Pontifícia Universidade Católica do Rio Grande do Sul

Faculdade de Engenharia – Faculdade de Informática

Curso de Engenharia de Computação

Desenvolvimento de Módulos de Hardware para Recepção e Transmissão de Quadros OTN

Volume Final do Trabalho de Conclusão

Autores
Roberto Germani Paiva
Samuel dos Santos Marczak

Orientador
Prof. Dr. Fernando Gehm Moraes

Porto Alegre, dezembro de 2007.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Siglas

3R	Reamplification, Reshaping and Retiming
10GbE	10-Gigabit Ethernet
ATM	Asynchronous Transfer Mode
BIP	Bit Interleaved Parity
BRAM	Block Ram
CBR	Constant Bit Rate
DWDM	Dense Wavelength Division Multiplexing
EDA	Electronic Design Automation
FAS	Frame Alignment Signal
FPGA	Field Programmable Gate Array
FEC	Forward Error Correction
FTFL	Fault Type & Fault Location
GbE	Gigabit Ethernet
GFP	Generic Framing Procedure
IaDI	Intra-Domain Interface
IP	Internet Protocol
IrDI	Inter-Domain Interface
ISE	Integrated Software Environment
ITUT	ITU Telecommunication Standardization Sector
JC	Justification Control
LAN	Local Area Network
LAN PHY	LAN Physical Layer
LFSR	Linear Feedback Shift Register
LUT	Lookup Table
LVDS	Low-Voltage Differential Signaling
MAN	Metropolitan Area Network
MFAS	Multi Frame Alignment Signal
MHz	Megahertz
NJO	Negative Justification Opportunity
ODU	Optical Channel Data Unit
ODU OH	Optical Channel Data Unit Overhead
OPU	Optical Channel Payload Unit
OPU OH	Optical Channel Payload Unit Overhead
OTN	Optical Transport Network
OTU	Optical Channel Transport Unit
OTU OH	Optical Channel Transport Unit Overhead
PJO	Positive Justification Opportunity
PSI	Payload Structure Identifier
PT	Payload Type
SDF	Standard Delay Format
SDH	Synchronous Digital Hierarchy
SerDes	Serializer-Deserializer
SM	Section Monitoring
SONET	Synchronous Optical Network
STA	Static Timing Analysis
TCM	Tandem Connection Monitoring
TDM	Time-Division Multiplexing
TTI	Trail Trace Identifier
VHDL	VHSIC Hardware Description Language

1 Introdução

Com o aumento da globalização, a troca de informações tornou-se uma ferramenta fundamental no andamento da economia, impulsionando a formação de redes metropolitanas rápidas, flexíveis e confiáveis. Com a demanda sempre crescente por novos serviços, a capacidade de transferir informações diversificadas e de diferentes níveis de complexidade em uma mesma infra-estrutura tornou-se o foco das operadoras de telecomunicações que prevêem uma acentuada demanda por serviços que exigem alta largura de banda, tais como: videoconferência, educação à distância, telemedicina, voz sobre IP (VoIP), entre outros.

Com a migração de tecnologias de rede para protocolos de maior velocidade (Gigabit Ethernet e 10 Gigabit Ethernet), passou-se a utilizar fibra óptica para transmissão de informações em redes locais (LANs) e metropolitanas (MANs). A fibra óptica não sofre interferências eletromagnéticas, o que significa que os dados que estão sendo transferidos correm menos riscos de serem corrompidos durante a transmissão. Outra vantagem é que a perda de potência do sinal por quilômetro é muito menor do que em sistemas que utilizam cabos coaxiais.

Nas transmissões por fibras ópticas, as portadoras possuem freqüências na faixa de infravermelho, valores da ordem de centenas de TeraHertz, fato que permite prever o emprego de elevadíssimas taxas de transmissão, de até milhares de Gigabits/s. A capacidade do sistema óptico pode ser aumentada ainda mais, utilizando-se técnicas de multiplexação por comprimento de onda.

Nos canais de fibra óptica é utilizada a técnica de multiplexação por comprimento de onda, WDM (*Wavelength Division Multiplexing*). Para um número elevado de canais multiplexados, a técnica WDM é chamada de DWDM (*Dense Wavelength Division Multiplexing*). Este é um sistema que multiplexa múltiplos comprimentos de onda que serão transmitidos através de uma única fibra óptica, independente das diferentes taxas de bits e formatos. Devido à capacidade de transportar diversas tecnologias e possibilitar o maior transporte de tráfego sobre a fibra, o DWDM torna-se uma peça importante na integração das redes de dados, voz e imagem de altíssima capacidade [NET02]. Este tipo de sistema tem como vantagem a alta flexibilidade em expansão de largura de banda, uma grande escalabilidade e a redução de funções custosas de multiplexação e demultiplexação elétrica.

Para atender a crescente demanda por banda com transporte de Tbps por fibra em enlaces DWDM e dar suporte a serviços de banda larga com taxas de 2,5 Gbps, 10 Gbps e 40 Gbps, uma nova camada de rede de transporte óptico foi implantada, a Optical Transport Network (OTN). A OTN é definida pela ITU-T [ITU03] como um composto de elementos de redes ópticas (Optical Network Elements) conectados por uma fibra óptica, capaz de fornecer funcionalidades de transporte, multiplexação, roteamento, gerência e supervisionamento dos canais ópticos transportando os sinais provenientes do cliente. A rede OTN não impõe limites nas taxas de comutação, tornando-a muito flexível no quesito largura de banda. Quando a taxa dos enlaces de

dados aumenta, novas taxas de comutação podem ser adicionadas.

A necessidade de suprir a crescente demanda por canais de comunicação de banda larga faz com que empresas desenvolvedoras de equipamentos de telecomunicação gerem soluções que suportem cada vez mais banda com um menor custo. Os enlaces de fibra ótica devem dar suporte a taxas de 10 Gbit/s ou superiores, e novas técnicas são necessárias para suprir esta expectativa e seu crescimento futuro.

A motivação do presente trabalho surgiu a partir do projeto realizado em parceria entre a TERACOM, empresa desenvolvedora de equipamentos de telecomunicação, e o Grupo de Apoio ao Projeto de Hardware (GAPH), da Faculdade de Informática (FACIN) da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). O projeto X10GIGA visa desenvolver um *transponder* capaz de transmitir sinais SDH e Gigabit Ethernet sobre redes do tipo OTN com enlaces ópticos em longas distâncias. O desenvolvimento será em protótipos de hardware que permitirão realizar a depuração e a validação de todos os módulos implementados.

1.1 Objetivos

O presente trabalho tem por objetivo desenvolver módulos utilizados no desenvolvimento do *transponder.* Estes devem ser capazes de transmitir e receber sinais sobre redes do tipo OTN, a uma taxa de comunicação de 10.7 Gbps. Ao final do trabalho de conclusão espera-se que os módulos desenvolvidos sejam capazes de:

- Alinhar as palavras de entrada, de modo que contenham a seqüência de alinhamento começando no *bit* zero. Quando identificado o FAS, um sinal de *fullmatch* deve ser gerado para alimentar a máquina de estados de alinhamento;
- Receber quadros OTN embaralhados e realizar o desembaralhamento e vice-versa;
- Extrair os cabeçalhos dos quadros OTN e armazená-los em memórias.

1.2 Módulos do Projeto Transponder

A arquitetura do projeto do *transponder* OTN é apresentada na Figura 1. Os módulos desenvolvidos no presente trabalho estão destacados com uma linha mais espessa, sendo eles: Alinhador de Quadro, Embaralhador, Desembaralhador e Extrator de Cabeçalhos. Os módulos Ajuste dos cabeçalhos e Inserção/Remoção de dados serão desenvolvidos futuramente, devido às limitações de tempo intrínsecas a um Trabalho de Conclusão de curso.

Apesar de fazerem parte do projeto, os módulos de decodificação e inserção de FEC são de responsabilidade de outro grupo de TC.

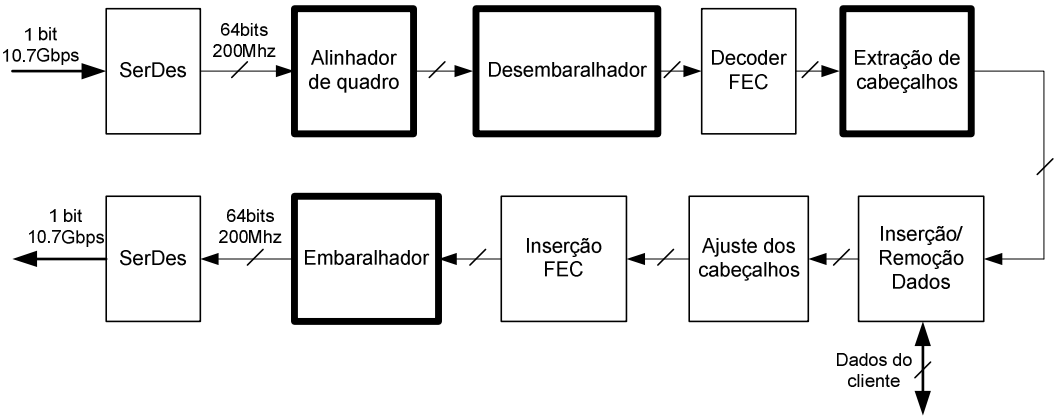

Figura 1 – Estrutura geral do transponder OTN a ser desenvolvida.

O módulo alinhador de quadro tem como restrição de projeto trabalhar com um fluxo de dados de largura 64 *bits*, operando a uma freqüência de 167,33 MHz. Essa restrição ocorre devido ao FPGA receber fluxos de dados de um módulo de interface com fibra óptica, externo ao contexto deste trabalho. Este, é responsável por paralelizar o fluxo de dados a 10,688 Gbps em 16 canais diferenciais (LVDS), cada um a uma taxa de 669,3125 Mbps. No decorrer do trabalho é adotado como freqüência interna de operação 200 MHz, o que resulta em uma taxa de 12,8 Gbps. Esta maior taxa de operação é adotada como cenário de pior caso, permitindo o desenvolvimento de um projeto mais robusto.

O módulo alinhador de quadro é composto por sub-módulos que consistem em identificar a ocorrência da seqüência de alinhamento, identificar em qual *bit* a seqüência está começando, realizar o deslocamento de modo que a seqüência de alinhamento comece no início da palavra de 64 *bits* e alimentar uma máquina de estados que realiza o sincronismo. Para ser atendida a restrição de tempo de propagação combinacional do projeto de 5 ns, cada sub-módulo é separado por um registrador, formando assim um *pipeline*.

O módulo embaralhador consiste em embaralhar os dados a serem transmitidos utilizando a técnica de geração de dados pseudo-aleatórios, conhecida como um LFSR (*Linear Feedback Shift Register*). Com isto, tem-se uma proteção para que não ocorram longas seqüências de 0s ou de 1s, o que garante suficiente troca de estado dos *bits* para facilitar a regeneração do relógio e também evita uma possível repetição da seqüência de alinhamento. A operação de embaralhar é realizada em todo o multi-quadro OTN G.709, com exceção do campo FAS. A funcionalidade de desembaralhar é a mesma de embaralhar, visto que a operação de embaralhamento é reversa. Basta passar um dado previamente embaralhado pelo módulo embaralhador que ele sairá desembaralhado.

Serão abordadas duas arquiteturas diferentes para implementação dos módulos embaralhador e desembaralhador. Primeiramente, para realizar o desenvolvimento do módulo utilizou-se a arquitetura padrão referida em [ITU03], onde a geração da seqüência de embaralhamento é feita em hardware por LFSRs. Tendo em vista um ganho no desempenho do módulo, foi utilizada uma abordagem mais simples, em que todas as seqüências do polinômio embaralhador foram inicializadas em memórias BRAM (*Block RAM*).

O módulo de extração dos cabeçalhos é utilizado para extrair o conteúdo dos cabeçalhos para uma memória, possibilitando a identificação de falhas, alarmes, tipo de dado transportado, origem e destino contidos nos quadros OTN G.709, o que será feito por um software embarcado externo ao contexto deste trabalho.

1.3 Estrutura do Documento

Este trabalho é dividido em nove capítulos. Além da presente introdução, temos dois capítulos teóricos: um sobre o padrão OTN G.709 e o outro sobre fluxo de projeto de sistemas digitais, quatro capítulos descrevem o projeto desenvolvido e os dois últimos capítulos contemplam a conclusão e as referências bibliográficas utilizadas.

Os Capítulos 2 e 3 incluem conceitos básicos necessários à compreensão deste documento. O Capítulo 2 apresenta conceitos básicos sobre o padrão OTN G.709. No Capítulo 3 são apresentadas as etapas e ferramentas do fluxo de projeto utilizado para desenvolvimento do presente trabalho.

Os Capítulos 4, 5 e 6 descrevem os módulos desenvolvidos. No Capítulo 4 é descrito o módulo Alinhador de Quadro e seus sub-módulos, assim como sua arquitetura. No capítulo 5 é apresentado o módulo Embaralhador e Desembaralhador bem como seu resultado em nível de validação e área. O módulo Extrator de Cabeçalhos é apresentado no Capítulo 6.

O Capítulo 7 apresenta a integração dos módulos descritos nos Capítulos anteriores, bem como os resultados da validação em nível de simulação e prototipação dos mesmos. Este ainda apresenta a estrutura de teste aplicada para validar os módulos e a integração com o módulo FEC.

As conclusões obtidas pelo grupo ao fim deste trabalho e as referências auxiliares para o desenvolvimento do mesmo são apresentadas nos Capítulos 8 e 9.

2 Padrão OTN G.709

A ITU-T define **Optical Transport Network (OTN)** [ITU03] [WAL03] como um composto de elementos de redes ópticas (*Optical Network Elements*) conectados por uma fibra óptica, capaz de fornecer funcionalidades de transporte, multiplexação, roteamento, gerência e supervisionamento dos canais ópticos transportando os sinais provenientes do cliente.

Os dados a serem transportados por redes OTN podem ter diversas origens. Cada dispositivo cliente se relaciona com a rede OTN através de seu protocolo nativo (Figura 2, índice 1). Um dispositivo OTN encapsula informações de múltiplos clientes (Figura 2, índice 2) em uma carga útil OTU1, OTU2 ou OTU3 e adiciona um cabeçalho de controle. Administradores podem monitorar e obter informações variadas em vários pontos da rede, através de interfaces de gerenciamento de rede (Figura 2, índice 3).

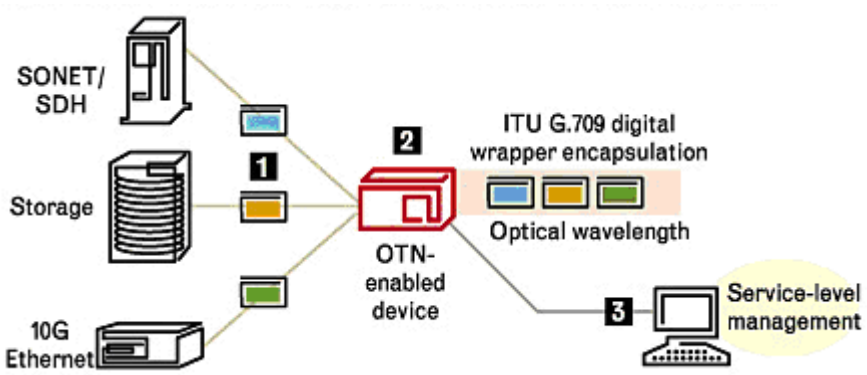

Figura 2 – Dispositivo OTN ligado a dispositivos clientes [ALE06].

O OTN G.709 encapsula os dados da carga útil e o cabeçalho provenientes do cliente em sua própria carga útil. Este comportamento garante a preservação das características originais dos dados enquanto eles estão sendo transportados por uma rede óptica.

2.1 A Abrangência do Padrão G.709

A arquitetura OTN [ITU05] define duas classes de interfaces: IrDI (*Inter-domain interface)* e IaDI (*Intra-domain interface)*, apresentadas na Figura 3. A interface OTN IrDI é definida com o processamento 3R (*Reamplification, Reshaping e Retiming*) em cada terminação da interface. Esta é a interface entre operadores e usuários. A interface OTN IaDI é definida como a interface dentro do domínio de um operador e um usuário.

O padrão de protocolos de interface G.709 é aplicado nas informações transferidas pelas interfaces IrDI e IaDI. É importante notar que o G.709 define somente a interface lógica, sem especificar interfaces ópticas ou elétricas.

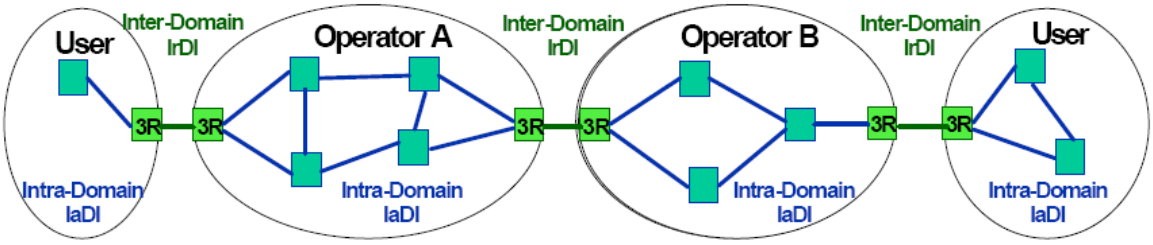

Figura 3 – Classes de interface do padrão G.709 [KAZ06].

2.2 Estrutura Lógica do Padrão OTN G.709

O quadro OTN G.709 inclui um cabeçalho de transporte que provê funcionalidades de administração, manutenção e o FEC (*Forward Error Correction*). O FEC ajuda a reduzir o número de erros na transmissão em enlaces com ruído, o que por sua vez possibilita o uso de cabos ópticos de maior comprimento.

