Universidade de Aveiro
2009
Departamento de Electrónica Telecomunicações e de Informática

Bruno Lopes Marques **Interligação de sistemas IP em redes SDH**

Universidade de Aveiro
2009
Departamento de Electrónica Telecomunicações e de Informática

Bruno Lopes Marques **Interligação de sistemas IP em redes SDH**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor Rui Luís Andrade Aguiar, Professor Auxiliar da Universidade de Aveiro, Departamento de Engenharia Electrónica e Telecomunicações

o júri

presidente

Prof. Dr. José Carlos da Silva Neves
professor catedrático da Universidade de Aveiro

Prof. Dr. Rui Luís Andrade Aguiar
professor auxiliar da Universidade de Aveiro

Prof. Dr. Carlos Alberto Batista Silva
professor auxiliar da Universidade do Minho

agradecimentos

Agradeço em particular à minha querida esposa pelo incentivo que me deu durante o período em que fiz esta dissertação. Agradeço também ao meu orientador e ao meu colega Miguel Osório por toda a ajuda prestada.

palavras-chave

IP, RPR, GMII, MII, SPI-3, HDLC, ATM, MPLS, Ethernet, SDH, VHDL, Verilog, SystemC, FPGA.

resumo

No presente trabalho propõe-se estudar as metodologias existentes de integração e implementação do protocolo IP sobre as diferentes tecnologias RPR, ATM, MPLS, Ethernet e HDLC suportadas na rede de transporte de dados SDH.

São apresentadas interfaces capazes de suportar um MAC da actual norma do RPR (IEEE 802.17) sobre um meio puramente Ethernet (IEEE 802.3) ou sobre o meio de transporte de dados SDH. Também serão apresentadas interfaces capazes de introduzir tráfego ATM sobre a rede SDH.

A primeira parte deste documento, correspondente ao primeiro e segundo capítulo, aborda as necessidades e identifica as interfaces suportadas por um sistema de transporte de dados desta natureza. São também apresentadas algumas soluções comercias existentes no mercado, desenvolvidos por alguns fabricantes e operadores assim como alguns exemplos de ambientes de utilização para este tipo de produtos.

A segunda parte corresponde ao terceiro capítulo, e é composta por uma abordagem ao trabalho desenvolvido pelo consorcio SIRAC que envolveu empresas e algumas entidades académicas e de investigação da área das telecomunicações na necessidade de estudo e de desenvolvimento de novas soluções tecnológicas. Foram feitas duas abordagens distintas em que primeiro se estudou uma tecnologia recente, que pretende afirmar-se no mercado (RPR), e por fim outra mais madura (ATM) que permitiu equacionar diferentes cenários de evolução da rede.

O quarto capítulo pertence à terceira parte deste documento, onde é descrito todo o desenvolvimento e testes elaborados às interfaces desenvolvidas no decorrer desta dissertação. É também apresentado um estudo referente à tecnologia de micro electrónica FPGA utilizada neste trabalho assim como alguns dados resultantes da síntese e "*place and route*" efectuados sobre o código HDL desenvolvido.

Por fim no quinto capítulo apresentam-se as respectivas conclusões.

São ainda apresentados alguns anexos onde se expõe alguns diagramas de algumas entidades lógicas da tecnologia FPGA, bem como código HDL de programação em FPGA para implementação de alguns circuitos de CRC (cyclic redundancy check) paralelos.

keywords

IP, RPR, GMII, MII, SPI-3, HDLC, ATM, MPLS, Ethernet, SDH, VHDL, Verilog, SystemC, FPGA.

Abstract

The present work is intended to study the existing methodologies for integration and implementation of the Internet Protocol IP with the RPR, ATM, MPLS, Ethernet and HDLC technologies, based on a SDH environment.

Some interfaces are presented to support a RPR (IEEE802.17) MAC on Ethernet or SDH technologies and also interfaces capable of introducing ATM traffic on SDH networks.

The first part of this document, corresponding to the first and second chapter, currently approaches the necessity of the interfaces used for a system of this nature, followed of a brief presentation of solutions and existing products in the market, developed by some manufacturers and operators.

The second part, corresponding to the third chapter, presents an approach of the work developed in the SIRAC consortium that involved some companies and academic and research institutions in order to study and develop new technology solutions including the RPR and ATM standard technologies.

The fourth chapter belong to the third part of this document, where it is described all the development and test made to the interfaces involved in this work. It also presents a FPGA technology study used in this work including synthesis and place and route analyses.

Finally in the fifth chapter the respective conclusions are presented.

Still some annexes are presented describing some FPGA unit logic diagrams and some HDL code for FPGA programming implementing a CRC (cyclic redundancy check) parallel algorithm

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

ANSI American national standard for information systems

ATM asynchronous transfer mode

AU-N administrative unit level n

B-ISDN broadband isdn

CIR committed information rate

CLB configurable logic block

CRC cyclic redundancy check

CSMA/CD carrier sense multiple access with collision detection

DWDM dense WDM

EIR excess information rate

FCS frame check sequence

FIFO first in first out

FPGA field programmable gate array

GFP generic framing procedure

GMII gigabit media independent interface

GRS GFP reconciliation sub-layer

HDL hardware description language

HDLC high-level data link control

HEC header error check

IEC international electro technical commission

IEEE institute of electrical and electronics engineers

IETF internet engineering task force

IP internet protocol

ISDN integrated services digital network

ISO international organization for standardization

ITU-T international telecommunication union

LAN local area network

LAPS link access procedure - SDH

LCP link control protocol

LRTT loop round trip time

LSB least significant bit

LUT lookup table

MAC medium access control

MAN metropolitan area network

MCFF multi choke fairness frame

MIB management information base

MII media independent interface

MLME MAC layer management entity

MPLS multiprotocol label switching

MSB most significant bit

MSPP multi service provision platform

OAM operations, administration, and maintenance

OC-N optical carrier level n

OIF optical Internetworking forum

OSI open systems interconnection

PCS physical coding sub-layer

PDH plesiochronous digital hierarchy

PHY physical layer

PMA physical medium attachment

PMD physical medium dependent

PPP point to point protocol

PRS-1 1 Gb/s 1Gb/s packet phy reconciliation sub-layer

PRS-10 10 Gb/s 10Gb/s packet phy reconciliation sub-layer

RFC request for comment

RPR resilient packet ring

RS reconciliation sub-layer

SDH synchronous digital hierarchy

SME station management entity

SNMP simple network management protocol

SONET synchronous optical network

SPI system packet interface

SPI-3 system packet interface level 3

SPI-4.1 system packet interface level 4 phase 1

SPI-4.2 system packet interface level 4 phase 2

STM-N synchronous transport module level n

STQ secondary transit queue

STS-N synchronous transport signal level n

TCP transmission control protocol

TDM time division multiplexing

VC virtual container

VC-N virtual container level n

VCI virtual channel identifier

VPI virtual path identifier

VPN virtual private network

WAN wide area network

WDM wavelength division multiplexing

WIS wan interface sub-layer

XAUI 10 gigabit attachment unit interface

XGMII 10 gigabit media independent interface

Termos e definições

agente: Uma entidade de gestão da rede utilizada para configurar uma estação e/ou colectar dados que descrevem a operação dessa estação.

anel aberto: Anel que foi cortado, portanto impedido de completar uma ligação em volta. Um anel aberto tem pelo menos uma estação fronteira detectada.

anel fechado: Anel intacto (sem cortes), que providencia um trajecto completo em todo o caminho à volta do anel. Um anel fechado não contém estações fronteira detectadas.

***bridge*:** Unidade funcional que liga duas ou mais redes na camada de Data Link do modelo OSI.

***broadcast*:** Acto de enviar uma trama endereçada a todas as estações numa rede.

camadas superiores: Conjunto de camadas protocolares acima da camada de *data-link*.

classe de serviço: Categorização do serviço do MAC em termos de limites de atraso, prioridade relativa, garantias de taxa de dados, ou características similares distintas.

cliente MAC: Entidade de camada que invoca a interface de serviço do MAC.

***fairness*:** Propriedade que, para uma qualquer ligação no anel, cada estação origem recebe uma proporção igual de capacidade elegível para *fairness*. Se todas as estações de origem tiverem pesos iguais, então as mesmas têm um acesso semelhante à capacidade disponível de todas as ligações.

flooding: Acto de transmitir uma trama tal que todas as estações no anel recebam essa trama uma vez.

keepalive: Troca de mensagens permitindo a verificação de que uma comunicação entre estações está activa.

layer management entity (LME): Entidade dentro de uma camada que executa a gestão local de uma camada. Fornece a informação sobre a camada, efectua controlo sobre a mesma, e indica a ocorrência de determinados eventos dentro da mesma.

management information base (MIB): Repositório de informação que descreve a operação de um equipamento específico de rede.

physical layer (PHY): Camada responsável por fazer a interface com o meio de transmissão.

ringlet: Ligação na qual o tráfego de dados circula unidireccional entre estações num anel composto por duas ou mais ligações.

sub-camada de adaptação: Sub-camada protocolar que tem como objectivo converter dados de um formato para outro.

sub-camada de reconciliação (RS): Sub-camada que providencia uma adaptação entre a interface de serviço PHY e a interface independente do meio da PHY.

sub-camada medium access control (MAC): Sub-camada que controla e medeia o acesso ao meio da rede.

topologia: Arranjo das ligações e estações que formam uma rede, conjuntamente com informação de atributos da estação.

1 Introdução

1.1 Necessidades de sistemas de transportes de dados

Nos últimos anos, tem havido uma influência cada vez mais acentuada por parte da Internet no dia a dia das pessoas, das empresas e da grande maioria dos organismos que compõem uma qualquer sociedade em geral. A tecnologia tem tido um papel impulsionador para a grande maioria dos países em geral, estando ao alcance quer dos países mais desenvolvidos, como também de alguns países menos desenvolvidos com economias ainda em fase de crescimento.

Os avanços verificados no domínio da Internet devem-se sobretudo ao trabalho elaborado pela comunidade científica internacional. Portugal foi sem duvida um país que aderiu claramente à Internet, datando de 1983 as primeiras experiências de ligação à Internet pelas nossas universidades [8].

Assim as empresas procuram cada vez mais redes capazes de suportar grandes transferências de dados, com bons índices de fiabilidade e disponibilidade, e actualmente tendo de ser capazes de assegurar serviços de dados voz e vídeo simultaneamente com significativa qualidade de serviço. É necessário por isso o desenvolvimento de tecnologias capazes de suportar todas estas necessidades actuais e futuras das sociedades em geral.

1.2 Enquadramento

Nos últimos anos a Internet espalhou-se a um ritmo extremamente elevado e a uma escala internacional mudando hábitos das empresas e da vida das pessoas. As tecnologias utilizadas nestas redes foram também evoluindo permitindo melhores desempenhos, eficiência e uma maior diversidade de serviços.

Nos finais dos anos 80 surgiram varias redes de alto débito, como por exemplo a FDDI (Fiber distributed data interface) uma MAN a 100Mbit/s, ESCON (Enterprise serial connection) que permitia a interligação entre computadores de grande porte ou com periféricos a 200Mbit/s, e as redes de SONET/SDH que fornecem ligações de comutação de circuitos extremo a extremo [9].

As redes SONET/SDH popularizadas nas redes de telecomunicações são possuidoras de um mecanismo eficiente para multiplexar ligações de baixo ritmo (ex. 155Mbit/s), de modo a obter ligações de ritmo mais elevado (ex. 10Gbits, 40 Gbit/s). O SONET/SDH também garante um elevado grau de fiabilidade e disponibilidade, entre 99.99% e 99.999%) permitindo a um operador obter uma indisponibilidade de apenas uma hora por ano, através de mecanismos rápidos de restauro do serviço no caso de avarias na rede. Contudo as redes SONET/SDH, tratando-se de redes de comutação de circuitos foram inicialmente projectadas e optimizadas para tráfego de voz. Tendo em consideração o aumento verificado ao nível de tráfego de pacotes IP existiu necessidade e espaço para o aparecimento de novas tecnologias capazes de preencher esta lacuna. É neste contexto que surgem tecnologias como o RPR (Resilient Packet Ring - IEEE

802.17) [1][4][26], ATM [22][27][28] e MPLS [25][27], que servem de suporte ao transporte de tráfego IP garantindo alguma qualidade de serviço. As tecnologias descritas são compatíveis com as actuais redes SONET/SDH usadas pela grande maioria dos operadores de telecomunicações.
Este trabalho enquadra-se neste paradigma de redes para pacotes, em particular nos aspectos do desenvolvimento de interfaces totalmente compatíveis com as tecnologias mais antigas SONET/SDH.

1.3 Objectivos da dissertação

No presente trabalho propõe-se estudar as metodologias existentes de integração e implementação do protocolo Internet IP sobre sistemas de transporte de dados SDH.
A tecnologia RPR aparece como um novo paradigma de rede de área metropolitana sobre uma arquitectura packet-based de próxima geração. Contudo terá de existir total compatibilidade entre as tecnologias SONET/SDH existentes para áreas metropolitanas bem como para as tecnologias de pacote Ethernet e IP.
Este trabalho pretende implementar um conjunto de interfaces, em tecnologia reconfigurável, capazes de usar as tecnologias RPR, ATM, Ethernet e HDLC para o correcto funcionamento de uma rede SDH. Foram desenvolvidas interfaces em linguagem SystemC [15][16][17] posteriormente convertidas em HDL (VHDL/Verilog) e sintetizadas numa FPGA da Xilinx. A implementação das interfaces pressupõe o conhecimento de alguns protocolos de telecomunicações que permitam o envio de tráfego IP através de sistemas de transporte de dados SDH como o RPR, ATM, MPLS, HDLC, GFP, LAPS e Ethernet. Para além disso, foi também explorada a possibilidade de uma rede totalmente orientada ao tráfego de pacote desenvolvendo para isso interfaces de sistema capazes de suportar simultaneamente as tecnologias RPR e Ethernet, para um outro sistema alternativo explorou-se a tecnologia ATM.

1.4 Estrutura da dissertação

Nos primeiros dois capítulos é feito um levantamento de algumas tecnologias existentes no mercado que permitem a inclusão de tráfego IP numa rede SDH com qualidade de serviço, focando as interfaces suportadas por um sistema de transporte de dados desta natureza. São também apresentadas algumas soluções comerciais existentes no mercado, desenvolvidas por alguns fabricantes e operadores assim como alguns exemplos de ambientes de utilização para este tipo de produtos.
O terceiro capítulo aborda o trabalho desenvolvido pelo consórcio SIRAC que envolveu empresas e algumas entidades académicas e de investigação da área das telecomunicações na necessidade de estudo e de desenvolvimento de novas soluções tecnológicas.
No quarto capítulo é feita uma descrição de todo o desenvolvimento e testes elaborados às interfaces desenvolvidas. É também apresentado um estudo sobre a arquitectura de uma FPGA da Xilinx utilizada neste trabalho.

Por fim no quinto capítulo apresentam-se as conclusões da tese.

1.5 Contribuições da tese

Ao longo desta dissertação de mestrado foram adquiridos conhecimentos sobre desenvolvimento e implementação de sistemas reconfiguráveis (FPGA). Permitiu-nos assim, conhecer diferentes arquitecturas de FPGAs existentes no mercado, a elaboração de código HDL, e também aprender algumas técnicas de place&route para que os sistemas desenvolvidos funcionem para as frequências máximas desejadas e área mínima ocupada. Foi também projectada e fabricada de raiz uma placa que permitiu o desenvolvimento e implementação das interfaces do MAC do RPR. O trabalho foi vasto no que diz respeito ao número de protocolos implementados como RPR, SDH, ATM, Ethernet, HDLC e interfaces associadas SPI, GMII e MII. Do trabalho realizado resultaram um conjunto de componentes virtuais para cada um protocolos referidos, assim como máquinas de estados, CRC, FIFOS etc. Estes componentes são facilmente reutilizáveis podendo ser adaptados a uma outra qualquer eventual implementação para um determinado projecto futuro. Desta dissertação de mestrado resultou também a publicação de um artigo [26].

2 Sistemas de transporte de dados

2.1 SDH

A tecnologia SDH (Syncronous Digital Hierarchy) [9][13][23][29] centra-se na camada física (definida pelo modelo OSI) servindo de suporte ao envio de informação através do meio físico. O meio físico escolhido para as redes SDH é composto por fibras ópticas, formando uma arquitectura de área metropolitana em anel. A tecnologia SDH surgiu para substituir a tecnologia PDH (Plesiochronous Digital Hierarchy) com o intuito de garantir o envio de grandes quantidades de tráfego de voz e permitindo ainda a interoperabilidade entre equipamentos de diferentes fornecedores.
Analogamente à tecnologia SDH desenvolvida na Europa surgiu também a tecnologia SONET que usa os mesmos conceitos de rede na América. A norma que define o SDH foi desenvolvida pelo International Telecommunication Union (ITU), G.707 [13] e mais tarde a extensão G.708.

A camada SDH fornece ligações de comutações de circuito extremo-a-extremo sendo capaz de multiplexar quer ligações de baixo débito (ex.: 155Mbit/s) como também de alto débito (ex.: 10Gbit/s, 40 Gbit/s). A tecnologia SDH possui um mecanismo de ponteiros muito eficiente que auxilia todo o sincronismo existente entre os diferentes nós da rede reduzindo também a necessidade de recurso a bufferização exaustiva no desenvolvimento dos equipamentos. Permite também extrair fluxos de ritmos mais baixos a partir de fluxos de ritmos mais elevados.
As redes SDH garantem uma disponibilidade entre os 99,99% e 99,999% o que equivale a uma indisponibilidade de uma hora por ano.

A informação em SDH é estruturada por tramas, que podem conter outros blocos de informação dentro (contentores virtuais). A estrutura de uma trama STM-1 é composta por 270 linhas e 9 colunas. Possui um overhead de 9 bytes e um payload de 261 bytes por linha. Cada trama STM-1 tem um tempo de propagação de 125µs, logo o débito é de 270x9x8/125 = 155,52Mbit/s. No caso de uma trama STM-4 termos uma trama com 270x4 = 1080 linhas e 9 colunas em que o tempo de propagação é também de 125µs o que resulta num débito de 622,08Mbit/s. A Figura 1 ilustra uma trama genérica STM-N:

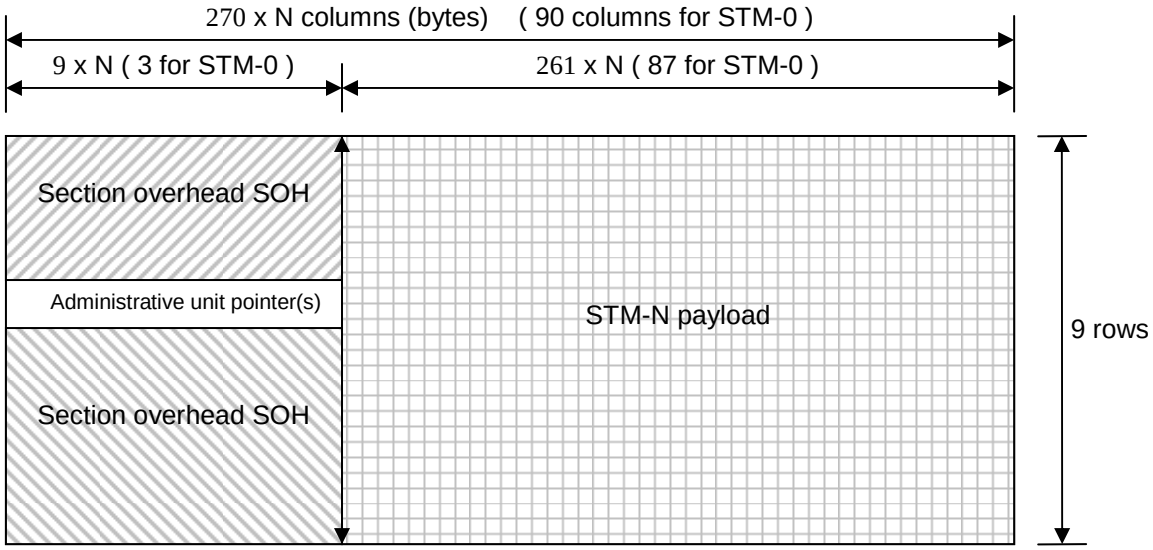


Figura 1 - estrutura da trama STM-N

A área representada pelas linhas 1 a 3 e 5 a 9 das colunas 1 a 9xN são dedicadas à secção de overhead (SOH). Esta secção inclui informação que permite ao equipamento reconstruir a trama, informação de manutenção, monitorização bem como outras operações funcionais.
Na linha 4 da coluna 1 a 9xN possui informação sobre ponteiros identificada como administrative unit pointers (AU-n). Este ponteiro permite um alinhamento dinâmico do contentor virtual VC-n dentro da trama AU-n.
Da linha 1 à 9 da coluna 9xN até 261xN temos a informação de payload da trama STM-N.

O parâmetro N pode assumir diferentes valores representando variados débitos como indicado na Tabela 1:

Synchronous digital hierarchy level	Hierarchical bit rate (kbit/s)
0	51 840
1	155 520
4	622 080
16	2 488 320
64	9 953 280
256	39 813 120

Tabela 1 - débitos e hierarquias SDH

A informação presente no payload pode ser estruturada de muitas maneiras permitindo multiplexar menos canais de informação com débitos elevados ou um maior número de canais com débitos

reduzidos. A estrutura VC (virtual container) é transportada dentro do payload da trama STM-N permitindo a flexibilidade de escolha entre número de canais e débito desejado a transportar na trama SDH.

VC type	VC bandwidth	VC payload bandwidth
VC-11	1 664 Kbit/s	1 600 Kbit/s
VC-12	2 240 Kbit/s	2 176 Kbit/s
VC-2	6 848 Kbit/s	6 784 Kbit/s
VC-3	48 960 Kbit/s	48 384 Kbit/s
VC-4	150 336 Kbit/s	149 760 Kbit/s
VC-4-4c	601 334 Kbit/s	599 040 Kbit/s
VC-4-16c	2 405 376 Kbit/s	2 396 160 Kbit/s
VC-4-64c	9 621 504 Kbit/s	9 584 640 Kbit/s
VC-4-256c	38 486 016 Kbit/s	38 338 560 Kbit/s

Tabela 2 - capacidade e tipos de VC

A Figura 2 é ilustrativa das inúmeras possibilidades existentes para definir a estrutura de multiplexagem SDH pretendida.

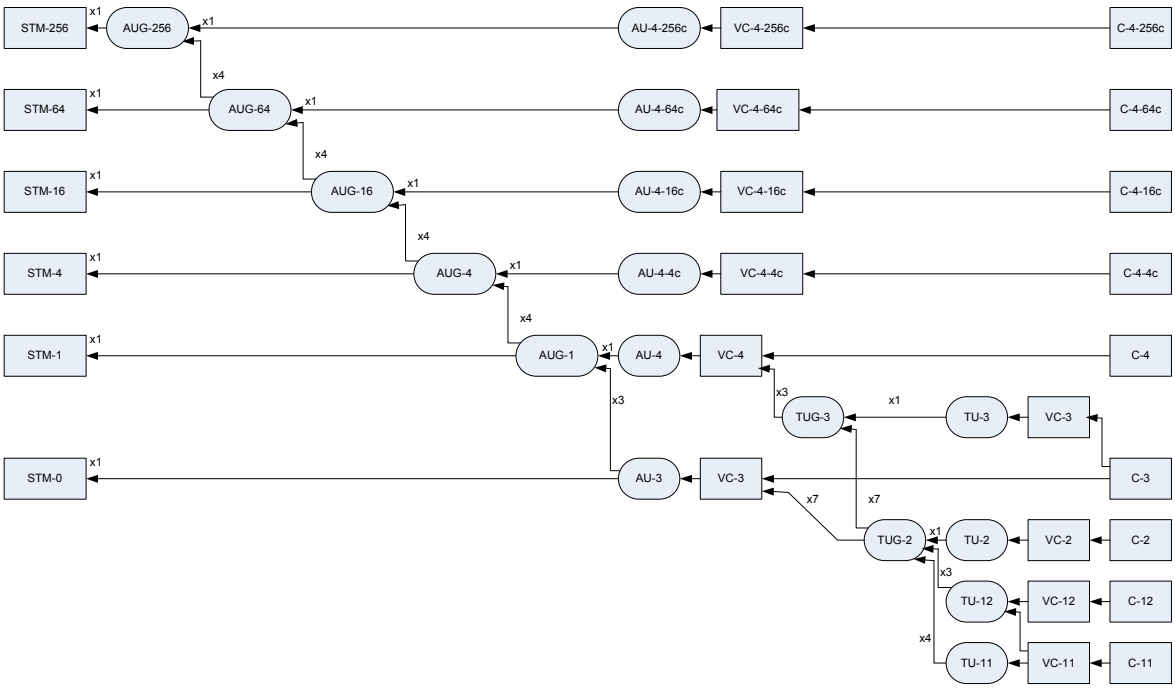


Figura 2 - Estrutura de multiplexagem SDH

2.2 RPR

O Resilient Packet Ring (RPR) [1][4] é uma tecnologia de transporte de dados entre estações interligadas entre si através de uma rede de área metropolitana numa arquitectura de duplo anel. O RPR tem sido alvo de estudos ao longo dos anos e é descrito pelo standard IEEE 802.17 [4]. Esta tecnologia de transporte pode servir como alicerce para um vasto leque de serviços multimédia emergentes no mercado das telecomunicações nos dias de hoje baseados em IP.

A tecnologia RPR tem para oferecer um misto das tecnologias Ethernet e SONET/SDH, suportando algumas potencialidades das tecnologias SDH (como 50ms para protecção), e simultaneamente um transporte de dados orientado ao pacote, que consegue obter elevados ganhos, através de uma multiplexagem estatística, utilizando da melhor forma possível toda a largura de banda disponível (incluindo a de protecção). A arquitectura de uma rede RPR é baseada numa configuração em anel-duplo tipo BLSR/2 que efectivamente utiliza a largura de banda de ambos os anéis o que resulta num consequente aumento da largura de banda que o operador passa a dispor. Cada anel é composto por ligações com fluxos de dados orientados no mesmo sentido. Os anéis são identificados como ringlet0 e ringlet1 como ilustra a Figura 3:

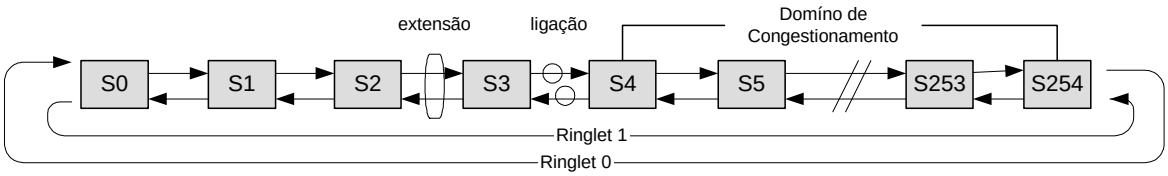


Figura 3 - Estrutura em anel duplo de uma rede RPR

O RPR permite uma utilização óptima e justa da largura de banda disponível através de algoritmos de justiça (fairness), e é capaz de fazer a distinção entre diferentes tipos de tráfego e possui alguns mecanismos de reutilização espacial. Suporta transferências de dados Unicast, Multicast e Broadcast e permite várias qualidades de serviço, através de protocolos de controlo de fluxo por qualidade de serviço que regulam o tráfego inserido pelo cliente. Suporta a descoberta automática da topologia, a inicialização dos parâmetros opcionais e a divulgação das capacidades da estação, o que permite que os sistemas se tornem operacionais sem intervenção manual.
Resumindo a tecnologia RPR combina algumas das mais valias de uma rede Ethernet como a simplicidade, familiaridade e custos competitivos, com as mais valias das redes SONET/SDH como um suporte eficiente para uma topologia em anel e uma boa recuperação no caso de ocorrência de falhas.

2.2.1. Modelo de camadas

A Figura 4 ilustra o modelo de camadas que suportam o RPR e como este se relaciona com o modelo OSI.

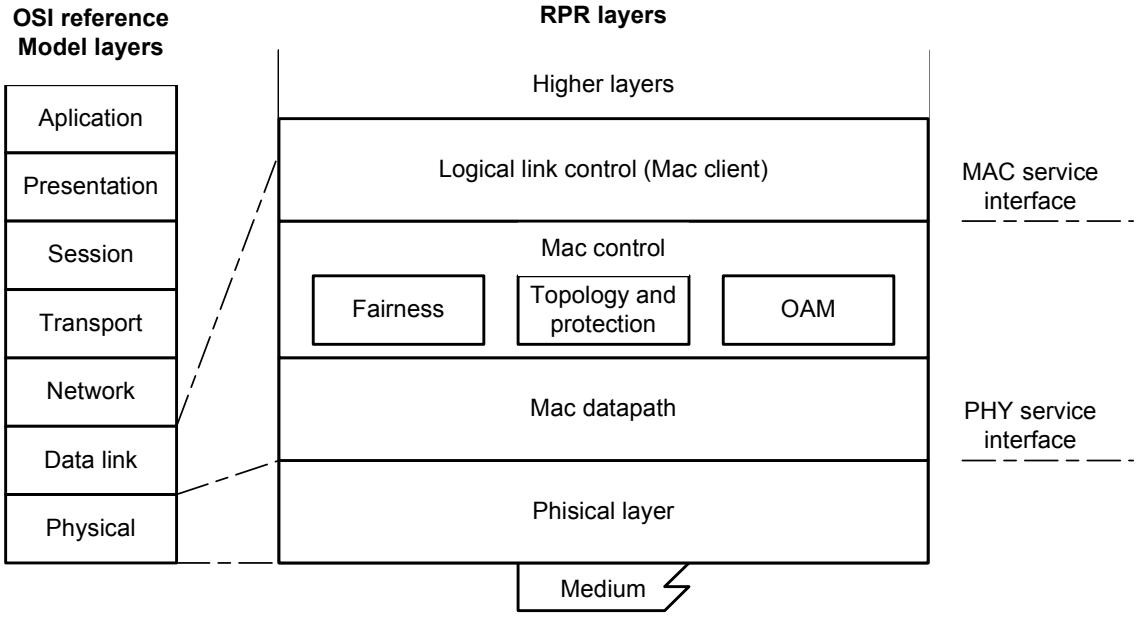


Figura 4 - modelo de camadas RPR

A sub-camada MAC de controlo representada na Figura 4 está descrita na cláusula 6.6. da norma 802.17. Controla a sub-camada de datapath, mantém o estado do MAC, coordena com as diferentes sub-camadas de controlo de outros MACs e controla a transferência de dados entre o MAC e o seu cliente.

A sub-camada MAC de datapath está descrita na cláusula 7 da norma 802.17 e disponibiliza funções de transferências de dados para cada um dos anéis.

A interface de serviço do MAC (presente no capítulo 6.4), permite quer a transferência de dados entre o MAC e os diversos clientes como também a transferência de informação local de controlo entre o MAC e o MAC do cliente. Para isso existem diversas primitivas lógicas de serviço que regulam a transferência de informação de dados e controlo entre as diferentes entidades.

A interface de serviço do PHY é utilizada pelo MAC para transmitir e receber tramas da linha. Existem sub-camadas de reconciliação distintas que especificam o mapeamento entre camadas físicas específicas e algumas interfaces independentes do meio (MIIs).

A norma 802.17 inclui a definição de uma sub-camada de reconciliação para cada um dos PHYs mais frequentes e permite simultaneamente o uso de outras sub-camadas de reconciliação que cumpram os requisitos da norma. Compreende o uso de PacketPHY's e também de SONET/SDH PHY's mantendo a interoperabilidade com sistemas já existentes. No decorrer desta dissertação

de mestrado será abordado a implementação de interfaces com o MAC RPR para cada um destes PHY's (ver ponto 4.3 e 4.4).

2.2.2 Interfaces disponíveis

A interface de serviço do MAC permite a transferência de dados e de informação de controlo entre o MAC e os diversos clientes.
Os serviços de uma camada ou sub-camada são o conjunto de possibilidades que estas oferecem às camadas ou sub-camadas da camada acima. Os serviços especificados na norma são descritos por primitivas de serviço abstractos e por parâmetros que caracterizam cada serviço. Esta definição de serviço é independente de qualquer implementação em particular.
As interfaces do MAC do RPR com o PHY estão definidas na cláusula 8 da norma 802.17. São aí descritas duas famílias distintas de sub-camadas de reconciliação (RS) opcionais podendo ser PacketPHYs ou SONET/SDH PHYs. Para assegurar a interoperabilidade do sistema deve ser adoptado pelo menos uma das interfaces descritas pela norma 802.17, estando o uso de interfaces com outros PHYs fora do âmbito da mesma.
Os PacketPHYs operam a 1Gb/s ou 10Gb/s de uma forma similar ao definido na norma IEEE Std 802.3-2002 e IEEE Std 802.3ae-2002.
Para o SONET/SDH PHY, a sub-camada de reconciliação (RS) disponibiliza interfaces para sub-camadas de adaptação que suportam generic frame procedure (GFP), byte-synchronous high-level data link control (HDLC)-like framing, ou link Access procedure-SDH (LAPS) framing para as redes SONET/SDH e PHYs operando desde 155Mb/s até 10Gb/s ou superior. A normalização GFP fornece um mecanismo genérico para adaptar tráfego de dados de clientes de camadas superiores sobre uma rede de transporte, usando uma variação do mecanismo de delineação de trama baseado no HEC definido do ATM (ITU-T Rec. I.432.1). O byte-synchronous HDLC-like framing para tramas do tipo RPR deverá ser realizado de acordo com a IETF RFC 1662 utilizando byte-stuffed framing, considerando que as tramas PPP sejam interpretadas como tramas do tipo RPR. Utiliza uma sequência de flags com valor 7E para sinalizar o início e fim de trama bem como uma flag de control escape, usada para impedir o reaparecimento do valor 7E no meio dos dados.
O LAPS consiste num modelo de encapsulamento de tramas Ethernet sobre um meio SONET/SDH, que inclui alguns mecanismo da camada de ligação de dados para o transporte de pacotes IP sobre redes SDH. O LAPS permite obter um serviço não orientado à conexão sobre um meio SONET/SDH. Permite também o encapsulamento de IPv6, IPv4, PPP, e de outros protocolos situados nas camadas superiores do modelo OSI.

Assim sendo as interfaces do MAC do RPR com o PHY deverão suportar a independência com a camada física, ritmos de transmissão de dados para o RPR de 155Mb/s até 10Gb/s, operar em full-duplex entre estações adjacentes no anel, permitir que o MAC do RPR opere sobre camadas

físicas síncronas e assíncronas e disponibilizar interfaces PHY que não afectem de maneira alguma o mecanismo de protecção do RPR.

A Figura 5 demonstra a relação entre o MAC RPR, sub-camada de reconciliação, sub-camada de adaptação e camada física.

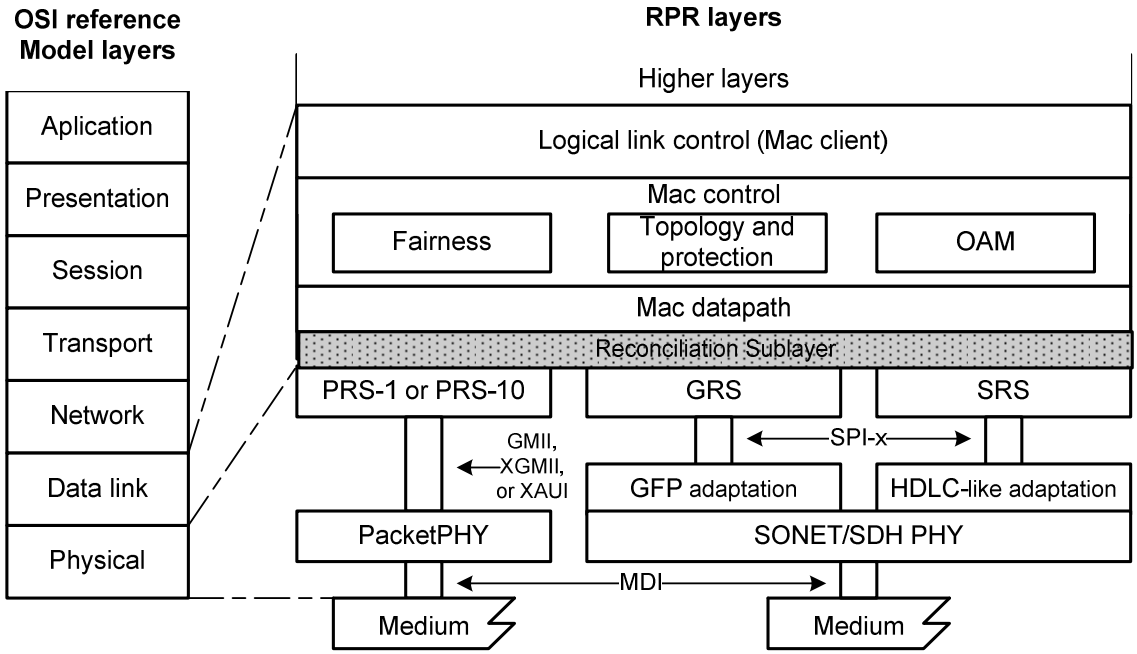


Figura 5 - relação entre o RPR, RS, PHY e o modelo ISO/IEC OSI

2.2.2.1 Interface de serviço do MAC com o cliente

As seguintes quatro primitivas de serviço são definidas para as interfaces de clientes non-bridge devendo por isso ser implementadas. O anexo F da norma 802.17 explica alguns requerimentos extra para clientes Bridge. O objectivo deste anexo consiste em demonstrar que a definição do MAC do RPR é compatível com Bridging transparente e VLAN Bridging, como descrito no IEEE 802.1D-2004 e IEEE802.1Q-2003.

- MA_DATA.request
- MA_DATA.indication
- MA_CONTROL.request
- MA_CONTROL.indication

As primitivas são ilustradas na Figura 6:

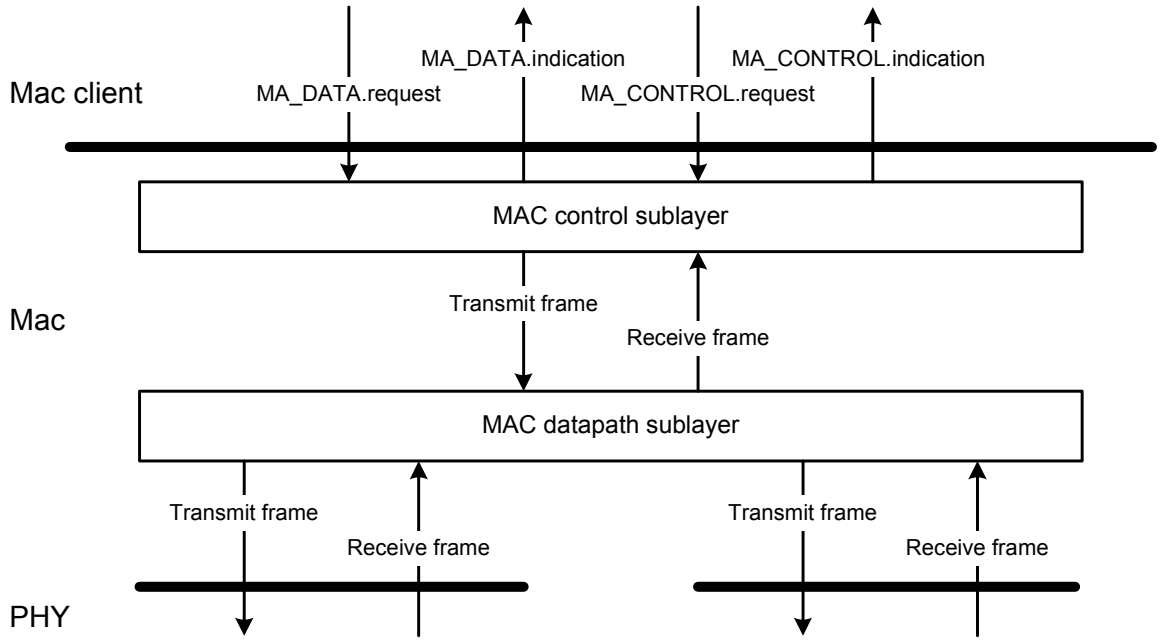


Figura 6 – modelo da interface de serviço do MAC

MA_DATA.request

A primitiva MA_DATA.request define a transferência de dados da entidade representativa do cliente MAC para uma entidade, ou para múltiplas entidades (no caso de endereços de grupo).

A primitiva MA_DATA.request será invocada pela entidade cliente sempre que houver dados para transmitir. A recepção da primitiva MA_DATA.request faz com que a entidade MAC crie uma trama de dados de formato básico ou estendido, preenchendo os campos que compõem a trama através da informação disponibilizada pela primitiva. Em seguida a trama correctamente formada é passada para as máquinas de estados de transmissão de modo a ser transferida para as entidades de sub-camada da entidade MAC. O MAC não reflecte as tramas de novo para o cliente. Se o cliente assinalar uma primitiva MA_DATA.request com um valor de destination_address igual ao seu endereço MAC local, o pedido será rejeitado.

O MAC não aceita pedidos de um cliente quando a indicação de envio apropriada não estiver presente. Os detalhes desta interacção são específicos de cada implementação.

As combinações de source_address e source_address_extended são sumariadas na Tabela 3.

source_address_extended	source_address	Resulting frame actions
not provided	not provided	// basic data frame ; sa = myMACAddress;
	= myMACAddress	// basic data frame ; sa = myMACAddress;

	≠ myMACAddress	// extended data frame ; sa = myMACAddress;
		saExtended = source_address ;
		da = destination_address ;
		daExtended = destination_address;

Provided	--	// extended data frame ; sa = myMACAddress;
		saExtended = source_address_extended;
		da = destination_address;
		daExtended = destination_address_extended;

Tabela 3 - Combinação de valores para o source address

A semântica que define a primitiva é a seguinte:

MA_DATA.request
(
destination_address,
source_address, // optional
MAC_service_data_unit,
frame_check_sequence, // optional
service_class,
ringlet_id, // optional
MAC_protection, // optional
mark_fe, // optional
strict_order, // optional
destination_address_extended, // optional
source_address_extended, // optional
flooding_form // optional
)

Os parâmetros da primitiva são os seguintes:

destination_address (endereço de destino)
Este parâmetro especifica um qualquer endereço MAC individual ou de grupo, diferente do endereço de MAC local, a ser colocado no campo da (destination MAC address) da trama RPR a ser transmitida.

source_address (endereço fonte)

O parâmetro source_address, no caso de existir e ser diferente do endereço de MAC da estação, representa um endereço MAC individual que será usado no campo saExtended (source MAC address, extended) da trama RPR a ser transmitida, e para determinar se a trama RPR se apresenta num formato de dados básico ou estendido (ponto 2.2.6). O parâmetro source_address será ignorado no caso do parâmetro source_address_extended for disponibilizado.

MAC_service_data_unit

Este parâmetro disponibiliza o payload da trama. Mais especificamente o parâmetro MAC_service_data_unit especifica os dados do MAC service a serem transmitidos pela entidade de sub-camada do MAC no campo service_data_unit da trama RPR transmitida. Existe também informação suficiente associada ao MAC_service_data_unit para que a entidade de sub-camada do MAC determine o comprimento dos dados transmitidos.

frame_check_sequence

Disponibiliza o valor do campo de controlo de erros FCS da trama RPR transmitida. Se o valor do parâmetro frame_check_sequence for omitido, o MAC calculará um valor correcto para o FCS da trama.

service_class

Indica a classe de serviço a ser utilizada pelo cliente MAC (ponto 2.2.6) como descrito na Tabela 4:

Service_class value	Corresponding sc value	Description
SC_CLASS_A	CLASS_A0 or CLASS_A1	classA service class
SC_CLASS_B	CLASS_B	classB service class
SC_CLASS_C	CLASS_C	classC service class

Tabela 4- Valores de classe de serviço

A classe de serviço é utilizada pelo MAC com o intuito de seleccionar um valor apropriado para o campo sc da trama RPR indicando qual o tratamento devido que a trama deverá sofrer pela entidade MAC.

ringlet_id

Indica qual o anel escolhido pelo cliente como descrito na Tabela 5.

Ringlet_id value	Corresponding ri value	Description
RI_0	RINGLET_0	ringlet0 preferred
RI_1	RINGLET_1	ringlet1 preferred
RI_DEFAULT	--	default ringlet
(null)		

Tabela 5 - Valores para o ringlet_id

A selecção do anel é importante pois permitirá preencher adequadamente o valor do campo ri (ringlet identifier) da trama transmitida antes de uma qualquer ocorrência do domínio dos mecanismos de protecção.

