DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Estudo e Implementação do Protocolo PRP em Sistemas Embarcados Relacionados ao Ambiente IEC 61850

Monografia submetida à Universidade Federal de Santa Catarina

como requisito para a aprovação da disciplina:

DAS 5511: Projeto de Fim de Curso

Luis Gustavo Perpetuo Costa Marques

Florianópolis, Julho de 2012

Estudo e Implementação do Protocolo PRP em Sistemas Embarcados Relacionados ao Ambiente IEC 61850

Luis Gustavo Perpetuo Costa Marques

Esta monografia foi julgada no contexto da disciplina
DAS 5511: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação

Eng. Carlos Alberto Dutra
Orientador Empresa

Prof. Max Hering de Queiroz
Orientador do Curso

Agradecimentos

Primeiramente aos meus pais, José Márcio e Maria Elisa, por todos os conselhos e ensinamentos, pelo apoio e incentivo durante toda a minha graduação e nos momentos mais difíceis da minha vida.

Agradeço a todos os professores do curso de Engenharia de Controle e Automação pelos conhecimentos repassados com muita dedicação ao longo da graduação.

Agradeço aos meus orientadores Max e Dutra pelos conselhos e contribuições que muito ajudaram no desenvolvimento deste trabalho.

Para finalizar, agradeço toda equipe do departamento de desenvolvimento da Reason, por propiciarem um ambiente muito agradável e incentivador para realização desse trabalho, e por todo suporte oferecido.

Resumo

Subestações possuem muitos equipamentos para supervisão, proteção e controle, conectados em rede. A norma IEC 61850 estabelece um padrão de rede de comunicação em subestações que garante interoperabilidade entre equipamentos de diversos fabricantes. A segunda edição da norma sugere como soluções de redundância de rede, os protocolos PRP/HSR presentes na norma IEC 62439.

A empresa Reason Tecnologia S/A, já consagrada nas áreas de desenvolvimento de equipamentos para oscilografia, qualidade de energia e sincronismo temporal, está desenvolvendo produtos na linha de proteção elétrica para subestações, compatíveis com a norma IEC 61850 e que adotam as soluções de redundância proposta por ela, o PRP/HSR.

O PRP utiliza o conceito de equipamento com duas portas Ethernet enviando simultaneamente dados duplicados em duas redes locais independentes. O envio de dados duplicados exige que o receptor consiga identificar e descartar as duplicações. A duplicação dos dados e detecção de mensagens duplicadas é feito na camada de enlace de dados através de uma entidade chamada “Link Redundancy Entity” (LRE), implementado em software ou hardware.

O trabalho descrito nesse documento foi realizado na Reason e criou um circuito lógico digital que realiza o LRE, com o objetivo de integra-lo aos novos produtos da empresa para que eles ofereçam a solução de redundância PRP. Esse circuito foi construído em um FPGA utilizando VHDL para programação. Ao longo desse documento é apresentado o projeto proposto e a implementação do circuito digital do LRE, assim como o algoritmo para detecção das mensagens duplicadas: um misto de tabela hash com buffer circular. Ao final do documento são apresentados os testes que validaram o funcionamento do LRE criado nesse trabalho e as perspectivas futuras do projeto.

Abstract

Substations have several equipment for monitoring, protection and control, all connected in a network. The IEC 61850 establishes a standard for communications network in substations that ensures interoperability between equipment from different manufacturers. The second edition of the standard suggests as network redundancy solution, the PRP/HSR protocols presented in IEC 62439.

The Reason Tecnologia S/A company already leads the market of electrical energy oscilography, power quality and time synchronization equipments, is developing power protection products for substations that are compliant to the IEC 61850 standard and adopt the network redundancy solutions proposed by it, the PRP/HSR.

The PRP uses the concept of equipments that have two Ethernet ports simultaneously sending duplicated data in two independent local networks. Sending duplicated data requires that the receiver be able to identify and discard the duplicates. The data duplication and the detection of duplicated messages occur at the data link layer by an entity called “Link Redundancy Entity” (LRE), implemented in software or hardware.

The work described in this document was produced in Reason and created a digital logic circuit that performs the LRE, in order to integrate it to the company’s new products. This circuit was built in a FPGA utilizing VHDL for programming. Throughout this paper we present the proposed design and implementation of the digital circuit of the LRE, as well as the algorithm to detect duplicate messages: a mixture of hash table with circular buffer. At the end of the document, the tests that validated the LRE’s operation are presented along with the project’s future prospects.

Sumário

Simbologia

ARP	Address Resolution Protocol
Avalon-MM	Avalon Memory Mapped Interface
Avalon-ST	Avalon Streaming Interface
BPDU	Bridge Protocol Data Units
CID	Configured IED Description
CRC	Cyclic Redundancy Check
DANP	Doubly Attached Node
FCS	Frame Check Sequence
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
HSR	High-Availability Seamless Redundancy
ICD	IED Capability Description
IEC	International Electrotechnical Comission
IED	Electronic Intelligent Device
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol/Intelectual Property
ISO	International Organization for Standardization
LAN	Local Area Network
LLC	Logical Link Control
LRE	Link Redundancy Entity
LSDU	Link Service Data Unit
MAC	Medium Access Control
MRP	Media Redundancy Protocol
OSI	Open Systems Interconnection

PRP	Parallel Redundancy Protocol
RCT	Redundancy Check Trailer
RSTP	Rapid Spanning Tree Protocol
RTL	Register Transfer Level
SAN	Singly Attached Node
SAS	Sistema de Automação de Subestação
SCD	System Configuration Language
SCL	Substation Configuration Language
SGDMA	Scatter-Gather Direct Memory Access
SNMP	Simple Network Management Protocol
SRAM	Static Random Access Memory
SSD	Substation Specification Description
TC	Transformador de corrente
TCP	Transmission Control Protocol
TP	Transformador de tensão
TSE MAC	Triple Speed Ethernet MAC
UDP	User Datagram Protocol
VHDL	Very High Speed Integrated Circuit HDL
VLAN	Virtual Local Area Network

Capítulo 1: Introdução

A sociedade atual é totalmente desenvolvida tendo como base o uso de energia elétrica. Essa eletricidade é gerada em usinas (hidroelétricas, termoelétricas, nucleares etc.) que convertem energia das mais variadas formas em energia elétrica.

O processo de transporte de energia gerada em usinas até o consumo em indústrias e residências acontece por meio de linhas de transmissão, e envolve a constante manipulação dessa eletricidade, seja para elevação/redução de tensão ou simplesmente distribuição. Essa manipulação é realizada por instalações chamadas subestações, as quais acabam atuando como interface entre as linhas de transmissão e os consumidores/geradores.

Visto que a eletricidade é crucial para o funcionamento da sociedade moderna, é extremamente importante que as linhas de transmissão permaneçam funcionando de forma segura e com o mínimo de interrupções para que o consumidor continue a ter eletricidade disponível para o uso. Por essa razão, além dos equipamentos essenciais para o seu funcionamento (transformadores, linhas, chaves etc.), as subestações contam também com uma grande quantidade de dispositivos para supervisão, proteção e controle, formando o Sistema de Automação de Subestações (SAS).

Para o funcionamento adequado do SAS, os equipamentos precisam se comunicar e entender, o que ao longo dos anos provou-se ser um grande desafio devido a grande quantidade de diferentes fabricantes desses dispositivos. A norma IEC 61850 surge como solução para interoperabilidade entre os equipamentos de um SAS, fornecendo um padrão de rede de comunicação para subestações.

Subestações de energia elétrica são sistemas de tempo real crítico, exigindo que qualquer anormalidade seja rapidamente identificada e medidas para resolver a situação também sejam rapidamente tomadas. Isso exige que uma rede de comunicação em uma SAS esteja sempre funcionando, e devido a essa alta disponibilidade exigida, redundância na rede é quase um item obrigatório. A primeira edição da IEC 61850 não apresentou uma solução de redundância e a falta de uma

solução comumente aceita estava ameaçando o conceito de interoperabilidade, já que os fabricantes começaram a colocar no mercado soluções de redundância de propriedade, dificultando a construção de uma subestação quando as interfaces não se encaixavam. Felizmente a norma IEC 62439 foi apresentada especificando vários métodos de redundância, um deles aplicável a subestações de qualquer tamanho e topologia.

O PRP (Parallel Redundancy Protocol) foi uma solução de redundância proposta na IEC 62439, a qual depende do funcionamento paralelo de duas redes locais, e oferece transição completamente transparente em caso de falha de links ou switches, cumprindo assim todos os rígidos requisitos de tempo real de automação de subestações.

1.1: Motivação, Objetivos e Justificativas

A Reason Tecnologia S/A é uma empresa que produz equipamentos da área de oscilografia, qualidade de energia, sincronismo temporal e mais recentemente relés de proteção. Com o desenvolvimento de relés de proteção, a Reason está prestes a entrar em um novo mercado, o de proteção elétrica, e precisa oferecer produtos com um diferencial em relação aos presentes no mercado a fim de conquistar clientes e se firmar nesse novo ambiente. Observando as tendências do setor de proteção elétrica para os próximos anos, a Reason reconheceu a importância de que seus novos produtos sejam totalmente compatíveis com a norma IEC 61850.

Redundância é um aspecto obrigatório em subestações, e todas as empresas que atuam no mercado de proteção elétrica oferecem sua solução para esse assunto, e a Reason, como também fará parte desse mercado necessita fazer o mesmo.

A segunda edição da IEC 61850 propõe como solução de redundância para redes de subestações, o PRP e o HSR. A Reason, como pretende oferecer produtos compatíveis com a norma, escolheu implementar esses dois protocolos de redundância em seus produtos, dando a origem ao projeto tema desse trabalho, realizado na empresa, que é a implementação do protocolo PRP.

O PRP exige, conforme será explicado ao longo deste trabalho, a criação de um dispositivo chamado “Link Redundancy Entity” (LRE). Esse dispositivo pode ser realizado em software ou hardware, sendo que nesse trabalho será criado em hardware como um circuito lógico digital. O objetivo do trabalho é analisar, projetar e implementar o LRE, dispositivo capaz de operar o protocolo PRP. Ao final do trabalho espera-se que o LRE atenda todas as especificações da norma PRP e possua uma interface apropriada para ser incluído nos produtos novos criados pela empresa.

1.2: Metodologia Utilizada

Esse trabalho foi realizado em várias etapas. São elas:

- Estudo da norma IEC 61850 e da norma IEC 62439: Nessa etapa, fez-se uma leitura das normas relevantes ao projeto para compreensão do funcionamento do PRP.
- Levantamento dos requisitos funcionais e não funcionais: Nessa etapa listou-se o que o LRE deveria ser capaz de realizar e as restrições quanto ao seu funcionamento.
- Baseado nos requisitos, análise e formulação da solução proposta: Nessa etapa definiu-se que o LRE seria criado como circuito lógico digital e, baseado nessa escolha, foi definido quais ferramentas seriam utilizadas.
- Aplicação prática da solução proposta.
- Testes e avaliação do resultado.

1.3: Estrutura do documento

Segue abaixo a organização desse documento:

O capítulo 2 fará uma contextualização do ambiente em que esse trabalho se insere, descrevendo melhor as motivações e objetivos. Será apresentado também a empresa Reason e o projeto PRP/HSR, o qual esse trabalho faz parte.

O capítulo 3 explicará o funcionamento do protocolo PRP. Nesse capítulo será apresentado o LRE, suas funcionalidades e como ele se relaciona com a interface de rede do equipamento em que será incluso.

O capítulo 4 apresentará as especificações quanto à realização do LRE e a escolha da tecnologia utilizada para implementação.

O capítulo 5 mostra toda a etapa do projeto do circuito lógico digital do LRE, apresentando diagramas que mostram quais componentes estão presentes e como eles se relacionam, e também máquinas de estados que descrevem o funcionamento de cada módulo.

O capítulo 6 fará uma descrição de como o projeto definido no capítulo 5 foi implementado. Este capítulo também irá apresentar os testes realizados e os resultados obtidos.

No capítulo 7 fará uma conclusão do trabalho e apresentação de perspectivas futuras.

Capítulo 2: Contextualização

Este capítulo tem por objetivo fazer uma contextualização do ambiente em que o projeto tema desse trabalho se insere, explicar importância do mesmo para empresa onde ele foi realizado, assim como fazer uma apresentação da própria empresa. Ao final do capítulo são citadas as disciplinas do curso, as quais forneceram conhecimento importante para o projeto.

2.1: Subestação de energia elétrica

Uma subestação é uma instalação elétrica de alta potência, contendo equipamentos para transmissão e distribuição de energia elétrica, além de equipamentos de proteção e controle.

A subestação funciona como ponto de controle e transferência em um sistema de transmissão de energia elétrica, direcionando e controlando o fluxo energético, transformando os níveis de tensão e funcionando como pontos de entrega para consumidores industriais e residenciais.

O principal equipamento da subestação é o transformador de potência, responsável por transformar os níveis de tensão. Na maior parte das subestações, disjuntores são utilizados para interromper curtos-circuitos ou sobrecargas de correntes que podem acontecer na rede, enquanto ocasionalmente, em subestações menores, fusíveis são utilizados para essa função. Chaves seccionadoras são utilizadas para controlar o fluxo energético, abrindo e fechando circuitos. Transformadores de corrente (TC) e tensão (TP) fazem medições dos valores de corrente e tensão das diversas linhas conectadas a subestação. Outros dispositivos como capacitores e reguladores de tensão também podem estar presentes nas subestações.

O Sistema de Automação de Subestação (SAS) é responsável pelo monitoramento, proteção e controle de uma subestação. Ele é composto por vários dispositivos multifuncionais chamados IEDs (Electronic Intelligent Devices), que realizam funções de aquisição de dados (Ex: obtenção de dados de TPs e TCs), supervisão e controle (Ex: operar um disjuntor). Estes dispositivos precisam trocar

dados para realizar suas funções e por isso estão conectados em uma rede de comunicação.

2.2: Comunicação em Sistemas de Automação de Subestações

A automação de subestações [1] visa à melhoria da qualidade no fornecimento de energia elétrica, reduzindo a quantidade e o tempo de interrupções através da supervisão do sistema elétrico em tempo real e direto, além da redução dos custos operacionais, através da automação de tarefas e centralização de ações operativas. Basicamente, um sistema de automação de subestação desempenha três tarefas: proteção, controle e monitoramento.

Comunicação sempre teve um papel crítico nas operações de tempo real crítico (hard real-time) de subestações [2]. No começo, o telefone foi utilizado para reportar o status de uma linha para o centro de comando, que por sua vez enviava operadores ao local para realizar as operações de chaveamento. A partir dos anos 60, com o surgimento da comunicação digital, sistemas de aquisição de dados foram instalados para realizar a tarefa de coletar dados de medição e enviá-los ao centro de controle.

Segundo [3], antes da introdução de equipamentos baseados em microprocessadores e da comunicação serial em subestações, proteção, controle e monitoramento eram realizados por diferentes dispositivos dedicados, o que resultava em responsabilidades sendo dividida entre vários departamentos dentro das empresas, tanto das concessionárias responsáveis pela instalação, como nas empresas fornecedoras de equipamento.

O desenvolvimento da tecnologia de microprocessadores e de comunicação, mais especificamente a introdução de comunicação serial e de dispositivos multifuncionais, levou a uma mudança nas soluções de automação de subestações. Com o objetivo de aperfeiçoar o sistema e reduzir custos, a integração de mais e mais funções em cada vez menos equipamentos se tornou uma tendência. Diferentes funções como proteção, controle e monitoramento poderiam ser combinadas em um único equipamento (cada IED para proteção de uma linha, por exemplo, deve ser capaz de receber dados de amostragem, enviar dados de controle para atuadores e enviar informações de status para um sistema supervisório). Consequentemente, a especificação de tais sistemas se tornou muito

diferente, a ponto que, atualmente, todas as funcionalidades disponíveis para proteção, controle e monitoramento de uma subestação não são tratadas separadamente, mas em um sistema comum e consistente.

Essa nova abordagem de comunicação aumentou a necessidade de troca de informações entre vários dispositivos envolvidos na automação e proteção da subestação, e com isso aumentou a pressão por um protocolo de comunicação comum que resolvesse o problema de interoperabilidade entre equipamentos de diversos fabricantes. A norma IEC 61850 “Communication Networks and Systems in Substations” surge como uma solução de padronização na comunicação entre os vários dispositivos dentro de uma subestação.

2.3: A norma IEC 61850

Este padrão foi desenvolvido tendo como objetivos principais: assegurar interoperabilidade entre os diferentes IEDs de uma subestação e atender aos diferentes tipos de arquitetura utilizados, além de suportar desenvolvimentos tecnológicos futuros sem requerer alterações significativas no software e hardware do SAS.

Para alcançar os objetivos acima, o novo padrão utiliza a abordagem orientada a objeto e subdividem as funções em objetos denominados nós lógicos que se comunicam entre si. As funcionalidades requeridas para a automação e proteção da subestação são focadas nos nós lógicos e não na quantidade de IEDs ou funções utilizadas.

Outro objetivo da norma IEC 61850 é possibilitar a comunicação entre IEDs com alta velocidade e confiabilidade elevada, possibilitando a substituição dos cabos de controle por redes de comunicação e reduzindo o custo global.

A grande conquista da IEC 61850 é a descrição de toda a comunicação da subestação na linguagem de configuração da Subestação (SCL) [4], que permitiu pela primeira vez que a engenharia de uma subestação utilizasse equipamentos de medição, proteção e controle (IEDs) de diferentes fabricantes, tornando o SAS em uma plataforma aberta de proteção e automação.

O processo de configuração de um SAS de uma subestação utilizando a linguagem SCL está demonstrado na Figura 1.

As diversas possibilidades e funcionalidades disponíveis em um determinado IED são representadas na linguagem SCL através do arquivo ICD (IED Capability Description), fornecido pelo fabricante. A especificação do SAS, transcrita em linguagem SCL, constitui o arquivo SSD (Substation Specification Description). Este arquivo, juntamente com os arquivos ICD dos diversos IEDs, configurados, através do emprego de uma ferramenta de engenharia adequada, de modo a atender aos requisitos especificados, dá origem ao arquivo SCD (Substation Configuration Description). Adicionalmente, o arquivo de cada IED, depois de configurado para um projeto especifico, passa a ser denominado arquivo CID (Configured IED Description) daquele IED.

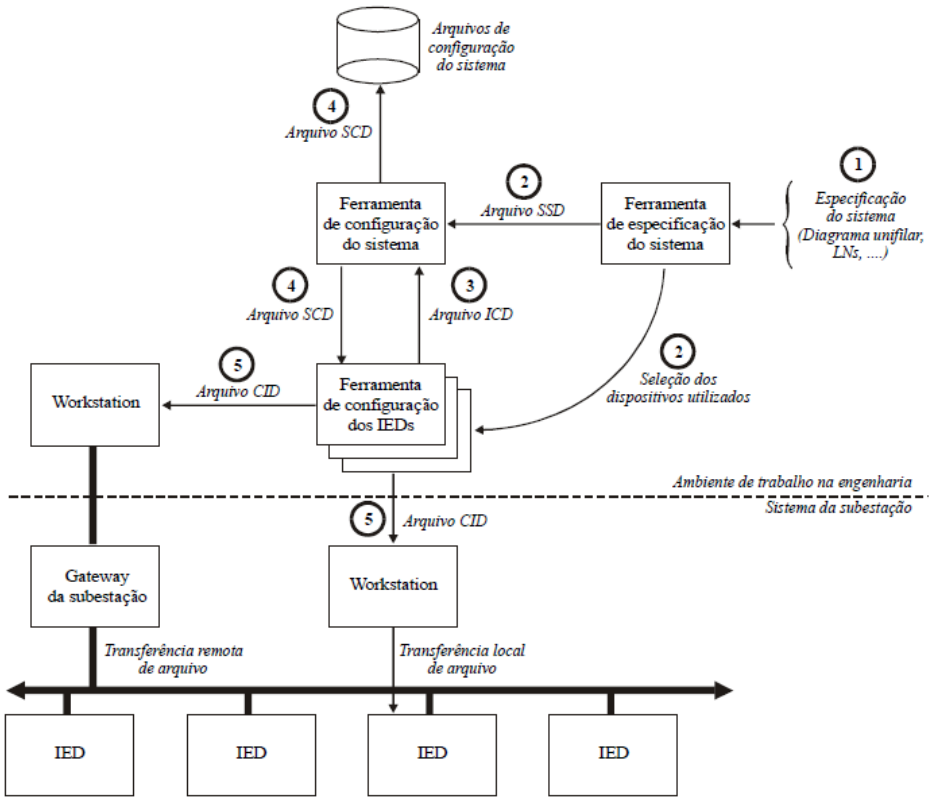


Figura 1: Modelo de referência para o fluxo de informações no processo de engenharia com a linguagem SCL

A especificação de um SAS utilizando o padrão IEC 61850 é, até certo ponto, semelhante à especificação de um sistema convencional. Devem ser fornecidas informações sobre o diagrama unifilar da subestação, a lista de pontos, as funcionalidades requeridas, os requisitos de desempenho, as interfaces com o processo e com outros IEDs, além de protocolos, condições ambientais, índices de confiabilidade admitidos ou critérios de tolerância a falhas aceitáveis etc. O uso do

padrão IEC 61850, porém, requer, adicionalmente, outros requisitos, como características do subsistema de comunicação (LAN e equipamentos de comunicação), documentação em linguagem SCL e procedimentos de teste específicos para o SAS que está sendo adquirido.

Uma grande vantagem do padrão IEC 61850 é a possibilidade de se reduzir sensivelmente a quantidade de cabos e de pontos de entrada e saída dos equipamentos digitais através do compartilhamento das múltiplas informações sobre o processo entre os diversos subsistemas. Assim, por exemplo, as informações analógicas e de estado relativas ao processo, podem ser adquiridas pelos relés das proteções e compartilhadas com os sistemas de supervisão, controle e automação, com custos menores e com maior confiabilidade.

Subestações de energia elétrica são sistemas de tempo real crítico. Na ocorrência de uma falta, os sistemas de proteção devem atuar no intervalor típico de no máximo 4ms, portanto em um sistema baseando na IEC 61850, a rede de comunicação deve estar sempre disponível, rápida e segura. Devido a essas exigências, redundância é quase um item obrigatório para garantir disponibilidade e confiabilidade. A primeira edição da IEC 61850 não apresentou uma solução de redundância e a falta de uma solução comumente aceita estava ameaçando o conceito de interoperabilidade, já que os fabricantes começaram a colocar no mercado soluções de redundância de propriedade, impedindo a construção de uma subestação quando as interfaces não se encaixavam. Felizmente, a norma IEC 62439 foi apresentada especificando vários métodos de redundância, um deles aplicável a subestações de qualquer tamanho e topologia.

2.4: Redundância em SAS e a norma IEC 62439

Como relatado em [5], em subestações é comum se utilizar o critério n+1 para redundância que garanta que o sistema seja tolerante a um ponto único de falha. Redundância é qualquer recurso utilizado o qual não seria necessário se não existisse falhas. Redundância pode ser dividida em redundância funcional e de hardware, e pode ser aplicada a diferentes componentes do SAS, como por exemplo, proteção, controle, automação ou rede de comunicação.

Redundância de hardware é a duplicação de dispositivos ou de alguns componentes (módulos) desses dispositivos. Redundância de componentes é

implementada pela duplicação de fontes de energia, módulo principal de processamento, placa de rede etc., enquanto redundância de dispositivo é a duplicação do equipamento inteiro, como por exemplo, a utilização de dois relés de proteção para a mesma tarefa.

Redundância de dispositivos é obtida através da utilização de equipamentos duplicados trabalhando em esquemas hot-hot (os dois dispositivos funcionam ao mesmo tempo, caso um falhe o outro continua funcionando) ou hot-standby (apenas um dispositivo funciona por vez, sendo que o segundo passa a funcionar a partir da falha do primeiro). Em ambos os casos, algum nível de troca de informação entre os dispositivos redundantes é necessário para garantir uma hierarquia de funcionamento e consistência da informação sendo trocada.

Redes de comunicação em subestações necessitam de alta disponibilidade para garantir operação contínua da planta. Além de equipamentos de excelente qualidade e manutenção contínua, a disponibilidade de uma rede pode ser aumentada através da utilização de elementos redundantes ativados automaticamente em casos de falha.

Redundância na rede de comunicação é a habilidade da arquitetura da rede ser tolerante a falhas. Esse tipo de redundância é realizado através de protocolos de redundância de rede, cuja arquitetura contém caminhos de comunicação alternativos e redundantes. Esses protocolos implementam mecanismos de auto reparo para uma recuperação rápida em caso de falhas, garantindo transferência de dados de maneira eficiente e otimizada através dos caminhos de comunicação mais curtos.

Redundância tem um impacto direto sobre a complexidade e custo de um SAS, e é definido dependendo do quão crítico é o sistema, exigências do consumidor ou da estrutura existente.

A primeira edição da IEC 61850 não explicita como realizar redundância na rede em um SAS, assunto o qual é abordado na norma IEC 62439.

A norma IEC 62439 apresenta vários protocolos de redundância para implementar redes de automação de alta disponibilidade, divididos em duas categorias:

- Redundância na rede: Cada equipamento está conectado somente a um switch (bridge), enquanto os switches implementam a redundância. Ex: RSTP, anéis duplicados.
- Redundância nos dispositivos: Cada equipamento possui duas interfaces de rede, acopladas a redes redundantes.

A seguir, os esquemas e protocolos de redundância mais utilizados atualmente em subestações serão brevemente explicados e por último será citado as características do protocolo PRP, tema desse trabalho e que será apresentado com detalhes no próximo capítulo.

2.4.1: RSTP (Rapid Spanning Tree Protocol)

O princípio de funcionamento do RSTP consiste de todos os switches da LAN coletando informações sobre os outros switches, através da troca de mensagens chamadas BPDU (Bridge Protocol Data Units). A troca dessas mensagens permite a escolha automática do switch raiz e habilitar ou desabilitar portas desnecessárias no switch. O switch “Raiz” é considerado o centro lógico da rede. Todos os caminhos os quais não são necessários para atingir a “Raiz” são colocados em modo backup. Todos os switches calculam o menor caminho para a “Raiz”. Cada switch pode somente ter um melhor caminho para encaminhar quadros para a “Raiz”. Quando da falha de um switch, os outros switches recalculam o melhor caminho para a “Raiz”, o que leva certo tempo de recuperação em torno de alguns milissegundos.

O RSTP é recomendado para redes de automação cujo tempo de recuperação de uma falha pode ser maior que 100 ms, portanto não é recomendado para tempo real crítico. O RSTP é um protocolo de redundância na rede presente na IEC 62439.

2.4.2: Dual Homing

Nesse esquema de redundância os dispositivos possuem duas conexões com a LAN através de diferentes switches. Apenas uma conexão está ativa por vez, enquanto a outra em standby (configuração hot-standby). Quando uma falha na conexão ativa é detectada, o dispositivo começará a se comunicar pela outra conexão.

O tempo de recuperação de falha em dual homing não é zero e dependendo da implementação pode ser de alguns milissegundos a até alguns segundos, o que não indicado para tempo real crítico.

2.4.3: IP duplo e duplicação da rede

Redundância através de completa duplicação da rede de comunicação é baseada na criação de duas redes separadas, e cada equipamento possuindo duas interfaces de rede com diferentes IPs. Esse tipo de redundância possibilita elevada disponibilidade já que todas as conexões e elementos de rede são duplicados e tempo de recuperação zero. A desvantagem é o seu elevado custo e a complexidade introduzida no nível de aplicação. Cada IED deve possuir dois endereços IPs e MACs o que torna a comunicação entre os dispositivos bastante complicada, já que os equipamentos devem suportar múltiplas aplicações rodando ao mesmo tempo e têm que lidar com mensagens duplicadas.

2.4.4: PRP (Parallel Redundancy Protocol) e HSR (High Availability Seamless Redundancy)

PRP/HSR [6] utiliza o conceito de IED com duas portas Ethernet. Cada IED tem apenas um endereço IP e MAC os quais são compartilhados pelas duas portas, que estão enviando simultaneamente dados duplicados. A duplicação dos dados é feito na camada de enlace de dados através de uma entidade chamada “Link Redundancy Entity” (LRE), implementado em software ou hardware, o qual garante que ambas as portas sejam interpretadas pela aplicação como somente uma interface de rede. Duplicação dos dados significa que o receptor irá receber todos os quadros duas vezes. O LRE fica responsável por detectar as mensagens duplicadas e descarta-las. PRP e HSR possibilitam tempo de recuperação de falha zero, pois os dados estão sempre disponíveis nas duas portas, não sendo necessária reconfiguração em caso de falha. A principal vantagem do PRP/HSR é o fato da aplicação nem perceber a existência do LRE ou que as mensagens estão sendo duplicadas, ou seja, o protocolo é transparente para a aplicação.

Existem diferenças sutis entre os protocolos PRP e HSR, mas a maior diferença é na topologia da rede. PRP funciona em qualquer topologia de rede, já que é baseado na duplicação total da mesma, já o HSR é um anel de nós, sem a presença de switches.

Esses dois protocolos têm uma importância especial por serem as soluções de redundância indicadas na segunda edição da IEC 61850.

2.5: A empresa Reason Tecnologia S/A

Fundada em 1991 em uma incubadora de empresas de tecnologia, e sediada em Florianópolis, Santa Catarina, Brasil, a Reason é uma empresa de capital nacional e atua no desenvolvimento de soluções na área de oscilografia, qualidade de energia e sincronismo temporal para os setores elétrico e industrial. Seus principais clientes são as companhias de geração, transmissão e distribuição de energia elétrica, grandes indústrias, empresas de engenharia, integradores e prestadores de serviço. Entre os seus clientes podemos citar: Celesc, Eletrosul, Copel, Chesf entre outras.

Seu primeiro produto foi o Registrador Digital de Perturbações RP-II. Em 1995 ingressou na Tecnópolis, onde lançou o Registrador Digital de Perturbações RP-III. Em 1997 lançou o RP-IV que se tornou o principal equipamento de oscilografia do Brasil, tornando-se referência nacional em tecnologia para o setor elétrico. Em 2007 lançou o RPV Registrador Digital Multifunção com a tecnologia inovadora da função de localização de faltas por ondas viajantes (travelling waves), que atualmente, é a tecnologia de melhor precisão e eficiência para reduzir o tempo de falta do sistema. Ainda em 2007 a empresa abriu uma filial nos Estados Unidos e, em paralelo, forneceu equipamentos para vários países da América Latina.

