Proposta de uma Rede de Comunicação Automotiva Baseada no FlexRay

Daniel C. F. Rezende e Walter C. Borelli

***Resumo* – Este artigo apresenta a criação de um modelo executável de uma rede intraveicular baseada no protocolo de comunicação FlexRay. Sendo assim, é proposta neste artigo uma descrição do projeto e do modelamento em SDL de uma rede FlexRay, visto que isto não é encontrado na especificação padrão do FlexRay. Foram propostas algumas melhorias no protocolo para possibilitar a criação de uma completa e funcional rede FlexRay. Também são apresentadas simulações e sua completa validação. Além disso, são apresentados diagramas de troca de mensagens que descrevem o correto comportamento do FlexRay que podem ser usados como referência nos testes para validação de qualquer implementação usando este protocolo.**

***Palavras-Chave* – Protocolos de Comunicação, Sistemas Intraveiculares, FlexRay, SDL.**

***Abstract* – This paper presents the creation of an executable SDL model of an in-vehicle network based on the FlexRay protocol. Thus, it is presented a description of the design and the modelling in SDL of a functional FlexRay network, since this is not accomplished by the FlexRay Protocol Specification. Some improvements in the FlexRay specification were also proposed. Results of the system simulation and validation are also presented. Besides, message sequence charts are presented describing the correct behavior of FlexRay. Such diagrams may be used as reference in tests for validation of any application using this protocol.**

***Keywords* – Communications Protocols, FlexRay, In-Vehicle Systems, SDL.**

I. INTRODUÇÃO

Recentemente tem havido um aumento significativo na quantidade de eletrônica introduzida no carro, e é esperado que esta tendência continue enquanto as montadoras introduzirem avanços adicionais em segurança, confiabilidade e conforto [1].

Motivados pelas aplicações intraveiculares, várias companhias vêm investindo no projeto de controladores que pudessem gerenciar o tráfego de informações com interfaces através de um meio físico reduzido, geralmente um barramento serial, porém capaz de possibilitar a multiplexação dessas informações. A esta forma de conexão é dado o nome de Rede Intraveicular (In-Vehicle Networking).

A introdução de sistemas avançados de controle que combinam sensores múltiplos, atuadores e centrais eletrônicas, também chamadas de ECUs (Electronic Control Unit) está começando a demandar exigências na tecnologia de uma comunicação não encontrada atualmente nos protocolos de comunicação existentes.

As exigências adicionais para as aplicações futuras do controle intraveicular incluem combinação de taxas de dados mais elevadas, comportamento determinístico e sustentação da tolerância à falhas. Para comunicação não-crítica um número de protocolos tornou-se popular, tais como LIN (Local Interconnect Network), CAN (Controller Area Network) e MOST (Media Oriented Systems Transport). Para aplicações críticas (safety-critical) e mais complexas (aplicações X-by-Wire e aplicações relacionadas à transmissão e motor) o FlexRay surge como protocolo mais adequada a operações em tempo real [2].

A introdução desses sistemas X-by-Wire, que essencialmente significa substituir componentes mecânicos e hidráulicos por componentes eletrônicos conectados por fios sem a necessidade de componentes mecânicos sobressalentes, será um passo significante para a indústria automobilística [3]. Sistemas de controle de tração e estabilidade (Steer-by-Wire), de freios (Brake-by-Wire) e direção (Drive-by-Wire) são exemplos de sistemas X-by-Wire.

Este artigo apresenta uma proposta de um modelo SDL executável para uma rede de comunicação para aplicação automotiva baseada no protocolo FlexRay. A rede foi modelada usando a linguagem SDL (Specification and Description Language). O propósito deste trabalho é criar um modelo executável de uma rede FlexRay e a partir do qual produzir diagramas de trocas de mensagens (MSC - Message Sequence Charts) que possam ser utilizados para validar aplicações deste protocolo. Tais diagramas são de suma importância na geração de pacotes de testes (Test Suites) para aplicações específicas. Este artigo descreve o projeto e o modelamento em SDL de uma rede funcional FlexRay, visto que isto não é realizado na especificação padrão do FlexRay [5] que apenas apresenta o comportamento dos mecanismos principais do protocolo. Algumas melhorias na especificação do FlexRay foram propostas para a elaboração desta rede usando SDL.

Este artigo é organizado como segue: Seção II descreve o protocolo de comunicação FlexRay, Seção III descreve o modelo do sistema e também apresenta as melhorias feitas na especificação do protocolo. Seção IV apresenta alguns resultados da simulação do sistema, sua validação e a geração dos diagramas de trocas de mensagens. Conclusões são apresentadas na Seção V.