MAC_protection

Indica se o MAC atribui protecção à trama ou não.

TRUE—Trama com protecção.

FALSE— Trama sem protecção.

(null)—O valor por defeito está a TRUE.

mark_fe

Este parâmetro indica se a trama é do tipo fairness eligible devendo ser tratada como tal ou não consoante a informação presente. A entidade de MAC irá colocar o valor apropriado no campo fe (fairness eligible). O parâmetro mark_fe é valido apenas para pedidos que incluem um pedido de classB para a classe de serviço e é ignorado para todos os outros pedidos. Isto acontece de modo a permitir que um cliente escolha quais dos dados marcados como classB apresentados ao MAC são caracterizados como fairness eligible, assim como quando manipulando múltiplos fluxos de tráfego de diferentes clientes, cada um com o sua alocação de largura de banda.

TRUE—O MAC força que a trama (classB) seja fairness eligible.

FALSE—O MAC determina se a trama é fairness eligible baseado no sendB.

(null)—O valor por defeito é FALSE.

strict_order

Serve para marcar e tratar uma trama de dados como strict ou relaxed, como explicado na norma 802.17, sendo utilizado pela entidade de MAC para seleccionar um valor para o campo so (strict order). O parâmetro strict_order é ignorado, sendo colocado a 1 em todas as tramas, se a variável forceStict estiver a TRUE.

TRUE—O so é colocado a 1 pelo MAC.

FALSE— O so é colocado a 0 pelo MAC.

(null)—O valor por defeito é TRUE.

destination_address_extended

Este parâmetro especifica um qualquer endereço MAC individual ou de grupo, diferente do endereço de MAC local, a ser usado de modo a compor o campo daExtended da trama RPR a ser transmitida. Se o parâmetro destination_address_extended for disponibilizado o MAC utiliza o formato extended. Sempre que o parâmetro destination_address_extended for disponibilizado, também o parâmetro source_address_extended terá de o ser, bem como o parâmetro flooding_form. Os efeitos da utilização deste parâmetro em bridging são apresentados no anexo F.1.7. da norma 802.17.

source_address_extended

O parâmetro source_address representa um endereço MAC individual que irá compor o campo saExtended (source MAC address, extended) da trama RPR a ser transmitida. Se o parâmetro source_address_extended for disponibilizado o MAC utiliza o formato extended. Sempre que o parâmetro source_address_extended for disponibilizado, o parâmetro source_address é ignorado, o parâmetro destination_address_extended também terá de ser disponibilizado bem como o parâmetro flooding_form. Os efeitos da utilização deste parâmetro em bridging são apresentados no anexo F.1.7. da norma 802.17.

flooding_form

Assinala que se utilize um determinado tipo de flooding form, como especificado na Tabela 6, indicando o valor que deverá ser colocado no campo fi (flooding indication) como descrito na norma 802.17.

Flooding_form value	Corresponding fi value	Description
FLOOD_NONE	FI_NONE	no flood
FLOOD_UNIDIR	FI_UNIDIR	unidirectional flood
FLOOD_BIDIR	FI_BIDIR	bidirectional flood
(null)	default flooding behavior selected by the MAC	

Tabela 6 - Valores para o flooding_form

O valor de FI_NONE serve para indicar ao MAC para que a trama seja not flood. Se o valor for diferente de FI_NONE indica ao MAC que o encaminhamento da trama seja feito para além do ringlet selection, utilizando o flooding form escolhido para a trama. Se o parâmetro flooding_form for omitido, o MAC utilizará a configuração por defeito de flooding. O parâmetro flooding_form terá de ser disponibilizado se os parâmetros destination_address_extended e source_address_extended forem também disponibilizados. Os efeitos da utilização deste parâmetro em bridging são apresentados no anexo F.1.7. da norma 802.17.

MA_DATA.indication

A primitiva MA_DATA.indication define a transferência de dados da entidade de sub-camada do MAC para a entidade de cliente do MAC.

A primitiva MA_DATA.indication é passada a partir da entidade da sub-camada do MAC, através a sub-camada de MAC control, para a entidade de cliente do MAC para assinalar a chegada de uma trama destinada à entidade da sub-camada do seu MAC local. Este tipo de tramas é reportado apenas se tiverem correctamente formadas e o seu destination address apontar para a entidade de MAC local (endereço de estação local, endereço de grupo, ou flooded). A manipulação de uma trama com o campo reception_status diferente de RECEIVE_OK não é coberta pela norma 802.17. O efeito da recepção da primitiva MA_DATA.indication pelo cliente MAC não é especificada. O MAC não reflecte as tramas de volta para o cliente. Se o MAC receber uma trama com um valor para o campo sa igual ao endereço do seu MAC local a primitiva MA_DATA.indication não irá ser enviada para o seu cliente original.

A semântica que define a primitiva é a seguinte:

MA_DATA.indication
(
destination_address,
source_address,
MAC_service_data_unit,

frame_check_sequence,
reception_status,
service_class,
ringlet_id,
fairness_eligible,
strict_order,
extended_frame,
destination_address_extended,
source_address_extended
)

Em seguida vamos definir os parâmetros da primitiva:

destination_address
Indica o valor do campo da (destination address) da trama RPR.

source_address
Indica o valor do campo sa (source address) da trama RPR.

MAC_service_data_unit
Disponibiliza o MAC service data unit como especificado no campo serviceDataUnit da trama. Existe informação suficiente associada ao MAC_service_data_unit para a entidade de cliente do MAC determinar o comprimento dos dados.

frame_check_sequence
Indica o valor do campo FCS no cabeçalho da trama.

reception_status
Indica à entidade de cliente do MAC o estado da trama recebida como descrito na Tabela 7.

Reception_status value	Description
RECEIVE_OK	The frame had no errors.
RECEIVE_FCS_ERROR	The frame had an FCS error (either invalid FCS or stomped FCS).

Tabela 7 - Valores do reception_status

O campo reception_status pode assumir o valor de RECEIVE_FCS_ERROR apenas se a variável copyBadFcs possuir a valor TRUE.

service_class
Indica a classe de serviço da trama enviada usando para isso os valores fornecidos na Tabela 4.

ringlet_id
Indicação para o bit ri (ringlet identifier) do cabeçalho da trama RPR utilizando os valores fornecidos pela Tabela 5.

fairness_eligible
Indicação para o bit fe (fairness eligible) do cabeçalho da trama RPR.

strict_order
Indicação para o bit so (strict order) do cabeçalho da trama RPR.

extended_frame
Indicação para o bit ef (extended frame) do cabeçalho da trama RPR.

destination_address_extended
Se o campo extended_frame possuir o valor TRUE indica o valor do campo daExtended da trama.

source_address_extended
Se o campo extended_frame possuir o valor TRUE indica o valor do campo saExtended da trama.

MA_CONTROL.request

A primitiva MA_CONTROL.request define a transferência de comandos de controlo da entidade de cliente do MAC para a entidade de sub-camada de controlo do MAC local.
É gerada pelo cliente MAC sempre que este necessite de fazer uso dos serviços da entidade de sub-camada do MAC control.

Esta primitiva não fornece um meio directo para que um cliente transmita uma trama de controlo do MAC local para um dos anéis. Contudo as tramas de controlo do MAC podem ser geradas de uma forma indirecta dando resposta a um pedido valido. Se uma entidade de cliente fizer um pedido de informação de status ou de parâmetros de configuração do MAC ou algo do género, este direccionará ou irá buscar a informação requerida ao MLME (MAC Layer Management Entity). Por exemplo informação de topologia de status da base de dados, single-queue ou dual-queue, switch manual e switch forçado.

A semântica que define a primitiva é a seguinte:

MA_CONTROL.request
(
opcode,
request_operand_list
)

O parâmetro opcode é descrito na Tabela 8 e descreve a operação de controlo a pedido do cliente MAC.

opcode	Significado	Operadores	Especificado em
OAM_ECHO_REQ	Pedido de transmissão de uma trama de echo request	echo request payload e parâmetros	12.4.1
OAM_FLUSH_REQ	Pedido de transmissão de uma trama flush	flush payload e parâmetros	12.4.3
OAM_ORG_REQ	Pedido de transmissão de uma trama OAM	payload especifico de organização e parâmetros	12.4.5

Tabela 8 - Valores para control opcodes

MA_CONTROL.indication

A primitiva MA_CONTROL.indication define a transferência de indicações do estado de controlo da sub-camada de controlo do MAC para o cliente MAC. É gerada pela sub-camada de controlo do MAC dentro de um conjunto de condições específicas para cada operação de controlo do MAC.

A semântica que define a primitiva é a seguinte:

MA_CONTROL.indication
(
opcode,
indication_operand_list
)
Os elementos do parâmetro indication_operand_list são especificados para cada um dos parâmetros de opcode possíveis na Tabela 9.

opcode value	Meaning	Operands	Specified in
OAM_ECHO_IND	Recepção de uma trama de echo reply	echo payload e parâmetros	12.4.2.
OAM_FLUSH_IND	Recepção de uma trama flush	flush payload e parâmetros	12.4.4
OAM_ORG_IND	Recepção de uma trama OAM	payload especifico de organização e parâmetros	12.4.6
TOPO_CHANGE	Alteração da topologia	topologia e base de dados	11.5
PROT_CHANGE	Alteração da protecção	topologia e base de dados	11.5
SENDA	sendA change	Verdadeiro/falso,ringlet_id	7.5.2
SENDB	sendB change	Verdadeiro/falso,ringlet_id	7.5.2
SENDC	sendC change	hopsToCongestion, ringlet_id	7.5.2
SINGLE_CHOKE_IND	Fim de agingInterval (quando singleChokeIndOption for verdadeira)	allowedRate, allowedRateCongested, hopsToCongestion, ringlet_id	10.4.2
MULTI_CHOKE_IND	Recepção de trama multi-choke fairness (MCFF)	frame.fairRate, frame.saCompact, frame.ttl, frame.ri	10.4.8

Tabela 9 - Valores para Control indication opcodes

2.2.2.2 Interface de serviço da camada física do MAC

A interface de serviço permite a transferência de dados do MAC entre a sub-camada de reconciliação (RS) e os PHY's através de primitivas lógicas de serviço.
As primitivas são as seguintes:

- PHY_DATA.request
- PHY_DATA.indication
- PHY_LINK_STATUS.indication
- PHY_READY.indication

PHY_DATA.request (OUTPUT_FRAME,LENGTH)

Esta primitiva define a transferência de dados do MAC para a sub-camada de reconciliação. Aceita como parâmetros um campo (OUTPUT_FRAME) com o trama RPR transferida do MAC para a sub-camada de reconciliação e um campo (LENGTH) com informação do comprimento da trama.
O MAC inicia a primitiva PHY_DATA.request para pedir a transmissão de uma trama. O mapeamento da primitiva relativamente a interfaces eléctricas especificas esta descrito nos anexos B e C da norma 802.17.

PHY_DATA.indication (INPUT_FRAME,STATUS,LENGTH)

Define a transferência de dados da sub-camada de reconciliação para o MAC. O campo INPUT_FRAME diz respeito a uma trama RPR transferida da sub-camada de reconciliação para o MAC. STATUS contém indicação se a trama foi recebida com erros e LENGHT possui informação do comprimento da trama. O mapeamento da primitiva relativamente a interfaces eléctricas especificas esta descrito nos anexos B e C da norma 802.17.

PHY_LINK_STATUS.indication (LINK_STATUS)

Esta primitiva possui informação relativa à ligação física entre o PHY e o MAC quando detectada pelo PHY. Deverá ser gerada pela sub-camada de reconciliação sempre que o parâmetro LINK_STATUS seja alterado.
O parâmetro LINK_STATUS poderá assumir um dos seguintes valores:

- NO_DEFECT - Não foi detectada qualquer degradação ou falha na ligação
- SIGNAL_FAIL - Foi detectada uma condição de falha no link
- SIGNAL_DEGRADE - Foi detectada uma condição de degradação de sinal do link
- DEGRADE_FAIL - Ambas as condições de falha e degradação do link foram detectadas

Nem todas as sub-camadas de reconciliação suportam a condição de SIGNAL_DEGRADE. No caso do SONET/SDH por exemplo, as sub-camadas de reconciliação suportam SIGNAL_DEGRADE, enquanto que nos PacketPHY poderão não suportar.
O mapeamento da primitiva relativamente a interfaces eléctricas especificas esta descrito nos anexos B e C da norma 802.17.

PHY_READY.indication (READY_STATUS)

Indica se a sub-camada de reconciliação está apta para receber uma nova trama do MAC. Os valores possíveis para o parâmetro READY_STATUS são READY e NOT_READY.
O mapeamento da primitiva relativamente a interfaces eléctricas especificas esta descrito nos anexos B e C da norma 802.17.

2.2.3 PacketPHY sub-camada de reconciliação

A norma 802.17 compreende duas sub-camadas de reconciliação possíveis para a interface com o PHY, a 1Gb/s PacketPHY reconciliation sublayer (PRS-1) e a 10Gb/s PacketPHY reconciliation sublayer (PRS-10). A primeira possibilita a interface com PacketPHYs a 1Gb/s e a segunda com PacketPHYs a 10Gb/s.
A 1 Gb/s PacketPHY reconciliation sublayer (PRS-1) faz o mapeamento das primitivas do MAC physical layer service com a gigabit media independent interface (GMII) especificada na cláusula 35 do IEEE Std 802.3-2002. Embora seja uma interface opcional, a GMII é utilizada como base para a descrição da PRS-1. Se for pretendido uma implementação que não a GMII, esta deve ter um funcionamento como se da interface GMII se tratasse.
A 10 Gb/s PacketPHY reconciliation sublayer (PRS-10) faz o mapeamento das primitivas do MAC physical layer service com a 10 gigabit media independent interface (XGMII) especificada na cláusula 46 do IEEE Std 802.3-2002. Alternativamente, uma extensão da XGMII (XGXS) pode ser utilizada de modo a disponibilizar a 10 gigabit attachment unit interface (XAUI) como definido na cláusula 47 do IEEE Std 802.3-2002. A XGMII e XAUI são interfaces opcionais. A XGMII é usada como base para a descrição da PRS-10. De uma forma semelhante ao que ocorre na PRS-1, no caso de se optar por não utilizar a XGMII ou XAUI, a interface deve operar de um modo funcional como se as mesmas estivessem implementadas. A PRS-1 e a PRS-10 estão definidas em detalhe no anexo B da norma 802.17.

Ambos os PacketPHYs são constituídos pelas camadas PCS (physical coding sublayer), PMA (physical medium attachment sublayer) e PMD (physical medium dependent sublayer).
As camadas PCS e PMA no caso do 1Gb/s PacketPHY, são definidas pela cláusula 36 da norma IEEE Std 802.3-2002 e a camada PMD pela cláusula 38 e 39 da norma IEEE Std 802.3-2002.
Para o 10Gb/s PacketPHY as camadas PCS e PMA são definidas pela cláusula 48 e 49 da norma IEEE Std 802.3-2002 para diferentes tipos de camadas PMD. No 10 Gb/s PacketPHY pode também opcionalmente ser incluída uma camada WIS (WAN interface sublayer) bem como, XAUI (10 gigabit attachment unit interface) e camadas XGMII (10 gigabit media independent interface).

Para os dois PacketPHYs descritos anteriormente o tamanho mínimo permitido para uma trama é de apenas 16 bytes e o máximo de 9216 bytes.

2.2.4. SONET/SDH sub-camada de reconciliação

Assim como nos PacketPHYs, a SONET/SDH sub-camada de reconciliação é responsável pelo mapeamento das primitivas da interface da camada MAC com os diferentes PHYs.
São possíveis duas sub-camadas de reconciliação, sendo uma a SONET/SDH reconciliation sublayer (SRS) e a outra a GFP reconciliation sublayer (GRS) que é intencionada para ser usada com o protocolo GFP.
Ambas são especificadas para as interfaces 8-bit SPI-3, 32-bit_SPI-3, SPI-4.1 e SPI-4.2, dependendo dos ritmos de transmissão como definido no optical Internetworking Fórum (OIF). Embora sejam interfaces opcionais, no caso de se optar por não utilizar nenhuma delas, a interface deve operar de um modo funcional como se as mesmas estivessem implementadas. As sub-camadas de reconciliação estão descritas em maior detalhe no anexo C da norma 802.17.
A SONET/SDH sub-camada de reconciliação quer para o lado este ou oeste de uma estação RPR deve operar com o mesmo ritmo.

2.2.5. SONET/SDH sub-camada de adaptação

A norma 802.17 do RPR apresenta três sub-camadas de adaptação distintas para o mapeamento das tramas RPR em SONET/SDH, podendo ser baseada em GFP, HDLC-like e por último uma sub-camada de adaptação baseada em LAPS.
Devido às capacidades de multiplexagem e de concatenação virtual do SONET/SDH, assim como dos efeitos resultantes do encapsulamento de tramas, não existe uma relação directa de um para um entre o bit rate do RPR MAC e o correspondente meio físico SONET/SDH.
O sinal C2 especificado no ANSI T1.105 (SONET) e ITU-T G.707 (SDH) a ser usado nas camadas GFP, HDLC-like e LAPS está descrito na Tabela 10:

Adaptation sublayer type	C2 signal label
GFP framing	$1B_{16}$
HDLC-like framing	16_{16}
LAPS framing	18_{16}

Tabela 10 - sinal C2 signal para GFP, HDLC-like e LAPS

As funcionalidades da camada do SONET/SDH são descritas nas normas ANSI T1.105 (SONET), ITU-T G.707 e G.783 (SDH). As camadas superiores de caminho, todos os possíveis caminhos

concatenados e todos os possíveis caminhos virtuais concatenados são suportados pela norma 802.17 do RPR.

As camadas de SONET/SDH normalmente incluem mecanismos de protecção adicionais aos disponibilizados pelo MAC do RPR. A interoperabilidade entre os diferentes mecanismos de protecção é descrita em detalhe na cláusula 11 da norma 802.17 do RPR.

2.2.6. Formato das tramas RPR

As tramas RPR podem ser de quatro tipos (data frame, control frame, fairness frame e idle frame) sendo que para o cliente apenas são enviadas tramas de dados. Estas tramas têm a seguinte estrutura:

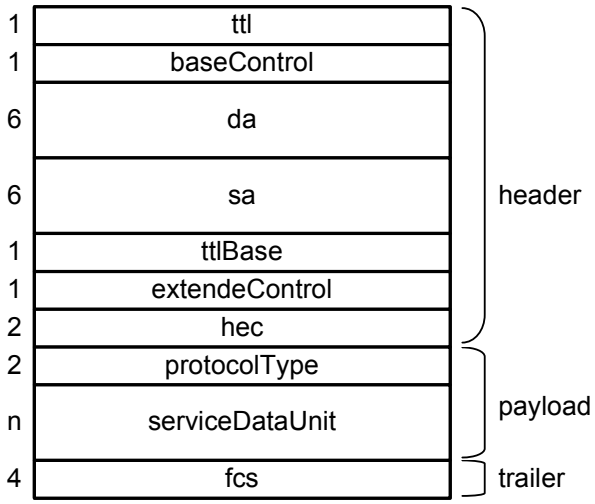


Figura 7 - Trama de dados RPR formato básico

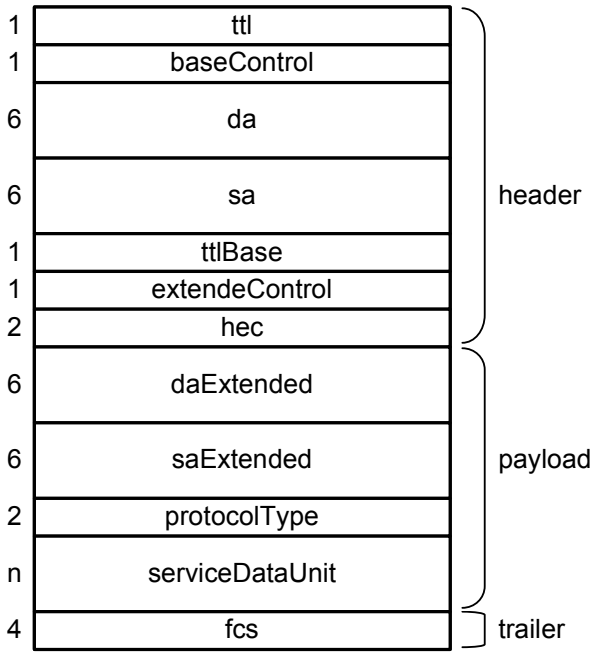


Figura 8 - Trama de dados RPR formato extended

As tramas de dados são identificadas por um campo de dois bits de largura ft (frame type) presente no byte baseControl. Quando os campos daExtended e saExtended Figura 8 são incluídos, a trama de dados é classificada como sendo uma trama de dados estendida. Para ser considerada como sendo uma trama de dados a mesma deverá ter presente o valor FT_DATA.

A estrutura do cabeçalho das tramas RPR diverge consoante o tipo de trama embora seja idêntica se for do tipo data frame e control frame assim como se for fairness frame e idle frame.

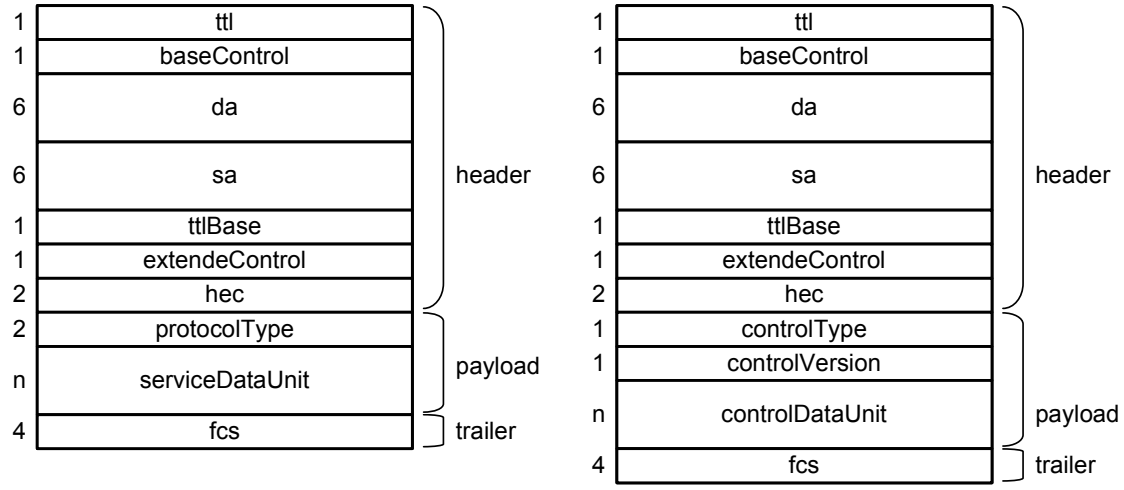


Figura 9 - estrutura de uma trama RPR de dados à esquerda e uma de controlo à direita

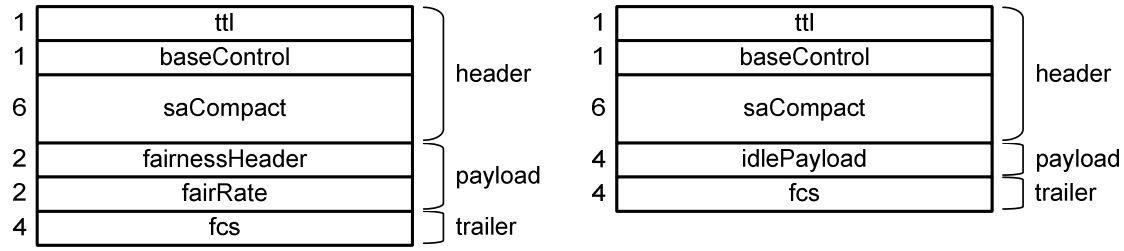


Figura 10 - estrutura de uma trama RPR de fairness à esquerda e uma de Idle à direita

O processamento dos pacotes RPR será distinto conforme o tipo de trama. Para distinguir os diferentes tipos de trama existe o campo frameType no interior do baseControl Figura 11. Note-se também, que por exemplo nas tramas de dados e controlo é necessário o cálculo do campo de HEC e FCS, ao passo que nas tramas de dados é apenas necessário o cálculo do FCS.

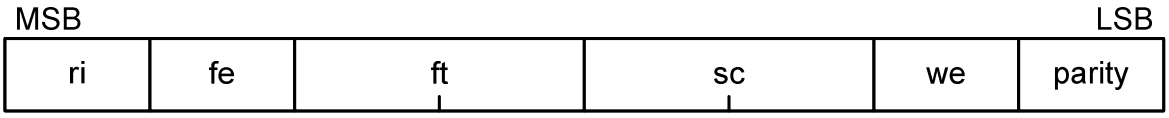


Figura 11 - campo baseControl

Nas interfaces RPR com o cliente desenvolvidas no decorrer desta dissertação de mestrado (ponto 4.3.1), o campo time to live ttl que especifica o numero de estações que a trama irá percorrer ao longo do anel assim como o campo ttlBase serão por defeito colocados com o valor máximo (FF em hexadecimal). O HEC, FCS e o parity bit do campo baseControl da trama RPR serão nulos pois o cálculo dos mesmos irá ser feito quando a trama for enviada para o anel ou seja na interface com a camada física (ponto 4.3.2).
Alguns campos iram ser alterados no módulo MAC do RPR como o ttl e o baseControl.

O campo baseControl como descrito anteriormente é composto por um primeiro bit ri que identifica o anel, sendo colocado a um ou a zero como apropriado. O campo fairness eligible (fe) irá ser colocado a zero indicando que a trama não é fairness eligeble, ao campo frame type (ft) será atribuído o valor 11 em binário por ser uma trama de dados (FT_DATA) e o bit wrap eligible (we) será colocado a um indicando que a trama é wrap eligible.
Os campos destination address (da) e source address (sa) da trama RPR assumirão os valores dos campos (da) e (sa) da trama Ethernet respectivos.
O campo extended control permanece a nulo pois as tramas do tipo extended data frame não irão ser consideradas.
O campo protocolType será calculado ficando com o valor do comprimento da trama RPR. Por ultimo os dados referentes à trama Ethernet serão inseridos dentro do campo serviceDataUnit da trama RPR finalizando a operação de encapsulamento.

O protocolo RPR permite também distinguir diferentes tipos de tráfego em mais ou menos prioritário, conforme os valores colocados no campo sc (service class) do campo baseControl, à semelhança dos primeiros três bits (User Priority) do campo Tag Control Information da trama Ethernet. Segundo o suplemento P802.17a/D1.0 da norma 802.17 temos:

Access_priority	MAC service class
0	classC
1	classC
2	classC
3	classC
4	classB
5	classB
6	classA
7	classA

Tabela 11 - User Priority e Service Class

2.2.7 Propostas comerciais

Os principais membros envolvidos no desenvolvimento da norma 802.17 são fabricantes de topo no mercado das telecomunicações como a Cisco, Corrigent, Infineon, Lantern Communications e Nortel. Todos eles têm interesse no desenvolvimento de produtos RPR baseados na norma 802.17.

Cisco

A CISCO possui uma Multiservice Provisioning Platform (MSPP) capaz de agregar uma variedade de cartas (ML-Series) com funcionalidades Ethernet e RPR, facilitando o acesso de clientes com tráfego de pacote ao seu produto.

Ao contrário da tecnologia SONET/SDH, na tecnologia RPR a largura de banda é gerida de uma forma flexível, o que permite aos operadores transmitir Gigabit Ethernet usando, por exemplo, apenas circuitos STS-12c/VC-4-4c.

Corrigent

A arquitectura CM-100 da Corrigent funde as tecnologias SONET/SDH com concatenação virtual, GFP (descrito no ponto) e LCAS com tecnologias de pacote como o RPR, MPLS ou Ethernet, alterando o paradigma actual das redes do género, misturando as usuais infra-estruturas de transporte SONET/SDH com a próxima geração de redes de transporte orientadas ao pacote.

A tecnologia LCAS (Link Capacity Adjustment Scheme) consiste num método de aumentar ou diminuir a largura de banda de contentores virtuais, permitindo a alocação dinâmica de largura de banda de clientes como Ethernet, mapeados em SDH.

A arquitectura CM-100 da Corrigent utiliza a tecnologia MPLS para engenharia de tráfego, gestão de utilizadores e serviços no anel, monitorização ponto a ponto automática das redes, manter qualidade de serviço ponto a ponto. A tecnologia RPR permite criar um meio partilhado de múltiplos nós, em que os utilizadores utilizam de um modo eficientemente toda a largura de banda do anel através das potencialidades da tecnologia RPR. O equipamento faz uso da tecnologia SONET/SDH servindo de suporte às tecnologias anteriores, configurando deste modo o equipamento RPR para funcionar sobre redes SONET/SDH.

Infineon

A Infineon, fabricante de circuitos integrados, e a ZTE Corporation, o segundo maior fornecedor de dados e sistemas de telecomunicações na China, fizeram uma parceria no sentido de desenvolver soluções RPR para os seus principais clientes. Ambos acreditam em espalhar a solução RPR por toda a Ásia, abrindo assim uma porta para que outros operadores mundiais se interessem e acelerem a sua aquisição. Para que isso ocorra, a Infineon anuncia o desenvolvimento de um circuito integrado (Frea™ PoS Framer/RPR MAC inte-grated circuit (IC)) baseado no draft 2.1 na norma 802.17 cujo objectivo é a integração do mesmo ou de uma futura versão compatível com a norma 802.17 nos equipamentos de MAN da ZTE Corporation. Como anuncia o fabricante, o circuito integrado será composto por um framer PoS (Packet-over-SONET), um RPR MAC (Media Access Control) e um XAUI SerDes (serializer/deserializer), 1 Mbyte de memória, uma interface SPI-4.2 16-bit 800 Mhz e uma 4-bit 3.125 GHz mate (XAUI), interface para permitir uma ligação eficaz entre dois chips necessários para uma completa solução RPR com capacidade de transmissão e recepção.

XILINX

Para além destes fabricantes existem também outros que têm trabalhado sobre a norma 802.17 no desenvolvimento de produtos relativos às competências de cada um dos fabricantes como a Alliance Semiconductor, Lucent, Luminous e Xilinx. Por exemplo o fabricante de FPGAs Xilinx anuncia um core para as suas FPGAs que executa o MAC RPR segundo a norma 802.17 bem como um conjunto de interfaces associadas.

A solução da XILINX é semelhante à solução desenvolvida no decorrer desta dissertação de mestrado no que diz respeito às interfaces de linha.

O core desenvolvido pelo fabricante XILINX foi optimizado para as FPGAs da família Virtex-4, que possui algumas diferenças de tecnologia relativamente às Virtex2Pro usada nesta dissertação de mestrado. As diferenças são sobretudo relativamente à existência de linhas de relógio regionais que foram adicionadas à tecnologia Virtex-4. O core, segundo anuncia o fabricante XILINX, é uma solução que segue a norma 802.17 e que implementa todos os blocos chave da tecnologia RPR

como o MAC datapath para este e oeste, Fairness, Topology, Protection e OAM. Trata-se de uma solução flexível pois suporta diferentes ritmos de transmissão (1G, 2.5G e 10G) para diversos PHYs.

A Tabela 12 ilustra as potencialidades do produto anunciado pelo fabricante XILINX.

• Débitos suportados	1.25G (GE), 2.5G (OC48), 10G (OC-192/10GE)
• Interfaces	Interfaces SPI 3.1, GMII, XAUI, XGMII and SPI 4.2 Interface de dados/endereços genérica de 32 bits para controlo. Interface genérica de 64 bits de dados, 8 bits de controlo para a interface cliente.
• Single Core	Anel duplo com steering e protecção wrapping 50ms
• Equipamento com redundância possuindo uma interface que poderá comunicar com outra carta.	
• Modo adicional Master/Slave	
• Classes de service Class A, Class B (CIR / EIR), Class C	
• Filas de transito únicas ou duplas	
• STQ configurável até 128MB, PTQ para 8 tramas incluindo tramas do tipo jumbo	
• Fairness Conservativo ou Agressivo	
• Todas as funções de tempo real em HW; OAMP, base de dados da topologia em SW	
• Suporte de tramas jumbo	
• Consegue suportar tráfego em carga máxima simultaneamente nos dois anéis	
• Disponibilização de SW incluindo drivers, MIB, SNMP	
• IEEE 802.17	

Tabela 12 - Potencialidades do core RPR XILINX

A performance e a optimização da utilização dos recursos são conseguidas, segundo o fabricante, através de uma correcta divisão dos blocos entre implementação em hardware e software como demonstra a Figura 12:

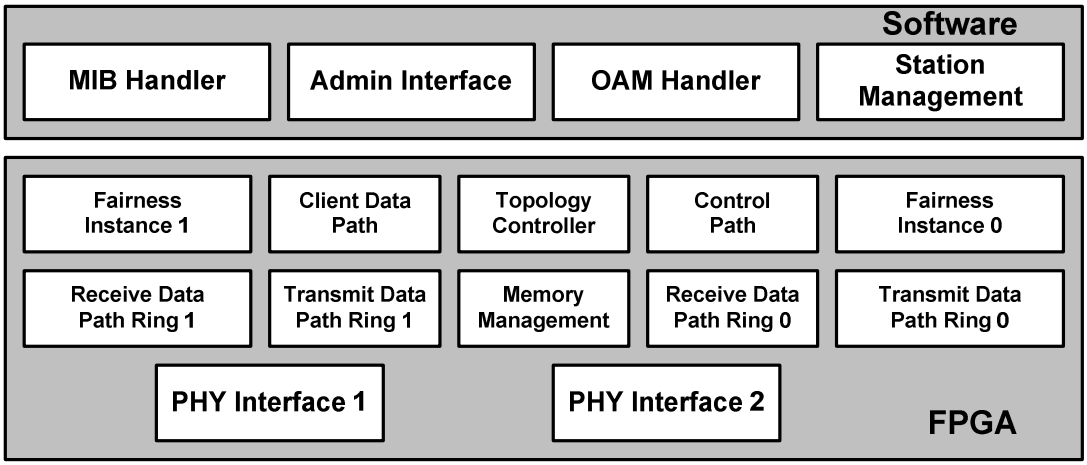


Figura 12 – blocos implementados em hardware e software

O core pode ser utilizado em variados sistemas como Multi-Service Provisioning Platforms (MSPP), IP-DSLAMs e equipamentos Wireless Backhaul. A Figura 13 demonstra uma aplicação do core RPR MAC numa MSPP linecard.

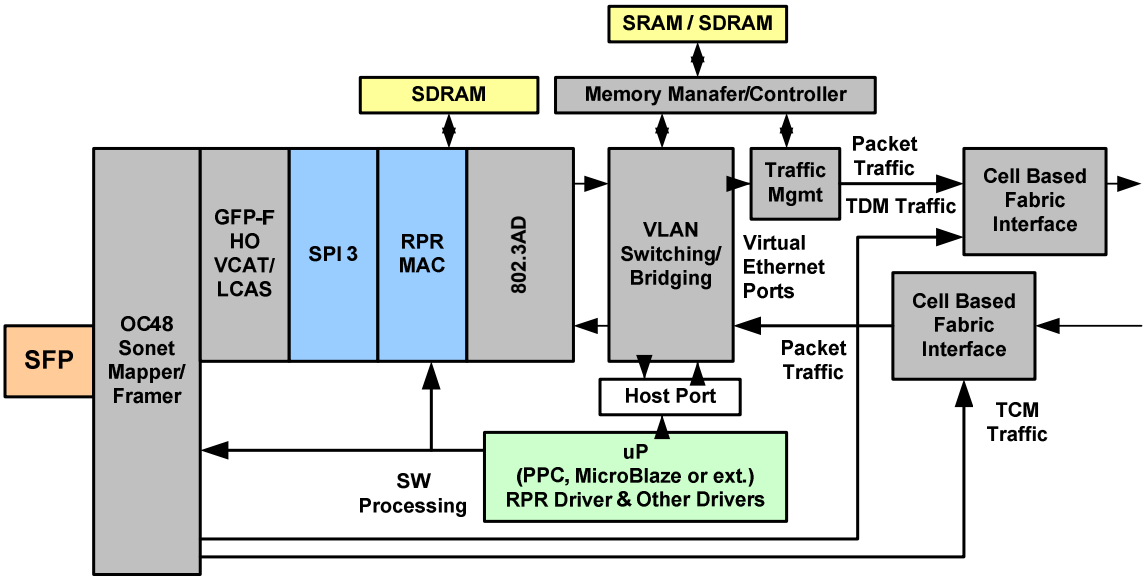


Figura 13 - core RPR MAC numa MSPP linecard

Alliance Semiconductor

A Alliance Semiconductor anuncia uma família de controladores RPR (AS95L2100) com o intuito de suportar a norma IEEE 802.17 com diversas interfaces a ritmos de transmissão variados OC-12, OC-48 e OC-192 para SONET e 2.5G, 10G para Ethernet. O controlador RPR incorpora o

resilient ring Media Access Control (MAC) juntamente com funções de camada1 e 2 como framers SONET, Ethernet MACs, PÓS/HDLC e GFP.

2.3 ATM -Asynchronous Transfer Mode

A tecnologia ATM (Asynchronous Transfer Mode) [22][23][27][28] situa-se ao nível da camada 2 do modelo OSI como descrito na Figura 14.

Surge para dar resposta a um conjunto de necessidades por parte dos operadores não previstas na tecnologia SDH, pensada sobretudo para tráfego de voz. Com o surgimento de novos tipos de tráfego vídeo/dados/voz e serviços multimédia é necessário arranjar alternativas à tecnologia SDH para colmatar certas lacunas.

A rede ATM é orientada à ligação para qualquer tipo de tráfego voz/dados/vídeo permitindo a interligação de redes melhorando a eficiência e a gestão. A largura de banda é também um factor a ter em conta sendo possível com este tipo de tecnologia obter flexibilidade para determinar qual o debito escolhido para a ligação desde Kbit/s até Gbit/s. Também é escalável e flexível em relação ao número de utilizadores e distancias geográficas necessárias para a rede.

O tráfego ATM é composto por um conjunto de pequenas células de tamanho fixo com 5 bytes do cabeçalho e restantes 48 bytes para dados. Isto faz com que haja acréscimo de informação devido ao cabeçalho da célula (~10%). Também no caso de ocorrência de congestionamento poderá originar perda de células.

Tal como a tecnologia RPR, também o ATM pode ser combinado com o SDH levando a uma solução interessante do ponto de vista dos operadores, podendo o ATM funcionar em qualquer rede física existente (par cobre, coaxial, fibra). O ATM Fórum [27] tem sido um dos principais impulsionadores pela aceitação da tecnologia ATM no mercado das telecomunicações. A organização ITU-T também tem contribuído com um conjunto de interfaces e sistemas de sinalização.

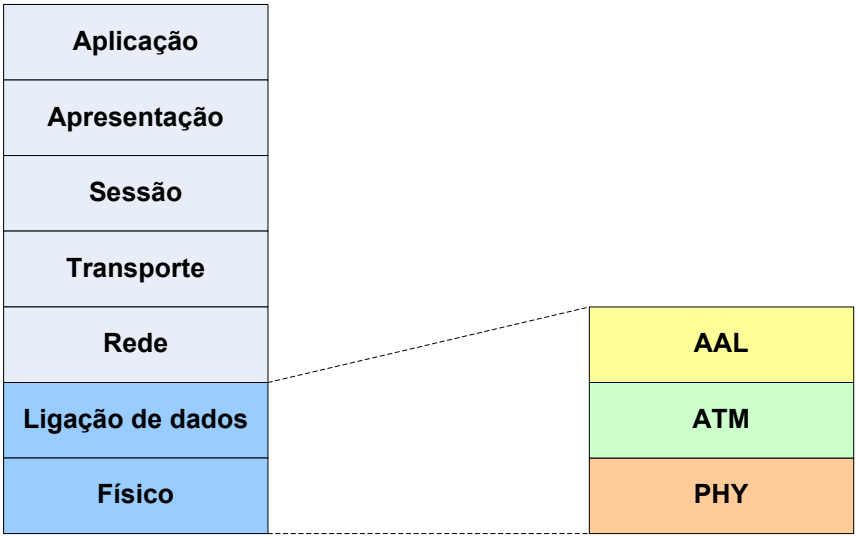


Figura 14 - correspondência entre modelo osi e tecnologia ATM

Devido à sua natureza assíncrona, a tecnologia ATM é mais eficiente que a maioria das tecnologias TDM. No caso TDM cada utilizador tem associado uma determinada largura de banda fixa. Se o utilizador a determinado momento quiser enviar mais informação do que o que lhe tiver atribuído não o poderá fazer. Por outro lado se o mesmo utilizador tiver pouca informação a enviar não poderá partilhar a seu excedente de largura de banda com nenhuma outra qualquer estação. Sendo a tecnologia ATM assíncrona isto já não acontece, havendo um melhor aproveitamento da largura de banda disponível. Contudo irá ocorrer um acréscimo de overhead na informação transmitida (correspondendo ao cabeçalho das células ATM).

Uma rede ATM é composta por equipamentos de comutação e terminação ATM. O equipamento de comutação tem de ler e processar o cabeçalho da célula em conformidade com as configurações presentes. Quanto aos equipamentos terminais resumem-se por exemplo a workstations, routers, LAN switch entre outros.
Os comutadores ATM suportam dois tipos de interfaces primárias, User-to-Network interface (UNI) e Network Node Interface NNI. A UNI é utilizada na ligação entre um comutador ATM e um qualquer equipamento terminal (workstation, router…). A NNI faz a ligação entre dois comutadores ATM distintos.

As interfaces UNI e NNI poderão ser subdivididas em públicas ou privadas, conforme os equipamentos de comutação forem propriedade privada de uma qualquer empresa ou do domínio da rede pública.

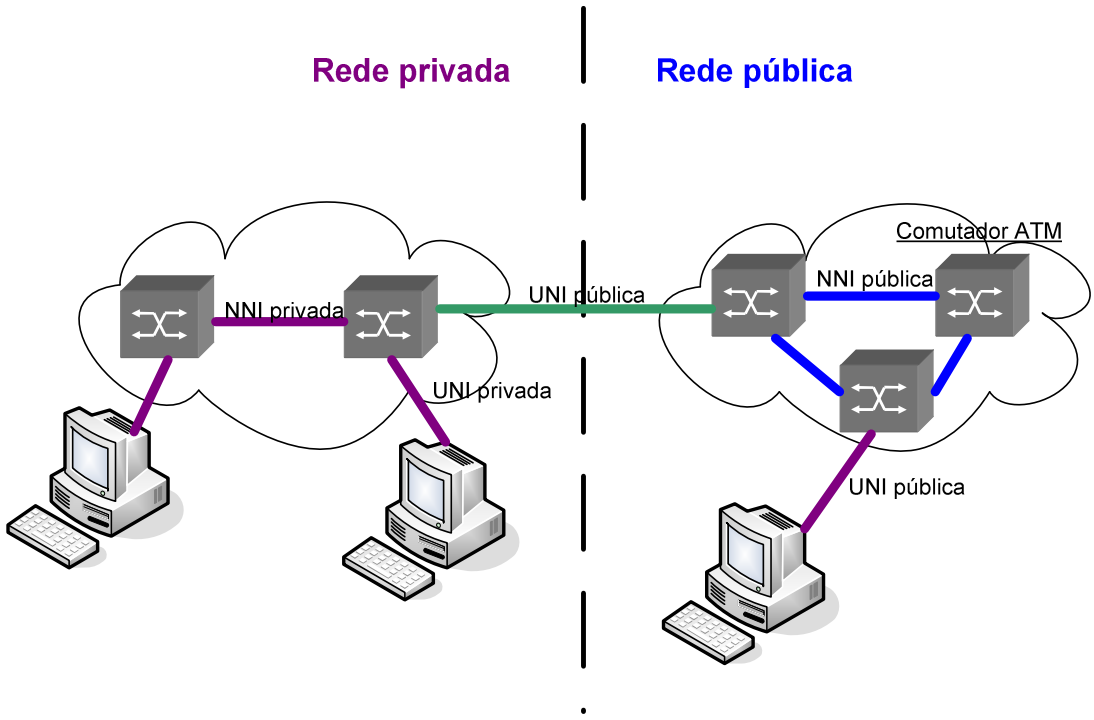


Figura 15 - especificações das interfaces ATM para redes públicas e privadas

2.3.1. Camada ATM

A camada ATM é análoga à camada de ligação de dados do modelo OSI e é responsável pelo transporte da informação através da rede fazendo uso de conexões virtuais para transportar a informação. Embora tratando-se de uma ligação ponto a ponto, esta apenas irá ocupar largura de banda efectiva no instante em que for transmitida uma qualquer célula nesta ligação. As conexões são subdivididas em caminho virtual (VP) e canal virtual (VC) com o intuito de simplificar a estrutura de endereçamentos ATM.

