UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

"IMPLEMENTAÇÃO DE UMA SOLUÇÃO MODULAR E PORTÁVEL DAS FUNÇÕES DE CONTROLE DO NÍVEL 2 DO SISTEMA DE SINALIZAÇÃO POR CANAL COMUM NÚMERO 7 UTILIZANDO DISPOSITIVOS DE LÓGICA PROGRAMÁVEL"

MITSUYOSHI NISHI DE CARVALHO

MANAUS
2013

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MITSUYOSHI NISHI DE CARVALHO

"IMPLEMENTAÇÃO DE UMA SOLUÇÃO MODULAR E PORTÁVEL DAS FUNÇÕES DE CONTROLE DO NÍVEL 2 DO SISTEMA DE SINALIZAÇÃO POR CANAL COMUM NÚMERO 7 UTILIZANDO DISPOSITIVOS DE LÓGICA PROGRAMÁVEL"

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade do Amazonas, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação.

Orientadora: Profa. Dra. Marly Guimarães Fernandes Costa

MANAUS
2013

Ficha Catalográfica
(Catalogação realizada pela Biblioteca Central da UFAM)

C331i

Carvalho, Mitsuyoshi Nishi de
Implementação de uma solução modular e portável das funções de controle do nível 2 do sistema de sinalização por canal comum número 7 utilizando dispositivos de lógica programável / Mitsuyoshi Nishi de Carvalho. - Manaus: UFAM, 2013.
104 f.; il. color.

Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal do Amazonas.
Orientadora: Profª Dra. Marly Guimarães Fernandes Costa

1. SS7-Sistema de Sinalização por Canal Comum número 7 2. MTP2-Message Transfer Part level 2 3. SDL- Specification and Description Language 4. VHDL- Linguagem de descrição de hardware I. Costa, Marly Guimarães Fernandes (Orient.) II. Universidade Federal do Amazonas III. Título

CDU (1997): 621.39(043.2)

MITSUYOSHI NISHI DE CARVALHO

IMPLEMENTAÇÃO DE UMA SOLUÇÃO MODULAR E PORTÁVEL DAS FUNÇÕES DE CONTROLE DO NÍVEL 2 DO SISTEMA DE SINALIZAÇÃO POR CANAL COMUM NÚMERO 7 UTILIZANDO DISPOSITIVOS DE LÓGICA PROGRAMÁVEL

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação de Sistemas.

Aprovado em 28 de fevereiro de 2013.

BANCA EXAMINADORA

Prof. Dr. Cícero Ferreira Fernandes Costa Filho, Presidente

Universidade Federal do Amazonas- UFAM

Marly Guimarães Fernandes Costa, Membro

Universidade Federal do Amazonas- UFAM

Prof. Dr. Jozias Parente de Oliveira, Membro

Universidade do Estado do Amazonas- UEA

Dedico este trabalho às três grandes mulheres da minha vida, minha avó, minha mãe e minha esposa.

AGRADECIMENTOS

A Deus, por ter me dado a oportunidade de viver essa experiência que proporcionou a minha aproximação a Ele;

À minha mãe, por ter me dado todas as condições para chegar até aqui;

À minha amada esposa, pela compreensão das minhas ausências, pelas noites em claro ao meu lado, pelas palavras de incentivo e pelo amor dedicado;

À minha orientadora Profa. Dra. Marly Guimarães pela confiança, pelas palavras de incentivo que vieram sempre nos momentos certos e por ter me guiado até a conclusão deste trabalho;

Ao amigo Victor de Afonso Valenzuela Diaz, pela sua dedicação e pela motivação nos momentos críticos;

Aos amigos que sempre me incentivaram e me apoiaram durante essa jornada;

Ao Centro de Pesquisa e Desenvolvimento de Tecnologia Eletrônica e da Informação – CETELI/UFAM pela disponibilização da infraestrutura utilizada na realização desse trabalho;

À Financiadora de Estudos e Projetos (FINEP) pelo apoio financeiro através do convênio n.° 01.10.0474.00 - FINEP/UNISOL/FUA/TRÓPICO.

AGRADEÇO.

RESUMO

O Sistema de Sinalização por Canal Comum número 7 (SS7) é um dos mais importantes sistemas de sinalização utilizado em redes atuais de telecomunicações e continua a ser usado em novas arquiteturas tanto de telefonia fixa como móvel. O presente trabalho de dissertação apresenta a implementação das funções de controle do nível 2 do Sistema de Sinalização número 7 (denominado nesse trabalho como MTP2-H) utilizando a linguagem de descrição de *hardware* VHDL. A especificação das funções de controle do nível 2 do SS7 é feita através de diagramas em linguagem SDL na recomendação Q.703 do ITU-T. Para realizar a implementação, foi desenvolvida uma metodologia para conversão de sistemas descritos em SDL para VHDL, composta por um conjunto de regras e um modelo padrão em VHDL que foram aplicados nos referidos diagramas SDL. A implementação foi realizada de forma a proporcionar características de modularidade e portabilidade ao código gerado. Com isso, o módulo desenvolvido poderá ser replicado o número de vezes em que for necessário em um componente de lógica programável (respeitando as limitações do componente) e também poderá ser sintetizado em componentes de diferentes fabricantes. A validação da implementação foi feita por meio de testes funcionais utilizando a ferramenta de simulação Modelsim. O código desenvolvido foi compilado em ferramentas de desenvolvimento de diferentes fabricantes para validar a característica de portabilidade e para estimar a quantidade de recursos necessários nos componentes de lógica programáveis.

Palavras-chave: SS7, MTP2, SDL, linguagem de descrição de *hardware*, VHDL.

ABSTRACT

The Common Channel Signaling System No. 7 (SS7) is one of the most important signaling systems used in today's telecommunication networks and continues to be used on new architectures of fixed and mobile telephony. This dissertation presents an implementation for the control functions of the Signaling System Number 7 level 2 (referred in this work as MTP2-H) using the VHDL as the hardware description language. The specification of the control functions for the SS7 level 2 is made by using SDL diagrams described in Recommendation Q.703 of ITU-T. To accomplish the implementation, a methodology was developed for conversion of systems described in SDL to VHDL, consisting of a set of rules and a standard VHDL model that were applied on those SDL diagrams. The implementation was performed in order to provide the modularity and portability characteristics to the generated code. This way, the developed module can be replicated as many times as necessary in a programmable logic component (respecting the limitations of the component) and can also be synthesized in components from different manufacturers. The validation of implementation was done by means of functional tests using the Modelsim simulation tool. The developed code was compiled in development tools from different manufacturers to validate the portability feature and to estimate the amount of resources required in programmable logic components.

Keywords: SS7, MTP2, SDL, hardware description language, VHDL.

ÍNDICE DE ILUSTRAÇÕES

ÍNDICE DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Nos anos 60 e 70 ocorreu a digitalização das redes de telecomunicações, tanto nos seus elementos de comutação, tais como centrais telefônicas, como nos meios de transmissão. Tal digitalização ocorreu concomitantemente, com a introdução de processadores digitais no controle das mesmas. Os elementos dessas redes passaram não somente a realizar suas funções sob o controle de tais processadores e dos programas neles armazenados, como passaram a se intercomunicar entre si também de forma inteligente. De fato, as redes de telecomunicações passaram a constituir verdadeiras redes de processadores. Nesse período, o CCITT (*International Telegraph and Telephone Consultative Committee*) como agência reguladora, verificou a necessidade de uma linguagem que pudesse ser usada tanto para descrever e especificar os protocolos e sistemas por ela definidos nas suas recomendações como para que outras agências reguladoras e fabricantes também pudessem usá-los nos seus desenvolvimentos. Isso motivou o desenvolvimento da linguagem SDL (*Specification and Description Language*) (OLSEN *et al.*, 1994). Ela está definida na recomendação Z.100 no documento ITU-T (1988).

Os protocolos de comunicação tem um papel fundamental nas redes de telecomunicações. Em termos da sua abrangência e diversidade de uso em diferentes tipos de redes e equipamentos, um dos principais conjuntos de protocolos é o SS7 – Sistema de Sinalização por canal comum Número 7. O SS7 está definido nas recomendações Q.700 – 795 no documento ITU-T (1980).

O uso de componentes programáveis permite que funções complexas implementadas total ou parcialmente em *software* possam ser realizadas em *hardware*. Isso permite diversas combinações de velocidades e quantidades de circuitos reproduzidos em um mesmo componente *hardware*. Por outro lado, o uso de uma linguagem de alto nível, como o VHDL, permite que o projeto dessas soluções possa ser feito sem entrar no nível de componentes *hardware* elementares, facilitando a simulação e a realização de alterações, seja para corrigir problemas encontrados ou para adicionar novas funcionalidades.

Este trabalho de dissertação aborda a definição de uma metodologia que possibilita o projeto e a implementação em componente de lógica programável usando a linguagem VHDL (*Very–high-speed integrated circuits Hardware Description Language*) a partir da especificação de protocolos feita em linguagem SDL (*Specification and Description Language*). No caso em tela, implementou-se um dos protocolos do sistema SS7, o MTP-2 (*Message Transfer Part level* 2).

Do exposto, a proposta contida nessa dissertação envolve os três grandes temas mencionados: O Sistema de Sinalização SS7, a linguagem de especificação SDL e a linguagem de descrição de *hardware* VHDL para componentes de lógicas programáveis.

1.1 MOTIVAÇÃO

Em 2009, o Centro de Pesquisa e Desenvolvimento em Tecnologia Eletrônica e da Informação, CETELI, órgão suplementar da Universidade Federal do Amazonas submeteu à Financiadora de Estudos e Projetos (FINEP) uma proposta para o desenvolvimento do projeto Hardware Universal Inovador (HUI). O projeto HUI foi aprovado, sob o convênio n.° 01.10.0474.00 - FINEP/UNISOL/FUA/TRÓPICO, para ser desenvolvido com o apoio financeiro do Fundo para o Desenvolvimento Tecnológico das Telecomunicações (FUNTTEL) no âmbito do Plano Nacional de Ciência e Tecnologia, tendo a empresa Trópico Telecomunicações como interveniente. Esse plano prevê o financiamento de centros de pesquisa tecnológica para o desenvolvimento de projetos em novas tecnologias cujos resultados são destinados a empresas nacionais, as intervenientes.

O projeto HUI consiste no desenvolvimento de uma nova plataforma *hardware* para os equipamentos de telecomunicações da empresa Trópico. O principal objetivo dessa plataforma é minimizar problemas atuais e futuros resultantes da grande longevidade desses equipamentos que chegam a atuar em campo por varias décadas. Os principais problemas são os de obsolescência e dificuldade de encontrar componentes eletrônicos tanto para manutenção como para expansão.

Com tempos de vida comerciais tão longos, novas versões *hardware* e *software* de partes desses equipamentos, que geralmente são de grande porte, são introduzidas ao longo desse tempo de vida. É, portanto, comum encontrar partes *hardware* e *software*

de diversas versões convivendo num mesmo equipamento. Dessa forma, as soluções desenvolvidas no projeto HUI precisam ser compatíveis com todas essas versões. Essa compatibilidade deve permitir que placas de vários tipos e versões possam ser substituídas por placas do projeto HUI.

Para reduzir ao máximo a quantidade de placas diferentes desenvolvidas nesse projeto, optou-se por desenvolver placas programáveis que podem atuar com funcionalidades distintas.

Além de acrescentar novas funcionalidades nos equipamentos já existentes, novos equipamentos vêm também sendo desenvolvidos pela empresa Trópico, principalmente para redes de nova geração (NGN e IMS, por exemplo) que integram a comutação por circuito (incluindo a telefonia convencional) com as redes de pacotes. A Trópico tem procurado, na medida do possível, usar uma plataforma básica comum para todos esses equipamentos. Isso estende os requisitos das placas do projeto HUI para atender também esses diversos tipos de novos equipamentos. Mais do que simplesmente atender as demandas deixadas pelas placas de versões anteriores, espera-se também das placas e soluções do projeto HUI o atendimento a demandas por novas funcionalidades e/ou novos equipamentos para as soluções da Trópico.

Uma das principais funcionalidades introduzidas ao longo dos projetos desenvolvidos na Trópico e que, sendo utilizada em diversos equipamentos da empresa, faz parte da sua plataforma básica, foi a sinalização SS7.

Duas dissertações de mestrado foram desenvolvidas dentro do projeto HUI, o de Lima (2012) e esta dissertação. O objetivo definido de ambas foi o desenvolvimento de uma solução para o nível 2 da SS7 (MTP2) que seja suficientemente flexível e modular para poder ser usada em vários equipamentos e placas. Para a divisão do conteúdo das duas dissertações foi usado o conceito mostrado por DIAZ (2009), onde a camada MTP-2 da SS7 foi dividida em duas partes denominadas MTP2-H (*Message Transfer Part level 2, High)* e MTP2-L (*Message Transfer Part level 2, Low*). O MTP2-H executa as funções de controle do nível 2 do SS7 e foi desenvolvido neste trabalho. O MTP2-L executa funções de tratamento das mensagens de sinalização ao nível de bit e foi desenvolvido no trabalho de Lima (2012).

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Este trabalho de dissertação tem por objetivo implementar as funções de controle do nível 2 do Sistema de Sinalização por Canal Comum número 7 utilizando uma linguagem de descrição de *hardware* que possa ser utilizado em componentes de lógica programáveis de diferentes fabricantes e que proporcione a característica de escalabilidade aos equipamentos de telecomunicações em que for utilizado.

1.2.2 Objetivos Específicos

- Desenvolvimento de uma metodologia para geração de sistemas digitais utilizando a linguagem de descrição de *hardware* VHDL a partir de uma especificação em SDL, o que inclui:
  - Definição de regras para conversão;
  - Elaboração de um modelo de conversão;
  - Validação do modelo.
- Implementação das funções de controle do nível 2 do SS7 (MTP2-H) o que incluiu o desenvolvimento e a validação das várias partes que a compõem:
  - *Link State Control* (LSC);
  - *Initial Alignment Control* (IAC);
  - *Reception Control* (RC);
  - *Transmission Control* (TXC);
  - *Processor Outage Control* (POC);
  - *Congestion Control* (CC);
  - *Alignment Error Rate Monitor* (AERM);
  - *Signal Unit Error Rate Monitor* (SUERM).
- Integração do MTP2-H, desenvolvido neste trabalho, com o MTP2-L desenvolvido por Lima (2012), formando o sistema MTP2, para validar a interface definida entre os dois módulos.

- Levantamento dos recursos do componente de lógica programável necessários para a implementação de uma ou mais instâncias do MTP2.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho de dissertação está organizado nos seguintes capítulos:

- Capítulo 1 – Introdução;
- Capítulo 2 – Revisão bibliográfica;
- Capítulo 3 – Fundamentação teórica;
- Capítulo 4 – Metodologia;
- Capítulo 5 – Resultados e discussões;
- Capítulo 6 – Conclusões.

O Capítulo 2 apresenta uma revisão bibliográfica de artigos e dissertações de mestrados que descrevem a implementação do SS7 em equipamentos de telecomunicações bem como artigos que descrevem métodos para a conversão de sistemas especificados em SDL para a descrição em VHDL.

O Capítulo 3 apresenta os principais conceitos utilizados no desenvolvimento deste trabalho, são eles: O Sistema de Sinalização por Canal Comum SS7 e a Linguagem de Especificação de Sistemas SDL.

No Capítulo 4 é descrito o procedimento utilizado para a implementação do MTP2-H, especificado na recomendação Q.703 do ITU-T. Fazem parte dessa implementação o desenvolvimento do modelo de conversão de sistemas especificados em SDL para a descrição em VHDL, bem como a definição de uma arquitetura para validação da implementação utilizando a ferramenta de simulação ModelSim-Altera®6.6d (Quartus II 10.1sp1) Starter Edition.

No Capítulo 5 são apresentados os principais resultados e as contribuições do trabalho. Isso inclui a definição da metodologia de geração de projetos utilizando o VHDL a partir de especificações descritas em SDL e a implementação do MTP2-H. Constam também nesse capítulo os resultados das compilações para os cenários com 2, 4, 8 e 16 instâncias do sistema MTP2.

No Capítulo 6 são apresentadas as conclusões do trabalho e sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Com o objetivo de identificar os trabalhos científicos publicados sobre o tema " Implementação de uma solução modular e portável das funções de controle do nível 2 do Sistema de Sinalização por Canal Comum número 7 utilizando dispositivos de lógica programável", foram realizadas pesquisas nas principais bases de dados literárias tais como IEEE e ACM, assim como no banco de teses e dissertações do portal de periódicos da CAPES.

Dessas buscas foram identificados poucos documentos científicos relacionados diretamente com o tema principal deste trabalho. A maioria está relacionada com o tema sinalização por canal comum número 7 voltado para a interação entre protocolos de outros sistemas de comunicação, por exemplo, o sistema de comunicação móvel, arquiteturas da rede de sinalização e análise de desempenho. Fez parte também das pesquisas os trabalhos que descrevem os métodos utilizados para a conversão de sistemas especificados em SDL para linguagem de descrição de *hardware* VHDL. Este é um assunto relevante, pois toda a especificação do MTP2 foi realizada em SDL.

Dos documentos científicos relacionados com o tema deste trabalho pode-se citar os seguintes artigos técnicos:

- *Implementing Signaling System No. 7 in Trópico RA System* (NETO; SILVEIRA, 1990);

- *Realization of a Signaling System No 7 Network for AT&T* (DONOHOE *et al.*,1986);

O trabalho de Neto e Silveira (1990) descreve a implementação do sistema de sinalização número 7 (SS7) na central telefônica Trópico RA. Este equipamento foi desenvolvido pelo Centro de Pesquisa e Desenvolvimento da Telebrás (CPqD), empresa brasileira, antiga fornecedora de serviços de telecomunicações. A central Trópico RA possui uma arquitetura de *hardware* distribuída composta por módulos funcionais. Todos os módulos se comunicam através de uma interface de *hardware* proprietária, formando uma rede, por onde trafegam sinais de controle e voz. Dentre os módulos que compõem a central podemos citar os módulos de comutação, de terminais, auxiliares e

de sinalização por canal comum. O módulo de comutação controla e estabelece a conexão interna dos sinais de voz que serão tratados por outros módulos na central. O módulo de terminais possui a função de prover o suporte aos vários tipos de padrões de terminação na rede de telefonia, tais como assinantes analógicos, terminais multifrequenciais R2, troncos digitais e analógicos, entre outros. O módulo auxiliar possui os recursos para a execução de *software* de funções de alto nível tais como roteamento de chamada e funções do SS7. O módulo de sinalização por canal comum possui os recursos para a execução das funções de tratamento das mensagens de sinalização e suporte aos terminais de sinalização. A implementação do SS7 na central teve como principais objetivos a flexibilidade, a modularidade e a otimização do desenvolvimento de forma que o impacto na estrutura de *software* de controle da central fosse a menor possível. A implementação foi baseada nas recomendações da ITU-T (*International Telecommunication Union – Telecommunication Standardization Sector*) série Q.700 a Q.795 que especifica o SS7. O protocolo do SS7 possui uma arquitetura dividida em camadas baseada no modelo de referência OSI para transporte de dados. Os blocos funcionais que compõem o protocolo do SS7 estão divididos em quatro camadas. As três primeiras camadas são responsáveis pelo transporte de dados na rede SS7 de forma confiável, com recursos para detecção e correção de erros das mensagens, tratamento e roteamento das mensagens na rede de sinalização. Ele recebe a denominação de subsistema de transferência de mensagens (*Message Transger Part* - MTP). Fazendo uma associação ao modelo de referência OSI, pode-se dizer que a camada 1 (MTP-1), corresponde à camada física, a camada 2 (MTP-2) corresponde ao enlace de dados e a camada 3 (MTP-3) corresponde a uma parte da camada de rede. A compatibilidade com a camada de rede do modelo OSI é obtida com o bloco funcional de subsistema de controle de conexão de sinalização (*Signaling Connection Control Part* – SCCP) associado ao MTP-3. A quarta camada do protocolo corresponde à camada de aplicações, denominada subsistema de usuário (*User Part* - UP), que tem como principais aplicações o subsistema de usuários de telefone (*Telephone User Part* - TUP) e o subsistema de usuários ISDN (ISDN *User Part* - ISUP). Os UPs de um equipamento utilizam os serviços do MTP para se comunicar com outros UPs na rede de sinalização. A Figura 1 mostra a arquitetura do protocolo do SS7 relacionado com o modelo de referência OSI.

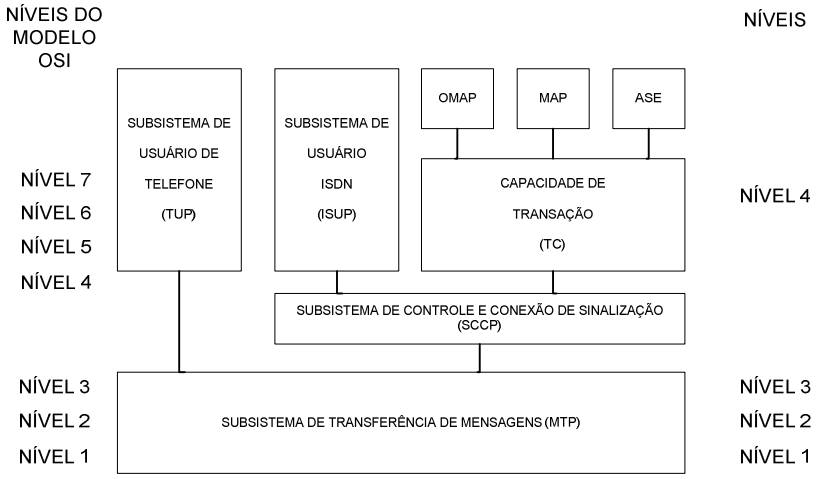


Figura 1. Arquitetura do protocolo do SS7.

FONTE: (NETO; SILVEIRA, 1990)

A implementação do SS7 na central Trópico RA foi dividida em etapas. Inicialmente foi prevista a implementação das funções do MTP e TUP, seguida pelo ISUP visando o teste piloto previsto para o ano de 1992. As outras funções (SCCP, OMAP e TCAP) seriam desenvolvidas após a definição da estratégia de implantação da Rede Inteligente (*Inteligent Network* - IN) nacional.

A camada 1 do MTP (MTP-1) foi implementada no módulo terminal que em conjunto com o módulo de comutação direciona as mensagens de SS7 para o módulo de sinalização por canal comum. Esse módulo é responsável em executar as funções do MTP-2 e as funções de tratamento de mensagens de sinalização (*Signaling Message Handling* – SMH) do MTP-3, que corresponde a uma parte das funções do MTP-3. As funções de gerenciamento da rede de sinalização (*Signaling Network Management* - SNM) do MTP-3, que corresponde a outra parte das funções do MTP-3, foram centralizadas no módulo auxiliar da central. As funções do TUP e ISUP foram implementadas nos módulos terminais por se tratarem de funções ligadas diretamente aos assinantes. O SCCP e o TCAP serão implementados nos módulos auxiliares. A Figura 2 mostra essa implementação dentro da arquitetura da central Trópico RA.

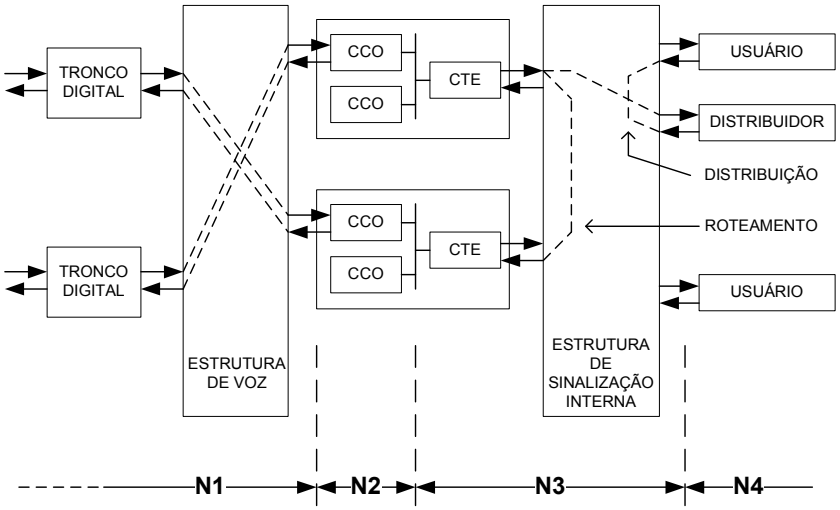


Figura 2. Implementação do SS7 na arquitetura da central Trópico RA.
FONTE: (NETO; SILVEIRA, 1990)

Cada módulo de sinalização por canal comum tem a capacidade de tratar quatro enlaces de canal comum. Ele é composto por duas placas que executam as funções do MTP-2 denominada CCO, com capacidade para tratar dois enlaces de SS7 cada uma, e uma placa que executa a função de tratamento de mensagens do MTP-3, denominada CTE, com capacidade para tratar mensagens de quatro enlaces de SS7. A arquitetura de implementação das funções de tratamento de mensagens de sinalização do MTP-3 adotada é a descentralizada. Cada placa CTE trata as mensagens de sinalização dos enlaces dos módulos ao qual ele pertence reduzindo o tempo gasto na transferência dessas mensagens entre os blocos funcionais MTP-2 e MTP-3.

Cada placa do módulo de sinalização por canal comum possui um microprocessador como elemento principal. A comunicação entre os microprocessadores das placas CCO e CTE é feita através de uma memória comum. A comunicação entre as funções de tratamento de mensagens do MTP-3 e os subsistemas de usuários (UP) são realizadas através de uma estrutura de barramento com vários processadores.

A memória comum está dividida em vários blocos que são utilizados para armazenar as mensagens de SS7 a serem transmitidas pelo MTP-2 bem como as mensagens recebidas pelos dois enlaces de SS7 suportado pela placa CCO. A estratégia de se utilizar uma memória comum para comunicação entre processadores foi adotada para que o tempo gasto na transferência das mensagens entre o MTP-2 e o MTP-3 fosse

o mínimo possível. Esse ganho é obtido porque na mesma área de memória que um dos microprocessadores escreve o outro lê e vice-versa.

Neto e Silveira (1990) descrevem também o protocolo de comunicação utilizando a memória comum, bem como a sua organização e a arquitetura de implementação da placa que executa as funções do MTP-2. Através de experimentos, verificou-se que é possível atingir uma taxa de 660 mensagens por segundo, nas duas direções, na comunicação entre os microprocessadores utilizando a memória comum. Segundo Neto e Silveira (1990), esse valor é significativo considerando que o microprocessador utilizado foi o Intel 8088.

O trabalho de Donohoe *et al.* (1986) descreve a transição do Sistema de Sinalização número 6 para o Sistema de Sinalização número 7 (SS7) da empresa norte-americana AT&T e algumas considerações na implementação do novo sistema. A AT&T é uma empresa que fornece serviço de telefonia nos Estados Unidos da América e também desenvolve os equipamentos utilizados em sua rede. A necessidade de oferecer mais serviços aos seus usuários motivou a transição para o novo sistema de sinalização.

Para a implantação do SS7 na rede da AT&T, foram utilizados novos equipamentos de transferência de mensagens de sinalização, conhecidos como STP (*Signaling Transfer Point*), com capacidade para tratar dez vezes mais mensagens em relação aos STPs do sistema de sinalização anterior. A velocidade de transmissão também aumentou de 2,4 e 4,8kbps para 56kbps. Os novos STPs estavam preparados para tratar mensagens dos dois sistemas de sinalização para minimizar o impacto da mudança. A topologia da rede de sinalização foi mantida havendo uma pequena redução na quantidade de STP.

A implementação do subsistema de transferência de mensagens (MTP) e do subsistema de controle e conexão de sinalização (SCCP) foram realizados no novo STP, cuja arquitetura *hardware* é composta por um processador central com nós microprocessados interligados numa topologia de conexão em anel, conforme mostra a Figura 3. As funções de MTP e SCCP são distribuídas entre os nós e o processador central que também é responsável em fornecer as funções de operação, administração e manutenção da rede em anel. Segundo Donohoe *et al.* (1986) esta topologia atende aos

requisitos de confiabilidade necessários para o sistema aliado à flexibilidade e redundância que garantem a ampliação e a evolução do equipamento para atender demandas futuras de serviços.