Em 2008 a Reason desenvolveu o RT420 Relógio Sincronizado por Satélites, que concentra todas as funcionalidades de outros receptores GPS comercializados anteriormente em um único equipamento. No final de 2008, a empresa abriu uma filial na Europa, em Berlim na Alemanha. Em 2009, lançou produtos voltados especialmente para monitoração e localização de faltas por ondas viajantes, e Registradores Digitais de Perturbações menores, mas mantendo as funcionalidades e características desenvolvidas para a família RPV. Em 2011 recebeu o prêmio FINEP de Inovação, competindo com empresas de todo o país.

2.5.1: Linha de Produtos

A Reason oferece linhas de produtos na área de Oscilografia com Registradores Digitais de Perturbações (RDP) e na área de sincronismo temporal com Relógios Sincronizados por GPS [7].

O Registrador Digital de Perturbações executa funções de oscilografia, além de outras requeridas atualmente para análise de uma ou mais instalações elétricas. Além do tradicional registro de forma de onda utilizado para análise de curtos, energização e outros transitórios, outras funcionalidades como registros fasoriais, registros de qualidade de energia, medição e difusão de sincrofasores também estão presentes.

Os produtos da linha de sincronismo temporal disponibilizam uma base de tempo única, referida ao sistema GPS, que permitem a análise de informações de diversos equipamentos. A linha de sincronismo temporal compreende os equipamentos RT420 (Relógio Sincronizado por Satélites GPS), RT412 (Transceptor Óptico), RT1320 (Repetidor de Tempo), RT1400 (Display IRIG-B).

Recentemente a empresa está desenvolvendo produtos na linha de Relés de Proteção Elétrica, ingressando em um novo mercado.

2.5.2: O Projeto PRP/HSR

Atualmente existem poucos produtos que fornecem a solução de redundância PRP/HSR, no entanto, a empresa tem acompanhado em diversos congressos e workshops que a tendência futura do mercado é adoção desses protocolos como solução padrão para redes de comunicação em subestações.

A proposta do projeto PRP/HSR é antecipar a tendência do mercado e criar um dispositivo capaz de operar esses protocolos para futuramente incluí-lo nos produtos Reason.

Esse dispositivo irá utilizar a mesma tecnologia usada nos IEDs os quais estão sendo desenvolvidos pela empresa, e servirá como uma solução a mais de redundância. As soluções utilizadas atualmente (Dual Homing, IP duplo) ainda serão incluídas como opção nos novos produtos, para permitir que esses equipamentos possam ser utilizados na atual realidade do mercado.

Inicialmente, o projeto irá focar na operação do protocolo PRP, tema desse trabalho, por ser de mais fácil implementação e também por já existir equipamentos de outros fabricantes que adotam essa tecnologia, facilitando testes de interoperabilidade. O protocolo HSR, fora do escopo desse trabalho, é quase uma consequência natural do PRP, pois utiliza quase toda a estrutura do mesmo, apenas com adição de um componente a mais que irá atuar como um switch, já que HSR foi desenvolvido para uma rede em anel.

O projeto acontece de forma paralela ao desenvolvimento de outros produtos, mas com uma obrigatória troca de informações, pois como foi esclarecido anteriormente, o dispositivo não será independente e por isso é preciso ter conhecimento da estrutura dos equipamentos em que ele será incluído, especialmente a estrutura associada à interface de rede, onde o PRP/HSR atua.

2.6: Contextualização com o curso

O presente documento foi desenvolvido como atividade para o Projeto de Fim de Curso (DAS 5511), no departamento de desenvolvimento da empresa Reason Tecnologia S/A. Várias disciplinas do curso estão relacionadas com o trabalho em maior ou menor participação, algumas delas são: Sistemas Digitais, Microprocessadores, Informática Industrial I e II, Eletrônica, Redes de Computadores e Sistemas Distribuídos.

2.7: Conclusão

Neste capítulo foi feito uma descrição do atual cenário e também a tendência futura de redes em subestações, ambiente em que este trabalho se insere. Foi feito também uma apresentação da empresa onde o trabalho foi realizado, assim como a motivação para o mesmo.

Nós próximos capítulos serão apresentados o embasamento teórico necessário para o projeto, um detalhamento de como foi realizado e apresentação de resultados.

Capítulo 3: O Protocolo PRP

O capítulo 3 fornecerá, inicialmente, o embasamento teórico necessário para compreensão do protocolo PRP, para, posteriormente, explicar o funcionamento do PRP em detalhes.

3.1: O modelo OSI

A grande importância da interconexão dos computadores através de redes de comunicação deu origem a uma necessidade que foi se tornando evidente à medida que os desenvolvimentos neste domínio foram acentuando-se: a padronização das redes de comunicação.

O modelo OSI (Open Systems Interconnection) é um modelo de arquitetura de redes de comunicação proposto pela ISO para resolver a necessidade de padronização. Nesse modelo, a rede de comunicação é dividida em camadas, em que funções similares são agrupadas. Uma camada oferece um serviço à camada acima e solicita serviço da camada abaixo.

As sete camadas do modelo OSI (aplicação, apresentação, sessão, transporte, rede, enlace de dados, física) e a forma como os dados são transferidos ao longo do modelo é ilustrada na Figura 2. Como se pode ver, o processo emissor vai enviar certa quantidade de dados ao processo receptor. Ele envia, então, os dados à camada de aplicação que introduz a estes um cabeçalho de aplicação, AH, e envia a mensagem resultante à camada de apresentação.

Esta camada, por sua vez, introduz a mensagem recebida um cabeçalho de apresentação, PH, enviando a mensagem para a camada inferior.

É importante ressaltar aqui que esta camada não toma conhecimento da existência e significado o cabeçalho de aplicação, considerando este como parte dos dados compondo a mensagem. Este processo de transferência de camada a camada vai se repetindo até o nível físico, quando os dados serão, enfim, transmitidos ao sistema destino. Neste sistema, os diversos cabeçalhos introduzidos nas camadas de rede do sistema fonte vão sendo interpretados e eliminados nas camadas correspondentes até que os dados cheguem ao processo receptor.

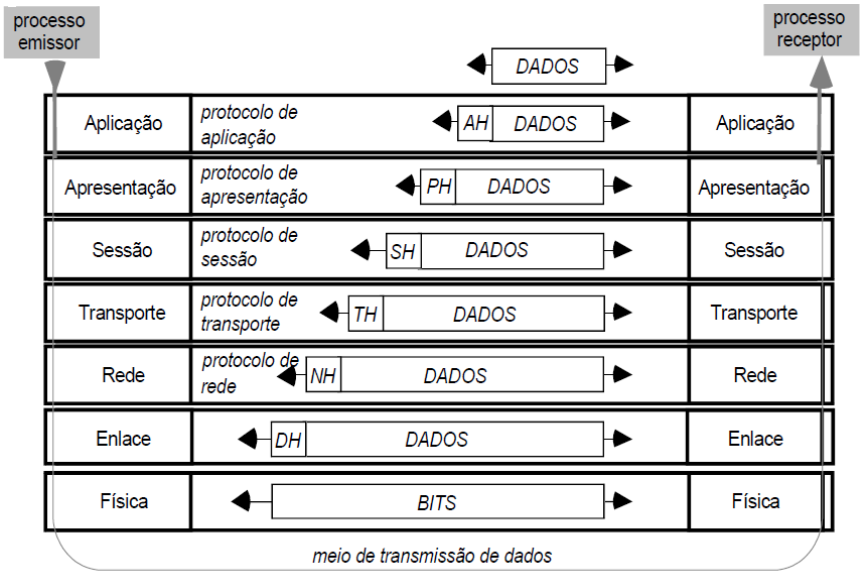


Figura 2: Arquitetura a sete camadas do modelo OSI

O conceito fundamental da transferência de dados é que cada camada foi projetada como se ela fosse realmente horizontal (entre camadas de rede de diferentes sistemas, por exemplo), quando na verdade a transmissão se dá de modo vertical.

Cada camada do modelo desempenha funções específicas importantes para o estabelecimento da comunicação entre os nós da rede, no entanto a seguir serão descritas apenas as funções das camadas, física, de enlace de dados e de rede, por serem mais relevantes neste trabalho. Os textos se baseiam em [8].

3.1.1: Camada Física

A camada física é responsável pela transferência de bits num circuito de comunicação. De maneira geral, a sua função é garantir que cada bit enviado de um lado será recebido do outro lado sem ter alterado o seu valor. As questões a serem resolvidas neste nível são do tipo:

- Os modos de representação dos bits 0 e 1 de maneira a evitar ambiguidades e confusões (valor de tensão para representação de 0 e 1, duração de sinal para representar bit etc.).
- Os tipos de conectores a serem utilizados nas ligações (número de pinos utilizados, funções de cada pino etc.)

- A maneira como as conexões são estabelecidas para a iniciação de um diálogo e como é feita a desconexão ao final deste.
- Modo de transmissão adotado (unidirecional, bidirecional etc.).
- Modo de conexão (ponto-a-ponto, multiponto etc.).
- Modo de tratamento dos erros (detecção, tratamento, etc.).

A concepção desta camada deve se relacionar à definição das interfaces elétricas e mecânicas, seus modos de funcionamento, o suporte de comunicação adotado, etc.

Exemplos de protocolos: IEEE 802.3, Bluetooth, RS232, USB etc.

3.1.2: Camada de enlace dos dados

A camada de enlace de dados tem por função principal a transformação do meio de comunicação (camada física) em uma linha livre de erros de transmissão para a camada de rede. Ela efetua esta função através da decomposição das mensagens em unidades de dados denominadas quadros (frames), que correspondem a algumas centenas de bytes. Estes quadros são transmitidos sequencialmente e vão gerar quadros de reconhecimento enviados pelo receptor. Nesta camada, as unidades de dados são enriquecidas com um conjunto de bits adicionais (no início e fim de cada quadro) de modo a permitir o reconhecimento destes e a definição de um endereço para o destinatário da mensagem.

Um problema típico deste nível é o da ocorrência de uma perturbação sobre a linha de transmissão que provoque a deturpação (perda) de quadro enviado. Esta deturpação ou perda deve ser reconhecida e tratada (controle de erros de transmissão de quadros). Neste caso, o quadro deve ser retransmitido para garantir a integridade da informação transferida. Por outro lado, deve-se também evitar múltiplas retransmissões de um mesmo quadro, o que pode provocar a sua duplicação, por exemplo, se o quadro de reconhecimento é perdido.

Outra função desta camada é evitar uma alta taxa de envio de dados da parte do emissor no caso do sistema receptor não ter capacidade de absorver a informação à mesma taxa. Este mecanismo deve permitir informar ao emissor a necessidade de armazenamento dos dados a transmitir (controle de fluxo de quadros).

Nas redes de difusão a camada de enlace de dados é usualmente decomposta em duas subcamadas, conforme proposta da IEEE:

- Subcamada de Controle de Acesso ao Meio (MAC – Medium Acess Control), responsável pelo acesso ordenado e compartilhado do canal de comunicação.
- Subcamada de Controle Lógico de Enlace (LLC – Logical Link Control), responsável pelo estabelecimento de conexões e oferecimento de serviços de comunicação às camadas acima.

Exemplos de protocolos: IEEE 802.3, Token Ring, PPP, ATM etc.

3.1.3: Camada de rede

Responsável pela gestão de sub-redes; ela defina a forma como os pacotes de dados serão encaminhados do emissor ao receptor (roteamento). Os caminhos a serem utilizados podem ser definidos em função de tabelas estáticas ou determinados dinamicamente no momento de cada diálogo em função das condições de tráfego da rede. Esta camada deve ainda efetuar a gestão dos problemas de congestionamento provocados pela presença de uma quantidade excessiva de pacotes de dados na rede. Ela deve, finalmente, resolver todos os problemas relacionados à interconexão de redes heterogêneas, como incompatibilidade no endereçamento e incoerências em relação aos tamanhos das mensagens.

Exemplos de protocolos: IP, ARP, ICMP, AppleTalk.

3.2: Rede Ethernet

Ethernet [9] é uma tecnologia de interconexão para LANs baseada no envio de quadros. Ela define cabeamento e sinais elétricos para a camada física, e formato de quadros e protocolos para a subcamada MAC do modelo OSI. A Ethernet foi padronizada na IEEE 802.3. Atualmente é a tecnologia de LAN mais utilizada.

Os padrões atuais do protocolo ethernet são os seguintes: 10 megabits/s (10Base-T, IEEE 802.3), 100 megabits/s (Fast Ethernet, IEE 802.3u), 1gigabits/s (Gigabit Ethernet, IEEE 802.3z), 10 gigabits/s (10 Gigabit Ethernet, IEEE 802.3ae).

A seguir será descrito subcamada MAC de uma rede ethernet, em especial o formato do quadro, o qual é a parte mais relevante para esse trabalho.

3.2.1: O quadro ethernet

O padrão IEEE 802.3 define um formato básico de quadro ethernet que todas as implementações de MAC devem possuir, mais alguns formatos opcionais que são usados para estender a capacidade do protocolo. Os sete campos básicos são: preâmbulo, delimitador de início de quadro (start of frame delimiter – SOF), endereço destino, endereço de origem, tamanho/ethertype, dados (LSDU) e checksum (frame check sequence – FCS). Quando se utiliza VLAN, o quadro é estendido com mais um campo de etiqueta de LAN virtual (802.1Q). Abaixo, na Figura 3, temos os três tipos de quadros ethernet.

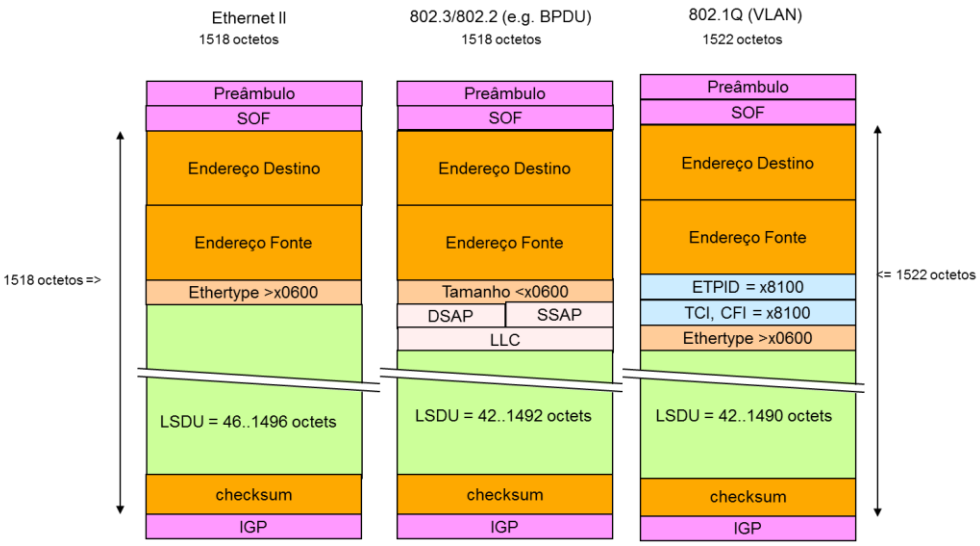


Figura 3: Tipos de quadros Ethernet

O preâmbulo consiste de sete bytes, sendo um padrão de 1s e 0s alternados que indica para a estação receptora que um quadro está chegando, e também fornece um meio de sincronizar a camada física com os bits que estão chegando.

O SOF consiste de 1 byte. É um padrão alternado de 1s e 0s, terminando com dois 1s consecutivos, indicando o início do quadro.

O endereço destino identifica qual estação deve receber o quadro. O bit mais a esquerda indica se o endereço é individual (indicado por um 0), ou de grupo (indicado por um 1) . Os restantes dos 46 bits são uma chave globalmente única que

identifica uma única estação (endereço unicast), um grupo definido de estações (endereço multicast), ou todas as estações na rede (endereço broadcast).

O endereço de origem consiste de seis bytes. O endereço de origem identifica a estação que enviou o quadro, e sempre é um endereço unicast.

Os endereços de destino e de origem são endereços MAC.

O campo tamanho/ethertype consiste de dois bytes. Esse campo indica ou o tamanho do campo de dados, ou o protocolo encapsulado nos dados. Caso valor de tamanho/ethertype seja menor ou igual a 1536, temos uma informação do tamanho do campo de dados em bytes. Caso o valor seja maior que 1536, então temos uma informação de ethertype, indicando um quadro do tipo “ethernet II”. Caso o valor de ethertype seja 0x8100, então temos um quadro que carrega informação de VLAN (formato 802.1Q na Figura 3).

O campo de dados, ou LSDU (Link Service Data Unit), é uma sequência de *n* bytes de qualquer valor, onde *n* é menor ou igual a 1500. Caso o valor de *n* seja menor que 46, a subcamada deve preencher esse campo com zeros até que tenha o tamanho mínimo de 46, procedimento chamado de “padding”.

O FCS consiste de quatro bytes. Essa sequência de bytes contém um CRC (cyclic redundancy check) de 32 bits que é criado pelo MAC que enviou o quadro, e recalculado pelo MAC receptor para checagem de quadros danificados. O FCS é gerado utilizando todos os bytes do quadro, com exceção do preâmbulo e o SOF.

Após o envio de um quadro, o MAC transmissor deve esperar um tempo equivalente à transmissão de 96 bits, antes de enviar um próximo quadro.

3.3: Parallel Redundancy Protocol (PRP)

A norma IEC 62439 oferece seis tipos diferentes de protocolos de redundância. Entre esses protocolos, o PRP foi escolhido pela IEC 61850, como solução de redundância para redes em subestações.

O PRP aplica o método de “redundância nos dispositivos” para oferecer redundância. Cada nó PRP (chamado de DANP por nó duplamente acoplado com PRP, “Doubly Attached Node with PRP”) é conectado a duas LANs que podem

apresentar diferentes topologias. As redes são completamente separadas e assume-se que elas são independentes quanto à falha.

Comparado a outros protocolos lançados na IEC 62439, PRP oferece transição transparente em caso de falha de conexões ou switches, atendendo o requisito de tempo real crítico para um sistema de automação de subestação. Outros protocolos como RSTP e MRP (Media Redundancy Protocol) necessitam de um tempo de recuperação.

No PRP, uma subcamada chamada “Link Redundancy Entity”, ou LRE, é adicionada a camada de enlace dados, acima do MAC. O LRE conecta a camada de rede aos dois controladores Ethernet e se comporta como se fosse uma única interface Ethernet. Quando transmitindo, o LRE adiciona o “Redundancy Check Trailer” (RCT) ao quadro e encaminham ambos os quadros para as duas portas do nó. Esses dois quadros navegam pela rede e chegam ao destino em tempos diferentes. Se o nó receptor está configurado no modo “duplicate accept”, o LRE encaminha os dois quadros do par para as camadas superiores. A rejeição do quadro duplicado deveria ocorrer nas camadas superiores. Nesse caso, a aplicação deveria ser capaz de rejeitar o quadro duplicado. Por exemplo, o TCP é feito para descartar pacotes duplicados. Aplicações usando UDP devem ser capazes de rejeitar duplicados, visto que é um protocolo sem conexão. Se o nó está configurado em modo “descartar duplicados”, o LRE deve encaminhar o primeiro quadro do par para as camadas superiores e rejeitar o segundo.

Para não sobrecarregar o processador, a função de rejeição de duplicados do LRE deve ser preferencialmente implementado em hardware.

A seguir será feito uma apresentação do protocolo PRP em detalhes conforme mostrado em [6].

3.3.1: Topologia de rede PRP

Esse protocolo de redundância implementa redundância nos nós ao invés de na rede, usando nós duplamente acoplados usando PRP (DANP).

Um DANP é a conectado a duas redes locais (LAN) de topologia similar, nomeadas LAN_A e LAN_B, as quais operam em paralelo. O nó de origem envia o mesmo quadro nas duas LANs e o nó destino recebe pelas duas LANs em tempos

diferentes, consumindo o primeiro e descartando o segundo. A Figura 4 demonstra um exemplo de topologia de rede redundante usando PRP.

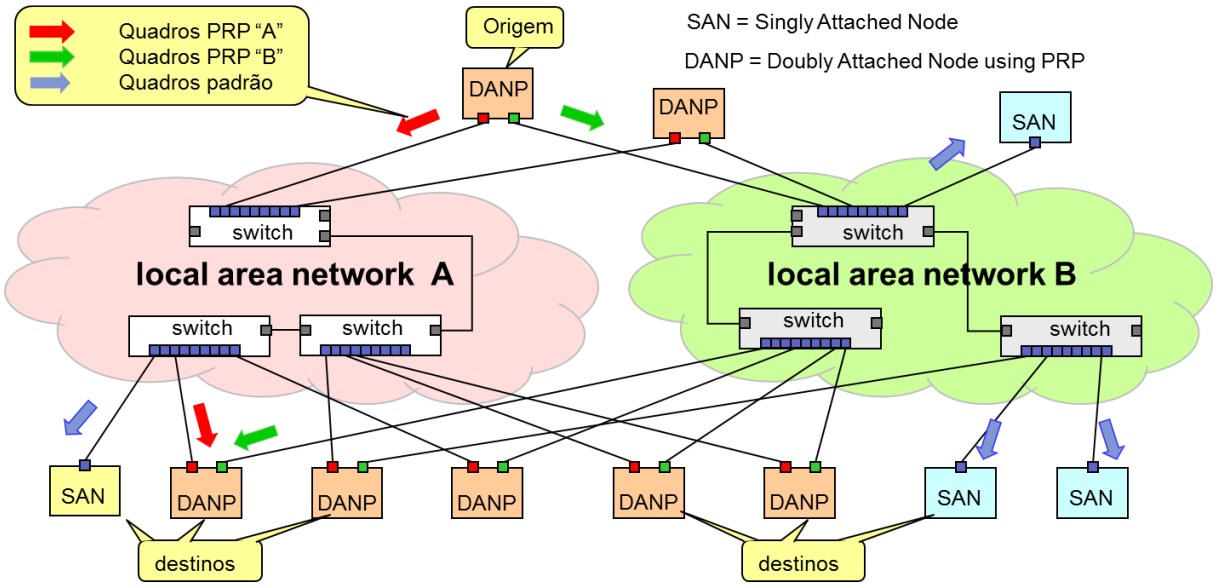


Figura 4: Exemplo de rede redundante usando PRP

As duas LANs são idênticas em protocolos em nível de MAC-LLC, mas elas podem se diferenciar em desempenho e topologia. Atrasos de transmissão também podem ser diferentes, principalmente se uma das redes se reconfigurar utilizando, por exemplo, o protocolo RSTP em caso de uma falha interna.

As duas LANs seguem regras de configuração que permitem que protocolos de manutenção de rede como ARP (Address Resolution Protocol) e SNMP (Simple Network Management Protocol) operem corretamente.

As duas LANs não possuem conexões entre elas e se assume que são independentes quanto a falhas internas. Redundância pode ser prejudicada por um ponto comum de falha, como uma fonte de energia compartilhada ou uma conexão direta que se falhar derruba as duas redes, por isso durante a montagem das redes deve-se tomar cuidado com esse tipo de problema.

3.3.2: Estrutura do nó PRP (DANP)

A Figura 5 mostra a estrutura do DANP. Cada nó possui duas portas de rede que operam em paralelo e que estão conectadas as mesmas camadas superiores da pilha de comunicação através do LRE.

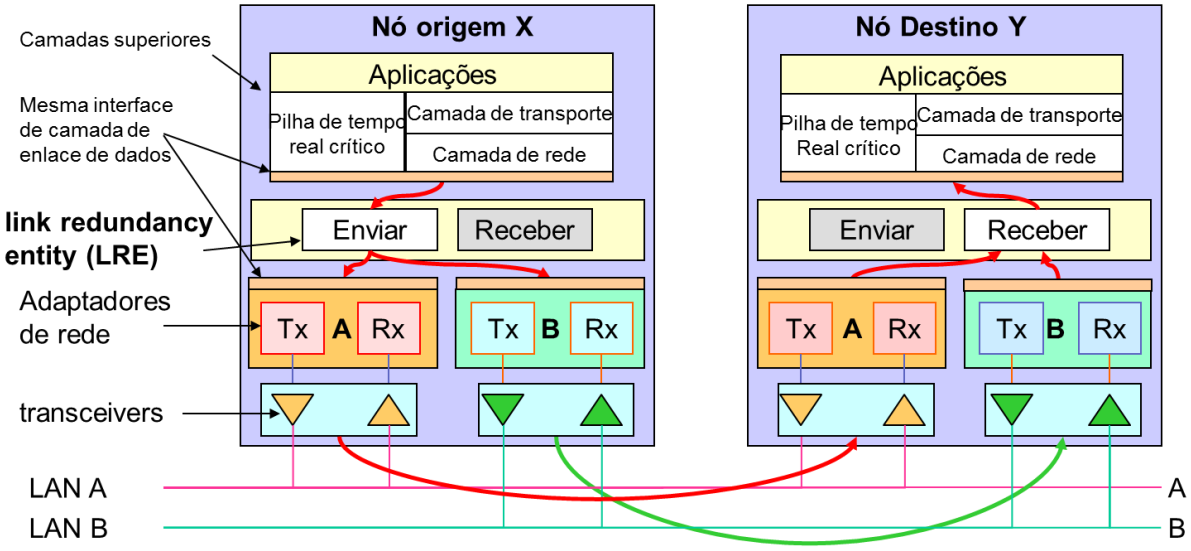


Figura 5: PRP com dois DANPs se comunicando

Para a comunicação, o LRE apresenta para as camadas superiores a mesma interface que um adaptador de rede não redundante, de tal modo que as camadas superiores não percebam a redundância.

O LRE possui duas tarefas: tratar os quadros duplicados e supervisionar a redundância.

Quando recebe um quadro das camadas superiores, o LRE adiciona ao quadro o RCT contendo um número de sequência e envia pelas duas portas ao mesmo tempo. Os dois quadros são quase idênticos, exceto pelo identificador de LAN e o checksum.

Os dois quadros navegam pelas redes com atrasos diferentes e idealmente chegariam ao mesmo tempo no nó destino.

Durante o recebimento dos quadros, o LRE encaminha o primeiro quadro que chegar às camadas superiores, e descarta o segundo se ele chegar. O LRE pode ser configurado para remover o RCT antes de encaminhar a mensagem.

O nó DANP possui o mesmo endereço MAC para os dois adaptadores de rede e apenas um conjunto de endereços IP para esse endereço, de tal forma que protocolos como ARP possam operar normalmente, e que o LRE seja transparente para as camadas superiores.

Para a manutenção da redundância e para checar os outros DANPs presentes na rede, o LRE envia periodicamente um quadro de supervisão PRP e também avalia os quadros que receber de outros nós.

3.3.3: Conectando nós que não utilizam PRP

Um nó que não utiliza PRP, conectado unicamente a rede (SAN), pode ser acoplado à rede PRP de duas maneiras. Um SAN pode ser conectado diretamente ou na LAN_A ou na LAN_B, pois DANPs geram um tráfego que SANs conseguem entender. SANs ignoram o RCT, pois não conseguem diferenciá-lo de um “padding” (preenchimento do quadro com zeros para que ele possa ter o tamanho mínimo para ser transmitido). Preferencialmente todos os SANs devem ser conectados a mesma LAN para que possam trocar mensagens entre si.

A outra maneira de conectar um SAN é através de um “Redundancy Box” (RedBox), que é um dispositivo que faz a transição de uma rede comum para a rede PRP. O RedBox recebe o quadro do SAN, duplica, insere o RCT e envia nas duas LANs. O RedBox recebe os quadros PRP da rede, repassa ao SAN e descarta os duplicados. O RedBox também envia quadros de supervisão em nome dos SANs os quais estão acoplados a ele. A estrutura do RedBox está apresentada na Figura 6.

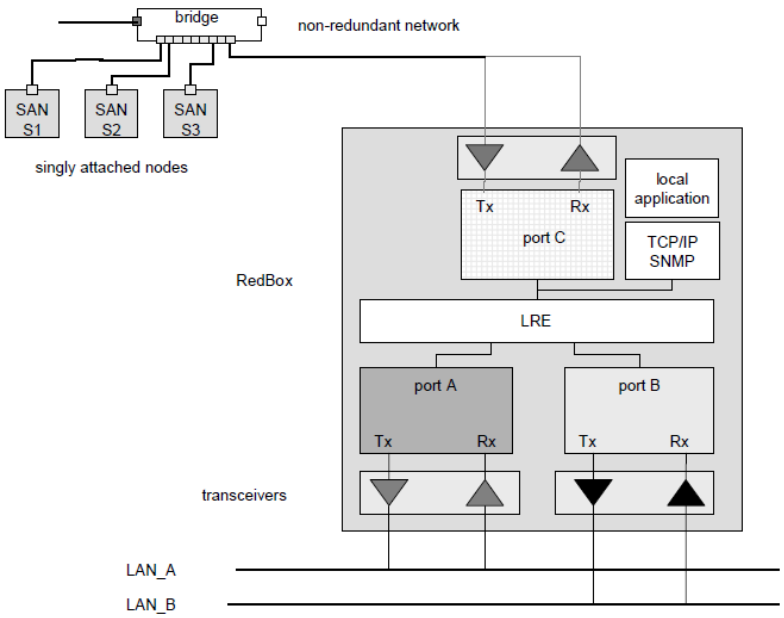


Figura 6: PRP RedBox, transição de rede comum para rede redundante

3.3.4: Tratamento de quadros duplicados

3.3.4.1: Duplicate Accept (modo teste)

O modo “Duplicate Accept” é usado somente para testes, para verificar se os quadros duplicados estão realmente sendo eliminados pela camada enlace e não por protocolos de camadas superiores.

Nesse modo, o nó de origem envia os quadros sem RCT, e o nó destino é configurado para aceitar os quadros sem checar duplicação.

3.3.4.2: Duplicate Discard

Para permitir que os receptores detectem quadros duplicados, o nó de origem adiciona um campo de seis bytes que contém um número de sequência em ambos os quadros de um par que enviar, fazendo que o quadro tenha o formato mostrado na Figura 7.

O LRE do receptor usa o número de sequência do RCT e o endereço MAC de origem para detectar duplicações. O LRE encaminha apenas o primeiro quadro do par para as camadas superiores.

Como opção, o receptor pode remover o RCT antes de encaminhar o quadro.

