UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES

MAPEAMENTOS DE PEQUENA ORDEM NA HIERARQUIA DIGITAL SÍNCRONA

LUÍS FLÁVIO COLLARES MACHADO
Orientador:Prof. Dr. : REGE ROMEU SCARABUCCI

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas - UNICAMP, como parte dos requisitos exigidos para a obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA

OUTUBRO - 1996

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

M18m

Machado, Luís Flávio Collares
Mapeamentos de pequena ordem na hierarquia digital síncrona / Luís Flávio Collares Machado.--Campinas, SP: [s.n.], 1996.

Orientador: Rege Romeu Scarabucci.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Mapeamento digital. 2. Sincronização. 3. Telecomunicações. I. Scarabucci, Rege Romeu. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES

MAPEAMENTOS DE PEQUENA ORDEM NA HIERARQUIA DIGITAL SÍNCRONA

LUÍS FLÁVIO COLLARES MACHADO
Orientador:Prof. Dr. : REGE ROMEU SCARABUCCI

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas - UNICAMP, como parte dos requisitos exigidos para a obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA

OUTUBRO - 1996

AGRADECIMENTOS:

Agradeço em primeiro lugar a CAPES que possibilitou a execução deste trabalho.

Agradeço ao Prof. Rege pelo apoio no esclarecimento das dúvidas e pela dedicação.

Ao LCAEE - Laboratório de Computação Aplicada a Engenharia Elétrica.

Aos colegas da FEE.

Aos meus pais que sempre me apoiaram.

RESUMO

A rede HDS(Hierarquia Digital Síncrona) já é uma realidade, hoje várias empresas estão implantando este sistema em todo o país. A HDS tornou-se um padrão mundial, não colocando na obsolescência as atuais redes PDH, pois estas podem ser interconectadas com as novas redes. Este novo padrão possibilita a interligação entre sistemas de 1,5 Mb/s e 2 Mb/s, não existindo mais fronteiras de incompatibilidade entre sistemas europeus e americanos. O gerênciamento mais abrangente, o uso da fibra óptica como meio de transmissão, uma estrutura mais inteligente de mapeamento de sinais e a possibilidade de cross-conexão consolidaram a HDS . Neste trabalho foram desenvolvidas três linhas de pesquisa: Um primeiro estudo desenvolve uma formulação para mapeamento e desmapeamento de canais de 64 kb/s nos containers VC-12 e VC-11. A seguir, blocos funcionais são projetados utilizando a linguagem VHDL[1]para mapeamento e desmapeamento de canais de 2Mb/s. Finalmente muitas simulações são realizadas com blocos funcionais anteriormente projetados, utilizando agora VHDL a nível estrutural, visando principalmente a observação do fenômeno de escorregamento de sinal(slips).

[1]VhsicHDL-Very High Speed Integrated Circuit Hardware Description Language

Glossário

- **AU** - Unidade Administrativa *(Administrative Unit).*
- **AUG** - Grupo de Unidades Administrativas *(Administrative Unit Group).*
- **C** - Container.
- **CI** - Indicador de Concatenação *(Concatanation Indicator).*
- **HPA** - Adaptação das Rotas de Alta Ordem *(Higher order Path Adaptation).*
- **HDS** - Hierarquia Digital Síncrona *(Synchronous Digital Hierarchy).*
- **ITU-TSS** - *ITU-Telecommunication Standardization Sector.*
- **LPA** - Adaptação das Rotas de Baixa Ordem *(Lower Order Path Adaptation).*
- **MSA** - Adaptação de Seção Multiplex *(Multiplex Section Adaptation).*
- **MSOH** - Supervisão de Seção Multiplex *(Multiplex Section OverHead).*
- **NDF** - Sinalização de Novos Dados *(New Data Flag).*
- **NPI** - Indicador de Ponteiro Nulo *(Null Pointer Indicator).*
- **PDH** - Hierarquia Digital Plesiócrona *(Plesiochronous Digital Hierarchy).*
- **PTR** - Ponteiro *(Pointer).*
- **POH** - Supervisão de Rota *(Path OverHead).*
- **RSOH** - Supervisão de Seção Regeneradora *(Regeneration Section OverHead).*
- **SOH** - Supervisão de Seção *(Section OverHead).*
- **STM** - Modulo de Transporte Síncrono *(Synchronous Transport Module).*
- **SONET** - Rede Ótica Padrão *(Standart Optical Network).*
- **TU** - Unidade Afluente *(Tributary Unit).*
- **TUG** - Grupo de Unidade Tributária *(Tributary Unit - Group).*
- **TS0**- Time Slot Zero.

- **VHDL - VhsicHDL** - *Very high speed integrated circuit Hardware Description Language.*
- **VC** - Container Virtual *(Virtual Container)*

Conteúdo

Página

Lista de Figuras

Página

Capítulo 1

Introdução

Com a definição da HDS (SDH - Synchronous Digital Hierarchy) iniciou-se um novo estágio na evolução das redes de telecomunicações no mundo. A HDS está propiciando uma revolução nos serviços de telecomunicações que vai desde os usuários finais, passando pelas operadoras e chegando aos fabricantes de equipamentos. Esta nova tecnologia representa um avanço que pode ser comparado com a introdução do PCM e da fibra ótica ha algumas décadas.

Os usuários dos serviços de telecomunicações ficam a cada dia mais exigentes, ocorrendo um aumento na demanda por serviços mais sofisticados. Serviços estes como: teleconferência, acesso remoto a base de dados, transferência de arquivos de multimídia que requerem uma maior flexibildade da rede e uma largura de banda virtualmente ilimitada.

A atual HDP *(Hierarquia Digital Plesiócrona)* foi desenvolvida em função de um plano de telefonia, não sendo compatível para um eficiente gerenciamento e em conexões de alta largura de banda. Usando essencialmente a mesma fibra, a rede HDS consegue operar com grandes larguras de banda e ainda reduz a quantidade de equipamentos. Um sofisticado sistema de gerenciamento introduz, mais significativamente, maior flexibilidade dentro da rede.

O sistema plesiócrono pode ser incorporado pela rede HDS devido a sua capacidade de interconexão. A HDS definiu uma estrutura que mapeia os tributários plesiócronos para dentro da rede síncrona . Portanto, os atuais equipamentos da HDP não ficarão obsoletos.

A operadora da rede terá uma redução nos custos associado à redução na quantidade de hardware da rede, uma eficiência crescente e uma redução no tempo de manutenção e operação. Um outro resultado que aumenta a confiança nesta nova hierarquia é o fato de não necessitar de um equipamento sobressalente.

O sofisticado gerenciamento capacita a rede síncrona que poderá dar uma vasta melhoria no controle das redes de transmissão, serviços de provisionamento mais rápido e avaliação dos resultados devido a uma melhor capacidade de reconfiguração e melhoria na restauração da rede. Esta nova rede tem ainda a capacidade de interconectar taxas diferentes de comunicação, EUA(SONET - 1,5 Mb/s) e EUROPA(SDH - 2 Mb/s). Suporta ainda serviços de MAN(Metropolitan Area Network), RDSI Faixa Larga e REDES PERSONALIZADAS de COMUNICAÇÃO.

Este trabalho está dividido em 5 capítulos, onde tem-se:
No capítulo 2: Um resumo da recomendação G.70X, que reune as recomendações G.707,G.708 e G.709. Fez-se um resumo, pois somente alguns ìtens serão importantes para os desenvolvimentos realizados nos outros capítulos. Apresenta-se o resumo de um trabalho já desenvolvido, que serviu de base para os desenvolvimentos feitos.
No capítulo 3: Desenvolve-se uma formulação matemática para a retirada direta dos canais, em um quadro síncrono, sem a necessidade da utilização de demultiplexadores.
No capítulo 4: Descreve-se um método de desmapeamento síncrono e assíncrono, baseado no mapeamento de sinais síncronos e assíncronos em modo flutuante de 2 Mb/s, utilizando como ferramenta a linguagem VHDL. Procura-se ainda, analisar o fenômeno do SLIP que ocorreria no caso de perda do relógio de transmissão, ou seja, trabalhando em free-running.
No capítulo 5: Mostram-se duas redes para análise e simulação. Serão mostrados os resultados obtidos em quatro simulações .
No capítulo 6: Conclusões.

Capítulo 2

Introdução à Multiplexação Síncrona

2.1 Recomendação G.70X- ITU-TSS

A Recomendação G-70X [1] trata, dentre outras coisas, da Estrutura de Multiplexação Síncrona a ser seguida pelos equipamentos da Hierarquia Digital Síncrona. Estes equipamentos têm a capacidade de mapear sinais dos mais diversos. A figura 2.1 mostra a multiplexação dos sinais afluentes definidos pela Recomendação G-703, que são sinais da Hierarquia Digital Plesiócrona:

- 1.544 kbits/s
- 2.048 kbits/s
- 6.312 kbits/s
- 34.368 kbits/s
- 44.736 kbits/s
- 139.264 kbits/s

Para isto foram definidas três operações:

- Mapeamento
- Alinhamento
- Multiplexação

Cada quadro na figura 2.1 simboliza uma sequência de bytes que é transmitida/processada uma vez a cada 125 μs [1]. O quadro que é realmente transmitido pela rede digital síncrona é o STM-N(Synchronous Transport Module); os demais quadros são resultados de processamentos realizados nos equipamentos HDS. Estes processamentos se referem às três operações listadas acima.

Legenda para a figura 2.1:

- STM - Módulo de Transporte Síncrono ("Synchronous Transport Module")
- AUG - Grupo de Unidades Administrativas ("Administrative Unit Group")
- VC - Container Virtual ("Virtual Container")
- AU - Unidade Administrativa ("Administrative Unit")
- TU - Unidade Afluente ("Tributary Unit")
- C - Container

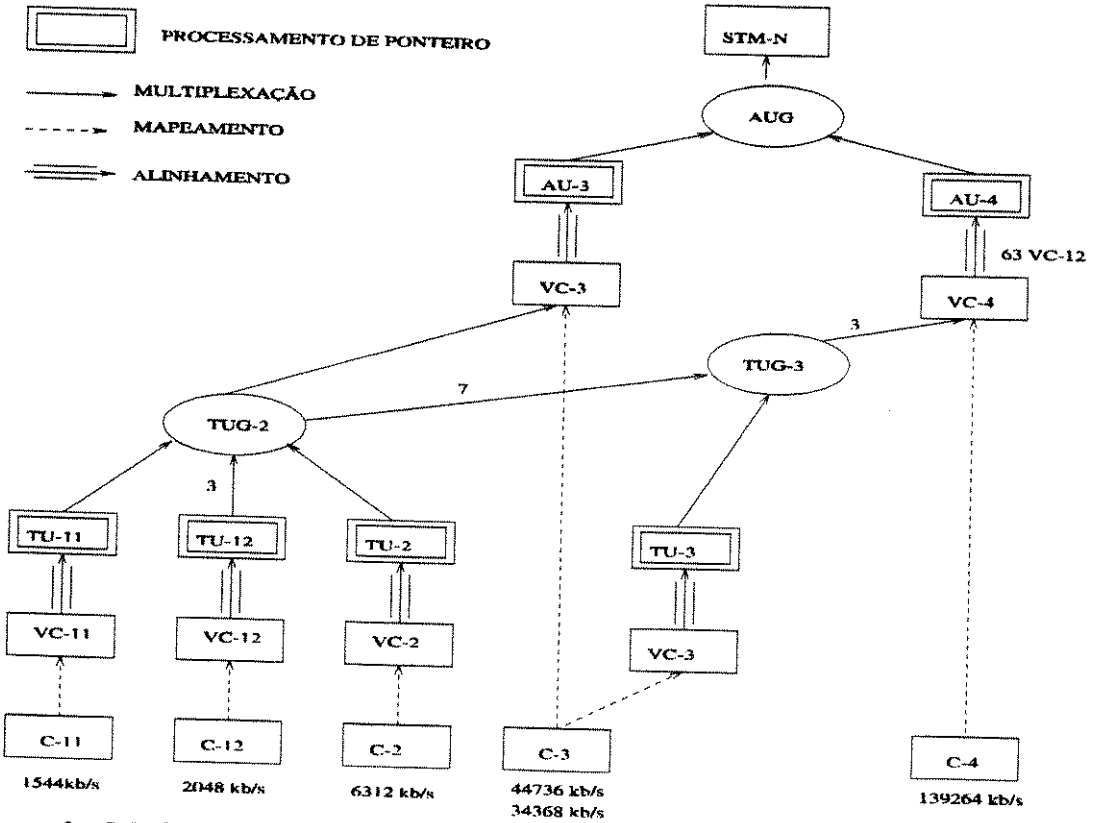


Figura 2.1 : Estrutura de Multiplexação HDS

[1]VC-1 e VC-2 são processados uma vez a cada 500μs

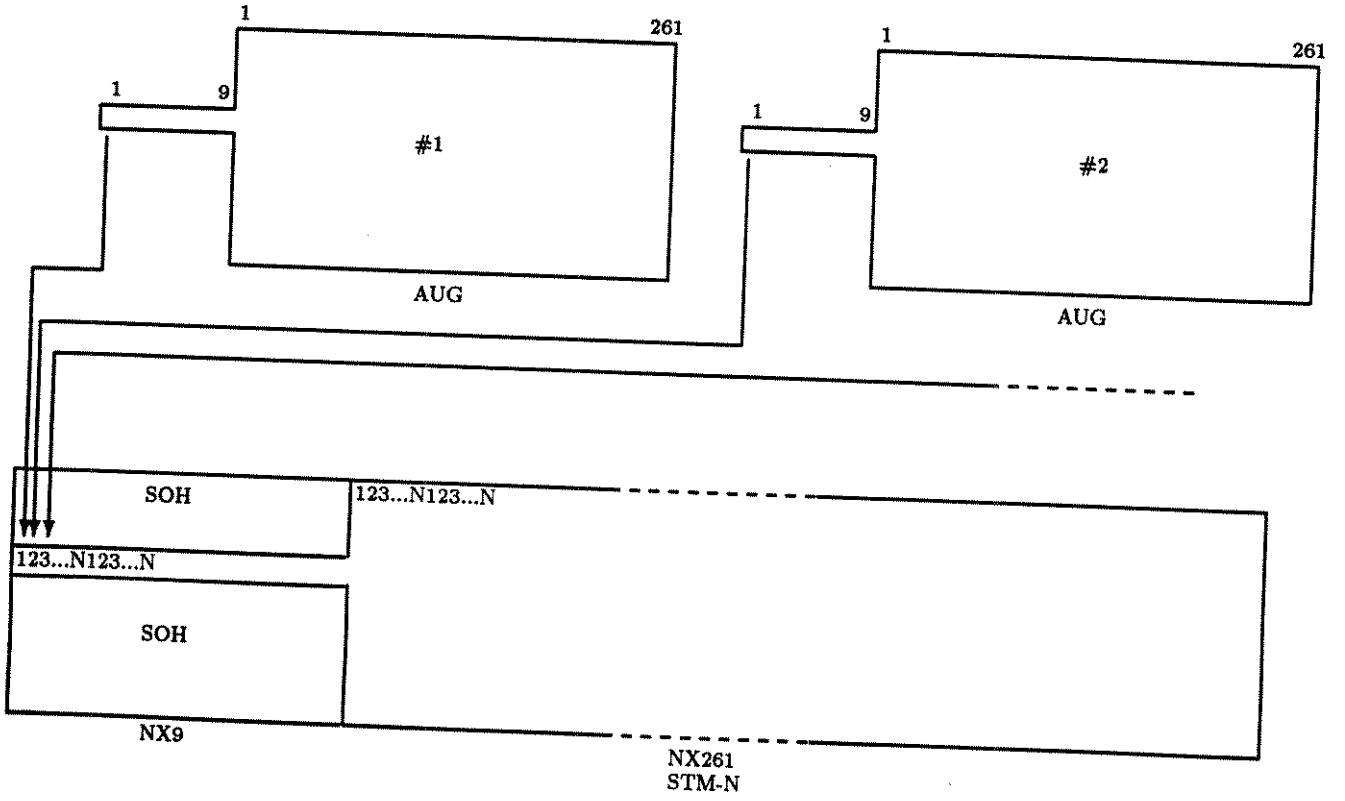
2.1.1 A Operação de Multiplexação

Multiplexação na Hierarquia Digital Síncrona é o processo pelo qual múltiplos quadros de menor ordem são adaptados para formar um quadro de maior ordem.

Multiplexação de AUGs dentro do STM-N

A multiplexação de N AUGs dentro de um STM-N é ilustrada na figura 2.2. O AUG é uma estrutura de 9 linhas por 261 colunas de bytes, mais 9 bytes colocados na quarta linha(ponteiro(s) do(s) AU(s)). O STM-N é composto de uma área RSOH (bytes de Supervisão de Seção Regeneradora), uma área de MSOH(bytes de Supervisão de Seção Multiplex) e áreas obtidas pela intercalação dos bytes de NxAUG's. As N AUGs são entrelaçadas byte a byte e têm fase fixa em relação ao STM-N.

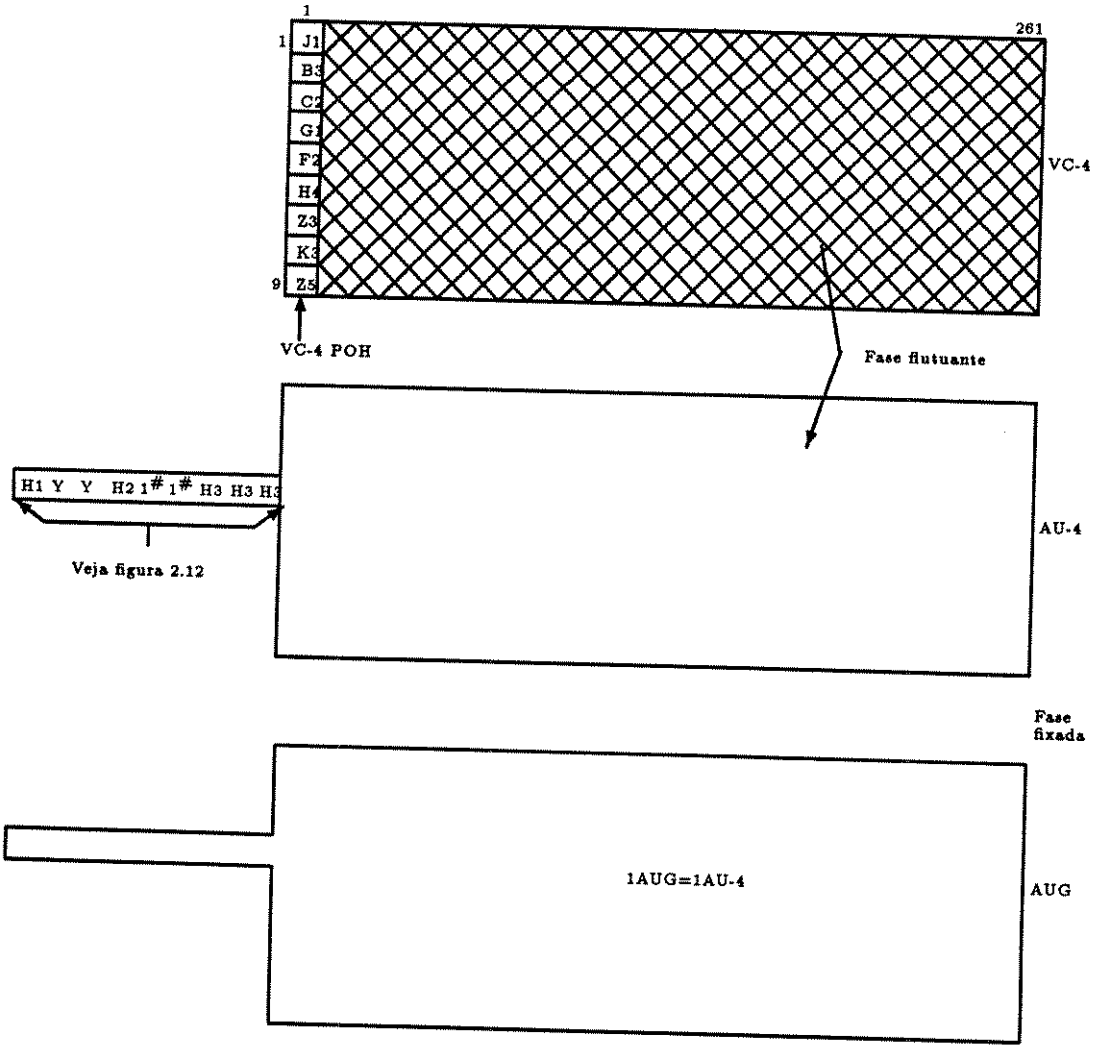
Figura 2.2 : Multiplexação de N AUGs dentro de um STM-N.



Multiplexação de AU-4s via AUG

A multiplexação de uma única AU-4 via AUG é ilustrada na figura 2.3. Os 9 bytes no começo da quarta linha são alocados para o ponteiro do AU-4. Os bytes restantes (9 linhas por 261 colunas) são alocados para o VC-4. A fase do VC-4 não é fixa em relação ao AU-4. A localização do primeiro byte do VC-4 em relação ao ponteiro do AU-4 é dada pelo valor do ponteiro. O AU-4 é colocado diretamente no AUG.

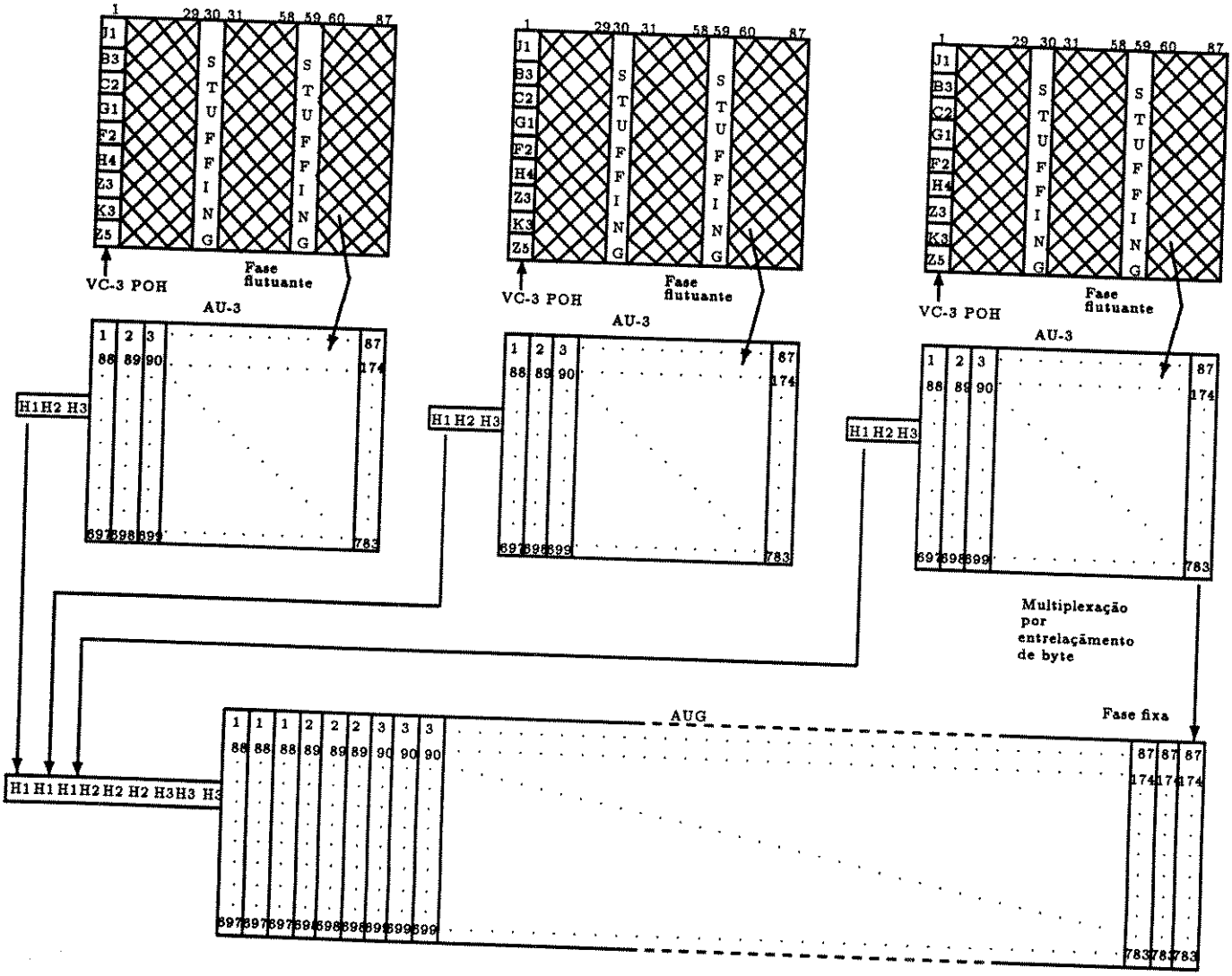
Figura 2.3: Multiplexação de AU-4s via AUG.



Multiplexação de AU-3s via AUG

A multiplexação de três AU-3 via AUG é ilustrada na figura 2.4. Os 3 bytes no começo da quarta linha são alocados para o ponteiro do AU-3. Os bytes restantes (9 linhas por 87 colunas) são alocados para o VC-3 mais 2 colunas de enchimento ("stuffing"). A fase do VC-3 mais as 2 colunas de "stuffing" não é fixa em relação ao AU-3. A localização do primeiro byte do VC-3 em relação ao ponteiro do AU-3 é dada pelo valor do ponteiro. As 3 AU-3 são entrelaçadas byte a byte para formar o AUG.

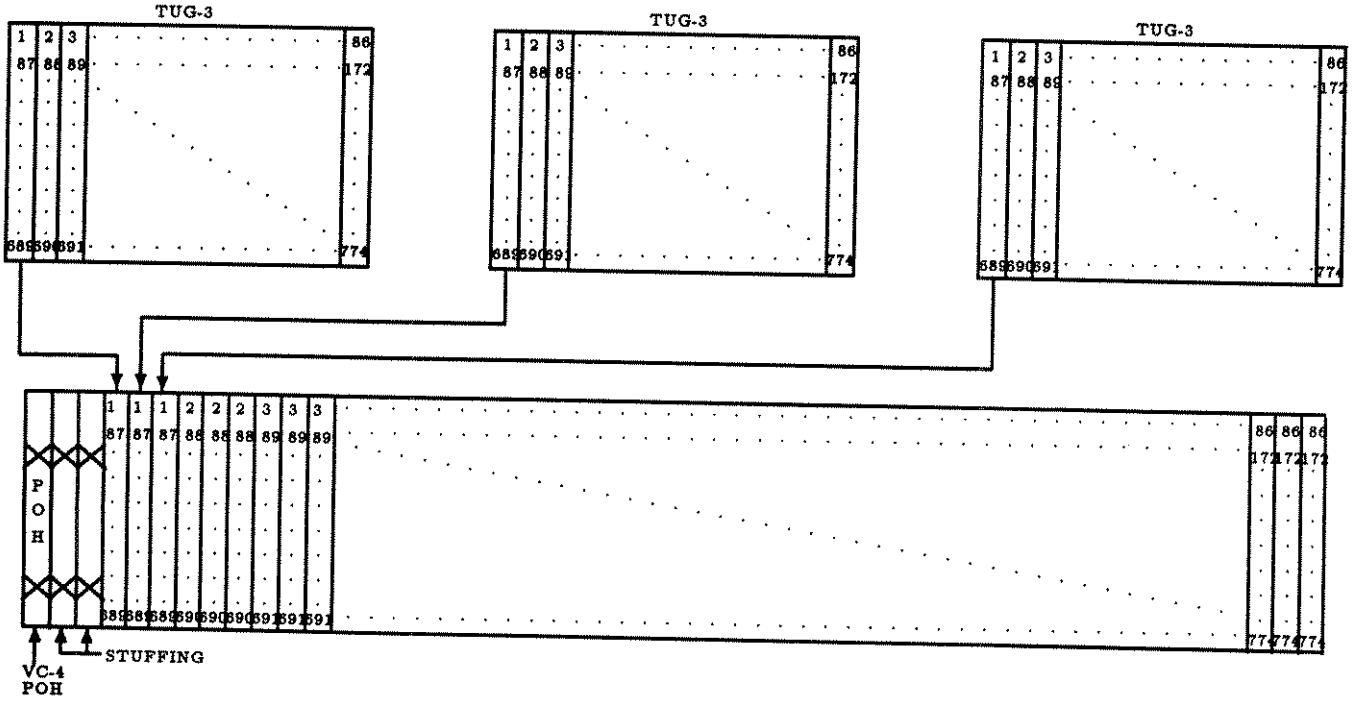
Figura 2.4 : Multiplexação de AU-3s via AUG.



Multiplexação de TUG-3s dentro do VC-4

A multiplexação de três TUG-3s dentro de um VC-4 é ilustrada na figura 2.5. O TUG-3 é uma estrutura de 9 linhas por 86 colunas de bytes. O VC-4 é composto por uma coluna de POH (Supervisão de Rota), duas colunas de "enchimento" e mais 258 colunas de carga útil. Os 3 TUG-3s, que têm fase fixa em relação ao VC-4, são entrelaçados byte a byte para formar a carga útil do VC-4. Como descrito anteriormente, a fase do VC-4 em relação ao AU-4 é dada pelo ponteiro do AU-4.

Figura 2.5 : Multiplexação de 3 TUG-3 dentro do VC-4.



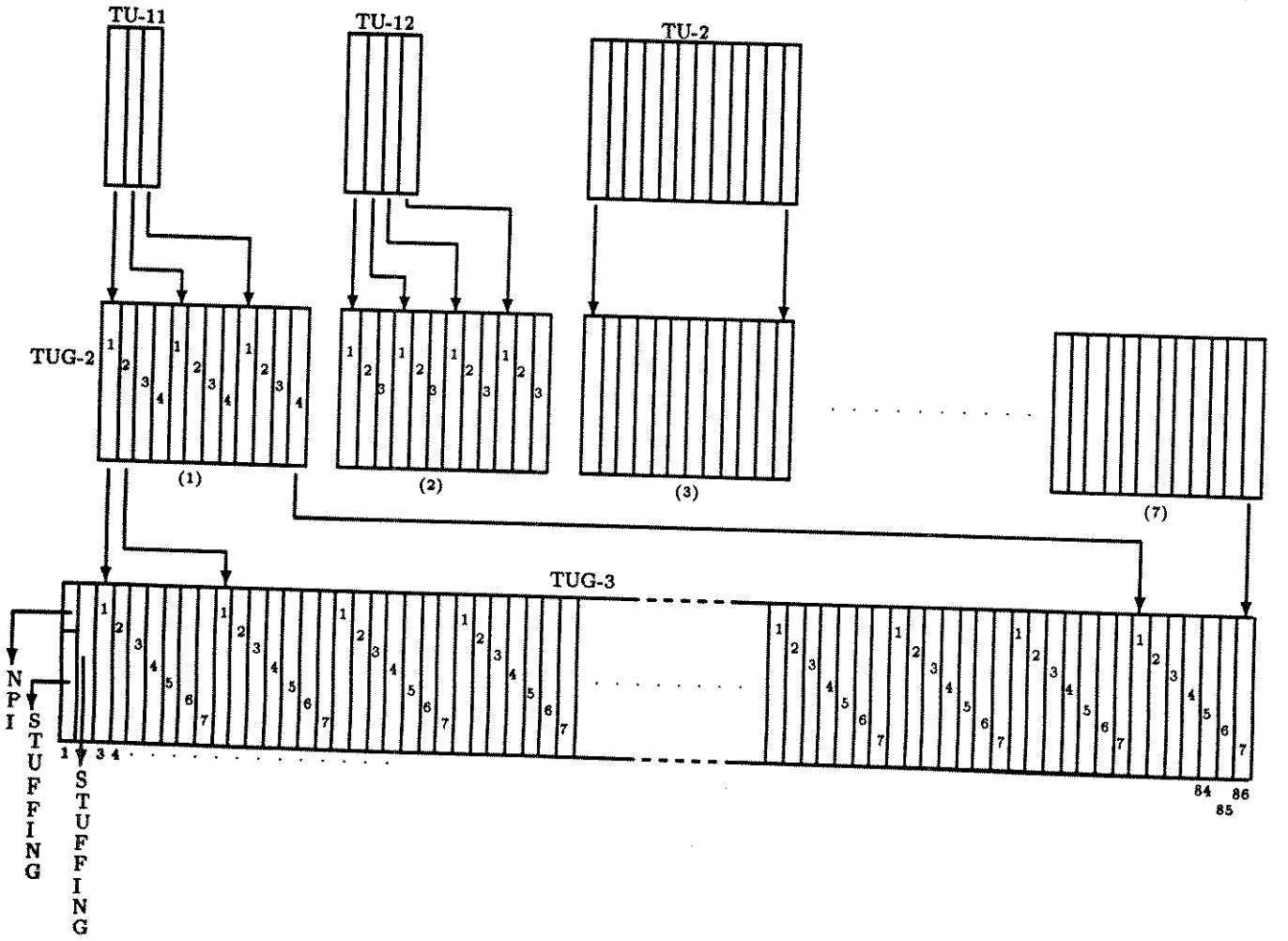
Multiplexação de TUG-2s via TUG-3

A multiplexação de 7 TUG-2s via TUG-3 é ilustrada na figura 2.6. Os 7 TUG-2s são entrelaçados byte a byte para formar 84 colunas de um TUG-3.

O TUG-3 é uma estrutura de 9 linhas por 86 colunas sendo que as duas primeiras colunas acomodam o seguinte:

- Um indicador de ponteiro nulo(NPI) contido nos dois primeiros bytes da primeira coluna. Este NPI pode ser usado para distinguir TUG-3s contendo TU-3s de TUG-3s contendo TUG-2s.
- A segunda coluna tem seus bytes preenchidos com "enchimento".

Figura 2.6 : Multiplexação de sete TUG-2s via um TUG-3.



Multiplexação de TUG-2s via VC-3

A multiplexação de 7 TUG-2s via TUG-3 é ilustrada na figura 2.7. Os 7 TUG-2s são entrelaçados byte a byte para formar 84 colunas de um VC-3. Um TUG-2 individual tem fase fixa em relação ao VC-3.

O VC-3 é composto por uma coluna de POH mais 84 colunas de carga útil.

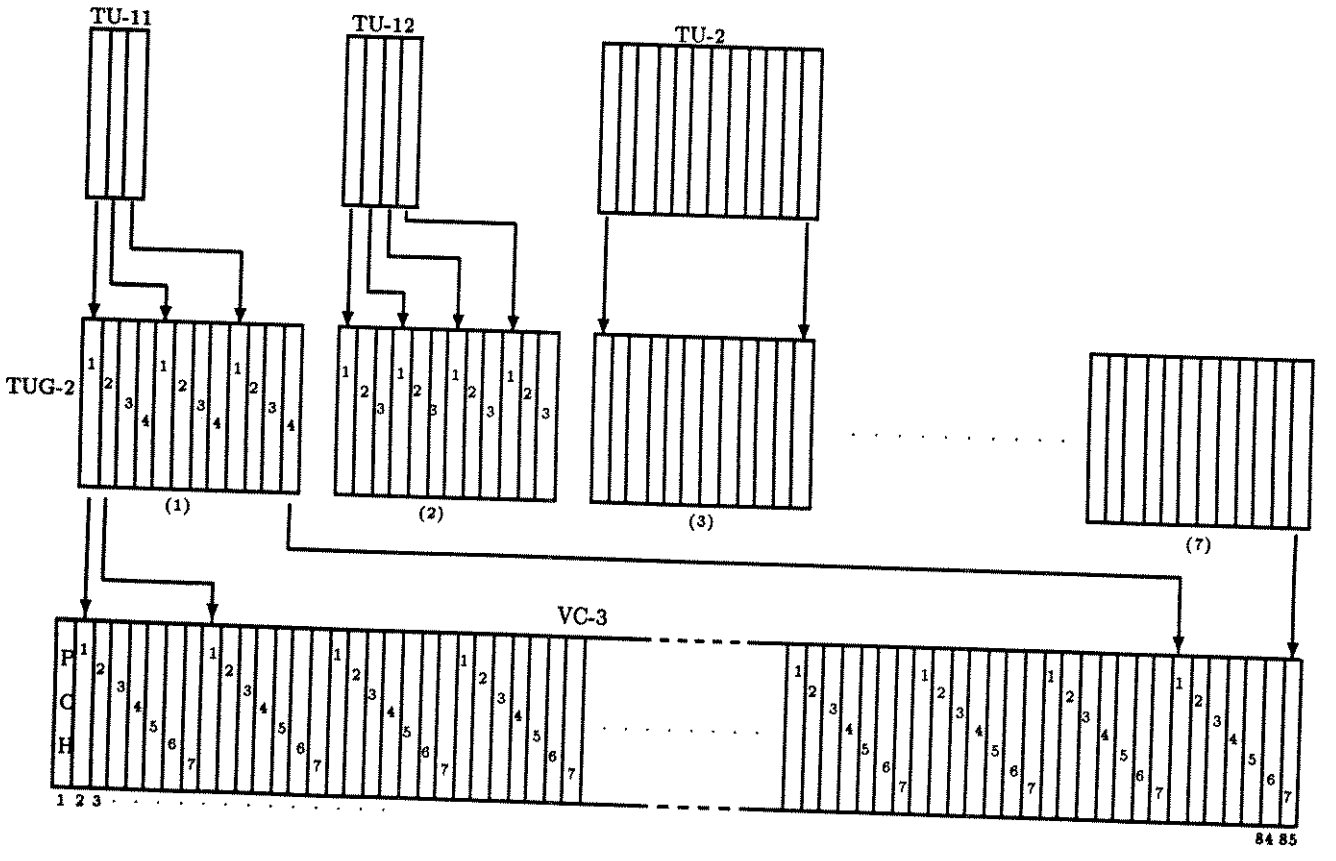
Multiplexação de TU-2s via TUG-2

A multiplexação de um único TU-2 via TUG-2 é ilustrada na figura 2.7. O TU-2 é colocado diretamente no TUG-2.

Multiplexação de TU-1s via TUG-2

A multiplexação de quatro TU-11 ou três TU-12 é ilustrada na figura 2.7. Os TU-1s são entrelaçados byte a byte para formar o TUG-2.

Figura 2.7 : Multiplexação de sete TUG-2s via VC-3.



2.1.2 A Operação de Alinhamento

Na Hierarquia Digital Síncrona, alinhamento é o processo pelo qual a informação relativa a diferença de fase entre um quadro de suporte de rota (VC-nm) e outro quadro de adaptação (TU-n ou AU-m), é processada e colocada em bytes específicos deste ultimo quadro. Estes bytes compoem o *Ponteiro* do quadro de adaptação que provê alinhamento flexível e dinâmico de um VC em relação a um AU/TU. Flexível no sentido de o VC poder "flutuar" acomodando diferenças de fase e dinâmico por permitir diferenças de frequência entre os quadros.

Os Ponteiros dos Quadros AU-3/4 e TU-3

Alocação dos Ponteiros

Os quadros AU-4, AU-3 e TU-3 reservam três de seus bytes para alocação de seus ponteiros. Estes bytes são H1,H2 e H3 como mostrado nas figuras 2.8,2.9 e 2.10, para AU-4, AU-3 e TU-3.

Figura 2.8 : Alocação do Ponteiro no Quadro AU-4

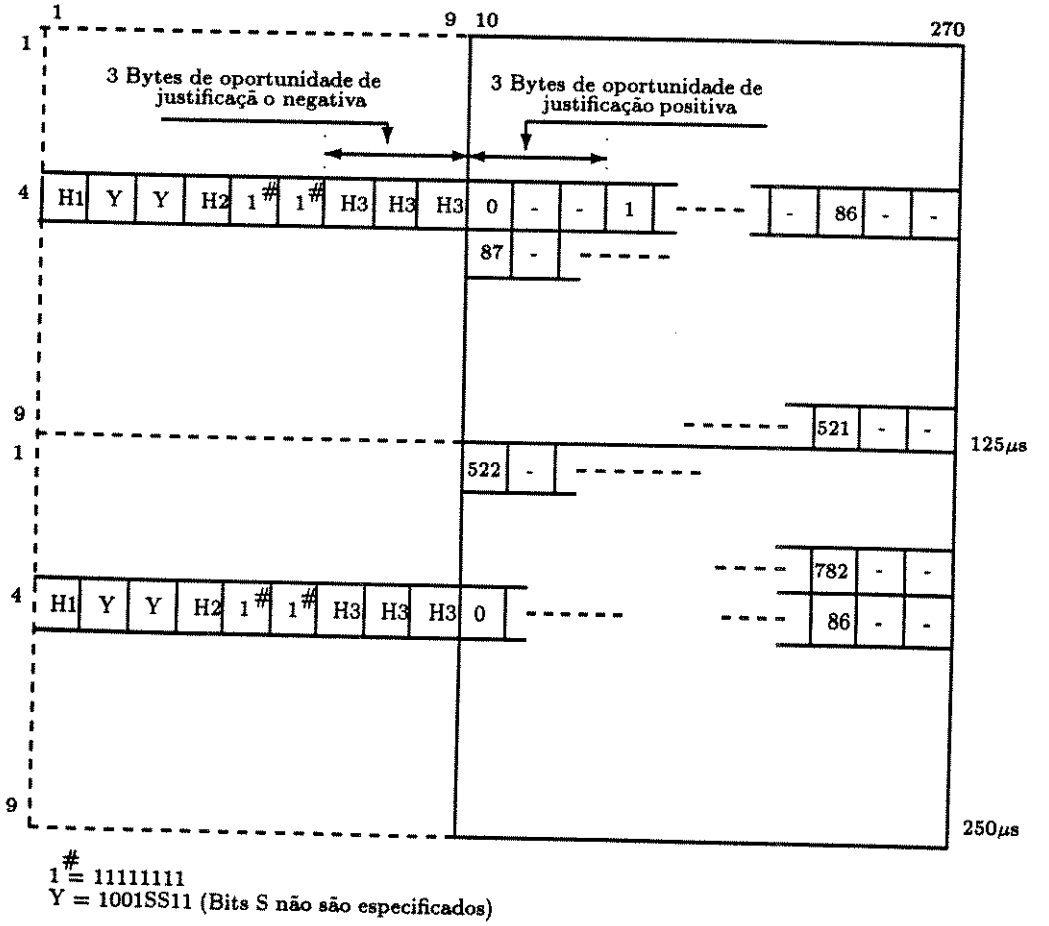


Figura 2.9 : Alocação do Ponteiro no Quadro AU-3.

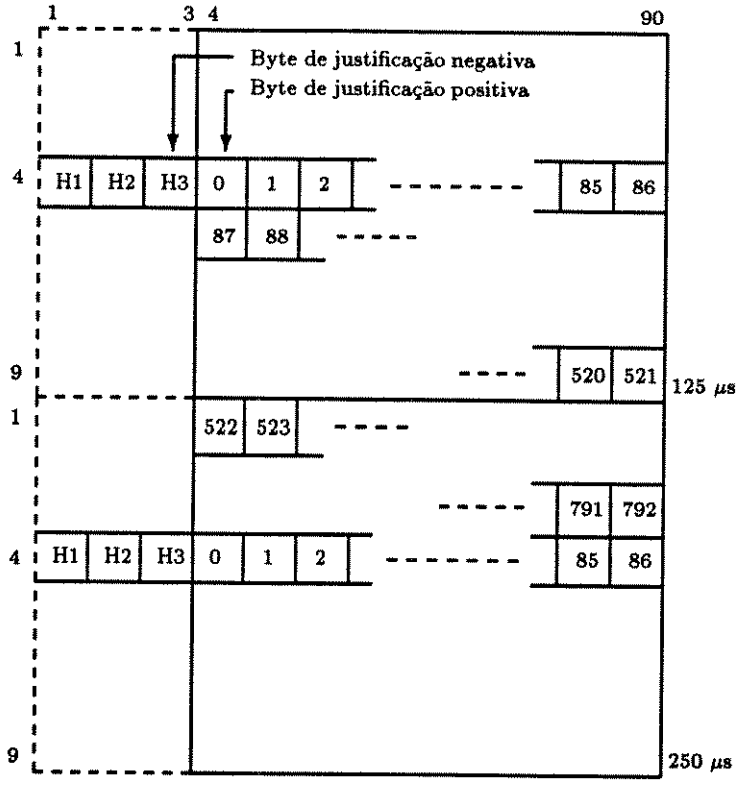
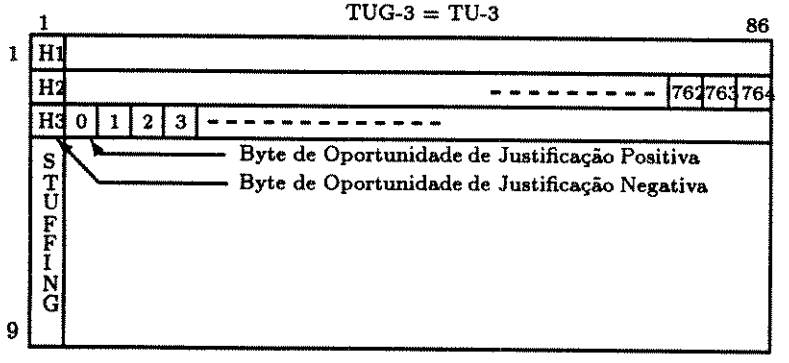


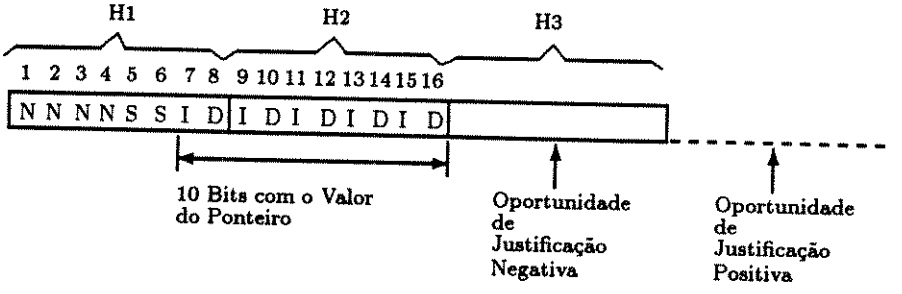
Figura 2.10 : Alocação do Ponteiro no Quadro TU-3.



Os Conteúdos dos Ponteiros

Os bytes H1 e H2 definem a posição do primeiro byte do VC contido no quadro AU/TU-3. Estes dois bytes formam a palavra mostrada na figura 2.11.

Figura 2.11 : Conteúdo dos Ponteiros dos Quadros AU/TU-3.



I - Bit de Incremento
D - Bit de Decremento
N - Sinalização de Novos Dados(NDF)

Novo Valor de Ponteiro(NDF)

- Abilitado: 1001
- Desabilitado: 0110

Justificação Negativa

- Inverte os 5 bits D
- Aceita votação majoritária

Justificação Positiva

- Inverte os 5 bits I
- Aceita votação majoritária

Valor do Ponteiro(bits 7-16):

AU-4 e AU-3: 0 - 782 decimal
TU-3: 0 - 764 decimal

Indicador de Concatenação(CI):

H1H2 = 1001SS1111111111
(bits S não são especificados)

Indicador de Ponteiro Nulo(NPI):

H1H2 = 1001SS1111100000
(bits S não são especificados)

Valor de SS	Tipo de AU/TU
10	AU-4, AU-3, TU-3

Nota 1 - NPI se aplica somente para TUG-3 transportando 7 TUG-2.

Nota 2 - Todos os bits do ponteiro são "1" quando ocorre AIS.

Os ultimos 10 bits (7 - 16) da palavra H1 + H2 carregam o valor do ponteiro. Como mostrado na figura 2.11, o valor do ponteiro é um número binário contido nas seguintes faixas:

- 0 a 782 para AU-4 e AU-3
- 0 a 764 para TU-3

Este número indica o deslocamento entre o ponteiro e o primeiro byte do VC. Para AU-3 e TU-3 este deslocamento é contado byte a byte, ou seja, se o ponteiro do quadro AU-3 contém o número 10 por exemplo, então entre o byte H3 e o primeiro byte do VC-3 relacionado a este AU-3 existem 10 bytes. O mesmo se passa com o TU-3.

Para o AU-4 o deslocamento é contado de três em três bytes, ou seja, se o seu ponteiro contém por exemplo o número 10, então existem 30 bytes entre o terceiro byte H3 e o primeiro byte do VC-4 relacionado a este AU-4.

A figura 2.11 também indica um valor de ponteiro adicional, o indicador de concatenação(CI). Quadros AU-4s podem ser concatenados para juntos formarem um quadro AU-4-Xc capaz de transportar cargas úteis que requerem capacidades maiores que a de um C-4. Os quadros VC-4s contidos em um AU-4-Xc formam uma estrutura chamada VC-4-Xc na qual a primeira coluna é usada para POH e todas as demais colunas estão disponiveis para a carga útil. O indicador de concatenação(CI), usado para indicar que a integridade desta carga útil multipla de C-4 carregada em um unico VC-4-Xc deve ser mantida, está contido no ponteiro de cada AU-4 usado para formar o AU-4-Xc, com exceção do primeiro AU-4. O primeiro AU-4 do AU-4-Xc deve ter operação normal de ponteiro. Todos os demais AU-4s contidos no AU-4-Xc devem ter seus ponteiros assim:

- Bits de 1 a 4 : CI = 1001.
- Bits 5 e 6 : não especificados.
- Bits 7 a 16 : dez bits "1"s.

O CI determina que o processador de ponteiro deve realizar as mesmas operaçoẽs ealizadas sobre o primeiro AU-4 do AU-4-Xc.

Quando quadros TUG-2s são multiplexados dentro de um quadro VC-3, o ponteiro do quadro TU-3 é preenchido com indicador de ponteiro nulo(NPI). Isto se dá porque os quadros TUG-2s já estão previamente sincronizados com o quadro TU-3. NPI é indicado por:

- Bits 1-4 : 1001.
- Bits 5-6 : não especificados.
- Bits 7-11 : 11111.
- Bits 12-16 : 00000.

A figura 2.11 também indica o valor de ponteiro para NPI.

Justificação de Frequência

Se existe uma diferença de frequência entre o quadro AU/TU-3 e o quadro VC correspondente, então o valor do ponteiro será incrementado ou decrementado conforme a necessidade, seguido de uma justificação(positiva ou negativa) de byte ou bytes(no caso do AU-4 a justificação se dá de 3 em 3 bytes). Operaçoẽs consecutivas de ponteiros devem ser separadas de no mínimo 3 quadros nos quais o valor do ponteiro permanece inalterado.

Se a taxa do VC é menor que a do AU/TU-3, então periodicamente o alinhamento do VC deve ser atrasado no tempo e o valor do ponteiro deve ser incrementado de um. Esta operação é indicada invertendo-se os bits 7, 9, 11, 13 e 15 (bits Is) do ponteiro. Para superar erros de transmissão, é tomada decisão majoritária nos cinco bits Is. No caso do AU-4, três bytes de justificação positiva aparecem imediatamente após o ultimo byte H3 do ponteiro contendo bits Is invertidos. No caso do AU-3/TU-3, um byte de justificação positiva aparece imediatamente após o byte H3 do ponteiro contendo bits Is invertidos. Ponteiros subsequentes irão conter o novo deslocamento. Este processo é ilustrado na figura 2.12 para AU-4.

Se a taxa do VC é maior que a do AU/TU-3, então periodicamente o alinhamento do VC deve ser adiantado no tempo e o valor do ponteiro deve ser decrementado de um. Esta operação é indicada invertendo-se os bits 8, 10, 12, 14 e 16 (bits Ds) do ponteiro. Para superar erros de transmissão, é tomada decisão

majoritária nos cinco bits Ds. No caso do AU-4, três bytes de justificação negativa aparecem nos bytes H3s do ponteiro contendo bits Ds invertidos. No caso do AU-3/TU-3, um byte de justificação negativa aparece no byte H3 do ponteiro contendo bits Ds invertidos. Ponteiros subsequentes irão conter o novo deslocamento. Este processo é ilustrado na figura 2.13 para AU-4.

Os Ponteiros dos Quadros TU-1 e TU-2

Diferentemente dos ponteiros dos quadros AU-4, AU-3 e TU-3 onde o processamento é feito uma vez a cada 125 μs, nos ponteiros dos quadros TU-1 e TU-2 o processamento é feito uma vez a cada 500 μs.

Alocação dos Ponteiros

Os quadros TU-1 e TU-2 reservam três de seus bytes para alocação de seus ponteiros. Estes bytes são V1, V2 e V3 como mostrado na figura 2.14.

Os Conteúdos dos Ponteiros

Os bytes V1 e V2 designam a localização do primeiro byte do VC contido no quadro TU. Estes dois bytes formam a palavra mostrada na figura 2.15.

