GUILHERME DE OLIVEIRA FIORELLI

Desenvolvimento de uma interface de comunicação entre placas processadoras padrão ETX e placas-módulos de equipamentos da plataforma Vectura

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em
Eletrônica

ORIENTADORA: Profª. Mônica de Lacerda Rocha

São Carlos
2008

DEDICATÓRIA

Ao meu pai, Paulo Miguel Fiorelli, exemplo de pessoa, marido e pai, que sempre me apoiou, incentivou e às vezes até cobrou resultados para que hoje eu estivesse onde estou, me formando em uma das melhores universidades e com o pensamento sempre voltado para o futuro. Obrigado pai por tudo, sei que de onde estiver sempre estará olhando por todos nós

AGRADECIMENTOS

Agradeço ao Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPqD), local de estágio e por conseqüência do trabalho de conclusão de curso, à Trópico proprietária dos equipamentos aqui descritos, ao gerente de hardware Victor Alfonso Valenzuela Diaz, que se dispôs a oferecer ajuda sempre que necessário e realmente o fez, a todos os funcionários e amigos da empresa que de certa forma colaboraram para o êxito desse documento, à minha orientadora Profª. Mônica de Lacerda Rocha que se dispôs a ler, verificar e aprovar minha monografia mesmo com o prazo curto. E finalmente agradeço à minha mãe que procurou organizar a casa para que eu pudesse ter sossego na hora de fazer esse documento.

Sumário

DEDICATÓRIA 2
AGRADECIMENTOS 3
Índice de Tabelas e Figuras 5
Tabela de Siglas 5
RESUMO 7
ABSTRACT 8
1. INTRODUÇÃO 9
2. NGN (Next Generation Network) 11
2.1. Camada de Transporte 12
2.2. Camada de Controle 12
2.3. Camada de Serviços 12
3. A Plataforma Vectura na NGN 13
3.1 Perspectiva para o futuro 14
4. Evolução do equipamento 15
5. A interface de comunicação 19
5.1 Interface Usando VHDL 20
5.2 Considerações finais do projeto (TCC) 21
6. Equipamento Vectura 23
6.1. Arquitetura 23
6.2 Aplicações e Configurações 26
6.3 Capacidade 26
6.4 Características principais do VES (TROPICO RA) 27
7. Conclusão 28
8. Referências Bibliográficas 29
Apêndice 1 – Erlang 30
Apêndice 2 - VHDL 33
A2.1 Introdução 33
A2.1.1 Por que usar VHDL? 33
A2.3 Par ENTITY (Entidade) / ARCHITETURE (arquitetura) 33
A2.4 Operações lógicas e comentários. 34
A2.5 COMPONENT (componente) 35
A2.6 PROCESS(processo)/ IF CASE WHEN 36
A2.7 Tipos de dados e biblioteca 1164 39
A2.8 Sinal e máquina de estado 40
Apêndice 3 - Canal Comum nº7 (SS#7 ou SS7) 43

Índice de Tabelas e Figuras

Figura 2.1: Modelo de 3 camadas 12
Figura 3.1: Equipamentos Vectura na NGN 13
Figura 4.1: Placa módulo com ETX utilizando IH4 16
Figura 4.2: Gaveta com os módulos ETX com IH4 17
Figura 4.3: Gaveta sem IH4 17
Figura 4.4: Arquitetura sem IH4 18
Figura 5.1: Placa Módulo com placa processadora atual 19
Figura 5.2: Placa Módulo com placa padrão ETX 20
Figura 5.3: Digrama funcional em blocos do MT8980D 20
Figura 6.1: Arquitetura da Plataforma Vectura 24
Tabela 6.1: Valores típicos de capacidade da CPA-T 27
Figura A1.1: Exemplo de ligação de duas centrais 30
Figura A1.2: Gráficos canais ocupados em função do tempo 30
Figura A1.3: Gráfico Erlang em função das horas do dia 31
Figura A2.1: Máquina de Estado 41
Figura A3.1: Camadas do SS7 43

Tabela de Siglas

ETX	Embedded Technology eXtended – placa processadora de padrão comercial.
NGN	Next Generation Network – redes de nova geração
SS7	Sistema de sinalização número 7 ou canal comum número 7 – conjunto de protocolos de sinalização
VHDL	Vhsic Hardware Description Language – linguagem de programação de dispositivos programáveis.
VES	Vectura Edge Switch – Central telefônica de processamento armazenado
CPA	Central telefônica de processamento armazenado
CPqD	Centro de Pesquisa e Desenvolvimento em Telecomunicações
FPGA	Field-programmable gate array – dispositivo programável
OPT	Open Packet Telecommunications - arquitetura da NGN
RTPC	Rede de telefonia pública comutada
VSS	Vectura Softswitch - Softswitch da Plataforma Vectura
TDM	Multiplexação por divisão de tempo
IP	Internet Protocol
MTD	Módulo tronco digital
MCC	Módulo de canal comum
MMA	Módulo Auxiliar
IH4	Interface de hardware proprietária da empresa Trópico.
MX	Módulo de comutação
MT	Módulo de terminais
MO	Módulo de operação
MZ	Módulo de sinalização
MS	Módulo de sincronismo

ITD	Circuito que trata os troncos digitais
ICC	Circuito que trata o canal comum
IFT	Circuito que gera as tensões de alimentação
EPLD	Dispositivo lógico programável
MPG	Módulo processador genérico
Eth	Interface Ethernet
BI	Bloco de implementação – software ou parte software
MTP	Message Transfer Part – parte de mensagem de transferência
ISUP	ISDN User Part
TUP	Telephone User Part
SCCP	Signaling Connection Control Part
TCAP	Transaction Capabilities Application Part
INAP	Internet Network Application Part
MAP	Mobile Application Part

RESUMO

ETX (Embedded Technology eXtended) é um padrão comercial de processamento do tipo "computer-on-module" (COM), altamente integrado e compacto. Cada COM de padrão ETX integra: um núcleo CPU (ou multi-núcleo), memória, grupos I/O comuns em PC's (serial, paralelo, etc.), USB, áudio, porta gráfica e Ethernet. Todos os sinais I / O e uma completa implementação de barramentos ISA e PCA são mapeados em quarto conectores de alta densidade na base da placa. Assim, as placas ETX são projetadas para serem inseridas numa placa-mãe e contêm recursos adicionais de memória, periféricos e interfaces. Esta padronização permite o desenvolvimento da placa mãe usando módulos ETX com placas-filhas e os módulos ETX podem ser fabricados por diferentes fabricantes e podem incorporar processadores mais avançados.

O projeto de formatura consiste no desenvolvimento de uma interface de comunicação entre as placas módulo de uma central telefônica, denominada Plataforma Vectura, e as placas processadoras padrão ETX. O intuito de utilização desse tipo de placa é aumentar a capacidade de processamento das placas módulo, migração futura para uma comunicação via ethernet entre placas, e acompanhar a evolução do mercado, uma vez que ETX é um padrão comercial. Outro fato importante é que como existem muitos fabricantes dos dispositivos empregados, o fornecimento dessas placas fica garantido e também seu preço competitivo.

Dessa forma, as placas ETX serão inseridas como placas periféricas numa placa mãe que contém recursos adicionais de memória, periféricos e interfaces. Essa placa mãe, juntamente com seus periféricos, forma o que denominamos aqui de placa módulo que dependendo de sua configuração, pode desempenhar vários papéis dentro de uma central telefônica, como o tratador de canal comum ou a módulo tronco digital. Esses módulos fazem parte de um equipamento denominado Vectura Edge Switch (VES) que por sua vez faz parte da Plataforma Vectura desenvolvido pelo CPqD (Centro de Pesquisa e Desenvolvimento em Telecomunicações) e comercializado pela empresa Trópico.

Palavras-chave

Plataforma Vectura, NGN, SS7, VHDL, ETX

ABSTRACT

ETX - Embedded Technology eXtended, is one of "computer-on-module" (COM) standard. It is a trade processing board standard, highly integrated and compact. Each COM with standard ETX integrates a core CPU (or multicore), memory, the I / O common on PCs (serial, parallel, etc..), USB, audio, graphics and Ethernet. All signs of I / O and a full implementation of ISA buses and PCA are mapped in four high-density connectors at the bottom of the board. Thus, the ETX boards are designed to be inserted into a motherboard that contains additional resources of memory, peripherals and interfaces. This standardization allows develop motherboard such using ETX modules with daughterboard that ETX module can be more than one manufacturer and can evolve to incorporate more advanced processors.

This Final Course Project (TCC) is an interface for communication development between the standard ETX (Embedded Technology eXtended) of processing boards and the board-module from Vectura Platform. These processing cards are commercially standardized with specific numbers of I/Os, connectors and standards of communication, for example, the Ethernet and the Serial communication. The reasons to use this type of card are the following: to increase the processing capacity in the module cards, future migration to the Ethernet communication between cards and to follow the development of processing cards. Another important aspect is, as there are many suppliers of these processing cards, what ensures market availability in the competitive price.

The ETX card will be connected as a daughter-board in a motherboard that contains additional resources of memories, interfaces and peripherals. This motherboard with its daughter-boards consists in what is called “board-module”. Depending on the configuration, the board-module can assume many functions in a Telephony Central, as common channel module or digital trunk module. These modules are parts of an equipment called “Vectura Edge Switch” VES, which is part of Vectura Platform, design by CPqD and sold by Trópico Company.

Keywords

Vectura Platform, NGN, SS7, VHDL, ETX

1. INTRODUÇÃO

Uma central telefônica do tipo processamento armazenado (CPA), também chamada na Plataforma Vectura de Vectura Edge Switch (VES), é formada por um conjunto de placas denominadas placas-módulo, cada uma desempenhando um papel especifico dentro da central. Dependendo da configuração dessas placas em relação as quantidades de cada uma e quais placas estariam em uma gaveta, a central telefônica pode ser configurada para tratar diversas intensidades de tráfego, onde cada equipamento VES tem capacidade máxima de tráfego de 12600 E (erlang – vide Apêndice 1). As CPA's fazem parte da camada de transporte da arquitetura OPT (Open Packet Telecommunication – figura 4.1) dentro do conceito de redes de nova geração NGN, descrito na seção 4 dessa monografia.