O RCT consiste dos seguintes campos:

- Número de sequência (SeqNr): 16 bits.
- Identificador de LAN (LanId): 4 bits.
- Tamanho do quadro (LSDUsize): 12 bits.
- Sufixo (PRPsuffix): 16 bits.

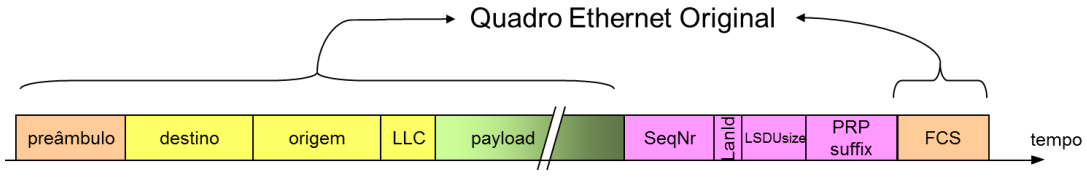


Figura 7: Quadro PRP

O PRPsuffix identifica quadros PRP e os diferencia de outros protocolos que também adicionam um “trailer” aos seus dados.

O LSDUsize contém o tamanho do LSDU mais o RCT em octetos conforme representado na Figura 8. Esse campo adiciona mais uma informação que diferencia o quadro PRP de um quadro comum.

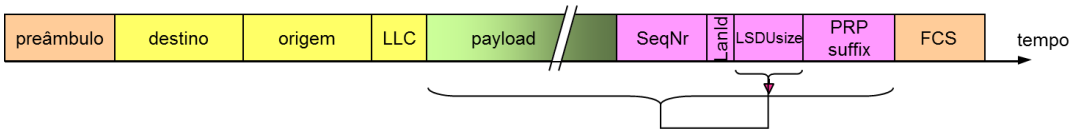


Figura 8: Campo LSDUsize

Se o receptor detecta que o final do quadro contém o sufixo PRP correto, os doze bits que antecedem o sufixo PRP correspondem ao tamanho da LSDU mais o RCT, e o identificador de LAN corresponde a LAN a qual a porta receptora está conectada, o quadro é um candidato a ser descartado.

Como quadros pequenos precisam de preenchimento (padding) para atender o tamanho mínimo de 64 octetos, o nó de origem sempre inclui o preenchimento fazendo com o quadro fique com o formato representado na Figura 9. O tamanho mínimo do quadro passa a ser de 70 octetos, já que o LRE ou o RedBox podem remover o RCT.

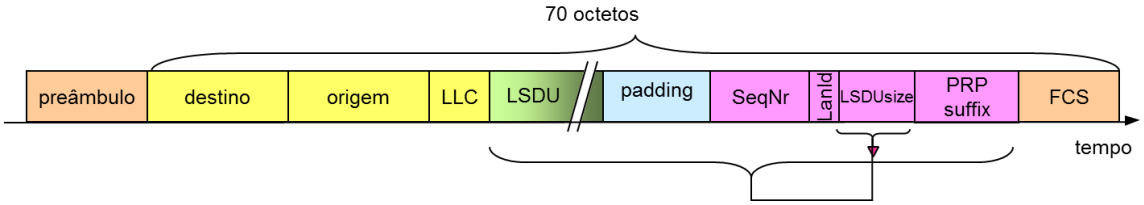


Figura 9: Quadro PRP com preenchimento (padding)

O campo LanId do RCT carrega um identificador distinto para LAN_A ou LAN_B, especificamente os códigos 1010 (“A”) e 1011 (“B”). Portanto, os quadros A e B diferem em um bit e no FCS. O receptor verifica se o quadro veio da LAN correta. O DANP não rejeita um quadro que chegou pela LAN incorreta, pois pode ser um quadro legítimo que possui a informação correta de sufixo PRP e tamanho em seus últimos 28 bits, mas incrementa os contadores de erro, pois isso indica um erro de configuração. Como esse é um tipo de erro permanente, é rapidamente detectado.

O campo SeqNr é incrementado por um para cada quadro que o DANP envia. A dupla {endereço MAC de origem, número de sequência} unicamente identifica

cópias do mesmo quadro. A norma não especifica um algoritmo para identificar cópias, mas apenas impõe as seguintes regras:

- Nunca descartar um quadro legítimo como cópia.
- Descartar quase todos os quadros duplicados

Como o SeqNr possui somente 16 bits, ele reinicia a contagem após 65536 quadros. Essa característica, associado com o fato de que o receptor não irá receber quadros com o número de sequência crescendo continuamente, exige que entradas mais antigas que um determinado tempo devam ser esquecidas pelo algoritmo de detecção de duplicados. Isso também permite que um nó possa reiniciar em qualquer posição do número de sequência.

A Figura 10 demonstra como o reinício da contagem do SeqNr afeta a detecção de duplicados.

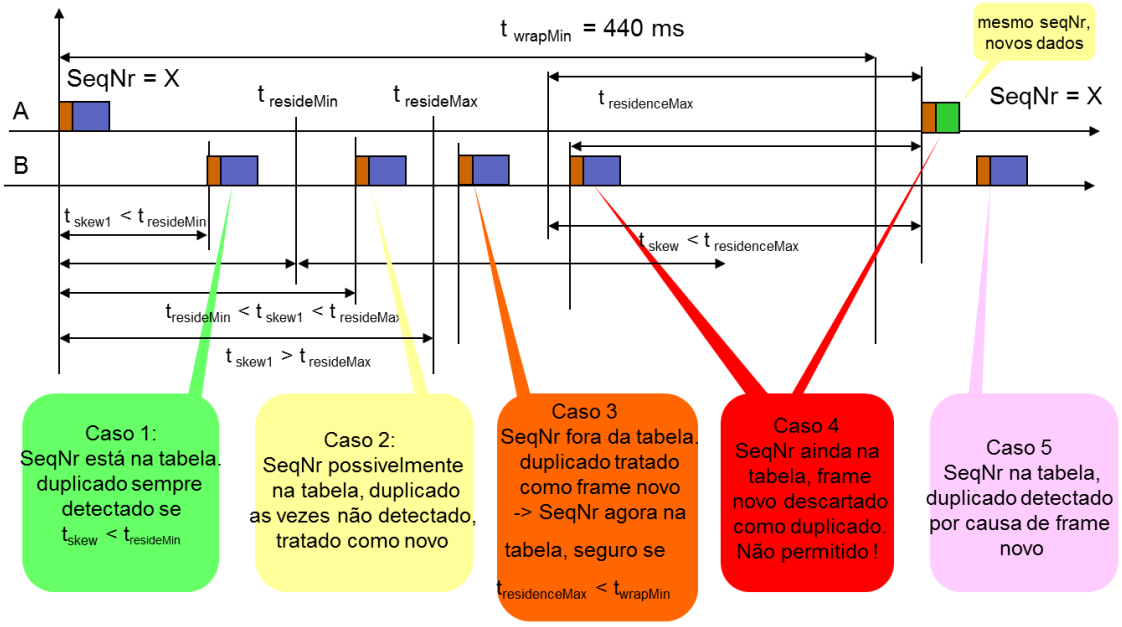


Figura 10: Limites do algoritmo de detecção de quadros duplicados

O $t_{wrapmin}$ é o tempo mínimo possível entre duas repetições do mesmo número de sequência por quadros legítimos após 65536 incrementos (depende das características da aplicação e da rede). Em uma rede de 100Mbit/s, $t_{wrapmin}$ é, teoricamente, de aproximadamente 440 ms.

O t_{skew} é a diferença de tempo entre a chegada de duas cópias do mesmo quadro.

O $t_{residenceMax}$ é o tempo máximo que uma entrada é utilizada no algoritmo de descarte de duplicações, enquanto $t_{residenceMin}$ é o tempo mínimo.

- Regra para descarte confiável: $t_{skew} < t_{residenceMin}$.
- Regra para liberação segura: $t_{residenceMax} < t_{wrapmin}$.

Quando um nó reiniciar, ele deve ficar um tempo maior que $t_{residenceMax}$ sem enviar quadros, para dar tempo que os outros nós envelheçam suas entradas no algoritmo de descarte de duplicados.

Durante o envio do quadro, o DANP deve proceder da seguinte maneira:

Se *o nó destino não é está registrado na tabela de nós como SAN.*

> ***Se*** *o quadro possui uma etiqueta VLAN e possui tamanho menor que 64 octetos (sem FCS).*
>
> > *Preencher o quadro (padding) para 64 octetos.*
>
> ***Se*** *o quadro não possui uma etiqueta VLAN e possui tamanho menor que 60 octetos.*
>
> > *Preencher o quadro (padding) para 60 octetos.*
>
> *Incrementar o número de sequência.*
>
> *Inserir o RCT com LanId "A" para porta A, e LanId "B" para porta B.*
>
> *Inserir o FCS.*
>
> *Enviar o quadro com LanId "A" pela porta A e o quadro com LanId "B" pela porta B.*

Senão

> *Enviar o quadro pela porta que o SAN está registrado.*

Durante a recepção, o DANP deve proceder da seguinte maneira:

Se *o quadro contém erro.*

> *Incrementar o contador de erro respectivo à porta em que foi recebido.*
>
> *Ignorar o quadro.*

Se *o quadro é uma duplicação*

> *Descartar.*

Senão

> *Encaminhar para as camadas superiores.*

3.3.5: Supervisão da redundância

O status de cada LAN é os dispositivos conectados a elas é monitorado, senão redundância não iria ajudar muito.

O DANP receptor confere se os quadros são corretamente recebidos em ambas às portas. O DANP mantém uma tabela de nós, onde registra o endereço MAC dos dispositivos dos quais já recebeu quadros, contadores de erro e o tempo

da ultima vez que recebeu um quadro de um determinado nó. Essa tabela pode ser acessada pelo supervisor da rede via SNMP. O formato da tabela é a seguinte:

Argumento	Definição	Tipo dos dados
MacAddress	Endereço MAC do nó (6 octetos)	OctetString6
CntReceivedA	Número de quadros recebidos desse nó pela LAN_A	Unsigned32
CntReceivedB	Número de quadros recebidos desse nó pela LAN_B	Unsigned32
CntErrWrongLanA	Número de quadros que foram recebidos com identificador de LAN incorreto pela LAN_A	Unsigned32
CntErrWrongLanB	Número de quadros que foram recebidos com identificador de LAN incorreto pela LAN_B	Unsigned32
TimeLastSeenA	Tempo da última vez em que foi recebido um quadro desse nó pela LAN_A	TimeTicks
TimeLastSeenB	Tempo da última vez em que foi recebido um quadro desse nó pela LAN_B	TimeTicks
SanA	Verdadeiro se o nó é provavelmente um SAN acessível pela LAN_A	Boolean1
SanB	Verdadeiro se o nó é provavelmente um SAN acessível pela LAN_B	Boolean1
Operation Mode	Indica o modo de operação “Duplicate Discard” ou “Duplicate Accept”, cujos valores são 20 e 21 respectivamente.	Unsigned5

Tabela 1: Tabela de nós

Uma entrada da tabela de nós é removida quando o nó fica um tempo maior que *NodeForgetTime* (60s por default) sem receber quadros pela LAN A e LAN B com endereço de origem igual ao atributo *MacAddress* da entrada. Cada DANP envia periodicamente um quadro de supervisão PRP que permite a checagem da integridade da rede e a presença dos nós, contribuindo para a formação da tabela de nós. O quadro de supervisão permite ao mesmo tempo identificar quais nós são DANP. O formato do quadro está abaixo:

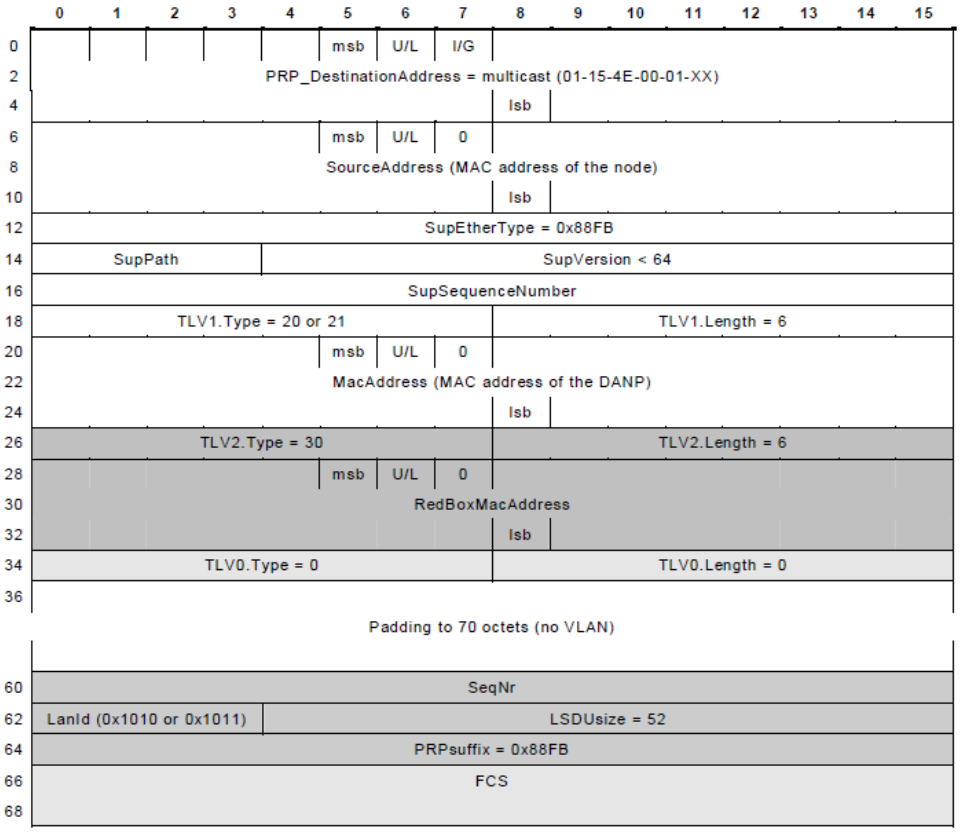


Figura 11: Formato do quadro de supervisão PRP

Abaixo está a descrição do conteúdo do quadro:

PRP_destinationAddress	Endereço Multicast reservado para esse protocolo: 01-15-4E-00-01-XX. O valor padrão para XX é “00”, caso o endereço esteja ocupado, pode ser qualquer outro entre “00” e “FF”.
SourceAddress	Endereço MAC do nó
SupEtherType	Ethertype 0x88FB reservado para esse protocolo
SupPath	Reservado. Sempre 0
SupVersion	Versão do protocol. Mantido em 1 para essa versão.
SupSequenceNumber	Número de sequência, incrementado em 1 para cada quadro de supervisão enviado
TLV1.Type	Modo de operação. 20 para “Duplicate Discard” e 21 para “Duplicate Accept”
TLV1.Length	Comprimento do endereço MAC seguinte. Sempre 6.
MacAddress	Endereço MAC usado pelas duas portas
TLV2.Type	Não usado se não for um RedBox
TLV2.Length	Não usado se não for um RedBox
RedBoxMacAddresses	Não usado se não for um RedBox
TLV0.Type	Sempre 0
TLV0.Length	Sempre 0

Tabela 2: Descrição do quadro de supervisão

3.4: Conclusão

O capítulo 3 apresentou o funcionamento do protocolo e mostrou que ele exige a criação da subcamada da camada de enlace de dados chamada LRE. Como o foco desse trabalho é a criação de um dispositivo capaz de operar o PRP, consequentemente o objetivo passa a ser a criação do LRE.

Nos próximos capítulos será apresentado todo o procedimento utilizado para implementar o LRE.

Capítulo 4: Especificação do LRE

Este capítulo é dedicado em apresentar os requisitos de implementação do LRE, a subcamada da camada de enlace de dados responsável por operar a redundância PRP, e apresentar a arquitetura do sistema computacional utilizado para desenvolvê-lo.

4.1: Requisitos do LRE

4.1.1: Requisitos Funcionais

O LRE deve ser capaz de enviar e receber quadros em dois adaptadores de rede, mais especificamente da subcamada MAC, operando com o mesmo endereço MAC.

O LRE deve ser capaz de prover a mesma interface para as camadas superiores de rede que um MAC ofereceria, e conseguir enviar e receber quadros das camadas superiores.

Na transmissão de quadros, o LRE deve receber os quadros a serem transmitidos, duplicar o quadro, inserir o RCT e enviar para subcamada MAC para posterior transmissão na rede. O LRE deve ser capaz de identificar no quadro o endereço destino, e no caso de endereços unicast, verificar na tabela de nós em qual LAN o endereço destino está conectado. Se o endereço destino for um SAN conectado na LAN A, o LRE deve enviar o quadro somente na LAN A, e se o SAN estiver na LAN B, enviar o quadro somente na LAN B. Caso o endereço destino seja um DANP, enviar o quadro nas duas portas com a respectiva identificação da LAN no RCT.

Na recepção de quadros, o LRE deve ser capaz de receber os quadros dos dois MACs, checar se o RCT está corretamente formado, e em caso de um RCT correto, obter o endereço de origem e o número de sequência e usá-los para checar quadros duplicados. Caso o quadro já tenha sido recebido, ele deverá ser descartado. Caso o quadro não tenha sido recebido ainda, o LRE deverá

encaminhá-lo para as camadas superiores, e retirar o RCT caso seja configurado para isso.

O LRE deve enviar um quadro de supervisão em certa frequência (alguns segundos, a ser configurado pelo usuário) com o formato especificado na seção 3.3.5.

Com as informações dos quadros recebidos, o LRE deve construir uma tabela de nós conforme especificado na seção 3.3.5.

O algoritmo de detecção de duplicados não pode descartar um quadro não recebido ainda como se fosse uma duplicação.

4.1.2: Requisitos não funcionais

O LRE deve ser capaz de tratar um volume de dados que corresponda no mínimo ao dobro da banda da rede, devido às duas portas utilizadas. Nesse projeto o LRE foi projetado para uma rede 100Mbit/s, portanto o LRE deve ser capaz de tratar 200Mbit/s.

Uma subestação possui tipicamente entre 20 e 40 IEDs, portanto o algoritmo de detecção de duplicados deve ser capaz de identificar pelo menos 90 por cento das duplicações em uma rede com esse número de IEDs em operação normal (atrasos entre mensagens enviadas na LAN A e LAN B muito pequenos).

O LRE deve ser capaz de tratar quadros mal formados ou com erros de checksum que forem recebidos tanto dos adaptadores de rede quanto das camadas superiores.

4.2: Tecnologia utilizada

Com base nos requisitos do LRE e também na tecnologia em que os novos produtos da empresa são construídos, optou-se por implementar o LRE em hardware e não software, utilizando um FPGA (Field Programmable Gate Array). A implementação em hardware permite aliviar o processador, evitando um processamento duplicado de quadros, e a escolha por FPGA é consequência do fato dos novos produtos serem desenvolvidos utilizando essa tecnologia.

O LRE foi realizado utilizando um FPGA Cyclone IV E da fabricante Altera. O LRE é, portanto, um circuito lógico digital.

Por ser implementado em FPGA e por ser um circuito lógico digital, alguns outros requisitos não funcionais são adicionados:

- O LRE como circuito digital bem construído, deve possuir atrasos de propagação dentro dos limites do clock utilizado de 100 MHz, e respeitar os requisitos temporais de funcionamento dos elementos de memória (flip-flops basicamente).
- O LRE deve respeitar os limites de elementos lógicos e memória disponíveis no FPGA.

Para desenvolvimento do trabalho, utilizou-se um kit de desenvolvimento que além do FPGA, possui LEDs, dois transceivers para rede Ethernet, display LCD, display sete segmentos, interface USB, memória SDRAM, memória Flash e outros elementos que não serão citados, pois não foram utilizados nesse trabalho.

4.2.1: FPGA

Lógica digital [10] mapeia um conjunto infinito de valores reais de uma grandeza física em apenas dois subconjuntos correspondendo a apenas dois possíveis valores lógicos – 0 e 1. Como resultado, circuitos de lógica digital podem ser analisados e desenvolvidos funcionalmente, usando álgebra booleana, tabelas e outros meios abstratos para descrever a operação de 0s e 1s em um circuito.

Um circuito lógico cujas saídas dependem apenas das entradas é chamado de circuito combinacional. Apenas três funções lógicas (também chamado de portas lógicas) são necessárias para construir qualquer circuito combinacional: AND, OR e NOT.

Um circuito com memória, cujas saídas dependem da entrada atual e da sequência de entradas passadas é chamado circuito sequencial. Um circuito sequencial é realizado por portas lógicas, flip-flops e latches. Flip-flop é um circuito sequencial que faz a amostragem de suas entradas e altera suas saídas somente em tempos determinados por um sinal de clock. Latch é um circuito sequencial que observa suas entradas continuamente e muda suas saídas a qualquer momento, independente do sinal de clock.

Um FPGA é um circuito integrado desenvolvido para ser configurado pelo cliente ou desenvolvedor após produção.

FPGA possui componentes lógicos programáveis chamados “blocos lógicos”, e uma hierarquia de interconexões que permitem os blocos serem conectados entre si. Os blocos lógicos podem ser configurados para realizar lógicas combinacionais complexas, ou apenas portas lógicas. Na maioria dos FPGAs, os blocos lógicos possuem também elementos de memória, os quais podem ser simples flip-flops ou blocos de memória mais complexos, podendo implementar lógicas sequenciais.

A configuração de um FPGA é geralmente feita usando HDL (Hardware Description Language). A habilidade de reconfiguração de um FPGA, e facilidade de não precisar “criar” um circuito integrado, mas apenas configurar, faz com que a utilização de FPGAs seja bastante vantajosa, pois reduz tempo de desenvolvimento de produto.

4.2.2: O kit de desenvolvimento

Para desenvolver o LRE foi utilizado o kit de desenvolvimento DE2-115 da Terasic, mostrado na Figura 12. Esse kit de desenvolvimento foi escolhido para o projeto por possuir dois adaptadores de rede, permitindo realizar a lógica do PRP.

Desse kit de desenvolvimento foi utilizado o FPGA cyclone IV E, os dois transceivers Gigabit Ethernet, o display LCD, os LEDs, os displays sete segmentos, uma memória SDRAM, a memória SRAM, a memória FLASH e os botões.

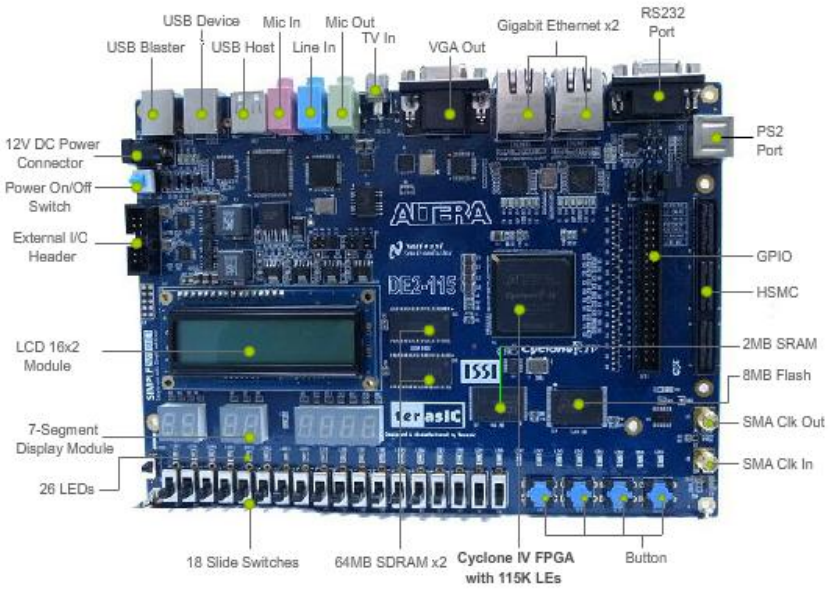


Figura 12: Kit de desenvolvimento DE2-115

4.3: Arquitetura da plataforma de desenvolvimento

Na Figura 13, é apresentada a arquitetura do sistema computacional proposta nesse trabalho e utilizada para desenvolver e validar o LRE, sendo toda ela criada no mesmo FPGA do kit de desenvolvimento. Essa arquitetura foi escolhida por ser próxima do encontrado nos novos produtos sendo desenvolvidos pela empresa, em especial a interface de rede (MACs e SGDMAs) que é igual. Todos os componentes, com exceção do LRE, são IPs (Intelectual Properties) da Altera e não foram criados nesse trabalho.

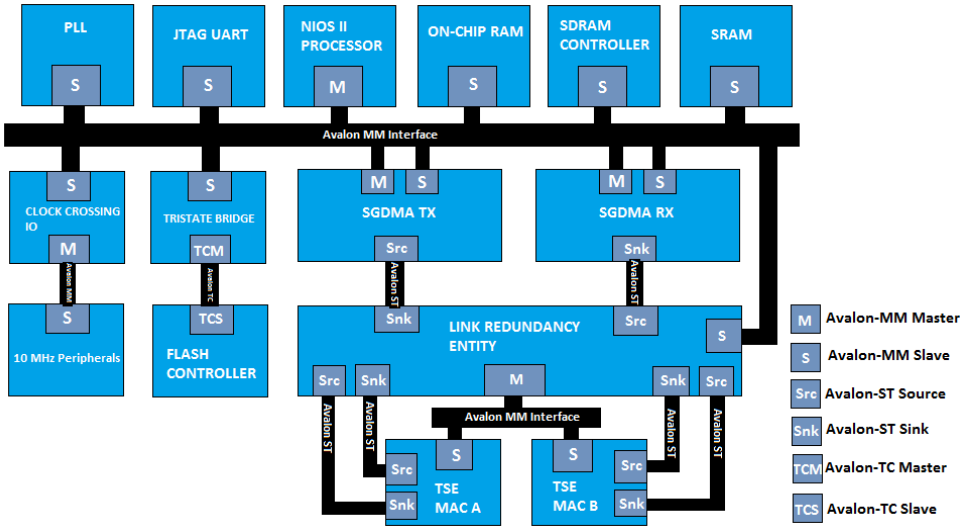


Figura 13: Arquitetura do sistema utilizado no desenvolvimento do LRE

As camadas superiores do modelo OSI são implementadas no processador NIOS II da Altera. Durante o desenvolvimento do LRE, o processador foi programado para operar um servidor web, que permite ao usuário acessar componentes (LEDs, display LCD, display sete segmentos) do kit de desenvolvimento, via internet. Escolheu-se por utilizar um servidor web, pois o mesmo permite uma boa validação da comunicação de rede (transmissão/recepção de pacotes, funcionamento de protocolos como ARP, DHCP, TCP etc.), utilizando PRP.

.Por ser inteiramente construído utilizando blocos lógicos de FPGA, o NIOS é considerado um processador softcore. A arquitetura utilizada é de 32 bits, ou seja, os barramentos de dados e de endereço utilizados pelo processador são de 32 bits.

O NIOS utiliza um barramento Avalon-MM (Avalon Memory Mapped Interface) para acessar os seguintes componentes: PLL, JTAG UART, interface de componentes do kit de desenvolvimento (LEDs, display sete segmentos, display LCD), controlador memória flash, controlador SDRAM, memória RAM on-chip, SGDMAs, SRAM e LRE. Nessa comunicação o NIOS é o mestre e os componentes são escravos.

A PLL é o componente responsável por gerar o clock de funcionamento do sistema (100 MHz). A PLL possui uma interface Avalon-MM escravo para poder ser acessada pelo processador.

A JTAG UART é o componente responsável por permitir a programação do processador.

Os componentes que controlam o display LCD, LEDs e display sete segmentos operam a 10 MHz, portanto o processador precisa de uma interface entre os sistemas que operam a clocks diferentes. Essa interface é o clock crossing IO.

A memória flash, a SDRAM, a SRAM e a memória RAM on-chip são acessadas e utilizadas pelo processador para executar o programa em que ele foi configurado.

As SGDMAs são as responsáveis por enviar e receber quadros dos MACs. Os quadros a enviar são lidos da memória SDRAM, onde o processador grava os pacotes. Os quadros recebidos são gravados na SDRAM, onde NIOS faz posteriormente a leitura. As SGDMAs também possuem uma interface Avalon-MM

mestre, utilizado para acessar a memória SDRAM. O fato de existir dois mestres para um mesmo barramento é uma das particularidades da interface Avalon-MM.

Em uma operação normal, os SGDMAs são a interface da camada de rede com os MACs, enviando e recebendo quadros deles. Na implementação do PRP, o LRE é criado para realizar as funcionalidades descritas na seção 3.3, e as SGDMAs agora se comunicam com o ele e não mais com os MACs. O LRE possui uma interface Avalon-MM escravo para poder ser configurado pelo NIOS e interfaces Avalon-ST (Avalon Streaming Interface) para comunicação com as SGDMAs e com os MACs.

Os TSE MACs (Triple Speed Ethernet MACs) são IPs desenvolvidos pela Altera que realizam a função da subcamada MAC da camada de enlace de dados, enviando e recebendo quadros dos transceivers. Os TSE MACs possuem interface Avalon-MM escravo para poderem ser configurados pelo LRE. O barramento Avalon-MM utilizado para configuração dos MACs é diferente do barramento utilizado pelo NIOS para configurar outros componentes.

4.3.1: A interface Avalon

O barramento Avalon [11] é uma interface de interconexão de componentes, desenvolvido para FPGAs produzidos pela Altera.

Muitos componentes utilizados nesse trabalho (como por exemplo, os MACs) são IPs da Altera e utilizam o barramento Avalon para troca de dados.

O Avalon fornece dois tipos de interface: Avalon Streaming Interface (Avalon-ST) e Avalon Memory Mapped Interface (Avalon-MM). O Avalon-ST é uma interface que suporta fluxo unidirecional de dados. O Avalon-MM é uma interface baseada em endereçamento para leitura/escrita, típico pra conexões mestre-escravo.

Avalon-ST é interessante para componentes que necessitam de banda elevada e cujo fluxo de dados é unidirecional. A transmissão de dados pode ser de forma continua, sem definição de pacotes, ou em pacotes.

Na Figura 14, temos um exemplo de transmissão de pacote utilizando Avalon-ST. Para iniciar a transmissão, a fonte dos dados (Avalon-ST Source) seta o sinal de inicio do pacote (SOF – Start of Packet), e disponibiliza os dados (data) no circuito de dados. Caso o nível de “ready” seja baixo, o componente fonte deve manter os

dados no próximo ciclo de clock, pois isso sinaliza que o receptor (Avalon-ST Sink) não está pronto para recebê-los. Assim que o sinal de “ready” for alto, a fonte deve, no próximo ciclo, fornecer novos dados e continuar esse ciclo até o final da transmissão do pacote. Quando os últimos dados do pacote forem transmitidos, o sinal de fim do pacote (EOF – End of Packet) é setado.