II. PROTOCOLO FLEXRAY

O protocolo FlexRay surgiu como resultado da cooperação entre BMW e DaimlerChrysler depois que as duas montadoras automotivas perceberam que as soluções atuais não satisfariam suas necessidades para aplicações futuras incluindo X-By-Wire. Como resposta a isso, o Consórcio FlexRay [1] foi formado com o objetivo de desenvolver um novo protocolo, chamado FlexRay [4] e [5]. Este novo protocolo deveria ser a solução não só para a introdução dos sistemas X-By-Wire, mas também para a substituição de

Daniel Cesar Felisberto Rezende e Walter da Cunha Borelli, Departamento de Telemática, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas-SP, E-mails: {danielre, borelli}@dt.fee.unicamp.br.

alguns protocolos adotados atualmente, assim reduzindo o número total de redes intraveiculares [6].

FlexRay é uma arquitetura eletrônica aberta, comum e escalável para aplicações automotivas. Este sistema pode ser operado no modo de canal único ou no modo com dois canais, fornecendo, assim, redundância quando necessário. O FlexRay permite transmissão de dados síncronos e de dados assíncronos. Através da transmissão síncrona, outros nós na rede recebem mensagens orientadas por tempo (time-triggered) em um tempo de latência predefinido; através da transmissão assíncrona, as mensagens chegam a seus destinos de forma mais rápida ou mais devagar, dependendo da prioridade atribuída a elas (event-triggered). Atualmente, o sistema FlexRay pode alcançar taxas de até 10 Mbps.

É esperado que o FlexRay seja o sistema de comunicação padrão para aplicações automotivas de controle interconectando ECUs em sistemas automotivos futuros de alta-velocidade [6].

III. Modelo do Sistema

Para criar uma rede FlexRay completa e funcional sua estrutura teve que ser projetada. Dessa forma, a arquitetura do sistema, a topologia da rede, a integração entre os principais mecanismos dentro do controlador de comunicação e a interface que conecta o controlador de comunicação ao canal de comunicação foram todos criados e descritos usando a linguagem SDL.

É considerado neste trabalho um sistema com três nós interconectados através da topologia de barramento. O sistema de barramento FlexRay modelado juntamente com o esboço da arquitetura do nó é ilustrado na Fig. 1.

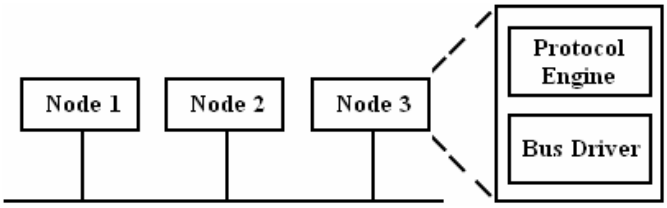

Fig. 1. Exemplo de sistema FlexRay e arquitetura do nó.

O sistema é uma entidade representada por um quadro que delimita a rede modelada. O que está fora do escopo deste quadro é chamado de ambiente e não está especificado em SDL. Fig. 2 ilustra o sistema da rede FlexRay modelada contendo quatro blocos. Esses blocos descrevem um conjunto de nós interconectados por um barramento. Os blocos chamados *Node1*, *Node2* e *Node3* representam os nós na rede. O bloco *Bus* representa o meio físico.

Cada bloco que representa um nó é conectado a cinco canais. Esses canais permitirão ao sistema modelado interagir com outras partes não especificadas neste modelo, que são: o CHI (interface entre o host e o controlador de comunicação) e alguns dos mecanismos do núcleo do protocolo FlexRay (codificação/decodificação e sincronismo do clock). O processo de codificação e decodificação é representado pelo termo CODEC. O mecanismo de sincronismo do clock é dividido em três processos: o processo de inicialização do sincronismo do clock (CSS), o processo de sincronismo do clock (CSP) e o processo de geração de macrotick (MTG). Os sinais que trafegam em cada direção através dos canais são indicados por listas de sinais anexadas às extremidades dos canais. A comunicação entre os nós da rede é alcançada codificando a carga útil de um pacote de dados (enviada pelo CHI) em frames ou símbolos e transmitindo através do barramento.

As descrições do comportamento dentro do Nó1 e do Nó2 são as mesmas. Somente o Nó3 tem descrição de comportamento diferente, pois ele foi adaptado para não ser um nó coldstart (ver seção IV).

Fig. 3 ilustra como o nó foi descrito em SDL. Cada bloco Nó contém um bloco controlador de comunicação (*Communication Controll*er) e um bloco controlador de barramento (*Bus Driver*). Nesta arquitetura o CHI é usado como interface de controle e de dados entre o sistema host e o mecanismo do protocolo FlexRay.

O bloco *Communication Controller* recebe e envia sinais para o CHI, para o bloco *Bus Driver* e para os outros componentes do mecanismo do protocolo FlexRay. O bloco *Bus Driver* troca sinais diretamente com o bloco *Communication Controller* e indiretamente com o CHI, o processo CSS e com o barramento através do ambiente.

