seminário de iniciação científica
ISSN 2558-6052

# ESTUDO E APLICAÇÃO DE VERIFICAÇÃO DE MODELOS EM FUNÇÕES VEICULARES MODELADAS EM ESTADOS FINITOS

Leonardo Gomes Pereira Neto[1]; Tayna Rios Espinosa [2]; Gustavo Lobato Campos[3]; Mário Luiz Rodrigues Oliveira4;

1 Leonardo Gomes Pereira Neto (IFMG), Engenharia Elétrica, IFMG Campus Formiga, Formiga - MG; leonardo.gpn@gmail.com
2 Tayna Rios Espinosa, Ciência da Computação, IFMG - Campus Formiga, São Paulo – SP
3 Gustavo Lobato Campos: Pesquisador do IFMG, Campus Formiga; @ifmg.edu.br
4 Mário Luiz Rodrigues Oliveira: Pesquisador do IFMG, Campus Formiga; mario.luiz@ifmg.edu.br

## RESUMO

Os produtos de *software* e *hardware* desenvolvidos atualmente envolvem alto nível de complexidade. A abordagem tradicional para verificação envolve realizar testes e simulações. Contudo tais abordagens não exploram todo o universo de possíveis estados do sistema e assim não são apropriadas para garantir que um produto de *software* ou *hardware* é isento de erros. Técnicas de métodos formais como verificação de modelos têm sido largamente usadas e bem aceitas na indústria e na academia e podem ser utilizadas em complemento a essa abordagem tradicional. A busca por produtos de *software* e *hardware* com qualidade tem motivado o uso dessas técnicas formais, principalmente porque testar programas concorrentes não é uma tarefa trivial e é suscetível a erros. Verificação de modelos é o problema de testar automaticamente e exaustivamente se um modelo que representa um sistema atende a uma dada especificação e pode ser aplicada a sistemas modelados em diagramas de estados finitos, tais como sistemas de *hardware* embarcados que implementam funções veiculares. O objetivo deste projeto de iniciação científica foi o estudo e a aplicação da técnica de verificação de modelos em funções veiculares modeladas em estados finitos. Para aplicação da técnica de verificação de modelo são necessárias 3 etapas: modelagem, especificação e verificação. Na modelagem faz-se a descrição das características e comportamentos do sistema a ser analisado em uma linguagem. Na especificação faz-se a descrição das propriedades do sistema a serem verificadas e na verificação é executado um procedimento de verificação com a finalidade de determinar se as propriedades são mantidas no modelo. Para realizar a verificação de modelos utilizou-se o produto de *software* denominado NuSMV e na etapa de modelagem a linguagem SMV, a qual é entendida pela ferramenta NuSMV. Na etapa de especificação utiliza-se lógica temporal (CTL). O projeto encontra-se em fase de finalização e como resultados parciais temos a aplicação da técnica de verificação de modelos nas funções veiculares analisadas e a capacitação de recurso humano na área de métodos formais, especificamente na técnica verificação de modelos.

## INTRODUÇÃO

Atualmente sistemas computacionais estão presentes em várias situações de nossas vidas e também auxiliam ou controlam vários processos organizacionais. Ao desenvolver um produto de software ou um novo hardware deve-se primar pela qualidade do produto desenvolvido. Uma quantidade razoável dos produtos de software e hardware construídos atualmente exibe alta complexidade (Silva, 2015) e há uma dificuldade de garantir a qualidade através de inspeções informais.

Uma abordagem possível para tentar garantir a qualidade do produto de software ou hardware é o uso de testes e simulações. Contudo tais abordagens não exploram todo o universo de possíveis estados do sistema e assim não são apropriadas para garantir que um software é isento de erros. Testes são feitos postulando-se uma hipótese sobre a execução do sistema e simulando uma execução deste em que esta hipótese ocorre para verificar se o comportamento do sistema corresponde ao esperado. O problema com esta metodologia é que as hipóteses sendo postuladas necessitam ser explicitamente especificadas. Como o número de comportamentos possível é muito grande (McMillan, 1992) estas hipóteses cobrem proporcionalmente um percentual muito pequeno das possibilidades de execução.

Por exemplo, os projetistas do Ariane 5 proveram o sistema de tolerância a falhas de hardware através de módulos duplicados de cálculo, mas não previram que se o software causasse o erro o mesmo seria calculado por ambos módulos ao mesmo tempo (Gleick, 1996 e Lions,1996). Uma solução alternativa e complementar ao uso de testes e simulações é a adoção de métodos formais.