A norma G.709 define um padrão para as interfaces e suas taxas de transmissão. Estas taxas foram derivadas das taxas já existentes dos padrões SONET/SDH [ITU00] onde o cabeçalho e a informação do FEC do G.709 foram levados em consideração. Assim, a interface resultante opera a taxas de transmissão aproximadamente 7% maiores que as taxas correspondentes dos padrões SONET/SDH.

A Tabela 1 lista as interfaces OTU1, OTU2 e OTU3 e suas taxas de transmissão definidas na G.709, comparando essas com as taxas correspondentes nos padrões SONET/SDH.

Tabela 1 – Taxas de comunicação utilizadas nas interfaces dos padrões G.709 e SONET/SDH.

Interfaces do padrão G.709	Taxas de Transmissão	Interfaces dos padrões SONET/SDH	Taxas de Transmissão
OTU-1	2.666 Gbps	OC-48/STM-16	2.488 Gbps
OTU-2	10.709 Gbps	OC-192/STM-64	9.953 Gbps
OTU-3	43.018 Gbps	OC-768/STM-256	39.813 Gbps

Ao contrário dos padrões SDH/SONET, o OTN pode ser modificado para atuar com taxas de transmissão especiais, podendo então, por exemplo, transportar pacotes do tipo 10G LAN PHY provenientes de chaveadores e roteadores IP/Ethernet com taxa de transmissão máxima. Isto é importante porque pacotes LAN PHY normalmente incluem cabeçalhos proprietários que aumentam a taxa de transmissão além de 10 Gbps.

O quadro OTN G.709 (Figura 4) é formado por 4080 *bytes,* constituídos de 16 subquadros. Cada subquadro é 1 linha de 255 *bytes*. Os 4080 *bytes* do quadro OTN estão divididos em: cabeçalho, que contém 16 *bytes*, carga útil, que contém 3808 *bytes* e o FEC, que contém os 256

bytes restantes. A ordem de transmissão é representada pelos números dentro de cada *byte* na Figura 4.

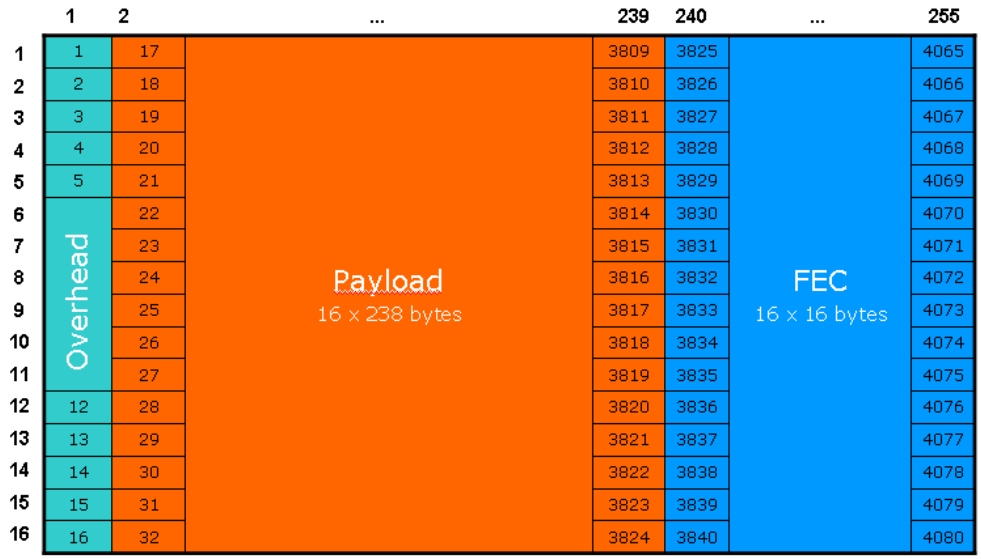

Figura 4 – Quadro OTN com 4080 bytes, mostrando a ordem de transmissão dos bytes.

O padrão G.709 [ITU02] define o multi-quadro OTN (Figura 5) como sendo quatro quadros OTN (Figura 4) agrupados em 4 linhas, cada uma com 4080 colunas. Cada quadro é representado como sendo uma linha na estrutura multi-quadro. Esta formação é denominada *digital wrapper multi-frame*. O multi-quadro é composto por 16320 *bytes*, sendo: 64 *bytes* de cabeçalho, 15232 *bytes* de carga útil e 1024 *bytes*, de FEC.

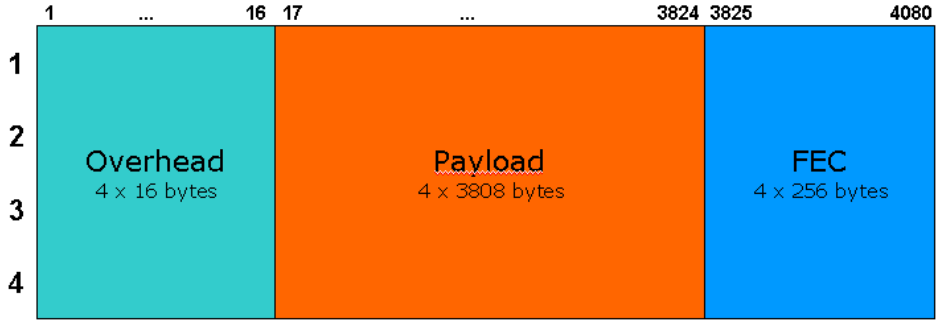

Figura 5 – Multi-quadro OTN: composto por quatro quadros OTN.

Ao contrário do padrão SDH/SONET, a taxa de transmissão é elevada mantendo-se a estrutura multi-quadro do G.709 (4 linhas x 4080 colunas). Isto é possível diminuindo o período em que o quadro é transmitido (no padrão SDH/SONET a estrutura do quadro é aumentada e o período do quadro de 125 μs é mantido). No caso da G.709, tem-se para a interface OTU1/ODU1/OPU1 o período definido em 48,971 μs, para OTU2/ODU2/OPU2 é definido em 12,191 μs e para OTU3/ODU3/OPU3 é definido em 3,035 μs.

O multi-quadro G.709 OTN é constituído por três camadas denominadas: OPU (*Optical Channel Payload Unit*), que é a camada utilizada para adaptar a informação que vem do cliente para

o transporte em um canal óptico; ODU (*Optical Channel Data Unit*), que é a camada utilizada para controlar o caminho percorrido pelos quadros e OTU (*Optical Channel Transport Unit*), que é a camada de transporte. Estas camadas estão representadas na Figura 6.

Estas camadas são encapsuladas como é mostrado na Figura 7. O OTU, por ser a camada de transporte, é a mais externa, ela encapsula o OPU e o ODU. O ODU é a camada de controle e encapsula o OPU, que é a camada inferior composta pela carga útil. Pode-se dividir o quadro G.709 OTN em três distintas áreas denominadas cabeçalho, carga útil e FEC que serão discutidas a seguir.

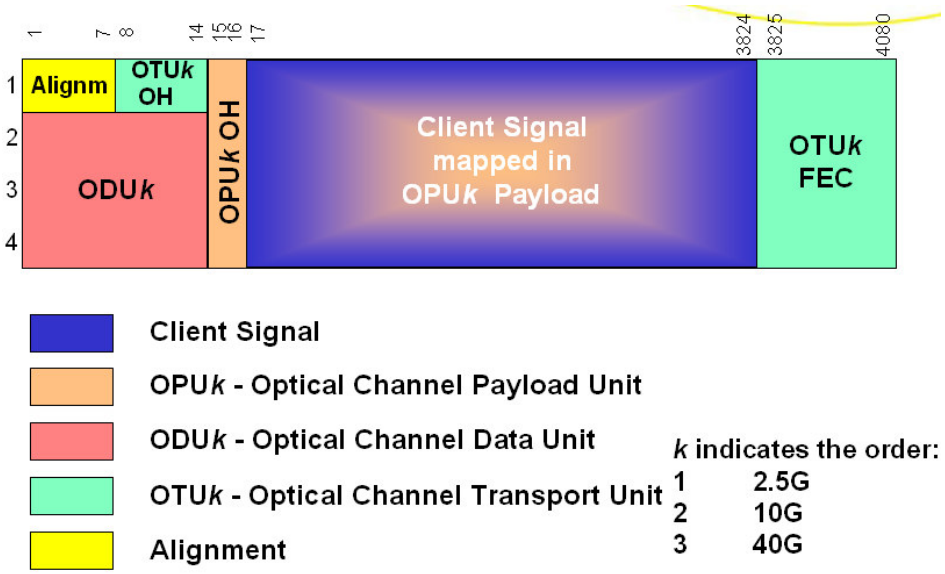

Figura 6 – Multi-quadro OTN dividido em camadas [VIS02].

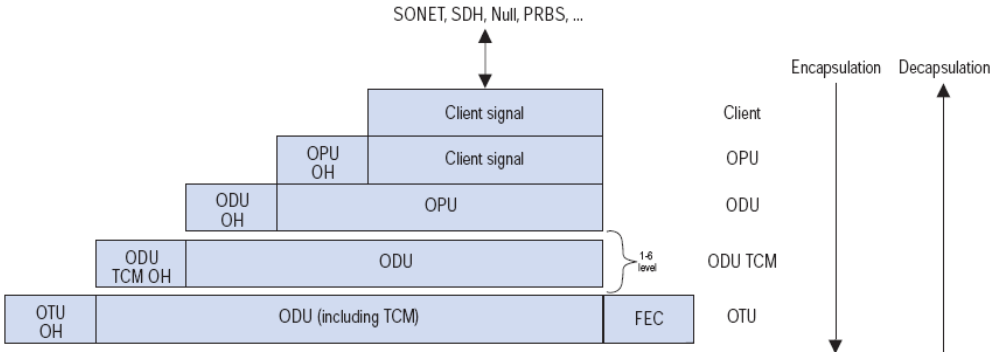

Figura 7 – Encapsulamento do multi-quadro OTN [GEN06].

2.2.1 Cabeçalho

O cabeçalho (*overhead*) representa a área do quadro onde estão definidos os *bytes* de controle da comunicação OTN. Como partes destes controles podem-se citar: funções supervisoras, condicionamento do transporte entre canais ópticos, monitoramento de conexões, adaptação do sinal do cliente para transporte sobre um canal óptico, controle de falhas e alarmes. Na G.709 o cabeçalho é dividido em quatro partes denominadas: *Frame Alignment Overhead*, *Optical Channel Payload Unit Overhead* (OPU OH), *Optical Channel Data Unit Overhead* (ODU OH) e *Optical Channel Transport Unit Overhead* (OTU OH). A Figura 8 ilustra os pontos de terminação de cada

uma das partes em uma rede OTN e a Figura 9 ilustra todos os campos do cabeçalho e suas nomenclaturas.

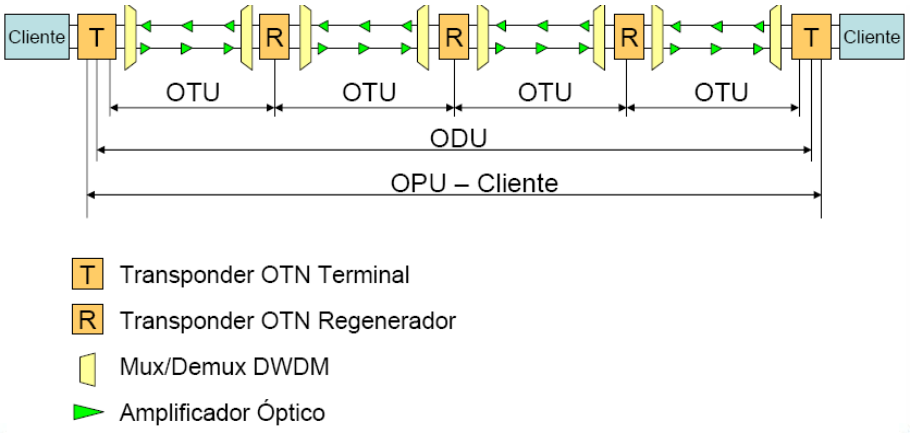

Figura 8 – Terminação OPU, ODU e OTU [NAK05].

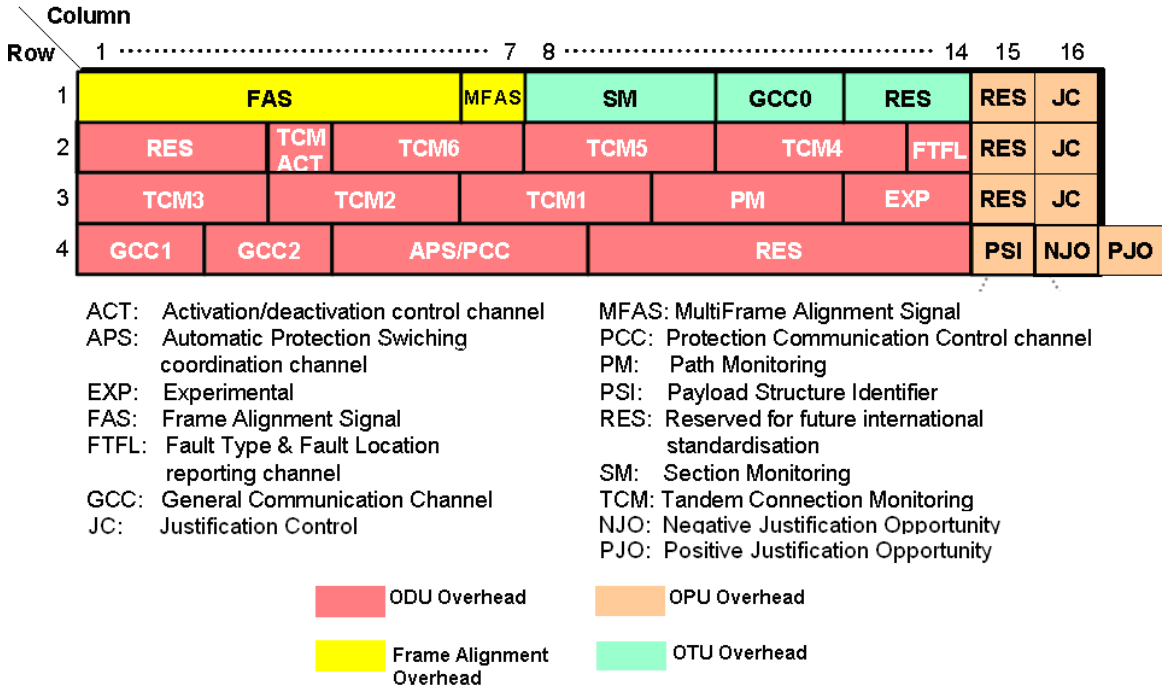

Figura 9 – Bytes que compõem o cabeçalho e seus nomes [VIS02].

2.2.1.1 Frame Alignment Overhead

O *Frame Alignment Overhead* é responsável por identificar os quadros OTN G.709 no sistema de transmissão, ou seja, é utilizado para determinar onde um quadro começa e onde ele termina. Ele é composto por dois campos com funcionalidades distintas. O campo FAS (*Frame Alignment Signal*) ilustrado na Figura 10 é composto por 6 *bytes* que contêm um valor fixo igual a F6F6F6282828 (hexadecimal), sendo F6 definido como OA1 e 28 como OA2. O campo MFAS (*Multiframe Alignment Signal*) é continuamente incrementado multi-quadro após multi-quadro de 0 até 255. Isto é útil em estruturas compostas a partir de mais de um multi-quadro, onde o significado da mensagem é determinado pela composição da informação ao longo de um período que pode

variar de 64 a 256 multi-quadros.

FAS OH Byte 2								FAS OH Byte 2								FAS OH Byte 3								FAS OH Byte 4								FAS OH Byte 5								FAS OH Byte 6							
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
OA1								OA1								OA1								OA2								OA2								OA2							

Figura 10 – Campos OA1 e OA2 do FAS.

2.2.1.2 Optical Channel Payload Unit Overhead (OPU OH)

O *Optical Channel Payload Unit Overhead* (OPU OH) é a estrutura utilizada para adaptar a informação que vem do cliente para o transporte em um canal óptico. Ele contém informações necessárias para executar a adaptação entre as taxas de transmissão do sinal do cliente e as taxas de recepção do sinal da carga útil do OPU.

O cabeçalho do OPU possui: *Payload Structure Identifier* (PSI), o qual contém 1 byte. No entanto, a informação total do PSI é composta por 256 bytes, sendo formada por vários multi-quadros. O primeiro byte do PSI é o P*ayload Type* (PT), os demais bytes estão reservados para futura padronização. O OPU possui também cabeçalhos associados com o mapeamento do sinal do cliente dentro da carga útil (JC, PJO e NJO).

O byte JC (*Justification Control*) é um cabeçalho que é utilizado quando há *jitter* entre o relógio do cliente e o relógio do OTU. Três *bytes* JC são utilizados para realizar o controle, sendo que é utilizado o voto majoritário, significando que é necessário ter dois entre os três *bytes* igualmente assinalados para realizar a justificativa positiva PJO (*Positive Justification Opportunity*) ou negativa NJO (*Negative Justification Opportunity*). A justificativa consiste em adicionar ou remover um *byte* da carga útil. É possível realizar a justificativa até um *byte* positivo (no caso do OPU ter recebido um *byte* a menos do que o esperado) ou um *byte* negativo (no caso do OPU ter recebido um *byte* a mais do que o esperado). O objetivo é deixar a carga útil sempre do mesmo tamanho. Se a diferença da freqüência do relógio do cliente e do relógio do OPU estiver fora do intervalo de *bit rate* em que é possível realizar a justificativa é enviado um sinal de alarme para o cliente. A Figura 11 descreve os possíveis valores de JC.

JC	NJO	PJO
00	justification byte	data byte
01	data byte	data byte
10*	justification byte	data byte
11	justification byte	justification byte
* O módulo que realiza o mapeamento não gera esta sequência. Devido a erros na transmissão dos bits, o módulo que realiza o desmapeamento poderá recebe-lá .		