Os últimos 10 bits (7 - 16) da palavra V1 + V2 carregam o valor do ponteiro. Como mostrado nas figuras 2.15 e 2.16 , o valor do ponteiro é um número binário contido nas seguintes faixas:

- 0 a 427 para TU-2
- 0 a 139 para TU-12
- 0 a 103 para TU-11

Este número indica o deslocamento entre o ponteiro e o primeiro byte do VC-1/VC-2. Este deslocamento é contado byte a byte, ou seja, se o ponteiro do quadro TU-1/TU-2 contém o número 10 por exemplo, então entre o byte V2 e o primeiro byte do VC relacionado a este TU existem 10 bytes. Os bytes V1, V2,

V3 e V4 não devem ser contados para a composição do valor do ponteiro. Os dois bits S(Bits 5 e 6 da palavra V1 + V2) indicam o tipo de TU, assumindo os valores:

- 00 para TU-2
- 10 para TU-12
- 11 para TU-11

Figura 2.12 : Operação de Ajustamento do Ponteiro do AU-4 - Justificação Positiva

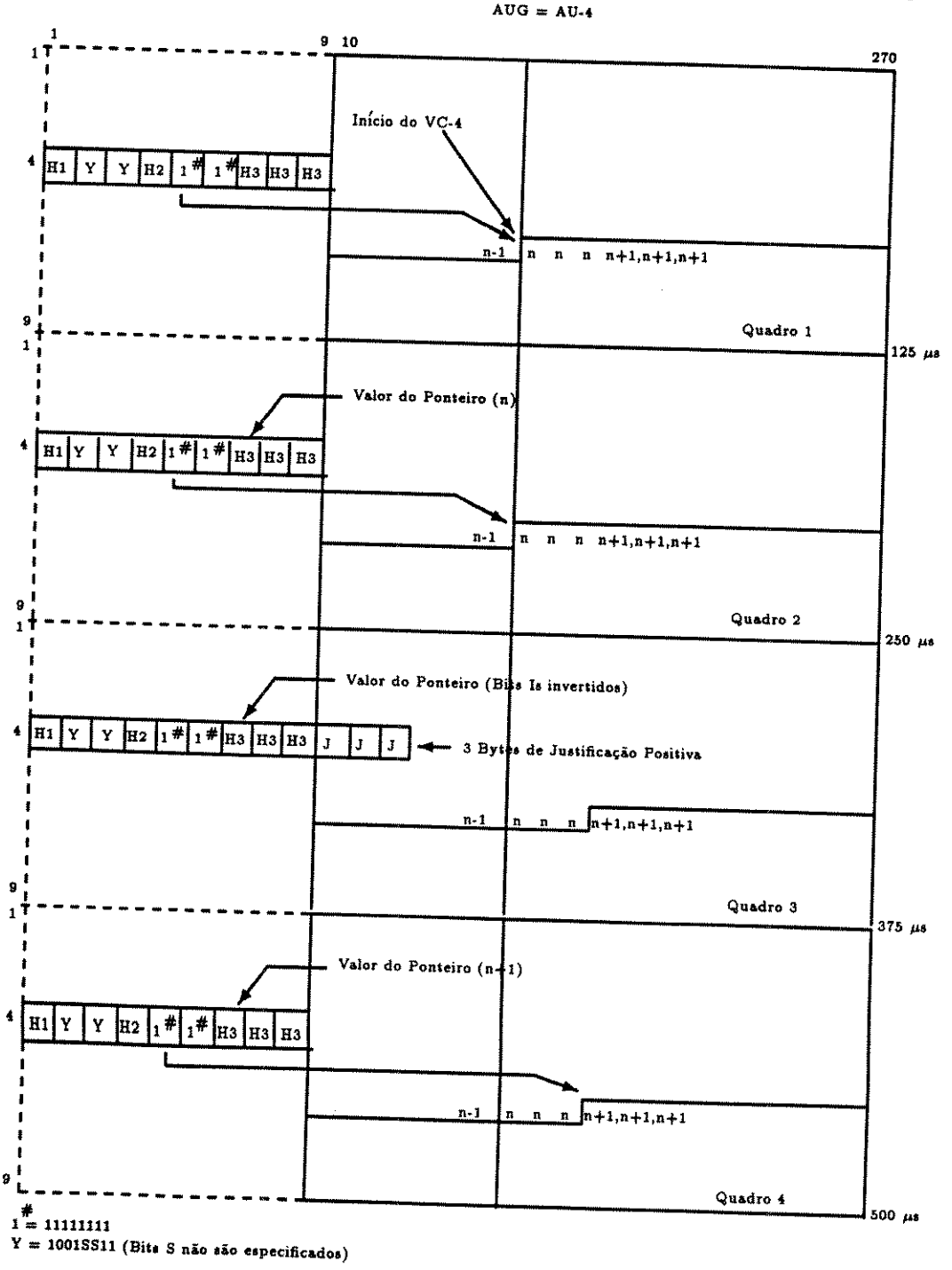


Figura 2.13 : Operação de Ajustamento do Ponteiro do AU-4 - Justificação Negativa

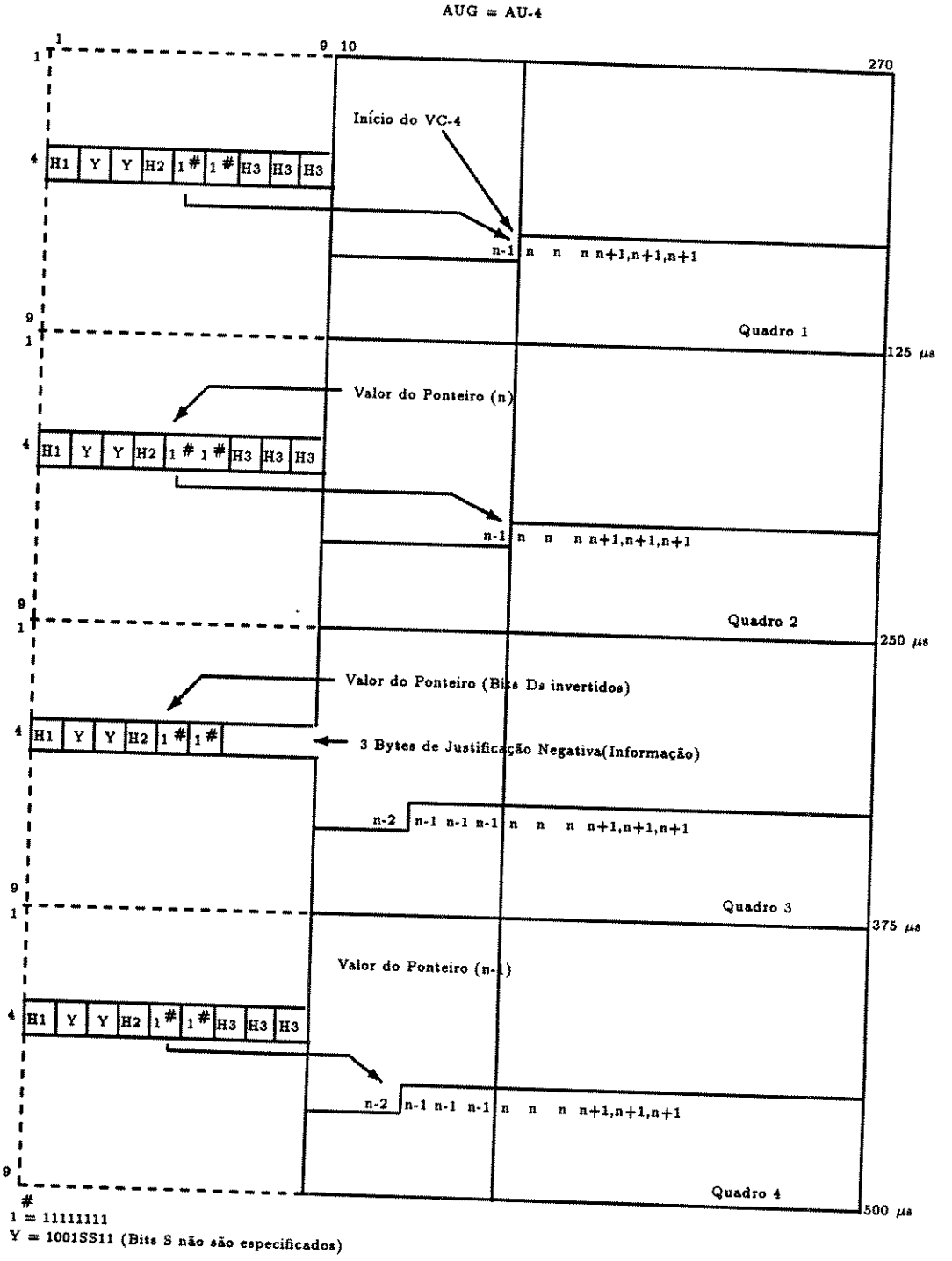
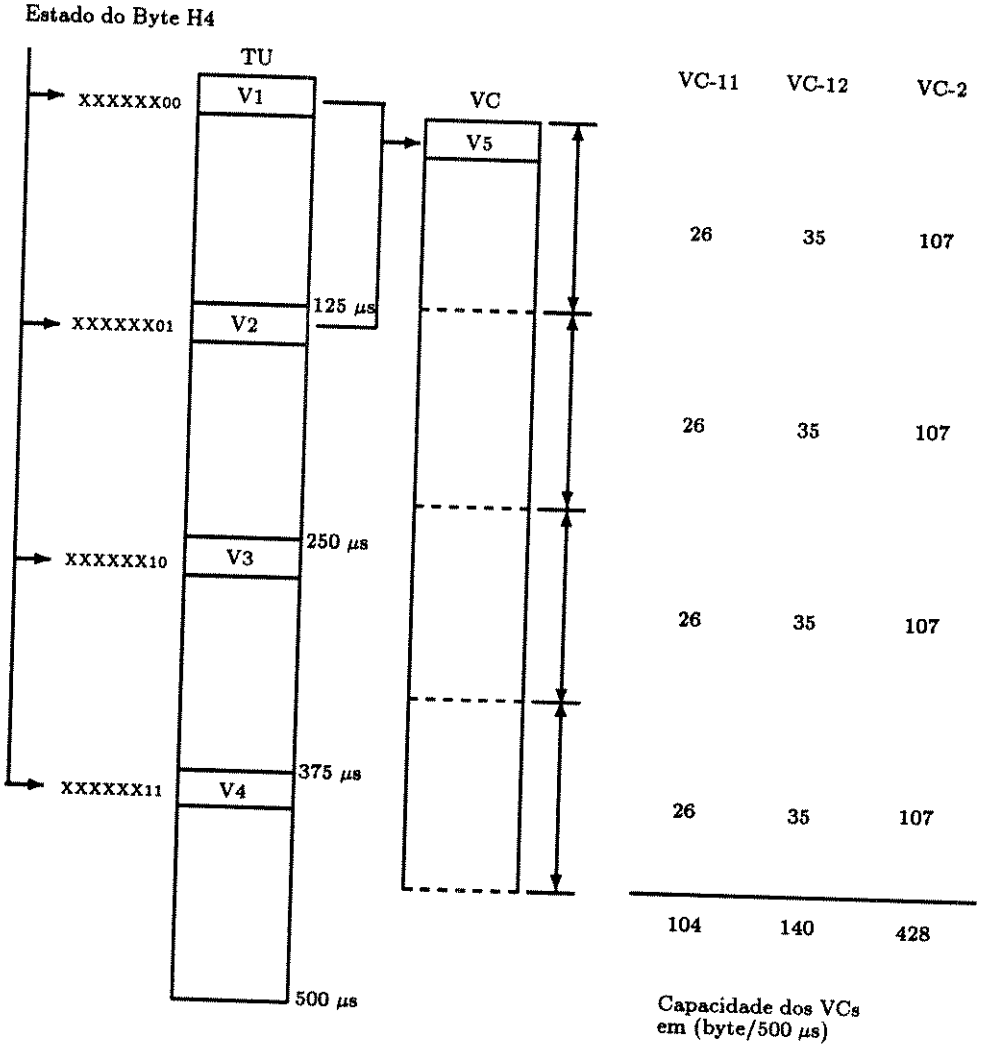


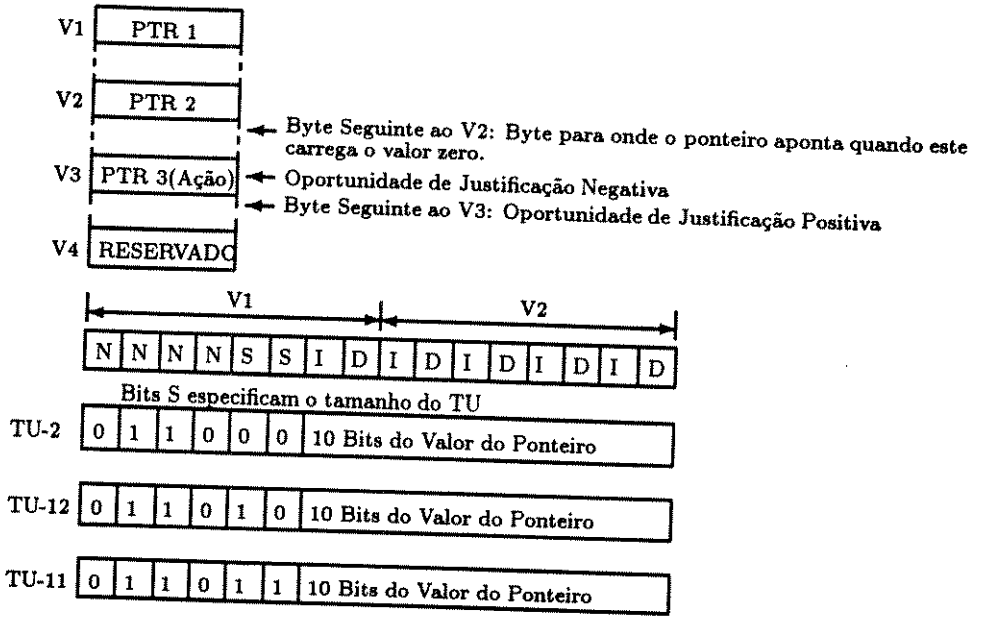
Figura 2.14 : Mapeamento de VCs em Estruturas TUs de Super Quadro.



V1 = Byte #1 do Ponteiro
V2 = Byte #2 do Ponteiro
V3 = Byte de Oportunidade de Justificação Negativa
V4 = Byte Reservado

Nota - V1, V2, V3 e V4 são parte do TU e são terminados no processador de ponteiro.

Figura 2.15 : Conteúdo dos Ponteiros dos Quadros TU-1 e TU-2.



I - Bit de Incremento
D - Bit de Decremento
N - Sinalização de Novos Dados(NDF)

Novo Valor de Ponteiro(NDF)

- Abilitado: 1001
- Desabilitado: 0110
- Estes valores somente são aceitos se recebidos com exatidão.

Justificação Negativa

- Inverte os 5 bits D
- Aceita votação majoritária

Justificação Positiva

- Inverte os 5 bits I
- Aceita votação majoritária

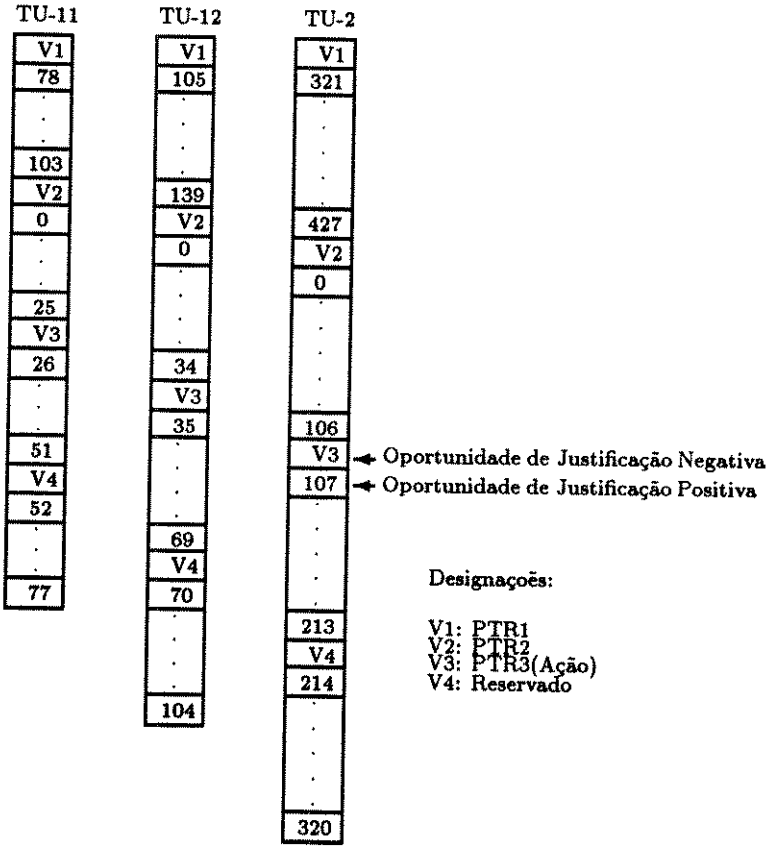
Valor do Ponteiro(bits 7-16):

TU-1: 0 - 427 decimal
TU-12: 0 - 139 decimaL
TU-11: 0 - 103 Decimal

Indicador de Concatenação(CI):

V1V2 = 1001SS1111111111
(bits S não são especificados)

Figura 2.16 : Alocação dos Bytes V1,V2,V3 e V4 do Quadro TU-1/TU-2.



Conteúdo de V5

Um outro conteúdo que deve ser definido é o de V5, que corresponde a carga de supervisão(POH) de VC12 ou VC11. O byte V5 provê funções de checagem de erro, status de rota e composição da carga útil. A figura 2.17 mostra o conteúdo deste ponteiro.

- Bits 1 e 2 são usados para monitoração de performance de erro. O BIP-2 (Bit Interleaved Parity - 2) funciona como um CRC com o resto sendo um polinômio do tipo ax^1+a_0. BIT#1 é setado paridade par para os bits impares do container e BIT#2 é setado paridade par para os bits pares do container. Convêm salientar que o cálculo dos bits de paridade inclui todos os bytes de um VC12 mas exclui os bytes V1,V2,V3 (exceto quando ocorre uma justifação negativa) e V4.

- Bit 3 indica erro na recepção da rota do lado remoto (REI - Remote Error Indication). Este bit é setado para um se um ou mais erros forem detectados no BIP-2 recebido do outro lado.

- Bit 4 a Indicação Remota de Falha na rota (RFI-Remote Failure Indication). Ele é setado para um se uma falha foi encontrada na montagem do VC, ou é setado para zero. Se uma falha é detectada e persistir durante um certo tempo um mecanismo de proteção do sistema é acionado.

- O conteúdo do container dado pelos três bits (5,6,7) está mostrados na figura 2.14 onde: Não equipado significa que o VC está livre, Equipado-Não especificado significa que o VC têm conteúdo mas não se sabe qual.As outras três possibilidades definem o tipo de mapeamento e os demais não estão sendo usados ainda.

- O RDI (Remote Defect Indication) indica um defeito remoto se foi recebido um AIS (Alarm Indication Signal) ou uma falha no sinal.É setado para um quando o defeito é detectado.

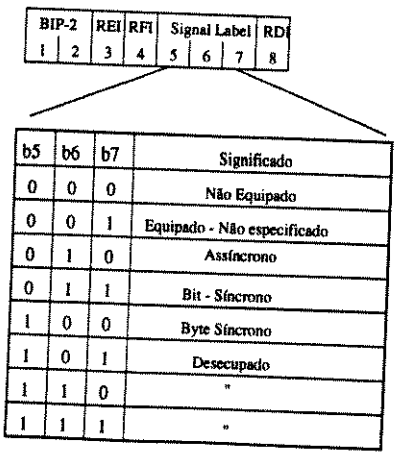
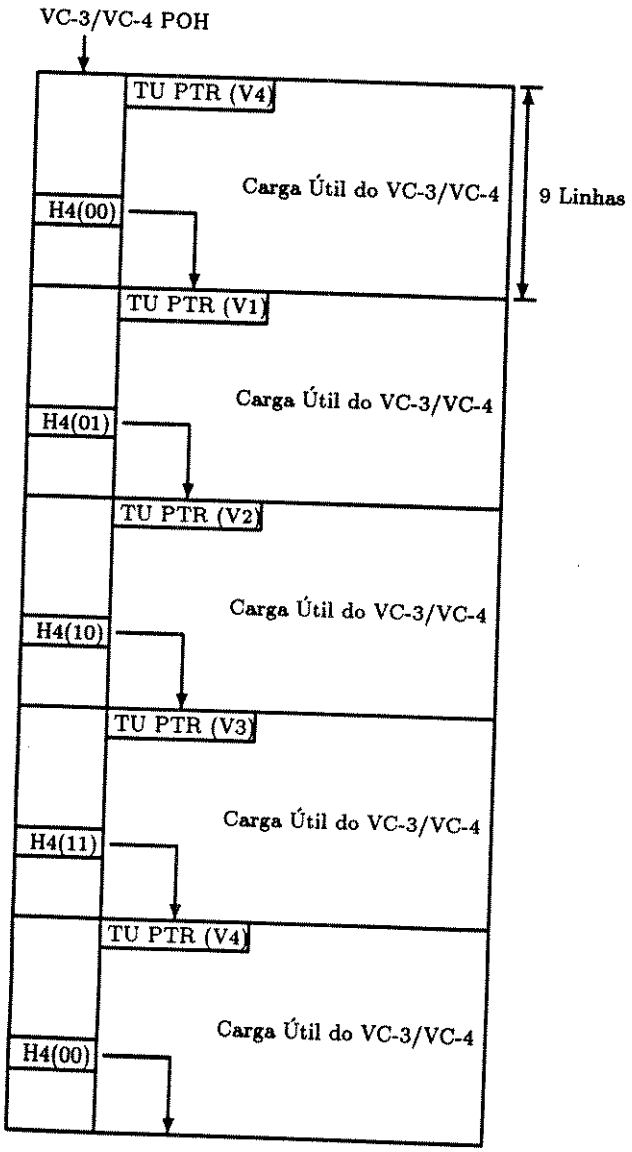


Figura 2.17 : Conteúdo do Byte V5.

O Byte Indicador de Super Quadro

Multiplos quadros TU-1/TU-2 podem ser processados conjuntamente para formar uma de três estruturas de superquadro existentes. Estas estruturas são multiplexadas dentro de um VC-3 ou VC-4 e devem ser identificadas por um byte indicador de superquadro(H4) que faz parte do POH do VC-3/VC-4. A figura 2.18 mostra os dois últimos bits do byte H4, usado para indicar a estrutura de superquadro composta por 4 quadros.

Figura 2.18 : Indicador de Fase de Superquadro.(byte H4)



Justificação de Frequência

Os ponteiros dos quadros TU-1/TU-2 são usados para justificação de frequência do quadro VC-1/VC-2, seguindo as mesmas regras estabelecidas para o ponteiro do quadro TU-3 usado para justificação de frequência do quadro VC-3. Nesta analogia, os bytes V1, V2 e V3 do quadro TU-1/TU-2 se associam respectivamente aos bytes H1, H2 e H3 do quadro TU-3, exercendo funções análogas no processo de justificação de frequência. Todo o processo de justificação está bem esclarecido nas figuras 2.16 e 2.17.

Sinalização de Novos Dados (NDF)

Os primeiros 4 bits(N bits) da palavra formada pelos bytes H1 e H2 (AU-4/AU-3/TU-3) ou V1 e V2 (TU-1/TU-2) carregam sinalização de novos dados(NDF) que permite uma mudança arbitrária no valor do ponteiro para comportar uma mudança de carga útil. No caso do AU-3/AU-4/TU-3 a decodificação de NDF pode ser aceita se no mínimo três dos quatro bits coincidirem com o código de NDF. Já no caso do TU-1/TU-2, os quatro bits devem coincidir exatamente com o código de NDF. Este código é o seguinte:

- 0110 para operação normal.
- 1001 para indicação de novo alinhamento.

O ponteiro seguinte à operação de NDF carrega o novo alinhamento que o VC deve tomar. Para o TU-1/TU-2 este ponteiro pode trazer também uma mudança no tamanho do VC. Quando isto ocorre, todos os ponteiros de TUs contidos em um TUG-2 devem indicar simultaneamente NDF com o mesmo tamanho de VC.

Geração e Interpretação do Ponteiro

Geração

1. Durante operação normal o ponteiro localiza o início do quadro VC dentro do quadro AU/TU. Os quatro bits de NDF são preenchidos com "0110".
2. O valor do ponteiro pode ser mudado somente pelas operaçoẽs 3, 4 ou 5.

3. Se justificação positiva é requerida, os bits Is do ponteiro corrente são invertidos e o byte ou bytes relativos a oportunidade de justificação positiva é preenchido ou são preenchidos com informação inválida. Os ponteiros subsequentes contêm o valor do ponteiro anterior à justificação, incrementado de 1. Nenhuma operação subsequente de incremento ou decremento no valor do ponteiro é permitida para no mínimo os próximos três quadros seguintes a esta operação.

4. Se justificação negativa é requerida, os bits Ds do ponteiro corrente são invertidos e o byte ou bytes relativos a oportuninade de justificação negativa é usado ou são usados para transportar informação válida. Os ponteiros subsequentes contêm o valor do ponteiro anterior à justificação, decrementado de 1. Nenhuma operação subsequente de incremento ou decremento no valor do ponteiro é permitida para no mínimo os próximos três quadros seguintes a esta operação.

5. Se o alinhamento do quadro VC muda por qualquer outra razão que não sejam as regras 3 ou 4, o novo valor de ponteiro deve vir acompanhado pelos bits de NDF preenchidos com "1001". Os bits de NDF são preenchidos com "1001" somente no primeiro ponteiro que contém o novo valor, os ponteiros seguintes trazem bits de NDF com "0110". A nova localização do VC é indicada pelo novo valor de ponteiro e já é válida no primeiro ponteiro com o novo valor. Nenhuma operação subsequente de incremento ou decremento no valor do ponteiro é permitida para no mínimo os próximos três quadros seguintes a esta operação.

6. Regra adicional válida para o ponteiro do AU-4: Se um sinal de AU-4-Xc está sendo transmitido, o ponteiro é gerado somente para o primeiro AU-4. O indicador de concatenação(CI) é gerado no lugar dos ponteiros dos outros AU-4s do AU-4-Xc. Todas as operaçoẽs realizadas pelo ponteiro do primeiro AU-4, se aplicam também aos ponteiros dos demais AU-4s do AU-4-Xc.

7. Regra adicional válida para o ponteiro TU-1/TU-2: Se o tamanho do TU dentro de um TUG-2 muda, então uma operação de NDF como descrito na regra 5 deve ser realizada simultaneamente em todos os TUs de tamanhos novos contidos neste TUG-2.

Interpretação

1. Durante operação normal o ponteiro localiza o início do quadro VC dentro do quadro AU/TU.

2. Qualquer mudança no valor do ponteiro é ignorada a não ser que um novo valor consistente seja recebido 3 vezes consecutivas ou ele é precedido por uma das regras 3, 4 ou 5. Qualquer valor consistente recebido 3 vezes consecutivas tem prioridade sobre as regras 3 ou 4.

3. Se a maioria dos bits Is do ponteiro são invertidos, uma operação de justificação positiva é indicada. Valores de ponteiros subsequentes devem ser incrementados de 1.

4. Se a maioria dos bits Ds do ponteiro são invertidos, uma operação de justificação negativa é indicada. Valores de ponteiros subsequentes devem ser decrementados de 1.

5. Se os bits de NDF são recebidos como sendo"1001", então o valor do ponteiro que acompanha esta operação de NDF deve substituir o valor do ponteiro corrente independente do estado do receptor.

6. Regra adicional válida para o ponteiro TU-3: Se o ponteiro do TU-3 traz indicação de ponteiro nulo(NPI), então qualquer modificação no seu valor é ignorada a não ser que um valor seja recebido de forma consistente por três vezes consecutivas.

7. Regra adicional válida para o ponteiro do AU-4: Se o ponteiro contém o indicador de concatenação(CI), então todas as operaçoẽs realizadas sobre o ponteiro do primeiro AU-4, se aplicam também aos ponteiros dos demais AU-4s do AU-4-Xc. Qualquer variação do CI é ignorada a não ser que um novo valor consistente de ponteiro seja recebido três vezes consecutivas.

8. Regra adicional válida para o ponteiro TU-1/TU-2: Se são recebidos simultaneamente os bits de NDF como "1001" e um novo tamanho arbitrário de TU em todos os TUs contidos em um TUG-2, então valores coincidentes de ponteiros e tamanhos devem substituir os correntes imediatamente.

2.1.3 A Operação de Mapeamento

Mapeamento é o procedimento pelo qual sinais afluentes são adaptados dentro dos VCs. Aqui se mostra o mapeamento dos sinais afluentes definidos na Recomendação G-703.
Serão definidos somente aqueles mapeamentos que nos interessam, ou seja, definiremos os mapeamentos dos tributários de pequena ordem. Os demais mapeamentos deixaremos de fora.

Mapeamento de Sinais Afluentes no VC-12

Mapeamento Assíncrono de 2.048 kbit/s

Um sinal de 2.048 kbit/s pode ser mapeado em um VC-12. A figura 2.19 mostra isto no modo assíncrono em um período de 500 μs.

Além de POH (V5), o VC-12 leva:

- 1.023 bits de dados (I).
- 6 bits de controle de justificação (C1 e C2).
- 2 bits de oportunidade de justificação (S1 e S2).
- 8 bits de "overhead" (O) reservados para o futuro.
- O restante dos bits é "stuffing".

Em cada 500 μs, os 3 bits de controle de justificação(C1) controlam o bit de oportunidade de justificação(S1) . Quando C1C1C1=000 então S1 é um bit de informação. Do contrário se C1C1C1=111 então S1 é um bit de justificação deve ser descartado. Para superar erros de transmissão, é tomada decisão majoritária nos 3 bits C1.

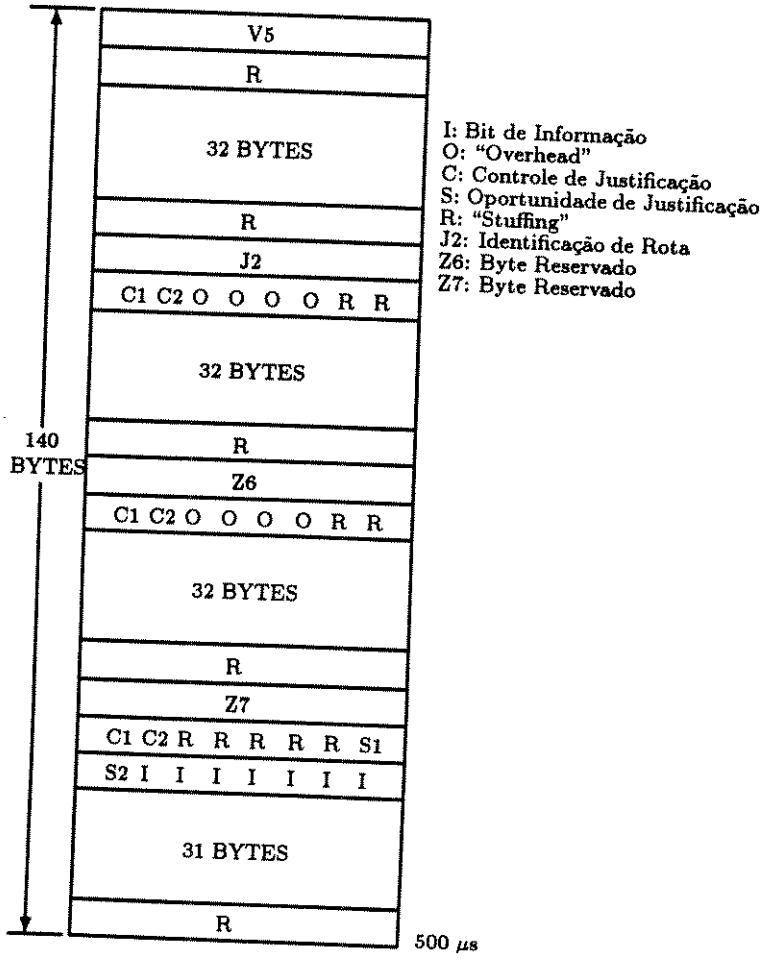
Em cada 500 μs, os 3 bits de controle de justificação(C2) controlam o bit de oportunidade de justificação(S2) . Quando C2C2C2=000 então S2 é um bit de informação. Do contrário se C2C2C2=111 então S2 é um bit de justificação deve ser descartado. Para superar erros de transmissão, é tomada decisão majoritária nos 3 bits C2.

No período de 500 μs existem 1.023 bits de informação com S2≠I. Portanto a taxa de transmissão neste caso deveria ser de:

$$\frac{1.023\ bits}{500\mu s} = 2.046\ kbit/s \tag{2.1}$$

Faltam 2 kbit/s para completar a taxa nominal de 2.048 kbit/s. Na taxa nominal, S2 é bit de informação sempre.

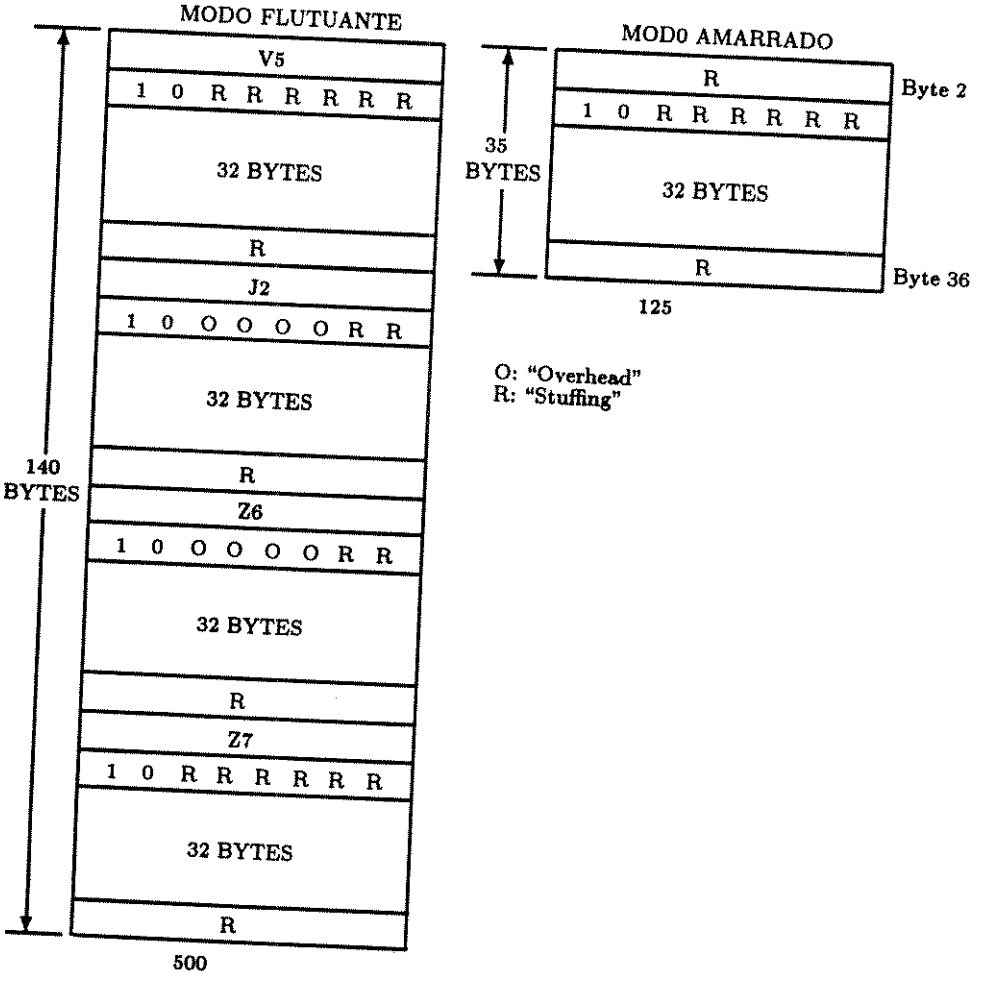
Figura 2.19 : Mapeamento Assíncrono de 2048 kb/s no VC-12.



Mapeamento de 2.048 kbit/s com Sincronismo de Bit

O mapeamento do sinal de 2.048 kbit/s com sincronismo de bit é mostrado na figura 2.20.

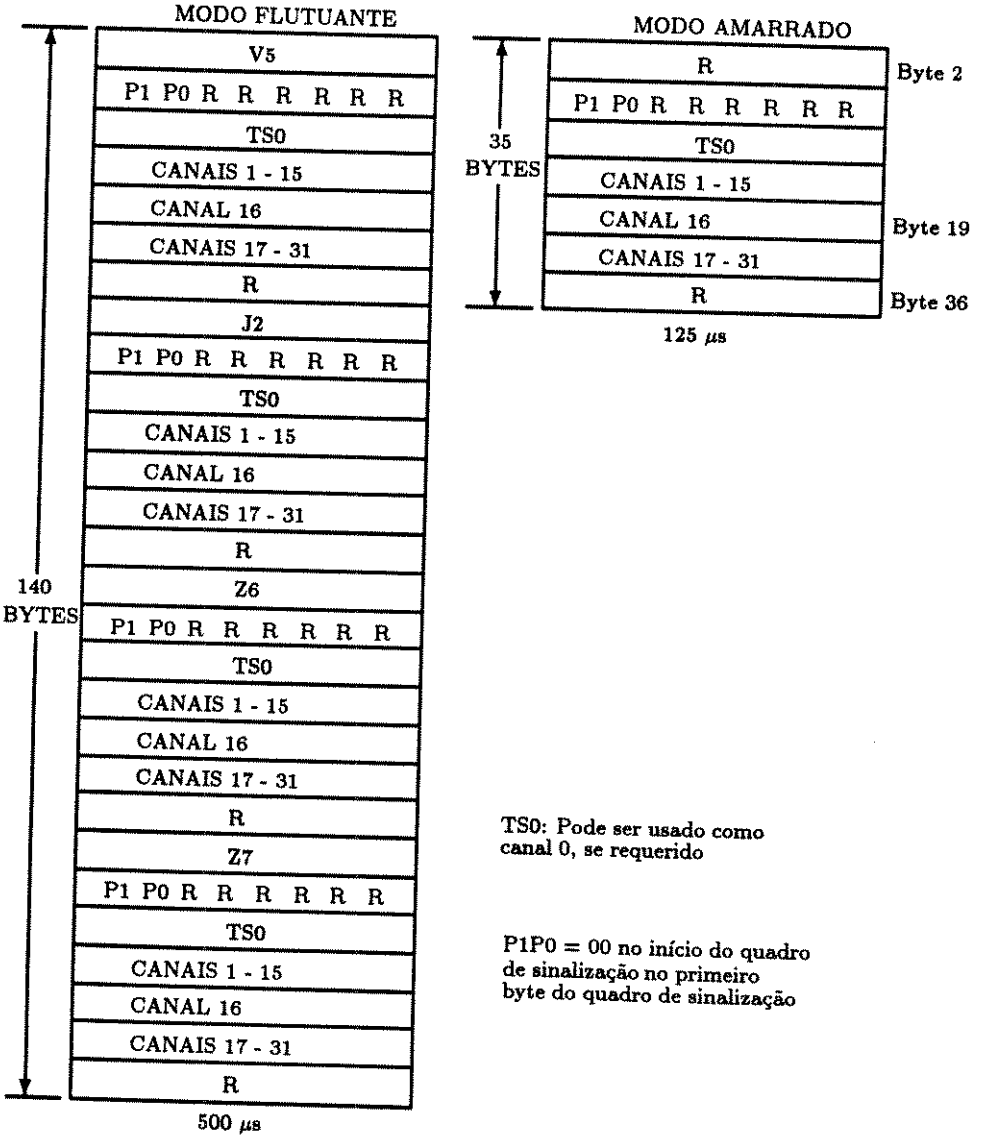
Figura 2.20 : Mapeamento de 2048 kb/s no VC-12 com Sincronismo de Bit.



Mapeamento de 2.048 kbit/s com Sincronismo de Byte

A figura 2.21 mostra o mapeamento com sincronismo de byte para 30 canais(2.048 kbit/s), empregando *sinalização associada a canal* (CAS).

Figura 2.21 : Mapeamento de 2048 kb/s com Sincronismo de Byte (30 Canais com Sinalização Associada a Canal)



Mapeamento de Sinais Afluentes no VC-11

Mapeamento Assíncrono de 1.544 kbit/s

Um sinal de 1.544 kbit/s pode ser mapeado em um VC-11. A figura 2.22 mostra isto no modo assíncrono em um período de 500 μs.

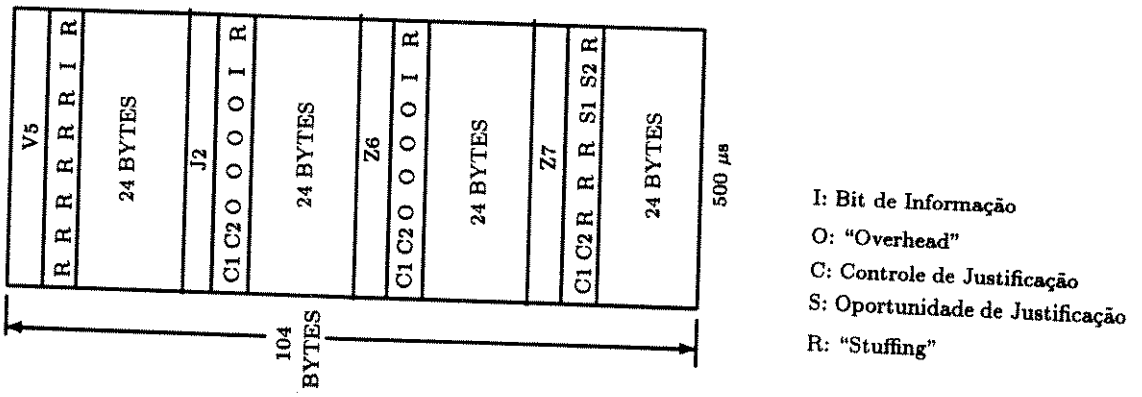


Figura 2.22 : Mapeamento Assíncrono de 1544 kb/s no VC-11.

Além de POH (V5), o VC-11 leva:

- 771 bits de dados (I).
- 6 bits de controle de justificação (C1 e C2).
- 2 bits de oportunidade de justificação (S1 e S2).
- 8 bits de "overhead" (O) reservados para o futuro.
- O restante dos bits é "stuffing".

Em cada 500 μs, os 3 bits de controle de justificação(C1) controlam o bit de oportunidade de justificação(S1) . Quando C1C1C1=000 então S1 é um bit de informação. Do contrário, se C1C1C1=111 então S1 é um bit de justificação. Para superar erros de transmissão, é tomada decisão majoritária nos 3 bits C1.

Em cada 500 μs, os 3 bits de controle de justificação(C2) controlam o bit de oportunidade de justificação(S2) . Quando C2C2C2=000 então S2 é um bit de informação. Ao contrário, se C2C2C2=111, então S2 é um bit de justificação. Para superar erros de transmissão, é tomada decisão majoritária nos 3 bits C2.

No período de 500 μs existem 771 bits de informação S2≠I Portanto a taxa de transmissão neste caso deveria ser de:

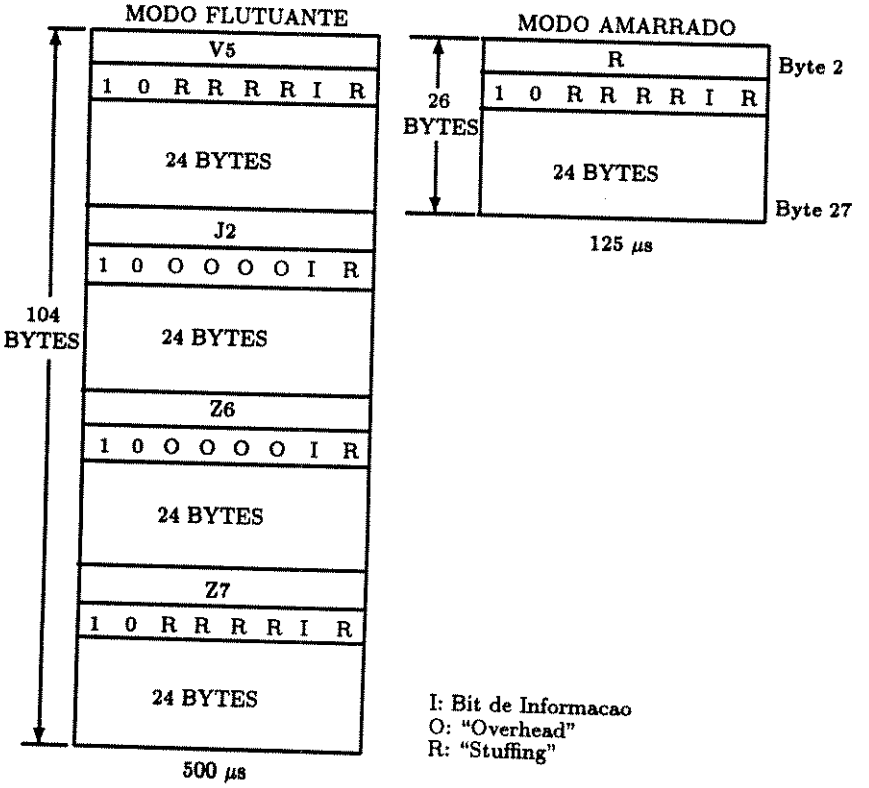
$$\frac{771\ bits}{500\mu s} = 1.542\ kbit/s \tag{2.2}$$

Faltam 2 kbit/s para completar a taxa nominal de 1.544 kbit/s. Na taxa nominal o bit S2 sempre carrega informação.

Mapeamento de 1.544 kbit/s com Sincronismo de Bit

O mapeamento do sinal de 1.544 kbit/s com sincronismo de bit é mostrado na figura 2.23.

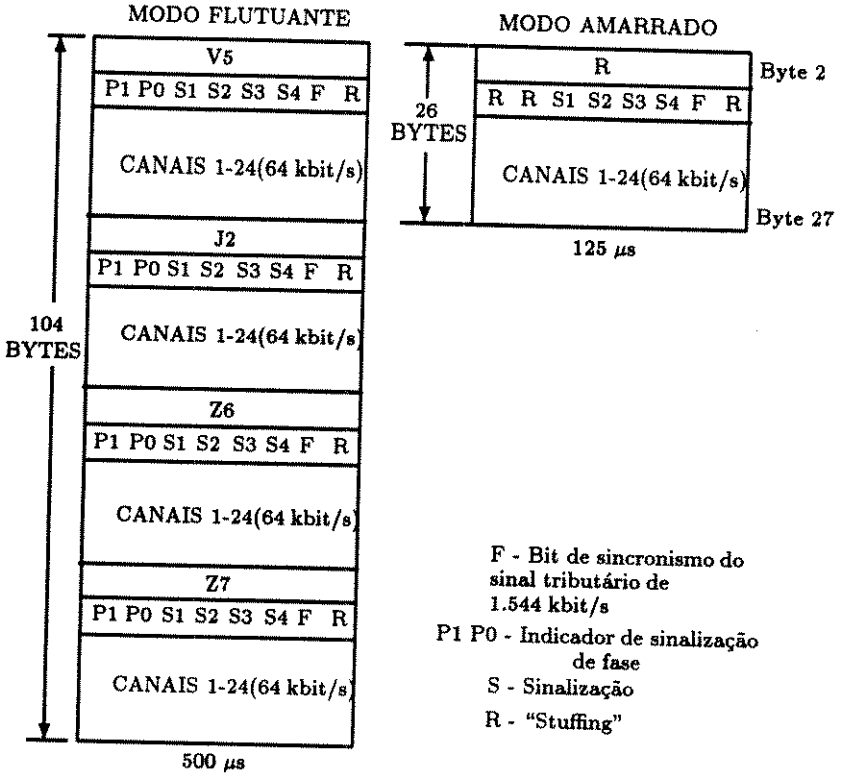
Figura 2.23 : Mapeamento de 1544 kb/s no VC-11 com Sincronismo de Bit.



Mapeamento de 1.544 kbit/s com Sincronismo de Byte

A figura 2.24 mostra o mapeamento com sincronismo de byte para 24 canais(1.544 kbit/s).

Figura 2.24 : Mapeamento de 1544 kb/s no VC-11 com Sincronismo de Byte.



2.2 Modelos em Linguagem VHDL Para Equipamentos da Hierarquia Digital Síncrona

2.2.1 Introdução

Este trabalho foi apresentado como tese de mestrado[2] em Agosto/93 [2]. Aqui é feito um resumo que servirá de base para o desenvolvimento que será mostrado nos próximos capítulos. Este trabalho utilizou como ferramenta para simulação a linguagem VHDL, que possibilita descrever circuitos digitais a nível comportamental.

2.2.2 Projetos de Alto Nível para Blocos Funcionais da Hierarquia Digital Síncrona

Foram projetados, neste trabalho, alguns equipamentos da HDS. A interligação destes equipamentos possibilitou analisar quase toda uma rede de transporte de informação, ou seja: 63 VC12 formando um STM-1. Analisou-se, também o mecanismo das justificações observando o comportamento dinâmico da carga ùtil dentro dos AUs e TUs. Para isto foram projetados os seguintes blocos funcionais:

- No sentido da transmissão (multiplexação):

- HPA_T - Adaptação de containers VC-12 em unidades afluentes TU-12 (modo flutuante).
- MSA_T - Multiplexação de 63 unidades afluentes TU-12 para formar a carga útil de um container VC-4.

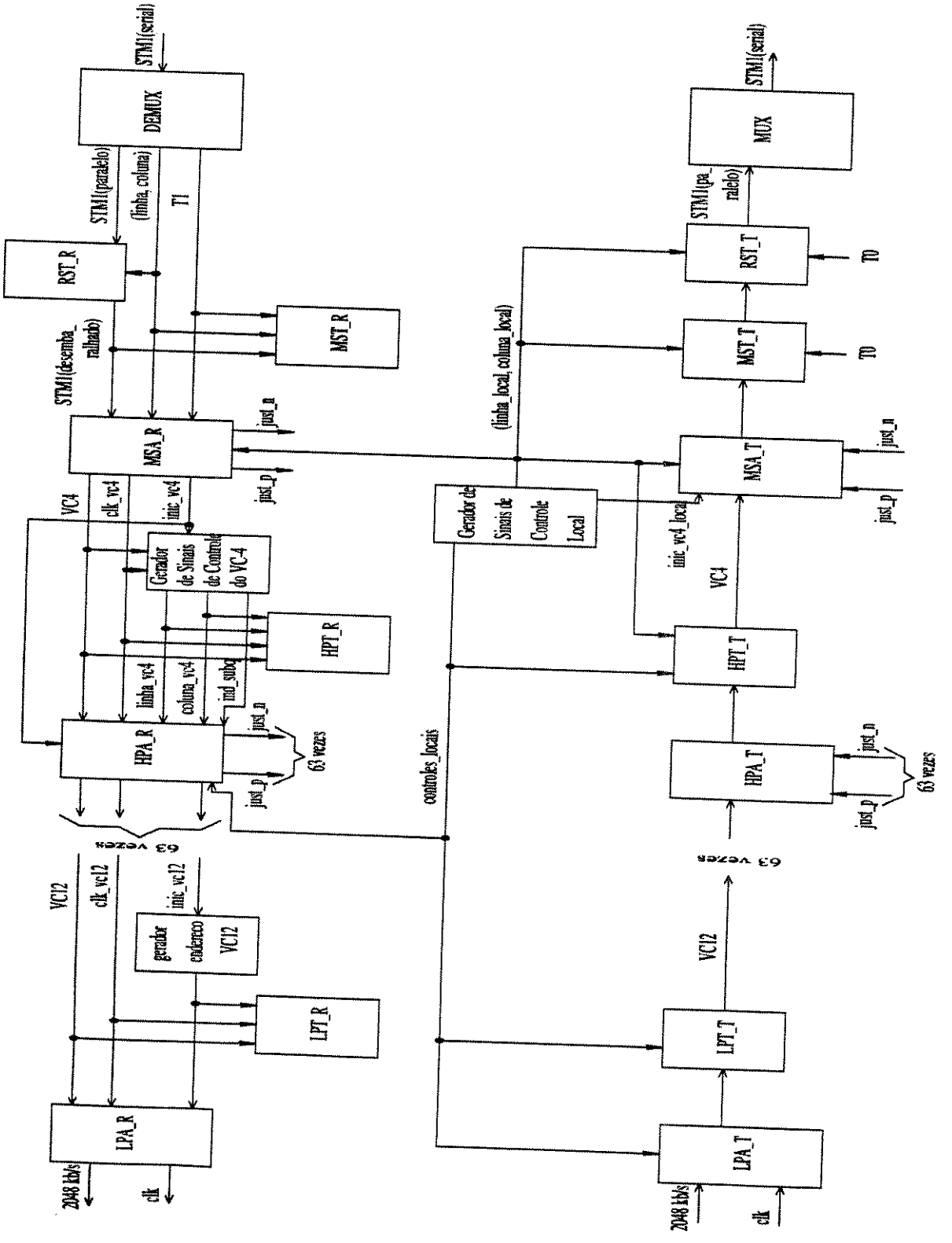
- No sentido da recepção (demultiplexação):

- HPA_R.
- MSA_R.