O tamanho do barramento de dados utilizado nesse trabalho foi de 32 bits, divididos em 4 símbolos de 8 bits. O sinal de “empty” é utilizado no final da transmissão dos pacotes para sinalizar quantos símbolos devem ser ignorados na última rodada de dados.

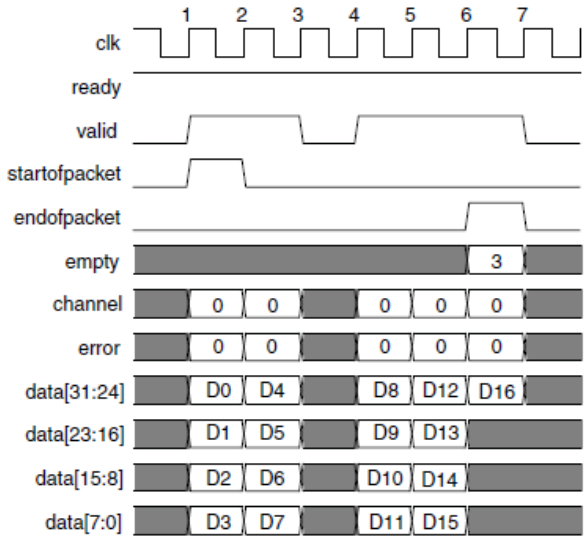


Figura 14: Exemplo de transferência de dados na interface Avalon-ST

O Avalon-MM é usado em sistemas baseado em trocas de informação entre mestre e escravos. Cada escravo e suas funcionalidades são mapeados para algumas faixas de endereço, os quais são utilizados pelo mestre para poder acessá-los.

A Figura 15 exemplifica processos de leitura e escrita no Avalon-MM. Quando o mestre deseja ler os dados de um endereço, ele disponibiliza o endereço no barramento “address” e seta o sinal “read”. Enquanto o escravo manter o sinal “waitrequest” setado, o mestre deve manter os valores de “address” e “read”. Quando o sinal “waitrequest” for baixo, os dados estarão disponíveis em “readdata”. Para escrita, o mestre disponibiliza o endereço, seta o sinal “write” e fornece os dados a serem escritos em “writedata”. O mestre deve manter os valores de “address” e “write” até “waitrequest” for baixo, sinalizando que a escrita finalizou. O

tamanho do barramento “writedata” e de endereçamento utilizado nesse trabalho foi de 32 bits.

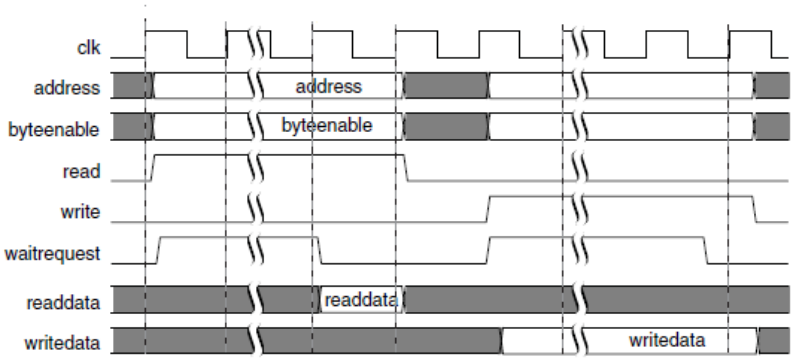


Figura 15: Exemplo de transferência de dados na interface Avalon-MM

4.4: Conclusão

Este capítulo listou os requisitos de implementação do LRE, os quais se basearam na norma PRP e também na tecnologia escolhida para criar o LRE, também apresentada nesse capítulo.

No próximo capítulo será mostrado como o LRE foi projetado nesse trabalho, apresentando a estrutura proposta e o funcionamento de cada componente do circuito lógico digital.

Capítulo 5: Projeto do LRE

Nesse capítulo será apresentado, em detalhes, o projeto do circuito digital do LRE implementado em um FPGA. Diagramas são utilizados para apresentar os módulos do circuito digital assim como o relacionamento entre eles, enquanto máquinas de estados são utilizadas para detalhar o funcionamento de cada componente do LRE.

As máquinas de estado são apresentadas de maneira mais simplificada, omitindo alguns detalhes de implementação (troca de sinais na interface Avalon-ST, troca de sinais no acesso a memória) e focando mais na funcionalidade, com o objetivo de facilitar a compreensão.

5.1: Arquitetura do LRE

Após a compreensão da norma PRP, da especificação dos requisitos do projeto e da escolha da tecnologia utilizada (FPGA), a próxima etapa do trabalho é projetar o circuito digital.

Para atender os requisitos do projeto e para realizar todas as funcionalidades necessárias para implementação da subcamada LRE do PRP, foi proposto nesse trabalho que o circuito digital do LRE tenha a estrutura mostrada na Figura 16.

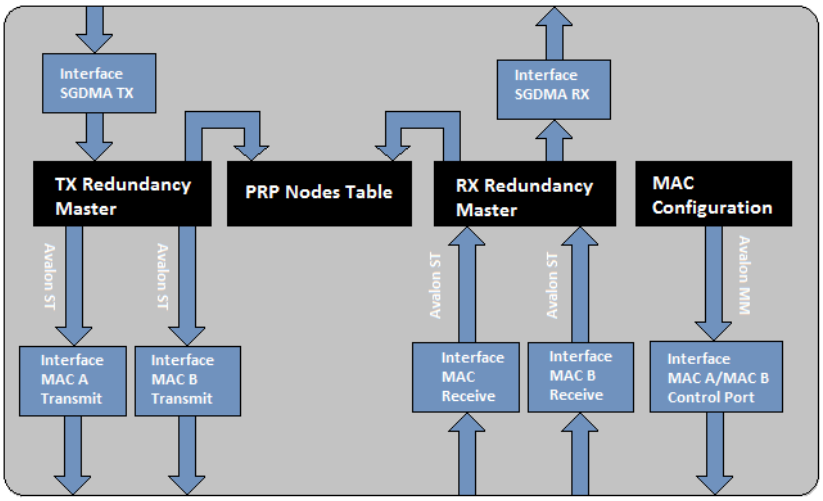


Figura 16: Estrutura do LRE

O LRE é formado por 4 componentes (cada componente é um circuito lógico digital) representados pelos blocos pretos do diagrama: “TX Redundancy Master”, “PRP Nodes Table”, “Rx Redundancy Master” e “MAC Configuration”. Esses componentes estão conectados e trocando dados entre eles e também com módulos externos ao LRE (a conexão com módulos externos é representada pelos blocos azuis).

As setas indicam as conexões entre os módulos, e o sentido delas indica o sentido do fluxo de dados e de acesso (solicitação da realização de alguma operação). O módulo receptor dos dados ou que está sendo acessado gera sinais de resposta durante as transações para o módulo transmissor, para que este possa saber os resultados de uma operação e quando ela foi finalizada. Esses sinais de resposta não são representados no diagrama.

O bloco “TX Redundancy Master” é responsável pela transmissão de pacotes PRP. Ele recebe quadros do componente “SGDMA TX” através de uma interface Avalon ST, duplica, insere o RCT e envia para os MACs A e B, também utilizando interface Avalon-ST, das portas A e B respectivamente. Antes de enviar um quadro, o bloco “TX Redundancy Master” consulta a tabela de nós (PRP Nodes Table), para conferir se o endereço destino das mensagens está na rede A ou B, ou ambas em caso de um DANP. O quadro é enviado nas redes em que o nó destino está conectado.

O bloco “PRP Nodes Table” reproduz a tabela de nós, e os processos de procura, inserção e atualização da tabela.

O bloco “RX Redundancy Master” é responsável pela recepção de pacotes PRP. Ele recebe os quadros dos MACs A e B (utiliza-se interface Avalon-ST), encaminha o primeiro quadro de um par para o “SGDMA RX” e descarta o segundo. O “RX Redundancy Master” também solicita a atualização da tabela de nós, toda vez que um quadro é recebido.

O bloco “MAC Configuration” é responsável pela configuração dos MACs, utilizando a interface Avalon-MM. Esse módulo não será apresentado em detalhes nesse trabalho, pois não realiza funções do PRP.

Nas seções 5.2, 5.3, 5.4 será apresentada a estrutura interna de “TX Redundancy Master”, “RX Redundancy Master” e “Nodes Table”.

5.2: TX Redundancy Master

A Figura 17 mostra a estrutura do “TX Redundancy Master”. O “TX Redundancy Master” é formado por 8 componentes: memória RAM, “Memory Access Control”, “TX Sink”, “Arbiter”, “TX Source”, “Avalon-ST Arbiter”, “Supervision Frame” e “Avalon-ST Source Control”.

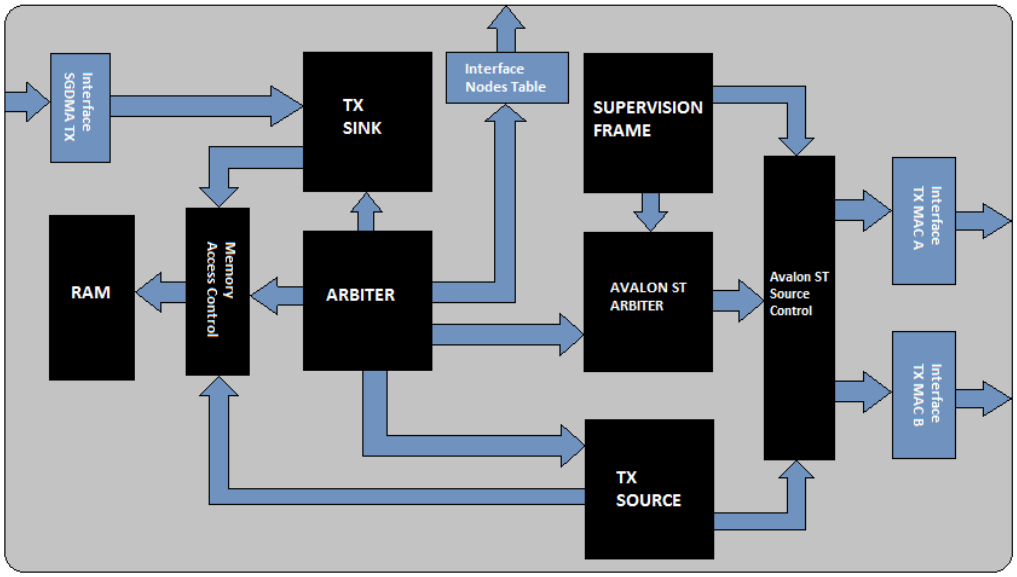


Figura 17: Estrutura do TX Redundancy Master

O componente “Memory Access Control” é um multiplexador que controla o acesso à memória RAM. O bloco “Arbiter” indica ao multiplexador se o “TX Sink” ou o “TX Source” irá acessar a memória.

O componente “Avalon-ST Source Control” também é um multiplexador que, nesse caso, seleciona se o componente “Supervision Frame” ou o “TX Source” irá acessar as interfaces Avalon-ST que comunicam com os MACs, baseado na decisão de “Avalon-ST Arbiter”.

O componente “TX Source” cuida da duplicação, inserção do RCT e transmissão dos quadros recebidos do “SGDMA TX”. Os quadros vindos do “SGDMA TX” são recebidos pelo componente “TX Sink” através de uma interface Avalon-ST, e salvos na RAM.

O bloco “Supervision Frame” se encarrega da transmissão de quadros de supervisão PRP.

O componente “Arbiter” é responsável por coordenar quem acessa a memória RAM. Ele libera o “TX Sink” para receber um quadro, e após a recepção do mesmo, libera o bloco “TX Source” para transmissão. Enquanto “TX Source” envia um quadro, “TX Sink” não recebe quadros por não ter acesso à memória. O “Arbiter” também acessa a tabela de nós para consultar em qual rede (LAN A ou B) o endereço destino se encontra e assim, enviar um quadro para a rede correta em caso de um SAN, ou para as duas em caso de um DANP.

5.2.1: TX Sink

O componente “TX Sink” possui como entradas os sinais da interface Avalon-ST Sink (valid, data, empty, error) e “read_frame”. O sinal “read_frame” é proveniente do componente “Arbiter” e libera o circuito para a recepção de um quadro do “SGDMA TX”. O funcionamento de “TX Sink” está representado na Figura 18.

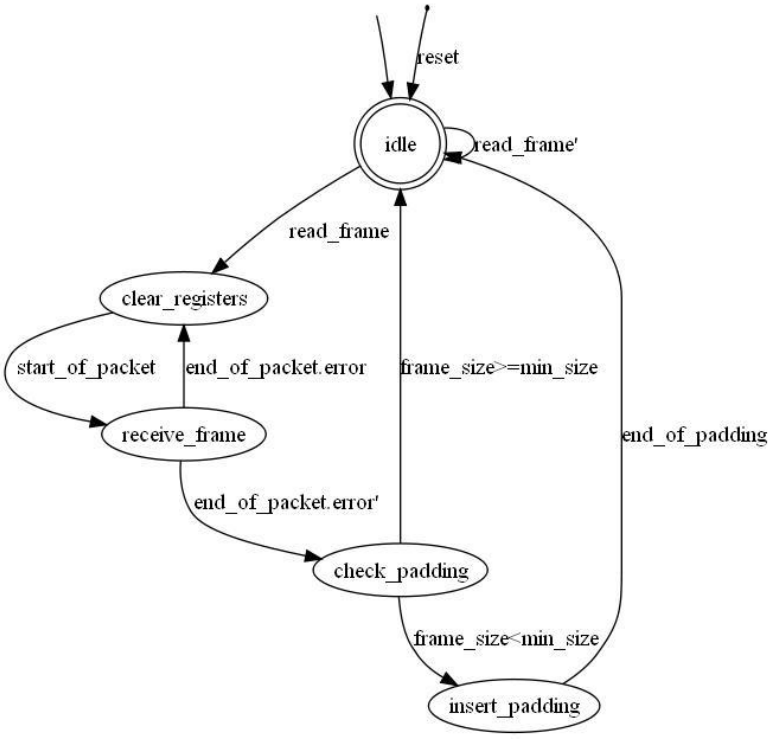


Figura 18: Máquina de estados de TX Sink

O recebimento do quadro acontece no estado “receive_frame”. À medida que o quadro é recebido, ele é guardado na memória e informações como endereço destino, tamanho do quadro, tamanho do payload são guardadas em registradores.

O “TX Source” insere “padding” caso seja necessário para o quadro obter o tamanho mínimo de 60 bytes (sem preâmbulo e checksum).

As saídas desse componente são: os sinais de acesso à memória, o endereço destino, o tamanho do quadro, sinais da interface Avalon-ST (ready) e um sinal indicando que o recebimento foi finalizado (“TX Source” está em “idle”).

O funcionamento da interface Avalon-ST e do acesso à memória não foram representados nessa máquina de estados.

5.2.2: TX Source

O componente “TX Source” possui como entradas: os sinais da interface Avalon-ST Source (ready) provenientes dos MACs A e B, o sinal “send frame”, o barramento de dados de leitura da memória (“read data”), o sinal “frame size”, e os sinais “send port A” e “send port B”. O sinal “send frame”, proveniente do “Arbiter”, indica que o componente deve enviar quadros para os MACs indicados pelos sinais “send port A” e “send port B”, também originados do “Arbiter”. O funcionamento de “TX Source” está representado na Figura 19.

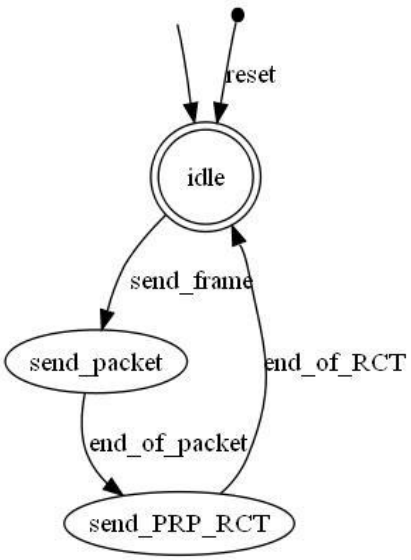


Figura 19: Máquina de estados de TX Source

O sinal “frame size” indica para o “TX Source” até qual endereço da memória existem dados do quadro a ser enviado, começando pelo endereço zero. O RCT é enviado ao final do quadro

Os sinais de saída são: sinais da interface Avalon-ST Source (valid, data, empty, error) para os MACs A e B, sinais de acesso à memória e o sinal “frame sent” que sinaliza que a máquina de estados está em “idle” e, portanto, o quadro foi enviado.

5.2.3: Supervision Frame

O componente “Supervision Frame”, cujo funcionamento está representado na máquina de estados da Figura 20, envia quadros de supervisão com frequência determinado por *lifeCheckInterval* (5s por default).

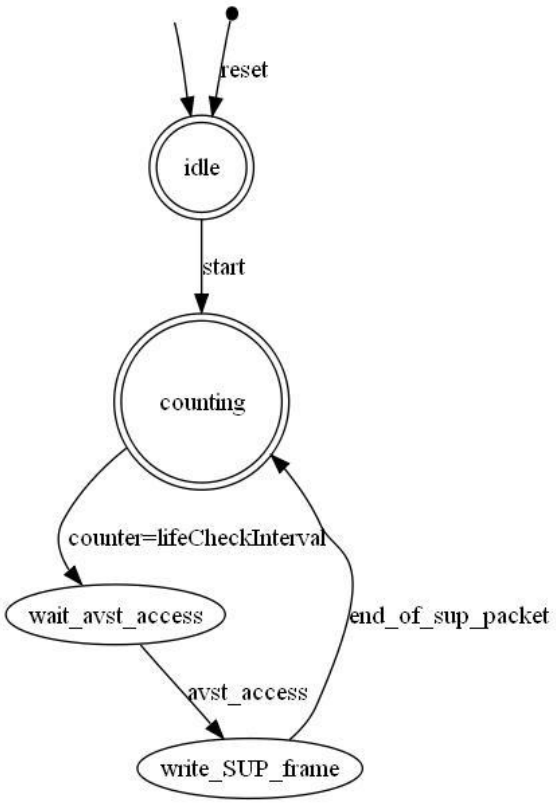


Figura 20: Máquina de estados de Supervision Frame

A contagem do tempo é iniciada quando o sinal de inicialização do sistema “start” é ativado. Antes de enviar o quadro, o componente precisa requisitar acesso à interface Avalon-ST Source, compartilhada com “TX Source”. O sinal de entrada “avst access”, oriundo do “Avalon-ST Arbiter”, sinaliza que o componente pode enviar o quadro. Após o envio, o componente volta a contar o tempo e indica para o “Avalon-ST Arbiter” através do sinal “ready” que a interface Avalon está liberada.

Os sinais de entrada são: sinais da interface Avalon-ST Source para os MACs A e B, o sinal “start” e o sinal “avst access”. Os sinais de saída são: sinais da interface Avalon-ST Source, o sinal “ready” e o sinal “avst request” (requisição de acesso à interface Avalon-ST).

5.2.4: Avalon-ST Arbiter

Esse componente é responsável por coordenar o acesso à interface Avalon-ST para envio de quadros aos MACs. Ele recebe as requisições de acesso do “Arbiter” e “Supervision Frame”. Quando no estado “grant_prp_sup_access”, ao acesso é liberado para “Arbiter”, enquanto no estado “grant_prp_sup_access”, o acesso é liberado para “Supervision Frame”. O funcionamento do circuito é representado na Figura 21.

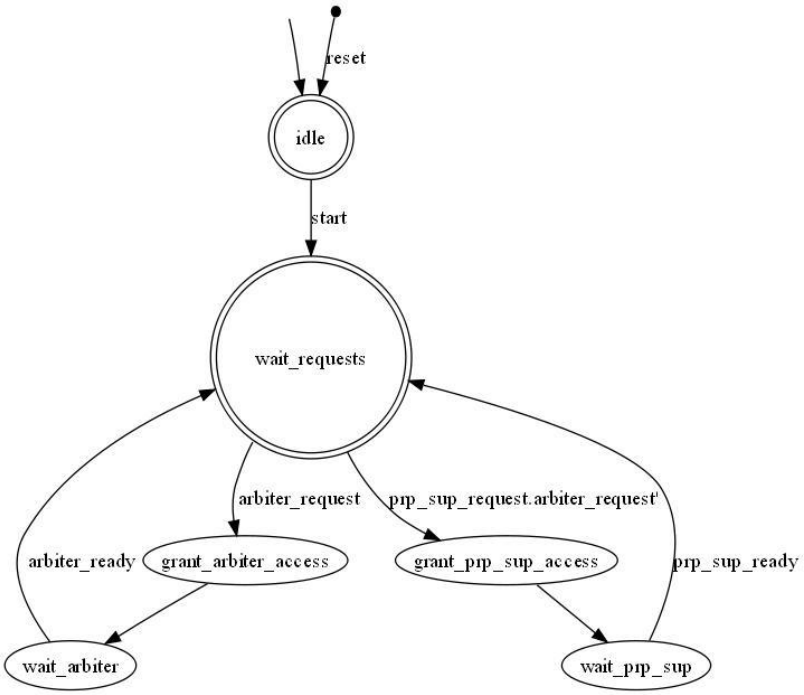


Figura 21: Máquina de estados de Avalon-ST Arbiter

O circuito prioriza o acesso de “Arbiter” quando os dois componentes solicitam acesso à interface Avalon.

O “Avalon-ST arbiter” também coordena o funcionamento do multiplexador “Avalon-ST Source Control”, fazendo com que este selecione os sinais provenientes de “Arbiter” quando nos estados “grant_arbiter_access” e “wait_arbiter”, e selecione

os sinais de “Supervision Frame” nos estados “grant_prp_sup_access” e “wait_prp_sup”.

5.2.5: Arbiter

O quadro a ser enviado é guardado em uma memória RAM acessado tanto por “TX Sink” quanto por “TX Source”, portanto é preciso coordenação nesse acesso. O módulo “Arbiter” coordena o acesso à memória, liberando a transmissão de um quadro para os MACs somente após ser recebido do “SGDMA TX”. O funcionamento está demonstrado na Figura 22.

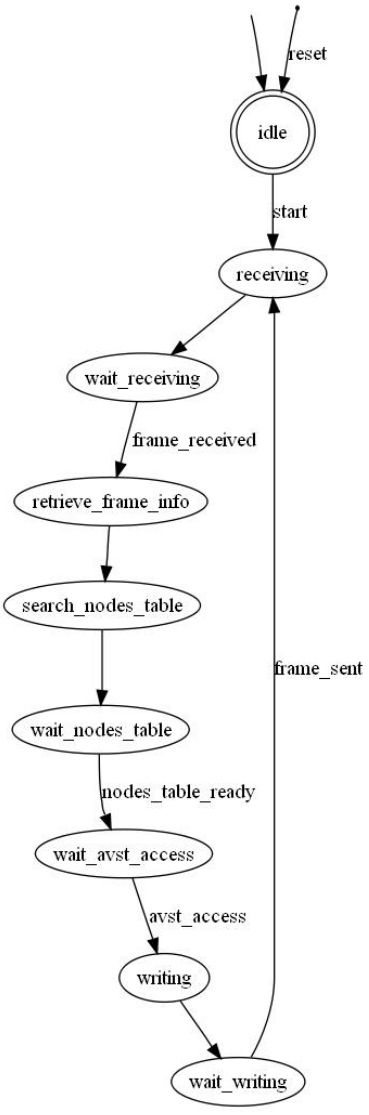


Figura 22: Máquina de estados de Arbiter

No estado “receiving”, esse módulo sinaliza para “TX Sink” a liberação para receber um quadro, e aguarda a finalização da recepção no estado “wait_reading”.

No estado “retrieve frame info”, o “Arbiter” obtém o endereço destino do quadro para poder consultar a tabela de nós no estado “search_nodes_table” e verificar em qual LAN o quadro deve ser enviado. Após o encerramento da procura, o “Arbiter” faz a requisição do acesso ao barramento “Avalon-ST”, e após receber a confirmação “avst_access” do “Arbiter Avalon-ST”, libera a transmissão do quadro para os MACs.

O “Arbiter” informa a “TX Source” o tamanho do quadro a ser enviado e em quais MACs eles devem ser enviados.

5.3: RX Redundancy Master

A Figura 23 mostra a estrutura do “RX Redundancy Master”. O “RX Redundancy Master” é formado por 13 componentes: “RAM port” A e B, “Memory Access Control” A e B, “RX Sink” A e B, “Arbiter”, “RX Source”, “Arbiter Nodes Table”, “Arbiter Dup Check”, “Nodes Table Access Control”, “Duplicate Check Access Control” e “Duplicate Check”.

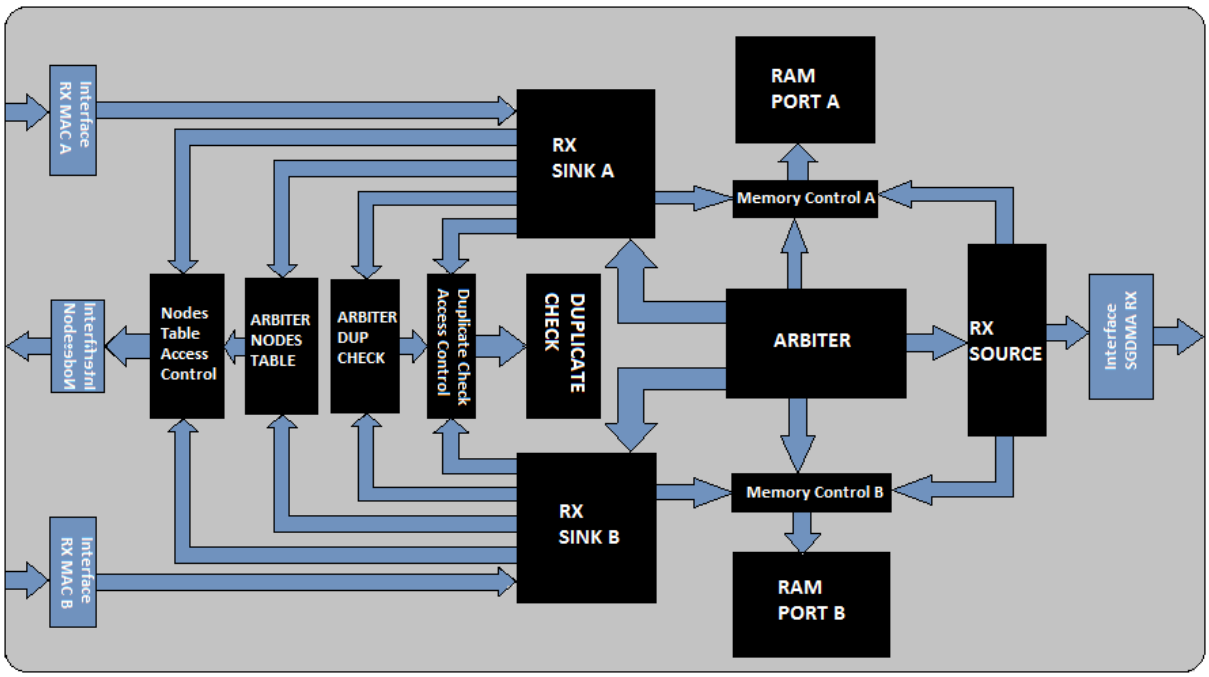


Figura 23: Estrutura do RX Redundancy Master

O componente “RX Sink A” é responsável pela recepção dos quadros enviados pelo MAC A, através da interface Avalon-ST. O “RX Sink A” consulta o componente “Duplicate Check” para verificar se o quadro recebido é uma

duplicação. Caso o quadro seja válido, ele é salvo na “RAM port A”, do contrário é descartado. O “RX Sink A” solicita também atualização da tabela de nós.

O componente “RX Sink B” funciona igual ao “RX Sink A”, porém recebe quadros do MAC B.

O componente “Nodes Table Access Control” é um multiplexador que controla se o “RX Sink A” ou o “RX Sink B” irá acessar a tabela de nós baseado na coordenação de “Arbiter Nodes Table”.

O componente “Duplicate Check Access Control” é um multiplexador que controla se o “RX Sink A” ou o “RX Sink B” irá acessar o “Duplicate Check” baseado na coordenação de “Arbiter Dup Check”.

O componente “Memory Control A” é um multiplexador que controla se o “RX Sink A” ou o “RX Source” irá acessar o “RAM port A” baseado na coordenação de “Arbiter”. O componente “Memory Control B” funciona igualmente a “Memory Control A”, porém para coordenar acesso de “RX Sink B” ao invés de “RX Sink A”.

O componente “Duplicate Check” é responsável por implementar o algoritmo de detecção de quadros duplicados.

O componente “TX Source” é responsável pela transmissão dos quadros válidos para o “SGDMA RX” através de uma interface Avalon-ST. Os quadros válidos são guardados nas memórias RAM.

O componente “Arbiter” é responsável por coordenar quem acessa as memórias RAM. Ele libera os blocos “RX Sink A” e “RX Sink B” para receber um quadro, e após a recepção do mesmo, libera o bloco “RX Source” para transmissão (quando o quadro é válido). Enquanto “RX Source” envia um quadro, o bloco “RX Sink” que recebeu esse quadro fica esperando o fim da transmissão para poder acessar a memória e guardar novos quadros.

5.3.1: RX Sink

O funcionamento dos componentes “RX Sink A” e “RX Sink B” está mostrado na Figura 24.

Figura 24: Máquina de estados de RX Sink

Após receber a liberação “read frame” do “Arbiter”, o componente começa a recepção de um quadro do MAC. Após a recepção do quadro, é feita uma checagem se o quadro possui um RCT, e se ele é bem formado.

No estado “wait nodes table”, o “RX Sink” solicita a “Arbiter Nodes Table” o acesso à tabela de nós para solicitar a atualização da mesma. No estado “update nodes table”, o endereço MAC de origem do quadro, a LAN em que foi recebida, a confirmação ou não de recebimento de um quadro de supervisão e a confirmação ou não de erro de LAN são repassados para tabela de nós para que ela possa ser atualizada. O componente não espera a finalização da atualização, já que não é necessária nenhuma resposta da tabela de nós para o processo de recepção continuar e o processo pode ser demorado. Caso o “RX Sink” entre no estado “update nodes table” e a tabela de nós está ocupada, o componente espera ela terminar seu processamento antes de oferecer novos dados.

