Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

LAÍS LEAL MENDES

SIMULAÇÃO DE CONVERSORES CC-CC EM TEMPO REAL COM FPGA

Campina Grande, Paraíba
Julho de 2015

LAÍS LEAL MENDES

SIMULAÇÃO DE CONVERSORES CC-CC EM TEMPO REAL COM FPGA

Relatório de estágio supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento de Energia

Orientador:
Professor Alexandre Cunha Oliveira, D. Sc.

Campina Grande, Paraíba
Julho de 2015

LAÍS LEAL MENDES

SIMULAÇÃO DE CONVERSORES CC-CC EM TEMPO REAL COM FPGA

Relatório de estágio supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento de Energia

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professor Alexandre Cunha Oliveira, D. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Dedico este trabalho a Deus primeiramente, por ter me permitido chegar até aqui e aos meus pais, os quais jamais mediram esforços na minha educação e sempre estiveram ao meu lado.

AGRADECIMENTOS

Agradeço a Deus, por cada etapa vencida para chegar até aqui, nas quais Ele sempre esteve ao meu lado.

Agradeço a UFCG, pelas condições oferecidas, que me permitiram concluir este trabalho.

Agradeço aos meus pais, Luiza e Riró, por todo o esforço realizado na minha educação, pelos valores morais que me ensinaram e por todo o apoio que sempre me dão.

Agradeço em especial ao meu orientador, Prof. Alexandre Cunha Oliveira, pela enorme paciência, pela amizade e pelo tempo dedicado a cada dúvida, a cada detalhe desse trabalho, o qual não seria possível sem o seu apoio.

Agradeço aos amigos feitos nesse curso, Rayan, Milena, Plateny, Thiago e Vidal, os quais sempre estiveram ao meu lado em toda a caminhada.

Enfim, agradeço a todos que de alguma forma, passaram pela minha vida e sonharam esse sonho comigo.

"Porque são nossos sonhos
e só nós sabemos o quanto
nos custa sonhá-los...."

RESUMO

Este trabalho consiste no relatório final elaborado ao término das atividades de Estágio Supervisionado, realizado no Laboratório de Eletrônica Industrial e Acionamento de Máquinas (LEIAM) da Universidade Federal de Campina Grande (UFCG). No estágio realizou-se a simulação de conversores CC-CC utilizando FPGA, onde foi desenvolvida a modelagem em tempo real dos conversores. Encontram-se aqui descritas as etapas do estágio, desde o desenvolvimento dos modelos dos conversores até sua implementação.

Palavras-chave: Conversores CC-CC. Conversores com FPGA. Conversor Boost. Conversor Buck. Conversor Buck-Boost.

ABSTRACT

This work is the final report that must be done at the end of internship activities, held at the Laboratory of Industrial Electronics and Machinery Driving (LEIAM) located at the Federal University of Campina Grande (UFCG). The internship was based on simulation of CC-CC converters using FPGA, where the real time modeling of the converters was done. The steps necessary to successfully accomplish the internship tasks are described in this document, from the development of the converters' models to them implementation.

Keywords: CC-CC Converters. Converters with FPGA. Boost converter. Buck Converter. Buck-Boost converter.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Este trabalho consiste na descrição das atividades desenvolvidas durante a disciplina de Estágio Supervisionado realizado no Laboratório de Eletrônica Industrial e Acionamento de Máquinas (LEIAM) da Universidade Federal de Campina Grande (UFCG). No estágio foi realizada a simulação de Conversores CC-CC utilizando FPGA, na qual é feita a modelagem em tempo real dos conversores.

O LEIAM localiza-se no bloco CH da UFCG e desenvolve atualmente projetos na área de qualidade de energia, fontes alternativas de energia e acionamento e controle de máquinas elétricas. Algumas empresas como a Eletrobrás financiam pesquisas realizadas no LEIAM.

Os projetos de pesquisa do laboratório são desenvolvidos pelos professores e alunos do doutorado, mestrado e graduação. A infraestrutura do laboratório dispõe de computadores, dispositivos de medição, instrumentação e aquisição de dados, máquina elétricas e dispositivos de eletrônica de potência.

Montagens utilizando indutores são uma das maiores dificuldades encontradas neste laboratório, uma vez que não existem muitas opções comerciais e desenvolver um indutor manualmente nem sempre é viável. Algumas montagens deste laboratório passam meses à espera do indutor adequado.

Este estágio teve como proposta, a simulação de conversores CC-CC em tempo real com o uso de FPGA. Com ele, objetivou-se observar a possibilidade de simulação em tempo real de determinados circuitos que utilizam indutores. Como os resultados obtidos foram satisfatórios e muito próximos dos reais, o uso de simulação em tempo real torna-se uma alternativa aos circuitos que utilizam indutores deste laboratório.

As etapas desenvolvidas no estágio foram: estudo dos circuitos dos conversores a serem desenvolvidos (Buck, Boost e Buck-Boost), discretização e obtenção da equação de diferenças dos modelos dos conversores CC-CC, modelagem dos conversores em FPGA e a análise comparativa dos resultados simulados com os resultados experimentais.

Nesse trabalho é feita inicialmente uma fundamentação teórica para explicitar o funcionamento dos conversores Buck, Boost e Buck-Boost, as equações que os definem e seus modelos discretizados. A etapa seguinte é composta da simulação dos conversores utilizando o PSIM, onde a FPGA é simulada pelo uso de DLLs. Posteriormente é mostrada a implementação dos módulos dos conversores, e da comunicação com o conversor D/A em FPGA. Por fim, é apresentada uma análise comparativa dos resultados simulados com os experimentais.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção é apresentada a fundamentação teórica dos temas que se mostraram indispensáveis para o desenvolvimento desse trabalho.

2.1 PONTO FIXO E PONTO FLUTUANTE

O ponto fixo e o ponto flutuante são representações para números fracionários. A vantagem do ponto flutuante em relação ao ponto fixo é que ele apresenta programação facilitada, uma vez que sua faixa dinâmica é automática.

Nesse trabalho usaremos as duas representações como critério de comparação, mas para a FPGA será usado ponto fixo que é a representação com a qual ela será operada.

Para fazer a transformação das variáveis dos conversores para ponto fixo, foi utilizada a equação:

$$N_Q = \frac{N}{Base} * (2^Q - 1)$$

Onde a "Base" utilizada para tensão foi de 100 V e de corrente foi de 50 V. O "Q" indica a base do ponto fixo. Utilizou-se a representação Q20 para as variáveis e Q17 para as saídas (Tensão no capacitor e corrente no indutor). "N" é a variável e "N_Q'' é a variável em ponto fixo.

2.2 APROXIMAÇÃO DE TUSTIN E EQUAÇÃO DE DIFERENÇAS

Para realizar a discretização dos modelos dos conversores CC-CC para uso no FPGA dois conceitos se fizeram de grande importância: a aproximação de Tustin e a equação de diferenças.

A aproximação de Tustin é uma técnica de discretização simples, na qual a aproximação mantém a característica de linearidade do sistema.

Para obter G(z) a partir de G(s) basta fazer a substituição de variáveis como segue:

$$s = \frac{2(Z-1)}{T(Z+1)}$$

Já a equação de diferenças pode ser escrita utilizando operações de avanço ou de atraso, as duas formulações são equivalentes. Porém para sistemas reais, faz necessário que a equação seja realizável (causal), podendo ser escrita apenas com operações de atraso. O método mais simples de

solução de uma equação de diferenças é o iterativo, que é a solução adequada para computadores. Para isso a equação deve se encontrar na forma:

$$Y[n] = -a_1 Y[n-1] + \cdots - a_n Y[n-N] + b_0 x[n] + \cdots.$$

Os modelos dos conversores, discretizados pela aproximação de Tustin, serão escritos de acordo com essa equação de diferenças, para facilitar a sua solução por computadores.

2.3 CONVERSORES CC-CC

Os conversores CC-CC são amplamente empregados em fontes de alimentação chaveadas e em acionamento de motores de corrente contínua. Apresentam duas topologias básicas: o conversor CC-CC abaixador de tensão (conversor Buck) e o conversor CC-CC elevador de tensão (conversor Boost). A combinação dessas duas estruturas resulta no conversor CC-CC abaixador-elevador de tensão (conversor Buck-Boost).

Os conversores CC-CC possuem dois modos de funcionamento, condução contínua ou condução descontínua. O modo de condução é caracterizado pela corrente no indutor. Em regime permanente, se a corrente não atinge o valor zero, então o conversor está operando no modo de condução contínua (CCM – continuous conduction mode). Se a corrente atinge o valor zero a cada etapa de comutação, então se está operando no modo de condução descontínua (DCM - descontinuous conduction mode). (Petry, 2014)

Os conversores CC-CC simulados nesse estágio apresentam-se em modo de condução contínua.

A seguir serão modelados os três principais tipos de conversores CC-CC. Seu modelo discretizado será determinado para sua implementação em FPGA.

2.3.1 CONVERSOR BOOST

O circuito do conversor Boost pode ser observado na Figura 1.

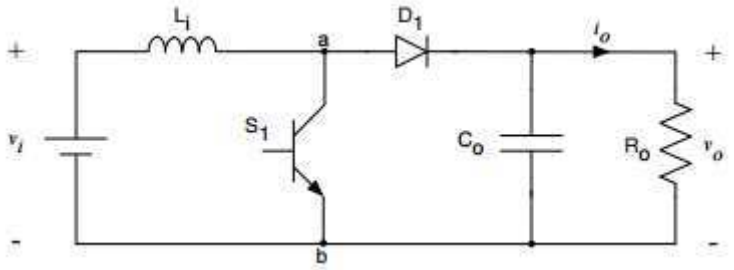

FIGURA 1 CIRCUITO DO CONVERSOR BOOST

Os modos de operação da chave podem ser observados na Figura 2.

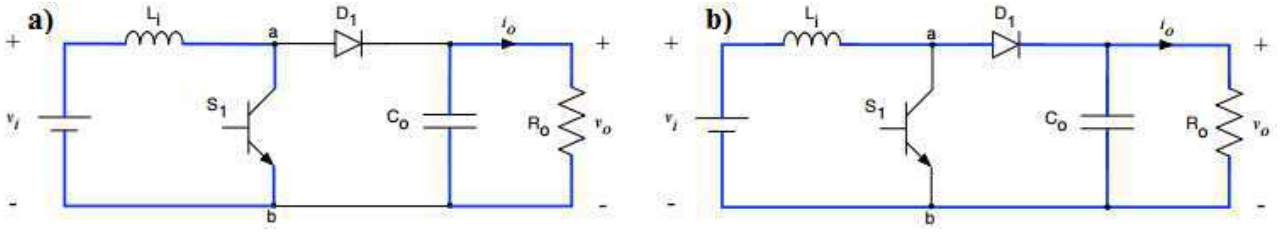

FIGURA 2 CIRCUITO CONVERSOR BOOST: A) CHAVE NA POSIÇÃO 1 B) CHAVE NA POSIÇÃO 2

Quando a chave está na posição 1, temos do lado esquerdo do circuito a):

$$Vl = Vi$$

Sendo $Vl = \frac{Ldi}{dt}$, e aplicando a transformada de Laplace:

$$Vl = L[sI - i(0)]$$

$$Vl = LsI - i(0)L$$

Aplicando a aproximação de Tustin, anteriormente explicada e desprezando i(0), tem-se:

$$Vl = \frac{2(Z-1)}{T(Z+1)}LI \quad (1)$$

Rearranjando a equação (1):

$$TZVl + TVl = 2ZLI - 2LI$$

Dividindo-a por Z:

$$TVl + Z^{-1}TVl = 2LI - 2LIZ^{-1}$$

Portanto:

$$2Li(k) = TVl(k) + TVl(k-1) + 2Li(k-1)$$

$$i(k) = \frac{T}{2L}[Vl(k) + Vl(k-1)] + i(k-1)$$

Como $Vl(k) = Vi$ e $Vi(k-1) = Vi(k) = Vi$, pois a tensão de entrada é constante, tem-se:

$$i(k) = \frac{T}{L}Vi + i(k-1)$$

Onde T é o passo de cálculo.

No lado direito do circuito a), temos:

$$ic = -ir$$

Sendo $ic = \frac{CdVc}{dt}$, e aplicando a transformada de Laplace:

$$Ic = C[sVc - vc(0)]$$

Desprezando $vc(0)$, e aplicando a aproximação de Tustin tem-se:

$$Ic = \frac{2(Z-1)}{T(Z+1)}CVc \quad (2)$$

Rearranjando a equação (2):

$$Vc(k) = \frac{T}{2C}[ic(k) + ic(k-1)] + CVc(k-1)$$

Substituindo $ic(k)$ por $-\frac{Vc(k)}{R}$:

$$Vc(k) = \frac{T}{2C}[-\frac{Vc(k)}{R} - \frac{Vc(k-1)}{R}] + CVc(k-1)$$

$$Vc(k) = \frac{T}{2CR}[Vc(k-1) + Vc(k-2)] + Vc(k-1)$$

Quando a chave está na posição 2:

$$vl = vi - vc$$

$$TZVl + TVl - TZVc - TVc = 2L(ZI - I)$$

$$\frac{T}{2L}[Vl(k) + Vl(k-1) - Vc(k) - Vc(k-1) + i(k-1) = i(k)$$

Portanto:

$$i(k) = \frac{T}{L}Vi - \frac{T}{2L}[Vc(k) + Vc(k-1)] + i(k-1)$$

$$i(k) = \frac{T}{L}Vi - \frac{T}{2L}[Vc(k-1) + Vc(k-2)] + i(k-1)$$

De $ic = il - ir$, temos:

$$Vc(k) = \frac{T}{2CR}[-Vc(k-1) - Vc(k-2)] + \frac{T}{2C}[i(k) + i(k-1)] + Vc(k-1)$$

Em resumo:

Chave na posição 1:

$$i(k) = \frac{T}{L}Vi + i(k-1)$$

$$Vc(k) = \frac{T}{2CR}[Vc(k-1) + Vc(k-2)] + Vc(k-1)$$

Chave na posição 2:

$$i(k) = \frac{T}{L}Vi - \frac{T}{2L}[Vc(k-1) + Vc(k-2)] + i(k-1)$$

$$Vc(k) = \frac{T}{2CR}[-Vc(k-1) - Vc(k-2)] + \frac{T}{2C}[i(k) + i(k-1)] + Vc(k-1)$$

De posse das quatro equações que modelam o funcionamento do Conversor Boost (equações de corrente no indutor e tensão no capacitor para as duas posições da chave) pode-se realizar sua simulação utilizando FPGA.

2.3.2 CONVERSOR BUCK

O circuito do conversor Buck pode ser observado na Figura 3.

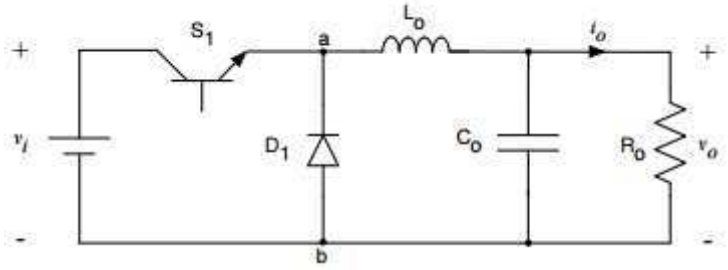

FIGURA 3 CIRCUITO DO CONVERSOR BUCK

Os modos de operação da chave podem ser observados na Figura 4.