Três dos principais mecanismos que compõem o protocolo FlexRay estão descritos como processos dentro do bloco *Communication Controller*. Fig. 4 ilustra os três processos no bloco *Communication Controller*.

A finalidade do processo *POC* (Protocol Operation Control) é reagir aos comandos do host e às condições do protocolo ativando mudanças coerentes nos mecanismos de maneira síncrona, e fornecer ao host o estado atual dos mecanismos em relação às mudanças.

O processo *MAC* (Media Access Control) define o controle de acesso ao meio físico (barramento) através do gerenciamento do ciclo de comunicação recorrente.

O processo *FSP* (Frame and Symbol Processing) verifica a correta temporização dos frames e símbolos em relação ao esquema TDMA, aplica testes sintáticos adicionais nos frames recebidos e verifica a correção semântica dos frames recebidos.

A especificação do FlexRay não descreve o controlador de barramento (*Bus Driver*) usando SDL, logo este teve que ser criado. O controlador de barramento é um componente eletrônico que consiste de um transmissor e um receptor que conecta o controlador de comunicação a um canal de comunicação.

O bloco *Bus Driver* é ilustrado na Fig. 5 e contém dois processos: *Transmitter* e *Receiver*. Para simplificar o sistema FlexRay, mas ainda seguindo as regras de operação do protocolo, os processos Transmissor e Receptor também foram projetados para realizar parte do trabalho do processo CODEC. Por exemplo, quando o sinal *transmit_frame_on_A (vType, vTF*) chega ao bloco *Bus Driver*, ele é guiado através da rota de sinal (signal route) *FromCC* para o processo *Transmitter* e depois ele é convertido para o sinal *Frame vType, vTF)* e finalmente enviado para o barramento.

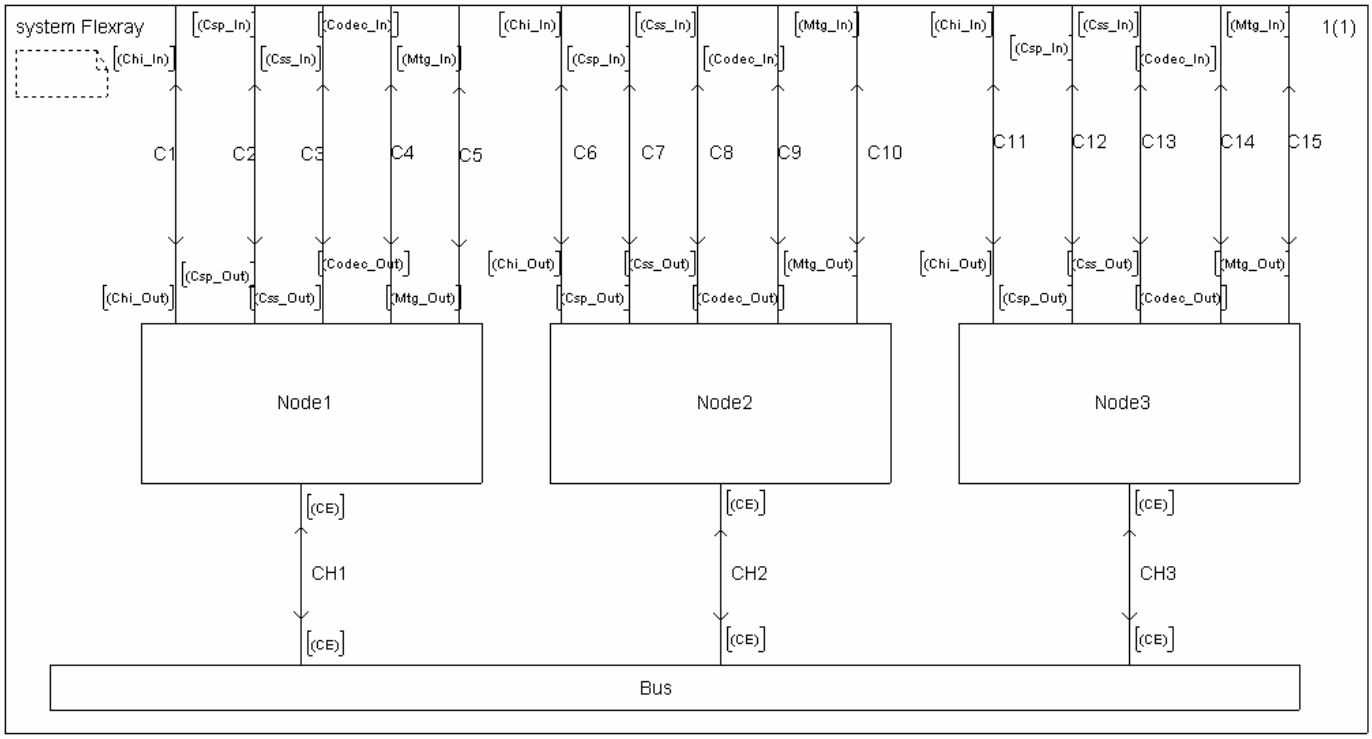

Fig. 2. Nível de sistema do modelo SDL.