Caso quadro recebido seja um quadro de supervisão PRP, ele é descartado, e o “RX Sink” passa a receber outro quadro. Caso o quadro não seja de supervisão, mas o RCT está mal formado, a checagem de duplicados não é realizada e o “RX Sink” sinaliza no estado “idle” que um quadro foi recebido.

Caso o quadro não seja de supervisão e o RCT é válido, o “RX Sink” solicita o acesso à checagem de duplicados para “Arbiter Dup Check”, e após receber a liberação (“dup_check_access”), a checagem é realizada. Caso o quadro já foi recebido anteriormente, ele é descartado e “RX Sink” passa a receber outro, do contrário, “RX Sink” finaliza a recepção.

5.3.2: RX Source

O módulo “RX Source”, cujo funcionamento está representado pela máquina de estados da Figura 25, é responsável por enviar os quadros recebidos dos MACs para o “SGDMA RX”.

Após receber a liberação “send frame” do “Arbiter”, o módulo começa a transmissão do pacote para o “SGDMA RX”.

Após a transmissão do LSDU, o “RX Source” envia também o RCT caso esteja configurado em “transparente reception”, do contrário o RCT é retirado.

O “RX Source” envia quadros recebidos por “RX Sink A” e “RX Sink B”, porém apenas um de cada vez. A escolha de qual quadro recebido repassar para as camadas superiores é definida pelo módulo “Arbiter”, e repassado para “RX Source”.

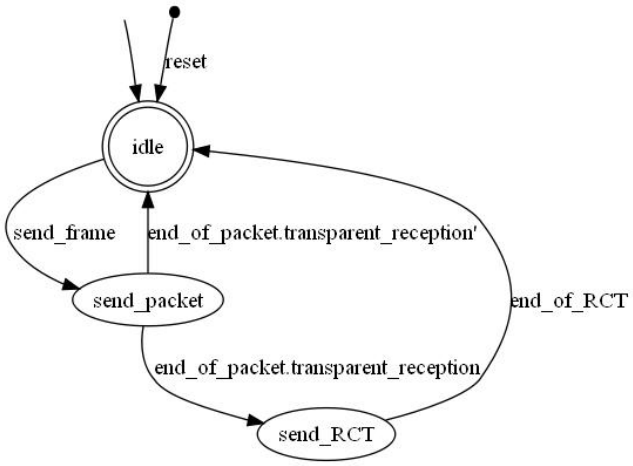


Figura 25: Máquina de estados de RX Source

5.3.3: Arbiter Dup Check e Arbiter Nodes Table

Na Figura 26 está representado o funcionamento dos componentes “Arbiter Dup Check” (controla acesso ao módulo “Duplicate Check”) e “Arbiter Nodes Table” (controla acesso ao módulo “Nodes Table”).

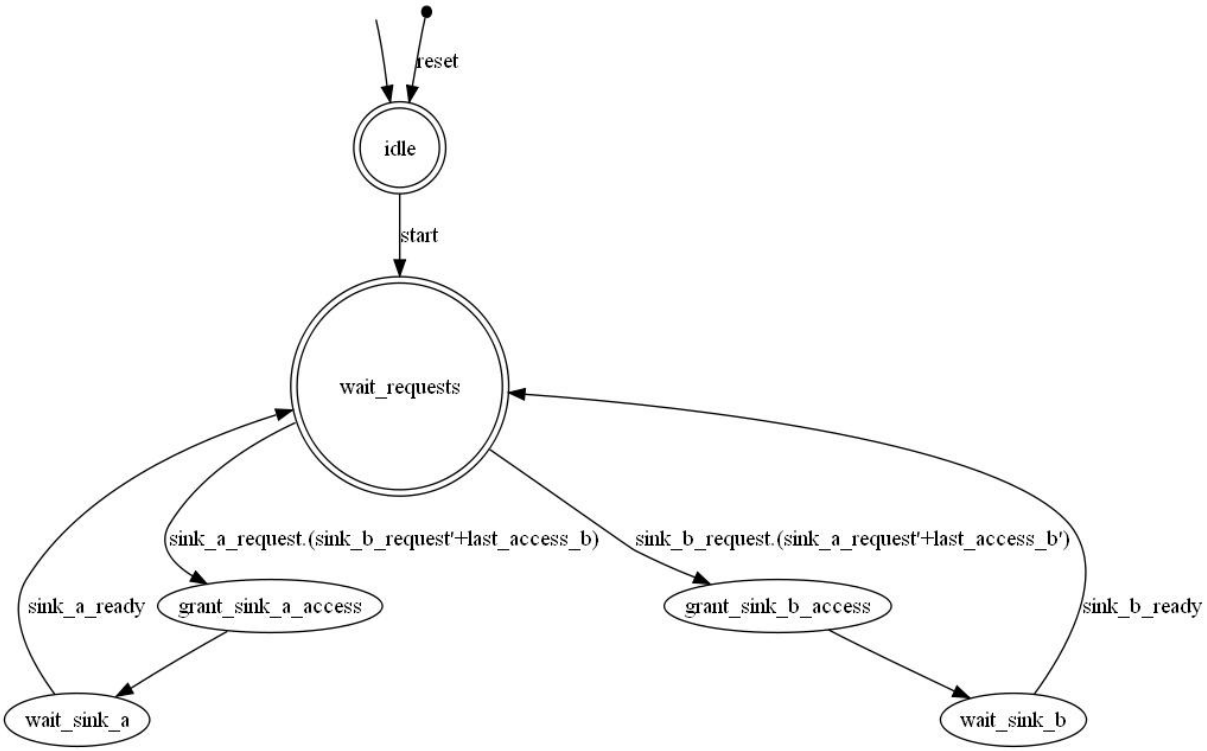


Figura 26: Máquina de estados de Arbiter Dup Check e Arbiter Nodes Table

Após a inicialização do sistema (sinal “start”), o módulo passa a esperar requisições de acesso de “RX Sink A” e “RX Sink B”. O acesso é liberado para “RX Sink A” no estado “grant_sink_a_access” e para “RX Sink B” no estado “grant_sink_b_access”.

‘ Quando os dois componentes “RX Sink” solicitam acesso ao mesmo tempo, o registrador “last_access_b” é consultado para saber se o “RX Sink A” ou “B” foi o último a acessar. Se o “RX Sink A” foi o último acessar (last_access_b=0), a liberação de acesso é concedida a “RX Sink B”, do contrário a liberação é concedida para “RX Sink B”.

O módulo “Arbiter Dup Check” controla o multiplexador “Duplicate Check Access Control”, sinalizando para qual componente (“RX Sink A” ou RX Sink B”) o multiplexador deve selecionar os sinais.

O módulo “Arbiter Nodes Table” controla o multiplexador “Nodes Table Access Control”, sinalizando de qual componente o multiplexador deve selecionar os sinais e repassar a tabela de nós.

5.3.4: Arbiter

O módulo “Arbiter”, cujo funcionamento está representado pela máquina de estados da Figura 27, coordena a recepção de quadros dos MACs e envio para a “SGDMA RX”.

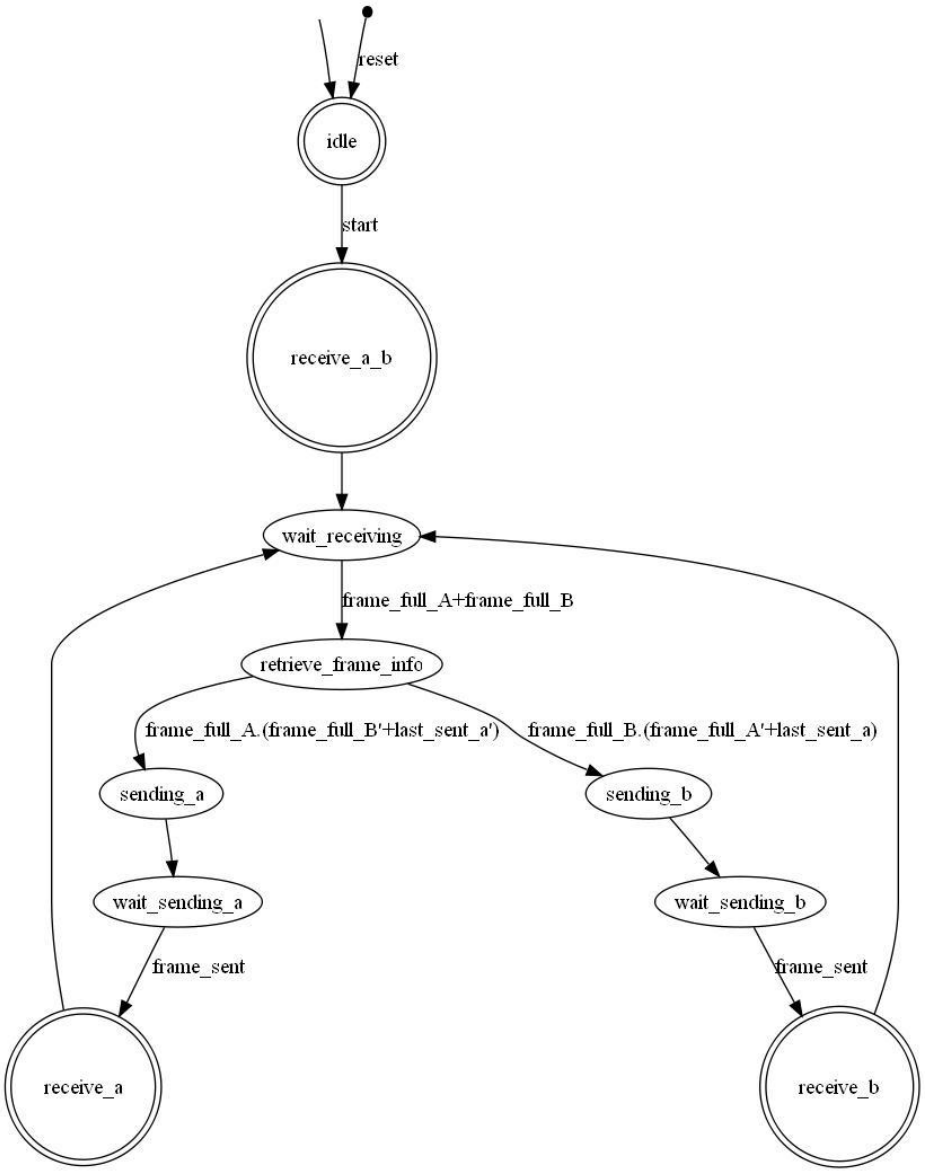


Figura 27: Máquina de estados de Arbiter

Após a inicialização do sistema, o “Arbiter” libera os componentes “RX Sink A” e “RX Sink B” para recepção de quadros (estado “receive_a_b”).

Quando um quadro é recebido por “RX Sink A” (sinal “frame_full_A”), o “Arbiter” obtém informação de tamanho do quadro no estado “retrieve_frame_info” para poder repassar para “RX Source” (essa informação indica até qual endereço da memória existe dados válidos do quadro). O “Arbiter” também informa para “RX Source” em qual memória se encontra o quadro a ser enviado (“RX Sink A”). Em seguida, o envio é liberado e após a finalização do mesmo, o “Arbiter” libera a recepção de outro quadro para “RX Sink A”. Quando um quadro é recebido por “RX Sink B”, o procedimento é parecido.

Quando a recepção de quadros por “RX Sink A” e “RX Sink B” acontece ao mesmo tempo, o registrador “last_sent_a” é consultado. Caso o último quadro repassado para a “SGDMA RX” foi recebido por “RX Sink A” (“last_sent_a=1”), o “Arbiter” libera “RX Source” para encaminha o quadro recebido por “RX Sink B”, do contrário, encaminha-se o quadro recebido por “RX Sink A”.

5.3.5: Duplicate Check

A Figura 28 mostra a estrutura do “Duplicate Check”. Ele é formado por 6 componentes: “Hash Table”, “Memory Controller”, “Duplicate Check”, “Arbiter”, “Aging” e “Hash Value Generator”.

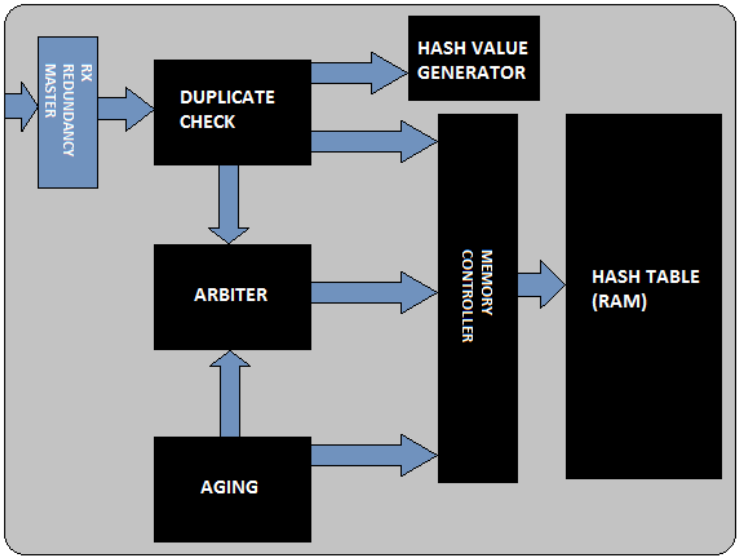


Figura 28: Estrutura do Duplicate Check

O componente “Duplicate Check” é responsável por realizar o algoritmo de identificação de quadros duplicados.

O componente “Aging” é responsável pela remoção do registro de um quadro no algoritmo de descarte de duplicados, devido à necessidade causada pelo reinício de contagem do número de sequência, conforme a necessidade descrita na seção 3.3.4.2.

O componente “Memory Controller” é um multiplexador que controla o acesso (“Duplicate Check” ou “Aging”) a tabela utilizada no algoritmo de descarte de duplicados (memória RAM), dependendo da coordenação do componente “Arbiter”.

O componente “Arbiter” libera o acesso à memória ao receber solicitações dos componentes interessados, dando prioridade ao componente “Duplicate Check”.

O componente “Hash Value Generator” é responsável por gerar um valor hash baseado no endereço MAC e no número de sequência, utilizado para acesso a tabela. O algoritmo implementado por esse componente e o formato da tabela de nós serão explicados a seguir.

5.3.5.1: Algoritmo de descarte de duplicados

Como descrito na seção 3.3 desse documento, o protocolo PRP se baseia na duplicação dos quadros a serem enviados, e posterior transmissão por duas LANs independentes. O nó receptor irá, em funcionamento normal, receber dois quadros contendo a mesma informação, encaminhará o primeiro para as camadas superiores e descartará o segundo.

O algoritmo utilizado para o descarte de quadros duplicados se baseia no trabalho realizado em [12]. Nesse trabalho, o autor desenvolveu um dispositivo em FPGA que implementasse o protocolo HSR, também descrito na norma IEC 62439, cujo funcionamento é semelhante ao PRP em muitos aspectos. Nesse trabalho, o autor realizou testes de vários algoritmos diferentes para descarte de duplicados e selecionou um que deu resultados bastante satisfatórios sem ter complexidade muito alta. O algoritmo utilizado foi um misto de tabela hash com buffer circular.

5.3.5.1.1: Princípio de funcionamento

Cada quadro PRP é unicamente identificado pelo seu endereço de origem e número de sequência. Portanto o endereço de origem e o número de sequência de

um quadro podem ser guardados como entrada em uma tabela, para posterior verificação de recebimento do mesmo.

Quando um quadro é recebido com sucesso (ou seja, nenhum erro ocorreu) em uma das portas, verifica-se se o conjunto *{endereço de origem, número de sequência}* já existe na tabela. Caso o conjunto já esteja presente na tabela, o quadro é descartado como um duplicado, do contrário guarda-se o conjunto na tabela e encaminha o quadro para as camadas superiores.

Para aumentar a eficiência da procura, uma tabela hash é utilizada.

5.3.5.1.2: Tabela hash com buffer circular

Na tabela hash com buffer circular, o endereço MAC de origem e o número de sequência são utilizados como chave. A tabela é divida em várias regiões, sendo que cada região é um buffer circular tamanho fixo *buffer_size,* conforme mostrado na Figura 29 (nesse caso, *buffer_size* igual a quatro). O conceito é semelhante a uma tabela hash encadeada, substituindo a lista encadeada por um buffer circular, não havendo mais necessidade de um processo de procura de espaço vazio na memória. O processo de leitura/escrita (Figura 30) no buffer circular utilizam os ponteiros de leitura/escrita. O princípio de operação é o seguinte:

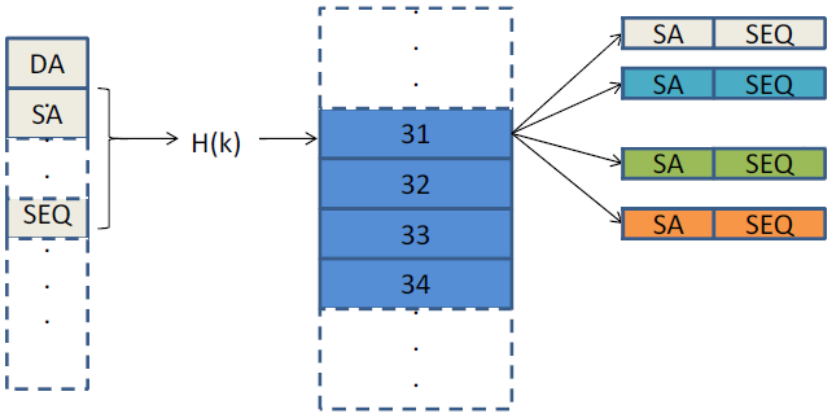


Figura 29: Tabela hash com buffer circular

- Para verificar a existência de um quadro recebido na tabela: Quando um quadro é recebido, o endereço de origem e o número de sequência formam uma chave utilizada pra calcular o valor hash. O valor calculado é mapeado para certa área da tabela (um buffer circular). A posição do ponteiro de leitura do buffer é a posição após a

ultima procura realizada nesse buffer. Se o conjunto *{endereço de origem, numero de sequencia}* não for encontrado na atual posição, o ponteiro de leitura move uma posição. Se a entrada for encontrada, o ponteiro de leitura se mantém na posição seguinte a posição em que a entrada foi encontrada. Quando o ponteiro se mover *buffer_size* vezes e a entrada não for encontrada, a procura termina.

- Para escrever uma nova entrada na tabela: Escrever a entrada na posição do ponteiro de escrita e mover o ponteiro uma posição. Se o ponteiro de escrita atingir o final do buffer, ele retorna ao início.

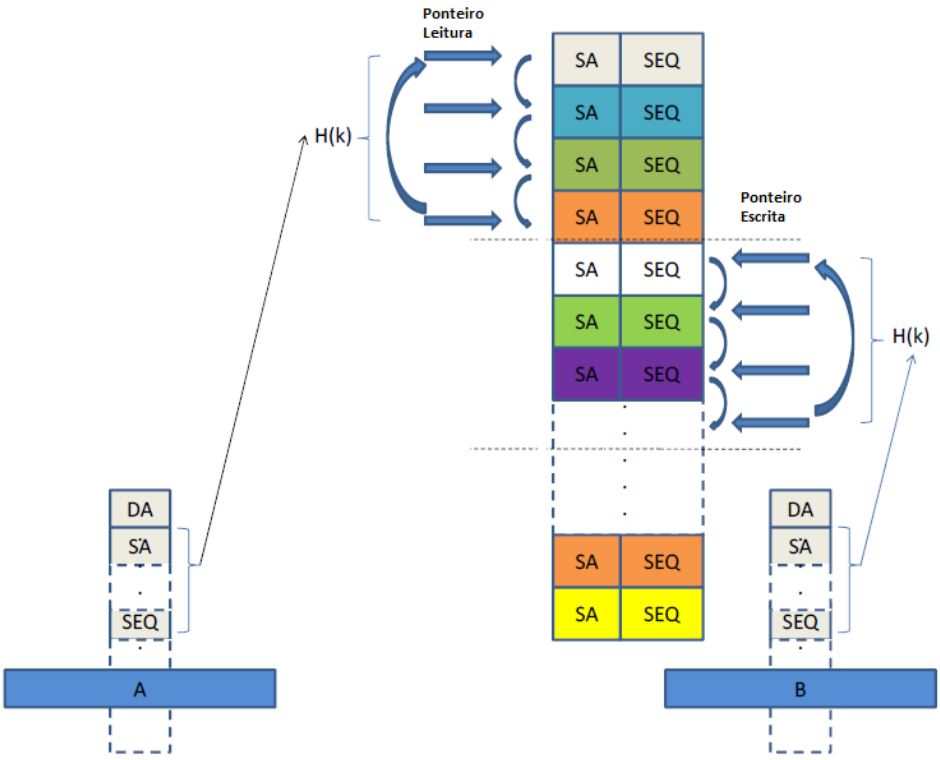


Figura 30: Operações de leitura e escrita

Esse algoritmo tem algumas vantagens em relação a outros algoritmos de tabelas hash tradicionais. O número de passos de procura é limitado a *buffer_size*, independente de a tabela estar cheia ou não, o que não acontece em tabelas hash encadeadas ou com endereçamento aberto, onde o desempenho do procedimento tende a se tornar mais lento à medida que a tabela vai se enchendo. O processo de escrita é bem simples e rápido, utilizando o conceito de buffer circular, enquanto em outros processos é mais complicado e demorado, envolvendo em muitos casos atrasos relacionados à procura de espaços vazios na memória.

Uma desvantagem desse método é que ele não procura uma entrada vazia no buffer circular, mas simplesmente sobrescreve a entrada mais antiga. A probabilidade de uma entrada não utilizada ser sobrescrita é maior que em outros métodos, o que aumenta o risco de um quadro duplicado ser considerado como válido.

Em [12], mostrou-se que esse algoritmo teve um desempenho semelhante a outros métodos de tabela hash, apresentando uma complexidade menor.

A função hash utilizada nesse trabalho foi $h(x) = x \bmod m$, onde:

$$x = hashPJW(endereço\ de\ origem\ (SA), número\ de\ sequência\ (SEQ))$$

$$m = tamanho\ da\ tabela.$$

5.3.5.1.3: A randomização de chaves não uniformes

Uma função hash como $h(x) = x \bmod m$, tem um bom desempenho, gerando sequências uniformemente distribuídas, somente quando a chave *x* é uniformemente distribuída. Quando a chave diverge de uma distribuição uniforme, a chance de colisão na tabela hash aumenta.

Infelizmente o conjunto *{endereço de origem, número de sequência}* não é uniformemente distribuído, devido aos seguintes motivos:

- Os 3 primeiros bytes do endereço MAC é identificação do fabricante. Em uma subestação, a quantidade de fabricantes diferentes de dispositivos não deve passar de 20, logo a variação é bem pequena.
- Apesar do número de sequência variar frequentemente (de 0 a 65536). Ele varia somente ao final da chave.

Uma solução normalmente utilizada para esse problema é aplicar uma função de randomização da chave, antes de utilizá-la na função hash. Um algoritmo popular, chamado *hashPJW*, recebe um string de entrada, e gera como saída um inteiro na faixa [0, 2^{32} – 1]. Essa transformação realizada por *hashPJW* normalmente produz números que aparecem uniformemente em um certo intervalo, mesmo quando as strings de entrada são bastante similares. O algoritmo da função *hashPJW,* a qual foi utilizada nesse trabalho, está na Figura 31.

```
int hashpjw(const void *key) {
   const char *ptr;
   int val;
   /*Hash the key by performing a
      number of bit operations on it. */
   val = 0;
   ptr = key;
   while (*ptr != '\0') {
      int tmp;
      val = (val << 4) + (*ptr);
      if (tmp = (val & 0xf0000000)) {
         val = val ^ (tmp >> 24);
         val = val ^ tmp;
       }
       ptr++;
    }
}
```

Figura 31: Algoritmo da função de randomização HashPJW

5.3.5.1.4: O reinício de contagem do número de sequência

Conforme mostrado na seção 3.3, o número de sequência possui 16 bits, portanto varia de 0 a 65536. Cada vez que um nó envia um quadro, ele incrementa o contador, e após 65536 quadros enviados, teremos um quadro com identificação igual a um quadro já enviado anteriormente. No pior dos casos, em uma rede Ethernet de 100Mbit/s, isso pode ocorrer a cada 440 ms.

Para poder resolver essa situação, cada entrada na tabela hash possui um contador de tempo que é atualizado com certa frequência por um processo paralelo de atualização de tempos. Quando esse contador atinge 440 ms, essa entrada é removida da tabela.

5.3.5.2: Duplicate check

O componente “duplicate check”, cujo funcionamento é mostrado na máquina de estados da Figura 32, faz a checagem se um determinado quadro recebido é uma duplicação. Esse componente recebe o endereço MAC de origem e o número de sequência do quadro como entradas (esses sinais são provenientes de “RX Sink A” ou “RX Sink B”, dependendo da seleção o multiplexador “Duplicate Check Access Control), e após o sinal de “start” inicia a verificação.

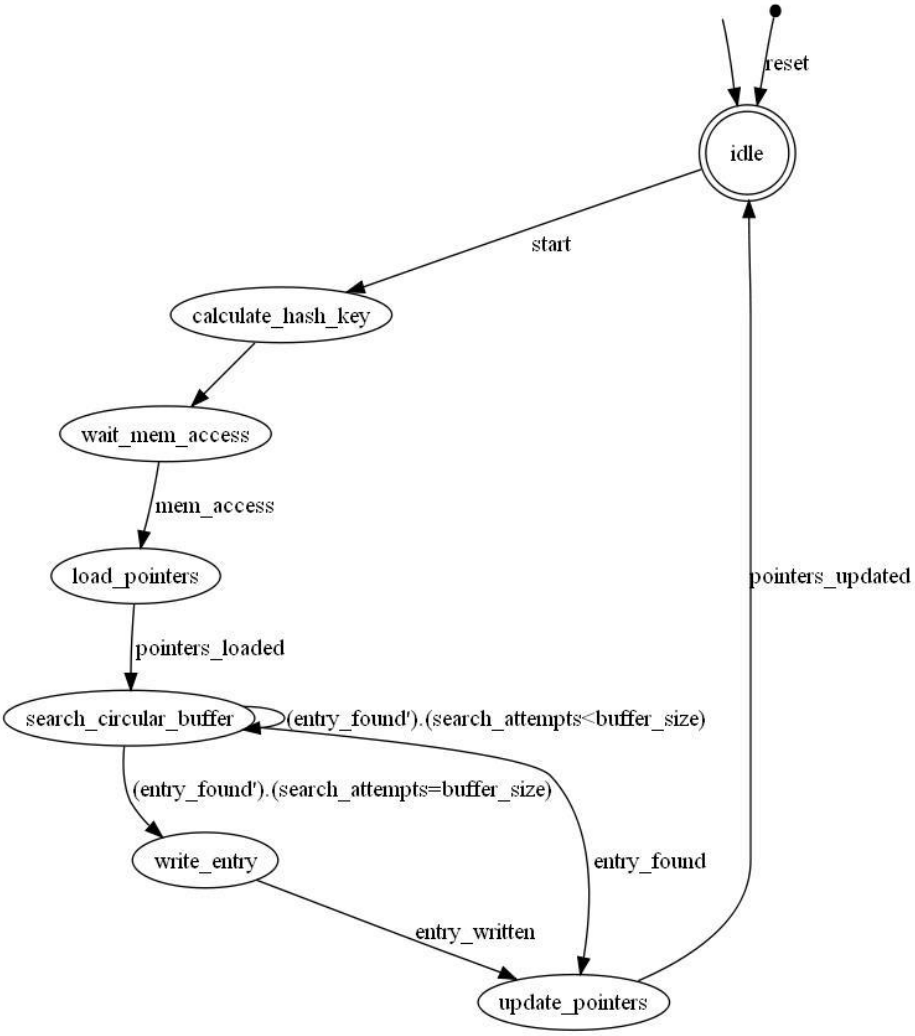


Figura 32: Máquina de estados de Duplicate Check

O primeiro passo da verificação é o cálculo do valor hash como descrito na no algoritmo de detecção de duplicados. O valor hash indica o endereço do buffer circular onde será feita as tentativas de achar a entrada.

Logo após o cálculo do valor hash, o componente faz a requisição do acesso à memória.

Em seguida, os ponteiros de leitura e escrita do buffer circular são carregados e utilizados para fazer a verificação. O primeiro endereço a ser procurado é o apontado pelo ponteiro de leitura. Caso a entrada seja encontrada, o ponteiro de leitura é incrementado de um e guardado na memória assim como o ponteiro de escrita, o componente retorna ao estado “idle” e sinaliza a finalização da verificação e que o quadro já foi recebido.

Caso a entrada não seja encontrada, o ponteiro de leitura é incrementado de um e acontece mais uma tentativa. Após varrer o buffer circular inteiro e a entrada não for encontrada, o endereço de origem e número de sequência é salvo na memória no endereço apontado pelo ponteiro de escrita, e ele é incrementado de um. Em seguida os ponteiros são guardados na memória, o componente retorna ao estado “idle” e sinaliza a finalização da verificação e que o quadro não foi recebido ainda.

5.3.5.3: Aging

Após a inicialização do sistema (sinal “start”), o “Aging” começa a contagem do número de pulsos de ciclo. Quando o número de pulsos atinge o valor *refreshTime,* inicia-se a atualização de tempo das entradas.

O primeiro passo da atualização é conseguir o acesso na memória, representado pelo estado “wait_mem_access” da máquina de estados do funcionamento de “Aging” da Figura 33.