Um VP consiste num conjunto de VC's capazes de serem comutados de forma transparente ao longo de uma qualquer rede ATM tendo em comum o valor presente no campo VPI do cabeçalho das células ATM. Os valores VPI's e VCI's têm um significado apenas local sendo reformulados à medida que atravessam um determinado comutador. A Figura 16 ilustra como se relaciona o meio físico com os diferentes valores de VPI's e VCI's:

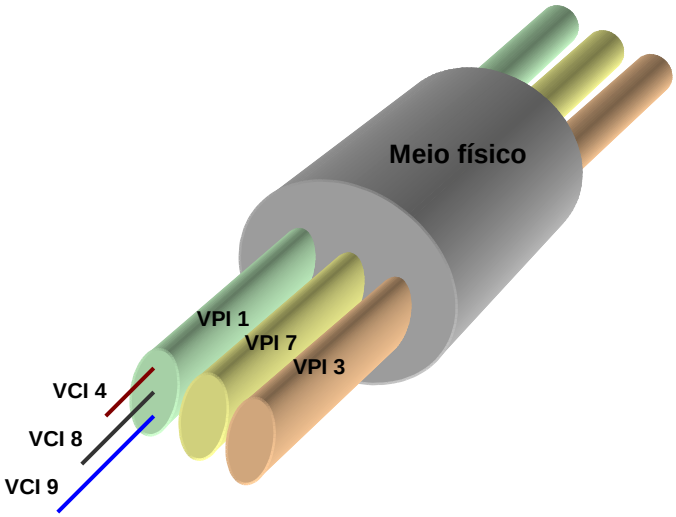


Figura 16 - relação entre VPI's VCI's e meio físico

Os campos de identificação de VP (VPI) e VC (VCI) vão sendo atribuídos pela rede, utilizador, negociando entre utilizadores e a rede (sinalização), ou são valores atribuídos por defeito referentes à normalização da tecnologia ATM.

A Tabela 13 apresenta um exemplo da tabela de comutação de VPI/VCI.

Porta	VPI	VCI	Porta	VPI	VCI
1	5	9	3	3	4
1	2	7	3	7	7
2	1	1	3	3	2

Tabela 13 - tabela de comutação de VPI/VCI

A Figura 17 ilustra um exemplo de como diferentes valores de VPI e VCI são processados através de um comutador de VP ou de VC tendo por base a Tabela 13 de comutação.

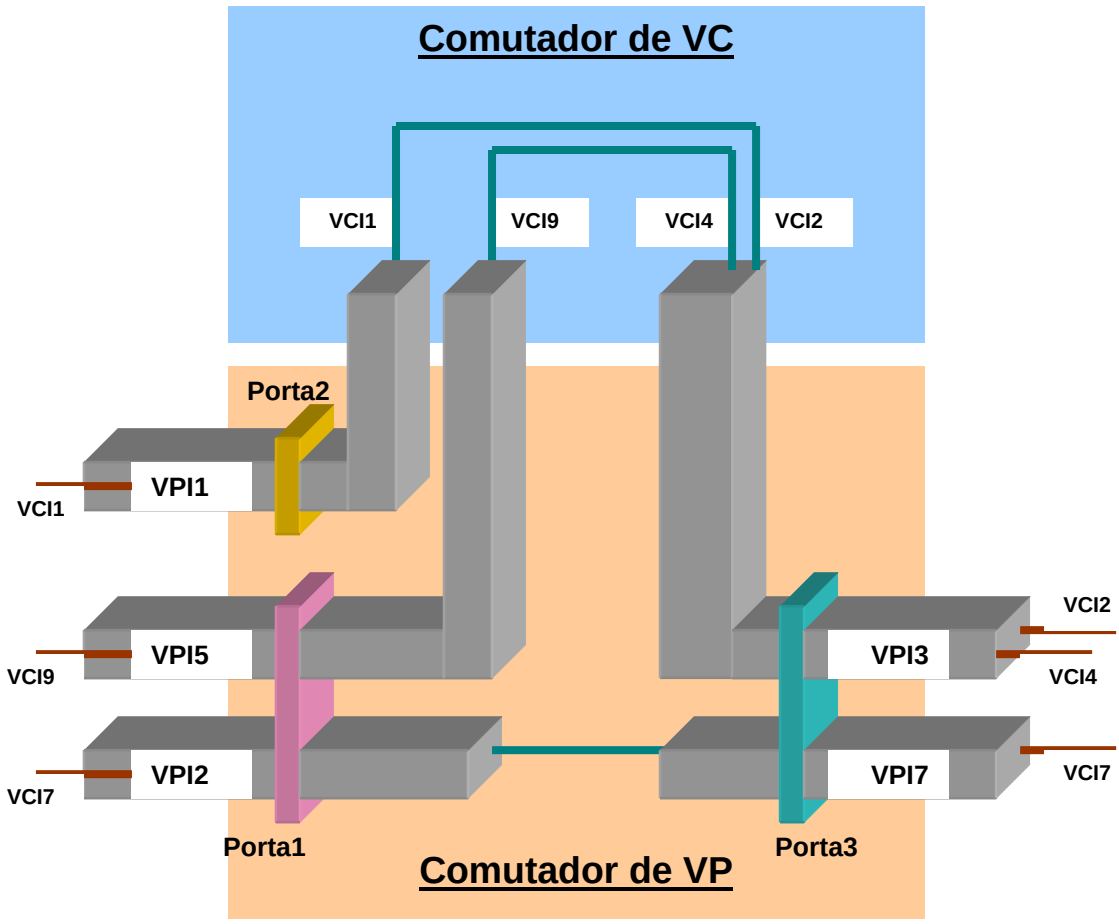


Figura 17 - comutação de VPI/VCI

2.3.2. Célula ATM

A estrutura do cabeçalho de uma célula ATM segue dois formatos distintos podendo ser um cabeçalho UNI ou NNI. Para as células UNI existe um campo extra de 4 bits de comprimento Generic Flow Control (GFC), sendo que o comprimento do campo VPI para o cabeçalho UNI é de apenas 8 bits e no NNI é de 12 bits.

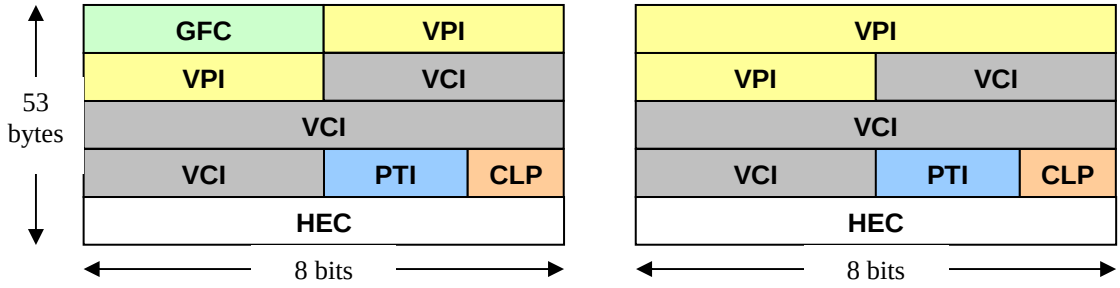


Figura 18 - cabeçalho de célula ATM UNI à esquerda e NNI à direita

Segue uma breve descrição de cada um dos campos do cabeçalho da célula ATM apresentados na Figura 18:

- **Generic Flow Control (GFC)** disponibiliza algumas funções locais como a identificação de estações múltiplas que partilham a mesma interface ATM. Tipicamente não é usado e o seu valor é por defeito colocado a zero

- **Virtual Path Identifier (VPI)** juntamente com o campo VCI identifica o próximo destino de uma célula à medida que a mesma circula ao longo de uma série de comutadores ATM.

- **Virtual Channel Identifier (VCI)** juntamente com o campo VPI identifica o próximo destino de uma célula à medida que a mesma circula ao longo de uma série de comutadores ATM. As Células são encaminhadas examinando primeiro o campo VPI e depois o campo VCI.

- **Payload Type (PT)** o primeiro bit indica se a célula contem dados de utilizador ou de controlo sendo colocado a 0 ou 1 respectivamente. O segundo bit, quando colocado a um, significa que se está perante um cenário de congestionamento. Quanto ao terceiro bit do campo, se este for um significa que a célula é a última de uma série de células que representam uma única trama AAL5 (ver secção 2.3.3.)

- **Cell Loss Priority (CLP)** Indica se a célula deve ou não ser descartada no caso de haver uma situação de congestionamento extrema à medida que a mesma vai circulando ao longo da rede. Se for igual a um, a célula deverá ser preferencialmente descartada relativamente a células que possuam o campo CLP a zero.

- **Header Error Control (HEC)** Calcula um código cíclico de oito bits segundo um polinómio especificado na recomendação [12] sobre os primeiro quatro bytes do cabeçalho.

2.3.3. Camada de adaptação ATM (AAL)

A camada de adaptação ATM (AAL) é análoga à camada de ligação de dados do modelo OSI. A camada AAL é responsável por isolar protocolos das camadas superiores (no modelo OSI) dos detalhes inerentes ao processamento da informação no formato das células ATM.
Sendo assim, a informação é transportada de modo transparente pela AAL não conhecendo a estrutura da mesma. Não existe qualquer relação entre o relógio da fonte e da rede sendo todo o inter funcionamento da responsabilidade da camada AAL. Este terá de conseguir lidar com o fluxo de dados, variações nos atrasos sofridos pelas células, perda de células, e células mal encaminhadas, entre outras questões. Estas tarefas são executadas nas pontas da rede ATM, frequentemente por adaptadores. O uso da camada AAL depende do tipo de tráfego/protocolo que se pretende transmitir através da rede ATM. Há cinco camadas AAL definidas.
Resumidamente a camada AAL fará uma adaptação da rede ATM para um determinado tipo de tráfego/protocolo a enviar tendo por base um acréscimo de overhead necessário ao seu funcionamento. Os 48 bytes da célula serão utilizados para envio da informação a transmitir resultante das camadas superiores do modelo OSI bem como da redundância inerente ao tráfego AAL necessário.

2.3.4. Camada física ATM

A camada física ATM divide-se em duas sub-camadas, a physical medium-dependent (PMD) e a sub-camada transmission convergence (TC).
A sub-camada PMD é responsável pelo envio dos bits, códigos de linha, e adaptações eléctricas/ópticas, entre outras funcionalidades. Um exemplo de um meio físico normalizado para a tecnologia ATM é o SONET/SDH sobre uma rede óptica. Mais adiante nesta dissertação podemos constatar a implementação em FPGA de um mecanismo capaz de remover e inserir células ATM sobre um contentor virtual VC4 da norma SDH. O VC4 é preenchido com as células ATM a um débito aproximado de 155 Mbit/s. No caso de não haver células suficientes para preencher todo o VC4 são utilizadas células IDLE especificadas na recomendação [12]. A partir do campo HEC do cabeçalho das células ATM é possível o equipamento delinear as células através da procura de um padrão de 5 bytes que possa ser um cabeçalho válido.

2.4 IP e ATM

A tecnologia IP [24], sendo uma tecnologia não orientada à conexão com tráfego de pacotes, é uma história de sucesso no mundo das tecnologias de informação e telecomunicações.

O IP permitiu que estruturas gigantescas de comunicação de dados, como a Internet, se desenvolvessem e alterassem significativamente o nosso modo de viver, o funcionamento de entidades, instituições e os negócios das empresas. O IP tem servido para o aparecimento de um conjunto diverso de serviços imprescindíveis nos dias de hoje como o comércio electrónico, intranets em empresas, emails entre outros.

Este enraizamento da Internet fez com que as infra-estruturas ATM orientadas à conexão já existentes tivessem de coabitar com a tecnologia IP. Contudo o aparecimento da tecnologia IP fez com que tecnologias já usadas como o ATM ou o Frame-Relay tivessem de ser reajustados aparecendo assim alguns modelos de sobreposição que permitem IP sobre ATM ou Frame Relay.

Esta solução está muito longe de ser a ideal pois obriga a que haja dois espaços de endereçamento distintos, a topologia da rede não é conhecida dos routers IP e temos diferentes protocolos de sinalização e routing. A agravar a situação temos também de considerar a inclusão do cabeçalho da célula ATM que diminui o envio de dados úteis na rede.

O ATM Fórum e o IETF têm vindo a reconhecer a necessidade de um mecanismo efectivo para a coabitação do ATM e IP. Um variado número de normas foi desenvolvido com o objectivo de mapear IP sobre redes ATM:

- RFC1577 e RFC2225 - Classical IP over ATM (IETF)
- NARP and NHRP, IETF
- LAN Emulation (LANE), ATM FORUM
- Multi-Protocol over ATM (MPOA), ATM FORUM
- Multi-Protocol Label Switching MPLS, IETF

Uma das estratégias usadas para se conseguir reunir o IP e o ATM é derivada da topologia de rede. O objectivo é tentar construir um misto de ligações virtuais ATM que formam atalhos entre pontos terminais da rede. Outra estratégia resulta da análise do tráfego IP que circula na rede, construindo posteriormente circuitos virtuais que carregam esses fluxos de dados.

O Classical IP over ATM (CIP), foi originalmente definido no RFC1577 e mais tarde substituído pelo RFC2225. A CIP utiliza as ligações ATM como se uma outra qualquer ligação de dados se tratasse (por exemplo: Ethernet, Token Ring). Tem como principal desvantagem que o encaminhamento dos pacotes é feito ao nível dos routers IP não aproveitando eficientemente possíveis ligações ATM virtuais directas. O IETF desenvolveu o NHRP (Next Hop Routing Protocol) que consegue determinar ligações ATM virtuais directas para determinados subdomínios

de rede IP. A LANE é uma norma desenvolvida pelo ATM Fórum que relaciona a tecnologia ATM com diversos protocolos de LAN como a tecnologia Ethernet. Permite transportar os datagramas IP sobre a tecnologia ATM como se de uma rede LAN se tratasse. O MPOA integra as funcionalidades da LANE preservando os benefícios de emulação de uma LAN e do NHRP incorporando as vantagens a estas associadas. A comunicação feita nos VCC's do ATM não requer routers ao nível da camada de ligação de dados. A tecnologia MPLS surge com o objectivo de dar resposta à necessidade crescente do uso de datagramas IP sobre a camada de ligação de dados. Trata-se de uma tecnologia pensada de raiz para o transporte de datagramas IP, eliminando os modelos de sobreposição que permitem IP sobre outras tecnologias já existentes como o ATM.

2.5 MPLS - Multi Protocol Label Switching

O MPLS permite a unificação e interoperabilidade através do uso em cada pacote de um cabeçalho curto e sempre do mesmo tamanho chamado label que tem apenas significado local. A camada MPLS [25] situa-se ao nível da camada datalink do modelo OSI logo abaixo do protocolo IP e garante a interligação entre routers LSR (Label Switching Router). Um conceito fundamental da tecnologia MPLS é o conceito de FEC (Forwarding Equivalent Class) que permite agrupar pacotes em diferentes categorias para serem posteriormente tratados de igual forma pelos routers LSR. O MPLS recorre à construção das tabelas de encaminhamento para o efeito. Uma ligação física numa rede MPLS é constituída por vários LSPs (Label Switched Paths) em que cada LSP é definido por uma "etiqueta" (variável em cada troço entre LSRs) sendo iniciado e terminado nos LSRs de fronteira de um domínio MPLS. À medida que os pacotes vão entrando na rede vão-lhes sendo atribuídos labels. Estes labels irão associar os pacotes a determinados FEC que serão expelidos de acordo nos LSR.

Na Figura 19 podemos observar quatro routers ligados entre si. Se optarmos por encaminhamento IP cada router tem de ter na sua tabela de encaminhamento informação relativa a seis redes IP distintas. Se o encaminhamento for com base nas labels dos dois LSPs (Label Switched Paths) distintos, então teremos apenas de ter duas entradas na sua tabela de encaminhamento.

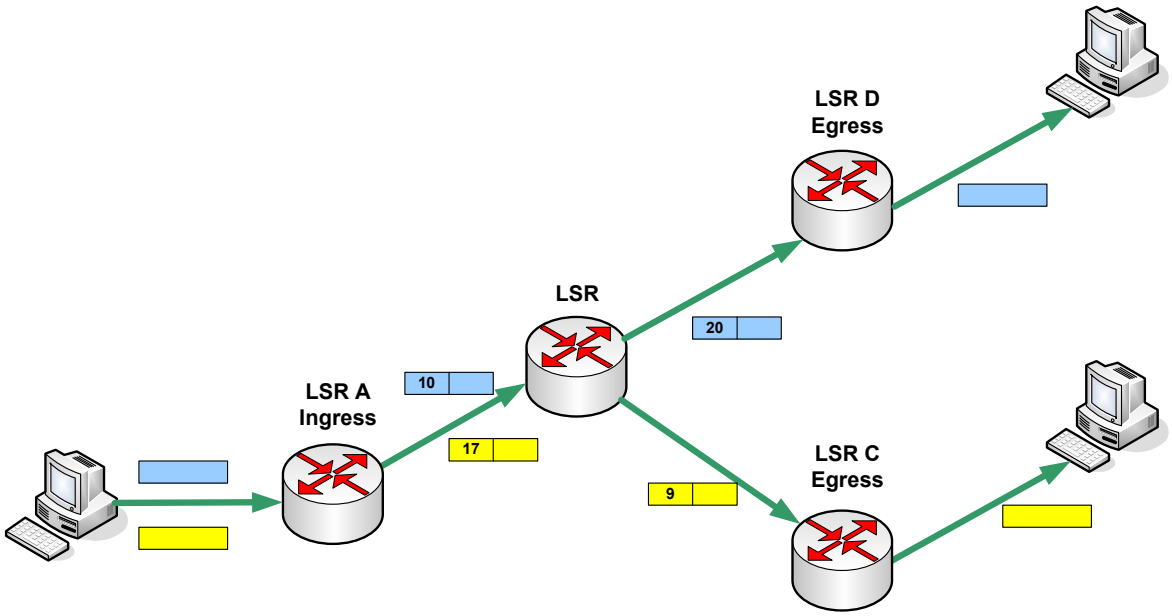


Figura 19 - Rede MPLS

O comutador de ingresso da rede é responsável por determinar a rota (LSP) que deve seguir o pacote, e para tal atribui o label correspondente à rota comutada. O comutador de egresso, sabendo que não está ligado a um equipamento MPLS, retira o label e envia o pacote usando o IP convencional. Se a tecnologia usada na camada 2 suportar um campo de label (como o ATM e o Frame-Relay), então é nesse campo que vai o label MPLS. Nas outras tecnologias de nível 2, o label MPLS é transportado num cabeçalho normalizado que é inserido entre o cabeçalho IP e o do nível 2 (802.3, etc.), continuando o equipamento a beneficiar da comutação de nível 2.

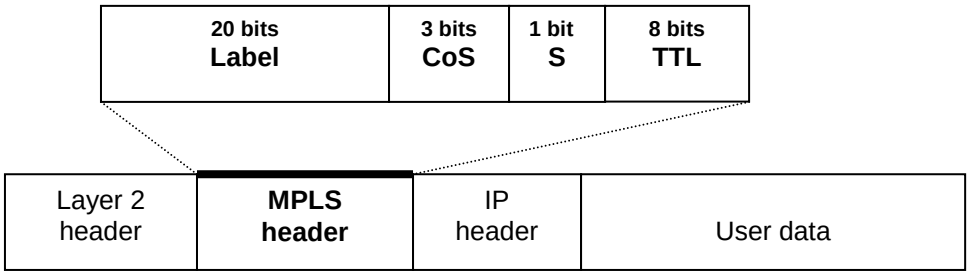


Figura 20 - cabeçalho normalizado MPLS

Segue uma breve descrição dos campos ilustrados na Figura 20:

- O campo Label possui o valor da label MPLS
- O campo CoS afecta o algoritmo de armazenamento e descartes
- O campo Stack S suporta a label de stack da hierarquia
- O campo TTL disponibiliza o TTL convencional do IP

3 O sistema SIRAC e as suas interfaces

Nos últimos anos têm surgido alterações ao nível das características do tipo de tráfego no sector das telecomunicações. O tráfego de voz tem vindo a ser substituído em termos de volume pelo tráfego de dados. Este factor tem vindo a condicionar um conjunto de equipamentos e redes em funcionamento há já alguns anos. Os mesmos foram projectados com o objectivo de servir apenas tráfego voz, conseguindo acomodar, com algum custo e pouca eficácia, os serviços de dados que vão continuamente aparecendo.

Esta evolução do tipo de tráfego tem vindo a suscitar inúmeros esforços por diferentes intervenientes do sector das telecomunicações, como a comunidade científica, académica, Instituições de I&D, alguns operadores e fabricantes. É neste cenário de evolução que o projecto SIRAC se enquadra, tendo por base o desenvolvimento de produtos para redes de nova geração. Alguns objectivos específicos do projecto SIRAC são o desenvolvimento de produtos baseados na tecnologia SDH já existente, aumentando a sua largura de banda e permitindo o transporte misto de circuitos TDM e de pacotes com multiplexagem estatística na rede de acesso. É esta fase do projecto que pressupõe o desenvolvimento e implementação do protocolo 802.17. Também se pretende o desenvolvimento de produtos Wave Division Multiplexing WDM, aumentando a largura de banda nas redes de transporte metropolitanas, e de Media Gateways MGW para o inter-funcionamento de voz entre a rede comutada e a rede de pacotes.

3.1 Arquitectura SDH de base desenvolvida

A arquitectura actual da rede consiste num anel óptico fechado SDH-STM1, permitindo o transporte e interligação entre diferentes serviços de dados e de voz simultaneamente. A rede é constituída por elementos ópticos e eléctricos que se interligam entre si através de unidades ONU (Optical Network Unit) como podemos ver na Figura 21. Consegue-se transportar no anel tráfego proveniente de diversas fontes como GSM, ATM, HDSL, ou seja, todo um conjunto de tecnologias que por sua vez definem os serviços comerciais associados.

O débito máximo atingível de um anel STM1 está limitado a 155,4Mbit/s (como vimos na secção 2.1). Contudo as exigências verificadas pela necessidade de procura de novos serviços por parte dos clientes não param de aumentar, sendo necessário desenvolver novas tecnologias e produtos. O aparecimento da tecnologia 3G, por exemplo, que introduz as vídeo chamadas, assim como outros serviços que a rede GSM não permite, é um bom exemplo. No entanto toda esta procura por novos serviços resulta numa necessidade de introduzir também alterações na rede de transporte SDH apresentada na Figura 21.

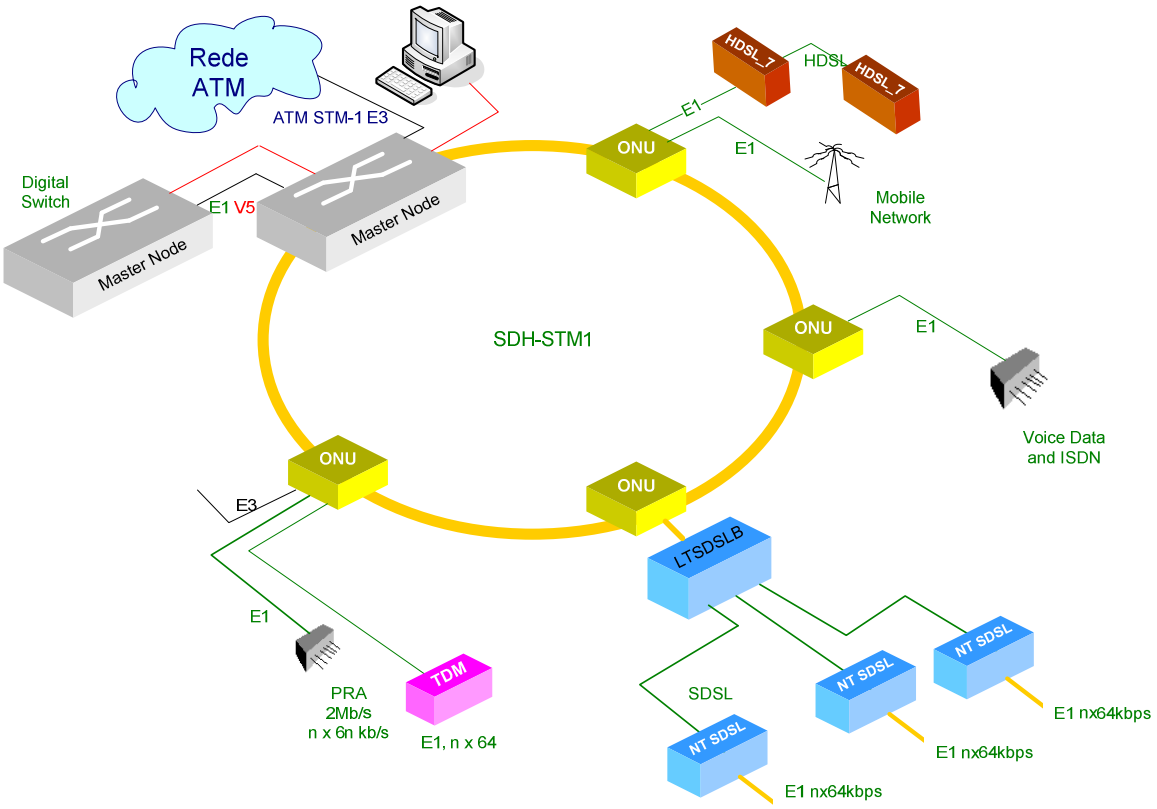


Figura 21 - Rede de transporte

Torna-se por isso necessário redimensionar a rede de modo a garantir aumentos de largura de banda. A alteração pode ser feito tanto no domínio eléctrico como no óptico. Neste caso optámos por intervir nos dois domínios simultaneamente. No primeiro iremos intervir ao nível do SDH-STM1 substituindo por um novo sinal SDH-STM16, o que implica aumentar a largura de banda no anel sobre um factor de dezasseis. No segundo, pretende-se incluir na rede um equipamento ADM (Add Drop Multiplexer), que possibilita que o anel possa ser ligado a uma rede com tecnologia DWDM passando a fazer parte integrante dessa mesma rede. Saliente-se que a tecnologia DWDM permite o envio de múltiplos canais em diferentes comprimentos de onda sendo que num desses comprimentos de onda podemos enviar todo o tráfego resultante no nosso anel SDH-STM16.

3.2 Evolução

Devido ao aparecimento crescente de novos serviços é necessário proceder a alterações no funcionamento tradicional de uma rede SDH, tal como representado na Figura 22.

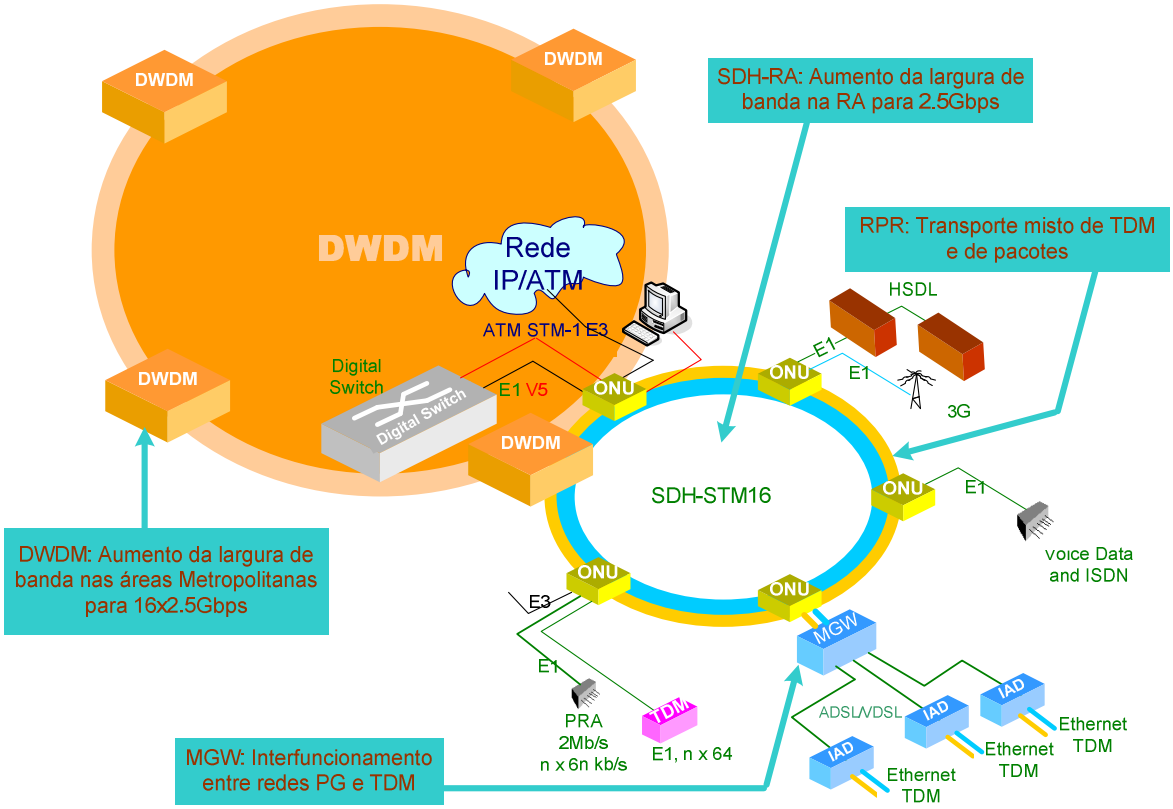


Figura 22 - Evolução da rede de transporte

Na Figura 22 aparece assim um conjunto de novas tecnologias, como o DWDM, que além de permitir o aumento da largura de banda altera o tipo de acesso ao meio (que deixa de ser puramente TDM). Outra tecnologia, o RPR, procura um compromisso óptimo para o transporte de tráfego de pacotes tentando rentabilizar tanto quanto possível a largura de banda do anel. Outra possibilidade mais simples do ponto de vista conceptual por ser uma tecnologia mais antiga e portanto sobre a qual se tem um maior domínio é actuar a nível do TDM, fazendo uma actualização da rede para um STM-16, cujo débito máximo é 16 vezes superior ao de um anel STM-1 ou seja 2.5Gbit/s aproximadamente. Misturar o WDM com o SDH, RPR e ATM é também possível, como iremos ver no decorrer desta dissertação de mestrado. Pretendemos desenvolver equipamentos capazes de mapear um anel RPR sobre um link STM-1, que por sua vez irá ser transportado num link STM-16. Em seguida iremos também desenvolver um outro circuito, capaz de introduzir células ATM sobre um link STM-1 e em seguida envia-lo no mesmo anel SDH STM-16, como podemos verificar na Figura 22. Este trabalho de mestrado situa-se nesta evolução de rede.

3.3 Interfaces do sistema

O trabalho reportado nesta dissertação de mestrado associado ao projecto SIRAC incide sobre o módulo RPR do projecto SIRAC, mais especificamente as interfaces do MAC do RPR com o anel SDH que vemos na Figura 22. Embora ambas as tecnologias DWDM e RPR consigam ganhos de largura de banda na rede, estas são bastante diferentes. No DWDM as alterações na rede ocorrem ao nível da camada física ou se quisermos ao nível da camada óptica. No RPR, teremos de desenvolver novos equipamentos para a rede de transporte com novas potencialidades como descrito na norma 802.17. O projecto SIRAC pretende fazer um conjunto de alterações sobre os equipamentos que compõem a camada óptica e também os que constituem a rede de transporte de dados. É apenas sobre os equipamentos da rede de dados que nos iremos focalizar no decorrer desta dissertação de mestrado desenvolvendo um conjunto de soluções RPR e ATM que permitem uma evolução da rede como descrito na Figura 22.

O módulo do RPR desenvolvido até ao momento pretende funcionar sobre um STM-4 a 622Mbit/s. Compreendendo que um link STM16 consegue agregar quatro links STM4, concluímos que o sistema pode ser incluído na rede representada na Figura 22. Acredita-se por isso na possibilidade, em termos futuros, de um incremento dos débitos do sistema RPR implementado. Contudo primeiro será necessário amadurecer o conhecimento desenvolvido neste protótipo RPR, para mais tarde desenvolver uma implementação com débitos ainda mais elevados. É de salientar também que o aparecimento de novas FPGAS é contínuo, cada vez com maiores capacidades e capazes de suportar frequências ainda mais elevadas, o que potencia um caminho fácil para evoluções futuras.

A interligação do módulo RPR com o anel pode ser feita com base em interfaces de linha SONET/SDH ou PacketPHY's. Nesta dissertação foram implementados os dois paradigmas de interface que a norma 802.17 compreende com o intuito de abarcar tanto quanto possível conhecimentos de como interligar o MAC do RPR em diferentes sistemas, acesso ao meio e garantindo sempre a sua compatibilidade. De uma forma semelhante, e considerando cenários de evolução algo diferentes, foram também desenvolvidas interfaces para sistemas ATM (ver secção 4.5). Embora não especificadas na norma 802.17, as interfaces ATM são em geral de grande interesse para sistemas de telecomunicações, pelo que também foram consideradas. Tal como no RPR, a tecnologia ATM também permite um melhor aproveitamento da largura de banda de um qualquer anel SDH quando introduzida no mesmo. Permite ter qualidade de serviço, tal como o RPR, entre outras vantagens, superando desta forma a rigidez da tecnologia SDH. Contudo, ao contrário da tecnologia RPR, a tecnologia ATM não foi desenvolvida tendo por base o paradigma de envio de pacotes. No entanto por se tratar de uma tecnologia antiga, e por isso bastante consolidada no mercado das telecomunicações, acaba por ser vantajoso a um qualquer operador investir neste tipo de tecnologia face ao RPR visto o risco ser inferior. Existe assim, alguma pressão de mercado para manter o ATM nas redes de transporte de dados. Eventualmente, um

operador de alguma dimensão e com alguma capacidade de investimento, poderá seguir no sentido de um sistema mais evolutivo como o RPR.

3.4 Projecto

O desenvolvimento feito na FPGA no que diz respeito ao RPR teve como resultado interfaces PacketPHY's capazes de suportar tráfego de pacotes Ethernet para o anel SDH. Para o cliente são também desenvolvidas interfaces Ethernet MII. No caso ATM a interface com o meio SDH baseia-se na recomendação I.432.1 do ITU [12]. É também utilizada uma interface UTOPIA bastante comum em sistemas electrónicos ATM, que interliga a FPGA a um Network Processor com diversas funcionalidades ATM.

A Figura 23 demonstra os encapsulamentos ocorridos numa trama enviada entre dois clientes distintos, que atravessa as interfaces, os módulos MAC e anel do sistema RPR. Suponhamos que o cliente do lado esquerdo da Figura 23 envia, por exemplo, uma trama de dados IP para o cliente do lado direito. A trama de dados começa por ser encapsulada numa trama Ethernet através do circuito integrado da INTEL representado pela Figura 23. Em seguida, a trama é encapsulada pela interface cliente numa trama RPR, é processada pelo MAC RPR, e é enviada para a interface física que a encapsula numa nova trama Ethernet. A trama viaja pelo anel e acaba por chegar ao cliente do lado direito da Figura 23. A interface física desencapsula o cabeçalho Ethernet obtendo-se a trama RPR enviada. A mesma é processada pelo MAC RPR, é desencapsulada pela interface cliente e pelo circuito integrado da INTEL chegando finalmente ao seu cliente de destino.

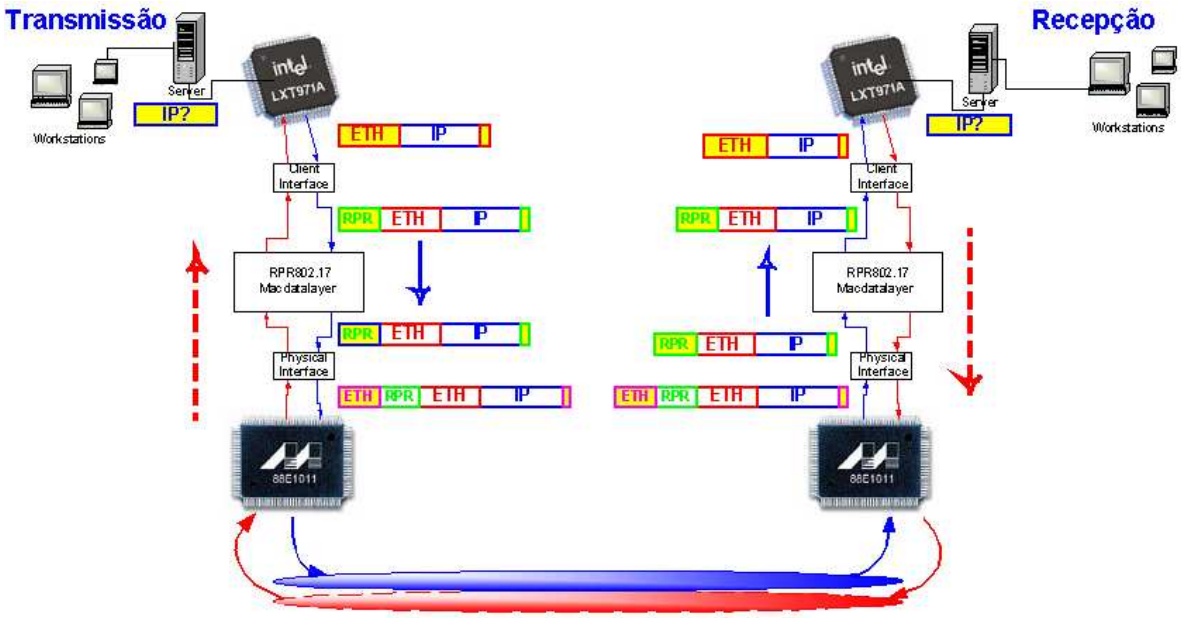


Figura 23 – Encapsulamentos ocorridos numa trama enviada entre dois clientes

No âmbito do projecto SIRAC a trama será enviada para o anel através de uma interface GMII implementada em FPGA sendo posteriormente mapeada num link STM4. Futuramente, existe a possibilidade de remover a parte SDH construindo um anel RPR e Ethernet apenas. Para já o projecto contempla um STM4 com uma largura de banda de 622Mbits.
O MAC do RPR irá receber/transmitir tramas Ethernet de e para o cliente através de uma interface MII. Esta interface está especificada na norma IEEE 802.3. Foi necessário desenvolver um módulo em FPGA (interface de cliente Rx) que recebesse as tramas vindas do cliente através da interface MII. Posteriormente o módulo desenvolvido encapsula a trama Ethernet recebida numa nova trama RPR e envia a mesma para o MAC do RPR através de uma interface SPI3. Foi também desenvolvido uma outra unidade (interface cliente Tx) que faz precisamente o inverso. Recebe as tramas RPR vindas do MAC, desencapsula a trama, recuperando assim uma trama Ethernet que é posteriormente enviada para o cliente através de uma interface MII. As interfaces de linha apresentadas na Figura 23 fazem exactamente o mesmo procedimento que as interfaces cliente, só que terão de trabalhar a débitos mais elevados e de ser compatíveis com a interface GMII. A secção seguinte apresenta uma descrição mais pormenorizada relativa às interfaces MII e GMII descritas na norma 802.3.

3.5 O MII e GMI

A expansão da Ethernet [2] [5] [6] [30] associada aos anos 80 deveu-se sobretudo à fácil instalação dos sistemas e significativa redução de custos [5]. Como tal no início dos anos 90 a Ethernet era já algo muito barato e fácil de instalar. Contudo as condições de mercado determinaram a necessidade de aumentar o ritmo de transmissão, de 10Mbit/s para 100 Mbit/s, mantendo as principais características da norma. Surge assim a Fast Ethernet. Foram assim feitas algumas modificações à norma que originaram o aparecimento da interface MII (Media Independent Interface) e a sub-camada de reconciliação, [2]. Estas duas entidades têm como principal missão substituir a interface AUI descrita na norma Ethernet a 10 Mbit/s.
O conceito de Ethernet a ritmos de transmissão da ordem do gigabit por segundo teve início no ano de 1995 [6], tendo-se formado novamente um grupo de estudo no seio do IEEE 802.3 [2]. Por esta altura deu-se um aumento significativo do número de equipamentos com capacidade de ligação 10/100 Mbit/s e consequente redução dos custos dos mesmos. Isto veio aumentar a necessidade de intervenção sobre os backbones e pontos de agregação das redes. A Gigabit Ethernet oferece o potencial para a realização de uma actualização ao core de uma rede de uma forma simples e eficiente. De uma forma similar ao ocorrido na transição de Ethernet a 10 Mbit/s para 100 Mbit/s houve mais uma vez a necessidade de alterar a interface MII. Optou-se pela criação de uma nova interface GMII (Gigabit Media Independent Interface) que não é mais que a interface MII com algumas modificações.

A Figura 24 ilustra o relacionamento do RPR, sub-camada de reconciliação e a camada física relativamente ao modelo OSI:

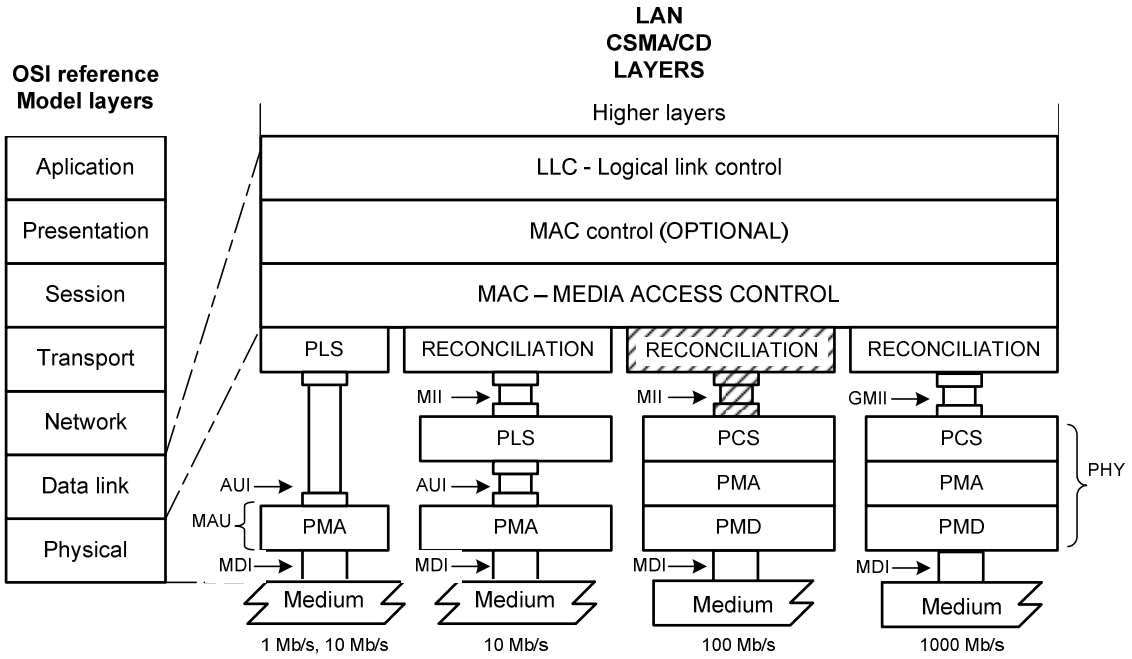


Figura 24 - Relacionamento do RPR RS e PHY relativamente ao modelo ISO/IEC OSI

3.5.1 Interface MII

A interface MII especifica as características dos sinais, conectores, e comprimentos de cabos, servindo assim de suporte para a informação transmitida da sub-camada de reconciliação para o PHY.

A sub-camada de reconciliação tem como função descodificar a informação enviada pelo MAC para a interface MII, de modo a que esta a consiga entender. Sob o ponto de vista do MAC, esta camada não tem qualquer utilidade, visto que apenas consegue traduzir os sinais enviados por este para a interface MII.

A interface MII possui um relógio de 25MHz juntamente com um barramento de dados com quatro bits de largura e alguns sinais de controlo. Os dados são transmitidos em sincronismo com o relógio, conseguindo-se velocidades de transferência de dados de 4bit x 25MHz = 100Mbit/s. Quando comparado com a interface AUI, a MII apresenta uma largura de banda 10 vezes superior. Fisicamente é necessário um conector de 40 pinos para fazer face ao fluxo de sinais que fluem em ambas as direcções para os sinais de controlo, de dados e relógio.

Enquanto que a interface AUI tem posicionar entre a camada que produz e recebe os sinais eléctricos (PLS) e a camada dependente do meio físico PMA (Physical Medium Attachment), a interface MII assegura uma maior independência do meio, separando claramente a camada data link da camada física.