Segundo Flores (2016), métodos formais constituem um conjunto de técnicas de elaboração de sistemas em que é aplicado o formalismo matemático na assistência às fases de especificação, desenvolvimento e verificação. No que tange especificamente a fase de verificação pode-se citar a verificação de modelos (model checking) e prova de teoremas.

O presente projeto tem por objetivo principal o estudo e a aplicação da técnica de verificação de modelos em funções veiculares modeladas em diagramas de estados finitos. Na sequência esse texto

seminário de iniciação científica
ISSN 2558-6052

apresenta a metodologia empregada assim como os resultados obtidos até o momento, por fim a relação da bibliografia consultada e referenciada ao longo do texto do presente projeto.

**METODOLOGIA:**

Para realizar a verificação de modelos tem-se trabalhando com o software denominado NuSMV (Projeto NuSMV, 2021), o qual é uma das implementações de um sistema de Symbolic Model Verifier e que utiliza a representação interna de BDDs (Bryant, 1986). O software NuSMV é um projeto open-source desenvolvido e mantido por grupos de pesquisa da fundação Bruno Kessler, das universidades de Gênova e Trento e da Carnegie Mellon University e nessa proposta é a ferramenta de model checking utilizada e pode ser considerado como o padrão de verificadores de modelo simbólico e por motivo é a ferramenta utilizada neste projeto.

Para a etapa de modelagem do processo de verificação de modelo utilizará a linguagem SMV, a qual é entendida pela ferramenta NuSMV. Na etapa de especificação utiliza-se lógica temporal (CTL) e o processo de verificação usa a técnica symbolic model checking nativa da ferramenta NuSMV.

Ao coordenador do projeto caberá a definição e a especificação das funções veiculares a serem implementadas. Ressalta-se que o coordenador do projeto possui experiência profissional em empresas do setor automotivo, tais como: montadoras e fornecedoras de soluções tecnológicas. Tal experiência o qualifica para definir o escopo das funções veiculares que serão especificadas, modeladas, implementadas e posteriormente verificadas. Para a implementação das funções veiculares serão adotadas as ferramentas disponíveis no produto de software Matlab/Simulink (Matlab/Simulink,2021). A equipe coordenadora do projeto se responsabiliza pela disponibilização do produto de software Matlab/Simulink aos discentes membros do projeto.

Assim para o cumprimento do objetivo do projeto são necessárias as etapas/metas definidas, descritas a seguir:

- Etapa 1 – Estudos bibliográficos e Modelagem de Funções Veiculares: estudar, entender e compreender funções veiculares. Modelagem de funções veiculares usando diagrama de estados finitos.
- Etapa 2 – Estudos bibliográficos a respeito de Métodos Formais: estudar teoria a respeito de métodos formais e lógica temporal. Estudar as técnicas de verificação de modelos e verificação de modelos simbólica.
- Etapa3 – Ferramentas para Implementação de Funções Veiculares: estudar, entender e compreender as ferramentas do produto de software Matlab/Simulink necessárias para a implementação de funções veiculares;
- Etapa 4 – Ferramentas para Verificação Formal: estudar e compreender a linguagem SMV e a ferramenta NuSMV;
- Etapa 5 – Desenvolvimento e Teste: implementar e testar as funções veiculares especificadas/modeladas por diagramas de estados finitos;
- Etapa 6 – Verificação Formal: realizar a verificação de modelos nas funções veiculares implementadas;
- Etapa 7 – Resultados: analisar os resultados da aplicação de verificação de modelos obtidos na etapa 6;
- Etapa 8 – Documentação: elaborar o relatório final em formato de artigo.

O Quadro 1 apresenta o cronograma deste projeto de iniciação científica para o tempo de execução previsto no edital, ou seja, 8 meses.

**Quadro 1:** Cronograma de execução do trabalho

| ETAPAS | MÊS DE EXECUÇÃO DA ETAPA | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | X | X | X | | | | | |
| 2 | X | X | X | | | | | |
| 3 | | X | X | | | | | |
| 4 | | | X | X | | | | |
| 5 | | | | X | X | X | | |
| 6 | | | | | X | X | | |

ISSN 2558-6052

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7 | | | | | | X | X | |
| 8 | | | | | | | | X |

O cronograma prevê mais de uma atividade por mês porque se planejou a participação de até três discentes no projeto, contudo tem-se dois alunos participantes do projeto, mas as atividades foram ajustadas pelos professores responsáveis com objetivo de viabilizar o desenvolvimento e entrega esperado.