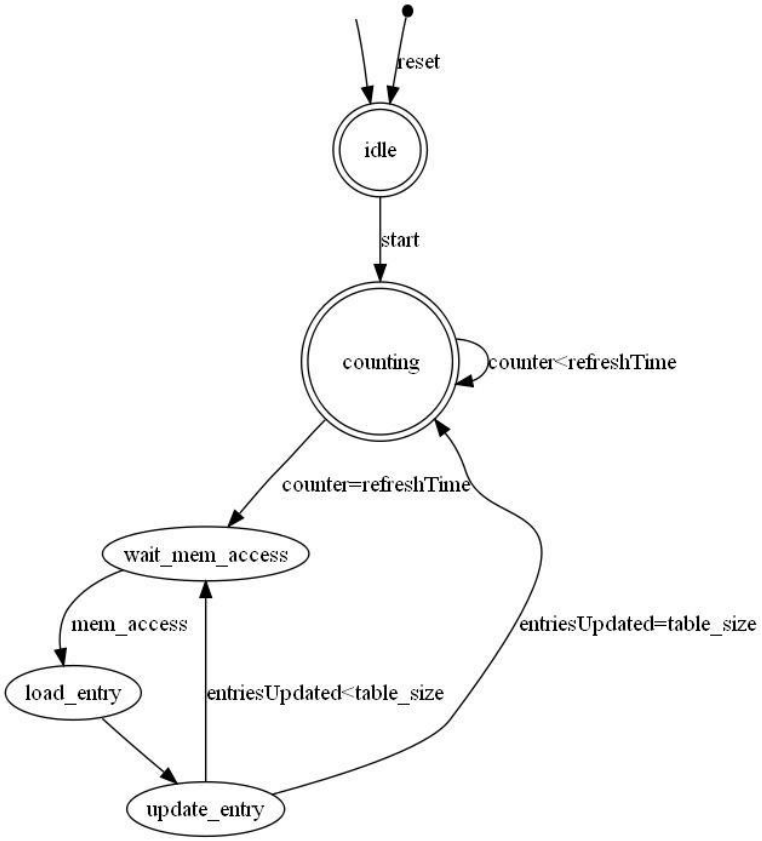


Figura 33: Máquina de estados de Aging

A atualização inicia pelo endereço 0 e vai até o valor *table_size* (tamanho da tabela hash).

Cada entrada possui, além do endereço de origem e o número de sequência, um vetor de 6 bits contando a quantidade de atualizações de tempo que essa entrada recebeu. A cada atualização esse vetor é incrementado por 1.

Quando o vetor de tempo atinge o valor *EntryForgetTime*, a entrada é deletada da memória. A atualização e remoção de entradas estão representadas pelo estado “update_entry”.

Cada vez que “Aging” atualiza uma entrada, ele solicita acesso a memória novamente para poder liberá-la para “Duplicate Check” caso seja necessário.

5.3.5.4: Arbiter

O funcionamento de “Arbiter”, representado pela máquina de estados da Figura 34, é parecido com o funcionamento de “Avalon-ST Arbiter”, porém o que está sendo coordenado nesse componente é o acesso à tabela hash, a qual é disputada pelos componentes “Aging” e “Duplicate Check”.

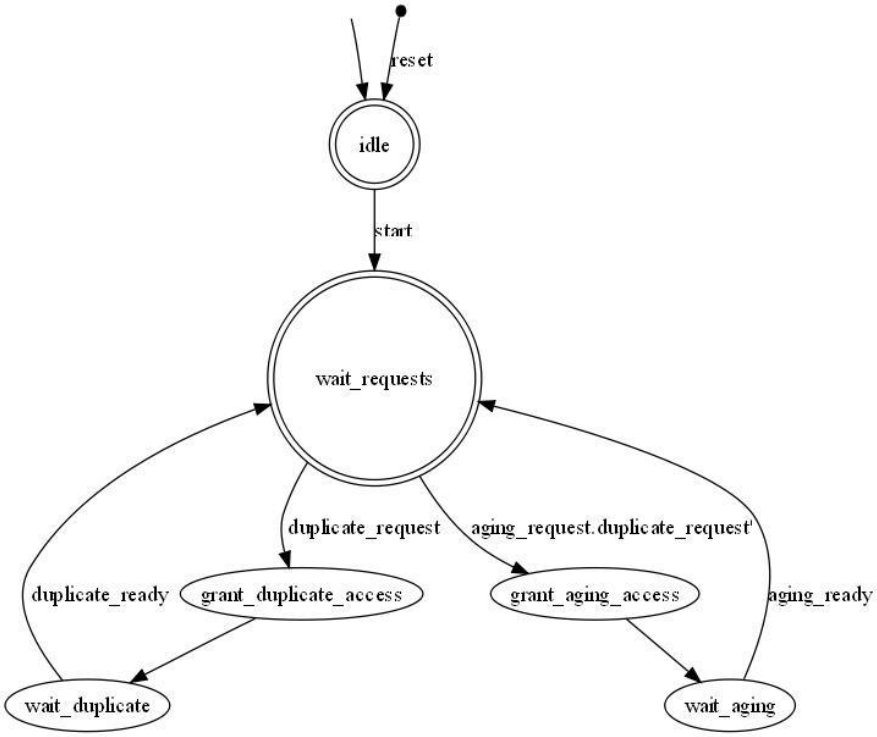


Figura 34: Máquina de estados de Arbiter

O “Arbiter” controla também o multiplexador “Memory Controller”. As entradas desse multiplexador são os sinais de acesso à memória de “Aging” e “Duplicate Check”, e o controle de “Arbiter”.

“Duplicate Check” possui prioridade de acesso em relação a “Aging”.

5.4: Nodes Table

A Figura 35 mostra a estrutura do “PRP Nodes Table”. Ele é formado por 8 componentes: “Hash Table”, “Memory Controller”, “Update Nodes Table”, “Arbiter”, “Aging”, “Search Process” e dois “Hash Value Generator”.

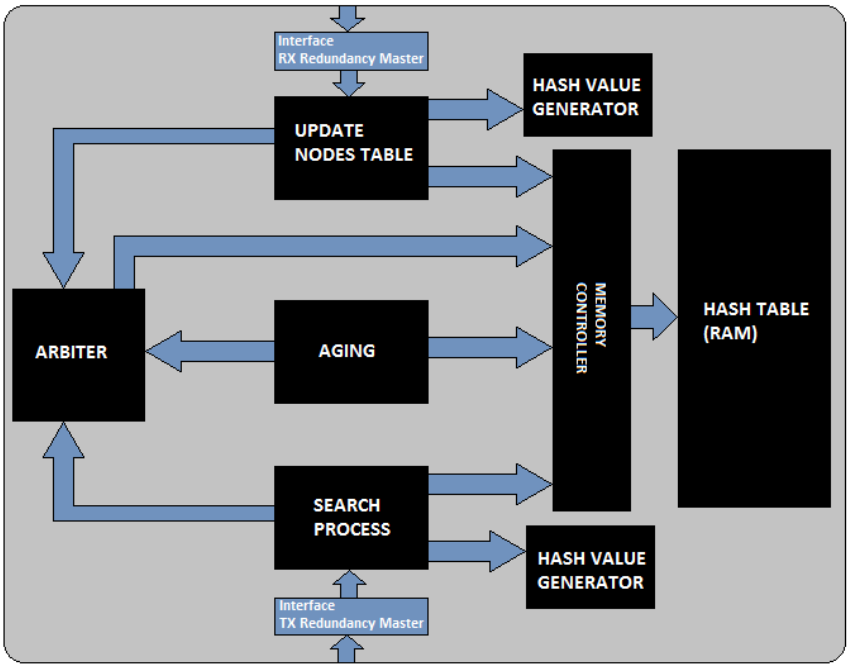


Figura 35: Estrutura do PRP Nodes Table

O componente “Update Nodes Table” é responsável pela atualização da tabela de nós, recebendo solicitações de acesso e informações necessárias para atualização (conforme tabela descrita na seção 3.3) do componente “RX Redundancy Master”.

O componente “Aging” é responsável por verificar quanto tempo cada entrada da tabela está sem ser atualizada, e excluir caso necessário. Conforme descrito na seção 3.3, uma entrada na tabela de nós deve ser excluída após certo tempo sem ser atualizada (entende-se por atualização, a recepção de um quadro com o endereço de origem registrado).

O componente “Search Process” realiza a verificação se um determinado nó está registrado na tabela, e se esse nó é um SAN (conectado a LAN A ou B) ou um DANP, conforme solicitação do componente “TX Redundancy Master”.

O componente “Memory Controller” é um multiplexador que controla o acesso a tabela de nós (memória RAM), dependendo da coordenação do bloco “Arbiter”.

O componente “Arbiter” recebe solicitações de acesso a tabela de nós dos componentes “Search Process”, “Update Nodes Table” e “Aging”, e coordena quem tem a liberação do acesso.

O componente “Hash Value Generator” é responsável por gerar um valor hash baseado no endereço MAC, utilizado para acesso a tabela. O algoritmo implementado por esse componente e o formato da tabela de nós serão explicados a seguir.

5.4.1: O algoritmo da tabela de nós

Conforme descrito na seção 3.3, o PRP exige que seja criado uma tabela de nós, onde são guardadas informações de todos os nós que estão se comunicando na rede. Essa tabela é construída baseada nos quadros recebidos. A tabela de nós possui alguns requisitos diferentes da tabela utilizada no algoritmo de descarte de duplicados, são eles:

- Uma entrada não pode ser sobrescrita.
- Visto que os nós estão se comunicando constantemente, a operação de inserção de novas entradas na tabela ocorre somente no inicio da operação da rede, após isso as operações predominantes são de atualização de entradas existentes.
- O processo de recepção não fica travado esperando uma resposta do processo de atualização da tabela de nós, ele apenas inicia o processo e continua suas tarefas, portanto os requisitos temporais são um pouco mais relaxados.
- Somente o endereço de origem é utilizado como chave.

Devido às observações acima, optou-se por utilizar uma tabela hash encadeada para a tabela de nós. Embora a escrita de novas entradas na tabela hash encadeada seja um pouco mais lento devido ao processo de alocação de

memória, esse tipo de processo, como foi explicado acima, ocorre com baixíssima frequência. Por outro lado, a tabela hash encadeada oferece degradação suave do desempenho da procura/atualização, à medida que a tabela for ficando cheia, além de nenhuma entrada ser sobrescrita. O tamanho da tabela é fixo, e uma vez cheio, nenhuma entrada nova é adicionada até ser liberado espaço.

O princípio de funcionamento da tabela de nós é o seguinte:

- Para atualização de uma entrada na tabela: Quando um quadro é recebido, seu endereço de origem é utilizado como entrada para uma função hash. O valor calculado mapeia para certa posição na tabela de registros, onde cada registro aponta para o início de uma lista encadeada. Se o registro não for vazio, inicia-se uma procura pelo nó na lista encadeada. Caso a entrada seja encontrada, ela é atualizada, senão inicia-se o processo de escrita.
- Para escrita de uma entrada na tabela: Se o valor hash mapear para uma posição vazia na tabela de registros, aloca-se um espaço na memória para registrar o nó (caso ainda exista espaço, senão cancela operação) e uma lista encadeada é criada com esse nó sendo o primeiro elemento dela. O endereço onde esse nó é criado é guardado na tabela de registros. Se o valor hash mapear para uma entrada existente, a lista encadeada é aumentada e atualizada.

A função hash e de randomização da chave utilizadas na tabela de nós é a mesma do algoritmo de descarte de duplicados.

Paralelamente ao processo de atualização/escrita na tabela de nós, também existe, como no processo de descarte de duplicados, um processo de atualização de tempo das entradas da tabela hash. Quando uma entrada fica certo tempo (na faixa de dezenas de segundos) sem ser atualizada, ela é eliminada da tabela. O processo de eliminação de uma entrada em uma lista encadeada é um pouco demorado, no entanto, isso não ocorre com frequência.

5.4.2: Update

O componente “update”, cujo funcionamento está representado na máquina de estados da Figura 36, faz a atualização da tabela de nós. Esse componente

recebe o endereço MAC de origem do quadro, a identificação da LAN onde o quadro foi recebido e o modo de operação do nó (em caso de quadro de supervisão) como entradas (esses sinais são provenientes de “RX Sink A” ou “RX Sink B”, dependendo da seleção o multiplexador “Nodes Table Access Control”), e após o sinal de “start” inicia a atualização.

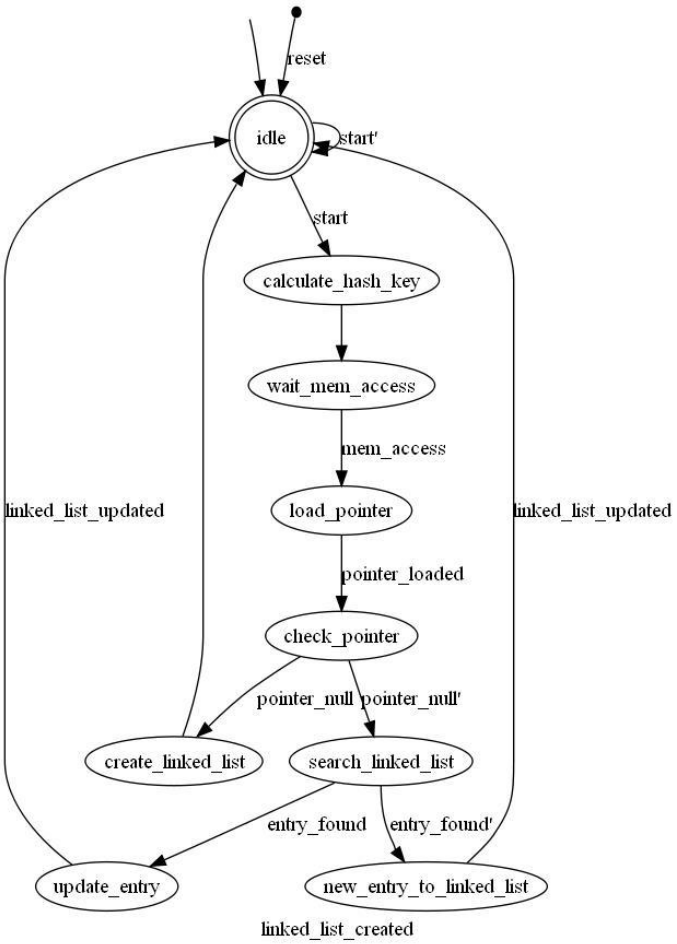


Figura 36: Máquina de estados de Update

O primeiro passo da verificação é o cálculo do valor hash, utilizando somente o endereço MAC como chave. O valor hash indica o endereço do ponteiro para lista encadeada onde será feita as tentativas de achar a entrada.

Logo após o cálculo do valor hash, o componente faz a requisição do acesso à memória.

Caso o ponteiro seja vazio, cria-se uma nova lista encadeada e o ponteiro é guardado na posição apontada pelo valor hash.

Caso o ponteiro não seja vazio, inicia-se a procura pela lista encadeada. Quando a entrada é encontrada, ela é atualizada (o formato de cada entrada na

tabela está definido na seção 3.3.5). Caso a entrada não seja encontrada, adiciona-se mais uma a lista encadeada.

5.4.3: Aging

O módulo “Aging” funciona de maneira muito parecida ao módulo “Aging” de “Duplicate Check”, conforme mostrado na máquina de estados da Figura 37

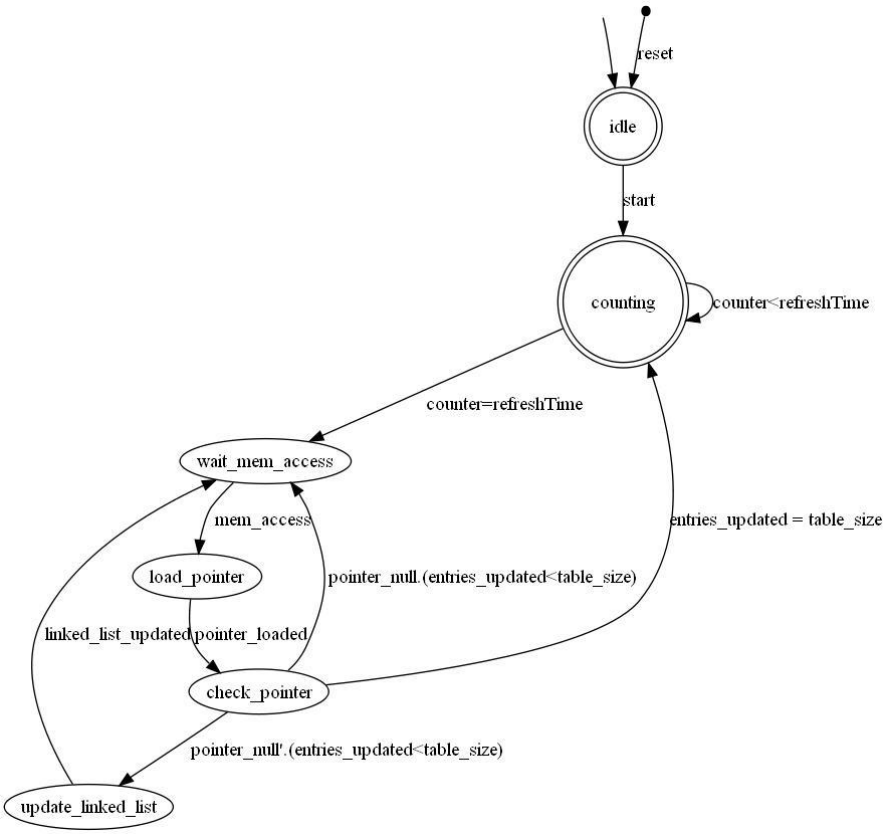


Figura 37: Máquina de estados de Aging

O componente faz uma varredura da tabela de ponteiros para listas encadeadas. A cada acesso a memória, o módulo verifica um ponteiro e se o ponteiro não for vazio, ele realiza a atualização dos tempos (incrementar os contadores de tempo) de cada entrada na lista encadeada (*TimeLastSeenA* e *TimeLastSeenB*).

Se *TimeLastSeenA* e *TimeLastSeenB* forem maiores que *NodeForgetTime*, a entrada é retirada da tabela.

5.4.4: Search

O componente “Search”, cujo funcionamento está representado pela máquina de estados da Figura 38, recebe como entrada um endereço MAC, e procura na tabela de nós a presença desse endereço. Caso o endereço esteja presente, o módulo retorna os valores *SanA* e *SanB*, do contrário ele retorna 0 para esses valores indicando para “TX Redundancy Master”, que o quadro é para ser enviado nas duas LANs. A procura pela entrada se inicia após o sinal de “start”.

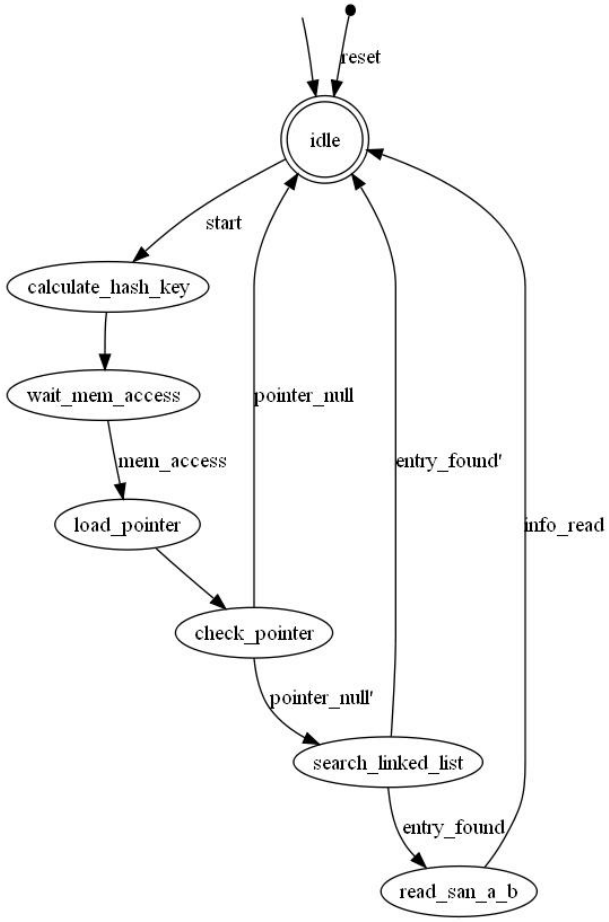


Figura 38: Máquina de estados de Search

O primeiro passo da verificação é o cálculo do valor hash, utilizando somente o endereço MAC como chave. O valor hash indica o endereço do ponteiro para lista encadeada onde será feita as tentativas de achar a entrada.

Logo após o cálculo do valor hash, o componente faz a requisição do acesso à memória.

Caso o ponteiro seja vazio, o módulo considera que a entrada não existe e finaliza a procura.

Caso o ponteiro não seja vazio, inicia-se a procura pela lista encadeada. Quando a entrada é encontrada, retorna os valores de *SanA* e *SanB*, senão considera que a entrada não existe e retorna para o estado “idle” e sinaliza que a procura finalizou.

5.4.5: Arbiter

O funcionamento de “Arbiter” também é parecido com o funcionamento de “Avalon-ST Arbiter” de “TX Redundancy Master”, porém o que está sendo coordenado nesse componente é o acesso à tabela de nós, a qual é disputada pelos componentes “Aging”, “Update” e “Search”. A Figura 39 apresenta a máquina de estados que representa o funcionamento de “Arbiter”.

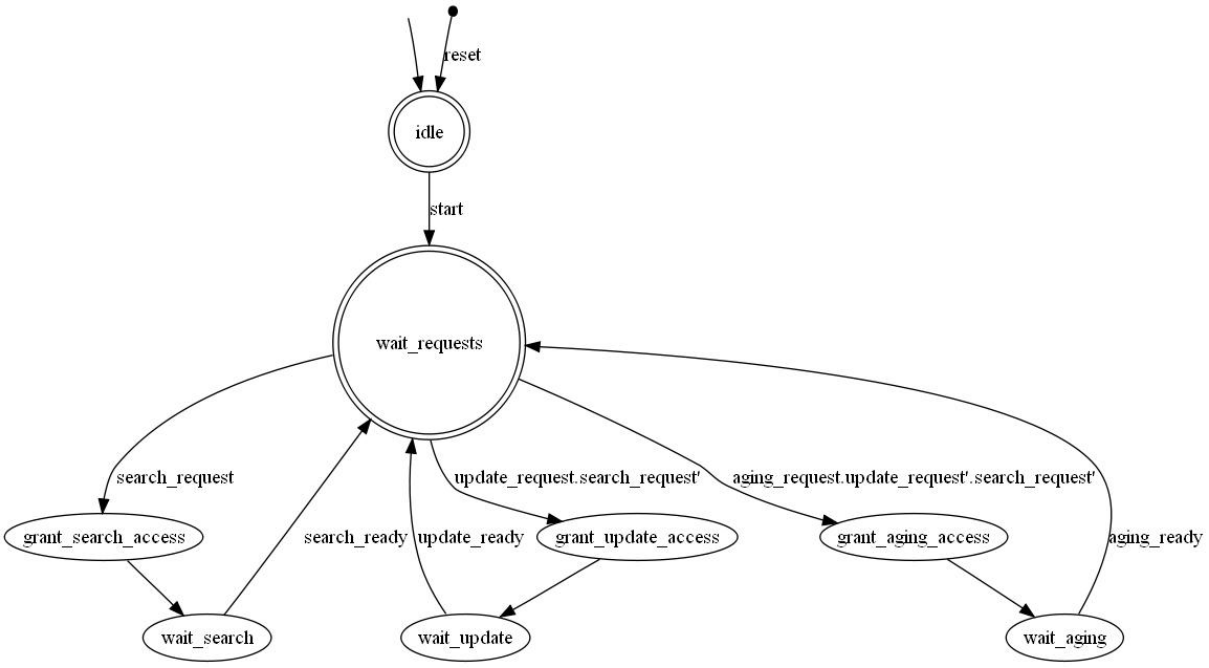


Figura 39: Máquina de estados de Arbiter

O “Arbiter” controla também o multiplexador “Memory Controller”. As entradas desse multiplexador são os sinais de acesso à memória de “Update”, “Aging” e “Search”, e o controle de “Arbiter”.

“Search” possui a prioridade de acesso mais alta, sendo seguido por “Update”. “Aging” possui a prioridade mais baixa.

5.5: Conclusão

Esse capítulo apresentou o projeto detalhado do circuito digital do LRE. Este circuito foi implementado no FPGA do kit de desenvolvimento, completando a arquitetura descrita no capítulo 4.

No próximo capítulo será mostrado como o FPGA foi programado para implementar o projeto apresentado neste capítulo, assim como os testes realizados e os resultados obtidos.

Capítulo 6: Implementação e Testes

Nesse capítulo é explicado como o projeto apresentado no capítulo 5 foi realizado em circuito digital, apresentando a linguagem VHDL, assim como as ferramentas utilizadas para a programação do FPGA.

Durante esse capítulo também serão mostrados as ferramentas utilizadas para testes, os testes realizados e os resultados obtidos.

6.1: Máquinas de estados síncronas

As máquinas de estado apresentadas no capítulo 5 foram implementadas em circuito lógico digital como máquinas de estados síncronas.

Máquinas de estado síncronas são circuitos sequenciais cujos elementos de memória (flip-flops) utilizam um mesmo sinal de clock. Esse tipo de máquina de estados modifica seu estado (saída dos flip-flops) somente com a mudança do sinal de clock (zero para um, ou o contrário).

A memória de estado é um conjunto de *n* flip-flops que guardam o estado atual da máquina, que tem 2^n estados distintos. O próximo estado da máquina é determinado pela lógica de próximo estado *F,* que é função do estado atual e das entradas. A lógica de saída *G* determina a saída, como função do estado atual e, opcionalmente, das entradas. Tanto *F* quanto *G* são circuitos combinacionais. A Figura 40 mostra um esquemático de uma máquina de estados.

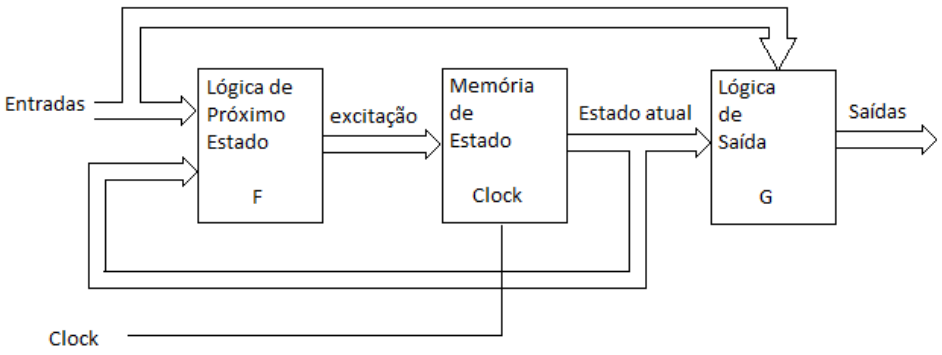


Figura 40: Máquina de estados síncrona

Na tradução das máquinas de estados do capítulo 5 para máquinas de estados síncronas, os estados foram representados pela memória de estado, enquanto os eventos de transição dos estados fazem parte da lógica de próximo estado. Os sinais de saída de cada componente fazem parte da lógica de saída.

6.2: VHDL

No desenvolvimento de circuitos digitais, uma abstração é um modelo simplificado do sistema, mostrando somente algumas características e ignorando detalhes associados. O motivo para uma abstração é reduzir a quantidade de dados para um nível manuseável, a tal ponto que somente informações importantes são mostradas. Uma abstração de alto nível contém somente as informações mais vitais. Uma abstração de baixo nível é mais precisa e contém informações antes ignoradas. Apesar de ser mais complexa, a abstração de baixo nível é a mais próxima de um circuito real. No desenvolvimento de circuitos digitais, começa-se pela abstração de mais alto nível, e à medida que o sistema é melhor compreendido, mais detalhes são incluídos até chegar ao circuito real.

Uma linguagem HDL deve ser capaz de descrever com precisão um circuito digital no nível desejado de abstração. Como linguagens HDL são modeladas para um hardware, sua semântica e uso são bem diferentes de linguagens tradicionais de programação.

A maioria das linguagens de programação tradicionais, como C, são modeladas sobre um processo sequencial. Em nível de abstração, um processo sequencial facilita o desenvolvimento de um algoritmo passo a passo. Em nível de implementação, o processo sequencial lembra a operação de um computador e permite uma tradução eficiente de um algoritmo para linguagem de máquina.

As características de um circuito digital, por outro lado, são bem diferentes de um modelo sequencial de processamento. Um sistema digital típico é normalmente construído por partes menores, com entradas e saídas conectadas entre si. Quando um sinal muda, as partes conectadas ao sinal são ativadas e um conjunto de novas operações é iniciado. Essas operações ocorrem de forma paralela, e cada uma irá levar certa quantidade de tempo, que representa o atraso de propagação, para completar. Depois de completada as operações, as partes atualizam suas saídas.

Se o valor da saída modificar, ela irá ativar uma nova rodada de operações. Essa descrição mostra várias características típicas de circuitos digitais, incluindo conexão de entradas e saídas, operações paralelas, atrasos de propagação etc. O modelo sequencial de programação não consegue representar essas características, e por isso existem as linguagens HDL.

Um código em HDL tem três funções principais:

- Documentação: Um programa HDL pode ser usado como uma especificação formal do sistema digital e documentação, substituindo a linguagem humana que pode ser muito ambígua.
- Entrada para um simulador: Simulação é usada para estudar e verificar a operação de um circuito digital, sem ter de construí-lo. Um simulador HDL fornece uma ferramenta para modelar operações paralelas em um computador, cujas operações são sequenciais. Um programa HDL, combinado com um código de geração de vetor de testes e coleta de dados, formam um *testbench*, o qual serve de entrada para o simulador. Durante execução, o simulador interpreta o código HDL e gera respostas.
- Entrada para uma ferramenta de síntese (synthesizer): O software de síntese recebe um programa HDL como entrada e cria o circuito a partir de componentes de uma biblioteca. A saída de um synthesizer é um novo HDL que representa a descrição estrutural do circuito criado.

Para a programação do FPGA do kit de desenvolvimento, foi utilizado nesse trabalho o VHDL (Very High Speed Integrated Circuit Hardware Description Language). VHDL é uma linguagem HDL desenvolvida pelo departamento de defesa dos Estados Unidos em meados dos anos 80, com finalidade de ser um padrão de documentação de hardware. Atualmente, o VHDL é um padrão IEEE.

O VHDL permite a descrição de um circuito digital em nível de abstração RTL, ou porta lógica. No nível de porta lógica, os circuitos digitais são construídos com portas lógicas simples (AND, OR, NOT), flip-flops e latches, os quais são feitos com elementos do nível de transistor. Nesse nível o comportamento do sistema é descrito por equações booleanas, e os dados são valores discretos. No nível de registradores (Register Transfer Level, ou RTL), os circuitos digitais são

confeccionados com módulos feitos por portas lógicas. Esses módulos podem ser operadores aritméticos, elementos de memória como registradores, e multiplexadores. O comportamento do circuito digital é descrito por máquinas de estado, e os dados são agrupados e interpretados por tipos de mais alto nível como estado ou inteiro.