No processo Receiver ocorre a conversão inversa. Quando o sinal *Frame (vType, vTF)* chega, seu parâmetro *vTF* (variável que contém os dados do frame a ser transmitido) é atribuído ao parâmetro *vRF* (variável que contém os dados do frame recebido) e um sinal *frame_decoded_on_A (vRF)* é enviado. O parâmetro *vType* (variável que contém o tipo de transmissão) é usado para diferenciar a recepção do elemento de comunicação.

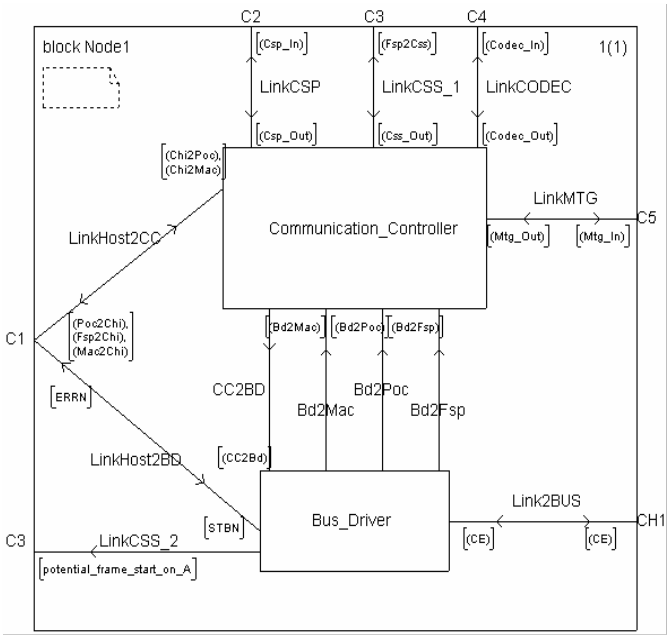

Fig. 3. Descrição do nó.

Uma interface para o CHI também foi especificada no bloco *Bus Driver* e consiste de um sinal *STBN* (Standby Not) que é usado para controlar o modo de operação do controlador do barramento e um sinal *ERRN* (Error Not) que é usado para indicar erros detectados.

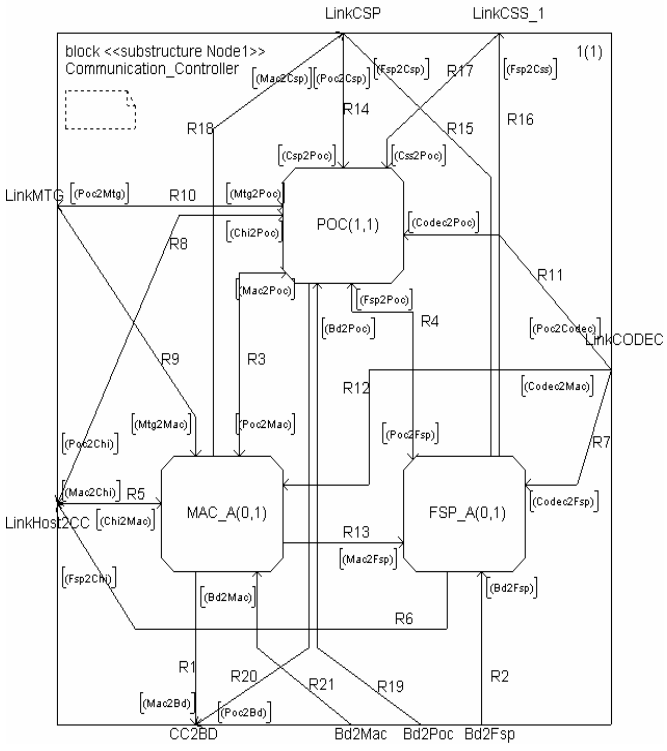

Fig. 4. Descrição do núcleo do protocolo.