O projeto de formatura está inserido no contexto de equipamentos de uma Plataforma de controle e comutação Vectura. Tal plataforma apresenta tanto partes proprietárias, ou seja hardware e software desenvolvidos pelo CPqD, como partes comerciais, como servidores de alto desempenho e switches Ethernet. Justamente na parte proprietária é que estão as placas módulos, as quais irão receber como placas periféricas, em uma evolução, as placas processadoras ETX, que terão como finalidades a de aumentar a capacidade de processamento das placas módulos e acompanhar a evolução dos processadores no mercado.

O trabalho de conclusão de curso refere-se à programação de dispositivos lógicos programáveis como FPGA's[1], onde serão criados mecanismos para que se tenha uma interface de comunicação entre as placas processadoras padrão ETX e as placas módulos, que por sua vez têm interfaces de comunicação proprietárias da empresa Trópico. A programação desses FPGA's é feita utilizando a linguagem de programação VHDL ("**V**HSIC **H**ardware **D**escription **L**anguage"- Linguagem de descrição de hardware) (Apêndice 2).

Em virtude de estarmos tratando de um sistema proprietário de comutação digital, este documento não tem como característica o fornecimento de quaisquer detalhes técnicos do sistema telefônico como os códigos de programação em VDHL desenvolvidos no projeto que possam vir a prejudicar e/ou desrespeitar a propriedade intelectual da empresa CPqD ou TRÓPICO.

Vale enfatizar aqui que o desenvolvimento de interface de hardware baseada em placas ETX faz parte de um projeto maior de evolução para comunicação entre placas totalmente interna à gaveta que se encontra em andamento com previsão de término para janeiro de 2009.

Esta monografia está organizada como se segue: a Seção 2 descreve os principais conceitos da tecnologia conhecida como NGN (Next Generation Network) e a Seção 3 situa a Plataforma

Vectura na NGN, introduzindo o cenário para a evolução do equipamento, descrita na Seção 4. A interface de comunicação, da qual faz parte o projeto de conclusão de curso, é apresentada na Seção 5 e mais informações do equipamento Vectura são apresentadas na Seção 6. A Seção 7 conclui o trabalho, que também apresenta três anexos: Apêndice 1, que define o Erlang, o Apêndice 2, que descreve os princípios de programação VHDL e o Apêndice 3, que apresenta os principais conceitos de canal comum.

---

[1]FPGA (Field-programmable gate array) é um dispositivo semicondutor contendo componentes lógicos programáveis chamados de "blocos lógicos" e interconexões programáveis.

2. NGN (Next Generation Network)

Os equipamentos da Plataforma Vectura, assim como o VES (onde estão as placas módulos do projeto de formatura), estão inseridos no contexto de redes de nova geração NGN que é um conjunto de novos conceitos de uma rede centrada em dados, que utilizam normalmente protocolo TCP/IP. Tal protocolo carrega informações de dados, voz e vídeo, que serão transmitidos por essa rede centrada em dados. A informações de voz provenientes das CPA's através de gateways são convertidas em pacotes TCP/IP e a partir de então passam a trafegar nessa rede obedecendo os respectivos protocolos de sinalização e direcionamento dessas mensagens.

Esse funcionamento do VES concomitantemente com a rede IP vem da necessidade de convergência em uma única rede de dados, necessidade essa devida ao mercado cada vez mais competitivo, onde operadoras de TV por assinatura (vídeo) vêm disponibilizando Internet (dados) e serviços de telefonia (voz) a seus clientes. Para entrar nesse atual mercado e ser competitivo é preciso oferecer qualidade de serviço e serviços cada vez mais diferenciados. Para isso há necessidade de uma migração, por parte das operadoras de rede telefonia fixa, para uma plataforma que integre os serviços de convergência das redes de próxima geração NGN.

A arquitetura desse tipo de rede, capacitada a funcionar junto com o legado tanto atendendo redes públicas de milhões de usuários, como se integrando de forma simples e completa com as redes centradas em dados já existentes, é a denominada Arquitetura OPT - Open Packet Telcommunications (Telecomunicações por Pacotes Aberta). Diferentemente das centrais telefônicas convencionais de controle distribuído, uma rede OPT tem controle centralizado e o que faz esse controle é um equipamento denominado Softswitch.

A arquitetura OPT tem uma alta flexibilidade devido a ser estruturada em 3 camadas: a camada de serviços, a camada de controle de chamadas e a camada de acesso e de transporte, como mostrado na Figura 2.1. A separação entre as camadas é feita através de interfaces abertas, proporcionando tal flexibilidade e interoperabilidade entre equipamentos e sistemas localizados nas diversas camadas.

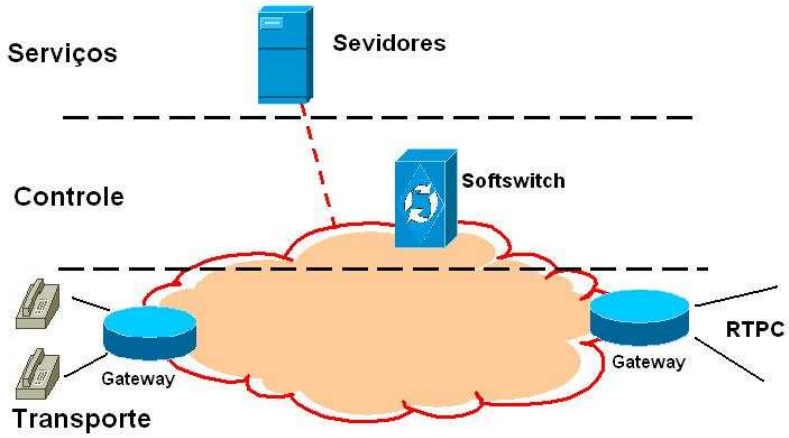


Figura 2.1: Modelo de 3 camadas

2.1. Camada de Transporte

É a camada responsável pelo transporte dos dados, podendo ser uma rede IP ou uma rede ATM. Nessa camada estão os gateways de mídia, que executam a codificação de mídias entre redes diferentes e toda a infra-estrutura da rede de dados, como os comutadores de pacotes ATM e roteadores IP.

2.2. Camada de Controle

Nessa camada ocorre o estabelecimento, supervisão e liberação das chamadas que trafegam pela rede de pacotes. É a parte mais estratégica da arquitetura das redes convergentes, sendo um ponto conveniente para o acesso a novos serviços e funcionando junto com o legado, que é constituído por redes CPA-T, por exemplo. Nesta camada o equipamento principal é o Softswitch., que controla os *Gateways* VoIP.

2.3. Camada de Serviços

Essa camada é responsável por controlar os serviços oferecidos aos usuários conectados às centrais periféricas da rede, operada por servidores na rede IP. Ao fazer o controle de processamento das chamadas entre as centrais conectadas àquela rede, o Softswitch trata os serviços para os usuários por meio dos servidores da rede.

3. A Plataforma Vectura na NGN

Como comentado no item anterior, o Vectura Edge Switch (VES) está inserido nesse contexto de NGN comunicando-se através de gateways com a rede centrada em dados. Porém outros equipamentos da plataforma Vectura também fazem parte dessa rede como o Vectura Softswitch (VSS) e o Vectura Signaling Server (VSI), o primeiro realizando o papel de controle de gerência dessa rede e o segundo como gateway de sinalização de canal comum número 7 (SS7- Apêndice 3).

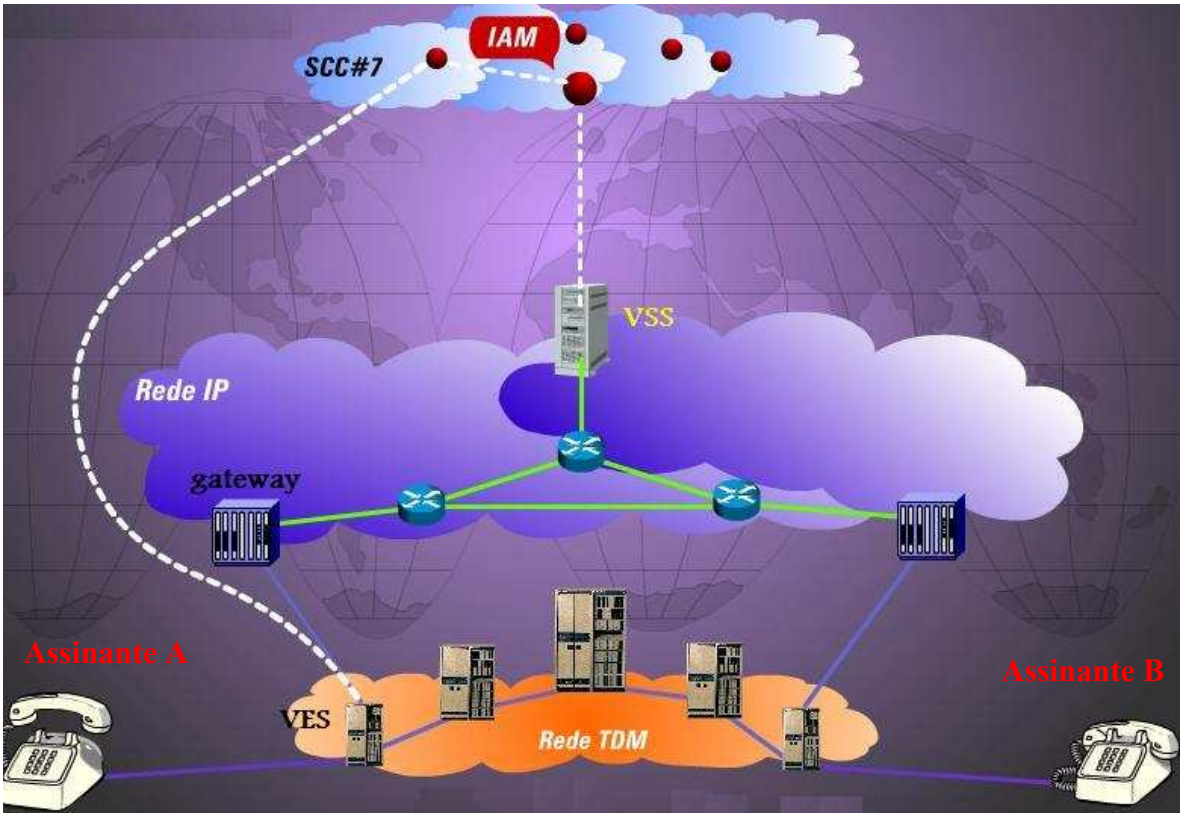


Figura 3.1: Equipamentos Vectura na NGN

A Figura 3.1 ilustra alguns dos equipamentos da plataforma Vectura se relacionando com as redes de próxima geração em uma chamada telefônica. O assinante A, conectado ao seu respectivo VES em uma rede TDM de telefonia fixa, deseja falar com o assinante B, localizado em um outro VES distante. Ao invés de usar a própria rede TDM para fazer essa ligação o VES irá encaminhar essa chamada para a rede IP, economizando assim mais canais de voz para que esses canais possam ser alocados para comunicação entre outros assinantes, aumentando assim a capacidade tráfego.