A camada de sub reconciliação mapeia os sinais provenientes da interface MII para a camada PLS. A Figura 25 demonstra as entradas e saídas da sub-camada de reconciliação (os sinais encontram-se detalhados na secção 3.5.2).

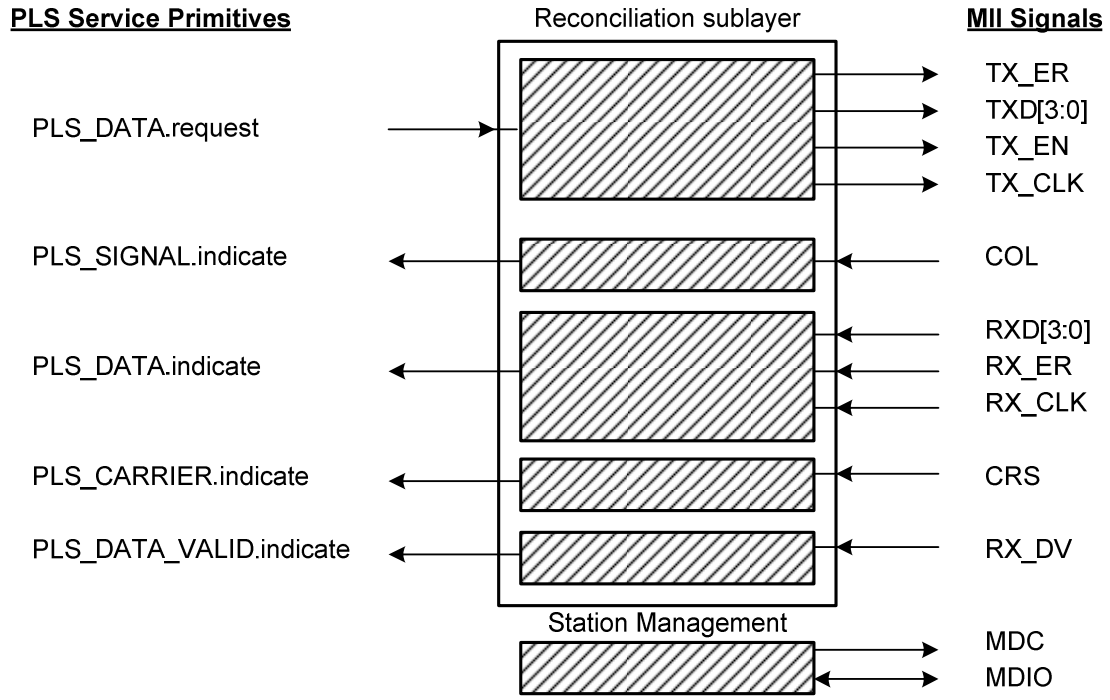


Figura 25 - Entradas e saídas da Reconciliation Sublayer (RS) para MII

O relógio de transmissão TX_CLK e recepção RX_CLK, proporciona os tempos de referência adequados para a comunicação entre a sub-camada de reconciliação e o PHY. Para uma transferência de dados a 100Mbit/s o relógio MII é de 25MHz, para 10Mbit/s o relógio é de 2,5Mbit/s.

As ligações de transmissão e recepção de dados na interface MII operam de uma forma independente, o que possibilita a operação em full duplex. Para usufruir das vantagens deste modo de funcionamento, também os componentes do PHY que se ligam à interface MII têm de suportar o modo de operação em full duplex.

As tramas de dados enviadas a partir da interface MII têm de ter o seguinte formato:

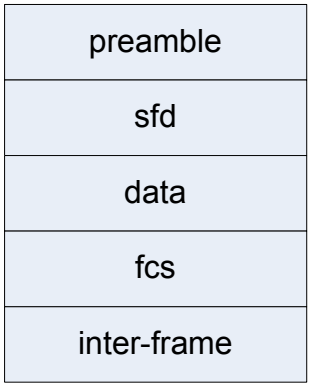


Figura 26 - formato de uma frame MII

Em que o campo preamble se refere a uma sequência usada para sincronização, seguida do campo sdf para sinalizar o inicio de pacote. O campo data contém os dados a transmitir enquanto que o fcs consiste no resultado de um algoritmo de crc calculado sobre o campo data. O inter-frame consiste num período de espera até que se inicie a transmissão de um novo pacote.
Cada octeto (byte) de dados tem de ser enviado em quatro bits de cada vez, devido ao barramento de dados da interface MII ter uma largura de apenas quatro bits.
Existe também um período entre tramas em que não são enviados dados chamado inter-frame, que resulta numa janela de observação. Este tempo é imposto para que exista um correcto funcionamento do MAC da Ethernet. É de salientar que o tempo deste período é dez vezes inferior na fast-Ethernet a 100Mbit/s quando comparado com a Ethernet a 10Mbit/s, tendo em conta a relação que existe entre relógios.

3.5.2 Interface GMII

A interface GMII está descrita na cláusula 35 da norma 802.3 do IEEE, e é baseada na interface MII definida na cláusula 22. De um modo similar à interface MII, a sub-camada de reconciliação (RS) tem de mapear os dados e respectivos sinais de controlo da interface GMII para a camada PLS. A interface GMII proporciona a interface entre o gigabit MAC e a camada física. A GMII e a sub-camada de reconciliação permitem que o gigabit MAC se conecte com diferentes PHYs. Contudo para que a gigabit Ethernet possa migrar para os equipamentos terminais como PC's etc., é necessário investigar soluções de baixo custos como o uso do meio físico UTP por exemplo.
A interface GMII é similar à interface MII tendo sofrido algumas modificações/alterações. O barramento de dados foi expandido de 4 bits para 8bits de largura. O relógio aumentou para 125MHz de modo a permitir um ritmo de transmissão de dados de 125Mhz*8bit=1000 Mbit/s.

A Figura 27 demonstra as entradas e saídas da sub-camada de reconciliação.

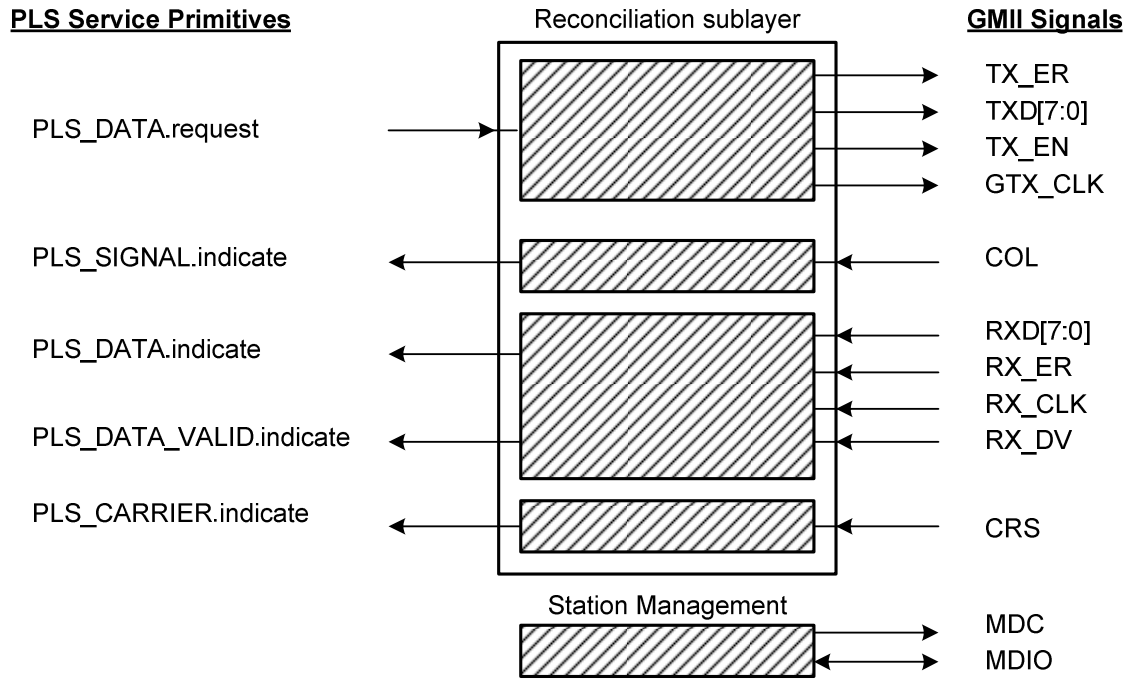


Figura 27 - Entradas e saídas da Reconciliation Sublayer (RS) para GMII

A interface GMII opera apenas com ritmos de transmissão de 1000 Mbit/s. Se for pretendido um PHY que opere também a outros ritmos de transmissão como 10/100 Mbit/s pode-se optar pelo uso da interface MII.

3.5.3 Interface MII e GMII – características dos sinais

Segue-se uma explicação sumária do comportamento e relação dos sinais que constituem a interface MII e a GMII.

3.5.3.1 TX_CLK (interface MII)

É um relógio contínuo que proporciona os tempos adequados para a transferência dos sinais TX_EN, TXD e TX_ER da sub-camada de reconciliação para o PHY. O sinal TX_CLK é amostrado pelo PHY. A frequência deste relógio deverá ser 25% do ritmo de transmissão nominal. Se o ritmo de transmissão do PHY for de 100Mb/s ou 10Mb/s o relógio devera ser de 25MHz ou 2.5MHz respectivamente. O duty cycle do relógio TX_CLK deverá ter um valor entre 35% e 65 inclusive.

3.5.3.2 RX_CLK (interface MII)

É um relógio contínuo que proporciona os tempos adequados para a transferência dos sinais RX_DV, RXD e RX_ER do PHY para a sub-camada de reconciliação. O PHY pode recuperar o relógio RX_CLK a partir dos dados recebidos ou pode deriva-lo a partir de um relógio local como o

TX_CLK por exemplo. Quando o sinal RX_DV tiver no estado activo o relógio RX_CLK deverá ser síncrono com os dados recebidos. A frequência deverá ser de 25% do ritmo de transmissão e o duty cycle de 35% ou 65% inclusive. No caso de haver perda do sinal recebido, o relógio deverá ser obtido a partir de um relógio local e não através dos dados recebidos.

3.5.3.4 GTX_CLK (interface GMII)

É um relógio contínuo usado para a operação a 1000Mbit/s. Proporciona os tempos adequados para a transferência dos sinais TX_EN, TX_ER e TXD da sub-camada de reconciliação para o PHY. Os valores presentes nos sinais TX_EN, TX_ER e TXD são amostrados pelo PHY no flanco ascendente do relógio GTX_CLK. A frequência nominal do relógio GTX_CLK é de 125MHz.

3.5.3.5 RX_CLK (interface MII e GMII)

O relógio RX_CLK proporciona os tempos adequados para a transferência dos sinais RX_DV, RX_ER e RXD da sub-camada de reconciliação para o PHY. Os valores presentes nos sinais RX_EN, RX_ER e RXD são amostrados pela sub-camada de reconciliação no flanco ascendente do relógio RX_CLK. O relógio RX_CLK é enviado pelo PHY. O PHY pode recuperar o relógio RX_CLK através dos dados recebidos ou pode derivar o mesmo utilizando um relógio local de referência, como por exemplo o relógio GTX_CLK recebido. No caso de perda de sinal e do relógio ser recuperado a partir dos dados, o relógio terá de ser extraído através de um relógio local. A frequência nominal do relógio RX_CLK é de 125MHz neste caso.

3.5.3.6 TX_EN (MII e GMII)

O sinal de controlo TX_EN serve para indicar a presença de dados válidos no barramento de dados TXD enviados pela sub-camada de reconciliação para transmissão.

Este sinal é activado pela sub-camada de reconciliação de uma forma síncrona com o primeiro octeto do preâmbulo da trama a ser transmitida. Permanecerá activo durante a transmissão dos dados sendo desactivado a quando da transmissão do ultimo grupo de dados presentes em TXD da trama enviada. A Figura 28 ilustra o funcionamento do sinal TX_EN no caso da interface GMII sendo semelhante para a interface MII.

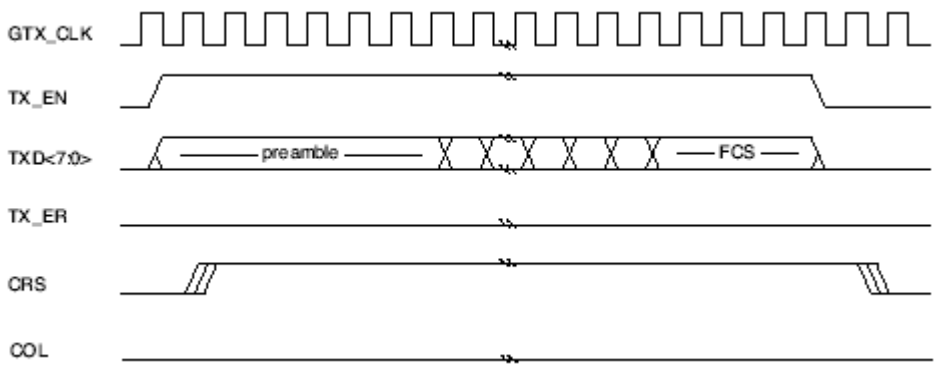


Figura 28 - transmissão simples de uma trama

3.5.3.7 TXD (MII e GMII)

Representa um conjunto de quatro ou oito sinais que constituem um barramento de dados por onde estes dados serão transmitidos, vindos da sub-camada de reconciliação (TXD<3:0> no caso da interface MII e TXD<7:0> no caso da interface GMII). Estes sinais deverão transitar de uma forma síncrona de acordo com o relógio TX_CLK ou GTX_CLK, no caso da interface MII ou GMII respectivamente. O sinal TX_EN deverá permanecer activo, ao invés do sinal TX_ER que estará desactivado quando da transmissão de dados validos para o PHY.

No caso de estarem ambos os sinais TX_EN e TX_ER desactivados, TXD não terá qualquer efeito sobre o PHY.

Quando TX_EN tiver desactivado e TX_ER activo, TXD será utilizado para indicar ao PHY a geração de carrier extend ou códigos de erros de grupo do tipo carrier extend. Com a introdução da norma 802.3z para Gigabit Ethernet em 1998, foi adicionado o campo extension, suficientemente comprido para que uma ocorrência de colisão fosse propagada para todas as estações na rede. O modo carrier extend utiliza-se apenas quando as interfaces estão configuradas em half duplex. Nesta dissertação de mestrado as interfaces MII e GMII serão configuradas em full duplex. A Tabela 14 especifica a relação dos sinais TXD<3:0>, TX_EN e TX_ER para a interface MII e a Tabela 15 para a GMII.

TX_EN	TX_ER	TXD[3:0]	Indicação
0	0	0 até F	Período entre pacotes
0	1	0 até F	Reservado
1	0	0 até F	Transmissão de dados normal
1	1	0 até F	Transmissão de propagação de erro

Tabela 14 - relação entre os sinais TXD<3:0>, TX_EN e TX_ER

TX_EN	TX_ER	TXD[7:0]	Description	PLS_DATA.request parameter
0	0	00 até FF	Período entre pacotes	TRANSMIT_COMPLETE
0	1	00 até 0E	Reservado	_
0	1	0F	Carrier Extend	EXTENDED (oito bits)
0	1	10 até 1E	Reservado	_
0	1	1F	Erro de Carrier Extend	EXTENDED_ERROR (oito bits)
0	1	20 até FF	Reservado	_
1	0	00 até FF	Transmissão de dados normal	ZERO, UM (oito bits)
1	1	00 até FF	Transmissão de propagação de erro	Nenhum parâmetro aplicável

Tabela 15 - relação entre os sinais TXD<7:0>, TX_EN e TX_ER

3.5.3.8 TX_ER (MII e GMII)

O sinal TX_ER é gerado pela sub-camada de reconciliação, transitando de uma forma síncrona com o relógio TX_CLK(MII) ou GTX_CLK(GMII). Se o sinal TX_ER for activado durante um ou mais períodos de relógio estando também o sinal TX_EN activado, o PHY deverá responder com um ou mais códigos de grupos que não constituem dados válidos da trama a ser transmitida. A Figura 29, ilustra o comportamento do sinal TX_ER ao longo da transmissão de uma trama com propagação de erros no caso da interface GMII.

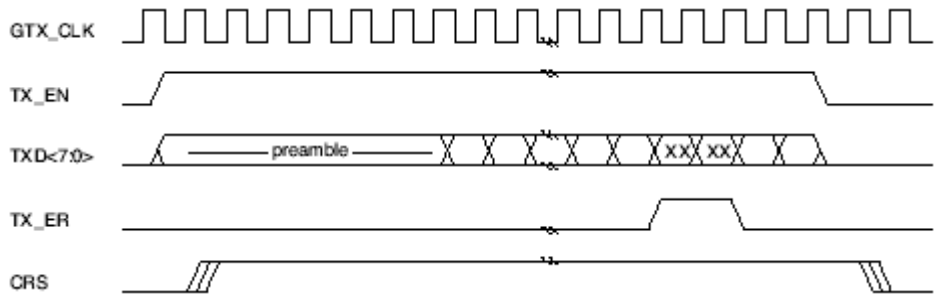


Figura 29 - transmissão de uma trama com erros

No caso do sinal TX_EN desactivo e TX_ER activo, a inserção de valores apropriados no barramento de dados TXD irá causar a geração de Carrier Extend ou de códigos de erros de grupo Carrier Extend por parte do PHY.

A transição de TX_EN de um estado activo e TX_ER desactivo para TX_EN desactivo e TX_ER activo, tendo TXD a indicação de carrier extend, deverá resultar na transmissão de uma indicação de fim de pacote por parte do PHY e de inicio de códigos de grupos de carrier extension. A Figura

30 descreve o comportamento do sinal TX_ER da interface GMII durante a transmissão de um carrier extension.

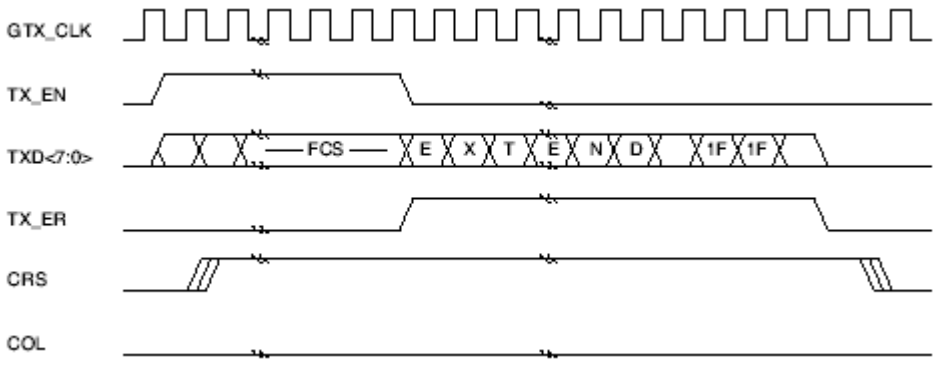


Figura 30 - Propagação de um erro com Carrier Extension

A propagação de um erro com carrier extension ocorre com TX_EN no estado desactivo, TX_ER activo e tendo TXD um valor apropriado. A transmissão de tramas em bursts também utiliza carrier extension entre os bursts das tramas transmitidas. A Figura 31 demonstra o comportamento dos sinais TX_ER e TX_EN durante a transmissão de bursts no caso da interface GMII.

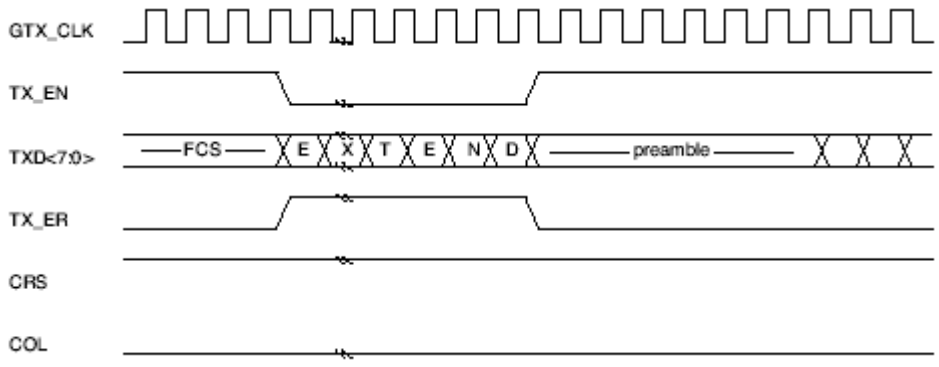


Figura 31 - Transmissão de burst

3.5.3.9 RX_DV (MII e GMII)

O sinal RX_DV é gerado pelo PHY para indicar a presença de dados validos no barramento de dados RXD. Este sinal é activado de uma forma síncrona com os primeiros dados do preâmbulo da trama a ser transmitida. Permanecerá activo durante a transmissão dos dados sendo desactivado a quando da transmissão dos últimos dados da trama a ser enviada. A Figura 32 ilustra o funcionamento do sinal RX_DV para a interface GMII.

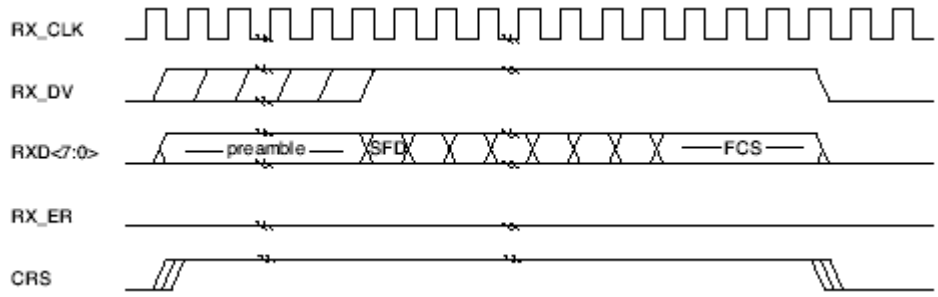


Figura 32 - Recepção simples de uma trama

3.5.3.10 RXD (MII e GMII)

O sinal RXD representa um conjunto de sinais servindo de suporte ao envio de dados do PHY para a sub-camada de reconciliação. Estes sinais deverão transitar de uma forma síncrona de acordo com o relógio RX_CLK. A Figura 32 ilustra a recepção pela sub-camada de reconciliação de uma trama vinda do PHY. Quando o sinal RX_DV estiver desactivado, o PHY pode disponibilizar algumas indicações para a sub-camada de reconciliação, activando o sinal RX_ER e colocando em RXD um valor de acordo com os valores da Tabela 16 e Tabela 17 no caso da interface MII e GMII respectivamente.

A Figura 33 ilustra a recepção pela sub-camada de reconciliação de uma trama com carrier extension.

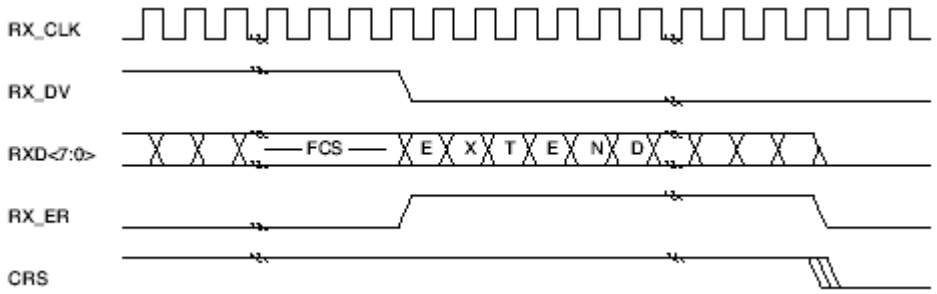


Figura 33 - Recepção de uma trama com carrier extension

A Figura 34 demonstra o comportamento dos sinais RX_ER e RX_DV durante a recepção de burst de tramas.

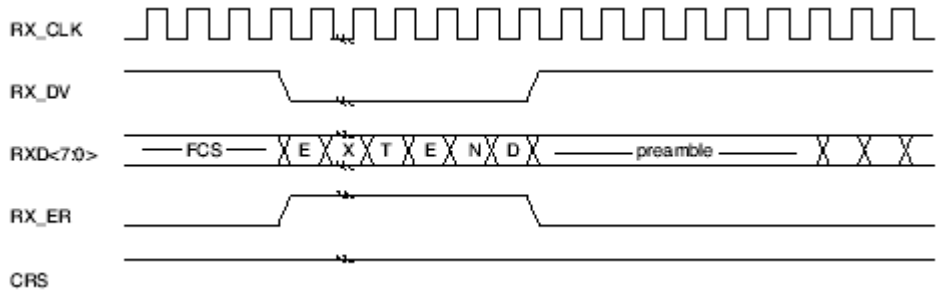


Figura 34 - Recepção de burst's

RX_DV	RX_ER	RXD[3:0]	Indication
0	0	0000 through 1111	Normal inter-frame
0	1	0000	Normal inter-frame
0	1	0001 through 1101	Reserved
0	1	1110	False Carrier indication
0	1	1111	Reserved
1	0	0000 through 1111	Normal data reception
1	1	0000 through 1111	Data reception with errors

Tabela 16 - relação entre os sinais RXD<3:0>, RX_DV e RX_ER

RX_DV	RX_ER	RXD[7:0]	Description	PLS_DATA.indicate parameter
0	0	00 through FF	Normal inter-frame	No applicable parameter
0	1	00	Normal inter-frame	No applicable parameter
0	1	01 through 0D	Reserved	_
0	1	0E	False Carrier indication	No applicable parameter
0	1	0F	Carrier Extend	EXTEND (eight bits)
0	1	10 through 1E	Reserved	_
0	1	1F	Carrier Extend Error	ZERO, ONE (eight bits)
0	1	20 through FF	Reserved	_
1	0	00 through FF	Normal data reception	ZERO, ONE (eight bits)
1	1	00 through FF	Data reception error	ZERO, ONE (eight bits)

Tabela 17 - relação entre os sinais RXD<7:0>, RX_DV e RX_ER

3.5.3.11 RX_ER (MII e GMII)

O sinal RX_ER é gerado pelo PHY, transitando de uma forma síncrona com o relógio RX_CLK. Se o sinal RX_ER for activado durante um ou mais períodos de relógio estando também o sinal RX_EN activado, poderão ser enviadas algumas indicações para a sub-camada de reconciliação. A Figura 35 ilustra o comportamento do sinal RX_ER durante a recepção de uma trama com erros.

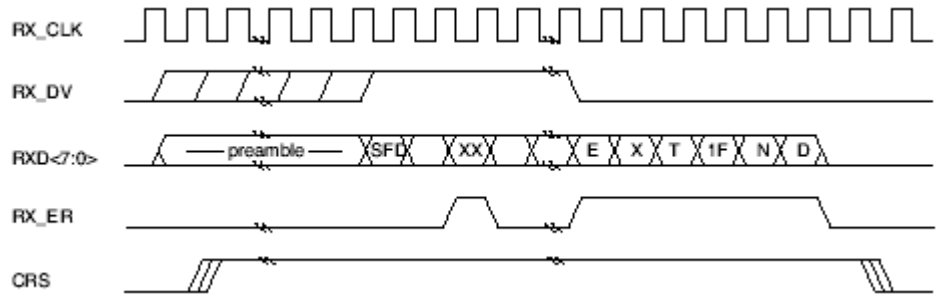


Figura 35- Exemplo de recepção com erros

3.5.3.12 CRS (MII e GMII)

No modo de operação em half duplex, o PHY irá activar este sinal no caso de o meio não estar no estado de IDLE e desactiva-lo em caso contrário. O PHY deverá garantir que o sinal CRS permanece activo no decorrer de uma condição de colisão.

O sinal CRS não necessita de ser síncrono com nenhum dos relógios TX_CLK/GTX_CLK ou RX_CLK. O funcionamento do sinal CRS não está especificado no caso do PHY operar em full duplex.

A Figura 36 e Figura 37, ilustram o comportamento do sinal no caso de ocorrer uma colisão durante a transmissão de uma trama.

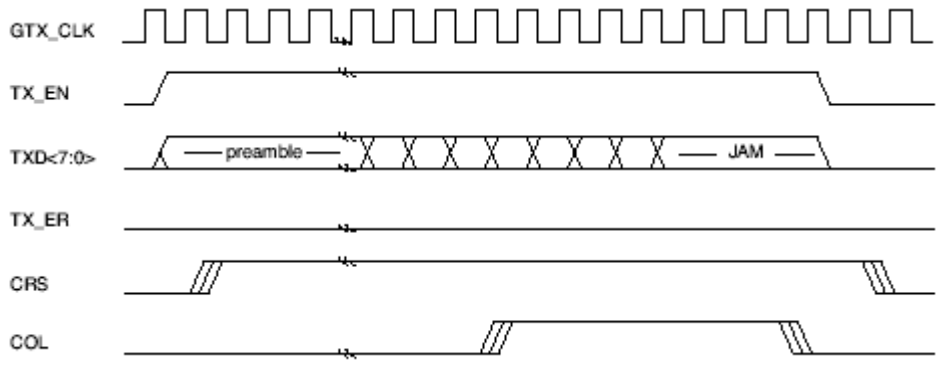


Figura 36 - Transmissão com colisão

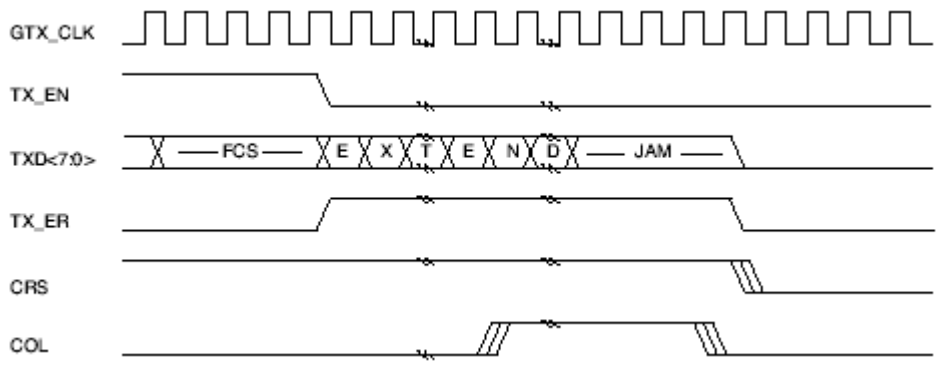


Figura 37 - Transmissão com colisão com carrier extension

O sinal de COL deverá ser activado pelo PHY quando for detectada uma qualquer colisão no meio e deverá permanecer activo enquanto a situação de colisão persistir. Não necessita ser síncrono com qualquer relógio TX_CLK/GTX_CLK ou RX_CLK. O funcionamento do sinal COL não está especificado no caso do PHY operar em full duplex.
A Figura 36 e Figura 37, ilustram o comportamento do sinal COL durante a transmissão de uma trama com uma colisão.

3.6 Interface ATM - UTOPIA

A interface UTOPIA é bastante comum em sistemas ATM facilitando a interligação entre diversos componentes electrónicos necessários para um sistema desta natureza. Na placa ATM desenvolvida para a rede da Figura 22, foi necessário o uso desta interface para interligar a FPGA a um Network Processor com capacidades ATM. O Network Processor irá fazer todo o trabalho inerente às funcionalidades ATM da placa como por exemplo a comutação entre diferentes VP/VC's, diferentes quantificações de qualidade de serviço, estatísticas da rede, etc. A FPGA terá de extrair/inserir as células ATM resultantes de link's STM-1 existentes na placa e em seguida enviar/receber as células do Network Processor através da interface UTOPIA. É de salientar que a interface pressupõe a existência de um "master" que neste caso será o Network Processor, e de um "slave", que será o módulo ATM presente na FPGA. Assim sendo, sempre que o Network Processor quiser enviar uma célula ATM para a FPGA, primeiro terá de averiguar se existe espaço suficiente na mesma. Do mesmo modo para receber correctamente as células ATM vindas da FPGA, terá de um modo periódico e sistemático averiguar a existência das mesmas nos FIFOS da FPGA. Note-se que na interface UTOPIA existem N FIFOS de células ATM e que cada um deles corresponde a um qualquer canal ATM existente entre a FPGA e o Network Processor. No nosso caso existem dois canais ATM entre os dispositivos que correspondem a cada um dos dois links STM-1 existentes na placa.

O funcionamento da interface UTOPIA é ilustrado na Figura 38:

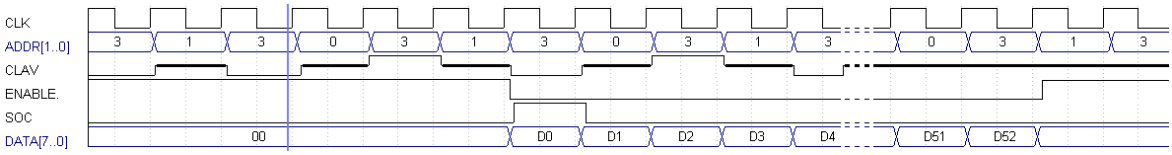


Figura 38 - Interface ATM UTOPIA

Note-se a existência de um barramento de endereços e de um sinal chamado CLAV. O barramento de endereços é um input na FPGA e um output no Network Processor, enquanto que o CLAV é um output na FPGA e input no Network Processor. O barramento de endereços identifica um dos dois canais a utilizar pelo Network Processor.

Sempre que este necessita de enviar uma célula para um dos dois canais da FPGA, primeiro põe no barramento de endereços um valor que identifica o canal. No nosso caso será zero (“00”) para o primeiro STM-1 e um (“01”) para o segundo STM-1. Caso haja espaço no FIFO para receber uma nova célula, a FPGA irá responder afirmativamente pondo o sinal CLAV a um. Note-se que a saída do sinal clav ocorre com um clock de latência. Neste preciso momento o barramento de dados possui o valor NULL, três (“11”), que não representa nenhum dos canais.

Quando o Network Processor pretende receber células da FPGA, o processo é exactamente o mesmo, só que neste caso o valor do sinal CLAV apenas transite para um quando a FPGA tiver células a enviar. Note-se que a responsabilidade de ler as células correctamente da FPGA, não permitindo que os FIFOS encham evitando a perda de células, é do Network Processor visto este ser considerado o “master”. No envio das células do Network Processor para a FPGA os sinais SOC e ENABLE são gerados pelo mesmo assim como o barramento de dados. Assim o Network Processor põe o sinal ENABLE a zero durante 53 ciclos seguidos pondo também o sinal SOC a um no primeiro. Quando é a FPGA a enviar as células o SOC é gerado pela FPGA bem com os dados. Contudo o ENABLE continua a ser gerado pelo Network Processor que assim que o CLAV vem a um põe o ENABLE a zero e com um clock de latência envia o SOC a um e inicia a transferência da célula através do barramento de dados. Note-se que em ambos os casos na ultima vez que o ENABLE teve a um no envio de uma célula, o valor presente no barramento de endereços indica qual o canal a que corresponde a célula.

4 Implementação e testes de interfaces

4.1 Implementação em FPGA

Neste trabalho houve a necessidade de escolher um dispositivo adequado às nossas necessidades. Para isso foram estudados dispositivos de alguns fabricantes de FPGAs lideres no mercado sendo escolhidas as FPGAS XC2VP70 e XC2V3000 da Xilinx. Possuem uma vasta diversidade de componentes no seu interior como iremos constatar em seguida. São dispositivos bastante idênticos com a principal diferença de a FPGA XC2VP70 possuir dois processadores internos. Além da necessidade da presença destes componentes, os tempos de computação e capacidade do dispositivo foram igualmente propriedades muito importantes a ter em conta na escolha do mesmo.

4.1.1 Características da FPGA

As FPGAs escolhidas possuem uma estrutura comum relativamente à maioria das FPGAS da Xilinx mais recentes. Apresentamos de seguida alguns dos componentes presentes nesta tecnologia bem como uma breve explicação de cada um dos mesmos.

4.1.1.1. CLB's

Os blocos elementares de uma FPGA são os CLBs. Os CLBs (configurable logic blocs) são recursos da FPGA que permitem a implementação de circuitos de lógica sequencial ou combinatória. Uma FPGA é constituída por um array de CLBs ligados entre si por intermédio de uma matriz de roteamento. Na escolha de uma FPGA, para além do número de Blocos de RAM assim como de outras funcionalidades, o principal requisito prende-se com o número de CLBs que a FPGA possui. Quanto mais CLBs a FPGA tiver, maior poderá ser o número/complexidade de funções lógicas a implementar. Cada CLB é constituído por dois pares de SLICES interligados entre si, dois SLICEM e dois SLICEL como se pode ver na Figura 39:

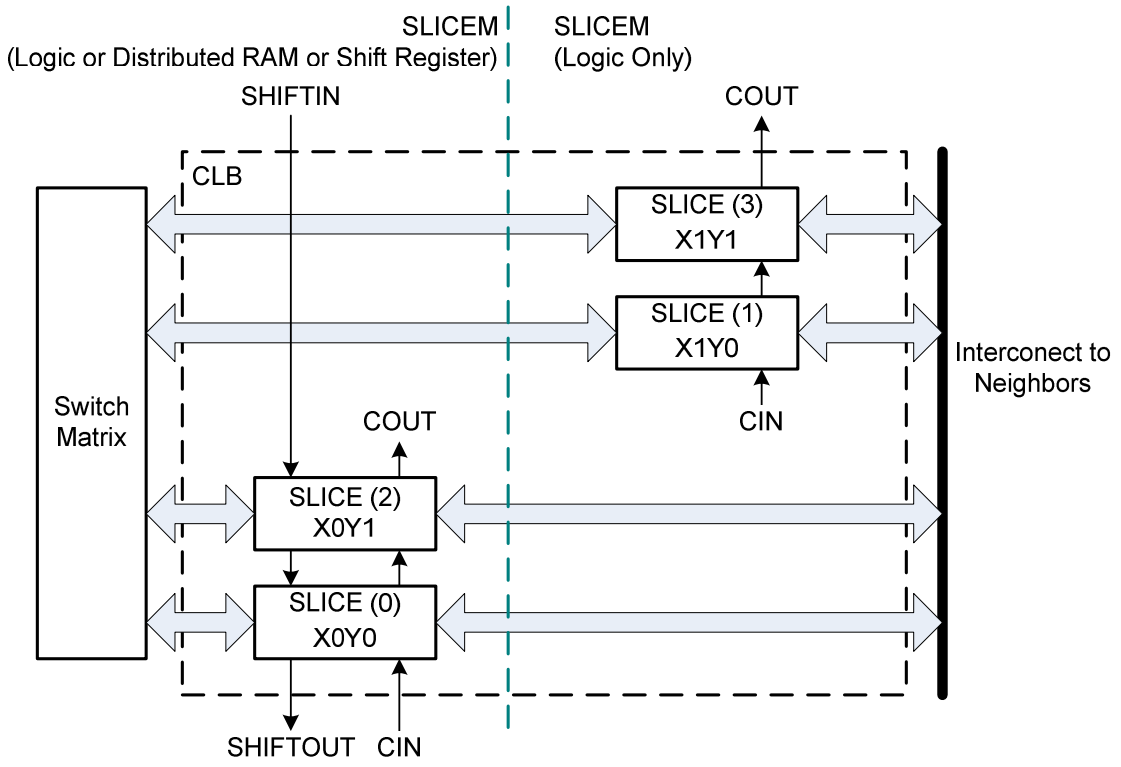


Figura 39 - Estrutura de um CLB

4.1.1.2. SLICE's

Os elementos comuns a cada um dos SLICE's são duas unidades capazes de implementar funções lógicas denominadas de LUT's (look-up tables), duas unidades de armazenamento, multiplexeres, carry logic, e algumas gates aritméticas. Estes elementos são utilizados por ambos os SLICEM e SLICEL de forma a gerar funções lógicas, de ROM, e aritméticas. Os SLICEM para além disso suportam o armazenamento de dados através de RAM distribuída e a capacidade de deslocar dados usando registos de 16 bit's. A Tabela 18 descreve os elementos que constituem um SLICE:

Slices	LUTs	Flip-Flops	MULT_ANDs	Arithmetic & Carry-Chains	Distributed RAM(1)	Shift Registers(1)
4	8	8	8	2	64 bits	64 bits

Tabela 18 - elementos que constituem um slice (SLICEM)

Os diagramas dos SLICEM e SLICEL estão descritos em detalhe no anexo A e anexo B respectivamente.

4.1.1.3. LUT's

As LUT's (look-up tables) são elementos capazes de implementar qualquer função booleana. Possuem quatro entradas distintas. As diferentes entradas e saídas das LUT's ligam a multiplexers presentes nos CLB's e podem ser combinadas de diversas formas, quer no mesmo CLB ou em CLB's distintos, conseguindo gerar funções lógicas de complexidade variável.

4.1.1.4. FLIP FLOP's

Os SLICE's possuem dois flip-flop's que podem ser configurados para funcionarem como flip-flop's do tipo D ou como latch como ilustrado na Figura 40. A entrada dos flip-flop's pode provir das saídas das LUT's ou directamente das entradas do SLICE. Os sinais de controlo clock (CLK), clock enable (CE) e set/reset (SR) são comuns a ambos os flip-flop's pertencentes ao mesmo SLICE e podem apresentar polaridades diferentes. O sinal de clock enable CE é activo a um por defeito e no caso deste não ser utilizado permanecerá activo, viabilizando a utilização do flip-flop. As entradas D dos flip-flops FFY ou FFX representados na Figura 40, serão ligadas às saídas das LUT's presentes no SLICE ou então às entradas do SLICE BY ou BX respectivamente. A selecção é feita através de dois miltiplexeres presentes no SLICE.

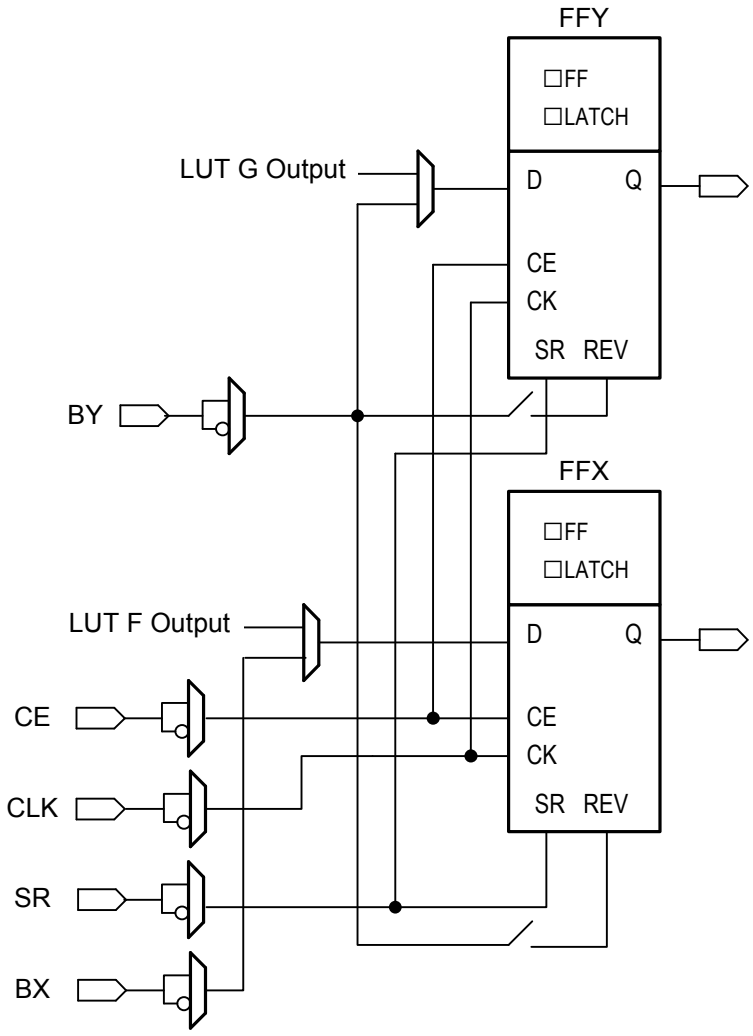


Figura 40 - flip-flop's pertencentes a um SLICE

4.1.1.4. RAM e memória distribuída

As LUT´s podem também ser configuradas como elementos de memória distribuída, sendo esta característica inerente apenas às LUT's de SLICE's do tipo SLICEM. Podem assim formar RAM´s síncronas de 16x1-bit. Por sua vez estas RAM's podem ser combinadas dentro de um mesmo CLB formando:

- Single-Port 16x4-bit RAM
- Single-Port 32x2-bit RAM
- Single-Port 64x1-bit RAM
- Dual-Port 16x2-bit RAM

Os elementos de RAM distribuída são recursos de escrita síncrona e leitura assíncrona. No caso de ser pretendido uma leitura também síncrona, a RAM distribuída pode ser combinada com um

flip-flop do mesmo SLICE para o efeito. O flip-flop e a RAM distribuída partilham o mesmo sinal de CLK de entrada. Numa operação de escrita o sinal Write Enable (WE), derivado do pino SR, deverá ser colocado a um. A Tabela 19 demonstra o número de LUT's ocupadas por cada configuração de RAM distribuída.

