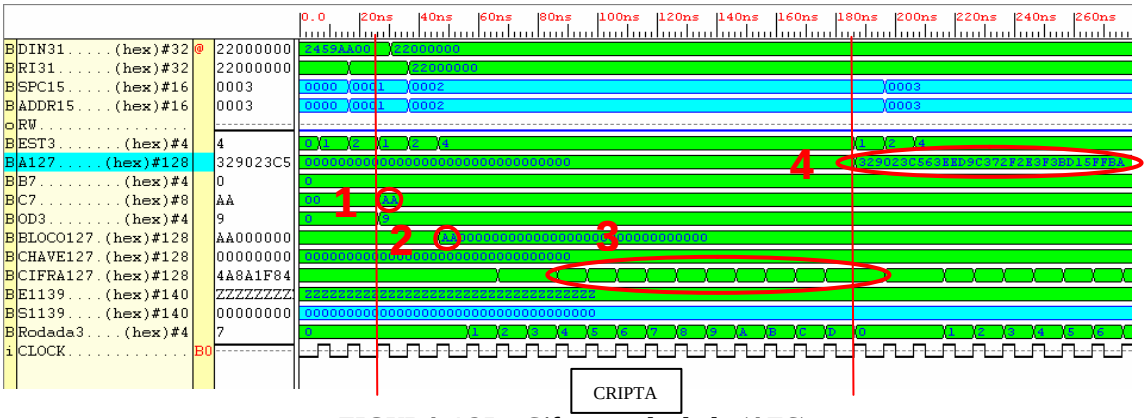


FIGURA 4.35 – Cifragem do dado (AES)

A simulação mostrada na figura 4.35 corresponde principalmente à instrução DECRIPTA, implementadas em VHDL na seguinte estrutura:

```
WHEN INSTR_CRIPTA =>
      RESET <= '0';
      CHAVE <="00000000000000000000000000000000000000000000"&
              "00000000000000000000000000000000000000000000"&
              "00000000000000000000000000000000000000000000";
      MODO <='1';
      BLOCO <= C & PAD;
      EST <= CRIPTO;
```

4.6.5 Estatísticas de Implementação e Desempenho – Primitivas de segurança

Após a finalização das simulações das primitivas de segurança, o ToPSeC foi implementado em dois modelos de FPGA para realizar um comparativo de implementação de desempenho. A tabela 4.16 mostra as estatísticas desses modelos de FPGAs através da ferramenta Xilinx 3.1 (BOURT, 1998).

A partir desta versão que inclui primitivas de segurança com os algoritmos de criptografia DES e AES, não foi possível continuar utilizando o FPGA Spartan2 S50FG256.

A implementação com os algoritmos de criptografia DES e AES é muito grande para esse modelo, sendo utilizado então os modelos de FPGA Virtex V600FG676 que já haviam sendo utilizados e a partir desta versão, o FPGA Virtexe V1000EFG680.

TABELA 4.16 – Estatísticas com parâmetros espaciais e temporais, com o algoritmo DES de criptografia

Virtex V600FG676										
Slices (CLBs)			**Flip-Flop**			**Luts de 4 entradas**			**Tempo de Propagação**	
Total	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Tp (ns)**	**Freq. Mhz**
379	6.912	5	64	13.824	1	752	13.824	5	24,798	40,035
Virtexe V1000EFG680										
Slices (CLBs)			**Flip-Flop**			**Luts de 4 entradas**			**Tempo de Propagação**	
Total	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Tp (ns)**	**Freq. Mhz**
379	12.288	3	64	24.576	1	752	24.576	3	21,036	47,538

A tabela 4.16 contém os dados estatísticos de ocupação e desempenho do processador utilizando o algoritmo de criptografia DES, onde é possível verificar que com a utilização desses modelos de FPGAs, é utilizados uma porcentagem relativamente alta se for comparado

com as versões anteriores; porém, esses modelos ainda possuem espaço para a inclusão de novos recursos no processador.