Para fazer esse encaminhamento o VES, via protocolo de canal comum SS7, envia uma mensagem para o VSS solicitando a comunicação com o assinante B do outro lado. Por sua vez, o

VSS irá enviar um comando tanto para o gateway da central do assinante A, quanto para a do assinante B. Em seguida, via protocolo SS7 o Softswitch, irá enviar uma mensagem para o VES do assinante B. Em caso de o assinante estar livre para receber ligação a central irá retornar uma mensagem para o VSS que encaminhará para a outra central avisando o respectivo *gateway* também. A partir de então a ligação é estabelecida e as informações de voz são transmitidas pela rede IP.

São os *gateways* que recebem as informações de voz das CPA's e fazem a compressão dessas informações, inserindo-as na rede IP usando protocolos do tipo TCP/IP. Essa informação não passa pelo VSS, que nesse caso faz o controle de chamada, desvio de tráfego de internet através de roteamento e tarifação por exemplo.

3.1 Perspectiva para o futuro

Existe uma forte tendência de que redes de telefonia migrem completamente para rede centrada em dados, onde os *Gateways* farão o papel do VES com os assinantes conectados diretamente nele, onde o sinal de voz será convertido em pacotes IP, já existem até telefones no mercado que já fazem essa codificação, embora não tenha a função de roteamento como os *Gateways* têm. Entretanto, estima-se que essa fase de transição dure ainda alguns anos.

4. Evolução do equipamento

Pensando em acompanhar essa tendência de migração para redes IP e também agregar mais funções e melhorias aos equipamentos da plataforma Vectura é que se vem buscando alternativas para esta evolução. Nesse âmbito entra o projeto de formatura o qual faz parte de um projeto maior, que será a evolução da placas módulo.

Essas placas são denominadas placas módulo, pois executam a função de um módulo como o módulo tronco digital (MTD- que trata os troncos digitais), MCC (módulo de canal comum) ou MMA (placa módulo de Módulo Auxiliar). Com a utilização de uma placa processadora como o padrão ETX, as placas terão sua capacidade de processamento aumentada em muitas vezes e assim poderão incorporar na mesma placa outros módulos, como o módulo de comutação (MX), que é responsável pela comutação entre os enlaces TDM.

Em uma primeira etapa de projeto essas placas equipadas com ETX deverão ser capazes de conversar com o equipamento (VES) como ele é hoje, ou seja, com módulo de comutação, sinalização (MZ) sincronismo (MS) e outros, cada qual em sua respectiva posição, sendo que essa comunicação entre placas é feita, por ser assim concebida, via interface de hardware proprietária da empresa Trópico, chamada IH4, e portanto as novas placas deverão comunicar-se também via IH4. A Figura 4.1 ilustra essa arquitetura e a Figura 4.2 ilustra como ficarão posicionadas na gaveta.

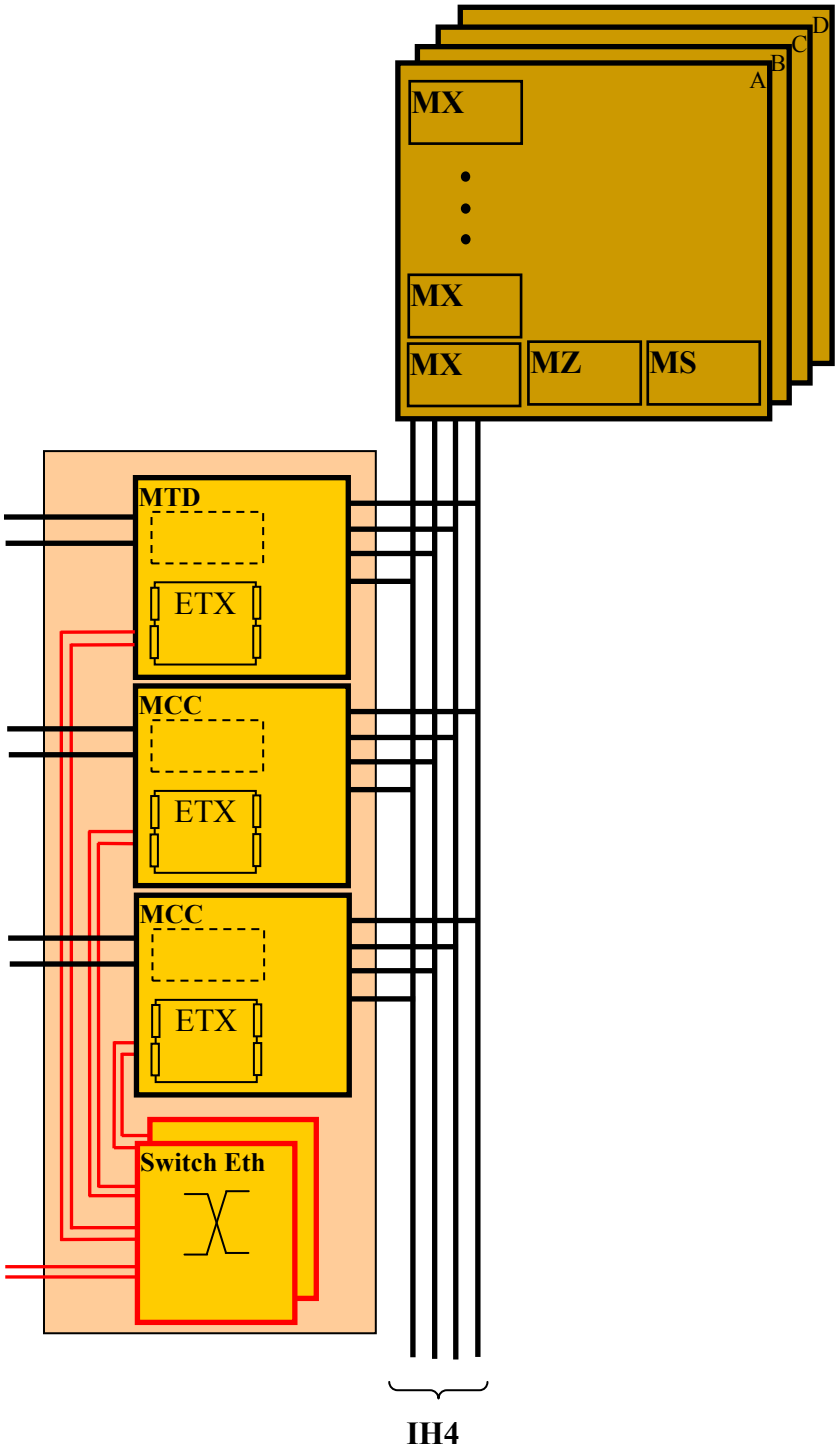


Figura 4.1: Placa módulo com ETX utilizando IH4

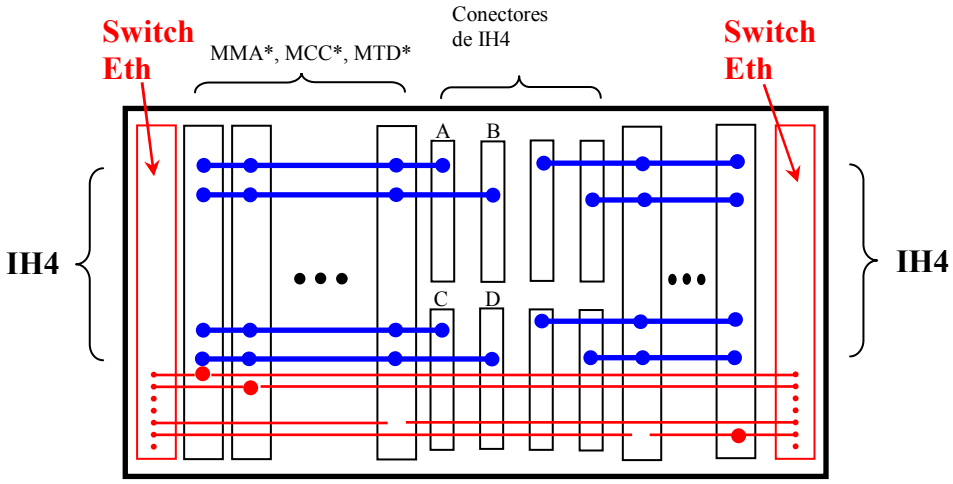


Figura 4.2: Gaveta com os módulos ETX com IH4

Como as placas processadoras padrão ETX são equipadas com conector Ethernet e também capazes de estabelecer uma comunicação por ele, podemos, como ilustra a figura 6.1 e 6.2, utilizar um Switch Ethernet para fazer a comunicação entre essas placas módulo.

Num segundo momento, as placas módulo com ETX passarão a incorporar as funções dos outros módulos, sendo assim a comutação temporal ficará interna à gaveta, não necessitando mais da interface de hardware IH4. As Figuras 4.3 e 4.4 ilustram a gaveta e a arquitetura respectivamente para esse segundo momento.