RAM	Number of LUTs
16 x 1S	1
16 x 1D	2
32 x 1S	2
64 x 1S	4

Tabela 19 - número de LUT's ocupadas por configuração de RAM

Nota: **1.** S = configuração single-port; D = configuração dual-port

Para configurações single-port, as RAM's distribuídas (LUT's) possuem um porto de endereço comum para escritas síncronas e leituras assíncronas. No caso de dual-port, as RAM's distribuídas possuem um porto para escritas síncronas e leituras assíncronas e um outro porto para leituras assíncronas. Assim sendo as LUT's possuem endereços de escrita e de leitura distintos.

No modo single-port os endereços de escrita/leitura partilham o mesmo bus de endereços que corresponde a A[3:0] na Figura 41 e A[4:0] na Figura 42. O dado de entrada é representado pelo sinal D, e é escrito na RAM quando o sinal WE for activo na transição do relógio WCLK. A saída pode ser lida directamente da RAM ou através de um flip-flop com um ciclo de relógio de latência.

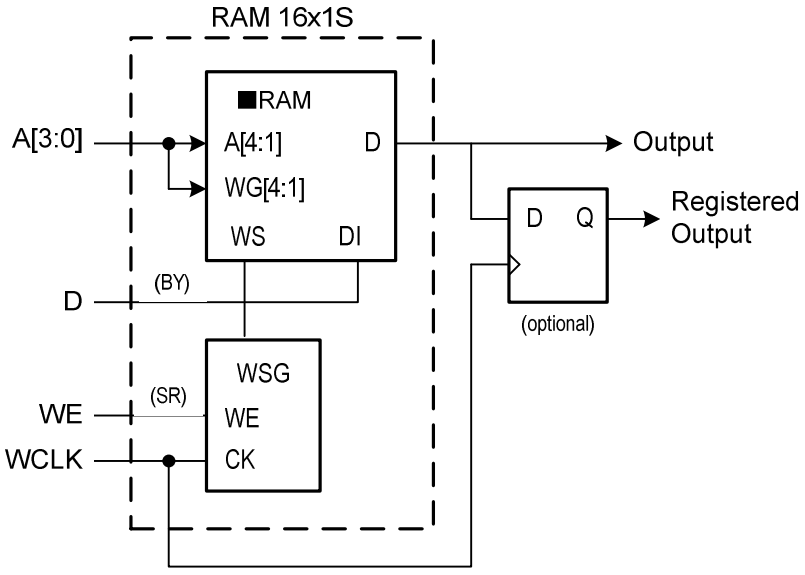


Figura 41 - Ram distribuída (RAM16x1S)

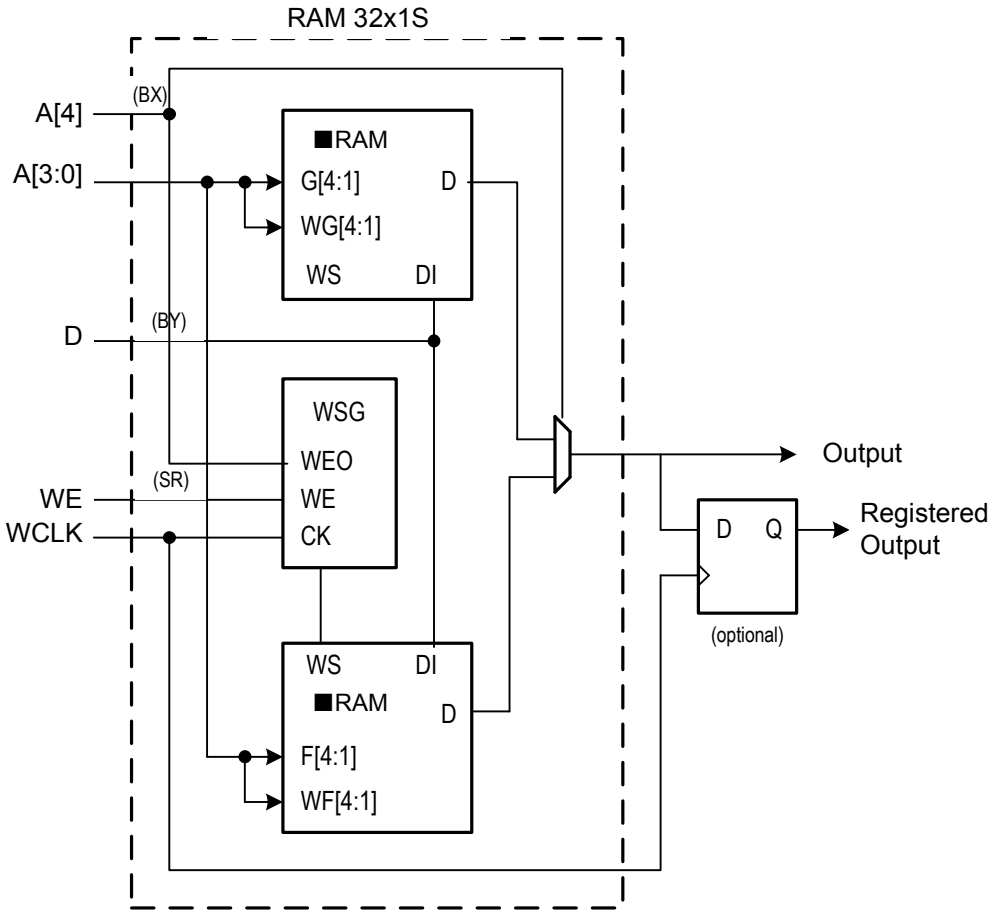


Figura 42 - Ram distribuída single-port (RAM32x1S)

Em dual port-mode (Figura 43) os endereços de escrita e leitura na primeira LUT são partilhados. Na segunda LUT o porto de escrita estará também ligado ao barramento de endereços de leitura/escrita da primeira LUT enquanto que o endereço de leitura será o segundo porto de leitura da DPRAM e é dado pelo barramento DPRA[3:0].

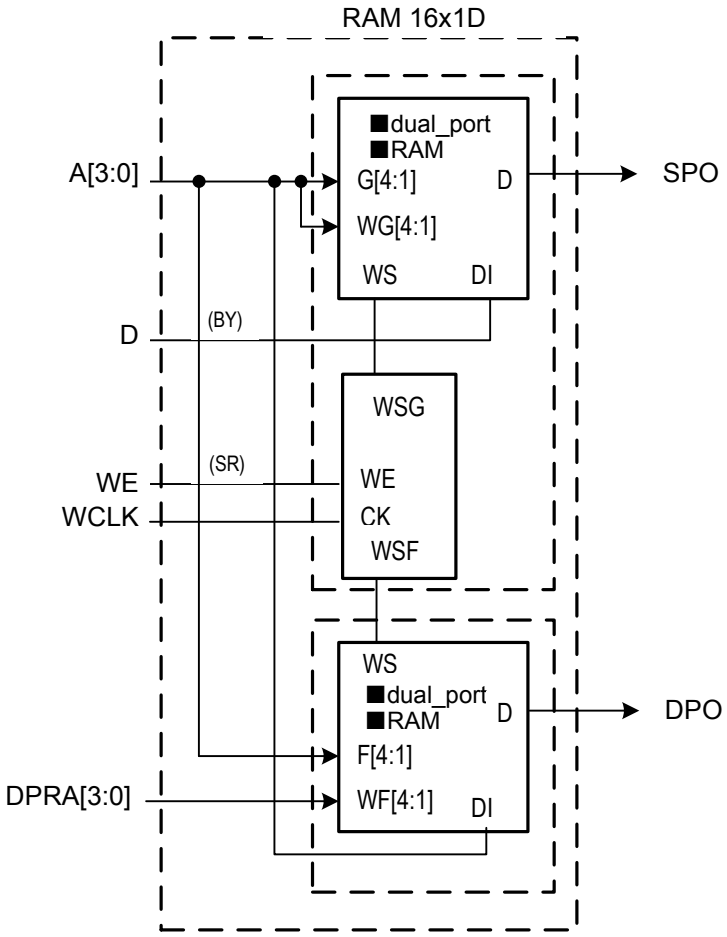


Figura 43 - Ram distribuída dual-port (RAM16x1D)

Como podemos ver pela Figura 43, uma dual port de 16x1 bit ocupa sempre duas LUT's. Como tal, tendo cada CLB 4 LUT's deste tipo, podemos realizar dois módulos DPRAM de 16x1 bit por cada CLB deste que os mesmos partilhem o mesmo relógio e write enable.

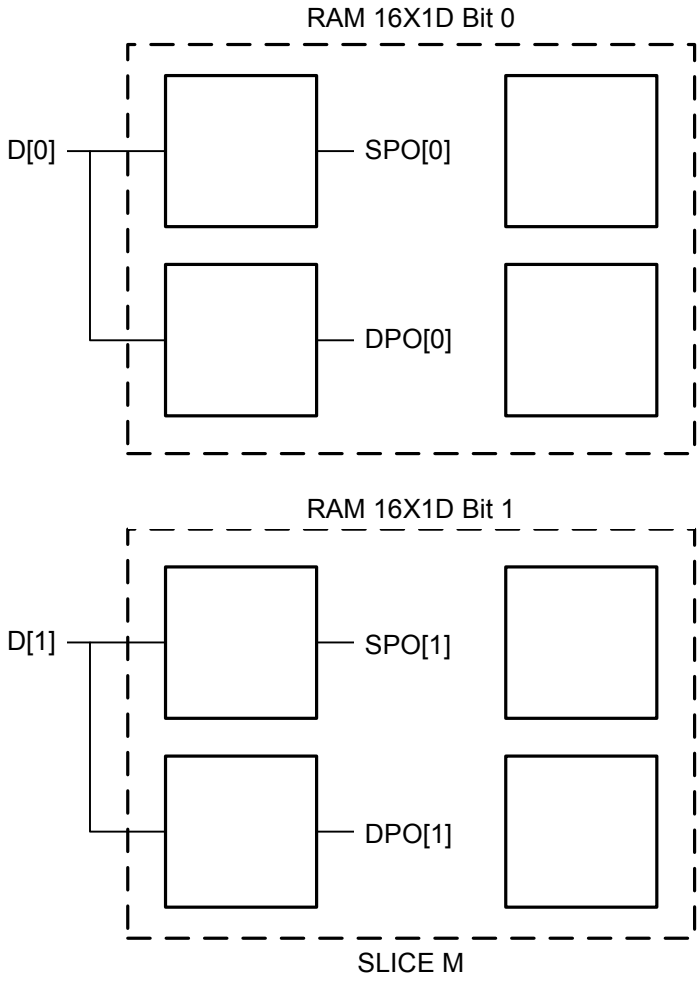


Figura 44 - Mapeamento de duas RAM16x1D

4.1.1.5. Dispositivo escolhido

Para esta dissertação de mestrado foram escolhidas as FPGAs XC2VP70 e XC2V3000 da Xilinx. As principais características destes dispositivos são apresentadas na Tabela 20:

Device	RocketI/O Transceiver Blocks	PowerPC Processor Blocks	Logic Cells	Slices	MaxDistr RAM(Kb)	18x18Bit Multiplier Blocks	18Kb Blocks	DCM's	Maximum User I/O Pads
XC2VP70	16 or 20	2	74,448	33,088	1,034	328	328	8	996
XC2V3000	0	0	32,256	14,336	0,448	96	96	12	720

Tabela 20 - Principais características das FPGAs XC2VP70 e XC2V3000 da Xilinx

4.2. Metodologia do projecto

Numa tentativa de primeira aproximação, as interfaces foram desenvolvidas utilizando uma linguagem de mais alto nível, SystemC, que as linguagens VHDL/Verilog. Esta linguagem implica uma maior abstracção do Hardware permitindo-nos descrever o sistema numa primeira fase. Contudo a linguagem SystemC não é sintetizável em FPGA, sendo necessário fazer uma conversão do código para linguagem HDL. A ferramenta Design Compiler da Synopsys referenciada na Figura 45 faz a conversão de código SystemC em código HDL (Verilog/VHDL) capaz de ser sintetizável numa FPGA. Verificou-se que a conversão não é directa na maioria dos casos, mas com algum conhecimento das linguagens de programação Verilog/VHDL, consegue-se através de pequenos ajustes sobre o código obter os resultados pretendidos. De seguida, utilizando a ferramenta FPGA Compiler II da Synopsys [18] fizemos um mapeamento do código HDL convertido para a FPGA de destino. É também necessário definir alguns constrangimentos ao código, como frequência mínima pretendida para os relógios, utilização de blocos de memória da FPGA ou de memória distribuída, entre outros. Este software gera um ficheiro com extensão .edif que será introduzido no software ISE [19] da Xilinx para fazer o Place&Route. Resulta então um ficheiro de extensão .jed para programação do FPGA, bem como alguns relatórios com a informação dos tempos atingidos para os relógios, espaço ocupado com lógica relacionada, número de blocos de memória utilizados, LUT´s usadas como memória distribuída, entre outros.

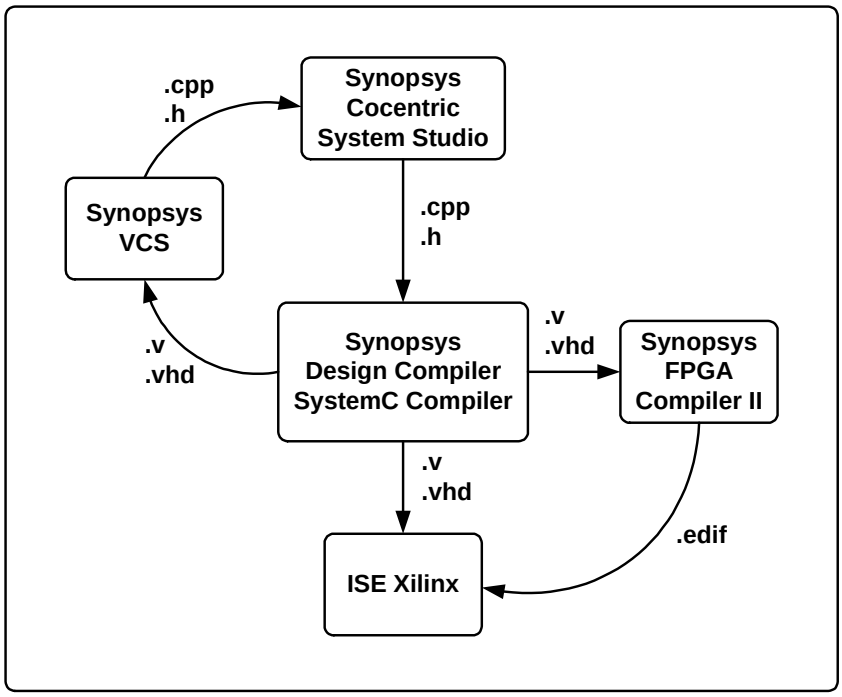


Figura 45 - Workflow para síntese em FPGA

As interfaces são constituídas por máquinas de estados, FIFO's e unidades de CRC (ciclic redundancy check). As máquinas de estados são modeladas por dois processos, um sequencial e um combinatório, em que o primeiro é sensível ao relógio e o segundo a uma variável que possui o valor do estado. No processo combinatório houve o cuidado de atribuir valores adequados a todas as saídas nos diferentes estados garantindo que nenhuma saída ficasse com valor indefinido. Na necessidade de armazenamento de dados entre os diferentes estados, estes valores são guardados através do processo sequencial na transição do relógio declarada na lista de sensibilidade do processo. Como ilustração, em seguida apresenta-se o código HDL da máquina de estados sm_client.vhd da interface cliente Rx.

```
0   ----------------------------------------------------------------------
1   -- sm_client.vhd
2   -- Recebe uma trama Ethernet com interface MII
3   -- retira os dados da trama e escreve-a num FIFO
4   ----------------------------------------------------------------------
5
6   LIBRARY ieee;
7   USE ieee.std_logic_1164.all;
8   USE ieee.std_logic_arith.all;
9   Use ieee.std_logic_unsigned.All;
10
11  Entity sm_client Is
12    Port(
13      RESET                 : IN  Std_Logic;
14      Clk                   : IN  Std_Logic;
15      ------- ETHERNET-------------
16      RXD                   : IN Std_Logic_Vector (3 Downto 0);
17      RX_EN                 : IN Std_Logic;
18      RX_ER                 : IN Std_Logic;
19      ------- FIFO ----------------
20      DATA_FIFO             : OUT  Std_Logic_Vector (35 Downto 0);
21      WR_EN                 : OUT  Std_Logic
22      ----------------------------
23    );
24  End sm_client;
25
26  Architecture sm_client Of sm_client Is
27  ---------------- Contador ----------------------------
28  signal NxCnt, Cnt : Std_Logic_Vector (4 Downto 0);
29  signal NxCntSReg, CntSReg : Std_Logic_Vector (5 Downto 0);
30  ---------------- Shift Reg ------------------------
31  signal NxShiftReg, ShiftReg : Std_Logic_Vector (123 Downto 0);
32  ---------------------------------------------------------
33  signal NxRMOD_FF, RMOD_FF : Std_Logic_Vector (1 Downto 0);
34  signal NxSC_SOP_REG, SC_SOP_REG : Std_Logic_Vector(3 Downto 0);
35  ------------------------------------------------------------------------------------------------------------------------------------
36  ---
37  -- Maquina de estados
38  type   NState is (CHECK_PREAMBLE,LOAD_SREG,ETH_DATA);
39  signal State: NState;
40  signal NxState: NState;
41  ------------------------------------------------------------------------------------------------------------------------------------
42  ---
43
44  Begin     -- Processo Sequencial --
45            ----------------------------------------------------------------
46            Sequencial_POS: Process (RESET, Clk)
47            Begin
48            ----------------------------------------------------------------
49            If (RESET='1') Then
50                    State<=CHECK_PREAMBLE;
51                    Cnt<=(others=>'0');
52                    CntSReg<="100000";
53                    ShiftReg<=(others=>'0');
```

```
54                      RMOD_FF<=(others=>'0');
55                      SC_SOP_REG<=(others=>'0');
56              Else
57                      If (Clk'Event And Clk='1') Then
58                              State<=NxState;
59                              Cnt<=NxCnt;
60                              CntSReg<=NxCntSReg;
61                              ShiftReg<=NxShiftReg;
62                              RMOD_FF<=NxRMOD_FF;
63                              SC_SOP_REG<=NxSC_SOP_REG;
64                      End If;
65              End If;
66              ------------------------------------------------------------------------
67              End Process Sequencial_POS;
68
69              ------------------------------------------------------------------------
70              Combinatorio: Process (State,Cnt,RX_EN,RX_ER,RXD)
71              Begin
72              ------------------------------------------------------------------------
73              NxShiftReg<=ShiftReg(119 Downto 0) & RXD;
74
75              if RX_EN='1' Then
76                 ----------------------------
77                 --- State MAChine ---
78                      case State is
79                              when CHECK_PREAMBLE =>  -- Verifica o Preambulo
80                                      if Cnt=15 AND RXD="1011" then -- AAAAAAAAAAAAAAB
81                                              NxState<=LOAD_SREG;
82                                      else
83                                              NxState<=CHECK_PREAMBLE;
84                                      end if;
85                              when LOAD_SREG =>  --
86                                      if Cnt=28 Then
87                                              NxState<=ETH_DATA;
88                                      else
89                                              NxState<=LOAD_SREG;
90                                      end if;
91                              when ETH_DATA =>
92                                      NxState<=ETH_DATA;
93                              when others =>
94                                      NxState<=CHECK_PREAMBLE;
95                      end case;
96
97                 --- State MAChine ---
98                 ----------------------------
99              else
100                     NxState<=CHECK_PREAMBLE;
101             end if;
102
103             if (State=CHECK_PREAMBLE AND RXD="1010") OR (State=LOAD_SREG) Then
104                     NxCnt<=Cnt+1; -- Incrementa contador
105             else
106                     NxCnt<="00000"; -- Inicializa contador
107             end if;
108
109             if State=ETH_DATA Then
110                     if CntSReg=7 Then
111                             NxCntSReg<="000000";
112                     else
113                             NxCntSReg<=CntSReg+1;
114                     end if;
115             else
116                     if CntSReg=32 Then
117                             if Cnt=28 Then
118                                     NxCntSReg<="000101";
119                             else
120                                     NxCntSReg<=CntSReg;
121                             end if;
122                     else
123                             NxCntSReg<=CntSReg+1;
124                     end if;
125             end if;
```

```
126
127             if CntSReg(2 DOWNTO 0)=7 Then
128                     WR_EN<='1';
129             else
130                     WR_EN<='0';
131             end if;
132
133             if State=ETH_DATA AND RX_EN='1' Then
134                     NxRMOD_FF<=not(CntSReg(2 DOWNTO 1));
135             else
136                     NxRMOD_FF<=RMOD_FF;
137             end if;
138
139             if Cnt=28 Then
140                     NxSC_SOP_REG<=RXD(3 DOWNTO 1) & '1'; -- Service Class & SOP
141             else
142                     if CntSReg(2 DOWNTO 0)=7 Then
143                             NxSC_SOP_REG<="0000";
144                     else
145                             NxSC_SOP_REG<=SC_SOP_REG;
146                     end if;
147             end if;
148
149             if CntSReg=31 Then
150                     DATA_FIFO<=RMOD_FF & "10" & ShiftReg(123 Downto 92); -- RMOD, EOP, SOP, DATA
151             else
152                     DATA_FIFO<=SC_SOP_REG & ShiftReg(123 Downto 92); -- Service Class, SOP, DATA
153             end if;

    End Process Combinatorio;

    End sm_client;
```

A máquina recebe a trama Ethernet vinda do cliente processando-a e escrevendo o resultado num FIFO. O processo sequencial, que está definido desde a linha 44 até à linha 67, é sensível aos sinais de RESET e de CLK. Quando o sinal de RESET transitar de zero para um, o processo é accionado e os sinais State, Cnt, CntSReg, ShiftReg, RMOD_FF, SC_SOP_REG assumem os valores definidos para cada um deles. Quando o sinal de CLK transitar de zero para um e o sinal RESET tiver com o valor zero os sinais State, Cnt, CntSReg, ShiftReg, RMOD_FF, SC_SOP_REG assumem os valores dos sinais NxState, NxCnt, NxCntSReg, NxShiftReg, NxRMOD_FF e NxSC_SOP_REG respectivamente. Estes últimos sinais tomam valores no interior do processo combinatório (da linha 70 à 153). O código anteriormente descrito, com sinais que tomam valores no interior de um processo combinatório (e que por sua vez servem como entradas para um processo sequencial que é sensível a uma transição de um sinal de relógio) assemelha-se ao comportamento de um flip-flop.

4.3 Implementação das interfaces - PacketPHYs

O módulo MAC do RPR da norma 802.17 do IEEE [1] compreende duas interfaces de interligação do mesmo com a camada física e outras duas com a camada cliente.

Optou-se pelo uso de interfaces Ethernet especificadas na norma 802.3 do IEEE [2], por ser uma interface usual na maioria dos equipamentos de telecomunicações desta natureza.

As interfaces cliente operam com ritmos de transmissão inferiores relativamente às da camada física, tendo sido escolhida a interface MII da norma 802.3 para as interfaces com o cliente e a interface GMII também da norma 802.3 para as interfaces com a camada física. O anexo B da norma 802.17 determina o uso da interface GMII como possível interface para interligação do MAC do RPR com a camada cliente.

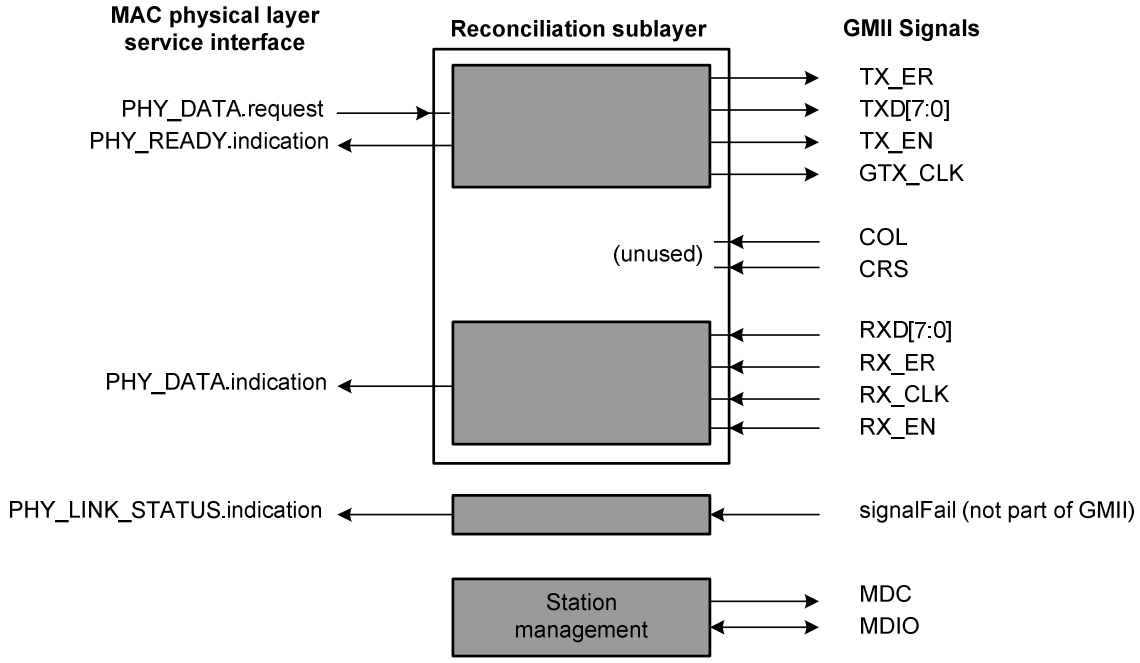


Figura 46 - interface GMII prevista na norma 802.17

Outro aspecto igualmente importante na implementação destas interfaces foi a necessidade de desenvolver algoritmos de CRC capazes de processar 8 bits de dados em simultâneo produzindo como resultado dessa operação um CRC de 16 ou 32 bits.

O anexo E da norma 802.17 expõe alguns algoritmos de como calcular o CRC16 e CRC32 para o HEC e FCS da trama RPR respectivamente. Contudo, apenas são apresentadas algoritmos série, isto é, os dados são recebidos e processados bit a bit. Para a nossa implementação em tecnologia de micro electrónica FPGA é necessário fazer um processamento de byte a byte. O cálculo de CRC através de circuitos de micro electrónica é um assunto que tem sido merecedor de diversos estudos nesta área há já largos anos [7] [20] [21].

A Figura 47 ilustra um circuito representativo do cálculo de um circuito CRC série:

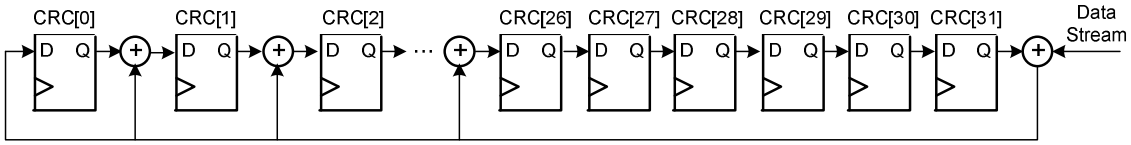


Figura 47 - Circuito CRC série

Contudo o que se pretende é de certa forma transfigurar o mesmo circuito da Figura 47 baseado em funções do tipo exor num aglomerado de equações também exor capazes de obter um mesmo resultado só que num modelo de um circuito CRC paralelo. As equações foram obtidas com algum esforço e compreensão destes algoritmos a partir dos artigos [7] [20] [21] publicados sobre esta matéria. Nos anexos C e D podemos encontrar o código HDL desenvolvido com o intuito de computar um CRC 16 e 32 paralelo de oito bits para os campos HEC e FCS da trama RPR.

4.3.1 Interfaces Cliente

São necessários dois módulos na interface cliente, um de recepção e outro de transmissão. O módulo de recepção terá de processar as tramas Ethernet recebidas do lado do cliente em tramas RPR enquanto que o de transmissão irá processar as tramas RPR recebidas do lado do anel em tramas Ethernet. Todo este processo resulta de uma operação de encapsulamento das tramas Ethernet em tramas RPR e vice-versa. A Figura 48 descreve o conteúdo de uma trama Ethernet constituída por um cabeçalho composto por um preâmbulo, endereços MAC de fonte e de destino e algumas tag's, carga paga (payload) e por ultimo um campo de controlo de erros FCS (frame check sequence).

7 bytes AA...A	*1 byte AB*	*6 bytes*	6 bytes	2 bytes	2 bytes	2 bytes	4 - 1500 bytes	4 bytes
Preamble	**Start Frame Delimeter**	**Dest. MAC address**	**Source MAC address**	**Tag Type 802.1Q**	**Tag Control Inf.**	**length/ Type**	**Data/Pad**	**FCS**

Figura 48 - Trama Ethernet

Na interface de transmissão o cabeçalho e FCS da trama RPR vinda do MAC do RPR são removidos, resultando numa nova trama Ethernet. Na interface de recepção a trama Ethernet será colocada no interior do campo ServiceDataUnit da trama RPR na operação de encapsulamento de trama.

4.3.1.1 Interface Cliente Rx

As tramas RPR possuem também um cabeçalho, um campo de dados e um campo de controlo de erros FCS. Na operação de encapsulamento das tramas Ethernet em tramas RPR é necessário preencher adequadamente todos estes campos da trama RPR. O IEEE elaborou um suplemento da norma 802.17 (P802.17a/D1.0) [3] com informação específica para esta tarefa. Na Figura 49, podemos observar à esquerda uma trama Ethernet recebida na interface cliente Rx e na direita, o encapsulamento da trama Ethernet numa trama RPR.

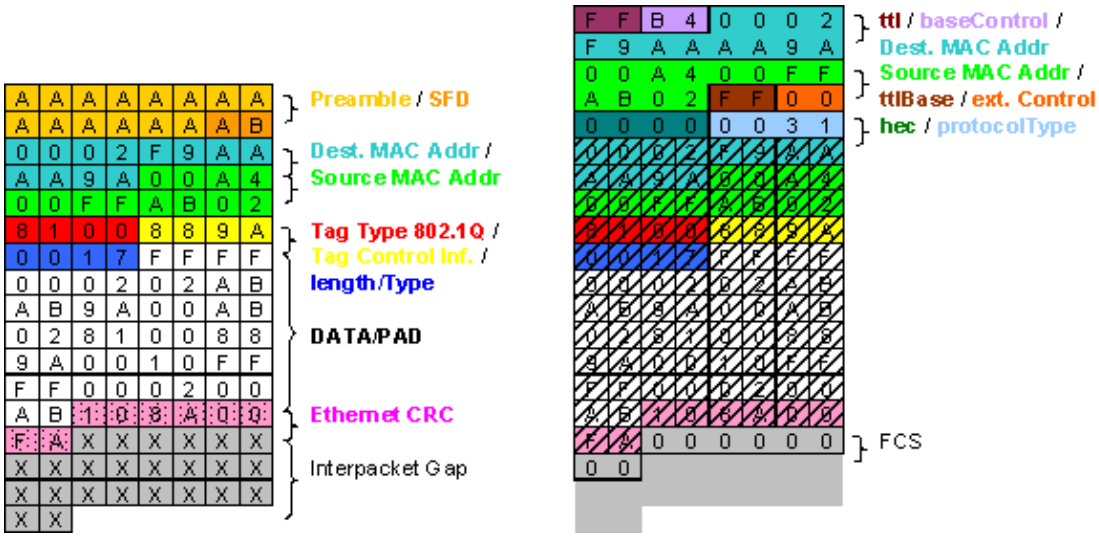


Figura 49 - encapsulamento de uma trama Ethernet numa trama RPR

A interface cliente Rx irá encapsular a trama Ethernet vinda do cliente numa nova trama RPR. O campo service class (SC) do byte baseControl da trama RPR irá ser preenchido de acordo com o valor presente nos bits de user priority do campo Tag Control Information da trama Ethernet. Assim sendo o campo Service Class fica com o valor 00 em binário para tramas de classe C, 01 para classe B e 10 ou 11 para classe A. No exemplo apresentado na Figura 49, o campo SC fica com o valor 10.

Na Figura 50 podemos observar os blocos que constituem a interface cliente RX.

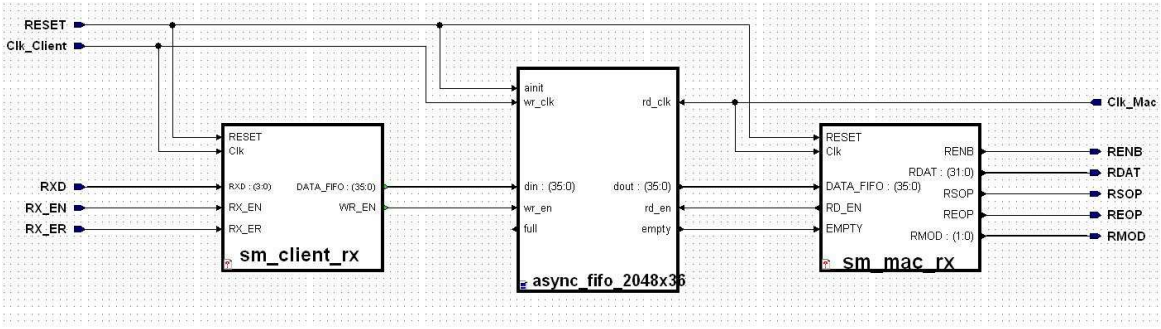


Figura 50 - esquema da interface cliente Rx

A primeira máquina de estados (sm_client) recebe a trama Ethernet vinda do cliente. A máquina aguarda a transmissão de um preâmbulo por parte do cliente e após a recepção do último byte do preâmbulo a máquina é accionada. Os dados são escritos num FIFO assíncrono de 32 bits de largura para dados e de 4 bits para controlo. Neste mesmo FIFO, para além dos dados será também escrito informação de início de trama (SOP) e de fim de trama (EOP). Quando for escrito, a primeira palavra de 32 bits de dados no FIFO haverá um bit de controlo que será colocado a um para indicar o início de trama (SOP) . Também serão preenchidos dois bits de controlo com a informação do, service class calculada a partir dos user priority bits da trama Ethernet. Na escrita da última palavra de 32 bits de dados, o bit de controlo de EOP é colocado a um, bem como os bits de controlo com informação de RMOD.

Esta máquina recebe quatro bits de dados a cada batimento de relógio, e escreve trinta e dois bits de dados no FIFO a cada oito batimentos de relógio. Os dados só poderão ser escritos no FIFO quando a informação para os bits de controlo de service class estiver disponível. Sendo que estes bits são calculados a partir da informação presente nos bits de user priority bits da trama Ethernet, apenas quando a interface cliente enviar esta informação é que a trama poderá começar a ser inserida no FIFO. Antes desta situação, os dados vindos da interface cliente necessários para a trama RPR que se pretende construir vão sendo armazenados num shift register devidamente dimensionado. Os inputs da máquina são os sinais RX_EN, e RXD[3:0]. O sinal RX_EN sinaliza que os dados presentes no barramento RXD são dados validos. Os outputs da máquina são os sinais WR_EN e DATA_FIFO[35:0]. O sinal WR_EN quando é activado, conduz à escrita dos valores presentes no barramento de dados DATA_FIFO no FIFO.

Todos os sinais referidos anteriormente são amostrados por um relógio com uma frequência de 25MHz. Na Figura 51 podemos observar uma simulação do comportamento da máquina e dos respectivos sinais.

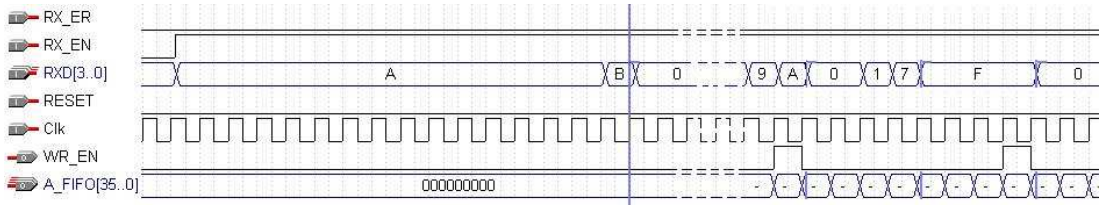


Figura 51 - simulação do modulo sm_client da interface client rx

A segunda máquina (sm_mac) será responsável por ler os dados presentes no FIFO, processá-los de maneira a estruturar uma trama do tipo RPR e de seguida envia-la para o MAC. A máquina lê os dados presentes no FIFO detectando o início de trama SOP e em seguida inicia a construção da trama RPR. No final a máquina detecta a presença de fim de trama EOP e informação de RMOD.

Os inputs da máquina serão os sinais DATA_FIFO[35:0] e EMPTY vindos do FIFO. O sinal de EMPTY no caso de activo sinaliza que o FIFO se encontra vazio. Como outputs os sinais são o RD_EN, RENB, RSOP, REOP, RMOD[1:0] e RDAT[31:0]. O sinal RD_EN serve como enable do FIFO para a leitura dos dados. O sinal RSOP e REOP sinalizam o primeiro e ultimo bytes da trama enviada pelo MAC do RPR, respectivamente. O sinal RENB, quando activo, valida o envio de dados presentes no barramento de dados RDAT. O sinal RMOD é válido apenas quando o sinal REOP está no estado activo de acordo com a Tabela 21.

RMOD[1:0]	RDAT[31:0]
RMOD[1:0]=”00”	RDAT[31:0] valid
RMOD[1:0]=”01”	RDAT[31:8] valid
RMOD[1:0]=”10”	RDAT[31:16] valid
RMOD[1:0]=”11”	RDAT[31:24] valid

Tabela 21 - campo RMOD da interface com o modulo MAC

Todos os sinais referidos anteriormente são batidos por um relógio com uma frequência de 3.125MHz. Este relógio é fornecido pelo MAC e tem uma frequência oito vezes inferior ao relógio de escrita no FIFO vindo do cliente. Isto acontece devido à relação de 4 para 32 bits de dados de largura recebidos a cada ciclo de relógio do cliente e do MAC do RPR respectivamente. Na Figura 52 podemos observar uma simulação do comportamento da máquina e dos seus respectivos sinais.

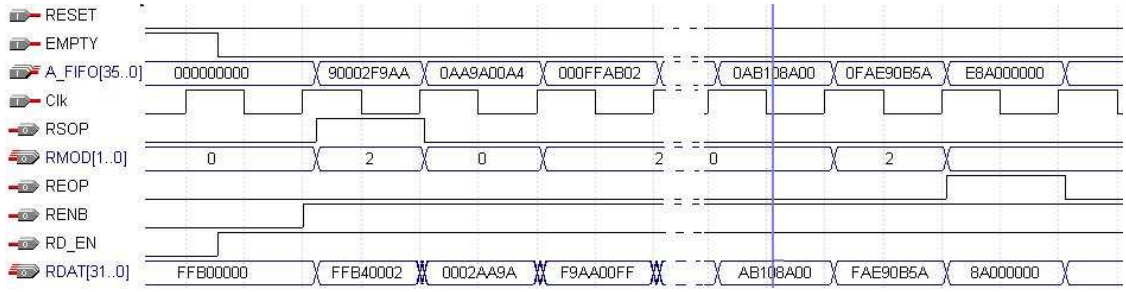


Figura 52 - simulação do módulo sm_MAC da interface client rx

Foi necessário recorrer a um FIFO assíncrono (assync_fifo) capaz de funcionar com um relógio de escrita diferente do relógio de leitura. O FIFO irá servir para adaptar os tempos necessários à transferência de dados do cliente MII para o MAC do RPR. Além disso serve também de amortecimento para os dados vindos do cliente que não possam ser no imediato transferidos para o MAC do RPR. Os sinais SOP e EOP são enviados através do FIFO, de modo a permitir uma melhor sincronização do sistema no caso de ocorrência de alguma anomalia.

4.3.1.2 Interface Cliente Tx

A interface cliente TX é responsável pelo envio das tramas do MAC do RPR para o cliente Ethernet. Para isso é necessário remover o cabeçalho das tramas RPR vindas do MAC RPR. A Figura 53 é demonstrativa deste procedimento.

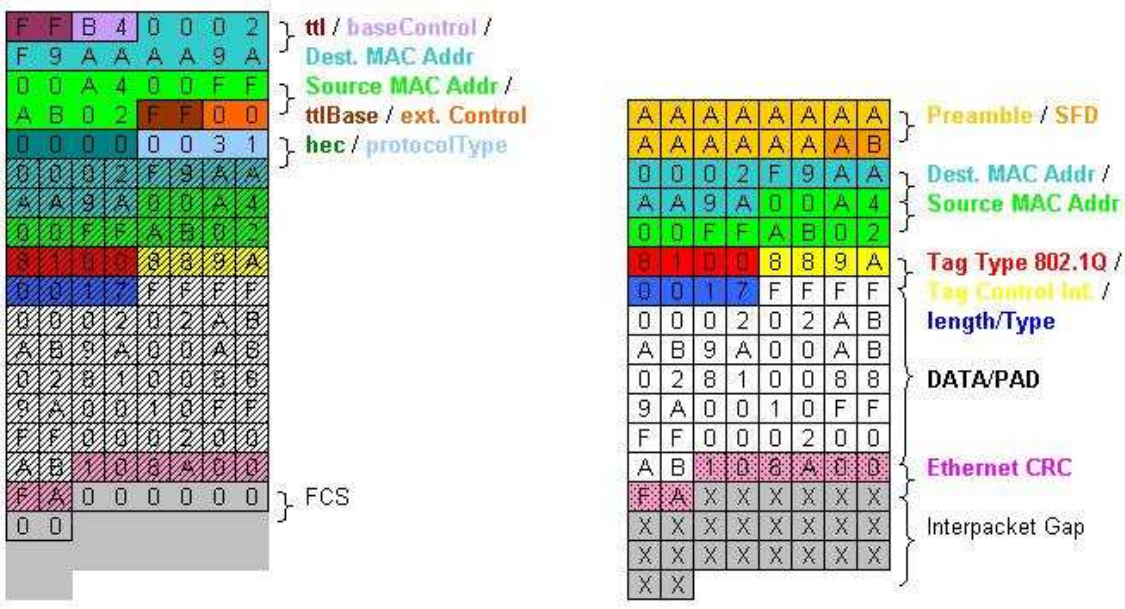


Figura 53 - encapsulamento de uma trama RPR numa trama Ethernet

O cabeçalho da trama RPR composto pelos campos ttl, baseControl, da, sa, ttlBase, extendedControl, hec e protocolType são removidos assim como o FCS. O campo serviceDataUnit que possui os dados referentes à trama Ethernet é extraído sendo adicionado um preâmbulo de acordo com a estrutura de dados de uma trama do tipo Ethernet.

Na Figura 54 podemos observar os blocos que constituem a interface cliente Tx.
A primeira máquina (sm_mac) recebe a trama vinda do MAC do RPR removendo-lhe o cabeçalho RPR e respectivo FCS. A máquina é responsável por escrever num FIFO o campo serviceDataUnit da trama RPR que contem os dados da trama Ethernet. Os inputs da máquina são os sinais TSOP, TEOP, TENB, TMOD[1:0] e TDAT[31:0]. O sinal TSOP e TEOP sinalizam o primeiro e ultimo bytes da trama enviada pelo MAC do RPR respectivamente. O sinal TENB, quando activo, valida o envio de dados presentes no barramento de dados TDAT. O sinal TMOD é valido apenas quando o sinal TEOP está no estado activo de acordo com a Tabela 22.