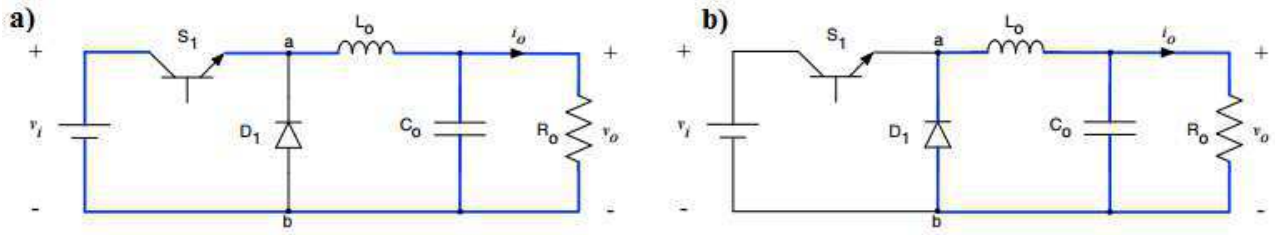

FIGURA 4 CIRCUITO CONVERSOR BUCK: A) CHAVE NA POSIÇÃO 1 B) CHAVE NA POSIÇÃO 2

Analogamente ao desenvolvimento observado na seção 2.3.1 para obtenção das equações que modelam o conversor Boost, foram obtidas as equações que modelam o conversor Buck:

Chave na posição 1:

$$i(k) = \frac{T}{L}Vi - \frac{T}{2L}[Vc(k-1) + Vc(k-2)] + i(k-1)$$

$$Vc(k) = \frac{T}{2C}[i(k) + i(k-1)] - \frac{T}{2CR}[Vc(k-1) + Vc(k-2)] + Vc(k-1)$$

Chave na posição 2:

$$i(k) = \frac{T}{2L}[Vc(k-1) + Vc(k-2)] + i(k-1)$$

$$Vc(k) = \frac{T}{2C}[i(k) + i(k-1)] - \frac{T}{2CR}[Vc(k-1) + Vc(k-2)] + Vc(k-1)$$

De posse das quatro equações que modelam o funcionamento do Conversor Buck (equações de corrente no indutor e tensão no capacitor para as duas posições da chave) pode-se realizar sua simulação utilizando FPGA.

2.3.3 CONVERSOR BUCK-BOOST

O circuito do conversor Buck-Boost pode ser observado na Figura 5.

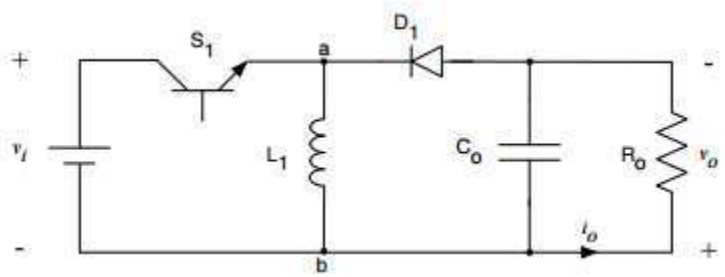

FIGURA 5 CIRCUITO DO CONVERSOR BUCK-BOOST

Os modos de operação da chave podem ser observados na Figura 6.

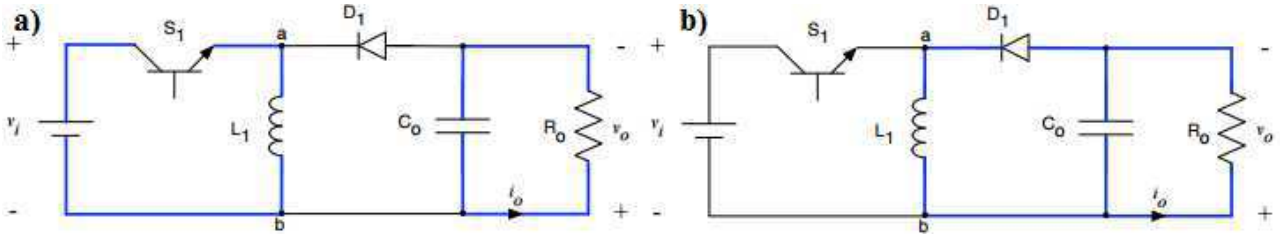

FIGURA 6 CIRCUITO CONVERSOR BUCK-BOOST: A) CHAVE NA POSIÇÃO 1 B) CHAVE NA POSIÇÃO 2

Analogamente ao desenvolvimento observado na seção 2.3.1 para obtenção das equações que modelam o conversor Boost, foram obtidas as equações que modelam o conversor Buck-Boost:

Chave na posição 1:

$$i(k) = \frac{T}{L}Vi + i(k-1)$$

$$Vc(k) = -\frac{T}{2CR}[Vc(k-1) + Vc(k-2)] + Vc(k-1)$$

Chave na posição 2:

$$i(k) = \frac{T}{2L}[Vc(k-1) + Vc(k-2)] + i(k-1)$$

$$Vc(k) = \frac{T}{2C}[i(k) + i(k-1)] - \frac{T}{2CR}[Vc(k-1) + Vc(k-2)] + Vc(k-1)$$

De posse das quatro equações que modelam o funcionamento do Conversor Buck-Boost (equações de corrente no indutor e tensão no capacitor para as duas posições da chave) pode-se realizar sua simulação utilizando FPGA.

3 SIMULAÇÃO DOS CONVERSORES CC-CC NO PSIM

Para a simulação dos conversores CC-CC no PSIM foram desenvolvidos dois circuitos de cada tipo de conversor. Um circuito para simular os conversores com seus componentes elétricos e um circuito para simular a implementação dos conversores em FPGA com o uso de DLLs.

Os códigos das DLLs foram escritos na linguagem de programação C (códigos no Anexo A) e os conversores foram representados pelas equações de diferenças anteriormente explicitadas. Após a simulação, esses códigos foram recodificados em verilog para implementação dos módulos em FPGA.

Analisaram-se as curvas de tensão de saída do conversor e corrente do indutor para os três tipos de conversores aqui representados. A análise para a simulação com FPGA foi ainda dividida em duas: análise em ponto fixo e análise em ponto flutuante.

As entradas das DLLs são os parâmetros do circuito, como o valor do indutor, do capacitor, do resistor e da tensão de entrada e as saídas são as curvas de corrente e tensão do circuito e de seus parâmetros.

3.1 SIMULAÇÃO CONVERSOR BOOST

Para a simulação do conversor Boost, os seguintes parâmetros foram considerados:

$Vi = 10\,V \quad C = 330\,\mu F \quad L = 50\mu H \quad R = 2{,}5\,\Omega \quad Vo = 13\,V \quad f = 20\,kHz \quad T = 100\,ns$

O circuito simulado correspondente ao conversor Boost pode ser observado na Figura 7.

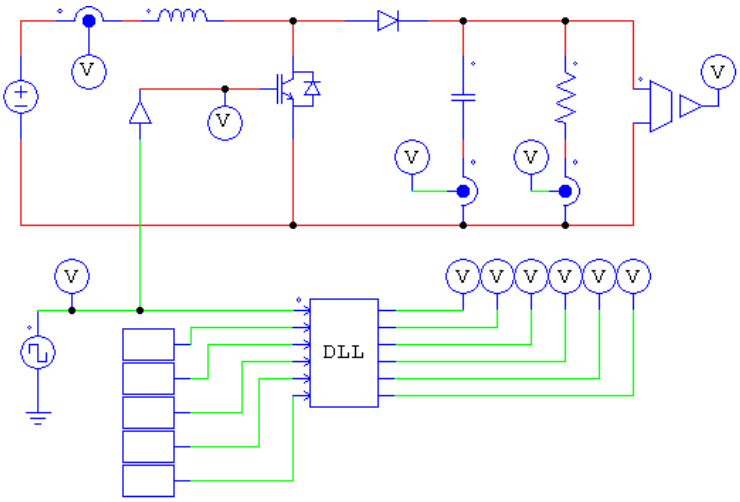

FIGURA 7 SIMULAÇÃO: CIRCUITO E DLL DO CONVERSOR BOOST

Os resultados da simulação encontram-se nas Figura 8, Figura 9, Figura 10 e Figura 11. A linha em vermelho representa a simulação do circuito elétrico, a linha em azul a simulação em ponto fixo e a linha em verde a simulação em ponto flutuante.

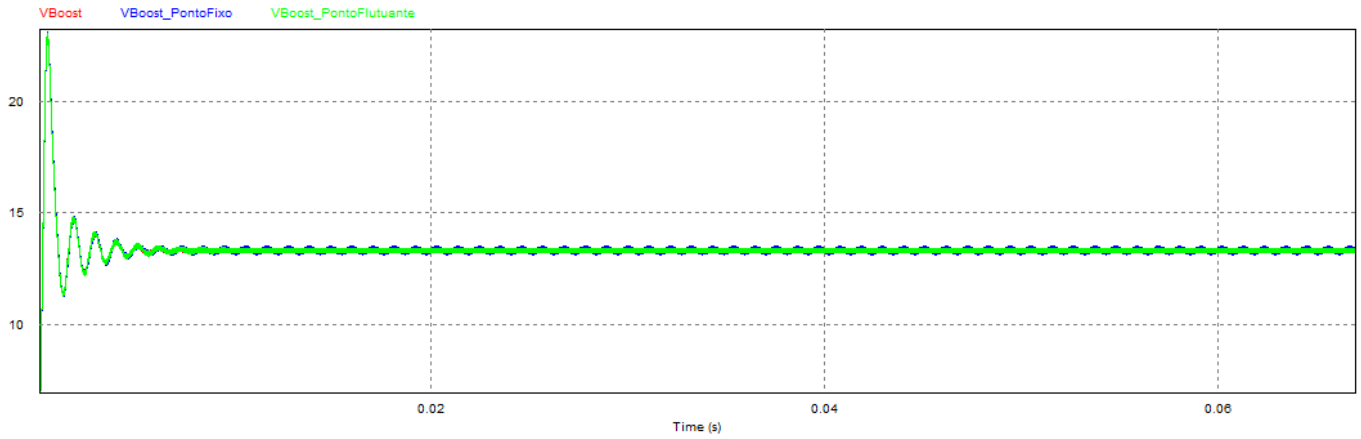

FIGURA 8 TENSÃO DE SAÍDA CONVERSOR BOOST: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

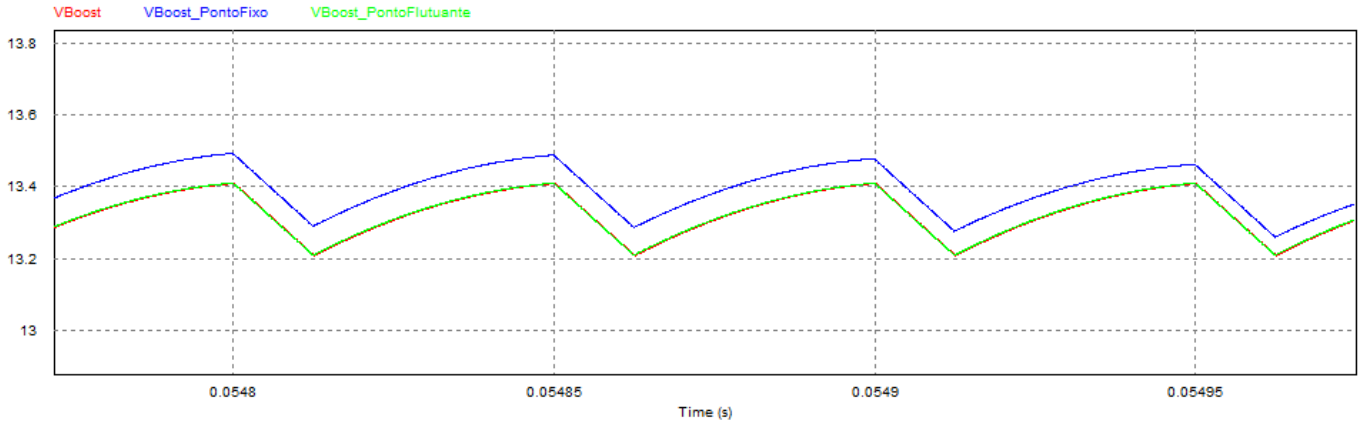

FIGURA 9 TENSÃO DE SAÍDA CONVERSOR BOOST EM REGIME PERMANENTE: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

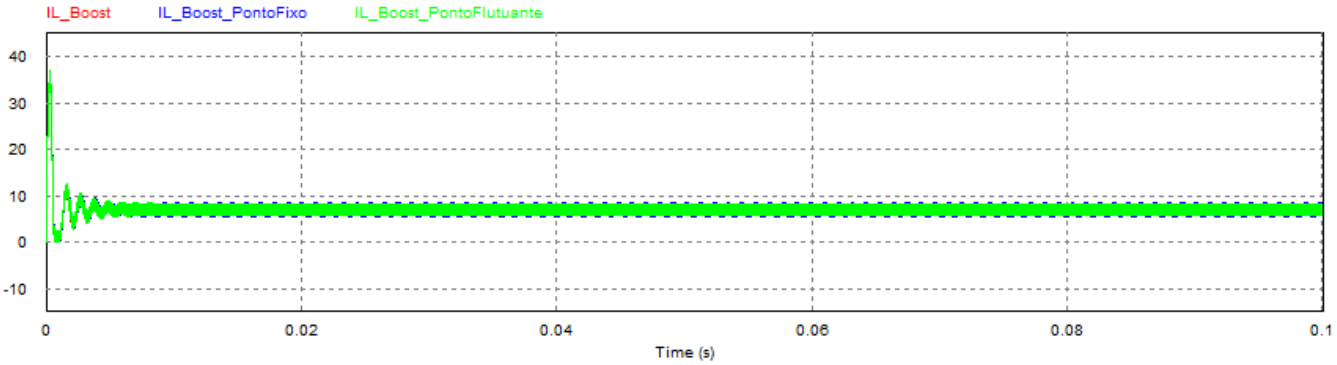

FIGURA 10 CORRENTE NO INDUTOR CONVERSOR BOOST: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

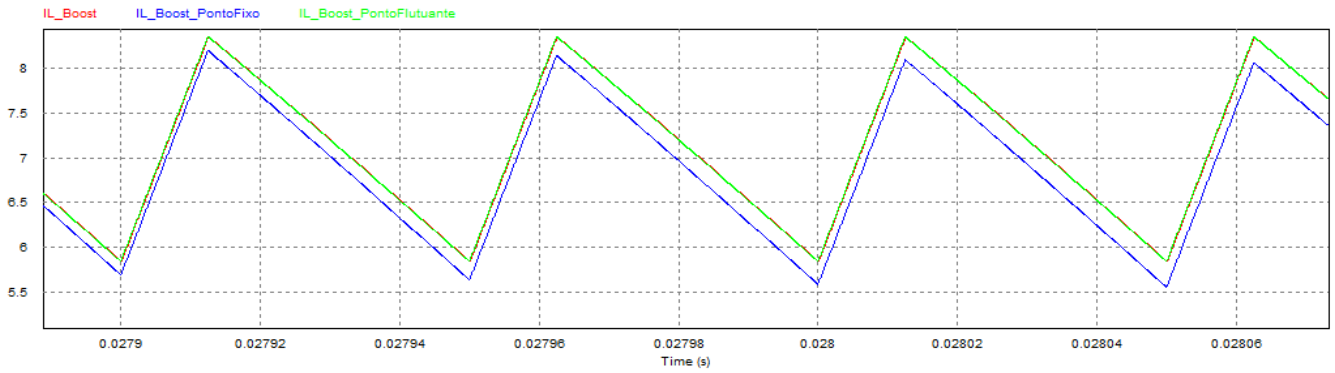

FIGURA 11 CORRENTE NO INDUTOR CONVERSOR BOOST EM REGIME PERMANENTE: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

É possível observar pela análise dos gráficos da Figura 8 e Figura 10 que os circuitos tiveram os resultados esperados tanto para a corrente no indutor quanto para a tensão de saída. Já as Figura 9 e Figura 11 trazem os mesmos gráficos, ampliados no regime permanente, podendo ser observado que o circuito simulado com componentes elétricos e o circuito simulado com DLL em ponto flutuante apresentam-se quase idênticos, apresentando-se ligeiramente diferente apenas o circuito simulado com DLL em ponto fixo, como esperado.

3.2 SIMULAÇÃO CONVERSOR BUCK

Para a simulação do conversor Buck, os seguintes parâmetros foram considerados:

$Vi = 10\,V \quad C = 330\,\mu F \quad L = 50\mu H \quad R = 2{,}5\,\Omega \quad Vo = 5\,V \quad f = 20\,kHz \quad T = 100\,ns$

O circuito simulado correspondente ao conversor Buck pode ser observado na Figura 12.