As medidas de propagação e freqüência do circuito com a implementação do algoritmo DES no FPGA Virtex são respectivamente 24,768 ns e 40,035 Mhz, e no FPGA Virtexe são respectivamente 21,036 ns e 47,538 Mhz.

Com esses números percebe-se que com o aumento da lógica utilizada nessa nova implementação devido à utilização de um algoritmo de criptografia, o melhor modelo de FPGA, no caso o Virtexe, possui um desempenho melhor em relação ao Virtex, mostrando que a partir do momento que aumenta a utilização da área do *chip* com uma lógica mais elaborada. O modelo de FPGA Virtexe se mostra mais favorável nessa utilização, inclusive pela ocupação demonstrada na tabela 4.16 que é praticamente duas vezes maior no Virtexe. No entanto ele ainda assim apresenta um desempenho melhor dentre os dois modelos utilizados.

Os *Floor Plans* das duas implementações do processador para o DES são mostrados abaixo nas figuras 4.36 e 4.37, que demonstram a alocação dos componentes do código em VHDL respectivamente nos *chip* Virtex V600FG676 e Virtexe V1000EFG680, ajudam a demonstrar os parâmetros espaciais e temporais apresentados na tabela 4.16, indicando a ocupação do chip nos modelos utilizados.

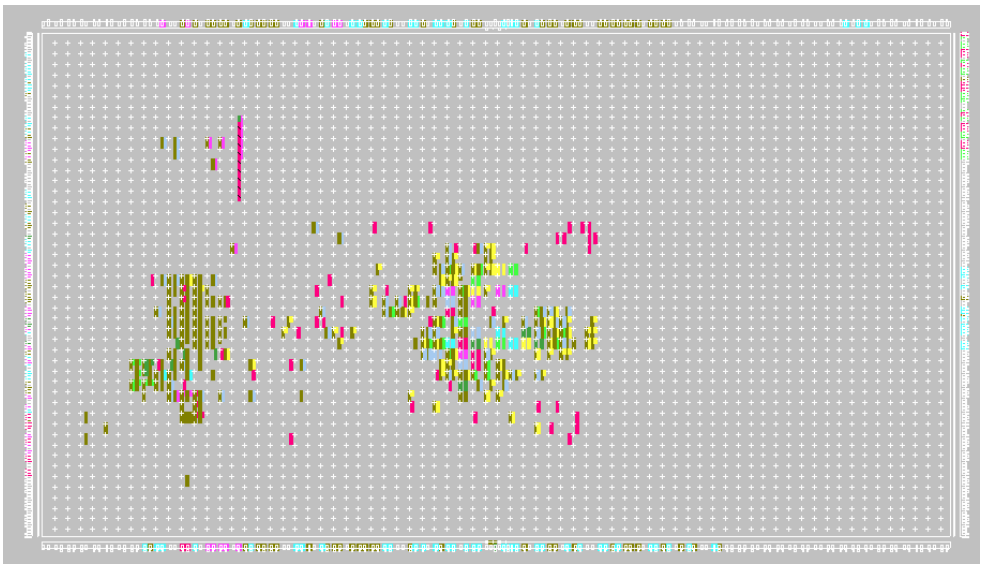


FIGURA 4.36 – *Floor Plan* Virtex V600FG676 – Primitiva de segurança com algoritmo DES