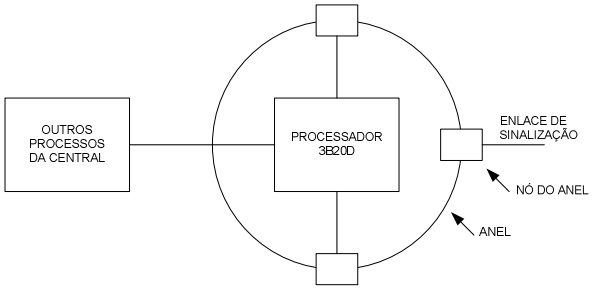


Figura 3. Arquitetura de implementação do MTP e SCCP no STP da AT&T.
FONTE: (DONOHOE *et al.*, 1986)

Da pesquisa ao banco de teses e dissertações da CAPES foi identificada a seguinte dissertação que aborda a implementação do subsistema de transferência de mensagens:

- Contribuições para a Implementação do Sistema de Sinalização por Canal Comum nº 7 em uma Central Telefônica Baseada em Ambiente de Processamento Distribuído (D'ÁVILA,1998);

O trabalho de D'Ávila (1998) apresenta um estudo do sistema de sinalização por canal comum número 7 (SS7) e como resultado prático a implementação do subsistema de transferência de mensagens (*Message Transfer Part* - MTP) na central telefônica ELCOM® desenvolvida pela empresa brasileira Batik Equipamentos. O projeto de implementação do MTP na central fez parte do convênio entre a empresa e o Departamento de Ciência da Computação (DCC) da Universidade Federal de Minas Gerais (UFMG).

Antes de abordar a implementação do MTP na central o autor faz uma descrição da estrutura *hardware* e *software* da central ELCOM. A central possui uma arquitetura *hardware* modular composta por várias unidades, onde cada uma possui a capacidade para executar funções de controle e comutação independentemente. Em cada unidade existe um computador, compatível com o padrão IBM-PC, interligado a várias placas através de um barramento proprietário da Batik. Essas placas executam funções específicas necessárias para o funcionamento da central. Os computadores que controlam as unidades se comunicam entre si através de uma rede local *ethernet*. As

matrizes de comutação das unidades são interligadas entre si através de planos de comutação intermodular. A Figura 4 apresenta um diagrama ilustrando a arquitetura *hardware* da central ELCOM.

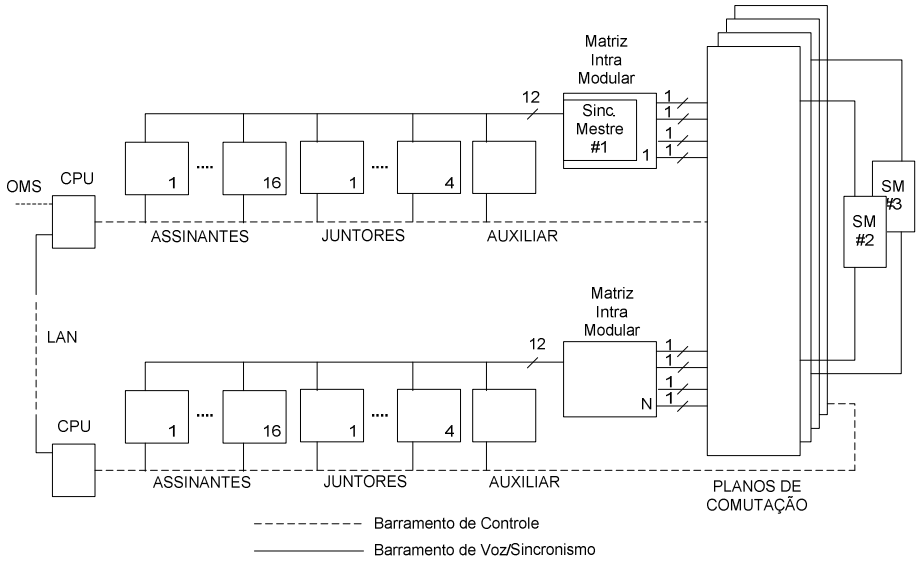


Figura 4. Arquitetura *hardware* da central ELCOM.

FONTE: (D'ÁVILA, 1998)

A estrutura *software* da central está dividida em quatro blocos principais: bloco de Sistema Básico, de Inicialização e Configuração, de Operação e Manutenção e de Processamento de Chamadas. O Programa Controlador, como é denominado o conjunto dos blocos de *software* funcionais é armazenado e executado em cada módulo *hardware* da central controlada pelo computador.

Segundo D'Ávila (1998) a implementação do SS7 na central ELCOM iniciou com a implementação do MTP e dos subsistemas de usuário TUP e ISUP. Para uma parte da implementação do MTP foi adotada uma solução comercial composta por um *hardware* que executa parte das funções do MTP-2 e um conjunto de código fonte de *software* com funções do MTP-3, funções MTP-2 não executadas pelo *hardware* e funções de comunicação entre o MTP-2 (*hardware* e *software*) e o MTP-3.

A implementação do MTP-1 foi executada pela Batik através da adaptação do *hardware* da central de forma a possibilitar a comunicação da central com o mundo externo através de enlaces PCM de 2,048Mbps. A solução desenvolvida permitia que o as informações de canal comum alocado em um determinado canal do enlace PCM externo fosse comutado a qualquer canal de qualquer enlace de PCM interno a central. Isso garante uma grande flexibilidade para a expansão das funções do MTP na central.

A implementação das funções do MTP-2 é composta em parte por uma placa terminal de SS7 em barramento ISA para computadores e a outra por um componente de *software* residente no computador ao qual a placa de SS7 está instalada. Ambas as soluções são comerciais.

As funções do MTP-3 foram implementadas completamente em *software*. Foi adotada a arquitetura de implementação centralizada referente às funções de gerenciamento da rede de sinalização do MTP-3. Portanto, essas funções eram executadas por um único elemento na central. A parte referente às funções de tratamento das mensagens de sinalização, por interagirem com as funções do subsistema de usuário, como por exemplo, o TUP, foram implementadas de forma distribuída nos módulos onde fossem instalados as funções do MTP-2. O desenvolvimento dos subsistemas de usuário TUP e ISUP foram realizados paralelamente ao desenvolvimento do MTP sendo o TUP desenvolvido pela Batik e o ISUP pelo aluno de mestrado Marcelo R. P. Miranda no convênio do DCC da UFMG com a Batik. Esses subsistemas eram executados pelas unidades que possuem módulos terminais.

A validação da implementação do SS7 na central ELCOM foi realizada através de testes de homologação pela Telebrás, tendo como resultado a aprovação. A implementação do SS7 teve também como resultado o aumento de desempenho da central relacionado à aplicação telefônica.

Abaixo citam-se alguns dos artigos que abordam o assunto de conversão de sistemas especificados em SDL para a linguagem de descrição de *hardware* VHDL.

- *Stoht – Na SDL-to-Hardware Translator* (BONATTI; FIGUEIREDO, 1995);
- *A Methodology and Algorithms for Efficient Interprocess Communication Syntesis from Description in SDL* (SVANTESSON *et al.*, 1998);

Bonatti e Figueiredo (1995) propõem em seu trabalho o uso da linguagem SDL para especificar o desenvolvimento de circuitos digitais. Eles definem um conjunto de elementos mais usados da linguagem SDL e algumas restrições para obter um mapeamento na linguagem de descrição de *hardware* VHDL. As regras para este

mapeamento foram definidas para a elaboração de um algoritmo que foi implementado em um aplicativo de conversão chamado Stoht (*SDL-to-Hardware Translator*).

Os elementos da linguagem SDL suportados são: sistema, bloco, subestrutura, processo, sinal, canal, rota de sinal, variável e tipo de dados inteiro e booleano. Para cada um dos elementos foi definido um modelo em VHDL. O uso de processos dinâmicos, fila de entradas de tamanho infinito e operações aritméticas com ponto flutuante não são suportados pelo algoritmo de conversão.

O algoritmo de mapeamento define um relógio global que servirá de base para o funcionamento do circuito digital resultante da conversão, ou seja, os circuitos serão síncronos com esse relógio. As transições das máquinas de estados existentes nos processos em SDL levarão um ciclo do relógio para serem processadas assim como os sinais enviados de um processo para o outro.

Cada processo em SDL, quando convertido, será formado por duas entidades em VHDL. Uma responsável pela comunicação e a outra pelo comportamento do processo. Na entidade de comunicação é possível utilizar três tipos diferentes de armazenamento do sinal recebido. O primeiro, denominado Protocolo 1, utiliza uma fila de entradas de somente uma posição para todos os sinais recebidos pelo processo. O segundo, denominado Protocolo 2, utiliza uma fila de entradas de uma posição para cada sinal recebido. O terceiro, denominado Protocolo 3, utiliza uma fila de entradas de tamanho igual a N para todos os sinais recebidos. As interfaces da entidade de comunicação foram padronizadas de modo a permitir a alteração dos protocolos sem que haja a necessidade de alterar a entidade de comportamento. Isso garante ao desenvolvedor maior flexibilidade. A entidade de comportamento é definida por uma máquina de estados finita modelada por uma estrutura CASE/WHEN em VHDL. Em cada estado da máquina uma lista de entradas é verificada, de acordo com o diagrama em SDL, utilizando a estrutura IF/ELSIF/THEN. A prioridade dos sinais de entradas é definida pela ordem de verificação na referida estrutura.

A indicação de ativação de um sinal em um processo é realizada invertendo-se o valor desse sinal. O processo de destino identificando essa inversão entende que o sinal foi ativado no processo de origem e o repassa à entidade de comportamento de acordo com o protocolo adotado.

O algoritmo definido foi implementado em um *software* chamado Stoht que converte descrições textuais do SDL (SDL/PR) para VHDL. Ele trabalha em conjunto com uma ferramenta de edição de SDL gráfico (SDL/GR) chamada Telelogic® SDT e com o compilador e sintetizador VHDL da Mentor Graphics®. O Stoth lê um arquivo contendo a descrição em SDL/PR, analisa a sintaxe e a semântica e gera os modelos em VHDL respeitando a estrutura do sistema. Ele gera também uma biblioteca com os arquivos necessários para a compilação. O Stoth é de domínio público e pode ser instalado na plataforma UNIX.

Svantensson *et al.* (1998) propõem uma metodologia para conversão de sistemas especificados em SDL para linguagem de descrição de *hardware* VHDL que contempla os sistemas em SDL com processos que são criados dinamicamente ou que possuam mais de uma cópia do mesmo processo.

Na metodologia apresentada, cada processo em SDL é implementado por dois blocos de *hardware*, um denominado de comunicação ($P_{Comunicação}$) e o outro de computação ($P_{Comptação}$). O bloco de computação implementa o comportamento do processo descrito em máquina de estados finita estendida (MEFE). O bloco de comunicação possui um circuito que possibilita a comunicação entre os processos do sistema SDL. Um bloco *hardware* adicional chamado supervisor também é utilizado. Esse bloco é responsável pelo roteamento dos sinais e pelo gerenciamento da comunicação entre os processos. A Figura 5 mostra um exemplo de como fica a implementação em VHDL de um bloco contendo três processos em SDL utilizando a metodologia proposta.

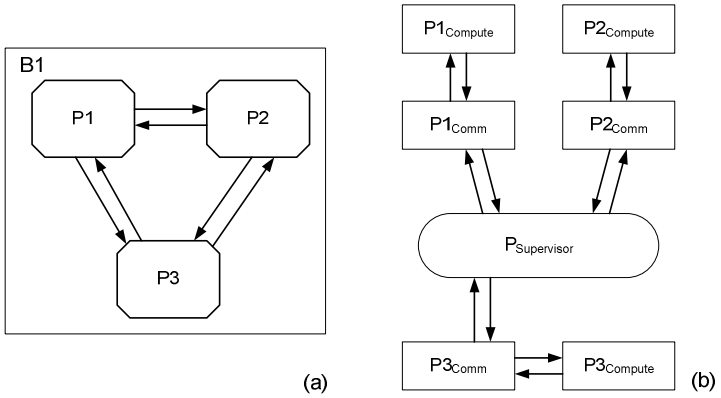


Figura 5. (a) Descrição de um bloco com três processos em SDL. (b) Implementação equivalente em VHDL.

FONTE: (BONATTI; FIGUEIREDO, 1995)

Cada bloco de computação recebe e envia os sinais via o bloco de comunicação correspondente. Observa-se que todos os sinais trocados entre os processos passam pelo bloco supervisor. Isso possibilita o compartilhamento do barramento de comunicação economizando recursos na sintetização do *hardware*. Para sistemas em SDL com muitos blocos, a funcionalidade do bloco supervisor é divida para simplificar a implementação. Nesse caso, para cada bloco SDL existirá um bloco supervisor que controlará a comunicação entre os processos e, entre os blocos SDL haverá um bloco supervisor que controlará a comunicação entre os blocos. A Figura 6 mostra um sistema contendo três blocos SDL e a respectiva implementação com a divisão dos blocos supervisores.

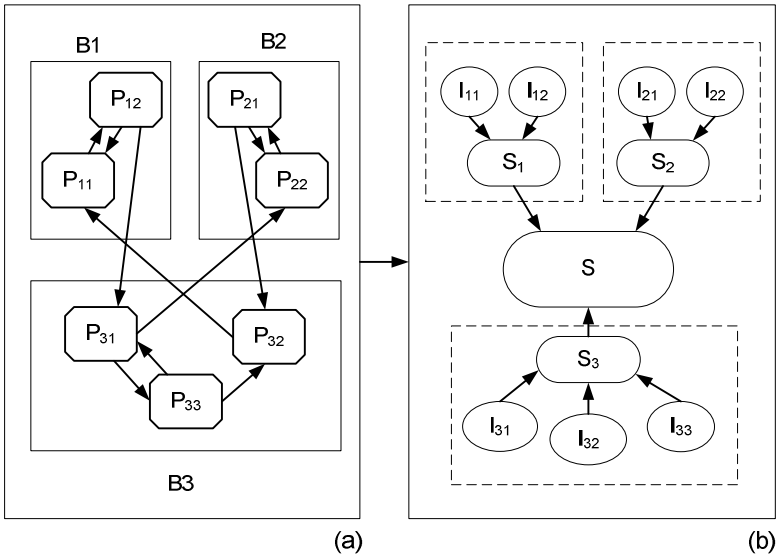


Figura 6. (a) Sistema em SDL composto por três blocos. (b) Implementação equivalente em VHDL. FONTE: (SVANTESSON *et al.*, 1998)

Os autores definiram cinco tipos de implementação do bloco de comunicação, são eles: *Send and forget*[1], *strobe*[1], *handshaking*[1], FIFO para uma instância de processo e FIFO para múltiplas instâncias do mesmo processo. *Send and forget* consiste no envio do sinal pelos processos sem a preocupação da entrega. *Strobe* consiste no envio de dados em conjunto com um sinal de controle. No *handshaking* o processo que gera o sinal aguarda uma resposta do processo que o recebeu. A FIFO para uma instância de processo armazena os sinais recebidos para somente uma instância do processo. A FIFO para múltiplas instâncias do mesmo processo armazena os sinais recebidos para cada uma das cópias com a devida identificação. A escolha do tipo de comunicação a ser

[1] Termos em inglês sem tradução direta para o português.

implementado é feita após uma análise da atividade de comunicação do processo levando-se em consideração alguns parâmetros definidos pelos autores.

A implementação do processo com múltiplas instâncias (cópias) no sistema em SDL é feita utilizando-se somente um processo implementado que é compartilhado no tempo pelas múltiplas instâncias utilizadas no sistema. Isso requer uma infraestrutura de gerenciamento para programar o uso desse processo pelas múltiplas instâncias no tempo correto, mantendo o estado de cada instância e com os recursos para o chaveamento de contexto. Para isso, os autores utilizaram um *buffer* para armazenar o estado e outro para armazenar as variáveis de cada instância. Um terceiro *buffer* também é utilizado para armazenar a lista de instâncias ativas com os ponteiros para acessar corretamente os *buffers* de estado e de variáveis. Essas funções são executadas por um bloco *hardware* denominado de gerenciador que em resumo possui funções similares aos encontrados em um sistema operacional.

Os autores aplicaram a metodologia em um exemplo de um controlador de tráfego em um cruzamento. Esse exemplo ilustra o uso de múltiplas instâncias de um mesmo processo. Eles concluem o trabalho mencionando que a metodologia desenvolvida possibilita a conversão de sistemas especificados em SDL que possuem tanto processos que são criados dinamicamente quanto os que utilizam múltiplas instâncias de um mesmo processo. Esses dois casos ainda não tinham sido abordados até então.

Como pode ser observado, a maioria dos artigos, dissertações e teses encontradas sobre o SS7 data das décadas de 80 e 90. Poucos são da década atual quando comparados a outras áreas de pesquisa. Identifica-se como possível causa o fato desta área de desenvolvimento tecnológico ser estratégica para as empresas. Dessa forma, a não divulgação dos resultados resguarda os interesses dessas empresas. Quando muito elas buscam a proteção e o registro da propriedade industrial, a partir do depósito de patentes. Uma busca nas bases de patentes do INPI, USPTO e Espacenet verificou-se a existência de centenas de patentes que estão diretamente relacionadas com o SS7 ou que utilizam os serviços do SS7. A Tabela 1 apresenta algumas das patentes relacionadas ao SS7.

Tabela 1. Lista de patentes relacionadas ao SS7.

Número	Data	Título
09/668,797	22/09/2000	Nó Número 7 de Sistema de Sinalização de Canal Comum
982074	25/09/1998	Processo e Aparelho para transmitir informação de Sinalização em uma rede de Telecomunicações entre um Par de Partes de Aplicação, e, Nó de conexão para acoplar informação de Sinalização de uma Rede de Sinalização de Canal Comum de um Sistema de Telecomunicações a uma rede baseada em Protocolo de Internet
05 005168.9	09/03/2005	Processo para Transmissão de Mensagens SS7, Disposição de Rede bem como Elemento de Rede

A patente de número 09/668,797 foi depositada nos Estados Unidos da América pela empresa Telefonaktiebolaget LM Ericsson. Ela é referente a um equipamento composto por vários processadores e uma estrutura de comutação que executa as funções do SS7 em uma rede de sinalização.

A patente de número 982074 foi depositada na Finlândia pela empresa Telefonaktiebolaget LM Ericsson. Ela é referente a um software e equipamento com capacidade para transmitir informações de sinalização na rede SS7 e também na rede IP.

A patente de número 05 005168.9 foi depositada na Organização Européia de Patentes pela empresa NOKIA SIEMSENS NETWORKS GMBH & CO. Ela é referente a um processo de transmissão de mensagens SS7, uma disposição de rede bem como um elemento de rede que facilite o acréscimo de elementos na rede de sinalização de forma mais fácil, pois toda inclusão de um Ponto de Sinalização na rede acarreta na alteração das tabelas de roteamento de vários Pontos de Sinalização.

O Sistema de Sinalização número 7 foi especificado pela ITU-T (International Telecommunication Union – Telecommunication Standardization Sector) com o primeiro lançamento em 1980, consolidado no Livro Amarelo (*Yellow Book*) (ITU-T, 1980) (ITU-T, 1980)com as recomendações da série Q.700-795, seguido por algumas revisões, tendo como última revisão lançada em 1993 (ITU-T, 1993a).

Toda implementação de SS7 é baseada nas recomendações da série Q.700. Nelas são especificadas todos os blocos funcionais do SS7 bem os testes das implementações. A seguir estão relacionadas as principais recomendações da série Q.700 agrupadas por blocos funcionais:

- Q.701 – Q.710: Especifica a implementação do Subsistema de Transferência de Mensagens (*Message Trasnfer Part* - MTP);
- Q.71X: Especifica a implementação do Subsistema de Conexão e Controle de Sinalização (*Signaling Connection Control Part* – SCCP);
- Q.72X: Especifica a implementação do Subsistema de Usuário de Telefone (*Telephone User Part* - TUP);
- Q.76X: Especifica a implementação do Subsistema de Usuário da Rede Digital de Serviços Integrados (*Integrated Services Digital Network User Part* - ISUP);
- Q.72X: Especifica a implementação do Subsistema de Usuário de Telefone (*Telephone User Part* - TUP);
- Q.78X: Especifica o procedimento de testes dos blocos funcionais MTP-2, MTP-3, TUP, ISUP, SCCP e TCAP.

3 FUNDAMENTOS TEÓRICOS

3.1 FUNDAMENTOS DE TELEFONIA

A invenção do telefone no século XIX deu início à comunicação por voz à longa distância. O telefone é composto basicamente por elementos transdutores que transformam sons em sinais elétricos e vice-versa, um elemento de sinalização (campainha ou sinal luminoso) e um acionador de chamada que indica a intenção do usuário em estabelecer uma chamada.

Um usuário do serviço de comunicação por telefone é denominado assinante. Para o estabelecimento da comunicação entre dois assinantes é necessário que os aparelhos telefônicos estejam eletricamente conectados. Essa conexão recebe o nome de linha telefônica.

Uma das arquiteturas que possibilita a comunicação entre todos os assinantes é a malha completamente conectada. Nesse caso, um sistema com *n* assinantes, resulta em $n(n-1)/2$ conexões, conforme mostra a Figura 7.

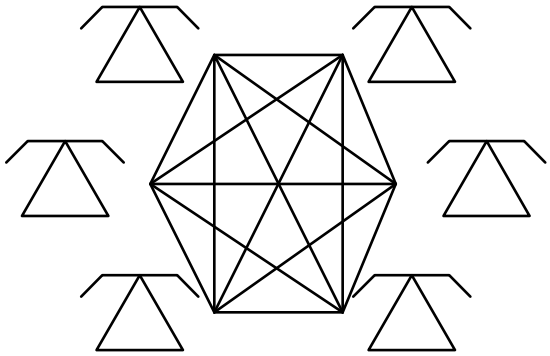


Figura 7. Exemplo de uma arquitetura de malha completamente conectada com seis assinantes, resultando em 15 conexões.

Uma grande limitação da arquitetura de malha completamente conectada referida advém do fato de que a inclusão de um novo assinante implica no acréscimo de ligação elétrica do assinante acrescido para cada um dos outros já existentes. Além disso, grande parte das linhas ficará ociosa porque um assinante estabelece somente uma conexão a outro assinante por vez. Uma forma mais eficiente é conectar todos os assinantes a um elemento central, formando uma ligação em estrela, conforme mostra a Figura 8. Este elemento "concentrador" é denominado central telefônica. Sua função é

controlar e estabelecer as conexões entre os assinantes. Nessa arquitetura, para *n* assinantes serão necessárias *n* linhas telefônicas, ou seja, o acréscimo de um assinante no sistema implica conexão de somente uma linha à central. Este valor é consideravelmente inferior quando comparado à arquitetura de malha completamente conectada, tornando esta última implementação viável técnica e financeiramente.

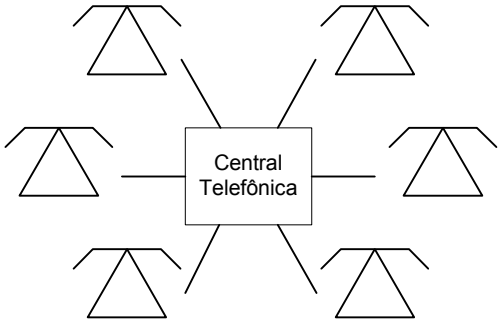


Figura 8. Exemplos de Arquitetura de ligação em estrela com seis assinantes e seis conexões.

Em uma central telefônica a operação de conexão entre dois assinantes recebe o nome de comutação.

As primeiras centrais telefônicas desenvolvidas logo após a criação do telefone eram manuais. O controle e o estabelecimento de chamadas eram executados por funcionários da operadora, através de uma mesa de operação (PINHEIRO, 2004). Com a evolução da tecnologia, as centrais passaram a ser automáticas do tipo: mecânicas, eletromecânicas, até chegarem às centrais digitais que deram origem às Centrais de Programa Armazenado (CPA).

3.1.1 Rede de Telefonia

Uma central telefônica concentra um número limitado de assinantes de uma localidade. Para possibilitar a comunicação entre dois assinantes de localidades distintas, as centrais telefônicas devem estar conectadas entre si. A essa conexão dá-se o nome de juntor ou tronco.

Através dos troncos, as centrais trocam as informações necessárias para o estabelecimento e finalização da chamada entre os seus assinantes, transporte dos sinais de voz, bem como os sinais necessários para o controle da rede e tarifação.

A estrutura formada por assinantes conectados às centrais e estas conectadas a outras através dos troncos formam a rede de telefonia, vide ilustração, Figura 9.

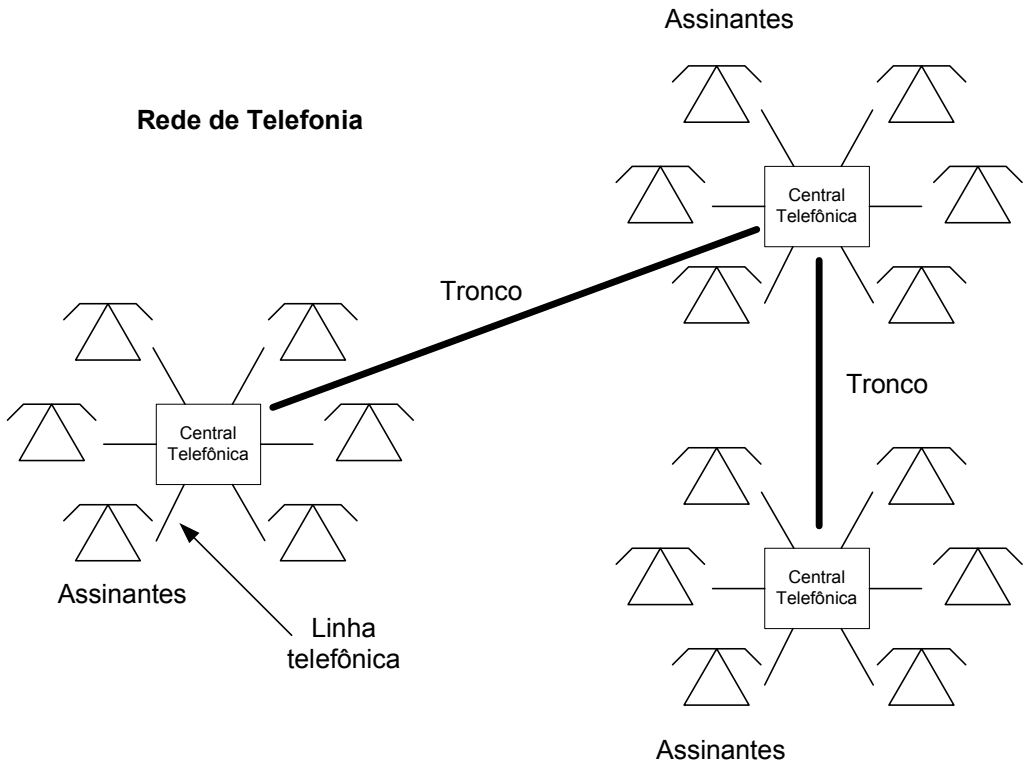


Figura 9. Exemplo de uma rede de telefonia.

As centrais telefônicas podem ser classificadas como nacional ou internacional de acordo com a sua localização e função na rede.

3.1.2 Sinalização

Ao retirar um telefone do gancho e digitar o número do telefone ao qual se deseja estabelecer uma conexão, são trocadas algumas informações entre o assinante e a central telefônica. Essas informações serão trocadas entre os elementos da rede, em momentos específicos, para o estabelecimento, controle e finalização da chamada. A essa troca de informações dá-se o nome de sinalização.

Inicialmente a sinalização estava limitada aos sinais relacionados com o controle das chamadas de voz, mas atualmente a sinalização abrange também a troca das informações relativas ao gerenciamento da rede de telefonia e aos serviços implementados por aplicações que utilizam a referida estrutura (SCHLANGER, 1986).