**RESULTADOS E DISCUSSÕES:**

Inicialmente foi realizado um treinamento no qual os alunos tiveram, em sua grande maioria, o primeiro contato com a ferramenta Stateflow do Software Matlab/Simulink. Este contato se deu por treinamento realizado pelo aluno de graduação Pedro Ivo, que desenvolveu roteiros de treinamento como base do seu TCC. Ao longo do treinamento foram trabalhados sete projetos que buscavam familiarizar os alunos com o ambiente de trabalho e as ferramentas.

Em seguida foram realizados testes com as funções implementadas no treinamento (inserir dados dos testes das funções no treinamento). Simultaneamente foi implementada uma função veicular (VF) teste para após implementar a VF de Cruise Control, ambas apresentadas à equipe pelo orientador do projeto, profissional com expertise no segmento automotivo. Destaca-se que não foram funções reais, uma vez que as montadoras não disponibilizam esse tipo de informação, mas segundo o próprio professor se aproxima das VFs quando o mesmo atuava no mercado. Devido ao grau de complexidade da função veicular de Cruise Control está foi dividida nas seguintes partes:

- CSM – Representado por função do Matlab que traduz os sinais do Column Switch para o formato usado na BCM;
- IPC – Representado como indicador luminoso recebe da BCM um sinal sobre o estado de funcionamento do Cruise Control. Representados as interconexões e o ponto de interesse ampliado;
- BCM – Responsável por traduzir os sinais da CSM para serem utilizados pela ECM e recebe o sinal que irá para o IPC da ECM (**Figura 1**);
- preECM – Usando a mesma lógica da CSM este é um módulo virtual que evita confusões com os nomes dos sinais e permite implementar adaptações que tornam essa função veicular auto suficiente e testável;
- ECM – Usa informações de quatro outras funções veiculares e da BCM para determinar o atual estado do Cruise Control e indiretamente as ações que ele irá executar (**Figura 2**).

Ao final do desenvolvimento da função veicular de Cruise Control foi acrescentado um contador de mensagens (MSGCounter) e as partes individuais foram integradas. Na implementação ainda foi necessário utilizar um bloco somador pronto que é integrado ao Simulink e as memórias foram substituídas por delays. Pois é preciso garantir que não ocorram duas mudanças entre uma leitura e outra. O intervalo de atualização do projeto foi padronizado em 50ms para o principal parâmetro (leitura do status do Cruise Control na lâmpada no Painel de Instrumentos (sistema IPC). Além disso foram feitas correções visuais no diagrama usando a implementação de buses que reduzem a poluição visual por fios (**Figura 3**) e ainda foi implementado o input e output da velocidade para teste na simulação.

Com a evolução no desenvolvimento das funções veiculares citadas, deu-se também a modelagem em SMV da VF utilizando o modelo feito no MatLab, tendo em vista a separação da complexidade da função veicular foi implementado as partes separadas da VF para realizar a junção após. Levando em consideração as regras que a linguagem do SMV impõe, teve-se certas mudanças em como abstrair o modelo feito em MatLab como por exemplo o uso de contadores e a ideia de tempo, mudanças essas que não comprometem o comportamento da VF. As VF's foram implementadas na linguagem SMV e parte desta implementação se mostra nas **Figuras 4, 5 e 6**.

Por fim, nas últimas reuniões foi discutido formas de realizar a especificação do sistema da VF para realizar a verificação do modelo, tal que será realizado uma leitura do documento que especifica a função veicular pelo aluno que realizou a modelagem em MatLab e será feita a especificação em lógica temporal CTL.