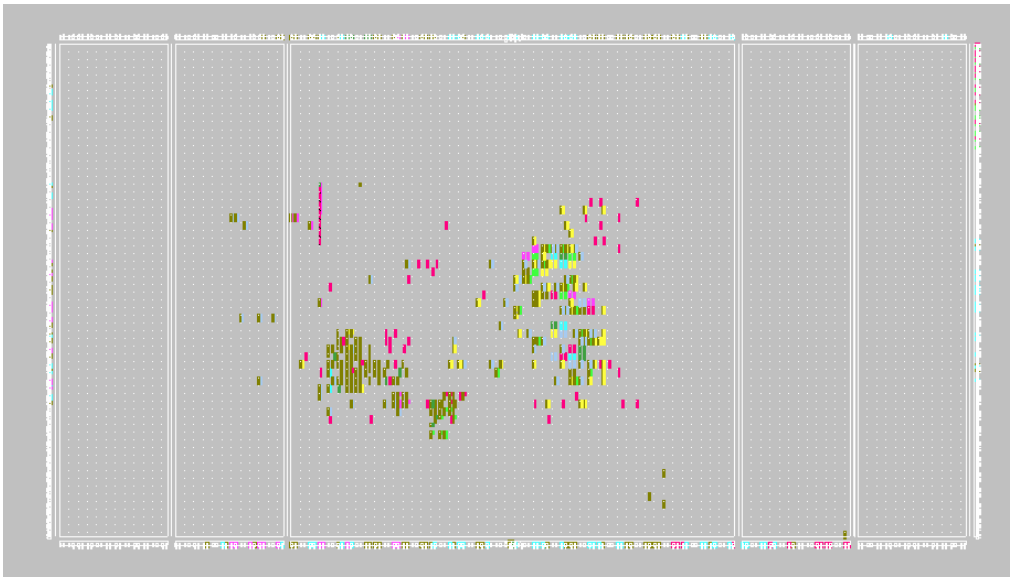


FIGURA 4.37 – *Floor Plan* Virtexe V1000EFG680 – Primitiva de segurança com algoritmo DES

A tabela 4.16 traz as estatísticas de implementação e desempenho do processador utilizando o algoritmo de criptografia DES. Agora, a tabela 4.17 traz os dados com a implementação do algoritmo AES.

Olhando a tabela 4.17 é possível verificar que, ao contrario da implementação do algoritmo DES, nesta implementação é utilizada uma porcentagem maior que a anterior é relativamente grande. Isso se deve à utilização do algoritmo de criptografia AES, que possui uma lógica que demanda maior espaço do *chip* e, conseqüentemente, utiliza mais recursos internos. Mas, com essa implementação ainda é possível a inclusão de novos recursos pelo fato dos modelos de FPGAs utilizados possuírem uma grande área de mapeamento.

TABELA 4.17 – Estatísticas com parâmetros espaciais e temporais, com o algoritmo AES de criptografia

Virtex V600FG676										
Slices (CLBs)			Flip-Flop			Luts de 4 entradas			Tempo de Propagação	
Total	Ocupado	%	Total	Ocupado	%	Total	Ocupado	%	Tp (ns)	Freq. Mhz
2.390	6.912	34	774	13.824	5	4,559	13.824	5	28,652	34,902
Virtexe V1000EFG680										
Slices (CLBs)			Flip-Flop			Luts de 4 entradas			Tempo de Propagação	
Total	Ocupado	%	Total	Ocupado	%	Total	Ocupado	%	Tp (ns)	Freq. Mhz
2.389	12.288	19	774	24.576	3	4.557	24.576	18	23,214	43,077

As medidas de propagação e freqüência no circuito no FPGA Virtex são respectivamente 28,652 ns e 34,902 Mhz, e no FPGA Virtexe são respectivamente 23,214 ns e 43,077 Mhz.

Os *Floor Plans* das duas implementações do processador utilizando o AES são mostrados abaixo nas figuras 4.38 e 4.39, que demonstram a alocação dos componentes do código em VHDL respectivamente nos *chip* Virtex V600FG676 e Virtexe V1000EFG680, ajudam a visualizar os parâmetros espaciais e temporais apresentados na tabela 4.17, indicando a ocupação do chip nos modelos utilizados.