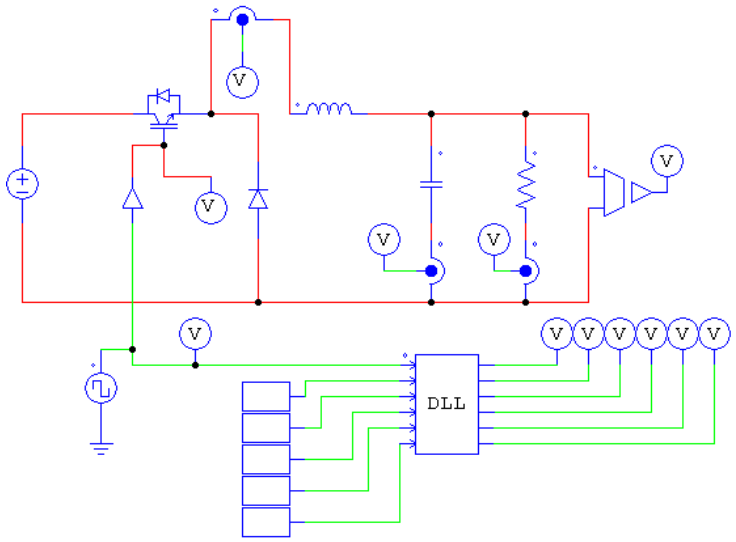

FIGURA 12 SIMULAÇÃO: CIRCUITO E DLL DO CONVERSOR BUCK

Os resultados da simulação encontram-se nas Figura 13, Figura 14, Figura 15 e Figura 16. A linha em vermelho representa a simulação do circuito elétrico, a linha em azul a simulação em ponto fixo e a linha em verde a simulação em ponto flutuante.

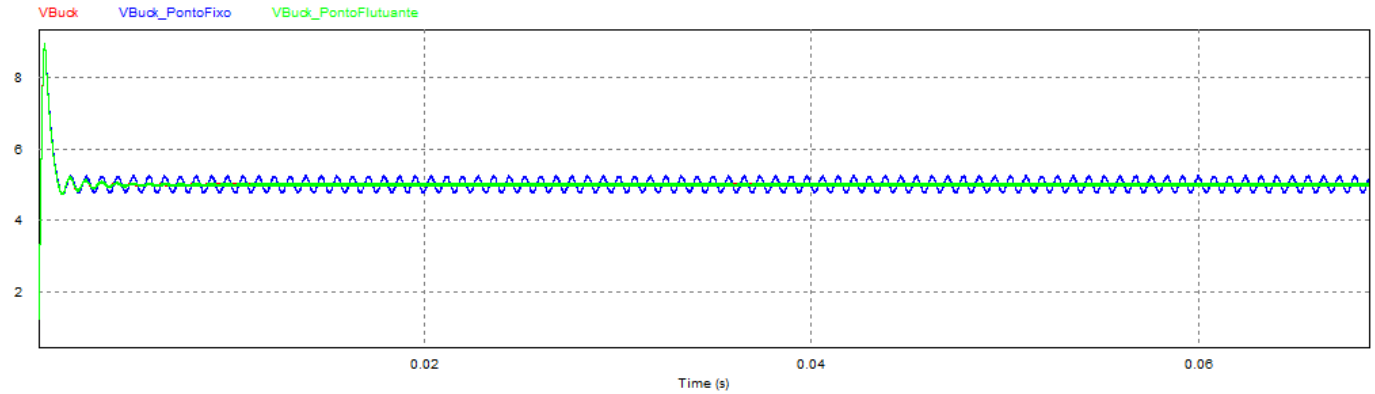

FIGURA 13 TENSÃO DE SAÍDA CONVERSOR BUCK: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

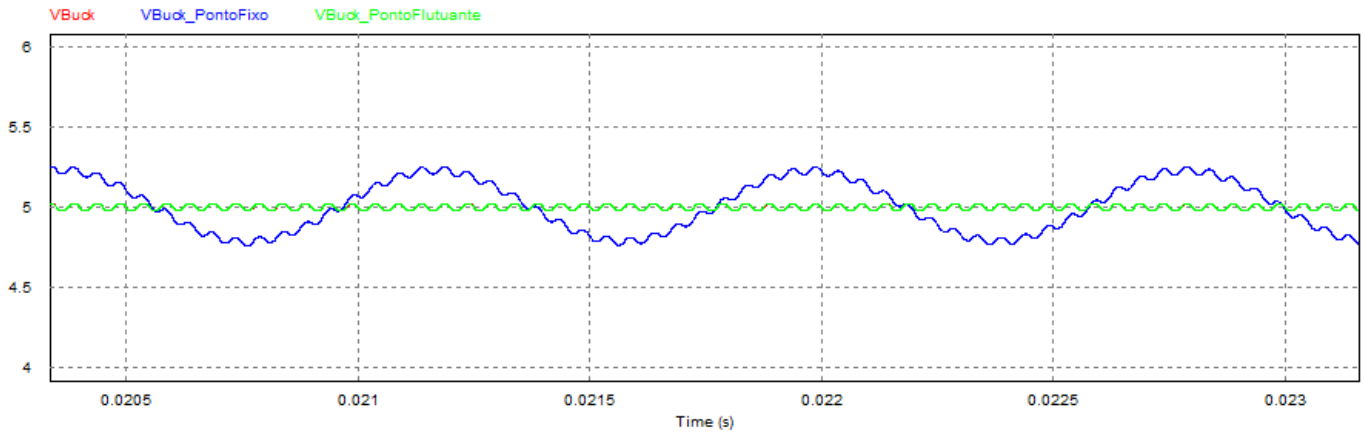

FIGURA 14 TENSÃO DE SAÍDA CONVERSOR BUCK EM REGIME PERMANENTE: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

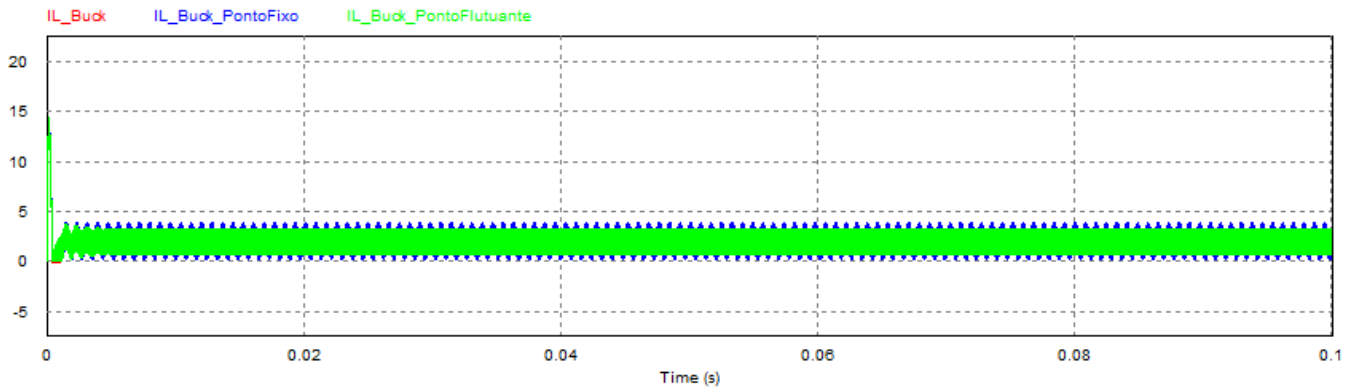

FIGURA 15 CORRENTE NO INDUTOR CONVERSOR BUCK: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

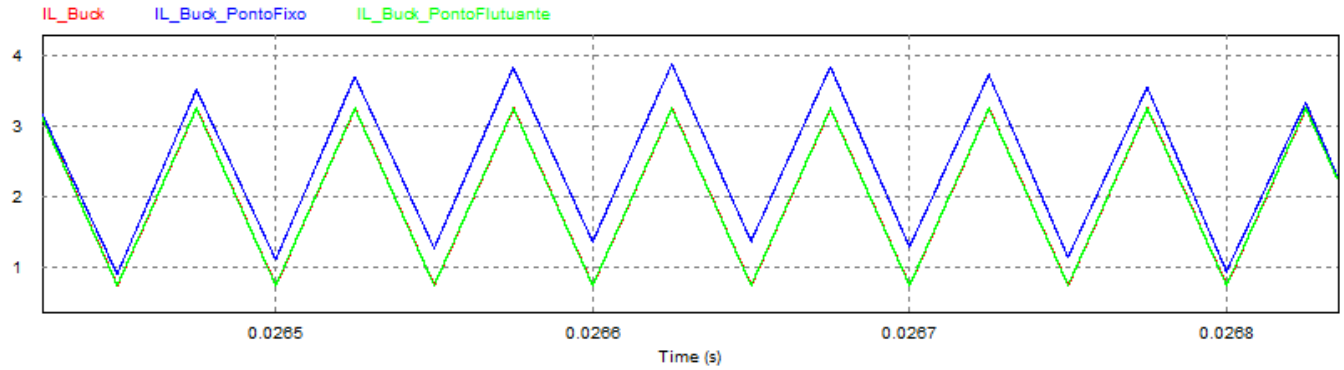

FIGURA 16 CORRENTE NO INDUTOR CONVERSOR BUCK EM REGIME PERMANENTE: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

É possível observar pela análise dos gráficos da Figura 13 e Figura 15 que os circuitos tiveram os resultados esperados tanto para a corrente no indutor quanto para a tensão de saída. Já as Figura 14 e Figura 16 trazem os mesmos gráficos, ampliados no regime permanente, podendo ser observado que o circuito simulado com componentes elétricos e o circuito simulado com DLL em ponto flutuante apresentam-se quase idênticos, apresentando-se ligeiramente diferente apenas o circuito simulado com DLL em ponto fixo, como esperado.

3.3 SIMULAÇÃO CONVERSOR BUCK-BOOST

Para a simulação do conversor Buck-Boost, os seguintes parâmetros foram considerados:

$Vi = 10\,V \quad C = 330\,\mu F \quad L = 50\mu H \quad R = 2{,}5\,\Omega \quad Vo = -10\,V \quad f = 20\,kHz\; T = 100\,ns$

O circuito simulado correspondente ao conversor Buck-Boost pode ser observado na Figura 17.

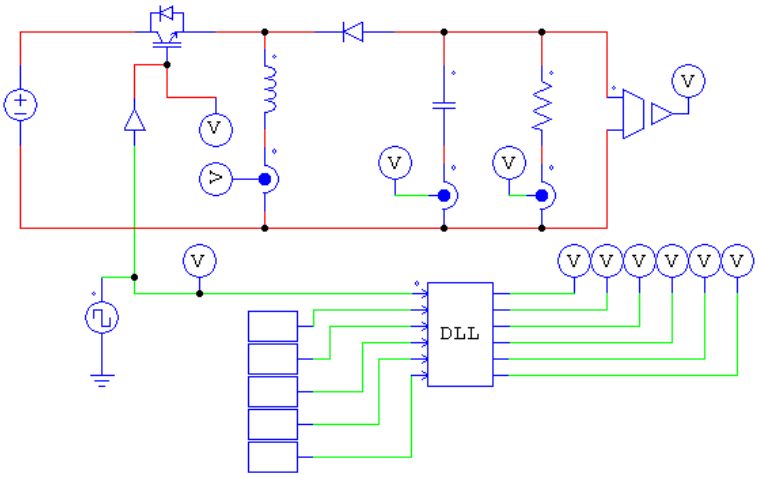

FIGURA 17 SIMULAÇÃO: CIRCUITO E DLL DO CONVERSOR BUCK-BOOST

Os resultados da simulação encontram-se nas Figura 18, Figura 19, Figura 20 e Figura 21. A linha em vermelho representa a simulação do circuito elétrico, a linha em azul a simulação em ponto fixo e a linha em verde a simulação em ponto flutuante.

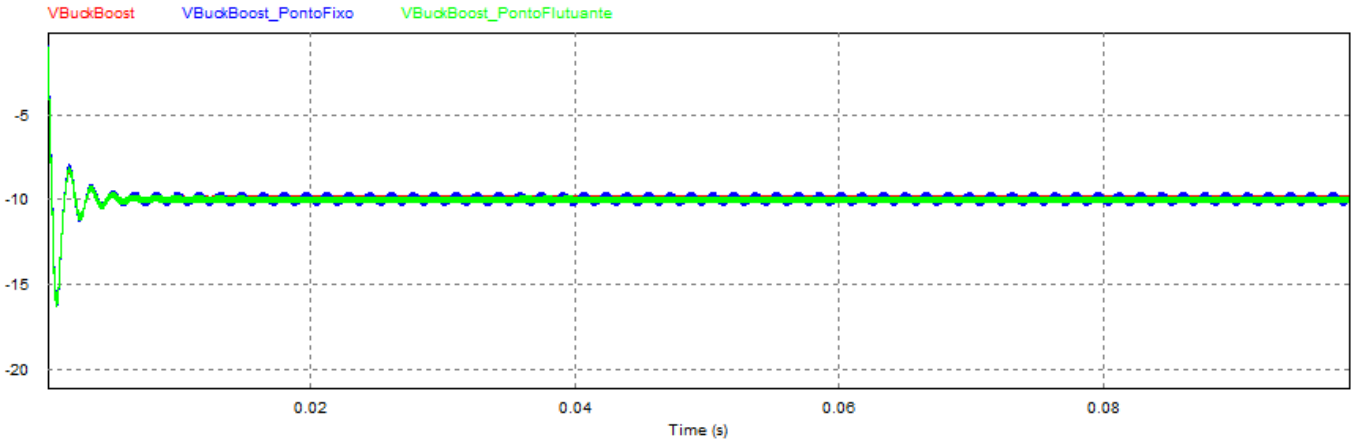

FIGURA 18 TENSÃO DE SAÍDA CONVERSOR BUCK-BOOST: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

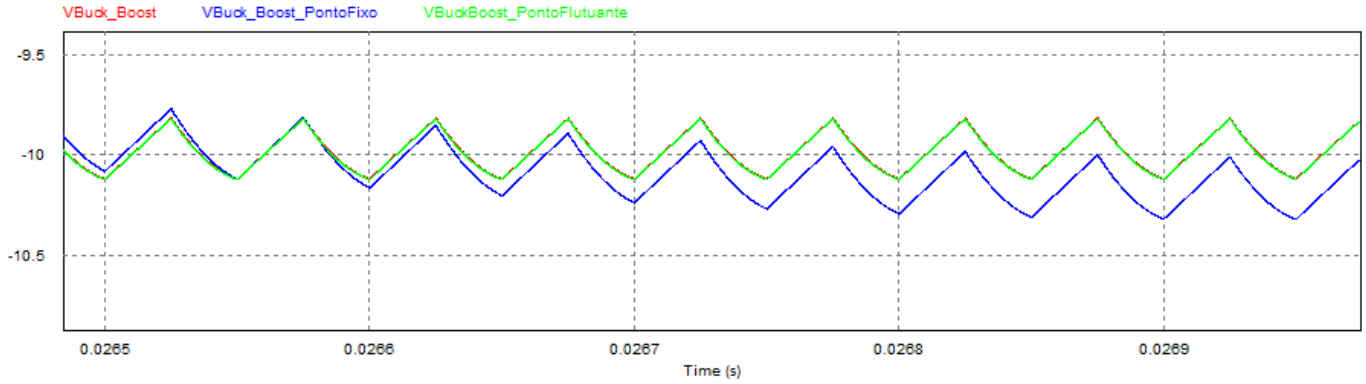

FIGURA 19 TENSÃO DE SAÍDA CONVERSOR BUCK-BOOST EM REGIME PERMANENTE: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