Figura 11 – Possíveis valores do JC e o comportamento dos bytes NJO e PJO.

2.2.1.3 Optical Channel Data Unit Overhead (ODU OH)

O *Optical Channel Data Unit Overhead* (ODU OH) é a estrutura utilizada para controlar o caminho percorrido pelo quadro. Ele inclui informação para manutenção de funções operacionais

para dar suporte a canais ópticos. O ODU OH é formado por *bytes* de controle dedicados a realizar supervisão ponto a ponto para os seis níveis de monitoramento TCM (*Tandem Connection Monitoring*), estrutura responsável pelo monitoramento dos sinais que será apresentada no item 2.3.2.

2.2.1.4 Optical Channel Transport Unit Overhead (OTU OH)

O *Optical Channel Transport Unit Overhead* (OTU OH) é a estrutura utilizada para transportar um ODU por um ou mais pontos de terminação dos canais de conexão óptica, onde a regeneração 3R entra em cena. O cabeçalho do OTU é composto pelo SM (*Section Monitoring*), o qual contém o *Trail Trace Identifier* (TTI), que é um byte composto por 64 multi-quadros que indica a origem e o destino dos quadros trafegando na rede, o *Bit Interleaved Parity* (BIP-8), que é um byte usado para detecção de erros, o SM contém também vários *bits* referentes a alarmes, sendo eles: *Backward Error Indicator* (BEI), *Backward Iincoming Alignment Error* (BIAE), *Backward Defect Indicator* (BDI) e *Incoming Alignment Error* (IAE). O OTU OH também é composto pelo GCC (*General Communication Channel*) que é um cabeçalho utilizado para a transmissão de informações entre pontos de terminação OTU e pelo FEC, que é um método de geração de código de correção de erros.

O OTU possui uma característica diferente dos demais cabeçalhos que é o embaralhamento (*scrambling*) dos seus *bytes*. O scrambling consiste em embaralhar a informação a ser transmitida utilizando um LFSR, de modo que não seja adicionado nenhum cabeçalho adicional. Com isto, tem-se uma proteção para que não ocorram longas seqüências de 0s ou de 1s, o que garante suficiente troca de estado dos *bits* para facilitar a regeneração do relógio e também evita uma possível repetição da seqüência de alinhamento. A operação de embaralhar é realizada em todo o multi-quadro OTN G.709, com exceção do campo FAS, e é realizada após a computação do FEC. A Figura 12 ilustra o LFSR, obtido a partir do polinômio gerador $1 + x + x^3 + x^{12} + x^{16}$.

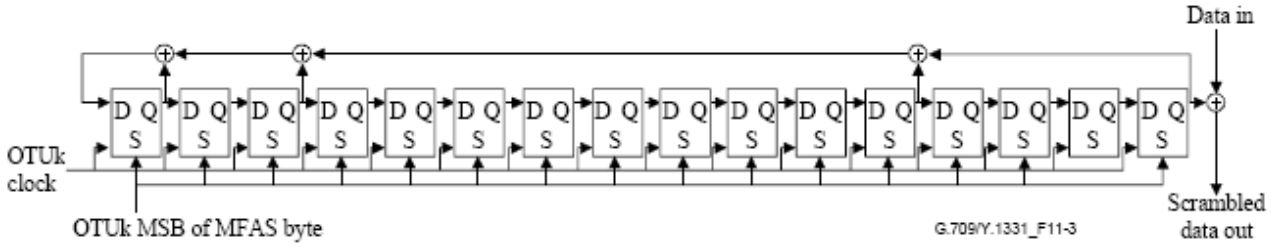

Figura 12 – O polinômio gerador $1 + x + x^3 + x^{12} + x^{16}$ [ITU02].

2.2.2 Carga Útil

A carga útil (*payload*) contém as informações e os controles provenientes dos clientes a serem transportados pela rede OTN. O protocolo a ser encapsulado pode ser SONET/SDH, GbE, 10GbE, ATM, IP e assim por diante. Esta facilidade de poder encapsular diversos protocolos é devida à característica do OTN, que permite o transporte transparente dos sinais do cliente.

2.3 Vantagens do OTN em Relação aos Padrões SONET/SDH

OTN oferece as seguintes vantagens em comparação com os padrões SONET/SDH:

- Correção de erro mais robusta;
- Mais níveis de *Tandem Connection Monitoring* (TCM);
- Transporte transparente dos sinais do cliente;
- *Switching Scalability* (Escalabilidade no chaveamento).

2.3.1 Forward Error Correction (FEC)

O FEC é uma das maiores justificativas para a utilização do padrão OTN G.709. A integridade da informação é um dos pontos fortes do G.709, que ainda conta com o código de paridade BIP-8 (*Bit Interleaved Parity*) aplicado sobre a carga útil. O FEC utiliza o método de geração de código de correção de erros *Reed-Solomon* para produzir informação redundante a ser transmitida sendo utilizada para localizar e corrigir erros pelo receptor.

O ganho gerado pela utilização de correção de erros FEC pode ser de até 6.2 dB, ou seja, devido ao menor número de erros na comunicação é possível transmitir um sinal a certa taxa BER (*Bit Error Rate*) utilizando 6.2 dB a menos de potência. Tal ganho pode ser utilizado para:

- Aumentar o comprimento máximo de um nodo ou o número de nodos resultando em uma extensão maior da fibra óptica;
- Aumentar o número de canais em um sistema DWDM o qual é limitado pela potência de saída dos amplificadores, isto é possível devido à menor potência por canal;
- Porém o mais importante é que o FEC aumenta o número de elementos de redes ópticas que podem ser cruzados por um caminho óptico antes que a regeneração 3R seja necessária. Isto possibilita a evolução dos enlaces ponto a ponto de hoje em dia para redes ópticas transparentes utilizando topologia malha.

2.3.2 Tandem Connection Monitoring (TCM)

O monitoramento dos sinais dos padrões SONET/SDH é dividido em seção, linha e caminho. Um problema surge quando se tem uma situação como mostrada na Figura 13, onde é necessário monitorar um segmento do caminho que passa por outro operador.

Na Figura 13 o Operador A precisa que o Operador B carregue o seu sinal. No entanto ele também precisa uma forma de monitorar o sinal quando ele passa pela rede do Operador B. Esta é a função do *tandem connection*. É uma camada entre monitoração de linha e monitoração de caminho. SONET/SDH foram modificados para permitir uma única *tandem connection*, o G.709 permite seis.

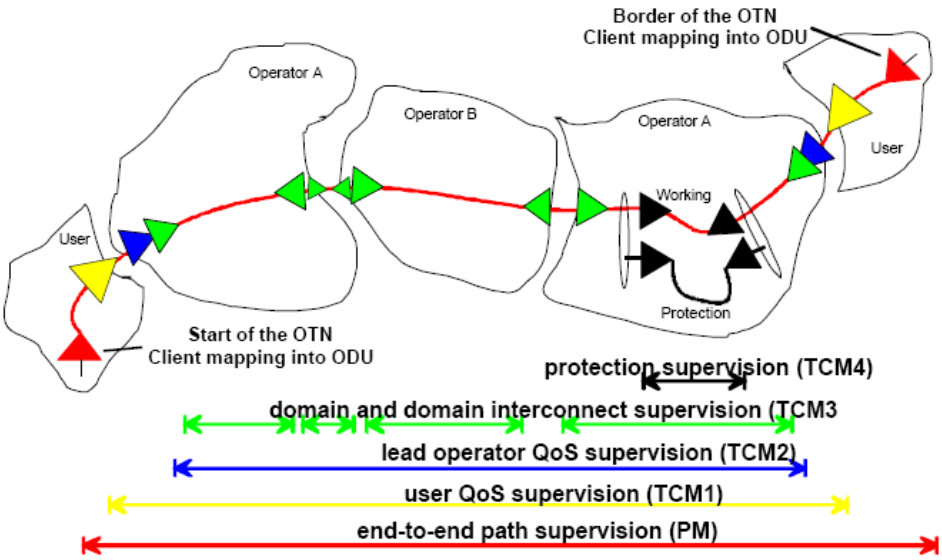

Figura 13 – Tandem Connection Monitoring [VIS02].

2.3.3 Transporte Transparente dos Sinais do Cliente

O G.709 define o OPUk [ITU03] o qual pode conter um sinal SONET/SDH inteiro. Isto significa que é possível transportar quatro sinais STM-16/OC-48 em um OTU2 sem precisar modificar nenhum cabeçalho do SONET/SDH. Assim, pode-se dizer que o transporte de sinais do cliente no OTN é:

- *Bit-transparent*, ou seja, a integridade de todo o sinal do cliente é mantida;
- *Timing-transparent,* significa que o modo de mapeamento assíncrono transfere o tempo de entrada para a outra extremidade;
- *Delay-transparent*, por exemplo, se quatro sinais STM-16/OC-48 são mapeados em sinais ODU1 e depois multiplexados em sinais ODU2, suas relações de tempo são preservadas até que sejam demapeados de volta para ODU1.

2.3.4 Switching Scalability

Quando o padrão SONET/SDH foi desenvolvido na década de oitenta, sua principal função era prover tecnologia de transporte para serviços de voz. Foram definidos dois níveis de chaveamento: (i) de ordem baixa a 1.5/2 Mbit/s para suportar diretamente os sinais de voz T1/E1 e (ii) de ordem alta a 50/150 Mbit/s para engenharia de tráfego. Níveis de chaveamento em maiores taxas de *bit* não foram pensados [ITU03].

Ao longo do tempo, a taxa de transmissão cresceu enquanto a taxa de chaveamento permaneceu fixa. Concatenações virtuais e contíguas foram introduzidas para solucionar parte dos problemas, já que elas suportam serviços acima das taxas de *bit* dos padrões SONET/SDH. Porém, a diferença entre as taxas de *bit* das linhas ou serviços e as taxas de *bit* de chaveamento continua a existir, mesmo com concatenação. O OTN fornece uma solução para este problema, sem que seja

colocada alguma restrição na taxa de chaveamento de *bit*. À medida que cresce a taxa de *bit* das linhas, novas taxas de chaveamento de *bit* são adicionadas. Um operador pode oferecer serviços a várias taxas de *bit* (2.5G, 10G, etc) independentemente da taxa de *bit* do comprimento de onda utilizado na transmissão óptica, utilizando as funções de multiplexação e multiplexação inversa contidas no padrão OTN.

3 Fluxo de Projeto

Devido à alta taxa de transmissão do padrão G.709, o projeto requer um controle cuidadoso da síntese de hardware. É necessário realizar o projeto dos módulos visando um alto desempenho para que os requisitos de *timing* sejam cumpridos. Para melhorar o desempenho de módulos codificados em VHDL a serem prototipados em FPGA, é necessário realizar uma codificação apropriada e realizar otimizações durante a síntese física. Este processo é conhecido como *timing closure* [WHA04]. A maioria das otimizações são realizadas por ferramentas EDA (*Electronic Design Automation*), tendo como base diretivas informadas pelo projetista. Dentre estas otimizações pode-se citar o planejamento topológico otimizado dos módulos no FPGA (planta baixa) e a síntese da árvore de relógio, bem como a otimização do roteamento e do posicionamento das células. Para verificar o cumprimento dos requisitos do projeto, é necessária a análise estática de *timing* (*Static Timing Analysis*), STA.

É utilizada a plataforma ISE (*Integrated Software Environment*) como ferramenta para o ambiente de projeto. A ferramenta engloba as etapas necessárias para a implementação de dispositivos reconfiguráveis da Xilinx: síntese lógica, síntese física, STA e simulações realizadas durante as diversas etapas do desenvolvimento do projeto. Cada etapa é realizada por uma ferramenta específica. O fluxo utilizado para que o projeto alcance o desempenho requisitado é apresentado na [WHA04].

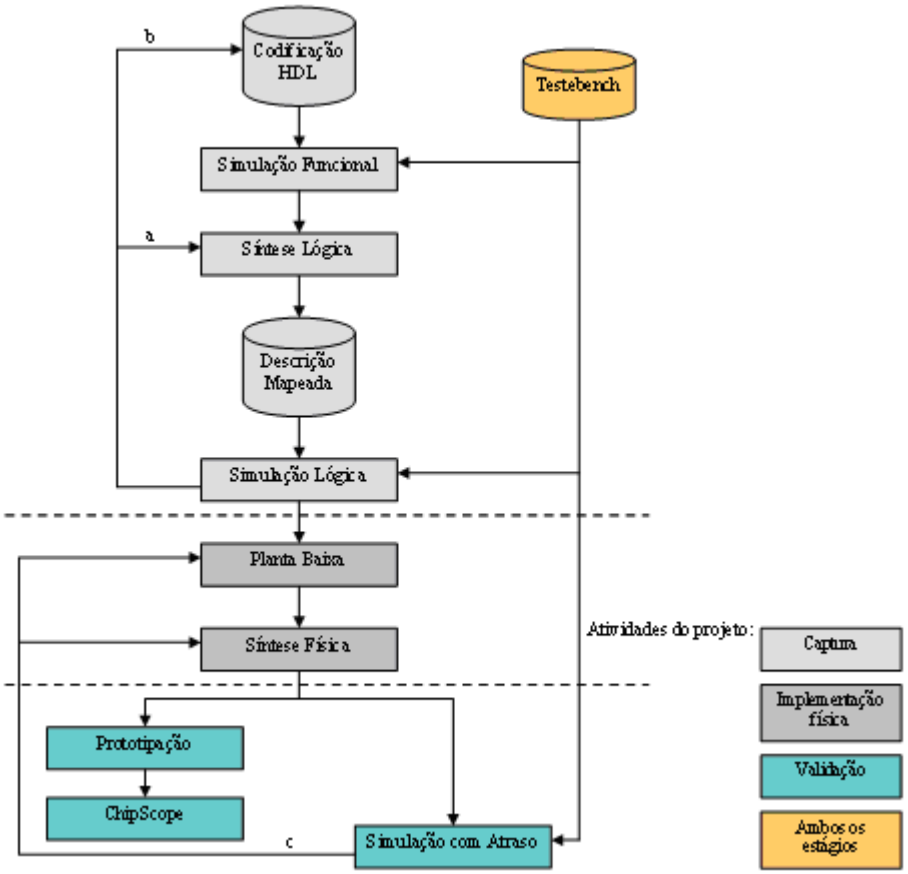

Figura 14 – Fluxo de projeto utilizado no desenvolvimento dos módulos propostos no trabalho.

Para realizar o projeto da arquitetura em um dispositivo reconfigurável de modo a atender as restrições de temporização e de área é necessário realizar três atividades de projeto, sendo elas: captura, implementação física e validação.

3.1 Captura

Durante a atividade de captura, o projeto é desenvolvido (codificado) e sintetizado. Durante a realização desta atividade verifica-se o correto comportamento do projeto em relação à codificação. Utiliza-se o simulador ModelSim para tal verificação. A próxima etapa é a realização da síntese lógica, etapa executada pela ferramenta XST (Xilinx Synthesis Technology).

A síntese lógica tem como objetivo principal fazer o mapeamento de descrição abstrata, tipicamente no nível de RTL (*Register Transfer Level*) para elementos de hardware de uma tecnologia específica, obtendo uma descrição mais detalhada do circuito em termos de portas lógicas [CAR01]. Espera-se ainda nesta etapa que ocorra uma minimização lógica do circuito, de forma a reduzir área do mesmo. Este processo é guiado por restrições (*constraints*). Existem diferentes requisitos e restrições tais como de área e de temporização onde é possível, por exemplo, definir a freqüência de relógio.

Obtido o circuito mapeado é possível realizar a simulação lógica, a qual consiste em verificar o correto funcionamento do circuito mapeado para uma tecnologia específica, ou seja, verificar se o mesmo realiza a função desejada. Nesta etapa podem ser analisados os atrasos dos caminhos críticos do circuito, verificando se o circuito atende as restrições temporais do projeto. Caso o circuito não funcione corretamente o projetista deve voltar para etapa de síntese lógica e ajustar algumas restrições (– a: ajuste de restrições), ou para a etapa de codificação HDL e recodificar o circuito para atender as restrições do projeto (– b: ajuste de restrições).

3.2 Implementação Física

A segunda atividade é a de implementação física do projeto. Nesta atividade inserem-se restrições de temporização referentes aos atrasos dos caminhos críticos verificados na etapa de simulação lógica, de forma a guiar o posicionamento e roteamento do projeto. Neste estágio é realizada a síntese física, a qual realiza o mapeamento físico em um dispositivo reconfigurável. Este processo tem por objetivo especificar a área em que os módulos se localizarão no dispositivo (planta baixa), em seguida realizando o posicionamento das células e definição da geometria das interconexões entre estas células (posicionamento e roteamento).

A planta baixa tem como objetivo determinar a geometria dos componentes físicos e o posicionamento dos pinos, procurando atender as restrições especificadas no projeto. Estes procedimentos consistem em identificar os blocos ou módulos que devem ficar juntos e organizá-los () a fim de aumentar o desempenho do circuito como um todo [WHA04]. Uma planta baixa

inicial é criada após a etapa de síntese lógica servindo de guia para o projeto físico. Se os tamanhos dos componentes ou dos fios verificados após a síntese física forem significativamente diferentes dos estimados na planta baixa inicial, pode-se realizar uma nova etapa de planta baixa de forma que as restrições impostas sejam posteriormente atendidas [WOL94].