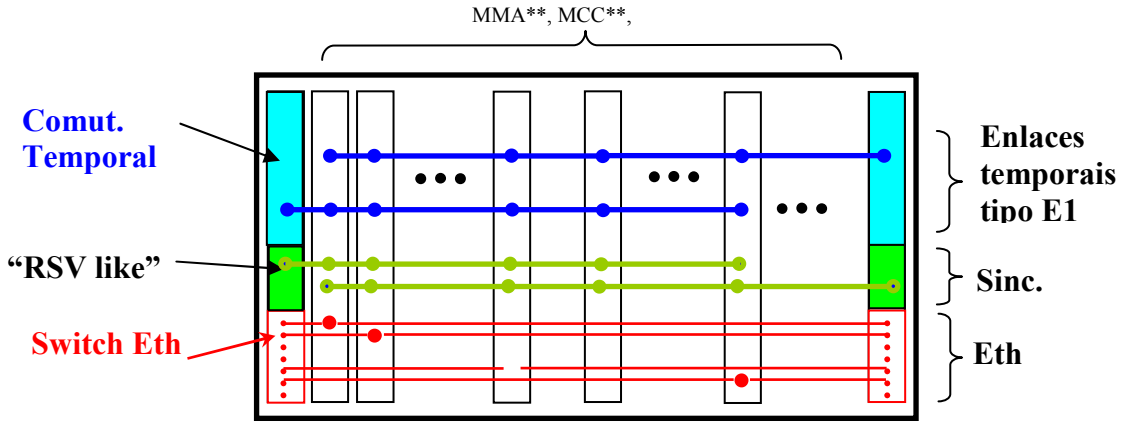


Figura 4.3: Gaveta sem IH4

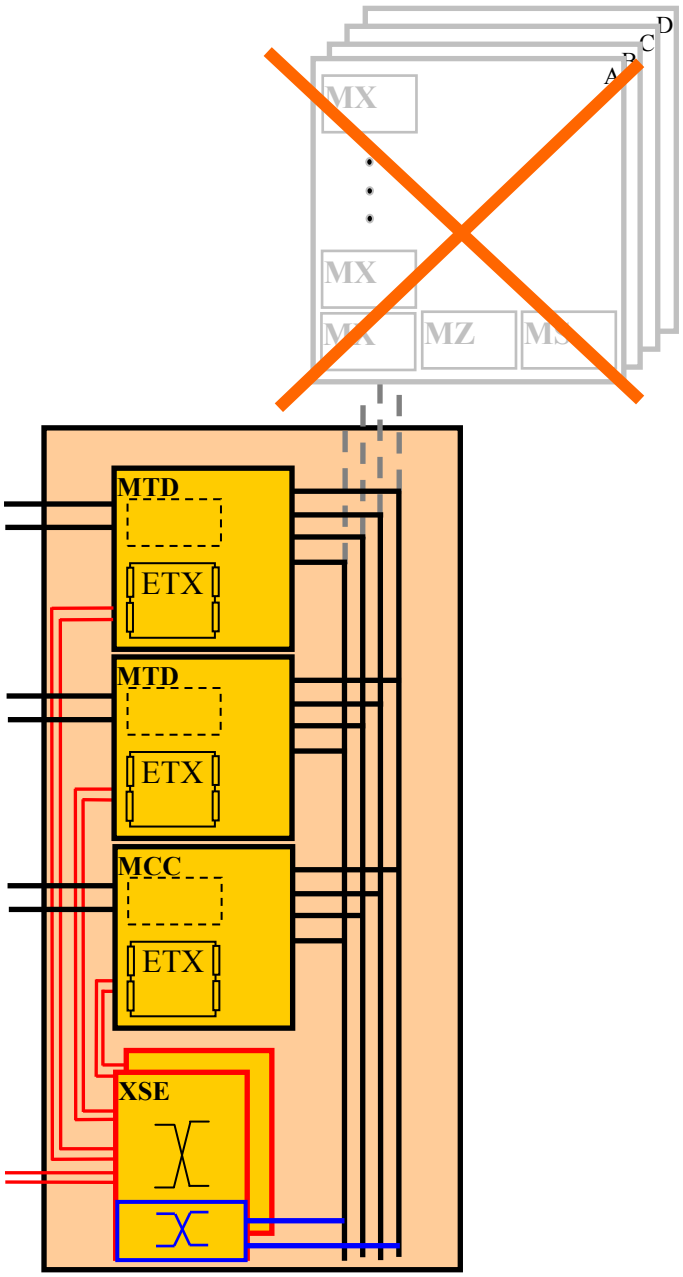


Figura 4.4: Arquitetura sem IH4

5. A interface de comunicação

O projeto de formatura faz parte do primeiro momento de evolução das placas módulo. Essa necessidade de conversar com equipamento VES, como ele é hoje, vem de uma premissa da empresa Trópico de que as evoluções nos equipamentos têm que ser na medida do possível compatíveis com o legado, ou seja, compatíveis com equipamentos já instalados e que estão em funcionamento.

Justamente por essa necessidade de compatibilidade é que as novas placas equipadas com ETX têm que apresentar uma interface de comunicação entre a interface de hardware IH4 e a placa processadora padrão ETX, além de interfaces de comunicação entre essa ultima e as placas ICC ITD, IFT que fazem as funções dos módulos que irão evoluir.

Os módulos atuais utilizam uma placa mãe CGT ou CMA, e placas filhas sendo uma placa processadora que hoje é a P586 e passará a utilizar ETX e a outra placa filha dependendo do módulo em questão poderá ser, para MCC a ICC+IFT, para MTD a ITD+IFT e para MMA só IF. IFT é o circuito responsável por gerar as demais tensões de alimentação além da fornecida pela central que serão usadas na placa - como se trata de um circuito pequeno ele é montado na mesma placa que a ICC e ITD.

A Figura 5.1 ilustra a placa como ela é hoje e a Figura 7.2 sua evolução usando placa processadora ETX.

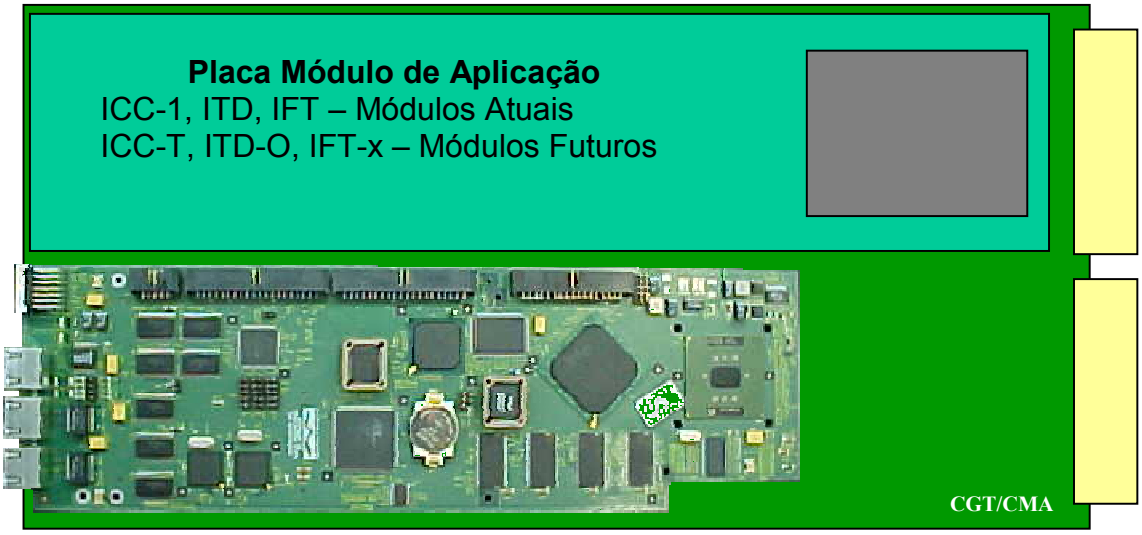


Figura 5.1: Placa Módulo com placa processadora atual

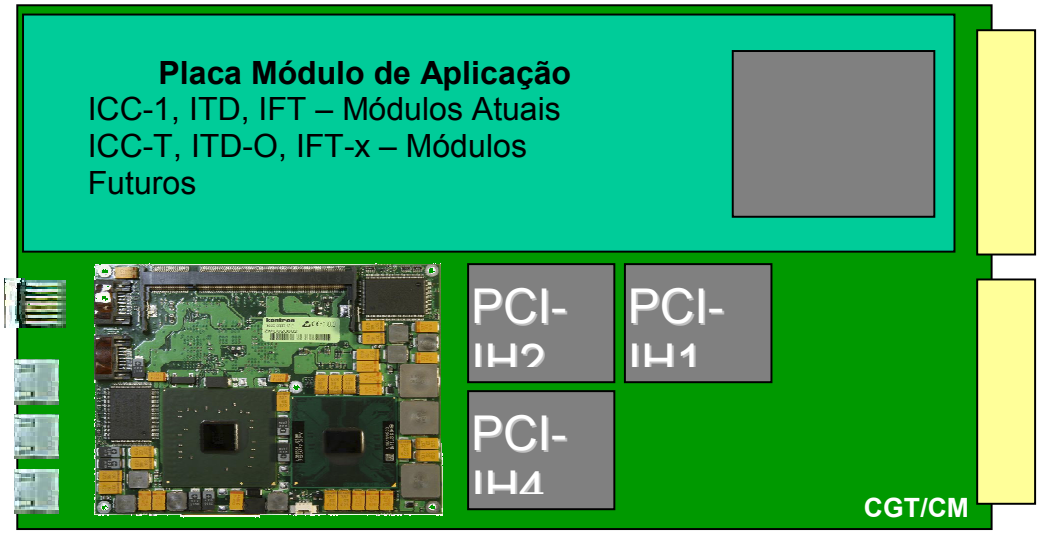


Figura 5.2: Placa Módulo com placa padrão ETX

5.1 Interface Usando VHDL

A interface de comunicação referida acima é feita utilizando dispositivos lógicos programáveis como EPLD ou FPGA, e para programar tais dispositivos é utilizado a linguagem de programação VHDL (apêndice 2).

Uma parte dessa interface que já foi feita e testada, mas só no nível de simulação, uma vez que precisa de mais partes ou da interface mais completa para ser testada com o dispositivo programado, é a implementação do comutador digital MT8980D em VHDL que apresenta o diagrama funcional em blocos esquematizado na Figura 5.3.

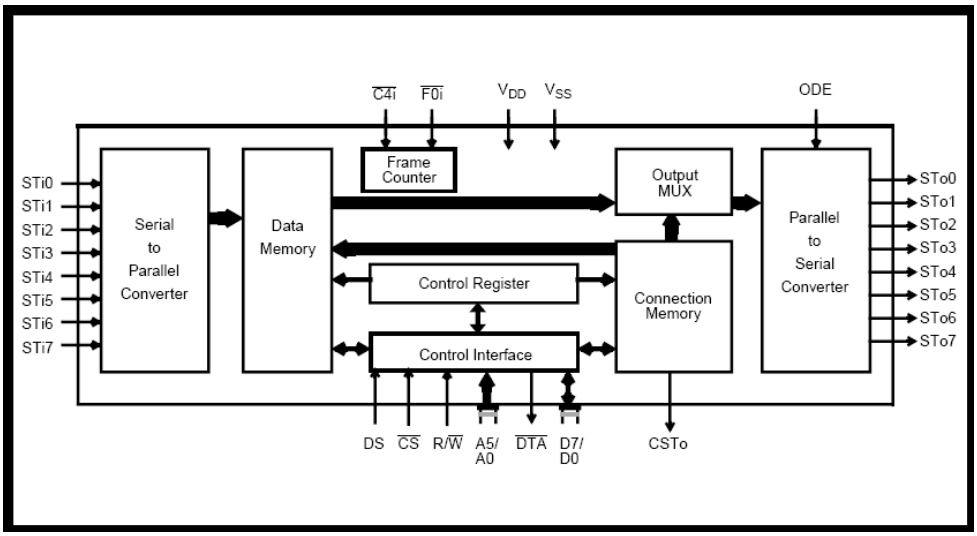


Figura 5.3: Digrama funcional em blocos do MT8980D

Esse dispositivo resumidamente recebe oito entradas cada uma com 32 canais de voz ou dados multiplexados no tempo e pode comutar cada um desses canais com qualquer canal da saída que também são multiplexados no tempo e dispostos em oito saídas seriais. Além disso, o dispositivo também conta com interface de controle e barramentos de escrita e leitura.