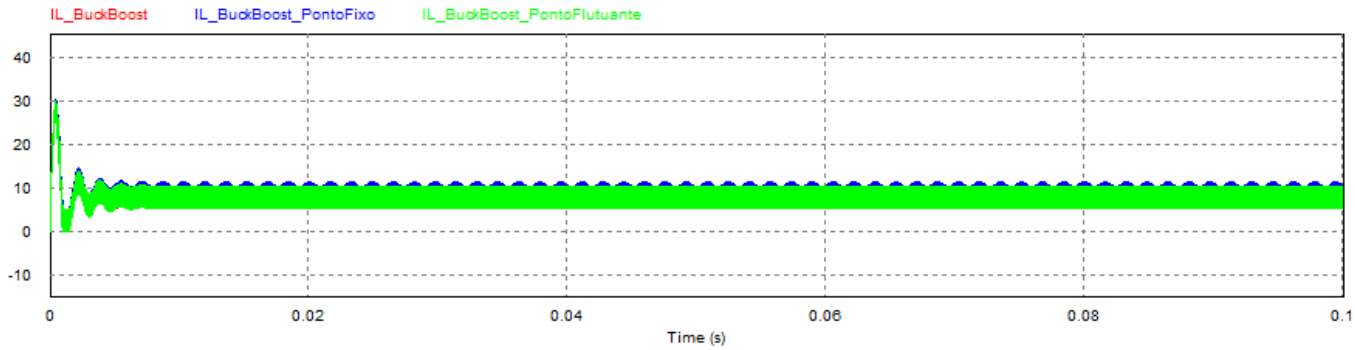

FIGURA 20 CORRENTE NO INDUTOR CONVERSOR BUCK-BOOST: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

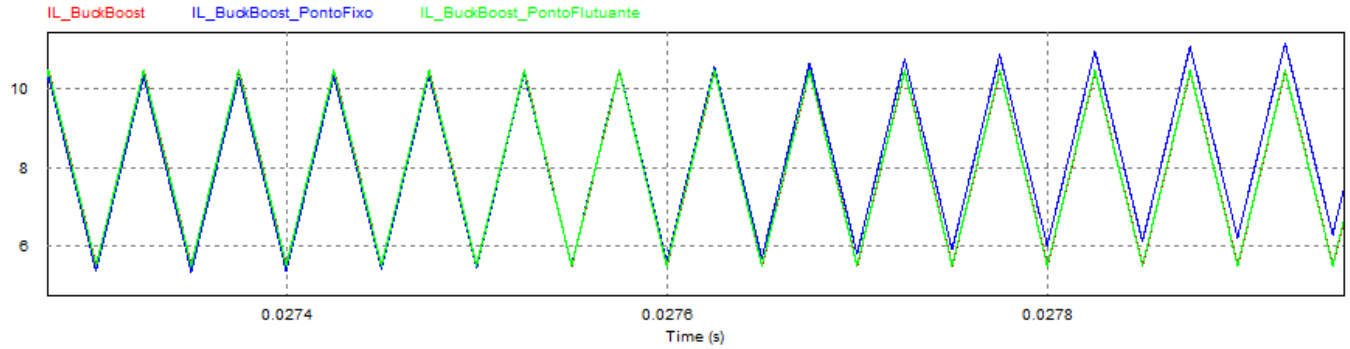

FIGURA 21 CORRENTE NO INDUTOR CONVERSOR BUCK-BOOST EM REGIME PERMANENTE: CIRCUITO, DLL PONTO FIXO, DLL PONTO FLUTUANTE

É possível observar pela análise dos gráficos da Figura 18 e Figura 20 que os circuitos tiveram os resultados esperados tanto para a corrente no indutor quanto para a tensão de saída. Já as Figura 19 e Figura 21 trazem os mesmos gráficos, ampliados no regime permanente, podendo ser observado que o circuito simulado com componentes elétricos e o circuito simulado com DLL em ponto flutuante apresentam-se quase idênticos, apresentando-se ligeiramente diferente apenas o circuito simulado com DLL em ponto fixo, como esperado, sendo a corrente no indutor para o circuito simulado com DLL em ponto fixo o melhor resultado se comparado com os outros dois tipos de conversores descritos nesse trabalho.

4 IMPLEMENTAÇÃO DOS MÓDULOS EM FPGA

A implementação dos módulos em FPGA foi realizada após a comprovação do modelo dos conversores CC-CC realizada utilizando as DLLs no PSIM. Foi utilizado o modelo em ponto fixo. Foram implementados 3 módulos: Mod_test, PLL e SPI. O código completo em verilog encontra-se no Anexo B.

4.1 MÓDULO PLL

O módulo PLL é um módulo pronto, disponível para uso no Quartus II. Colocou-se em sua entrado o clock de 50 Mhz da placa FPGA e ele gerou na saída um clock de 200 Mhz.

4.2 MÓDULO MOD_TEST

No módulo Mod_Test foram realizadas as seguintes implementações:

- Utilização do Clock de 200 Mhz gerado no PLL;
- Geração do sinal de controle dos conversores;
- Contador que conta até 20 para que a cada 100ns os dados sejam calculados. (Passo utilizado na simulação). Cada dado calculado gera um ponto da curva.;
- Geração dos sinais de dados dos conversores de acordo com a opção de conversor escolhida (utilizando as chaves). A cada 50 us uma amostra dos pontos da curva é enviada ao SPI.
- Envio dos oito canais de dados para SPI (1 canal por vez, ou seja, 24 bits por vez);
- Instanciação dos demais módulos.

4.3 MÓDULO SPI

O módulo SPI (Serial Periferic Interface) é a interface periférica serial responsável pela comunicação entre o FPGA e o Conversor D/A. O módulo SPI recebe o dado de simulação vindo do módulo Mod_Test e o transfere para o conversor Digital/Analógico. São 8 canais de dados e 24 bits por canal. A palavra de 24 bits recebida é composta por quatro bits de controle, quatro bits de endereço e 16 bits de dados, sendo esses bits enviados ao conversor do MSB ao LSB.

No diagrama de tempo da Figura 22, é possível observar como é feita a transferência dos bits:

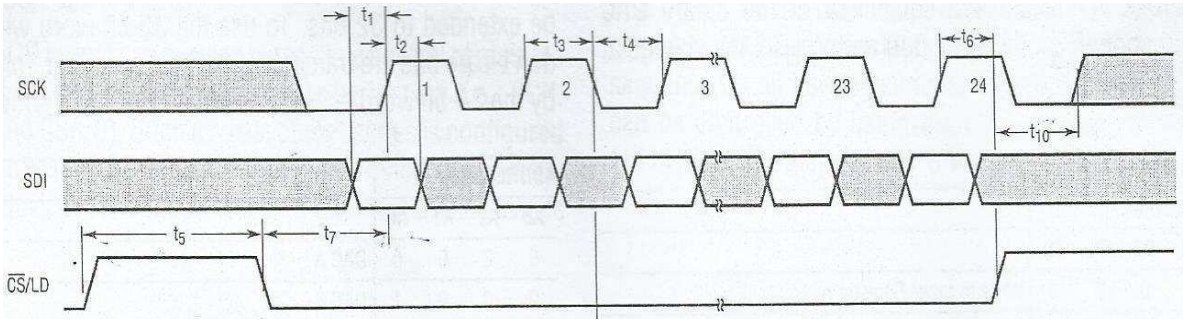

FIGURA 22 DIAGRAMA DE TEMPO SPI

Os bits são recebidos pelo SDI enquanto CS está em zero, quando CS vai a um, o comando é executado. O pulso de clock é dado t_1 segundos após o dado ser recebido, para garantir que o mesmo foi corretamente enviado. O clock mínimo é definido por t3 e t4 como sendo de 50 Mhz, porém utilizamos uma frequência muito menor, sendo o clock utilizado de 50 us.

5 ANÁLISE DOS RESULTADOS

Os resultados obtidos no experimento estão de acordo com o esperado. Todas as curvas apresentaram comportamento semelhante ao simulado, como pode ser observado nas figuras que seguem. A ligação do FPGA com o conversor D/A encontra-se na Figura 23.

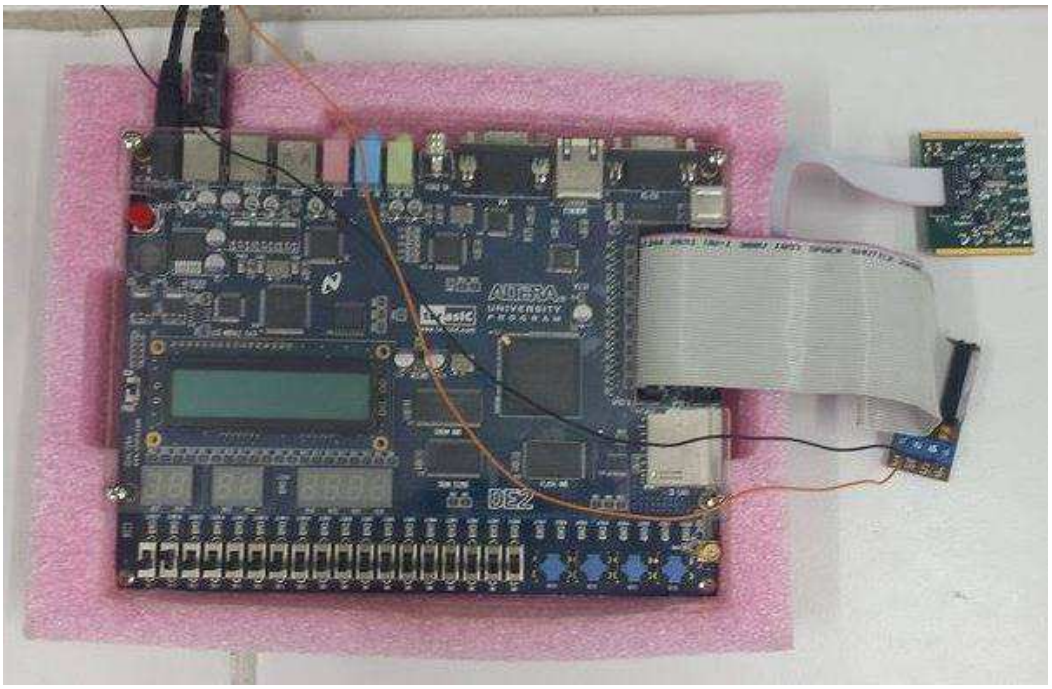

FIGURA 23 FPGA CONECTADO AO CONVERSOR D/A.

5.1 CONVERSOR BOOST

As curvas de corrente, tensão e corrente e tensão em um mesmo gráfico, referentes à simulação e a montagem experimental do conversor Boost podem ser observadas como segue:

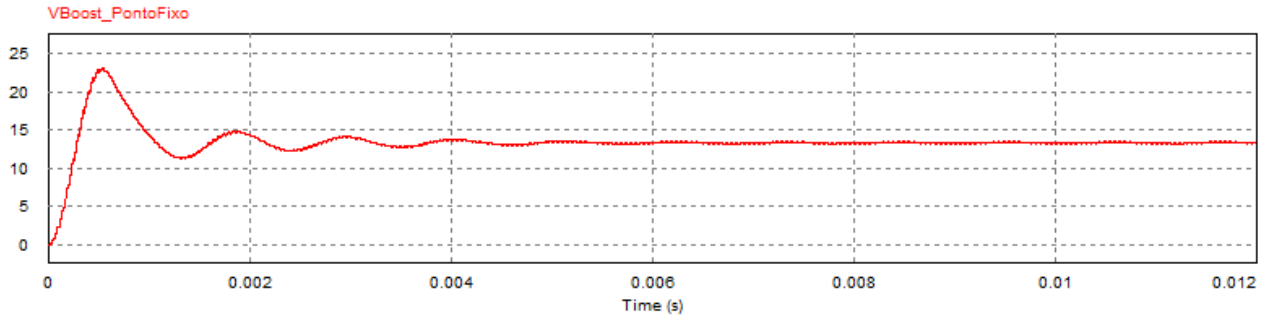

FIGURA 24 BOOST: TENSÃO PONTO FIXO- SIMULAÇÃO PSIM

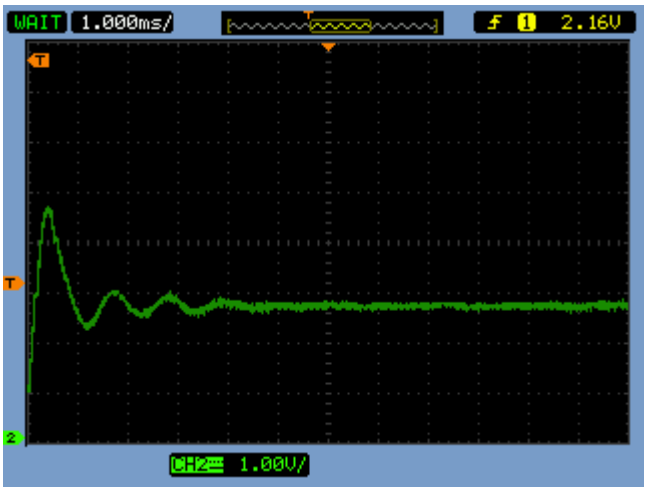

FIGURA 25 BOOST: TENSÃO EXPERIMENTAL

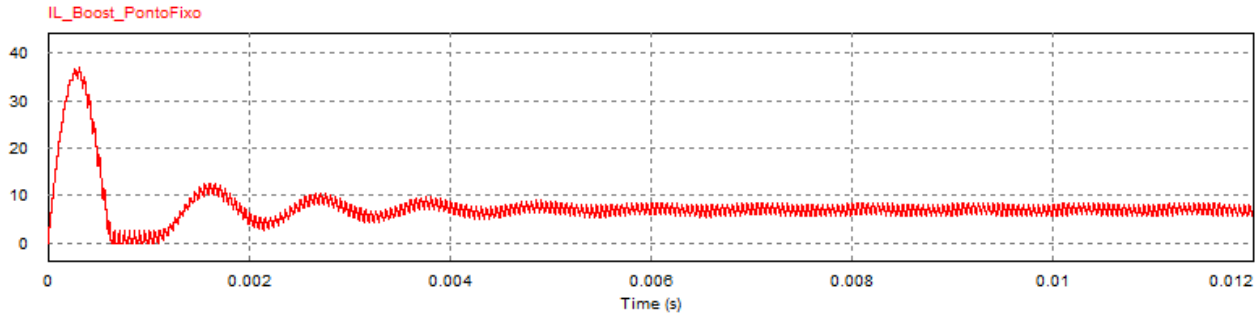

FIGURA 26 BOOST: CORRENTE PONTO FIXO- SIMULAÇÃO PSIM

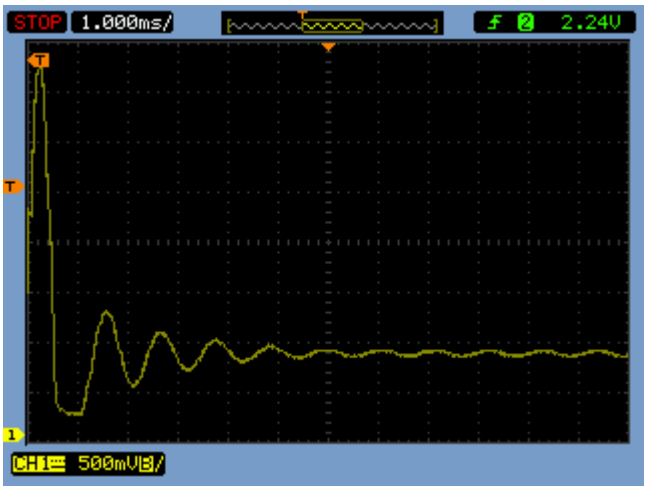

FIGURA 27 BOOST: CORRENTE EXPERIMENTAL

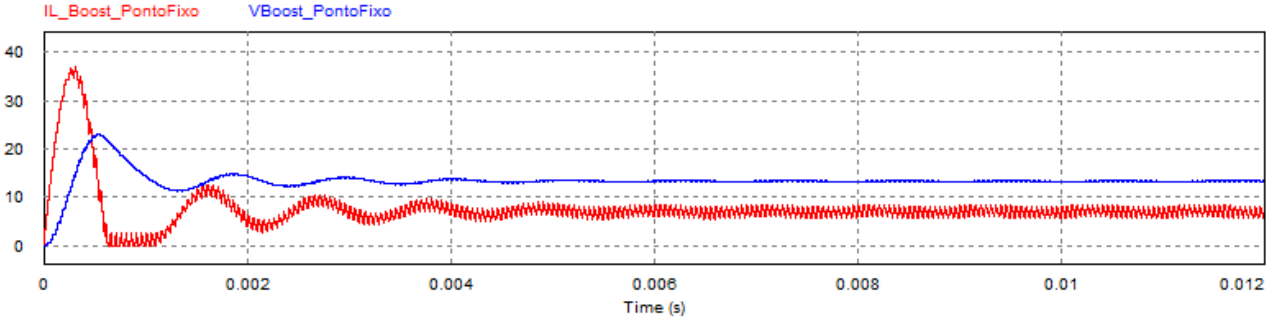

FIGURA 28 BOOST: TENSÃO E CORRENTE PONTO FIXO- SIMULAÇÃO PSIM

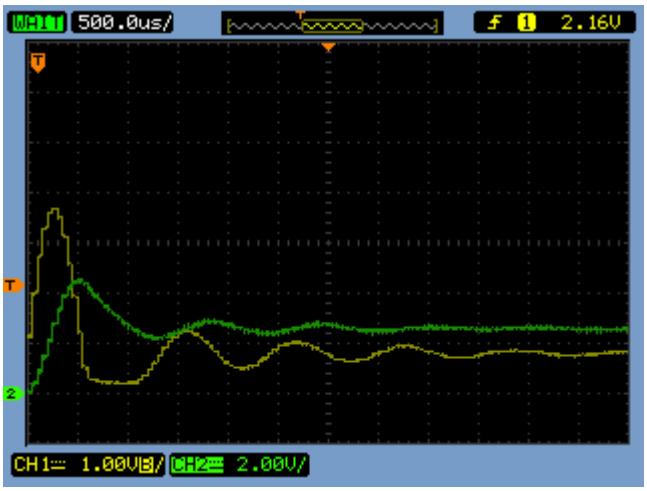

FIGURA 29 BOOST: TENSÃO E CORRENTE EXPERIMENTAL

5.2 CONVERSOR BUCK

As curvas de corrente, tensão e corrente e tensão em um mesmo gráfico, referentes à simulação e a montagem experimental do conversor Buck podem ser observadas como segue:

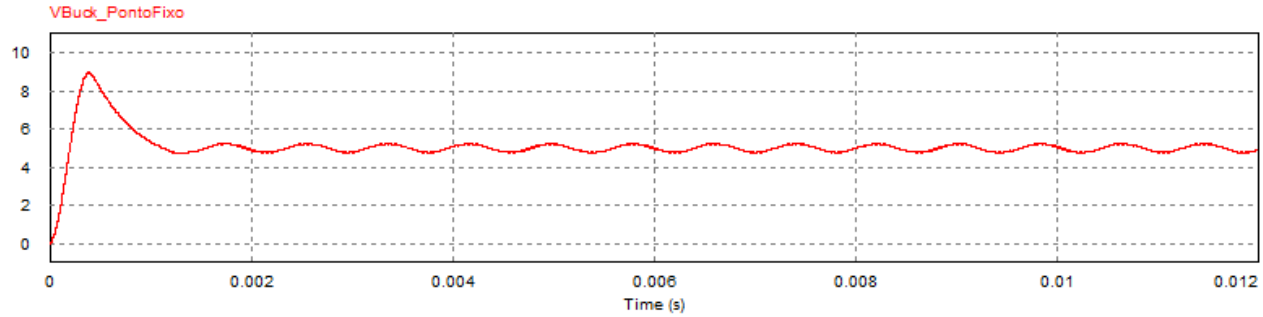

FIGURA 30 BUCK: TENSÃO PONTO FIXO- SIMULAÇÃO PSIM

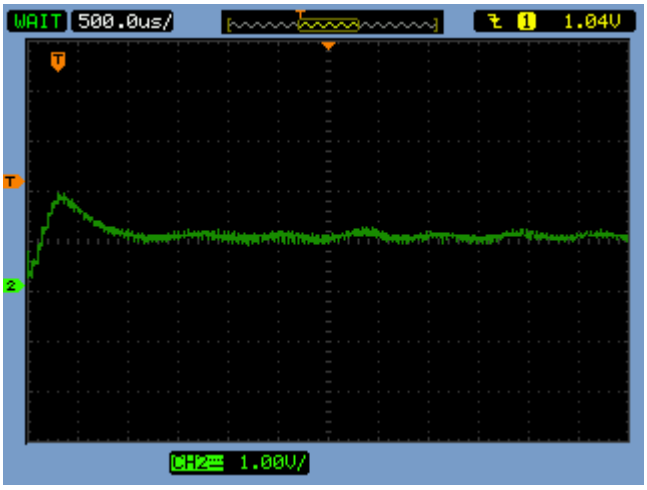

FIGURA 31 BUCK: TENSÃO EXPERIMENTAL

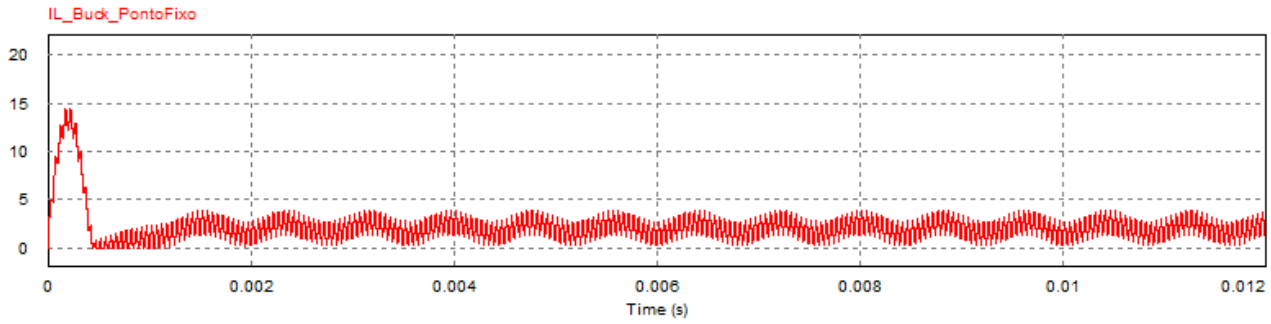

FIGURA 32 BUCK: CORRENTE PONTO FIXO- SIMULAÇÃO PSIM

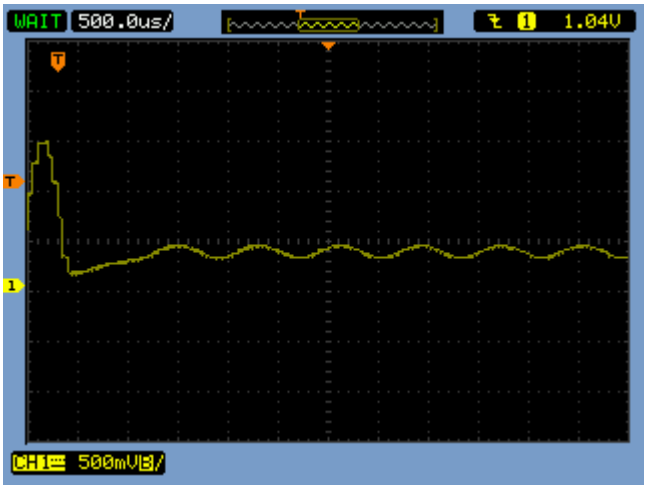

FIGURA 33 BUCK: CORRENTE EXPERIMENTAL

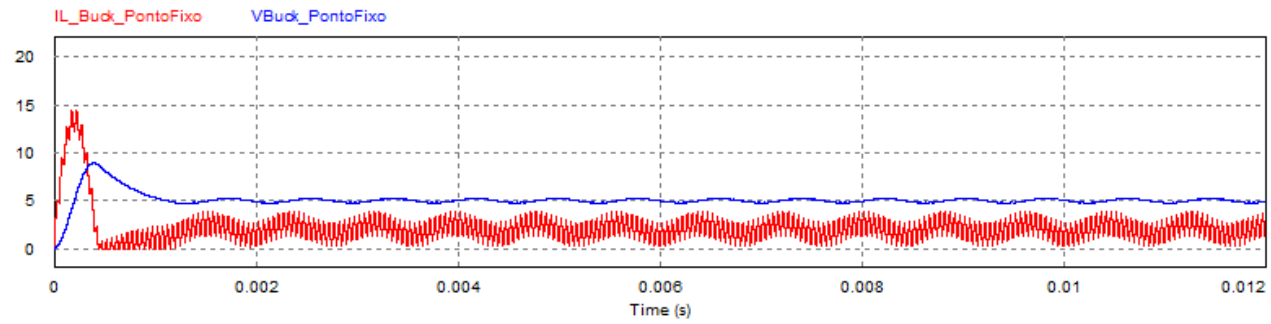

FIGURA 34 BUCK: TENSÃO E CORRENTE PONTO FIXO- SIMULAÇÃO PSIM

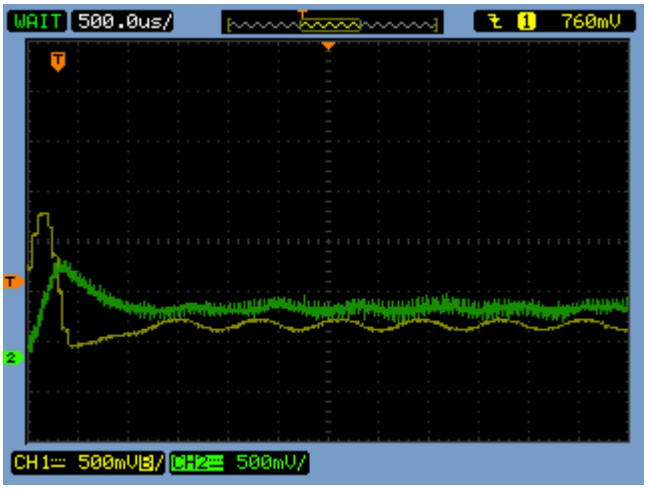

FIGURA 35 BUCK: TENSÃO E CORRENTE EXPERIMENTAL

5.3 CONVERSOR BUCK-BOOST

As curvas de corrente, tensão e corrente e tensão em um mesmo gráfico, referentes à simulação e a montagem experimental do conversor Buck-Boost podem ser observadas como segue:

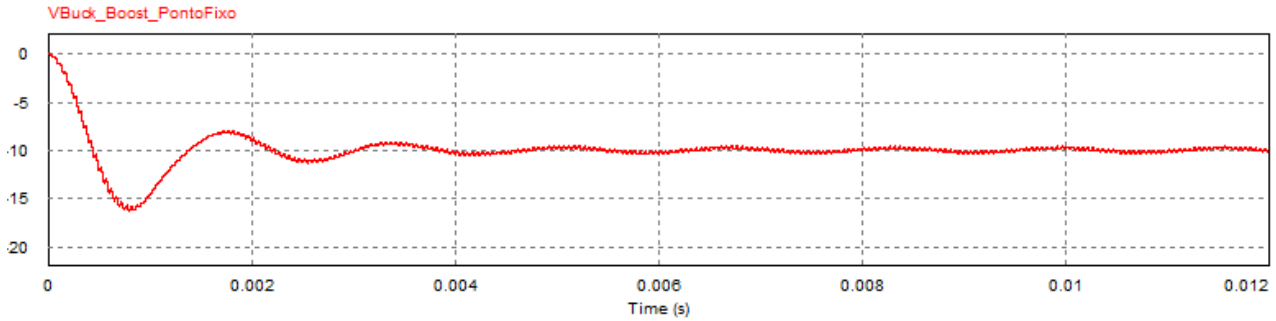

FIGURA 36 BUCK-BOOST: TENSÃO PONTO FIXO- SIMULAÇÃO PSIM

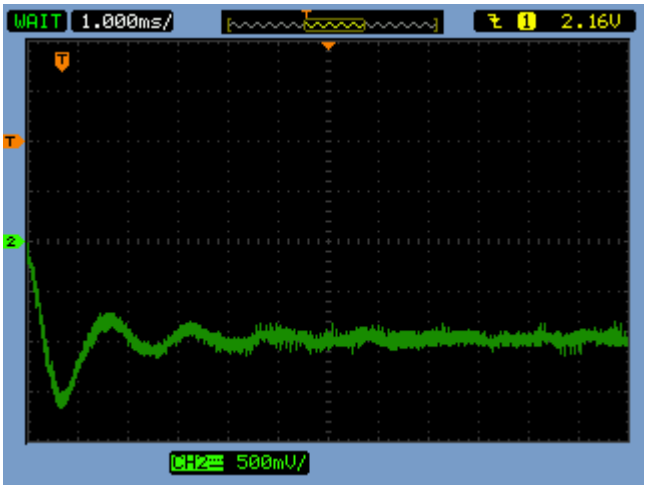

FIGURA 37 BUCK-BOOST: TENSÃO EXPERIMENTAL

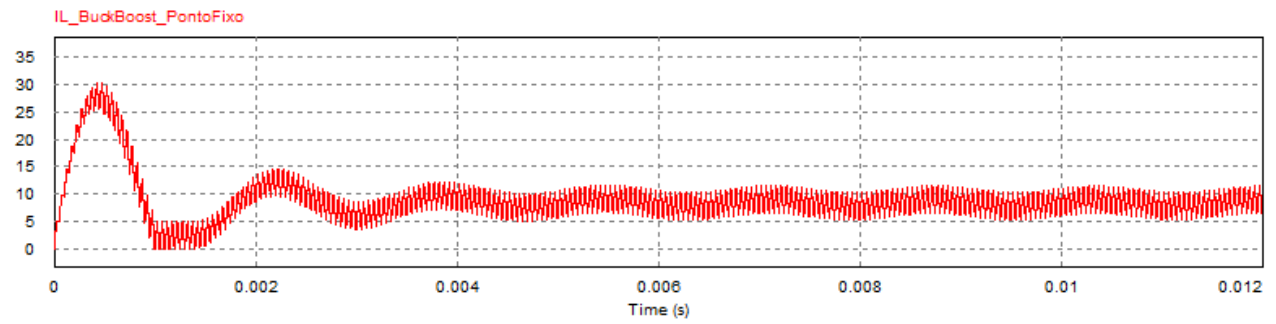

FIGURA 38 BUCK- BOOST: CORRENTE PONTO FIXO- SIMULAÇÃO PSIM

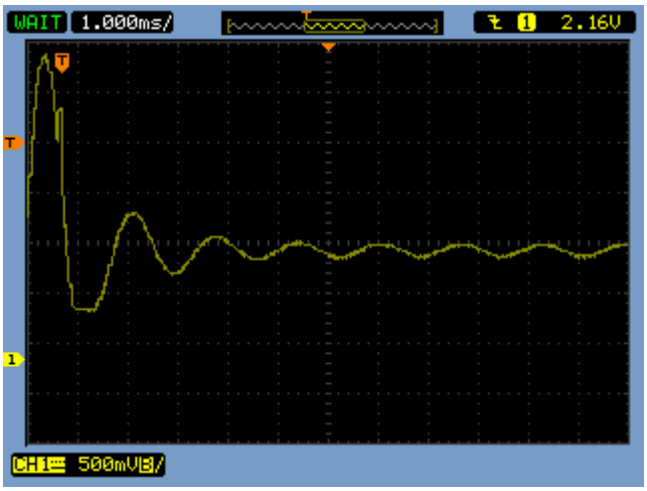

FIGURA 39 BUCK-BOOST: CORRENTE EXPERIMENTAL

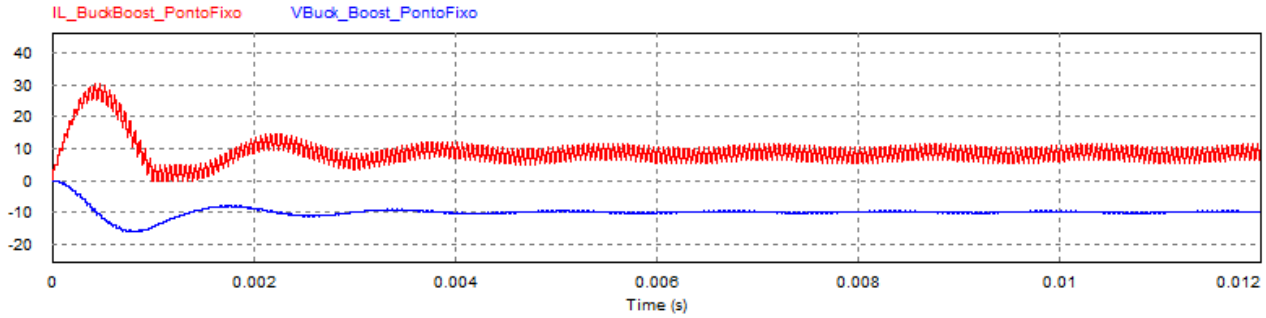

FIGURA 40 BUCK-BOOST: TENSÃO E CORRENTE PONTO FIXO- SIMULAÇÃO PSIM

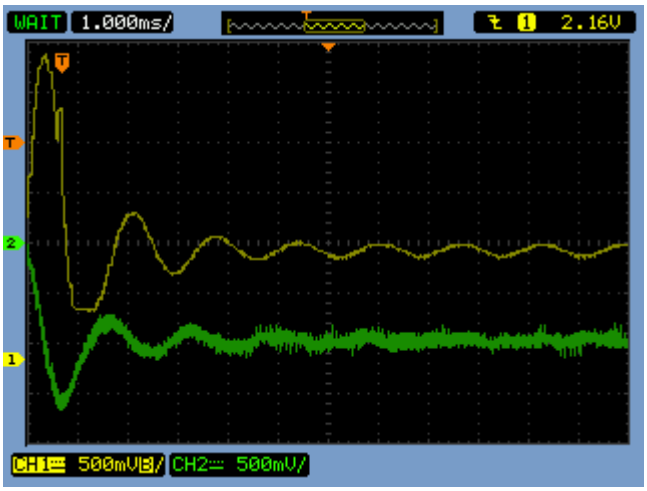

FIGURA 41 BUCK-BOOST: TENSÃO E CORRENTE EXPERIMENTAIS

6 CONCLUSÃO

A principal contribuição desse trabalho foi a simulação de conversores CC-CC em tempo real utilizando FPGA. Essa técnica é especialmente importante na simulação de conversores, pois muitas vezes o projeto de um conversor CC-CC torna-se bastante complexo. Essa complexidade dá-se principalmente pelo uso de indutores, os quais são difíceis de encontrar no mercado e muitas vezes a fabricação "caseira" não é viável.