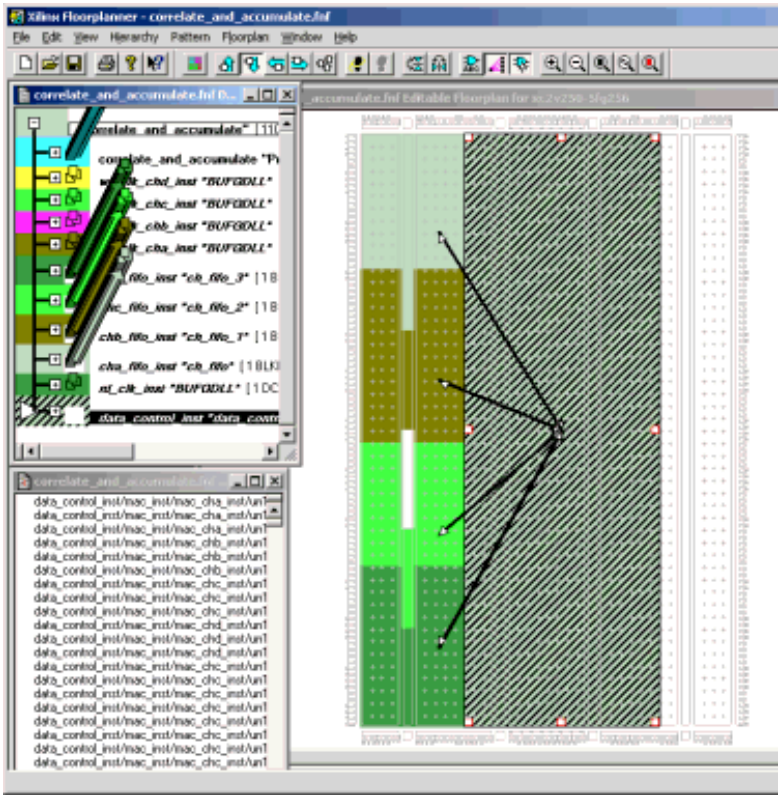

Figura 15 – Planejamento topológico do circuito [WHA04].

As etapas de posicionamento e roteamento são realizadas automaticamente pelas ferramentas de CAD [BRO92]. O posicionamento atribui uma localização específica para os diversos blocos do projeto dentro do dispositivo reconfigurável a partir de estimativas dos tamanhos dos blocos e dos números de fios entre esses blocos [WOL94]. A busca por uma minimização dos atrasos em caminhos críticos, do comprimento total, da densidade máxima e do número de conexões faz parte desta etapa. A qualidade do posicionamento define a dificuldade do roteamento.

O roteamento estabelece rotas de conexão entre as portas lógicas e entre estas e os pinos de E/S. Esta etapa é geralmente dividida em roteamento global e roteamento detalhado. O roteamento global divide a região de roteamento em menores seções que podem ser projetadas independentemente, simplificando assim o roteamento detalhado dessas seções. O roteamento detalhado realiza a conexão propriamente dita entre células e pinos. Há ainda outras etapas de roteamento, realizadas por ferramentas específicas, como roteamento de alimentação e roteamento de relógio.

Após o posicionamento das células e a definição da geometria das conexões é possível conhecer o atraso de cada célula e conexão do circuito. Com isso, é possível realizar a simulação

pós-layout, o qual contém informações de atraso do circuito, a fim de realizar uma simulação com atrasos reais, conhecida como simulação SDF (*Standard Delay Format*).

3.3 Validação

A atividade final do fluxo de projeto é a validação do projeto através de simulação com atrasos e prototipação, a qual consiste em validar o projeto em hardware. Considera-se a simulação com atraso uma etapa importantíssima no desenvolvimento do projeto, pois é onde são considerados os atrasos de cada célula e de conexões. Após realizar uma análise será possível saber se os requisitos de temporização do projeto foram ou não cumpridos. Caso as restrições do projeto não venham a ser cumpridas, o projetista deve voltar para as etapas de planta baixa e síntese física para ajustar estes requisitos (– c: ajuste de requisitos). Na validação, a análise de timing estático determina se o projeto alcança o desempenho desejado. A partir da atividade de validação o projetista descobre em que etapas do fluxo de projeto é possível fazer alterações para que ocorra um aumento no desempenho do circuito.

A validação em hardware é o processo que determina se o projeto está funcionando corretamente em um dispositivo reconfigurável. Esta etapa é realizada em nível de implementação física, através de testes em um dispositivo reconfigurável. A análise lógica dos sinais internos do circuito que se encontram no dispositivo é realizada com o uso da ferramenta ChipScope. Esta permite ao projetista visualizar o funcionamento do hardware da mesma forma que faria com um analisador lógico, sem a necessidade da utilização de equipamentos de teste.

Para realizar as simulações apresentadas neste documento foi desenvolvido um *testbench*, este é responsável pela geração de estímulos de entrada para verificação do comportamento do circuito. É importante a utilização do mesmo *testbench* para a simulação comportamental e temporal, pois é necessário verificar se os atrasos introduzidos pela implementação do circuito numa dada tecnologia não alteram a sua funcionalidade.

Para o presente trabalho as etapas de síntese e simulações foram realizadas tendo como alvo o dispositivo FPGA Virtex4 XC4VFX100 com speed grade -10.

4 Módulo Alinhador de Quadro

Um módulo de interface com a fibra óptica, externo ao contexto deste trabalho, é responsável por paralelizar o fluxo de dados a 10,688 Gbps em 16 canais diferenciais (LVDS), cada um a uma taxa de 669,3125 Mbps. O FPGA é responsável por receber cada um destes fluxos, paralelizando-os em 4 bits (Figura 16). Desta forma, a freqüência interna de trabalho é reduzida a 167,33 MHz, e os dados disponibilizados em uma largura de 64 bits.

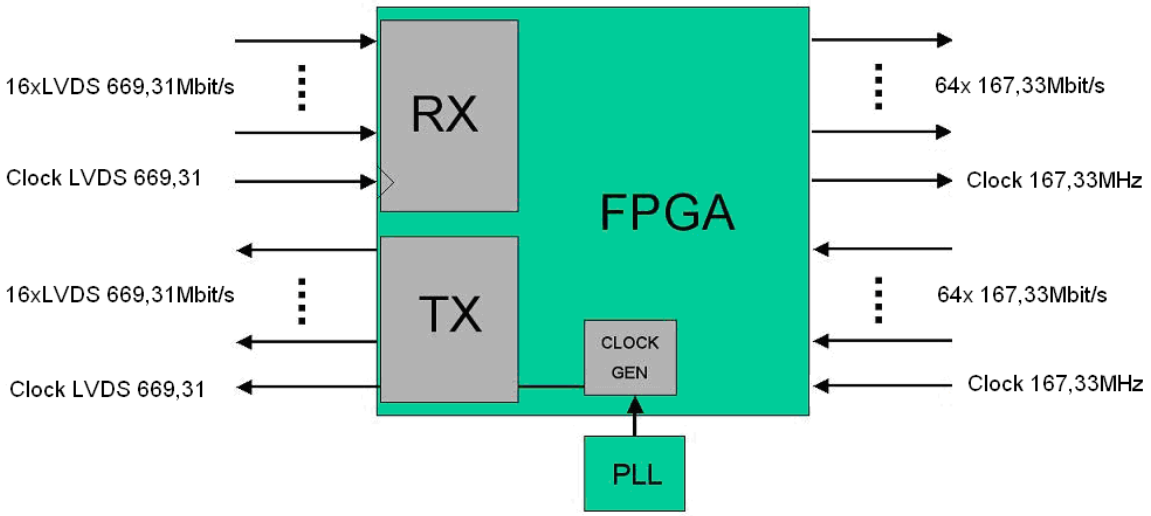

Figura 16 – Interface com a fibra óptica.

O módulo alinhador de quadro tem então como restrição de projeto trabalhar com um fluxo de dados de largura 64 *bits*, operando a uma freqüência de 167,33 MHz. No decorrer do trabalho é adotada como freqüência de operação 200 MHz, o que resulta em uma taxa de 12,8 Gbps. Esta maior taxa de operação é adotada como cenário de pior caso, permitindo o desenvolvimento de um projeto mais robusto, levando em conta que as principais considerações quanto ao alinhamento de multi-quadros são alta confiabilidade e resposta rápida [CHO03].

Para delimitar o início de cada multi-quadro OTN é utilizada uma seqüência de *bytes* definida pela palavra em hexadecimal “F6F6F6282828”. De acordo com o padrão OTN, pode-se utilizar um subconjunto destes seis *bytes* para realizar o alinhamento e o sincronismo inicial. Optou-se por utilizar um subconjunto de três *bytes*, sendo eles os seguintes: “F6XXF628XXXX”, sendo que “X” representa posições inicialmente desconsideradas na procura pela seqüência de alinhamento. Esta seqüência de *bytes* pode estar começando em qualquer *bit* dos 64 *bits* recebidos nos pinos de entrada, inclusive o caso de estar começando nos últimos *bits*. O que nos levará a ter os *bits* restantes no início da próxima palavra de 64 *bits* recebida pelos pinos de entrada (Figura 17, caso 1) denominada palavra futura.

O início de cada multi-quadro é composto por esta seqüência, por isso então terá que se

realizar a identificação bem como o deslocamento dos *bits* das palavras de 64 *bits* recebidas.

O módulo alinhador de quadro é formado por sub-módulos que consistem em identificar a ocorrência da seqüência de alinhamento, identificar em qual bit a seqüência está começando, realizar o deslocamento de modo que a seqüência de alinhamento comece no início da palavra de 64 *bits* (Figura 17, caso 2) e alimentar a máquina de estados que realiza o sincronismo.

Figura 17 – Exemplo de desalinhamento e posterior alinhamento.

Uma vez realizado o alinhamento, todas as palavras seguintes do multi-quadro terão que ser deslocadas o mesmo número de *bits*.

4.1 Arquitetura

Como citado anteriormente, o alinhador de quadro é dividido em sub-módulos. Cada sub-módulo foi separado por um registrador, formando então um *pipeline* (Figura 18). Esta divisão é necessária para que seja atendida a restrição de tempo de propagação combinacional do projeto, igual a 5 ns.

Primeiramente, os 64 *bits* da palavra atual (palavra armazenada no registrador do 1º estágio) e os 64 *bits* da palavra futura (palavra que ainda não entrou no primeiro estágio) são processados pelo sub-módulo **comparador parcial**, que é responsável por identificar a ocorrência da seqüência de alinhamento a partir da verificação de um subconjunto de três *bytes*. É necessário realizar esta verificação 64 vezes, pois a seqüência de alinhamento pode estar começando em qualquer *bit* da palavra atual. Uma vez identificada esta subseqüência de alinhamento, são passados 64 *bits* ao sub-módulo **gerador de endereço**. Havendo alinhamento um destes *bits* estará em nível lógico '1' indicando onde o início da seqüência de alinhamento começou. De posse do índice onde este *bit* está localizado é gerado o endereço em que o mesmo se encontra e o sinal *match* indica a ocorrência da seqüência de alinhamento atuando como *enable* do registrador de endereço. O módulo **deslocador** utiliza este endereço, que corresponde ao número de *bits* a serem deslocados para realizar o deslocamento correto entre a palavra atual e a palavra futura. Por fim, é realizada uma última verificação pelo **comparador completo**, que confere se os 6 *bytes* da seqüência de alinhamento estão corretos e alimenta a máquina de estados que realiza o controle do sincronismo de multi-quadros, que será apresentada posteriormente na integração dos módulos desenvolvidos.

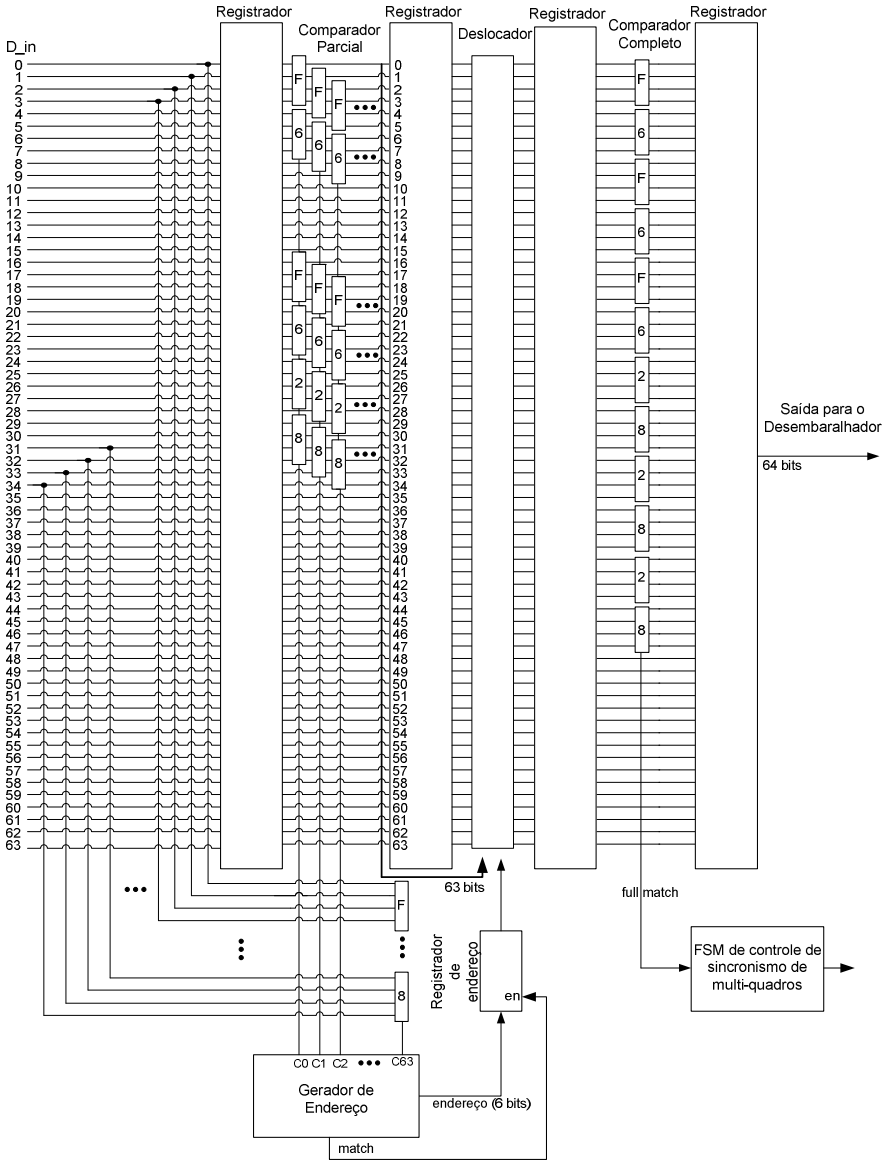

Figura 18 – Arquitetura do módulo alinhador de quadro com pipeline.

Para a implementação dos módulos que compõem o alinhador de quadro fez-se uso de lógica combinacional tanto na sua arquitetura como na codificação, para ganho de desempenho tanto em área utilizada como em velocidade de processamento. Devido aos requisitos do projeto, esta não foi uma escolha, e sim, uma necessidade.

A partir de estudos realizados utilizando a opção FPGA EDITOR presente na ferramenta ISE, foi possível identificar como a codificação estava sendo mapeada fisicamente. Foi verificado que a ferramenta de síntese inferia multiplexadores quando eram realizadas operações lógicas com mais de quatro entradas, o que resultava em um acréscimo no caminho crítico. Assim, optou-se por utilizar portas lógicas de quatro entradas nos módulos desenvolvidos.

4.1.1 Comparador Parcial

O sub-módulo comparador parcial, cujo código VHDL encontra-se no Apêndice I, é responsável pela identificação de um subconjunto de três dos seis *bytes* da seqüência de alinhamento, sendo estes "F6XXF628XXXX". Como a seqüência de alinhamento pode estar começando em qualquer *bit* dos 64 *bits* da palavra recebida, são necessárias 64 instâncias deste sub-módulo. O sinal de saída, que indica a ocorrência da seqüência de alinhamento, é um vetor de 64 posições onde cada posição do vetor corresponde ao número de *bits* deslocados em que a seqüência foi encontrada. Cada *bit* deste vetor é a saída de uma instância do comparador parcial.

A Figura 19 ilustra uma instância em que a verificação é feita exatamente nos primeiros *bits* da palavra atual. Neste caso, se a saída indicada por C[0] estiver em nível lógico '1', significa que a seqüência de alinhamento já está na posição correta e não precisa ser deslocada. Já a Figura 20 ilustra uma instância em que a verificação está começando no último *bit* da palavra atual, o *bit* 63. Os demais *bits*, do 64 ao 94 são da palavra futura, sendo o *bit* 64 equivalente ao *bit* 0 da palavra futura e o *bit* 94 equivalente ao *bit* 30 da palavra futura. Neste caso, se a saída indicada por C[63] estiver em nível lógico '1', significa que a seqüência de alinhamento está desalinhada em 63 *bits*.

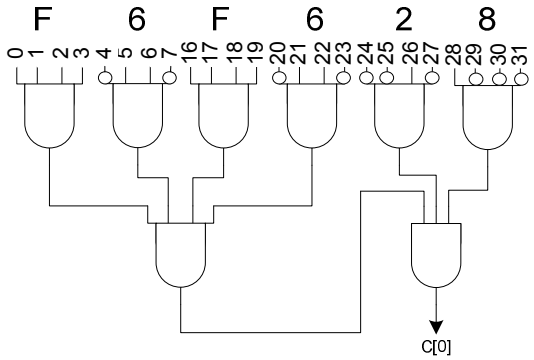

Figura 19 – Instância do Comparador Parcial sem deslocamento.

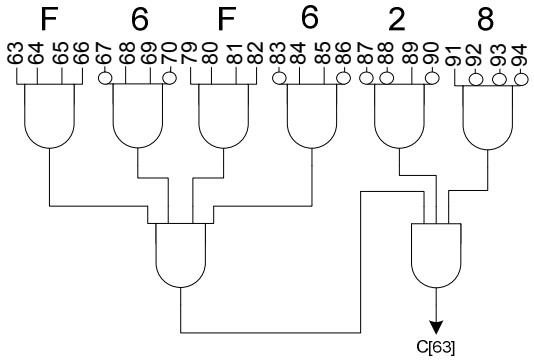

Figura 20 – Instância do Comparador Parcial com deslocamento de 63 bits.