A seguir temos um exemplo de descrição de um circuito digital de detecção de paridade par em VHDL. Os componentes descritos no capítulo 5 foram traduzidos para o VHDL seguindo estrutura semelhante.

```
library ieee;
use ieee.std_logic_1164.all;

entity even_detector is

  port (
    a    : in  std_logic_vector;
    even : out std_logic);

end entity even_detector;

architecture sop_arch of even_detector is
  signal p1, p2, p3, p4 : std_logic;
begin  -- architecture sop_arch
  even <= (p1 or p2) or (p3 or p4);
  p1   <= (not a(0)) and (not a(1)) and (not a(2));
  p2   <= (not a(0)) and a(1) and a(2);
  p3   <= a(0) and (not a(1)) and a(2);
  p4   <= a(0) and a(1) and (not a(2));
end architecture sop_arch;
```

Figura 41: Circuito de detecção de paridade par em VHDL

Como se pode observar, o código consiste de duas grandes unidades: *entity* e *architecture*. O “entity” especifica as portas de entrada e saída do circuito. A unidade “architecture” especifica a operação ou organização interna do circuito.

Um dos principais usos do VHDL é simulação. Fazer uma simulação em VHDL é similar a fazer um experimento com um circuito físico, no qual conectamos as entradas do circuito a algum estímulo e observamos o resultado. Simular uma descrição VHDL é como fazer uma experiência virtual, no qual o circuito físico é substituído pelo código VHDL. Na Figura 42 temos o “testbench” do detector de paridade par. Nesse ambiente de simulação é instanciado um componente do detector de paridade, e suas entradas e saídas são conectadas ao processo gerador de vetor de teste e ao verificador. É importante ressaltar que algumas diretivas utilizadas no “testbench” (como, por exemplo, “wait”), não podem ser utilizadas para criar circuitos digitais porque não são realizáveis em circuito físico.

Para verificação do funcionamento correto do algoritmo de detecção de duplicados foi utilizado testbenchs escritos em VHDL que simulavam a recepção de

quadros pelo LRE, para em seguida analisar os resultados produzidos pelo algoritmo.

```
library ieee;
use ieee.std_logic_1164.all;
entity even_detector_testench is
end even_detector_testbench;

architecture tb_arch of even_detector_testbench is
  component even_detector
    port(
      a: in std_logic_vector(2 downto 0);
      even: out std_logic;
      end component;
      signal test_in: std_logic_vector(2 downto 0);
      signal test_out: std_logic;
      begin
        -- instantiate the circuit under test
        uut: even_detector
          port map(a=>test_in, even=>test_out);
        -- test vector generator
        process
        begin
          test_in <= "000";
          wait for 200 ns;
          test_in <= "001";
          wait for 200 ns;
          test_in <= "010";
          wait for 200 ns;
          test_in <= "011";
          wait for 200 ns;
        end process;
        -- verifier
        process
          variable error_status: boolean;
        begin
          wait on test_in;
          wait for 100 ns;
          if((test_in="000" and test_out='1') or
             (test_in="001" and test_out='0') or
             (test_in="010" and test_out='0') or
             (test_in="011" and test_out='1'))
          then
            error_status := false;
          else
            error_status := true;
          end if;
          -- error reporting
          assert not error_status
            report "test failed."
            severity note;
        end tb_arch;
```

Figura 42: Testbench do circuito detector de paridade par

Programar em VHDL exige alguns cuidados, por exemplo, é preciso escrever o código de tal maneira que garanta que o “synthesizer” consiga interpretar e gerar o circuito digital desejado no FPGA. Alguns circuitos como memória RAM, tem sua maneira própria de ser representado em VHDL sem que haja dúvidas que o “synthesizer” irá conseguir interpretar. Na Figura 43 está demonstrado um modelo de declaração de registradores, e na Figura 44 está representado um modelo de memória RAM. As memórias e registradores utilizados pelo LRE foram descritas em VHDL utilizando estrutura semelhante à mostrada nessas figuras.

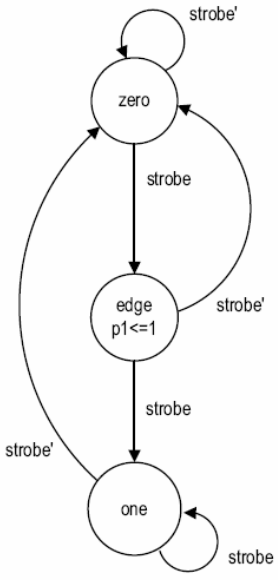
```
-- registers
process(clk, reset)
begin
  if reset = '1' then
    state_reg         <= idle;
    key_reg           <= (others => '0');
    val_reg           <= (others => '0');
    bytes_counter_reg <= (others => '0');
  elsif rising_edge(clk) then
    state_reg         <= state_next;
    key_reg           <= key_next;
    val_reg           <= val_next;
    bytes_counter_reg <= bytes_counter_next;
  end if;
end process;
```

Figura 43: Exemplo de declaração de registradores em VHDL

```
process(clk)
begin
  if rising_edge(clk) then
    if (pointer_memory_we = '1') then
      pointer_memory(CONV_INTEGER(pointer_memory_write_address)) <= pointer_memory_writedata;
    else
      pointer_memory_readdata <= pointer_memory(CONV_INTEGER(pointer_memory_read_address));
    end if;
  end if;
end process;
```

Figura 44: Exemplo de declaração de memória RAM em VHDL

Na Figura 45 está o código que representa uma máquina de estados síncrona.




```
library ieee;
use ieee.std_logic_1164.all;

entity edge_detector1 is
  port (
    clk, reset : in  std_logic;
    strobe     : in  std_logic;
    p1         : out std_logic);
end entity edge_detector1;

architecture moore_arch of edge_detector1 is
  type state_type is (zero, edge, one);
  signal state_reg, state_next : state_type;
begin  -- architecture moore_arch
  -- state register
  process (clk, reset) is
  begin  -- process
    if reset = '1' then
      state_reg <= zero;
    elsif clk'event and clk = '1' then
      state_reg <= state_next;
    end if;
  end process;
  -- next state logic
  process (state_reg, strobe) is
  begin  -- process
    case state_reg is
      when zero =>
        if strobe = '1' then
          state_next <= edge;
        else
          state_next <= zero;
        end if;
      when edge =>
        if strobe = '1' then
          state_next <= one;
        else
          state_next <= zero;
        end if;
      when one =>
        if strobe = '1' then
          state_next <= one;
        else
          state_next <= zero;
        end if;
    end case;
  end process;
  -- moore output logic
  p1 <= '1' when state_reg = edge else '0';
end architecture moore_arch;
```

Figura 45: Exemplo de máquina de estado em VHDL

O código é dividido em três partes como no modelo de máquinas de estado síncronas: declaração de registradores de estado, lógica de próximo estado e lógica de saída. As máquinas de estado síncronas que realizam os circuitos descritos nas máquinas de estados do capítulo 5 foram descritas em VHDL seguindo estrutura semelhante à mostrada na Figura 45, porém adicionando os sinais, registradores, estados e outras particularidades específicas de cada circuito.

Existem muitos outros detalhes de semântica e sintaxe em VHDL, o qual este documento não entrará em detalhe. Mais sobre a linguagem pode ser encontrado em [13].

6.2.1: Programando um FPGA

Na Figura 46 temos o processo de programação de um FPGA. Este processo foi utilizado para programar, a partir do código VHDL criado, o FPGA do kit de desenvolvimento para que realizasse a arquitetura mostrada na seção 4.3.

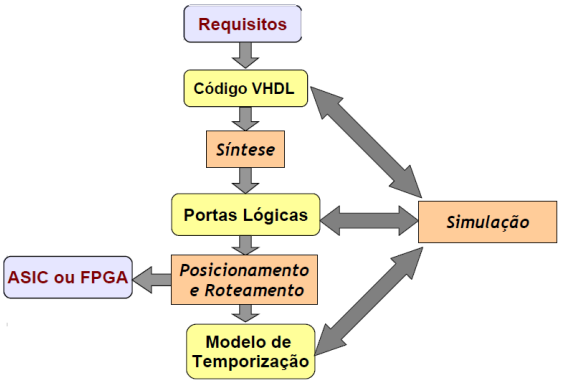


Figura 46: Processo de programação de um FPGA

A entrada do processo é um arquivo VHDL que descreve a funcionalidade do circuito. Essa descrição é feita em nível de abstração RTL.

Em seguida a descrição RTL passa por um processo de síntese o qual gera um código VHDL em nível de abstração de porta lógica. O resultado desse passo normalmente é chamado de “netlist”

O próximo passo é planejar a configuração dos blocos lógicos do FPGA para atender o especificado no netlist. A ferramenta de roteamento e posicionamento gera arquivos de leiaute e configuração, os quais não são VHDL, e um arquivo com informações de temporização do circuito digital.

O último passo é a programação do FPGA, em que os blocos lógicos são configurados e interconectados de tal forma que gere o circuito digital descrito no arquivo VHDL que serviu de entrada para o processo.

6.3: Ferramentas utilizadas

Para compilação do código em VHDL e programação do FPGA foi utilizado o programa Quartus II da Altera (Figura 47). O Quartus também realiza análise temporal para verificar se os requisitos temporais dos elementos de memória são respeitados e se os atrasos de propagação estão dentro do aceitável para uma determina frequência.

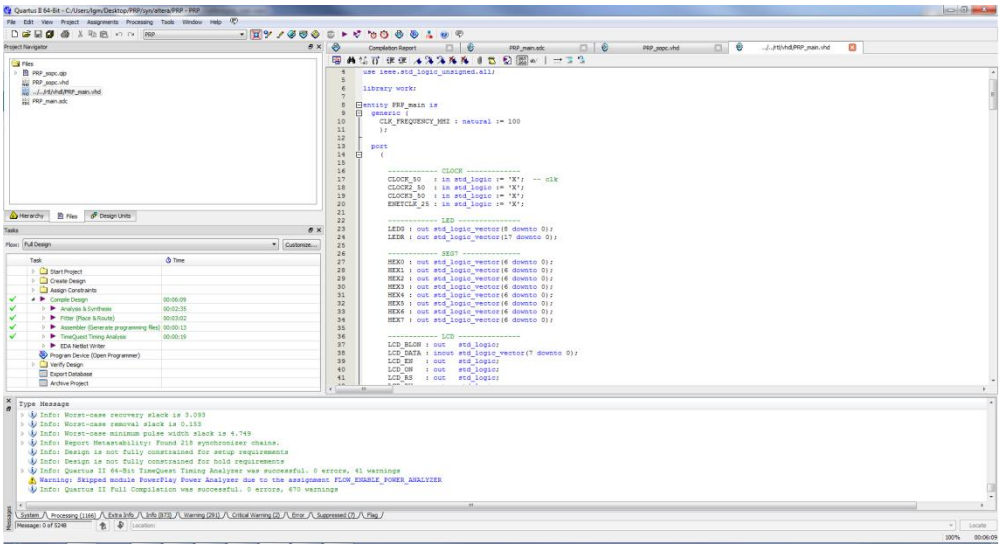


Figura 47: Ambiente Quartus II

Para simulação do funcionamento dos circuitos digitais foi utilizado o software ModelSim da Mentor Graphics. Esse software realiza suas simulações baseado em testbenchs escritos em VHDL, e possui uma interface gráfica (Figura 48) para mostrar os sinais gerados na simulação.

Para testes de verificação dos quadros sendo transmitidos na rede, foi utilizado o programa Wireshark. O Wireshark é um programa de análise de protocolos de rede, que realiza a captura e leitura de todos os quadros que estão sendo transmitidos. Ele foi utilizado nesse trabalho para verificar se o LRE estava inserindo o RCT corretamente nos quadros sendo transmitidos.

Para verificar se o LRE estava descartando os quadros duplicados corretamente, foi utilizada a ferramenta tcpdump. O tcpdump também é um

programa de análise de protocolo de rede, porém ele é capaz de ser executado pelo sistema operacional utilizado pelo processador NIOS, e por isso foi utilizado para verificar se o LRE encaminhava corretamente os quadros para as camadas superiores.

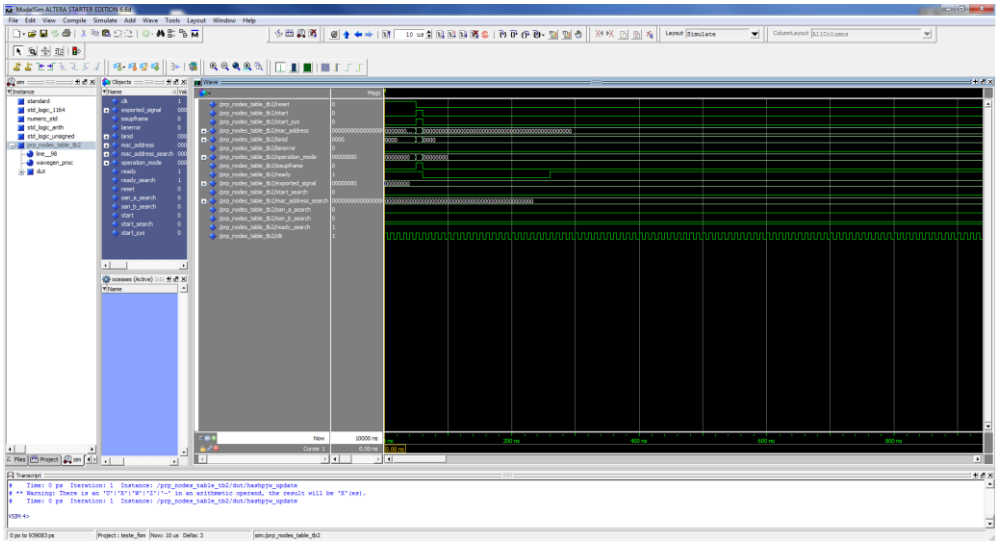


Figura 48: Ambiente ModelSim

6.4: Testes de comunicação

O FPGA do kit de desenvolvimento foi programado para realizar a arquitetura descrita na seção 4.3. O processador NIOS foi programado para operar um servidor Web em que o usuário poderia realizar as seguintes funções via internet: escrever um número nos displays sete segmentos, escrever textos no display LCD e acender os LEDs verdes do kit de desenvolvimento. A interface do servidor Web está apresentada na Figura 49.

Todos os quadros enviados pelo NIOS são interceptados pelo LRE, o qual insere o RCT e transmite o quadro pelas duas portas de rede.

Durante o desenvolvimento, para simplificação, as duas portas de rede do kit de desenvolvimento foram conectadas na mesma LAN (o PRP exige a conexão em LANs diferentes).

O funcionamento correto do servidor Web permitiu verificar que o LRE consegue receber quadros e repassar para as camadas superiores, assim como receber quadros das camadas superiores e transmiti-los pela rede, de forma correta e transparente para a aplicação.

Figura 49: Interface do servidor Web

O servidor Web utiliza protocolos como ARP, DHCP, TCP, HTTP etc., portanto, o funcionamento correto permitiu verificar que a inserção do RCT nos quadros não interfere nos protocolos das camadas superiores, como era de se esperar.

No funcionamento normal do servidor Web, os dois adaptadores de rede do kit de desenvolvimento estão conectados na LAN, enviando e recebendo quadros simultaneamente. Ao desconectar uma das portas, o servidor continuou funcionando corretamente, satisfazendo o requisito básico do PRP que é fornecer a redundância de comunicação na rede.

Na Figura 50 temos um exemplo de quadro DHCP enviado pelo servidor Web (o servidor Web solicita um IP para o servidor DHCP da LAN) com o RCT nos seis últimos bytes (00-2C-B1-5A-88-FB). O número de sequência é “x002C”, o

identificador de LAN é “xB” (LAN B), o LSDU size é “x15A” (346 bytes em número decimal) e o sufixo PRP é “x88FB”. Nesse caso, somente a porta B estava conectada na rede.

É importante dizer que o Wireshark mostra os quadros que o adaptador de rede do computador repassa para o programa, e antes de repassar o quadro, o preâmbulo e o checksum são retirados.

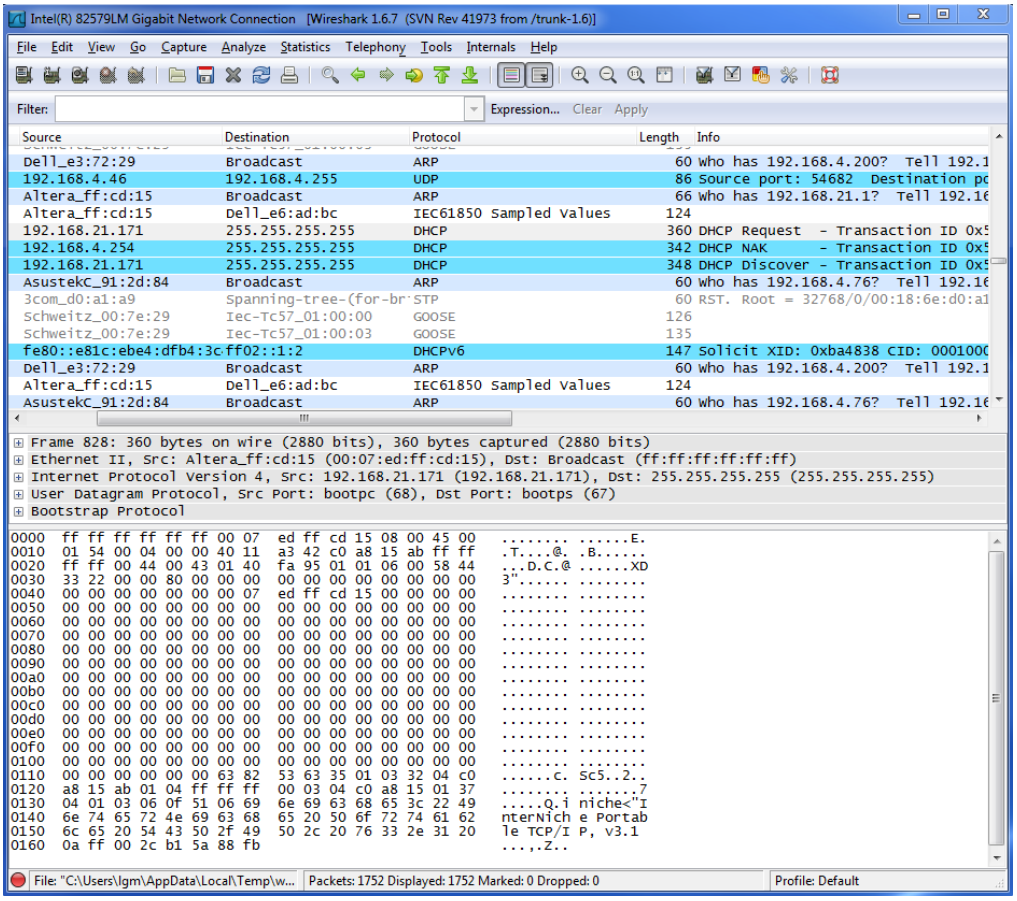


Figura 50: Exemplo de quadro DHCP com RCT, capturado pelo Wireshark

Na Figura 51, temos exemplo de um quadro do protocolo “IEC 61850 Sampled Values” que foi recebido das camadas superiores pelo LRE. O quadro foi duplicado, recebeu a inserção do RCT com identificação de LAN correto, e enviado pelas duas portas de rede.

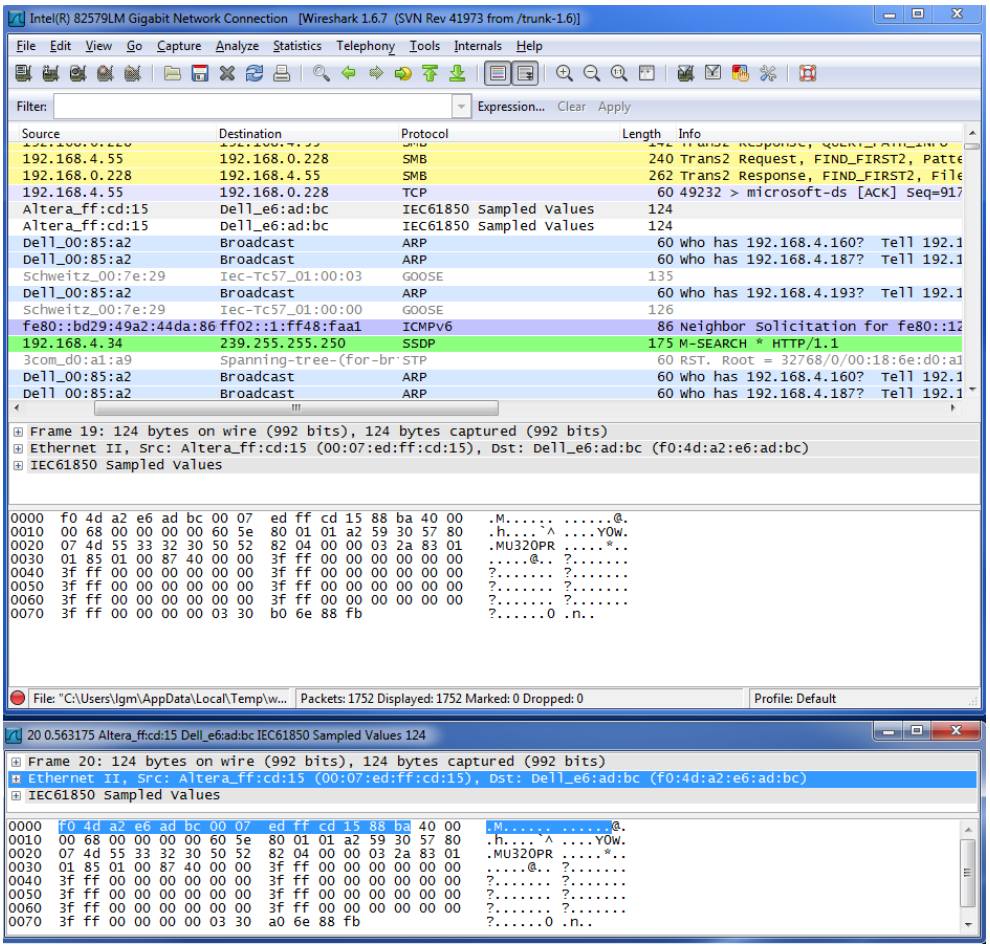


Figura 51: Exemplo de quadro IEC 61850 Sampled Values

Nos testes realizados, o LRE foi programado para enviar, a cada cinco segundos, um quadro de supervisão PRP com o formato descrito na seção 3.3.5. A Figura 52 mostra os dois quadros de supervisão (LAN A e LAN B) capturados pelo Wireshark.

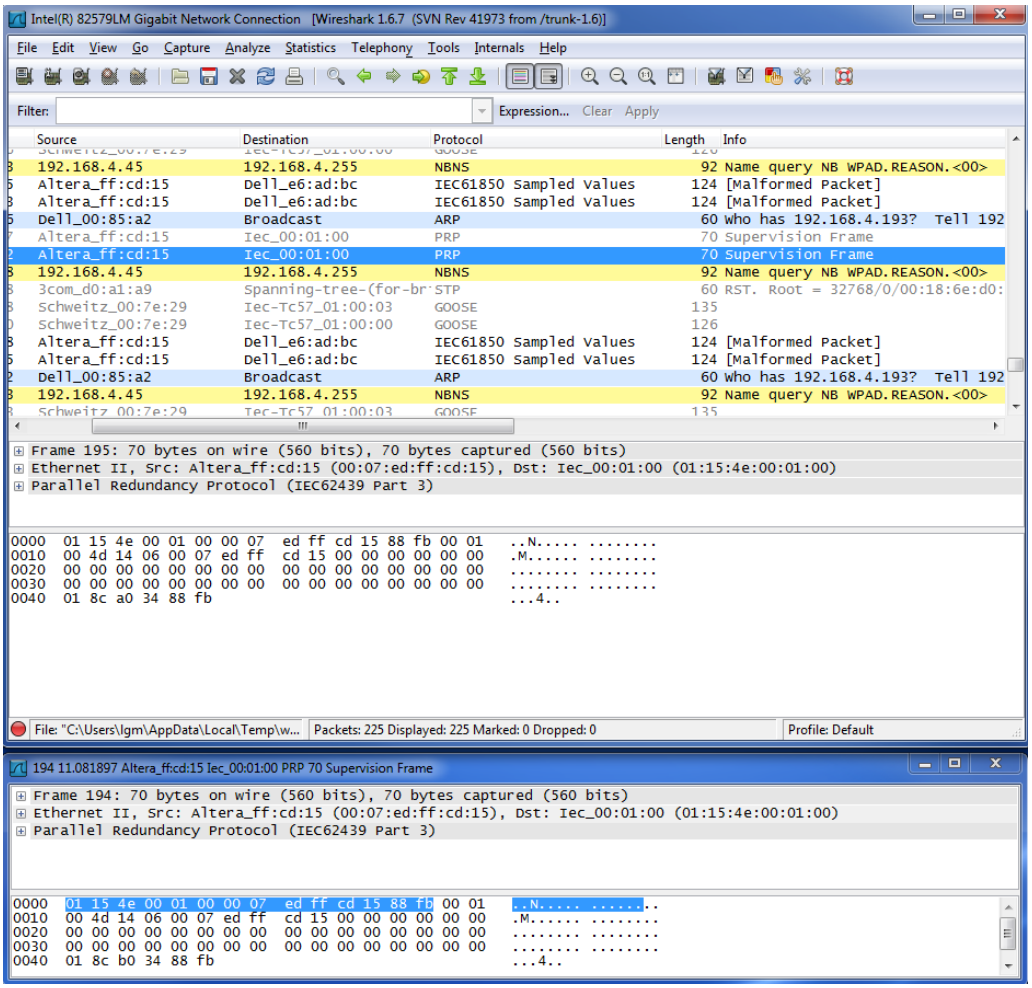


Figura 52: Quadro de supervisão PRP

6.5: Testes do algoritmo de descarte de duplicados

Para verificar o correto funcionamento do LRE, seus componentes foram testados em simulação feita no software ModelSim. Para a simulação, foi criado programas testbench em VHDL, para gerar sinais de estímulo e observar na simulação as respostas.

Na Figura 53, temos a resposta do componente “Duplicate Check” em uma simulação. No testbench, foram gerados sinais que simulassem a requisição de verificação se um quadro com endereço de origem *src_mac_address* e número de sequência *SeqNr.* Em seguida simula-se a requisição de verificação de um quadro com o mesmo endereço de origem e número de sequência, como se fosse o segundo quadro de um par. Para finalizar, ocorre a chegada de um quadro com mesmo endereço de origem, porém com número de sequência diferente, simulando a chegada de um novo quadro do mesmo nó.

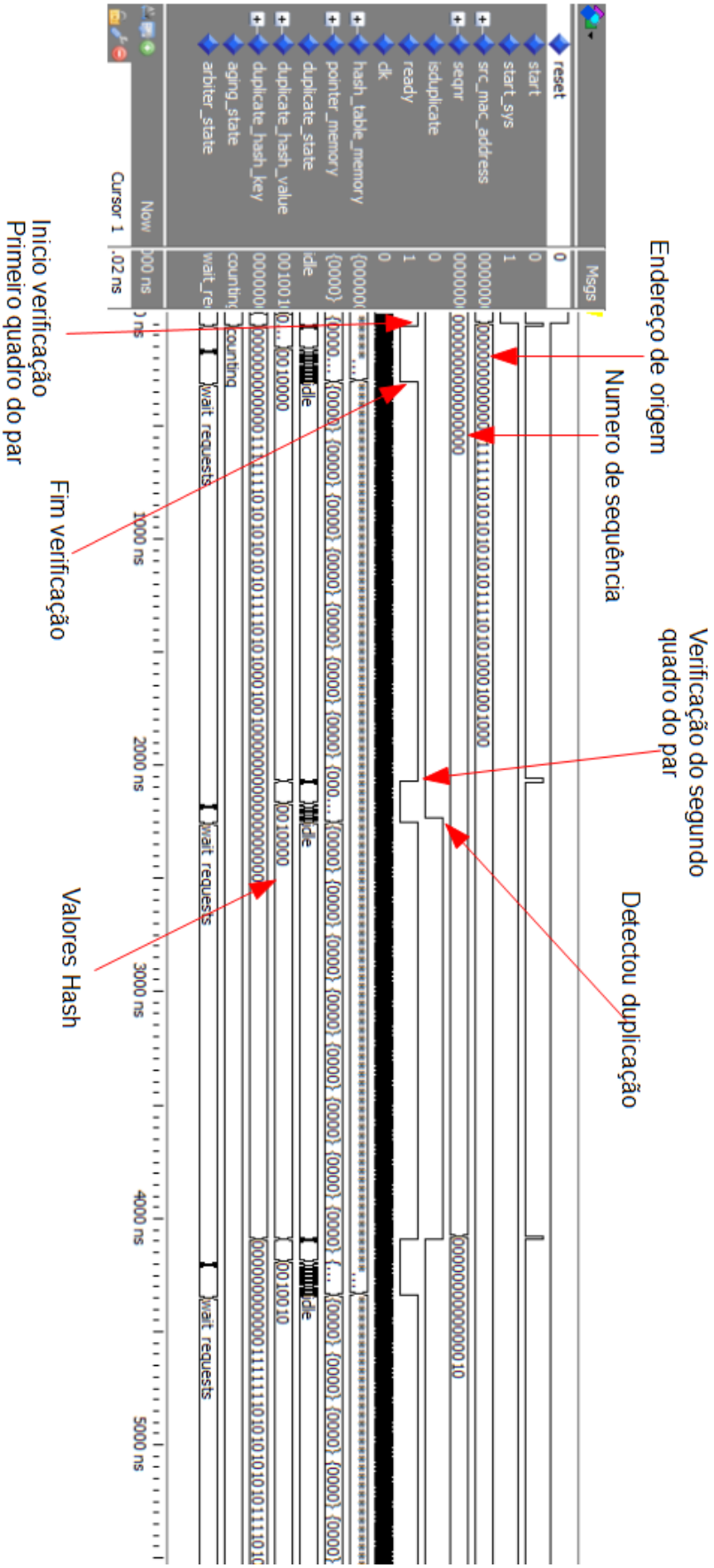


Figura 53: Simulação do componente Duplicate Check

Esse teste mostra como o algoritmo de detecção de duplicados funciona, porém para validar seu funcionamento, testes mais rigorosos foram feitos.