Utilizando FPGA para simulação de conversores CC – CC, a análise das mudanças ocasionadas pela troca de qualquer dos componentes do circuito torna-se muito mais fácil, uma vez que ela não é mais realizada com a troca do componente físico.

A simulação de sistemas complexos em hardwares como a FPGA tem se tornado uma abordagem cada vez mais utilizada, pois além de manter um alto grau de precisão e fidelidade do sistema real, ainda gera uma diminuição de custos e tempo de desenvolvimento, permitindo a reprodução de sistemas complexos e muitas vezes de difícil montagem.

BIBLIOGRAFIA

Brumati, M. (2005). *Eletrônica de Potência.* Serra - ES.

Erickson, R. W., & Maksimovic, D. (2001). *Fundamentals of Power Eletronics.* Bolder, Colorado: Kluwer Academic Publishers.

França, B. W. (2009). *Hardware in the loop para desenvolvimento de software embarcado em DSPs utilizando ambiente PSCAD/EMTDC.* Rio de Janeiro.

Nascimento, E. J. (s.d.). Análise no domínio do tempo de sistemas em tempo discreto. Univasf.

Petry, C. A. (2014). Capítulo 12: Conversores CC-CC: Conversor Buck.

Ribeiro, R. V. (2011). *Sistema Fotovoltaico Autônomo Baseado em Conversores CC-CC.* Rio de Janeiro.

ANEXO A – PSIM: DLL DOS CONVERSORES

- DLL conversor Boost

```
//DLL para ser utilizada como defasador de 1/4 de período
//Valor de entrada
//In (0) = Sinal de entrada
//Valor de saida
//Out (0) = Sinal de saída

#include <math.h>
#include <stdio.h>

__declspec(dllexport) void simuser (t, delt, in, out)
// Note that all the variables must be defined as "double"
double t, delt;
double *in, *out;
{
double static vetor[6]={0,0,0,0,0,0};
double Tpwm=1*delt;
int ind=0;
double static Tch=0.0;
double static Tch1=0.0;
static double h=0;
static double vin=0;
static double indut=0;
static double indutf=0;
static double cap=0;
static double capf=0;
static double res=0;
static int gate=0;
static double il[3]={0,0,0};
static double vl[3]={0,0,0};
static double ic[3]={0,0,0};
static double vc[3]={0,0,0};
static double ir[3]={0,0,0};
static double vbase = 100;
static double ibase = 50;
```

```
static signed long int k1 = 0;
static signed long int k2 = 0;
static signed long int k3 = 0;
static signed long int k4 = 0;
static signed long int k5 = 0;
static signed long int k6 = 0;
static signed long int ilf[3]={0,0,0};
static signed long int vlf[3]={0,0,0};
static signed long int icf[3]={0,0,0};
static signed long int vcf[3]={0,0,0};
staticsigned long int irf[3]={0,0,0};
static signed long int vinf=0;
gate=in[0];
indut=in[1];
cap=in[2];
vin=in[3];
res=in[4];

//////////////////////////////////////////////////////////////////
// Rotina em ponto flutuante
if (t> Tch)
{
     h = Tpwm;
     Tch = Tch + Tpwm;
if(gate)
{
          il[1]=il[2];
//        il[2]=h*vin/indut+il[1];
          il[2]=vin*(h/indut)+il[1];
          if(il[2]<0.0) il[2]=0.0;
          vc[0]=vc[1];
          vc[1]=vc[2];
//        vc[2]=h*(-1*vc[1]/res-vc[0]/res)/(2*cap)+vc[1];
          vc[2]=-(vc[1]+vc[0])*(h/(res*2*cap))+vc[1];
          if(vc[2]<0.0) vc[2]=0.0;
}
else
{
          il[1]=il[2];
//        il[2]=h*vin/indut-h*(vc[1]+vc[0])/(2*indut)+il[1];
```

```
            il[2]=vin*(h/indut)-(vc[1]+vc[0])*(h/(2*indut))+il[1];
            if(il[2]<0.0) il[2]=0.0;
            vc[0]=vc[1];
            vc[1]=vc[2];
//          vc[2]=h*(il[2] + il[1] - vc[1]/res - vc[0]/res)/(2*cap)+vc[1];
            vc[2]=(il[2] + il[1])*(h/(2*cap)) - (vc[1] + vc[0])*(h/(res*2*cap)) + vc[1];
            if(vc[2]<0.0) vc[2]=0.0;
}
}
// Rotina em ponto fixo
if (t> Tch1)
{
     h = Tpwm;
     Tch1 = Tch1 + Tpwm;
indutf = indut/(vbase/ibase);
capf = cap*(vbase/ibase);
k1 = 1048575*(double)(h/indutf);
k2 = 1048575*(double)(h/(2*capf));
k3 = 1048575*(double)(h/(2*indutf));
k4= 1048575*(double)(h/(2*capf));
k6 = 1048575*(double)(1/(res/(vbase/ibase)));
k2 = ((signed long long int)(k2*k6))>>20;
k5 = -33554431;
vinf = 1048575*(double)(vin/vbase);
if(gate)
{
            ilf[1]=ilf[2];
            ilf[2]=(((signed long long int)(k1*vinf))>>20) + ilf[1];
            if(ilf[2]<0) ilf[2]=0;
            vcf[0]=vcf[1];
            vcf[1]=vcf[2];
            vcf[2]=(((signed long long int)(-k2*(vcf[1] + vcf[0])))>>20)+vcf[1];
            if(vcf[2]<0) vcf[2]=0;
}
else
{
            ilf[1]=ilf[2];
            ilf[2]=(((signed long long int)(k1*vinf))>>20) - (((signed long long int)
            (k3*(vcf[1]+vcf[0])))>>20) + ilf[1];
            if(ilf[2]<0) ilf[2]=0;
```

```
            vcf[0]=vcf[1];
            vcf[1]=vcf[2];
            vcf[2]=(((signed long long int)(k4*(ilf[2] + ilf[1])))>>20) - (((signed long long int)
            (k2*(vcf[1] + vcf[0])))>>20) + vcf[1];
            if(vcf[2]<0) vcf[2]=0;
}
}
out[0]=vc[2];
out[1]=il[2];
out[2]=vc[2]/res;
out[3]=il[2]-vc[2]/res;
out[4]=(vcf[2]*vbase)/1048575;
out[5]=(ilf[2]*ibase)/1048575;
}
```

- DLL conversor Buck

```
//DLL para ser utilizada como defasador de 1/4 de período
//Valor de entrada
//In (0) = Sinal de entrada
//Valor de saida
//Out (0) = Sinal de saída

#include <math.h>
#include <stdio.h>

__declspec(dllexport) void simuser (t, delt, in, out)

// Note that all the variables must be defined as "double"
double t, delt;
double *in, *out;
{
double static vetor[6]={0,0,0,0,0,0};
double Tpwm=1*delt;
int ind=0;
double static Tch=0.0;
double static Tch1=0.0;
static double h=0;
static double vin=0;
static double indut=0;
```

```
static double indutf=0;
static double cap=0;
static double capf=0;
static double res=0;
static int gate=0;
static double il[3]={0,0,0};
static double vl[3]={0,0,0};
static double ic[3]={0,0,0};
static double vc[3]={0,0,0};
static double ir[3]={0,0,0};
static double vbase = 100;
static double ibase = 50;
static signed long int k1 = 0;
static signed long int k2 = 0;
static signed long int k3 = 0;
static signed long int k4 = 0;
static signed long int k5 = 0;
static signed long int k6 = 0;
static signed long int ilf[3]={0,0,0};
static signed long int vlf[3]={0,0,0};
static signed long int icf[3]={0,0,0};
static signed long int vcf[3]={0,0,0};
static signed long int irf[3]={0,0,0};
static signed long int vinf=0;
gate=in[0];
indut=in[1];
cap=in[2];
vin=in[3];
res=in[4];

//////////////////////////////////////////////////////////////////////
// Rotina em ponto flutuante
if (t> Tch)
{
     h = Tpwm;
     Tch = Tch + Tpwm;
if(gate)
{
          il[1]=il[2];
//        il[2]=h*vin/indut-h*(vc[1]+vc[0])/(2*indut)+il[1];
```

```
            il[2]=vin*(h/indut)-(vc[1]+vc[0])*(h/(2*indut))+il[1];
            if(il[2]<0.0) il[2]=0.0;
            vc[0]=vc[1];
            vc[1]=vc[2];
//          vc[2]=h*(il[2] + il[1] - vc[1]/res - vc[0]/res)/(2*cap)+vc[1];
            vc[2]=(il[2] + il[1])*(h/(2*cap)) - (vc[1] + vc[0])*(h/(res*2*cap)) + vc[1];
            if(vc[2]<0.0) vc[2]=0.0;
}
else
{
            il[1]=il[2];
//          il[2]=h*vin/indut-h*(vc[1]+vc[0])/(2*indut)+il[1];
            il[2]=-(vc[1]+vc[0])*(h/(2*indut))+il[1];
            if(il[2]<0.0) il[2]=0.0;
            vc[0]=vc[1];
            vc[1]=vc[2];
//          vc[2]=h*(il[2] + il[1] - vc[1]/res - vc[0]/res)/(2*cap)+vc[1];
            vc[2]=(il[2] + il[1])*(h/(2*cap)) - (vc[1] + vc[0])*(h/(res*2*cap)) + vc[1];
            if(vc[2]<0.0) vc[2]=0.0;
}
}
// Rotina em ponto fixo
if (t> Tch1)
{
     h = Tpwm;
     Tch1 = Tch1 + Tpwm;
indutf = indut/(vbase/ibase);
capf = cap*(vbase/ibase);
k1 = 1048575*(double)(h/indutf);
k2 = 1048575*(double)(h/(2*capf));
k3 = 1048575*(double)(h/(2*indutf));
k4= 1048575*(double)(h/(2*capf));
k6 = 1048575*(double)(1/(res/(vbase/ibase)));
k2 = ((signed long long int)(k2*k6))>>20;
k5 = -33554431;
vinf = 1048575*(double)(vin/vbase);
if(gate)
{
            ilf[1] = ilf[2];
            ilf[2] = (((signed long long int)(k1*vinf))>>20) - (((signed long long int)
```

```
            (k3*(vcf[1]+vcf[0])))>>20) + ilf[1];
            if(ilf[2]<0) ilf[2]=0;
            vcf[0] = vcf[1];
            vcf[1] = vcf[2];
            vcf[2] = (((signed long long int)(k4*(ilf[2] + ilf[1])))>>20) - (((signed long long int)
            (k2*(vcf[1] + vcf[0])))>>20) + vcf[1];
            if(vcf[2]<0) vcf[2]=0;
}
else
{
            ilf[1] = ilf[2];
            ilf[2] = -(((signed long long int)(k3*(vcf[1]+vcf[0])))>>20) + ilf[1];
            if(ilf[2]<0) ilf[2]=0;
            vcf[0] = vcf[1];
            vcf[1] = vcf[2];
            vcf[2] = (((signed long long int)(k4*(ilf[2] + ilf[1])))>>20) - (((signed long long int)
            (k2*(vcf[1] + vcf[0])))>>20) + vcf[1];
            if(vcf[2]<0) vcf[2]=0;
}
}
out[0]=vc[2];
out[1]=il[2];
out[2]=vc[2]/res;
out[3]=il[2]-vc[2]/res;
out[4]=(vcf[2]*vbase)/1048575;
out[5]=(ilf[2]*ibase)/1048575;
}
```

- DLL conversor Buck-Boost

```
//DLL para ser utilizada como defasador de 1/4 de período
//Valor de entrada
//In (0) = Sinal de entrada
//Valor de saida
//Out (0) = Sinal de saída

#include <math.h>
#include <stdio.h>

__declspec(dllexport) void simuser (t, delt, in, out)
```

```
// Note that all the variables must be defined as "double"
double t, delt;
double *in, *out;
{
double static vetor[6]={0,0,0,0,0,0};
double Tpwm=1*delt;
int ind=0;
double static Tch=0.0;
double static Tch1=0.0;
static double h=0;
static double vin=0;
static double indut=0;
static double indutf=0;
static double cap=0;
static double capf=0;
static double res=0;
static int gate=0;
static double il[3]={0,0,0};
static double vl[3]={0,0,0};
static double ic[3]={0,0,0};
static double vc[3]={0,0,0};
static double ir[3]={0,0,0};
static double vbase = 100;
static double ibase = 50;
static signed long int k1 = 0;
static signed long int k2 = 0;
static signed long int k3 = 0;
static signed long int k4 = 0;
static signed long int k5 = 0;
static signed long int k6 = 0;
static signed long int ilf[3]={0,0,0};
static signed long int vlf[3]={0,0,0};
static signed long int icf[3]={0,0,0};
static signed long int vcf[3]={0,0,0};
staticsigned long int irf[3]={0,0,0};
static signed long int vinf=0;
static int formato = 20;
static long int pot2= 0;
gate=in[0];
```

```
indut=in[1];
cap=in[2];
vin=in[3];
res=in[4];

//////////////////////////////////////////////////////////////////
// Rotina em ponto flutuante
if (t> Tch)
{
      h = Tpwm;
      Tch = Tch + Tpwm;
if(gate)
{
             il[1]=il[2];
//           il[2]=h*vin/indut+il[1];
             il[2]=vin*(h/indut)+il[1];
             if(il[2]<0.0) il[2]=0.0;
             vc[0]=vc[1];
             vc[1]=vc[2];
//           vc[2]=h*(-1*vc[1]/res-vc[0]/res)/(2*cap)+vc[1];
             vc[2]= -(vc[1]+vc[0])*(h/(res*2*cap))+vc[1];
             if(vc[2]>0.0) vc[2]=0.0;
}
else
{
             il[1]=il[2];
//           il[2]=h*vin/indut-h*(vc[1]+vc[0])/(2*indut)+il[1];
             il[2]=(vc[1]+vc[0])*(h/(2*indut))+il[1];
             if(il[2]<0.0) il[2]=0.0;
             vc[0]=vc[1];
             vc[1]=vc[2];
//           vc[2]=h*(il[2] + il[1] - vc[1]/res - vc[0]/res)/(2*cap)+vc[1];
             vc[2]=-(il[2] + il[1])*(h/(2*cap)) - (vc[1] + vc[0])*(h/(res*2*cap)) + vc[1];
             if(vc[2]>0.0) vc[2]=0.0;
}
}
// Rotina em ponto fixo
if (t> Tch1)
{
      h = Tpwm;
```

```
      Tch1 = Tch1 + Tpwm;
indutf = indut/(vbase/ibase);
capf = cap*(vbase/ibase);
pot2 = pow(2,formato)-1;
k1 = pot2*(double)(h/indutf);
k2 = pot2*(double)(h/(2*capf));
k3 = pot2*(double)(h/(2*indutf));
k4 = pot2*(double)(h/(2*capf));
k6 = pot2*(double)(1/(res/(vbase/ibase)));
k2 = ((signed long long int)(k2*k6))>>formato;
vinf = pot2*(double)(vin/vbase);
if(gate)
{
            ilf[1]=ilf[2];
//          ilf[2]=(((signed long long int)(k1*vinf))>>18) + ilf[1];
            ilf[2]=(((signed long int)(k1*vinf))>>formato) + ilf[1];
            if(ilf[2]<0) ilf[2]=0;
            vcf[0]=vcf[1];
            vcf[1]=vcf[2];
//          vcf[2]= -(((signed long long int)(k2*(vcf[1] + vcf[0])))>>20) + vcf[1];
            vcf[2]= -(((signed long int)(k2*(vcf[1] + vcf[0])))>>formato) + vcf[1];
            if(vcf[2]>0) vcf[2]=0;

}
else
{
            ilf[1] = ilf[2];
//          ilf[2] = (((signed long long int)(k3*(vcf[1]+vcf[0])))>>20) + ilf[1];
            ilf[2] = (((signed long int)(k3*(vcf[1]+vcf[0])))>>formato) + ilf[1];
            if(ilf[2]<0) ilf[2]=0;
            vcf[0] = vcf[1];
            vcf[1] = vcf[2];
//          vcf[2] = -(((signed long long int)(k4*(ilf[2] + ilf[1])))>>20) - (((signed long long int)
            (k2*(vcf[1] + vcf[0])))>>20) + vcf[1];
            vcf[2] = -(((signed long int)(k4*(ilf[2] + ilf[1])))>>formato) - (((signed long long int)
            (k2*(vcf[1] + vcf[0])))>>formato) + vcf[1];
            if(vcf[2]>0) vcf[2]=0;
}
}
out[0]=vc[2];
```

```
out[1]=il[2];
out[2]=vc[2]/res;
out[3]=il[2]-vc[2]/res;
out[4]=(vcf[2]*vbase)/pot2;
out[5]=(ilf[2]*ibase)/pot2;
}
```