[2]Tese de Mestrado de Aleandro Macedo

A figura 2.25 mostra um diagrama de blocos funcionais. Os sinais entram na forma de um STM-1(paralelo) no bloco MSA_R saindo como um VC4 e entram no HPA_R, onde saem como 63 VC12. O sentido inverso também é possível, ou seja, 63 VC12 sendo processados para formar um STM-1, passando pelos blocos HPA_T e MSA_T. A seguir é feita uma descrisão resumida destes blocos.

Figura 2.25 : Arranjo dos Funcionias HDS para Terminal de Linha.

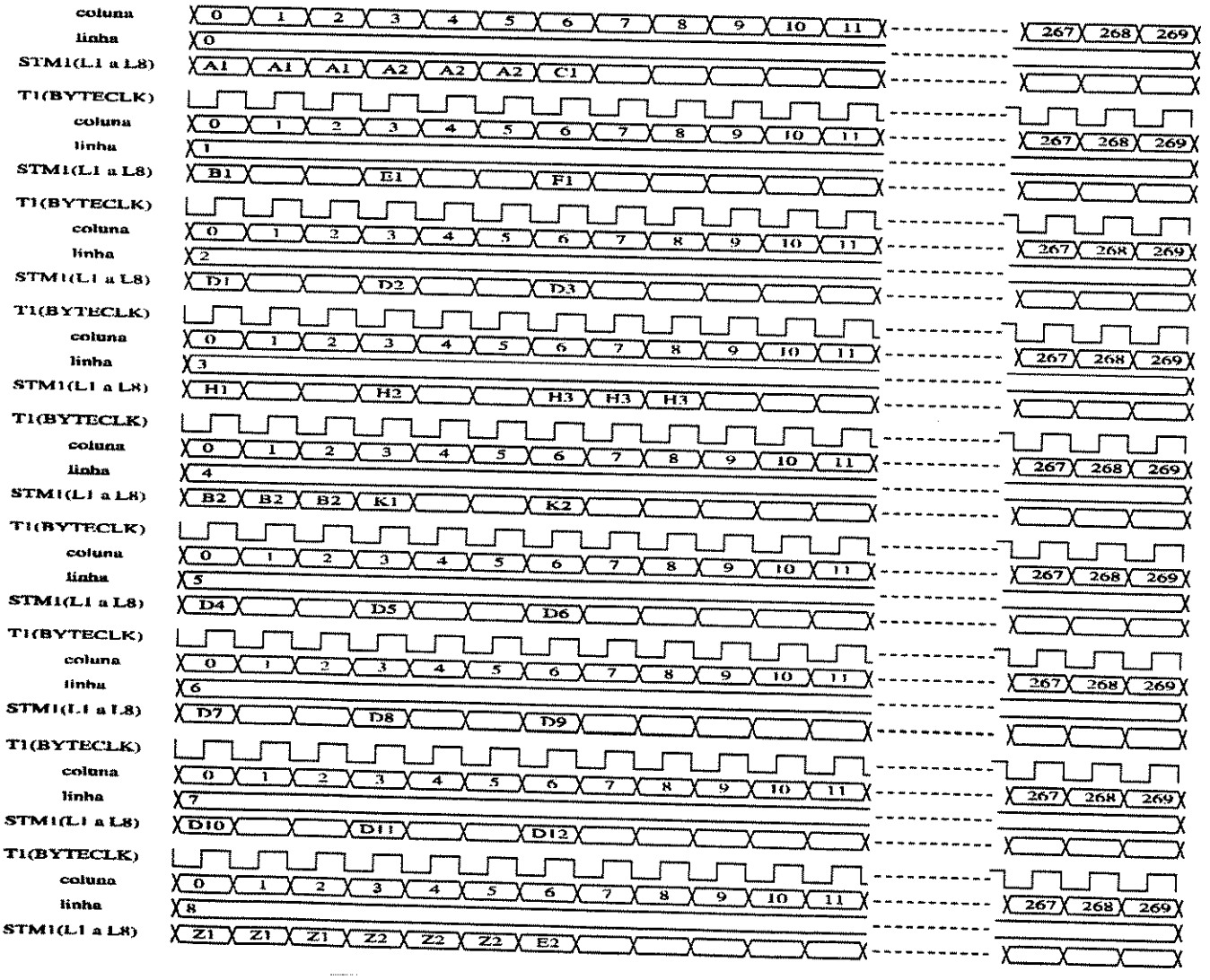


Sinais de entrada

A figura 2.26 mostra o diagrama temporal dos sinais de entrada. Observa-se na figura que estes sinais foram arranjados de forma semelhante a um quadro de 9 linhas por 270 colunas afim de destacar a relação com o quadro STM-1.

O sinal *linha* é do tipo inteiro e evolui de 0 a 8, sendo capaz de identificar as 9 linhas do quadro STM-1. O sinal *coluna* também é do tipo inteiro e evolui de 0 a 269, sendo capaz de identificar as 270 colunas do quadro STM-1. O relógio *T1* apresenta 2430 (9×270) pulsos em $125\mu s$, portanto um pulso para cada byte do quadro STM-1. Finalmente, o sinal *STM1* provém de um barramento de 8 bits. Ele traz um novo byte do quadro STM-1 a cada pulso de *T1*.

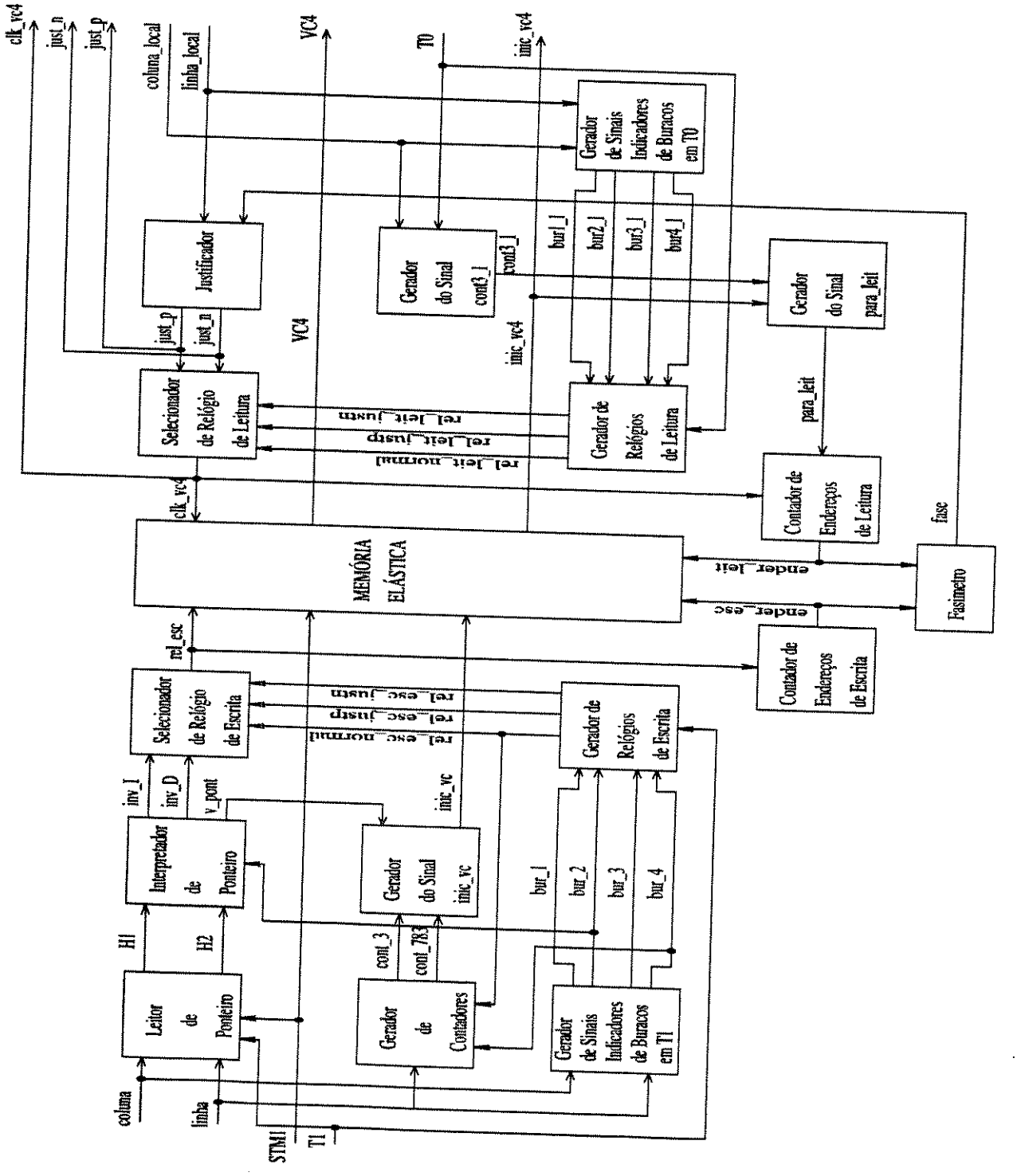
Figura 2.26 : Diagrama temporal dos sinais *coluna e linha* em relação ao relógio T1(BYTECLK) e o sinal STM-1.



2.2.3 Adaptação de Seção Multiplex na Recepção (MSA_R)

A função de Adaptação de Seção Multiplex na Recepção é de retirar do sinal que está chegando os bytes que formam um VC4. A figura 2.27 mostra os diversos blocos que tratam este sinal. Este bloco retira do sinal que está chegando somente os bytes válidos que formam um VC4. Entra um sinal STM-1 de um lado e saí o VC4 do outro. O clock local faz a retirada destes sinais.

Figura 2.27 : Detalhamento do Bloco SA_R.



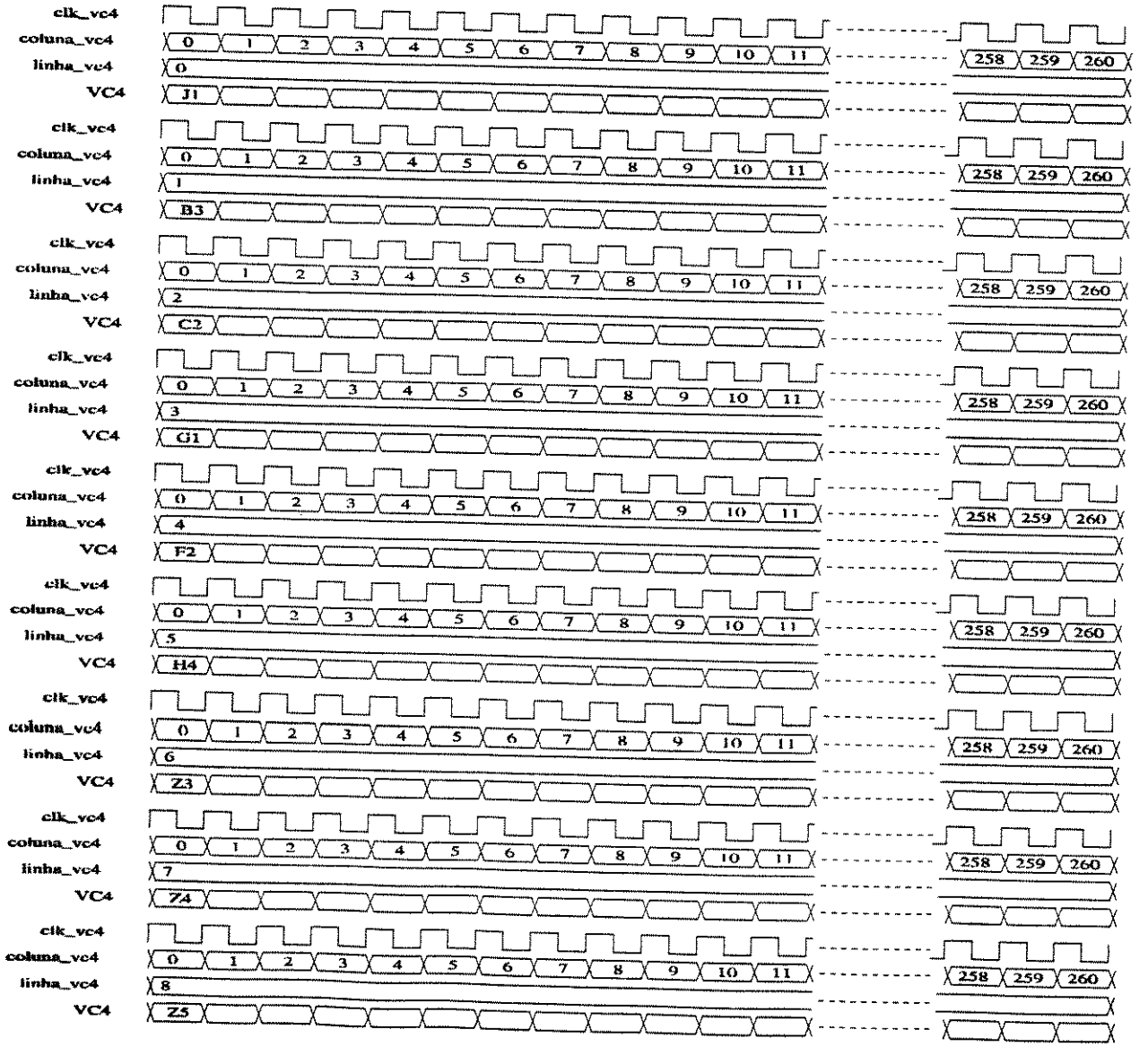
Memória Elástica

A base do funcionamento do bloco MSA_R é a existência de uma memória elástica onde se escrevem os bytes do quadro VC-4 com o relógio esburacado de escrita *rel_esc* derivado do relógio *T1*. Estes bytes são lidos pelo relógio esburacado *clk_vc4* que é derivado do relógio *T0* (clock local do equipamento).

A figura 2.28 mostra o diagrama temporal dos sinais *linha_vc4* e *coluna_vc4*. Apesar da figura 2.28 apresentar o relógio *clk_vc4* com clocks iguais, existem locais onde este relógio fica regularmente parado por 9 pulsos e ocasionalmente ocorrem paradas de 6 e 12 pulsos quando há justificação negativa e positiva respectivamente.

Se o relógio *clk_vc4* não fosse desdentado então se poderia ilustrar os sinais *coluna_vc4* e *linha_vc4* de uma forma mais didática como na figura 2.28.

Figura 2.28 : Diagrama temporal dos sinais *linha_vc4* e *coluna_vc4* durante um quadro VC-4 completo.



2.2.4 Adaptação de Rotas de Alta Ordem na Recepção (HPA_R)

O bloco HPA_R está detalhado na figura 2.29.

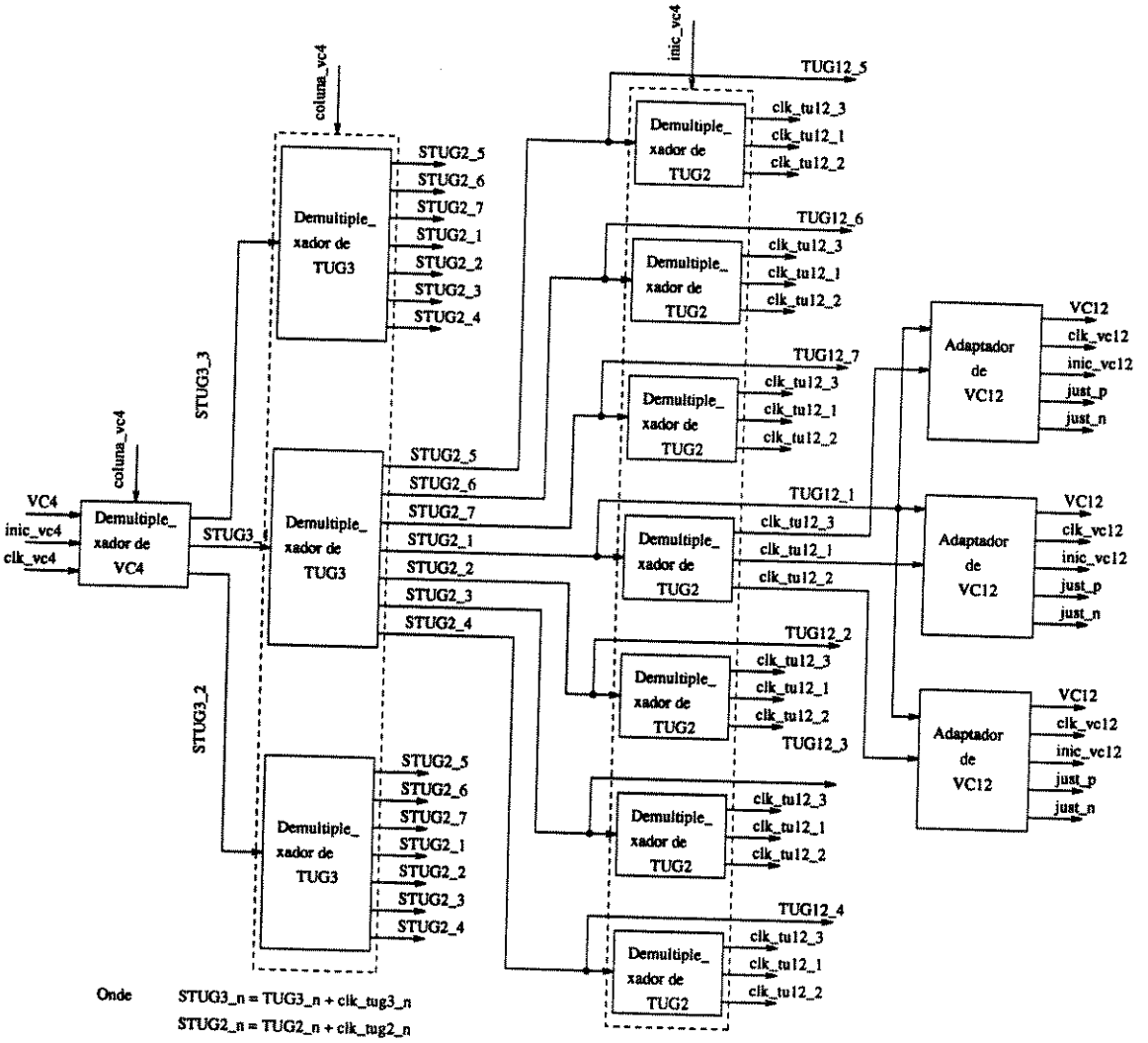


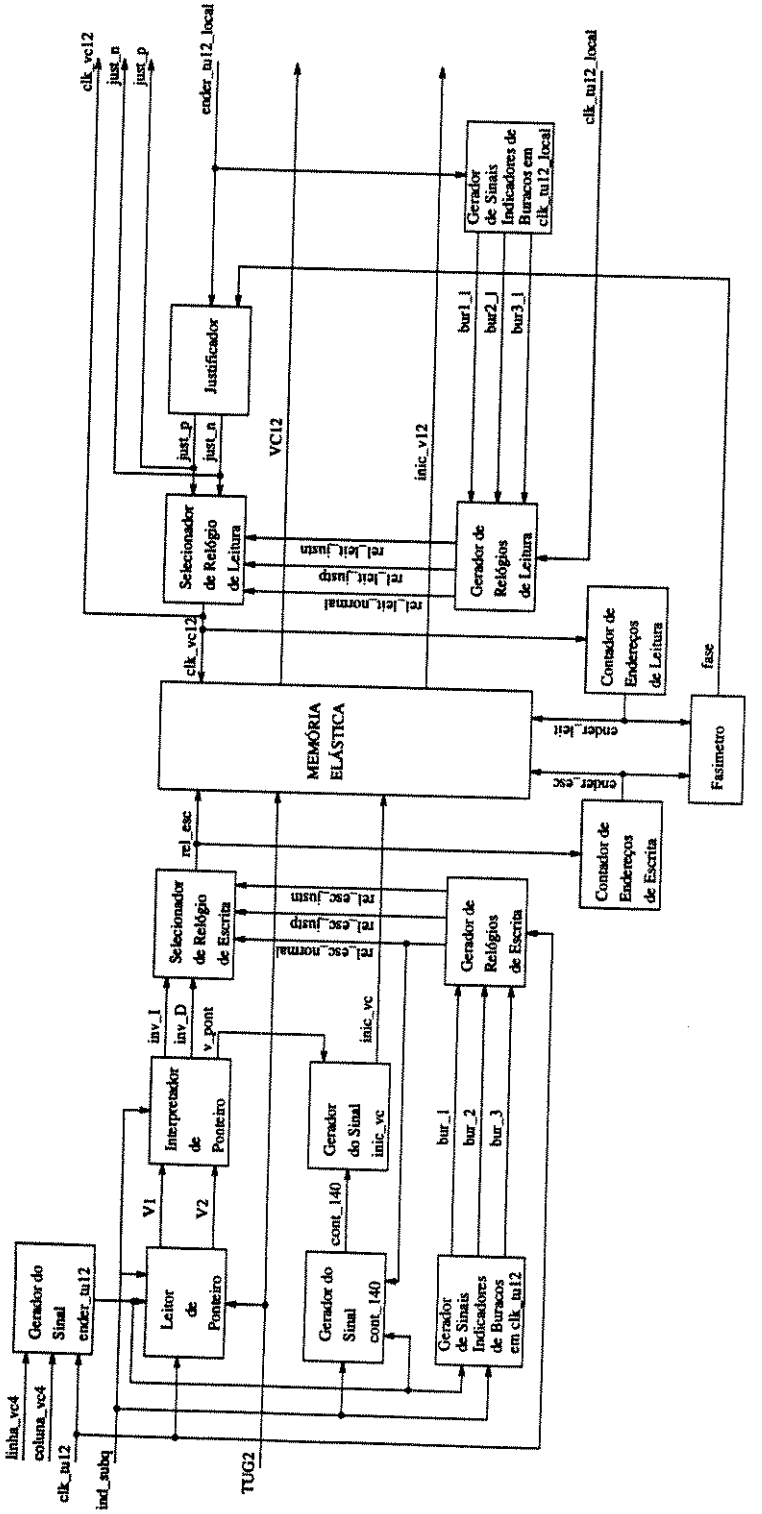
Figura 2.29 : Adaptação de Rotas de Alta Ordem na Recepção(HPA_R).

A figura 2.29 representa um bloco Demultiplexador de VC-4, 3 blocos Demultiplexadores de TUG-3, 21 blocos Demultiplexadores de TUG-2 e 63 blocos Adaptadores de VC-12. Deste bloco são retirados os 63 VC12.

Adaptador de VC-12 na Recepção

Este bloco está detalhado na figura 2.30. Este bloco faz a retirada do VC12 fazendo um processamento de ponteiro. Este processamento é semelhante ao bloco anterior que utilizou uma memória elástica para adaptar o sinal.

Figura 2.30 : Detalhamento do Bloco Adaptador de VC-12.



Interpretação do Ponteiro

Este bloco serve para retirar o valor do ponteiro de V1 e V2 respeitando as regras de interpretação de ponteiro vistas anteriormente. Com o valor do ponteiro sabe-se onde inicia o VC12. A figura 2.31 representa os sinais que saem deste bloco.

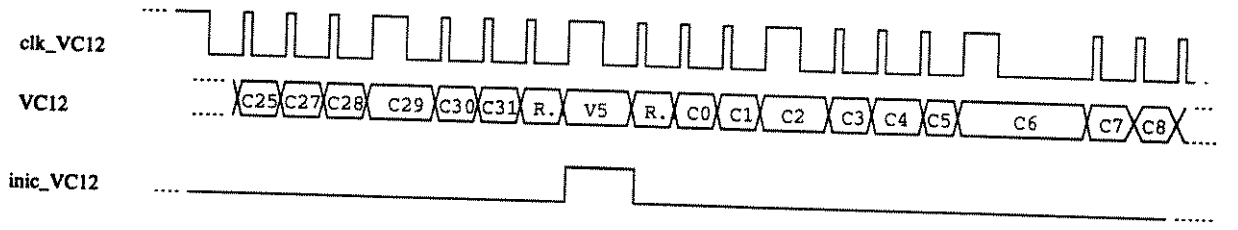
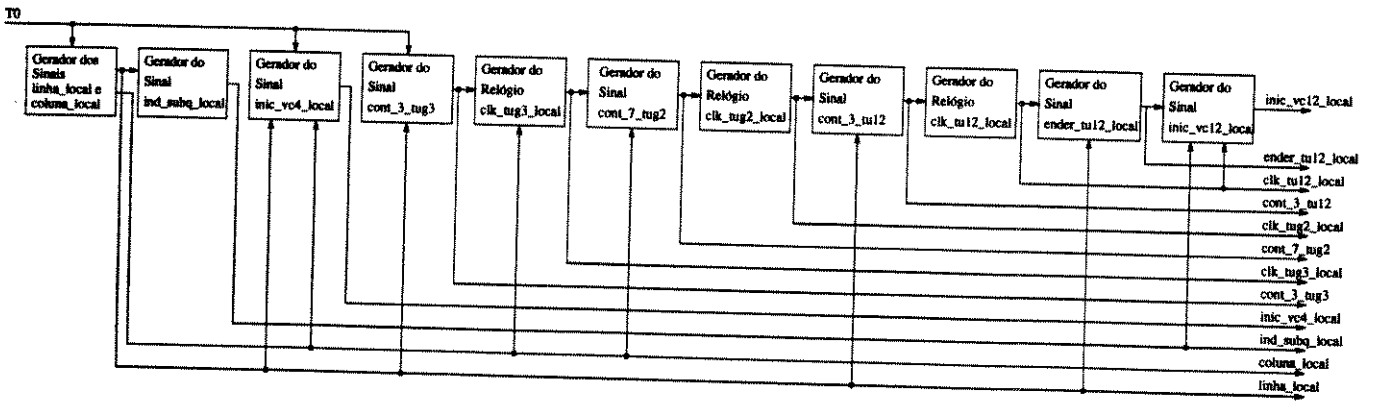


Figura 2.31 : Sinais de Saída do HPA_R.

2.2.5 Gerador dos Sinais de Controle Locais

A figura 2.32 apresenta o detalhamento deste bloco. Sua função é gerar sinais de controle e temporização para a montagem das estruturas a serem transmitidas. Observa-se que estes sinais são usados em todas as estruturas montadas.

Figura 2.32 : Detlhamento do Bloco Gerador dos sinais de Controle Locais.

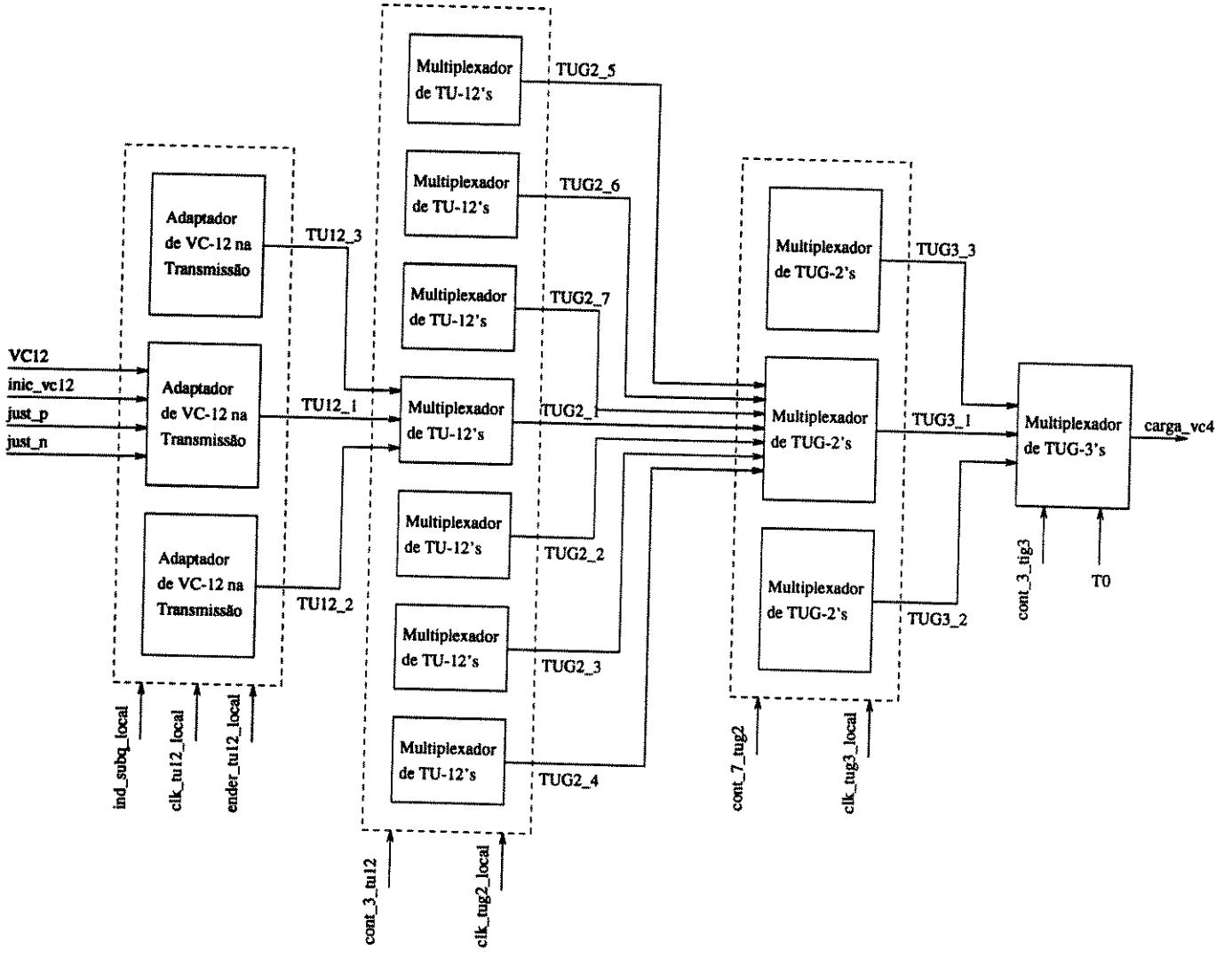


2.2.6 Adaptação de Rotas de Alta Ordem na Transmissão (HPA_T)

O bloco HPA_T está detalhado na figura 2.33.

A figura 2.33 representa um bloco Multiplexador de TUG-3's, 3 blocos Multiplexadores de TUG-2's, 21 blocos Multiplexadores de TU-12's e 63 blocos Adaptadores de VC-12 na Transmissão.

Figura 2.33 : Adaptação de Rotas de Alta Ordem na Transmissão(HPA-T)



Adaptador de VC-12 na Transmissão

O Adaptador de VC12 na Transmissão está detalhado através do diagrama de blocos na figura 2.34. Este bloco rescebe os sinais VC12 e inicio de VC12 para gerar um valor de ponteiro e seguir para a multiplexação nos próximos blocos.

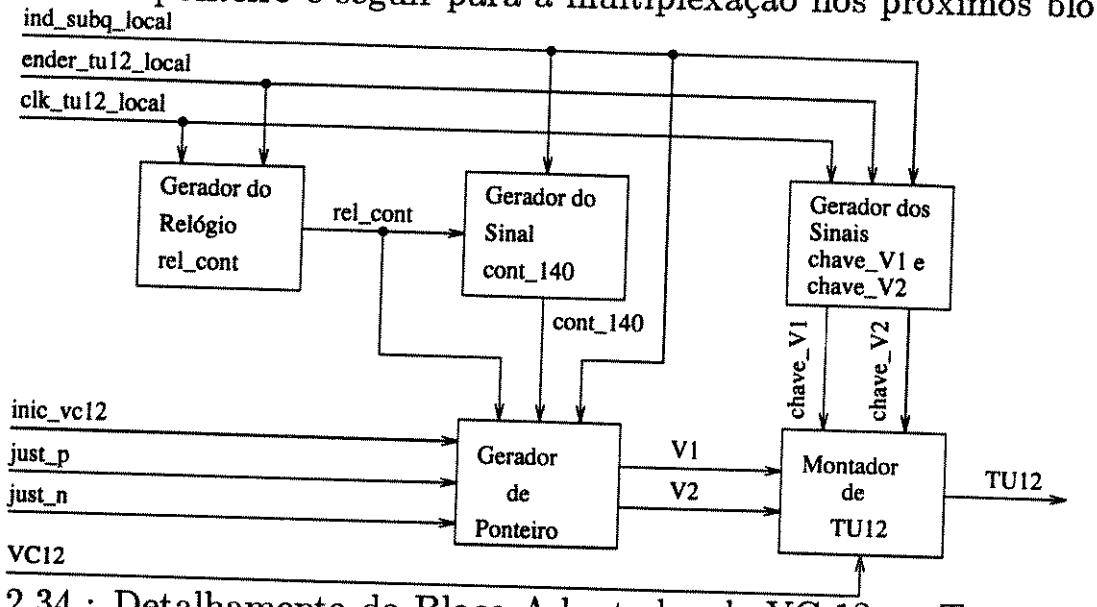


Figura 2.34 : Detalhamento do Bloco Adaptador de VC-12 na Transmissão.

2.2.7 Adaptação de Seção Multiplex na Transmissão (MSA_T)

A função de Adaptação de Seção Multiplex na Transmissão está detalhada através do diagrama de blocos na figura 2.35. Este bloco faz o inverso do bloco MSA_T, ou seja, apartir de um VC4 obtem-se um STM-1 existindo também um bloco gerador de ponteiro.

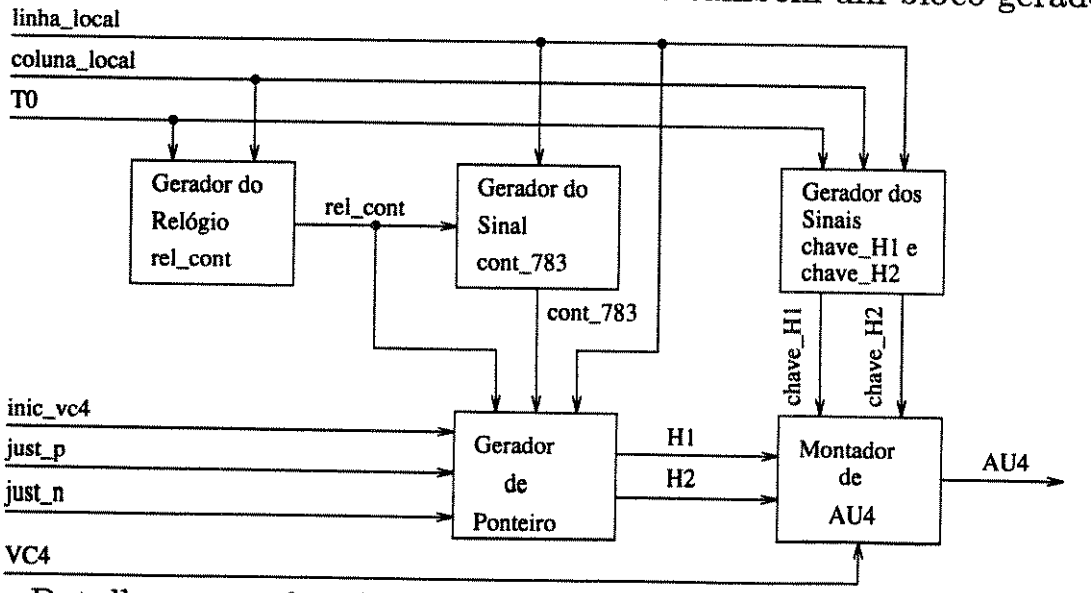


Figura 2.35 : Detalhamento do Bloco SA_T.

No final deste trabalho foram sugeridos alguns temas para desenvolvimentos futuros. Um destes temas foi o de desenvolvimento dos blocos de Desmapeamento/Mapeamento (LPA_R e LPA_T) . Neste próximo capítulo faz-se o desenvolvimentos de blocos funcionais com o objetivo de retirada de canais utilizando como base os Mapeamentos de baixa ordem descritos anteriormente.Mostraremos o desenvolvimento somente para sinai de 2Mb/s*(HDS-BRASIL)*. Apesar de termos desenvolvido blocos para sinais de 1.5Mb/s*(SONET-EUA)* estes blocos não serão mostrados.

Capítulo 3

Numeração de Afluentes nas Estruturas dos Módulos de Transporte da Rede Síncrona.

3.1 Introdução:

Este capítulo mostrará algumas formulações que possibilitam a localização de um canal DS0 dentro de um AU-4[10]. Analisando-se a hierarquia da estrutura *HDS(Hierarquia Digital Síncrona)* observa-se as várias possibilidades para "colocar" os mais diversos sinais para dentro desta estrutura. Este "colocar" sinais compreende as operações de *Mapeamento*, *Alinhamento*, *Multiplexação*, que são suas operações básicas. Um determinado canal quando é Mapeado para dentro da estrutura HDS, irá assumir uma posição dentro de uma TU-Unidade Tributária. Esta posição é possível de ser determinada. Com isto, pode-se gerar sinais específicos para a retirada destes canais. Convém salientar que isto era impossível de se conseguir na hierarquia plesiócrona, já que nesta ocorre um entrelaçamento de bits, diferente da HDS, onde temos entrelaçamento de bytes e ponteiros indicadores de início de carga útil.

3.2 Conceituação.

As formulações que aqui serão desenvolvidas referem-se a afluentes de 2 Mb/s e 1.5 Mb/s. Os sinais tributários que chegam a um equipamento

HDS pelas portas de 2 Mb/s ou 1.5 Mb/s são mapeados para dentro de um Container Virtual VC-12 ou VC-11. Este Container Virtual é alinhado dentro de uma Unidade Tributária TU-12 ou TU-11, esta Unidade Tributária é multiplexada dentro de um Grupo de Unidade Tributárias TUG-2. Este grupo de unidades tributárias é multiplexado em um grupo de unidades tributárias de maior porte TUG-3. O TUG-3 é multiplexado em um Container Virtual de grande porte VC-4 que é alinhado dentro de uma Unidade Administrativa AU-4. O AU-4 é colocado em um Grupo de Unidades Administrativas, que são multiplexadas para formar um sinal de grande porte e transmitido pela rede síncrona na forma de um Módulo de Transporte Síncrono STM-1. Na figura 3.1 mostra-se, de forma resumida, as operações básicas de mapeamento HDS. A figura 2.1 mostra a estrutura HDS completa.

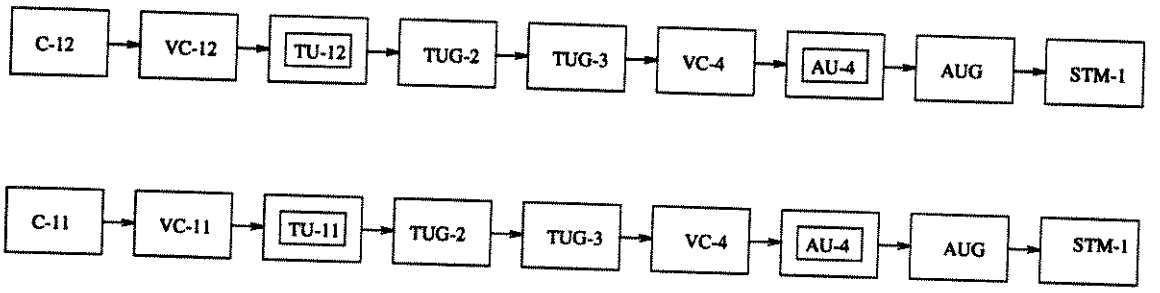


Figura 3.1: Estrutura para Transporte de VC-11 e VC-12

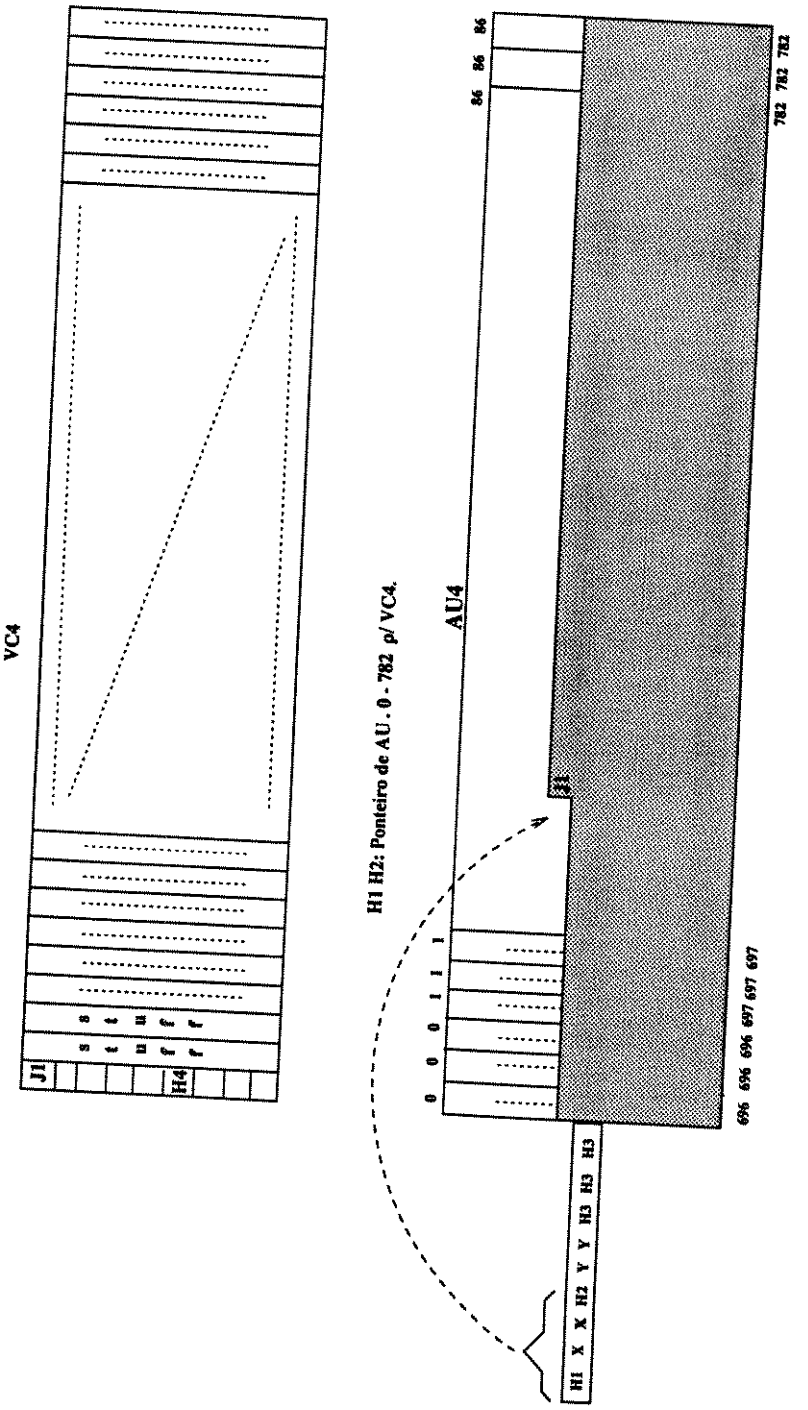
Multiplexação.

Um quadro STM-N é formado por N AUG onde, 1 AU-4 ≡1 AUG comporta somente 1 VC-4. Um quadro STM-1 pode ser representado por uma estrutura retangular de 270 colunas por 9 linhas, representando 270x9 bytes, que são transmitidos da esquerda para a direita, linha após linha. Tem-se então 2430 bytes em 125 μs, dando uma taxa de 155520 kb/s. Nesta estrutura STM-1, existem 9 colunas reservadas para SOH[1] e 261 que servem para acomodar a carga útil. As 261 colunas por 9 linhas formam a estrutura de um container virtual VC-4. Na figura 3.2 está a estrutura de um VC-4 *(Observe que a numeração é feita de* 3 *em* 3 *posições).* Então: 261 x 9 = 2349 bytes em 125 μs, que corresponde a

[1]SOH-Section OverHead

uma taxa de 150.336 kb/s. Dentro de um VC-4, reserva-se uma coluna para POH[2], com seu primeiro byte J1 definindo o início do VC-4.

Figura 3.2 : Alinhamento de um VC-4 no AU-4.



[2]POH-Path OverHead

Dentro de um VC-4 são multiplexados 3 TUG-3, onde cada TUG-3 é uma estrutura de 86 colunas por 9 linhas. Observa-se que 86 x 3 não dá as 261 colunas. Para se conseguir isto, tem-se mais 2 colunas de enchimento, além da de POH. Ver figura 2.5.

3 TUG-3 dentro de um VC-4:

$$\begin{array}{lrlrl} 3TUG-3 & \Leftrightarrow 3\times 86 & colunas & = 258 & colunas; \\ POH\,do\,VC-4 & - & - & = 1 & coluna; \\ bytes & de & enchimento & = 2 & colunas; \\ \textbf{TOTAL} & \Leftrightarrow & VC-4 & = 261 & colunas. \end{array}$$

- *Cada* **TUG-3** *pode transportar uma carga de até* **86x9x64***kb/s* = **49536** *kb/s.*

Dentro de um TUG-3 são multiplexados 7 TUG-2, onde cada TUG-2 é uma estrutura de 12 colunas por 9 linhas. Observe que **12 x 7 = 84** colunas ocupadas pelos *TUG-2* dentro de um *TUG-3.* Então é preciso incluir 2 colunas de enchimento para cada TUG-2. (observar a figura 3.3).

7 TUG-2 dentro de um TUG-3:

$$\begin{array}{rrlrl} 7TUG-2 \Leftrightarrow 7\times 12 & colunas & = 84 & colunas; \\ bytes & enchimento & = 2 & colunas; \\ \textbf{TOTAL} \Leftrightarrow & TUG-3 & = 86 & colunas. \end{array}$$

A multiplexação de 7 **TUG-2** *em um* **TUG-3** *é mostrada na* figura 3.3.

- *Cada* **TUG-2** *pode transportar uma carga de até* **12x9x64***kb/s* = **6912** kb/s.

Dentro de um TUG-2 *(que possui 12 colunas)* são multiplexados 3 TU-12, onde cada TU-12 é uma estrutura composta de 4 colunas por 9 linhas. Então, um TUG-2 é completamente preenchido por 3 TU-12, sem a necessidade de bits de *enchimento*, observar a figura 3.4.

- *3 TU-12 dentro de um TUG-2:*

$$3TU-12 \Leftrightarrow 3 \times 4 \quad colunas \quad = \quad 12 \quad colunas;$$
$$\textbf{TOTAL} \qquad \Leftrightarrow \quad TUG-2 \quad = \quad 12 \quad colunas;$$

A *figura 3.4 mostra a multiplexação de 3* **TU-12** *em um* **TUG-2**.

- *Cada* **TU-12** *pode transportar uma carga de até* $\mathbf{4x9x64}kb/s = \mathbf{2304}kb/s$.

Figura 3.3 : Multiplexação de 7 TUG-2 em TUG-3.

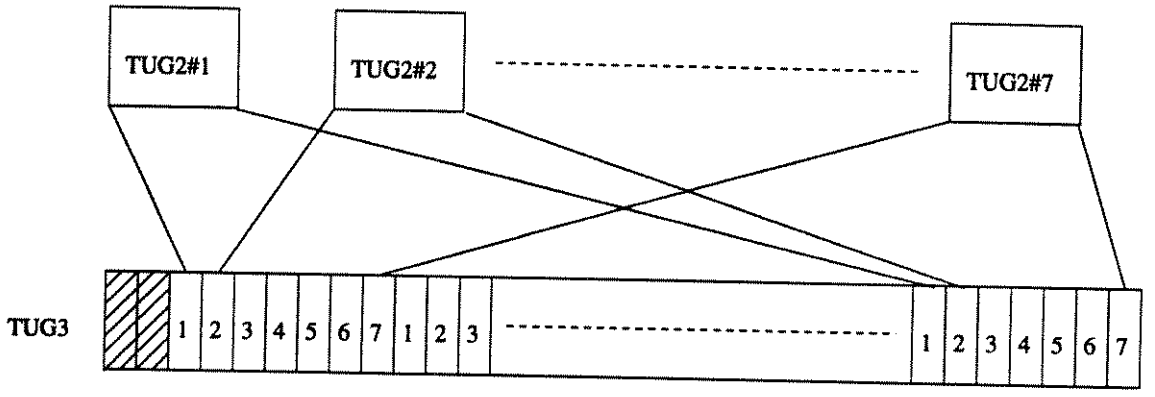
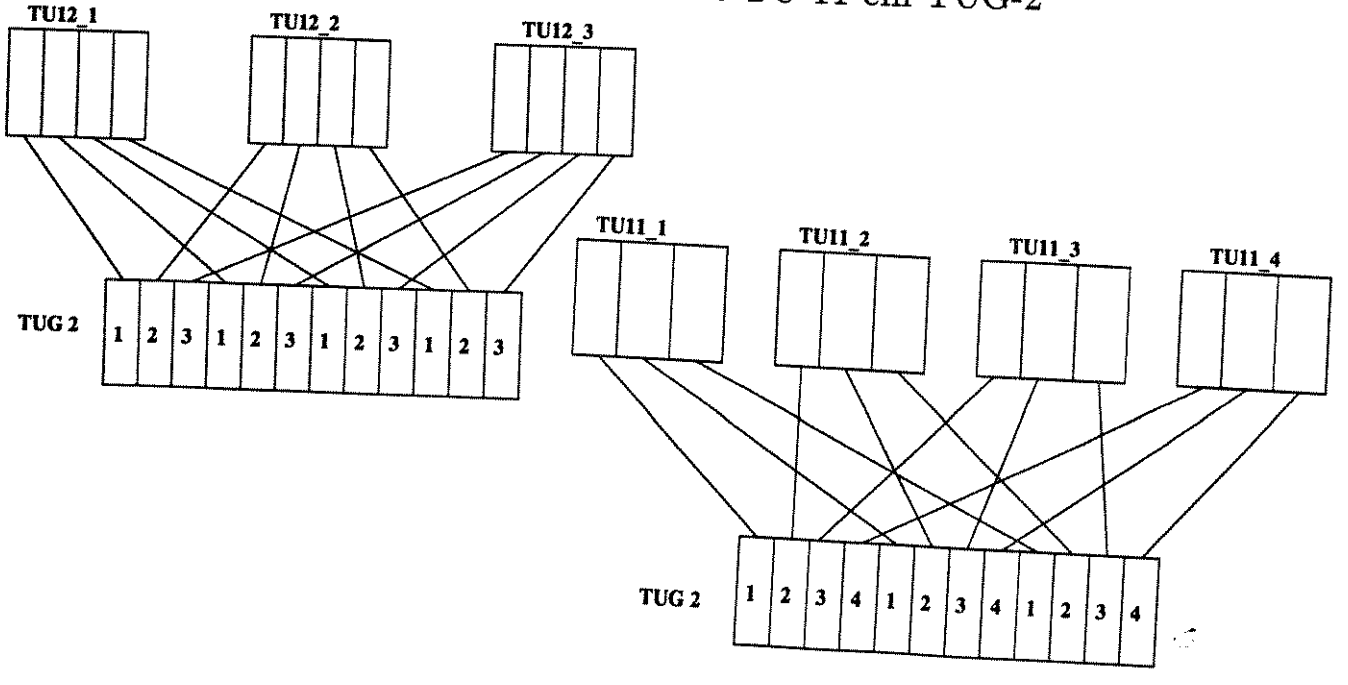


Figura 3.4 : Multiplexação de 3 TU-12 e TU-11 em TUG-2



Dentro de um TUG-2 são multiplexados 4 TU-11, onde cada TU-11 é uma estrutura composta de 3 colunas por 9 linhas. Então, um TUG-2 é completamente preenchido por 4 TU-11, sem a necessidade de bits de *enchimento*, observar a figura 3.4.

4 TU-11 dentro de um TUG-2:

$$\begin{aligned} 4TU-11 &\Leftrightarrow 4\times 3 \quad colunas &= 12 \quad colunas; \\ \textbf{TOTAL} &\Leftrightarrow TUG-2 &= 12 \quad colunas; \end{aligned}$$

- *Cada* **TU-11** *pode transportar uma carga de até* $\mathbf{3x9x64}kb/s = \mathbf{1728}kb/s$.