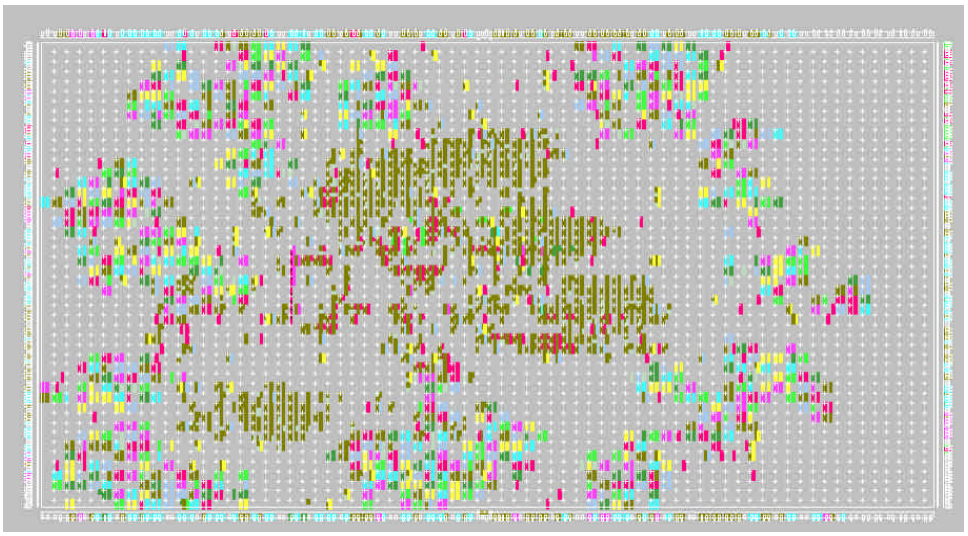


FIGURA 4.38 – *Floor Plan* Virtex V600FG676 – Primitiva de segurança com algoritmo AES

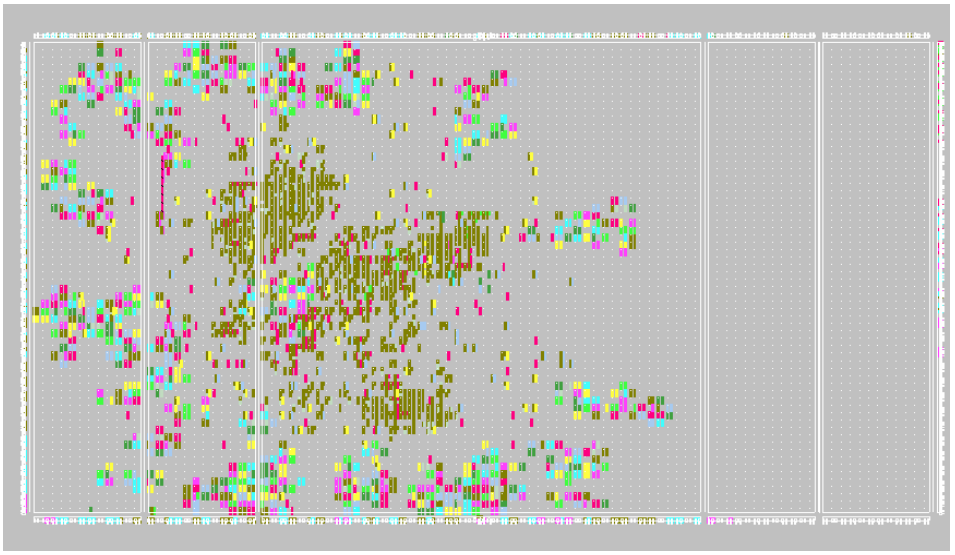


FIGURA 4.39 – *Floor Plan* Virtexe V1000EFG680 – Primitiva de segurança com algoritmo AES

Por fim, analisando as duas implementações que possuem os algoritmos de criptografia DES e AES, se percebe que a implementação com o DES é mais rápida em relação ao AES pelo fato de ser menor e, desta forma, utilizar uma lógica menor ocupar menos espaço e recursos do *chip*.