Portanto, no sinal de saída composto por 64 *bits*, o *bit* que estiver em nível lógico '1' indica o *bit* de início da seqüência de alinhamento. A próxima etapa consiste em transformar o índice onde este *bit* se encontra em um endereço para realizar o deslocamento necessário.

4.1.2 Gerador de Endereço

O sub-módulo gerador de endereço é responsável pela conversão do sinal *controle*, proveniente da saída do comparador parcial, para um endereço de 6 *bits*. Este endereço corresponde ao índice em que a saída do comparador parcial estava em nível lógico '1' (Figura 21). Também cabe ao gerador de endereço a tarefa de acionar o sinal *match* toda vez que o sinal controle tiver um de seus *bits* em nível lógico '1'.

```
Sinal de saida do comparador parcial:
00000000000000000000000000000000000001000000000000000000000000

Endereço correspondente em binário: 100111
```

Figura 21 – Exemplo de conversão realizada pelo Gerador de Endereço.

A arquitetura utilizada no sub-módulo gerador de endereço está representada na Figura 22. Cada instância computa um *bit* do endereço através da operação lógica OU. As entradas das portas lógicas OU correspondem aos *bits* que, para sua identificação por um endereço binário, irão utilizar aquele *bit* de endereço.

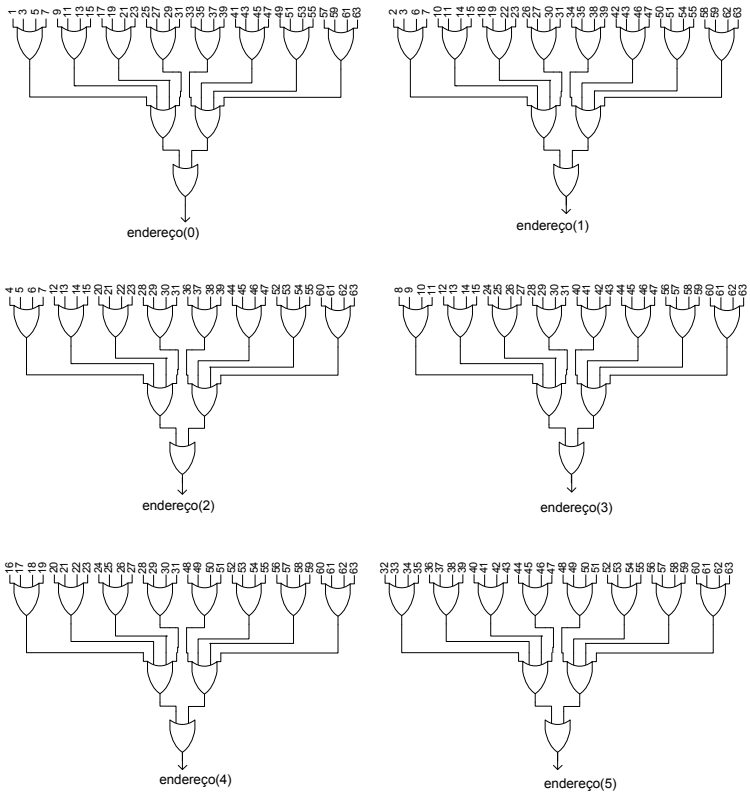

Figura 22 – Instancias do Gerador de Endereço.

O endereço gerado pelo módulo corresponde ao *bit* em que a seqüência de alinhamento está iniciando, que por sua vez indica também o número de *bits* que deverão ser deslocados pelo sub-módulo deslocador.

4.1.3 Deslocador

O sub-módulo deslocador tem como objetivo realizar o deslocamento da seqüência de alinhamento de N posições, onde N varia de 0 a 63. O número de posições a serem deslocadas é passado pelo gerador de endereço.

Usou-se um arranjo logarítmico de multiplexadores 2x1, pois uma LUT (*LookUp Table*) tem 4 entradas. Assim, otimizou-se a área e o tempo de processamento.

A Figura 23 ilustra um deslocador de 8 *bits*, o qual possui três estágios (colunas de multiplexadores). Neste caso, cada estágio realiza o deslocamento de 1, 2 ou 4 *bits*. Para o deslocamento de 64 *bits* são necessários seis estágios ($2^6 = 64$). O princípio de funcionamento do deslocador de 64 *bits* é exatamente o mesmo do de 8 *bits*, porém o número de multiplexadores cresce exponencialmente, o que torna sua ilustração gráfica muito complicada.

O endereço recebido entra como controle dos multiplexadores. Tendo como exemplo a Figura 23, teríamos um endereço de 3 *bits*, sendo que as entradas identificadas pelos números 1 até 8 podem ser entendidas como a palavra atual, e as entradas identificadas por N_1 até N_7 podem ser entendidas como a palavra futura. Caso se tivesse, por exemplo, o endereço 010, teríamos que deslocar 2 *bits* pois 010 em binário corresponde a 2 em decimal. De acordo com a figura, o *bit* menos significativo é o controle dos multiplexadores que estão ligados diretamente a saída, o *bit* mais significativo é o controle dos multiplexadores que estão ligados na entrada e o *bit* intermediário é o controle dos multiplexadores intermediários. Neste exemplo, colocando os multiplexadores intermediários em nível lógico ‘1’ se terá um deslocamento de 2 *bits*, o que está de acordo com a figura.

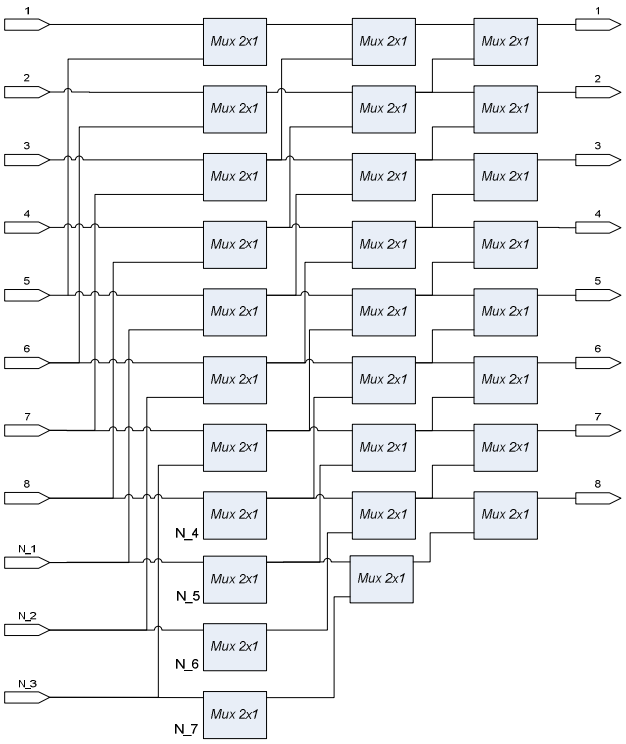

Figura 23 – Deslocador de 8 bits.

Devido à restrição de *timing* deste projeto, dividiu-se o deslocador de 64 *bits* em três estágios de *pipeline* para diminuir o tempo de processamento, sendo que cada estágio do *pipeline* é formado por duas colunas de multiplexadores.

4.1.4 Comparador Completo

O sub-módulo Comparador Completo é muito parecido com o Comparador Parcial. A diferença está na quantidade de *bytes* verificados, que no caso do comparador completo são verificados todos os 6 *bytes* da seqüência de alinhamento, sendo estes "F6F6F6282828". Outra diferença está no número de instâncias. Como o comparador completo encontra-se após o módulo que realiza o deslocamento, ele só precisa verificar a ocorrência da seqüência de alinhamento no início da palavra atual. Essa verificação assegura que a seqüência de alinhamento está com todos os seus *bytes* corretos, uma vez que no módulo comparador parcial são verificados apenas 3 dos 6 *bytes*. Se fossem utilizadas 64 instâncias do comparador completo haveria além do acréscimo de área, um tempo de processamento muito elevado caso fossem verificados todos os 6 *bytes*.

A Figura 24 ilustra o módulo Comparador Completo, o qual gera na saída o sinal *fullmatch,* este sinal alimenta a máquina de estados que controla o sincronismo dos multi-quadros sinalizando a ocorrência da seqüência de alinhamento.

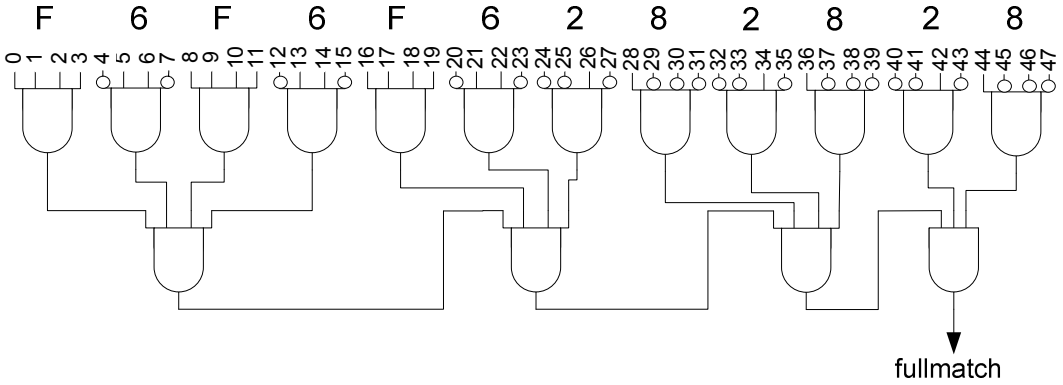

Figura 24 – Comparador Completo.

4.2 Validação

A validação do módulo Alinhador de Quadro se dá através de simulação com atraso, a qual contém os tempos de atraso das portas lógicas utilizadas para a implementação do módulo e também tempos de atraso referentes ao roteamento, permitindo assim uma verificação do módulo bastante realista.

Para realizar a verificação do módulo Alinhador de Quadro é preciso gerar estímulos externos como entrada do módulo. No caso do módulo sendo validado têm-se como entradas os seguintes sinais: clock, reset e "palavra futura" (de fato é o D_in da Figura 18, página 21).

A forma de onda apresentada na Figura 25 mostra a simulação com atraso do Alinhador de

Quadro.

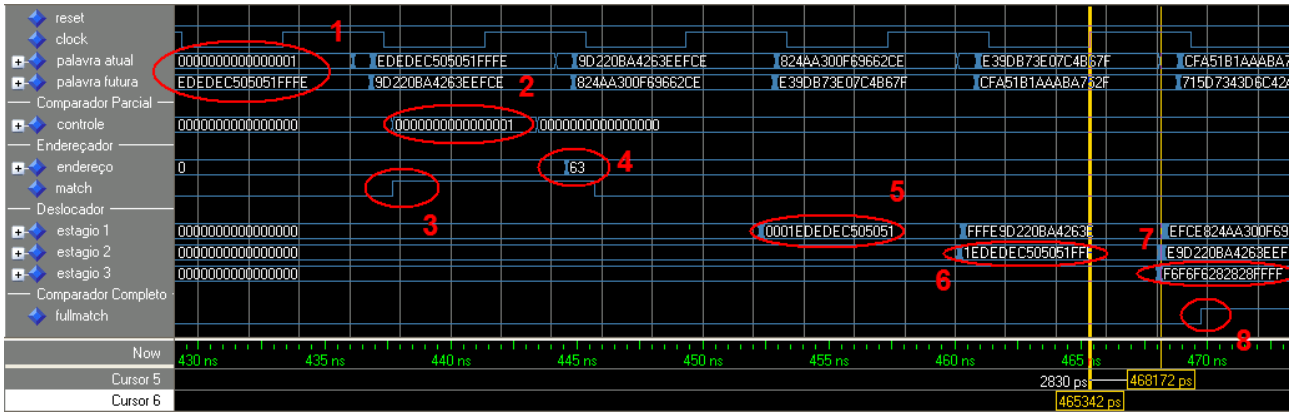

Figura 25 – Simulação do módulo Alinhador de Quadro com atraso.

Para verificar o correto funcionamento do módulo, inicialmente foi gerada a seqüência de alinhamento deslocada 63 *bits* de sua posição original (Figura 25, índice 1).

Temos na *palavra atual* o valor "0000000000000001" (hexadecimal), que corresponde a "00000000000000000000000000000000000000000000000000000000000000**1**" (binário) e "EDEDEC505051FFFE" (hexadecimal) na *palavra futura*, que corresponde a "**1110110111101101111011000101000001010000010100**1111111111111110" (binário). Da mesma maneira que foi mostrado na Figura 17 (página 20) é possível observar a seqüência de alinhamento deslocada, neste caso em 63 *bits*.

Após um ciclo de relógio é indicado o *bit* inicial da seqüência de alinhamento pelo sinal *controle* (Figura 25, índice 2), que aparece com valor "0000000000000001" (hexadecimal) que corresponde a "00000000000000000000000000000000000000000000000000000000000000001" (binário), e também o sinal *match* passa a ter nível lógico '1' (Figura 25, índice 3), indicando que a seqüência de alinhamento foi encontrada.

A próxima etapa consiste em gerar o endereço a ser utilizado pelo deslocador (Figura 25, índice 4). O endereço gerado é 63, valor que confere com o índice do *bit* com valor lógico '1' no sinal *controle* e com o número de *bits* que a seqüência de sincronismo está deslocada. Tendo o endereço, têm-se os três estágios do deslocador (Figura 25, índices 5, 6 e 7). A palavra de saída do sub-módulo deslocador tem o valor "F6F6F6282828FFFF" (hexadecimal) que corresponde a "1111011011110110111101100010100000101000001010001111111111111111" (binário), demonstrando que o deslocamento das palavras atual e futura foi feito corretamente. Por fim, o alinhamento é confirmado pelo sinal *fullmatch* (Figura 25, índice 8) em nível lógico '1'.

Tendo então a sua funcionalidade validada, também é mostrado na Figura 25 o tempo de propagação combinacional do módulo alinhador de quadro, que é 2,830 ns, indicado entre dois cursores amarelos. Tempo que está de acordo com o requisito que se tem no projeto, que é de 5 ns.

4.3 Resultado de Área

No atual projeto, além do requisito de tempo de propagação combinacional tem-se também um requisito que diz respeito à utilização de área da placa em que os módulos serão prototipados. Os dados de utilização de área do FPGA referentes ao módulo alinhador de quadro são apresentados na Tabela 2.

Tabela 2 – Utilização do FPGA pelo módulo alinhador de quadro.

Resumo da utilização do dispositivo Xilinx Virtex4 XCV4FX100 (valores estimados)			
Utilização Lógica	**Utilizado**	**Disponível**	**Utilização**
Número de Slices	632	42176	1%
Número de Slice Flip Flops	478	84352	0%
Número de LUTs de 4 entradas	903	84352	1%

É possível observar que o módulo alinhador de quadro utiliza apenas 903 LUTs de um total de 83352 LUTs disponíveis no dispositivo, o que representa uma utilização de aproximadamente 1%.

5 Módulo Embaralhador e Desembaralhador

Como visto anteriormente, a operação de *scrambling* consiste em embaralhar a informação a ser transmitida utilizando um LFSR, que é uma técnica de geração de dados pseudo-aleatório. Com isto, tem-se uma proteção para que não ocorram longas seqüências de 0s ou de 1s, o que garante suficiente troca de estado dos *bits* para facilitar a regeneração do relógio e também evita uma possível repetição da seqüência de alinhamento. A operação de embaralhar é realizada em todo o multi-quadro OTN G.709, com exceção do campo FAS. A funcionalidade do embaralhador e do desembaralhador é idêntica, uma vez que a operação de embaralhamento é simétrica. Basta passar um dado previamente embaralhado pelo sub-módulo que ele sairá desembaralhado.

Neste sub-módulo serão abordadas duas arquiteturas diferentes para a implementação do embaralhador e do desembaralhador.

5.1 Arquitetura Utilizando Gerador LFSR

Primeiramente, para se realizar a implementação do módulo embaralhador utilizou-se a arquitetura padrão referida em [ITU03], onde a geração da seqüência de embaralhamento é feita em hardware por LFSRs. Para o embaralhador do OTN foi definido como padrão o polinômio gerador $1 + x + x^3 + x^{12} + x^{16}$. A Figura 26 mostra um diagrama de blocos do módulo LFSR em hardware.

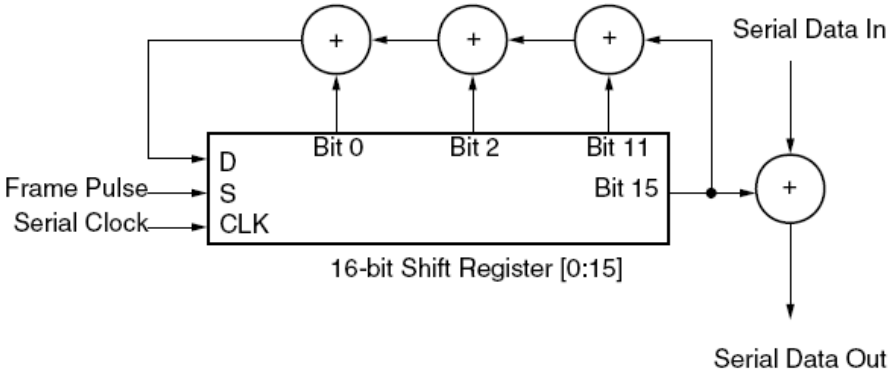

Figura 26 – Arquitetura de embaralhamento utilizando LFSR [SAW02].

Por se tratar de um LFSR serial, esta arquitetura tem de ser replicada 64 vezes para que os 64 *bits* de entrada possam ser embaralhados em um único ciclo de relógio. As instâncias replicadas precisam também estar interligadas para que haja propagação dos sinais. A Figura 27 ilustra um exemplo da arquitetura replicada, neste caso para uma entrada de 8 *bits*.