seminário de iniciação científica
ISSN 2558-6052

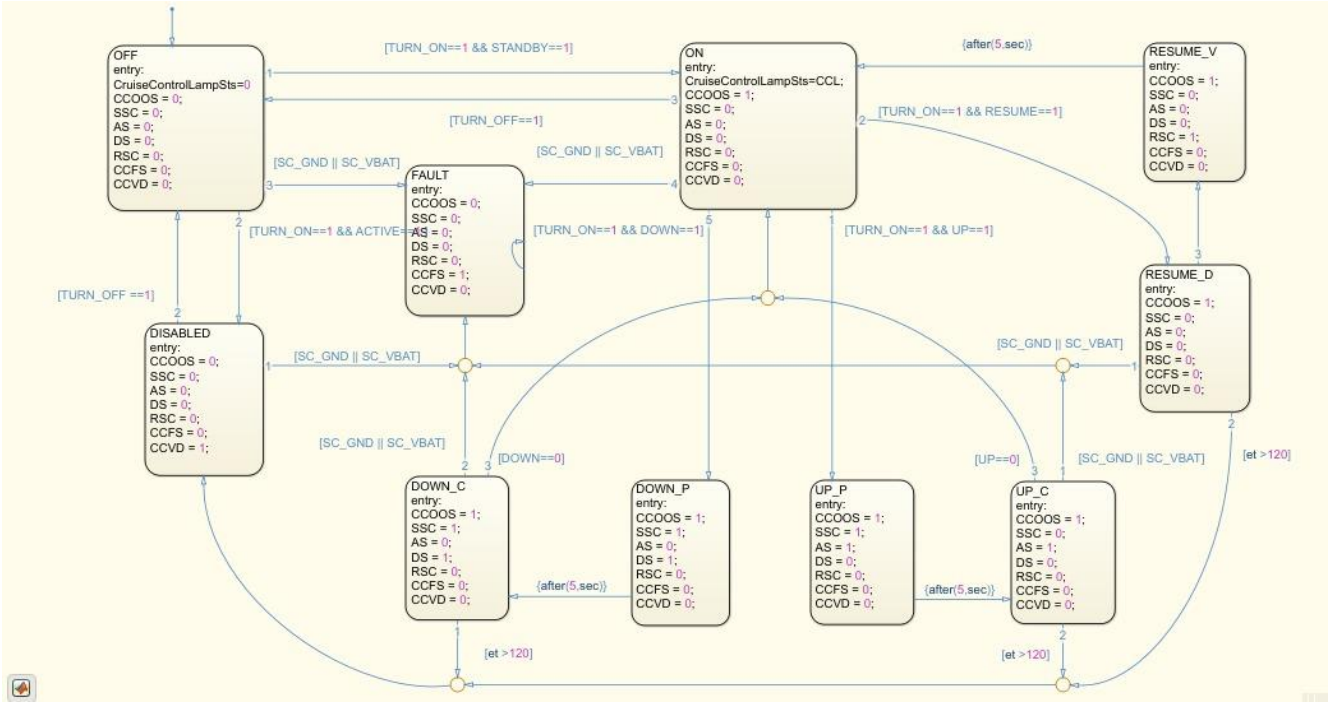


**Figura 1:** Representação da modelagem para o fluxo de estados da BCM.

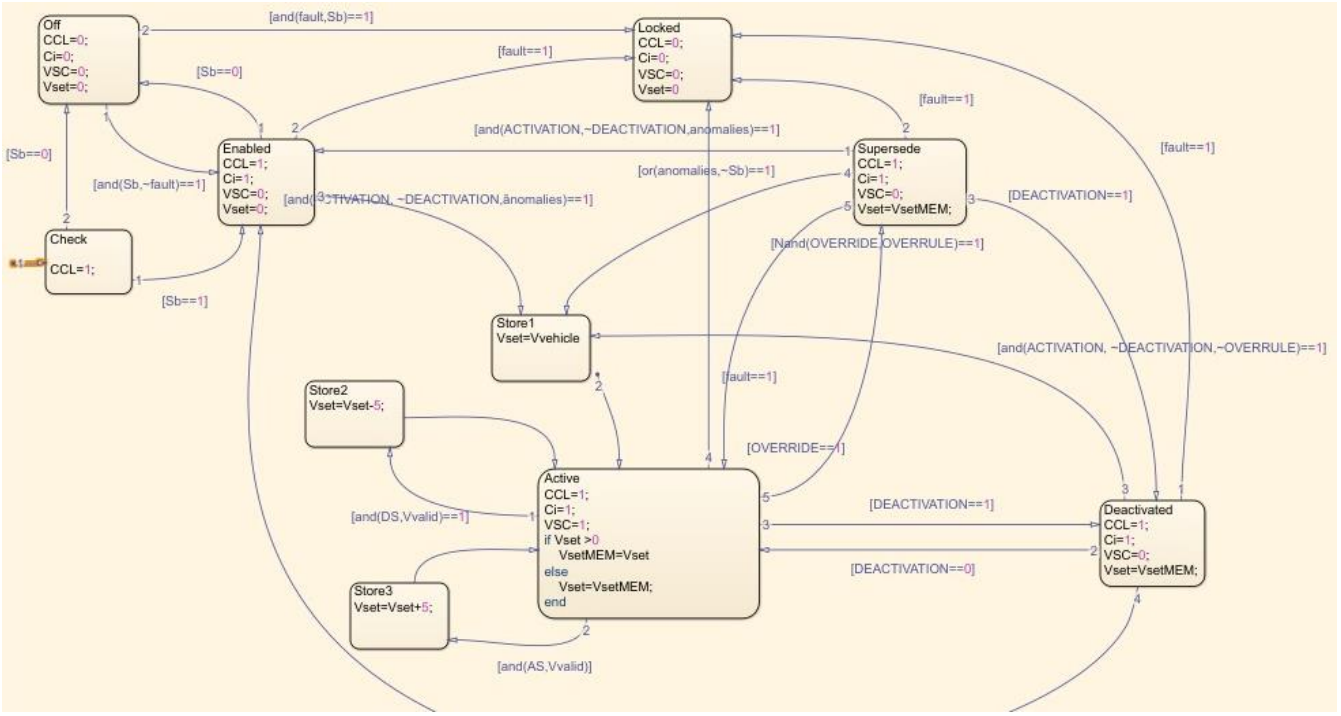


**Figura 2:** Representação da modelagem para o fluxo de estados da ECM

seminário de iniciação científica
ISSN 2558-6052

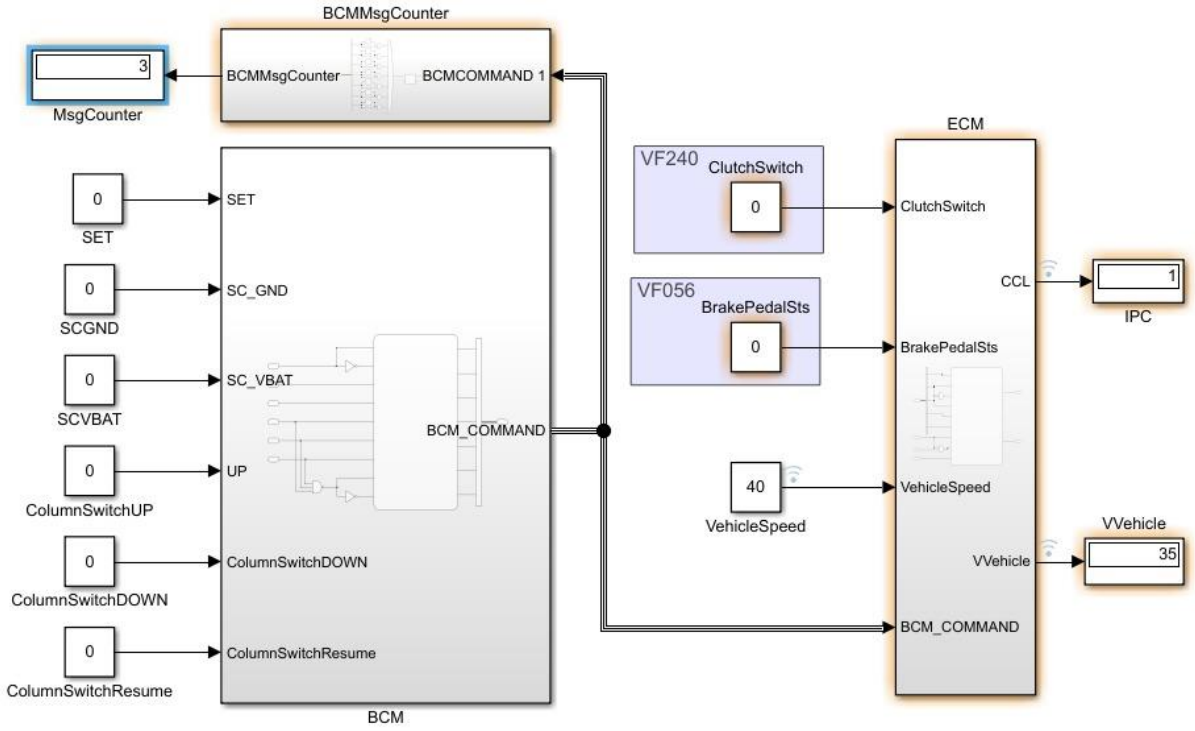


**Figura 3:** Diagrama com ligações corrigidas

```
MODULE main

VAR
    estado: {Check, Off, Enabled, Deactivated, Active, Store1, Store2, Store3, Locked, Supersede};
--variaveis da VF
    CCOOS: boolean;
    SSC: boolean;
    CCFS: boolean;
    CCVD: boolean;
    UP: boolean;
    DOWN: boolean;
    ValidVehicleSpeed: boolean;
    --variaveis da VF056
        BrakePedalSts: boolean;
    --variaveis da VF240
        ClutchSwitch: boolean;
--variaveis de saida
    CCL: boolean;
    Vvehicle: 0..201;
--variaveis de entrada
    Sb: boolean;
    fault: boolean;
    anomalies: boolean;
    AS: boolean;
    ACTIVATION: boolean;
    DS: boolean;
    DEACTIVATION: boolean;
    OVERRIDE: boolean;
--variaveis internas
    Vset: 0..201;
    VSC: boolean;
    Ci: boolean;
```

**Figura 4:** Modelagem das variáveis da função veicular em SMV

ISSN 2558-6052

```
--TRANSICOES DA MODELO ECM
    next(estado):= case
                    (estado = Check) & (!Sb) = TRUE : Off;