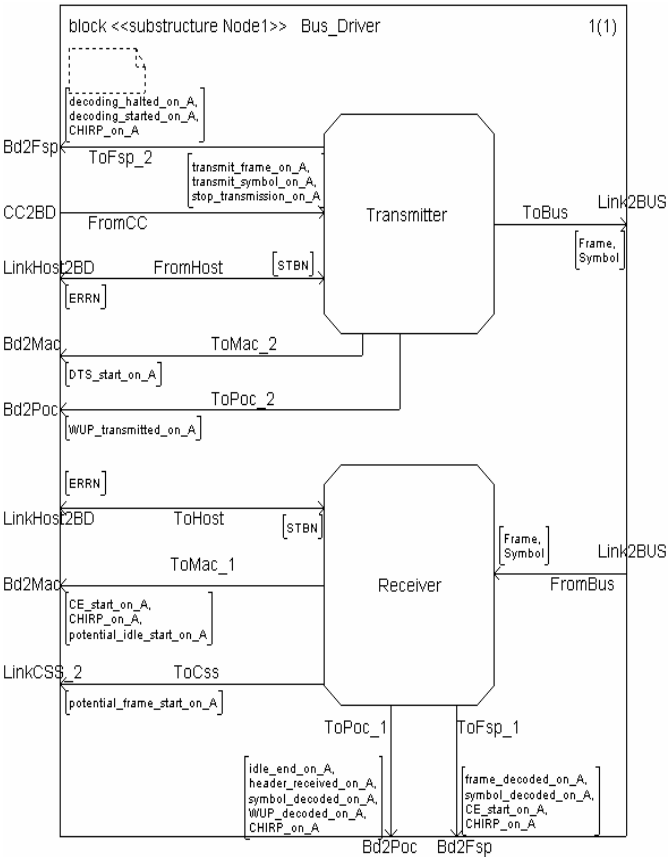

Fig. 5. Descrição do controlador de barramento.

Além da descrição completa da estrutura da rede, algumas melhorias e modificações também foram feitas em relação à especificação do FlexRay a fim de gerar um sistema executável usando SDL.

As principais alterações foram:

- Substituição de todas as construções de macros descritas em [5] por construções de procedimentos.
- Criação do bloco *Bus Driver*.
- Adaptação dos tipos existentes de temporizadores do FlexRay para um único tipo de temporizador baseado na unidade microtick.
- Substituição de todas as construções task contendo textos informais relacionados ao CHI por sinais de saída usando as construções de output.
- Substituição de alguns termos descritos em [5] por novos termos devido a conflito com palavras-chave utilizadas pela ferramenta SDL.
- Criação de um procedimento *CycleCounter* para verificar o ciclo de comunicação atual.
- Substituição da construção task *import_vTCHI_from_the_CHI* no procedimento *ASSEMBLE_STATIC_FRAME_A* por um sinal de entrada recebido do CHI.

Alguns mecanismos do protocolo especificados em [5] usam construções de macros com o exclusivo propósito de simplificar a apresentação do protocolo em SDL. Porém, esta prática não é aconselhável ao implementar sistemas reais portáteis. Macros são geralmente dependentes de certas ferramentas SDL, logo o projetista poderia ter dificuldade ao tentar compilar seu modelo utilizando diferentes editores de SDL. Dessa forma, todas as construções de macros foram substituídas por construções de procedimentos.

A criação do bloco *Bus Driver*, como explicado anteriormente, foi fundamental na geração de um modelo SDL executável, já que este componente não é especificado em [5].

A representação do tempo no protocolo FlexRay é baseada em uma hierarquia que inclui microticks, macroticks e sampleticks. Vários mecanismos do FlexRay necessitam de temporizadores que medem certo número de microticks ou macroticks. A especificação do FlexRay usa uma extensão dos temporizadores do SDL para realizar tal função, porém esta extensão não é suportada pela ferramenta SDT utilizada (TAU SDL Design Tool v.4.2).

Como existe uma relação definida entre a unidade de tempo microtick (μT), a unidade de tempo macrotick (MT) e a unidade de tempo da amostragem de bits sampleticks (ST), o temporizador microtick foi definido como sendo o único tipo de temporizador presente em todo o sistema. Logo, todos os outros temporizadores (macrotick e sampletick) tiveram seus valores recalculados como segue: 1MT é igual a 40μT e 1ST é igual a 1μT. Assim, os temporizadores sampletick foram diretamente convertidos para temporizadores microtick e os temporizadores macrotick tiveram seus valores multiplicados por 40, valor representado pela constante *pMicroPerMacroNom*. Por exemplo, o temporizador *tMinislot* que tem valor igual a 40MT (representado pela constante *gdMinislot*) é declarado como sendo multiplicado pela constante *pMicroPerMacroNom* (número inteiro de microticks por cada macrotick). Fig. 6 ilustra a declaração dos temporizadores no processo MAC_A.

```
timer tSlotBoundary := gdStaticSlot * pMicroPerMacroNom;
timer tMinislot := gdMinislot * pMicroPerMacroNom;
```

Fig. 6. Declaração dos temporizadores.

Alguns termos descritos em [5] entraram em conflito com as palavras-chave da ferramenta SDL, por exemplo, *ALL*, *STATE*, *CHANNEL* e *ACTIVE*. A solução encontrada foi substituir esses termos por outras palavras livres de conflito. Os termos foram substituídos, respectivamente, por *TODOS*, *ESTADO*, *CANAL* e *ATIVO*.