Mas se for considerado que o algoritmo DES cifra blocos de informação de 64 *bits* e o AES cifra blocos de 128 *bits*, e levando-se em conta que para acompanhar o funcionamento dos algoritmos de criptografia, a arquitetura do processador também sofre alterações estruturais, como aumento do tamanho dos registradores e portas de entrada e saída que irão utilizar essa informação cifrada. O AES se mostra uma alternativa de segurança interessante, já que trabalha com um bloco de informação cifrada duas vezes maior que o DES, possuindo um desempenho de 800 Mbits/seg. para um processo de cifragem com 10 iterações em 160 ns contra 355 Mbits/seg. do DES para processo de cifragem com 16 iterações em 180 ns para isso. No entanto, para a escolha do algoritmo de criptografia que será utilizado, deve-se pesar a qual tipo de aplicação será empregado o processador, verificando os requisitos de desempenho e segurança exigidos pela situação.

Capítulo 5

Conclusão

Acerca das atuais deficiências nas redes de computadores, foi feito um estudo neste trabalho, constatando que há a necessidade de redes mais rápidas e seguras.

Visando este foco, uma das alternativas encontradas para esses problemas é a utilização dos processadores de rede, pois esse tipo de equipamento possui um processamento específico para o roteamento de pacotes em uma rede, sendo que alguns modelos também oferecem algum tipo de segurança.

Atualmente existem vários modelos de processadores de rede, cada um com uma tecnologia, arquitetura e roteamento diferentes. Das tecnologias empregadas no desenvolvimento de processadores de rede, verificou-se que o FPGA é a mais adequada a este projeto por possuir baixo custo, bom desempenho, fácil correção e manutenção do projeto e principalmente, por permitir uma fácil prototipação, oferecendo uma situação real de análise de desempenho a esse projeto, por se tratar de um trabalho acadêmico.

Na questão da segurança, foi estudada a criptografia como forma de prover segurança em uma transmissão de uma rede. Dentre os vários algoritmos existentes, optou-se por utilizar o DES e o AES, por possuírem segurança forte, projeto simples, bom desempenho, serem livres e serem considerados padrões de segurança.

Com base nos estudos realizados, a proposta desse trabalho foi a criação de um processador em FPGA que ofereça suporte a duas topologias, sendo elas Anel Unidirecional e

Árvore Binária, a escolha em tempo real dentre essas topologias na transmissão de uma informação e a busca de um caminho alternativo dentro da topologia no caso da impossibilidade de recebimento da informação por um nó.

Também foram incorporados os algoritmos de criptografia DES e AES para oferecer segurança à informação trafegada.

Os resultados desse projeto foram satisfatórios, sendo que na versão inicial, que corresponde apenas à inclusão da topologia Anel Unidirecional, o processador possuía uma freqüência de 91,050 Mhz, e na versão 4 onde o processador, além de possuir a topologia Anel Unidirecional, também possui a topologia Árvore Binária, a escolha entre as duas topologias e a busca por um caminho alternativo na topologia Anel Unidirecional, o processador passou a ter uma freqüência de 61,009 Mhz no mesmo modelo de FPGA Spartan2 S50FG256.

Com a inclusão da criptografia, seguindo os passos de (OLIVEIRA 2006), foi necessária a troca de modelo de FPGA. A lógica empregada nos dois algoritmos de criptografia é bem maior do que a que estava sendo utilizados até então, passando o processador a ter uma freqüência de 40,035 Mhz utilizando o algoritmo DES e 34,902 Mhz utilizando o algoritmo AES no FPGA Virtex V600FG676.

Com isso, o impacto com a utilização de criptografia fica visível, pois mesmo alterando-se o modelo de *chip*, o desempenho do processador cai cerca de 20 Mhz entre a última versão e as implementações de criptografia, mas ainda assim, demonstrando um resultado satisfatório, se forem consideradas todas as funcionalidades implementadas.

Como trabalhos futuros, deseja-se implementar a busca pelo caminho alternativo na topologia Árvore Binária. Também deseja-se incluir outros algoritmos de criptografia, como RC5 e RSA, incluir mais topologias como Hipercubo, bem como o suporte para o seu funcionamento nesse processador e incluí-la na escolha em tempo real das topologias oferecidas por esse processador. Por fim, realizar a reconfiguração parcial, melhorando a funcionalidade da escolha de topologias.