                    (estado = Check) & (Sb) = TRUE : Enabled;
                    (estado = Off) & (Sb) & (!fault) : Enabled;
                    (estado = Off) & (Sb) & (fault) : Locked;
                    (estado = Enabled) & (!Sb) : Off;
                    (estado = Enabled) & (fault) : Locked;
                    (estado = Enabled) & (ACTIVATION) & (!DEACTIVATION) & (!anomalies) : Store1;
                    (estado = Supersede) & (ACTIVATION) & (!DEACTIVATION) & (!anomalies): Enabled;
                    (estado = Supersede) & (fault) : Locked;
                    (estado = Supersede) & (DEACTIVATION) : Deactivated;
                    (estado = Supersede) & (anomalies | !Sb) : Store1;
                    (estado = Supersede) & (OVERRIDE) : Active;
                    (estado = Deactivated) & (fault) : Locked;
                    (estado = Deactivated) & (!DEACTIVATION) : Active;
                    (estado = Deactivated) & (ACTIVATION) & (!DEACTIVATION): Store1;
                    (estado = Deactivated) & (anomalies | !Sb): Enabled;
                    (estado = Active) & (DS) & (ValidVehicleSpeed): Store2;
                    (estado = Active) & (AS) & (ValidVehicleSpeed) : Store3;
                    (estado = Active) & (DEACTIVATION) : Deactivated;
                    (estado = Active) & (fault) : Locked;