A sinalização pode ser classificada de acordo com a funcionalidade, como: sinalização de acesso, registro e linha.

A sinalização de acesso compreende as informações trocadas entre o assinante e a central telefônica, e vice-versa, tais como tons audíveis de assinante ocupado, tom de discar, fone no gancho, fone fora do gancho e número do assinante de destino discado.

A sinalização de linha corresponde à troca de informações entre centrais para o estabelecimento da conexão e desconexão do canal de voz, entre os assinantes, bem como a supervisão do estado da conexão entre as centrais relacionadas à chamada.

A sinalização de registro são as informações trocadas entre centrais e estão relacionadas aos assinantes, tais como número discado e número de origem, estado do assinante, tipo de assinante, entre outros.

A sinalização de linha e de registro ocorre entre as centrais telefônicas envolvidas entre os assinantes de origem e destinos através dos troncos.

A sinalização entre as centrais telefônicas através de troncos podem ocorrer de duas formas, uma por sinalização por canal associado e outro por sinalização por canal comum.

A sinalização por canal associado consiste no transporte de uma parte das informações relativas à chamada através de um canal específico de sinalização no tronco (canal 16) e a outra parte através do próprio canal de voz associado à chamada. O nome canal associado está relacionado fato do canal de sinalização estar associado somente aos canais de voz do tronco ao qual ele pertence. As informações eram transportadas por códigos multifrequenciais de acordo com uma tabela padrão. Essa técnica limitava a quantidade e o tipo de informações possíveis de serem transportados, consequentemente a quantidade de serviços oferecidos, e gerava um atraso grande para o estabelecimento de chamadas.

A sinalização por canal comum foi desenvolvida para possibilitar um transporte de informações não só relacionadas a chamadas de voz, mas também de gerência, controle e serviços de forma confiável e eficiente. Os sinais são transportados de forma totalmente digital e estruturada o que permite um único canal de sinalização em um determinado tronco transportar informações relativas a vários canais de voz (cerca de 1000), inclusive de outros troncos, em um tempo muito menor quando

comparado com a sinalização por canal associado. O nome canal comum está relacionado ao fato de um canal poder transportar informações de várias chamadas (canais de voz), não só do tronco ao qual ele pertence, como também de vários outros troncos que interligam as centrais.

3.2 SISTEMA DE SINALIZAÇÃO POR CANAL COMUM NÚMERO 7

A primeira implementação da sinalização por canal comum foi realizada em 1976 pela empresa AT&T utilizando a Sinalização por Canal Comum número 6 (*Common Channel Signalling number 6* - CCS6) baseada na recomendação da *International Telegraph and Telephone Consultative Committee* (CCITT) (ITU-T, 1988a).

A CCS6 possuía uma estrutura de protocolo monolítica devido à limitação da banda de transmissão disponível na época, que era de 2,4kbps. A quantidade de mensagens era bastante restrita sendo direcionada somente para sinalização de tronco entre centrais. Em 1980 passou por uma evolução que permitiu a troca de mensagens entre uma central telefônica e um banco de dados na rede. Com a evolução, deu-se início a uma nova arquitetura de rede, onde centrais podiam consultar base de dados centralizadas e oferecer mais serviços, como por exemplo, ligações gratuitas com o número 800 (serviço equivalente ao 0800 utilizado no Brasil) e cartões de pagamento de chamadas (MODARRESSI; SKOOG, 1990).

Em meados da década de 70, o CCITT iniciou os estudos para a elaboração do sistema de sinalização por canal comum número 7 (*Signalling System number 7* – SS7). A primeira versão foi lançada em 1980 (ITU-T, 1993a) e passou por evoluções até chegar a uma estrutura de protocolos mais próxima ao modelo de referência OSI (*Open Systems Interconnections*) para transporte de dados. Essa abordagem facilita a ampliação dos serviços fornecidos inicialmente pelo sistema.

De acordo com as especificações do CCITT:

> O objetivo global do SS7 é fornecer uma padronização internacional de propósito geral de um sistema de sinalização por canal comum:

> • otimizado para operação em redes de telecomunicações digitais em conjunto com centrais de programa armazenado (CPA);
>
> • que possa atender aos requisitos atuais e futuros de transferência de informações entre processadores na rede de telecomunicações para controle de chamada, controle à distância e sinalização de gerenciamento e manutenção;
>
> • que forneça um meio confiável para transferência de informações na sequência correta sem perdas ou duplicação. (ITU, 1993, p.1, tradução nossa).

O SS7 foi especificado tanto para atender aos requisitos de transporte de dados relacionados com as chamadas telefônicas, quanto para qualquer tipo de informação dos serviços implementados na rede.

Ele utiliza enlaces de sinalização que interligam as centrais e outros componentes da rede para as transferências das informações. Está preparado para operar com taxas de transferência de 64kbps em sistemas digitais e atende também aos requisitos de transferências de mensagens em sistemas analógicos com velocidades mais baixas.

3.2.1 Rede de Sinalização do SS7

A rede de sinalização é formada por pontos de sinalização (PS) conectados através de enlaces de sinalização. Os PS são todos os elementos da rede de telefonia que implementam o SS7. Os três principais tipos são (LIN, 1996):

- PAS – Ponto de Acesso de Sinalização (*Service Switching Point* – SSP): são as centrais telefônicas que possuem assinantes e estão conectados à rede SS7. Eles podem ser a origem ou o destino das mensagens de sinalização na rede.
- PTS – Ponto de Transferência de Sinalização (*Signalling Transfer Point* – STP): são as centrais que recebem e encaminham as mensagens do SS7 entre os elementos da rede, funcionando como roteadores. Eles verificam o

endereço de destino da mensagem e a direciona para o enlace correspondente. Por serem elementos concentradores de enlaces, as suas aplicações na rede são sempre redundantes. Assim, em caso de falha de um PTS, o outro poderá suprir a demanda sem prejudicar a troca de mensagens da rede.

- PCS – Ponto de Controle de Serviço (*Service Control Point* – SCP): são os elementos da rede que possuem a base de dados para os serviços de consulta, como por exemplo o serviço 800 (chamada gratuita). Eles fornecem as informações solicitadas pelos PAS através dos PTS.

A definição de ponto de sinalização é uma definição lógica, ou seja, podem existir situações em que uma central pode assumir mais de uma função de PS, como por exemplo, a de PAS e PTS, conforme o tipo de mensagem a ser tratada (ITU-T, 1993a).

A Figura 10 mostra um exemplo de rede de sinalização com os principais PS.

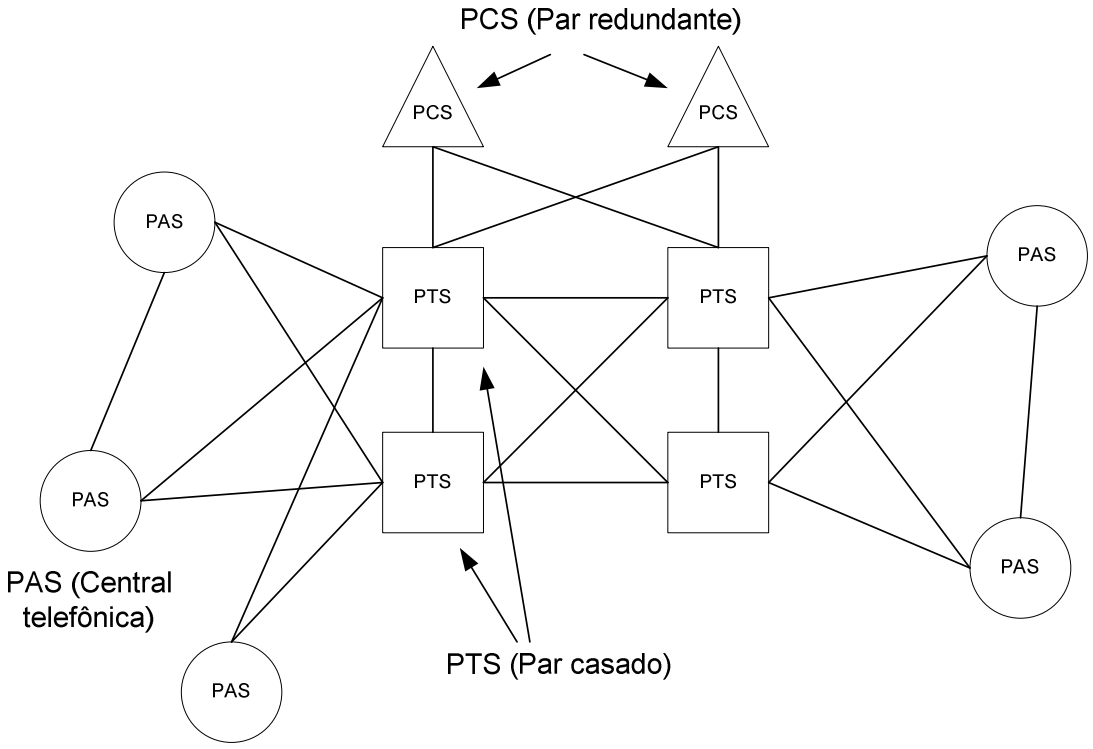


Figura 10. Rede de sinalização e seus elementos.

O enlace de sinalização consiste no meio de transmissão de dados bidirecional por onde as informações de SS7 são transportadas entre os pontos de sinalização. Para garantir a confiabilidade no tráfego das informações, são disponibilizados entre dois PS, um determinado número de enlaces denominado conjunto de enlaces (*link set*).

Dependendo da funcionalidade dos PS, é possível que existam mais de um conjunto de enlaces entre eles.

Todo PS é identificado na rede, seja nacional ou internacional, por um código único conhecido pelo termo em inglês *point code*. Através do código, é possível determinar o caminho por onde a mensagem irá passar até chegar ao seu destino. As regras para a determinação dos códigos estão descritas na Recomendação Q.704 da (ITU-T, 1993a).

Dois pontos de sinalização possuem relação de sinalização quando existe a possibilidade do subsistema de usuário de ambos trocarem mensagens de SS7. Como o PTS, de forma geral, não implementa o subsistema de usuário, ele não terá uma relação de sinalização com outro PS. Porém, ele será um elemento intermediário entre dois pontos de sinalização que possuam a relação de sinalização.

A associação entre o caminho que uma mensagem percorre entre dois PS e a relação de sinalização entre eles determina o modo de sinalização. Dois modos de sinalização são previstos no SS7, são eles: modo associado e quase-associado.

No modo associado, os PS de origem e destinos são adjacentes, ou seja, os pontos de origem e destinos estão diretamente conectados. Já no modo quase-associado, a mensagem vai seguir um caminho pré-determinado entre a origem e o destino passando por alguns PTS. A Figura 11 ilustra esse conceito.

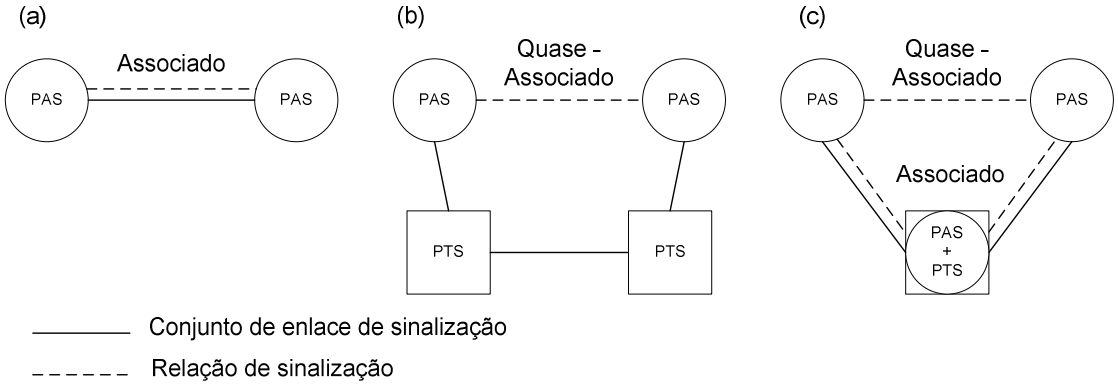


Figura 11. Exemplos de modos de sinalização.

(a) Modo associado. (b) Modo quase-associado. (c) Modo associado e quase-associado

O caminho percorrido por uma mensagem de sinalização entre dois PS é denominado de rota. Na rede, é possível que existam mais de uma rota entre os PS. O

que vai determinar por qual rota a mensagem será encaminhada são as condições de tráfego da rede e a tabela de roteamento presente em cada PS.

3.2.2 Arquitetura do SS7

A fim de atender aos requisitos de modularidade e flexibilidade para evoluções futuras, o protocolo do SS7 foi baseado em camadas seguindo a idéia do modelo de referência OSI de transporte de dados. Inicialmente, ele foi modelado somente para o controle dos circuitos de telefonia, composto por quatro camadas, dividido em dois blocos funcionais: o Subsistema de Transferência de Mensagens (*Message Transfer Part* – MTP, versão em inglês) que compreende as camadas/níveis 1 a 3 e o Subsistema do Usuário (*User Part* – UP, versão em inglês) que compreende a camada/nível 4. O subsistema de transferência de mensagens é responsável pelo transporte confiável das mensagens do SS7 de forma seqüenciada entre os PS. O subsistema do usuário são todas as entidades que utilizam o MTP para transporte de dados. A primeira versão previa como principais subsistemas de usuário o *Signalling Connection Control Part* (SCCP), o *Telephone User Part* (TUP) e o ISDN *User Part* (ISUP). A Figura 12 mostra a relação entre os níveis funcionais.

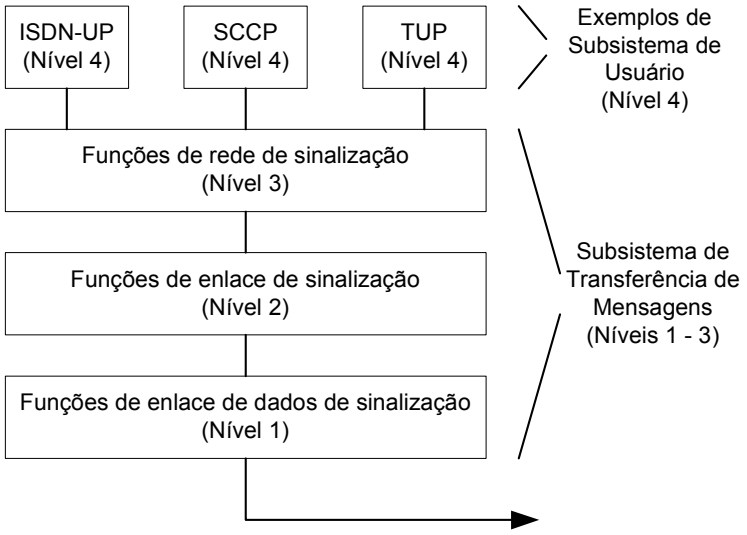


Figura 12. Relação entre os níveis funcionais da primeira versão da arquitetura do SS7.
FONTE: (ITU-T, 1993a)

O SS7 evoluiu e passou a ter uma arquitetura mais próxima do modelo OSI, porém, ele não prevê elementos funcionais que sejam equivalentes às camadas de

Apresentação, Sessão e Transporte. A Figura 13 apresenta a relação entre os elementos e o modelo OSI.

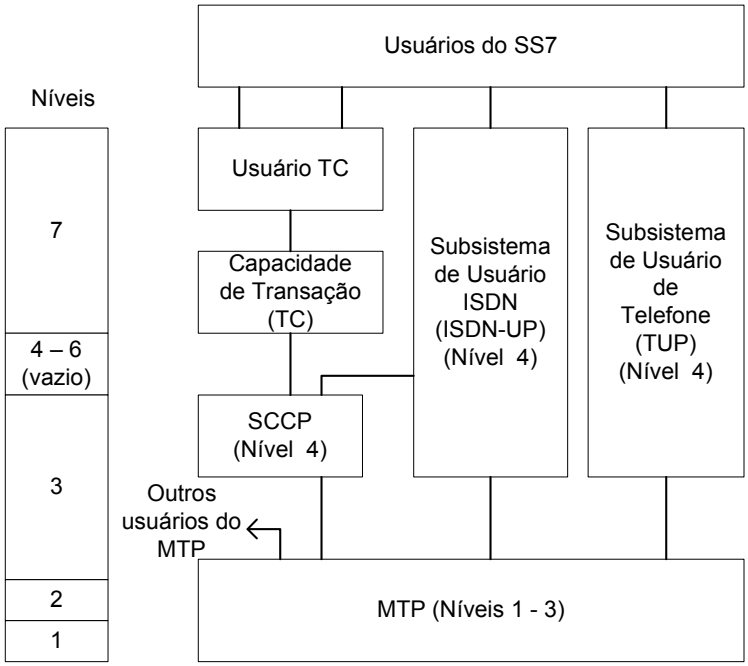


Figura 13. Arquitetura do SS7 e a relação entre os elementos funcionais e o modelo de referência OSI.

FONTE: (ITU-T, 1993a)

O bloco funcional MTP está diretamente relacionado com as três primeiras camadas do modelo OSI. A camada 1 de enlace de sinalização de dados (MTP-1) define as características físicas e elétricas e o meio por onde serão transportados os dados entre os PS. A camada 2 de enlace de sinalização (MTP-2), em conjunto com o MTP-1, fornece os procedimentos para o transporte da mensagem de sinalização de forma confiável entre os PS diretamente conectados. A camada 3 de rede de sinalização (MTP-3) corresponde a uma parte da camada 3 do modelo OSI. Ele é responsável em controlar o roteamento das mensagens na rede e na distribuição do tráfego entre os enlaces de sinalização em caso de falha, assim como em direcionar as mensagens aos Subsistemas de usuários e aos enlaces de sinalização.

As funcionalidades de cada camada do MTP bem como as interfaces entre eles serão detalhadas no próximo item por serem parte do tema principal do trabalho.

O SCCP foi desenvolvido para complementar o MTP-3 e fornecer a funcionalidade equivalente à camada de rede do modelo OSI. Ele fornece funções para a transferência de dados relacionados ou não ao controle dos circuitos de telefonia

baseado em serviços de rede orientados à conexão (*connection-oriented*) e sem conexão (*connectionless*) (ITU-T, 1993a).

O bloco *Transactions Capabilities* (TC) fornece as funções que possibilitam aos seus usuários estabelecerem conexões com outros nós da rede de sinalização não relacionados aos circuitos de telefonia. Em relação ao modelo OSI, ele está classificado como a base da camada 7 de aplicações e faz a interface entre as aplicações do SS7 e o SCCP.

O *Telephone User Part* (TUP) possui as funções de sinalização de controle das chamadas telefônicas na rede nacional e internacional e utiliza diretamente o MTP para o transporte das mensagens. Já o *Integrated Services Digital Network User Part* (ISDN *User Part* - ISUP) fornece todas as funções do TUP bem como as funções de conexão de dados do ISDN (MODARRESSI; SKOOG, 1990). Ele tem interface direta com o MTP para o transporte de mensagens de sinalização relacionadas ao controle de chamada telefônica e também com o SCCP para conexões de dados. O TUP e o ISUP correspondem à camada 7 de aplicação do modelo OSI.

As aplicações que utilizam o TC proporcionam ao usuário do telefone e do ISDN o acesso aos serviços básicos e suplementares da rede.

3.3 SUBSISTEMA DE TRANSFERÊNCIA DE MENSAGENS – MTP

O Subsistema de Transferência de Mensagens (MTP) é o bloco funcional do protocolo do SS7 responsável em garantir a transferência e a entrega das mensagens de sinalização, passadas pelos Subsistemas de Usuário (*User Part* – UP), aos seus destinos na rede SS7. Ele possui funções capazes de tomar as ações necessárias para garantir a transferência das mensagens em situação de falha na rede ou no sistema. Os principais UP que utilizam os serviços do MTP são o SCCP, o TUP e o ISUP. O MTP é especificado pelas recomendações Q.701, Q. 702, Q.703 e Q.704 do ITU-T (ITU-T, 1993a) e este texto é todo baseado nessas recomendações.

Funcionalmente, o MTP está dividido em três camadas/níveis: enlace de dados de sinalização (nível 1), enlace de sinalização (nível 2) e funções de rede de sinalização (nível 3). Cada nível fornece determinados serviços ao nível imediatamente superior e estabelece uma conexão com o seu par na rede através dos protocolos de níveis.

A Figura 14 apresentada na recomendação Q.701 da ITU-T, ilustra a estrutura em níveis das funções do SS7. A estrutura apresentada não é uma especificação de implementação do sistema, mas pode ser considerada como um possível modelo.

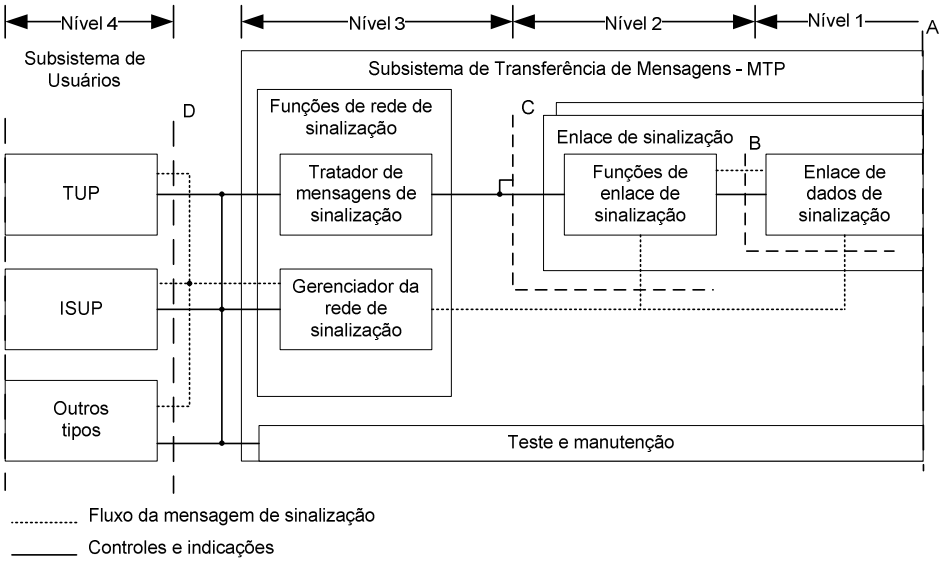


Figura 14. Estrutura em níveis das funções do SS7.

FONTE:(ITU-T, 1993a)

As interfaces funcionais entre os níveis representados A, B, C e D podem ou não existir, tudo vai depender da implementação adotada.

O nível 1 do MTP (MTP-1), que corresponde ao enlace de dados de sinalização, assim como no modelo de referência OSI, é responsável pela camada física do protocolo. A função desse nível é transportar as informações de sinalização, bit a bit, através das interfaces físicas existentes entre os pontos de sinalização na rede. Ele corresponde a um caminho de transmissão de dados de sinalização bidirecional, no qual existe um canal para transmissão e um canal para recepção operando na mesma taxa de dados (Ref. Q.702). O enlace de dados de sinalização digital é composto por um canal de transmissão e um equipamento de terminação digital em cada extremo que dá acesso ao enlace de transmissão incluindo comutadores digitais de cada lado, conforme mostra a Figura 15. O enlace de transmissão pode ser do tipo terrestre ou via satélite.

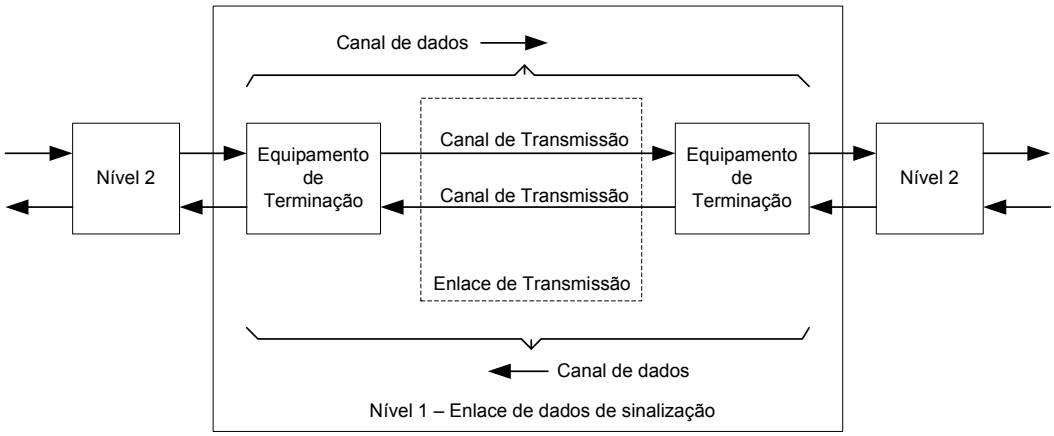


Figura 15. Diagrama de blocos do enlace de dados de sinalização digital.

FONTE: (ITU-T, 1993a)

O comutador digital é utilizado para extrair os dados de sinalização de um determinado canal do enlace de transmissão e encaminhá-los ao nível 2 (MTP-2) e vice-versa. A utilização do comutador digital possibilita também a reconfiguração automática dos canais de transmissão entregues ao MTP-2, pois com ele é possível configurar o uso de qualquer um dos canais do enlace de transmissão. A comutação é do tipo semipermanente, ou seja, um determinado canal será associado a um bloco de MTP-2 do início ao fim da conexão entre os pontos de sinalização sendo alterado somente em caso de falha.

No padrão ITU-T G.702, pelo enlace de transmissão, são enviados quadros de dados no formato PCM de 32 canais a 2048 kbps. No padrão ANSI, são enviados quadros no formato PCM de 24 canais mais um bit de quadro a 1544 kbps.

Como no Brasil o padrão utilizado é o ITU-G702, neste trabalho será utilizado tal padrão. Dos 32 canais disponíveis no enlace de transmissão, somente 31 podem ser utilizados para transmissão de informações de voz ou sinalização. Um canal é dedicado para sincronismo e indicação de alarme. O nível 1 deve estar preparado também para transmitir e receber sinais analógicos através de equipamentos *modem*.

O nível 2 do MTP (MTP-2), que corresponde ao enlace de sinalização, é responsável em transferir de forma confiável, mensagens de sinalização entre os pontos de sinalização através do enlace de dados de sinalização (MTP-1). As mensagens de sinalização são transmitidas através de unidades de sinal (*signal unit* – SU) que podem ser de três formas: unidades de sinal de mensagens (*message signal unit* – MSU), unidade de sinal de estado do enlace (*link status signal unit* – LSSU) e unidade de sinal

de preenchimento (*fill-in signal unit* – FISU). As MSUs são utilizadas para transportar as mensagens de sinalização e de teste do enlace, as LSSUs são utilizadas para transportar as mensagens referentes ao estado do enlace e as FISUs não transportam nenhuma informação útil, sendo utilizada somente para garantir o alinhamento do enlace. A Figura 16 apresenta o formato de cada uma dessas unidades de sinal.

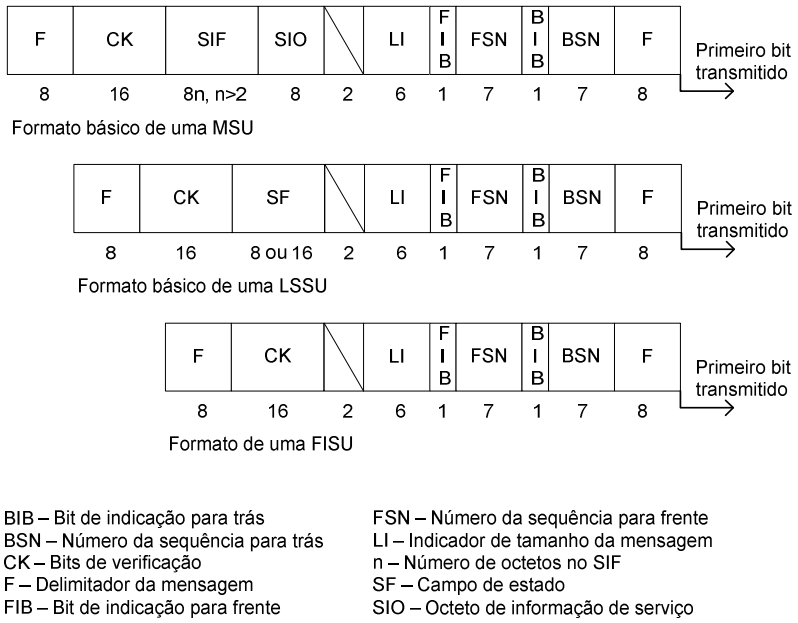


Figura 16. Formato das unidades de sinal.