Referências

AESWINNER, National Institute of Standards and Technology (NIST). Outubro. 2000. Disponível em: http://www.nist.gov/public_affairs/releases/g00-176.htm. Acesso em: 06 mai. 2006.

AGERE System, **The Challenge for Next Generation Network Processors**. Abril. 2001. Disponível em: www.agere.com/docs/challenge_new.pdf. Acesso em: 17 jul. 2006.

AGERE System, PayloadPlus NP, Family Product Brief, 2002.

AGERE System, **Building Next Generation Network Processos**. Abril. 2001. Disponível em: www.agere.com/docs/building_new.pdf. Acesso em: 17 jul. 2006.

AGERE System, **Advanced *PayloadPlus*® APP300**: Access Network Processors. 2005. Disponível em: www.agere.com/docs/pb05-039np.pdf. Acesso em: 17 jul. 2006.

ALLEN, J. R., BASS, B. M., BASSO, C., BOIVIE, R. H., CALVINAC, J. L., DAVIS, G. T., FRELECHOUX, L., HEDDES, M., HERKERSDORF, A., KIND, A., LOGAN, J. F., PEYRAVIAN, M., RINALDI, M. A., SABHIKHI, R. K., SIEGEL, M. S., WALDVOGEL, M., **IBM PowerNP Network Processor**: Hardware, Software and Applications. Maio. 2001. Disponível em: www.research.ibm.com/journal/rd/472/allen.pdf. Acesso em: 20 out. 2006.

BALPARDA, Daniel, **Criptografia Métodos e Algoritmos**. Rio de Janeiro: Editora Book Express, 2ª Edição, 2001.

BOURT, Dave Van den, **The Pratical Xilinx Designer Lab Book**. Editora Prentice-Hall, 1998.

CISCO, Cisco Systems. **Technology of Edge Aggregation**: Cisco 10000 Series Edge Services Router. Janeiro. 2002. Disponível em: www.cisco.com/univercd/cc/td/doc/product/aggr/10000/swconfig/swref/tech-edg/**sysdsfr**.pdf. Acesso em: 25 mar. 2006.

CWYNAR, T., LIMA, S., **Network Processors and their Impact in Networking Industry**. Agosto. 2000. Disponivel em: http://netlab.cs.tsinghua.edu.cn/~ljsheng/npu/from_servio/np.ppt. Acesso em: 07 mar. 2005.

DAEMEN, Joan, RIJMEN, Vincent, **AES Proposal: Rijndael** – The Rijndael Block Cipher. Setembro. 1999. Disponível em: http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf. Acesso em: 11 out. 2006.

DIER, J. G., LINDEMANN, Mark, PEREZ, Ronald, SAILER, Reiner, DOORN, Leendert van, SMITH, Sean W., WEINGART, Steve. **Building the IBM 4758 Secure Coprocessor**. IEEE *Computer*, p57-65, out, 2001. Disponível em: http://www.cs.dartmouth.edu/~sws/pubs/comp01.pdf. Acesso em: 17 set. 2006.

FIPS46-2, **Federal Informatio Processing Standards Publication 46-2**, Data Encryption Standard (DES). Dezembro. 1993. Disponível em: http://www.itl.nist.gov/fipspubs/fip46-2.htm. Acesso em: 23 ago. 2006.

FREITAS, H. C., C. A. P. S. Martins, **Projeto de Processador com Microarquitetura Dedicada para Roteamento em Sistemas de Comunicação de Dados**, WSCAD'00, 2000.

FREITAS, H. C., C. A. P. S. Martins, **RCNP**: Processador de Rede com Suporte a Multi-protocolo e Topologias Dinâmicas, WSCAD'01, 2001. Disponível em: http://scholar.google.com/url?sa=U&q=http://inf.pucminas.br/professores/cota/papers/wscad2001.pdf. Acesso em: 21 set. 2006.

FREITAS, H. C., C. A. P. S. Martins, **R2NP**: Processador de Rede RISC Reconfigurável, WSCAD'02, 2002. Disponível em: http://inf.pucminas.br/professores/cota/papers/wscad2002.pdf. Acesso em: 21 set. 2006.