A implementação desse dispositivo em VHDL à primeira vista pode parecer complexo, porém se a dividirmos em sub-partes, como o conversor paralelo serial, então a implementação torna-se factível. Como se trata de uma propriedade intelectual da empresa Trópico o código em VHDL do programa não constará neste documento.

Entretanto, como exemplo, o conversor série – paralelo ou paralelo - série pode ser implementado facilmente utilizando o conceito de registrador de deslocamento usando para isso estruturas de CASE e WHEN contidas no Apêndice 2.

5.2 Considerações finais do projeto (TCC)

Usando placas ETX de alto desempenho e usando a mesma Base-Board, podem-se implementar Módulos MPG (Módulo Processador Genérico) que se conectam a outros módulos unicamente através de interfaces Ethernet. Como nesses módulos a IH4 não será usada, os circuitos de interface de IH4 poderão ser subequipados na Base-Board. Também se pode imaginar novos tipos de módulos com capacidade de MPG, inclusive rodando BI's(blocos de implementação) que são softwares, e nesse caso programados em linguagem C ou C++ que alem de Ethernet, também possuam IH4.

Os principais objetivos da utilização das placas ETX são:

- Permitir que os processadores que controlam os produtos da plataforma Vectura possam acompanhar mais rapidamente a evolução tecnológica das placas processadoras comerciais;
- Permitir que se disponha de uma gama de placas processadoras, desde placas mais baratas e de menor capacidade, para atender módulos com baixa demanda de processamento, até placas processadoras de alto poder de processamento, para módulos que assim o exijam;
- Permitir que, sobre uma base constituída pelo sistema operacional Linux, possam rodar tanto BIs em linguagem CHILL (principalmente BIs de legado) como novos BIs em linguagem C, C++;

- Permitir que a recarga de software dos processadores possa ser realizada a partir de memória Flash, podendo ser atualizada via rede interna do equipamento Vectura a partir de uma memória de massa somente nos casos de atualização de software;
- Minimizar o número de cabos internos à gaveta de Placas Módulo. Isto é possível graças aos Switches Ethernet incorporados na gaveta.

Considerando que a modularidade é importante no projeto de um equipamento de controle e comutação, os diversos tipos de placas módulo possuirão uma base de hardware e software comum e uma gaveta poderá comportar diferentes tipos de placas módulos.

Os altos custos inerentes aos processos de desenvolvimento e industrialização não permitem que se tenha uma quantidade muito grande de módulos específicos diferentes. Por outro lado, atende à necessidade no mercado de sistemas mais flexíveis, capazes de atender não somente as funções atualmente demandadas como também demandas futuras.

Portanto, outra premissa que também foi estabelecida é que a arquitetura do projeto tem que estar preparada para acompanhar e adaptar-se às evoluções e também ser compatível, se possível, com as arquiteturas anteriores. Este último requisito é importante pois permite que a fabricação de módulos de versões anteriores seja interrompida sem deixar de atender a necessidade do fornecimento de módulos sobressalentes.

Com relação à operação e manutenção, uma série de requisitos é atendida como: operação local ou remota da central; supervisão automática de falhas pelo próprio equipamento; detecção e diagnóstico de defeitos; testes e medições de desempenho, etc.

6. Equipamento Vectura

6.1. Arquitetura

O aspecto mais importante na arquitetura de um sistema de controle e comutação é a sua própria estrutura interna de controle. A Plataforma Vectura utiliza uma arquitetura com estrutura de controle completamente distribuída entre vários processadores de pequeno e médio porte e descentralizada, diferentemente de outras arquiteturas que utilizam menos processadores de alto desempenho realizando as funções de forma mais centralizada. Entre outras vantagens, evita-se assim que o custo com processadores não seja muito elevado, já que dessa forma esses últimos não precisam ter seus requisitos de confiabilidade tão elevada. A complexidade do software também se torna menor, uma vez que cada processador desempenha funções específicas associadas apenas ao grupo de órgãos controlado. Com a descentralização do controle a confiabilidade global do sistema torna-se mais alta devido à ocorrência de falhas só afetarem um pequeno número de dispositivos. Essa confiabilidade é também otimizada porque cada função básica é realizada de forma distribuída por uma quantidade grande de módulos, cada um controlado por um processador.

Cada módulo é constituído por um processador e, eventualmente, mais algum hardware controlado. Conforme mostrado na Figura 6.1, os módulos são interconectados através de dois sistemas de comunicação entre processadores, um baseado em protocolo Ethernet e outro de alta velocidade em que a maior parte do protocolo é realizada em Hardware, otimizando os seus tempos de tratamento. Os processadores que se conectam com o restante do sistema através de interfaces Ethernet são implementados exclusivamente através de servidores comerciais. Os switches Ethernet também são comerciais. Todas as outras partes do sistema são desenvolvimentos proprietários da Trópico.

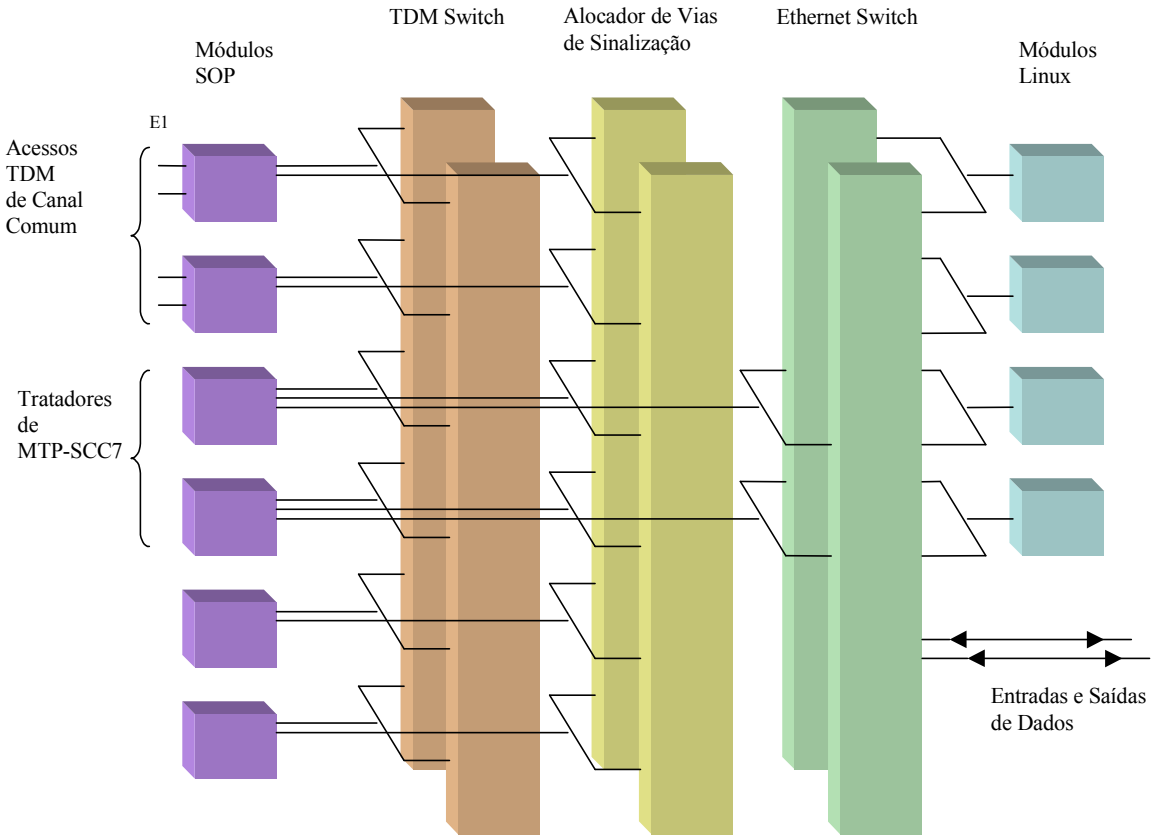


Figura 6.1: Arquitetura da Plataforma Vectura

Os equipamentos da Plataforma Vectura são fisicamente constituídos de módulos construtivos básicos: Módulo de Comutação (MX), Módulo de sinalização (MZ), Módulo de Sincronismo (MS), Módulo de operação (MO), Módulo de Canal Comum (MC), Módulo de Terminais (MT) e Módulo Auxiliar (MA). O conjunto de módulos MX, MZ e MS é denominado de conjunto de Módulos de Interconexão, pois toda comunicação interna tem origem ou passagem por um desses módulos. Esses módulos são implementados em até quatro planos redundantes: De um a oito MX constituem um Plano de Comutação; um ou dois MZ's constituem um Plano de Sinalização Interna e um MS constitui um Plano de Sincronismo. Até 64 interfaces internas denominadas Interfaces Intermodulares IH4, ou simplesmente IH4 conectam os módulos de interconexão de cada plano com todos os outros módulos do equipamento. Cada um desses módulos do equipamento se conecta a uma IH4 de cada plano. Até 20 módulos podem ser conectados Cada IH4 em barramento. A topologia dessa conexão intermodular é portanto "barramento estrela".

As placas módulo implementam os seguintes tipos de módulos: MA, MC e MT. Atualmente elas usam placas processadoras proprietárias. O objetivo do projeto considerado neste TCC é a evolução dessas placas módulo de modo a passarem a usar placas processadoras comerciais tipo ETX.

Outros módulos relacionados com esse projeto são os módulos de interconexão, ou seja, os módulos MX, MZ e MS, pois são os que fornecem as interfaces intermodulares IH4 através das quais as placas módulo se comunicam com o restante do equipamento. A seguir o funcionamento desses módulos será explicado.