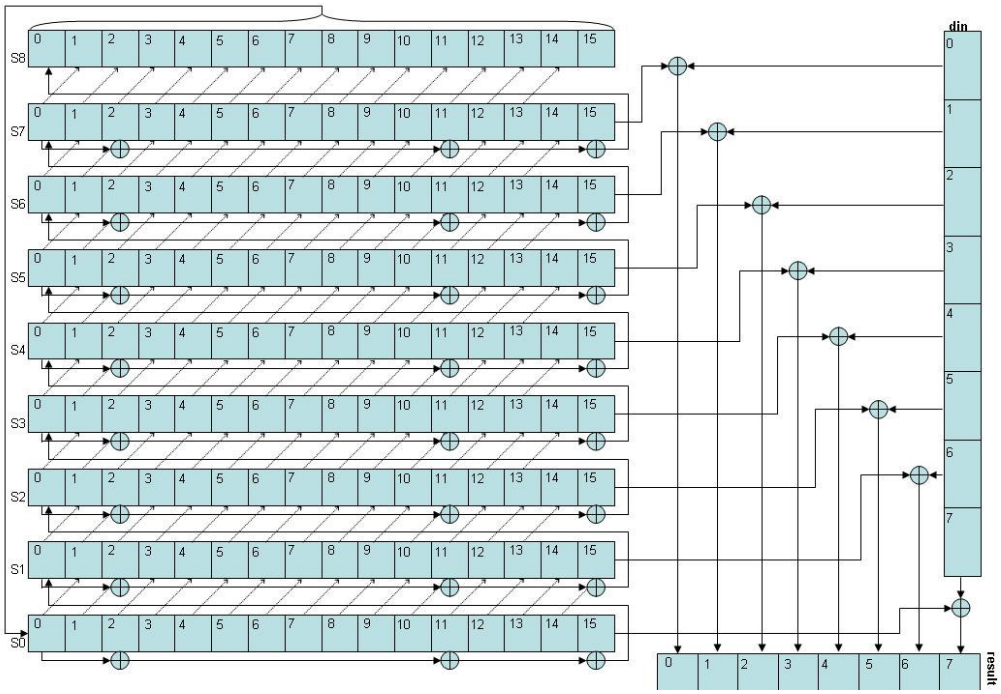

Figura 27 – Arquitetura de embaralhamento utilizando LFSR replicada e interligada.

Contudo, devido ao caminho crítico esta arquitetura não conseguiu cumprir o requisito de *timing* do projeto. A Tabela 3 apresenta uma comparação do desempenho da arquitetura de embaralhamento utilizando LFSR entre diferentes tamanhos de palavras. O FPGA utilizado para as medições foi um Virtex-II com *speed grade* -5.

Tabela 3 – Desempenho do embaralhador para palavras de tamanhos diferentes [SAW02].

Data Path	Flip-Flops	Look-Up Tables	Clock MHz	Bandwidth
8-bit SONET	16	15	>400	>3.2 Gb/s
16-bit SONET	24	25	>360	>5.7 Gb/s
32-bit SONET	40	50	>360	>11.5 Gb/s
64-bit SONET	72	87	>320	>20 Gb/s
128-bit SONET	256	261	>320	>40 Gb/s
256-bit SONET	384	391	>320	>80 Gb/s
512-bit SONET	640	653	>320	>160 Gb/s
8-bit OTN	25	35	>350	>2.8 Gb/s
16-bit OTN	34	56	>175	>2.8 Gb/s
32-bit OTN	55	124	>85	>2.7 Gb/s

É possível observar que no caso do padrão OTN, para uma palavra de 32 *bits* a máxima freqüência de relógio é 85 MHz. Poder-se-ia concluir que para 64 *bits* este valor ficaria em aproximadamente 40 MHz. Utilizando uma Virtex-4 esta freqüência provavelmente aumentará, mas mesmo assim não é suficiente para se chegar aos 200 MHz requeridos no projeto. Portanto, teve-se que elaborar uma nova arquitetura que tivesse um tempo de processamento menor.

5.2 Arquitetura Utilizando Memórias BRAM

Esta arquitetura foi desenvolvida tendo em vista um ganho no desempenho. Para isto, ao invés de se computar as seqüências do polinômio utilizando LFSR em hardware, o que realmente demanda certo processamento, foi utilizada uma abordagem mais simples em que todas as seqüências do polinômio embaralhador foram inicializadas em memórias BRAM (*Block RAM*). Dado que o polinômio gerador padrão do OTN é de grau 16, tem-se então $2^{16}-1$ *bits*, o que equivale a 65535 *bits* pseudo-aleatórios. Para se alocar 65535 *bits* foram utilizadas quatro BRAM de 1024 x 16 *bits*. A Figura 28 mostra a arquitetura desenvolvida, onde o sinal entrada é a palavra de 64 *bits* proveniente do deslocador e será submetida à operação lógica "XOR" com a palavra de 64 *bits* proveniente das memórias BRAM, identificada na figura pelo sinal *s* também de 64 *bits*.

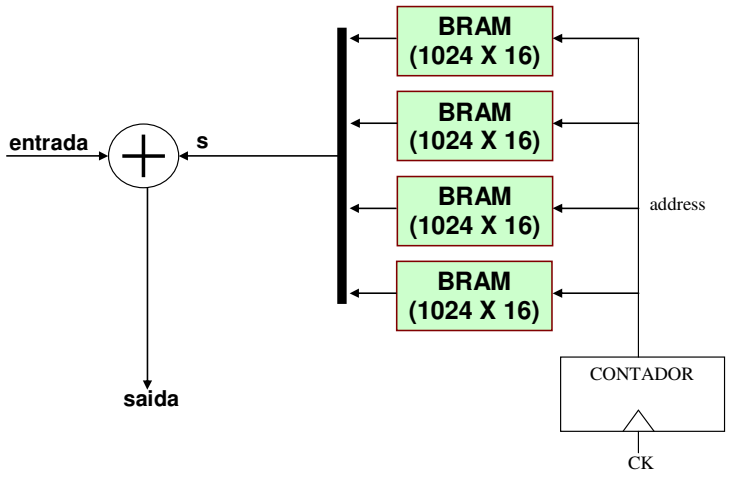

Figura 28 – Arquitetura de embaralhamento utilizando memórias BRAM.

As seqüência do polinômio contidas nas memórias foram organizados de forma a otimizar o acesso, portanto cada palavra de 64 *bits* foi dividida entre as quatro BRAM, sendo então 16 *bits* por memória. Desta forma é possível ter os 64 *bits* com leituras em paralelo.

Para realizar o endereçamento das memórias utilizou-se um contador. Este contador é parte do sub-módulo chamado gerador de endereço da memória. Este módulo tem também a tarefa de realizar a repetição da seqüência do polinômio quando necessária, pois como pôde ser visto anteriormente, o multi-quadro OTN tem 16320 *bytes*, ou seja, 130560 *bits*, o que não fecha com os 65535 *bits* gerados pelo polinômio. Portanto após 65535 *bits* a seqüência terá que ser repetida, completando quase duas seqüências em um multi-quadro. Ainda há um pequeno detalhe, por serem 65535 *bits* gerados pelo polinômio, fica faltando um último bit na última palavra de 64 *bits*. Esta ausência de um *bit* requer que seja executada uma operação de deslocamento de um *bit* entre a palavra em que este *bit* está faltando e a próxima palavra, que por sua vez é a primeira palavra a ser repetida. Para tornar esta operação possível foram utilizadas memórias com dupla porta, permitindo leituras de posições distintas de uma mesma memória em paralelo. Esta operação é feita em todos os multi-quadros.

5.3 Validação

Para a validação do sub-módulo embaralhador utilizou-se da seguinte estratégia: usar a própria seqüência do polinômio como entrada do sub-módulo embaralhador acrescida da seqüência de alinhamento. Assim, como visto na Figura 28, tem-se uma operação lógica "XOR" entre duas palavras que serão sempre iguais, resultando em uma saída sempre em zero com exceção da seqüência de alinhamento que não é embaralhada.

Utilizando esta estratégia conseguiu-se realizar a visualização e a validação das formas de ondas mais facilmente. A Figura 29 mostra o comportamento do módulo embaralhador em uma simulação com atraso utilizando a estratégia adotada.

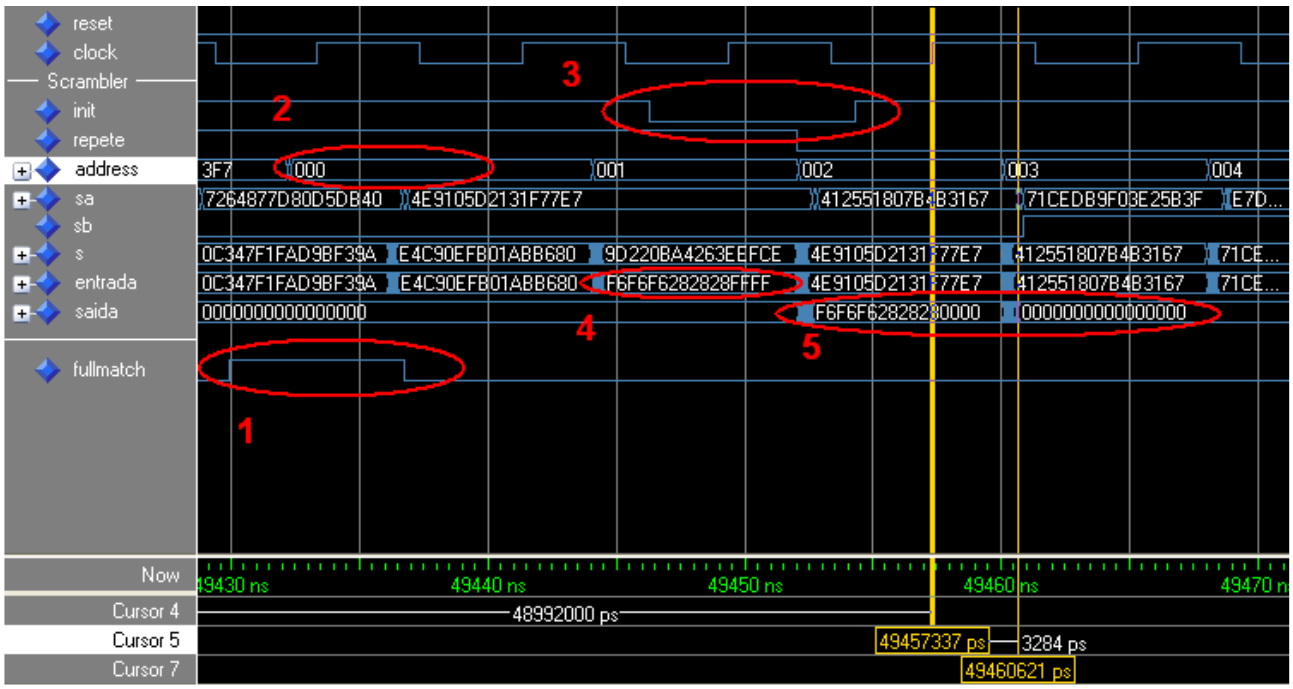

Figura 29 – Simulação do módulo embaralhador com atraso.

Após a identificação da seqüência e alinhamento, representada pelo sinal *fullmatch* proveniente do sub-módulo alinhador de quadro (Figura 29, índice 1), o gerador de endereço do embaralhador é resetado e começa a incrementar o seu valor (Figura 29, índice 2). O sinal *sa* começa a receber o conteúdo da memória, o sinal *sb* é o primeiro *bit* da próxima palavra na memória e somente é utilizado no momento de realizar o deslocamento devido à ausência de um *bit* mencionada anteriormente. O sinal *s* então é a composição do sinal *sa* e *sb*, de acordo a existência, ou não, do deslocamento. Um ciclo de relógio depois de identificada a seqüência de alinhamento, a mesma pode ser observada na entrada do sub-módulo embaralhador (Figura 29, índice 4) e o sinal *init*, que é responsável pela inicialização do embaralhamento, é acionado. A saída então começa a ser gerada, primeiramente com a seqüência de alinhamento, que não é embaralhada (Figura 29, índice 5). Um ciclo de relógio após a seqüência de alinhamento aparecer na saída percebe-se que o embaralhador começou a atuar, deixando a saída do sub-módulo sempre em zero. É possível observar no ciclo anterior a saída zerada, o sinal *s* e o sinal *entrada* com o mesmo valor, no caso,

“4E9105D2131F77E7”.

Com isto é possível validar o sub-módulo embaralhador e por conseqüência o desembaralhador, que são na prática o mesmo. Para se obter o valor original sem embaralhamento basta colocar a saída gerada pelo embaralhamento como entrada no desembaralhamento.

Tendo então a sua funcionalidade validada, também é mostrado na Figura 29 o tempo de propagação combinacional do módulo Embaralhador, que é 3,284 ns, indicado entre dois cursores amarelos. Tempo que está de acordo com o requisito que se tem no projeto, que é de 5 ns.

5.4 Resultado de Área

Seguindo o mesmo procedimento apresentado no módulo alinhador de quadro. Temos os dados de utilização referentes ao módulo embaralhador apresentados na Tabela 4.

Tabela 4 – Utilização do FPGA pelo módulo embaralhador.

Resumo da utilização do dispositivo Xilinx Virtex4 XCV4FX100 (valores estimados)			
Utilização Lógica	**Utilizado**	**Disponível**	**Utilização**
Número de Slices	115	42176	0%
Número de Slice Flip Flops	180	84352	0%
Número de LUTs de 4 entradas	205	84352	0%
Número de RAMB16s	4	376	1%

Assim como ocorreu no módulo alinhador de quadro, o número de LUTs utilizadas representa muito pouco, neste caso foram utilizadas 205 LUTs, quantidade que é inferior a 1% da quantidade total disponível no dispositivo. Neste caso também foram utilizadas 4 memórias RAMB16, o que representa 1% da quantidade total de RAMB16 disponível.

6 Módulo Extrator de Cabeçalhos

Como visto anteriormente, o cabeçalho representa a área do quadro onde estão definidos os *bytes* de controle da comunicação OTN. As informações contidas no cabeçalho estão divididas em quatro quadros OTN, sendo 16 *bytes* por quadro, totalizando 64 *bytes* por multi-quadro OTN (Figura 9, página 9). Nestes *bytes* existem campos relacionados a informações de identificação da origem e do destino, identificação do operador, identificação do tipo de dado a ser ou sendo transportado bem como identificação de falhas e alarmes variados. Estas informações precisam ser interpretadas e processadas, o que será feito pelo software embarcado presente na plataforma de prototipação externa ao contexto deste trabalho a qual este projeto será integrado.

A extração dos cabeçalhos consiste em identificar em quais palavras do quadro cada tipo de cabeçalho se encontra e armazená-los em memórias BRAMs. Os cabeçalhos serão armazenados em três memórias lógicas distintas, referente aos três diferentes tipos de cabeçalhos (ODU, OTU e OPU). Nestas memórias estarão armazenados os dados referentes aos últimos 256 multi-quadros.

6.1 Arquitetura

A arquitetura responsável pela extração dos cabeçalhos é composta por sete BRAMs de 2048x1 *bytes*. Estas memórias possuem duas portas de 8 *bits* disponíveis para escrita e leitura dos dados. Os cabeçalhos foram organizados nas memórias de forma a otimizar a escrita em paralelo dos dados, tornando possível a extração dos cabeçalhos contidos nas palavras de entrada em no máximo três ciclos de relógios. A Figura 30 ilustra as sete memórias com seus cabeçalhos associados.

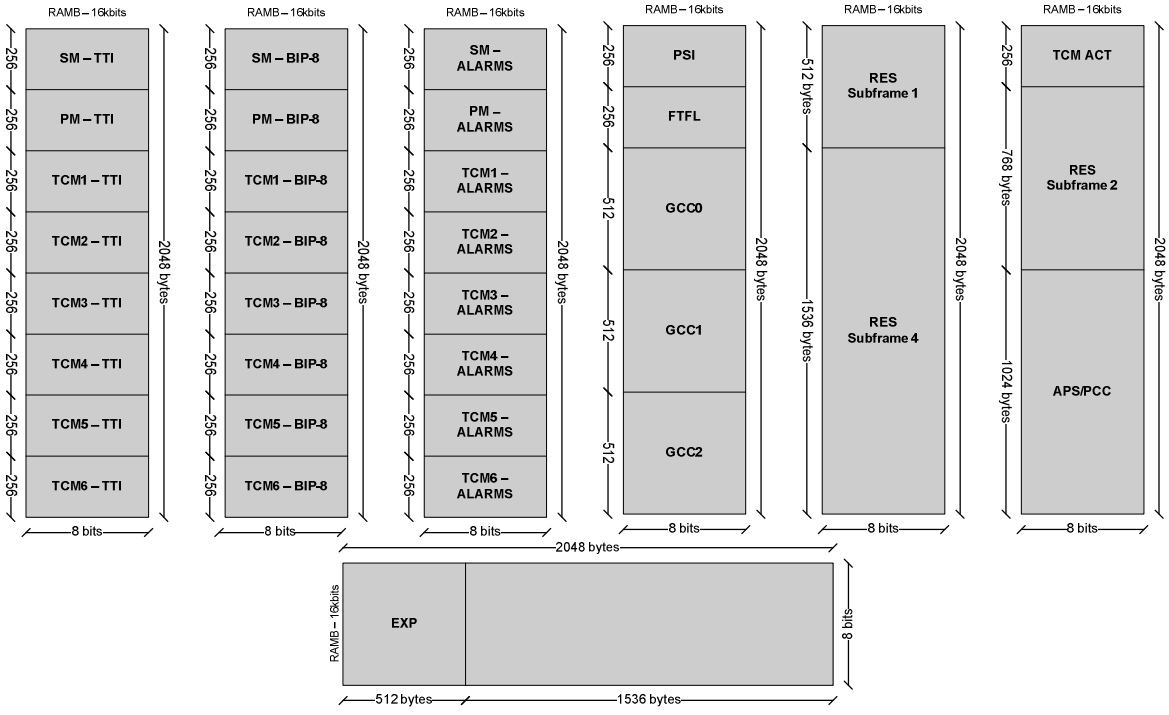

Figura 30 – Organização dos cabeçalhos nas memórias.