Como uma subestação de pequeno e médio porte tem entre 20 a 40 IEDs, e a quantidade de fabricantes diferentes não costuma variar muito (entre 10 e 20), foi criado um testbench que simulasse a existência de 40 nós, de 14 fabricantes diferentes, enviando quadros para o nó sendo testado. A simulação considerou que os nós enviavam quadros ao mesmo tempo, e que o segundo quadro de um par de todos os nós sempre chegassem antes de uma nova rodada de mensagens. Nessa simulação ocorreram 50 rodadas de mensagens, totalizando 4000 mensagens recebidas (LAN A e LAN B).

A tabela hash utilizada para simulação possuía 128 buffers circulares com quatro posições cada um, totalizando 512 entradas.

Foram realizados três testes diferentes com essa configuração. Na primeira simulação, considerou-se que todos os nós mandavam mensagens com mesmo número de sequência, logo a chave para a função hash somente era diferente pelo endereço de origem. Para esse teste o resultado foi bem ruim, com eliminação de algo próximo de 10% das mensagens duplicadas apenas.

No segundo teste, considerou-se que todos os nós mandavam mensagens com número de sequência crescendo uniformemente, porém com valores diferentes uns dos outros. Para esse teste houve eliminação de 100% dos duplicados e não foi descartado nenhum quadro válido como duplicado.

No terceiro teste, considerou-se que todos os nós mandavam quadros com números de sequência aleatórios. O resultado foi eliminação de 100% dos duplicados e não foi descartado nenhum quadro válido como duplicado.

A situação do primeiro teste foi muito ruim, porém é algo muitíssimo improvável de acontecer, visto que o número de sequência é incrementado para qualquer quadro inválido, independente do destino, portanto ele não invalida o algoritmo.

Os resultados do segundo e terceiro testes são muito bons e validam o algoritmo. Os resultados bons eram esperados, pois a número de valores hash (128) é bem maior que o número de IEDs (40).

Na Figura 55, temos a simulação da atualização dos tempos das entradas na tabela de duplicados. A taxa de atualização de tempos escolhido para a simulação foi de 10 ms, e o tempo máximo de permanência de uma entrada foi 400 ms (40 atualizações), já que a rede utilizada era de 100 Mbit/s (o tempo mínimo de reinício de contagem do número de sequência é de 440 ms).

Nos testes realizados, o componente “Aging” de “Duplicate Check” funcionou corretamente, atualizando corretamente as entradas e eliminando quando o tempo de permanência ultrapassasse 400 ms.

Para realizar um teste real do funcionamento do algoritmo de descarte de duplicados, foram utilizados dois kits de desenvolvimento iguais, sendo que um funcionou como transmissor de quadros e o outro como receptor. Os dois links de comunicação (LAN A e LAN B) do transmissor e receptor foram ligados diretamente via cabo crossover. O FPGA do kit transmissor foi programado com o circuito digital da “Merging Unit”, um dos produtos sendo desenvolvidos pela Reason, com a adição do LRE realizado nesse trabalho para operar o PRP. O kit enviava mensagens do protocolo GOOSE (protocolo utilizado na norma IEC 61850) pelas duas portas de rede (LAN A e LAN B), com o RCT adicionado, a cada segundo. O FPGA do kit receptor foi programado com a mesma arquitetura do transmissor, porém o processador foi configurado para executar a ferramenta tcpdump. Através da ferramenta tcpdump do receptor, constatou-se que apenas uma das mensagens GOOSE chegava ao processador a cada segundo, portanto o LRE estava descartando corretamente o quadro duplicado. O arquivo log produzido pelo tcpdump está na Figura 54

Todos os testes realizados com o algoritmo de descarte de duplicados mostraram também que, o tempo máximo gasto para fazer a checagem é de 420 ns.

```
00:45:45.563477 00:70:ed:11:12:1d (oui Unknown) > 01:0c:cd:01:00:00 (oui Unknown), ethertype 802.1Q (0x8100), length 179:
        0x0000:  e000 88b8 1000 009b 0000 0000 6181 9080  ............a...
        0x0010:  184d 5533 3230 434f 5245 2f4c 4c4e 3024  .MU320CORE/LLN0$
        0x0020:  474f 2447 4f43 4230 3181 0207 d082 184d  GO$GOCB01......M
00:45:46.565474 00:70:ed:11:12:1d (oui Unknown) > 01:0c:cd:01:00:00 (oui Unknown), ethertype 802.1Q (0x8100), length 179:
        0x0000:  e000 88b8 1000 009b 0000 0000 6181 9080  ............a...
        0x0010:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0020:  0000 0000 0000 0000 0000 0000 0000 0000  ................
00:45:47.567497 00:70:ed:11:12:1d (oui Unknown) > 01:0c:cd:01:00:00 (oui Unknown), ethertype 802.1Q (0x8100), length 179:
        0x0000:  e000 88b8 1000 009b 0000 0000 6181 9080  ............a...
        0x0010:  184d 5533 3230 434f 5245 2f4c 4c4e 3024  .MU320CORE/LLN0$
        0x0020:  474f 2447 4f43 4230 3181 0207 d082 184d  GO$GOCB01......M
00:45:48.569479 00:70:ed:11:12:1d (oui Unknown) > 01:0c:cd:01:00:00 (oui Unknown), ethertype 802.1Q (0x8100), length 179:
        0x0000:  e000 88b8 1000 009b 0000 0000 6181 9080  ............a...
        0x0010:  184d 5533 3230 434f 5245 2f4c 4c4e 3024  .MU320CORE/LLN0$
        0x0020:  474f 2447 4f43 4230 3181 0207 d082 184d  GO$GOCB01......M
00:45:49.571490 00:70:ed:11:12:1d (oui Unknown) > 01:0c:cd:01:00:00 (oui Unknown), ethertype 802.1Q (0x8100), length 179:
        0x0000:  e000 88b8 1000 009b 0000 0000 6181 9080  ............a...
        0x0010:  184d 5533 3230 434f 5245 2f4c 4c4e 3024  .MU320CORE/LLN0$
        0x0020:  474f 2447 4f43 4230 3181 0207 d082 184d  GO$GOCB01......M
```

Figura 54: Log da ferramenta tcpdump para o teste com dois kits

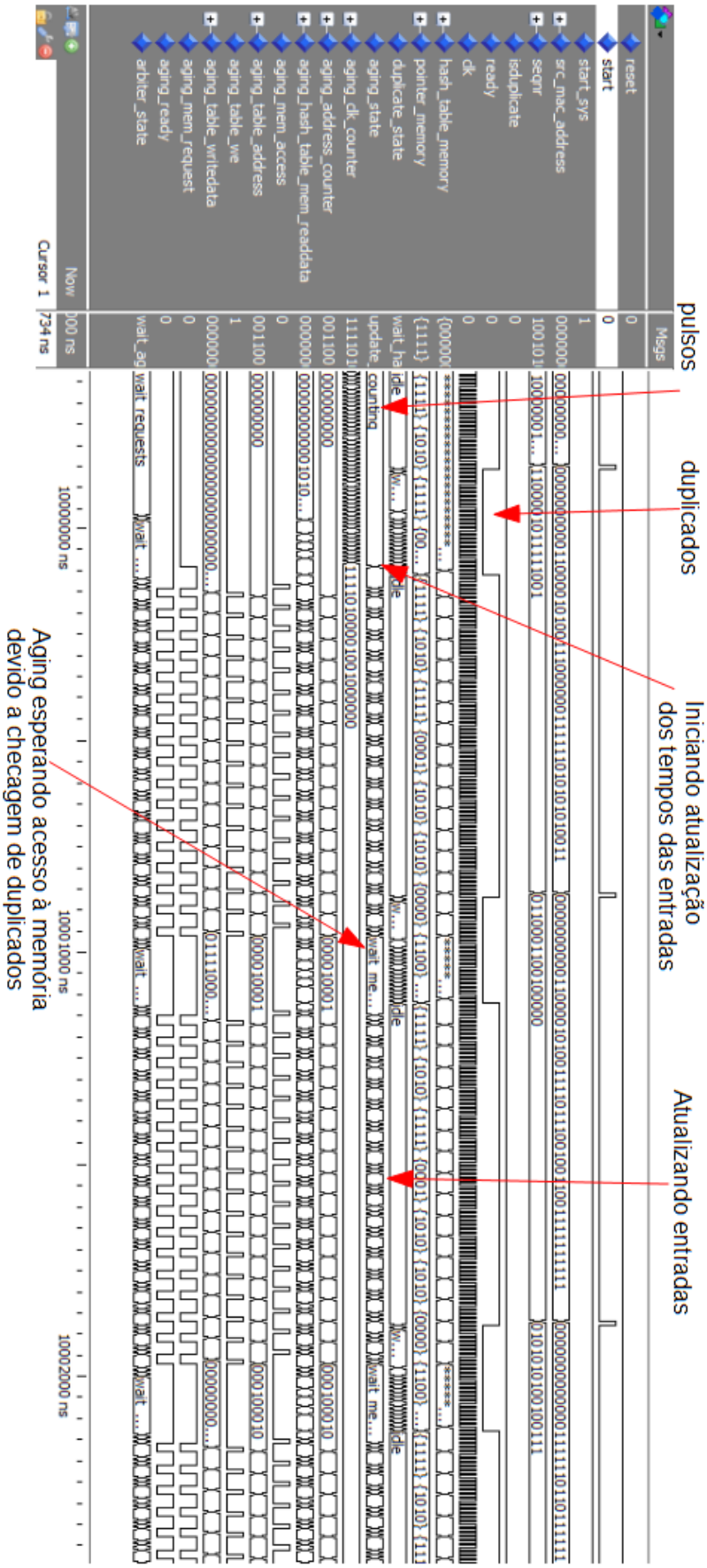


Figura 55: Simulação do componente Aging

6.6: Testes da tabela de nós

Na Figura 56, temos a resposta do componente “Nodes Table” em uma simulação no ModelSim. No testbench, foram gerados sinais que simulassem a requisição de atualização (componentes “Update”) da tabela de nós, dando como entrada um quadro de supervisão que chegou pela LAN A com endereço de origem *src_mac_address* e número de sequência *SeqNr.*

Foram feitos outros testes que simulassem a atualização da tabele de nós com diferentes configurações possíveis do conjunto de entradas (endereço de origem, LAN em que foi recebido, erro de LAN, quadro de supervisão ou não, modo de operação) e a função se comportou como esperado, atualizando entradas e criando novas entradas quando necessário. O tamanho da tabela utilizada foi de 128 entradas.

O tempo mínimo que o componente “Update” gasta para realizar sua função é de 300 ns. O tempo máximo possível é de 7.6 microssegundos, e acontece quando o “Update” tem que criar uma nova entrada, inserindo um novo elemento em uma lista encadeada de tamanho 127 (tendo que verificar 127 entradas antes de concluir que o nó não tem registro na tabela) e tendo que verificar a memória inteira antes de achar uma posição vazia. Esse pior tempo é muitíssimo improvável de acontecer, já que temos que considerar a existência de 128 nós na rede, e o endereço de origem de todos eles gera o mesmo valor hash.

Nos testes realizados, o componente “Aging” de “Nodes Table” funcionou corretamente, atualizando corretamente as entradas (a cada 1s) e eliminando quando o tempo de permanência ultrapassasse 60s.

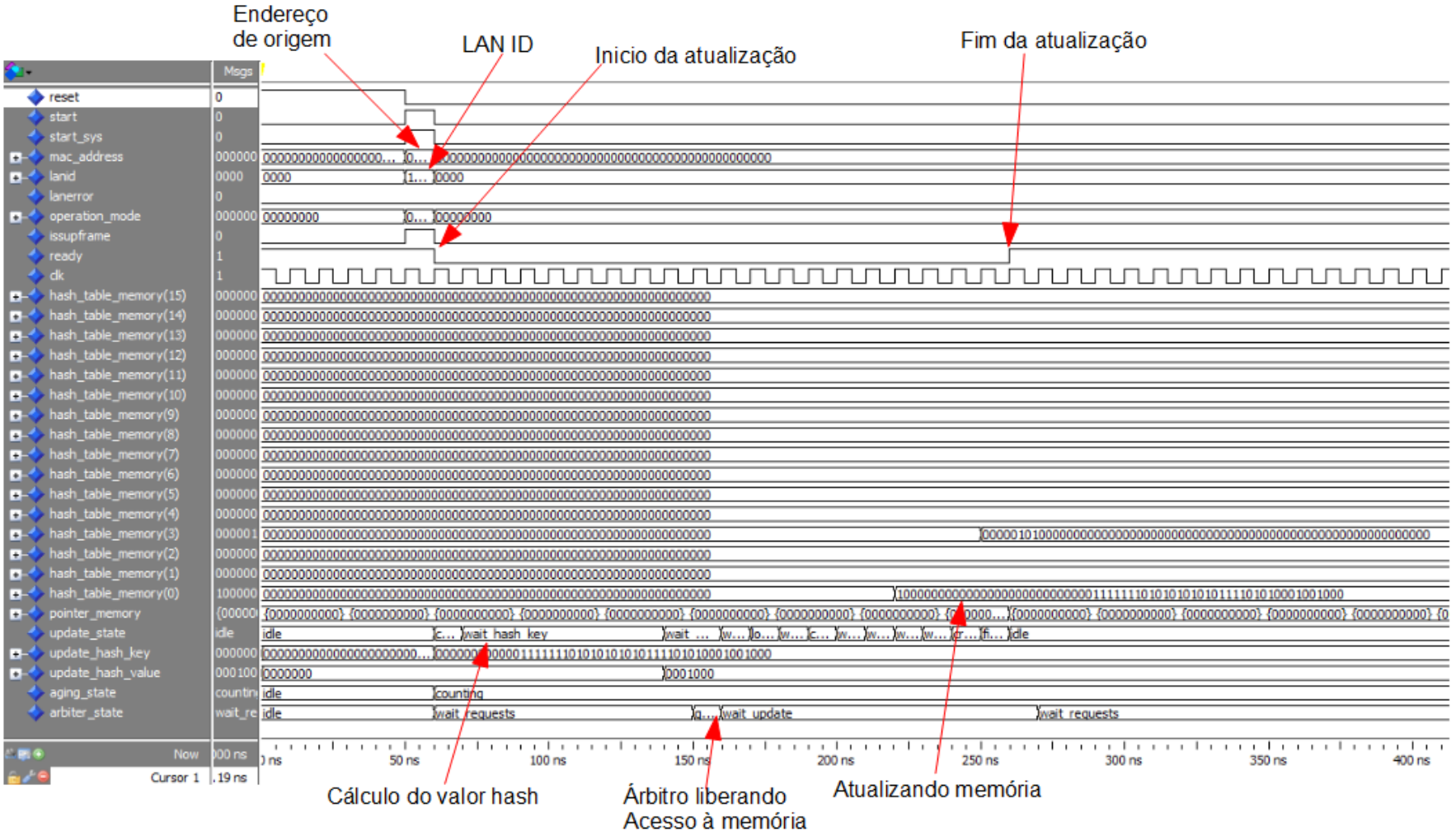


Figura 56: Simulação do funcionamento de Nodes Table

Na Figura 57, temos a resposta do componente “LRE” em uma simulação. No testbench, foram gerados sinais que simulassem a recepção de dois quadros de um par, ao mesmo tempo. Somente um quadro do par é repassado para as camadas superiores enquanto o outro é descartado.

Na Figura 58, temos o testbench que simula a recepção de um quadro de um SAN com endereço de origem *src_mac_address* pela porta B. Em seguida, o nó envia um quadro com endereço destino *src_mac_address.* O quadro é enviado somente pela porta B, pois a procura na tabela de nós indica que o endereço destino é um SAN acessível pela porta B.

Durante os testes, foi calculado um tempo de 1290 ns (129 pulsos de clock a 100 MHz) para o LRE receber dois quadros de tamanho mínimo (70 bytes), fazer a checagem de duplicados, atualizar a tabela de nós, e encaminhar ambos para as camadas superiores. Esse tempo está de acordo com que se espera para uma rede 100 Mbit/s (essa é a velocidade da rede para qual os equipamentos da empresa são projetados), onde um quadro de tamanho mínimo leva 6700 ns para ser transferido, e, portanto, o LRE será capaz de tratar todos os quadros recebidos em tempo hábil.

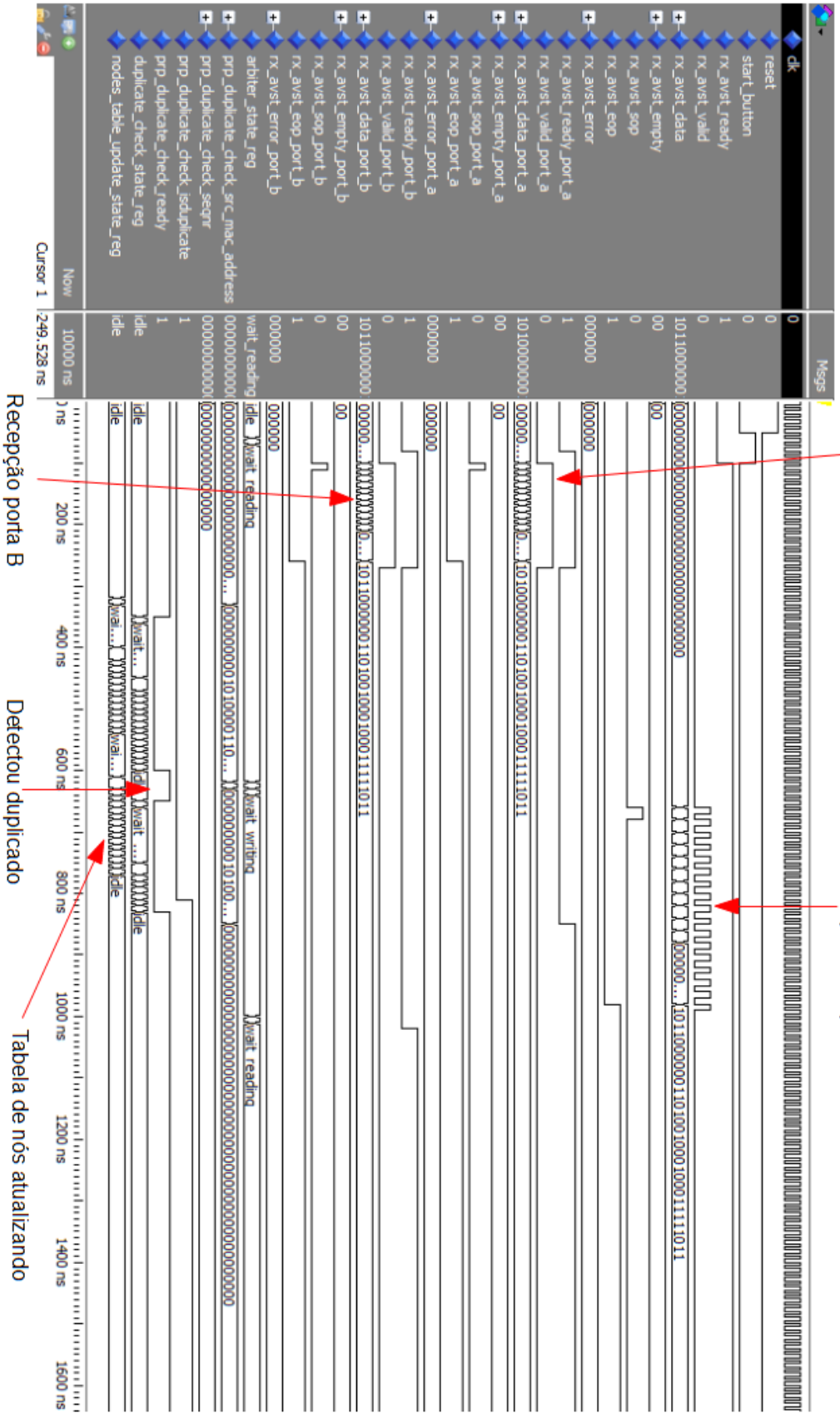


Figura 57: Simulação de recepção de quadro pelo componente LRE

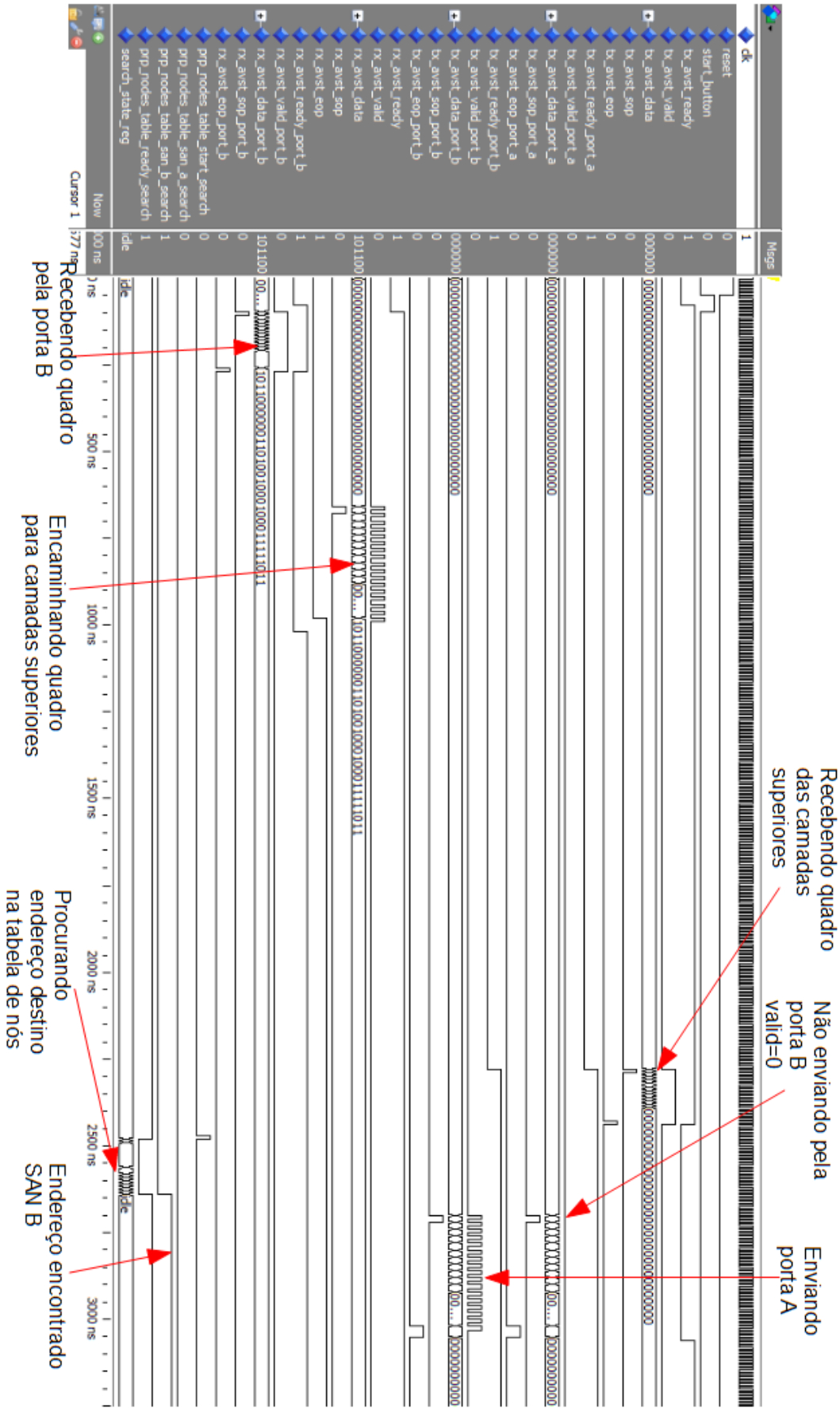


Figura 58: Simulação de transmissão de quadro pelo componente LRE

6.7: Recursos utilizados

Após a compilação do código VHDL, o software Quartus II disponibilizou um relatório de recursos utilizados do FPGA pelo circuito lógico do LRE. Os principais dados foram resumidos na Tabela 3.

Total de elementos lógicos	**4.375/114.480 (4%)**
Combinacional sem registradores	1945
Somente registradores	500
Combinacional com registradores	1930
Registradores totais	**2,430/117,053 (2%)**
Registradores lógicos dedicados	2,432/114,480 (2%)
Registradores de I/O	0/2,573 (0%)
LABs totais: parcialmente ou completamente utilizados	**324/7,155 (5%)**
M9Ks	**16/432 (4%)**
Total de bits de memória	**107,520/3,981,312 (3%)**
Total de bits utilizados na implementação da memória	**147,456/3,981,312 (4%)**
Média de interconexão utilizada (Total/H/V)	**1%/1%/1%**

Tabela 3: Recursos do FPGA utilizados pelo LRE

Na tabela de recursos utilizados, podemos verificar que o LRE utiliza uma porcentagem pequena dos elementos lógicos e bits de memória disponíveis, o que satisfaz o requisito não funcional de utilizar poucos recursos.

Essa pouca utilização de recursos é importante, pois o PRP será integrado nos novos produtos e o FPGA utilizado nesses produtos já possui bastante recurso utilizado para desempenhar outras funções.

As análises temporais realizadas pelo Quartus também demostraram que o circuito lógico do LRE respeita todas as restrições temporais para uma frequência de 100 MHz (frequência utilizada no projeto).

6.8: Conclusão

Quanto à recepção de quadros, os testes realizados no ModelSim permitiram validar o funcionamento do algoritmo de detecção de duplicados e da tabela de nós,

enquanto o teste realizado com os dois kits de desenvolvimento mostraram que algoritmo funciona em um equipamento real. Ainda falta testar o LRE em um ambiente PRP real, com duas LANs separadas e vários equipamentos se comunicando usando a norma, para verificar se a recepção funciona corretamente em uma rede real com tráfego maior.

Quanto à transmissão de quadros, a ferramenta Wireshark possibilitou verificar que a transmissão conforme a norma PRP funcionou de forma correta.

Os tempos calculados para transmissão e recepção de quadros, permitiram verificar que o LRE consegue tratar dados em uma velocidade compatível com uma rede 100 Mbit/s.

Por último, a análise temporal e de recursos utilizados, realizado pelo software Quartus, confirmou que o circuito digital está bem construído, respeitando os requisitos temporais e utilizando uma quantidade pequena de elementos lógicos e de bits de memória.

Capítulo 7: Conclusões e Perspectivas

Nesta monografia foi proposta e implementada uma arquitetura de circuito digital em FPGA que realizasse a operação do protocolo de redundância PRP. O FPGA foi programado utilizando a linguagem VHDL e ao longo do documento, foi apresentada a arquitetura do circuito digital do LRE (subcamada da camada de enlace de dados que opera a redundância PRP) e a descrição do seu funcionamento. O algoritmo de detecção de quadros duplicados utilizado e implementado no circuito do LRE foi um misto de tabela hash com buffer circular.

Testes realizados com o auxílio das ferramentas ModelSim, tcpdump e Wireshark, comprovaram que o circuito proposto e criado nessa monografia para o LRE, atende os requisitos estabelecidos para funcionamento do PRP: enviar e receber quadros conforme a norma, descartar quadros duplicados, construir tabela de nós corretamente e ser capaz de tratar os quadros recebidos em tempo hábil.

Apesar do módulo PRP criado funcionar corretamente, ele não está pronto para a inclusão nos produtos desenvolvidos para empresa. Ainda é necessária a criação de um device driver para que o LRE possa ser configurado pela aplicação e os dados da tabela de nós possam ser acessados pela aplicação. A norma PRP também prevê a criação de um agente SNMP para que os dados sobre redundância possam estar disponíveis na rede.

Com a inclusão do PRP nos produtos da empresa, espera-se que esses produtos ganhem mais um diferencial que os tornem mais atraentes para o mercado, conquistando novos clientes e ajudando no crescimento da empresa.

O projeto PRP/HSR ainda está longe de ser finalizado, pois após finalização do módulo PRP, restará ainda a criação do módulo HSR. Espera-se que o conhecimento adquirido com o trabalho no PRP possa acelerar o processo de desenvolvimento do dispositivo HSR, porque a maior parte do modo de operação e requisitos do HSR são semelhantes ao PRP.

A realização desse trabalho foi de fundamental importância para formação do estudante como engenheiro de controle e automação, ao permiti-lo ter uma

experiência de trabalho em ambiente profissional empresarial, possibilitando a interação com problemas reais de uma empresa e a busca por soluções.

Bibliografia:

[1] Mello, N. F. (2006). Automação Digital de Subestações de Energia Elétrica.

[2] Mackiewicz, R. E. (2006). Overview of IEC 61850 and Benefits.

[3] Mesmaeker, I. D., Rietmann, P., Brand, K.-P., & Reinhardt, P. (2005). Substation Automation based on IEC 61850. CIGRÉ. Cairo.

[4] Pereira, A. C., Zanirato, E., Abboud, R., Pellizzoni, R., & Caceres, D. (Maio de 2009). Sistemas de Proteção e Automação de Subestações de Distribuição e Industriais Usando a Norma IEC 61850. CIGRÉ. Puerto Iguazú.

[5] Castillo, C. R., Pozuelo, J. F., Goraj, M., Pereda, R., & Amezaga, A. (2012). Redundancy challenges on IEC 61850 systems and Migration Paths for IEC 61850 Substation Communication Networks. Paris.

[6] IEC 62439-3 International Electrotechnical Commission, Industrial Networks – Highly Available Automation Networks, Parallel Redundancy Protocol and High availability Seamless Redundancy.

[7] Reason Tecnologia S/A. (s.d.). Produtos. Acesso em 30 de Junho de 2012, disponível em Reason Tecnologia S/A: http://www.reason.com.br/pt/produtos.

[8] Stemmer, M. R. (2001). DAS 5331 - Sistemas Distribuídos e Redes de Computadores para Controle e Automação Industrial. UFSC, Departamento de Automação e Sistemas, Florianópolis.

[9] Cisco Systems (s.d.). Ethernet Technologies. Acesso em 25 de Junho de 2012, disponível em Cisco DocWiki:

http://docwiki.cisco.com/wiki/Ethernet_Technologies

[10] Wakerly, J. F. (s.d.). Digital Design: Principles & Practices (3ª ed.). Prentice Hall.

[11] Altera Corporation. (Maio de 2011). Avalon Interface Specifications. San Jose, CA.

[12] Jiang, X. (Fevereiro de 2009). High Availability Seamless Ring Protocol Implementation in FPGA. Tese de Mestrado, ETH, Zurich, Computer Engineering and Networks Laboratory.

[13] Chu, P. P. (2006). RTL Hardware Design Using VHDL: Coding for Efficiency, Portability and Scalability. New Jersey: Wiley-Interscience.