Os MX's, conforme explicado anteriormente, são responsáveis pela comutação entre os enlaces TDM intermodulares que conectam os módulos com canais em 64 kbit/s (MT e MC). Cada MX pode comutar até 16 enlaces TDM - Time Division Multiplex bidirecionais, cada um dos quais transporta 32 canais em 64 kbit/s (por canal), com acessibilidade plena a qualquer número de terminais associados (a limitação é dada por tráfego e não por posições físicas no equipamento). Como indicado, até oito MX podem ser utilizados em cada plano de comutação, de acordo com os requisitos de tráfego e de confiabilidade. Com quatro planos, cada um com oito MX's o equipamento é capaz de tratar até 12600 E.

O MO faz o interfaceamento com um conjunto de periféricos de entrada e saída, cobrindo as quantidades e tipos de periféricos que atendem os requisitos das empresas operadoras. Além disso incorporam os softwares de gerência do equipamento.

MT é o módulo que liga a central ao meio externo através de linhas de terminais, tais como linhas de assinantes ou troncos digitais.

MC é o módulo de Canal Comum SS7 que realiza os níveis 2 e 3 da Sinalização por Canal Comum SS7.

MZ é responsável pelo controle de comunicação entre os processadores, ou seja, é ele que arbitra a rede interna de processadores para evitar colisão, ele verifica se tem via livre e aloca-a para a comunicação, quando o processador termina a comunicação ele retira a requisição da via avisando assim o MZ que a mesma já está liberada.

MS é responsável pelo sincronismo, gerando o clock que é enviado a todos os processadores da central, através de vários recursos e caminhos redundantes para propiciar uma alta tolerância às falhas.

MA ou Módulos Auxiliares são módulos que não controlam nenhum hardware associado, possuindo unicamente funções de processamento.

6.2 Aplicações e Configurações

Os equipamentos da plataforma Vectura, VSS (Vectura Softswitch), VSI (Vectura Signaling Server) e VES (Vectura Edge Switch) podem ser configurados de acordo com sua aplicação como central local (que contém assinantes), central trânsito (que faz a conexão entre duas centrais), central tandem (central local e trânsito juntas), SoftSwitch de uma rede NGN (ver descrição na Seção 2), PTS (ponto de transferência de sinalização de uma rede SS7 e Signaling Server. Algumas combinações de mais de uma aplicação no mesmo equipamento também são possíveis. Devido à sua distribuição e descentralização das funções de controle e sua modularidade, o equipamento tem uma alta flexibilidade com relação a sua configuração. Por exemplo, no caso da aplicação central telefônica, dependendo da configuração pode se obter uma central só com 16 assinantes até capacidades superiores a 100.000 assinantes, variando para isso as quantidades de módulos terminais, de comutação, de canal comum, etc.

Por outro lado na aplicação de central telefônica, o VES pode também operar como central satélite, utilizando numeração dependente de centrais digitais de maior porte em zonas urbanas. Nesta aplicação é utilizada uma série de vantagens adicionais em relação a um estágio remoto:

- Comutação interna (com tarifação independente), para chamadas locais, aliviando o tráfego na central mãe;
- Distribuição descentralizada de tons ou anúncios gravados;
- Facilidades especiais de operação e manutenção;
- Interligação com outras centrais para chamadas de saída, dando uma flexibilidade maior para encaminhamento;
- Utilização da Unidade Distante (UD), possibilitando otimização de rede e equipamentos.

6.3 Capacidade

A Tabela 6.1 contém, a título de exemplo, os valores típicos de capacidades da aplicação VES. Em sua configuração máxima o equipamento pode tratar até 12.600 E, mantida ainda uma capacidade extra de tratamento de sobrecarga.

Como a central trabalha com conceito de redundância, temos nela 4 planos de comutação funcionando mas operando abaixo da capacidade máxima. Isso garante que se algum plano falhar a central continuará com seu funcionamento normal e todas as suas funções e sem nenhuma redução

de tráfego. Se um segundo plano falhar a central ainda funciona com suas funções integralmente mantidas porém com redução do tráfego (degradação suave).

Capacidade máxima de tráfego (4 planos de comutação)	12600 E
Número de terminais de assinantes	10.000
Número de junções (entrada e saída)	10.000

Tabela 6.1: Valores típicos de capacidade da CPA-T

6.4 Características principais do VES (TROPICO RA)

A seguir estão as principais características :

- Controle por programa armazenado (CPA)
- Central com sistema digital
- Comutação digital com conversão A/D de sinais (aplicação VES)
- Controle distribuído e descentralizado
- Redundância ativa nas funções de comutação, sinalização e distribuição de sincronismo
- Geração de sincronismo triplicada
- Partição de carga sem troca de mensagem de atualização entre processadores (OM carrega cada processador)
- Degradação suave em presença de falhas
- Estrutura de voz e sinalização independentes (aplicação VES)
- Padronização de interfaces
- Ampliação sem precisar para a central

7. Conclusão

O projeto de formatura, desenvolvimento de uma interface de comunicação entre as placas módulo e a placa processadora padrão ETX, faz parte de um projeto maior que é a evolução das placas módulos bem como a gaveta na qual são inseridas e as interfaces de comunicação entre elas. Por se tratar de uma evolução e não de uma exigência de algum cliente, esse projeto ainda está em andamento e com previsão de término para janeiro de 2009.

Entretanto, apesar de não estar concluída essa evolução, pequenas partes como a implementação do MT8980D já foi concluída e com êxito, aguardando somente a próxima etapa. Além disso o projeto de formatura serviu também de aprendizado com todas as etapas envolvidas em projeto de desenvolvimento, como por exemplo as definições de funções que o equipamento vai ter, o custos envolvidos com o projeto, cronograma, divisão do projeto em subprojetos como o caso do próprio tema do TCC e outros.

Para ambientalizar e contextualizar o projeto, foi necessário o aprendizado de conceitos, como redes de próxima geração, protocolo de sinalização por canal comum número 7 e também conhecimento dos equipamentos da Plataforma Vectura e como se relacionam com esses conceitos.

8. Referências Bibliográficas

[1] Tutorial SS7**http://resource.intel.com/telecom/support/ss7/SS7tutorial/tutorial.html**

[2] Tutorial ETX www.etx.org

[3] Borges, Engº Aderbal Alves , *TROPICO R – Central local/Tandem digital do sistema Trópico 1991*

[4] Filho, Helio J. Malavazi, *Livro Ouro TROPICO RA, 2000*

[5] AVNET electronics marketing, *Introduction to VHDL for FPGA- featuring ISE 9.2i Tools 2008*

[6] Diaz, Victor Alfonso Valenzuela, *A evolução de uma arquitetura centrada em voz para (também) centrada em dados – XXVI* SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES, set 2008

[7] Artigas, Fernando Cavalheiro de Oliveira, *Redes NGN: Desafios e Serviços para Convergência* – II Concurso Teleco de Trabalhos de Conclusão de Curso (TCC) 2006

Apêndice 1 – Erlang

Este apêndice doe baseado em uma apresentação tutorial do Engenheiro Eduardo Tude sobre os conceitos básicos sobre erlang e tráfego telefônico. O erlang é utilizado para dimensionamento de centrais telefônicas. Com centrais adequadamente dimensionadas, o congestionamento em um sistema telefônico passa a depender basicamente do número de troncos entre as centrais.

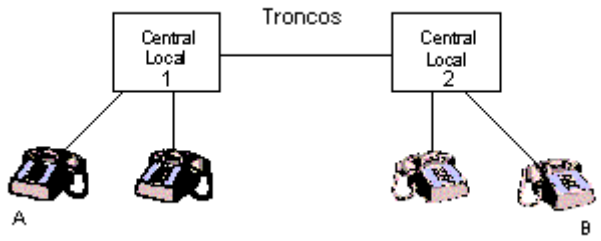


Figura A1.1: Exemplo de ligação de duas centrais

Qual o número de troncos necessários para garantir que as chamadas bloqueadas devido ao número insuficiente de troncos entre 1 e 2, seja inferior a 5% em um período de maior movimento?

Para responder a esta questão apresenta-se inicialmente como se caracteriza tráfego telefônico, para em seguida apresentar a fórmula desenvolvida por Erlang para este dimensionamento.

Caracterização do Tráfego Telefônico

A Figura A1.2 a seguir exemplifica como ocorre a ocupação dos troncos entres as centrais A e B em função das chamadas.

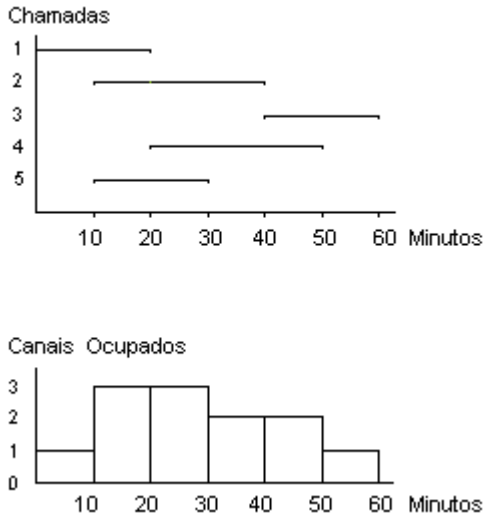


Figura A1.2: Gráficos canais ocupados em função do tempo

A intensidade de tráfego em um sistema telefônico pode ser definida como o somatório dos tempos das chamadas telefônicas (ocupação dos canais telefônicos) em um determinado período de tempo, normalmente uma hora.

Erlang é uma unidade de medida de intensidade de tráfego telefônico para um intervalo de uma hora.

Exemplo: Na Figura A1.2 o tempo total de ocupação de canais é de 120 minutos, ou 2 horas. A intensidade de tráfego é de 2 erlangs.

Em um sistema telefônico as chamadas se originam aleatoriamente e independentemente uma das outras. O tráfego telefônico varia com:

- A hora do dia.
- O dia da semana.
- A semana do ano.

Figura A1.3: Gráfico Erlang em função das horas do dia

Para dimensionar um sistema é preciso estabelecer o número médio de chamadas e a duração média de cada chamada na Hora de Maior Movimento (HMM). Com estes dados pode-se calcular a intensidade de tráfego para a qual o sistema será dimensionado. Uma vez implantado, o desempenho do sistema pode ser acompanhado através de medições periódicas.

Apresenta-se a seguir a Fórmula de Erlang que permite o dimensionamento do número de troncos em um sistema telefônico.