FONTE: (ITU-T, 1993a)

Para garantir a transferência das mensagens entre os pontos de sinalização, o MTP-2 possui as seguintes funções: delimitação e alinhamento da unidade de sinal, correção e detecção de erro das mensagens, alinhamento do enlace, supervisão dos erros das mensagens, controle do estado do enlace e controle de fluxo. A Figura 17 apresenta a interação entre os blocos funcionais do MTP-2.

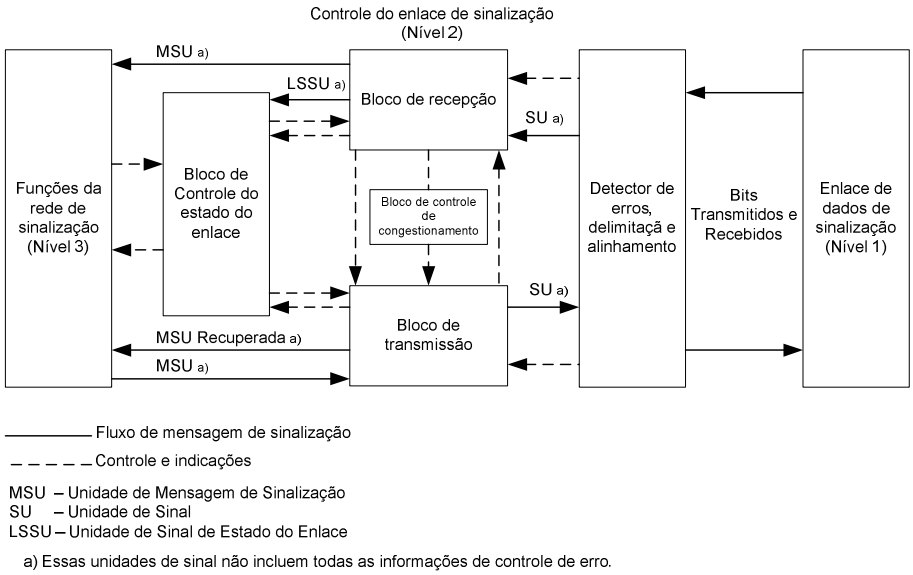


Figura 17. Interação entre os blocos funcionais do MTP-2.

FONTE: (ITU-T, 1993a)

- A delimitação da SU é feita através da inclusão de um valor padrão de 8 *bits* (01111110) no início e no fim da SU. Ele é denominado pelo termo em inglês *flag*. Quando o receptor identificar uma sequencia de 7 ou mais *bits* em 1 ou receber uma mensagem maior do que o tamanho limite, será gerada uma perda de alinhamento. Essa perda de alinhamento alterará o modo de funcionamento do monitor da taxa de erro de SU.
- A detecção de erro na mensagem é feita através da verificação de um conjunto de 16 *bits*, denominado de *bits* de verificação (*check bits* – CK), enviada em toda unidade de sinal. Ele é gerado por um código corretor de erros específico tendo como base de cálculo os dados da mensagem. Ao receber uma mensagem, é feita uma comparação entre o CK calculado pelo receptor, baseado nos dados recebidos, com o CK da mensagem. Caso haja divergência entre os valores, a mensagem é descartada e a indicação de erro é gerada.
- A recomendação Q.703 especifica dois métodos para correção de erros: um é o método básico e o outro é o método de retransmissão preventiva cíclica. O método básico consiste em enviar uma mensagem e armazená-la até que uma confirmação de recepção positiva seja recebida. Se a confirmação positiva for recebida, a mensagem armazenada será descartada e considerada como transmitida com sucesso, caso uma confirmação negativa seja recebida, a mensagem será retransmitida. Esse

método é aplicado quando o meio de transmissão utilizado é o terrestre com um atraso de propagação de no máximo 15 ms.

- O método de retransmissão preventiva cíclica consiste em armazenar uma SU após o envio até que uma confirmação positiva seja recebida. Enquanto a confirmação positiva não for recebida e não houver nenhuma SU nova para ser transmitida, as SUs não confirmadas serão retransmitidas ciclicamente. Nesse método não existe confirmação negativa. Se a quantidade de SU sem confirmação atingir um certo limite, a transmissão de novas SUs será suspensa e as SUs sem confirmação serão retransmitidas ciclicamente até que elas sejam reduzidas. Esse método é aplicando quando o tempo de propagação no meio de transmissão terrestre é superior a 15ms ou em transmissão via satélite.
- O procedimento de alinhamento inicial ocorre em duas situações: no momento da primeira inicialização do enlace ou quando é feita uma restauração de alinhamento após uma falha no enlace. O procedimento consiste na troca de mensagens de estado do enlace entre os dois pontos de sinalização durante um determinado tempo de prova. Se a troca de mensagens ocorrerem conforme o esperado, o enlace será considerado alinhado e estará pronto para a troca de mensagens entre os pontos de sinalização.
- A supervisão de erros no enlace de sinalização é feito através do monitor de taxa de erro da unidade de sinal (*signal unit error rate monitor* - SUERM) e do monitor de taxa de erro de alinhamento (*alignment error rate monitor* - AERM). O SUERM é usado quando o enlace já está alinhado e indica a taxa de erros detectados nas mensagens de sinalização trocadas entre os pontos de sinalização da rede e será utilizado como um dos critérios para configurar o enlace fora de serviço. O AERM é usado quando o enlace não está alinhado e indica os erros ocorridos durante o procedimento de alinhamento inicial de enlace. Ele será utilizado para determinar se o alinhamento pode ou não ser aprovado, ou seja, se a máquina de estado do enlace pode ir para o estado Em Alinhamento.

- O bloco de controle do estado do enlace (*link state control*) é responsável em controlar o estado do enlace bem como fornecer as diretivas para as outras funções do enlace de sinalização (MTP-2).
- O procedimento de controle de fluxo é iniciado quando um congestionamento de mensagens é detectado pelo lado do receptor de umas das pontas do enlace de sinalização. Esse estado é informado ao transmissor na outra ponta do enlace, através do envio de unidades de sinais de estado de enlace (LSSU). Nesse momento, todas as confirmações de mensagens são suspensas até que o congestionamento seja solucionado. O lado transmissor na outra ponta do enlace receberá periodicamente as LSSU informando o estado congestionado do enlace. Se o congestionamento durar por muito tempo, o lado transmissor na outra ponta do enlace indicará uma falha no enlace.

O nível 3 do MTP (MTP-3) corresponde às funções da rede de sinalização. Ele tem como função principal garantir a transferências das mensagens de sinalização mesmo em situações de falha no enlace ou em algum dos elementos que fazem parte da rede de sinalização e que possam interferir na comunicação. As funções da rede de sinalização estão divididas em duas categorias: tratador de mensagens de sinalização e gerenciamento da rede de sinalização. A Figura 18 apresenta as principais funções da rede de sinalização.

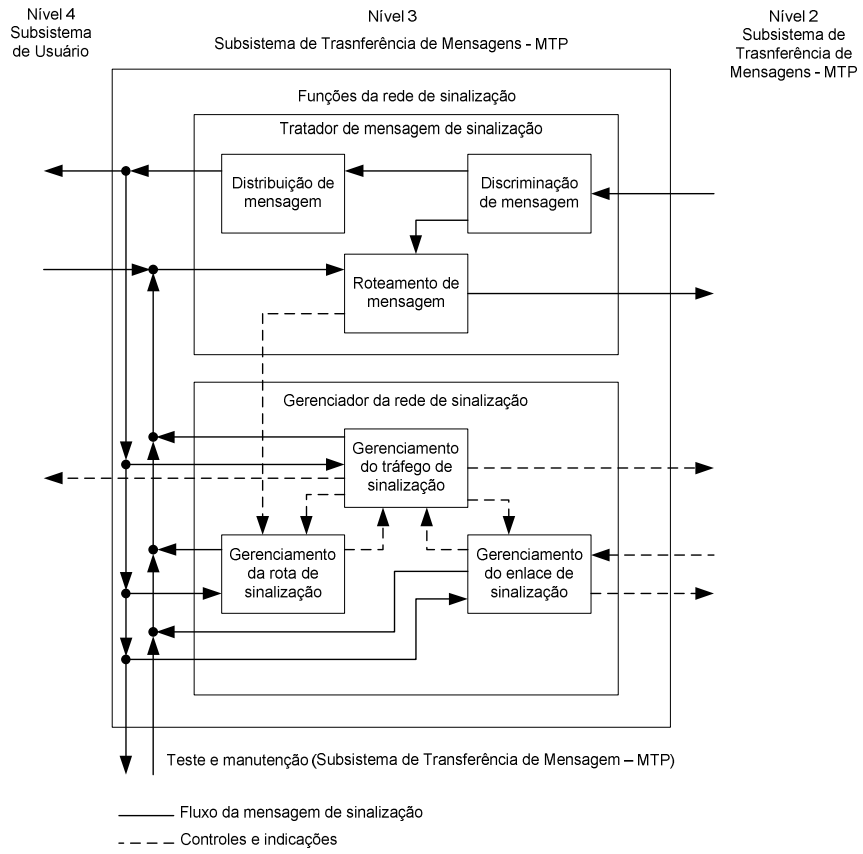


Figura 18. Funções do MTP-3.

FONTE: (ITU-T, 1993a)

O tratador de mensagens de sinalização possui as funções de roteamento, discriminação e distribuição das mensagens de sinalização. Essas funções são necessárias para garantir que uma mensagem enviada pelo subsistema de usuário (*user part* - UP) de origem, seja entregue ao subsistema de usuário de destino. A função de roteamento determina o enlace de sinalização de saída da mensagem de modo que a mesma consiga chegar ao seu destino e utiliza como base para referência os códigos únicos dos pontos de sinalização na rede (*point code*) contidos nas MSUs. A função de discriminação verifica se a mensagem recebida é para o próprio ponto de sinalização ou para outro destino. Caso o ponto de sinalização seja um STP, ele encaminhará a mensagem para o bloco de roteamento de mensagem que executará sua função. Do contrário a mensagem será descartada. A função de distribuição encaminha a mensagem para o subsistema de usuário correspondente pertencente ao próprio ponto de sinalização.

O gerenciamento da rede de sinalização é composto pelas funções de gerenciamento de tráfego, de enlace e de rota de sinalização. Essas funções têm por

objetivo garantir a confiabilidade da rede através da reconfiguração da rede em situação de falha ou do controle do tráfego das mensagens nos enlaces em situações de congestionamento. As falhas na rede podem ocorrer tanto no enlace como também nos pontos de sinalização. Quando uma falha ocorrer, as funções de reconfiguração serão ativadas imediatamente de tal forma que se evite ao máximo a perda de mensagem, duplicação ou alteração da sequencia de mensagens. É possível também que novos enlaces sejam ativados e alinhados para melhorar a distribuição do tráfego de mensagens e contornar os problemas de congestionamento.

3.4 SDL

3.4.1 História

Com a digitalização da rede ocorrida nos anos 60 a 80, os nós de comutação passaram de eletromecânicos e com controle relativamente simples para digitais com controle por programas armazenados executados em computador. Em particular, os protocolos de sinalização usados na intercomunicação entre nós passaram a ser altamente sofisticados. Nesse período o CCITT, como agência reguladora sentiu falta de uma linguagem que pudesse ser usada tanto para descrever e especificar os protocolos e sistemas por ela definidos nas suas recomendações como para que outras agências reguladoras e fabricantes também pudessem usá-los nos seus desenvolvimentos (OLSEN *et al.*, 1994). O CCITT iniciou, portanto, o desenvolvimento da linguagem SDL – *Specification and Description Language* em 1972, tendo sua primeira versão lançada em 1976 seguida por novas versões lançadas em 1980, 1984, 1988, 1996, 1999 e 2011 (BELINA; HOGREFE, 1989) (SDL FORUM).

A SDL foi desenvolvida inicialmente para especificação de sistemas de telecomunicações, mas teve sua utilização também no modelamento de outras aplicações, tais como sistemas de tempo real e sistemas distribuídos (BELINA; HOGREFE, 1989). Ela possui recursos para especificar tanto a estrutura quanto o comportamento de sistemas.

A SDL possui duas formas de representação, uma é a gráfica (SDL/GR – *Graphic Representation*) e a outra é a textual (SDL/PR – *Phrase Representation*). As duas representações são equivalentes. A recomendação Q.703 do ITU-T utiliza a

SDL/GR e, portanto, neste trabalho nos restringiremos à descrição dela. A seguir serão apresentados alguns elementos básicos da linguagem. Para mais detalhes consultar a recomendação Z.100 ITU-T (1988) ou posterior.

3.4.2 Estrutura de um sistema em SDL

A linguagem SDL possui três níveis hierárquicos: sistema (*System*), blocos (*Blocks*) e processos (*Process*). Além de abstrair informação não relevante no nível em que se está trabalhando, essa hierarquização permite acompanhar subdivisões funcionais, criar módulos de tamanhos adequados e reusar especificações já existentes.

O sistema em SDL é composto por um ou mais blocos que contém um ou mais processos (ou subestruturas que podem conter outros blocos). Os processos funcionam de forma autônoma e concorrente e se comunicam através de troca sinais. Eles definem o comportamento do sistema e utilizam como base uma máquina de estados finita estendida (MEFE). A MEFE se difere da máquina de estado finita porque além de armazenar os estados ela manipula e armazena variáveis (SINNOTT; HOGREFE, 2001). A Figura 19 mostra um exemplo de sistema especificado em SDL nos três níveis da hierarquia.

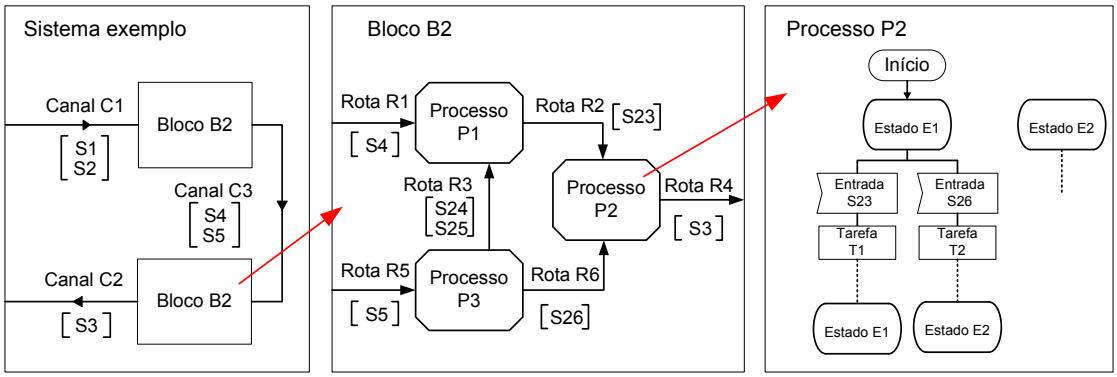


Figura 19. Exemplo de um sistema em SDL nos três níveis da hierarquia.

A interface de comunicação entre os blocos é denominada de canal (*channel*) e a interface entre os processos é denominada de rota de sinais (*signal route*). Tais interfaces são definidas formalmente o que permite que o desenvolvimento seja realizado de forma paralela por equipes independentes, garantindo consistência entre partes diferentes do sistema.

3.4.3 Elementos básicos utilizados na descrição dos processos

As máquinas de estados definidas nos processos são formadas por estados e transições. Uma transição ocorre quando o processo recebe um sinal dentre os que fazem parte da lista de sinais esperados no estado em que a máquina se encontra. Na transição várias ações poderão ser tomadas, por exemplo: execução de tarefas, desvio de sequência de execução e geração sinais de saídas. Essas ações são definidas por símbolos no SDL/GR. A Figura 20 mostra os principais símbolos utilizados para descrever um processo.

Símbolo	Descrição	Símbolo	Descrição
Comment	Utilizado para inserir notas de determinadas partes do sistema.	Output	Indica o envio de um sinal para outro processo.
Text extenssion	Contém textos associados aos símbolos a qual está conectado.	Internal Output	Indica o envio de um sinal a ser utilizado no mesmo processo.
Start	Indica o início de um processo.	Task	Executa uma tarefa. Por exemplo: atribuição de um valor a uma variável.
State	Simboliza o estado do processo.	Save	Indica que o sinal recebido que não faz parte da lista de sinais que ativam uma transição no estado seja armazenada novamente na fila de entradas.
Input	Indica a espera de um sinal vindo de outro processo para gerar uma transição.	Decision	Utilizada para determinar um caminho entre dois ou mais de acordo com o resultado da pergunta.
Internal Input	Indica a espera de um sinal gerado pelo processo ao qual ele pertence.	Connector	Utilizado para indicar a continuação de um fluxo do diagrama quando o mesmo precisa ser dividido.

Figura 20. Principais símbolos utilizados na descrição de um processo.

A Figura 21 mostra um exemplo MEFE que descreve um processo. Ele possui dois estados, três entradas e uma saída. Esse processo, quando iniciado, aguarda o recebimento do sinal **Inicia/Para**. Ao receber esse sinal, a variável **X** será incializada com o valor 0 e a máquina irá para o estado **Contando**. Nesse estado, a variável **X** será incrementada ou decrementada de acordo com o recebimento dos sinais **Incrementa** e **Decrementa**. Se o valor de **X** atingir um valor maior que 10, uma saída será enviada ao processo de destino e a máquina voltará para o estado **Parado**. Se o sinal **Inicia/Para** for recebido enquanto a máquina estiver no estado **Contando**, a máquina irá para o estado **Parado**.

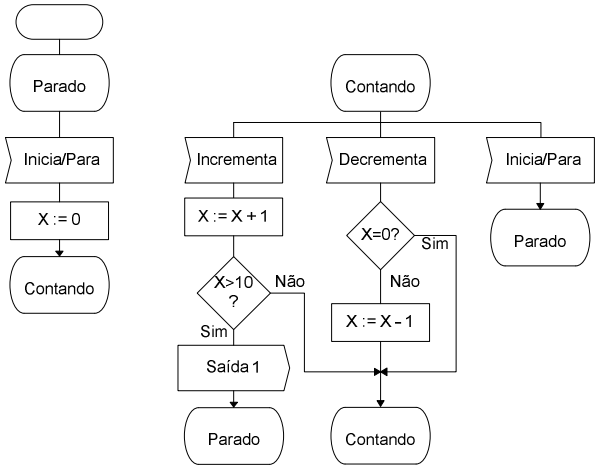


Figura 21. Exemplo de uma MEFE de um processo.

3.4.4 Comunicação

Conforme mencionado anteriormente, os processos funcionam paralelamente e se comunicam por meio de sinais. Essa característica aumenta a possibilidade de um processo receber um sinal durante a execução de uma transição ou mais de um sinal simultaneamente. Por esses motivos, a recomendação Z.100 define o uso de uma fila de entrada do tipo FIFO (*first-in first-out*) para cada processo. O armazenamento deve ser feito respeitando a ordem de chegada dos sinais e nos casos de simultaneidade a atribuição da ordem será aleatória. Os sinais possuem identificação única e podem ou não conter parâmetros (ITU-T, 1988b).

O processo, estando em um estado, verifica se tem sinais na fila de entrada. Se houver um sinal disponível e o mesmo fizer parte da lista de sinais verificados nesse estado, ele será consumido. Caso contrário, o sinal será descartado e o próximo sinal da fila será verificado. Se não houver nenhum sinal na fila, o processo permanecerá no mesmo estado até a chegada de um sinal válido (ITU-T, 1988b). A Figura 22 mostra um exemplo da operação da fila de entradas.

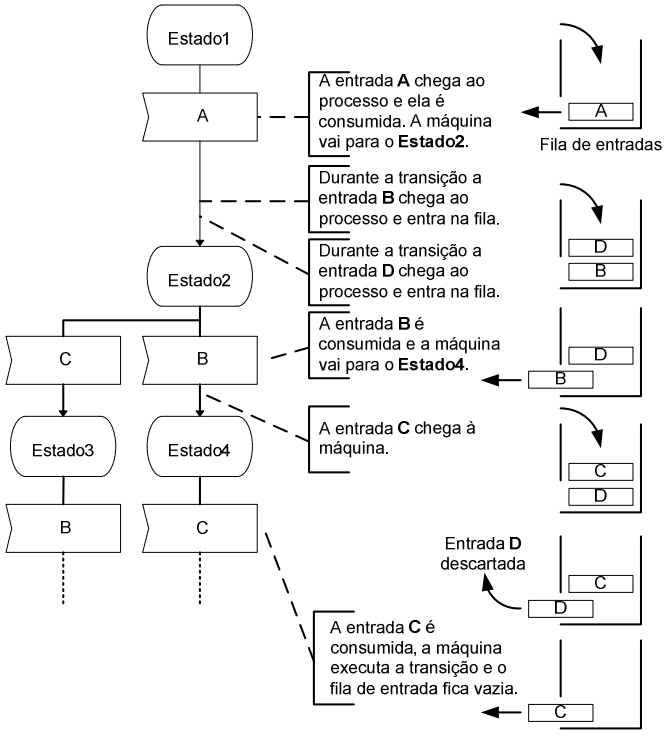


Figura 22. Exemplo de operação da fila de entradas.

Inicialmente a máquina encontra-se no **Estado1**. Ao receber o sinal **A**, a máquina o consome e vai para o **Estado2**. Durante a transição, os sinais **B** e **D** chegam, nessa ordem, e são armazenados na fila de entradas. Ao chegar no **Estado2**, a máquina verifica o próximo sinal na fila. Como o sinal **B** é esperado nesse estado, ele é consumido gerando uma transição para o **Estado4**. Durante a transição, o sinal **C** chega e é armazenado na fila. No **Estado4**, a máquina verifica a fila e encontra o sinal **D**. Como ele não é esperado nesse estado, o sinal é descartado. O próximo da fila é o sinal **C**. Este gera uma transição na máquina e a mesma continua a execução das tarefas.

Há situações em que não se deseja descartar o sinal que não é esperado em um determinado estado para ser consumido em outro. Nesses casos é possível utilizar um símbolo para salvar o sinal novamente na fila para ele ser consumido no próximo estado. O símbolo é o **Salvar** (*Save*). A Figura 23 mostra um exemplo de aplicação deste símbolo.

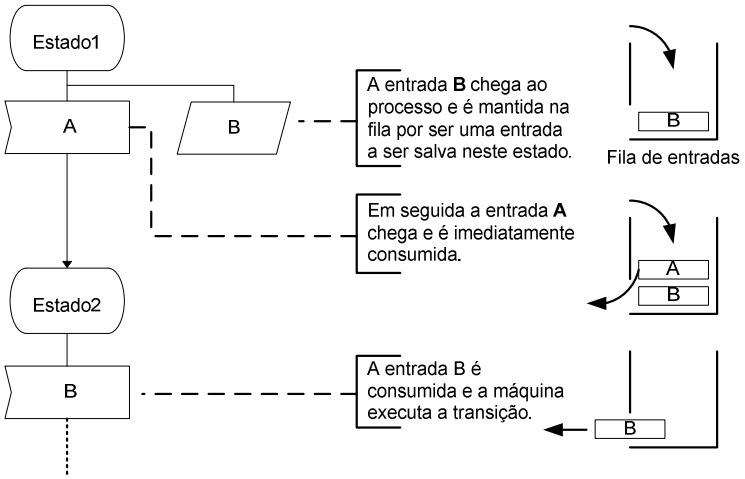


Figura 23. Exemplo do uso do símbolo Salvar.

Ao receber o sinal **B**, a máquina não o consome porque ele está configurado para ser salvo nesse estado. Portanto, a entrada permanece na fila de entradas e a máquina fica aguardando um novo sinal. Ao receber o sinal **A**, o mesmo é consumido imediatamente gerando uma transição para o **Estado2**. Nesse estado, o próximo sinal na fila é o sinal **B**, salvo no estado anterior. Ele é consumido e a máquina executa a transição.

4 METODOLOGIA

4.1 ESCOPO

Conforme descrito anteriormente, faz parte dos objetivos deste trabalho de dissertação, o desenvolvimento de um projeto em linguagem de descrição de *hardware* (VHDL – *Very High Speed Hardware Description Language*) do nível 2 do Subsistema de Transferência de Mensagens da SS7 – MTP2. O MTP2 é uma das camadas da SS7 sendo que cada camada é especificada funcionalmente por uma recomendação do ITU-T (1993a). Em particular, a recomendação Q.703 descreve as funcionalidades do MTP2.

A linguagem SDL, desenvolvida pelo próprio ITU-T (1988) é usada extensivamente nas recomendações dessa agência reguladora, em particular na especificação de sistemas de tempo real, tais como os vários protocolos por ela definidos. Geralmente tais especificações são compostas de textos, figuras gerais e diagramas SDL.

Os trabalhos de Bonatti e Figueiredo (1995), Svantesson *et al.*(1998) e Horn *et al*. (1999) descrevem metodologias utilizadas para a conversão de sistemas descritos e especificados em linguagem SDL para soluções *hardware* implementadas em VHDL. Na maioria dos casos, esses trabalhos tem a preocupação de desenvolver uma solução genérica e frequentemente implementada em algum aplicativo que automatiza essa conversão. No caso particular deste trabalho, no entanto, o foco foi em atender as seguintes premissas:

a) Ter uma solução o mais eficiente possível em termos de otimização dos recursos dos componentes de lógica programável. Isso permite que uma maior quantidade de instâncias do MTP-2 possam ser implementadas num mesmo componente programável;
b) Ter uma solução "a prova de futuro" (*future proof*), ou seja, que consiga continuar a ser utilizada mesmo quando o fornecedor ou a tecnologia de componentes for descontinuada e substituída, ou quando as ferramentas e aplicativos de desenvolvimento forem substituídos.

Dessa forma, optou-se por não utilizar nenhum aplicativo comercial ou externo de conversão, e sim utilizar uma forma padrão de realizar essa conversão, procurando uma solução o mais eficiente possível. A pesquisa de trabalhos correlatos foi importante, pois, permitiu aproveitar ideias já experimentadas e juntá-las num conjunto de regras específicas para este trabalho.