Mapeamento.

Para se mapear um container VC-12 dentro da estrutura de uma Unidade Tributária TU-12 *(que é fixa em relação a estrutura do VC-4)*, monta-se um multiquadro de 4 quadros de 125 μs, como mostrado na figura 3.5, com as seguintes alocações de bytes:

- Bytes do TU-12:

  - *Ponteiro de TU :bytes* **V1** *e* **V2.**
  - *Oportunidade de Justificação Negativa: byte* **V3**
  - *Reserva de TU: byte* **V4**

  TOTAL⇔ 4 bytes.

- Bytes do VC-12:

  - *POH: byte* **V5.**
  - *Identificação de Rota :byte* **J2.**
  - *Bytes reservados:* **Z6** *e* **Z7.**

  TOTAL⇔ 4 bytes.

- *Máxima Taxa de Bytes para Carga Útil:*

$$(4 \times 36) - 4 - 4 \Leftrightarrow 34 \times 4 \times 16kb/s = \mathbf{2176}kb/s.$$

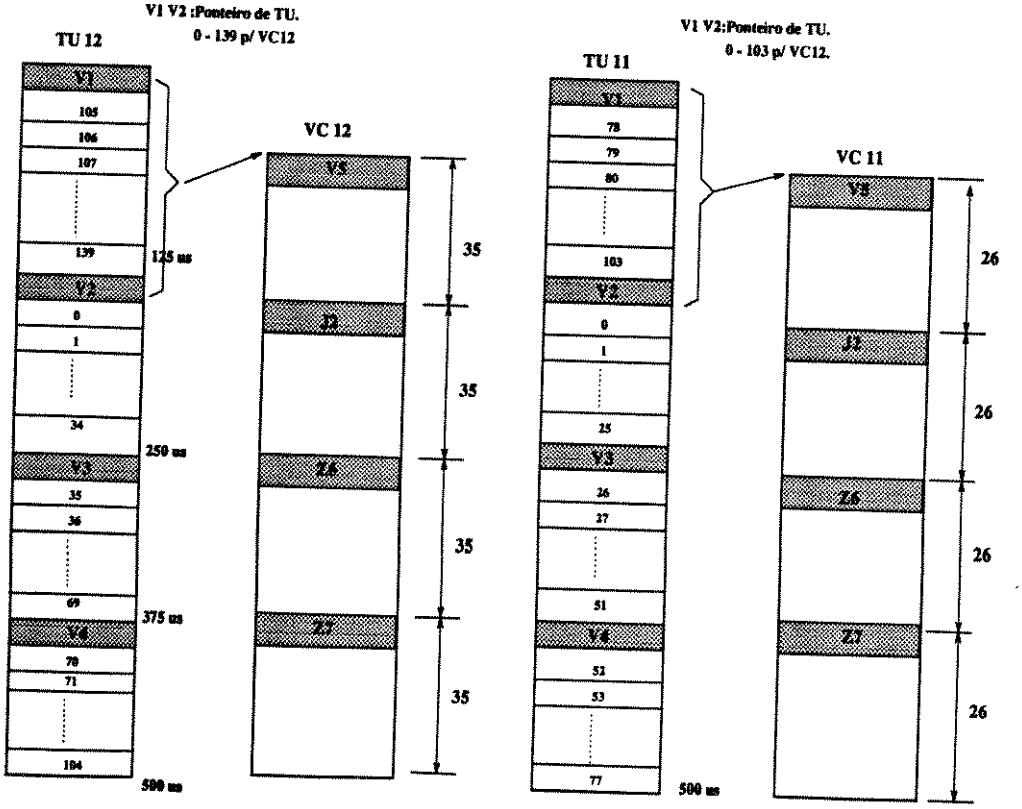
Para se mapear um container VC-11 dentro da estrutura de uma Unidade Tributária TU-11 *(que é fixa em relação a estrutura do VC-4)*, monta-se um multiquadro de 4 quadros de 125 μs, como mostrado na figura 3.5.

A alocação dos bytes V1, V2, V3 e V4 é a mesma descrita para o TU-12; idem para os bytes de POH de VC-12.Figura 3.5.

- *Máxima taxa de Bytes para Carga Útil:*

$$4 \times 27 - 4 - 4 \Leftrightarrow 25 \times 4 \times 16kb/s = \mathbf{1600}kb/s.$$

Figura 3.5 : Container Virtual Mapeado em uma Unidade Tributária.



Alinhamento.

A posição que será ocupada por um determinado container dentro de uma Unidade Tributária (**TU**) ou de uma Unidade Administrativa (**AU**) é determinada através da operação de alinhamento, que possibilita uma alocação dinâmica da carga útil, através de justificação de ponteiro. A figura 3.2 ilustra, através da interpretação do ponteiro *(bytes H1 e H2)*, a posição que a carga útil VC-4 irá ocupar dentro do AU-4. Existem 783 posições de bytes dentro de um AU-4. A figura 3.5, da mesma forma, mostra que através da interpretação do poteiro *(V1 e V2)*, a carga útil VC-12 pode ocupar 140 posições diferentes dentro de um TU-12. No caso do VC-11, existem 104 posições dentro de um TU-11. Na figura 3.6 são mostrados os mapeamentos de sinais de 2048 kb/s e de 1544 kb/s na forma síncrona nos containers VC-12 e VC-11, respectivamente, como recomendados pelo **ITU - TSS**. É para estes mapeamentos que as equações serão deduzidas:

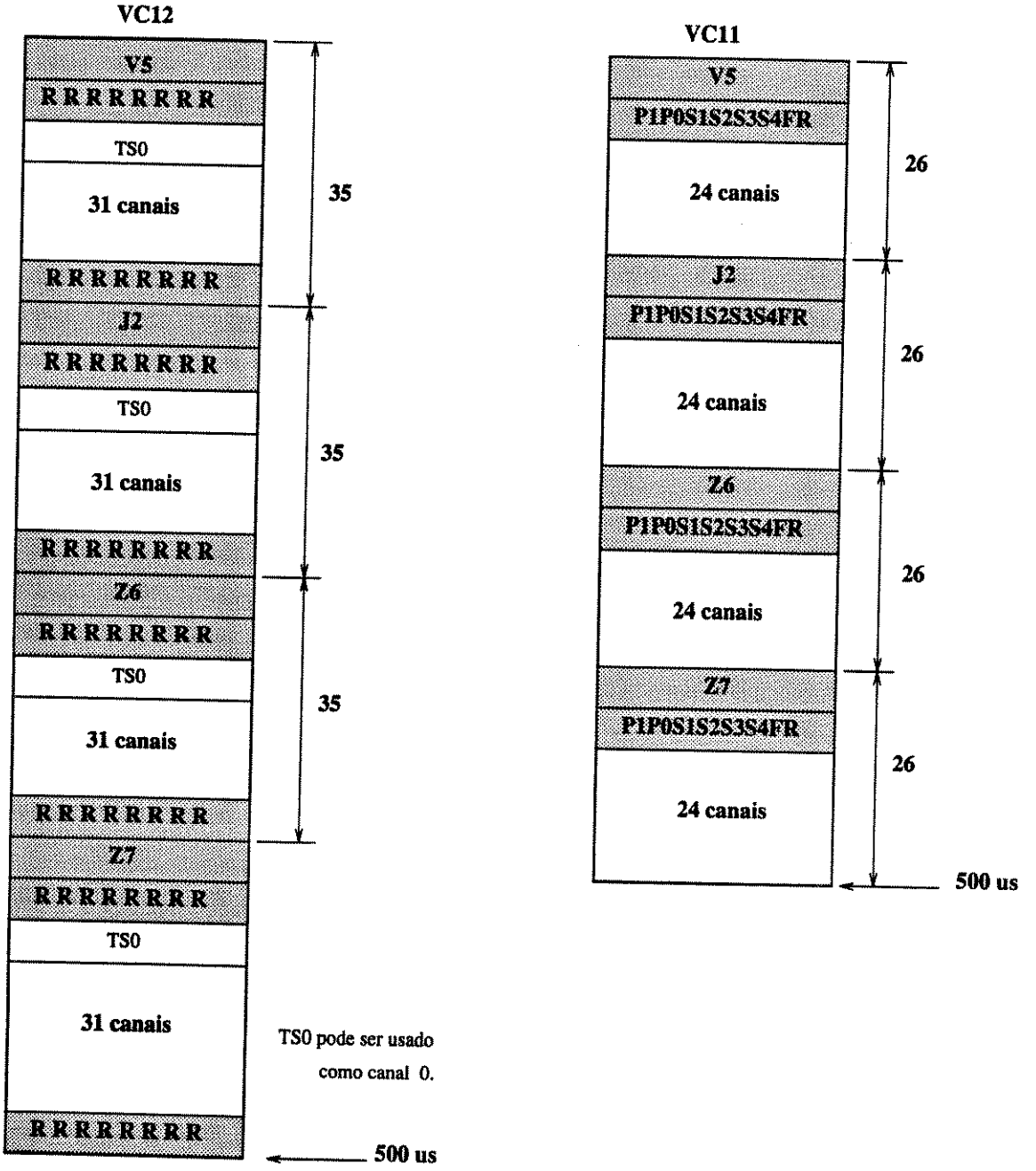
VC-12:

- *mapeamento de 32 canais de 64 kb/s (2048 kb/s).* **(pode ir até 34 canais de 64 kb/s).**

VC-11:

- *mapeamento de 24 canais de 64 kb/s (1536 ks/s) com canal adicional de 64 kb/s para sinalização de linha e sincronização.***(Máximo de 25 canais de 64 kb/s).**

Figura 3.6 : Mapeamento Síncrono de Byte para 2048 kb/s(VC-12) e 1544 kb/s(VC-11).



3.3 Formulação para Tributários de 2Mb/s

Neste trabalho descreve-se as posições que um determinado canal **n** de 64 kb/s poderá ocupar dentro de um AU-4. Com isto, pode-se determinar onde está um canal dentro deste quadro, possibilitando inserção/derivação/processamento de qualquer canal de forma direta.

Como foi visto anteriormente, 3 TU-12 são multiplexados dentro de 1 TUG-2 e 7 TUG-2 dentro de 1 TUG-3. Então dentro de cada TUG-3 tem-se 21 TU-12 multiplexados. Como dentro de 1 VC-4 são multiplexados 3 TUG-3, a capacidade de um VC-4 é de **63** TU-12. Dentro de um TU-12 tem-se um VC-12 e neste estão colocados vários canais. Para se chegar a uma formulação que possibilite a localização de um determinado canal, leva-se em conta os seguintes aspectos:

- *O canal pode estar dentro de um dos* **3** *TUG-3: Corresponde a variável* **k**.

  k pode assumir valores de *0,1,2.* $\Leftrightarrow$ TUG-3#0, TUG-3#1,TUG-3#2.

- *Escolhido o TUG-3#k, o canal pode estar dentro de um dos* **7** *TUG-2. A variável ℓ será usada para o TUG-2.*

  ℓ pode assumir valores de *0,1,2,3,5,6*$\Leftrightarrow$ TUG-2#0,TUG-2#1,..,TUG-2#6.

- *Escolhidos* TUG-3# k *e* TUG-2#ℓ, *o canal pode estar dentro de um dos* **3** *TU-12. A variável* **m** *será usada para o TU-12.*

  m pode assumir valores de *0,1,2*$\Leftrightarrow$ TU-12#0,TU-12#1,TU-12#2.

- *Valor de coluna ocupado pela carga útil* $\Rightarrow$ *variável* **c**.

  c terá valores de *0,1,..11* para TUG-2; terá valores de *0,1,2,3* para TU-12 e terá valores de *0,1,2* para TU-11.

- *Valor da linha dentro de um quadro* $\Rightarrow$ *variável* **r**.

r pode assumir valores de *0,1,2,..,8.*

- *Valor do ponteiro de TU-12* $\Rightarrow$ *Variável* $\mathbf{p_{v5}}$.

$\mathbf{p_{v5}}$ pode assumir valores de *0,..,139.*

- *Valor do ponteiro de VC-4* $\Rightarrow$ *Variável* p_{J1}.

p_{J1} pode assumir valores de *0,..,782.*

3.3.1 Numeração de Coluna para *TUG-2.*

O desenvolvimento que segue é feito para VC-4 que inicia na posição "ZERO" do ponteiro de AU-4. A figura 3.7 ajudará no desenvolvimento das equações que se seguem. Para localizar o 1° byte de um TUG-2, que é onde inicia a carga útil, precisa-se inicialmente descontar os bytes de POH e de STUFF(enchimento). Na figura 3.7, existem dois tipos de numeração em um AU-4: Uma numeração com contagem de 3 em 3 *(000,111,222,...,868686)* e uma numeração com contagem de 1 em 1 *(0,1,2,3,..,260).* No primeiro tipo, não há como definir a qual dos 3 TUG-3 pertenceria um TUG-2. No segundo tipo define-se a numeração com um número para cada posição. As equações serão desenvolvidas considerando os dois tipos de numeração.

- a) *Numeração de 3 em 3 no AU-4:*

Observe que a carga útil inicia na posição **3**, pois as colunas anteriores não trazem informação (carga útil). Vai-se determinar as posições ocupadas pelo 1° TUG-2 dentro do 1° TUG-3: O terceiro byte do 1° TUG-3 está ocupado pelo primeiro byte do 1° TUG-2, que corresponde a posição **3** do AU-4; o quarto byte do 1° TUG-3 está ocupado pelo segundo *TUG-2*, que corresponde a posição **4** do AU-4; o quinto byte do 1° TUG-3 está ocupado pelo terceiro TUG-2, que corresponde a posição **5** do *AU-4*, etc.. Seguindo este raciocínio, depois que os **7** TUG-2 ocuparem o *TUG-3* com seus primeiros bytes, o 1° *TUG-2* com seu segundo byte está na posição **10** do AU-4. Observe então que a distâcia é de **7** entre o 1° e o 2° byte do 1° TUG-2. Como um TUG-2 possui 12 colunas em uma linha, então

tem-se esta constante **7** multiplicando a variável coluna (**c**). Variando **c**, tem-se as **12** posições ocupadas por um TUG-2 em uma mesma linha (**r**). Para saber as posições ocupadas pelos demais TUG-2 basta introduzir no somatório a variável ℓ. Através do incremento de **r**, calcula-se as 108 posições ocupadas pelo TUG-2 em um TUG-3. Portanto a posição de um **TUG-2** de órdem ℓ em um AU-4, contando de **3** em **3**, é dada por:

$$\mathbf{P_{TUG-2}}(\ell, \mathbf{c}, \mathbf{r}) = \mathbf{3} + \ell + \mathbf{7}\times\mathbf{c} + \mathbf{87}\times\mathbf{r} \tag{3.1}$$

$$P_{TUG-2} : 108 valores$$
$$c : 12 valores$$
$$r : 9 valores$$

O valor 87 vem da contagem de 3 em 3 nos 261 bytes de uma linha.

- b)*Numeração de 1 em 1 no AU-4*:

Com esta numeração distingue-se a qual TUG-3 pertence um TUG-2. Descontando as posições de POH e de STUFF, a carga útil inicia na posição 9. A variável **k** distingue qual é o TUG-3. A distância entre um TUG-2 e outro é de **3** *(1° TUG-2 na posição 9 e 2° TUG-2 na posição 12)*. Esta distância de 3 separa todos os TUG-2 dentro de um mesmo TUG-3. Do produto desta constante 3 pela variável ℓ, tem-se as posições ocupadas pelos **7** TUG-2. Na figura 3.7 a distância entre os bytes de um mesmo TUG-2 é de **21**. Do produto da constante **21** pela variável **c** *(colunas)* tem-se as posições ocupadas por um mesmo TUG-2#ℓ em uma mesma *linha.*

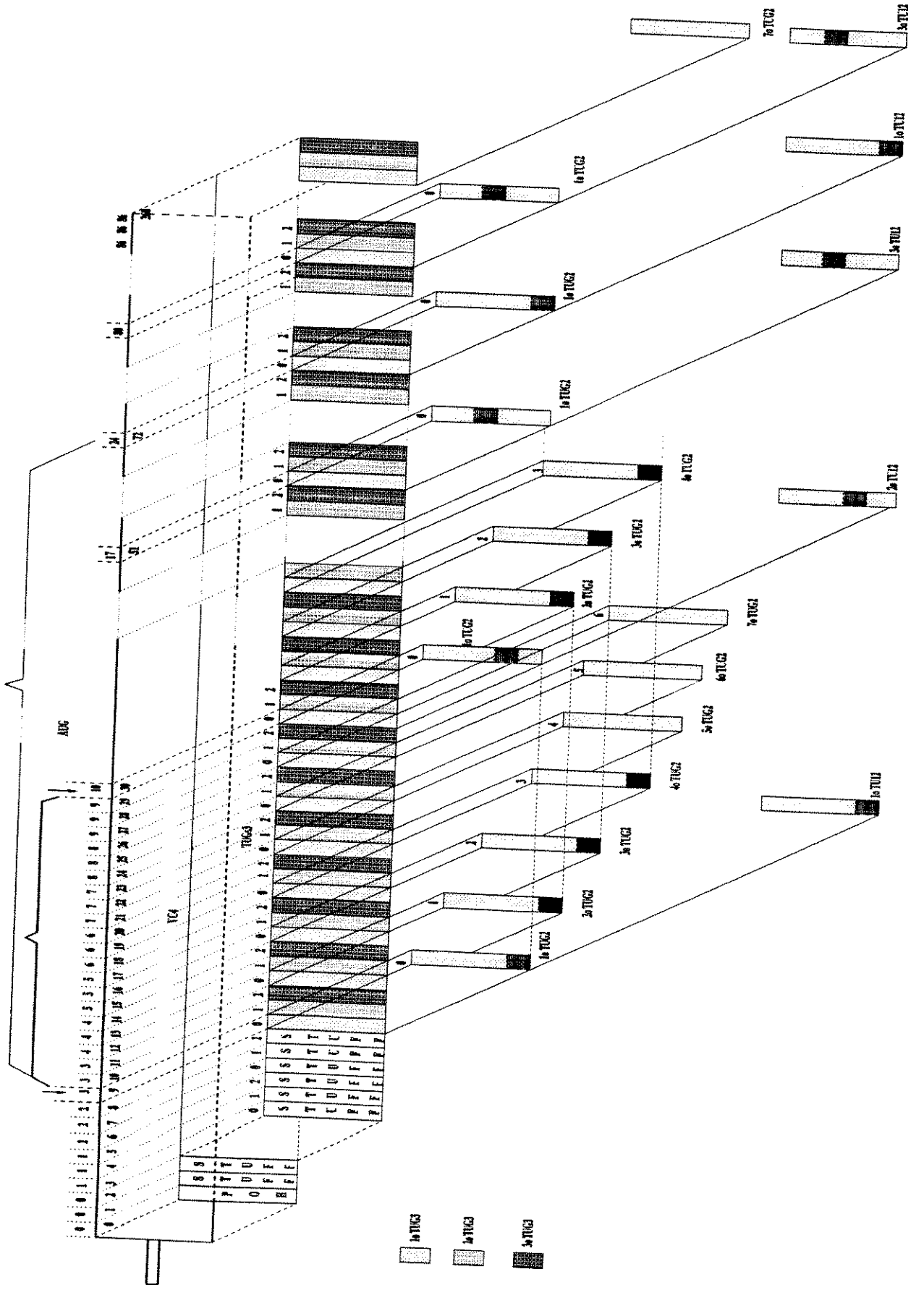
$$\mathbf{P_{TUG-2}}(\mathbf{k}, \ell, \mathbf{c}) = \mathbf{9} + \mathbf{k} + \mathbf{3}\times\ell + \mathbf{21}\times\mathbf{c} \tag{3.2}$$

A *equação*-**3.2** informa a coluna do *VC-4* que carrega **TUG-2-ℓ** dentro do **TUG-3-k**, isto é:

$p_{TUG-2}(k,\ell,c)\in[9,260]$.

Na equação-**3.1** *não há distinção entre* TUG-3# 0,1,2.

Figura 3.7 : Posicionamento dos TU-12 dentro de um AU-4 com Ponteiro de VC-4 em ZERO.



A *equação*-**3.1** calcula as posições ocupadas por um TUG-2 dentro de um AU-4. Estas posições estão na Tabela 1 que mostra as 108 posições ocupadas pelo TUG-2#0 dentro de um TUG-3(existem mais 6 tabelas de numeração para um mesmo TUG-3). Até aqui não foi introduzido o valor do ponteiro, isto é, o valor deste é "*ZERO*",ou seja, o **J1** do VC-4 está na posição zero do AU-4. A *equação*-**3.2**, fornece todas as posições dentro de uma linha, ou seja, informa a coluna do VC-4 que carrega o TUG-2#ℓ dentro do TUG-3#**k**.

TABELA 1-Posições Ocupadas pelo TUG-2#0- Equação-3.1(Ponteiro nulo).

r c	0	1	2	3	4	5	6	7	8	9	10	11
0	3	10	17	24	31	38	45	52	59	66	73	80
1	90	97	104	111	118	125	132	139	146	153	160	167
2	177	184	191	198	205	212	219	226	233	240	247	254
3	264	271	278	285	292	299	306	313	320	327	334	341
4	351	358	365	372	379	386	393	400	407	414	421	428
5	438	445	452	459	466	473	480	487	494	501	508	515
6	525	532	539	546	553	560	567	574	581	588	595	602
7	612	619	626	633	640	647	654	661	668	675	682	689
8	699	704	713	720	727	734	741	748	755	762	769	776

A Tabela 2 mostra todas as posições ocupadas pelos **21** TUG-2 dentro dos 3 TUG-3, observando que esta numeração é para uma mesma linha **r**.

TABELA 2-Colunas Ocupadas pelos 21 TUG-2-Equação-3.2(ponteiro nulo)

k	**ℓ c**	0	1	2	3	4	5	6	7	8	9	10	11
0	0	9	30	51	72	93	114	135	156	177	198	219	240
0	1	12	33	54	75	96	117	138	159	180	201	222	243
0	2	15	36	57	78	99	120	141	162	183	204	225	246
0	3	18	39	60	81	102	123	144	165	186	207	228	249
0	4	21	42	63	84	105	126	147	168	189	210	231	252
0	5	24	45	66	87	108	129	150	171	192	213	234	255
0	6	27	48	69	90	111	132	153	174	195	216	237	258
1	0	10	31	52	73	94	115	136	157	178	199	220	241
1	1	13	34	55	76	97	118	139	160	181	202	223	244
1	2	16	37	58	79	100	121	142	163	184	205	226	247
1	3	19	40	61	82	103	124	145	166	187	208	229	250
1	4	22	43	64	85	106	127	148	169	190	211	232	253
1	5	25	46	67	88	109	130	151	172	193	214	235	256
1	6	28	49	70	91	112	133	154	175	196	217	238	259
2	0	11	32	53	74	95	116	137	158	179	200	221	242
2	1	14	35	56	77	98	119	140	161	182	203	224	245
2	2	17	38	59	80	101	122	143	164	185	206	227	248
2	3	20	41	62	83	104	125	146	167	188	209	230	251
2	4	23	44	65	86	107	128	149	170	191	212	233	254
2	5	26	47	68	89	110	131	152	173	194	215	236	257
2	6	29	50	71	92	113	134	155	176	197	218	239	260

obs1: *Observar que na equação 3.1 temos a previsão para:*

$$\begin{aligned} 9 \times 12 &= 108 \Leftrightarrow 1 \quad \mathbf{TUG-2} \\ 108 \times 7 &= 756 \Leftrightarrow 7 \quad \mathbf{TUG-2} \\ 756 \times 3 &= 2268 \Leftrightarrow 21 \quad \mathbf{TUG-2} \end{aligned}$$

obs2: *Por outro lado, na equação 3.2 temos:*

$$21 \times 12 \quad = 252 bytes \Leftrightarrow em \ \mathbf{1} \ linha$$
$$9 \times 252 \ = 2268 bytes \Leftrightarrow em \ \mathbf{9} \ linhas$$

Na obtenção da formulação os bytes de POH,STUFF... *foram descontados. Então, somando estes, tem-se:***9 x 9 + 2268 = 2349** *bytes, que dá a carga bruta do* VC-4.

3.3.2 Numeração de coluna para *TU-12.*

- a) *Numeração de 3 em 3 no AU-4.*

Dentro do 1º TUG-2, correspondendo a posição 3 do VC-4, está o 1º TU-12. O 2º TU-12 está a uma distância de **7** do 1º TU-12, ou seja, na posição **10** do AU-4, e o 3º TU-12 está na posição **17** do AU-4 *(correspondendo sempre ao mesmo TUG-2 - neste caso 1º TUG-2).* Tem-se então, uma constante de valor 7 que incrementa a variável **m** (TU-12), diferenciando qual o TU-12. O 2º byte do 1º TU-12 está a uma distância de **21** *(o 1º byte do 1º TU-12 na posição 3 e o 2º byte do mesmo TU-12 na posição 24).* Estabelecendo a constante 21 para incrementar a variável **c**, modifica-se então a equação-3.1 para determinar as **36** posições ocupadas por um TU-12 dentro de um AU-4.

$$\mathbf{P_{TU-12}(\ell, m, c, r) = [3 + \ell + 7 \times m + 21 \times c + 87 \times r]} \tag{3.3}$$

- b) *Numeração de 1 em 1 no AU-4.*

A distância entre o 1º TU-12 e o 2º é de **21** *(1º TU-12 na posição* **9** *e 2º TU-12 na posição* **30***).* A distância entre o 1º byte do 1º *TU-12* e o 2º byte do mesmo TU-12 é de **63** *(1º byte na posição* **9** *do AU-4 e o 2º byte na posição* **72** *do AU-4).* Definidas estas constantes, a equação-3.2 é transformada para:

$$\mathbf{P_{TU-12}(k, \ell, m, c) = [9 + k + 3 \times \ell + 21 \times m + 63 \times c]} \tag{3.4}$$

$$p_{TU-12} : 36 valores$$
$$k : 0, 1, 2$$
$$\ell : \mathbf{0,\ldots,6}$$
$$m : 0, 1, 2$$
$$c : 0, 1, 2, 3$$

A variável **c** *corresponde à coluna do* TU-12. *São* **4** *posições ocupadas por um* TU-12 *em uma mesma linha.*

- Exemplo de aplicação da *equação*-**3.3**:

(ℓ,m): **(0,0)**; isto é, 1º *TU-12* dentro do 1º *TUG-2.*

c: Varia de **0,..,3** e incrementa a variável **r**.

Dado um TU-12 (ℓ,m) dentro de qualquer TUG-3 então fazemos r=0 e achamos as posições no VC-4 das colunas c=0,c=1,c=2,c=3 do TU-12; repetimos para r=1,2,..,8. Teremos portanto as **36** *posições ocupadas por este TU-12, mostradas na Tabela 3.*

Podemos observar na Tabela 3 que existem algumas posições marcadas.Estas são as posições ocupadas por estes *TU-12* do exemplo acima.

TABELA 3-Posições Ocupadas TU-12#0TUG2#0-Equação-3.3(Ponteiro nulo).

c	0	0	0	1	1	1	2	2	2	3	3	3
r m	0	1	2	0	1	2	0	1	2	0	1	2
0	3	10	17	24	31	38	45	52	59	66	73	80
1	90	97	104	111	118	125	132	139	146	153	160	167
2	177	184	191	198	205	212	219	226	233	240	247	254
3	264	271	278	285	292	299	306	313	320	327	334	341
4	351	358	365	372	379	386	393	400	407	414	421	428
5	438	445	452	459	466	473	480	487	494	501	508	515
6	525	532	539	546	553	560	567	574	581	588	595	602
7	612	619	626	633	640	647	654	661	668	675	682	689
8	699	704	713	720	727	734	741	748	755	762	769	776

Exemplo de aplicação da *equação*--3.4:

(k,ℓ,m)= **(0,0,0)**; isto é, colunas ocupadas pelo 1º *TU-12* dentro do 1º *TUG-2* dentro do 1º *TUG-3.*

São os valores: **9,72,135** e **198** mostrados na Tabela 4.

(k,ℓ,m)= **(2,6,2)** ; isto é, colunas ocupadas pelo 3º *TU-12* dentro do 7º *TUG-2* dentro do 3º *TUG-3.*

São os valores:**71,134,197** e **260** mostrados na Tabela 4.

TABELA 4-Colunas Ocupadas pelos 63 TU-12 em um AU 4-Equação-3.4 .

m	-	0	1	2	0	1	2	0	1	2	0	1	2
c	-	0	1	2	3	4	5	6	7	8	9	10	11
k	ℓ	-	-	-	-	-	-	-	-	-	-	-	-
0	0	9	30	51	72	93	114	135	156	177	198	219	240
0	1	12	33	54	75	96	117	138	159	180	201	222	243
0	2	15	36	57	78	99	120	141	162	183	204	225	246
0	3	18	39	60	81	102	123	144	165	186	207	228	249
0	4	21	42	63	84	105	126	147	168	189	210	231	252
0	5	24	45	66	87	108	129	150	171	192	213	234	255
0	6	27	48	69	90	111	132	153	174	195	216	237	258
1	0	10	31	52	73	94	115	136	157	178	199	220	241
1	1	13	34	55	76	97	118	139	160	181	202	223	244
1	2	16	37	58	79	100	121	142	163	184	205	226	247
1	3	19	40	61	82	103	124	145	166	187	208	229	250
1	4	22	43	64	85	106	127	148	169	190	211	232	253
1	5	25	46	67	88	109	130	151	172	193	214	235	256
1	6	28	49	70	91	112	133	154	175	196	217	238	259
2	0	11	32	53	74	95	116	137	158	179	200	221	242
2	1	14	35	56	77	98	119	140	161	182	203	224	245
2	2	17	38	59	80	101	122	143	164	185	206	227	248
2	3	20	41	62	83	104	125	146	167	188	209	230	251
2	4	23	44	65	86	107	128	149	170	191	212	233	254
2	5	26	47	68	89	110	131	152	173	194	215	236	257
2	6	29	50	71	92	113	134	155	176	197	218	239	260

3.3.3 Introdução dos Ponteiros.

a) Ponteiro de VC-4

A carga útil dentro de um AU-4 tem comportamento dinâmico, isto é, não têm uma posição fixa. Ela pode começar em qualquer uma das 783 posições dentro de uma *Unidade Administrativa.* Através da interpretação do ponteiro retira-se o seu valor.

A introdução da variável p_{J1} *(valor do ponteiro)* na equação-3.4, possibilita que a carga útil assuma qualquer posição dentro de uma Unidade Administrativa -AU-4. A equação-3.4 para TU-12 passa para:

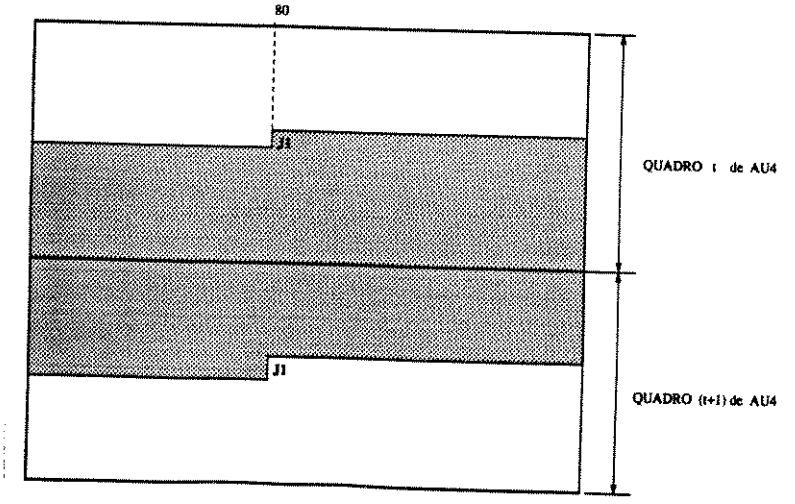
$$f_{TU-12} = p_{J1} mod(261) + 9 + k \tag{3.5}$$

$$\mathbf{P_{TU-12}(p_{J1}, k, l, m, c) = [f_{TU-12} + 3\times\ell + 21\times m + 63\times c] mod(261)} \tag{3.6}$$

Onde:
a.mod(y) = a-y.q onde
y.q $\leq a \leq$ *y.(q+1) com q=0,1,2,..,(inteiro).*

Na figura 3.8, tem-se um exemplo. Observe o posicionamento da carga útil VC-4 dentro do AU-4. Dado um valor para p_{J1}, o VC-4 será posicionado de acordo com este valor.(neste caso, p_{J1}=**80**). A figura abaixo mostra como se posiciona o VC-4, e observe que alguns bytes pertencem ao próximo AU-4.

Figura 3.8 : VC-4 alinhado no AU-4.



Para se introduzir a variável p_{J1} na equação (3.3), deve-se observar que o valor máximo é 782, devido a numeração de 3 em 3 do AU-4. Tem-se então:

$$\mathbf{p_{TU-12}(p_{J1}, l, m, c, r) = [p_{J1} + 3 + \ell + 7{\times}m + 21{\times}c + 87{\times}r]mod(783)} \quad (3.7)$$

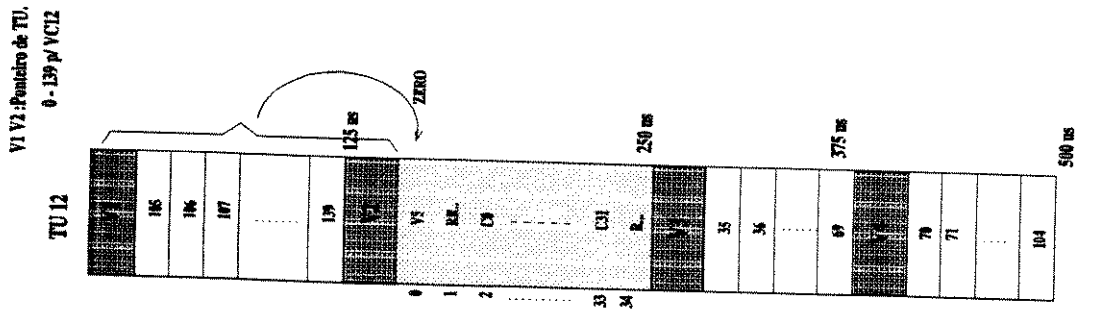
b) Ponteiro de VC-12 : O ponteiro de VC-12 pode assumir valores de **0,...,139**. São 140 valores para o ponteiro em um multiquadro, ou seja, em 500 μs. Em 125 μs, o ponteiro só poderá ter **35** valores: O objetivo aqui é determinar onde se encontra um determinado canal dentro de um quadro síncrono de transporte de informação, baseado no mapeamento síncrono mostrado na figura 3.6 e padronizado pelo ITU-TSS. Para simplificar o raciocínio, supoẽ-se inicialmente que *V1* e *V2* apontam para o valor *ZERO* dentro de um *TU-12*. A posição do **canal 0** está a **2** bytes do *V5*. A posição do **canal 1**, está a **3** bytes do *V5* e assim sucessivamente, ver figura 3.9. Então pode-se escrever para o canal n:

$$\mathbf{p_{VC-12}(p_{v5}, n) = [p_{v5}.mod(35) + 2 + n]mod(35);} \quad (3.8)$$

n: numero do canal - *(0,1,..,31)*;

Um exemplo de posicionamento do *VC-12* está mostrado na figura 3.9, com o valor nulo de ponteiro, posicionando a carga útil logo abaixo do byte *V2*.(TS0 foi utilizado como canal#0)

Figura 3.9 : VC-12 dentro de TU-12.



3.3.4 Formulação Geral:

Os conceitos vistos até aqui serão reunidos em duas equações básicas para localização de um canal dentro de um quadro síncrono.

- *a) Numeração de 1 em 1 no AU-4*

A *equação*-**3.6** pode localizar qualquer byte de um TU-12 dentro de um AU-4. O objetivo é o de localizar qualquer **canal** dentro de um AU-4. A *equação*-**3.8** possibilita localizar um canal dentro de um TU-12. Definido qual é o TU-12, localizam-se seus canais. As equações-3.6 e 3.8 serão reunidas em uma outra equação. A variável **c** na equação-3.6 é a que incrementa as posições de um TU-12, então, é no lugar desta variável que entra a equação-3.8.
A figura 3.5 mostra que as posições de V1,V2,V3 e V4 pertencem ao TU-12, mas não ao VC-12. Estabelecendo um relacionamento entre a posição p=p_{VC-12} e a coluna **c** tem-se na figura 3.10 para o TU-12.

r \ c	0	1	2	3
0	Vi	p=0	p=1	p=2
1	p=3	p=4	p=5	p=6
2	p=7	p=8	p=9	p=10
3	p=11	p=12	p=13	p=14
4	p=15	p=16	p=17	p=18
5	p=19	p=20	p=21	p=22
6	p=23	p=24	p=25	p=26
7	p=27	p=28	p=29	p=30
8	p=31	p=32	p=33	p=34

Figura 3.10 : Relação c x p e r x p no TU-12.

Ou seja, temos:

$$c = (p+1) mod 4 \tag{3.9}$$

$$r = \lfloor \frac{(p+1)}{4} \rfloor \tag{3.10}$$

onde: $\lfloor z \rfloor$significa Maior inteiro menor que z.

Então, substituindo a *equação*-**3.9** acima em (3.6) temos a posição do canal **n** (p_n) no AU-4.

$$p_n = [p_{J1} mod(261) + 9 + k + 3 \times \ell + 21 \times m + 63 \times (p+1) mod(4)] mod(261) \quad (3.11)$$

Uma outra forma de descrever a *equação*-**3.11** é substituir o termo que envolve mod(4). A figura 3.11 mostra a relação entre a variável y_1=x e y_2=xmod(4). Tem-se:

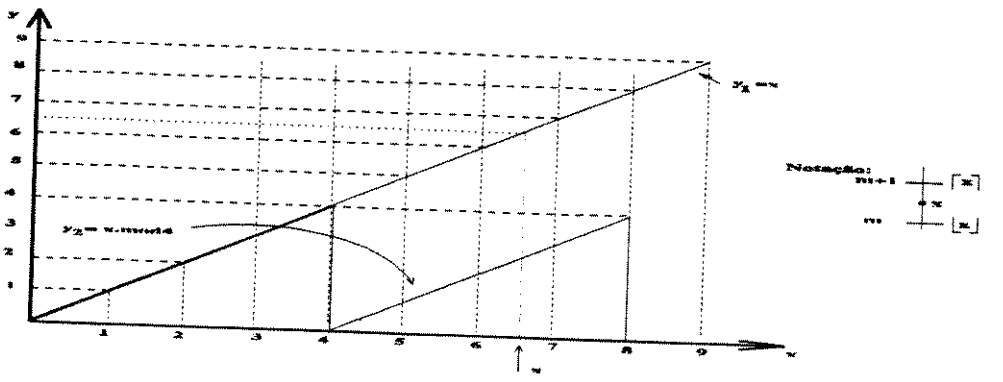


Figura 3.11 : Relação de X com X.mod(4).

$$y_1 - y_2 = 4.n \quad (3.12)$$

Onde:

$$\begin{aligned} n = 0 \quad & para \quad 0 \leq \quad x < 4 \\ n = 1 \quad & para \quad 4 \leq \quad x < 8 \\ n = 2 \quad & para \quad 8 \leq \quad x < 12 \\ etc \quad & \quad . \quad . \quad . \end{aligned}$$

fica:

$$n = \lfloor \frac{x}{4} \rfloor$$

Logo, (3.12) fica

$$x - xmod4 = 4 \lfloor \frac{x}{4} \rfloor$$

ou

$$xmod4 = x - 4 \lfloor \frac{x}{4} \rfloor. \quad (3.13)$$

Fazendo-se

$$x = (p+1)$$

na equação (3.11) e usando (3.13), obtemos:

$$f_n = p_{J1} mod(261) + 9 + k \tag{3.14}$$

$$\mathbf{p_n = [f_n + 3\times\ell + 21\times m + 63\times(p+1) - 252\times\lfloor\frac{(p+1)}{4}\rfloor]mod(261)}; \tag{3.15}$$

- *b) Numeração de 3 em 3 no AU-4.*

De modo semelhante ao caso a), reescrevendo a *equação*-3.7, tem-se:

$$p_n = [p_{J1} + 3 + \ell + 7\times m + 21\times c + 87\times r] mod(783)$$

O objetivo aqui também é o de introduzir p em (3.7). Os termos que variam as posições dos bytes de um mesmo TU-12 são **c** e **r**, cujos relacionamentos com p estão mostrados nas equações (3.9) e (3.10).

Tem-se então:

$$\mathbf{p_n = [p_{J1} + 3 + \ell + 7\times m + 21\times(p+1) + 3\times\lfloor\frac{(p+1)}{4}\rfloor]mod(783)} \tag{3.16}$$

3.3.5 Formulação para Localização de *V1 e V2* - TU-12

Introdução:

A obtenção de uma formulação que indique as posições ocupadas pelos bytes V1 e V2 é necessária para a retirada destes bytes e, através da sua interpretação, determinar onde está o byte V5. A interpretação do byte **H4** mostrado na figura 3.2 possibilita saber qual o byte que está sendo retirado: V1,V2,V3 ou V4 do VC-4.

Localização dos ponteiros de TU-12.

- Estas equações são uma simplificação das equações anteriores, onde, ao aplicá-las, localiza-se sempre o 1^o byte de um TU12 qualquer, mostrando a posição do seu POH.

$$\mathbf{p_{v1v2} = (p_{J1} + 3 + \ell + 7 \times m) mod (783)} \tag{3.17}$$

$$\mathbf{p_{v1v2} = [p_{J1} mod (261) + 9 + k + 3 \times \ell + 21 \times m] mod (261)} \tag{3.18}$$

3.3.6 Exemplos e Interpretações

Vamos utilizar a *equação*-**3.16** em alguns exemplos.

- 1^o *CASO* : $p_{J1} = 0$, $p_{v5} = 0$, $\ell = 0$, m $= 0$;

Neste primeiro exemplo, tem-se as diversas posições ocupadas pelos **32** canais dentro de um AU-4. Observando a figura 3.12, a primeira posição a ser ocupada é a 66 e nas posições anteriores deste *TU-12*, tem-se: *3 = V2, 24 = V5, 45 = R...* Assim são **32** posições ocupadas entre 66,..,741. Até agora tem-se 35 posições ocupadas por este TU-12; mas como um TU-12 possui 36 bytes, então, na posição **762** temos mais 1 byte de enchimento. Observe que na posição 3 tem-se V2 porque o ponteito p_{v5} aponta para a posição *zero* e anterior a esta está V2 (Observar a figura 3.5). Observe também que a diferença entre as posições 66 e 90 é de 24 bytes e entre 90 e 111 é de 21 bytes. Os três bytes a mais advém da passagem de uma linha para outra onde, onde no início de cada linha tem-se bytes ocupados por *POH* e *stuff*, que são descontados.

- 2^o *CASO* : $p_{J1} = 0$, $p_{v5} = 100$, $\ell = 0$, m $= 0$;

Neste caso, a primeira posição a ser ocupada é a **720**, como mostra a figura 3.12. Observe que as posições ocupadas dentro de um AU-4 não se alteram, pois o valor do ponteiro p_{J1} continua sendo zero. O que muda são as posições ocupadas pelos canais dentro da estrutura. O canal zero, por exemplo, está agora na posição 720. Deslocado 21 posições atrás, mais precisamente na posição **699**, tem-se R... e V5 está na posição **675**. Observe todo o deslocamento que ocorreu nas posições dos canais. Como $p_{v5} = 100$ a posição **3** está agora com **V4**. Observe a *figura 3.5.*

- 3^o *CASO* : $p_{J1} = 0$, $p_{v5} = 139$, $\ell = 0$, m $= 0$;

As conclusões aqui são identicas as do 2^o *CASO*. Observe com relação ao 1^o CASO o deslocamento que ocorreu na ocupação das posições. Antes o canal zero estava na posição **66**; agora está na posição **45**; isto é perfeitamente correto já que o ponteiro de p_{v5} está apontando para uma posição anterior dentro de um TU-12. Assim, todos os canais foram deslocados de uma posição para trás.

- 4^o *CASO* : $p_{J1} = 1$, $p_{v5} = 0$, $\ell = 0$, m $= 0$;

Observe que agora ocorre uma mudança nos valores das posições que este TU-12 ocupa no AU-4. Isto deveria de fato acontecer já que agora o valor do ponteiro p_{J1} foi alterado. Com este novo valor de ponteiro, as posições foram deslocadas para frente de **1**. Compare novamente com o 1^o CASO.

- 5^o *CASO* : $p_{J1} = 1$, $p_{v5} = 100$, $\ell = 0$, m $= 0$;

Comparando com o 2^o CASO, observe os deslocamentos ocorridos nas posições ocupadas por um TU-12: Os endereços foram deslocados de uma posição pra frente devido ao valor do p_{J1}. Comparando com o caso anterior, verificam-se as novas posições ocupadas pelos canais dentro de uma mesma numeração de TU-12. Na posição **4** tem-se V4 e V5 está agora a duas posições atrás do canal zero, ou seja, na posição **676**.

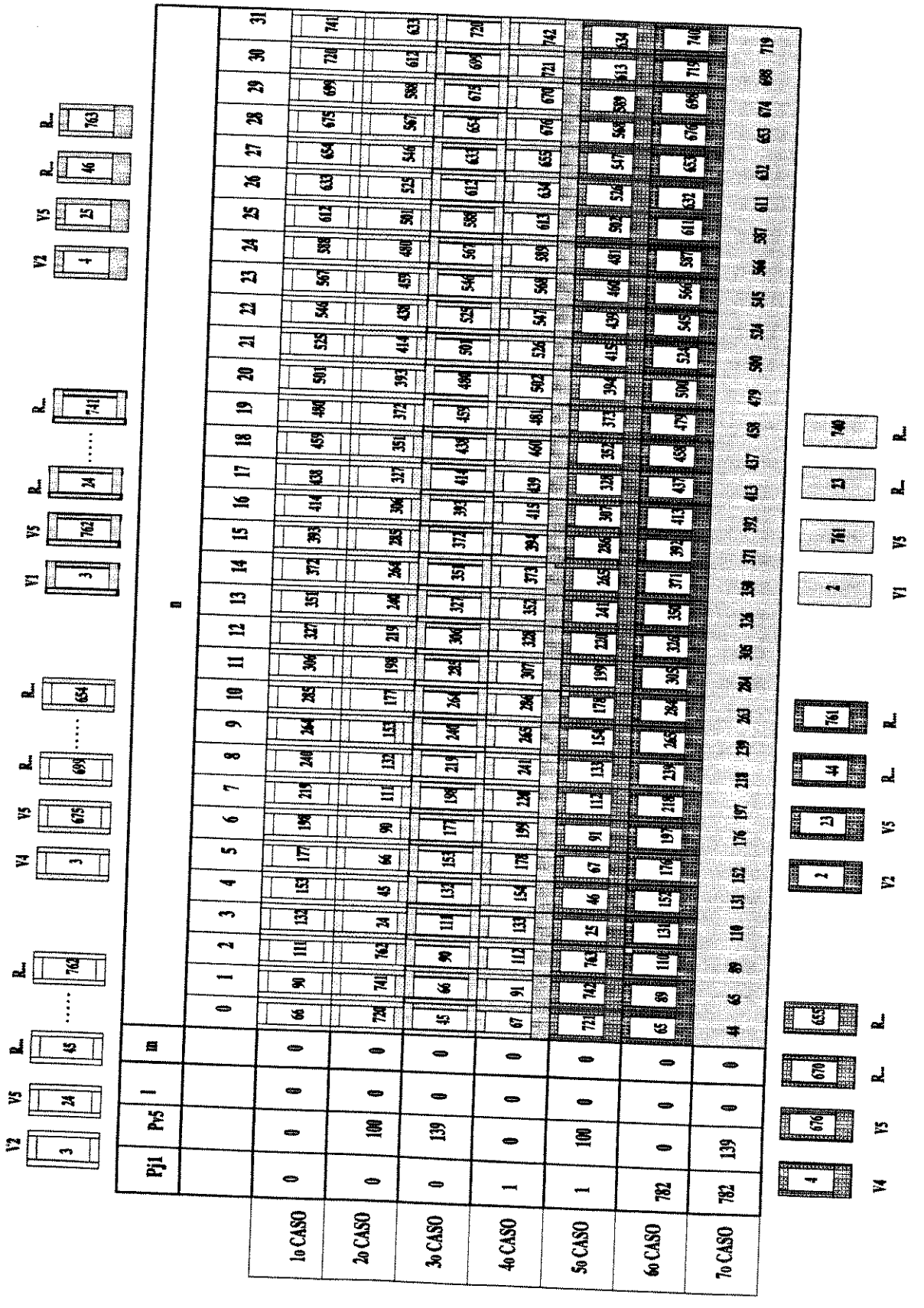
- 6^o *CASO* : $p_{J1} = 782$, $p_{v5} = 0$, $\ell = 0$, m $= 0$;

Neste caso, o valor do p_{J1} posiciona a carga útil em 782, ou seja, a carga útil está no último byte do AU-4, deslocando um byte para trás, no próximo quadro, os endereços. Comparando o caso 6 com o caso **1** nota-se este deslocamento nos endereços.

- 7^o *CASO* : $p_{J1} = 782$, $p_{v5} = 139$, $\ell = 0$, m $= 0$;

Este exemplo usa os valores extremos nos ponteiros. Como o valor do ponteiro p_{J1} é o mesmo do exemplo anterior os valores dos endereços continuam os mesmos, o que muda são as posições assumidas pelos canais dentro deste endereços, pois alterou-se o valor do ponteiro p_{v5}. Este caso pode também ser comparado com o caso **3**.

Figura 3.12 : Numeração de um VC-12.



3.4 Formulação para Tributários de 1.5 Mb/s

As equações aqui desenvolvidas seguem o mesmo raciocínio utilizado anteriormente, com algumas diferenças devido as suas estruturas, ou seja, um *VC-12* é diferente de um *VC-11*. No restante as estruturas se equivalem, o que implica em equações semelhantes àquelas já desenvolvidas.

Como foi visto anteriormente, 4 TU-11 são multiplexados dentro de 1 TUG-2 e 7 TUG-2 dentro de 1 TUG-3; então, dentro de cada TUG-3 tem-se 28 TU-11 multiplexados. Como dentro de 1 VC-4 são multiplexados 3 TUG-3, a capacidade de um VC-4 é de **84** TU-11. Dentro de um TU-11 tem-se um VC-11 e, neste estão colocados vários canais.

- *Escolhidos* TUG-3# k *e* TUG-2#ℓ, *o canal pode estar dentro de um dos* **4** *TU-11. A variável* **m** *será usada para o TU-11.*

  m pode assumir valores de *0,1,2,3*$\Leftrightarrow$ TU-11#0,TU-11#1,TU-11#2,TU-11#3.

- *Valor do ponteiro de TU-11. Variável* $\mathbf{p_{v5}}$.

  $\mathbf{p_{v5}}$ pode assumir valores de *0,..,103.*

3.4.1 Numeração de Coluna para TU-11

Na figura 3.13 a distância entre o 1^o byte de um TU-11 e o 2^o byte do mesmo TU-11 é de 84 para uma numeração contínua e de 28 para uma numeração de 3 em 3. Anteriormente viu-se que um TUG-2 comporta 4 TU-11; como existem 12 bytes ocupados por um TUG-2 em uma linha, então tem-se 3 bytes ocupados em uma linha por um mesmo TU-11.