                    (estado = Active) & (OVERRIDE) : Supersede;
                    (estado = Store1) = TRUE : Active;
                    (estado = Store2) = TRUE : Active;
                    (estado = Store3) = TRUE : Active;
                    TRUE: estado;
                esac;
```

**Figura 5:** Modelagem das variáveis da função veicular em SMV

```
ASSIGN

    init(estado):= Check;

--TRANSICOES DAS VARIAVEIS DE ENTRADA
    next(DEACTIVATION):= case
                            BrakePedalSts | ClutchSwitch = TRUE : TRUE;
                            !(BrakePedalSts | ClutchSwitch) = TRUE : FALSE;
                            TRUE: DEACTIVATION;
                         esac;

    next(OVERRIDE):= case
                        ClutchSwitch = TRUE : TRUE;
                        ClutchSwitch = FALSE : FALSE;
                        TRUE: OVERRIDE;
                     esac;

    next(Sb):= case
                CCOOS = TRUE : TRUE;
                CCOOS = FALSE : FALSE;
                TRUE: Sb;
               esac;

    next(fault):= case
                    CCFS = TRUE : TRUE;
                    CCFS = FALSE : FALSE;
                    TRUE: fault;
                  esac;
```

**Figura 6:** Modelagem das variáveis da função veicular em SMV

**CONCLUSÕES:**

Com a conclusão deste projeto espera-se que os seguintes resultados sejam alcançados:

- Capacitação de recurso humano na área de métodos formais, especificamente na técnica verificação de modelos;
- Aplicação da técnica verificação de modelos nas funções veiculares implementadas;

Os produtos de software e hardware desenvolvidos atualmente envolvem alto nível de complexidade. A abordagem tradicional para verificação envolve realizar testes e simulações. Contudo tais abordagens não exploram todo o universo de possíveis estados do sistema e assim não são apropriadas para garantir que um produto de software ou hardware está isento de erros. Portanto, o uso do Model Checking permite que verifiquemos todo o universo de possíveis estados do sistema, transformando anos de testes e simulações em um tempo razoavelmente menor. A Verificação de Modelos também permite que os erros que os métodos de simulação ou validação técnica sejam identificados nas fases iniciais do projeto, permitindo assim evitar perdas de tempo e dinheiro.

A abordagem visual computacional com modelos práticos deu aos alunos um suporte muito forte para o aprendizado de Lógica Temporal. A distância entre o treinamento e as aplicações reais agora reside apenas na familiarização com conceitos e na sinergia com os métodos. A ordem em que os conhecimentos foram passados permitiu que os alunos enxergassem mais rapidamente vieses da intuição que poderiam prejudicar a aplicação da Verificação de Modelos. Os modelos de Máquina de Estados Finitos são comuns na indústria automobilística e vários problemas evitáveis acontecem por falta de profissionais capacitados não somente na Modelagem do Sistema Física, mas também na análise por métodos formais do modelo concebido.

**REFERÊNCIAS BIBLIOGRÁFICAS:**

Bryant, Randal E. **Graph-Based Algorithms for Boolean Function Manipulation**. IEEE Transactions on Computer. Volume C-25. Edição: 8. 1986.

Caldas, Ruiter Braga. **Modelagem, Verificação Formal e Codificação de Sistemas Reativos Autônomos**. 2009. Tese doutorado em Ciência da Computação. UFMG. Belo Horizonte. Disponível em https://www.dcc.ufmg.br/pos/cursos/defesas/279D.PDF. Acesso em 25 fevereiro de 2021.

Clarke, Edmund M. et al.. **Model Checking. 2ª edição**. MIT Press. Londres, 2018.

CUNHA, Alessandro F. **O que são sistemas embarcados?**. 2011. [s.l: s.n.].

Ferreira, Nelson Franca Guimarães. **Verificação Formal de Sistemas Modelados em Estados Finitos**. 2006. Dissertação mestrado em Engenharia. USP. São Paulo. Disponível em: http://www.teses.usp.br/teses/disponiveis/3/3141/tde-19092006-134100/en.php. Acesso em 24 fevereiro de 2021.

Flores, Fabio Henrique Lacerda. **Verificação formal na indústria**. 2016. Monografia especialização em Informática - UFMG. Belo Horizonte. Disponível em: https://repositorio.ufmg.br/handle/1843/ESBF-A9EGUS. Acesso em 25 fevereiro de 2021.

Genta G., Morello L.(eds) **The Automotive Chassis. Volume 2: System Design**. Mechanical Engineering Series. Springer, 2009.

Gleick, James. **A Bug and a Crash**. http://www.around.com/ariane.html, 1996. Acesso em 14 fevereiro de 2021.

Lions, J. L. ARIANE 5 Flight 501 Failure.http://sunnyday.mit.edu/nasa-class/Ariane5-report.html. Acesso em 23 fevereiro de 2021.

Matlab/Simulink (2021). Disponível em: https://www.mathworks.com. Acesso em 14 fevereiro de 2021.

McMillan, Kenneth L. **Symbolic Model Checking: An approach to the state explosion problem**. 1992. Tese de Doutorado. Carnegie Mellon University. Disponível em https://apps.dtic.mil/sti/pdfs/ADA250924.pdf. Acesso em 25 fevereiro de 2021.

**Projeto NuSMV** (2021). Disponível em: http://nusmv.fbk.eu. Acesso em 12 fevereiro de 2021.

SOUZA, Andrey Gustavo de; CAMPOS, Gustavo Lobato. **Rede can veicular: levantamento bibliográfico e apresentação de conceitos iniciais**. ForScience: revista científica do IFMG, Formiga, v. 5, n. 1, e00234, jan./jun. 2017.