4.2 SDL NA ESPECIFICAÇÃO DO MTP-2

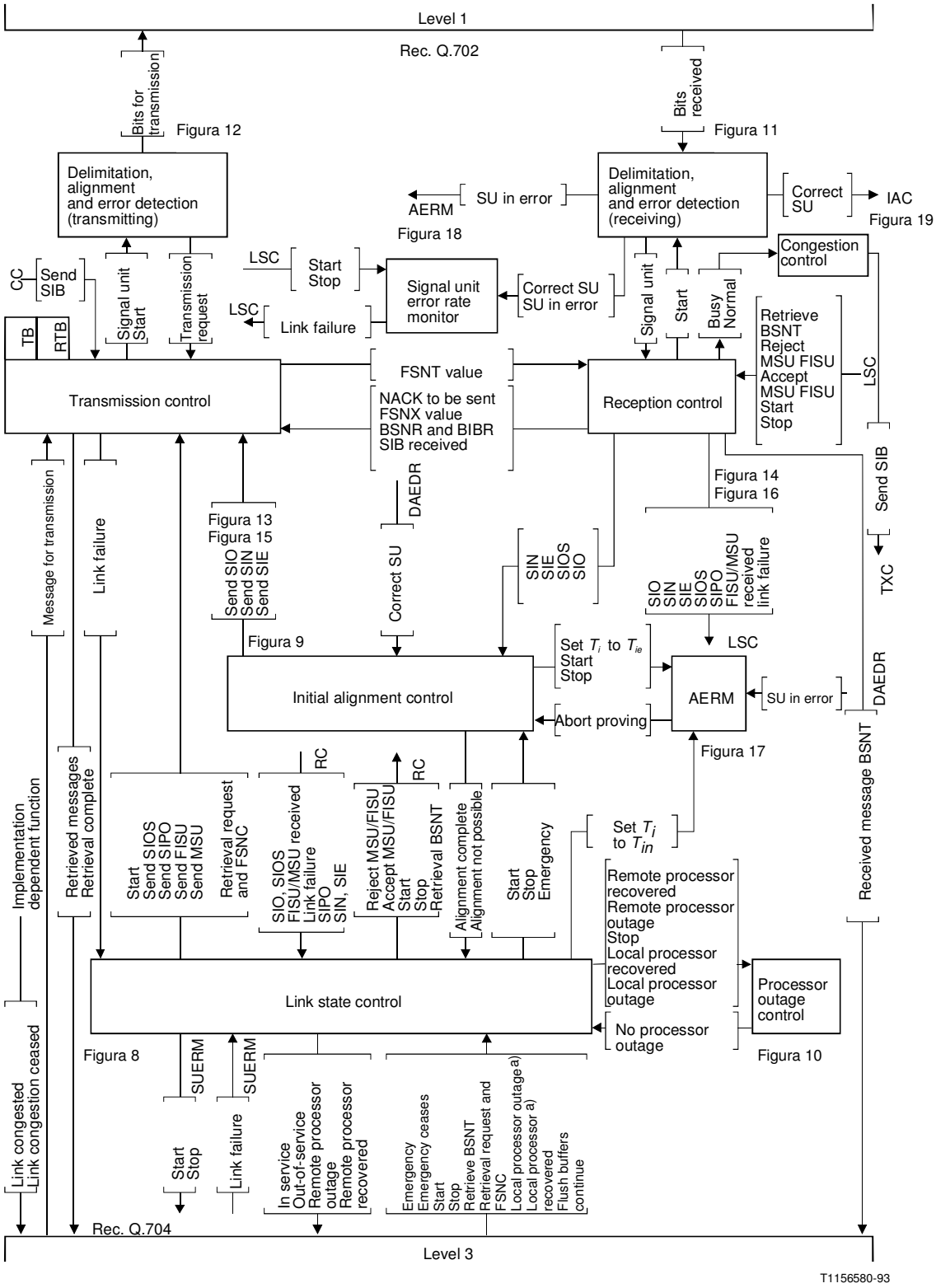
A Figura 24 apresenta o diagrama SDL de mais alta hierarquia do sistema MTP2 que é composto por blocos funcionais. Para cada bloco existe um diagrama que define o seu comportamento denominado de processo (*process*). São eles:

- *Link State Control* (LSC);
- *Initial Alignment Control* (IAC);
- *Reception Control* (RC);
- *Transmission Control* (TXC);
- *Processor Outage Control* (POC);
- *Congestion Control* (CC);
- *Alignment Error Rate Monitor* (AERM);
- *Signal Unit Error Rate Monitor* (SUERM);
- *Delimitation, alignment and error detection in transmission* (DAEDT);
- *Delimitation, alignment and error detection in reception* (DAEDR).

Os processos são máquinas de estados finitas estendidas e são detalhados na Q.703 através de descrição por texto e também por diagramas.

É possível dividir o MTP2 em duas partes de acordo com as características funcionais: MTP2 com as funções de alto nível (MTP2-H) e MTP2 com funções de baixo nível (MTP2-L).

Os blocos DAEDT e DAEDR atuam sobre os pacotes de dados (denominados mensagens) ao nível de bit garantindo a correta transmissão e recepção dos dados. Por exemplo, ao calcular tanto na transmissão como na recepção o CRC (*Cyclic Redunancy Check*), ou ao inserir e retirar bits de preenchimento que impedem que os "*flags*" de abertura e encerramento sejam erradamente reproduzidos. Essas funções são consideradas funções de baixo nível.



a) Somente para opção nacional de bloqueio de processador inativo.

NOTAS
1 – Foram utilizados neste diagrama nomes de mensagens abreviadas (ex. os códigos de origem – destino foram omitidos).
2 – Veja as abreviações e os temporizadores utilizados neste diagrama no item 12.2 da Q.703.

Figura 24. Diagrama do MTP2 evidenciando os blocos funcionais.

(Fonte: ITU-T, 1993a).

Os blocos funcionais LSC, IAC, RC, TXC, POC, CC, AERM e SUERM executam funções de controle que garantem a transmissão e recepção de mensagens na ordem correta conforme determina a recomendação. Essas funções são consideradas funções de alto nível. Este trabalho de dissertação apresenta a implementação da MTP2-H. Cada bloco do diagrama apresentado na Figura 24 possui somente um processo que é uma máquina de estados finita estendida. A descrição desses processos é feita na recomendação Q.703 também por meio de diagramas SDL com o foco na descrição do comportamento.

A metodologia definida para realizar a conversão do sistema em SDL para VHDL foi dividida em etapas de acordo com o diagrama da Figura 25.

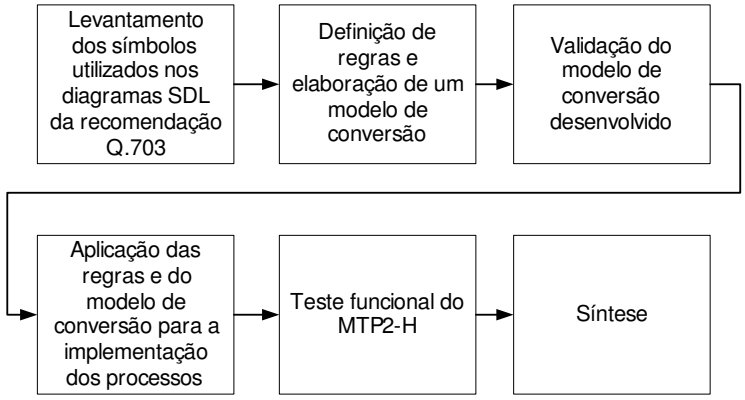


Figura 25. Etapas que compõem a metodologia definida para a conversão do sistema em SDL para VHDL.

A seguir será apresentado o detalhamento de cada etapa da conversão.

4.3 LEVANTAMENTO DOS SÍMBOLOS UTILIZADOS NOS DIAGRAMAS SDL DA RECOMENDAÇÃO Q.703

Nesta etapa iniciou-se a pesquisa de trabalhos e livros relacionados à descrição do SDL, em seguida partiu-se para a pesquisa de trabalhos relacionados à conversão de SDL para VHDL.

O estudo da linguagem SDL foi baseado no artigo de Belina e Hogrefe (1989) e na recomendação Z.100 da ITU-T que descreve a linguagem SDL. O artigo de Belina e Hogrefe (1989) é um tutorial que apresenta uma visão geral do SDL. A partir da leitura dele, o estudo pela recomendação Z.100 ficou mais fácil uma vez que as

recomendações do ITU-T têm como principal objetivo serem completas e precisas e não serem didáticas.

Como a linguagem é abrangente, o estudo completo demandaria muito tempo. Portanto, para tornar o estudo mais objetivo, focando na necessidade do trabalho, realizou-se um levantamento de todos os símbolos (*constructs*) utilizados nos diagramas da recomendação Q.703.

Os diagramas SDL dos processos do MTP2-H utilizam somente os símbolos apresentados na Figura 26.

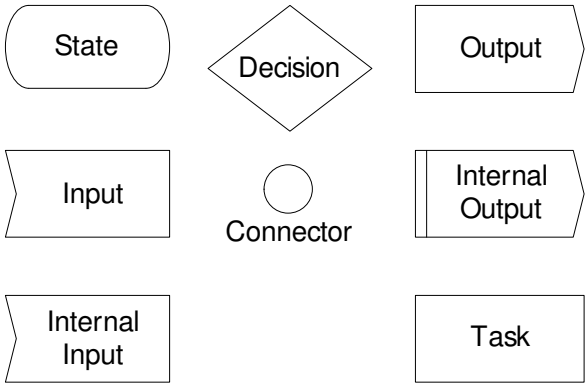


Figura 26. Símbolos utilizados nos diagramas dos processos da Q.703.

A função de cada símbolo foi estudada e em seguida partiu-se para o estudo de cada processo do MTP2-H. O diagrama SDL que define o comportamento de cada processo representa uma máquina de estados finita estendida (MEFE).

Uma vez entendido o funcionamento de cada máquina, foi possível estabelecer algumas regras para a conversão dos diagramas para VHDL.

4.4 DEFINIÇÃO DE REGRAS E ELABORAÇÃO DE UM MODELO DE CONVERSÃO

Conforme visto no capítulo de fundamentação teórica, em SDL um sistema é definido por um conjunto de blocos com um comportamento definido e que se comunicam através de canais de sinais. Cada bloco pode ser formado por um conjunto de processos ou somente um processo.

Classificando um sistema SDL como uma estrutura hierárquica, pode-se dizer que o diagrama do sistema representa o topo da hierarquia e o processo representa o nível mais baixo dela, conforme mostra a Figura 27.

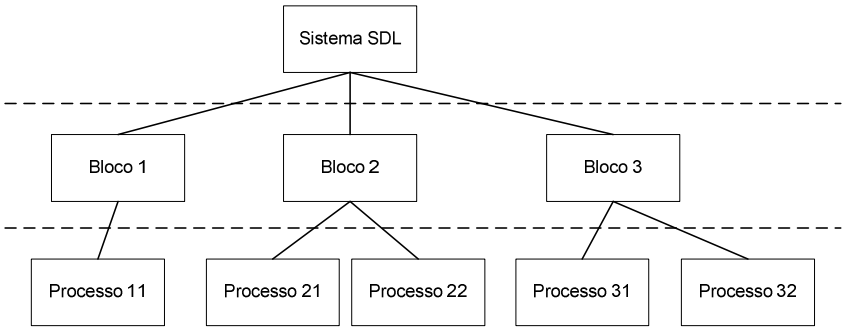


Figura 27. Níveis hierárquicos do diagrama SDL.

Por sua vez, em VHDL, são descritos sistemas digitais que podem ser compostos por subsistemas interconectados, seguindo o mesmo conceito de sistemas hierárquicos. Os subsistemas podem ser formados por um ou mais componentes, onde em cada um é definido um comportamento por meio da descrição e interconexão de circuitos digitais. Nesse caso, pode-se considerar o sistema como sendo o topo da hierarquia e os componentes o nível mais baixo, conforme mostra a Figura 28.

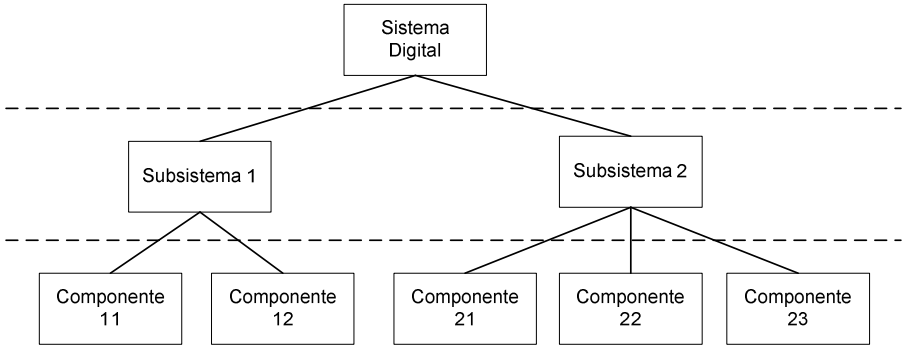
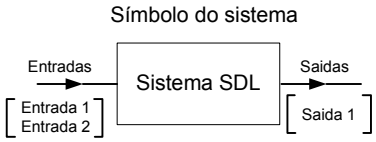


Figura 28. Hierarquia em Sistemas Digitais descritos em VHDL.

Fazendo uma comparação entre a Figura 27 e a Figura 28 é possível observar uma correspondência entre as representações em SDL e VHDL. O sistema em SDL pode ser representado por um sistema digital em VHDL, um bloco por um subsistema e um processo por um componente. Assim é possível estabelecer um modelo em VHDL para cada elemento do sistema SDL. Cada bloco da Figura 28 é descrito em VHDL por meio de uma entidade. As diferenças entre as representações residem no fato de que: nos níveis de sistema e de subsistema, a arquitetura da entidade descreve a interligação de instâncias de subsistemas e componentes e no nível de componente, a arquitetura

descreve o comportamento do bloco através de circuitos digitais combinacionais e/ou sequenciais interligados.

A Figura 29 e a Figura 30 mostram os símbolos de um sistema e de blocos em SDL e as respectivas representações em VHDL.



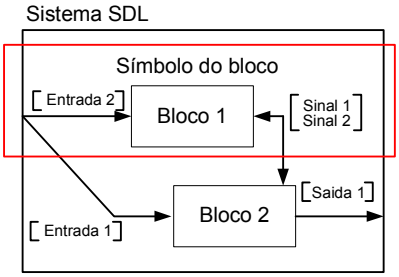
(a)

```
entity Sistema_SDL is
   -- Relação dos sinais de entrada e saída do sistema SDL.
end Sistema_SDL;

archtecture estrutura of Sistema_SDL is
   -- Declaração dos sinais, componentes e tipos.
begin
   -- Instância dos componentes que representam os blocos.
end estrutura;
```

(b)

Figura 29. Modelo para o símbolo de sistema.

a) Símbolo do sistema em SDL. (b) Representação em VHDL do sistema.



(a)

```
entity Bloco1 is
   -- Relação dos sinais de entrada e saída do bloco.
end Bloco1 ;

archtecture estrutura of Bloco1 is
   -- Declaração dos sinais, componentes e tipos.
begin
   -- Instância dos componentes que representam os processos.
end estrutura;
```

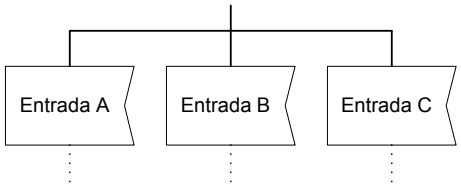
(b)

Figura 30. Modelo para o símbolo de bloco.

(a) Símbolo do bloco em SDL. (b) Representação em VHDL do bloco.

O processo é descrito em SDL com o uso de vários símbolos. Para cada um é possível encontrar uma equivalência em VHDL. No caso deste trabalho, a equivalência ficou restrita aos símbolos utilizados na recomendação Q.703. Esses símbolos estão relacionados na Figura 26. A seguir serão apresentados os símbolos em SDL e suas representações em VHDL.

A verificação do recebimento das entradas em SDL podem ser representadas em VHDL por:



(a)

```
...
case Entrada_s is

   when Entrada_A =>
      -- Ação relacionada à ativação da entrada A.
   when Entrada_A =>
      -- Ação relacionada à ativação da entrada B.
   when Entrada_A =>
      -- Ação relacionada à ativação da entrada C.
   when others =>
      -- Ação relacionada a um valor de entrada desconhecido.

end case;
...
```

(b)

Figura 31. Modelo para a verificação de sinais de entrada.

(a) Verificação do recebimento das entradas em SDL. (b) Representação em VHDL.

A ativação de uma saída em SDL pode ser representada em VHDL por:

(a)

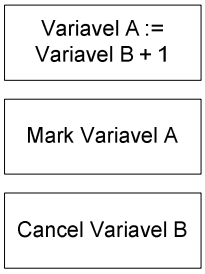
```
...
Saida_1 <= '1' ;-- Valor relativo à ativação da saida 1.
...
ou
...
Saida_1 <= not Saida_1 ;-- Valor relativo à ativação da saida 1.
...
```

(b)

Figura 32. Modelo para ativação de sinais de saída.

(a) Ativação do sinal de saída em SDL. (b) Representação em VHDL.

A execução de uma tarefa em SDL pode ser representada em VHDL por:



(a)

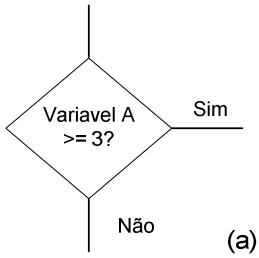
```
Variavel_A := Variavel_B + 1;
ou
Variavel_A <= Variavel_B + 1;

Variavel_A := '1';
ou
Variavel_A <= '1';

Variavel_B := '0';
ou
Variavel_B <= '0';
```

(b)

Figura 33. Modelo para execução de uma tarefa.

(a) Execução de uma tarefa e SDL. (b) Representação em VHDL.

A tarefa de decisão em SDL pode ser representada em VHDL por:



(a)

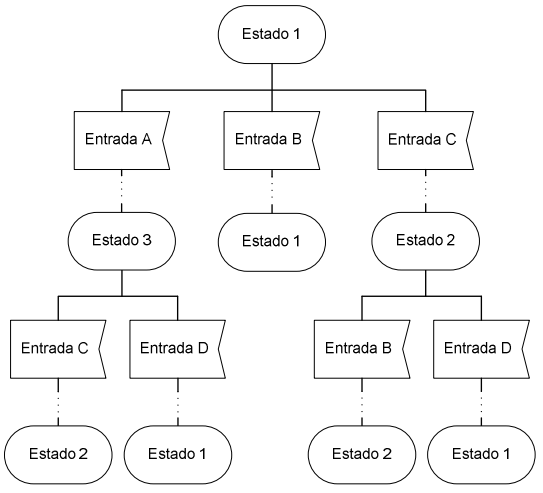
```
...
if (Variavel_A >= 3) then
   -- Ações relativas ao resultado verdadeiro da verificação.
else
   -- Ações relativas ao resultado falso da verficação.
end if;
...
```

(b)

Figura 34. Modelo para execução de uma decisão.
(a) Tarefa de decisão em SDL. (b) Representação em VHDL.

Os símbolos de verificação de entradas e ativação de saídas internas ao processo podem ser representados em VHDL da mesma forma com que é feito para os símbolos de entradas e saídas normais. A diferença é que esses sinais não entram nem saem da entidade onde eles são tratados.

A representação em VHDL de uma máquina de estados descrita em SDL pode ser feita como mostrado na .



(a)

```
...
case prox_estado is
   when Estado 1 =>
      case Entrada is
         when Entrada_A =>
            -- Ações relativas à Entrada A...
            prox_estado <= Estado 3;
         when Entrada_B =>
            -- Ações relativas à Entrada B...
            prox_estado <= Estado 1;
         when Entrada_C =>
            -- Ações relativas à Entrada C...
            prox_estado <= Estado 2;
         when others =>
            -- Ações relativas à entrada desconhecida...
      end case;
   when Estado 2 =>
      case Entrada is
         when Entrada_B =>
            -- Ações relativas à Entrada B...
            prox_estado <= Estado 2;
         when Entrada_D =>
            -- Ações relativas à Entrada D...
            prox_estado <= Estado 1;
         when others =>
            -- Ações relativas à entrada desconhecida...
      end case;
   when Estado 3 =>
      case Entrada is
         when Entrada_C =>
            -- Ações relativas à Entrada C...
            prox_estado <= Estado 2;
         when Entrada_D =>
            -- Ações relativas à Entrada D...
            prox_estado <= Estado 1;
         when others =>
            -- Ações relativas à entrada desconhecida...
      end case;
   when others =>
      -- Ações em caso de um valor de estado indefinido...
end case;
...
```

(b)

Figura 35. Modelo para máquina de estados finita estendida.
(a) Diagrama em SDL e (b) Representação equivalente em VHDL.

Nem sempre é possível implementar a máquina em VHDL de forma idêntica ao diagrama SDL. Existem casos em que é necessário criar estados intermediários ou subestados para permitir a execução de todas as tarefas previstas na transição. Isso normalmente ocorre em situações onde há a necessidade de se transferir pacotes de dados entre processos e também onde é necessário o tratamento de um conjunto de dados.

Uma vez definidos os modelos para cada elemento do sistema SDL pode-se definir como regra:

Aplicação dos modelos em VHDL estabelecidos para os elementos do sistema SDL.

Em SDL, por ser uma linguagem de alto nível, detalhes como frequência de verificação das entradas em cada estado ou o tempo de execução das tarefas em uma transição não são definidos. Esses parâmetros são totalmente dependentes da implementação. Em VHDL, é recomendável que o funcionamento da máquina seja determinado por um relógio, ou seja, que ele seja síncrono. Portanto, pode-se estabelecer como regra:

Uso de um relógio que deve servir como base para o funcionamento da máquina de estados.

Aplicando o modelo de máquina de estados apresentado anteriormente em um modelo de circuito síncrono tem-se uma máquina de estados síncrona que pode ser representada pelo código da Figura 36.

A comunicação entre os processos no SDL é realizada por meio de sinais que podem ou não conter parâmetros. Os sinais possuem identificação única que determinam o seu destino. A ativação do sinal é indicada pelo símbolo **Saída** e a verificação do recebimento de um determinado sinal é indicada pelo símbolo **Entrada**.

```
...
Maq_Estados: process (Clk_i, Reset_i)
begin

   if (Reset_i = '1') then
      -- Ações relativas ao acionamento do sinal Reset…
   elsif rising_edge(Clk_i) then
      case prox_estado is
         when Estado 1 =>
            case Entrada is
               when Entrada_A =>
                  -- Ações relativas à Entrada A...
                  prox_estado <= Estado 3;
               when Entrada_B =>
                  -- Ações relativas à Entrada B...
                  prox_estado <= Estado 1;
               when Entrada_C =>
                  -- Ações relativas à Entrada C...
                  prox_estado <= Estado 2;
               when others =>
                  -- Ações relativas à entrada desconhecida...
            end case;
         when Estado 2 =>
            case Entrada is
               when Entrada_B =>
                  -- Ações relativas à Entrada B...
                  prox_estado <= Estado 2;
               when Entrada_D =>
                  -- Ações relativas à Entrada D...
                  prox_estado <= Estado 1;
               when others =>
                  -- Ações relativas à entrada desconhecida...
            end case;
         when Estado 3 =>
            case Entrada is
               when Entrada_C =>
                  -- Ações relativas à Entrada C...
                  prox_estado <= Estado 2;
               when Entrada_D =>
                  -- Ações relativas à Entrada D...
                  prox_estado <= Estado 1;
               when others =>
                  -- Ações relativas à entrada desconhecida...
            end case;
         when others =>
            -- Ações em caso de um valor de estado indefinido...
      end case;
   end if;
end process Maq_Estados;
...
```

Figura 36. Modelo em VHDL de uma máquina de estados finita estendida síncrona.

Em VHDL é necessário estabelecer um padrão de acionamento de sinal, pois, a verificação de uma determinada entrada depende da forma em que ela foi acionada. O acionamento pode ser realizado por:

- Nível – alto ou baixo;
- Borda – transição do nível baixo para alto (borda de subida) ou transição do nível alto para baixo (borda de descida);
- Transição – mudança de nível, tanto de baixo para alto quanto de alto para baixo;
- Valor binário – valor binário em um barramento de dados.

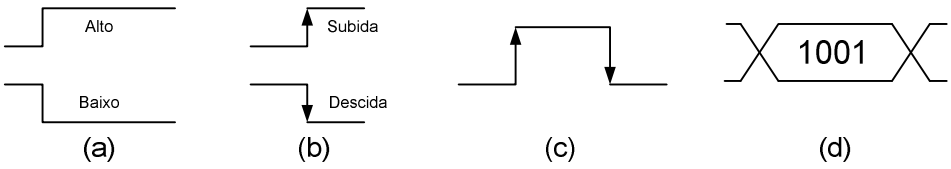


Figura 37. Tipos de ativação de sinais.
(a) Nível. (b) Borda. (c) Transição. (d) Valor binário em um barramento.

Pode-se, portanto, estabelecer como regra:

Definição do padrão de acionamento dos sinais de saída e verificação dos sinais de entrada.

Como indicado anteriormente, o SDL prevê o uso de uma fila de entrada em cada processo para armazenar os sinais recebidos. Na prática é necessário avaliar, para cada processo, o tamanho e até mesmo a necessidade do uso desse recurso. Dependendo do processo é possível utilizar alguns recursos mais simples, como por exemplo, o marcador (*flag* - sinal de um bit) para armazenar a recepção de um sinal.

O marcador funciona como um indicador de recepção de sinal de entrada que ainda não foi tratado. Ele assumirá um valor que indica a recepção de um sinal, por exemplo, valor igual a 1, e o manterá até que o sinal recebido seja consumido pela máquina. Quando um sinal de entrada recebido for consumido pela máquina, o marcador receberá o valor complementar ao da indicação de recepção, por exemplo, 0. Esse recurso é muito útil também nas situações de recepção simultânea de dois ou mais sinais de entrada. Isso porque todos os sinais recebidos serão registrados e permanecerão ativos até que eles sejam consumidos. Porém, uma vez registradas as recepções, não será possível determinar, através dos marcadores, a ordem da recepção. Nesse caso, a solução é a determinação de uma ordem de prioridade entre os sinais. Assim, a máquina fará a verificação das entradas na ordem da prioridade previamente definida.

Em VHDL, o marcador pode ser representado por um *flip-flop* tipo D síncrono com um relógio e com sinal de *reset* assíncrono. A Figura 38 apresenta o código em

VHDL do circuito de marcador onde a verificação da ativação do sinal de entrada é feita por nível alto.

```
Flag: process (Reset_i, Flag_rst_s, Clk_i)

begin

   ---------- Reseta o Flag --------------------
   if Reset_i = '1' or Flag_rst_s = '1' then Flag_s <= '0';

   elsif rising_edge(CLK_i) then
   ---------- Seta o flag ----------------------------
      if Entrada_s = '1' then Flag_s <= '1'; end if;

   end if;

end process Flag;
```

Figura 38. Código que descreve um circuito de marcador em VHDL.

Apesar de ser eficiente, de simples implementação e de utilizar pouco recurso do componente de lógica programável, o uso do marcador é limitado. Ele não é recomendado em processos com a recepção sucessiva de um mesmo sinal, cujos tempos entre elas sejam menores do que o tempo de atendimento das entradas recebidas de maior prioridade. Isso porque as recepções sucessivas vão se transformar em somente uma recepção e as outras serão suprimidas. Nesses casos é recomendado o uso de fila de entrada.

A implementação de uma fila de entrada do tipo FIFO em VHDL demanda recursos de lógica e de memória. O ponto mais crítico está no bloco de memória, pois, dependendo dos tamanhos das filas de entradas de todos os processos do sistema poderá ser necessário o uso de memórias externas. Alguns componentes de lógica programável possuem blocos internos de memórias que podem ser utilizados. Outra alternativa seria o uso de registradores para fazer o papel de memória da fila de entrada. Porém, essas funções são dependentes da ferramenta de compilação, da tecnologia do componente e aplicada somente em casos onde se requer pouco espaço de memória. A escolha entre uma solução ou outra para implementar a parte de memória da fila de entrada depende muito do tamanho necessário na aplicação. E a determinação do tamanho só é possível através de experimentações, pois, depende da frequência de verificação das entradas realizada pela máquina, do tempo entre as recepções dos sinais e da quantidade de sinais de entrada no processo.

Pelo exposto, pode-se estabelecer como regra:

Definição do tipo de solução para armazenamento dos sinais recebidos nos processos, marcador ou fila de entrada do tipo FIFO.

Neste trabalho foi adotado tanto o uso de marcador quanto de fila de entrada. O sistema possui tanto processos simples como complexos, o que exigiu o uso das duas soluções para o armazenamento dos sinais recebidos. Nos processos em que se utilizou o marcador, foi aplicado o modelo apresentado na Figura 38. No caso da fila de entrada foi utilizado um *IP Core FIFO Queue*, liberado para uso, obtido no site da *OpenCores*[2]. Algumas alterações foram necessárias para atender aos requisitos do trabalho.

Para facilitar a interpretação e manutenção dos códigos em VHDL, foram propostas e definidas regras de escrita dos nomes dos sinais, variáveis, constantes e entidades. Entidade é uma designação genérica para sistema, bloco ou processo, procedimento.

Para os nomes das entidades adotou-se como regra o nome abreviado de cada bloco SDL com todas as letras maiúsculas, por exemplo: LSC, IAC, POC, etc.