TMOD[1:0]	TDAT[31:0]
TMOD[1:0]=”00”	TDAT[31:0] valid
TMOD[1:0]=”01”	TDAT[31:8] valid
TMOD[1:0]=”10”	TDAT[31:16] valid
TMOD[1:0]=”11”	TDAT[31:24] valid

Tabela 22 - campo TMOD da interface com o módulo MAC

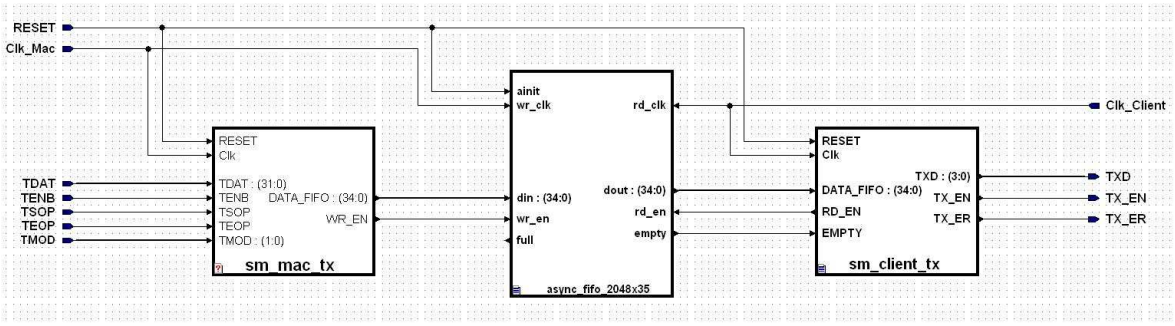


Figura 54 - esquema da interface cliente Tx

Os outputs da máquina são os sinais WR_EN e DATA_FIFO[31:0]. O sinal WR_EN quando é activado conduz à escrita dos valores presentes no barramento de dados DATA_FIFO no FIFO.
Todos os sinais referidos anteriormente são batidos por um relógio com uma frequência de 3.125MHz. Na Figura 55 podemos observar uma simulação do comportamento da máquina e seus respectivos sinais.

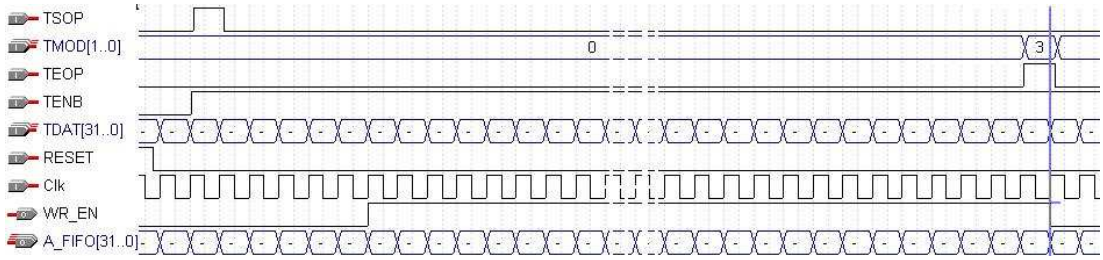


Figura 55- simulação do módulo sm_MAC da interface client tx

A segunda máquina (sm_client) será responsável por ler os dados presentes no FIFO, processá-los e envia-los de seguida para o cliente. Aos dados lidos do FIFO será adicionado um preâmbulo de uma trama do tipo Ethernet.
Os inputs da máquina serão os sinais EOP, EMPTY e o barramento de dados DATA_FIFO[31:0] vindos do FIFO. O sinal EOP sinaliza o último byte do pacote lido do FIFO sendo a sua utilização necessária para sincronização do sistema. O sinal de EMPTY no caso de activo sinaliza que o FIFO se encontra vazio. Como outputs, os sinais são o RD_EN, TXD[3:0] e TX_EN. O sinal RD_EN serve como enable do FIFO para a leitura dos dados. O sinal TX_EN, quando activo, indica que os dados presentes no barramento TXD são válidos

Todos os sinais referidos anteriormente são amostrados por um relógio com uma frequência de 25MHz. Este relógio é fornecido pelo cliente e tem uma frequência oito vezes superior ao relógio de escrita no FIFO vindo do MAC do RPR. Isto acontece pois os dados vindos do MAC do RPR são de 32 bits ao passo que para o cliente são enviados apenas quatro bits por cada ciclo de relógio. Na Figura 56 podemos observar uma simulação do comportamento da máquina e dos respectivos sinais.

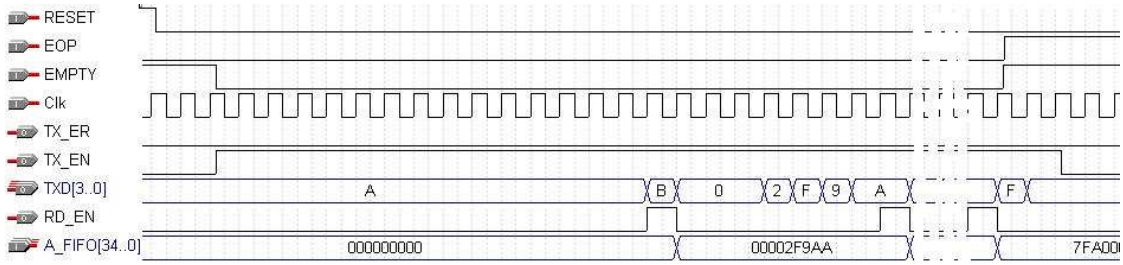


Figura 56 - simulação do módulo sm_client da interface client tx

Foi necessário recorrer a um FIFO assíncrono capaz de funcionar com um relógio de escrita diferente do relógio de leitura. O FIFO irá servir para adaptar os tempos necessários para a transferência de dados do MAC do RPR para o cliente MII. Além disso serve também de amortecimento para os dados vindos do MAC do RPR que não possam ser no imediato transferidos para o cliente. O sinal EOP é enviado através do FIFO de modo a permitir uma melhor

sincronização do sistema no caso de ocorrência de alguma anomalia. A interface cliente Tx foi simulada como ilustra a Figura 57:

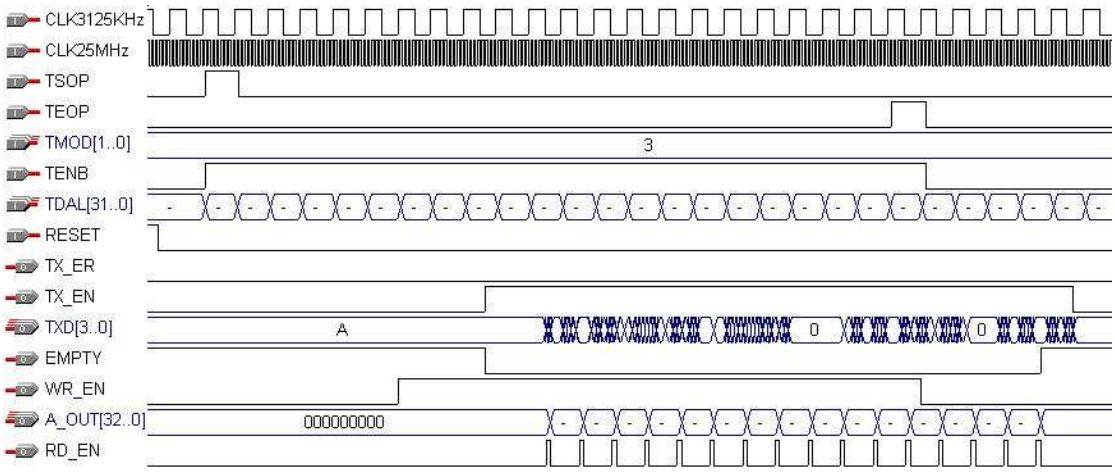


Figura 57 - simulação da interface client tx

4.3.2 Interfaces de linha

A interface com a camada física, embora com ritmos de transmissão superiores, é em tudo semelhante ao que se passa do lado do cliente. Contudo existe uma dificuldade acrescida, não só pelo aumento dos ritmos de transmissão, como também pela possibilidade do envio de diferentes tipos de tramas RPR (control frame, fairness frame e idle frame), para além das tramas de dados (data frame).

As interfaces com camada física terão assim de determinar o FCS da trama RPR, erros de paridade do baseControl e o HEC.

São necessários dois módulos na interface de linha, um de recepção Rx e outro de transmissão Tx. O módulo de recepção terá de processar as tramas Ethernet recebidas do lado do anel em tramas RPR enquanto que o de transmissão irá processar as tramas RPR recebidas do lado do MAC do RPR em tramas Ethernet. Todo este processo resulta de uma operação de encapsulamento das tramas Ethernet em tramas RPR e vice-versa.

4.3.2.1 Interface linha Rx

Na interface de recepção, o cabeçalho e FCS da trama Ethernet são removidos resultando numa nova trama RPR. A interface terá de averiguar qual o tipo de trama RPR analisando para isso os bits FrameType do campo baseControl de modo a confirmar se o bit de paridade deste campo

está correcto e calcular o respectivo HEC se necessário comparando-o com o valor de HEC recebido. O mesmo procedimento terá de ser feito relativamente ao FCS.

Nesta interface os dados são recebidos em formato de Byte, sendo para isso necessário desenvolver mecanismos capazes de computar um CRC que a cada ciclo de relógio seja capaz de calcular um valor de CRC para cada 8 bit's de entrada. Isto acontece tanto para o cálculo do HEC, que resulta num CRC de 16 bit's, como para o cálculo do FCS, CRC de 32 bit's. Existem alguns trabalhos publicados nesta área, de como calcular um CRC paralelo [7] [20] [21], que resultam em algumas equações baseadas em exor's que têm como variável de entrada os bits de dados sobre os quais pretendemos calcular um CRC e também as saídas resultantes no ciclo de relógio anterior. Estas equações bem como a sua implementação são explicadas com maior detalhe no ponto 4.3.3 desta tese de mestrado.

Na Figura 58 podemos observar os blocos que constituem a interface de linha RX:

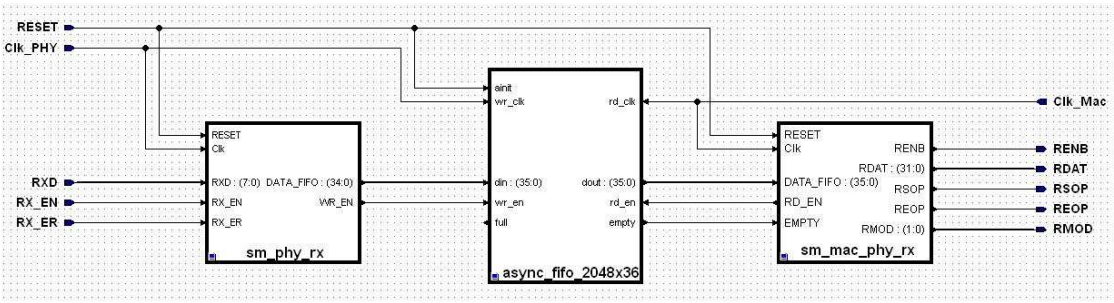


Figura 58 - esquema da interface linha Rx

A primeira máquina (sm_phy) recebe a trama Ethernet vinda do anel. A máquina aguarda a transmissão de um preâmbulo por parte do anel e após a recepção do último byte do preâmbulo a máquina é accionada. Os dados são escritos num FIFO assíncrono de 32 bits de largura para dados e de 3 bits para controlo. Neste mesmo FIFO, para além dos dados será também escrito informação de fim de trama (EOP). Na escrita da última palavra de 32 bits de dados o bit de controlo de EOP é colocado a um, bem como os bits de controlo com informação de RMOD.

Esta máquina recebe oito bits de dados a cada batimento de relógio e escreve trinta e dois bits de dados no FIFO a cada quatro batimentos de relógio. Os dados vindos da interface cliente necessários para a trama RPR que se pretende construir vão sendo armazenados num shift register devidamente dimensionado para o efeito, e quando este estiver cheio os dados são transferidos para o FIFO. Os inputs da máquina são os sinais RX_EN e RXD[7:0]. O sinal RX_EN sinaliza que os dados presentes no barramento RXD são dados validos. Os outputs da máquina são os sinais WR_EN e DATA_FIFO[34:0]. O sinal WR_EN quando é activado conduz à escrita dos valores presentes no barramento de dados DATA_FIFO no FIFO. Existem também os sinais

HEC_EN e FCS_EN que serão usados pelas máquinas de cálculo do HEC e FCS respectivamente.

Todos os sinais referidos anteriormente são batidos por um relógio com uma frequência de 125MHz. Na Figura 59 podemos observar uma simulação do comportamento da máquina e dos respectivos sinais.

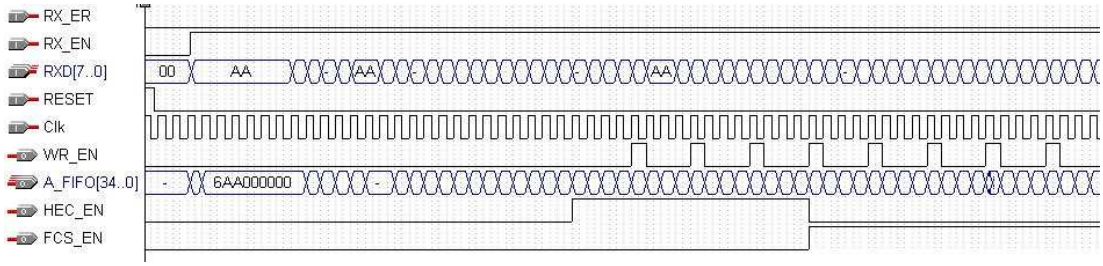


Figura 59 - simulação do módulo sm_phy da interface linha rx

A segunda máquina (sm_mac) será responsável por ler os dados presentes no FIFO, processá-los de maneira a estruturar uma trama do tipo RPR e envia-la de seguida para o MAC. A máquina lê constantemente os dados presentes no FIFO detectando o fim de trama EOP e iniciando em seguida a construção de uma nova trama RPR. No final a maquina detecta a presença de fim de trama EOP e informação de RMOD e o processo repete-se.

Os inputs da máquina serão os sinais DATA_FIFO[34:0] e EMPTY vindos do FIFO. O sinal de EMPTY, no caso de activo, sinaliza que o FIFO se encontra vazio. Como outputs os sinais são o RD_EN, RENB, RSOP, REOP, RMOD[1:0] e RDAT[31:0]. O sinal RD_EN serve como enable do FIFO para a leitura dos dados. O sinal RSOP e REOP sinalizam o primeiro e ultimo bytes da trama enviada pelo MAC do RPR respectivamente. O sinal RENB, quando activo, valida o envio de dados presentes no barramento de dados RDAT. O sinal RMOD é valido apenas quando o sinal REOP está no estado activo de acordo com a Tabela 21.

Todos os sinais referidos anteriormente são batidos por um relógio com uma frequência de 31,25MHz. Este relógio é fornecido pelo MAC e tem uma frequência quatro vezes inferior ao relógio de escrita no FIFO vindo do anel. Isto acontece devido à relação de oito para trinta e dois bits de dados de largura recebidos a cada ciclo de relógio do anel e do MAC do RPR respectivamente. Na Figura 60 podemos observar uma simulação do comportamento da máquina e dos seus respectivos sinais.

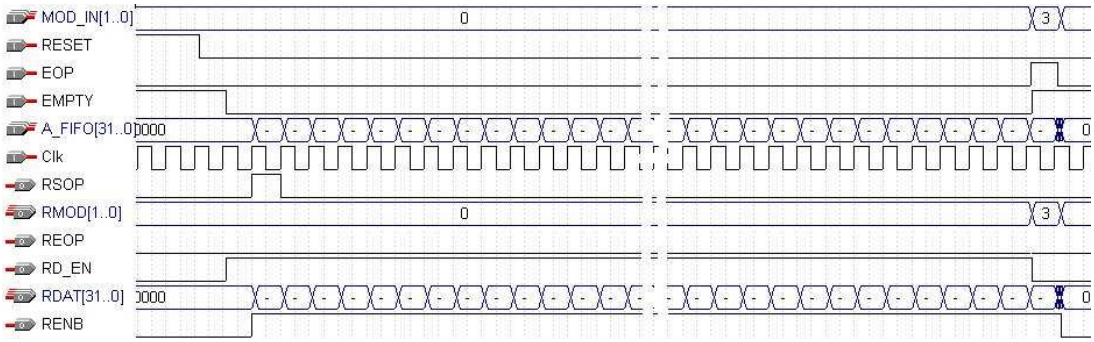


Figura 60 - simulação do módulo sm_MAC da interface linha rx

Foi necessário recorrer a um FIFO assíncrono capaz de funcionar com um relógio de escrita diferente do relógio de leitura. O FIFO irá servir para adaptar os tempos necessários para a transferência de dados da interface do anel GMII para o MAC do RPR. Além disso serve também de amortecimento para os dados vindos do anel que não possam ser no imediato transferidos para o MAC do RPR. O sinal EOP é enviado através do FIFO de modo a permitir uma melhor sincronização do sistema no caso de ocorrência de alguma anomalia.

4.3.2.2 Interface linha Tx

A interface de linha Tx é responsável pelo envio das tramas do MAC do RPR para o anel. Para isso é necessário encapsular a trama RPR recebida do MAC do RPR numa nova trama Ethernet.
Na interface de transmissão a trama RPR será colocada no interior do campo Data/Pad da trama Ethernet.

O cabeçalho da trama Ethernet composto pelos campos preâmbulo, sfd, da, sa, tag type, tag control e length/type terão de ser devidamente preenchidos de acordo com a informação vinda nas tramas RPR recebidas do MAC. O campo preâmbulo será preenchido com 7 bytes “AA” em hexadecimal, o campo sfd com os valores “AB”, os endereços de destino e fonte da e sa serão preenchidos com os valores dos mesmos endereços presentes na trama RPR. O campo tag type será preenchido com os bytes “81” e “00”, a tag control com zeros “00” e length/type com o comprimento da trama calculado através do valor do comprimento recebido na trama RPR no respectivo campo.

Na Figura 61 podemos observar os blocos que constituem a interface linha Tx.
A primeira máquina recebe a trama vinda do MAC do RPR escrevendo a mesma num FIFO devidamente dimensionado para o efeito. Os inputs da máquina são os sinais TSOP, TEOP, TENB, TMOD[1:0] e TDAT[31:0]. O sinal TSOP e TEOP sinalizam o primeiro e ultimo bytes da trama enviada pelo MAC do RPR respectivamente. O sinal TENB quando activo valida o envio de

dados presentes no barramento de dados TDAT. O sinal TMOD é valido apenas quando o sinal TEOP está no estado activo de acordo com a Tabela 22.

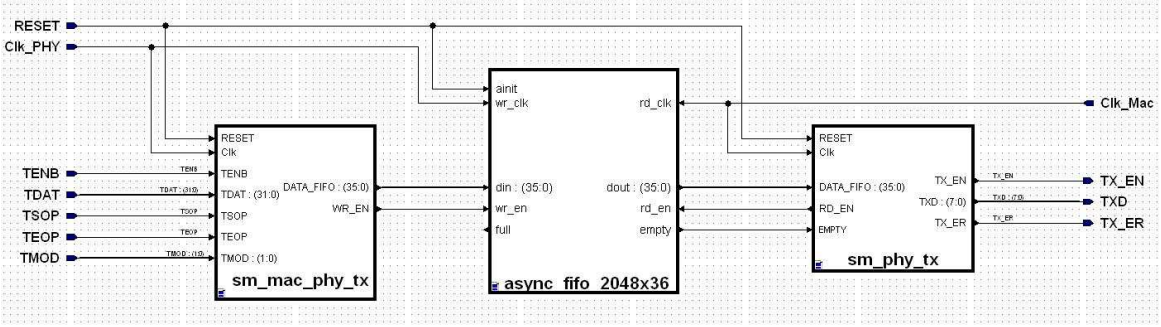


Figura 61 - esquema da interface linha Tx

Os outputs da máquina são os sinais WR_EN e DATA_FIFO[31:0]. O sinal WR_EN quando é activado conduz à escrita dos valores presentes no barramento de dados DATA_FIFO no FIFO.
Todos os sinais referidos anteriormente são batidos por um relógio com uma frequência de 31,25MHz. Na Figura 61 podemos observar uma simulação do comportamento da máquina e dos respectivos sinais.

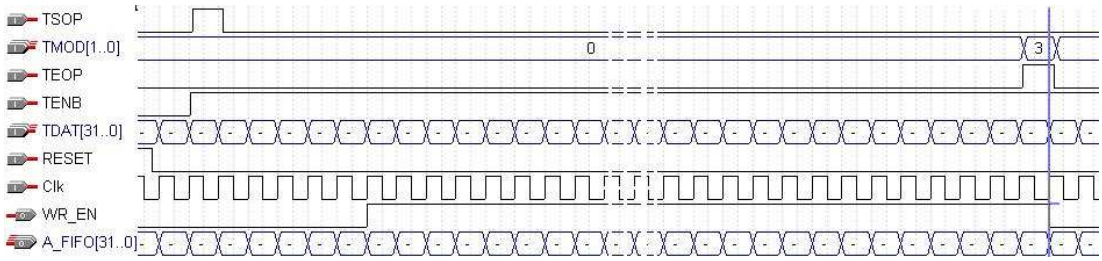


Figura 62- simulação do módulo sm_MAC da interface linha tx

A segunda máquina será responsável por ler os dados presentes no FIFO, processá-los e envia-los de seguida para o anel. Aos dados lidos do FIFO será adicionado um preâmbulo de uma trama do tipo Ethernet bem como os restantes campos devidamente preenchidos que compõem uma trama do tipo Ethernet.
Os inputs da máquina serão os sinais EOP, EMPTY e o barramento de dados DATA_FIFO[31:0] vindos do FIFO. O sinal EOP sinaliza o último byte do pacote lido do FIFO sendo a sua utilização necessária para sincronização do sistema. O sinal de EMPTY no caso de activo sinaliza que o FIFO se encontra vazio. Como outputs os sinais são o RD_EN, TXD[7:0] e TX_EN. O sinal RD_EN serve como enable do FIFO para a leitura dos dados. O sinal TX_EN quando activo indica que os dados presentes no barramento TXD são válidos
Todos os sinais referidos anteriormente são batidos por um relógio com uma frequência de 125MHz. Este relógio é fornecido pelo anel e tem uma frequência quádrupla do relógio de escrita no FIFO vindo do MAC do RPR. Isto acontece pois os dados vindos do MAC do RPR são de 32

bits ao passo que para o cliente são enviados apenas oito bits por cada ciclo de relógio. Na Figura 63 podemos observar uma simulação do comportamento da máquina e dos respectivos sinais.

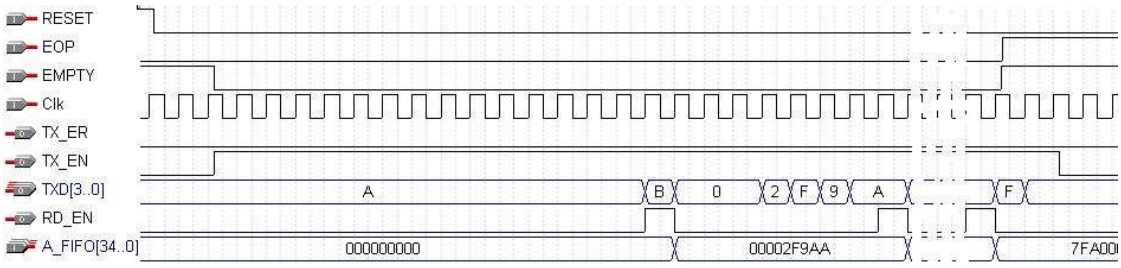


Figura 63 - simulação do módulo sm_client da interface client tx

Foi necessário recorrer a um FIFO assíncrono capaz de funcionar com um relógio de escrita diferente do relógio de leitura. O FIFO irá servir para adaptar os tempos necessários para a transferência de dados do MAC do RPR para a interface do anel GMII. Além disso serve também de amortecimento para os dados vindos do MAC do RPR que não possam ser no imediato transferidos para o anel. O sinal EOP é enviado através do FIFO de modo a permitir uma melhor sincronização do sistema no caso de ocorrência de alguma anomalia. A interface anel Tx foi simulada como ilustra a Figura 64:

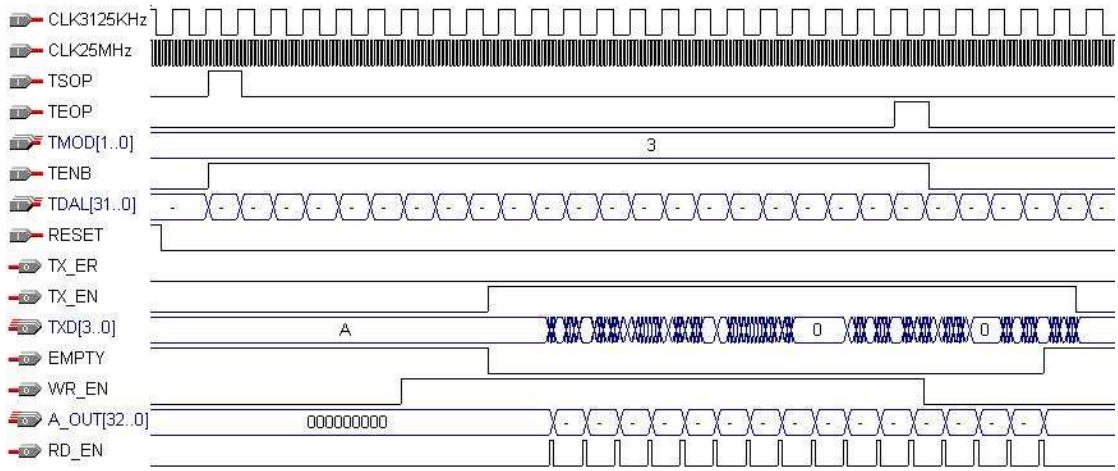


Figura 64 - simulação da interface linha tx

4.4 Implementação das Interfaces – SONET/SDH

A norma 802.17, além dos packet PHY's Ethernet já explicados anteriormente, permite também o uso de interfaces de linha SONET/SDH.
A interface SONET/SDH consiste numa sub-camada de reconciliação que mapeia as primitivas de serviço lógicas da interface de serviço da camada física do MAC recorrendo a interfaces eléctricas standard. Neste trabalho foi implementada a interface SPI Level 3 (SPI-3) [14] de 32-bit de dados capaz de funcionar a ritmos de transmissão que vão desde 155 Mb/s até 2.5 Gb/s como definido no Optical Internetworking Forum (OIF).
Com esta interface é possível o envio e recepção de tramas através de um link STM1, STM4 ou STM16 com ritmos de transmissão de dados úteis de 149.76 Mb/s, 599.04 Mb/s e 2396.160 Mb/s respectivamente.

Além disso, é necessário encapsular/desencapsular as tramas RPR num formato de dados capaz de ser interpretado pela camada SONET/SDH. A norma 802.17 permite o mapeamento das tramas RPR em GFP, HDLC-like e LAPS, tendo sido escolhido o formato HDLC-like.

Este tipo de mapeamento é funcionalmente idêntico ao definido no IETF RFC 2615 [11] para o protocolo point-to-point (PPP) sobre SONET/SDH e IETF RFC 1662 [10] para PPP em HDLC-like framing excepto no seguinte:

- Os parâmetros de link são estatisticamente configurados. O Link control protocol (LCP) não é utilizado.
- O mapeamento não suporta tramas do tipo PPP, suportando apenas tramas RPR.
- As funcionalidades PPP-specific e legacy support não são suportadas nem definidas.
- Os campos de address, control e FCS fazem parte da trama RPR não sendo por isso utilizados no mapeamento HDLC-like.
- O scrambler X43+1 está sempre activo.

O byte-synchronous HDLC-like framing para tramas do tipo RPR deverá ser realizado de acordo com a IETF RFC 1662 utilizando byte-stuffed framing, considerando que as tramas PPP sejam interpretadas como tramas do tipo RPR.
A sequência de flags utilizadas para indicar o início e fim de trama bem como a flag de control escape, usada para transparência, são apresentadas na Tabela 23:

Byte sequence	Name	Value
Flag sequence	flag	$7E_{16}$
Control escape	control	$7D_{16}$

Tabela 23 - Flag sequence e Control escape para o formato de dados HDLC-like

4.4.1. Processamento HDLC

Nas interfaces de linha SONET/SDH foi necessário a inclusão de lógica capaz de implementar a formatação de dados HDLC. As flags 7E sinalizam o início e fim de trama, como tal não podem aparecer no interior da trama RPR que se pretende formatar em dados HDLC. Portanto sempre que o caracter 7E ocorrer, o controlador HDLC irá substituir esse valor por 7D-5E. O caracter 7D será considerado como sendo o caracter de escape precisando também de ser substituído sempre que se verificar no interior da trama RPR ficando com o valor 7D-5D. O receptor irá proceder à operação inversa.

Serão necessários dois módulos na interface de linha SONET/SDH, um de recepção e outro de transmissão. O módulo de recepção terá de processar a informação HDLC-like contida no interior do contentor das tramas SDH recebidas do lado do anel em tramas RPR enquanto que o de transmissão irá processar a informação recebida do lado do MAC do RPR no formato de dados HDLC-like.

Por razões de estabilidade da rede, os dados do payload da trama HDLC são baralhados ainda antes de serem inseridos no interior dos contentores SDH. Esta operação impossibilita que um utilizador mal intencionado envie padrões que se traduzam em problemas de sincronismo para a camada SONET/SDH, emulando o padrão de sincronismo de scrambler da trama SONET/SDH ou replicando a palavra de alinhamento da trama SONET/SDH. O scrambler é derivado do polinómio x43+1 e é idêntico ao usado em ATM.

Os esquemas seguintes são ilustrativos dos scrambler e descrambler implementados:

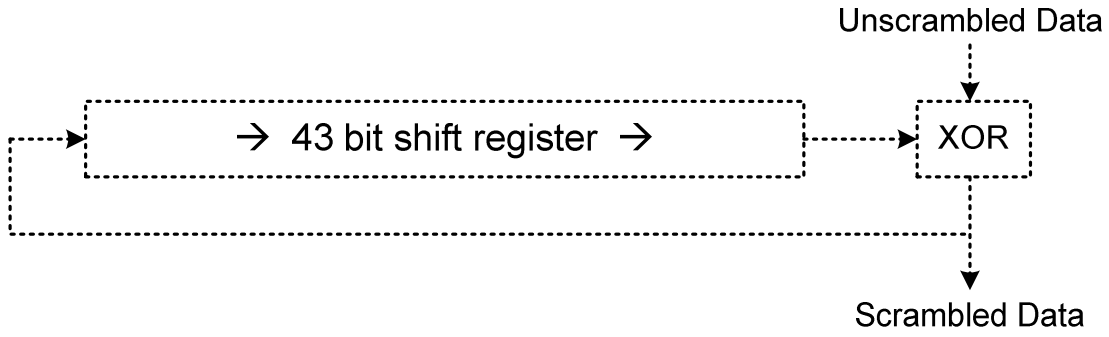


Figura 65 - Scrambler HDLC

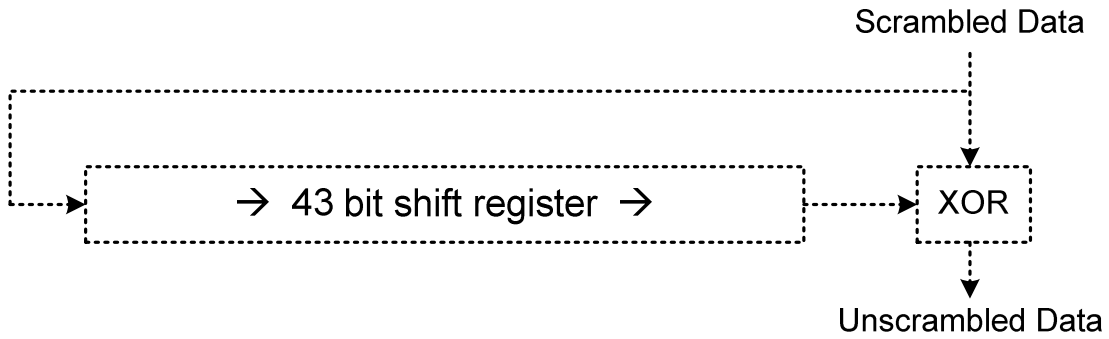


Figura 66 - Descrambler HDLC

O código HDL do scrambler e descrambler HDLC pode ser consultado nos anexos E e F.

A Figura 67 demonstra a estrutura de uma trama do tipo HDLC-like compreendendo tramas RPR segundo a norma 802.17:

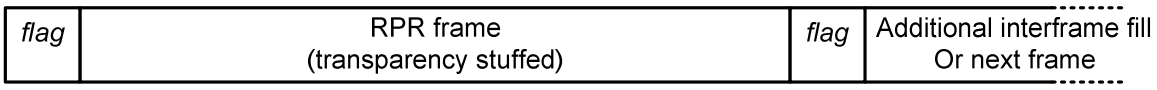


Figura 67 - Estrutura de uma trama RPR no formato HDLC-like

Os campos address, control e protocol são suprimidos.

4.4.2. Interfaces de linha Rx SONET/SDH

Na interface de recepção são detectadas as flags de início e de fim da trama HDLC-like, delineando assim a trama recebida. Em seguida a trama HDLC-like é processada removendo-lhe o bit stuffing introduzido, e recuperando deste modo as tramas RPR recebidas. A interface terá de averiguar qual o tipo de trama RPR, analisando para isso os bits FrameType do campo

baseControl para confirmar se o bit de paridade deste campo está correcto e calcular o HEC. O mesmo procedimento terá de ser feito relativamente ao FCS.

Na Figura 68 podemos observar os blocos que constituem a interface de linha RX SONET/SDH.

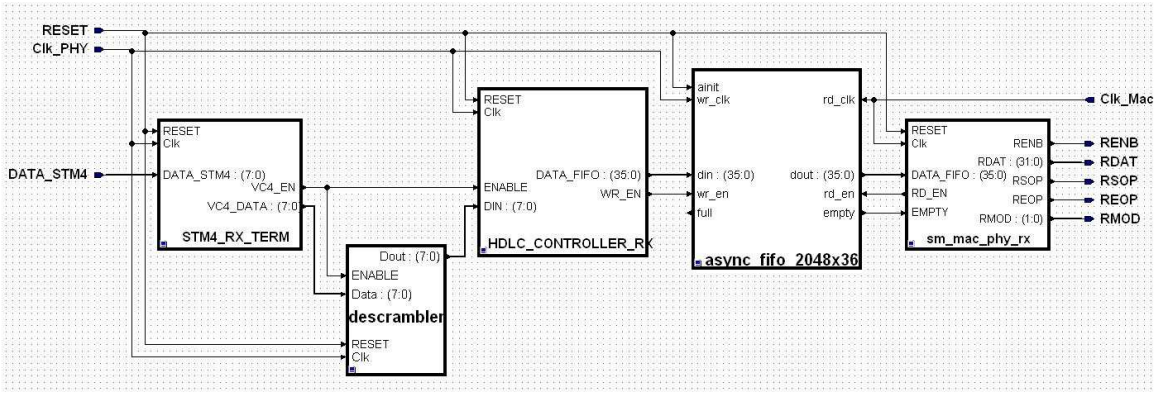


Figura 68 - esquema da interface SDH Rx

A primeira máquina (STM4_RX_TERM) recebe e termina o STM4 recebido do anel. A máquina é responsável por executar funções inerentes ao protocolo SDH. A implementação desta máquina não fez parte desta dissertação de mestrado, estando apenas presente na Figura 68 de modo a ilustrar o procedimento necessário para a recepção de tramas RPR através de PHY's SONET/SDH. O output da máquina é um sinal VC4_EN que sempre que está activo sinaliza dados validos no barramento VC4_DATA(7:0). A máquina remove o overhead da trama STM4 retornando os dados presentes no interior do contentor VC4. Em seguida existe um descrambler baseado no polinómio 1+x^43 que recupera os dados HDLC anteriormente baralhados. Temos então o HDLC_CONTROLLER_RX que é responsável por identificar e remover as flags do protocolo HDLC escrevendo o resultado no FIFO. A última máquina será responsável por ler os dados presentes no FIFO, processá-los de maneira a estruturar uma trama do tipo RPR e envia-la de seguida para o MAC. A máquina é idêntica à segunda máquina utilizada na interface de linha Rx dos Packet PHY's.

Foi também necessário recorrer a um FIFO assíncrono capaz de funcionar com um relógio de escrita diferente do relógio de leitura. O FIFO irá servir para adaptar os tempos necessários para a transferência de dados do anel para o MAC do RPR. Além disso serve também de amortecimento para os dados vindos do anel que não possam ser no imediato transferidos para o MAC do RPR. O sinal EOP é enviado através do FIFO de modo a permitir uma melhor sincronização do sistema no caso de ocorrência de alguma anomalia.

4.4.3. Interfaces de linha Tx SONET/SDH

A interface de linha Tx é responsável pelo envio das tramas do MAC do RPR para o anel.
A trama RPR será formatada numa nova trama HDLC, em seguida os dados serão baralhados e colocados no interior dos contentores VC4 dos link's STM-4.

Na Figura 69 podemos observar os blocos que constituem a interface linha Tx:

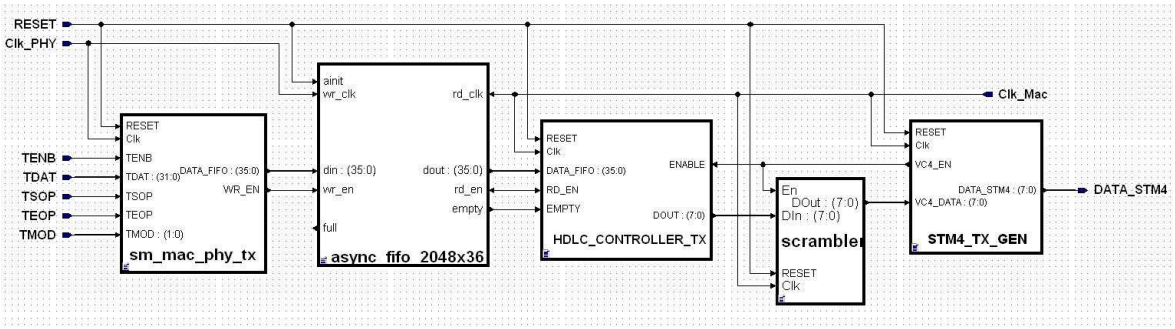


Figura 69 - esquema da interface linha SDH Tx

4.4.4. Aspectos de implementação das interfaces

Depois de terminado o desenvolvimento dos códigos HDL das interfaces, procedeu-se ao mapeamento do mesmo na FPGA XC2VP70 da Xilinx. O resultado final relativamente ao numero de slices ocupados na FPGA para o conjunto das interfaces de linha e de cliente de é aproximadamente 5966 slices ou seja 18% de ocupação do dispositivo. Note-se que para as interfaces de linha SONET/SDH o resultado é ligeiramente superior (26%) e deve-se sobretudo ao acréscimo de lógica resultante da inclusão dos módulos HDLC e SDH.
Em termos de blocos de memória a ocupação é de apenas 5% para o conjunto composto pelas interfaces de linha e de cliente e de 8% para o conjunto com as interfaces de linha SONET/SDH aproximadamente. Note-se que a FPGA possui 328 blocos de memória de 18Kbit. Assim por exemplo se pretendermos um FIFO com uma profundidade de apenas um pacote Ethernet de 1500 Bytes bastaria apenas usar um bloco de memoria pois: 18*1024/8 = 2043 Bytes! Contudo a tecnologia inerente à FPGA permite a combinação de mais do que um bloco de memória permitindo-nos ter FIFOS de maior ou menor profundidade.

Outro factor determinante a ter em conta no correcto funcionamento das interfaces são os tempos máximos e mínimos aceitáveis de relógios e dos restantes sinais que compõem o sistema. Para isso é necessário definir correctamente nas ferramentas de síntese qual a frequência/período associada a cada linha de relógio. Note-se que todo o trabalho está subjacente num modelo de lógica síncrona em que os estados das variáveis vão sendo alterados apenas nas transições

positivas ou negativas das linhas de relógio.

Nas interfaces de linha a frequência do relógio é de aproximadamente 125MHz, um valor relativamente elevado, o que pressupõem algumas considerações extra quando for feito o mapeamento nesta FPGA.
Um primeiro passo foi colocar a lógica referente à primeira máquina de estados que recebe as tramas do lado da linha nos slices mais próximos dos pinos de saída e entrada na FPGA para esta interface. O mesmo foi feito para a máquina de estados que transmite as tramas para a linha. Nas interfaces de linha SONET/SDH fez-se o mesmo procedimento.
No caso das ferramentas de mapeamento e place&route não atingirem os tempos especificados, as mesmas retornam um ficheiro com informação relativa à identificação do local e dos sinais em que tal não foi possível no sistema. Nestes casos, uma técnica chamada de pipelining e que produz bons resultados consiste na inclusão de flip-flop's no meio destes mesmos sinais [18]. Note-se que as ferramentas fazem os cálculos baseando-se nas distâncias dos caminhos que existem entre flip-flop's. Ao incluirmos flip-flop's nesses caminhos estamos a encurtar esses mesmos caminhos e por conseguinte a facilitar o desempenho do sistema de optimização.
Existem alguns sinais que não necessitam de tempos tão apertados como por exemplo as linhas de endereços, chip-select's e de dados usados pelo processador no interior da FPGA. Todos estes sinais cujos tempos são bastante reduzidos são declarados na ferramenta como false-paths. Assim a ferramenta não se irá preocupar com os mesmos e atingirá melhores resultados com outros sinais cruciais para o sistema.

4.5 Controlo do sistema/interfaces

O MAC do RPR inclui uma entidade de gestão da camada do MAC (MLME) independente que reside num plano separado de gestão. Para a ocorrência de uma correcta operação do MAC do RPR, uma entidade de gestão da estação (SME) deverá estar presente, residindo num plano separado de gestão. As funções exactas do SME não são especificadas no standard 802.17 mas em geral esta entidade pode ser vista como sendo responsável por tais funções como a recolha de informação dependente da camada das diferentes entidades de gestão de camada e similarmente alterando o valor de alguns parâmetros de camadas especificas. A entidade SME tipicamente executa todas estas funções tendo por base determinados sistemas de entidades de gestão e implementa protocolos de gestão normalizados.

A Figura 70 ilustra a relação entre as diversas entidades de gestão.

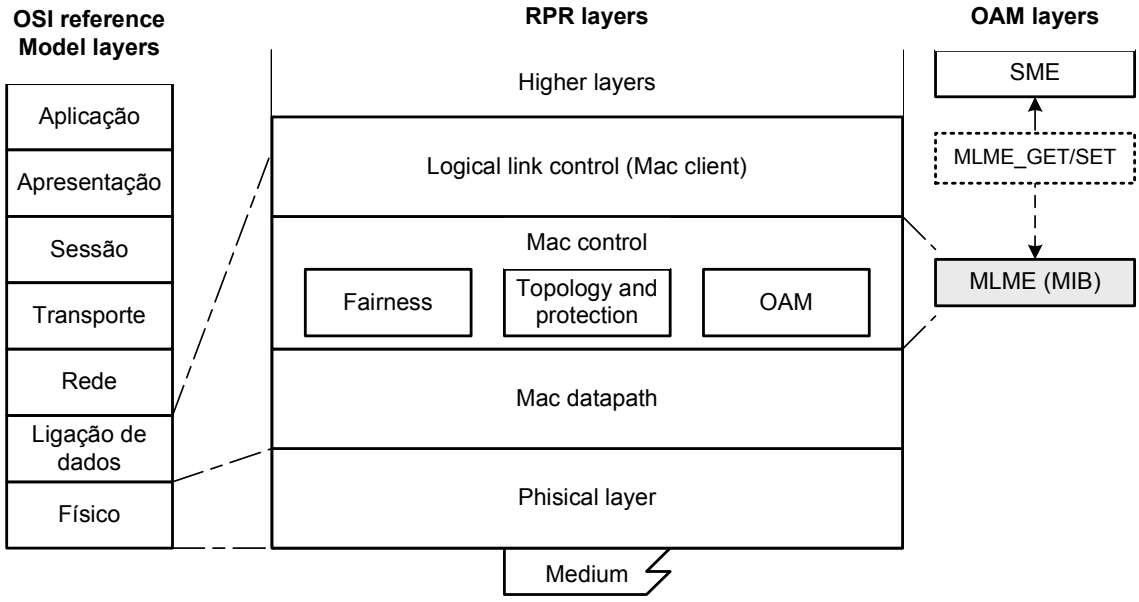


Figura 70 - Relacionamento das entidades de gestão com o modelo ISO/IEC OSI

A interface de serviço de gestão neste modelo define-se como interface SME_MLME.
Os mecanismos de gestão de protecção definidos neste standard são baseados na informação conhecida pela entidade de MAC. Os mecanismos de protecção da camada do PHY são independentes e endereçados apenas no respectivo PHY especificado.

4.5.1 Interfaces de serviço MLME

A informação de gestão específica para cada camada é representada como sendo a informação de gestão base (MIB) para aquela camada. As entidades de gestão de camada do PHY e MAC são vistas como cada uma implementando o MIB para a sua camada. O modelo genérico das primitivas de gestão relacionadas com o MIB expande-se através das interfaces de serviço de gestão de modo a permitir que a interface de serviço da entidade do utilizador receba o valor de um atributo do MIB, active um valor de um atributo do MIB ou que seja notificado sobre eventos assíncronos. As seguintes primitivas são definidas para a camada de gestão do MAC.