ANEXO B– FPGA: CONVERSORES EM VERILOG

- Módulo Mod_Test

```
module Mod_Teste(
                    ////////////////////    Clock Input              ////////////////////
                    CLOCK_27,                                        //27 MHz
                    CLOCK_50,                                        //50 MHz

                    ////////////////////    Push Button              ////////////////////
                    KEY,                                             //Pushbutton[3:0]

                    ////////////////////    DPDT Switch              ////////////////////
                    SW,                                              //Toggle Switch[17:0]

                    ////////////////////    7-SEG Dispaly    ////////////////////
                    HEX0,                                            //Seven Segment Digit 0
                    HEX1,                                            //Seven Segment Digit 1
                    HEX2,                                            //Seven Segment Digit 2
                    HEX3,                                            //Seven Segment Digit 3
                    HEX4,                                            //Seven Segment Digit 4
                    HEX5,                                            //Seven Segment Digit 5
                    HEX6,                                            //Seven Segment Digit 6
                    HEX7,                                            //Seven Segment Digit 7

                    ////////////////////////    LED              ////////////////////////
                    LEDG,                                            //LED Green[8:0]
                    LEDR,                                            //LED Red[17:0]

                    ////////////////////////    UART    ////////////////////////
                    UART_TXD,                                        //UART Transmitter
                    UART_RXD,                                        //UART Receiver

                    ////////////////////    LCD Module 16X2          ////////////////
                    LCD_ON,                                          //LCD Power ON/OFF
                    LCD_BLON,                                        //LCD Back Light ON/OFF
                    LCD_RW,                  //LCD Read/Write Select, 0 = Write, 1 = Read
                    LCD_EN,                                          //LCD Enable
```

```
                LCD_RS,             //LCD Command/Data Select, 0 = Command, 1 = Data
                LCD_DATA,                                   //LCD Data bus 8 bits

                ////////////////////    GPIO    ////////////////////////////
                GPIO_0,                                     //GPIO Connection 0
                GPIO_1                                      //GPIO Connection 1
        );

////////////////////////  Clock Input             ////////////////////////
input                   CLOCK_27;                           //27 MHz
input                   CLOCK_50;                           //50 MHz

////////////////////////  Push Button             ////////////////////////
input   [3:0]   KEY;                                //Pushbutton[3:0]

////////////////////////  DPDT Switch             ////////////////////////
input   [17:0]  SW;                                 //Toggle Switch[17:0]

////////////////////////  7-SEG Dispaly   ////////////////////////
output  [6:0]   HEX0;                               //Seven Segment Digit 0
output  [6:0]   HEX1;                               //Seven Segment Digit 1
output  [6:0]   HEX2;                               //Seven Segment Digit 2
output  [6:0]   HEX3;                               //Seven Segment Digit 3
output  [6:0]   HEX4;                               //Seven Segment Digit 4
output  [6:0]   HEX5;                               //Seven Segment Digit 5
output  [6:0]   HEX6;                               //Seven Segment Digit 6
output  [6:0]   HEX7;                               //Seven Segment Digit 7

////////////////////////////        LED             ////////////////////////////
output  [8:0]   LEDG;                               //LED Green[8:0]
output  [17:0]  LEDR;                               //LED Red[17:0]

////////////////////////////        UART    ////////////////////////////
output                  UART_TXD;                   //UART Transmitter
input                   UART_RXD;                   //UART Receiver

////////////////////    LCD Module 16X2         ////////////////////////////
inout   [7:0]           LCD_DATA;                   //LCD Data bus 8 bits
output                  LCD_ON;                     //LCD Power ON/OFF
output                  LCD_BLON;                   //LCD Back Light ON/OFF
```

```
output          LCD_RW;     //LCD Read/Write Select, 0 = Write, 1 = Read
output          LCD_EN;                 //LCD Enable
output          LCD_RS;     //LCD Command/Data Select, 0 = Command, 1 = Data

//////////////////////  GPIO    ////////////////////////////////
inout   [35:0]  GPIO_0;                 //GPIO Connection 0
inout   [35:0]  GPIO_1;                 //GPIO Connection 1

        //
        reg [7:0] cont;
        reg gate;
        reg gate1;
        reg [15:0] compara;
        reg [15:0] triangular;
        //wire gate;

        //assign gate = GPIO_0[0];

        //          vbase = 100;
        //          ibase = 50;

        //          indutor = 50uH
        //          capacitor = 330uF
        //          resistor = 2.5 ohms
        //          Vin = 10V

        //  indutf = indut/(vbase/ibase);
        //  capf = cap*(vbase/ibase);

        //  k1 = 1048575*(double)(h/indutf);        // k1 = 4194
        //  k2 = 1048575*(double)(h/(2*capf));      // k2 = 79
        //  k3 = 1048575*(double)(h/(2*indutf));    // k3 = 2097
        //  k4 = 1048575*(double)(h/(2*capf));      // k4 = 79

        //  k6 = 1048575*(double)(1/(res/(vbase/ibase)));   // k6 = 838860
        //  k2 = ((signed long long int)(k2*k6))>>20;       // k2 = 63


        //  vinf = 1048575*(double)(vin/vbase);         // vinf = 104857
```

```
reg sinal;
reg [25:0] cont1;
reg signed [63:0] k1;
reg signed [63:0] k2;
reg signed [63:0] k3;
reg signed [63:0] k4;

reg signed [63:0] k5;
reg signed [63:0] k6;
reg signed [63:0] k7;
reg signed [63:0] k8;
reg signed [63:0] k9;
reg signed [63:0] k10;
reg signed [63:0] k11;
reg signed [63:0] k12;

reg signed [63:0] vinf;

reg signed [63:0] intermed1;
reg signed [63:0] intermed2;
reg signed [63:0] intermed3;
reg signed [63:0] intermed4;

reg signed [63:0] ilf [2:0];
reg signed [63:0] vcf [2:0];

always @ (posedge fio_c0)
begin
        compara = 2500;
        triangular = triangular + 1;
        if(triangular>9999) triangular = 0;

        if(compara>triangular) gate=1;
        else gate =0;

end
always @ (posedge fio_c0)
begin

        k1 <= 4194;
```

```
k2 <= 63;
k3 <= 2097;
k4 <= 79;
vinf <= 104857;

k6 = -k2;
k5 = -k2;
k5 = k5*k4;
k5 = k5 + 32767;


cont = cont + 1;
if(cont>19)
begin
        gate1 = gate1 ^1;
        case(SW[17:16])
                2'b00 :begin        //conversor boost
                        if(gate)
                        begin
                        ilf[1]=ilf[2];
                        intermed1 = k1*vinf;
//                      ilf[2]=(((k1*vinf))>>>20) + ilf[1];
                        ilf[2]=(intermed1>>>20) + ilf[1];
                        if(ilf[2]<0) ilf[2]=0;

                        k7 = intermed1>>>20;
                        k8 = ilf[2]>>>4;
//                      k7 = k7+32767;

                        vcf[0]=vcf[1];
                        vcf[1]=vcf[2];
                        intermed1 = vcf[1] + vcf[0];
                        intermed2 = k6*intermed1;
//                      vcf[2]=(((k6*(vcf[1] + vcf[0])))>>>20) + vcf[1];
                        vcf[2]=(intermed2>>>20) + vcf[1];
                        if(vcf[2]<0) vcf[2]=0;

                        k9 = vcf[2];

                        end
```

```
                    else
                      begin
                      ilf[1]=ilf[2];
                      intermed1 = k1*vinf;
                      intermed2 = vcf[1]+vcf[0];
                      intermed3 = k3*intermed2;
//                    ilf[2]=(((k1*vinf))>>>20) - (((k3*(vcf[1]+vcf[0])))>>>20) + ilf[1];
                      ilf[2]=(intermed1>>>20) - (intermed3>>>20) + ilf[1];
                      if(ilf[2]<0) ilf[2]=0;

//                    k7 = intermed1>>>20;

                      vcf[0]=vcf[1];
                      vcf[1]=vcf[2];
                      intermed1 = ilf[2] + ilf[1];
                      intermed2 = k4*intermed1;
                      intermed3 = vcf[1] + vcf[0];
                      intermed4 = k2*intermed3;
//                    vcf[2]=(((k4*(ilf[2] + ilf[1])))>>>20) - (((k2*(vcf[1] +
                    vcf[0])))>>>20) + vcf[1];
                      vcf[2]=(intermed2>>>20) - (intermed4>>>20) + vcf[1];
                      if(vcf[2]<0) vcf[2]=0;
                      end
            end

                    2'b01 :   begin     //conversor buck
                            if(gate)
                              begin
                              ilf[1] = ilf[2];
                              ilf[2] = (((k1*vinf))>>>20) –
                              (((k3*(vcf[1]+vcf[0])))>>>20) + ilf[1];
                              if(ilf[2]<0) ilf[2]=0;

                              k8 = ilf[2]>>>4;

                              vcf[0] = vcf[1];
                              vcf[1] = vcf[2];
                              vcf[2] = (((k4*(ilf[2] + ilf[1])))>>>20) –
                            (((k2*(vcf[1] + vcf[0])))>>>20) + vcf[1];
                              if(vcf[2]<0) vcf[2]=0;
```

```
            k9 = vcf[2];
            end
        else
            begin
            ilf[1] = ilf[2];
            ilf[2] = -(((k3*(vcf[1]+vcf[0])))>>>20) + ilf[1];
            if(ilf[2]<0) ilf[2]=0;

            vcf[0] = vcf[1];
            vcf[1] = vcf[2];
            vcf[2] = (((k4*(ilf[2] + ilf[1])))>>>20) –
            (((k2*(vcf[1] + vcf[0])))>>>20) + vcf[1];
            if(vcf[2]<0) vcf[2]=0;
            end
end

2'b10 :begin        //conversor buck-boost
            if(gate)
            begin
            ilf[1]=ilf[2];
            ilf[2]=(((k1*vinf))>>>20) + ilf[1];
            if(ilf[2]<0) ilf[2]=0;

            k8 = ilf[2]>>>4;

            vcf[0]=vcf[1];
            vcf[1]=vcf[2];
            vcf[2]= -(((k2*(vcf[1] + vcf[0])))>>>20) + vcf[1];
            if(vcf[2]>0) vcf[2]=0;

            k9 = -vcf[2];
            end
      else
       begin
       ilf[1] = ilf[2];
       ilf[2] = (((k3*(vcf[1]+vcf[0])))>>>20) + ilf[1];
       if(ilf[2]<0) ilf[2]=0;

       vcf[0] = vcf[1];
```

```
                                vcf[1] = vcf[2];
                                vcf[2] = -(((k4*(ilf[2] + ilf[1])))>>>20) - (((k2*(vcf[1] +
                                vcf[0])))>>>20) + vcf[1];
                                if(vcf[2]>0) vcf[2]=0;
                                end
                        end

                            2'b11    :       begin    //reservado
                                             end
                    endcase

                    cont=0;
                    k10 = k9>>>3;
            end
end

PLL PLL1(.areset(SW[0]),
                .inclk0(CLOCK_50),
                .c0(fio_c0),
                .locked(LEDG[8])
                        );

reg [19:0] cont2;
reg [19:0] cont_start;
reg [19:0] cont_ndata;
reg [5:0] cont_busy;

reg signed [20:0] IL_offset;
reg signed [20:0] VC_offset;

reg [23:0] data_spi [7:0];

reg start;
reg start1;
reg new_data;
reg [3:0]index_dado;
//reg index_dado1;
reg busy_state;

always@(posedge fio_c0)
```

```
begin

        IL_offset = 32767 + ilf[2]>>>5;
        VC_offset = 32767 + vcf[2]>>>5;

        data_spi[7] = 24'h270000;
        data_spi[6] = {8'h06,k10[15:0]};
        data_spi[5] = {8'h05,k8[15:0]};
        data_spi[4] = {8'h04,k7[15:0]};
        data_spi[3] = {8'h03,16'd32767};
        data_spi[2] = {8'h02,k5[15:0]};
        data_spi[1] = {8'h01,IL_offset};//2FFF;
        data_spi[0] = {8'h00,VC_offset};//{24'h001FFF;

        cont2 = cont2 + 1;
        if(cont2 >= 10000)
        begin
                cont2 = 0;
                if(!fio_busy)
                begin
                        if(!start) cont_start = 0;
                        if(!new_data) cont_ndata = 0;
                        start = 1;
                        index_dado = 0;
                end
        end

        if(start)
        begin
                cont_start = cont_start + 1;
                if(cont_start == 10)
                begin
                        new_data = 1;
                        start = 0;
                        cont_start = 0;
                end
        end

        if(new_data)
        begin
```

```
                cont_ndata = cont_ndata + 1;
                if(cont_ndata == 10)
                begin
                        new_data = 0;
                        cont_ndata = 0;
                        if(index_dado < 7) busy_state = 1;
                        cont_busy = 0;
                end
        end

        if( (busy_state) && (!fio_busy) )
        begin
                cont_busy = cont_busy + 1;
                if(cont_busy == 10)
                begin
                        busy_state = 0;
                        if(!start) cont_start = 0;
                        if(!new_data) cont_ndata = 0;
                        start = 1;
                        index_dado = index_dado + 1;
                        cont_busy=0;
                end
        end
end


spi spi1(
        .sck_in(fio_c0),
        .miso(fio_miso),
        .dado(data_spi[index_dado]),
        .new_data(new_data),
        .start(start),
        .sck(fio_sck),
        .mosi(fio_mosi),
        .cs(fio_cs),
        .busy(fio_busy));

assign GPIO_0[11] = start;
assign GPIO_0[13] = new_data;
assign GPIO_0[15] = fio_busy;
```

```
assign GPIO_0[17] = fio_mosi;
assign GPIO_0[19] = fio_sck;
assign GPIO_0[21] = fio_cs;

assign GPIO_0[12] = fio_mosi;
assign GPIO_0[16] = fio_sck;
assign GPIO_0[24] = fio_cs;

always@(posedge fio_c0)
begin
        cont1 = cont1+1;
        if(cont1>=25000000)
        begin
                cont1 = 0;
                sinal = sinal ^1;
        end
end

//      Turn on all display
//assign  HEX0              =       7'h00;
//assign  HEX1              =       7'h00;
//assign  HEX2              =       7'h00;
//assign  HEX3              =       7'h00;
//assign  HEX4              =       7'h00;
//assign  HEX5              =       7'h00;
//assign  HEX6              =       7'h00;
//assign  HEX7              =       7'h00;
//assign  LEDG[7:0]       =       9'h1FF;
//assign  LEDR              =       18'h3FFFF;

//assign  GPIO_0[24] = fio_c0;
//assign  GPIO_0[13] = sinal;
assign  LCD_ON          =       1'b1;
assign  LCD_BLON        =       1'b1;
assign LEDG[0] = sinal;
assign LEDR = k5;

//      All inout port turn to tri-state
assign  LCD_DATA        =       8'hzz;
```

```
//assign  GPIO_0          =       36'hzzzzzzzzz;
//assign  GPIO_1          =       36'hzzzzzzzzz;
assign    GPIO_1[0] = gate1;
assign    GPIO_1[1] = gate;

assign LEDG[2:1] = SW[17:16];
endmodule
```