Os sinais, por serem aplicados em diferentes pontos do código VHDL, tiveram denominações variadas. Já na denominação de constantes e de variáveis foram adotados padrões únicos. A Tabela 2 mostra a relação dos padrões definidos.

Tabela 2. Padrões de escrita do código em VHDL.

Tipo	Padrão de escrita	Descrição
Entidade	<NOME DO BLOCO ABREVIADO>	Nome da entidade
Sinal	<Nome do sinal>_<Origem>**_i**	Sinal de entrada da entidade oriundo de outra entidade do sistema
	<Nome do sinal>**_i**	Sinal de entrada geral da entidade
	<Nome do sinal>_<Destino>**_o**	Sinal de saída da entidade que vai para outra entidade do sistema
	<Nome do sinal>**_o**	Sinal de saída geral da entidade

[2] www.opencores.org

Tipo	Padrão de escrita	Descrição
	<Nome do sinal>**_f_s**	Sinal com função de marcador
	<Nome do sinal>**_rst_s**	Sinal com função de reinicialização do estado do marcador
	<Nome do sinal>**_s**	Sinal de uso geral na entidade
Variável	<Nome da variável>**_v**	Variável de uso nos processos da entidade
Constante	<Nome da constante>**_c**	Constante da entidade

A Figura 39 e a Figura 40 mostram alguns exemplos de aplicação dos padrões de escrita definidos.

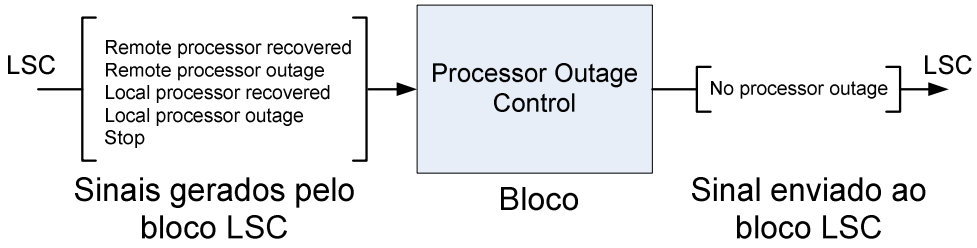


Figura 39. Diagrama SDL de um bloco com seus sinais.

```
...
entity POC is                                          <- Nome da entidade

-- Todos os sinais gerados ou recebidos pelo bloco POC
   port(
      Reset_i : in std_logic;                          <- Sinais do sistema digital
      CLK_i : in std_logic;                               necessários para o
                                                          funcionamento da entidade
      --### Entradas
      Stop_LSC_i                 : in std_logic;
      Local_Proc_Outage_LSC_i    : in std_logic;
      Remote_Proc_Outage_LSC_i   : in std_logic;       <- Sinais de entrada gerados
      Local_Proc_Recov_LSC_i     : in std_logic;          pelo bloco LSC
      Remote_Proc_Recov_LSC_i    : in std_logic;

      --### Saídas
      No_Proc_Outage_LSC_o       : out std_logic      <- Sinal de saída que vai
                                                          para o bloco LSC
   );

end POC;

architecture RTL of POC is

constant Saidas_qtd_c   : integer := 1;                <- Declaração de sinais e
signal Stop_LSC_f_s     : std_logic;                      constantes
-- Declaração de constantes e sinais...

begin

   -- Descrição dos processos que compõem a arquitetura…

end RTL;
```

Figura 40. Exemplos de aplicação dos padrões de escrita do código em VHDL.

Uma vez definidos os padrões para escrita dos nomes das entidades, sinais, variáveis e constantes, pode-se estabelecer como regra:

Uso dos padrões de escrita dos nomes das entidades, dos sinais, das variáveis e das constantes relacionados na Tabela 2.

A Tabela 3 mostra o resumo das regras propostas para a conversão do diagrama em SDL para a descrição em VHDL.

Tabela 3. Resumo das regras estabelecidas para a conversão de um sistema em SDL para VHDL.

Regra	Descrição
1	*Aplicação dos modelos em VHDL estabelecidos para os elementos do sistema SDL.*
2	*Uso de um relógio que vai servir como base para o funcionamento da máquina de estados.*
3	*Definição do padrão de acionamento dos sinais de saída e verificação dos sinais de entrada.*
4	*Definição do tipo de solução para armazenamento dos sinais recebidos nos processos, marcador ou fila de entrada do tipo FIFO.*
5	*Uso dos padrões de escrita dos nomes das entidades, dos sinais, das variáveis e das constantes relacionados na .*

Aplicando as regras em cada elemento de um sistema descrito em SDL obtém-se um modelo equivalente em VHDL que foi utilizado como base na conversão do sistema da MTP2-H.

Para definir um modelo em VHDL de um processo do diagrama do MTP2 apresentado na Figura 24, fez-se um levantamento das funcionalidades necessárias para que a máquina de estados funcionasse de acordo com a descrição. Como resultado, verificou-se que um processo pode ser composto por três blocos funcionais, um de comunicação, um de temporização e um da própria máquina de estados do processo. A Figura 41 mostra o diagrama em blocos com os circuitos digitais necessários para representar um processo em SDL.

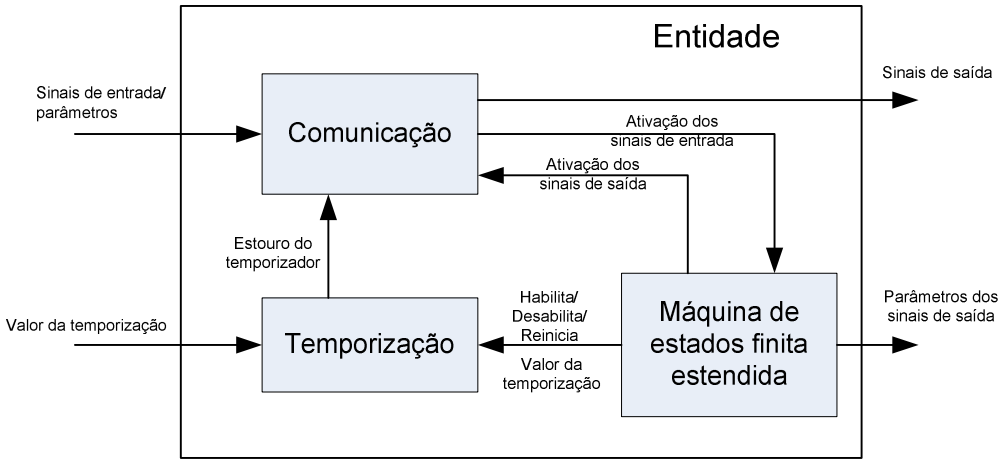


Figura 41. Diagrama em blocos de um processo em SDL

Antes de descrever cada bloco do diagrama é necessário definir as interfaces das entidades que representam os processos e dos blocos funcionais de cada entidade. Conforme visto anteriormente, a ativação de um sinal pode ser indicada em circuito digital de quatro formas: nível, borda, transição ou valor em barramento. Neste trabalho foi definido que as ativações dos sinais entre as entidades que representam o processo em SDL seriam indicadas por nível alto e as ativações dos sinais entre os blocos funcionais dentro da entidade seriam realizadas por transição e valores em um barramento.

Por se tratar de um sistema síncrono, a verificação da recepção do sinal de entrada é feita na borda de subida de cada ciclo do relógio do circuito. Isso significa que a ativação de um sinal deverá durar somente um ciclo do relógio, caso contrário, será identificada recepção consecutiva. Situação similar ocorre com a ativação de sinais indicada por valores em barramento entre os blocos funcionais internos às entidades. A Figura 42 e a Figura 43 mostram exemplos de como são feitas as verificações das ativações dos sinais.

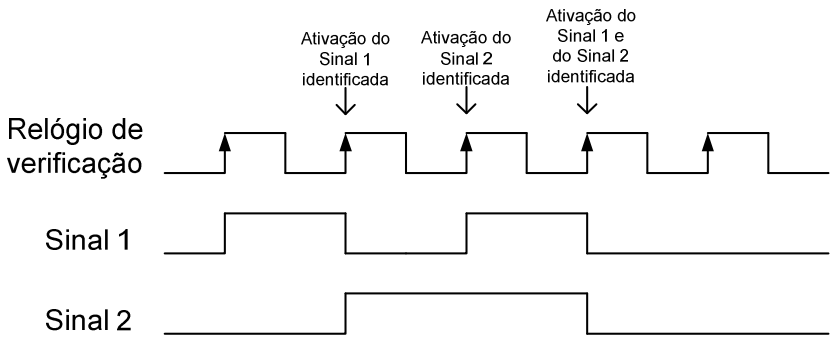


Figura 42. Verificação da ativação dos sinais indicados por nível alto.

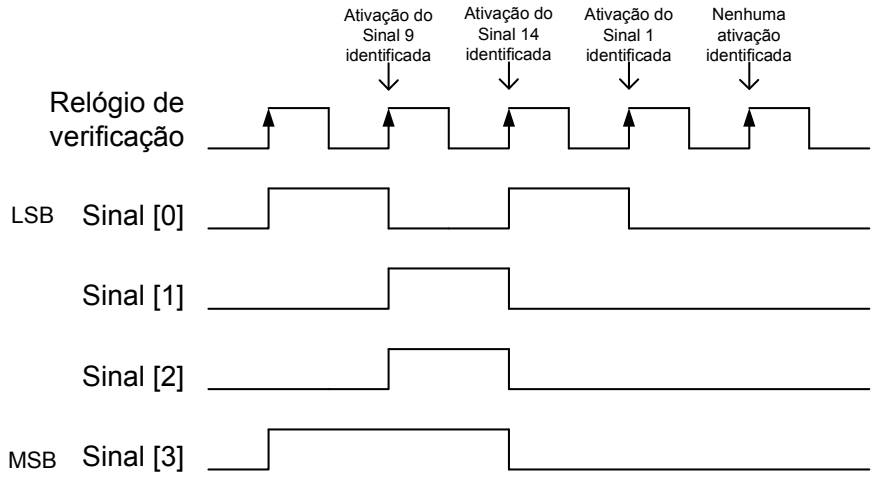


Figura 43. Verificação da ativação dos sinais indicados por valores.

A ativação de sinais por transição foi utilizada para otimizar a implementação da máquina de estados e dos temporizadores. Nesse caso, a informação de ativação não se encontra unicamente nas bordas de subida ou unicamente nas bordas de descida, mas nas duas. Os sinais vão para o bloco de comunicação que ao identificar uma transição gera um pulso positivo, no padrão definido para ativação de sinais de saída de entidade. A otimização vem do fato de ser necessária somente a inversão do sinal na implementação da máquina ou do temporizador para indicar a ativação, caso contrário, seria necessário gerar o pulso em cada circuito o que tornaria a implementação mais trabalhosa. A Figura 44 mostra como fica a relação entre a transição dos sinais e o pulso de saída no bloco de comunicação.

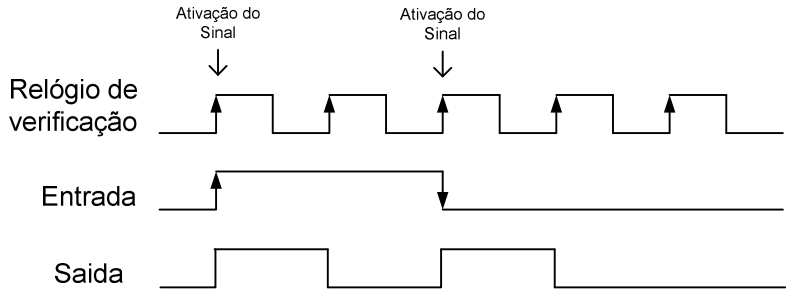


Figura 44. Geração do sinal de saída da entidade.

O bloco de comunicação é responsável pela verificação dos sinais de entrada recebidos e geração dos sinais de saída do processo. As recepções dos sinais de entrada são armazenadas e repassadas para a máquina de estados. Para o armazenamento delas foram adotadas duas soluções: uma com uso de marcadores e a outra com uso de fila de entrada do tipo FIFO. A aplicação de uma ou outra depende do nível de atividade do

processo, conforme visto anteriormente. Para cada solução elaborou-se um modelo para o bloco de comunicação com as mesmas interfaces para manter compatibilidade. A geração dos sinais de saída é realizada a partir da detecção da transição de sinais internos gerados por outros circuitos. A Figura 45 mostra os tipos de sinais recebidos e gerados pelo bloco.

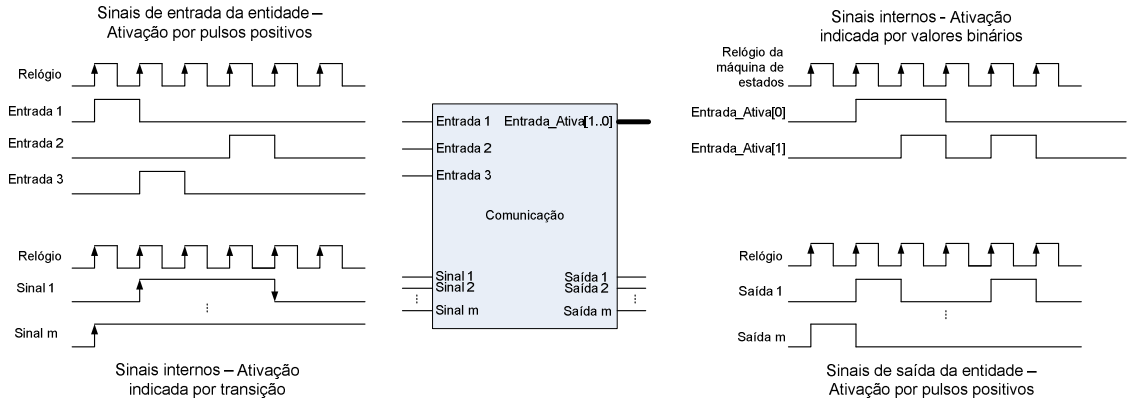


Figura 45. Interface do bloco de comunicação.

O relógio utilizado na verificação dos sinais de entrada, sinais internos e na geração dos sinais de saída do processo possui uma frequência pelo menos duas vezes maior que a frequência do relógio utilizado na máquina de estados. Com essa medida foi possível reduzir em um ciclo de relógio o tempo entre a ativação de um sinal de entrada e a execução da transição referente a ela. A Figura 46 mostra a diferença entre usar o relógio de verificação das entradas igual ou maior ao da máquina de estados em um circuito usando o marcador para armazenar os sinais recebidos. Isso se aplica também no circuito que utiliza fila de entrada. Na avaliação foi considerado que a máquina de estados executa a transição em apenas um ciclo de relógio. A transição da máquina é indicada pela geração do sinal MarcadorX_rst.

O bloco de temporização pode conter um ou mais temporizadores. Cada um gera um sinal indicando a passagem do tempo pré-determinado após sua habilitação. O valor de cada temporização pode ser passado pela máquina durante a execução ou pode ser um valor padrão carregado durante a inicialização do sistema. O sinal que indica o estouro do tempo vai para o bloco de comunicação e entra na fila de ativações dos sinais de entrada do processo. A Figura 47 mostra os sinais que compõem a interface do bloco de temporização.

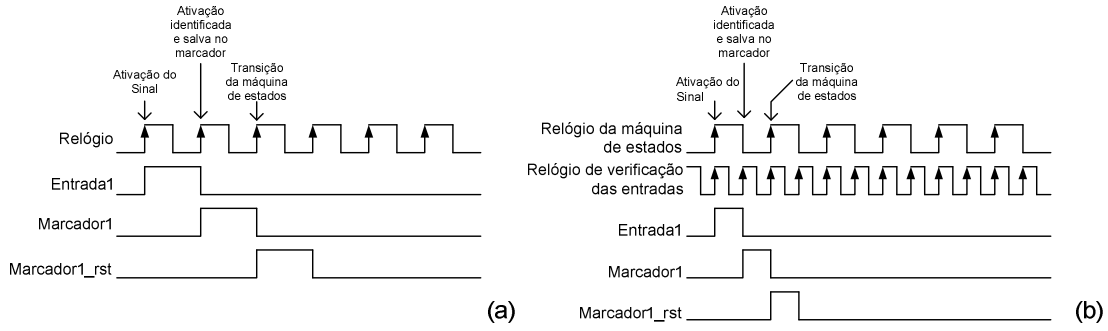


Figura 46. Diferença no uso de relógio com frequências iguais ou diferentes nos circuitos de comunicação e de máquina de estados.

(a) **Relógios com a mesma frequência. (b) Relógio do circuito de comunicação com o dobro da frequência do relógio da máquina de estados.**

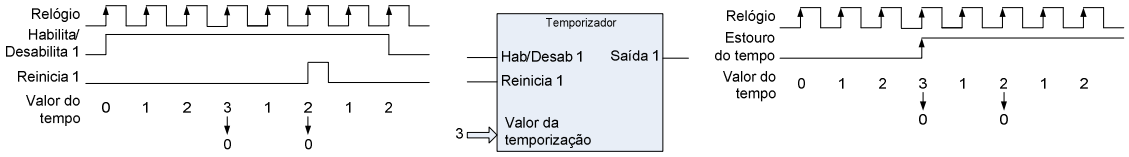


Figura 47. Interface do bloco de temporização.

O bloco da máquina de estados contém o circuito com a máquina que representa o comportamento do processo. Ele recebe as entradas ativas do bloco de comunicação, altera as variáveis internas e gera os sinais de saída de acordo com a transição de estados. A máquina foi implementada conforme o modelo apresentado na Figura 36. Modelo em VHDL de uma máquina de estados finite estendida síncrona.

. A Figura 48 mostra os sinais que compõem a interface do bloco de máquina de estados.

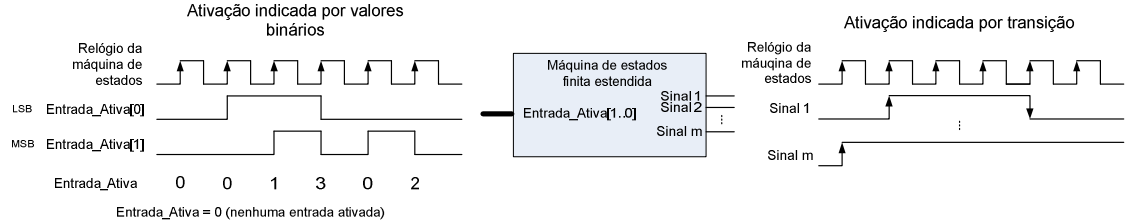


Figura 48. Interface do bloco da máquina de estados finita estendida.

Os blocos de comunicação e de temporização possuem funções fixas com variação somente da quantidade de sinais de entrada e saída. Como eles são comuns a

quase todos os processos do MTP2-H, optou-se por cria-los como componentes e instanciá-los nas entidades que os utilizam. O código em VHDL do modelo de um processo é apresentado no Apêndice 1.

4.5 VALIDAÇÃO DO MODELO DE CONVERSÃO DESENVOLVIDO

A validação do modelo foi realizada por meio de testes funcionais de alguns dos processos implementados utilizando uma ferramenta de simulação. A ferramenta utilizada foi o ModelSim-Altera 6.6d (Quartus II 10.1sp1) Starter Edition e os processos foram: o LSC, o CC e o TXC.

Os testes consistiram em criar um bloco testador descrito em VHDL (*testbench*) que gerasse as ativações das entradas e verificasse as saídas do processo de acordo com o vetor de ativação dos sinais de entrada e o vetor de resultado. A Figura 49 mostra a arquitetura de teste dos modelos.

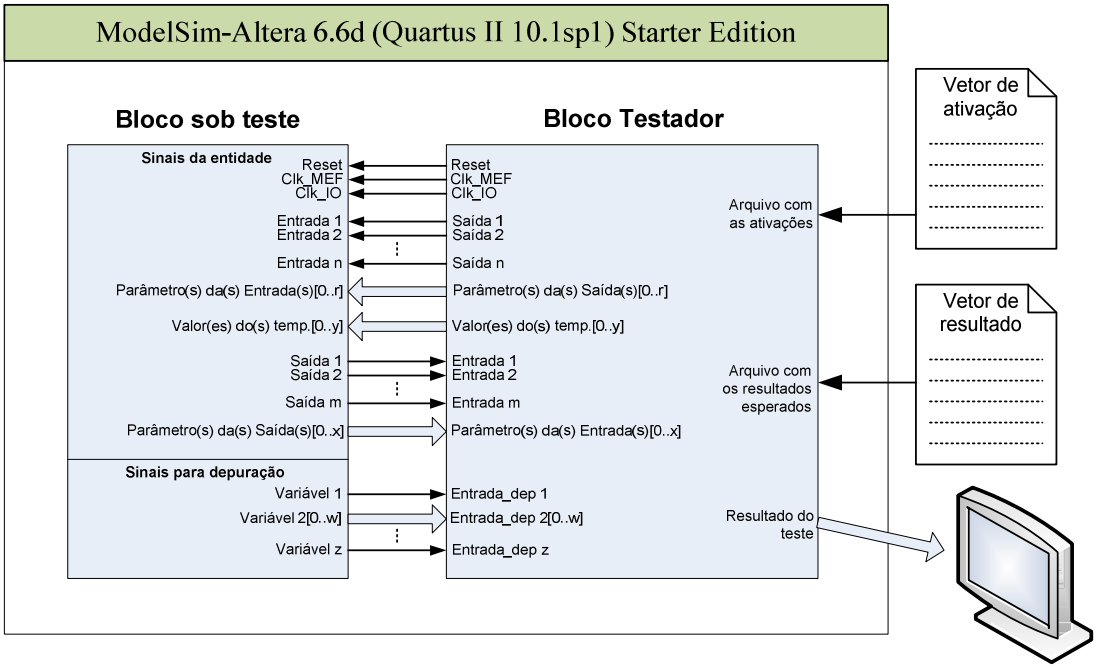


Figura 49. Arquitetura de teste dos modelos em VHDL.

Os dois vetores são arquivos textos e foram gerados a partir de duas tabelas, uma de ativações das entradas e a outra de ativações das saídas e sinais internos. O vetor de sinais recebidos pelo processo foi gerado de forma a levar a máquina a executar todas as transições previstas no diagrama SDL do processo sob teste. O vetor de resultados contém todas as ativações dos sinais de saída do processo e os valores dos

sinais internos conforme as transições da máquina. Cada linha do arquivo com os vetores representa o estado dos sinais em um ciclo do relógio da máquina de estados. Portanto, o bloco testador lê uma linha do arquivo com o vetor de sinais recebidos, gera os sinais correspondentes e compara os sinais de saída e os sinais de depuração com o vetor de resultados a cada ciclo de relógio da máquina de estados. O resultado do teste é apresentado na tela de mensagens da ferramenta de simulação. O fluxograma da validação dos modelos é apresentado na Figura 50.

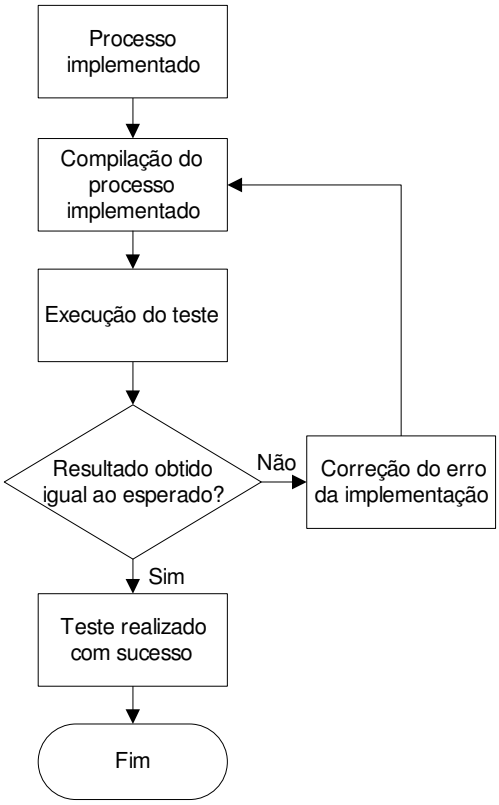


Figura 50. Fluxograma de validação dos modelos.

4.6 APLICAÇÃO DAS REGRAS E DO MODELO DE CONVERSÃO PARA A IMPLEMENTAÇÃO DOS PROCESSOS

Uma vez validado, o modelo de conversão foi aplicado para a conversão dos processos IAC, RC, POC, AERM e SUERM. Em seguida, realizou-se a integração de todos em uma entidade formando o MTP2-H. A interligação dos processos seguiu a descrição mostrada na Figura 24.

Para atender aos requisitos da recomendação Q.703, os blocos RC e TXC necessitam de espaços de memória temporários (conhecidos como *buffer*) para armazenar as mensagens transmitidas e recebidas. A recomendação não descreve como o *buffer* deve ser implementado e nem como será feito o controle das mensagens armazenadas. São funcionalidades totalmente dependentes da implementação. Neste trabalho foram criados mais dois blocos funcionais para realizarem os controles desses *buffers*: controlador do *buffer* de recepção (CtrBuffRX) e o controlador do *buffer* de transmissão (CtrBuffTX).

O CtrBuffRX é responsável em controlar o acesso ao *buffer* de recepção (RB) pelos blocos DAEDR, RC e a MTP-3. Ele controla também o estado do *buffer* gerando sinais, ao bloco RC, que indicam se o mesmo está cheio, quase cheio ou com espaço suficiente para receber mensagens. O *buffer* é do tipo FIFO cujo tamanho não está especificado na Q.703, ficando a critério da implementação. Para efeito de validação definiu-se um espaço de 4.096 bytes que é suficiente para armazenar 15 SUs com o tamanho máximo de 272 bytes (octetos). O tamanho do *buffer* está relacionado com a capacidade que a MTP-3 terá em tratar as SUs recebidas, ou seja, quanto mais rápida for menor poderá ser o *buffer*. Essa avaliação deve ser realizada levando-se em consideração o tráfego de mensagens na rede onde o equipamento for instalado e a arquitetura do mesmo no que se refere à MTP-3.

O CtrBuffTX controla o acesso aos *buffers* de transmissão (TB) e retransmissão (RTB) de SUs. Os *buffers* são acessados pela MTP-3 e pelo TC. A MTP-3 escreve no *buffer* de transmissão as SUs a serem enviadas a outro PS. O bloco TC envia as mensagens armazenadas no TB e as grava no RTB para possível reenvio seguindo as regras do método de correção de erro. A Q.703 prevê o armazenamento de até 127 SUs enviadas sem confirmação de recebimento. Cada SU armazenada possui um número de identificação que é usado no controle de correção de erro de transmissão realizado pelo bloco TC. Neste trabalho foi definida uma faixa dedicada para cada uma das 127 SUs com tamanho de 272 bytes, resultando num *buffer* de 34.544 bytes. O tamanho do TB não está definido na Q.703, mas está diretamente relacionado à quantidade de SUs enviadas pela MTP-3 ao PS de destino. Para efeito de validação deste trabalho foi definido o tamanho de 4.096 bytes que pode ser reduzido ou ampliado dependendo da aplicação.

Considerando os tamanhos dos *buffers* de transmissão, retransmissão e de recepção definidos, uma instância da MTP-2 *High* necessitará de 42.736 bytes da memória de sinalização.

O TXC e o RC possuem também sinais que vão para os blocos DAEDT e DAEDR. Esses blocos já foram implementados por Lima (2012), que denominou de DAED, e já possuem uma interface padronizada. O padrão especificado foi adotado de forma a possibilitar a integração do sistema MTP2-H com o DAED, formando o sistema MTP2.

No Apêndice 2 é apresentado o diagrama em SDL do processo *Congestion Control* (CC) e a descrição em VHDL obtida como exemplo do resultado da conversão.

No bloco CC, assim como nos blocos POC, AERM e SUERM foram utilizados os componentes de comunicação com marcadores. No restante utilizaram-se os componentes de comunicação com fila de entrada.

4.7 TESTE FUNCIONAL DO MTP2-H

Como normalmente feito em projetos realizados em lógica programável, na fase em que os circuitos existem apenas virtualmente no computador, a validação funcional da MTP2-H foi realizada usando vetores de teste e simulação.