Para gerar um modelo SDL mais próximo de uma implementação real algumas construções task, contendo textos informais relacionados à exportação de variáveis do CHI, foram substituídas por sinais de saída usando as construções output. O mesmo acontece no procedimento *ASSEMBLE_STATIC_FRAME_A* localizado no processo *MAC_A*, mas ao contrário do exemplo anterior, uma construção input de sinal de entrada está substituindo a construção task original *import_vTCHI_from_the_CHI*. Então agora, o sistema pode receber dados do host pelo CHI.

As decisões de mudança dos modos nos mecanismos feita pelo processo *POC* no final de cada ciclo depende se o número atual do ciclo é par ou ímpar, porém [5] não descreve nenhum mecanismo para verificar o estado do ciclo. Assim, foi criado um procedimento chamado *CycleCounter* para verificar o ciclo de comunicação atual durante a operação do FlexRay.

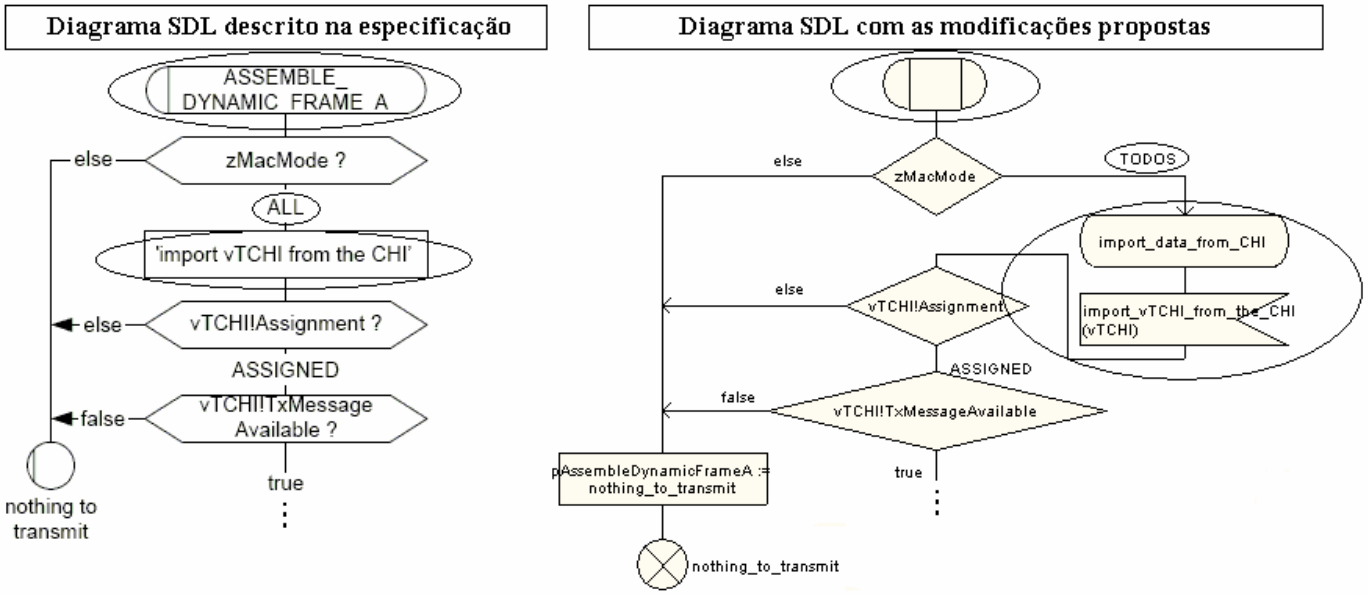

Fig. 7. Do lado esquerdo, o diagrama SDL obtido em [5]. Do lado direito, o diagrama SDL proposto. Os círculos destacam onde as alterações ocorreram.

Na Fig. 7 estão ilustrados três das modificações mencionadas acima que foram feitas em [5] a fim de aprimorar o modelo SDL, que são: a descrição do diagrama usando procedimento ao invés de macro, a substituição do termo *ALL* por *TODOS* e a substituição da construção task *import_vTCHI_from_the_CHI* pelo sinal de entrada *import_vTCHI_from_the_CHI (vTCHI)* através da construção input, onde *vTCHI* é a variável que armazena os dados enviados pelo host.

IV. Simulação, Validação e Geração de MSCs da Rede FlexRay

Depois de especificar a rede FlexRay proposta, as funcionalidades desejadas foram completamente testadas. O sistema foi integralmente validado através da ferramenta Validator disponível no pacote SDT da Telelogic.

Nesta fase foi possível detectar alguns erros dinâmicos (erros gerados em tempo de execução) como, por exemplo, o consumo implícito de sinal (implicit signal consumption) no procedimento *Wakeup*. Este problema aconteceu porque foram ativados dois temporizadores e quando um deles expirou, seu sinal foi consumido, ao passo que o sinal proveniente do outro temporizador foi descartado. Tal erro foi facilmente descoberto depois da execução da ferramenta Validator, notando-se que não havia um comando para desativar o segundo temporizador depois do término do primeiro. A solução foi adicionar um comando para realizar tal função (Fig. 8).