Para que seja possível realizar a extração dos cabeçalhos devemos saber em quais bytes eles se encontram. A partir das informações apresentadas nas Figura 5 e Figura 9 (páginas 7 e 9) obtiveram-se os dados apresentados na Tabela 5. Esta tabela indica em qual palavra, *byte* e quadro estão localizados os dados referentes aos cabeçalhos em um multi-quadro. Além de indicar quantos ciclos de relógio são necessários para realizar o armazenamento nas memórias.

Tabela 5 – Localização dos cabeçalhos no multi-quadro

Palavra	Bytes	Parte do Cabeçalho	Quadro	Ciclos de Relógio
1	0 a 7	OH1	1	1
2	8 a 15	OH2	1	
511	4080 a 4087	OH1	2	2
512	4088 a 4095	OH2	2	
1021	8160 a 8167	OH1	3	2
1022	8168 a 8175	OH2	3	
1531	12240 a 12247	OH1	4	3
1532	12248 a 12255	OH2	4	

Como visto anteriormente, cada quadro OTN contém 16 *bytes* de cabeçalhos. O campo indicado como Parte do Cabeçalho indica se esta palavra esta localizada nos primeiros 8 *bytes* (OH1) ou nos 8 *bytes* seguintes (OH2).

Lembrando, cada quadro possui 255 colunas por 16 linhas. A cada ciclo de relógio recebe-se simultaneamente 8 *bytes*, ou seja, meia coluna. Assim, o tempo total para a recepção de um quadro inteiro é de 510 ciclos de relógio. A cada quadro tem-se então 508 ciclos de relógio para que a extração seja realizada. Estes ciclos correspondem ao tempo que o *transponder* fica recebendo a carga útil e o FEC. Na arquitetura desenvolvida utiliza-se no pior caso 3 ciclos por quadro para a extração.

A partir destas informações é possível extrair todos os *bytes* do cabeçalho, no entanto precisamos extrair cada campo separadamente, uma vez que temos campos de uma mesma palavra que são armazenados em memórias distintas. Esta operação é realizada de acordo com a separação dos campos do cabeçalho apresentada na Figura 9 e o armazenamento nas memórias é feito de acordo com a Figura 30. Com isto tem-se um sinal para cada campo do cabeçalho contendo o seu endereço atual na memória, uma vez que são armazenados os últimos 256 valores. Campos do cabeçalho em que suas informações são compostas a partir de mais de um multi-quadro necessitam de um sinal que avise quando estas informações estão prontas para serem lidas, como é o caso dos campos FTFL e TTI.

Esta arquitetura encontra-se codificada quase que em sua totalidade, faltando pequenos detalhes. Porém, devido às limitações de tempo intrínsecas a um Trabalho de Conclusão de curso não foi possível realizar a validação do módulo Extrator de Cabeçalhos até o momento da entrega do presente documento.

7 Integração e Prototipação

Após a validação individual de cada um dos módulos que compõe o projeto, realizou-se a integração dos mesmos, o que inclui também a máquina de estados que controla o sincronismo dos multi-quadros. A Figura 31 ilustra o diagrama de blocos da arquitetura integrada. Lembrando que o sub-módulo deslocador possui três estágios de *pipeline* internos.

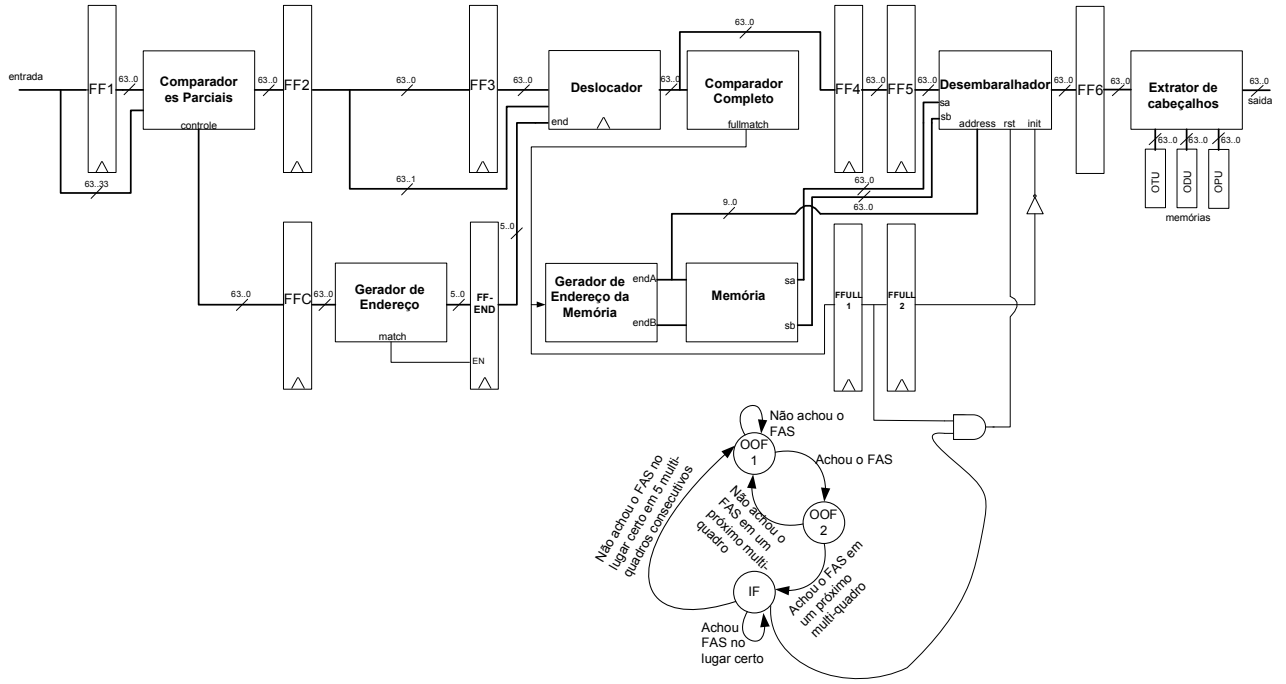

Figura 31 – Diagrama de blocos da arquitetura integrada.

A máquina de estados ilustrada na Figura 31 é responsável pelo sincronismo dos multi-quadros. Ela foi elaborada de acordo com sua especificação em [ITU04]. O estagio inicial é o *out-of-frame*, indicado na figura por OOF 1. Neste estado, não haverá transição enquanto não for encontrada a seqüência de alinhamento, referida na figura como FAS. Uma vez encontrada a seqüência de alinhamento entra-se no estado OOF 2. Este estado deve confirmar a presença da seqüência de alinhamento no próximo multi-quadro, entrando então no estado *inframe,* referido na figura como IF. Caso a seqüência de alinhamento não seja confirmada em OOF 2, o próximo estado volta a ser o OOF 1.

Ao se entrar no estado IF, assume-se que os multi-quadros estão sincronizados. A máquina de estados continuará no estado IF enquanto não encontrar cinco multi-quadros consecutivos sem a seqüência de alinhamento. Neste caso o próximo estado será o OOF 1, caracterizando a perda do sincronismo.

7.1 Estrutura de Teste

Para realizar a validação dos módulos integrados foi desenvolvido um programa em C++ capaz de gerar multi-quadros para serem inseridos como entrada dos módulos. O programa gerador de quadros foi desenvolvido para facilitar a criação dos arquivos texto contendo os multi-quadros utilizados nas simulações feitas da arquitetura, visto que seria praticamente impossível a criação manual destes arquivos. Para utilizar a ferramenta é necessário fornecer algumas informações a respeito dos multi-quadros, como: quantidade de multi-quadros a serem gerados; tipo de sinal de teste (nulo ou pseudo-aleatório); gerar o multi-quadro com ou sem embaralhamento e a quantidade de bits de desalinhamento desejada.

A partir desta ferramenta podem-se testar todas as funcionalidades da arquitetura gerando multi-quadros de forma automatizada. A Figura 32 mostra a interface da ferramenta geradora de multi-quadros.

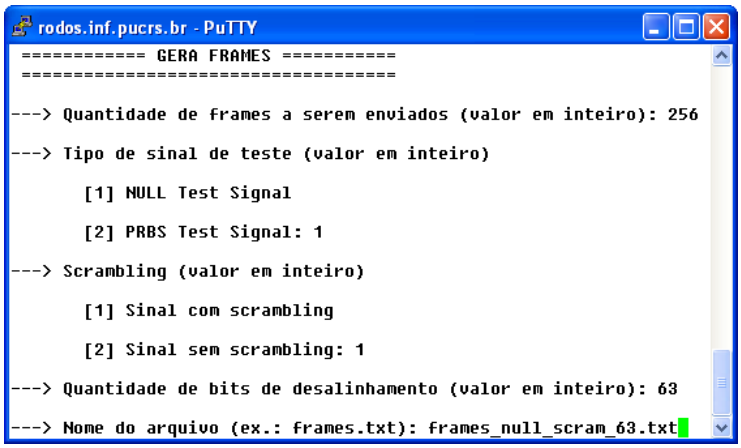

Figura 32 – Ferramenta geradora de multi-quadros.

Após ser facilitada a criação de estímulos para a arquitetura, queria-se facilitar a sua validação. Foi com esta finalidade que foi criado o módulo de teste chamado assinatura.

Este módulo tem como função calcular uma assinatura para todas as palavras de 64 *bits* que saírem da arquitetura. Assim como no embaralhador, a assinatura é formada por um LFSR, só que neste caso é um LFSR paralelo que utiliza o polinômio $1 + x^3 + x^4 + x^{64}$. A vantagem de se ter uma assinatura é que não é necessário analisar todas as saídas geradas. Com a assinatura basta analisar a última assinatura para se validar todas as saídas geradas pela arquitetura desde o último *reset*, porém esta abordagem só faz sentido se a assinatura for também gerada por *software* para se poder realizar a comparação. Esta funcionalidade foi incluída no programa gerador de multi-quadros.

De posse do gerador de multi-quadros e da assinatura foi criada a estrutura de teste apresentada na Figura 33.

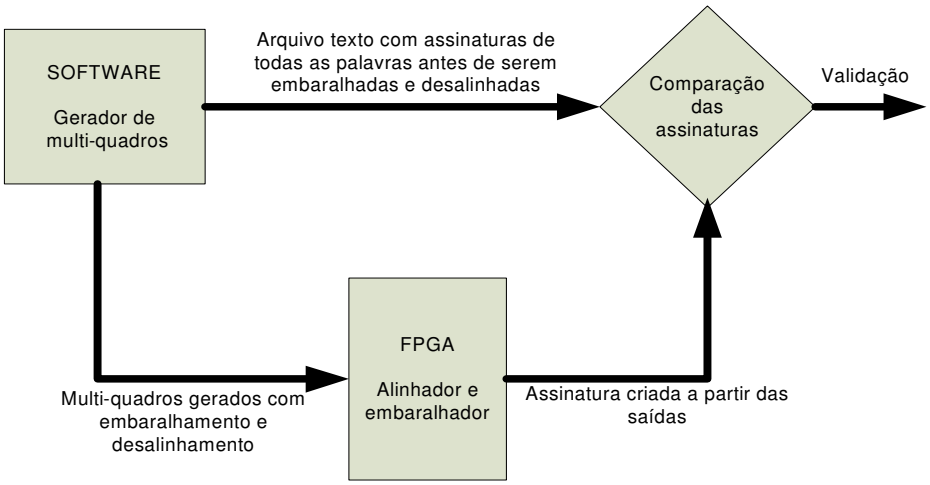

Figura 33 – Estrutura de teste da arquitetura.

7.2 Validação em Simulação com Atraso

Utilizando a estrutura de teste apresentada foi realizada a validação por simulação dos módulos integrados. Assim como foi realizado nos módulos separados, realizou-se simulação com atraso de portas lógicas e roteamento. O *software* gerador de multi-quadros gerou as assinaturas para um sinal nulo acrescido da seqüência de alinhamento e posteriormente foram gerados os multi-quadros com deslocamento embaralhamento que serão utilizados como estímulos de entrada na simulação. A Figura 34 ilustra as formas de ondas obtidas pela simulação com atraso.

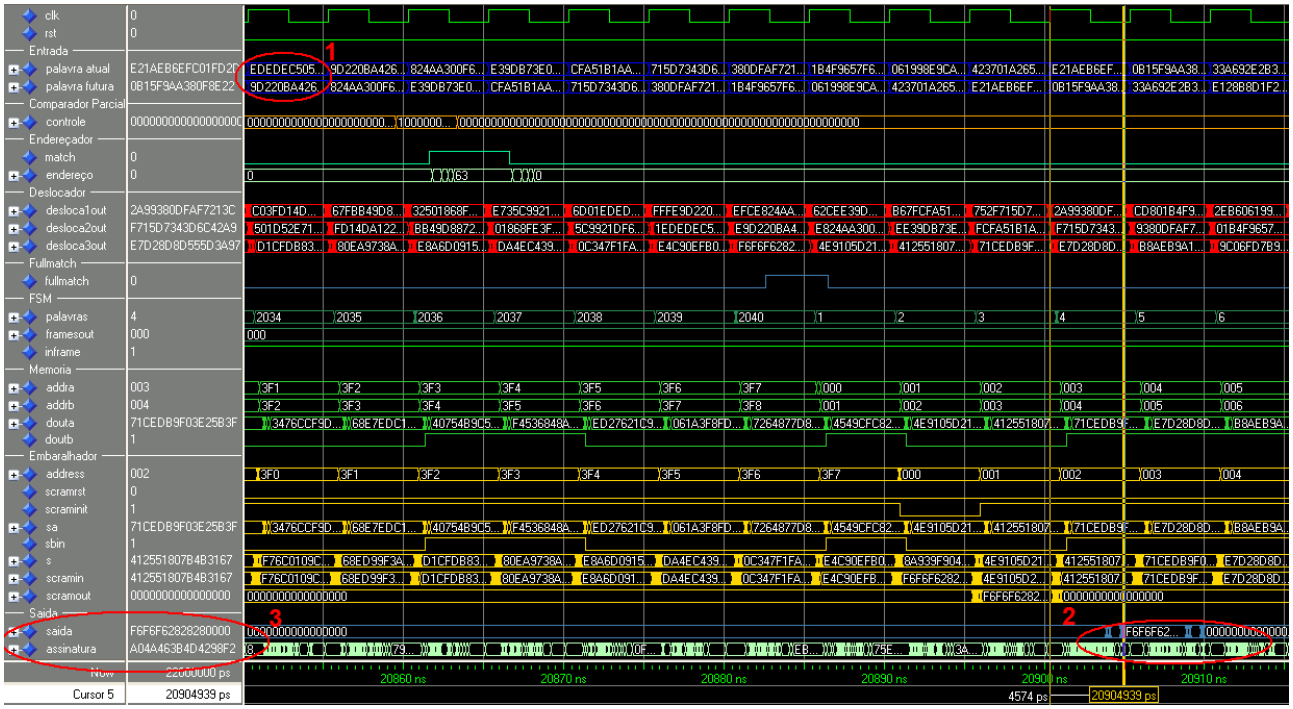

Figura 34 – Simulação da arquitetura integrada com atraso, no dispositivo XCV4FX100-10.

Assim como na validação do módulo embaralhador, a entrada é a própria seqüência do

polinômio embaralhador, uma vez que um sinal nulo foi embaralhado. A diferença é que agora a entrada, além de embaralhada, está com desalinhamento.

É possível observar na Figura 34 (índice número 1) o mesmo estímulo utilizado nas simulações anteriores, que é a seqüência de alinhamento deslocada em 63 *bits.* Após passar por alguns estágios de *pipeline* é possível observar no sinal *saída* a seqüência de alinhamento devidamente deslocada bem como a assinatura calculada (Figura 34, índices 2 e 3). O tempo de propagação combinacional da arquitetura com os módulos integrados foi de aproximadamente 4,5 *ns*.

7.3 Validação em Hardware

Após a validação em nível de simulação, foi realizada a validação em *hardware* dos módulos já integrados. Para tanto, utilizou-se o FPGA Xilinx Virtex4 XCV4FX100 com *speed grade* -10 rodando na freqüência de 200 MHz. A estrutura de teste para a validação em *hardware* se manteve a mesma, diferenciando somente no modo de injetar os estímulos de entrada. Ao invés de se ler os multi-quadros de um arquivo texto, os estímulos de entrada foram lidos de BRAMs inicializadas com o mesmo conteúdo.

Para a visualização dos sinais dos módulos prototipados, utilizou-se a ferramenta ChipScope Pro Analyzer de versão 8.1.02i da Xilinx Inc. (Figura 35).