Para o teste funcional do MTP2-H foram consideradas duas arquiteturas possíveis de serem utilizadas. Uma consiste no desenvolvimento de blocos que gerem e verifiquem os sinais necessários para o funcionamento do bloco MTP2-H, conforme mostra a Figura 51. A outra consiste em utilizar o bloco DAED implementado por Lima (2012) junto com o MTP2-H para formar o MTP2 e simular o entroncamento de dois pontos de sinalização, conforme mostra a Figura 52.

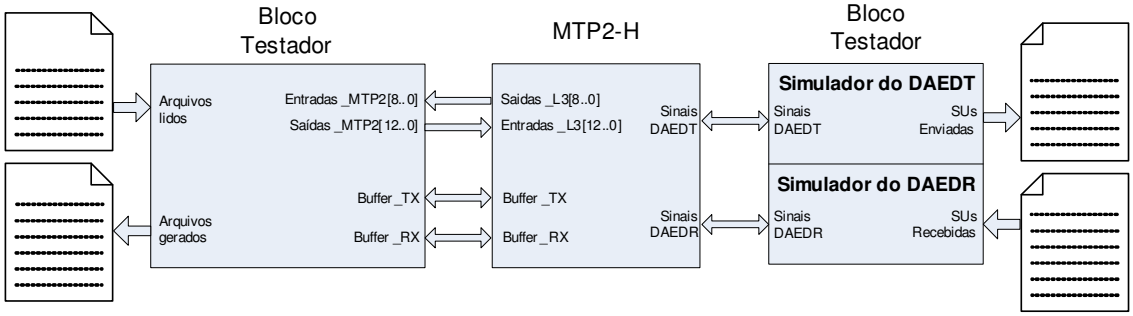


Figura 51. Arquitetura 1 para teste do bloco MTP2-H.

Na arquitetura 1 é necessário desenvolver um bloco simulador do DAEDT, um do DAEDR e o bloco testador. Nessa arquitetura, os blocos Simulador de DAEDT e Simulador de DAEDR simulam somente as ativações dos sinais da interface com o MTP2-H. Além disso, o bloco DAEDT gera um arquivo simulando uma transmissão e o bloco DAEDR lê um arquivo simulando a recepção de mensagens. Os arquivos simulam o outro ponto de sinalização que gera e recebe essas mensagens. Eles são compostos por mensagens completas com os campos de controle de fluxo e tipo de mensagem. O bloco testador simula a interface do bloco MTP2-H com o nível 3, gera e lê arquivos com as ativações dos sinais e conteúdo das mensagens recebidas, simula também os *buffers* de transmissão e recepção de mensagens e também verifica se as mensagens recebidas são iguais às transmitidas.

Uma dificuldade de se utilizar essa arquitetura para teste é a sincronização das mensagens recebidas com as transmitidas. Isso porque cada mensagem recebida deverá conter o reconhecimento da última mensagem transmitida corretamente. A outra é a implementação dos próprios blocos de simulação do DAEDT e DAEDR considerando que esses blocos já foram implementados por Lima (2012).

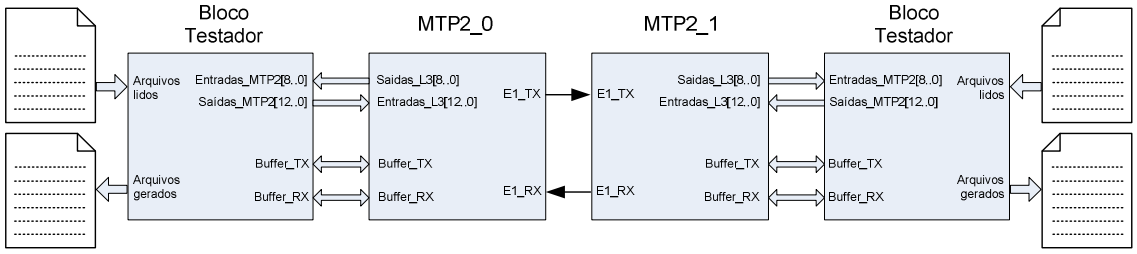


Figura 52. Arquitetura 2 para teste do bloco MTP2-H.

A arquitetura 2 apresenta uma estrutura mais próxima do real, sendo necessário somente implementar o bloco testador. Os blocos DAEDT e DAEDR são aproveitados do trabalho de Lima (2012). O bloco testador, nesse caso, contém as mesmas funções da arquitetura anterior, porém duplicadas, já que são dois sistemas MTP2 ao invés de um. Uma vez interligado todo o sistema conforme a Figura 52, basta gerar os estímulos necessários para que os dois MTP2 troquem mensagens. Os arquivos lidos pelo bloco testador contêm as ativações e as mensagens a serem transmitidas. Conforme a recomendação Q.703, as mensagens devem ser transmitidas na ordem correta, portanto, a validação do funcionamento do MTP2-H é realizada comparando-se o arquivo que contém as mensagens transmitidas pelo MTP2_0 com o arquivo que contém as mensagens recebidas pelo MTP2_1, e vice-versa. Por ser mais próximo da aplicação

real e pela facilidade na implementação, a arquitetura 2 foi adotada para validar o sistema MTP2-H.

A Figura 53 mostra com mais detalhe a arquitetura 2 com os principais sinais que interligam os blocos.

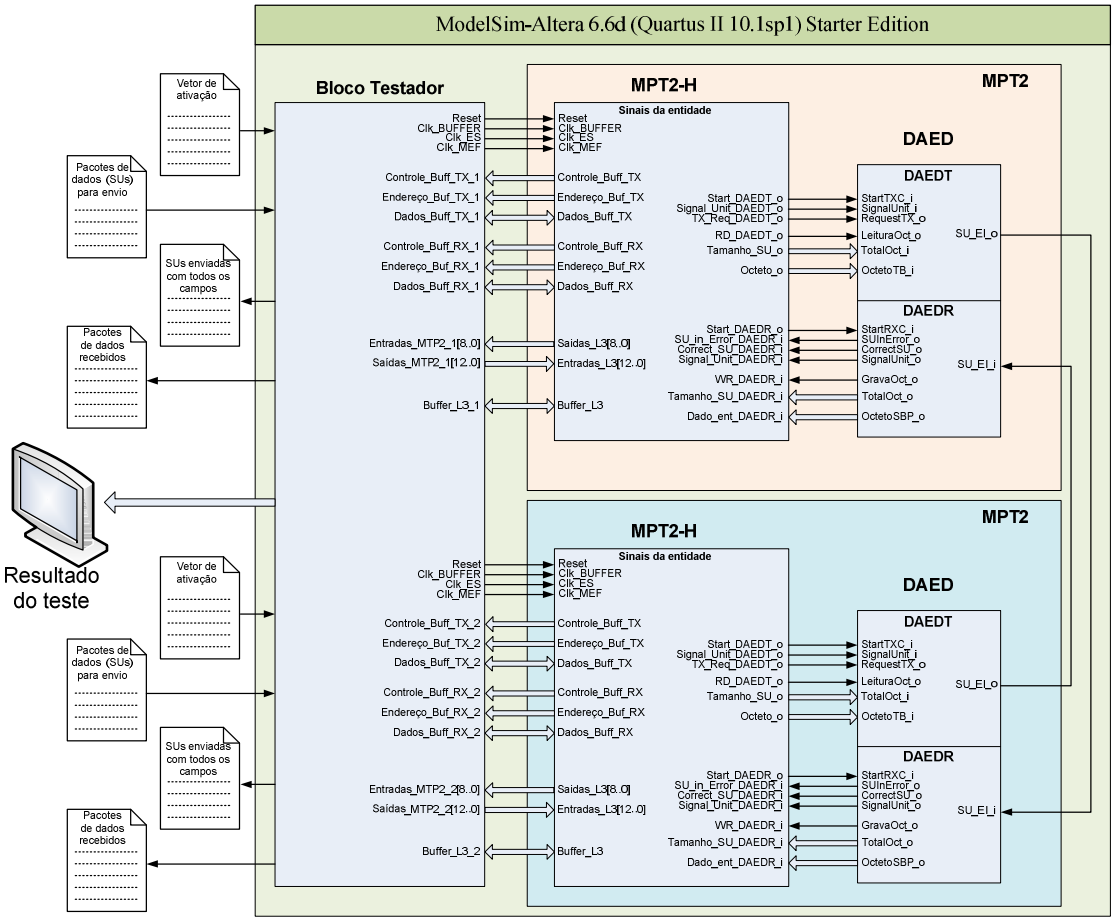


Figura 53. Arquitetura de teste do MTP2-H projetada.

Os arquivos de ativações são gerados a partir de uma tabela da mesma forma que o utilizado na validação do modelo de conversão. Os arquivos com os pacotes de dados a serem transmitidos e recebidos possuem a mesma estrutura com várias mensagens de tamanhos variados. Um *script* para o Matlab® foi desenvolvido para gerar os arquivos com os pacotes de dados onde a quantidade de mensagens e o tamanho de cada uma foi gerada de forma pseudoaleatória. A estrutura do arquivo é apresentada na Figura 54. Cada linha contém um *byte* da mensagem que pode ser o tamanho ou o próprio conteúdo. Os arquivos são comparados por um circuito no bloco testador que apresenta o resultado do teste na tela de mensagens do simulador.

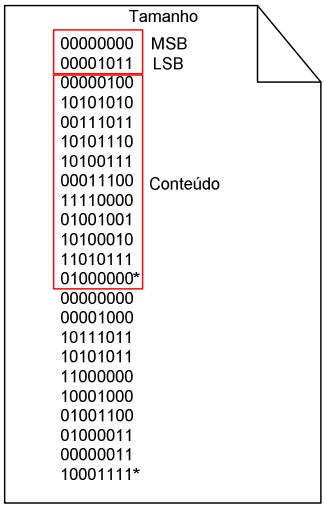


Figura 54. Estrutura do arquivo com os pacotes de dados a serem transmitidos e recebidos.

Os arquivos de mensagens completas (com os campos de controle de erro – FSN, FIB, BSN, BIB e LI) enviadas foram utilizados para depuração nos casos de falha, bem como para verificar se o controle da numeração das mensagens estava sendo realizado corretamente. Eles foram gerados de forma a facilitar a interpretação dos campos de controle que compõem a mensagem completa. Essa tarefa é realizada pelo mesmo circuito que gera o arquivo. No arquivo estão discriminados o tipo da mensagem, os campos de numeração da mensagem transmitida e reconhecida, o tamanho e os dados. A Figura 55 mostra a estrutura do arquivo.

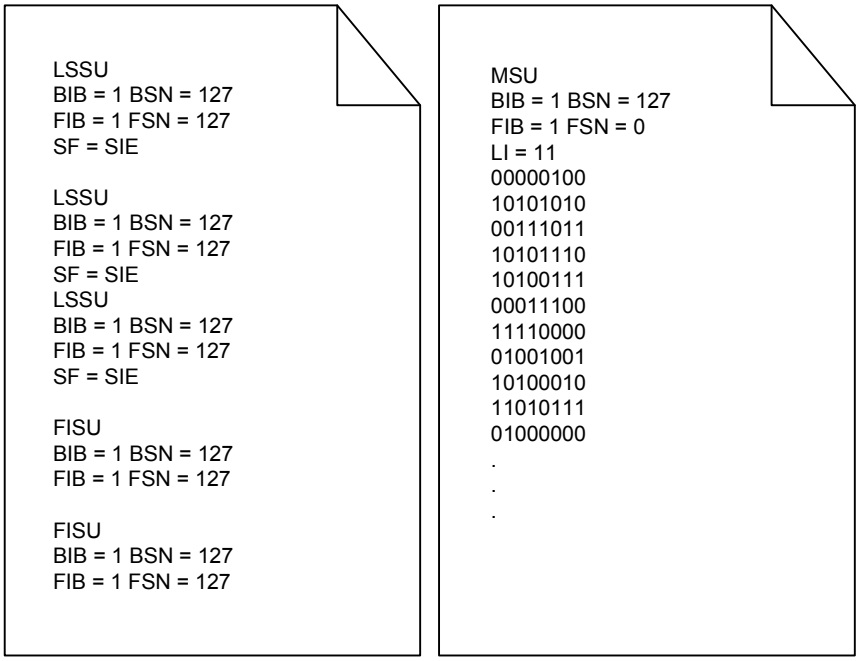


Figura 55. Estrutura do arquivo de mensagens completas enviadas.

Como recurso adicional de depuração utilizou-se também a geração de formas de onda do simulador. Nele é possível visualizar o estado dos sinais de interesse em cada instante da simulação, o conteúdo de variáveis e das memórias com as mensagens transmitidas e recebidas. A Figura 56 mostra uma parte da janela com as formas de onda do teste do MTP2-H.

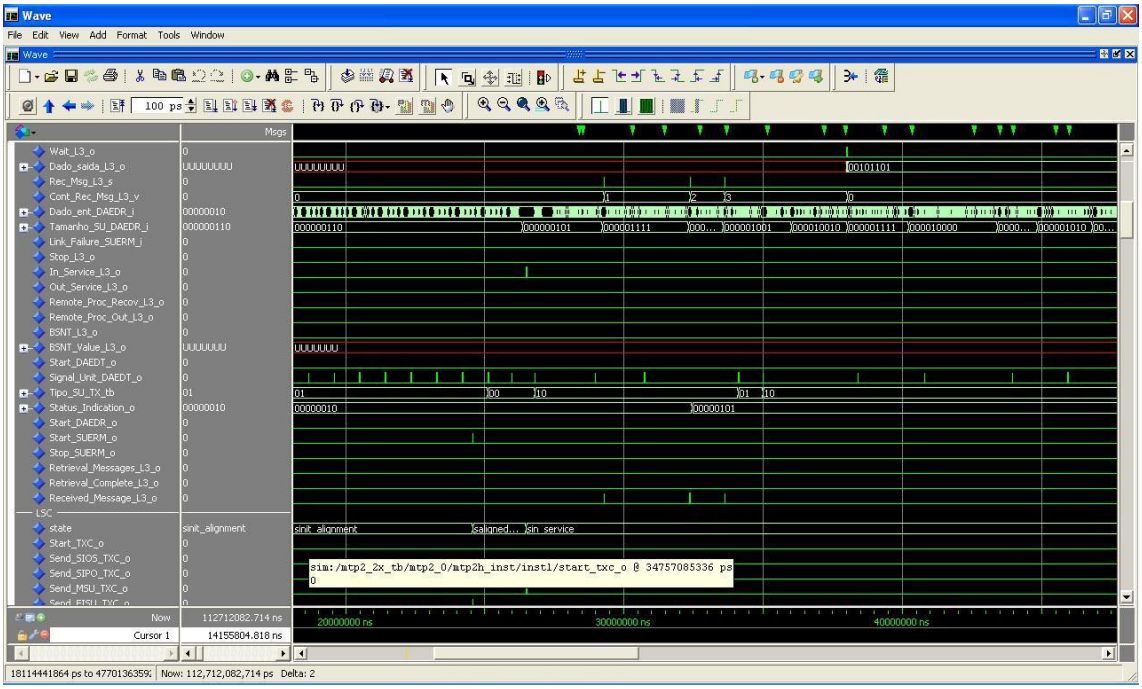


Figura 56. Formas de onda do teste do MTP2-H geradas pelo simulador.

Utilizando o modelo de conversão e todos os recursos para depuração apresentados, conseguiu-se validar a implementação do sistema MTP2-H.

4.8 SÍNTESE

Nesta etapa os códigos em VHDL implementados são compilados em ferramentas de desenvolvimento disponibilizadas por fabricantes de componentes de lógicas programáveis. Esse procedimento é necessário para verificar se os códigos geram (síntese) ou não o circuito desejado e indicam qual a quantidade de recursos do componente são necessários para a aplicação.

Para realizar a síntese da implementação do MTP2-H foram utilizadas duas ferramentas, uma é o Quartus® II 10.1 SP2 pertencente à empresa Altera® e a outra é o Vivado® 2012.4 pertencente à empresa Xilinx®.

5 RESULTADOS E DISCUSSÕES

Duas linhas principais de resultados foram desenvolvidas dentro deste trabalho. A primeira trata-se de uma metodologia, composta por regras e modelos, para a conversão de sistemas descritos em SDL para VHDL. A segunda é a implementação do MTP2-H baseada nessa metodologia.

5.1 METODOLOGIA PARA A CONVERSÃO DE SISTEMAS DESCRITOS EM SDL PARA VHDL

A metodologia foi desenvolvida a partir das descrições em SDL da recomendação Q.703 do ITU-T (1993a), porém a aplicação dela não está limitada somente a essa recomendação. Ela poderá ser utilizada em outras aplicações que possuam características (definidas na metodologia) semelhantes.

O modelo de conversão desenvolvido na metodologia foi dividido em blocos para facilitar a implementação e otimizar o uso dos recursos do componente de lógica programável. Os dois principais blocos funcionais são o de comunicação e o da máquina de estados finita estendida. Essa divisão foi baseada nos trabalhos de Bonatti e Figueiredo (1995) e de Svantesson *et al.* (1998). Eles propõem a criação de um bloco de comunicação independente ao da máquina de estados derivada do processo em SDL como forma de flexibilizar a implementação de acordo com as características das trocas de sinais entre os processos. Para este trabalho foram desenvolvidos dois tipos de bloco de comunicação, um utilizando marcadores e outro utilizando uma fila de armazenamento de ativação do tipo FIFO. As interfaces definidas para esses blocos foram padronizadas para que a adoção de um ou outro não impactasse no código dos outros blocos pertencentes ao modelo de conversão.

5.2 IMPLEMENTAÇÃO DO MTP2-H BASEADA NA METODOLOGIA DESENVOLVIDA

A metodologia de conversão foi aplicada aos blocos LSC, IAC, RC, TXC, CC, POC, SUERM e AERM especificados na recomendação Q.703. Estes, conforme mencionado, compõem o sistema MTP2-H. Adicionalmente foram desenvolvidos também blocos para controle dos *buffers* de transmissão e recepção de mensagens,

necessários para o funcionamento do MTP2. Da forma como foi implementada, uma instância do MTP2-H necessitará de 42.736 bytes de memória para os referidos *buffers*.

A validação de todo o desenvolvimento foi realizada através de simulações utilizando a ferramenta *ModelSim-Altera 6.6d (Quartus II 10.1sp1) Starter Edition*. Para o teste do MTP2-H foram criados cenários que cobrissem o maior número possível de situações encontradas durante o funcionamento real. Foram eles:

1. Inicialização do enlace;
2. *Buffer* de recepção cheio;
3. Falha no enlace (problema físico na linha de transmissão/recepção).

Os testes de inicialização do enlace foram realizados utilizando-se um arquivo com as ativações dos sinais de entrada necessários para que o MTP2 realizasse o procedimento de alinhamento inicial. Esse arquivo foi gerado manualmente, pois deve obedecer a uma sequência de ativações definida.

Os testes de *buffer* de recepção cheio foram realizados utilizando-se um arquivo com as ativações dos sinais de entrada, para que o MTP2 faça a transmissão e recepção das mensagens, em conjunto com um pacote de mensagens gerados por um *script* do Matlab®. O pacote de mensagens utilizado era composto por 32 mensagens com o tamanho variando de 4 a 24 bytes gerados de forma pseudoaleatória. Para facilitar a análise do funcionamento dos blocos envolvidos, o *buffer* de recepção foi temporariamente reduzido para armazenar somente 64 bytes de dados. Com isso, a falha ocorre em um tempo menor do que com o tamanho normal do *buffer* e a depuração se torna mais rápida.

O teste de falha no enlace foi realizado simulando uma falha na conexão do sinal de transmissão do bloco MTP2_0 um tempo após o estabelecimento da comunicação entre os blocos MTP2. Para esse teste foram utilizados os mesmos arquivos de ativação de sinais de entrada e de pacote de mensagens do teste de *buffer* de recepção cheio.

Foram necessárias aproximadamente trinta interações no total para corrigir todos os problemas e validar a implementação. Os principais problemas encontrados

foram as falhas na sequência de transição da máquina ocasionados principalmente por troca do nome dos próximos estados das máquinas dos blocos envolvidos.

Após a validação da implementação, foram realizados mais testes com os cenários citados utilizando 10 pacotes (arquivos) com 100 mensagens cada, que variavam de 4 a 120 bytes. Para todos os pacotes, nenhuma falha foi detectada.

Os testes foram realizados em conjunto com o sistema MTP2-L desenvolvido por Lima (2012) formando o sistema MTP2. Para além dos testes funcionais realizados, a recomendação Q.781 do ITU-T (1993b) indica testes adicionais para fins de homologação.

Os códigos VHDL desenvolvidos neste trabalho não utilizam funções específicas de fabricantes de componentes. Essa foi uma medida adotada para que fosse possível compilá-los e utilizá-los em componentes de lógica programável de qualquer fabricante. Isto garante ao projeto a portabilidade do código, além de lhe conferir a propriedade de ser "a prova de futuro", ou seja, que o torna imune a problemas tais como obsolescência de um componente ou a sua descontinuidade. A Figura 57 e a Figura 58 apresentam os resultados da compilação do sistema MTP2-H nas ferramentas de desenvolvimento dos fabricantes de componentes Altera® e Xilinx®. Em ambas, o projeto foi sintetizado e implementado nos componentes definidos.

Além da portabilidade, outra característica importante a ser mencionada é a escalabilidade, ou seja, o sistema MTP2-H poderá ser instanciado o número de vezes em que for necessário no componente de lógica programável. Para tanto, um bloco de controle de acesso à memória com os *buffers* de transmissão e recepção e uma interface entre o nível 3 (MTP3) e as várias instâncias do nível 2 (MTP2-H) deverão ser desenvolvidos em trabalhos futuros.

Conforme já mencionado, este trabalho de dissertação está sendo desenvolvido no âmbito do projeto HUI. Entre as metas deste projeto encontra-se o desenvolvimento de uma placa que possui como uma de suas funções a execução do nível 2 do Sistema de Sinalização por Canal Comum número 7 (MTP2). Essa placa já foi desenvolvida e está preparada para tratar 4 enlaces de sinalização por canal comum. A parte do MTP2

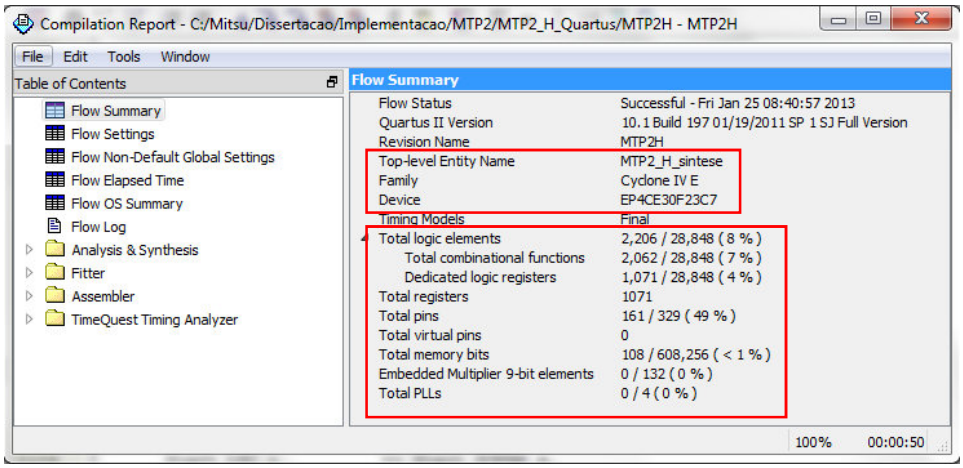


Figura 57. Resultado da compilação e síntese do MTP2-H na ferramenta de desenvolvimento Quartus II 10.1 SP1 da Altera®.

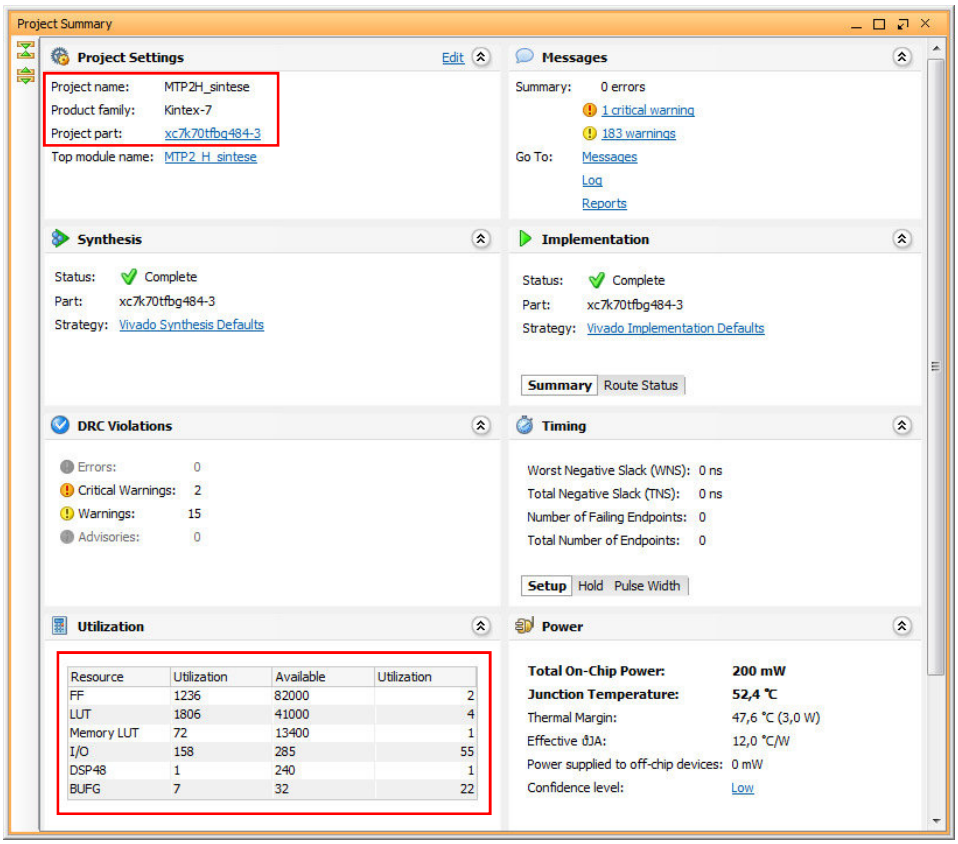


Figura 58. Resultado da compilação e síntese do MTP2-H na ferramenta de desenvolvimento Vivado 2012.4 da Xilinx®.

na referida placa é formada por um componente de lógica programável do tipo FPGA (EP4CE30F23C7 da família Cyclone® da Altera®) e duas memórias do tipo SSRAM de 2 MBytes cada. Esse *hardware* é baseado na arquitetura proposta por Neto e Silveira (1990), onde cada memória está associada a um processador (implementado no FPGA) que tem a capacidade de tratar dois enlaces de canal comum número 7. Uma ampliação da capacidade de tratamento implica na alteração do *hardware* da placa.

Tal limitação não está presente na proposta desenvolvida nessa dissertação e na de Lima (2012), conforme pode-se verificar a seguir.

Para se ter uma estimativa da quantidade de instâncias do MTP2 possível de implementar na referida placa, realizou-se algumas compilações simulando o uso de 2, 4, 8 e 16 instâncias do MTP2. A Tabela 4 mostra os resultados dessas compilações e as respectivas necessidades em termos de recursos do componente e da quantidade de memória para os *buffers* de transmissão e recepção de mensagens.

Tabela 4. Recursos do componente EP4CE30F23C7 e quantidade e memória versus quantidade de instâncias do MTP2.

Quantidade do MTP2	Recursos utilizados no componente EP4CE30F23C7 (%)	Quantidade de memória para os *buffers* de transmissão e recepção (kB)
2	19	84
4	38	168
8	76	336
16	Não suportado	672

Com os resultados apresentados verifica-se que com a proposta desenvolvida será possível implementar até 8 instâncias do MTP2 na placa sem alteração de *hardware*, dobrando a capacidade de tratamento de enlaces de canal comum. Para se utilizar as 16 instâncias, será necessário alterar somente o componente de lógica programável, pois as memórias existentes na placa atendem aos requisitos de espaço necessário para os *buffers* de transmissão e recepção.