3.4.2 Formulação Geral.

Indo diretamente as formulações, já que o raciocínio é identico ao feito para as equações anteriores, tem-se:

$$\mathbf{p_{VC-11} = [p_{v5} mod(26) + 2 + n] mod(26)} \tag{3.19}$$

$$p = p_{VC-11};$$

A posição de um canal n (n pode ir de 0 a 24) transportado em um TU11(ℓ,m) é dada por:

$$\mathbf{p_n = [p_{J1} + 3 + \ell + 7 \times m + 28 \times (p + 1) + 3 \times \lfloor \frac{(p + 1)}{3} \rfloor] mod(783);} \tag{3.20}$$

Em (3.20) não há distinção de TUG3. A posição de um canal n transportado em TU11(k,ℓ,m) é dada por:

$$f_n = p_{J1} mod(261) + 9 + k \tag{3.21}$$

$$\mathbf{p_n = f_n + 3 \times \ell + 21 \times m + 84 \times (p + 1) + 9 \times \lfloor \frac{(p + 1)}{3} \rfloor] mod(261);} \tag{3.22}$$

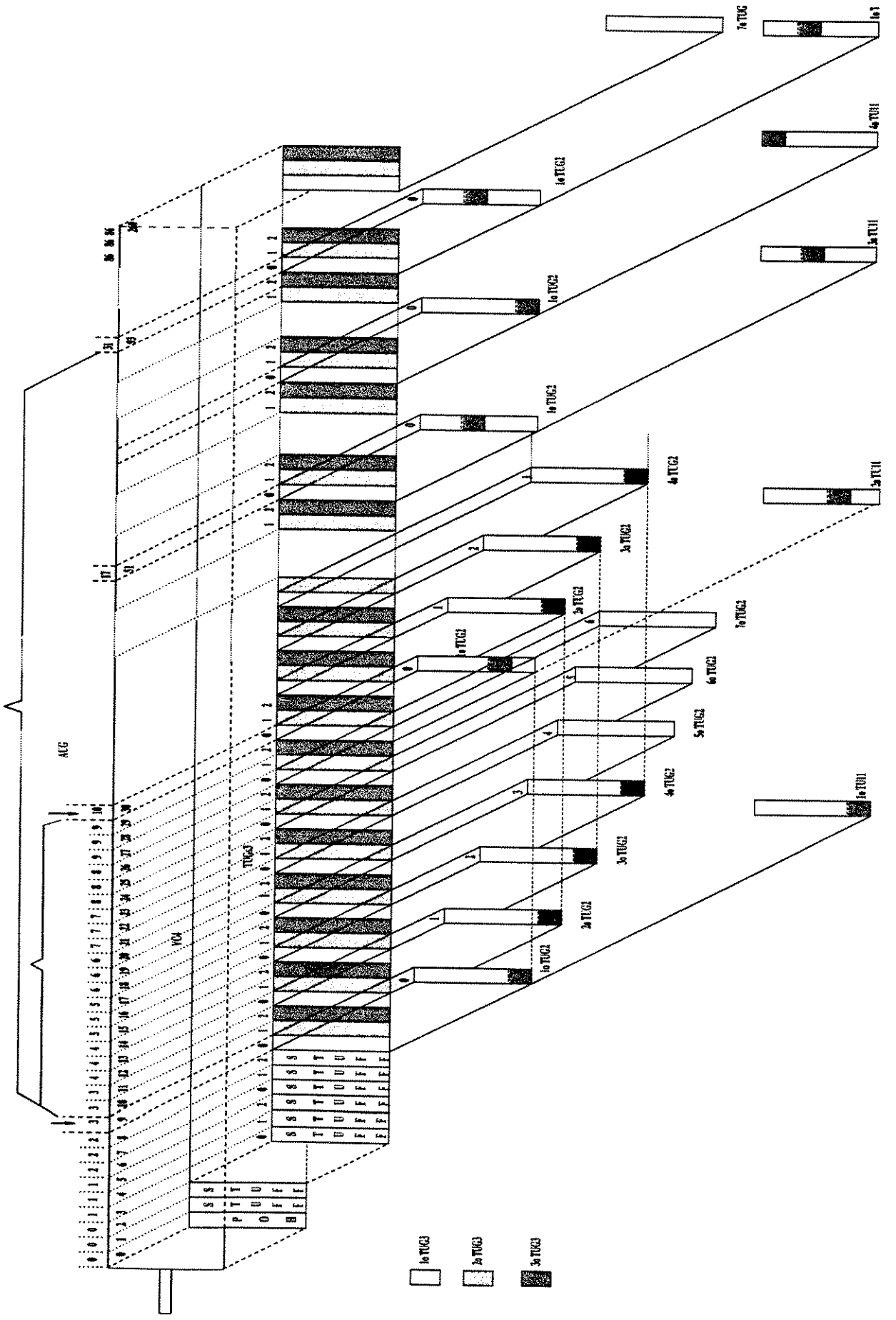
3.4.3 Formulação para Localização de *V1 e V2* - TU-11

- Estas equações são uma simplificação das equações anteriores.

$$\mathbf{p_{v1v2} = [p_{J1} + 3 + \ell + 7 \times m] mod(783)} \tag{3.23}$$

$$\mathbf{p_{v1v2} = [p_{J1} mod(261) + 9 + k + 3 \times \ell + 28 \times m] mod(261)} \tag{3.24}$$

Figura 3.13 : Diagrama Geral de Formulação para TU-11.



3.4.4 Exemplos e Interpretações.

Vamos utilizar a *equação*(**3.20**) nos exemplos.

- 1^o *CASO* : $p_{J1} = 0$, $p_{v5} = 0$, $\ell = 0$, m $= 0$;

Observando a figura 3.14 a primeira posição a ser ocupada é a 90 e nas posições anteriores deste *TU-12*, tem-se: *3 = V2, 31 = V5, 59 = P1P0...* assim são **27** posições ocupadas entre 3,..,755, com os 24 canais alocados de 90,..755.

- 2^o *CASO* : $p_{J1} = 0$, $p_{v5} = 100$, $\ell = 0$, m $= 0$;

Neste caso, a primeira posição a ser ocupada é a **727**, como mostra a figura 3.14. Observe que as posições ocupadas dentro de um AU-4 não se alteram, pois o valor do poteiro p_{J1} continua sendo zero. O que muda são as posições ocupadas pelos canais dentro da estrutura. O canal zero, por exemplo, está agora na posição 727. Deslocado **28** posições atrás, mais precisamente na posição **699**, tem-se P1P0... e V5 está na posição **668** com uma distância de **31** bytes isto porque descontou-se POH e Enchimento. Observe todo o deslocamento que ocorreu nas posições dos canais. Como $p_{v5} = 100$ a posição **3** está agora com **V1**. Observar a figura 3.5.

- 3^o *CASO* : $p_{J1} = 0$, $p_{v5} = 103$, $\ell = 0$, m $= 0$;

Observe com relação ao 1^o CASO o deslocamento que ocorreu na ocupação das posições. Antes o canal zero estava na posição **90**; agora está na posição **59**; isto é perfeitamente correto já que o ponteiro de p_{v5} está apontando para uma posição anterior dentro de um TU-11. Assim, todos os canais foram deslocados de uma posição para trás.

- 4^o *CASO* : $p_{J1} = 1$, $p_{v5} = 0$, $\ell = 0$, m $= 0$;

Observe que agora ocorre uma mudança nos valores das posições que este TU-11 ocupa no AU-4. Isto deveria de fato acontecer já que agora o valor do ponteiro p_{J1} foi alterado. Com este novo valor de ponteiro, as posições foram deslocadas para frente de **1**. Compare novamente com o 1^o CASO.

- 5^o *CASO* : $p_{J1} = 1$, $p_{v5} = 100$, $\ell = 0$, m = 0;

Comparando com o 2^o CASO, observe os deslocamentos ocorridos nas posições ocupadas por um TU-11; os endereços foram deslocados de uma posição pra frente devido ao valor do p_{J1}. Comparando com o caso anterior, verificam-se as novas posições ocupadas pelos canais dentro de uma mesma numeração de TU-11. Na posição **4** tem-se V1 e V5 está agora a duas posições atrás do canal zero, ou seja, na posição **669**.

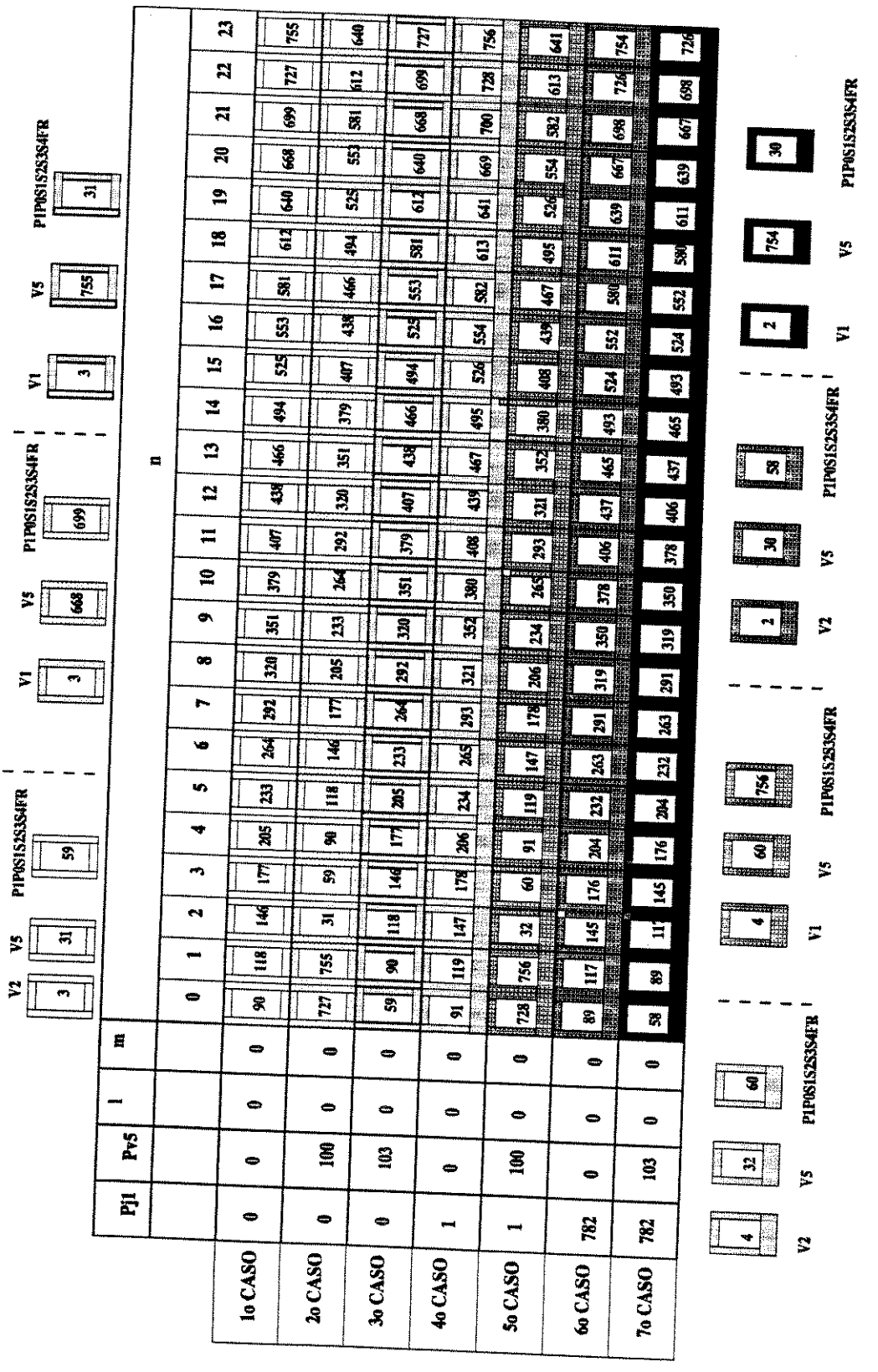
- 6^o *CASO* : $p_{J1} = 782$, $p_{v5} = 0$, $\ell = 0$, m = 0;

Neste caso, o valor do p_{J1} posiciona a carga útil em 782, ou seja, a carga útil está no último byte do AU-4, deslocando um byte para trás, no próximo quadro, os endereços. Comparando este caso com o caso **1** notam-se estes deslocamento nos endereços.

- 7^o *CASO* : $p_{J1} = 782$, $p_{v5} = 103$, $\ell = 0$, m = 0;

Este exemplo usa os valores extremos nos ponteiros. Como o valor do ponteiro p_{J1} é o mesmo do exemplo anterior os valores dos endereços continuam os mesmos, o que muda são as posições assumidas pelos canais dentro deste endereços, pois alterou-se o valor do ponteiro p_{v5}. Este caso pode também ser comparado com o caso **3**.

Figura 3.14 : Numeração de um VC-11.



3.4.5 Comentários Finais:

As formulações desenvolvidas em 3.14 (2 Mb/s) e 3.19(1.5 Mb/s) possibilitam a localização de qualquer canal **n** dentro de um quadro síncrono, utilizando as equações 3.15 e 3.16 para localizar os bytes V1 e V2, que são fixos em relação ao VC4,para determinar a posição de V5. Com isto pode-se usar um mux/derivador inteligente para a retirada de canais de 64 kb/s. Estas formulações podem ser usadas para o desenvolvimento de ckts paralelos para a retirada e análise de bytes específicos de gerenciamento. Com isto pode-se, por exemplo, determinar que tipo de mapemento de carga útil(ATM,MAN(DQDB),FDDI,etc) trás um determinado VC-4, através da interpretação do byte C2. Uma outra aplicação poderia ser a retirada direta de canais de voz específicos que transportam determinada informação pela rede. Com isso tem-se a possibilidade de rastrear um determinado agregado com maior velocidade de processamento, já que alguns circuitos intermediários podem ser anulados. Em aplicações onde se tem uma rede em anel que trafegem sinais de 2.5 Gb/s pode ser interesante a retirada de alguns bytes específicos, para análise, sem se fazer uma demultiplexação total do sinal agregado.

Capítulo 4

Adaptação das Rotas de Baixa Ordem.

4.1 Introdução

Neste capítulo apresenta-se o projeto dos blocos funcionais de Adaptação das Rotas de Baixa Ordem na direção de recepção. As operações de Alinhamento e Multiplexação já foram analisadas anteriormente. Agora aborda-se a operação Desmapeamento em direção ao terminal de afluentes, partindo-se dos desenvolvimentos feitos para os demais blocos funcionais de uma estrutura SDH. Estuda-se aqui três tipos de mapeamento de baixa ordem: Mapeamento síncrono de byte, mapeamento síncrono de bit e mapeamento assíncrono, todos a uma taxa de 2048 kb/s. Utiliza-se também a linguagem VHDL[3] para desenvolver estes blocos, já que esta linguagem permite uma análise dos sinais que trafegam pelos equipamentos SDH. Com este trabalho temos mais uma parte dos blocos funcionais dos equipamentos SDH.

4.2 Recepção Síncrona

4.2.1 Introdução

A recepção de sinais síncronos consiste num desmapeamento[11], ou seja, retirada do POH, dos bytes de enchimento de um VC-12 e finalmente da carga útil. Para se fazer esta retirada, tem-se que saber onde inicia o VC-12, o relógio do VC-12 e o próprio sinal VC-12 (estes sinais são fornecidos pelos blocos funcionais anteriormente desenvolvidos). Deve-se obter na saida 63 sinais de 2048 kb/s, cada um com 32 canais de 64 kb/s e um clock de 2048 kHz. Observa-se que nestes sinais de saída não poderão ocorrer "buracos", seja no próprio sinal de informação, seja no seu clock correspondente.

4.2.2 Diagrama Temporal dos Sinais de Entrada

- O clk_vc12 é um relógio com buracos, pois o mesmo é derivado do relógio do TU-12. O TU-12 possui 36 bytes em 125 μs e o VC-12 tem 35 bytes em 125 μs. Deve-se destacar que este clock pode vir alterado devido a possíveis justificações no TU12.
- O sinal inic_vc12 é necessário para se saber a posição do V5, e a partir daí, retirar o seu POH e bytes R de enchimento.
- O sinal VC-12 é o sinal de onde serão retirados os 32 canais.
- A figura 4.1 mostra os sinais de entrada.
- A figura 4.2 mostra o diagrama básico do receptor.

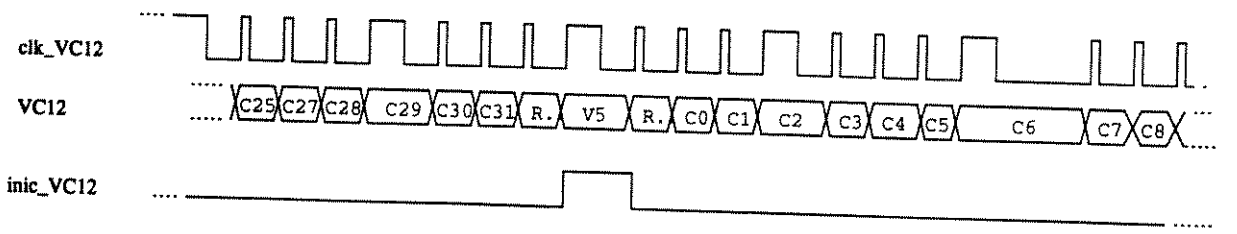


Figura 4.1 : Diagrama temporal dos sinais de *Entrada do Receptor*

4.2.3 Gerador dos endereços de VC-12

- Este bloco é modelado pelo processo abaixo.

```
ender_vc12:PROCESS(inic_vc12,clk_vc12)
BEGIN
    IF(inic_vc12='1') AND (inic_vc12'EVENT) THEN
       ender_vc12 <= 0;
   END IF;
    IF(clk_vc12='1') AND (clk_vc12'EVENT) THEN
      IF(ender_vc12=34) THEN
         ender_vc12 <= 0;
      ELSE
         ender_vc12 <= ender_vc12 + 1;
      END IF;
   END IF;
         clk_saida_vc12 <= NOT clk_vc12;
END PROCESS;
```

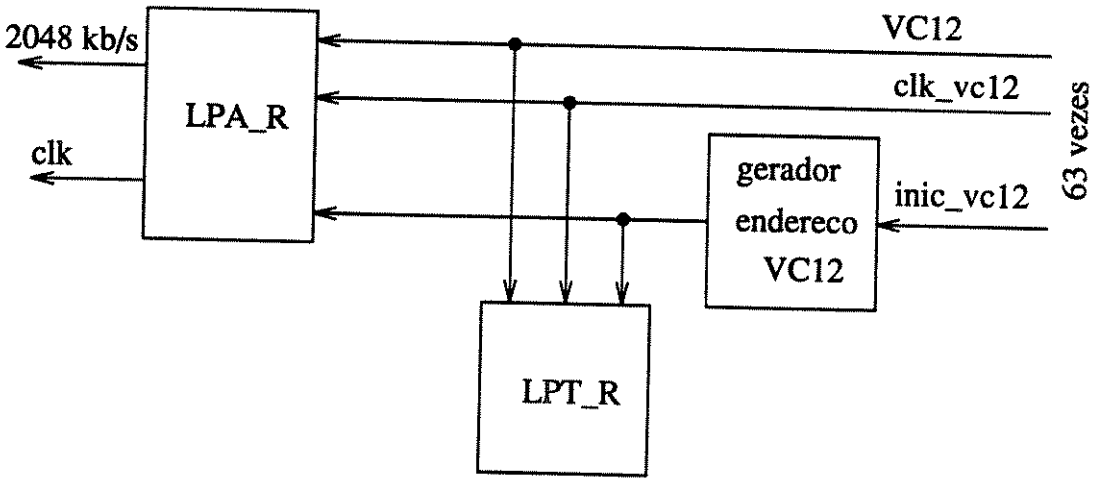


Figura 4.2 : Diagrama de Blocos do Receptor

Este bloco funcional gera 35 sinais para endereço de VC-12 a partir do início de VC-12, onde está a posição V5. Verifica-se na figura 4.3 que na subida do sínal *inic_vc12* o *ender_vc12* é zerado, independentemente do *clk_vc12*. Apartir deste instante, a cada subida do clk_vc12 o endereço é incrementado, até que chegue a 34, quando então é zerado. Pode-se, com isto, identificar cada elemento dentro de um quadro VC-12. Observe na figura 4.3 que tem-se três períodos distintos, ou seja, t1, t2, t3. Um TU-12 possui 36 bytes dentro de um AU-4, onde em cada linha estão colocados 4 bytes de um mesmo quadro; estes bytes possuem uma certa ordem de colocação e uma certa distância entre eles. Em t2 estão suprimidos os bytes de **SOH**(*ponteiro de AU)* e bytes de enchimento; os outros 3 bytes estão a uma mesma distâcia fixa t1. O período t3 é exatamente onde se retira um byte do TU-12 que pode corresponder a V1,V2,V3 ou V4 mais **SOH**. O clk_vc12 é derivado do clk_tu12 possuindo então estes "buracos". Lembramos que o clk_vc12 contém pulsos a mais ou a menos, devido a possíveis justificações de ponteiro de TU, em torno do byte V3. O clk_vc12 é então invertido e colocado para fora deste bloco em direção aos demais blocos. Será visto, mais à frente, a necessidade desta inversão e deste sinal.

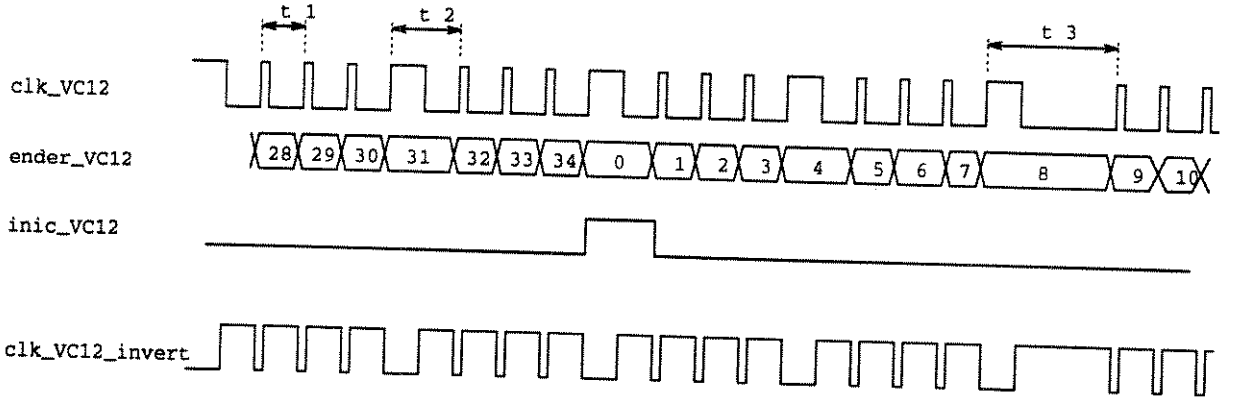


Figura 4.3 : Diagrama temporal do sinal *ender_vc12*

4.2.4 LPA_R

Utilizando os sinais clk_vc12 e inic_vc12 este bloco irá, através de um relógio de escrita com buracos, escrever dentro de uma memória somente 32 bytes de um quadro VC12, sendo que 3 bytes serão suprimidos, pois os mesmos não são dados de um Container. Os dados serão lidos do outro lado da memória através de um relógio de leitura sem buracos. O tamanho escolhido para a Memória Elástica é de 64 bytes, espaço suficiente para armazenar até 64 amostras de VC12. Este tamanho é tomado levando-se em conta um atraso máximo não muito grande (8ms) e espaço suficiente para amortecer as variações de taxa devidas aos buracos. No item 4.2.9 tem-se mais comentários.

- O diagrama de blocos do LPA_R está mostrado na figura 4.6. O seu funcionamento é descrito através dos blocos a seguir.

Geração de bur_1

- Este bloco é modelado pelo processo abaixo:

```
bur_1:
    bur_1 <='1' WHEN (ender_vc12=34) OR (ender_vc12 <= 1) ELSE
                '0';
```

Conforme observado na figura 4.4 o sinal bur_1 é gerado a partir dos endereços de VC-12. O sinal é zero em todos os instantes, exceto onde temos as posições dadas por R,V5,R. Isto possibilitará a retirada dos bytes do cabeçalho de um VC-12. Nos próximos quadros todos os bytes de cabeçalho serão retirados já que temos distâncias iguais entre os mesmos (J2,Z6,Z7).

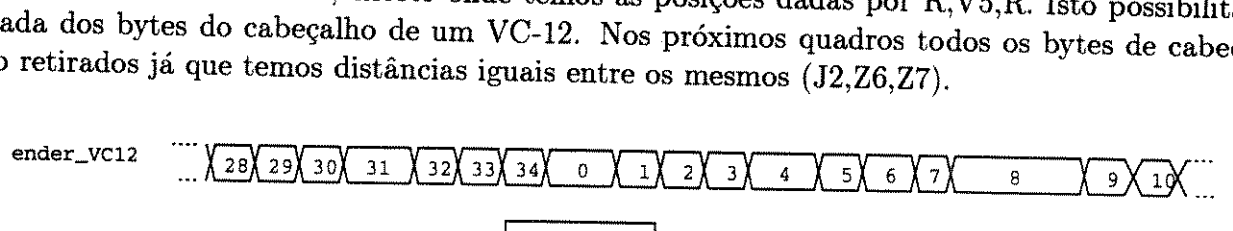


Figura 4.4 : Diagrama temporal do sinal *bur_1*

4.2.5 Geração do Relógio de Escrita

- Este bloco é modelado pelo processo abaixo:

```
rel_esc:
   rel_esc <= clk_vc12 AND NOT bur_1;
```

Este bloco fornece o clock para a geração de endereços de escrita na memória elástica e escreve dentro desta os 32 bytes válidos. A figura 4.5 mostra este sinal. Observe uma diferença entre o final do sinal bur_1 e o rel_esc, obtido através do sinal clk_vc12_invert, para obter o sinal rel_esc no instante exato em que tem-se dados válidos estáveis na entrada da memória elástica.

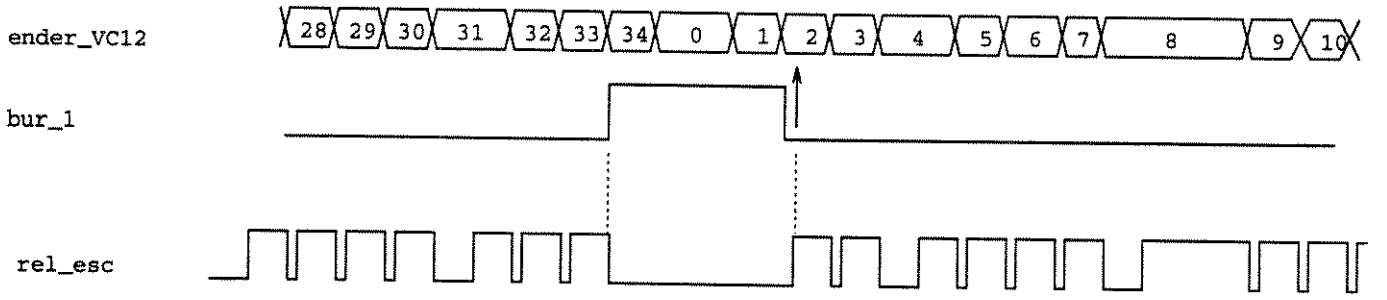
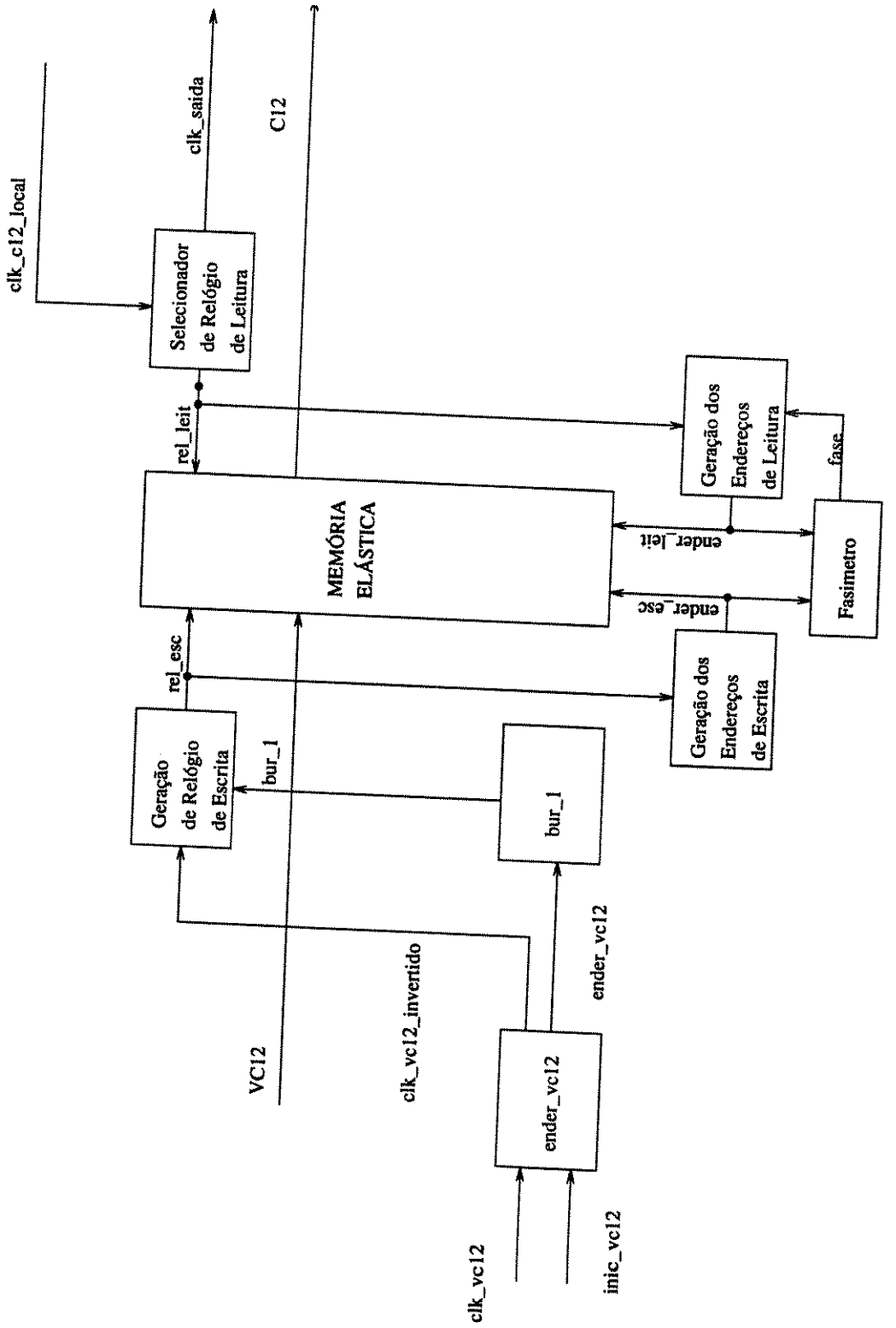


Figura 4.5: Diagrama temporal do *rel_esc*

Figura 4.6 : Detalhamento do Bloco de Desmapeamento de Sinais Síncronos.



4.2.6 Geração de Endereço de Escrita

- Este bloco é modelado pelo processo abaixo:

```
Cont_end_escrita:PROCESS(rel_esc)
BEGIN
   IF (rel_esc='0') AND (rel_esc'EVENT)THEN
      IF (ender_esc = 63) THEN
         ender_esc <= 0;
      ELSE
         ender_esc <= ender_esc + 1;
      END IF;
   END IF;
END PROCESS;
```

Este bloco funcional gera 64 endereços de escrita na memória elástica, onde os dados são colocados um a um. A cada borda de descida do **rel_esc** é incrementado o **ender_esc**. A figura 4.7 mostra a geração dos endereços de escrita.

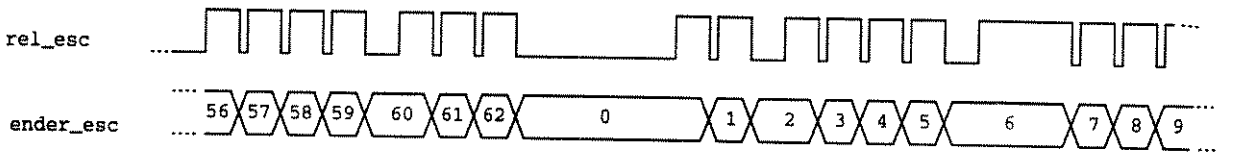


Figura 4.7 : Diagrama temporal do sinal *ender_esc*

4.2.7 Geração de Endereço de Leitura

- Este bloco é modelado pelo processo abaixo:

```
cont_ender_ender_leitura:PROCESS(rel_leit)
VARIABLE ender:INTEGER:='0'; BEGIN
    IF (rel_leit='0') AND (rel_leit'EVENT)THEN
      IF (ender_leit = 63) THEN
         ender := 0;
      ELSIF (fase <= 5)THEN
         ender := ABS(ender_leit - 32);
      ELSIF (fase >= 59)THEN
         ender := ender_leit +32);
      ELSE
         ender := ender_leit +1;
      END IF;
         ender_leit <= ender;
   END IF;
END PROCESS;
```

Este bloco gera os endereços de leitura.Foi modelado em função de se tentar observar SLIPS numa rede síncrona. O endereço de leitura é controlado em função da fase da memória elástica,

sendo encrementado ou decrementado de 32 bytes dependendo do limiar que foi atingido. Em 4.2.9 a explicação está mais detalhada. A figura 4.8 mostra a geração dos endereços de leitura.

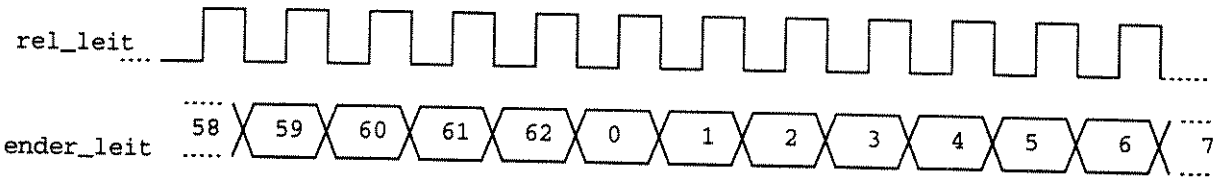


Figura 4.8 : Diagrama temporal do *ender_leit*

4.2.8 Geração da Fase

- Este bloco é modelado pelo processo abaixo:

```
fasímetro:PROCESS(ender_esc,ender_leit)
VARIABLE v_fase:INTEGER:=0;
BEGIN
   v_fase:=ender_esc-ender_leit;
      IF(v_fase < 0)THEN
         fase <= 64 + v_fase;
      ELSE
         fase <= v_fase;
      END IF;
END PROCESS;
```

Este bloco mede a distância entre o endereço de escrita e o endereço de leitura reportando este valor para a geração do endereço de leitura. A figura 4.9 mostra o sinal gerado por este bloco.

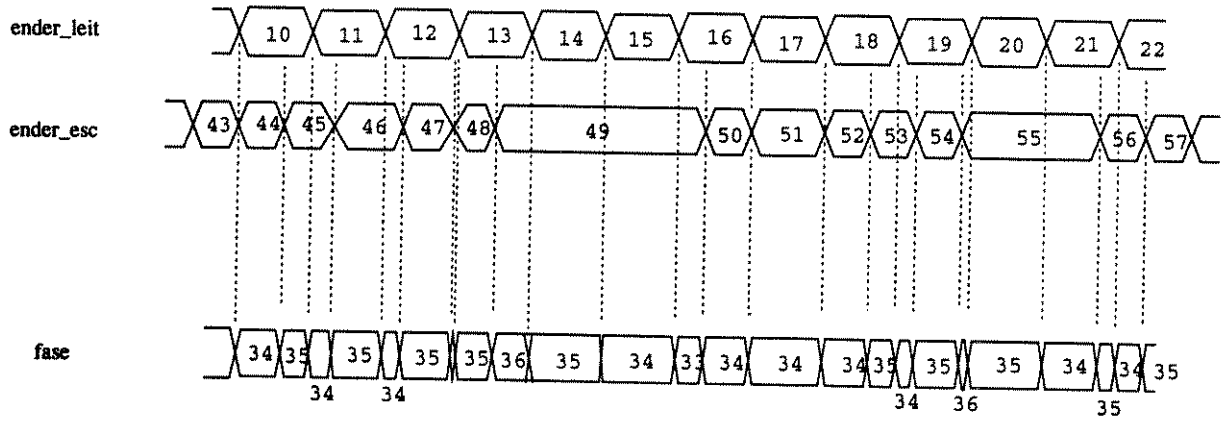


Figura 4.9 : Diagrama temporal da *Fase*

4.2.9 Memória Elástica

- Este bloco é modelado pelo processo abaixo:

```
escrita_memória_elástica:PROCESS(rel_esc)
BEGIN
   IF (rel_esc = '1') THEN
      mem_elast(ender_esc) <= VC-12;
   END IF;
END PROCESS;
leitura_memória_elástica:PROCESS(rel_leit)
BEGIN
   IF (clk_c12_local = '1')THEN
      C-12 <=mem_elast(ender_leit);
   END IF;
END PROCESS;
```

Este bloco rescebe somente os bytes válidos de VC-12, ou seja, os 32 canais síncronos. Através do rel_esc é feito o endereçamento de escrita e é neste endereço que o dado de VC-12 será escrito. Observe que se o relógio de escrita **(rel_esc)** estiver sincronizado com o relógio de leitura **(rel_leit)** não ocorrerá o problema de encher ou esvaziar a memória elástica.

- O clk de VC-12 derivado do TU-12 é mais rápido que o de C-12,ou seja, 36 bytes x 64 kb/s = 2304 kb/s;
- O clk de C-12 é mais lento com 32 bytes x 64 kb/s = 2048 kb/s;
- Como o clk de VC-12 é um clk com buracos,temos:32/36 x 2304 kb/s = 2048 kb/s; deste modo a taxa média de escrita e leitura é idêntica.

O relógio de leitura faz a retirada dos dados da memória elástica, sendo os dados lidos em uma frequência dada por um relógio local. A figura 4.10 mostra o sinal de saída da memória elástica. Observe que este sinal, bem como o seu clock, não possue buracos. Tem-se então na saída um sinal síncrono de **2048** kb/s.

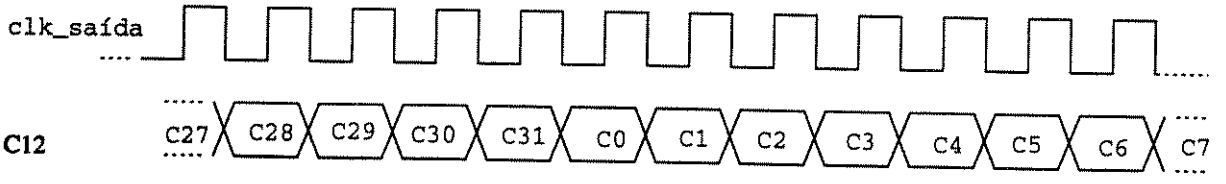


Figura 4.10 : Diagrama temporal dos sinais de saída do lpa_r

É bom lembrar que em uma rede síncrona não há ocorrência de SLIPS [4]. Porém, o bloco de geração de endereços de leitura foi modelado a fim de "forçar" este SLIP (*uso de clock síncrono local independente da fonte do sinal).* Quando o clk na origem estiver na condição de *free-running* observaremos escorregamentos com o LPA_R proposto. Fez-se isto para introduzir o conceito de um receptor universal, ou seja, que opere com sinais síncronos e assíncronos. O objetivo aqui é mostrar que o relógio de saída deve ser obtido do sinal que chega, uma condição que leva naturalmente a se implementar só um tipo de receptor, capaz de receber sinais síncronos ou assíncronos.

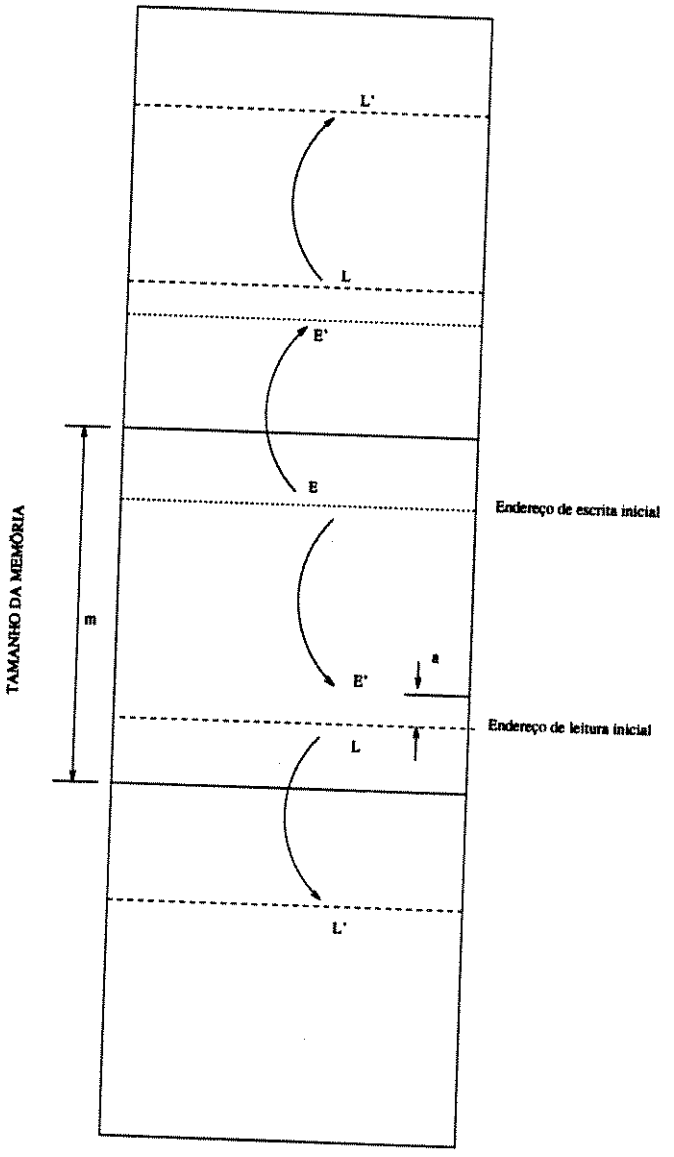
4.2.10 SLIP

Nesta interface **(memória elástica)** a passagem de informação de um lado para outro se processa através do uso de uma memória elástica(BUFFER) de tamanho determinado, de tal modo que se usa o rel_esc para escrever, com um relógio que vem de dentro do equipamento e se usa um relógio local para ler, ou seja, retirar os dados. Quando existem diferenças entre os relógios é possivel haver "OVERFLOW" ou "UNDERFLOW" na memória elástica. Quando o relógio de escrita é mais rápido que o de leitura, a ocupação da memória com novos bits aumenta com o tempo e, eventualmente quando o relógio de escrita passa à frente do relógio de leitura, ocorre um "OVERFLOW" e N bits são suprimidos, onde N é o numero de bits da memória. Ao contrário quando o relógio de leitura é mais rápido que o de escrita ocorre um "UNDERFLOW" e N bits são lidos na memória duas vezes, resultando uma repetição de N bits nos dados de saída. A isto dá-se o nome de SLIP.

4.2.11 SLIP CONTROLADO

É conveniente, para fins de transmissão, que quando for ocorrer um SLIP este seja de um numero inteiro de quadros, isto é importante principalmente na trasmissão de voz, pois exatamente um quadro depois teremos o sinal amostrado com uma fase bem próxima da amostra anterior fazendo com que o "estalo" seja imperceptível.É usual então suprimir ou repetir exatamente um quadro na ocorrência de um slip. Este evento é denominado SLIP CONTROLADO. Por outro lado, mesmo quando os relógios de escrita e de leitura tem a mesma frequência média, slips podem ocorrer se os relógios estiverem alinhados e ouver flutuação de fase entre eles. Neste caso, ultrapassagens mútuas entre os relógios provocam múltiplos slips. Por esta razão, é comum forçar uma fase inicial entre os dois relógios de N/2 bits. Observe no bloco de geração de endereços de leitura que o ender_esc=0, e que o ender_leit=32 isto força uma fase inicial de 32 bytes entre os enderços de escrita e de leitura. No caso de diferenças entre o relógio de escrita e de leitura deve-se também adotar um limiar de slip antes que os endereços venham a se alinhar. Para isto o tamanho da memória elástica deve ser de dois quadros. Observe que a memória é de 64 bytes. Com relação a flutuação de fase , o ITU-TSS estabelece que a flutuação na entrada da central de comutação pode ter uma amplitude máxima de 18 μs. Deste modo, a fim de evitar mútiplos slips, não se deve permitir que os relógios de escrita e de leitura estejam com alinhamento menor que 18 μs(limiar de slip). Quando os relógios da interface estão perfeitamente sincronizados, não há ocorrência de slips e utiliza-se pequenas memórias de transferência somente para permitir uma certa margem para pequenas variações de fase causadas por flutuações aleatórias de fase (jitter) e flutuações lentas(wander)no atraso de fase. No nosso caso odotamos que slips podem ocorrer e nas simulações mostraremos este fenomeno. A figura 4.11 mostra o mecanismo de SLIP CONTROLADO quando o rel_esc está bem mais rápido que o rel_leit. Pode-se observar que quando atingido o limiar de slip de 18 μs ou 5 bytes o endereço de leitura é deslocado de 32 bytes. Observando-se a figura 4.11 verifica-se que quando o endereço de escrita inicial **E** se desloca para **E'** o endereço de leitura **L** dá um "pulo" , se afastando do endereço de escrita. Este "pulo" pode ser tanto para frente como para trás.

Figura 4.11 : Representação do SLIP CONTROLADO



4.3 Recepção Síncrona/Assíncrona

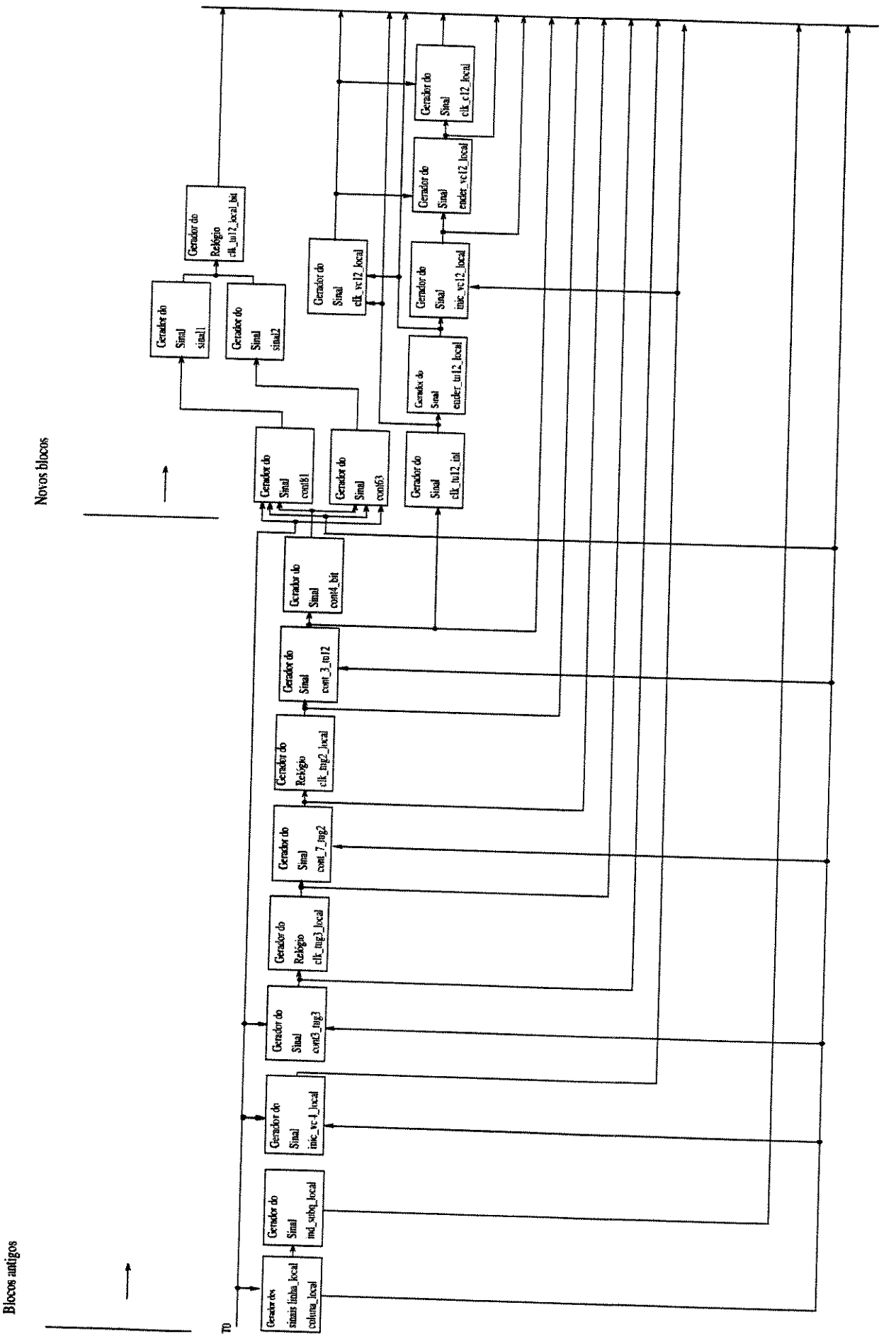
4.3.1 Introdução

Como na recepção síncrona de byte, descrita anteriormente, aqui o objetivo além da retirada dos bits de cabeçalho é também interpretar as possíveis justificações de mapeamento que podem ocorrer. Neste bloco tem-se uma análise bit a bit de um multiquadro VC12. Através da interpretação do POH do VC12 sabemos se os dados são síncronos ou assíncronos; depois, interpretando C1 e C2 , descobre-se a ocorrência de justificação positiva, justificação negativa ou se não ocorre justificação em S1 e S2. A seguir são mostrados os blocos que fazem este tratamento: Gerador de Sínais de Controle Local Bit, adaptador de VC12 na recepção na forma de bit, adaptação das rotas de baixa ordem síncona e assíncrona.

4.3.2 Gerador dos Sinais de Controle Local Bit

A figura 4.12 apresenta o detalhamento deste bloco. Este bloco é responsável juntamente com adapt_vc12_r_bit por "transformar" os sinais clk_vc12 e VC12 em clk_vc12_bit e VC12_bit. Vai-se, logo abaixo, descrever os sinais gerados por este bloco, descrevendo somente os sinais que nos interessam; os blocos que já foram projetados anteriormente[2] não serão descritos.

Figura 4.12 : Detalhamento do Bloco *Gerador dos Sinais de Controle Local Bit*



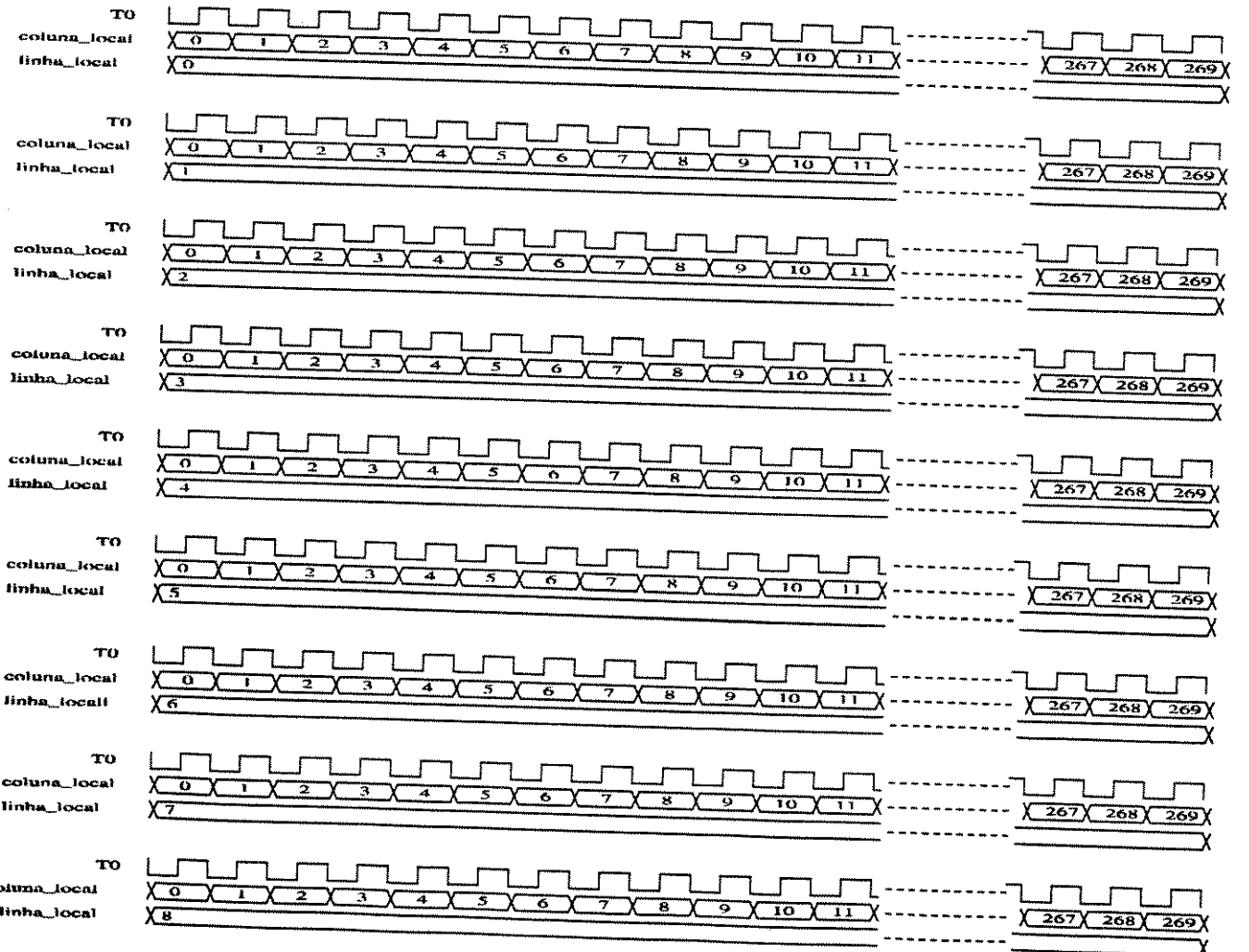
Gerador dos Sinais *linha_local* **e** *coluna_local*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_dos_Sinais_linha_local_e_coluna_local :
PROCESS(T0)
BEGIN
    IF (T0 = '0') AND (T0'EVENT) THEN
        IF (coluna_local = 269) THEN
            IF (linha_local = 8) THEN
                linha_local <= 0;
                coluna_local <= 0;
            ELSE
                linha_local <= linha_local + 1;
                coluna_local <= 0;
            END IF;
        ELSE
            coluna_local <= coluna_local + 1;
        END IF;
    END IF;
END PROCESS;
```

O diagrama temporal dos sinais *coluna_local* e *linha_local* está na figura 4.13. Observe naquela figura que as bordas de subida do relógio *T0* e os valores dos sinais *linha_local* e *coluna_local* marcam os instantes de inserção dos 2430 bytes do quadro STM-1 local a ser transmitido.

Figura 4.13 : Diagrama temporal dos sinais de *coluna_local* e *linha_local*



Gerador do Sinal *ind_subq_local*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_do_Sinal_ind_subq_local :
PROCESS(linha_local)
BEGIN
    IF (linha_local = 8) AND (linha_local'EVENT) THEN
       IF (ind_subq_local = 3) THEN
          ind_subq_local <= 0;
       ELSE
          ind_subq_local <= ind_subq_local + 1;
       END IF;
    END IF;
END PROCESS;
```

O diagrama temporal do sinal *ind_subq_local* está mostrado na figura 4.14. Observe que quando o sinal *linha_local* atinge o valor 8 o sinal *ind_subq_local* é incrementado ou zerado se o seu valor for 3. Isto mostra a formação de um multiquadro.