HARPER, S. J., **A Secure Adaptive Netwo rk Processor**. Virgínia. 2003. Tese (Doutorado em Emgenharia Elétrica) Instituto Politécnico da Virgínia, USA, 2003.

HENRIKSON, T., **Hardware Architecture for Protocol Processing**. Suécia. 2001. Tese (Doutorado Engenharia Elétrica) Instituto de Tecnologia da Linköpings University, Suécia, 2001.

IBM Corp., **IBM Network Processor** (IBM32NPR161EPXCAC100). Product Overview. 1999.

INTEL, Intel Corp., **Intel® IXP1200 Network Processor Family**. Intel Corporation, Part number 27851301. Fevereiro. 2002. Disponível em: http://www.intel.com/design/network/applnots/27851301.pdf. Acesso em: 18 mai. 2006.

INTEL, Intel Corp., **Intel® IXP1200 Network Processor**, Intel Corporation, Part number 278316018. Março. 2004. Disponível em: ftp://download.intel.com/design/network/specupdt/27831618.pdf. Acesso em: 18 mai. 2006.

JESSEN, J., DHIR, A., "Programmable Network Processor Plataform". Article about of the available of SPEEDRouter ™ in Virtex ™ - II Network Processor Core, 2004.

KALTE, H. LANGER, D., VONNAHME, E., BRINKMANN, A., RÜCKERT, U., **Dynamically Reconfigurable System-on-Programmable-Chip**. 10th Euromicro Workshop on Parallel Distributed an Network-based Processing (EUROMICRO-PDP 2002), Gran Canary Island, Spain, 09-11 January, 2002. S. 235-242. Disponível em: http://wwwhni-alt.upb.de/publikationen/publication.php3?pub_id=174&parent=%2Fmitarbeiter%2Fmitarbeiter.php3%3Fid%3D115%26pub%3Dja. Acesso em: 20 set. 2006.

LOPEZ, Manuel J. Lucena, **Criptografía y Seguridad en Computadores 4a Edición**. Júlio 2006. E-Book disponível em: http://wwwdi.ujaen.es/~mlucena/wiki/pmwiki.php?n=Main.LCripto. Acesso em 12 out. 2006.

MENEZES, A., OORSCHOT, P. van, VANSTONE, S., **Handbook of Applied Cryptography**. Flórida: Editora CRC Press, Volume 6, 1996.

MORENO, E. D., PEREIRA F.D., CHIARAMONTE, R. B., **Projetos, Desenvolvimento e Aplicações de Sistemas Digitais em Circuitos Programáveis** (FPGAs). Marília: Editora Bless. 2003.

MORENO, E. D., PEREIRA, Fabio D., CHIARAMONTE, Rodolfo B., **Criptografia em Software e Hardware** – Implantação e Desenpenho. Marília: Editora Novatec. 2005.

MUZZI, Fernando A., **Um Protótipo Modular em FPGA do padrão de Segurança PKCS#11**. Marília. 2005. Dissertação (Mestrado em Ciência da Computação) Programa de Pós Graduação em Ciência da Computação (PPGCC) – Centro Universitário Eurípedes de Marília (UNIVEM), Marília, 2005.

OLIVEIRA, Alexande Ponce de, **PERS** – Um Processador específico para Redes de Sensores com primitivas de Segurança. Marília. 2005. Dissertação (Mestrado em Ciência da Computação) Programa de Pós Graduação em Ciência da Computação (PPGCC) – Centro Universitário Eurípedes de Marília (UNIVEM), Marília, 2006.

PEREIRA, Fábio D., **Um Criptoprocessador VLIW para Algoritmos Criptográficos Simétricos**. Marília. 2004. Dissertação (Mestrado em Ciência da Computação) Programa de Pós Graduação em Ciência da Computação (PPGCC) – Centro Universitário Eurípedes de Marília (UNIVEM),, Marília, 2004.