A Altera® dispõe de um conjunto de componentes para a migração vertical totalmente compatível com o EP4CE30F23C7, cuja variação está somente na sua capacidade. A Figura 59 mostra os componentes compatíveis com o utilizado.

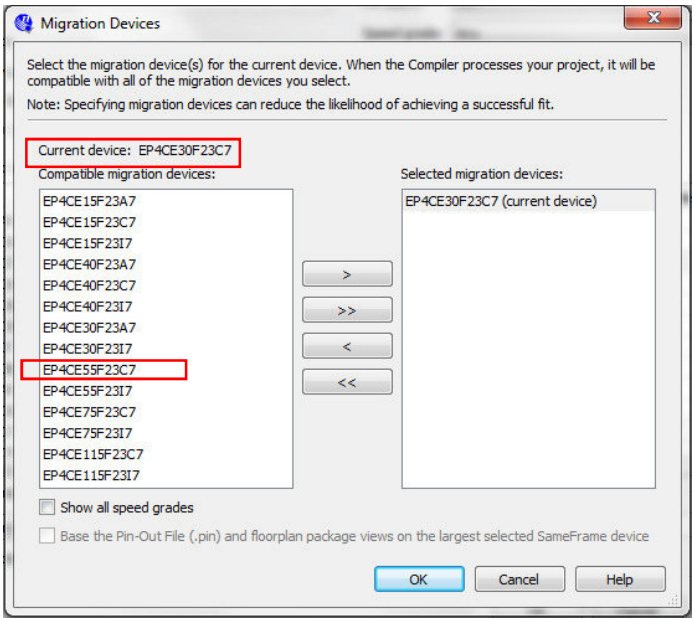


Figura 59. Lista de componentes compatíveis com o EP4CE30F23C7 para migração vertical.

Uma nova compilação foi realizada utilizando o componente EP4CE55F23C7. Nele as 16 instâncias do MTP2 ocuparam 77% de sua capacidade.

6 CONCLUSÃO

Neste trabalho de dissertação foi desenvolvido uma metodologia para conversão de sistemas descritos em SDL para a linguagem de descrição de *hardware* VHDL. A metodologia é composta por um conjunto de regras associada a um modelo de conversão. A aplicação dessa metodologia aos diagramas SDL da recomendação Q.703 resultaram na implementação do MTP2-H, objetivo principal deste trabalho.

Na codificação em VHDL da implementação não foi utilizado nenhum recurso específico de fabricante de componente, deixando o código preparado para ser sintetizado e utilizado em componentes de lógica programáveis de qualquer fabricante. Essa característica torna a solução a "prova de futuro", ou seja, a descontinuidade ou obsolescência de determinado componente de lógica programável não impedirá o seu uso ao longo do tempo. Outra característica obtida com a implementação é a modularidade, ou seja, o sistema MTP2-H poderá ser replicado o número de vezes em que for necessário no equipamento em que for utilizado, bastando ter recursos de lógica e de memória disponíveis. Essas características não eram facilmente obtidas nas soluções apresentadas por Neto e Silveira (1990) e por D'Ávila (1998). Em ambos os casos, a implementação foi realizada utilizando-se uma arquitetura mista onde parte das funções eram executadas por *software* e parte em um *hardware* específico. Hoje, ambas as soluções estão obsoletas, visto que utilizaram componentes e soluções comerciais que já não são mais fabricados.

O ganho obtido com o desenvolvimento desse trabalho em conjunto com o desenvolvido por Lima (2012) já pode ser observado na placa do projeto HUI. Nessa placa, o *hardware* existente permite o tratamento de somente 4 enlaces de canal comum. Com o uso do MTP2 em VHDL, utilizando o mesmo *hardware*, será possível dobrar a quantidade de enlaces tratados. A quantidade de enlaces poderá aumentar ainda mais caso seja utilizado um componente de maior capacidade.

A elaboração de uma metodologia associada ao desenvolvimento e aplicação do modelo de conversão foram essenciais na implementação do MTP2-H, visto a quantidade de blocos funcionais envolvidos. Nesse caso, sem uma metodologia definida

e um modelo de conversão, a implementação teria sido mais trabalhosa e, em consequência, teria demandado um tempo maior em relação ao realizado.

Além do modelo de conversão ter sido desenvolvido baseado nos diagramas encontrados na recomendação Q.703, ele possui uma estrutura básica de comunicação e máquina de estados que podem ser utilizadas como base para a aplicação em outros tipos de diagramas SDL, já que estão previstos o uso dos principais símbolos para a representação dos processos em SDL.

6.1 TRABALHOS FUTUROS

Vislumbra-se como atividades de pesquisa e o desenvolvimento em continuidade ao trabalho ora apresentado as seguintes sugestões:

- O uso de fila de entradas do tipo FIFO para o armazenamento dos sinais recebidos pelo processo requer o uso de componentes de lógica programável com blocos de memória interno. Para avaliar a real necessidade do uso da fila de entradas nos processos é necessário fazer um levantamento das atividades dos sinais recebidos por cada processo em funcionamento real num equipamento de telecomunicações. Se for identificada a possibilidade de se utilizar somente blocos de comunicação com armazenamento dos sinais por meio de marcadores, será possível ampliar a aplicação do MTP2-H também em componentes sem memória interna, reduzindo o custo do *hardware* do equipamento onde for implementado.

- A sintetização de mais de uma instância do MTP2-H foi comprovada nos resultados desse trabalho. Para a utilização delas em um equipamento de telecomunicações, será necessário definir um barramento de controle que possibilite a comunicação entre o nível 3 (MTP3) com as várias instâncias utilizadas sem comprometer o desempenho do equipamento onde será aplicado.

- A validação da implementação do MTP2-H foi realizada por meio de testes funcionais utilizando uma ferramenta de simulação. Testes práticos em um equipamento de telecomunicações seguindo a recomendação Q.781 do ITU-T está previsto para homologar o uso da solução.

REFERÊNCIAS BIBLIOGRÁFICAS

(DIAZ, 2009) DIAZ, V. A. V., Relatório Técnico: Anteprojeto de Implementação Hardware da MTP-2 Lower part do Nível 2 da SS7. Campinas, 2009.

(NETO; SILVEIRA, 1990) NETO, I. L. S., SILVEIRA, L. M., "Implementing Signaling System No. 7 in Trópico RA System". *Telecommunications Symposium, 1990. ITS '90 Symposium Record., SBT/IEEE International*, pp. 137-142, Setembro de 1990.

(DONOHOE *et al.*, 1986) DONOHOE, D. C.; JOHANNESSEN, G. H.; STONE, R. E., "Realization of a Signaling System No 7 Network for AT&T". *IEEE Journal on Selected Areas in Communications*, vol. SAC-4, n.º 8, Novembro de 1986.

(D'ÁVILA, 1998) D'ÁVILA, M. H. C.,*Contribuições para a Implementação do Sistema de Sinalização por Canal Comum nº 7 em uma Central Telefônica Baseada em Ambiente de Processamento Distribuído*. Departamento de Ciência da Computação da Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, MG, Março de 1998. (Dissertação, Mestrado em Ciência da Computação).

(BONATTI; FIGUEIREDO, 1995) BONATTI, I.S., FIGUEIREDO, R.J.O., "Stoht – An SDL-to-Hardware Translator". *Design Automation Conference, 1995. Proceedings of the ASP-DAC '95/CHDL '95/VLSI '95., IFIP International Conference on Hardware Description Languages. IFIP International Conference on Very Large Scal*, pp. 33-36, Agosto/Setembro de 1995.

(SVANTESSON *et al.*, 1998) SVANTESSON, B., KUMAR, S., HEMANI, A., "A methodology and algorithms for efficient interprocess communication synthesis from system description in SDL". *VLSI Design, 1998. Proceedings., 1998 Eleventh International Conference on*, pp. 78-84, Janeiro de 1998.

(SCHLANGER, 1986) SCHLANGER, G. G. "An Overview of Signaling System No. 7". *IEEE Journal on Selected Areas in Communications*, vol. SAC-4, n.º 3, Maio de 1986.

(MODARRESSI; SKOOG, 1990) MODARRESSI, A. R., SKOOG, R. A. "Signaling System No. 7: A Tutorial". *IEEE Communications Magazine*, vol. 28, n.º 7, pp. 19-35, Julho de 1990.

(LIN, 1996) LIN, Y. B. “Signaling System Number 7”. *IEEE Potentials*, vol. 15, n.º 3, pp. 5-8, Agosto/Setembro de 1996.

(PINHEIRO, 2004) PINHEIRO, P. R. G., “Tutoriais de Telefonia Fixa”, *Teleco* - *http://www.teleco.com.br/tutoriais/tutorialciclos/default.asp*, Agosto de 2004.

(ITU-T, 1980) INTERNATIONAL TELECOMMUNICATION UNION, Telecommunication Standardization Sector (ITU-TSS), Genebra. Recomendações Q.700-795; “Specifications of Signalling System No. 7”. CCITT Yellow Book – 1980, vol. VI, Fascículo VI.6., 1980.

(ITU-T, 1988a) INTERNATIONAL TELECOMMUNICATION UNION, Telecommunication Standardization Sector (ITU-TSS), Genebra. Recomendações Q.251-300; “Specifications of Signalling System No. 6”. CCITT Blue Book – 1988, vol. VI, Fascículo VI.3. Revisadas pelo ITU-T Study Group XI (1988-93), Novembro de 1988.

(ITU-T, 1988b) INTERNATIONAL TELECOMMUNICATION UNION (ITU-T), Genebra. Recomendações Z.100; “Specification and Description Languages - SDL”. CCITT Blue Book – 1988, Anexos A – F. vol. VI.20 – VI.24, 1988.

(ITU-T, 1993a) INTERNATIONAL TELECOMMUNICATION UNION, Telecommunication Standardization Sector (ITU-TSS), Genebra. Recomendações Q.700-716; “Specifications of Signaling System No. 7”. CCITT Blue Book – 1988, vol. VI, Fascículo VI.7. Revisadas pelo ITU-T Study Group XI (1988-93), Helsinki, Março de 1993.

(ITU-T, 1993b) INTERNATIONAL TELECOMMUNICATION UNION, Telecommunication Standardization Sector (ITU-TSS), Genebra. Recomendação Q.781; “Signalling System No. 7 – MTP Level 2 Test Specification”. Revisada pelo ITU-T Study Group XI (1988-93), Helsinki, Março de 1993.

(ITU-T, 1993c) INTERNATIONAL TELECOMMUNICATION UNION, Telecommunication Standardization Sector (ITU-TSS), Genebra. Recomendação Q.781; “Signalling System No. 7 – MTP Level 3 Test Specification”. Revisada pelo ITU-T Study Group XI (1988-93), Helsinki, Março de 1993.

(BELINA; HOGREFE, 1989) BELINA, F., HOGREFE, D., “The CCITT-Specification and Description Language SDL”. *Computer Networks and ISDN Systems.* vol. 16, n.° 4, pp. 311-341, Março de 1989.

(SDL FORUM). http://www.sdl-forum.org/SDL/index.htm

(OLSEN *et al.*, 1994) OLSEN, A., FAERGEMAND, O., MOLLER-PEDERSEN, B., SMITH, J.R.W., REED, R., “System Engineering Using SDL-92”. Primeira edição 1994, Editora Elsevier Science & Technology Books.

(SINNOTT; HOGREFE, 2001) SINNOTT, R. O., HOGREFE, S., “Finite state machine based: SDL”. No livro *Formal methods for distributed processing*. Cambridge University Press, pp. 55-76, 2001.

(LIMA, 2012) LIMA, H. F. O., *Implementação de uma solução modular e escalável das funções DAED para o nível 2 do Sistema de Sinalização por Canal Comum número 7 usando dispositivos de lógica programável.* Universidade Federal do Amazonas (UFAM), Manaus, AM, Dezembro de 2012. (Dissertação, Mestrado em Engenharia Elétrica).

(HORN et al., 1999) HORN, W., SVANTESSON, B., KUMAR, S., JANTSCH, A., HEMANI, A. "Hardware synthesis of an ATM multiplexer from SDL to VHDL: A case study". In Proceedings of the IEEE Computer Society Workshop on VLSI '99, pp.100-105, Abril de 1999.

APÊNDICE 1

A seguir é apresentado o código em VHDL do modelo de um processo desenvolvido neste trabalho.

```
library std;
use std.textio.all;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

--------------------------------------------------------------------------------
-- Definição dos sinais (pinos) de entrada/saida
--------------------------------------------------------------------------------
entity Modelo_processo is

   -- Todos os sinais gerados ou recebidos pelo bloco
   port(
      Reset_i       : in std_logic;   -- Reset. Ativo em alto.
      Clk_ent_i     : in std_logic;   -- Relógio de verificação das entradas.
      Clk_maq_i     : in std_logic;   -- Relógio da máquina de estados finita.

      --### Entradas do processo
      Entrada1_X_i  : in std_logic;   -- Sinal de entrada 1 vindo do bloco X.
      Entrada2_Y_i  : in std_logic;   -- Sinal de entrada 2 vindo do bloco Y.
      Entrada3_Z_i  : in std_logic;   -- Sinal de entrada 3 vindo do bloco Z.

      --### Saidas do processo
      Saida1_W_o    : out std_logic; -- Sinal de saida 1 que vai para o bloco W.
      Saida2_W_o    : out std_logic; -- Sinal de saida 2 que vai para o bloco W.

      --### Valor padrão
      Tempo1_i      : in std_logic_vector(m downto 0) -- Valor padrão do temporizador 1.
   );

end Modelo_processo;

--------------------------------------------------------------------------------
-- Definição da arquitetura
--------------------------------------------------------------------------------
architecture RTL of Modelo_processo is

-- Declaração dos componentes, constantes, tipos e sinais utilizados na entidade...

--------------------------------------------------------------------------------------------
begin

   Entradas_s     <= T_overflow_i & Entrada3_Z_i & Entrada2_Y_i & Entrada1_X_i;

   Saida1_W_o     <= Saida1_W_o_s;
   Saida2_W_o     <= Saida2_W_o_s;

   T_value_s      <= T_value_i;
```

```
-------------------------------------------------------------------------------
-- Conexão dos sinais do componente de Comunicação.
-------------------------------------------------------------------------------
Comun_Flag: Com_Flag_CC
   generic map(
      Entrada_ativa_qtd_c    => Ent_ativa_qtd_c,
      Entradas_qtd_c         => Ent_qtd_c,
      Sinais_internos_qtd_c  => Sinais_int_qtd_c
   )
   port map(
      Reset_i              => Reset_i,
      CLK_ent_i            => CLK_ent_i,
      CLK_maq_i            => CLK_maq_i,
      Entradas_i           => Entradas_s,
      Sinais_internos_i    => Sinais_internos_s,
      Saidas_o             => Saidas_s,
      Entrada_ativa_o      => Entrada_ativa_s
   );

-------------------------------------------------------------------------------
-- Conexão dos sinais do componente Temporizador.
-------------------------------------------------------------------------------
T: Temporizador
   generic map(
      T_Value_qtd_g    => T_Value_qtd_c
   )
   port map(
      Reset_i          => Reset_i,
      CLK_maq_i        => CLK_maq_i,
      T_habilita_i     => T_enable_s,
      T_reinicia_i     => T_reinicia_s,
      T_valor_i        => T_value_i,
      T_overflow_o     => T_overflow_o_s
   );
```

```
    ------------------------------------------------------------------------
    -- Máquina de estados do Modelo de Processo.
    ------------------------------------------------------------------------
    ModeloProcesso_mef: process (CLK_maq_i, Reset_i)
    begin

       if Reset_i = '1' then

          -- Inicializa os parâmetros da máquina...

       elsif rising_edge (CLK_maq_i) then

          --### Máquina de estados ###--
          case state is

             -- ### Estado 1 ###--
             when sEstado1 =>
                case Entrada_ativa_s is
                   when Entrada1_X_c =>
                      -- Ações relativas à ativação da Entrada1_X...
                   when T_overflow_c =>
                      -- Ações relativas ao estouro do tempo T...
                   when others =>
                      -- Ações relativas ao valor de entrada desconhecido...
                end case;

             -- ### Estado 2 ###--
             when sEstado2 =>
                case Entrada_ativa_s is
                   when Entrada1_X_c =>
                      -- Ações relativas à ativação da Entrada1_X...
                   when Entrada2_Y_c =>
                      -- Ações relativas à ativação da Entrada2_Y...
                   when Entrada3_Z_c =>
                      -- Ações relativas à ativação da Entrada3_Z...
                   when others =>
                      -- Ações relativas ao valor de entrada desconhecido...
                end case;
             -- ### Estado desconhecido ###--
             when others =>
                -- Ações relativas ao estado desconhecido...
          end case;
       end if;
    end process ModeloProcesso_mef;
end RTL;
```

APÊNDICE 2

A seguir é apresentado o diagrama em SDL e o código em VHDL do processo *Congestion Control* obtido como exemplo da conversão utilizando a metodologia e o modelo desenvolvido neste trabalho.

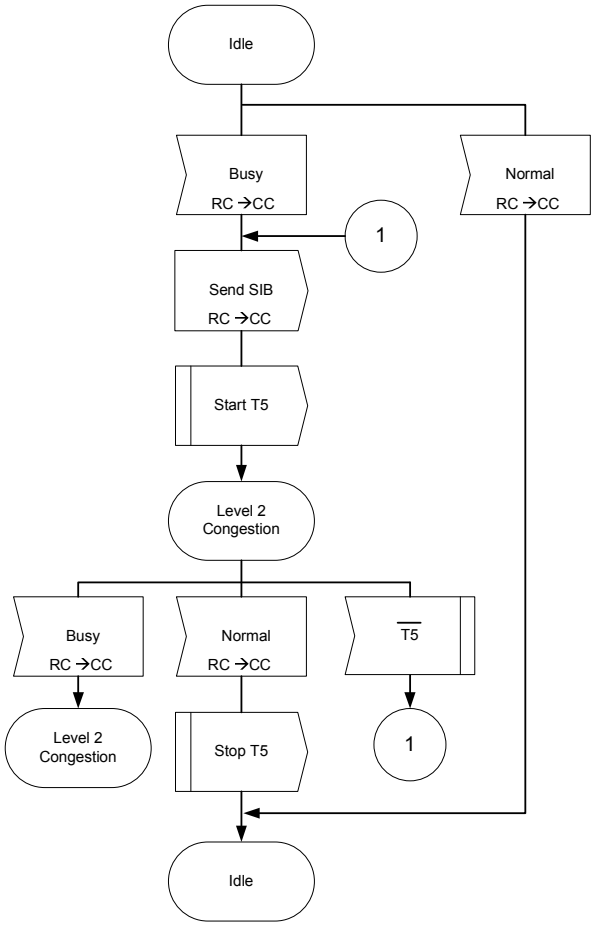


Figura 60. Diagrama em SDL do processo *Congestion Control*.

```
library std;
use std.textio.all;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

--------------------------------------------------------------------------------
-- Definição dos sinais (pinos) de entrada/saida
--------------------------------------------------------------------------------
entity CC is

   -- Todos os sinais gerados ou recebidos pelo bloco CC
   port(
      Reset_i          : in std_logic;   -- Reset. Ativo em alto.
      CLK_ent_i        : in std_logic;   -- Relógio de verificação das entradas.
      CLK_maq_i        : in std_logic;   -- Relógio da máquina de estados
      T5_value_i       : in std_logic_vector (12 downto 0);  -- O valor padrão do T5
                                                             -- é 7680. (1111000000000).
      --### Entradas
      Busy_RC_i        : in std_logic;   -- Indica "enlace" ocupado.
      Normal_RC_i      : in std_logic;   -- Indica estado normal do "enlace".

      --### Saidas
      Send_SIB_TXC_o   : out std_logic   -- Solicita o envio de SIB.
);
end CC;

--------------------------------------------------------------------------------
-- Definição da arquitetura
--  A arquitetura é a descrição em VHDL da máquina de estados do bloco de Controle
--  de Congestionamento (CC) do MTP2 descrito na figura 19 da Q.703.
--  Nesta entidade estão sendo utilizados somente flags para armazenar as ativações das
--  entradas. Devido as ações executadas para cada entrada serem pequenas, não foi
--  identificada a necessidade de se utilizar uma FIFO.
--------------------------------------------------------------------------------
architecture RTL of CC is

-------------------------------------------------------------------------
--### Declaração de componentes
-------------------------------------------------------------------------
constant Ent_ativa_qtd_c  : integer := 2;
constant Ent_qtd_c        : integer := 3;
constant Sinais_int_qtd_c : integer := 2;
constant T_Value_qtd_c    : integer := 13;  -- Quantidade de bits do sinal que
                                            -- contém o valor padrão da contagem.
component Com_Flag_CC is
   generic(
      Entrada_ativa_qtd_c     : integer;
      Entradas_qtd_c          : integer;
      Sinais_internos_qtd_c   : integer
   );
   -- Todos os sinais gerados ou recebidos pelo bloco Comunicacao
   port(
      Reset_i              : in std_logic;   -- Reset. Ativo em alto.
      CLK_ent_i            : in std_logic;   -- Relógio de verificação das entradas.
      CLK_maq_i            : in std_logic;   -- Relógio da máquina de estados
      --### Entradas
      Entradas_i           : in std_logic_vector (Entradas_qtd_c-1 downto 0);
                                             -- Entradas da entidade.
      Sinais_internos_i    : in std_logic_vector (Sinais_internos_qtd_c-1 downto 0);
                                             -- Entradas com os sinais internos do Processo em SDL.
      --### Saidas
      Saidas_o             : out std_logic_vector (Sinais_internos_qtd_c-1 downto 0);
                                             -- Saídas da entidade e internos.
      Entrada_ativa_o      : out std_logic_vector (Entrada_ativa_qtd_c-1 downto 0)
                                             -- Saída com o valor da entrada ativa enviada para a
                                             -- máquina de estados.
   );
end component;
```

```
component Temporizador is
    generic(
        T_Value_qtd_g: integer
    );
    -- Todos os sinais gerados ou recebidos pelo bloco Temporizador
    port(
        Reset_i          : in std_logic;   -- Reset. Ativo em alto.
        CLK_maq_i        : in std_logic;   -- Relógio da máquina de estados
        --### Entradas
        T_habilita_i     : in std_logic;   -- Habilitação do Temporizador.
        T_reinicia_i     : in std_logic;   -- Reinicia a contagem.
        T_valor_i        : in std_logic_vector (T_Value_qtd_g-1 downto 0); -- Valor da contagem
        --### Saidas
        T_overflow_o     : out std_logic   -- Saída de overflow da contagem.
    );
end component;

-------------------------------------------------------------------------------
--### Declaração de sinais
-------------------------------------------------------------------------------

-- Criação do tipo da máquina de estados principal do CC.
type state_type is (
    sIdle,
    sLevel2_congestion
);

-- Declaração da máquina de estados principal
signal state                : state_type := sIdle;
signal Entrada_ativa_s      : std_logic_vector (Ent_ativa_qtd_c-1 downto 0);
                              -- Valor correspondente à entrada ativa na ordem.
signal Entradas_s           : std_logic_vector (Ent_qtd_c-1 downto 0);
                              -- Concatenação de todas as entradas e os sinais
                              --  verificados na máquina do CC.

-- Declaração das constantes dos sinais de entrada.
constant Busy_RC_c          : std_logic_vector (Ent_ativa_qtd_c-1 downto 0) := "01";
constant Normal_RC_c        : std_logic_vector (Ent_ativa_qtd_c-1 downto 0) := "10";
constant T5_overflow_c      : std_logic_vector (Ent_ativa_qtd_c-1 downto 0) := "11";

-- Declaração dos sinais utilizados na máquina.
signal T5_enable_s          : std_logic;
signal T5_value_s           : std_logic_vector (T_Value_qtd_c-1 downto 0);

-- Declaração dos sinais para geração dos pulsos das saídas.
signal Sinais_internos_s        : std_logic_vector(Sinais_int_qtd_c-1 downto 0);
alias Send_SIB_TXC_o_s      : std_logic is Sinais_internos_s(0);
alias T5_overflow_o_s       : std_logic is Sinais_internos_s(1);

signal Saidas_s             : std_logic_vector(Sinais_int_qtd_c-1 downto 0);
alias Send_SIB_TXC_s        : std_logic is Saidas_s(0);
alias T5_overflow_s         : std_logic is Saidas_s(1);
-------------------------------------------------------------------------------------------
begin

    Entradas_s          <= T5_overflow_s & Normal_RC_i & Busy_RC_i;
    Send_SIB_TXC_o      <= Send_SIB_TXC_s;
    T5_value_s          <= T5_value_i;
```

```
-------------------------------------------------------------------------
-- Conexão dos sinais do componente de comunicação.
-------------------------------------------------------------------------
Comun_Flag: Com_Flag_CC
   generic map(
     Entrada_ativa_qtd_c      => Ent_ativa_qtd_c,
     Entradas_qtd_c           => Ent_qtd_c,
     Sinais_internos_qtd_c    => Sinais_int_qtd_c
   )
   port map(
     Reset_i                  => Reset_i,
     CLK_ent_i                => CLK_maq_i,
     CLK_maq_i                => CLK_maq_i,

     Entradas_i               => Entradas_s,
     Sinais_internos_i        => Sinais_internos_s,

     Saidas_o                 => Saidas_s,
     Entrada_ativa_o          => Entrada_ativa_s
   );

-------------------------------------------------------------------------
-- Conexão dos sinais do componente Temporizador.
-------------------------------------------------------------------------
-- Nome    : Temporizador T5
-- Função  : Gera um pulso de um ciclo de clock (T5_overflow_s) quando atinge o
--           valor do tempo configurado. O tempo pode ser um valor entre 80 a 120ms.
--           A entrada T5_enable_s habilita/desabilita o temporizador. Enquanto
--           o temporizador estiver habilitado ele fica contando. O valor
--           adotado será de 100ms.
-------------------------------------------------------------------------

T5: Temporizador
  generic map(
     T_Value_qtd_g  => T_Value_qtd_c
  )
  port map(
     Reset_i             => Reset_i,
     CLK_maq_i           => CLK_maq_i,
     T_habilita_i        => T5_enable_s,
     T_reinicia_i        => '0',
     T_valor_i           => T5_value_i,
     T_overflow_o        => T5_overflow_o_s
);
```

```
    ------------------------------------------------------------------------------
    -- Máquina de estados do Bloco de Controle de Congestionamento (CC)
    -- de acordo com o diagrama SDL da figura 19 da Q.703.
    ------------------------------------------------------------------------------

    CC_me: process (CLK_maq_i, Reset_i)

    begin

       if Reset_i = '1' then
          -- Inicializa os parâmetros da máquina
          state                <= sIdle;
          T5_enable_s          <= '0';
          Send_SIB_TXC_o_s     <= '0';

       elsif rising_edge (CLK_maq_i) then
          --### Máquina de estados CC ###--
          case state is
             -- ### Estado Incial ###--
             when sIdle =>

                case Entrada_ativa_s is

                   when Busy_RC_c =>
                      Send_SIB_TXC_o_s    <= not Send_SIB_TXC_o_s;
                      T5_enable_s         <= '1';                    -- Start T5.
                      state <= sLevel2_congestion;

                   when Normal_RC_c =>
                      state <= sIdle;

                   when others =>
                      state <= sIdle;

                end case;
             -- ### Estado Nível 2 congestionado ###--
             when sLevel2_congestion =>

                case Entrada_ativa_s is

                   when Busy_RC_c =>
                      state <= sLevel2_congestion;

                   when Normal_RC_c =>
                      T5_enable_s <= '0';                            -- Stop T5.
                      state <= sIdle;

                   when T5_overflow_c =>
                      Send_SIB_TXC_o_s    <= not Send_SIB_TXC_o_s;
                      T5_enable_s         <= '1';                    -- Start T5.
                      state <= sLevel2_congestion;

                   when others =>
                      state <= sLevel2_congestion;

                end case;
             -- ### Estado desconhecido  ###--
             when others =>
                state <= sIdle;

          end case;
       end if;
    end process CC_me;
end RTL;
```