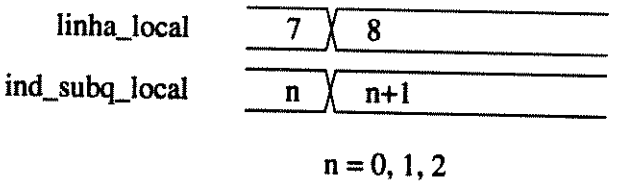


Figura 4.14 : Diagrama temporal do sinal *ind_subq_local*

Gerador do Sinal *cont_3_tug3*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_do_Sinal_cont_3_tug3 :
PROCESS(T0, linha_local)
BEGIN
    IF (T0 = '0') AND (T0'EVENT) THEN
       IF (cont_3_tug3 = 2) THEN
          cont_3_tug3 <= 0;
       ELSE
          cont_3_tug3 <= cont_3_tug3 + 1;
       END IF;
    END IF;
    IF (linha_local = 0) AND (linha_local'EVENT) THEN
       cont_3_tug3 <= 0;
    END IF;
END PROCESS;
```

A figura 4.15 mostra o diagrama temporal do sinal *cont_3_tug3* em relação ao sinais *linha_local* e *T0*. Observe que o sinal *cont_3_tug3* rotula os pulsos de *T0* com os valores 0, 1 e 2. No ponto no qual o sinal *linha_local* atinge o valor 0 se faz o alinhamento do sinal *cont_3_tug3*, isto é, seu valor deve ser zero durante o primeiro byte do quadro STM-1 local. Este sinal serve para identificar cada um dos três TUG3 dentro de um VC4.

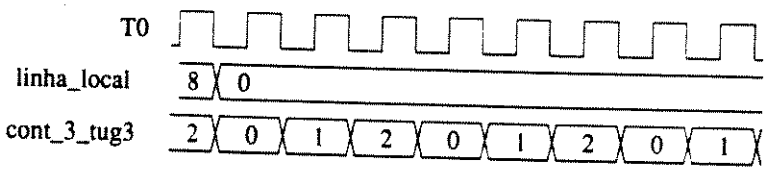


Figura 4.15 : Diagrama temporal do sinal *cont_3_tug3*

Gerador do Relógio *clk_tug3_local*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_do_Relógio_clk_tug3_local :
clk_tug3_local <= '1' WHEN (coluna_local > 11) AND (cont_3_tug3 = 0) ELSE
    '0';
```

A figura 4.16 mostra o diagram temporal do relógio *clk_tug3_local* em relação aos sinais *coluna_local*, *cont_3_tug3* e *T0*. A coluna acima de 11 mostra a "retirada" do SOH+POH+STUFF.

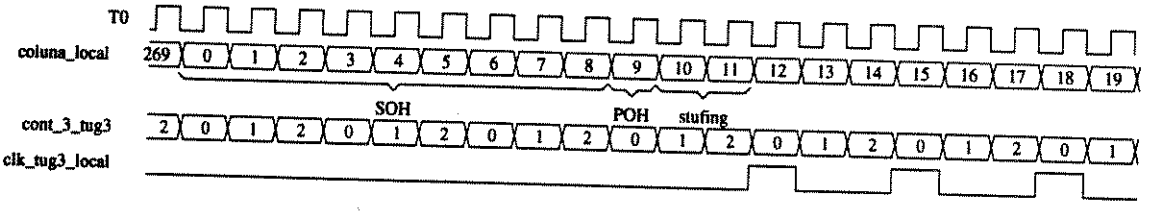


Figura 4.16 : Diagrama temporal do relógio *clk_tug3_local*

Gerador do Sinal *cont_7_tug2*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_do_Sinal_cont_7_tug2 :
PROCESS(clk_tug3_local, coluna_local)
BEGIN
    IF (coluna_local <= 15) OR (coluna_local >= 268) AND (coluna_local'EVENT) THEN
       cont_7_tug2 <= 7;
    END IF;
    IF (clk_tug3_local = '0') AND (clk_tug3_local'EVENT)AND
        (coluna_local > 14) AND (coluna_local < 267) THEN
       IF (cont_7_tug2 = 6) OR (cont_7_tug2 = 7)THEN
          cont_7_tug2 <= 0;
       ELSE
          cont_7_tug2 <= cont_7_tug2 + 1;
       END IF;
    END IF;
END PROCESS;
```

A figura 4.17 mostra o diagrama temporal do sinal *cont_7_tug2* em relação aos sinais *coluna_local* e *clk_tug3_local*.

Pela figura 3.3 vê-se que as 2 primeiras colunas do quadro TUG3 são de "stuffing". Por isso o sinal *cont_7_tug2* assume o valor 7 durante os 2 primeiros pulsos do relógio *clk_tug3_local* para informar que se trata de "stuffing". Este sinal identifica os 7 TUGs2 dentro de um TUG3.

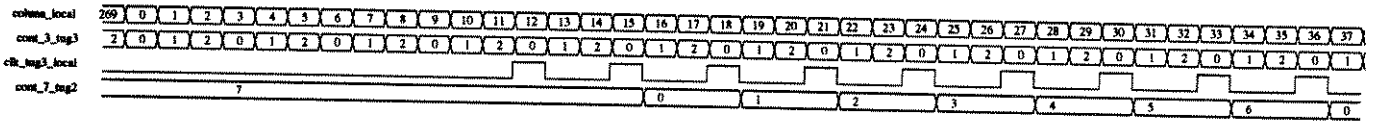


Figura 4.17 : Diagrama temporal do sinal *cont_7_tug2*

Gerador do Relógio *clk_tug2_local*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_do_Relógio_clk_tug2_local :
clk_tug2_local <= '1' WHEN (cont_7_tug2 = 0) ELSE
    '0';
```

A figura 4.18 mostra o diagrama temporal do relógio *clk_tug2_local* em relação ao sinal *cont_7_tug2* e ao relógio *clk_tug3_local*.

Observe que o relógio *clk_tug2_local* vai a nível lógico 1 quando o sinal *cont_7_tug2* assume o valor zero. Observe também que entre 2 bordas de subida consecutivas do relógio *clk_tug2_local*

ocorrem 7 bordas de subida do relógio *clk_tug3_local*. Tem-se até aqui um clock que identifica TUG3#0(TUG2#0).

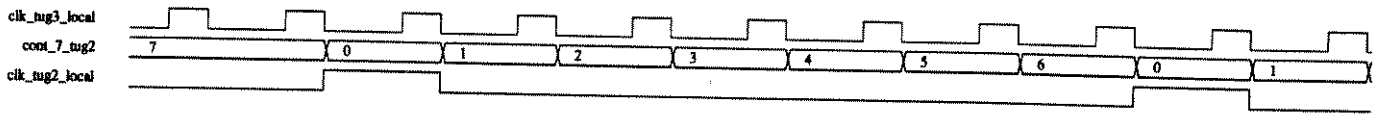


Figura 4.18 : Diagrama temporal do relógio *clk_tug2_local*

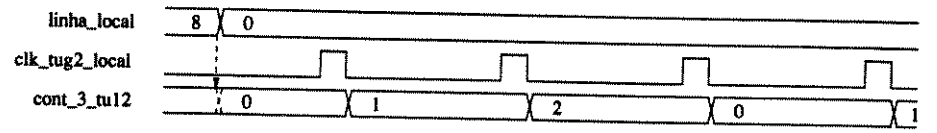
Gerador do Sinal *cont_3_tu12*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_do_Sinal_cont_3_tu12 :
PROCESS(clk_tug2_local, linha_local)
BEGIN
    IF (linha_local = 0) AND (linha_local'EVENT) THEN
       cont_3_tu12 <= 0;
    END IF;
    IF (clk_tug2_local = '0') AND (clk_tug2_local'EVENT) THEN
       IF (cont_3_tu12 = 2) THEN
          cont_3_tu12_bit <= 0;
       ELSE
          cont_3_tu12 <= cont_3_tu12 + 1;
       END IF;
    END IF;
END PROCESS;
```

A figura 4.19 mostra o diagrama temporal do sinal *cont_3_tu12* em relação aos sinais *linha_local* e *clk_tug2_local*. O ponto no qual o sinal *linha_local* assume o valor zero é usado para alinhar o sinal *cont_3_tu12*. Neste ponto o sinal *cont_3_tu12* deve ter valor zero. A cada borda de descida do relógio *clk_tug2_local* o sinal *cont_3_tu12* é incrementado ou assume o valor zero se o seu valor for 2. Aqui tem-se a identificação de 3 TUs12 dentro de um TUG2.

Figura 4.19 : Diagrama temporal do sinal *cont_3_tu12*



Gerador do sinal *cont4_bit*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_do_Sinal_cont4_bit:
PROCESS(cont_3_tu12)
BEGIN
   IF(cont_3_tu12=0)THEN
     IF(cont4_bit=3)THEN
        cont4_bit<=0;
     ELSE
        cont4_bit<=cont4_bit+1;
     END IF;
  END IF;
END PROCESS;
```

Este sinal serve para identificar quatro clocks de um TU12 em uma linha, podendo-se identificar sinais que possuem tempos diferentes. Logo à frente observaremos a necessidade deste sinal. A figura 4.20 mostra o diagrama temporal do sinal *cont4_bit* em função do *cont_3_tu12*.

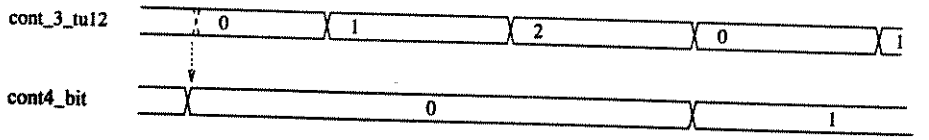


Figura 4.20 : Diagrama temporal do sinal *cont4_bit*

4.3.3 Gerador dos *sinais cont81* e *cont63*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_dos_contadores_cont81_e_cont63:
PROCESS(T0,cont4_bit)
BEGIN
   IF(T0='0')AND(cont4_bit=2)THEN
      IF(cont81=81)THEN
         cont81<=0;
      ELSE
         cont81<=cont81+1;
      END IF;
   END IF;
   IF(T0='0')AND(cont4_bit≠2)THEN
      IF(cont63=63)THEN
         cont63<=0;
      ELSE;
         cont63<=cont63+1;
      END IF;
   END IF;
   IF(linha_local=0)AND(cont4_bit'EVENT)THEN
         cont81<=0;
   END IF;
   IF(linha_local=0)AND(cont4_bit'EVENT)THEN
         cont63<=0;
   END IF;
END PROCESS;
```

Tem-se como objetivo gerar um sinal na forma de clk_tu12_bit, para identificar cada bit de um quadro VC12. Em uma numeração contínua de um AU4, sabe-se que a distância entre os bytes de um mesmo TU12 é de 63, para 3 destes bytes, o quarto byte fica a uma distância maior, pois deve-se descontar os bytes de SOH, enchimento e POH: Então; 18+63=81. Os sinais cont63 e cont81 identificam cada bit de um TU12. A figura 4.21 mostra o diagrama temporal destes sinais em função de *T0* e *cont4_bit*. Observe que o sinal *linha_local=0* faz o alinhamento destes endereços. Estes 18 bytes correspondem a: 9 de SOH + 1 POH + 2 STUFFING + 6 STUFFING.

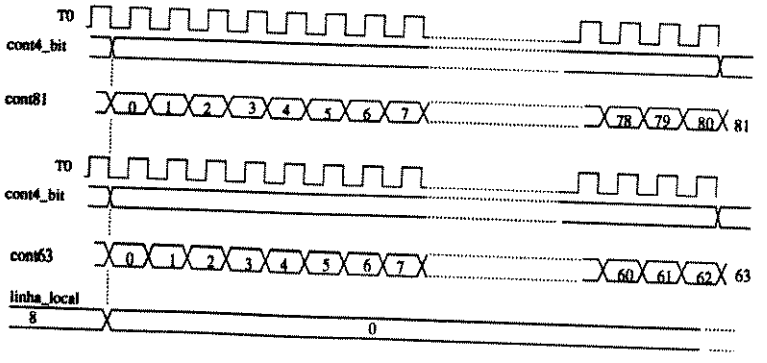


Figura 4.21 : Diagrama temporal dos sinais *cont81 e con63*

Gerador de *sinal1* e *sinal2*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_sinal1_sinal2:
  sinal1<='1' WHEN(cont81>=5)AND(cont81<10)ELSE
      '1' WHEN(cont81>=15)AND(cont81<20)ELSE
      '1' WHEN(cont81>=25)AND(cont81<30)ELSE
      '1' WHEN(cont81>=35)AND(cont81<40)ELSE
      '1' WHEN(cont81>=45)AND(cont81<50)ELSE
      '1' WHEN(cont81>=55)AND(cont81<60)ELSE
      '1' WHEN(cont81>=65)AND(cont81<70)ELSE
      '1' WHEN)cont81>=75)AND(cont81<80)ELSE
      '0';
  sinal2<='1'WHEN(cont63>=4)AND(cont63<8)ELSE
      '1' WHEN(cont63>=12)AND(cont63<16)ELSE
      '1' WHEN(cont63>=20)AND(cont63<24)ELSE
      '1' WHEN(cont63>=28)AND(cont63<32)ELSE
      '1' WHEN(cont63>=36)AND(cont63<40)ELSE
      '1' WHEN(cont63>=44)AND(cont63<48)ELSE
      '1' WHEN(cont63>=52)AND(cont63<56)ELSE
      '1' WHEN(cont63>=60)AND(cont63<62)ELSE
      '0';
```

Estes dois sinais: sinal1 e sinal2 fornecem 8 clocks dentro de cada byte de um TU12 . A figura 4.22 mostra o diagrama temporal destes dois sinais em função de *cont81* e *cont63*.

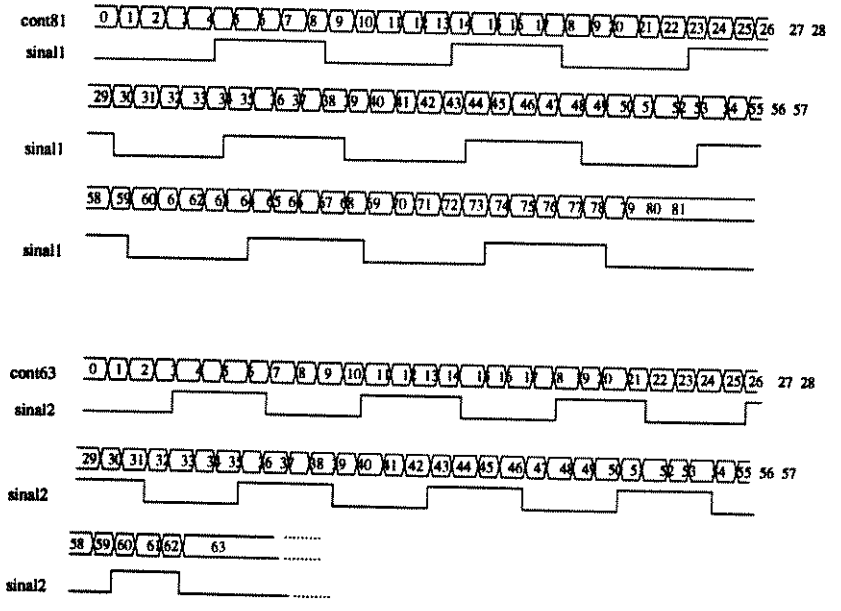


Figura 4.22 : Diagrama temporal de *sinal1 e sinal2*

Gerador do sinal *clk_tu12_local_bit*

Modelo em VHDL:

```
Gerador_clk_tu12_local_bit:
PROCESS(sinal1,sinal2)
BEGIN
   clk_tu12_local_bit<=sinal1 OR sinal2;
END PROCESS;
```

A figura 4.23 mostra o diagrama temporal do relógio de TU12 na forma de bit, gerado através de sinal1 e sinal2. Tem-se aqui um clock de bit dentro de um TUG3#0(TUG2#0(TU12#0)).

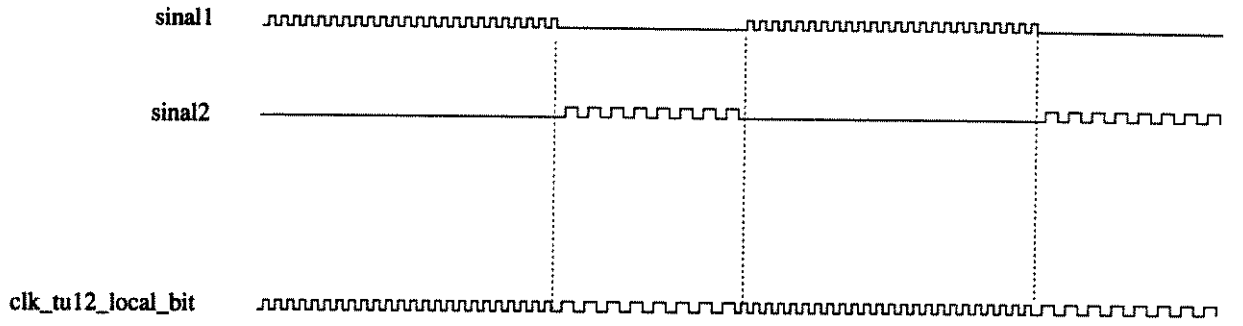


Figura 4.23 : Diagrama temporal do *clk_tu12_local_bit*

Gerador do Relógio *clk_tu12_int*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_do_Relógio_clk_tu12_int :
clk_tu12_int <= '1' WHEN (cont_3_tu12 = 0) ELSE
    '0';
```

A figura 4.24 mostra o diagrama temporal do relógio *clk_tu12_int* em relação aos sinais *cont_3_tu12* e *clk_tug2_local*. Observe que o relógio *clk_tu12_int* vai a nível lógico 1 quando o sinal *cont_3_tu12* assume o valor zero.

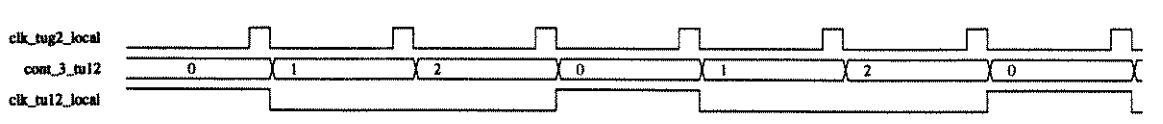


Figura 4.24 : Diagrama temporal do relógio *clk_tu2_int*

Pode-se observar na figura 4.25 a relação existente entre o clk_tu12_int*(clk de byte)* com o clk_tu12_local_bit*(clk de bit)*. Este sinal fará com que se possa analisar um VC12 em forma de bit.

Figura 4.25 : Diagrama temporal do relógio *clk_tu2_int x clk_tu12_local_bit*

Gerador do Sinal *ender_tu12_local*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_do_Sinal_ender_tu12_local :
PROCESS(clk_tu12_int, linha_local)
BEGIN
    IF (linha_local = 3) AND (linha_local'EVENT) THEN
       ender_tu12_local <= 0;
    END IF;
    IF (clk_tu12_int = '0') AND (clk_tu12_int'EVENT) THEN
       IF (ender_tu12_local = 35) THEN
          ender_tu12_local <= 0;
       ELSE
          ender_tu12_local <= ender_tu12_local + 1;
       END IF;
    END IF;
END PROCESS;
```

A figura 4.26 mostra o diagrama temporal do sinal *ender_tu12_local* em relação aos sinais *linha_local* e *coluna_local*. Observe que o ponto no qual o sinal linha_local muda seu valor de 2 para 3 serve para alinhar o sinal *ender_tu12_local*(assume o valor zero neste ponto). A cada borda de descida do relógio *clk_tu12_int* o sinal *ender_tu12_local* é incrementado ou zerado se o seu valor for 35.

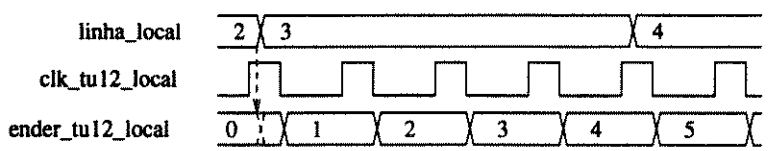


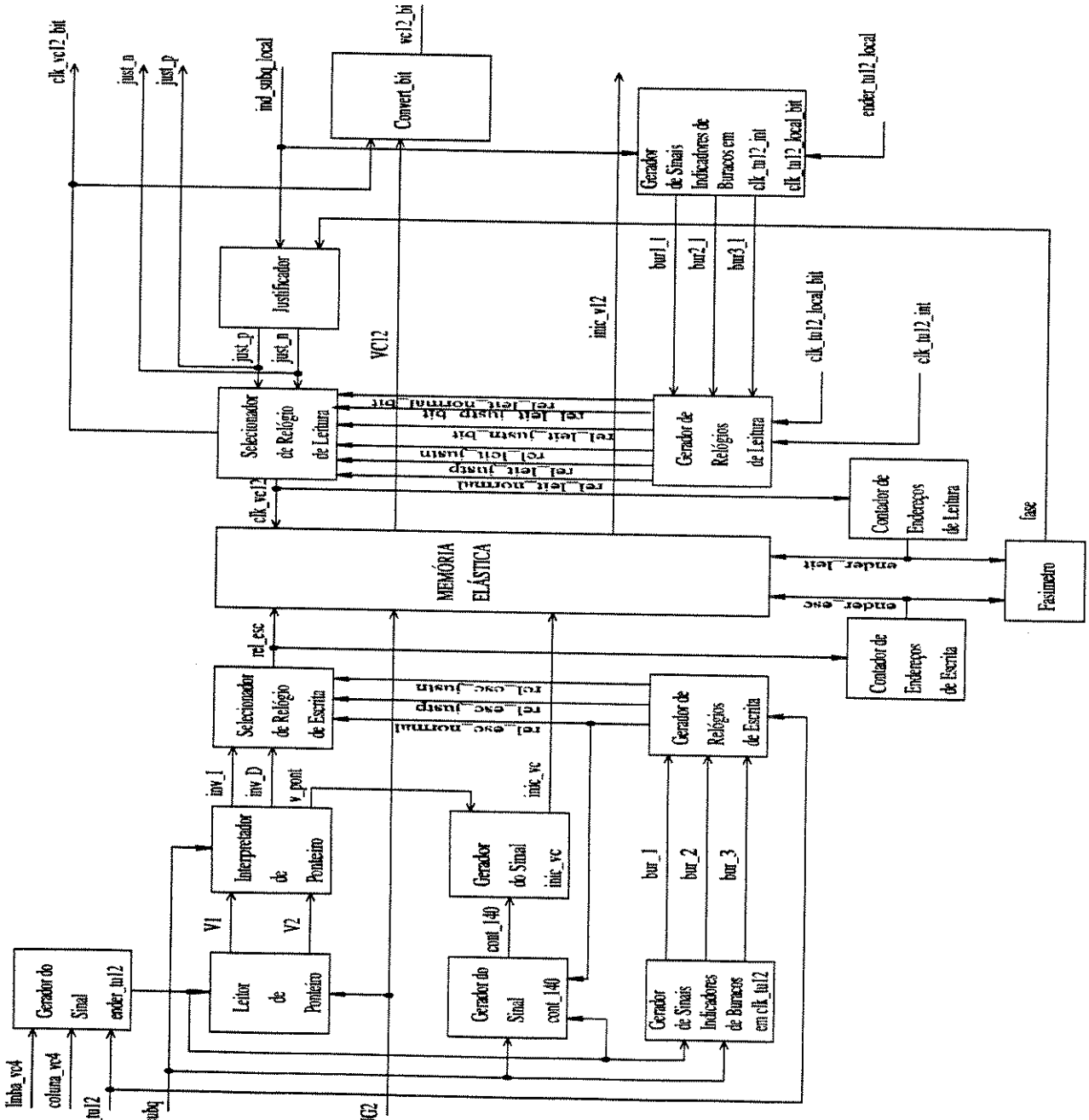
Figura 4.26 : Diagrama temporal do sinal *ender_tu12_local*

4.3.4 Adaptador de VC12 na Recepção para a Forma de Bit

4.3.5 Introdução

O bloco a seguir faz a adaptação de um TU12 para a retirada de um VC12; este bloco já foi projetado anteriormente[2] e aqui fez-se uma adaptação para que os dados retirados, apartir dele, fossem em forma de bit. Neste bloco ocorre a leitura de V1 e V2 a sua interpretação e um possível processo de justificação em V3. Do lado direito do bloco tem-se a retirada de um VC12 como bit e como byte, existe um processamento paralelo entre bit e byte para que os dados possam sair sincronizados deste bloco. Em conjunto com o gerador de sinais de controle local bit ele "manda" para o próximo bloco os dados de VC12, o início de VC12 e o clock do VC12. Os sinais na forma de VC12 podem ser retransmitidos para dentro dos equipamentos ou detectados localmante. A figura 4.27 mostra o seu diagrama de blocos.

Figura 4.27 : Detalhamento do bloco *adaptador de VC-12 para bit*



Gerador dos Sinais Indicadores de Buracos *clk_tu12_local_bit e clk_tu12_int*

- Este bloco é modelado pelo processo abaixo.

Gerador_dos_Sinais_Indicadores_de_Buracos em clk_tu12_local_bit e clk_tu12_int:
bur1_l <='1' WHEN ender_tu12_local = 0 AND ind_subq_local ≠2 ELSE
'0';
bur2_l <='1' WHEN ender_tu12_local = 0 AND ind_subq_local = 2 ELSE
'0';
bur3_l <='1' WHEN ender_tu12_local = 1 AND ind_subq_local = 2 ELSE
'0';

Conforme mostra o diagrama temporal na figura 4.28, os sinais bur1_l,bur2_l e bur3_l servem para identificar as posições onde ficam V1,V2,V3 e V4 e as posições para justificação de byte.

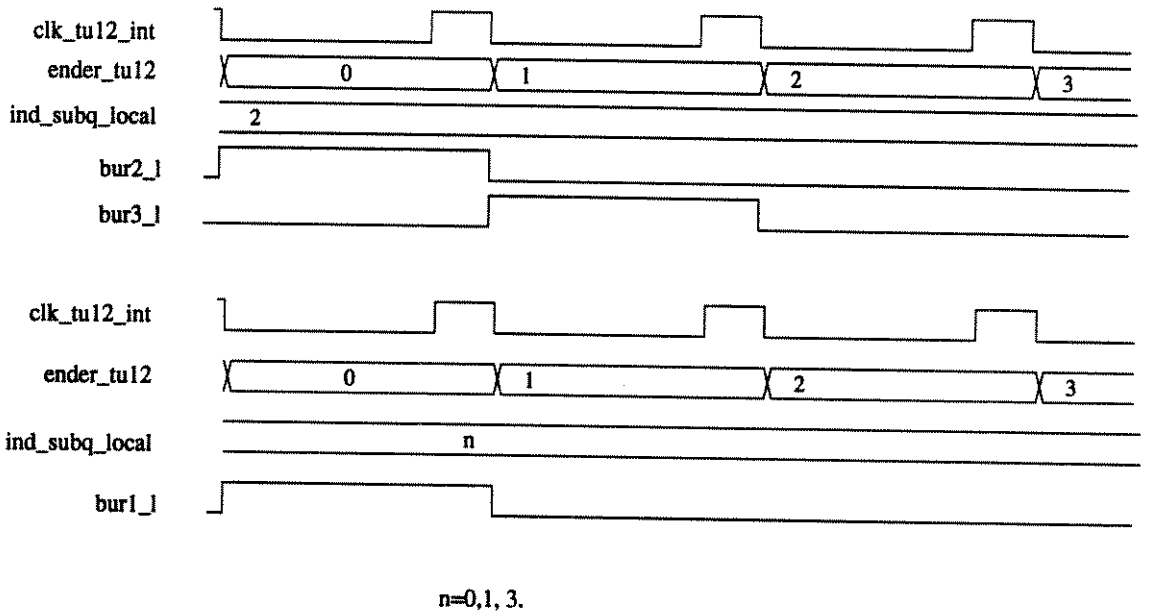


Figura 4.28 : Diagrama temporal dos *buracos*

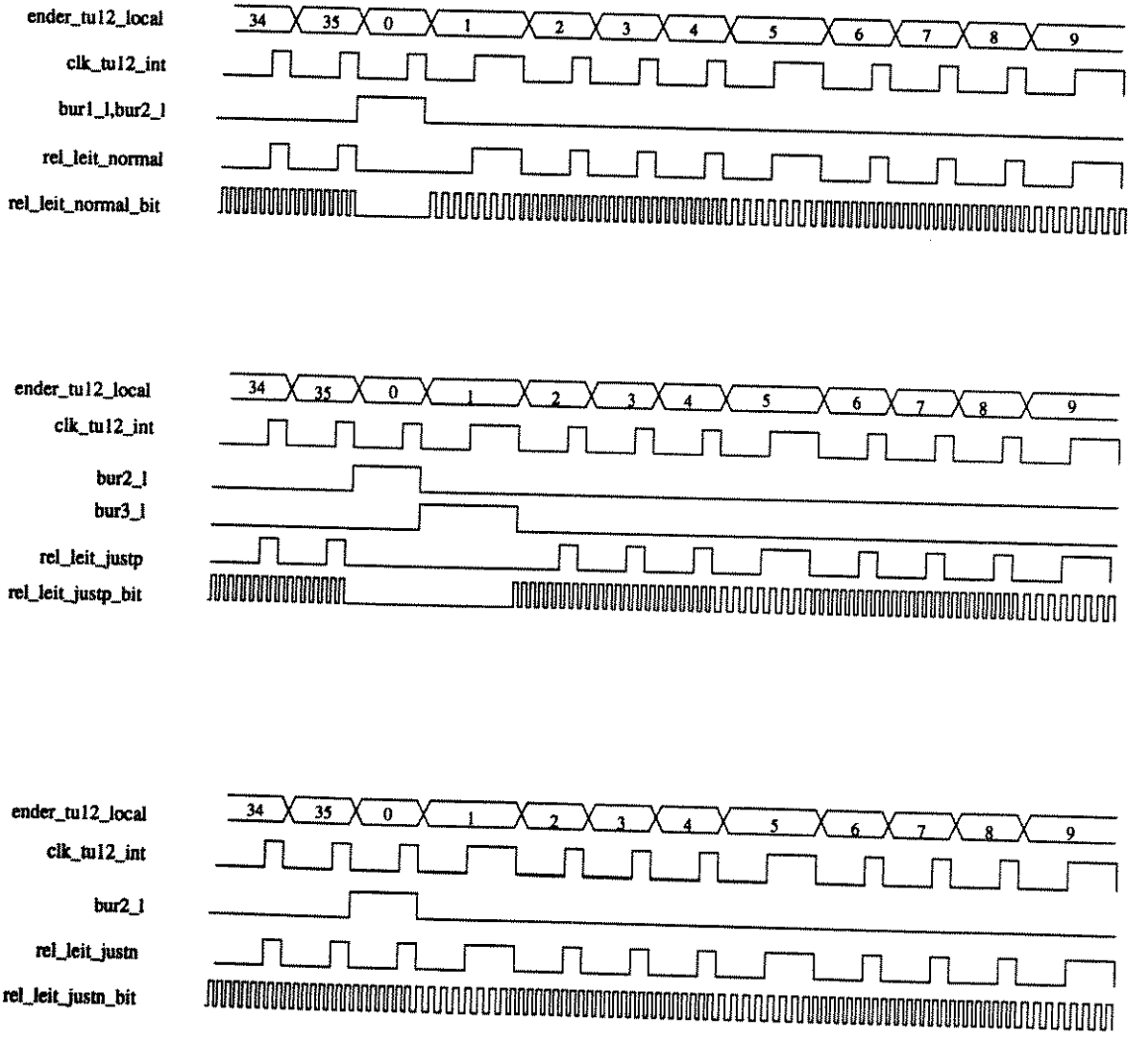
Gerador de Relógios de Leitura

- Este bloco é modelado pelo processo abaixo.

Gerador_Relógios_de_Leitura:
rel_leit_normal <= clk_tu12_int AND NOT(bur1_l) AND NOT(bur2_l);
rel_leit_justp <= clk_tu12_int AND NOT(bur1_l) AND NOT(bur2_l) AND NOT(bur3_l);
rel_leit_justn <= clk_tu12_int AND NOT(bur1_l);
rel_leit_normal_bit <= clk_tu12_local_bit AND NOT(bur1_2l) AND NOT(bur2_l);
rel_leit_justp_bit <= clk_tu12_local_bit AND NOT(bur1_l) AND NOT(bur2_l) AND NOT(bur3_l);
rel_leit_justn_bit <= clk_tu12_local_bit AND NOT(bur1_l);

Este bloco gera sinais para a leitura de dados na memória elástica em função dos clks de TU12 e dos buracos que determinam as posições das justificações e da retirada de um byte do TU12 para formar o VC12. Conforme o diagrama temporal na figura 4.29 os sinais gerados *(clk_leit e clk_leit_bit)* são muito semelhantes, ou seja, possuem buracos nas mesmas posições.

Figura 4.29 : Diagrama temporal dos relógios de *leitura*



Selecionador de Relógios de Leitura

- Este bloco é modelado pelo processo abaixo.

```
Selecionador_Relógios_de_Leitura:
   clk_vc12 <= rel_leit_justn WHEN just_n = '1' ELSE
               rel_leit_justp WHEN just_p = '1' ELSE
               rel_leit_normal;
   clk_vc12_bit <= rel_leit_justn_bit WHEN just_n='1' ELSE
               rel_leit_justp_bit WHEN just_p = '1' ELSE
               rel_leit_normal_bit;
```

Este bloco seleciona o clk que irá retirar os dados da memória elástica, em função das justificações *(just_n,just_p)*. Tem-se o clk_vc12 que retira os dados da memória na forma de byte e tem-se o clk_vc12_bit que varia de acordo com as justificações que possam ocorrer. Observe que isto "amarra" os dois clks *(clk_vc12 com clk_vc12_bit)*. É usado o clk_vc12 para a retirada dos dados da memória já que esta, recebe dados na forma de byte. O clk_vc12_bit será usado pelos blocos mais a frente.

Justificador

- Este bloco é modelado pelo processo abaixo.

```
Justificador:
PROCESS(ind_subq_local)
BEGIN
   IF(ind_subq_local = 3) AND (ind_subq_local'EVENT)THEN
      IF(fase <=2)THEN
         just_p<='1';
      ELSE
         just_p<='0';
      END IF;
      IF (fase >=6)THEN
         just_n<='1';
      ELSE
         just_n<='0';
      END IF;
   END IF;
END PROCESS;
```

Este bloco faz com que ocorram justificações em função dos valores de fase *(memória enchendo ou esvaziando)*. Possuem um limiar inferior de 2 e um superior de 6 como pode ser observado acima.

Convert_bit

- Este bloco é modelado pelo processo abaixo.

```
Convert_bit:
PROCESS(clk_vc12_bit,inic_vc12)
VARIABLE contador:INTEGER:=1;
BEGIN
   IF(clk_vc12_bit='0')AND(clk_vc12_bit'EVENT)THEN
      VC12_bit<=VC12(contador);
         IF(contador=8)THEN
            contador:=1;
         ELSE
            contador:=contador+1;
         END IF;
   END IF;
   IF(inic_vc12='1')AND(inic_vc12'EVENT)THEN
      contador:=1;
   END IF;
END PROCESS;
```

Apartir deste bloco, tem-se o sinal na forma de bit. Utiliza-se aqui uma variavel chamada *contador* para converter os dados de byte para bit, onde cada dado será lido bit a bit. O inic_vc12, serve para alinhar os dados a serem convertidos, (neste instante a variável é ajustada). A figura 4.30 mostra o diagrama temporal destes sinais com mais detalhes.

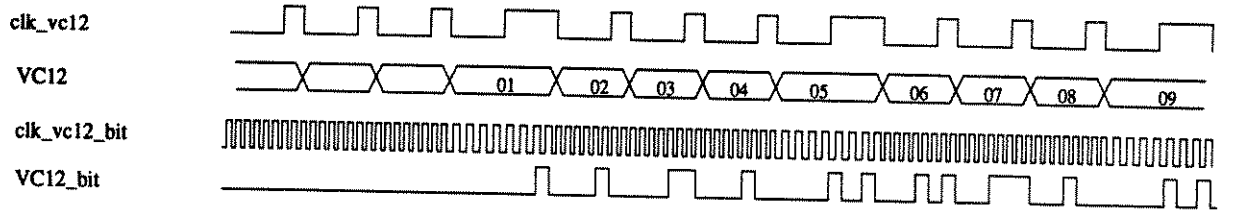


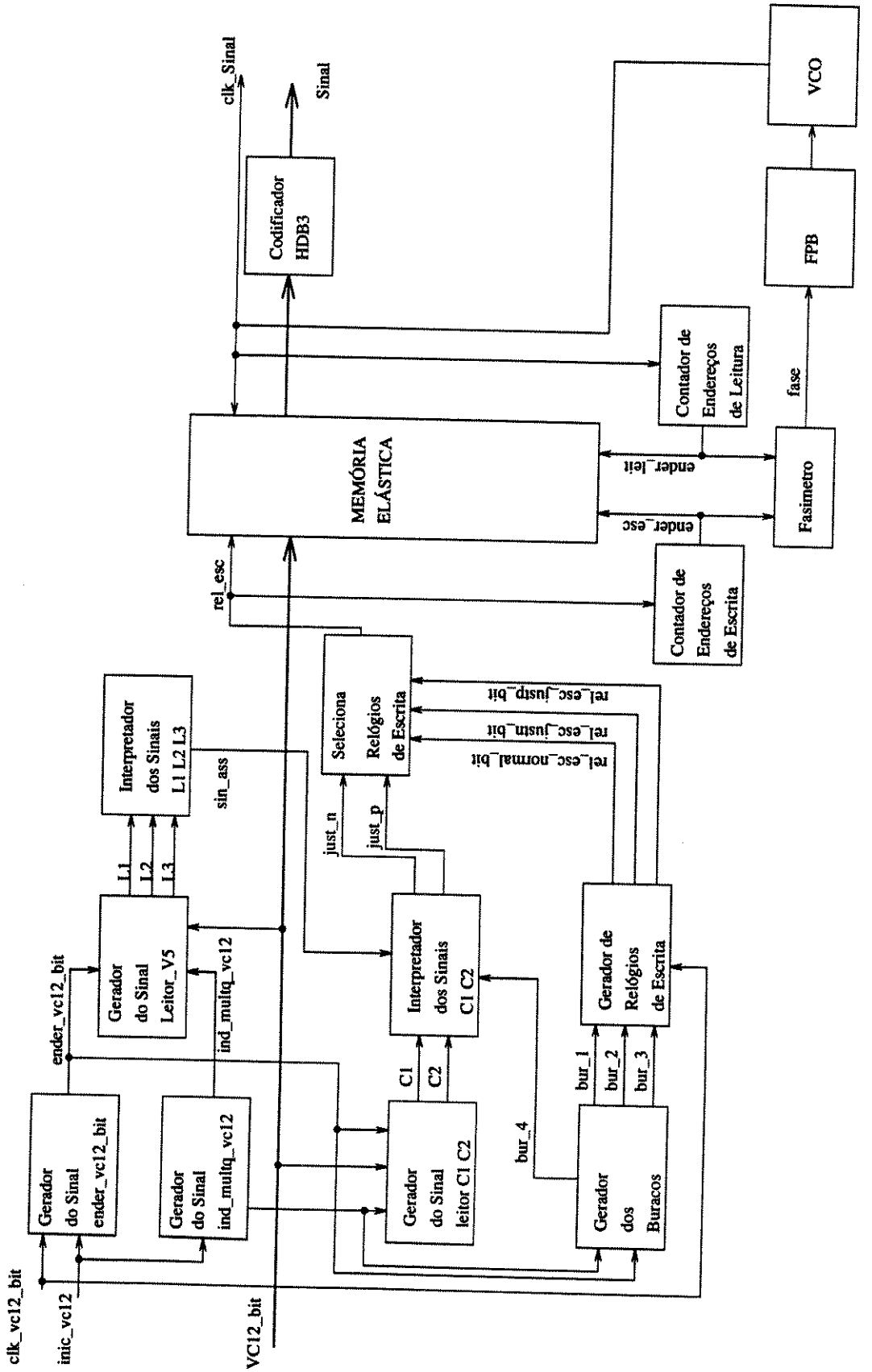
Figura 4.30 : Diagrama temporal do *vc12_bit e clk_vc12_bit*

4.3.6 Adaptação das Rotas de Baixa Ordem na Forma Síncrona e Assíncrona

Introdução

Em linhas gerais, este bloco interpreta os sinais do VC12 na forma de bit e libera para a saída um sinal de 2048 kb/s . A esquerda da memória elástica tem-se os blocos que interpretam os dados do VC12. Inicialmente é feita uma geração de endereços para um quadro VC12, gerados para que fosse possível identificar cada elemento dentro de um quadro. Mas são necessárias interpretações em um multiquadro, por isto tem-se o sinal *indicador de multiquadro VC12.* Estes são os sinais básicos para a interpretação de cada bit dentro de um VC12. A partir daí tem-se os blocos que interpretam estes sinais possibilitando a escrita correta dos dados na memória. O bloco do fasímetro faz uma comparação entre o endereço de escrita e o de leitura para verificar as velocidades dos respectivos relógios. A direita da memória elástica tem-se um PLL que não foi implementado, bem como o bloco Codificador-HDB3. Nesta parte do trabalho, descreve-se a retirada de sinais na forma de serial-bit, utilizando como referência os mapeamentos descritos no Capítulo 2*(Mapeamento assíncrno e síncrono de 2Mb/s em modo flutuante).* O diagrama de blocos mostrado na figura 4.31 está o bloco responsável por este "desmapeamento". A seguir vai-se descrever cada um dos blocos que compõe o lpa_r_sin_ass.

Figura 4.31 : Detalhamento do Bloco *lpa_r_sin_ass*



Bloco *Gerador_de_ender_vc12*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_do_Sinal_ender_vc12:
PROCESS(inic_vc12, clk_vc12_bit)
BEGIN
    IF (clk_vc12_bit='0') AND (clk_vc12_bit'EVENT) THEN
      IF (ender_vc12_bit=279)THEN
          ender_vc12_bit<=0;
       ELSE
          ender_vc12_bit<=ender_vc12_bit+1;
       END IF;
    END IF;
   IF (inic_vc12='1')AND(inic_vc12'EVENT)THEN
          ender_vc12_bit<=0;
    END IF;
END PROCESS;
```

Este bloco gera endereços para cada bit de um quadro VC12; isto possibilita fazer a identificação do dado que está chegando. O sinal *inic_vc12* serve para alinhar os endereços de um VC12. A figura 4.32 mostra o diagrama temporal de como estes endereços são gerados, lembrando que em um quadro VC12 tem-se 280 bits(35 bytes).

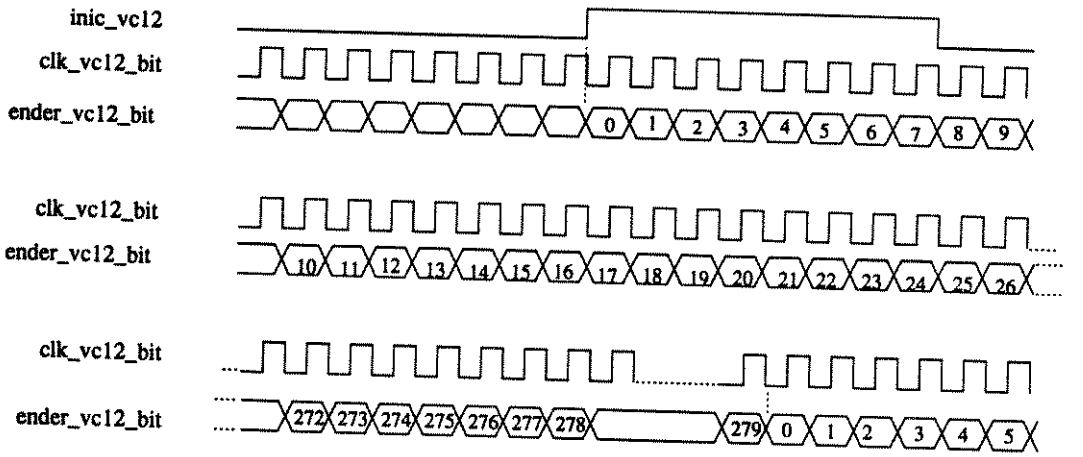


Figura 4.32 : Diagrama temporal da geração do *ender_vc12_bit*

Bloco *Gerador_do_ind_multq_vc12*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_do_Sinal_ind_multq_vc12:
PROCESS(ender_vc12_bit,inic_vc12)
BEGIN
    IF (ender_vc12_bit=1) AND (ender_vc12_bit'EVENT) THEN
      IF (ind_multq_vc12=4)THEN
          ind_multq_vc12<=1;
       ELSE
          ind_multq_vc12<=ind_multq_vc12+1;
       END IF;
    END IF;
   IF (inic_vc12='1')AND(inic_vc12'EVENT)THEN
          ind_multq_vc12<=0;
    END IF;
END PROCESS;
```

Cada quadro, dentro de um multiquadro, possui um cabeçalho diferente. Para fazer a identificação de quadro por quadro, criou-se este sinal. A figura 4.33 mostra o diagrama temporal deste sinal em função do *ender_vc12_bit* e *inic_vc12*. Após a variação total dos endereços de um VC12, o indicador de multiquadro é incrementado. O inic_vc12 serve para alinhar este sinal, acertando a posição de V5.

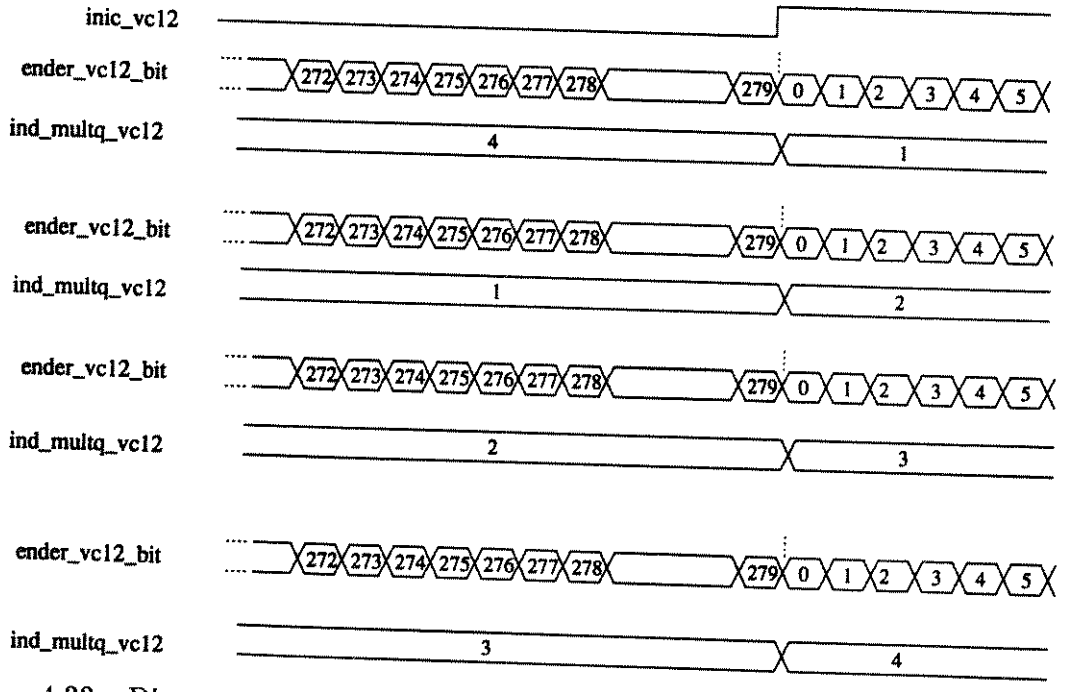


Figura 4.33 : Diagrama temporal do *ind_multq_vc12*

Bloco *leitor_de_v5*

- Este bloco é modelado pelo processo abaixo.

```
leitor_de_v5:
PROCESS(VC12_bit,ind_multq_vc12,ender_vc12_bit,clk_vc12_bit)
BEGIN
    IF (clk_vc12_bit='1') THEN
      IF (ender_vc12_bit=5)AND(ind_multq_vc12=1)THEN
         L1<=VC12_bit;
       ELSIF(ender_vc12_bit=6)AND(ind_multq_vc12=1)THEN
         L2<=VC12_bit;
       ELSIF(ender_vc12_bit=7)AND(ind_multq_vc12=1)THEN
         L3<=VC12_bit;
       ELSIF(ender_vc12_bit=0)AND(ind_multq_vc12=4)THEN
         L1<='0';
         L2<='0';
         L3<='0';
       END IF;
    END IF;
END PROCESS;
```

A figura 4.34 mostra o diagrama temporal de L1,L2 e L3. Na subida do *clk_vc12_bit* o *ender_vc12_bit* e o *ind_multq_vc12* são verificados e faz-se a leitura dos dados de *VC12_bit*. Nos endereços 5,6 e 7 os dados são lidos. No final de um multiquadro os sinais L1,L2 e L3 são zerados.Este leitor de V5 serve para retirar dados do POH do VC12 a fim de se saber que tipo de dado está chegando no receptor.

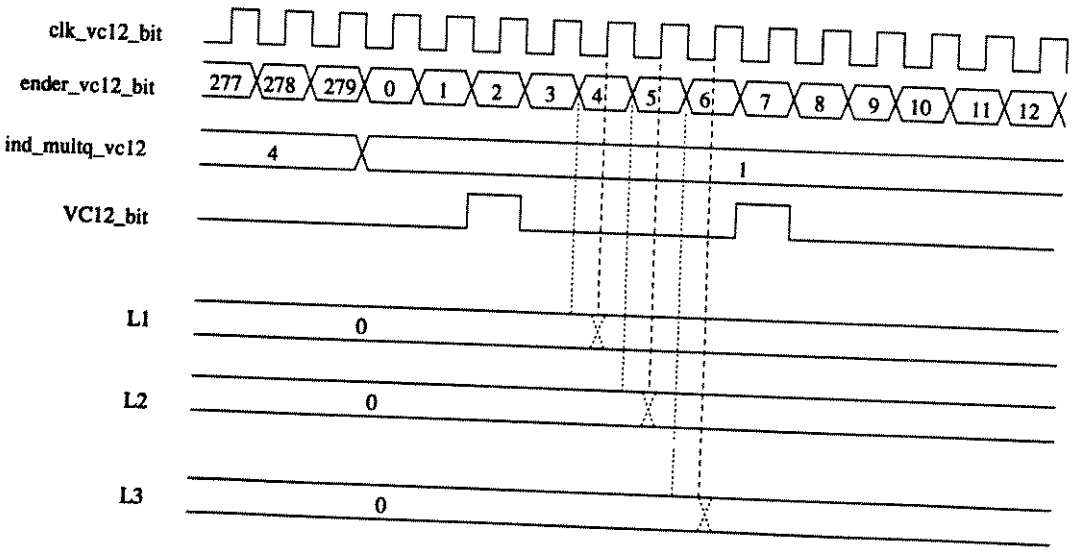


Figura 4.34 : Diagrama temporal de L1,L2 e L3.