Esta fórmula, conhecida como Fórmula de Erlang B, é:

$$P_b = \frac{\frac{A^N}{N!}}{\sum_{i=0}^{N} \frac{A^i}{i!}}$$

Onde:

A = Tráfego Oferecido

N = Número de Canais para escoar o tráfego

Pb = Probabilidade de Bloqueio.

A fórmula de Erlang apresentada, também conhecida como Erlang B, foi desenvolvida utilizando algumas hipóteses simplificadoras. Uma delas é que qualquer chamada bloqueada é perdida, não existindo uma nova tentativa de discagem.

Apêndice 2 - VHDL

A2.1 Introdução

VHDL foi desenvolvido pelo Ministério de Defesa dos EUA para ser usado sobre VHSIC (Very High Speed Integrated Circuit) e VHDL significa **V**hsic **H**ardware **D**escription **L**anguage. Portanto VHDL trata-se de uma linguagem de programação de hardware, usada para descrever sistemas e pode ser usada para projetar e simular circuitos digitais. Ele foi adotado pelo IEEE em 1987 como o padrão IEEE 1076-1987.

A2.1.1 Por que usar VHDL?

A linguagem de programação VHDL é bastante simples e estruturada de forma que podemos citar as seguintes vantagens:

- Capacidade de desenvolvimentos mais complexos do que os esquemáticos;
- Não está vinculado a nenhum fabricante;
- É uma linguagem padronizada;
- Capacidade de reutilização de projetos e/ou partes de projeto;
- Capacidade de abstração de projetos.

A2.3 Par ENTITY (Entidade) / ARCHITETURE (arquitetura)

ENTITY é a base de todo projeto em vhdl e sempre vem acompanhado de ARCHITECTURE. Uma entidade pode ter mais de uma arquitetura, mas uma arquitetura só pode ter uma entidade associada. Entidades definem a interface, isto é os I/Os, para o projeto. A arquitetura define a função do projeto. Portanto a entidade seria o bloco com as entradas e saídas e a arquitetura as funções dentro do bloco.

A seguir temos os *templates* da entidade e da arquitetura:

```
entity entity_name is
port (port_list);
end entity_name;

-- exemplo

entity DFLOP is
Port (D,clk : in std_logic;
        Q :out std_logic
         );
end DFLOP;
```

```
architecture architecture_name of entity_name is
    declaration section
begin
    concurrent statements
end architecture_name;
```

- Detalhes da entidade

No *Port* é onde especificamente declaramos as entradas e saídas sendo da seguinte forma: (*Port_names* : MODE type), ou seja, o MODE pode ser **in, out, inout** ou **buffer**, onde o **buffer** é usado quando queremos que uma saída realimente o próprio circuito.

- Detalhes da arquitetura

Em *declaration section,* são declarados os sinais, constantes e componentes localizados na arquitetura. Em *concurrent statements,* é definido o circuito propriamente dito.

A2.4 Operações lógicas e comentários.

O VHDL predefine as seguintes operações lógicas:

⇒ NOT – mais alta precedência

⇒ AND

⇒ NAND

⇒ OR

⇒ NOR

⇒ XOR

⇒ XNOR

As outras operações lógicas não têm precedência entre elas. Se houver dois ou mais operadores diferentes em uma equação, a ordem de precedência é da esquerda para a direita.

Para se fazer comentários no código VHDL usam-se duplo sinal de menos (--), depois dele e na mesma linha é que estará o comentário. O comentário só funciona para uma linha, portanto caso deseje-se comentar mais de uma linha é necessário inserir (--) em todas as linhas.

```
-- Exemplo de uma porta AND de duas entradas
library IEEE;
use IEEE.std_logic_1164.all;

entity and2vhdl is
port (
```

```
        In_a : in std_logic;
        In_b : in std_logic;
        Out_c : out std_logic
        );
end and2vhdl;

architecture and2vhdl_arch of and2vhdl is
begin
        Out_c <= In_a and In_b;
end and2vhdl_arch;
```

A2.5 COMPONENT (componente)

Esse bloco é usado dentro da arquitetura antes do **begin**, componente tem que ser declarado antes de ser usado. Componentes são usados para apontar sínteses de projetos em outras entidades. Essas entidades podem estar no diretório do projeto em si ou em bibliotecas do software de programação.

```
Component component_name
generic ( generic_list); -- Opcional
port ( port_list);
end component;
```

Port list contém os sinais, modos e tipos, *generic list* é usado para passar a informação para o chamado da entidade. É importante dizer que a lista de entradas e saídas do *port list* tem ser declarada igualmente e na mesma ordem na hora de instanciar no projeto.

A seguir temos um exemplo de componente:

```
entity MAP is
        port (A, B, CLK1 : in std_logic;
                Q1 : out std_logic);
end MAP;
architecture MAP_arch of MAP is
        signal D1 : std_ logic;
        component Comb_logic
        port ( X,Y : in std_logic;
                O : out std_logic);
        end component;
        component DFF
        port (D,clk : in std_logic;
                Q : out std_logic);
        end component;
begin
…….
```

A seguir temos um *template* para instanciar um componente dentro de um projeto:

```
<instance_name> : <component_name>
```

```
generic map (
  <generic_name> => <value>,
  <other generics>...
)
port map (
  <port_name> => <signal_name>,
  <other ports>...
);
```

O **port map** é usado para mapear os sinais em um componente, pode mapear a posição ou o nome associado. Também é usado em conjunto com um componente que foi instanciado a parte. Um exemplo disso está abaixo.

```
component COMB2 (a,b : in std_logic;
      out1 : out std_logic);
end component;
signal in1,in2,o1: std_logic;
begin
      U1:COMB2 port map (in1, in2, o1); -- ordem dependente, mapeamento por posição

      U2:COMB2 port map (a => in1, out1 => o1, -- não depende da ordem
                         b => in2); --associação por nome
```

A2.6 PROCESS(processo)/ IF CASE WHEN

Estruturas de IF, WHEN e CASE usadas em outras linguagens de programação também são usadas no código VHDL, a seguir temos os *templates* das três estruturas.

IF:

```
 if <condition> then  <statement>
elsif <condition> then  <statement>
else   <statement>
end if;
```

WHEN:

```
 <name> <= <expression> when <condition> else
        <expression> when <condition> else
        <expression>;
```

CASE COM 3 BITS DE ENTRADA:

```
case (<3-bit select>) is
  when "000" => <statement>;
  when "001" => <statement>;
  when "010" => <statement>;
  when "011" => <statement>;
  when "100" => <statement>;
  when "101" => <statement>;
  when "110" => <statement>;
  when "111" => <statement>;
  when others => <statement>;
end case;
```

Essas estruturas são usadas normalmente com o PROCESS, pois no VHDL as instruções não são executadas linha por linha, como em C ou C++ por exemplo, e sim em paralelo, ou seja, como se tudo fosse executado de uma única vez. Portanto para executar expressões que dependam de outra temos que usar esse tipo de estrutura.

```
Label:-- label opcional
process (<sinais>)
-- declaração de processos locais
begin
-- declarações seqüenciais
-- opcional wait declarações
end process;
```

Processos existem dentro da arquitetura, podem ter variáveis locais, contêm declarações seqüenciais, têm lista de sensibilidade ou declarações de espera, mas nunca os dois. Processos são executados somente quando o sinal ou sinais da lista de sensibilidade muda(m) de valor, logo eles podem ser usados para fazer circuitos dependentes de *clock*.

A seguir temos exemplos de códigos usando PROCESS junto com IF,WHEN e CASE:

Um MUX de quatro entradas usando IF

```
library IEEE;
use IEEE.std_logic_1164.all;
entity MUX is
        port ( MUX_IN1, MUX_IN2, MUX_IN3, MUX_IN4 : in std_logic;
                                    SEL : in std_logic_vector (1 downto 0);
                            MUX_OUT : out std_logic);
end MUX;
architecture IF_MUX_arch of MUX is
```

```
begin
        process (SEL, MUX_IN1, MUX_IN2, MUX_IN3, MUX_IN4)
        begin
                if SEL = "00" then
                        MUX_OUT <= MUX_IN1;
                elsif SEL = "01" then
                        MUX_OUT <= MUX_IN2;
                elsif SEL = "10" then
                        MUX_OUT <= MUX_IN3;
                else
                        MUX_OUT <= MUX_IN4;
                end if;
        end process;
end IF_MUX_arch;
```

Agora usando WHEN.

```
entity MUX is
port (
        MUX_IN1,MUX_IN2 : in std_logic;
        MUX_IN3,MUX_IN4 : in std_logic;
        SEL             : in std_logic_vector(1 downto 0);
        MUX_OUT         : out std_logic
        );
end MUX;
architecture WHEN_MUX_arch of MUX is
        begin
                MUX_OUT <= MUX_IN1 when SEL="00" else
                           MUX_IN2 when SEL="01" else
                           MUX_IN3 when SEL="10" else
                           MUX_IN4;
end WHEN_MUX_arch;
```

O mesmo MUX com CASE

```
architecture CASE_MUX_arch of MUX is
begin
        process (MUX_IN1, MUX_IN2, MUX_IN3, MUX_IN4, SEL)
          begin
            case sel is
                when "00" =>
                        MUX_OUT <= MUX_IN1;
                when "01" =>
                        MUX_OUT <= MUX_IN2;
                when "10" =>
                        MUX_OUT <= MUX_IN3;
                when others =>
                        MUX_OUT <= MUX_IN4;
                end case;
```

```
        end process;
end CASE_MUX_arch;
```

A2.7 Tipos de dados e biblioteca 1164

Um conjunto de possíveis valores ordenados define um tipo particular de dado, VHDL é uma linguagem fortemente "tipada", ou seja, todas as variáveis têm que ser atribuídas a um tipo. A seguir temos exemplos desses tipos de dado:

- Boolean FALSE, TRUE
- Bit ('0','1')
- bit_vector("101010")
- Integers: range -(2^31-1) to 2^31-1
- Floating real: -1.E38 to 1.0E38
- Time
- Character
- String
- Enumerated (User defined)
- Records, file & access types (Used in Simulation only)
- Std_logic e Std_ulogic