- MLME_GET.request
- MLME_SET.request
- MLME_EVENT.indication
- MLME_RESET.request

As primitivas MLME_GET.request, MLME_SET.request, e MLME_EVENT.indication descritas neste capítulo deverão ser suportadas. A primitiva MLME_RESET.request poderá ser suportada

(opcionalmente). Este serviço é utilizado para iniciar as entidades de gestão, o MIB e as entidades de datapath. Poderá incluir uma lista de atributos para que alguns parâmetros sejam iniciados com alguns valores por defeito.

4.5.1.1. MLME_GET.request

A primitiva MLME_GET.request é utilizada para retornar o valor dum atributo da MIB.
Esta primitiva é gerada pelo SME sempre que este desejar obter atributos a partir do MIB do RPR.
Esta primitiva retorna o valor de atributo MIB apropriado se o status for “success”, caso contrario retorna erro.

A semântica que define a primitiva é a seguinte:

MLME_GET.request
(
mib_attribute,
mib_attributevalue,
status
)
Os parâmetros da primitiva MLME_GET.request são descritos abaixo.

mib_attribute
Atributo de leitura no MIB do RPR.

mib_attributevalue
O valor do mib_atribute retornado pela MLME_GET.request.

Status
O status do pedido retornado como um resultado da MLME_GET.request.
SUCCESS—O pedido foi atendido com sucesso.
FAIL—O pedido falhou.

4.5.1.2. MLME_SET.request

A primitiva MLME_SET.request é utilizada para atribuir um novo valor a um atributo da MIB.
Esta primitiva é gerada pelo SME sempre que este desejar submeter atributos no MIB do RPR.
Esta primitiva retorna um status igual a “success” se for bem sucedida, caso contrario retorna erro.

A semântica que define a primitiva é a seguinte:

MLME_SET.indication
(
mib_attribute,
mib_attributevalue,
status
)

Os parâmetros da primitiva MLME_GET.request são descritos abaixo.

mib_attribute
Atributo de escrita no MIB do RPR como definido no anexo E na norma 802.17.
mib_attributevalue
O valor do mib_atribute
status
O status do pedido retornado como um resultado da MLME_SET.request.
SUCCESS—O pedido foi atendido com sucesso.
FAIL—O pedido falhou.

4.5.1.3 MLME_EVENT.indication

A primitiva MLME_EVENT.indication é utilizada pelo MLME para notificar de forma assíncrona o SME sobre possíveis alterações no estado ou de eventos.

Esta primitiva é gerada pelo MLME sempre que este desejar notificar o SME sobre a ocorrência de um evento assíncrono, com os seguintes:
a) qualquer item do ponto 11.2.9 da norma 802.17.
b) Alteração do estado operacional da interface RPR (ponto 13.3.1.2 da norma 802.17.)
c) RPR keep alive timeout (ponto 11.6).
d) Recepção de uma mensagem de echo reply (12.4.1).
e) Recepção de uma trama flush retornada (12.1.5).

A semântica que define a primitiva é a seguinte:
MLME_EVENT.indication
(
event
)

Os parâmetros da MLME_EVENT.indication são descritos abaixo.

event
Um evento RPR como descrito no ponto 13.2.3.3. na norma 802.17 e na Tabela 24:

Meaning	Critical severity	Specified in
Miscabling	YES	11.4.9
Instabilidade da topologia	YES	11.2.9
Inconsistência da topologia	YES	11.2.9
Excedido o número máximo de estações	YES	11.2.9
Configuração de protecção não condiz	YES	11.2.9
Medida de LRTT incompleta	YES	11.2.9
Entrada invalida de topologia	YES	11.2.9
Endereço MAC duplicado	YES	11.2.9
Largura de banda reservada excedida	NO	11.2.9
Alteração do estado operacional da interface RPR	NO	13.3.1.2
RPR Keep alive timeout	NO	11.6
Recepção de mensagem de echo reply	NO	12.4.1
Recepção de trama flush retornada	NO	12.1.5

Tabela 24 - indicação de eventos

4.5.1.4. MLME_RESET.request

A primitiva MLME_RESET.request é utilizada para inicialização das entidades de gestão, dos atributos da informação de MIB das entidades. Esta primitiva é gerada pela entidade SME sempre que esta desejar executar um reset sobre os atributos da informação de MIB para os seus valores de defeito. Retorna um status igual a “success” se o reset for bem sucedido, caso contrário retorna uma indicação de erro.
A semântica que define a primitiva é a seguinte:

MLME_RESET.request
(
list of mib_attributes, // optional
status
)

Os parâmetros da primitiva MLME_RESET.request são descritos abaixo.

List of mib_attributes
Um conjunto de atributos de escrita da informação de MIB do RPR como descrito no anexo D da norma. Quando especificados, apenas estes serão inicializados com os seus valores por defeito, caso contrário todos os valores serão inicializados.

status
O status do pedido retornado como um resultado da MLME_RESET.request.
SUCCESS—O pedido foi atendido com sucesso.
FAIL—O pedido falhou.

4.5.2 Serviços MLME

Esta secção especifica os serviços disponibilizados pela MLME e SME. Estes serviços são descritos de uma forma abstracta e não implicam uma implementação em particular ou o uso de uma qualquer interface descrita. As primitivas da interface de serviço MLME de um modo em geral são do tipo ACTION.request. A SME utiliza os serviços disponibilizados pela MLME através da interface de serviço da MLME.

4.5.2.1 Configuração da interface RPR

A interface RPR possui o seu próprio estado administrativo que especifica o que é desejado para a interface. O estado administrativo autoriza ou proíbe às camadas superiores e inferiores o envio de tramas para e a partir do anel. Quando o estado administrativo da interface RPR (ifAdminStatus) é desactivado (RFC 2863) a estação deverá implementar isto usando pelo menos uma das seguintes considerações:
a) A estação transita para o modo de passthrough, isto é, as tramas são emitidas através do trajecto de trânsito sem decréscimo do ttl ou não são aplicadas quaisquer outras regras de processamento/decomposição de tramas.

b) A estação desabilita caminhos, adiciona/remove interfaces de cliente e para de gerar ou responder a qualquer mensagem de topologia/protecção/fairness.

c) A estação desabilita ambas as interfaces dos dois anéis.

A interface RPR possui o seu próprio estado operacional derivado do:

a) Estado administrativo da interface RPR como definido na RFC 2863

b) Estado operacional para ambas as interfaces dos dois anéis baseados no:

1) Estado operacional da camada do PHY

2) Tempo limite de keepalive em ambas as interfaces dos dois anéis indicando perda de conectividade das interfaces do anel.

3) ID do anel, indicando se a estação se encontra ligada no lado correcto do anel ou não.

O estado operacional da interface RPR estará desabilitado se qualquer uma das situações seguintes ocorrer:

a) Estado administrativo da interface RPR está desabilitado

b) As interfaces do anel estão todas desligadas devida a:

1) A camada PHY está desabilitada

2) O tempo de Keepalive excedeu o limite máximo

3) O ID do anel nas interfaces dos anéis em um dos lados da estação não condiz

A configuração deverá permitir a monitorização do estado de auto-configuração e dos protocolos de descoberta da topologia com o propósito de manutenção. Se, por exemplo, uma condição de má configuração for detectada, deverá ser enviada uma notificação para o sistema de gestão.

O desempenho e algumas medidas de contagem deveram ser suportadas envolvendo contagens de fluxos de tramas seleccionadas. As tramas deverão seguir do cliente para o MAC, MAC para o cliente, PHY para o MAC ou do MAC para o PHY como ilustra a Figura 71. Os nomes das contagens são baseados em medidas locais ou em informação dependente do conteúdo da trama.

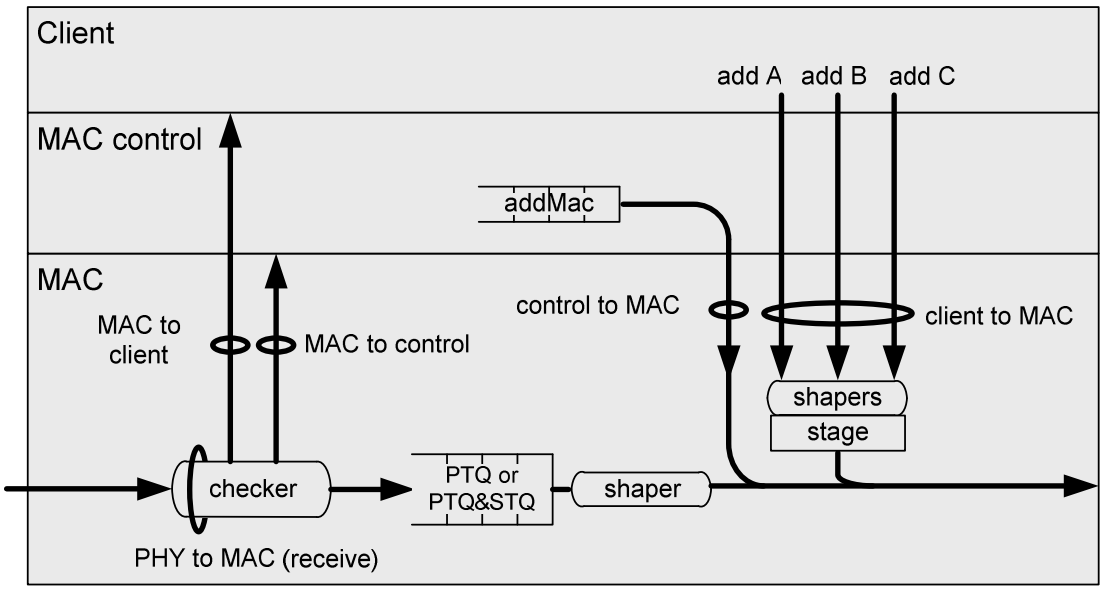


Figura 71 - locais para contadores de medidas

As estatísticas do RPR que são guardadas dependem das propriedades da trama processada. O formato geral de um contador é composto pela localização, período, direcção, endereço, classe e unidades. Esta classificação está sumariada na Tabela 25 e Tabela 26.

Type	Variable	Values	Description
Location	<place>	Span, Client, MAC Control	Places where statistics are collected
Reporting period	<period>	Current interval, Past intervals, Total intervals, Running counters	Time periods for which statistics are reported
Address Type	<addr>	Unicast, Multicast+Broadcast, Broadcast	Frame types sent/received
Class	<class>	ClassA, ClassB-CIR, ClassB-EIR, ClassC	Service class
Units	<units>	Octets, Frames	What unit is being counted
Direction	<dir>	In,Out	Whitch direction from the point of view of the “higher layer”
Error type	<err>	TooLong, TooShort, BadFcs, BadHec, TtlExpired, SelfSourced, PmdAborted	Error type

Tabela 25 - Definições estatísticas

O desempenho dos parâmetros deverá ser acumulada em intervalos de cinquenta minutos e após 96 intervalos (24 horas) são armazenados por um agente. Os parâmetros de performance continuaram a ser coleccionados mesmo quando a interface estiver em baixo.

Group	MAC counter instances	MAC counter names
MAC client frames	Reporting period Direction Address type Class / subclass	RprClient<period><dir><addr><class>Frames e.g., rprClientCurrentInUcastClassAFrames rprClientIntervalOutMcastClassCFrames
MAC client bytes	Reporting period Direction	RprClient<period><dir><addr><class>Octets e.g., rprClientCurrentInUcastClassAOctets

	Address type Class / subclass	rprClientIntervalOutMcastClassCOctets
Span frames	Reporting period Direction Address type Class / subclass	RprSpan<period><dir><addr><class>Frames e.g., rprSpanCurrentInUcastClassAFrames rprSpanIntervalOutMcastClassCFrames
Span bytes	Reporting period Direction Address type Class / subclass	RprSpan<period><dir><addr><class>Octets e.g., rprSpanCurrentInUcastClassAOctets rprSpanIntervalOutMcastClassCOctets
Span errors	Reporting period Error type Address type Class / subclass	RprSpan<period><Error><err> e.g., rprSpanCurrentErrorTtlFrames rprSpanIntervalErrorBadHecFrames

Tabela 26 - Estatísticas do MAC

4.5.3 Interface com microprocessador

Na interface com o microprocessador existe a necessidade de troca de informação entre o dispositivo FPGA e o powerPC da Motorola, da placa desenvolvida para o sistema. Como primeira abordagem e reutilizando algum know-how já adquirido optou-se por utilizar o powerPC presente na placa ao invés dos processadores contidos na FPGA. Contudo no futuro existe a possibilidade de migrar para os processadores contidos na FPGA. Para isso não será necessário fazer qualquer tipo de alteração ao Hardware mas sim desenvolver algum FirmWare para o efeito.

Para a troca de informação entre o FPGA e o processador existem linhas de dados e de endereços na placa que ligam ambos os dispositivos. Na FPGA foram também criados um conjunto de registos de leitura e de escrita para auxílio ao processador através destas mesmas linhas. Assim se for necessário, por exemplo, aceder a alguma estatística de contadores a operação pode ser realizada através de um registo de leitura para o processador. Se por outro lado for necessário executar alguma configuração sobre o MAC do RPR, interfaces ou um qualquer outro módulo, a operação pode ser feita através de um registo de escrita que por sua vez irá accionar a respectiva configuração do módulo na FPGA.

A lógica implementada neste tipo de interface com o microprocessador baseia-se em decoder's, latche's, buffer's tristate, portas OR, AND e inversores.

4.6 Implementação das interfaces ATM

A tecnologia ATM não se baseia num meio físico de transporte específico, é compatível com as actuais redes físicas (par cobre, coaxial, fibra). Deste modo, pretende-se incluir no sistema capacidades ATM, interligando assim esta tecnologia com um sistema de transporte de dados SDH. Mais concretamente vamos utilizar a interface SDH/STM-1 mencionada no ponto 2.1 e inserir as células ATM como descrito na recomendação [12] do ITU.

O delineamento de célula consiste na identificação dos limites das células ATM. A célula ATM contém um campo de HEC (Header Error Control) que irá servir para se conseguir o delineamento de célula. Pretende-se que o sinal ATM seja transportado de um modo transparente a partir de uma qualquer interface como descrito anteriormente. Será então usado um algoritmo de scrambling que permita uma maior segurança e robustez do mecanismo de delineamento de célula. Este mecanismo irá baralhar a informação contida no payload da célula através da função contida no scramber, permitindo assim um melhor desempenho da transmissão das células. O scrambler não deverá alterar o header da célula pois isso implica um mau funcionamento do mecanismo de delineamento de célula.

4.6.1 Header Error Control (HEC)

O campo de Header Error Control (HEC) incide sobre o cabeçalho da célula ATM. O código de correcção de erros utilizado para esta função é capaz de corrigir erros de um bit no header e também de detectar erros superiores a um bit.

A transmissão terá de computar e preencher o campo de HEC da célula ATM. Na recepção existem dois modos de operação como demonstrado na Figura 72:

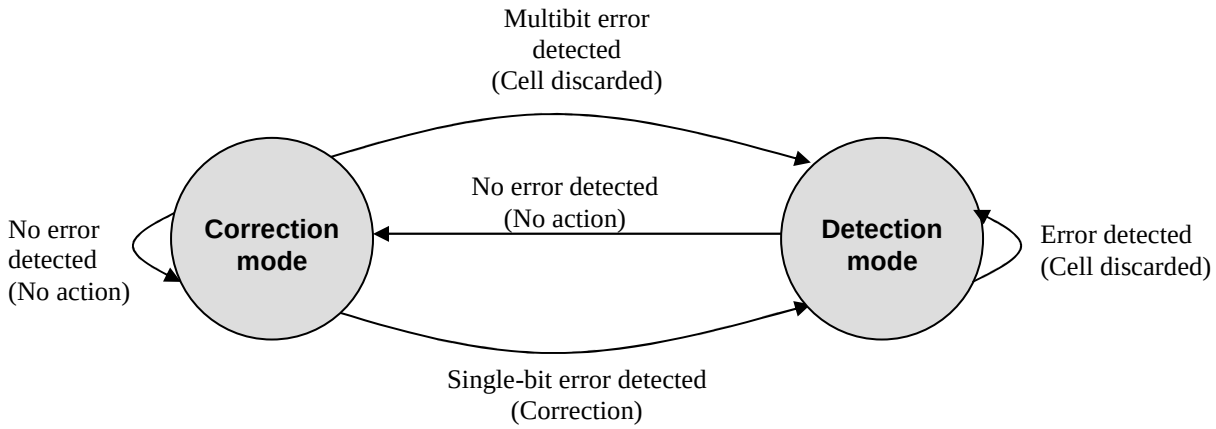


Figura 72- HEC: Operação de recepção

Por defeito, temos activo o modo capaz de corrigir um bit errado do cabeçalho da célula ATM sempre que o mecanismo de delineamento de célula se encontrar no estado de SYNC. Cada cabeçalho de célula é examinado individualmente e no caso de ocorrência de erros é despoletada uma de duas acções possíveis. A acção despoletada depende do estado em que se encontra o receptor. No caso do modo de correcção apenas um erro de bit pode ser corrigido transitando de seguida o receptor para o modo de detecção. No modo de detecção todas as células detectadas com erros no cabeçalho são descartadas. Quando for encontrado um cabeçalho sem qualquer bit errado o receptor transita de novo para o modo de correcção. O termo "no action" na Figura 72 significa que não irá ocorrer qualquer correcção e que nenhuma célula será descartada.

O transmissor calcula o valor de HEC ao longo do todo o cabeçalho da célula ATM e insere o resultado num campo apropriado. A notação utilizada para descrever o HEC é baseada nas propriedades inerentes aos códigos cíclicos [7][20][21]. O campo de HEC consiste numa sequência de 8-bit resultante duma divisão modulo dois pelo polinómio gerador x8 + x2 + x + 1 sobre o conteúdo do cabeçalho da célula ATM excluindo o campo de HEC. Na transmissão o registo que guarda o valor do resto da divisão é posto a zero sendo alterado pela operação de divisão do cabeçalho excluindo o campo de HEC pelo polinómio gerador. O resto da divisão representa o HEC do cabeçalho da célula.
Para melhorar significativamente a performance do algoritmo de delineamento de célula no caso de ocorrência de desvios (slips) em alguns bits da célula, é recomendável que se tome o seguinte procedimento:

- Os bits resultantes na operação de computação do HEC são adicionados (modulo 2) com um padrão de 8-bits antes de serem colocados no campo de HEC do cabeçalho de célula ATM.
- O padrão recomendado é "01010101" (MSB...LSB)
- O receptor deverá subtrair (adição modulo 2) exactamente o mesmo padrão da sequência de 8-bits do HEC antes de calcular o sindroma do cabeçalho.

A operação não afecta de modo nenhum as capacidades de detecção/correcção do HEC.
Como exemplo, se os primeiros quatro octetos do cabeçalho forem zeros o cabeçalho gerado será "0000 0000 0000 0000 0000 0000 0000 0000 0101 0101".

4.6.2 Operação de Scrambler

Para garantir o desempenho do mecanismo de delineamento de célula nos sistemas baseados em SDH é aconselhável a utilização de um mecanismo adicional de scrambing. A norma [12] do ITU define o polinómio para a função do scrambler.

A operação do scrambler relativamente ao diagrama de estados do delineamento de célula do HEC é o seguinte:

- O scrambler baralha apenas os bits de informação de dados da célula ATM.
- Durante os cinco octetos do cabeçalho o mecanismo de scrambler é suspenso e o seu estado armazenado.
- No estado de HUNT o descrambler é desactivado.
- Durante os primeiros cinco octetos do cabeçalho o scrambler é desactivado e o seu estado guardado.
- Nos estados PRESYNC e SYNC o descrambler é ligado durante os 48 bytes correspondentes ao payload da célula.
- No arranque, os primeiros 43 bits do payload da célula serviram para sincronizar o scrambler e descrambler, o que faz com que esta primeira célula saia corrompida.

4.6.3 Delineamento de célula

O delineamento de célula resulta do cálculo da correlação entre os bits do header da célula a proteger (32 bits) e os bits de controlo introduzidos no header pelo mecanismo de HEC usando para isso um algoritmo de CRC baseado no polinómio x8 + x2 + x + 1.

A Figura 73 demonstra um diagrama de estados do mecanismo de delineamento de célula.

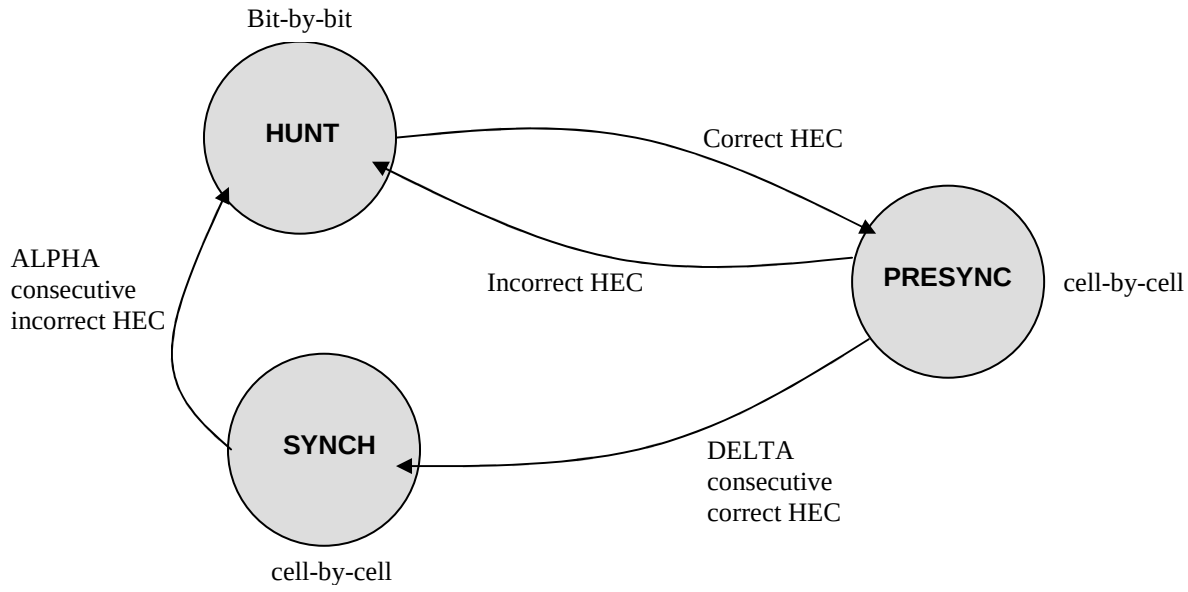


Figura 73 - diagrama de estados de delineamento de célula

Estado HUNT: No estado de HUNT é feita uma verificação sistemática para todos os bits da célula recebida até ser encontrado um HEC. Em seguida a máquina salta para o estado de PRESYNC. Note-se que a partir deste momento já existe uma referência para o sítio da célula onde eventualmente se encontra o HEC.

Estado PRESYNC: No estado de PRESYNC o processo de delineamento é executado célula a célula, tendo por base a referência para a posição do HEC determinada no estado anterior.
Assim sendo a maquina está constantemente a verificar se na posição referida se encontra um HEC válido. O processo repete-se sistematicamente até que o HEC seja confirmado DELTA vezes consecutivas, avançando em seguida para o estado de SYNC. No caso de ser encontrado um HEC incorrecto, o processo retorna ao estado de HUNT. Assim sendo, o número total consecutivo de HECs validos necessários para o processo avançar do estado de HUNT para SYNC é de DELTA+1.

Estado SYNC: No estado de SYNC o processo de delineamento de célula será considerado como perdido sempre que um HEC incorrecto seja obtido pelo menos ALPHA vezes consecutivas.

Os parâmetros ALPHA e DELTA são escolhidos para que o processo de delineamento de célula seja o mais robusto e seguro possível. O parâmetro ALPHA determina a robustez contra falsas indicações de desalinhamentos relativos a erros de bit no canal. O parâmetro DELTA determina a robustez contra um mau delineamento de célula no processo de resincronização do sistema.
Para um sistema SDH os valores escolhidos na recomendação [12] são de ALPHA = 7 e DELTA = 6. Estes valores são obtidos, segundo a mesma recomendação, utilizando várias probabilidades de erro e analisando em seguida o desempenho do algoritmo de delineamento de célula descrito anteriormente para diferentes valores de ALPHA e de DELTA. Esses resultados representam valores médios para um sistema SDH assumindo que os 8 bits do HEC são utilizados.

4.6.4 Células IDLE

As células Idle serão usadas também para o algoritmo de delineamento de célula e verificação do HEC na recepção.
As mesmas são inseridas e descartadas para que os débitos necessários para o canal de transporte de dados sejam cumpridos. As células IDLE são identificadas através do cabeçalho como demonstrado na Tabela 27.

	Octed 1	Octed 2	Octed 3	Octed 4	Octed 5
Header pattern	0000 0000	0000 0000	0000 0000	0000 0001	HEC=Valid code 0101 0010
Nota 1. O conteúdo do payload da célula é repetido 48 vezes com o valor “0101 0010” Nota 2. As células IDLE não passam para a camada ATM.					

Tabela 27 - célula IDLE

4.6.5 Desenvolvimento das interfaces ATM

Na Figura 74 podemos observar os blocos que processam as células ATM entre a camada SDH e ATM. Na interface de recepção é necessário extrair as células ATM que existem dentro da trama SDH, usando para isso a máquina de delineamento de célula ATM. Em seguida as células passam por um descrambler ATM como descrito no ponto 4.6.3, sendo armazenadas num FIFO que serve de fronteira para a camada ATM. Existe também uma máquina (sm_idle_cell_remove) que remove as células IDLE impedindo que estas sejam desnecessariamente escritas no FIFO.

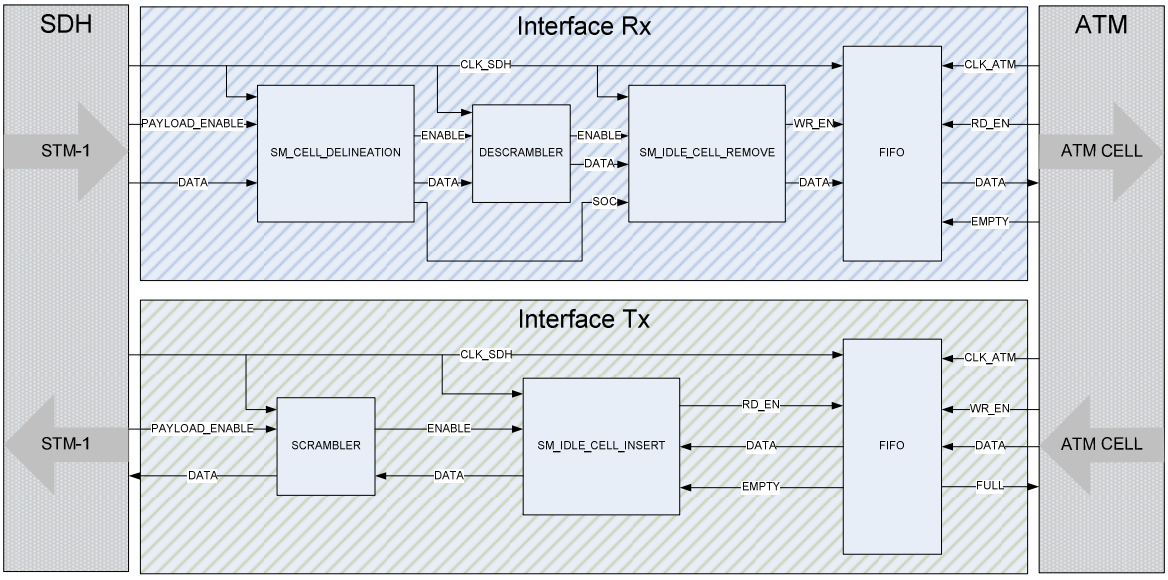


Figura 74 - Digrama de blocos das interfaces ATM

Na interface de transmissão é necessário inserir células ATM na trama SDH. No caso de a camada ATM não enviar um débito de células suficiente para preencher a carga paga da trama SDH, é necessário inserir células IDLE como explicado no ponto 4.6.4. Para isso, existe a máquina de inserção de células IDLE. As células lidas do FIFO fronteira com a camada ATM são primeiro processadas pelo scrambler como descrito no ponto 4.6.4.

Depois de terminado o desenvolvimento dos códigos HDL das interfaces, procedeu-se ao mapeamento do mesmo na FPGA XC2V3000 da Xilinx. O resultado final relativamente ao número de slices ocupados na FPGA para as interfaces ATM é de aproximadamente 3970 slices, ou seja 24% de ocupação do dispositivo. Em termos de blocos de memória a ocupação é de apenas 2 blocos de 18Kbit em que cada bloco corresponde a cada um dos FIFOS de transmissão e recepção.

4.7 Testes Funcionais

4.7.1 Testes RPR

Para o teste das interfaces RPR foi usada uma placa desenvolvida no decorrer do projecto SIRAC, apresentado no ponto 3 (Figura 75). A placa foi projectada e fabricada de raiz com o objectivo de possibilitar o desenvolvimento e teste do MAC do RPR e suas interfaces. Para esta dissertação de mestrado apenas se planeou a parte correspondente às interfaces RPR. Para melhor entendimento da norma 802.17 no que respeita ao MAC do RPR, ver [4].

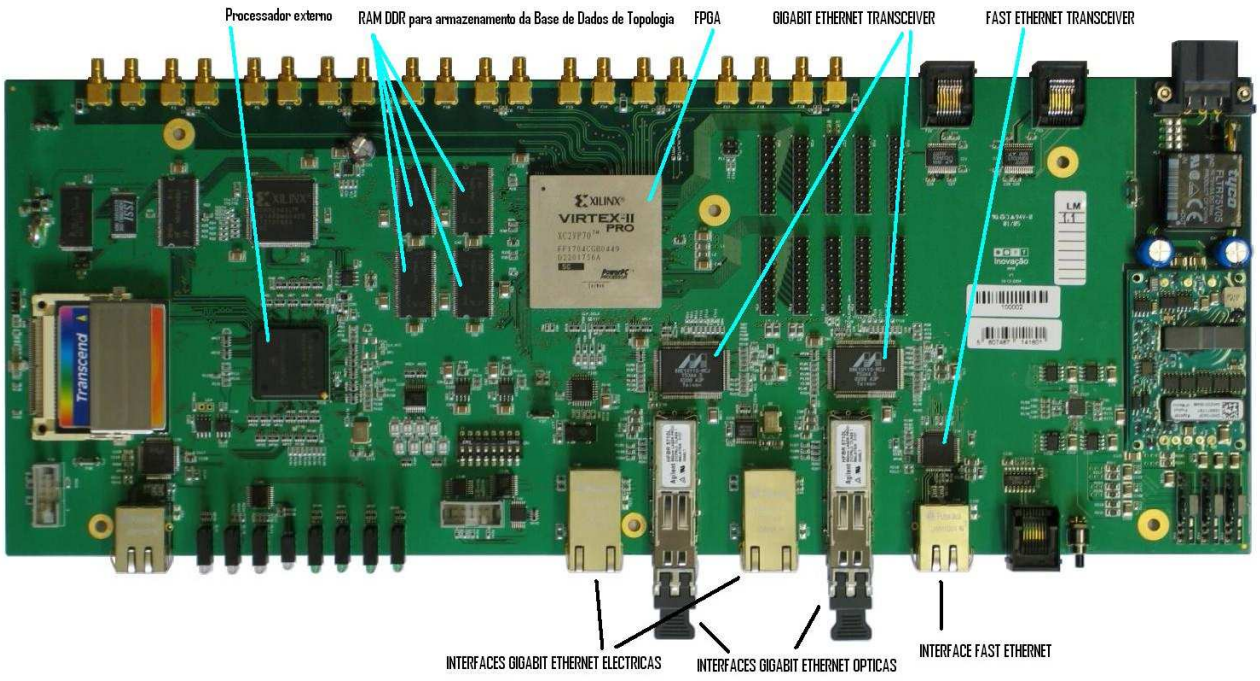


Figura 75 - Placa de desenvolvimento RPR

Os principais elementos que compõem esta placa são um processador externo, memória RAM DDR, a FPGA XC2VP70 da Xilinx, duas interfaces Gigabit Ethernet e duas Fast Ethernet.

Numa primeira fase foi usado um Logic Analyser para analisar os sinais de saída nos pinos I/O da FPGA e um gerador de ondas para simular o relógio de sincronismo. Para isso foi desenvolvido código de teste (test-bench) sintetizável na FPGA. O principal objectivo é emular o envio de um pacote Ethernet ou RPR consoante a interface a testar. Por observação dos resultados obtidos no Logic Analyser e na simulação consegue-se testar o funcionamento das interfaces. Os testes foram muito úteis identificando problemas. Em muitos casos, verificou-se a necessidade de

recorrer de novo à ferramenta de simulação para analisar possíveis discrepâncias existentes entre os resultados de simulação e de teste. De seguida foi necessário identificar o porquê das diferenças observadas, proceder à sua correcção e novamente simular e testar o módulo desenvolvido seguindo o fluxo de desenvolvimento explicado no ponto 4.2. Todo este trabalho teve alguma morosidade e só com alguma prática é que se consegue melhorar todo o processo de desenvolvimento, simulação e de teste.

Usámos também um outro método auxiliar que nos permitiu acelerar significativamente o teste dos módulos. Este método consiste simplesmente na utilização do microprocessador da Motorola presente na placa para retorno de dados de contadores de tramas colocados em diferentes locais estratégicos do sistema. Estes contadores podem ser do tipo tramas validas, tramas inválidas, CRC errado etc.… um conjunto de possibilidades que nos auxilia o teste dos módulos já implementados na placa. Através destes contadores consegue-se saber se as interfaces estão a operar correctamente. Finalmente fizemos uso do equipamento de teste N2X da Agilent para gerar tramas Ethernet para as interfaces MII e GMII. Foram feitos vários testes às interfaces variando o débito e tamanho dos pacotes como se pode ver na Figura 76.

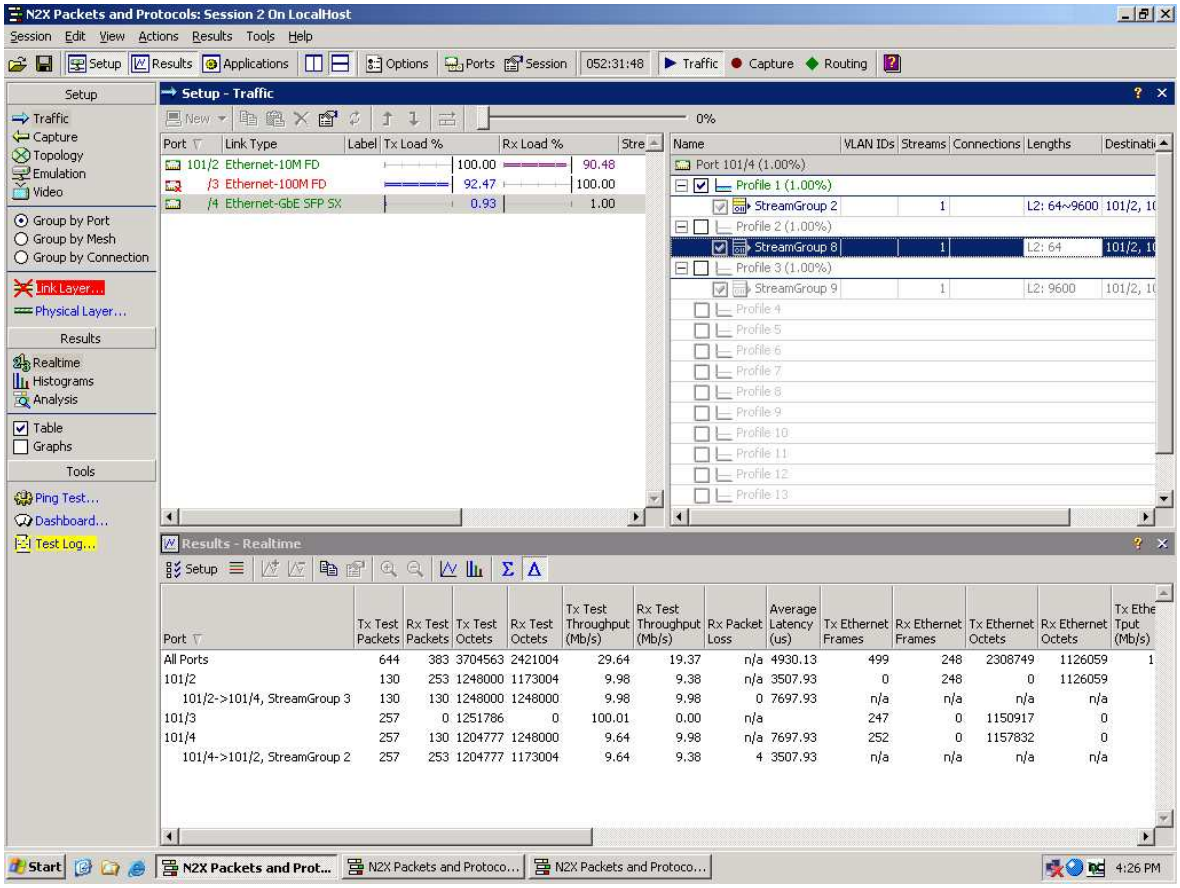


Figura 76 - Envio de tramas Ethernet através do N2X

4.7.2 Testes ATM

Para o teste das interfaces ATM, utilizámos uma nova carta com duas interfaces STM-1, um Network Processor com funcionalidades ATM e uma ligação a um backplane. Esta ligação permite que a carta possa ser introduzida num qualquer slot de um equipamento modular SDH. No futuro pretende-se que a carta RPR apresentada no ponto anterior, possua também uma ligação ao backplane compatível com este mesmo equipamento SDH. A modularidade do equipamento em que se pode alternar aleatoriamente a introdução de cartas nos diversos slots permite ao operador grande flexibilidade no dimensionamento da sua rede. Permite-lhe também escolher quais as tecnologias mais apropriadas para diferentes cenários.

Foi utilizada uma placa SDH/ATM como ilustra a Figura 77:

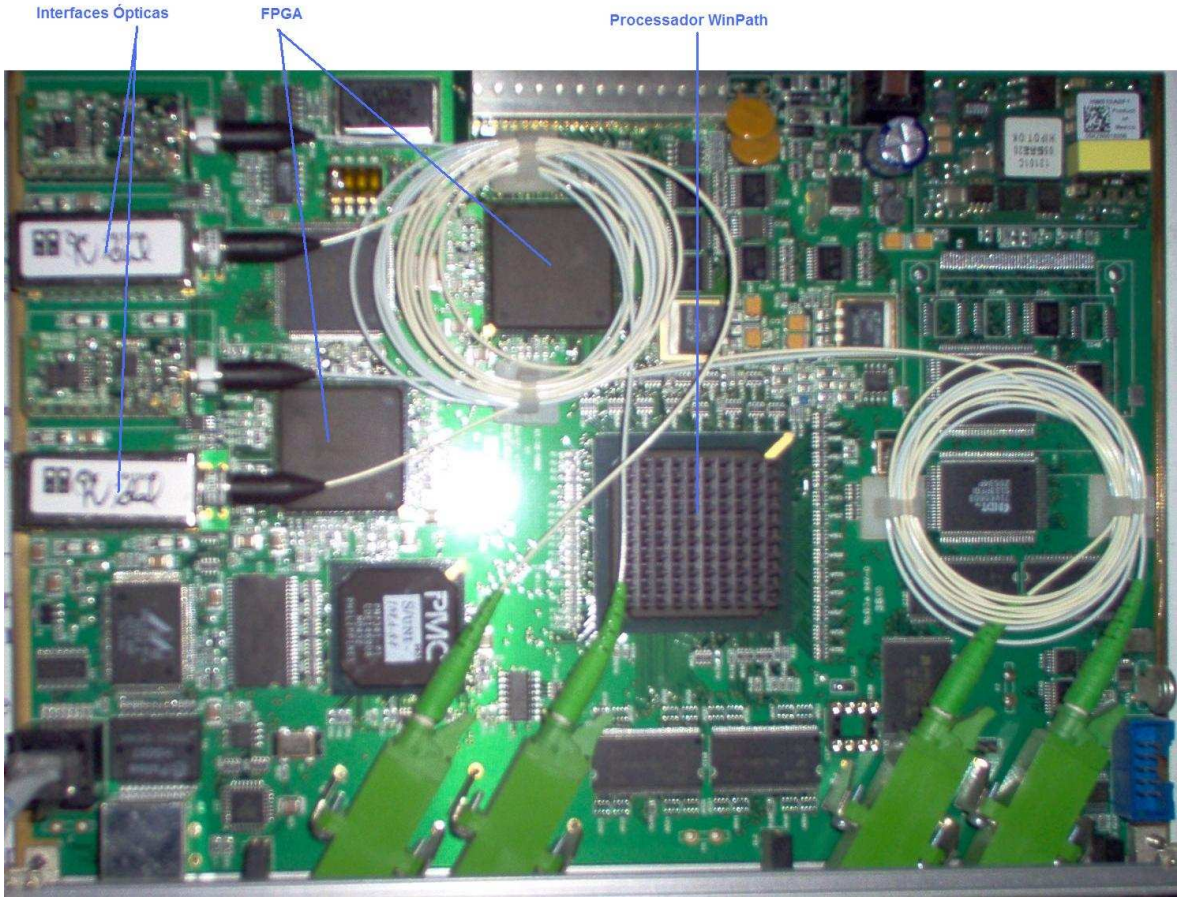


Figura 77 - Placa de desenvolvimento ATM

Os principais elementos que constituem a placa são a presença de duas interfaces SDH ópticas, um processador WinPath da Wintegra e duas FPGAs XC2V3000 da família Virtex II da Xilinx. Estas FPGAs possuem a mesma arquitectura da FPGA XC2VP70 usada nas interfaces RPR com

a excepção de não possuírem processadores no interior da FPGA. Apenas uma FPGA está ligada às interfaces ópticas SDH e ao processador WinPath, sendo esta a FPGA a usar para implementação das interfaces ATM com o PHY. Fizemos uso do equipamento de teste Aurora Forte da TrendCommunications e inserimos uma trama STM-1 com células ATM no seu interior. Em seguida ligamos as interfaces ATM aqui descritas e monitorizamos os valores de contadores propositadamente introduzidos para contagem de células válidas, inválidas e IDLE. Para além dos contadores utilizámos também pinos de teste com informação do estado dos FIFOS (empty e full). O facto de possuirmos um equipamento de teste facilitou-nos o desenvolvimento destas interfaces ATM comparativamente com as interfaces para o módulo MAC do RPR em que foi necessário recorrer a técnicas exaustivas de simulação e a desenvolvimento de código de teste HDL (test-bench).

Configuramos o mesmo para o envio de um link STM-1 como ilustra a Figura 78:

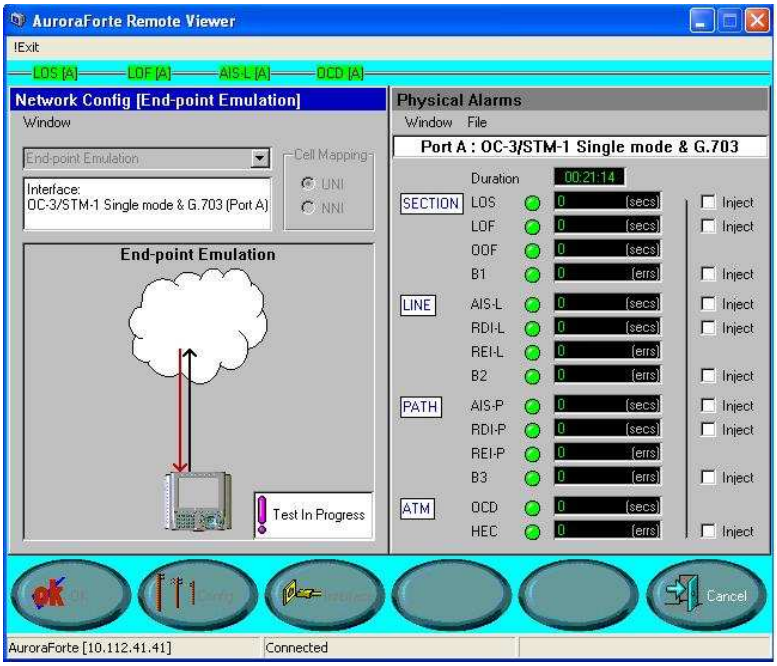


Figura 78 - Esquema de teste AuroraForte

Configuramos também o envio de 200047 células/segundo de dados ATM com o valor de VP/VC de 0/32 (ver Figura 79) e com uma estrutura de dados padronizada PRBS.