Depois da correção de todos os erros dinâmicos, o sistema foi novamente validado e então todos os símbolos esperados foram alcançados. Fig. 9 mostra o gráfico de cobertura dos símbolos gerado pela ferramenta de validação do SDL.

Após a completa validação do sistema, foram simuladas as principais funcionalidades do sistema FlexRay para a garantia da validade do sistema. Para tanto foi utilizada a ferramenta Simulator, que também é parte integrante do pacote SDT. Esta ferramenta permite o acompanhamento total da simulação gerando diagramas de trocas de mensagens (MSC - Message Sequence Charts), que permitem visualizar a interação entre os diversos componentes do sistema.

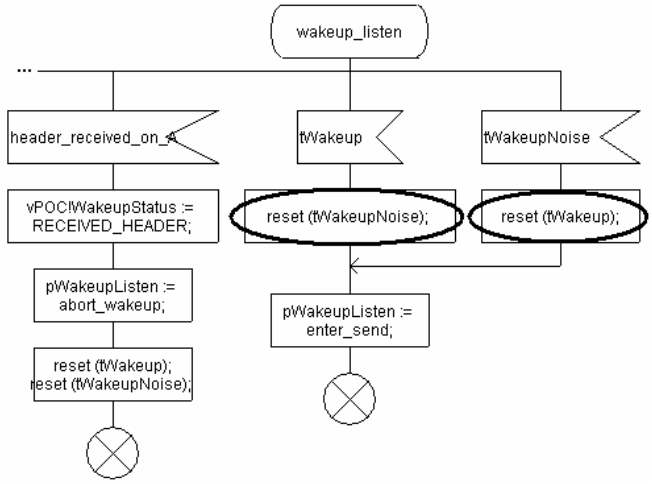

Fig. 8. O comando reset foi adicionado para evitar sinal implícito.

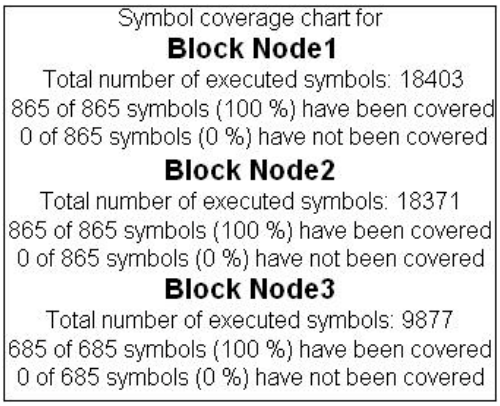

Fig. 9. Cobertura de símbolos dos Nós 1, 2 e 3.

Para a simulação do modelo FlexRay proposto foram utilizados os cenários para o início de comunicação da rede

descritos em [5]. O ato de começar o processo de inicialização da rede é chamado de coldstart. Somente alguns nós são permitidos inicializar a rede, chamados nós coldstart. De acordo com a especificação, três diferentes tipos de caminhos podem ser seguidos pelo nó dependendo de sua configuração. O primeiro caminho refere-se ao nó coldstart pioneiro na inicialização; o segundo caminho refere-se aos nós coldstart que não foram os pioneiros na inicialização e o terceiro caminho refere-se aos nós que não são coldstart. A partir desta descrição foram gerados nove diagramas de trocas de mensagens (MSC) mostrando detalhadamente o correto comportamento do mecanismo do protocolo dentro de cada tipo de nó. Para cada tipo de caminho, três cenários diferentes foram simulados, dependendo da verificação de erro executada ao final de cada ciclo de comunicação. Dessa forma todas as situações possíveis que ocorrem durante a inicialização e a operação normal de uma rede FlexRay foram simuladas e descritas em forma de diagramas. Tais diagramas podem ser utilizados na criação de pacotes de testes para implementação de qualquer aplicação usando o FlexRay. Estes pacotes de testes podem ser produzidos por ferramentas específicas (TTCN) e estão também disponíveis no software SDT usado neste trabalho.