STD_LOGIC e STD_ULOGIC são tipos de variáveis que foram incorporadas somente com a biblioteca 1164 (IEEE.std_logic_1164**.all**) STD_LOGIC é um subtipo do STD_ULOGIC, onde esse ultimo é definido da seguinte forma:

```
TYPE std_ulogic IS ('U', -- Uninitialized
                    'X', -- Forcing Unknown
                    '0', -- Forcing 0
                    '1', -- Forcing 1
                    'Z', -- High Impedance
                    'W', -- Weak Unknown
                    'L', -- Weak 0
                    'H', -- Weak 1
                    '-' -- Don't care
                    );
```

Como podemos ver esse tipo de variável pode assumir o valor 0, 1, alta impedância, desconhecido, etc. Podemos declarar vetores usando esse tipo de variável da seguinte forma:

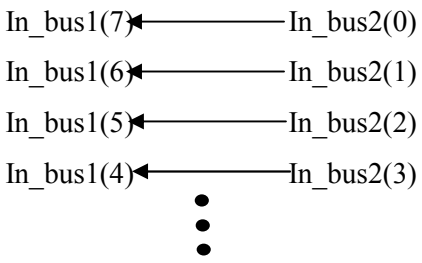
```
library IEEE;
use IEEE.std_logic_1164.all;
entity busses is
port (
        In_bus1, In_bus2 : in std_logic_vector (7 downto 0);
```

```
        In_bus3         : in std_logic_vector (0 to 7);
        Out_bus         : out std_logic_vector (7 downto 0)
        );
end busses;
```

Os vetores são sempre preenchidos da esquerda para a direita, e os índices são atribuídos ascendentemente ou descendentemente, dependendo da palavra chave **to** ou **downto**. Para exemplificar isso temos a seguinte situação:

```
In_bus1, In_bus2    : in std_logic_vector (7 downto 0);
In_bus3             : in std_logic_vector (0 to 7);
Out_bus             : out std_logic_vector (7 downto 0)
……
In_bus1 <=“10110010”; -- IN_BUS1(7)= 1, IN_BUS1(0) = 0
In_bus2(3) < =‘1’ ; -- IN_BUS2 = (U,U,U,U,1,U,U,U,)
In_bus2(6 downto 4) < =“101” ; -- IN_BUS2 = (U,1,0,1,U,U,U,U,)
In_bus3 <=“10110010”; -- IN_BUS3(7)= 0, IN_BUS3(0) = 1
```

Ou seja, se tivermos a operação In_bus1<= In_bus2, os índices ficariam dispostos assim



A2.8 Sinal e máquina de estado

Quando temos a saída de um bloco construtivo indo para a entrada de outro bloco construtivo dentro do mesmo projeto é necessário que se declare essa ligação como um sinal. A declaração do sinal segue de forma análoga à declaração de um dado e I/Os. Sempre atentando para o fato de que o sinal deve ser do mesmo tipo e do mesmo tamanho da entrada e saída com a qual é conectada. Por exemplo, se temos a saída de um contador de 8 bits indo para um barramento de dados de uma memória, onde **out_cont8** e **mem_dado8** foram declarados dentro de suas respectivas entidade como STD_LOGIC_VECTOR(7 **DOWNTO** 0), o sinal **cont_memo8** terá que ser declarado depois de **architeture** e antes do **begin** como:

```
signal cont_memo8 : std_logic_vector (7 downto 0);
```

Em muitas situações iremos nos deparar com situações de projeto onde fica muito mais fácil desenvolver o código VHDL a partir de uma máquina de estado. Para isso consideremos a seguinte máquina de estado:

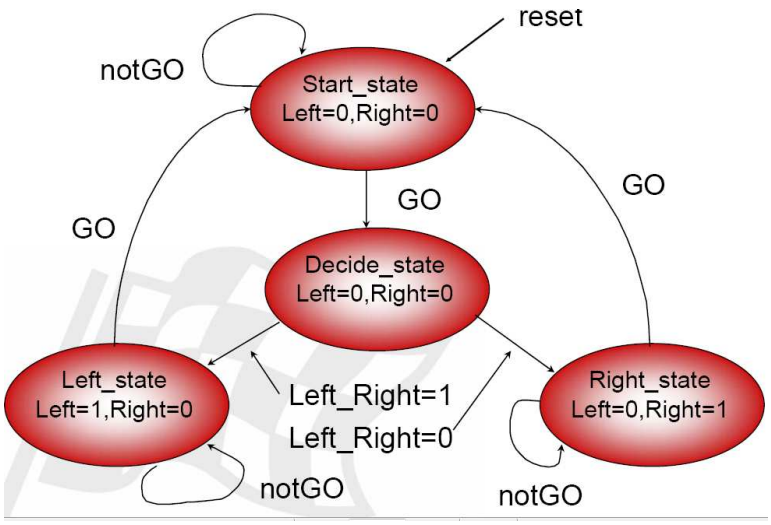


Figura A2.1: Máquina de Estado

O caminho para se codificar o VHDL de uma máquina de estado como essa é usar duas variáveis de estado, um para o estado presente e outra para o estado futuro e escrever dois processos sendo:

- Processo 1

Criação do próximo estado a partir do estado presente de entradas de controle;

Atribuição das saídas de acordo com o estado presente.

- Processo 2

Reset assíncrono, habilitadores opcionais;

Clock de próximo estado dentro do estado atual.

Processo 1

Para criar o próximo estado é necessário criar um processo sensível ao estado presente e a todas as entradas de sinal de controle. E dentro do processo fazer a declaração de CASE que tenha um WHEN para cada estado do diagrama de estados.

Dentro de cada WHEN, defina os sinais de saída para aquele estado e escreva um bloco IF que represente cada um dos possíveis caminhos que deixam esse estado e atribua o próximo estado para cada um desses caminhos.

O código VHDL da máquina de estado da figura7.2 pode ser feito da seguinte forma:

```
-- Implementação da máquina de estado
library IEEE;
```

```
        use IEEE.std_logic_1164.all;
        entity state1 is
          port (
                LEFT_RIGHT, GO      : in std_logic;
                                reset     : in std_logic;
                                clk       : in std_logic;
                                RIGHT : out std_logic;
                                LEFT : out std_logic);
        end state1;
        architecture state1_arch of state1 is
                type PossibleStates is ( Start_state, Decide_state, Right_state, Left_state);
                signal present_state, next_state : PossibleStates; --Enumerated Type
begin
   Make_Next_State:
   process (present_state, GO, LEFT_RIGHT)
   begin
   case present_state is
         when Start_state =>
         RIGHT <='0'; LEFT <='0';-- Atribuição do estado de saída para o estado
                 if GO = '0' then next_state <= Start_state;
                else
                        next_state <= Decide_state;
                end if;
         when Decide_state =>
                RIGHT <='0'; LEFT <='0';-- Controle sucursal
                if LEFT_RIGHT = '1' then next_state <= Left_state;
                else
                        next_state <= Right_state;
                end if;
         when Right_state =>
         RIGHT <='1'; LEFT <='0';-- Saída para direita
                if GO = '0' then next_state <= Right_state;
                else
                        next_state <= Start_state;
                end if;
         when Left_state =>
         RIGHT <='0'; LEFT <='1';-- Saída para esquerda
                if GO = '0' then next_state <= Left_state;
                else
                        next_state <= Start_state;
                end if;
end case;
end process Make_Next_State;
```

Apêndice 3 - Canal Comum nº7 (SS#7 ou SS7)

Sistema de sinalização número 7 (SS7), ou canal comum número 7, é um conjunto de protocolos que descreve uma comunicação entre elementos da rede telefônica pública. É utilizada também na comunicação entre o SoftSwitch e a rede telefônica pública. SS7 serve tanto para que duas centrais ou nós da rede telefônica conversem, em termos de troca de informação, por exemplo de controle de chamadas e configuração de serviços como para que uma central se comunique com uma base de dados de serviços (como 0800, 0400, 0300). SS7 é uma sofisticada e poderosa forma de sinalização por canal comum.

Igualmente a muitos protocolos de sinalização, SS7 é constituído por uma arquitetura de camadas. As 3 camadas mais baixas juntas formam a parte de mensagem de transferência ou MTP (*Message Transfer Part*). A MTP é responsável pela segurança e confiabilidade de encaminhamento das mensagens, cujo conteúdo é fornecido por camadas mais superiores. MTP usa *link* de sinalização para o roteamento das mensagens ao devido destino requisitado. Camadas superiores têm funções diferentes e são implementadas de acordo com a exigência da rede. O controle de chamada (isto é, o estabelecimento e desconexão de chamadas) é tratado por uma série de 4 camadas de protocolos de controle de chamada como o ISUP ou TUP. Outras funções são construídas em cima de outra camada denominada SCCP. A figura 6.1 mostra as camadas do SS7.

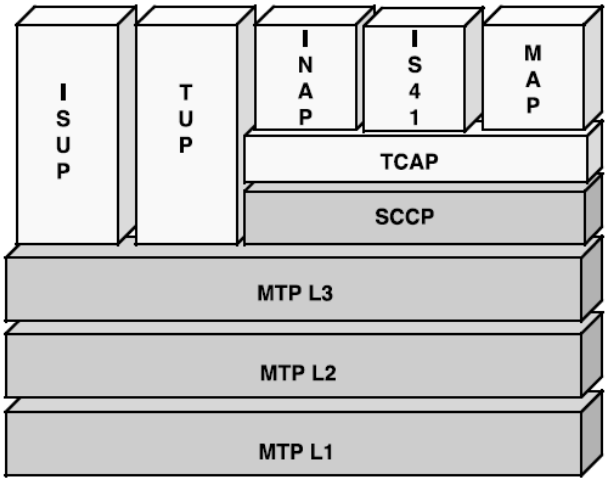


Figura A3.1: Camadas do SS7

O SS7 estabelece um quadro para que haja troca de dados entre sistemas dentro da rede via canal de sinalização dedicado. O *link* de sinalização é a base da arquitetura SS7. Ele possibilita que haja comunicação entre as entidades no âmbito da rede, permite a troca de informação e é essencial

para a eficácia dos recursos de segurança que compõe a rede de forma combativa contra possíveis invasões no sistema.

As normas SS7 incluem agora especificações para uma ampla diversidade de gestão de tarefas de telefonia e têm-se revelado bem sucedidas. Como a tendência é avançar no sentido da convergência entre a rede de telefonia pública de circuito-comutado e a comunicação por pacotes IP, SS7 tornou-se tema de uma atenção significativa por parte dos desenvolvedores na procura da integrar melhor essas duas arquiteturas de rede levando-as para a convergência. Sendo assim, o entendimento do SS7 é componente vital para a compreensão da atual e da próxima geração de redes públicas (NGN).