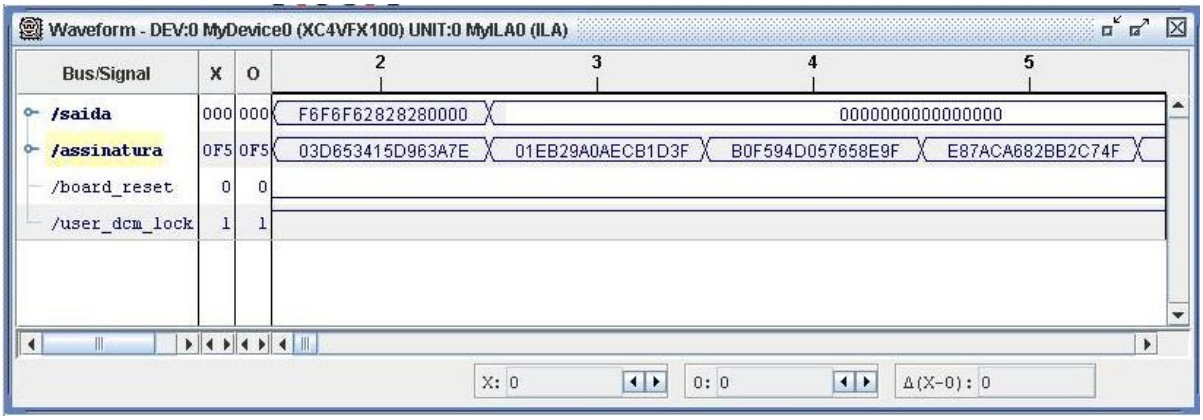

Figura 35 – Validação em hardware através do ChipScope.

No ChipScope foram conferidos os sinais saída e assinatura. A arquitetura foi validada de acordo com a estrutura de teste apresentada na Figura 33.

Futuramente será utilizado um analisador de protocolo, o qual é capaz de gerar quadros OTN utilizados de estímulos para a arquitetura desenvolvida bem como analisar a saída da mesma, verificando assim a correta implementação do padrão OTN.

7.4 Resultados

Os módulos integrados foram sintetizados, lógica e fisicamente, no software ISE versão 9.1i da Xilinx, fazendo uso da ferramenta de síntese XST. Um dos principais requisitos deste projeto é o desempenho. Por meio dos resultados obtidos nas validações realizadas pode-se afirmar que a arquitetura desenvolvida se mostrou muito eficiente, conseguindo operar na freqüência adotada como referência no início do projeto: 200 MHz.

Outro importante requisito deste projeto é a ocupação de área. Em FPGA a área é medida por utilização de LUTs. Tento em vista que o módulo responsável pela codificação e decodificação do FEC, sob responsabilidade de outro grupo, requer muita área, precisa-se utilizar o menor número possível de LUTs. A arquitetura desenvolvida se mostrou novamente eficiente ocupando 1222, ou 1%, de 84352 LUT's disponíveis na Virtex4 XCV4FX100.

7.5 Integração com o Módulo FEC

Um dos objetivos da atual parceria entre a TERACOM Telemática Ltda e o Grupo de Apoio ao Projeto de Hardware (GAPH) é a integração dos módulos codificador e decodificador FEC com os módulos desenvolvidos neste trabalho. Na Figura 36 pode-se observar a integração de todos os módulos do projeto entre a TERACOM e GAPH.

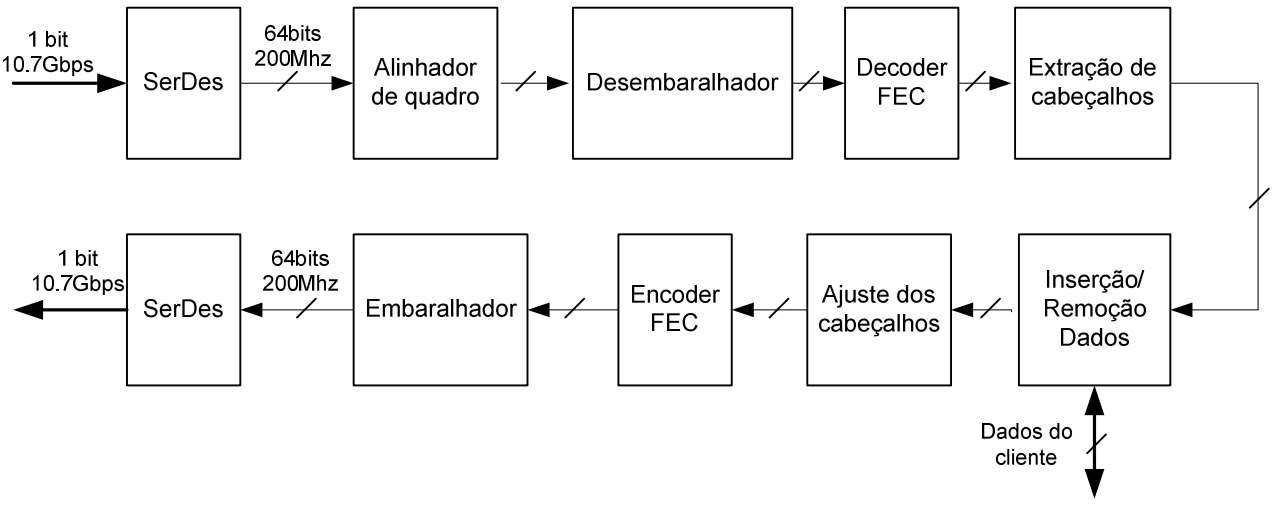

Figura 36 – Integração dos módulos do *transponder*.

O objetivo do módulo *SerDes* é receber o fluxo de dados provenientes da fibra óptica a 10.7 Gbps e disponibilizá-los em palavras de 64 bits a uma freqüência de 200 MHz. Este módulo também realiza a operação inversa, ou seja, transmite à fibra óptica os dados provenientes do módulo Embaralhador. Como visto anteriormente, o módulo Alinhador de quadro é responsável por identificar a ocorrência da seqüência de alinhamento e realizar o deslocamento, de modo que esta seqüência de alinhamento comece no início da palavra de 64 bits. Os módulos Embaralhador e

Desembaralhador são responsáveis por embaralhar e desembaralhar os dados, de forma que não ocorram longas seqüências de 0s ou de 1s, o que garante uma suficiente troca de estado dos bits facilitando a regeneração do relógio e evitando uma possível repetição da seqüência de alinhamento. O módulo Extração dos Cabeçalhos é utilizado para extrair o conteúdo dos cabeçalhos possibilitando a identificação de falhas, alarmes, tipo de dado transportado, origem e destino contidos nos quadros OTN G.709. O módulo Inserção/Remoção Dados é responsável pelo mapeamento e demapeamento do sinal do cliente dentro da carga útil do quadro OTN. O módulo Ajuste dos cabeçalhos é responsável pela montagem dos cabeçalhos dos quadros a serem transmitidos.

Os módulos Decoder e Encoder FEC correspondem a bancos de 16 codificadores e 16 decodificadores, pois um quadro possui o total de 16 sub-quadros com cálculos de FEC independentes.

Na Figura 37 encontra-se a arquitetura empregada nos módulos Decoder e Encoder FEC. Como comentado anteriormente, os dados são disponibilizados em palavras de 64 bits por ciclo da freqüência de 200 MHz. Como existem 16 cálculos de FEC independente, a freqüência de operação do codificador e decodificador foi reduzida para 100 MHz. Isto significa que duas palavras de 64 bits, ou uma coluna do quadro, são entregues aos módulos por ciclo da freqüência de 100 MHz. Dessa forma, é possível calcular o FEC dos 16 sub-quadros paralelamente e independentemente.

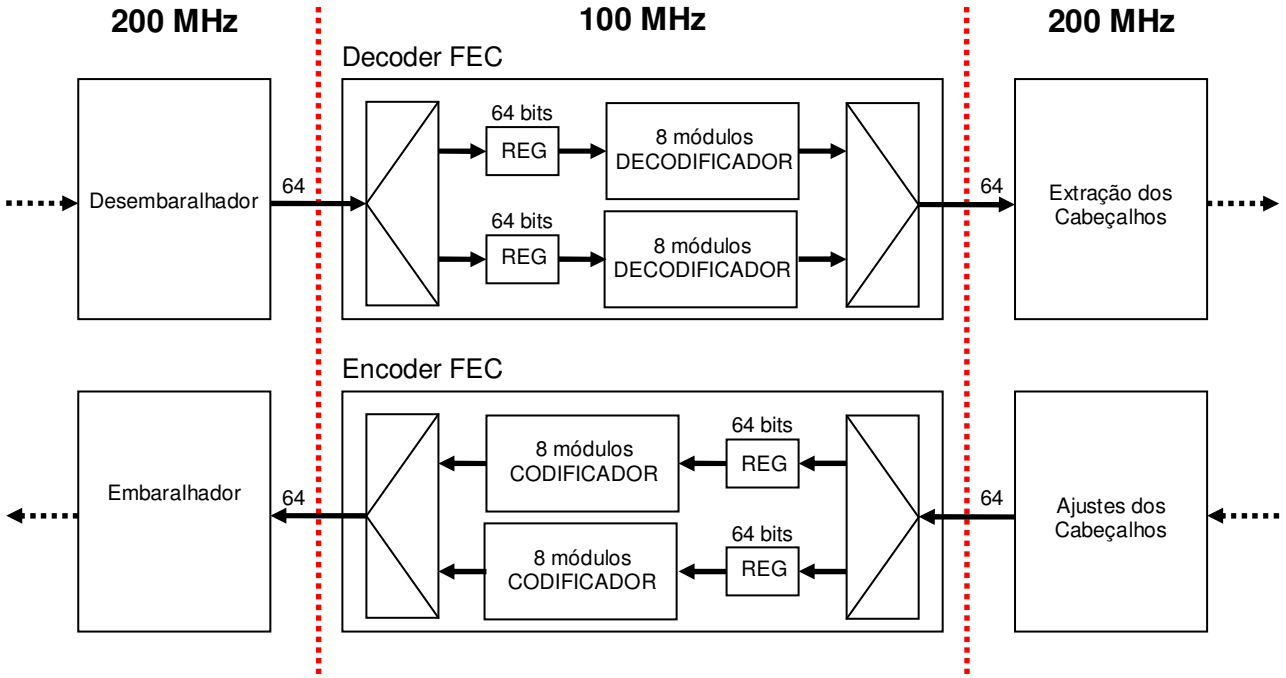

Figura 37 – Módulos Decoder e Encoder FEC integrados.

Esta tarefa de integração encontra-se em andamento e ainda não foi plenamente concluída. Podemos dizer que já temos toda a estrutura de integração montada e em fase de simulação funcional, porém ela ainda não foi plenamente testada, pois para esta tarefa seria necessária uma

ferramenta de geração de multi-quadros OTN completos, com a inclusão dos bytes do FEC, que também está em desenvolvimento. Para um teste completo do *transponder,* também seria necessário um domínio de transmissão de dados em fibra óptica no FPGA, que ainda não possuímos.

8 Conclusão

O presente Trabalho de Conclusão apresentou o estudo detalhado que foi realizado sobre a norma OTN G.709, o qual foi indispensável para o posterior desenvolvimento de módulos utilizados no projeto de um *transponder* OTN.

Este trabalho teve por característica a multidisciplinaridade, requerendo dos autores a aplicação de conhecimentos apreendidos durante todo o curso de Engenharia de Computação. Podem ser citadas as disciplinas de arquiteturas e organização de computadores, algoritmos de programação, laboratório de programação, teste e confiabilidade de sistemas e redes de computadores entre outras.

A arquitetura desenvolvida adotou como método de projeto o uso de lógica combinacional dividida por estágios de *pipeline*, permitindo a construção de um sistema com alto desempenho que atendesse às restrições de área e desempenho existentes no projeto. O trabalho envolveu o desenvolvimento de software, utilizado na criação de estímulos para os módulos e também na estrutura de teste, bem como o desenvolvimento de todo o hardware referente aos módulos Alinhador de Quadro, Embaralhador, Desembaralhador e Extrator de Cabeçalhos. Destacam-se como dificuldades encontradas ao longo do trabalho a elaboração, desenvolvimento e codificação das arquiteturas de forma a otimizar o desempenho dos módulos, uma vez que o requisito principal do projeto diz respeito ao tempo de propagação combinacional do sinal. Algumas arquiteturas, por não atenderem a este requisito, tiveram que ser reprojetadas ou adequadas.

Ao final deste trabalho podemos dizer que praticamente todos os nossos objetivos foram atingidos. Os módulos Alinhador de Quadro, Embaralhador e Desembaralhador foram desenvolvidos e estão funcionando corretamente em FPGA na freqüência adotada como freqüência de operação, necessária para a comunicação a 10.7 Gbps. Ficando pendente somente a validação do módulo Extrator de Cabeçalhos, o qual já possui sua arquitetura definida e a mesma está quase totalmente codificada.

Como trabalhos futuros, além de almejarmos realizar algumas otimizações no projeto, teremos a validação do módulo Extrator de Cabeçalhos bem como o desenvolvimento dos módulos Inserção/Remoção de Dados e Ajuste dos Cabeçalhos. Ainda teremos a validação completa dos módulos do *transponder* após a integração com os módulos FEC e o estudo das técnicas de transmissão e recepção de dados em fibra-óptica usando FPGAs. Acreditamos que através destas atividades será possível concluir o projeto X10GIGA.

Referências

[ALE06] Alexander, S. "OTN offers transparent service delivery". Capturado em: http://www.networkworld.com/news/tech/2006/020606-techupdate.html#graphic, Ago 2007.

[BRO92] Brown, S. et al. "Field-programmable gate arrays". Boston: Kluwer Academic, 1992, pp. 43.

[CAR01] Carro, L. "Projeto e Prototipação de Sistemas Digitais". Porto Alegre: Editora da UFRGS, 2001, 176p.

[CHO03] Choi, D. "Frame Alignment in a Digital Carrier - A Tutorial". IEEE Communications Magazine, Vol. 28, No 2. Fev 1990.

[GEN06] Gendron, R; Gidaro, A. "The G.709 Optical Transport Network - An Overview". Capturado em: http://documents.exfo.com/appnotes/anote153-ang.pdf, Nov 2007.

[ITU00] ITU-T. "Network node interface for the synchronous digital hierarchy (SDH)". ITU-T Recommendation G.707, Out 2000.

[ITU03] ITU-T. "Interfaces for the Optical Transport Network (OTN)". ITU-T Recommendation G.709/Y.1331, Mar 2003.

[ITU04] ITU-T. "Characteristics of optical transport network hierarchy equipment functional blocks". ITU-T Recommendation G.798, Jun 2004.

[ITU05] ITU-T. "Architecture for the Optical Transport Network". ITU-T Recommendation G.872, Set 2001.

[KAZ06] Kazi, K. "Optical Networking Standards: A Comprehensive Guide". New York: Springer, 2006, 109p.

[NAK05] Nakamura, R. "Novas Tecnologias de Comunicações Ópticas". Capturado em: http://www.rnp.br/_arquivo/sci/2005/nakamura-roberto_novas-tecnologias.pdf, Ago 2007.

[NET02] Neto, J. "DWDM – Dense Wavelength Division Multiplexing". Capturado em: http://www.gta.ufrj.br/grad/02_2/dwdm/, Nov 2007.

[SAW02] Sawyer, N. "SONET and OTN Scramblers/Descramblers". Xilinx, Nov 2002. Capturado em: www.xilinx.com/bvdocs/appnotes/xapp651.pdf, Nov 2007.

[VIS02] Vissers, M. "Optical Transport Network & Optical Transport Module". Capturado em: http://ties.itu.ch/ftp/public/itu-t/tsg15opticaltransport/OTN/g709-intro-v2.ppt, Ago 2007.

[WAL03] Walker, T. “Optical Transport Network (OTN) Tutorial”. Capturado em: http://www.itu.int/ITU-T/studygroups/com15/otn/OTNtutorial.pdf, Ago 2007.

[WHA04] Whatcott, R. “Timing Closure - 6.1i”. Capturado em: http://www.xilinx.com/xlnx/xweb/xil_tx_display.jsp?iLanguageID=1&category=&sGlobalNavPick=&sSecondaryNavPick=&multPartNum=1&sTechX_ID=rw_tim_closure_61i#statictiming, Ago 2007.

[WOL94] Wolf, W. “Modern VLSI Design: A Systems Approach”. Englewood Cliffs (NJ): PTR Prentice Hall, 1994, pp. 257-268.

Apêndice I – Código do Sub-módulo Comparador Parcial

Código VHDL do sub-módulo comparador parcial:

```
--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-- Comparadores parciais
--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_unsigned.all;

entity comparador is
port(   controle : out std_logic_vector(63 downto 0) ;
        atual    : in  std_logic_vector(63 downto 0) ;
        futuro   : in  std_logic_vector(30 downto 0)
      );
end comparador;

architecture comparador of comparador is

signal p0,p1,p2,p3,p4,p5: std_logic_vector(63 downto 0);
signal comp             : std_logic_vector(63 downto 0);
signal compara          : std_logic_vector(63 downto 0);
signal atualefuturo     : std_logic_vector(94 downto 0);

begin

atualefuturo <= atual & futuro(30 downto 0);
add:for i in 0 to 63 generate
begin
        p0(i) <=(atualefuturo(94-i) and
                 atualefuturo(93-i) and
                 atualefuturo(92-i) and -- F
                 atualefuturo(91-i));
        p1(i) <=((not atualefuturo(90-i)) and
                 atualefuturo(89-i) and
                 atualefuturo(88-i) and -- 6
                 (not atualefuturo(87-i)));
        p2(i) <=(atualefuturo(78-i) and
                 atualefuturo(77-i) and
                 atualefuturo(76-i) and -- F
                 atualefuturo(75-i));
        p3(i) <=((not atualefuturo(74-i)) and
                 atualefuturo(73-i) and
                 atualefuturo(72-i) and -- 6
                 (not atualefuturo(71-i)));
        p4(i) <=((not atualefuturo(70-i)) and
                 (not atualefuturo(69-i)) and
                 atualefuturo(68-i) and -- 2
                 (not atualefuturo(67-i)));
        p5(i) <=(atualefuturo(66-i) and
```

```
                (not atualefuturo(65-i)) and
                (not atualefuturo(64-i)) and -- 8
                (not atualefuturo(63-i)));

        comp(i) <= (p0(i) and p1(i) and p2(i) and p3(i));
        compara(i) <= (comp(i) and p4(i) and p5(i));

end generate;

controle <= compara;

end comparador;
```