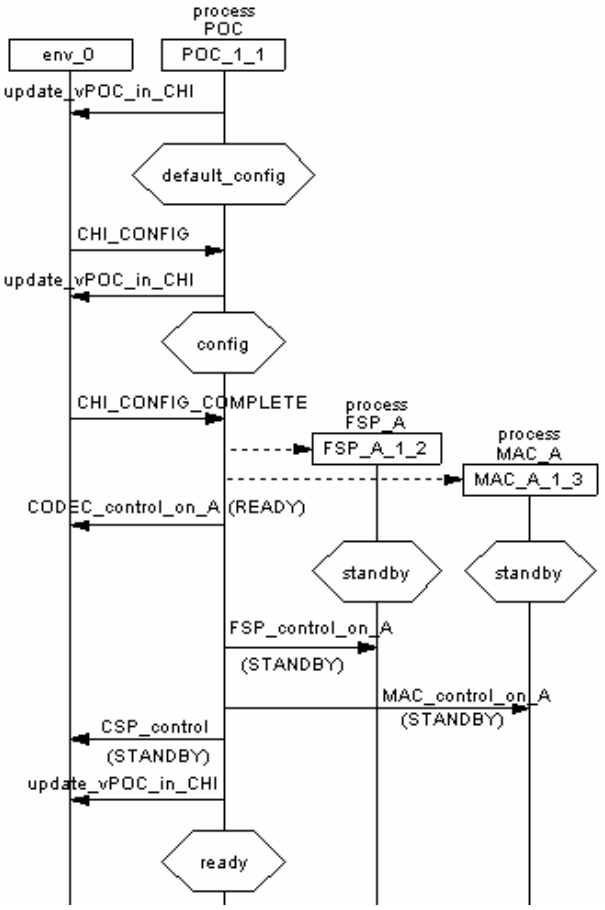

Fig. 10. MSC que descreve o processo de inicialização da rede FlexRay.

A Fig. 10 ilustra o início de um dos diagramas gerados. Esse diagrama refere-se ao cenário de um nó coldstart pioneiro na inicialização da rede. Antes de entrar no estado *default_config* o processo *POC* inicializa o valor de suas variáveis. Após isto feito, ele aguarda um sinal do host para entrar no estado *config* e assim habilitar a configuração da rede. Ao receber o sinal *CHI_CONFIG_COMPLETE* o processo *POC* cria todos os outros processos principais do protocolo, passa para o estado *ready* e aguarda a fase de inicialização da rede.

V. CONCLUSÃO

O trabalho apresentado neste artigo propôs a geração de um modelo SDL executável de uma rede intraveicular baseada no protocolo de comunicação FlexRay. Para criar uma completa e funcional rede FlexRay sua estrutura teve que ser projetada. Dessa forma, a arquitetura do sistema, a topologia da rede, a integração entre os principais mecanismos dentro do controlador de comunicação e a interface que conecta o controlador de comunicação ao canal de comunicação foram todos criados e descritos usando a linguagem SDL. O sistema foi modelado usando a especificação do protocolo FlexRay para fornecer um modelo executável dos mecanismos do protocolo e a partir do qual produzir diagramas de trocas de mensagens que possam ser utilizados para validar aplicações deste protocolo. Tais diagramas são de suma importância na geração de pacotes de testes (Test Suites) para aplicações específicas. Algumas melhorias na especificação do FlexRay foram propostas para a elaboração desta rede usando SDL.

Para a concretização deste trabalho fez-se uso do conjunto de ferramentas SDT, utilizando-se o Simulator para a simulação dos casos de uso mais importantes do sistema FlexRay e o Validator para a validação completa do sistema. Utilizando-se de tais ferramentas, todas as possíveis situações puderam ser testadas, o que garante a validade do modelo proposto. Pode-se desta forma, facilitar o processo de uma eventual implementação, uma vez que a lógica do sistema já está verificada e não possui erros.

REFERÊNCIAS

[1] FlexRay Consortium. FlexRay. Available at: <http://www.flexray.com> Access: Jul. 2006.

[2] E. Armengaud et al. “A monitoring concept for an automotive distributed network - the FlexRay example”. In: 7th IEEE International Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2004), 2004.

[3] G. Leen, D. Hefferman, “In-vehicle networks, expanding automotive electronic systems”. In: IEEE Transaction on Computers, Jan. 2002, pp. 88-93.

[4] J. Berwanger et al. “FlexRay – the communication system for advanced automotive control systems”. In: SAE 2001 World Congress, Society of Automotive Engineers, Detroit, MI, Apr. 2001.

[5] FlexRay Consortium. FlexRay Communications System - Protocol Specification Version 2.1 Revision A. Dec. 2005.

[6] T. Nolte, H. Hansson, L. Bello, “Implementing next generation automotive communications”. In: Proceedings of the 1st Embedded Real-Time Systems Implementation Workshop (ERTSI'04) in conjunction with the 25th IEEE International Real-Time Systems Symposium (RTSS'04), Lisbon, Portugal, Dec. 2004.

[7] Telelogic AB. Telelogic TAU 4.2 SDL Suite Getting Started: Technical report, Telelogic AB Sweden, Sep. 2001.

[8] R. Braek, “SDL basics”. In: Computer Networks and ISDN Systems, Jun. 1996, pp. 1585-1602.

[9] L. Doldi, Validation of Communications Systems with SDL. Chichester, Hoboken (NJ), Wiley, 2003.