Bloco *Interpreta_ponteiro_v5*

- Este bloco é modelado pelo processo abaixo.

```
interpreta_v5:
PROCESS(L1,L2,L3)
BEGIN
   IF(L1='0')AND(L2='1')AND(L3='0')THEN
      ass<='1';
   ELSE
      ass<='0';
   END IF;
END PROCESS;
```

A figura 4.35 mostra o diagrama temporal do interpretador de ponteiro, analisando-se o caso de transmissão assíncrona. O sinal *ass* depende dos valores de *L1,L2 e L3.*

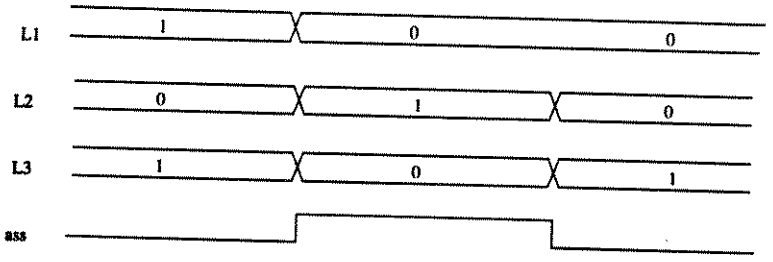


Figura 4.35 : Diagrama temporal do *Interpretador de V5*

Bloco *leitor_de_C1C2*

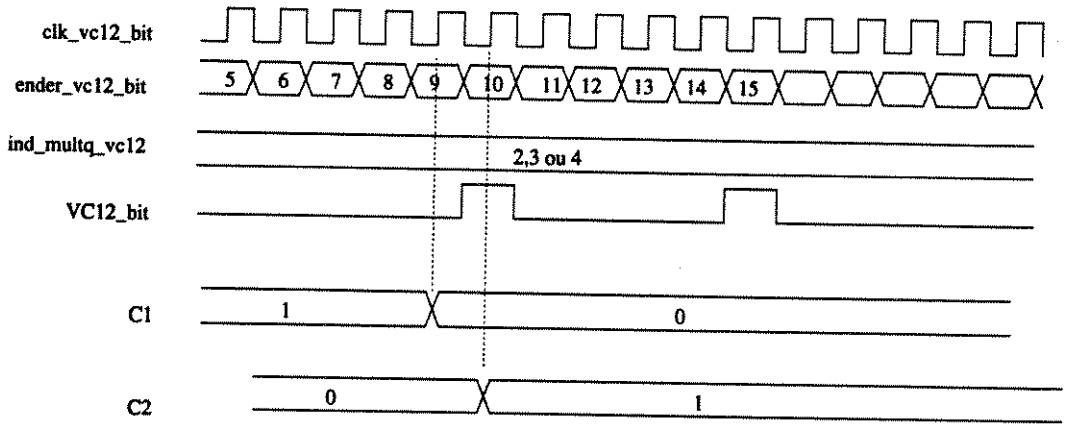
- Este bloco é modelado pelo processo abaixo.

```
leitor_de_C1C2:
PROCESS(ender_vc12_bit,VC12_bit,ind_multq_vc12,clk_vc12_bit)
BEGIN
   IF(clk_vc12_bit='1')AND(clk_vc12_bit'EVENT)THEN
      IF(ind_multq_vc12≠1)THEN;
         CASE ender_vc12_bit IS
            WHEN 9 => C1 <=VC12_bit;
            WHEN 10 => C2 <=VC12_bit;
            WHEN OTHERS => NULL;
         END CASE;
      END IF;
   END IF;
END PROCESS;
```

A figura 4.36 mostra o diagrama temporal do bloco *leitor_de_C1C2* em função do *ender_vc12_bit, VC12_bit, ind_multq_vc12 e clk_vc12_bit.* Na subida do *clk_vc12_bit* se o *ind_multq_vc12* for diferente

de 1 *(tem-se C1 e C2 somente nos três ùltimos quadros)* o dado de *VC12_bit* é passado para *C1* ou *C2* conforme o *ender_vc12_bit.*

Figura 4.36 : Diagrama temporal do *leitor de C1 C2*



Bloco *Gerador_de_Buracos*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_de_bur_1,bur_11,bur_12,bur_2,bur_3,bur_4:
   bur_1<='1' WHEN ender_vc12_bit>=1 AND ender_vc12_bit <=16 ELSE
      '0';
   bur_11<='1' WHEN ender_vc12_bit>=273 AND ender_vc12_bit <=279 ELSE
      '0';
   bur_12<='1' WHEN ender_vc12_bit=0 ELSE
      '0';
   bur_2<='1' WHEN ender_vc12_bit=16 AND ind_multq_vc12 =4 ELSE
      '0';
   bur_3<='1' WHEN ender_vc12_bit=17 AND ind_multq_vc12 =4 ELSE
      '0';
   bur_4<='1' WHEN ender_vc12_bit>=9 AND ind_multq_vc12 ≠1 ELSE
      '0';
```

Este bloco fornece sinais que serão usados para a retirada dos bits do cabeçalho de um VC12 e dos bits de enchimento. A figura 4.37 mostra o diagrama temporal destes sinais.

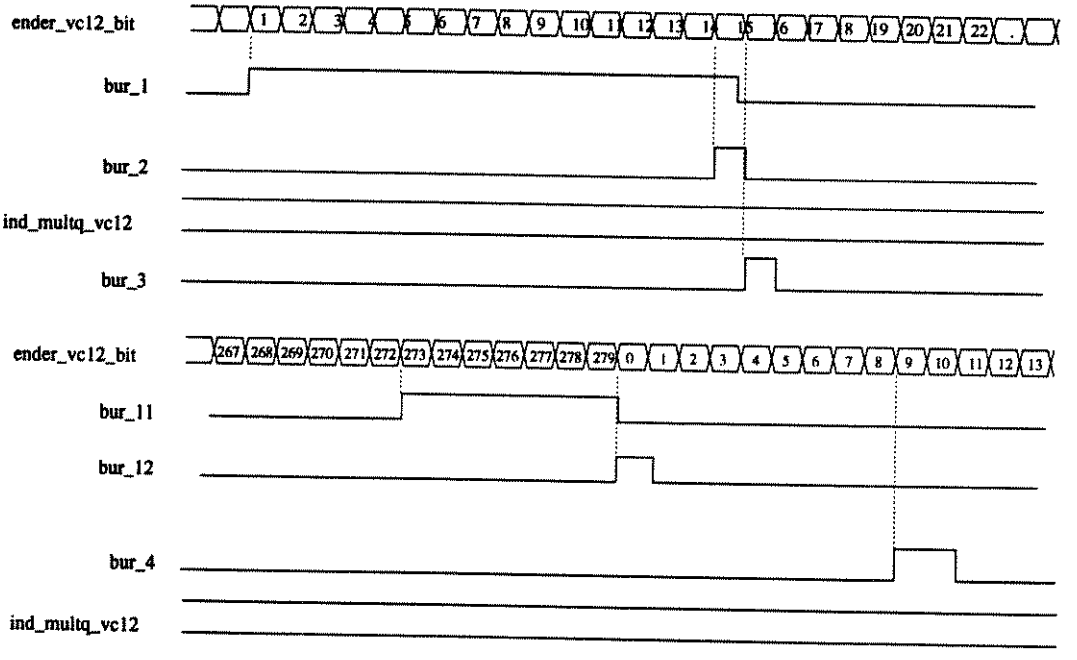


Figura 4.37 : Diagrama temporal dos *Buracos de escrita*

Bloco *Interpretador_de_C1C2*

- Este bloco é modelado pelo processo abaixo.

```
leitor_de_C1C2:
PROCESS(ender_vc12_bit,bur_4,ind_multq_vc12)
BEGIN
   IF(bur_4='1')AND(bur_4'EVENT)THEN
      IF(C1='0')THEN;
         int1<=int1+1;
      ELSIF(C1='1')THEN
            just_n<='0';
      END IF;
   END IF;
   IF(bur_4='1')AND(bur_4'EVENT)THEN
      IF(C2='1')THEN;
         int2<=int2+1;
      ELSIF(C2='0')THEN
            just_p<='0';
      END IF;
   END IF;
   IF(ass='0')THEN
            just_n<='0';
            just_p<='0';
   END IF;
   IF(ind_multq_vc12=1)THEN
            just_n<='0';
            just_p<='0';
   END IF;
   IF(ind_multq_vc12=1)THEN
         int1<=0;
         int2<=0;
   END IF;
   IF(int1>=2)THEN
            just_n<='1';
   END IF;
   IF(int2>=2)THEN
            just_p<='1';
   END IF;
END PROCESS;
```

A figura 4.38 mostra os sinais *ender_vc12_bit,bur_4* e *ind_multq_vc12* que atuam sobre os sinais de justificação. O sinal *bur_4* identifica o local onde tem-se C1 e C2. Os sinais internos *int1* e *int2* são alterados em função de uma votação majoritária em C1 e C2. Se os dados são síncronos não ocorrem justificações. Quando um novo multiquadro inicia, as justificações são zeradas. Isto que foi descrito segue as regras de interpretação de C1 e C2, vistas no capítulo 2.

Bloco *Gerador_dos_rel_escrita*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_rel_esc:
   rel_esc_normal<=(clk_vc12_bit)AND NOT(bur_1) AND NOT(bur_11) AND NOT(bur_12);
   rel_esc_just_n<=((clk_vc12_bit)AND NOT(bur_1) AND NOT(bur_11) AND NOT(bur_12))
            OR NOT(bur_2);
   rel_esc_just_p<=((clk_vc12_bit)AND NOT(bur_1) AND NOT(bur_11) AND NOT(bur_12))
            OR NOT(bur_3);
```

A figura 4.39 mostra o diagrama temporal dos relógios de escrita na memória elástica em função dos buracos e do *clk_vc12_bi.*

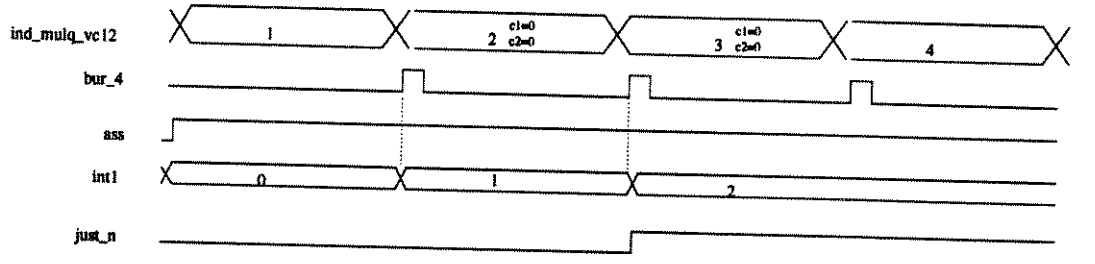


Figura 4.38 : Diagrama temporal do *Interpretador de C1 C2*

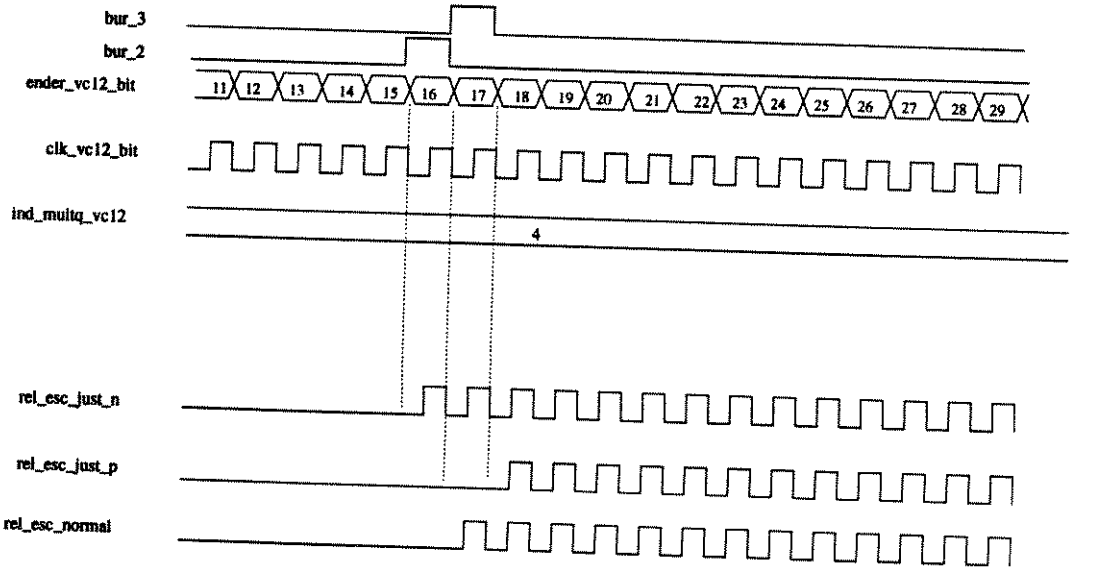


Figura 4.39 : Diagrama temporal do *rel_esc*

Bloco *Selecionador_dos_rel_escrita*

- Este bloco é modelado pelo processo abaixo.

```
Seleciona_rel_esc:
   rel_esc_bit<=rel_esc_just_n WHEN just_n='1' ELSE
               rel_esc_just_p WHEN just_p='1' ELSE
               rel_esc_normal;
```

O *rel_esc_bit* fornece o sinal para escrita na memória elástica conforme a ocorrência ou não de justificação. A figura 4.40 mostra como se comporta este bloco.

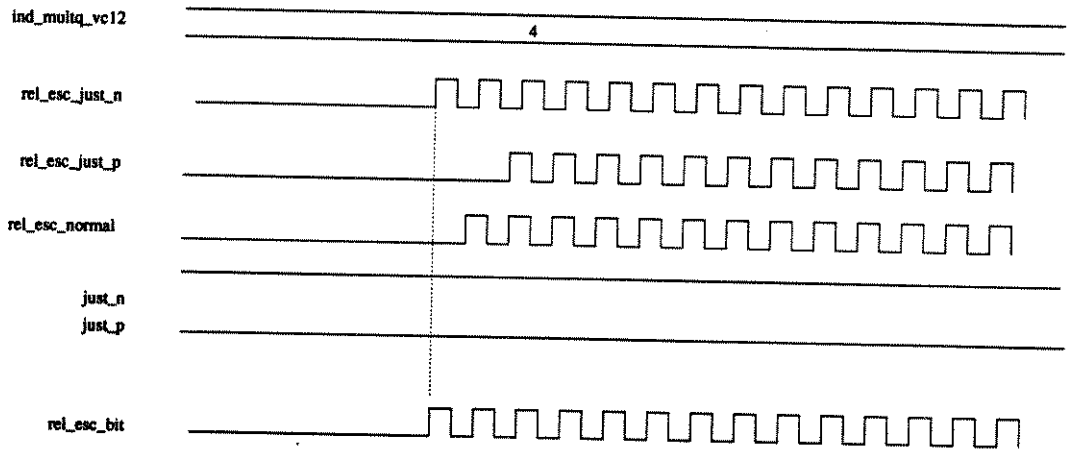


Figura 4.40 : Diagrama temporal do *rel_esc*

Bloco *Gerador_ender_escrita*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_ender_esc:
PROCESS(rel_esc_bit)
BEGIN
   IF(rel_esc_bit='0')AND(rel_esc_bit'EVENT)THEN
      IF(ender_esc_bit=63)THEN
         ender_esc_bit<=0;
      ELSE
         ender_esc_bit<=ender_esc_bit+1;
      END IF;
   END IF;
END PROCESS;
```

Fornece o endereço para escrita na memória elástica na descida do sinal *rel_esc_bit.*

Bloco *Gerador_ender_leitura*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_ender_leit:
PROCESS(rel_leit_bit)
BEGIN
   IF(rel_leit_bit='0')AND(rel_leit_bit'EVENT)THEN
      IF(ender_leit_bit=63)THEN
         ender_leit_bit<=0;
      ELSE
         ender_leit_bit<=ender_leit_bit+1;
      END IF;
   END IF;
END PROCESS;
```

A cada descida do *rel_leit_bit* o endereço de leitura é incrementado até chegar a 63, quando então é zerado.

Bloco *Gerador_da_mem_elástica*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_mem_elástica:
PROCESS(rel_esc_bit)
BEGIN
   IF(rel_esc_bit='1')AND(rel_esc_bit'EVENT)THEN
      mem_elast_bit(ender_esc_bit)<= VC12_bit;
   END IF;
END PROCESS;
PROCESS(rel_leit_bit)
BEGIN
   IF(rel_leit_bit='1')AND(rel_leit_bit'EVENT)THEN
      sinal<=mem_elast_bit(ender_leit_bit);
   END IF;
END PROCESS;
```

Nesta memória elástica é que os dados são escritos. Na subida do *rel_esc_bit* o dado entra num endereço de memória. Na subida do *rel_leit_bit* o dado é lido de um endereço de leitura.

Bloco *Gerador_do_fasímetro*

- Este bloco é modelado pelo processo abaixo.

```
Gerador_fase_bit:
PROCESS(rel_esc_bit)
variable v_fase_bit:INTEGER:=0;
BEGIN
   v_fase_bit:=ender_esc_bit-ender_leit_bit;
   IF(v_fase_bit<0)THEN
      fase_bit<=64+v_fase_bit;
   ELSE
      fase_bit<=v_fase_bit;
   END IF;
END PROCESS;
```

Mede a distância entre o endereço de escrita e o endereço de leitura reportando este valor para o PLL.

Capítulo 5

Simulações e Análises

5.1 Introdução

Os blocos funcionais descritos anteriormente não estão completos, pois precisa-se fazer a declaração das "Entidades". Na linguagem VHDL , utiliza-se esta declaração para descrever as entradas e as saídas de cada bloco funcional. Vai-se, logo a seguir, descrever duas redes onde, através da união destes blocos funcionais, temos os equipamentos que as formam. Para isto foi utilizada a linguagem VHDL a nível estrutural[5]. Nestes equipamentos serão feitas as simulações e análises visando um maior entendimento da hierarquia estudada.

5.2 Retirada de Canais Utilizando Mapeamento Síncrono de byte em 2048 kb/s

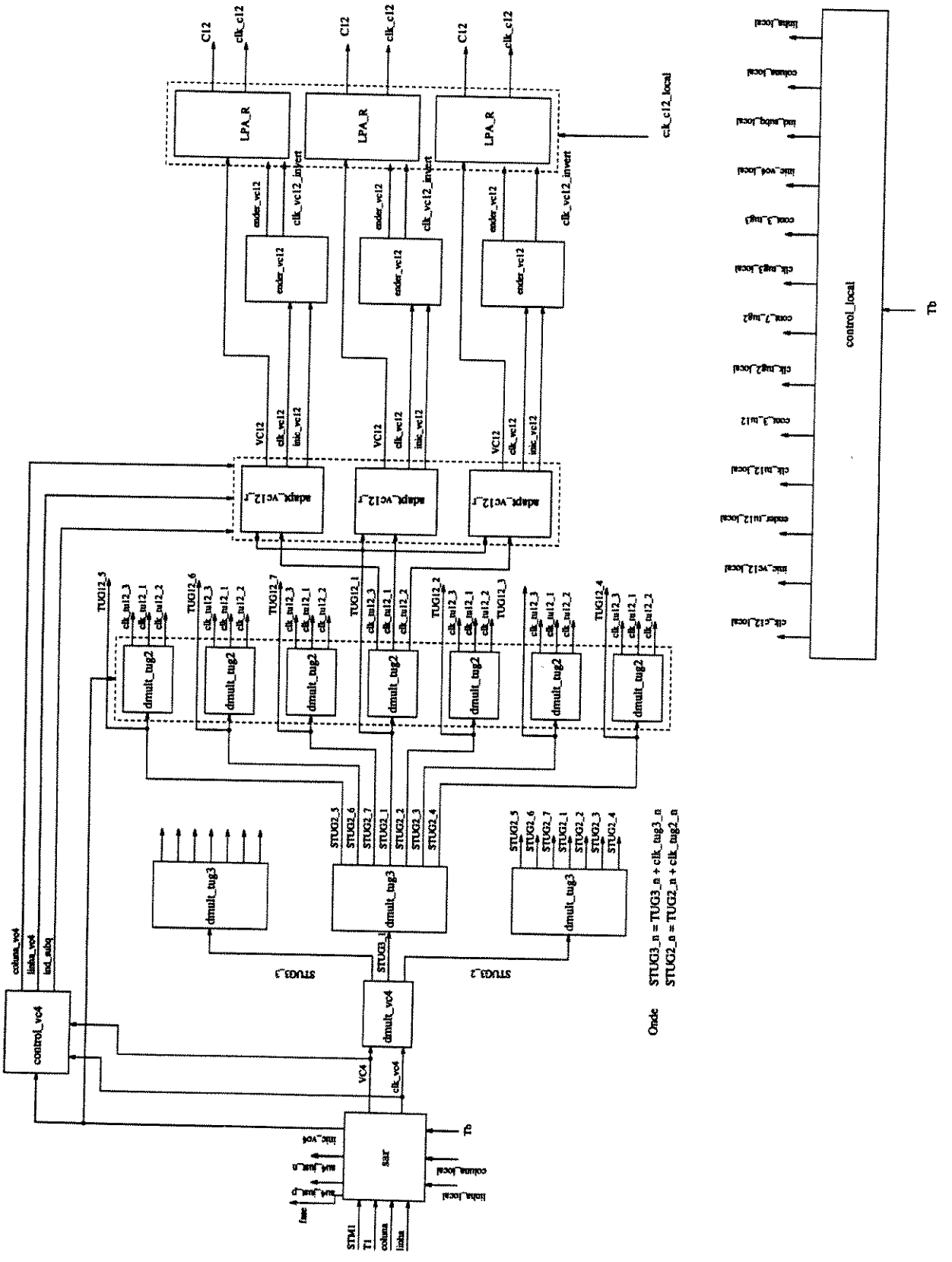
A fim de testar os blocos desenvolvidos anteriormente, vai-se mostrar uma rede síncrona de transporte de informação em um enlace ponto a ponto entre dois equipamentos terminal multiplex SDH de 155520 kb/s. Os sinais entrão em uma ponta do equipamento de transmissão já na forma de Virtual Containers de pequena ordem (VC's12) são multiplexados para formar um STM-1, trafegam por uma rede simulada e são recolhidos do outro lado, no equipamento de recepção. Este equipamento é também um terminal multiplex de 155520 kb/s, o sinal é demultiplexado para gerar os Virtuais Containers de pequena ordem e em seguida desmapeado para formar quadros de 32 canais síncronos com 2048 kb/s (C12), utilizando-se o TS0 como canal ZERO, vai-se verificar a retirada correta dos 32

canais na recepção. Um outro fenômeno a ser observado, é do SLIP, quando se tem um equipamento SDH cujos tributários são síncronos, e por algum defeito na referência de sincronismo os equipamentos começam a trabalhar através do relógio interno, ou seja, com 4.6 ppm, este fenômeno pode ocorrer e é este efeito que também tentaremos mostrar nesta simulação.

5.2.1 -Equipamento de Desmapeamento Síncrono

Este equipamento recebe um sínal STM1 de 155520 kb/s, realiza as operações de adaptação, demultiplexação e desmapeamento síncrono de byte . Na sua saída tem-se 63 sinais , cada um com 32 canais síncronos de 2048 kb/s. No capítulo 4, o item 4.2 tratou da recepção síncrona, os blocos desenvovidos fazem a retirada somente da carga ùtil de um VC12 retirando também os bytes de enchimento. Os blocos lpa_r e gerador de endereço de VC12 são os responsáveis pela retirada destes 32 canais A figura 5.1 mostra este equipamento e logo após tem-se a descrição das suas entidades. As entidades sar,dmult_vc4,control_vc4, dmult_tug3, dmult_tug2, adapt_vc12_r e control_local foram desenvolvidas anteriormente[2].

Figura 5.1 :*Desmapeamento Síncrono*



ENTIDADES:

Entidade *Sar*(**Seção de adaptação na recepção**)

Declaração de Entidade em VHDL:

```
ENTITY sar IS
    PORT(STM1: IN BIT_VECTOR(1 TO 8);
        linha, coluna, linha_local, coluna_local: IN INTEGER:=0;
        T0, T1: IN BIT;
        just_p, just_n, inic_vc4, clk_vc4: BUFFER BIT;
        fase: BUFFER INTEGER :=0;
        VC4: OUT BIT_VECTOR(1 TO 8));
END sar;
```

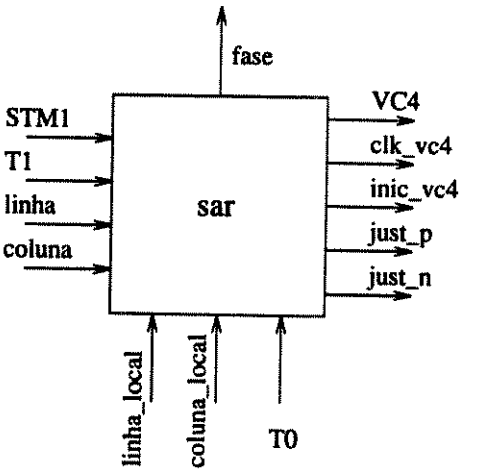


Figura 5.2 : Ilustração do Componente *Sar*.

Entidade *control_vc4*

Declaração de Entidade em VHDL:

```
ENTITY control_vc4 IS
    PORT(VC4: IN BIT_VECTOR(1 TO 8);
        inic_vc4, clk_vc4: IN BIT;
        linha_vc4, coluna_vc4: BUFFER INTEGER;
        ind_subq: OUT INTEGER);
END control_vc4;
```

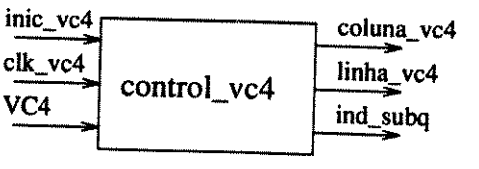


Figura 5.3 : Ilustração do Conponente *Control_vc4*.

Entidade *dmult_vc4* (Demultiplexador de VC4)

Declaração de Entidade em VHDL:

```
ENTITY dmult_vc4 IS
    PORT(VC4: IN BIT_VECTOR(1 TO 8);
       clk_vc4, inic_vc4: IN BIT;
       coluna_vc4: IN INTEGER;
       TUG3_1, TUG3_2, TUG3_3: OUT BIT_VECTOR(1 TO 8);
clk_tug3_1, clk_tug3_2, clk_tug3_3: BUFFER BIT);
END dmult_vc4;
```

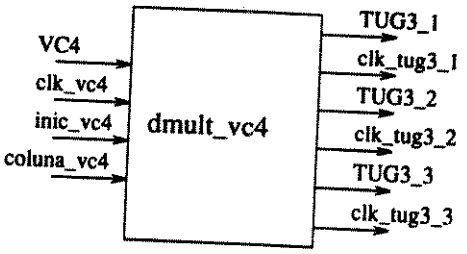


Figura 5.4 : Ilustração do Componente *dmult_vc4*

Entidade *dmult_tug3* (Demultiplexador de TUG-3)

Declaração de Entidade em VHDL:

```
ENTITY dmult_tug3 IS
    PORT(TUG3: IN BIT_VECTOR(1 TO 8);
       coluna_vc4: IN INTEGER;
       clk_tug3: IN BIT;
       TUG2_1, TUG2_2, TUG2_3, TUG2_4, TUG2_5, TUG2_6,
       TUG2_7: OUT BIT_VECTOR(1 TO 8);
       clk_tug2_1, clk_tug2_2, clk_tug2_3, clk_tug2_4,
       clk_tug2_5, clk_tug2_6, clk_tug2_7: BUFFER BIT);
END dmult_tug3;
```

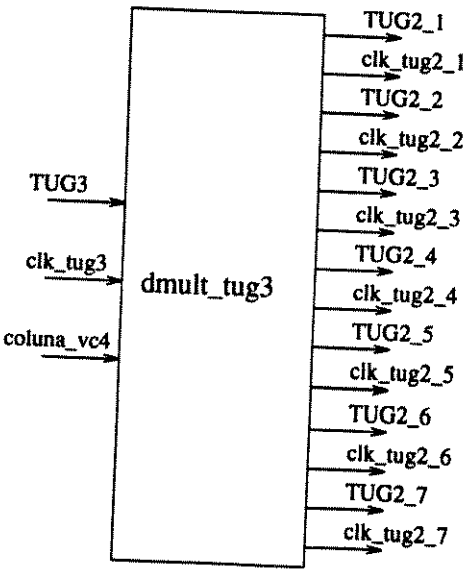


Figura 5.5 : Ilustração do Componente *dmult_tug3*

Entidade *dmult_tug2* (Demultiplexador de TUG-2)

Declaração de Entidade em VHDL:

```
ENTITY dmult_tug2 IS
    PORT(clk_tug2, inic_vc4: IN BIT;
      clk_tu12_1, clk_tu12_2, clk_tu12_3: OUT BIT);
END dmult_tug2;
```

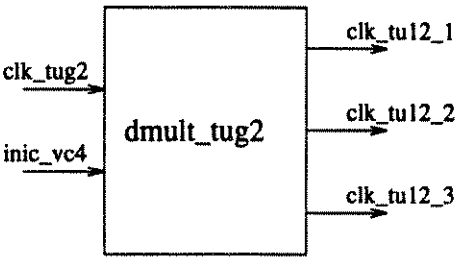


Figura 5.6 : Ilustração do componente *dmult_tug2*

Entidade *adapt_vc12_r* (Adaptador de VC12 na Recepção.)

Declaração de Entidade em VHDL:

```
ENTITY adapt_vc12_r IS
   PORT(TUG2: IN BIT_VECTOR(1 TO 8);
      ind_subq, ind_subq_local, ender_tu12_local, coluna_vc4, linha_vc4: IN INTEGER:=0;
      clk_tu12, clk_tu12_local: IN BIT;
      just_p, just_n, inic_vc12, clk_vc12: BUFFER BIT;
      fase: BUFFER INTEGER:=0;
      VC12: OUT BIT_VECTOR(1 TO 8));
END adapt_vc12_r;
```

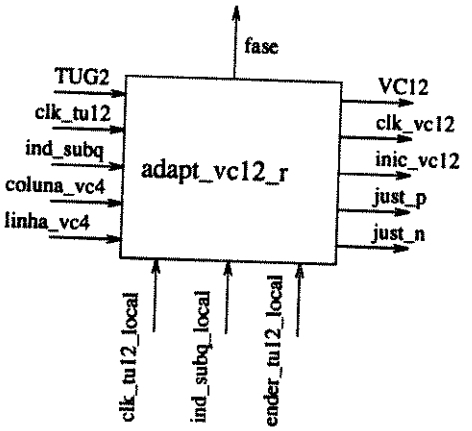


Figura 5.7 : Ilustração do Componente *adapt_vc12_r*

Entidade *ender_vc12* (Endereço de VC12)

Declaração de Entidade em VHDL:

```
ENTITY ender_vc12 IS
   PORT(inic_vc12: IN BIT;
      clk_vc12:IN BIT;
      ender_vc12:BUFFER BIT;
      clk_vc12_inv: OUT BIT);
END ender_vc12;
```

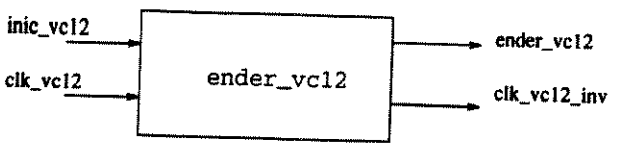


Figura 5.8 : Ilustração do Componente *ender_vc12*

Entidade *lpa_r* (Recepção de VC12)

Declaração de Entidade em VHDL:

```
ENTITY lpa_r IS
   PORT(VC12: IN BIT_VECTOR(1 TO 8);
1cmender_vc12:IN INTEGER;
      clk_vc12,clk_c12_local:IN BIT;
      sinal: OUT BIT_VECTOR(1 TO 8)
      clk_sinal:OUT BIT);
END lpa_r;
```

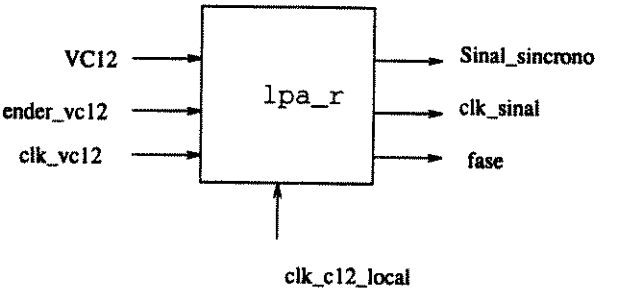
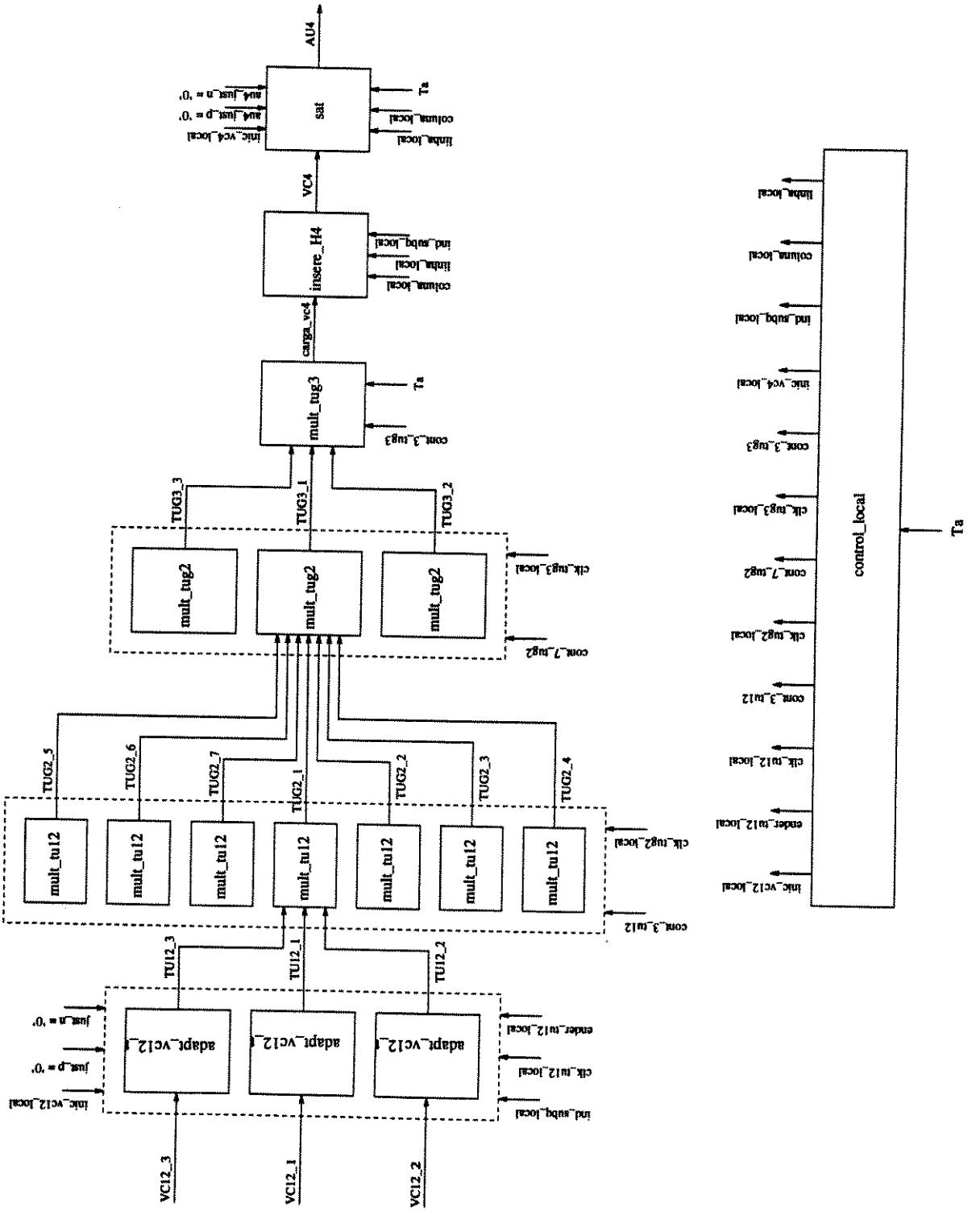


Figura 5.9 : Ilustração do Componente *lpa_r*

Equipamento de Transmissão de VC12

Na direção de transmissão tem-se a estrutura mostrada na figura 5.10, o equipamento recebe 63 sinais afluentes VC12 e executa as operações de adaptação e multiplexação, resultando na sua saída o sinal AU-4. Como não estamos tratando de gerênciamento, o sinal AU-4 que não possui SOH é entendido como um STM-1 apesar de não possuir os bytes de cabeçalho . Todas as entidades aqui mostradas já foram desenvolvidas anteriormente[2].

Figura 5.10 : *Equipamento de Transmissão de VC12.*



ENTIDADES

Entidade *adapt_vc12_t* **(Adaptador de VC12 na Transmissão.)**

Declaração de Entidade em VHDL:

```
ENTITY adapt_vc12_t IS
   PORT(VC12: IN BIT_VECTOR(1 TO 8);
      ind_subq_local, ender_tu12_local: IN INTEGER:=0;
      clk_tu12_local, inic_vc12, just_p, just_n: IN BIT;
      TU12: OUT BIT_VECTOR(1 TO 8));
END adapt_vc12_t;
```

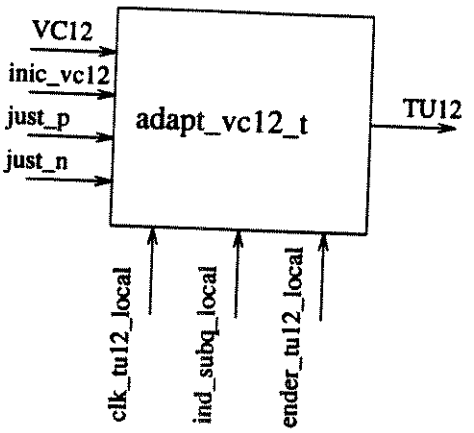


Figura 5.11 : Ilustração do Componente *adapt_vc12_t*.

Entidade *mult_tu12* **(Multiplexador de TU-12.)**

Declaração de Entidade em VHDL:

```
ENTITY mult_tu12 IS
   PORT(TU12_1, TU12_2, TU12_3: IN BIT_VECTOR(1 TO 8);
      cont_3_tu12: IN INTEGER;
      clk_tug2_local: IN BIT;
      TUG2: OUT BIT_VECTOR(1 TO 8));
END mult_tu12;
```

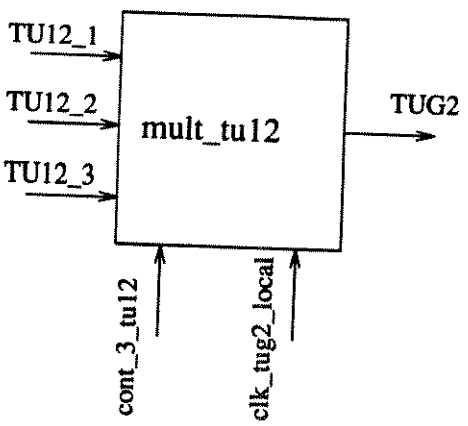


Figura 5.12 : Ilustração do Componente *mult_tu12*

Entidade *mult_tug2* **(Multiplexador de TUG-2.)**

Declaração de Entidade em VHDL:

```
ENTITY mult_tug2 IS
   PORT(TUG2_1, TUG2_2, TUG2_3, TUG2_4, TUG2_5, TUG2_6, TUG2_7:
            IN BIT_VECTOR(1 TO 8);
      cont_7_tug2: IN INTEGER;
      clk_tug3_local: IN BIT;
      TUG3: OUT BIT_VECTOR(1 TO 8));
END mult_tug2;
```

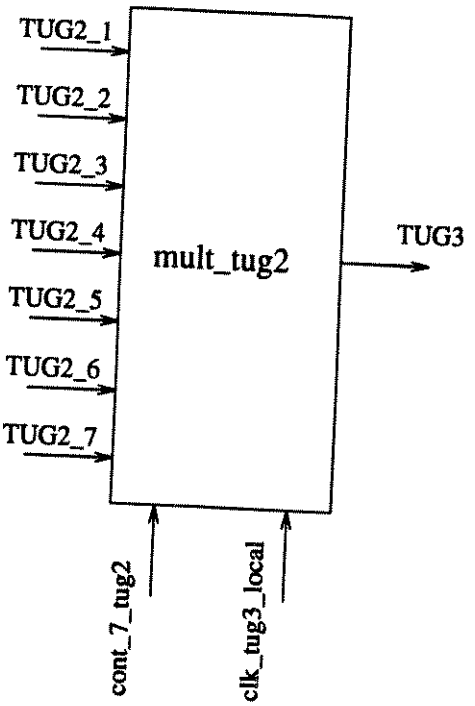


Figura 5.13 : Ilustração do Componente *mult_tug2*

Entidade *mult_tug3* **(Multiplexador de TUG-3.)**

Declaração de Entidade em VHDL:

```
ENTITY mult_tug3 IS
   PORT(TUG3_1, TUG3_2, TUG3_3: IN BIT_VECTOR(1 TO 8);
      cont_3_tug3: IN INTEGER;
      T0: IN BIT;
      carga_vc4: OUT BIT_VECTOR(1 TO 8));
END mult_tug3;
```

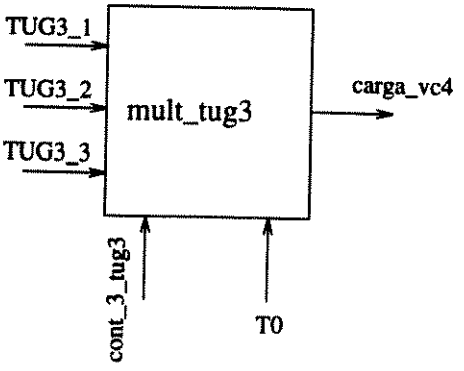


Figura 5.14 : Ilustração do Componente *mult_tug3*

Entidade *insere_H4* (Gerador e Inseridor de Byte H4.)

Declaração de Entidade em VHDL:

```
ENTITY insere_h4 IS
   PORT(carga_vc4: IN BIT_VECTOR(1 TO 8);
      linha_local, coluna_local, ind_subq_local: IN INTEGER:=0;
      T0: IN BIT;
      VC4: OUT BIT_VECTOR(1 TO 8));
END insere_h4;
```

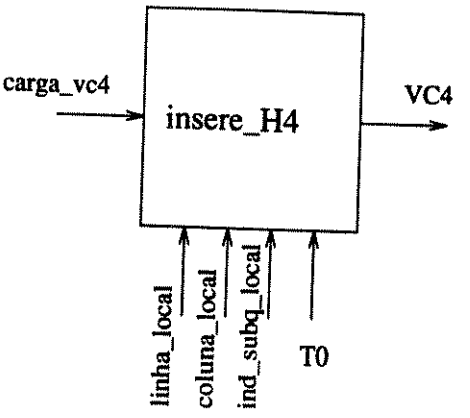


Figura 5.15 : Ilustração do Componente *insere_h4*

Entidade *sat* (Adaptação de Seção Multiplex na Transmissão.)

Declaração de Entidade em VHDL:

```
ENTITY sat IS
   PORT(VC4: IN BIT_VECTOR(1 TO 8);
      linha_local, coluna_local: IN INTEGER:=0;
      T0, inic_vc4, just_p, just_n: IN BIT;
      AU4: OUT BIT_VECTOR(1 TO 8));
END sat;
```

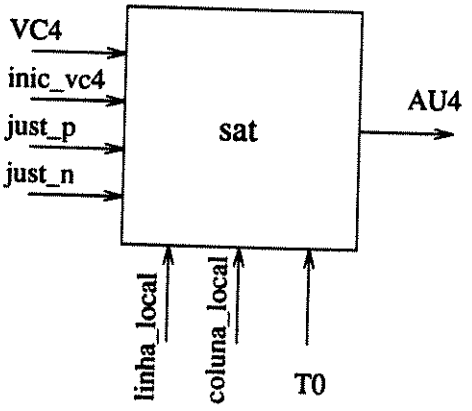


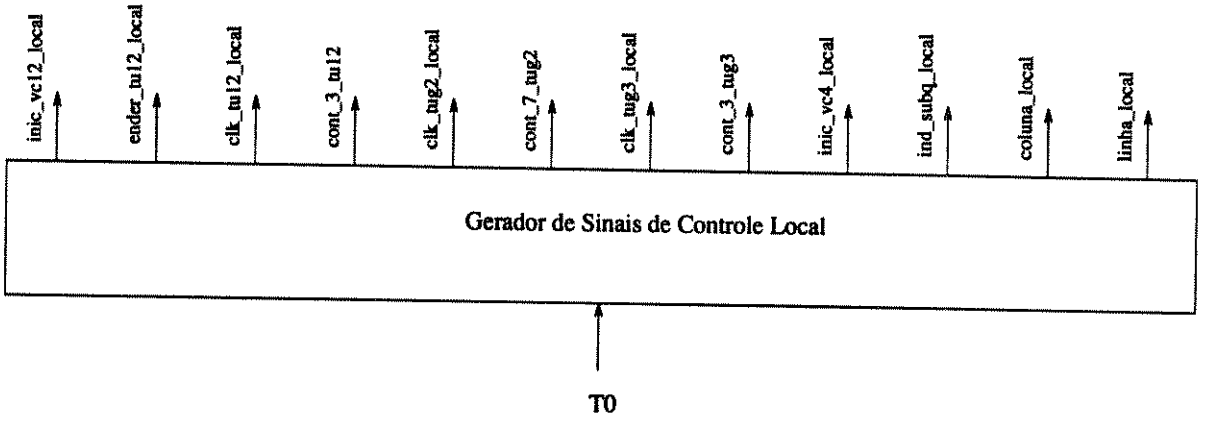
Figura 5.16 : Ilustração do componente *Sat*

Entidade *control_local* (Gerador de Sinais de Controle Local.)

Declaração de Entidade em VHDL:

```
ENTITY control_local IS
   PORT(T0: IN BIT;
      linha_local, coluna_local, ind_subq_local, cont_3_tug3,
      cont_7_tug2, cont_3_tu12, ender_tu12_local: BUFFER INTEGER:=0;
      inic_vc4_local, clk_tug3_local, clk_tug2_local, clk_tu12_local, inic_vc12_local: BUFFER BIT);
END control_local;
```

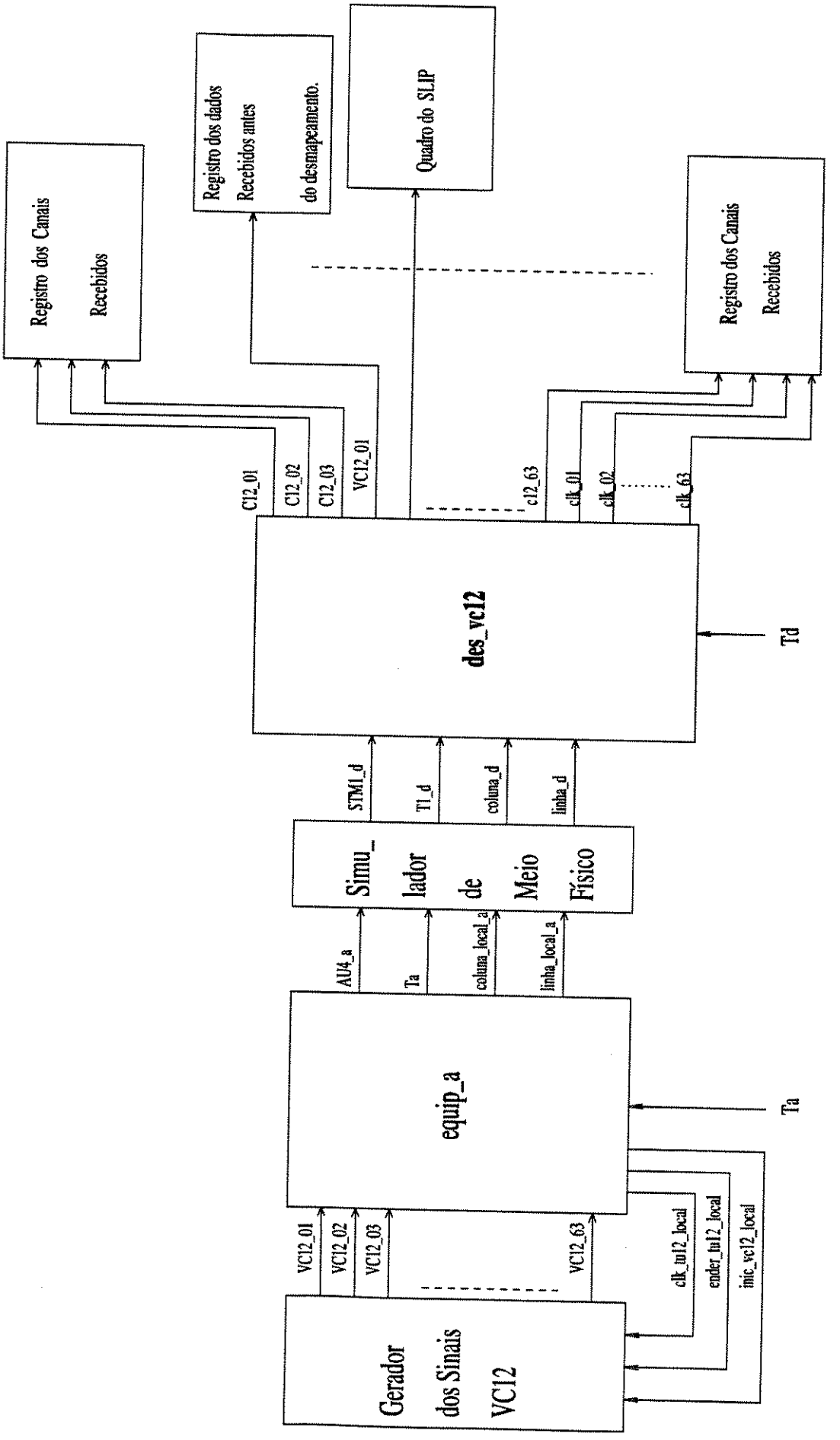
Figura 5.17 : Ilustração do Componente *Control_local*



5.2.2 Realização do Enlace#1:

Fazendo-se uma conexão do *trasm_vc12* com o *des_vc12*, tem-se a ligação entre o equipamento de transmissão e Recepção numa topologia ponto a ponto. São gerados 63 sinais VC12 com dados hexadecimais que entram no equipamento terminal multiplex de transmissão saindo na forma de STM-1. Entre o *transm_vc12* e o equipamento *des_vc12* tem-se um simulador de meio físico que provoca mudanças de fase no agregado STM-1. Na recepção, o equipamento de desmapeamento faz a retirada dos 63 tributários. Os dados são armazenados em arquivos para uma posterior análise. A figura 5.18 mostra o diagrama em blocos do enlace#1. Foram usadas duas referências de relógio nas simulações; uma referência que se baseia em um sinal de relógio externo alimentando os dois pontos do enlace, simulando a sincronização entre os equipamentos, e uma outra referência de relógio interna para verificar o SLIP; nesta referência teremos o equipamento remoto recebendo um relógio externo e o equipamento local operando com relógio interno no modo free-running.

Figura 5.18 : *Enlace#1*



Gerador de Sinais VC12

Este bloco gera uma sequência de bytes na ponta do enlace que devem ser recebidas do outro lado sem alterações. Estes bytes já estão na forma de Virtuais Containers de pequena ordem com seus 35 bytes por quadro STM-1. Estes virtuais containers estão todos equipados, ou seja, transportam carga ùtil em suas estruturas bem como cabeçalho e enchimento. Os dados de saída deste gerador são da seguinte forma:

- VC12_01=01,02,03,04,05.....63;
- VC12_02=02,03,04,05.....63,01;
- VC12_03=03,04,05.....63,01,02;
- VC12_04=04,05,.....63,01,02,03;
- VC12_05=05,.....63,01,02,03,04

Isto se segue para os 63 tributários.

Modelo em VHDL:

```
ger_sinais_vc12:PROCESS(clk_tu12_local_a,inic_vc12_local_a)
BEGIN
   IF(clk_tu12_local_a = '1') AND (clk_tu12_local_a'EVENT)AND(ender_tu12_local_a ≠0)THEN
    FOR i IN 62 DOWNTO 1 LOOP
      VC12(i) <= VC12(i +1) AFTER 2ns:
    END LOOP
   VC12(63) <= VC12(1) AFTER 2ns;
   END IF;
   IF(inic_vc12_local_a = '1')AND(inic_vc12_local_a'EVENT)THEN
      FOR i IN 1 TO 63 LOOP
         VC12(i)<=inteiro_pra_byte(i);
      END LOOP;
   END IF;
END PROCESS;
```

Simulador de Meio Físico

Este bloco provoca apenas uma inversão de fase no clk do sinal recebido considerando que não ocorreu wander ou jitter.

Modelo em VHDL:

```
ger_coluna_linha_b:PROCESS(Ta)
BEGIN
   IF (Ta = '1')THEN
      coluna_b <= coluna_local_a;
      linha_b <= linha_local_a;
   END IF;
END PROCESS;
T1_b <= NOT(Ta);
STM1_b <= AU4_a;
```

Relógio

Este sinal é gerado a fim de se registrar certos fenômenos que ocorrem durante as simulações.

Modelo em VHDL:

```
relógio:PROCESS
BEGIN
    WAIT FOR 10 ns;
       tempo <= tempo+1.0e-9;
END PROCESS;
```

5.2.3 Simulações no Enlace#1

O ITU-T recomenda limite máximo de 4.6ppm de variação de relógio quando operando em **free-running**. Foi feita uma simulação onde utilizou-se um relógio com variação de 46 ppmil, para se obter mais rapidamente o resultado nas simulações.

Relógios Ta, Td e clk_c12_local_teste:

```
Ta <= NOT(Ta) AFTER 24.589067504 ns;

Td <= NOT(Td) AFTER 24.589067504 ns;

clk_c12_local_teste <= NOT(clk_c12_local_teste) AFTER 1953.125 ns;
```

Cálculos:

- Ta× 46 ppmil = 7.153.920 b/s;
- taxa = 7.153.920 + 155.520 kb/s = 162.673.920 b/s;

taxa =2.048.000× 46 ppmil + 2.048.000kb/s = 94.208 b/s + 2.048.000 kb/s = 2.142.208 b/s

5.2.4 O Fenômeno do SLIP

Nos cálculos acima, o relógio de escrita terá um valor de 2.142 kb/s e o de leitura com 2048 kb/s; com isto observa-se que o relógio de escrita na memória elástica de desmapeamento vai ser bem mais rápido que o de leitura, ou seja, a memória elástica irá em direção a estourar rápidamente. Como nesta simulação os sinais dos tributários são considerados síncronos, não necessitando de grande memória elástica, é certo que com a perda de uma referência externa a memória irá transbordar, ocorrendo o SLIP. Nesta nossa simulação utilizou-se uma memória elástica grande para que o fenômeno do slip pudesse ser observado.

Seguindo as idéias do item 4.2.10 estabeleceu-se um limiar para que o SLIP fosse do tipo controlado, através de uma adaptação na memória elástica de desmapeamento. Aqui fixou-se um limiar superior de 59 bytes e um inferior de 5 bytes, com a fase da memória elástica partindo de 32(diferença entre o endereço de escrita e o de leitura) , maiores esclarecimentos podem ser obtidos no capítulo 4.

O tempo para que ocorra o 1º SLIP é de:

- (32-5)×8 = 216 bits ver figura 4.11.

- Tempo = 216/94.208 = 2.3 ms em média;

ou seja:

Nº quadros = 2,3 ms / 125 μs = 18 quadros.

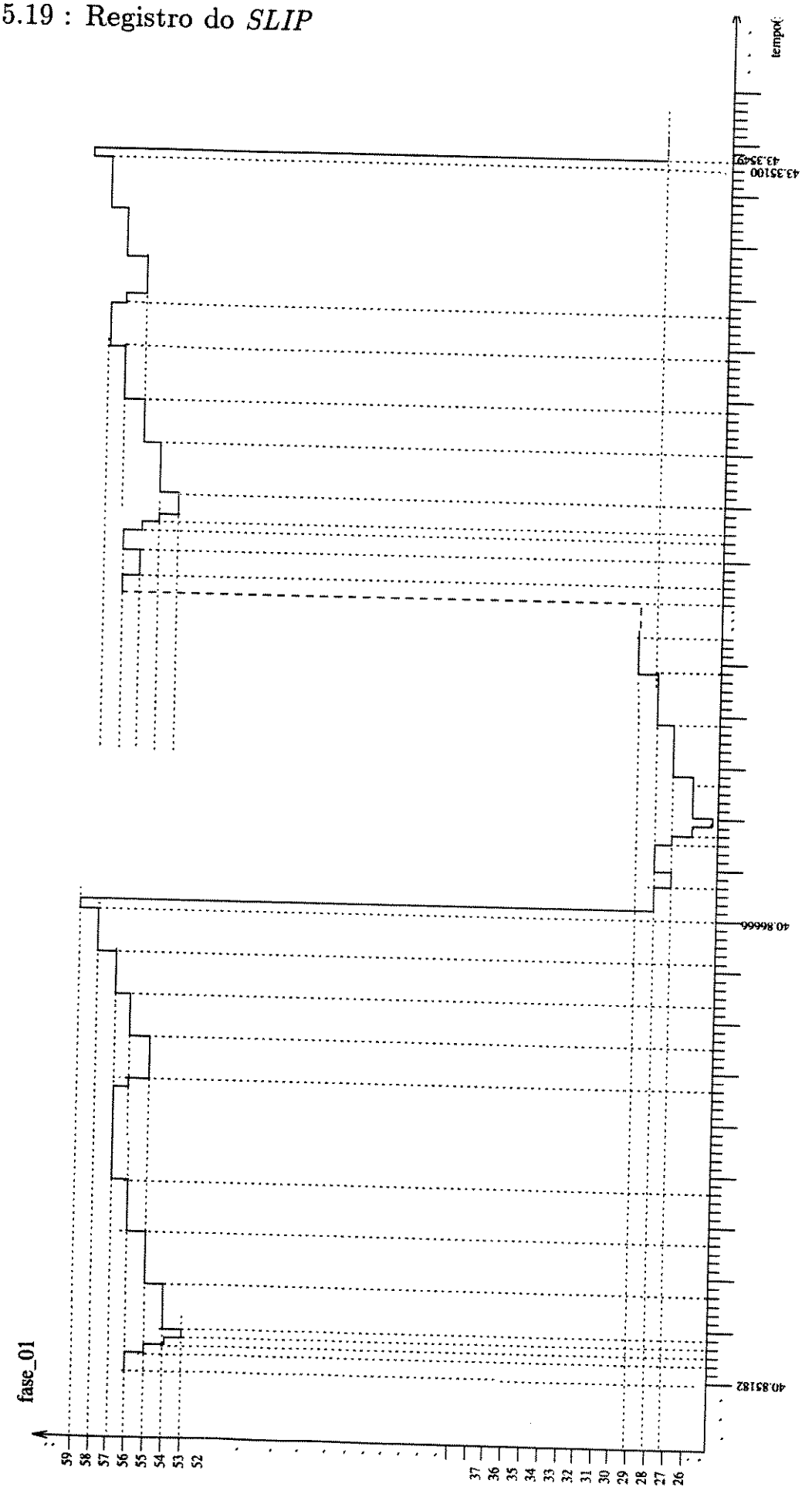
O Registro de SLIP:

Pode-se observar na figura 5.19 que quando a fase está em 59, ou seja o endereço de escrita está proximo do de leitura , ocorre um salto para a frente de 32 bytes no endereço de leitura, fazendo com que a fase passe para 28. Observa-se também que a fase começa a aumentar; isto já era esperado pois fizemos o relógio na origem ser mais rápido que o relógio local. Conforme os cálculos anteriores, SLIPS devem ocorrer dentro de intervalos de **aproximadamente** 2.3 ms. (**Aproximadamente** devido às muitas variações que ocorrem na entrada dos dados na memória elástica). Este fenômeno realmente ocorre quando os tributários são síncronos e os relógios estão em modo livre, como foi observado.

Na observação do gráfico da figura 5.19 verifica-se o tempo entre dois SLIPS:

(43.35491ms - 40.86666ms = 2.48 ms)

Figura 5.19 : Registro do *SLIP*



5.2.5 Retirada dos Canais Síncronos de Byte Com Relógio Local

Uma análise gráfica será feita, logo a seguir, para verificar a retirada simples de 32 canais em 125μs.

A figura 5.20 mostra, logo abaixo, a retirada completa dos 32 canais.

A figura 5.21 mostra o comportamento da fase.

Observações Gráficas:

Na figura 5.20:

Observa-se que entre os dois primeiros tempos (12.33577ms-12.21078ms = 124.99μs) tem-se a retirada de 32 canais síncronos começando com o dado 06 indo até o dado 37 perfazendo os 32 canais.

Após a retirada do último dado deste C12 (37) tem-se o desconto dos bytes de cabeçalho e enchimento que correspondem aos valores 38,39,40. O próximo quadro começa com o dado 41 indo até o dado 09 com os tempos de (12.46077ms-12.33577ms=125μs) perfazendo também os 32 canais.

Se observarmos o gráfico da figura 5.19 verificamos que a cada retirada de 32 canais corresponde o tempo de 125μs e retirada completa dos bytes de cabeçalho e enchimento. Observe que o comportamento de retirada dos canais se repete dentro de um intervalo de 500μs, ou seja, a cada multiquadro.

Na figura 5.21:

Uma observação importante a ser feita, neste caso, é que a fase tende a subir em direção a "estourar" a memória elástica, isto era esperado já que o relógio de escrita foi gerado de forma que fosse bem mais rápido que o de leitura.

Pode-se entender os instantes em que o relógio de escrita dá uma parada, devido aos bytes de cabeçalho e enchimento que não são escritos na memória elástica quando a fase da memória elástica começa a diminuir, isto é mostrado no tempo 12.30452 ms quando a fase cai de 36 para 35. O relógio de escrita na memória elástica possui buracos e o relógio de leitura é constante.

Figura 5.20 : *Gráfico da Retirada da Carga Útil.*

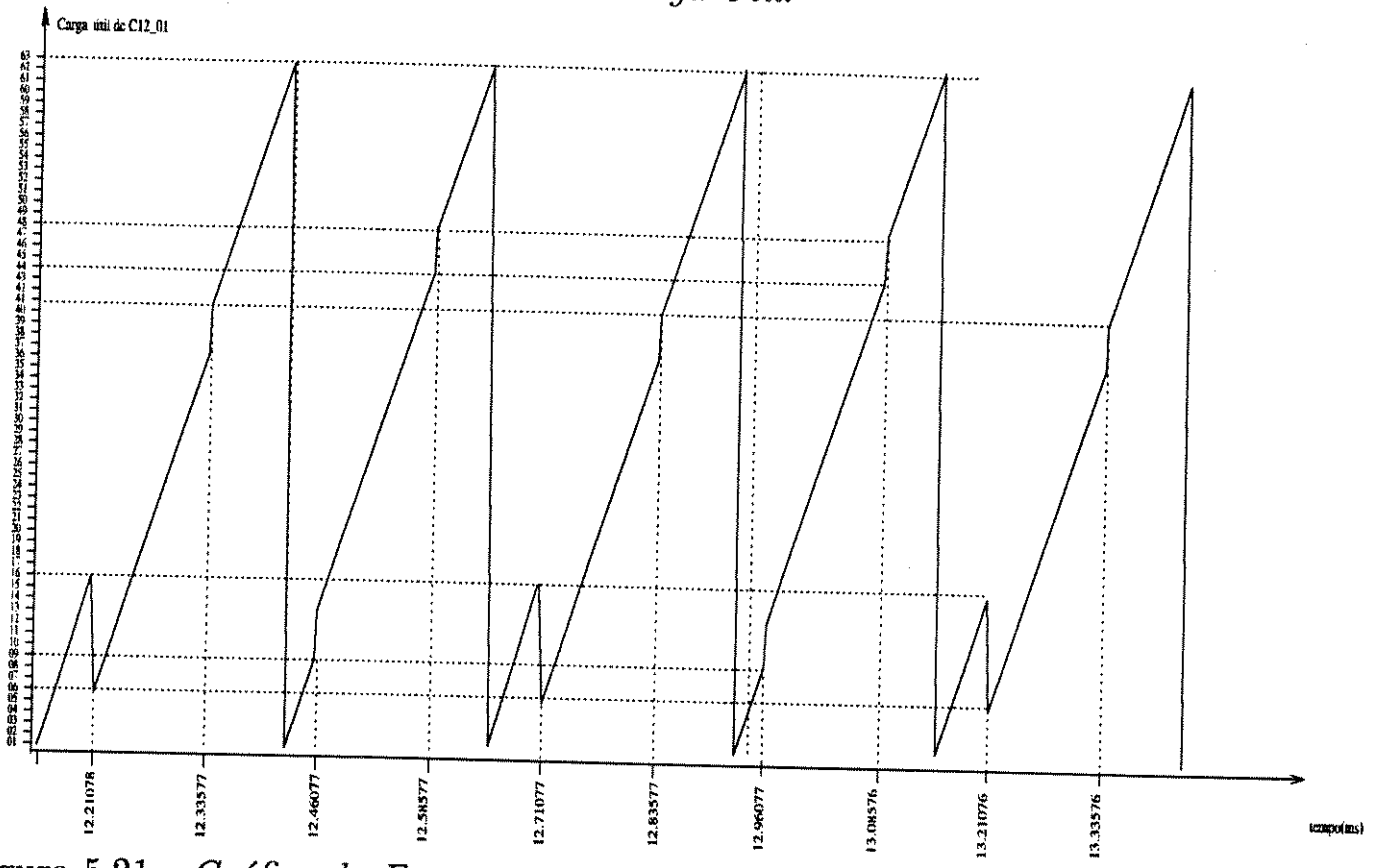
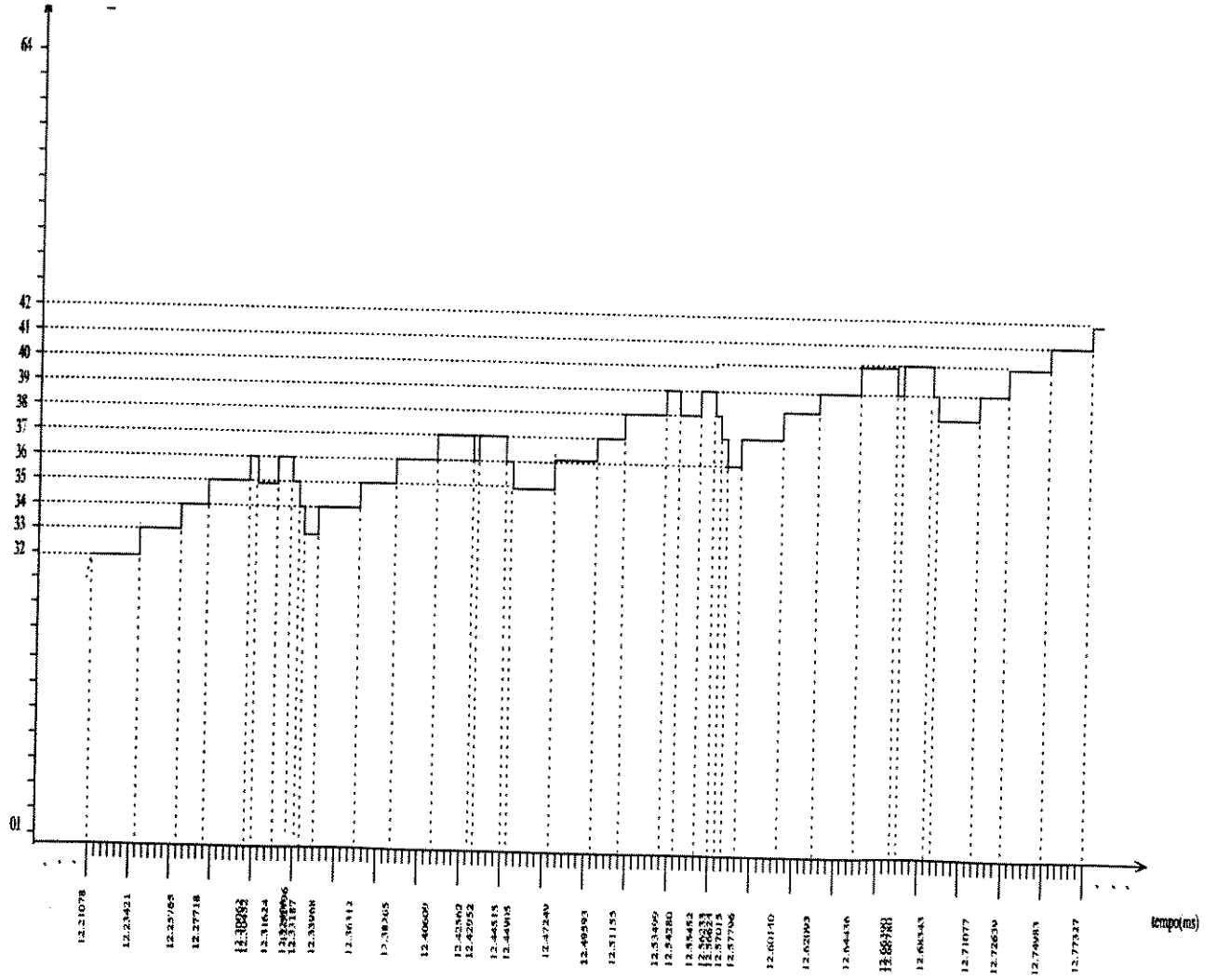


Figura 5.21 : *Gráfico da Fase.*



Observação:

- Fazendo-se alguns cálculos podemos verificar alguns detalhes referentes ao gráfico da fase.

O relógio de escrita chega com uma frequência de +94208 Hz na memória elástica de desmapeamento.

94.208 b/s = 11,776 bits a cada 125μs.

Em 500μs teríamos 47,104 bits sendo colocados a mais na memória elática. Isto corresponde a um aumento na fase de um valor de 6. Se observarmos no gráfico da fase entre o tempo de 12.21078ms e 12.71077ms verificamos que a fase aumenta de 32 para 38, tendo um aumento de 6 como foi calculado a pouco.

5.2.6 Retirada de Dados Com Referência Externa de Relógio

Uma outra análise a ser feita, é a de comparar dados que entram em um lado da memória elástica de desmapeamento e os dados que saem desta memória. Esta simulação será com os relogios dos equipamentos trabalhando nos valores nominais, ou seja, sem variações entre o relógio remoto e o relógio local, usando uma fonte externa para sincronizar os equipamentos, sendo que esta fonte é a mesma para os dois equipamentos. Os equipamentos estão trabalhando como se um relógio primário de referência fosse usado diretamente nos equipamentos da rede.

Relógios Ta, Td e clk_c12_local_teste:

```
Ta <= NOT(Ta) AFTER 25.720164609 ns;

Td <= NOT(Td) AFTER 25.720164609 ns;

clk_c12_local_teste <= NOT(clk_c12_local_teste) AFTER 1953.125 ns;
```

Quando ocorre a escrita de um dado em uma memória elástica,.o dado irá assumir uma posição determinada pelo endereço de escrita, o mesmo ocorre quando se faz a leitura deste dado, ou seja, vai-se ler este dado em uma posição determinada pelo endereço de leitura que depende da fase desta memória elástica.Para saber se os dados que saem da memória elástica estão corretos , deve-se registrar os dados de entrada, a fase desta memória e os dados de saída. Se para um determinado dado

de entrada a fase tem o valor 32, este mesmo dado deverá sair em aproximadamente 32 bytes depois ou 125 μs, se a fase estiver com o valor 34 o dado deverá sair em aproximadamente 34 bytes depois ou 132 μs .Daí deduz-se que:

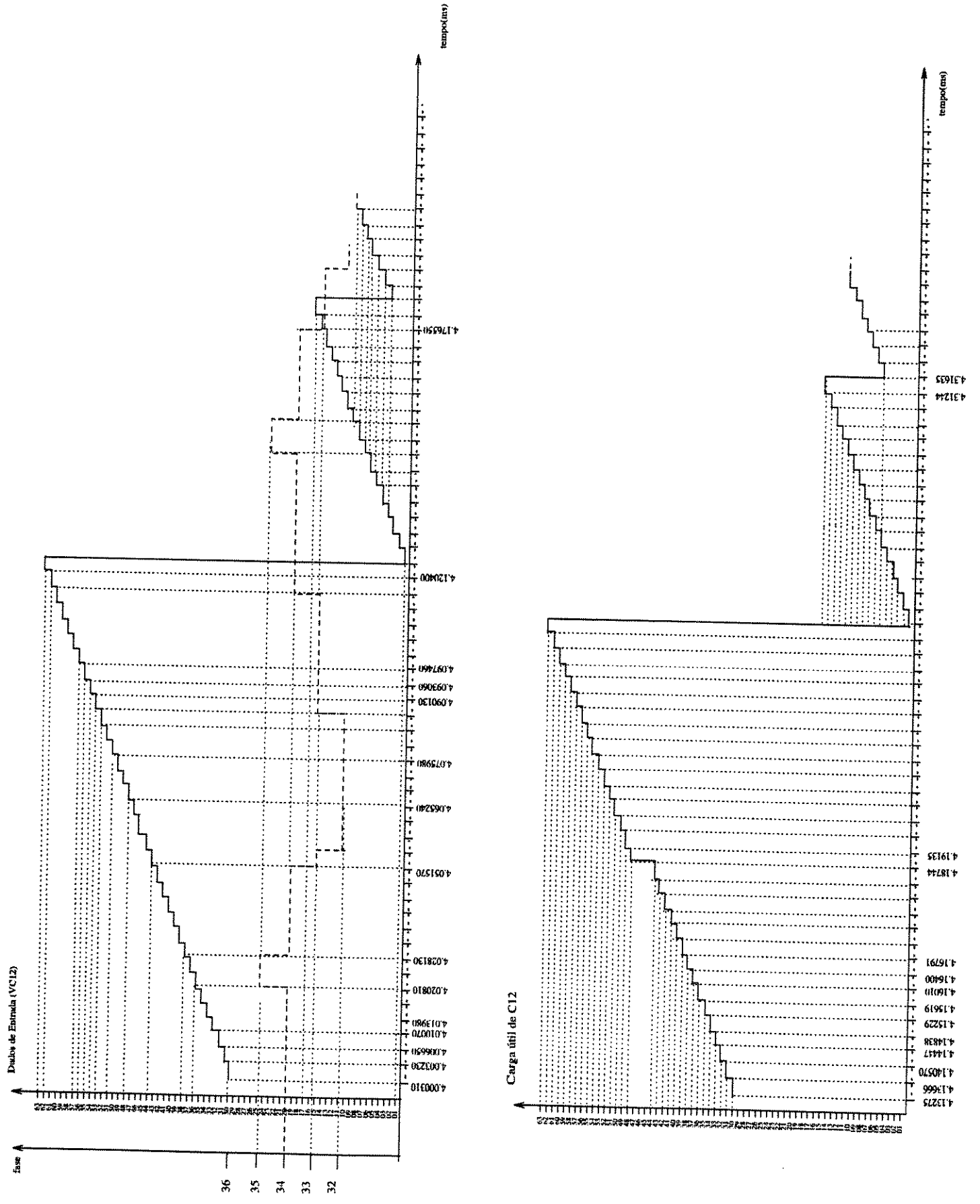
32 bytes - 125 μs
X bytes - Y μs;

A figura 5.22 mostra como é a retirada dos bytes de C12.

O primeiro gráfico da figura 5.22 mostra a escrita de dados úteis na memória elástica e a fase correspondente em relação à leitura. Este são os dados que foram gerados e processados.

O segundo gráfico mostra a retirada dos dados através do relógio de leitura na forma de byte(C12), o objetivo é verificar a correta retirada destes dados. A seguir mostramos como se interpreta estes gráficos, utilizando alguns cálculos para verificar a correta retirada dos dados.

Figura 5.22 : *Retirada Correta dos Canais*



Cálculos

Dados de entrada:

- VC12=30 _ tempo(ms)=4.000310 _ fase = 34;
- VC12=36 _ tempo(ms)=4.020810 _ fase = 35;
- VC12=38 _ tempo(ms)=4.028130 _ fase = 35;
- VC12=44 _ tempo(ms)=4.051570 _ fase = 34;
- VC12=48 _ tempo(ms)=4.065240 _ fase = 32;

Dados de saída:

- C12=30 _ tempo(ms)=4.13275;
- C12=36 _ tempo(ms)=4.15619;
- C12=38 _ tempo(ms)=4.16400;
- C12=44 _ tempo(ms)=4.18744;
- C12=48 _ tempo(ms)=4.19135;

Para verificar se os dados retirados estão corretos basta subtrair o tempo de C12 pelo de VC12 do respectivo dado e comparar com o tempo dado pela fase.

- 4.13275ms - 4.000310ms = 132.44μs ;
- 4.15619ms - 4.020810ms = 135.4μs;
- 4.16400ms - 4.028130ms = 135.87μs;
- 4.18744ms - 4.051570ms = 135.87μs;
- 4.19135ms - 4.065240ms = 126.11μs;

Observe que entre os tempos (4.31635ms-4.19135ms=125μ) (**Gráfico do Conteúdo de C12**)retira-se exatamente 32 canais. Os blocos funcionais, descritos no capítulo 4 para uma rede SDH, podem ser implementados das mais variadas formas, inclusive simulando redes em anel com a retirada de tributários de 2 Mb/s, em algumas estações, e a continuação da informação para outras estações do enlace. Optou-se, no exemplo anterior, por uma simulação simples para verificar o trafego dos dados a nível de STM-1 e a retirada de canais através do mapeamento síncrono de byte. Os equipamenos SDH estão sendo implementados básicamente para o mapeamento de tributários síncronos, seria interessante desenvolver uma placa que aceita-se os tributários síncronos e plesiócronos utilizando dois tipos de mapeamento. A seguir tem-se um exemplo de um equipamento que aceitaria os dois tipos de tributários através da interpretação dos bytes do POH de VC12.

5.3 Retirada de Sinais Utilizando Mapeamento Síncrono/ Assíncrono de 2048 kb/s.

Nesta parte do trabalho mostra-se uma outra rede que possibilita a retirada tanto de tributários síncronos quanto assíncronos, onde os seus sinais serão retirados de forma serial, ou seja, na forma de bit.

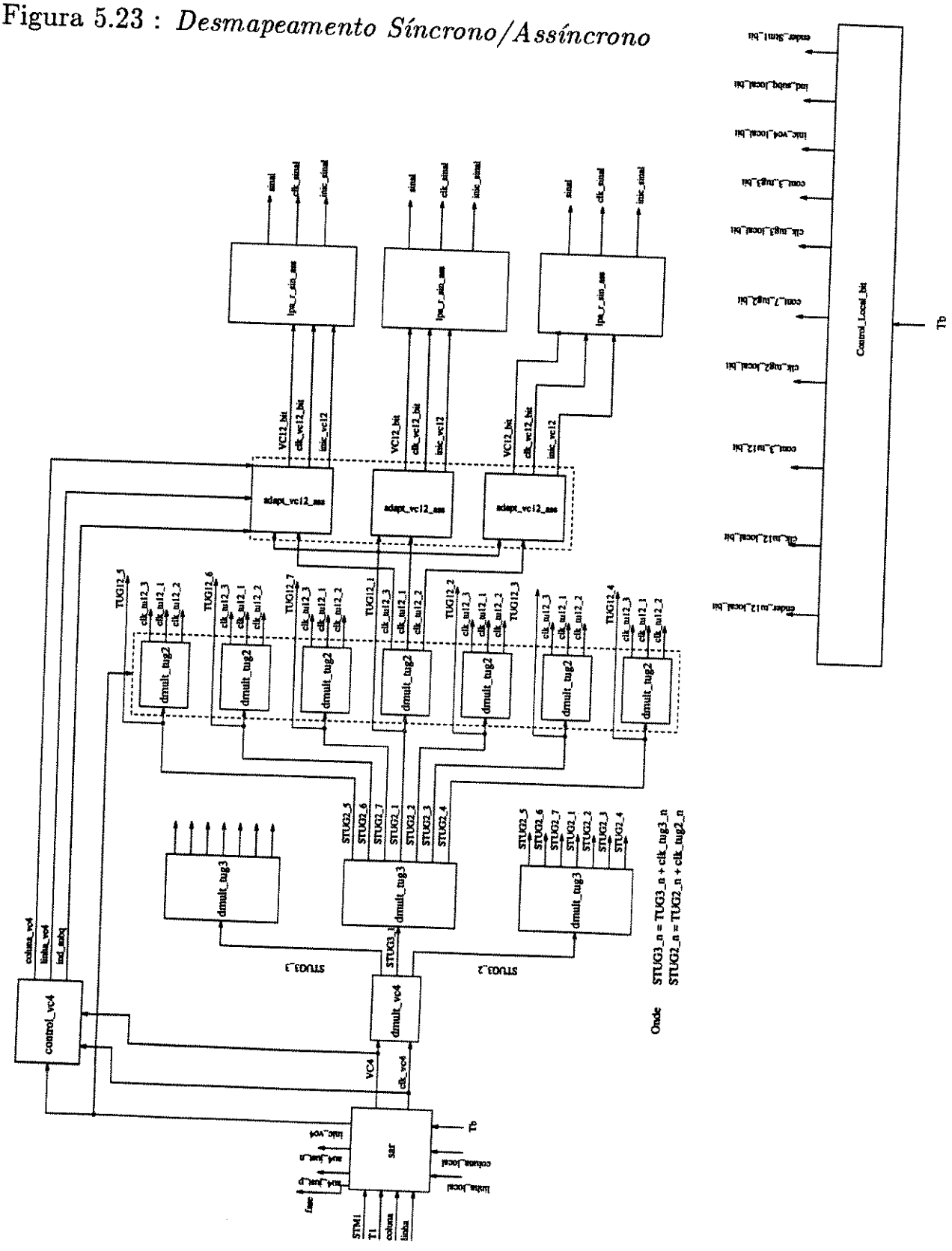
5.3.1 Desmapeamento Síncrono/Assíncrono

5.3.2 Realização do Enlace#2

Este é um outro arranjo de blocos funcionais de forma a se ter um equipamento que faça a retirada de sinais síncronos e assíncronos. Da mesma forma que foi feito anteriormente, vamos mostrar as entidades que formam este equipamento; são estas entidades responsáveis pelo desmapeamento(O diagrama de blocos da figura 4.31 no capítulo 4 mostra o desmapeamento Síncrono/Assícrono). O sinal entra em uma ponta do equipamento na forma de um STM1-byte, é submetido às operações de adaptação, demultiplexação e desmapeamento síncrono/assíncrono tendo na sua saída um sinal de 2048 kb/s em forma de bit. Este sinal é codificado através de um codificador HDB3, por exemplo, e liberado para a linha. A figura 5.23 mostra o diagrama de blocos deste equipamento. Não temos conhecimento se este tipo de equipamento, que aceita tanto tributários síncronos quanto assíncronos, tenha sido

implementado pelos fabricantes. Aqui será feita uma simulação com a retirada de tributários, mas sem ocorrer as justificações de mapeamento nos bits S1 e S2 que são características da PDH. Os blocos funcionais desenvolvidos anteriormente[2] foram: sar,control_vc4,dmult_vc4,dmult_tug3,dmult_tug2. Os demais blocos foram desenvolvidos neste trabalho.

Figura 5.23 : *Desmapeamento Síncrono/Assíncrono*



ENTIDADES:

Entidade *adapt_vc12_r_bit* (Adaptador de VC12 na Recepção-bit.)

Declaração de Entidade em VHDL:

```
ENTITY adapt_vc12_r_bit IS
   PORT(TUG2: IN BIT_VECTOR(1 TO 8);
      ind_subq, ind_subq_local_bit, ender_tu12_local_bit,coluna_vc4,linha_vc4: IN INTEGER:=0;
      clk_tu12, clk_tu12_local_bit: IN BIT;
      just_p, just_n, inic_vc12, clk_vc12_bit: BUFFER BIT;
      fase: BUFFER INTEGER:=0;
      VC12_bit: OUT BIT_VECTOR(1 TO 8));
END adapt_vc12_r_bit;
```

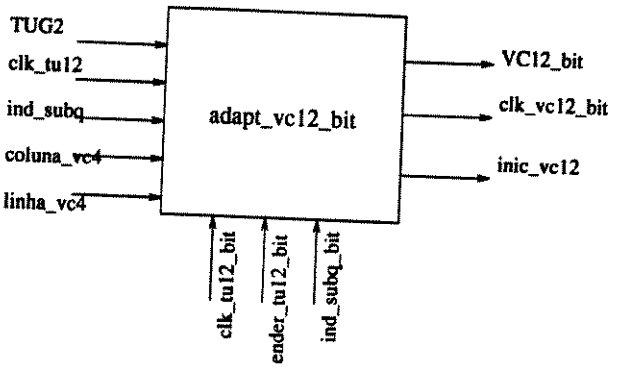


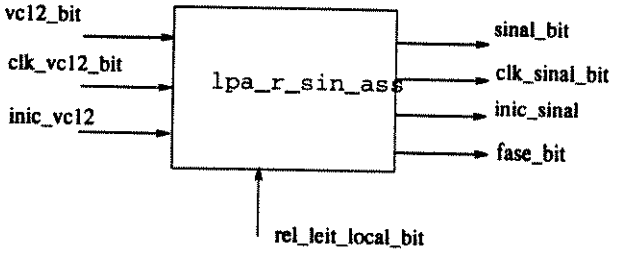
Figura 5.24 :Ilustração do Componente *adapt_vc12_r_bit*

Entidade *lpa_r_sin_ass* (Rescepção de VC12 sin/ass)

Declaração de Entidade em VHDL:

```
ENTITY lpa_r_sin_ass IS
   PORT(VC12_bit: IN BIT;
      clk_vc12_bit,inic_vc12_rel_leit_local:IN BIT;
      sinal: OUT BIT;
      clk_sinal,inic_sinal:OUT BIT);
END lpa_r_sin_ass;
```

Figura 5.25 : Ilustração do componente *lpa_r_sin_ass*

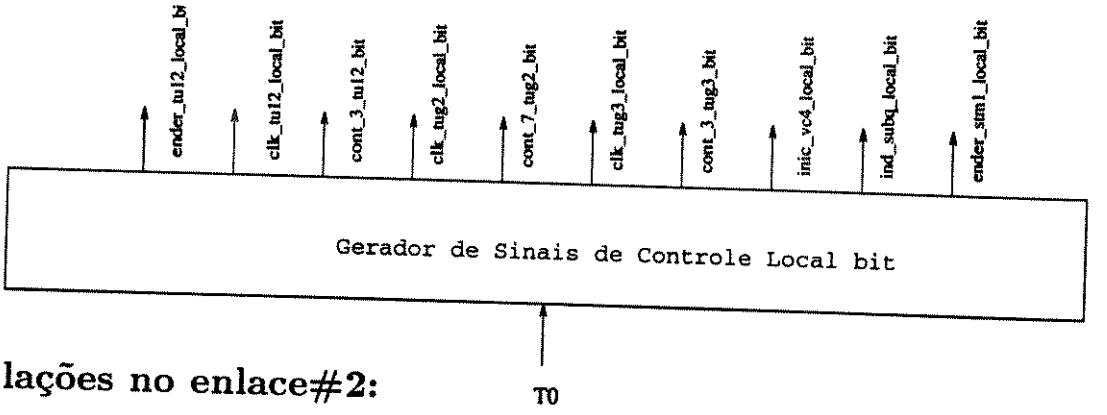


Entidade *control_local_bit* **(Gerador de Sinais de Controle Local bit.)**

Declaração de Entidade em VHDL:

```
ENTITY control_local_bit IS
   PORT(T0: IN BIT;
      ender_stm1_local_bit, ind_subq_local_bit, cont_3_tug3_bit,
      cont_7_tug2_bit, cont_3_tu12_bit, ender_tu12_local_bit: BUFFER INTEGER:=0;
      inic_vc4_local_bit, clk_tug3_local_bit, clk_tug2_local_bit, clk_tu12_local_bit: BUFFER BIT);
END control_local_bit;
```

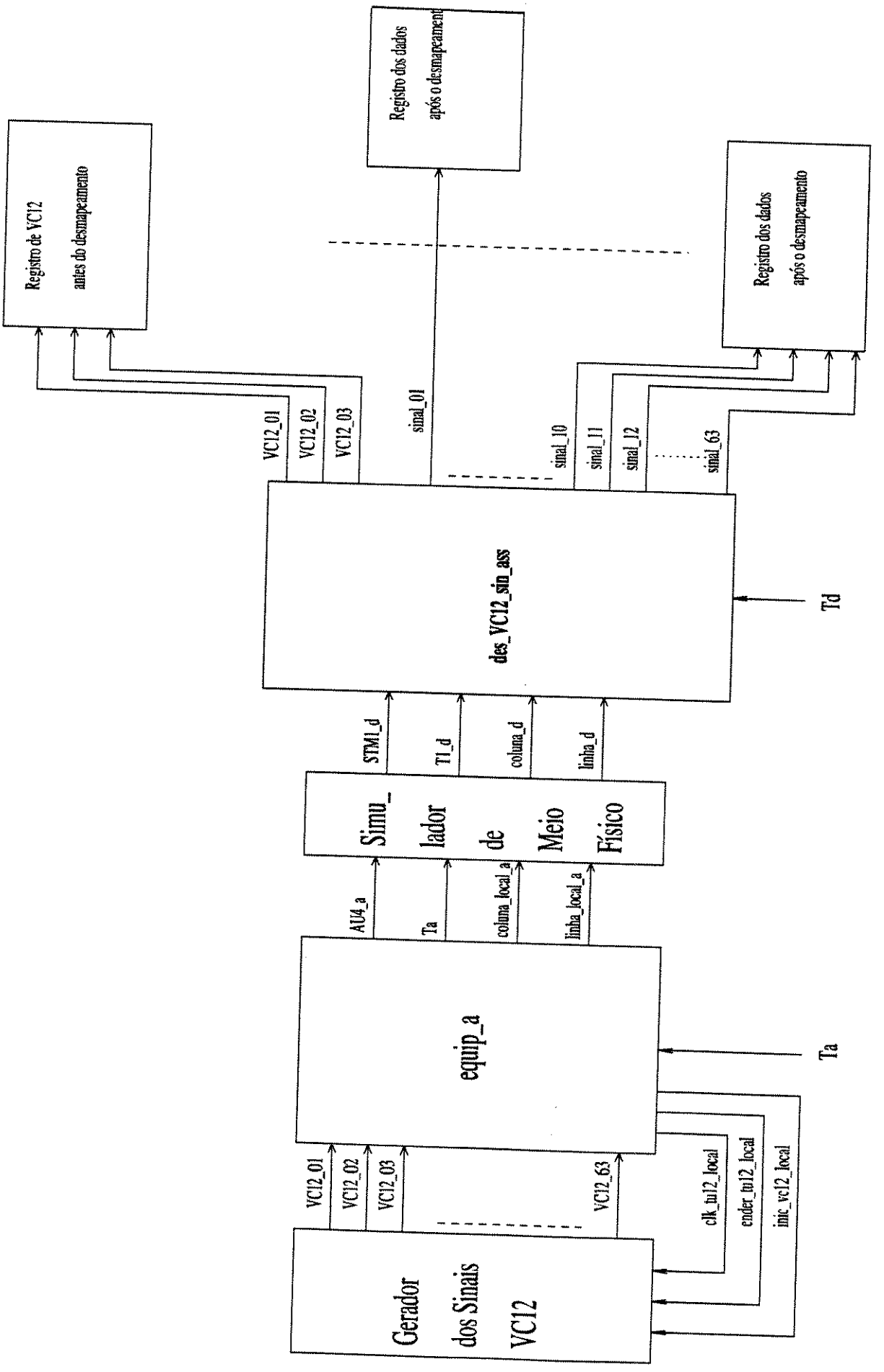
Figura 5.26 : Ilustração do Componente *control_local_bit*



5.3.3 Simulações no enlace#2:

Também utiliza-se aqui o *equip_a* interligado através do simulador de meio físico com o equipamento *des_vc12_sin_ass*. Estes dados serão registrados e analisados. A figura 5.27 mostra o enlace#2.

Figura 5.27 : *Enlace#2*



O ITU-TSS recomenda que para sistemas plesiócronos os relógios devem trabalhar com uma variação máxima de 50 ppm e esta será usada no relógio remoto. O relógio local terá valor nominal. A diferença é que agora tem-se um relógio local de leitura na forma de bit **rel_leit_bit**. Após a demultiplexação de um sinal agregado, a interpretação do POH de VC_12 define se tributários são síncronos ou assíncronos. Conforme esta interpretação teremos um ou outro tipo de desmapeamento. A gerência de rede é quem define o tipo de tributário, atuando no byte V5.. Fez-se com que o POH de VC12 tivesse valores correspondentes a L1=1,L2=0,L3=0 configurando um mapeamento síncrono, mas outros valores de POH podem ser usados para que configure um mapeamento assíncrono.

Relógios Ta, Td e rel_leit_bit:

```
Ta <= NOT(Ta) AFTER 25.718878665 ns;

Td <= NOT(Td) AFTER 25.718878665 ns;

rel_leit_bit <= NOT(rel_leit_bit) AFTER 244.140625000 ns;
```

Nesta fase vamos mostrar a retirada correta dos dados em forma de bit. A figura 5.28 mostra os gráficos dos dados de entrada na memória elástica (VC12) e os dados de saída. Também aqui deve-se analisar os dados que entram no lpa_r_sin_ass, a fase_bit e os dados de saida (C12_bit), pois o raciocínio é o mesmo seguido nas simulações anteriores, ou seja, sabendo-se o dado de entrada e a fase pode-se perfeitamente dizer onde estará o dado de saída.

Para verificar se os dados estão sendo retirados corretamente basta seguir a fórmula abaixo.

- tempo_dado_saída = tempo_dado_entrada + (fase_bit x 1/2048 kb/s);

Seguindo a equação:

Para dado de VC12_01 = 30(00011110);

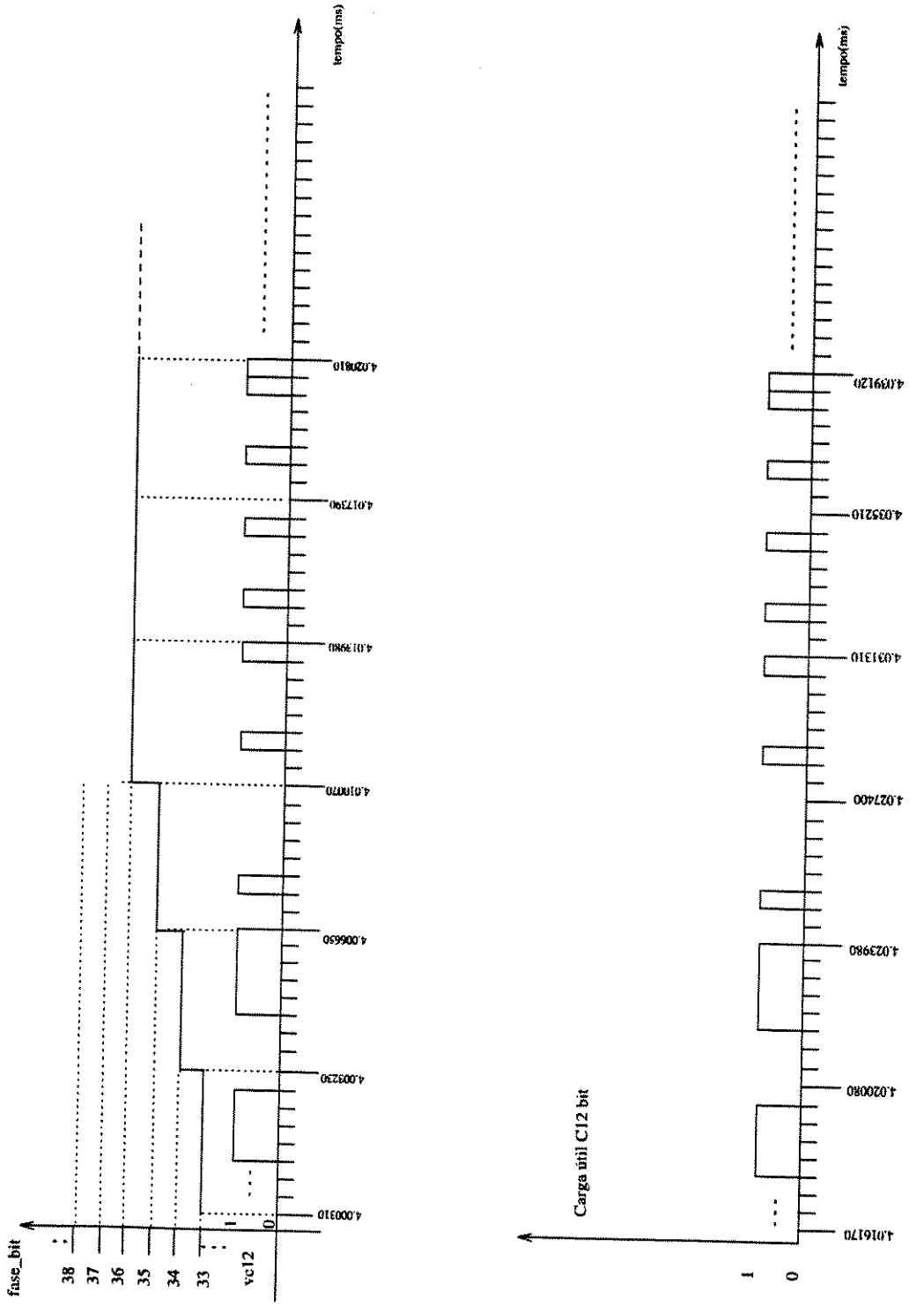
Para fase da memória=33;

tempo_dado_saida = 4.000310ms + (33 x 1/2048 kb/s); tempo_dado_saida = 4.016423ms;

Observe que o primeiro dado desmapeado inicia com o tempo = 4.016170ms

com uma pequena diferença em relação ao valor 4.016423ms, e a sequência de bits mostra o valor 30(Dados de Saída=00011110). Uma simples observação da figura 5.28 mostra que os dados estão saindo de forma correta. Estes dados seguem para a interface óptica para a transmissão. Estas simulações foram feitas de forma que não ocorresse justificação de mapeamento, ou seja, interpretar informação e enchimento nos bits S1 e S2 respectivamente, lembrando que em operação normal S1 carrega enchimento e S2 carrega informação. No próximo capítulo teremos algumas conclusões importantes a respeito dos estudos anteriores.

Figura 5.28 :*Retirada da Carga Útil bit.*



Capítulo 6

Conclusões

6.1 Conclusões

Neste trabalho, desenvolvemos vários estudos relativos à Hierarquia Digital Síncrona, com enfoque especial para mapeamentos e desmapeamentos nos níveis de 2Mb/s e 64 Kb/s. Também mostramos equipamentos para a retirada de sinais síncronos com fontes distintas de síncronismo. Estudou-se o desenvolvimento de um bloco funcional tanto para tributários síncronos quanto assíncronos em função destes estudos deduziram-se as seguites conclusões:

No capítulo 3:

Apresentamos formulações para canais de 64 Kb/s mapeados em containers VC-12 e VC-11. No mapeamento síncrono de byte tem-se a visibilidade direta de canais de 64kb/s, uma formulação específica que possibilite a retirada direta destes canais pode ser útil em certos equipamentos SDH, como os ADM's de grande porte. O acesso rápido a um determinado canal, dentro deste equipamento, reduziria as operações de demultiplexação, utilizada em todos os níveis desta hierarquia. Um outro exemplo de utilização, seria o acesso a um determinado byte V5 de um

certo tributário que vem por um agregado de um terminal remoto. Com este acesso, pode-se determinar o tipo de mapeamento que foi utilizado por este VC, sem passar por algumas etapas de demultiplexação. Enfim, pode-se desenvolver, com estas formulações para 2Mb/s e 1.5 Mb/s (considerando que o mapeamento é do tipo síncrono de byte), dispositivos de processamento paralelo que acessem estas informações com maior velocidade. As formulações podem ser utéis também para acessar rápidamente as funções de gerenciamento. Através dos pontos Sn, retirados dos POH's dos VC's e SOH, sendo reportadas informações de falha,desempenho,configuração a SEMF e depois a MCF, que, por sua vez, alimenta, pela interface Q, o Centro de Gerênciamento de Rede(CGR).

No capítulo 4:

Foram desenvolvidos os blocos funcionais em linguagem VHDL com o objetivo de montar equipamentos para o estudo da HDS. Desenvolveram-se blocos para a Recepção Síncrona , Recepção Síncrona/Assíncrona, com um destaque especial para os blocos LPA_R e lpa_r_sin_ass.

No capítulo 5:

Foram estudadas ocorrências de SLIP. Conclui-se que para equipamentos SDH com entradas de tributários síncronos, se ocorrer a perda de síncronismo, passando os equipamentos a operar com relógio interno e pelo fato de as memórias elásticas usadas nestes equipamentos serem de pequeno porte (preve-se na literatura que para mapeamento síncrono de byte deve-se usar uma memória elástica de pequeno porte, somente para acomodar variações de fase lentas "wander"

e variações rápidas "jitter"), o fenômeno do SLIP ocorrerá. Os escorregamentos nas nossas simulações foram de um numero inteiro de quadros, ou seja, o slip foi do tipo controlado; portanto, as memórias elásticas usadas por estes equipamentos devem prever este fenômeno.

A retirada correta dos dados para mapeamento síncrono de byte demonstrou a correta simulação de um enlace ponto a ponto de 155 Mb/s com os relógios dos equipamentos utilizando uma fonte de sincronismo externa. Quando as simulações foram feitas com os relógios em sincronismo, observou-se que a fase da memória elástica de desmapeamento aumenta muito lentamente, permanecendo quase constante. Deve-se usar um PLL na saída da memória elástica de desmapeamento; com isto, o relógio de leitura acompanha as variações do relógio de escrita e dados não deixarão de ser lidos.

Os atuais equipamentos SDH possuem somente entradas para tributários síncronos da rede PDH mas conforme as normas do ITU-TSS seria possível distinguir o tipo de carga útil transportada pelo VC através da interpretação do byte V5, com o desenvolvimento do bloco funcional para desmapear sinais síncronos e assíncronos e suas simulações ficou demonstrado que é possível o desenvolvimento de bloco de mapeamento/desmapeamento híbrido, ou seja, tanto para sinais síncronos quanto para sinais assíncronos.Este tipo de bloco funcional aparentemente ainda não foi implementado pelos fabricantes. Pode-se usar o bloco funcional descrito no capítulo 4 ìtem 4.3.6 como modelo.

Futuros Trabalhos

- **Realizar a nível de ASIC os circuitos já descritos em VHDL. Tem-se varias ferramentas que possibilitam chegar ao desenvolvimento de um chip partindo-se do desenvolvimento de um sistema em alto nível. Deixo como sugestão utilizar as ferramentas de síntese da Menthor Graphics ou da Synopsys para desenvolver circuitos integrados. No caso de se querer um desenvolvimento rápido de um chip, a sugestão é que se utilize as ferramentas da Altera para FPGA.**
- **Desenvolver um processamento paralelo para retirada de canais específicos tomando como base os desenvolvimentos matemáticos feitos no capítulo 3.**
- **Desenvolver os demais blocos funcionais da HDS, especifamente os blocos de gerênciamento.**

Bibliografia

[1]Recomendação G.70X-Network Node interface for Synchronous Digital Hierarchy, ITU TSS,1994.

[2]Macedo A. S.,"Modelos em VHDL para Equipamentos da Hierarquia Digital Síncrona", Tese Mestrado-Unicamp,1993.

[3]Perry D. L.,"VHDL", McGraw-Hill,1991.

[4]Fudoli,T.R.T.,"Redução de Jitter de justificação na Hierarquia Digital Síncrona, Tese Mestrado-Unicamp, 1992.

[5]Armstrong J.R.,"Chip-Level Modeling with VHDL", Prentice Hall, 1989.

[6]V8.0 An Introduction to Modeling in VHDL Student Workbook, Menthor Graphics,1994.

[7]Synchronous Transmission Systems, Northem Telecom, 1994.

[3]Lamport L.,"Latex: A Document Preparation System,Addison-Wesley Publishing Company, 1986.

[9]SDH- Introdução a Rede Síncrona.

UNICAMP-Curso de Pós-Graduação.

Prof. Rege R. Scarabucci.

[10]Machado,Luis Flávio Collares;Scarabucci,Rege Romeu:Mathematical Representation for DS0 in SDH and SONET-Paper submetido ao 15TH INTERNATIONAL TELETRAFFIC CONGRESS,Washington, D.C.,USA,1996.

[11]Machado,Luis Flávio Collares; Scarabucci, Rege Romeu: Mapeamento Síncrono Utilizando VHDL: Trabalho apresentado no SEGUNDO WORKSHOP IBERCHIP,São Paulo, Brasil,1996.