PRADO, Ricardo P., **NPSoC** – Arquitetura e Protótipo de um Novo Processador de Rede. Marília. 2004. Dissertação (Mestrado em Ciência da Computação) Programa de Pós Graduação em Ciência da Computação (PPGCC) – Centro Universitário Eurípedes de Marília (UNIVEM), Marília, 2004.

ROSA, Rafael Antonio da Silva, **Análise do Algoritmo Vencedor do AES**: O Rijndael. S. José dos Campos. 2003. Iniciação Científica e Pós-Graduação Instituto Tecnológico de Aeronáutica (ITA), S. José dos Campos, 2003. Disponível em: http://www.bibl.ita.br/ixencita/artigos/FundRafaelAntonio1.pdf. Acesso em: 17 abr. 2005.

SHAH, Niraj, **Understanding Network Processors**. Berkeley. 2001. Dissertação de Mestrado University of California, Berkeley, 2001. Disponível em www.cs.berkeley.edu/~plishker/UnderstandingNPs.pdf. Acesso em: 05 nov. 2006.

SILVA, Glaucio E. Ferreira da, **Análise comparativa entre os quatro Algoritmos de Chave Simétrica submetidos ao Projeto NESSIE** – Segunda Etapa. Gravataí. 2003. Trabalho de Conclusão de Curso (Ciência da Computação) Universidade Luterana do Brasil, Gravataí, 2003.

STALLINGS, William, **Arquitetura e Organização de Computadores**. São Paulo: Prentice-Hall, 5ª Edição, 2002.

STENBERG, A., **Network Processor Core Architecture**. Suécia. 2002. Dissertação de Mestrado (Departamento de Ciência da Computação e Engenharia Elétrica, Divisão de Engenharia da Computação) Lulea University of Technology, Suécia, 2002.

STENSTRÖM, Sven, **Implementation of a Network Processor Based Exchange Terminal**. Stockohlm. 2002. Dissertação de Mestrado (Departamento de Microeletrônica e Tecnologia da Informação) Royal Institute of Tecknology (KTH), Suécia, 2002. Disponível em: http://web.it.kth.se/~matsbror/exjobb/msc_theses/stenstrom-thesis_021205.pdf. Último acesso: 14 set. 2006.

SURYANARAYANAN, D., BYRD, G. T., MARSHALL, J., **A Methodology and Simulator for the Study of Network Processors**. Workshop on Network Processor, Cambridge. 2002. Disponível em: http://www.ecs.umass.edu/ece/wolf/courses/ECE697J/Fall2002/presentations/ECE697J-02-11-26.pdf. Último Acesso: 12 ago. 2006.

SESHADRI, M. S., BENT, J., KOSAR, T., **Intelligent Routing Network Processors**: Guiding Design through Analysis. Wisconsin. 2003. Relatório Técnico (Departamento de Ciência da Computação) Universidade de Wisconsin, Wisconsin, 2003.

TANEMBAUM, Andrew S., **Redes de Computadores**. Rio de Janeiro: Prentice-Hall, 4ª Edição, 2001.

TROXEL, I. A., GEORGE, A. D., ORAL, S., **Design and Analysis of a Dinamically Reconfigurable Network Processor**. Gainesville. 2002. IEEE Conference on Local Computer Networks (LNC), Tampa, Florida, November 6-8, 2002. Disponível em: www.hcs.ufl.edu/pubs/LCN2002.pdf. Acesso em: 12 set. 2006.

WOLF, T., **Design of an Instruction Set for Modular Network Processors**. IBM Report RC21865, October 27, 2000. Disponível em: http://www.ecs.umass.edu/ece/wolf/pubs/2000/ibm.html. Último Acesso: 27 ago. 2006.

WOLF, T., FRANKLIN, M. A., SPITZNAGEL, E. W., **Design Tradeoffs for Embedded Network Processors**. Relatório Técnico, WUCS-00-24, Washington University in St. Louis, July 10, 2000. Disponível em: http://www.arl.wustl.edu/Publications/2000-04/wucs0024.pdf. Último Acesso: 21 jun. 2006.