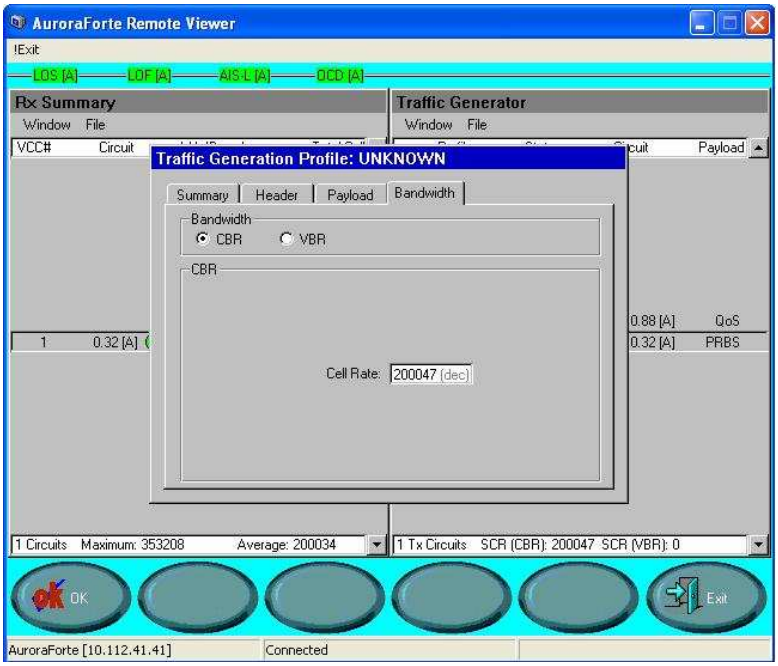


Figura 79 - Número de células ATM enviadas

Finalmente abrimos a janela de células ATM recebidas no equipamento Aurora Forte e constatámos que o valor recebido era o esperado 200047 células/segundo como ilustra a Figura 80:

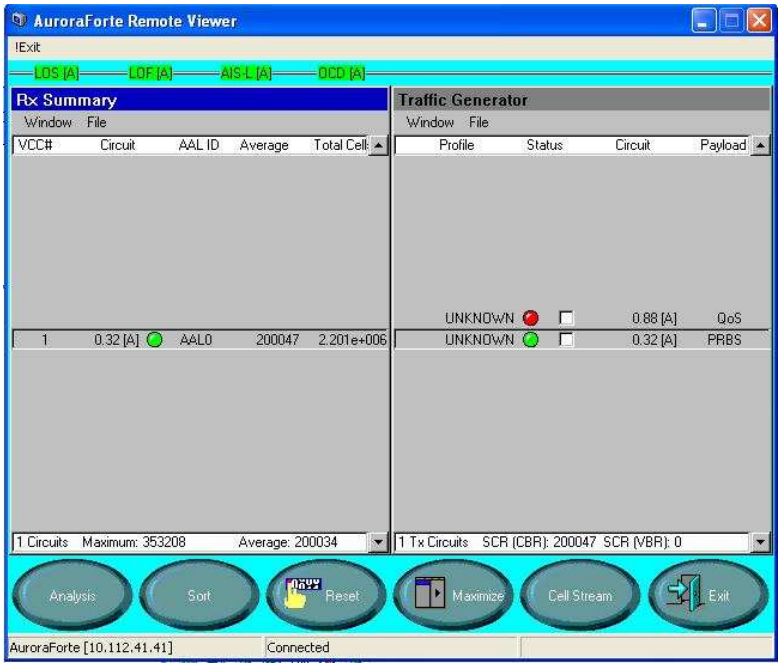


Figura 80 - Células ATM recebidas no AuroraForte

5 Conclusões e trabalho futuro

As redes de telecomunicações actuais, estão sujeitas a um novo paradigma de transporte de dados devido ao aumento do tráfego internet verificado no decorrer dos últimos anos. O tráfego de voz tem vindo a ser substituído pelo tráfego internet de pacotes IP o que torna as tecnologias usadas nas redes de telecomunicações mais antigas inadequadas e em alguns casos obsoletas. Surge assim, a necessidade de substituir ou actualizar alguns dos equipamentos que constituem estas redes.

Esta dissertação de mestrado tem como principal missão a implementação de diversas interfaces para sistemas desta natureza, dotando-os de tecnologias capazes de operar sobre o novo paradigma de transporte de dados, e de simultaneamente manterem a interoperabilidade com os equipamentos mais antigos presentes ainda no mercado das telecomunicações.

O SONET/SDH tem sido uma das tecnologias mais utilizadas para as redes de transporte de dados nos últimos anos. Embora sendo uma tecnologia bastante fiável, esta apresenta algumas lacunas que vêm sendo ultrapassadas com o surgimento de outras tecnologias. O ATM por exemplo, quando misturado com o SONET/SDH permite o envio de tráfego com diversos níveis de qualidade de serviço entre outras funcionalidades. No entanto estas tecnologias não são orientadas para o tráfego de pacotes, ao invés do RPR pensada de raiz para este fim. Existe assim uma grande diversidade de tecnologias nas redes de telecomunicações actuais, sendo necessária a coexistência e uma boa interoperabilidade entre todas elas. Nesta dissertação de mestrado implementámos interfaces para as tecnologias SDH, ATM e RPR possibilitando o desenvolvimento de uma rede de telecomunicações bastante versátil.

As interfaces foram desenvolvidas através de circuitos de electrónica reconfigurável (FPGA's). Estes dispositivos são compostos por um array de logica reconfigurável capaz de implementar os protocolos de telecomunicações desejados. Numa primeira fase optámos por desenvolver as interfaces em SystemC, por ser uma linguagem de alto nível o que nos permitiu ter uma certa abstracção do hardware da FPGA. Esta fase teve como principal objectivo descrever os requisitos dos sistemas, quais os sinais a usar, as máquinas de estados envolvidas, entre outras funcionalidades. Contudo para a correcta programação de um FPGA ao contrário de um ASIC, é necessário orientar a fase de desenvolvimento do código à a arquitectura do dispositivo. Existem alguns condicionalismos como a velocidade e área da FPGA escolhida. Foi nesta fase em que decidimos converter o código SystemC em código HDL através de um software de conversão. O código resultante já pode ser usado na programação do FPGA, mas antes tem de passar por um conjunto de ferramentas que o iram converter nas portas lógicas existentes no dispositivo. Estas

ferramentas conseguem optimizar o código para que área ocupada da FPGA seja minimizada e a velocidade de funcionamento do circuito maximizada.
O fluxo do projecto necessário para o desenvolvimento do código das interfaces primeiro em SystemC e depois em código HDL incluiu algum trabalho de simulação. Foram desenvolvidos algumas porções de código de teste HDL também sintetizável em FPGA que foram úteis na fase de simulação e também mais tarde na fase de prototipagem em FPGA.

O conhecimento adquirido no decorrer do desenvolvimento desta dissertação de mestrado foi bastante enriquecedor quer ao nível do estudo de protocolos de telecomunicações como ATM, RPR, SONET/SDH, Ethernet, HDLC bem como a sua implementação em FPGA. Desta dissertação de mestrado resultou um fluxo de projecto bastante interessante, bem como um conjunto de interfaces para diferentes sistemas de telecomunicações que podem ser combinadas entre si gerando novos produtos e serviços. Os resultados foram bastante satisfatórios, na medida em que a placa ATM resultou num produto comercial muito interessante, adquirido por operadores nacionais e estrangeiros.
Quanto à solução RPR o resultado foi positivo na medida em que se conseguiu implementar uma tecnologia de nova geração ainda em fase de normalização, o que se traduziu em conhecimento de valor acrescentado. Contudo não nos foi possível ainda fazer testes de rede conclusivos ao RPR que nos permitam explorar as suas totais capacidades. Os testes elaborados foram apenas testes funcionais e não de velocidade. Contudo as interfaces RPR PacketPhy ou SONET/SDH desenvolvidas, em conjunto com o módulo HDLC podem ser combinadas com um MAC RPR de um qualquer fabricante existente no mercado podendo também dar origem a um novo produto.
O facto de se ter desenvolvido código HDL para sistemas reconfiguráveis (FPGA), permite que se proceda a eventuais correcções ou actualizações que se venham a revelar necessárias de futuro. A metodologia utilizada é por isso bastante interessante para um qualquer fabricante de equipamentos desta natureza pois permite um contínuo aperfeiçoamento do produto. Saliente-se que neste sector de mercado o tempo de vida das tecnologias é bastante diminuto, sendo necessário um contínuo desenvolvimento de novas tecnologias, aperfeiçoando a gama de produtos em tempo útil para que os mesmos não se tornem obsoletos. O fluxo de projecto apresentado nesta dissertação de mestrado permite assim uma grande flexibilidade no desenvolvimento de novos produtos.

Como trabalho futuro poderá ser explorada a possibilidade de desenvolvimento de uma rede piloto assente na tecnologia RPR, que permita fazer testes de desempenho comparando com outras tecnologias. Poderemos também tentar desenvolver outras que se vão apresentando como variantes das tecnologias apresentadas nesta dissertação de mestrado. Por exemplo no caso ATM existem alguns fabricantes desta tecnologia que aplicaram os conhecimentos aqui adquiridos com o intuito de desenvolver equipamentos em tecnologia MPLS.

Surgem também novas normalizações que sugerem o aparecimento de tecnologias orientadas ao pacote na rede de transporte de dados que substituam o SONET/SDH para além do RPR. A maioria dessas tecnologias usa a tecnologia Ethernet abordada neste trabalho para o transporte dos dados na rede core. A principal vantagem prende-se com o exagerado número de interfaces Ethernet presente nas redes de telecomunicações por todo o mundo bem como o baixo custo das mesmas. O facto de grande parte do tráfego presente nas redes de telecomunicações ser Ethernet é também um factor bastante decisivo. Portanto embora seja muito difícil prever quais as tendências que iram existir de futuro no mercado das telecomunicações global, podemos sem duvida pensar que a tecnologia SONET/SDH irá ser gradualmente substituída por novas tecnologias orientadas ao tráfego de pacote na rede core.

6 Referências

[1] IEEE 802.17 WORKING GROUP, “Local and metropolitan area networks IEEE Draft P802.17/D2.7: Resilient packet ring (RPR) access method and physical layer specifications.”, IEEE, New York, 2003.

[2] IEEE 802.3 WORKING GROUP, “Local and metropolitan area networks IEEE standards: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.”, IEEE std 802.3, New York, 2002.

[3] IEEE 802.17 WORKING GROUP, “Local and metropolitan area networks IEEE Draft P802.17a/D1.0: Resilient packet ring (RPR) access method and physical layer specifications.”, IEEE, New York, 2003.

[4] Miguel Osório – “Implementação de IP sobre novas camadas físicas”, dissertação de mestrado, Universidade de Aveiro.

[5] Howard W. Johnson, “Fast Ethernet: drawn of a new network”,Prentice Hall PTR, New Jersey, 1996

[6] Jayant Kadambi, Ian Crayford, Mohan Kalkunte, “Gigabit Ethernet: Migrating to high-bandwidth LANs”, Prentice Hall PTR, New Jersey, 1998

[7] Chris Borrelli, “IEEE 802.3 Cyclic Redundancy Check”, Xilinx XAPP209 (v1.0), March 23, 2001

[8] Reportium XXI Consulting ©; O Mercado das TMT’s em Portugal; Edição 2003/2004

[9] Ferreira da Rocha, Doutor José Rodrigues – “Redes Ópticas - redes da primeira geração – o SONET/SDH”, apontamentos da disciplina de Redes Ópticas, Universidade de Aveiro.

[10] IETF RFC 1662: PPP in HDLC Like Framing, W. Simpson, July 1994.

[11] IETF RFC 2615: PPP over SONET/SDH, A. Malis, W. Simpson, June 1999.

[12] ITU-T Recommendation I.432.1, “B-ISDN user-network interface – Physical layer specification: General characteristics.”, INTERNATIONAL TELECOMMUNICATION UNION

[13] ITU-T Recommendation G.707/Y.1322, “Network node interface for the synchronous

digital hierarchy (SDH).", INTERNATIONAL TELECOMMUNICATION UNION

[14] OIF System Packet Interface Level 3 (SPI-3): OC48 System Interface for Physical and Link Layer Devices. Implementation agreement: OIF-SPI3-01.0, June 2004.

[15] Open SystemC Initiative. Functional Specification for SystemC™ 2.0. Update for SystemC 2.0.1, Version 2.0-Q April 2002 (//www.systemc.org/).

[16] "SystemC User's Guide", Version 2.0.1, 2002.

[17] T.Grotker, S.Liao, G. Martin, Stuart Swan, "System Design with SystemC", Kluwer Academic, 2002

[18] Synopsys Inc., "FPGA Compiler II / FPGA Express HDL Reference Manual", Version 1999.05.

[19] Xilinx® Integrated Software Environment (ISE 6.2i).

[20] Rajesh Nair, Gerry Ryan, Farivar Farzaneh "A Symbol Based Algorithm for Hardware Implementation of Cyclic Redundancy Check (CRC)", Bay Networks, Inc, Santa Clara

[21] Giuseppe CAmpobello, Giuseppe Patanè, Marco Russo "Parallel CRC Realization", IEEE Transactions on Computers, VOL. 52, NO. 10. October 2003

[22] Rainer Handel, Manfred N Huber, Stefan Schroder, "ATM Networks Concepts, Protocols, Applications", Addison-Wesley, 1998

[23] Stamatios V. Kartalopoulos, "Understanding SONET/SDH and ATM", IEEE PRESS, 1999

[24] J M Pitts, J A Schormans, "Introduction to IP and ATM Design and Performance", John Wiley & Sons, Ltd, England, 2000

[25] Uyless Black, "MPLS & Label Switching Networks", Prentice Hall, New Jersey, 2002

[26] Rui L. Aguiar, Bruno Marques, Miguel Osório, "Developing an IEEE 802.17 station in SystemC", FEUP REC2006 - II Jornadas Sobre Sistema Reconfiguráveis, 2006

[27] ATM, MFA, IP/MPLS Forum http://www.ipmplsforum.org/

[28] Pedro Mendes – “On-board fast packet switching for ATM”, M.Sc. in Telematics, University of Surrey, 1995.

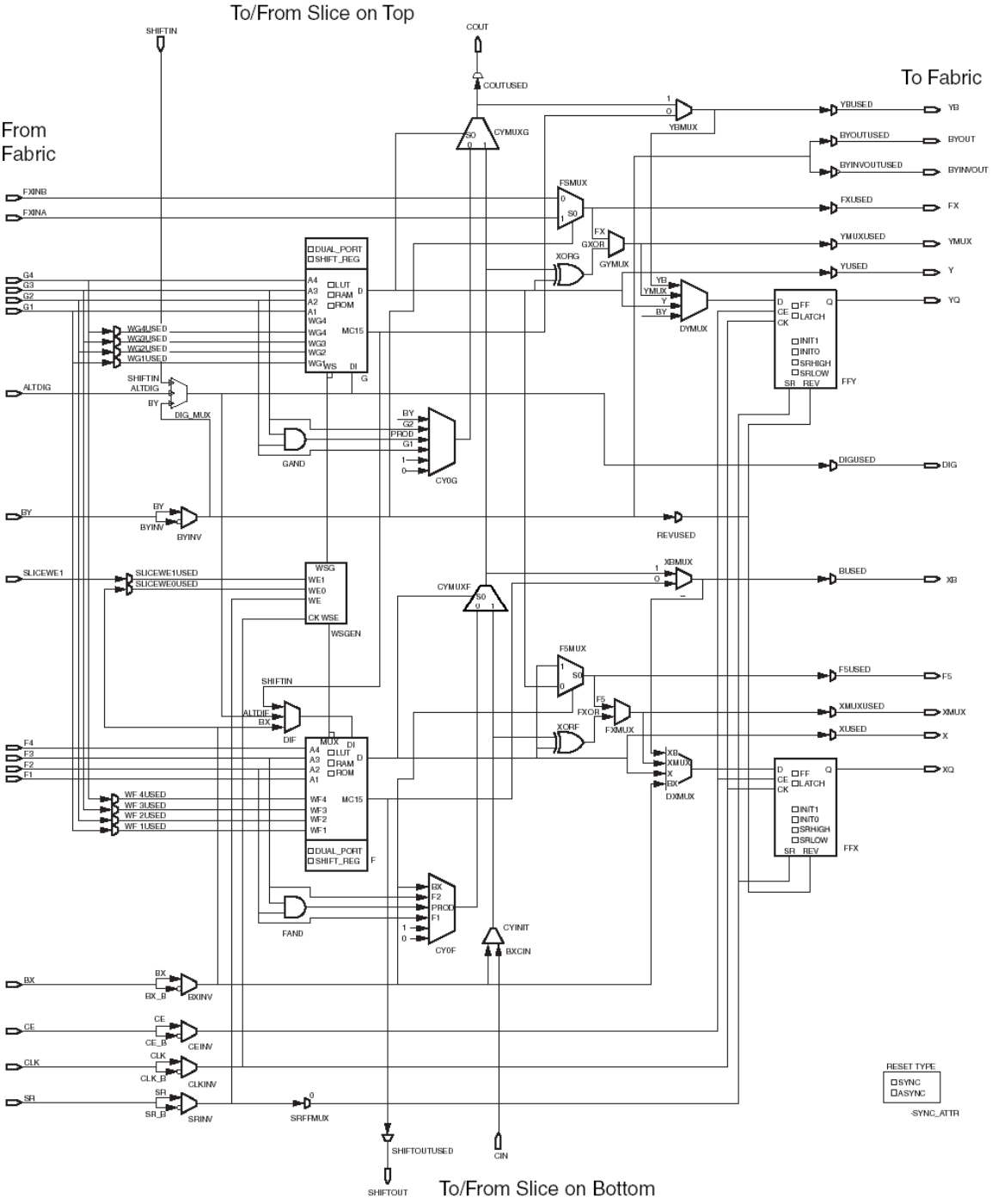
[29] Umberto Mazzei, Arlando Palamidessi, Paolo Passeri, Francesco Balena “Evolution of the Italian Telecommunication Network Towards SDH”, IEEE Communicatios Magazine, August 1990.

[30] Depari A., Ferrari P., Flammini A., Mariolo D., Taroni A. “Multi-probe Measurement Instrument for Real-Time Ethernet networks”, IEEE international Workshop on Factory Communication Systems, June 2006.

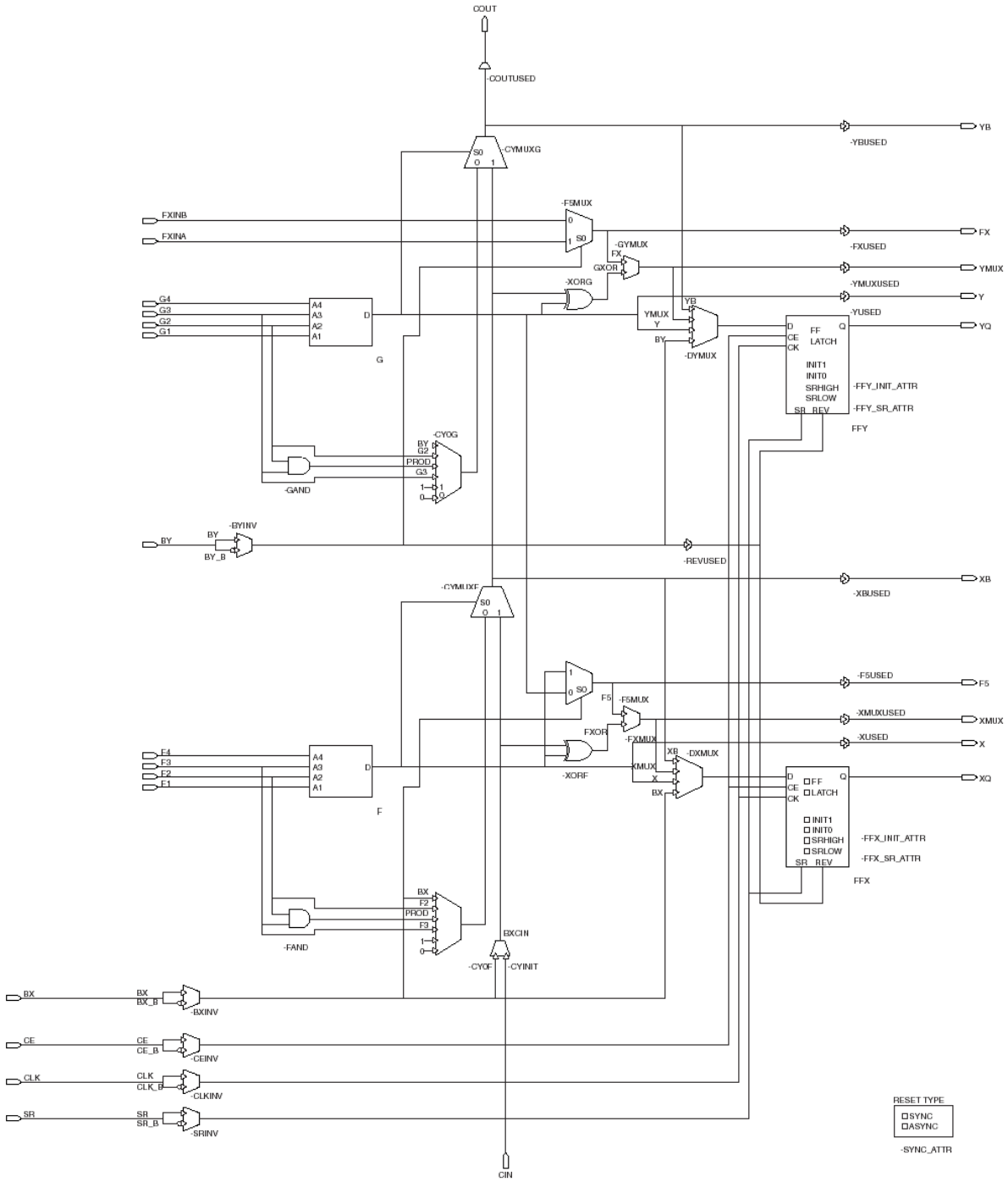
ANEXO A – Diagramas de SLICE

Os diagramas dos slices da XC2VPro são os seguintes:

A.1-SLICE M



A.2-SLICE L


COUT
-COUTUSED
YB
-YBUSED
-CYMUXG
-F5MUX
FXINB
FXINA
FX
-FXUSED
-GYMUX
GXOR
YMUX
-XORG
-YMUXUSED
Y
-YUSED
YQ
G4
G3
G2
G1
G
YMUX
Y
BY
-DYMUX
FF
LATCH
INIT1
INIT0
SRHIGH
SRLOW
SR REV
-FFY_INIT_ATTR
-FFY_SR_ATTR
FFY
-CYOG
PROD
-GAND
-BYINV
BY_B
-REVUSED
XB
-XBUSED
-CYMUXF
-F5USED
F5
-F5MUX
-XMUXUSED
XMUX
FXOR
-FXMUX
-XUSED
X
-XORF
XB
-DXMUX
XMUX
X
BX
XQ
F4
F3
F2
F1
F
-FFX_INIT_ATTR
-FFX_SR_ATTR
FFX
BXCIN
-FAND
-CYOF
-CYINIT
BX_B
-BXINV
CE_B
-CEINV
CLK_B
-CLKINV
SR_B
-SRINV
RESET TYPE
SYNC
ASYNC
-SYNC_ATTR
CIN

ANEXO B – código HDL para CRC paralelo

B.1- HEC: CRC16 paralelo RPR

```
0   ////////////////////////////////////////////////////////////////
1   //
2   //    CRC Width = 16
3   //    Data Width = 8
4   //    CRC Init   = 0
5   //    Polynomial = [0 -> 16]
6   //       1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1
7   //
8   ////////////////////////////////////////////////////////////////
9
10  module CRC_ccit_8 (
11    CRC_HEC,
12    d,
13    enable,
14    clk,
15    reset
16    );
17
18  output [15:0] CRC_HEC;
19
20  input  [7:0]  d;
21  input         enable;
22  input         clk;
23  input         reset;
24
25  reg [15:0] CRC_reg;
26
27  ////////////////////////////////////////////////////////////////
28  // Internal Signals
29  ////////////////////////////////////////////////////////////////
30  wire   [15:0] next_CRC;
31
32  always @ (posedge clk or posedge reset)
33  begin
34    if (reset)
35      begin
36        CRC_reg <= 16'hFFFF; // Inicializa a máquina de CRC
37      end
38    else
39      if (enable)
40        begin
41          CRC_reg <= next_CRC; // Atribui novo CRC
42        end
43      else
44        if (~enable)
45          begin
46            CRC_reg <= 16'hFFFF; // Inicializa a máquina de CRC
47          end
48  end
49
50  // OUTPUT
51  assign CRC_HEC = CRC_reg;
52
53  ////////////////////////////////////////////////////////////////
54  // CRC XOR equations
55  ////////////////////////////////////////////////////////////////
56
57  assign next_CRC[0] = d[7] ^ d[3] ^ CRC_reg[12] ^ CRC_reg[8];
58  assign next_CRC[1] = CRC_reg[13] ^ CRC_reg[9] ^ d[6] ^ d[2];
59  assign next_CRC[2] = d[5] ^ d[1] ^ CRC_reg[14] ^ CRC_reg[10];
60  assign next_CRC[3] = d[4] ^ d[0] ^ CRC_reg[15] ^ CRC_reg[11];
61  assign next_CRC[4] = d[3] ^ CRC_reg[12];
62  assign next_CRC[5] = CRC_reg[13] ^ d[2] ^ d[7] ^ d[3] ^ CRC_reg[12] ^ CRC_reg[8];
63  assign next_CRC[6] = CRC_reg[13] ^ CRC_reg[9] ^ d[1] ^ CRC_reg[14] ^ d[6] ^ d[2];
64  assign next_CRC[7] = d[0] ^ d[5] ^ d[1] ^ CRC_reg[14] ^ CRC_reg[10] ^ CRC_reg[15];
```

```
65  assign next_CRC[8] = d[4] ^ d[0] ^ CRC_reg[15] ^ CRC_reg[11] ^ CRC_reg[0];
66  assign next_CRC[9] = d[3] ^ CRC_reg[12] ^ CRC_reg[1];
67  assign next_CRC[10] = CRC_reg[13] ^ CRC_reg[2] ^ d[2];
68  assign next_CRC[11] = d[1] ^ CRC_reg[14] ^ CRC_reg[3];
69  assign next_CRC[12] = d[0] ^ CRC_reg[15] ^ d[7] ^ d[3] ^ CRC_reg[12] ^ CRC_reg[8] ^ CRC_reg[4];
70  assign next_CRC[13] = CRC_reg[13] ^ CRC_reg[9] ^ CRC_reg[5] ^ d[6] ^ d[2];
71  assign next_CRC[14] = d[5] ^ d[1] ^ CRC_reg[14] ^ CRC_reg[10] ^ CRC_reg[6];
72  assign next_CRC[15] = d[4] ^ d[0] ^ CRC_reg[15] ^ CRC_reg[11] ^ CRC_reg[7];
73  endmodule
74
```

B.2- FCS: CRC32 paralelo RPR

```
0   ////////////////////////////////////////////////////////////////
1   //
2   //    CRC Width = 32
3   //    Data Width = 8
4   //    CRC Init   = F
5   //    Polynomial = [0 -> 32]
6   //      1 1 1 0 1 1 0 1 1 0 1 1 1 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1
7   //
8   ////////////////////////////////////////////////////////////////
9
10  module CRC32_8 (
11    CRC_FCS,
12    d,
13    enable,
14    clk,
15    reset
16    );
17
18  output    [31:0] CRC_FCS;
19
20  input  [7:0]  d;
21  input         enable;
22  input         clk;
23  input         reset;
24
25  reg    [31:0] CRC_reg;
26
27  ////////////////////////////////////////////////////////////////
28  // Internal Signals
29  ////////////////////////////////////////////////////////////////
30  wire   [31:0] next_CRC;
31
32  always @ (posedge clk or posedge reset)
33  begin
34    if (reset)
35                begin
36      CRC_reg =  {16'hFFFF, 16'hFFFF}; // Inicializa a máquina de CRC
37     end
38    else
39              if (enable)
40      begin
41       CRC_reg <= next_CRC; // Atribui novo CRC
42      end
43     else
44      if (~enable)
45        begin
46         CRC_reg =  {16'hFFFF, 16'hFFFF}; // Inicializa a máquina de CRC
47        end
48  end
49
50
51
52  // OUTPUT
53  assign CRC_FCS = CRC_reg;
54
55  ////////////////////////////////////////////////////////////////
56  // CRC XOR equations
57  ////////////////////////////////////////////////////////////////
58
59  assign next_CRC[0] = CRC_reg[30] ^ d[1] ^ CRC_reg[24] ^ d[7];
60  assign next_CRC[1] = d[6] ^ d[7] ^ d[0] ^ CRC_reg[30] ^ CRC_reg[31] ^ d[1] ^ CRC_reg[24] ^ CRC_reg[25];
61  assign next_CRC[2] = CRC_reg[26] ^ d[5] ^ d[6] ^ d[7] ^ CRC_reg[30] ^ d[0] ^ d[1] ^ CRC_reg[31] ^
62  CRC_reg[24] ^ CRC_reg[25];
63  assign next_CRC[3] = d[4] ^ CRC_reg[26] ^ d[5] ^ CRC_reg[27] ^ d[6] ^ d[0] ^ CRC_reg[31] ^ CRC_reg[25];
64  assign next_CRC[4] = d[4] ^ CRC_reg[26] ^ d[5] ^ CRC_reg[27] ^ CRC_reg[28] ^ d[7] ^ CRC_reg[30] ^ d[1]
65  ^ CRC_reg[24] ^ d[3];
66  assign next_CRC[5] = d[4] ^ CRC_reg[27] ^ d[6] ^ CRC_reg[28] ^ d[7] ^ CRC_reg[29] ^ CRC_reg[30] ^ d[0]
67  ^ d[1] ^ CRC_reg[31] ^ d[2] ^ CRC_reg[24] ^ d[3] ^ CRC_reg[25];
```

```
68  assign next_CRC[6] = CRC_reg[26] ^ d[5] ^ d[6] ^ CRC_reg[28] ^ CRC_reg[29] ^ d[0] ^ CRC_reg[30] ^
69  CRC_reg[31] ^ d[1] ^ d[2] ^ d[3] ^ CRC_reg[25];
70  assign next_CRC[7] = d[4] ^ CRC_reg[26] ^ d[5] ^ CRC_reg[27] ^ d[7] ^ CRC_reg[29] ^ d[0] ^ CRC_reg[31]
71  ^ d[2] ^ CRC_reg[24];
72  assign next_CRC[8] = d[4] ^ CRC_reg[27] ^ d[6] ^ CRC_reg[28] ^ d[7] ^ CRC_reg[24] ^ CRC_reg[0] ^ d[3] ^
73  CRC_reg[25];
74  assign next_CRC[9] = CRC_reg[26] ^ d[5] ^ d[6] ^ CRC_reg[28] ^ CRC_reg[29] ^ d[2] ^ d[3] ^ CRC_reg[25]
75  ^ CRC_reg[1];
76  assign next_CRC[10] = d[4] ^ CRC_reg[26] ^ CRC_reg[2] ^ d[5] ^ CRC_reg[27] ^ d[7] ^ CRC_reg[29] ^ d[2]
77  ^ CRC_reg[24];
78  assign next_CRC[11] = d[4] ^ CRC_reg[27] ^ d[6] ^ CRC_reg[3] ^ CRC_reg[28] ^ d[7] ^ CRC_reg[24] ^ d[3]
79  ^ CRC_reg[25];
80  assign next_CRC[12] = CRC_reg[26] ^ d[5] ^ d[6] ^ CRC_reg[28] ^ d[7] ^ CRC_reg[4] ^ CRC_reg[29] ^
81  CRC_reg[30] ^ d[1] ^ d[2] ^ CRC_reg[24] ^ d[3] ^ CRC_reg[25];
82  assign next_CRC[13] = d[4] ^ CRC_reg[26] ^ d[5] ^ CRC_reg[27] ^ d[6] ^ CRC_reg[29] ^ d[0] ^ CRC_reg[30]
83  ^ CRC_reg[5] ^ CRC_reg[31] ^ d[1] ^ d[2] ^ CRC_reg[25];
84  assign next_CRC[14] = d[4] ^ CRC_reg[26] ^ d[5] ^ CRC_reg[27] ^ CRC_reg[28] ^ CRC_reg[30] ^ d[0] ^ d[1]
85  ^ CRC_reg[31] ^ CRC_reg[6] ^ d[3];
86  assign next_CRC[15] = d[4] ^ CRC_reg[27] ^ CRC_reg[28] ^ CRC_reg[29] ^ d[0] ^ CRC_reg[31] ^ d[2] ^
87  CRC_reg[7] ^ d[3];
88  assign next_CRC[16] = CRC_reg[28] ^ d[7] ^ CRC_reg[29] ^ d[2] ^ CRC_reg[24] ^ d[3] ^ CRC_reg[8];
89  assign next_CRC[17] = CRC_reg[9] ^ d[6] ^ CRC_reg[29] ^ CRC_reg[30] ^ d[1] ^ d[2] ^ CRC_reg[25];
90  assign next_CRC[18] = CRC_reg[26] ^ d[5] ^ CRC_reg[10] ^ CRC_reg[30] ^ d[0] ^ d[1] ^ CRC_reg[31];
91  assign next_CRC[19] = d[4] ^ CRC_reg[27] ^ CRC_reg[11] ^ d[0] ^ CRC_reg[31];
92  assign next_CRC[20] = CRC_reg[28] ^ CRC_reg[12] ^ d[3];
93  assign next_CRC[21] = CRC_reg[29] ^ CRC_reg[13] ^ d[2];
94  assign next_CRC[22] = d[7] ^ CRC_reg[14] ^ CRC_reg[24];
95  assign next_CRC[23] = d[6] ^ d[7] ^ CRC_reg[30] ^ d[1] ^ CRC_reg[15] ^ CRC_reg[24] ^ CRC_reg[25];
96  assign next_CRC[24] = CRC_reg[26] ^ d[5] ^ d[6] ^ d[0] ^ CRC_reg[31] ^ CRC_reg[16] ^ CRC_reg[25];
97  assign next_CRC[25] = d[4] ^ CRC_reg[17] ^ CRC_reg[26] ^ d[5] ^ CRC_reg[27];
98  assign next_CRC[26] = d[4] ^ CRC_reg[18] ^ CRC_reg[27] ^ CRC_reg[28] ^ d[7] ^ CRC_reg[30] ^ d[1] ^
99  CRC_reg[24] ^ d[3];
100 assign next_CRC[27] = d[6] ^ CRC_reg[19] ^ CRC_reg[28] ^ CRC_reg[29] ^ d[0] ^ CRC_reg[31] ^ d[2] ^ d[3]
101 ^ CRC_reg[25];
102 assign next_CRC[28] = CRC_reg[26] ^ d[5] ^ CRC_reg[20] ^ CRC_reg[29] ^ CRC_reg[30] ^ d[1] ^ d[2];
103 assign next_CRC[29] = d[4] ^ CRC_reg[27] ^ CRC_reg[21] ^ CRC_reg[30] ^ d[0] ^ d[1] ^ CRC_reg[31];
104 assign next_CRC[30] = CRC_reg[28] ^ d[0] ^ CRC_reg[22] ^ CRC_reg[31] ^ d[3];
105 assign next_CRC[31] = CRC_reg[29] ^ CRC_reg[23] ^ d[2];
106 endmodule
107
```

ANEXO C – código HDL para para scrambler/descrambler

C.1- Scrambler 1+x^43 paralelo

```
0   ////////////////////////////////////////////////////////////////
1   //
2   // Scrambler HDLC byte =>  1+X^43
3   //
4   ////////////////////////////////////////////////////////////////
5
6   LIBRARY ieee;
7   USE ieee.std_logic_1164.all;
8   USE ieee.std_logic_arith.all;
9
10  ENTITY scrambler IS
11    PORT(
12      DIn    : IN    std_logic_vector (7 DOWNTO 0);
13      RESET  : IN    std_logic;
14      Clk    : IN    std_logic;
15      En     : IN    std_logic;
16      DOut   : OUT   std_logic_vector (7 DOWNTO 0)
17    );
18  END scrambler ;
19
20  Architecture scrambler Of scrambler Is
21
22     signal D: std_logic_vector(7 downto 0);
23     signal C: std_logic_vector(42 downto 0);
24     signal NxCRC, CRC: std_logic_vector(42 downto 0);
25
26   begin
27              ----------------------------------------------------------------------
28              Sequencial: Process (RESET, Clk)
29              Begin
30              ----------------------------------------------------------------------
31              If RESET='1' Then
32                      CRC<="0000000000000000000000000000000000000000000";  -- Inicializa
33              Else
34                      If (Clk'Event And Clk='1') Then
35                              CRC<=NxCRC;
36                      End If;
37              End If;
38              ----------------------------------------------------------------------
39              End Process Sequencial;
40
41              Comb: Process (CRC, En)
42              Begin
43
44    C <= CRC;
45    D <= DIn;
46
47    if (En = '1') then
48
49      NxCRC(0) <= D(0) xor C(35);
50      NxCRC(1) <= D(1) xor C(36);
51      NxCRC(2) <= D(2) xor C(37);
52      NxCRC(3) <= D(3) xor C(38);
53      NxCRC(4) <= D(4) xor C(39);
54      NxCRC(5) <= D(5) xor C(40);
55      NxCRC(6) <= D(6) xor C(41);
56      NxCRC(7) <= D(7) xor C(42);
57      NxCRC(8) <= C(0);
58      NxCRC(9) <= C(1);
59      NxCRC(10) <= C(2);
60      NxCRC(11) <= C(3);
61      NxCRC(12) <= C(4);
62      NxCRC(13) <= C(5);
63      NxCRC(14) <= C(6);
64      NxCRC(15) <= C(7);
```

```
65      NxCRC(16) <= C(8);
66      NxCRC(17) <= C(9);
67      NxCRC(18) <= C(10);
68      NxCRC(19) <= C(11);
69      NxCRC(20) <= C(12);
70      NxCRC(21) <= C(13);
71      NxCRC(22) <= C(14);
72      NxCRC(23) <= C(15);
73      NxCRC(24) <= C(16);
74      NxCRC(25) <= C(17);
75      NxCRC(26) <= C(18);
76      NxCRC(27) <= C(19);
77      NxCRC(28) <= C(20);
78      NxCRC(29) <= C(21);
79      NxCRC(30) <= C(22);
80      NxCRC(31) <= C(23);
81      NxCRC(32) <= C(24);
82      NxCRC(33) <= C(25);
83      NxCRC(34) <= C(26);
84      NxCRC(35) <= C(27);
85      NxCRC(36) <= C(28);
86      NxCRC(37) <= C(29);
87      NxCRC(38) <= C(30);
88      NxCRC(39) <= C(31);
89      NxCRC(40) <= C(32);
90      NxCRC(41) <= C(33);
91      NxCRC(42) <= C(34);
92
93      DOut <= NxCRC(7 downto 0);
94    else
95      NxCRC<=CRC;
96      DOut <=DIn;
97    end if;
98
99   End Process COMB;
100
101 end scrambler;
```

C.2- descrambler 1+x^43 paralelo

```
0   ////////////////////////////////////////////////////////////
1   //
2   // Descrambler HDLC byte =>  1+X^43
3   //
4   ////////////////////////////////////////////////////////////
5   LIBRARY ieee;
6   USE ieee.std_logic_1164.all;
7   USE ieee.std_logic_arith.all;
8
9   ENTITY descrambler IS
10    PORT(
11      Data   : IN     std_logic_vector (7 DOWNTO 0);
12      RESET  : IN     std_logic;
13      Clk    : IN     std_logic;
14      ENABLE : IN     std_logic;
15      Dout   : OUT    std_logic_vector (7 DOWNTO 0)
16    );
17  END descrambler ;
18
19  Architecture descrambler Of descrambler Is
20
21     signal D: std_logic_vector(7 downto 0);
22     signal C,NxC: std_logic_vector(42 downto 0);
23     signal NxCRC, CRC: std_logic_vector(42 downto 0);
24
25   begin
26              Sequencial: Process (RESET, Clk)
27              Begin
28              ------------------------------------------------------------------------
29              If RESET='1' Then
30                      CRC<="00000000000000000000000000000000000000000000000";  -- Inicializa
31                      C<="00000000000000000000000000000000000000000000000";
32              Else
33                      If (Clk'Event And Clk='1') Then
34                              CRC<=NxCRC;
35                              C<=NxC;
36                      End If;
37              End If;
38              ------------------------------------------------------------------------
39              End Process Sequencial;
40
41              Comb: Process (CRC)
42              Begin
43  D <= Data;
44
45  if (ENABLE = '1') then
46
47  NxC(0) <= D(0);
48  NxC(1) <= D(1);
49  NxC(2) <= D(2);
50  NxC(3) <= D(3);
51  NxC(4) <= D(4);
52  NxC(5) <= D(5);
53  NxC(6) <= D(6);
54  NxC(7) <= D(7);
55  NxC(8) <= C(0);
56  NxC(9) <= C(1);
57  NxC(10) <= C(2);
58  NxC(11) <= C(3);
59  NxC(12) <= C(4);
60  NxC(13) <= C(5);
61  NxC(14) <= C(6);
62  NxC(15) <= C(7);
63  NxC(16) <= C(8);
64  NxC(17) <= C(9);
65  NxC(18) <= C(10);
66  NxC(19) <= C(11);
```

```
67   NxC(20) <= C(12);
68   NxC(21) <= C(13);
69   NxC(22) <= C(14);
70   NxC(23) <= C(15);
71   NxC(24) <= C(16);
72   NxC(25) <= C(17);
73   NxC(26) <= C(18);
74   NxC(27) <= C(19);
75   NxC(28) <= C(20);
76   NxC(29) <= C(21);
77   NxC(30) <= C(22);
78   NxC(31) <= C(23);
79   NxC(32) <= C(24);
80   NxC(33) <= C(25);
81   NxC(34) <= C(26);
82   NxC(35) <= C(27);
83   NxC(36) <= C(28);
84   NxC(37) <= C(29);
85   NxC(38) <= C(30);
86   NxC(39) <= C(31);
87   NxC(40) <= C(32);
88   NxC(41) <= C(33);
89   NxC(42) <= C(34);
90   NxCRC(0) <= d(0) xor C(35);
91   NxCRC(1) <= d(1) xor C(36);
92   NxCRC(2) <= d(2) xor C(37);
93   NxCRC(3) <= d(3) xor C(38);
94   NxCRC(4) <= d(4) xor C(39);
95   NxCRC(5) <= d(5) xor C(40);
96   NxCRC(6) <= d(6) xor C(41);
97   NxCRC(7) <= d(7) xor C(42);
98   NxCRC(8) <= C(0);
99   NxCRC(9) <= C(1);
100  NxCRC(10) <= C(2);
101  NxCRC(11) <= C(3);
102  NxCRC(12) <= C(4);
103  NxCRC(13) <= C(5);
104  NxCRC(14) <= C(6);
105  NxCRC(15) <= C(7);
106  NxCRC(16) <= C(8);
107  NxCRC(17) <= C(9);
108  NxCRC(18) <= C(10);
109  NxCRC(19) <= C(11);
110  NxCRC(20) <= C(12);
111  NxCRC(21) <= C(13);
112  NxCRC(22) <= C(14);
113  NxCRC(23) <= C(15);
114  NxCRC(24) <= C(16);
115  NxCRC(25) <= C(17);
116  NxCRC(26) <= C(18);
117  NxCRC(27) <= C(19);
118  NxCRC(28) <= C(20);
119  NxCRC(29) <= C(21);
120  NxCRC(30) <= C(22);
121  NxCRC(31) <= C(23);
122  NxCRC(32) <= C(24);
123  NxCRC(33) <= C(25);
124  NxCRC(34) <= C(26);
125  NxCRC(35) <= C(27);
126  NxCRC(36) <= C(28);
127  NxCRC(37) <= C(29);
128  NxCRC(38) <= C(30);
129  NxCRC(39) <= C(31);
130  NxCRC(40) <= C(32);
131  NxCRC(41) <= C(33);
132  NxCRC(42) <= C(34);
133
134  Dout <= NxCRC (7 DOWNTO 0);
135
136  else
137    NxCRC<=CRC;
138    NxC<=C;
```

```
139     Dout <= Data;
140   end if;
141
142             End Process COMB;
143   end descrambler;
```