- Módulo PLL

```
// megafunction wizard: %ALTPLL%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altpll

// ============================================================
// File Name: PLL.v
// Megafunction Name(s):
//                         altpll
//
// Simulation Library Files(s):
//                         altera_mf
// ============================================================
// ************************************************************
// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
//
// 11.0 Build 157 04/27/2011 SJ Web Edition
// ************************************************************


//Copyright (C) 1991-2011 Altera Corporation
//Your use of Altera Corporation's design tools, logic functions
//and other software and tools, and its AMPP partner logic
//functions, and any output files from any of the foregoing
//(including device programming or simulation files), and any
//associated documentation or information are expressly subject
//to the terms and conditions of the Altera Program License
//Subscription Agreement, Altera MegaCore Function License
//Agreement, or other applicable license agreement, including,
//without limitation, that your use is for the sole purpose of
//programming logic devices manufactured by Altera and sold by
```

```
//Altera or its authorized distributors.  Please refer to the
//applicable agreement for further details.


// synopsys translate_off
`timescale 1 ps / 1 ps
// synopsys translate_on
module PLL (
	areset,
	inclk0,
	c0,
	locked);

	input	areset;
	input	inclk0;
	output	c0;
	output	locked;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
	tri0	areset;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

	wire  sub_wire0;
	wire [5:0] sub_wire1;
	wire [0:0] sub_wire5 = 1'h0;
	wire  locked = sub_wire0;
	wire [0:0] sub_wire2 = sub_wire1[0:0];
	wire  c0 = sub_wire2;
	wire  sub_wire3 = inclk0;
	wire [1:0] sub_wire4 = {sub_wire5, sub_wire3};

	altpll	altpll_component (
				.areset (areset),
				.inclk (sub_wire4),
				.locked (sub_wire0),
				.clk (sub_wire1),
				.activeclock (),
```

```
                .clkbad (),
                .clkena ({6{1'b1}}),
                .clkloss (),
                .clkswitch (1'b0),
                .configupdate (1'b0),
                .enable0 (),
                .enable1 (),
                .extclk (),
                .extclkena ({4{1'b1}}),
                .fbin (1'b1),
                .fbmimicbidir (),
                .fbout (),
                .fref (),
                .icdrclk (),
                .pfdena (1'b1),
                .phasecounterselect ({4{1'b1}}),
                .phasedone (),
                .phasestep (1'b1),
                .phaseupdown (1'b1),
                .pllena (1'b1),
                .scanaclr (1'b0),
                .scanclk (1'b0),
                .scanclkena (1'b1),
                .scandata (1'b0),
                .scandataout (),
                .scandone (),
                .scanread (1'b0),
                .scanwrite (1'b0),
                .sclkout0 (),
                .sclkout1 (),
                .vcooverrange (),
                .vcounderrange ());
    defparam
            altpll_component.clk0_divide_by = 1,
            altpll_component.clk0_duty_cycle = 50,
            altpll_component.clk0_multiply_by = 4,
            altpll_component.clk0_phase_shift = "0",
            altpll_component.compensate_clock = "CLK0",
            altpll_component.gate_lock_signal = "NO",
            altpll_component.inclk0_input_frequency = 20000,
```

```
altpll_component.intended_device_family = "Cyclone II",
altpll_component.invalid_lock_multiplier = 5,
altpll_component.lpm_hint = "CBX_MODULE_PREFIX=PLL",
altpll_component.lpm_type = "altpll",
altpll_component.operation_mode = "NORMAL",
altpll_component.port_activeclock = "PORT_UNUSED",
altpll_component.port_areset = "PORT_USED",
altpll_component.port_clkbad0 = "PORT_UNUSED",
altpll_component.port_clkbad1 = "PORT_UNUSED",
altpll_component.port_clkloss = "PORT_UNUSED",
altpll_component.port_clkswitch = "PORT_UNUSED",
altpll_component.port_configupdate = "PORT_UNUSED",
altpll_component.port_fbin = "PORT_UNUSED",
altpll_component.port_inclk0 = "PORT_USED",
altpll_component.port_inclk1 = "PORT_UNUSED",
altpll_component.port_locked = "PORT_USED",
altpll_component.port_pfdena = "PORT_UNUSED",
altpll_component.port_phasecounterselect = "PORT_UNUSED",
altpll_component.port_phasedone = "PORT_UNUSED",
altpll_component.port_phasestep = "PORT_UNUSED",
altpll_component.port_phaseupdown = "PORT_UNUSED",
altpll_component.port_pllena = "PORT_UNUSED",
altpll_component.port_scanaclr = "PORT_UNUSED",
altpll_component.port_scanclk = "PORT_UNUSED",
altpll_component.port_scanclkena = "PORT_UNUSED",
altpll_component.port_scandata = "PORT_UNUSED",
altpll_component.port_scandataout = "PORT_UNUSED",
altpll_component.port_scandone = "PORT_UNUSED",
altpll_component.port_scanread = "PORT_UNUSED",
altpll_component.port_scanwrite = "PORT_UNUSED",
altpll_component.port_clk0 = "PORT_USED",
altpll_component.port_clk1 = "PORT_UNUSED",
altpll_component.port_clk2 = "PORT_UNUSED",
altpll_component.port_clk3 = "PORT_UNUSED",
altpll_component.port_clk4 = "PORT_UNUSED",
altpll_component.port_clk5 = "PORT_UNUSED",
altpll_component.port_clkena0 = "PORT_UNUSED",
altpll_component.port_clkena1 = "PORT_UNUSED",
altpll_component.port_clkena2 = "PORT_UNUSED",
altpll_component.port_clkena3 = "PORT_UNUSED",
```

```
                altpll_component.port_clkena4 = "PORT_UNUSED",
                altpll_component.port_clkena5 = "PORT_UNUSED",
                altpll_component.port_extclk0 = "PORT_UNUSED",
                altpll_component.port_extclk1 = "PORT_UNUSED",
                altpll_component.port_extclk2 = "PORT_UNUSED",
                altpll_component.port_extclk3 = "PORT_UNUSED",
                altpll_component.valid_lock_multiplier = 1;


endmodule

// ============================================================
// CNX file retrieval info
// ============================================================
// Retrieval info: PRIVATE: ACTIVECLK_CHECK STRING "0"
// Retrieval info: PRIVATE: BANDWIDTH STRING "1.000"
// Retrieval info: PRIVATE: BANDWIDTH_FEATURE_ENABLED STRING "0"
// Retrieval info: PRIVATE: BANDWIDTH_FREQ_UNIT STRING "MHz"
// Retrieval info: PRIVATE: BANDWIDTH_PRESET STRING "Low"
// Retrieval info: PRIVATE: BANDWIDTH_USE_AUTO STRING "1"
// Retrieval info: PRIVATE: BANDWIDTH_USE_CUSTOM STRING "0"
// Retrieval info: PRIVATE: BANDWIDTH_USE_PRESET STRING "0"
// Retrieval info: PRIVATE: CLKBAD_SWITCHOVER_CHECK STRING "0"
// Retrieval info: PRIVATE: CLKLOSS_CHECK STRING "0"
// Retrieval info: PRIVATE: CLKSWITCH_CHECK STRING "1"
// Retrieval info: PRIVATE: CNX_NO_COMPENSATE_RADIO STRING "0"
// Retrieval info: PRIVATE: CREATE_CLKBAD_CHECK STRING "0"
// Retrieval info: PRIVATE: CREATE_INCLK1_CHECK STRING "0"
// Retrieval info: PRIVATE: CUR_DEDICATED_CLK STRING "c0"
// Retrieval info: PRIVATE: CUR_FBIN_CLK STRING "c0"
// Retrieval info: PRIVATE: DEVICE_SPEED_GRADE STRING "6"
// Retrieval info: PRIVATE: DIV_FACTOR0 NUMERIC "1"
// Retrieval info: PRIVATE: DUTY_CYCLE0 STRING "50.00000000"
// Retrieval info: PRIVATE: EFF_OUTPUT_FREQ_VALUE0 STRING "200.000000"
// Retrieval info: PRIVATE: EXPLICIT_SWITCHOVER_COUNTER STRING "0"
// Retrieval info: PRIVATE: EXT_FEEDBACK_RADIO STRING "0"
// Retrieval info: PRIVATE: GLOCKED_COUNTER_EDIT_CHANGED STRING "1"
// Retrieval info: PRIVATE: GLOCKED_FEATURE_ENABLED STRING "1"
// Retrieval info: PRIVATE: GLOCKED_MODE_CHECK STRING "0"
// Retrieval info: PRIVATE: GLOCK_COUNTER_EDIT NUMERIC "1048575"
```

```
// Retrieval info: PRIVATE: HAS_MANUAL_SWITCHOVER STRING "1"
// Retrieval info: PRIVATE: INCLK0_FREQ_EDIT STRING "50.000"
// Retrieval info: PRIVATE: INCLK0_FREQ_UNIT_COMBO STRING "MHz"
// Retrieval info: PRIVATE: INCLK1_FREQ_EDIT STRING "100.000"
// Retrieval info: PRIVATE: INCLK1_FREQ_EDIT_CHANGED STRING "1"
// Retrieval info: PRIVATE: INCLK1_FREQ_UNIT_CHANGED STRING "1"
// Retrieval info: PRIVATE: INCLK1_FREQ_UNIT_COMBO STRING "MHz"
// Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "Cyclone II"
// Retrieval info: PRIVATE: INT_FEEDBACK__MODE_RADIO STRING "1"
// Retrieval info: PRIVATE: LOCKED_OUTPUT_CHECK STRING "1"
// Retrieval info: PRIVATE: LONG_SCAN_RADIO STRING "1"
// Retrieval info: PRIVATE: LVDS_MODE_DATA_RATE STRING "Not Available"
// Retrieval info: PRIVATE: LVDS_MODE_DATA_RATE_DIRTY NUMERIC "0"
// Retrieval info: PRIVATE: LVDS_PHASE_SHIFT_UNIT0 STRING "deg"
// Retrieval info: PRIVATE: MIG_DEVICE_SPEED_GRADE STRING "Any"
// Retrieval info: PRIVATE: MIRROR_CLK0 STRING "0"
// Retrieval info: PRIVATE: MULT_FACTOR0 NUMERIC "4"
// Retrieval info: PRIVATE: NORMAL_MODE_RADIO STRING "1"
// Retrieval info: PRIVATE: OUTPUT_FREQ0 STRING "100.00000000"
// Retrieval info: PRIVATE: OUTPUT_FREQ_MODE0 STRING "0"
// Retrieval info: PRIVATE: OUTPUT_FREQ_UNIT0 STRING "MHz"
// Retrieval info: PRIVATE: PHASE_RECONFIG_FEATURE_ENABLED STRING
"0"
// Retrieval info: PRIVATE: PHASE_RECONFIG_INPUTS_CHECK STRING "0"
// Retrieval info: PRIVATE: PHASE_SHIFT0 STRING "0.00000000"
// Retrieval info: PRIVATE: PHASE_SHIFT_STEP_ENABLED_CHECK STRING "0"
// Retrieval info: PRIVATE: PHASE_SHIFT_UNIT0 STRING "deg"
// Retrieval info: PRIVATE: PLL_ADVANCED_PARAM_CHECK STRING "0"
// Retrieval info: PRIVATE: PLL_ARESET_CHECK STRING "1"
// Retrieval info: PRIVATE: PLL_AUTOPLL_CHECK NUMERIC "1"
// Retrieval info: PRIVATE: PLL_ENA_CHECK STRING "0"
// Retrieval info: PRIVATE: PLL_ENHPLL_CHECK NUMERIC "0"
// Retrieval info: PRIVATE: PLL_FASTPLL_CHECK NUMERIC "0"
// Retrieval info: PRIVATE: PLL_FBMIMIC_CHECK STRING "0"
// Retrieval info: PRIVATE: PLL_LVDS_PLL_CHECK NUMERIC "0"
// Retrieval info: PRIVATE: PLL_PFDENA_CHECK STRING "0"
// Retrieval info: PRIVATE: PLL_TARGET_HARCOPY_CHECK NUMERIC "0"
// Retrieval info: PRIVATE: PRIMARY_CLK_COMBO STRING "inclk0"
// Retrieval info: PRIVATE: RECONFIG_FILE STRING "PLL.mif"
// Retrieval info: PRIVATE: SACN_INPUTS_CHECK STRING "0"
```

```
// Retrieval info: PRIVATE: SCAN_FEATURE_ENABLED STRING "0"
// Retrieval info: PRIVATE: SELF_RESET_LOCK_LOSS STRING "0"
// Retrieval info: PRIVATE: SHORT_SCAN_RADIO STRING "0"
// Retrieval info: PRIVATE: SPREAD_FEATURE_ENABLED STRING "0"
// Retrieval info: PRIVATE: SPREAD_FREQ STRING "50.000"
// Retrieval info: PRIVATE: SPREAD_FREQ_UNIT STRING "KHz"
// Retrieval info: PRIVATE: SPREAD_PERCENT STRING "0.500"
// Retrieval info: PRIVATE: SPREAD_USE STRING "0"
// Retrieval info: PRIVATE: SRC_SYNCH_COMP_RADIO STRING "0"
// Retrieval info: PRIVATE: STICKY_CLK0 STRING "1"
// Retrieval info: PRIVATE: SWITCHOVER_COUNT_EDIT NUMERIC "1"
// Retrieval info: PRIVATE: SWITCHOVER_FEATURE_ENABLED STRING "1"
// Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"
// Retrieval info: PRIVATE: USE_CLK0 STRING "1"
// Retrieval info: PRIVATE: USE_CLKENA0 STRING "0"
// Retrieval info: PRIVATE: USE_MIL_SPEED_GRADE NUMERIC "0"
// Retrieval info: PRIVATE: ZERO_DELAY_RADIO STRING "0"
// Retrieval info: LIBRARY: altera_mf altera_mf.altera_mf_components.all
// Retrieval info: CONSTANT: CLK0_DIVIDE_BY NUMERIC "1"
// Retrieval info: CONSTANT: CLK0_DUTY_CYCLE NUMERIC "50"
// Retrieval info: CONSTANT: CLK0_MULTIPLY_BY NUMERIC "4"
// Retrieval info: CONSTANT: CLK0_PHASE_SHIFT STRING "0"
// Retrieval info: CONSTANT: COMPENSATE_CLOCK STRING "CLK0"
// Retrieval info: CONSTANT: GATE_LOCK_SIGNAL STRING "NO"
// Retrieval info: CONSTANT: INCLK0_INPUT_FREQUENCY NUMERIC "20000"
// Retrieval info: CONSTANT: INTENDED_DEVICE_FAMILY STRING "Cyclone
II"
// Retrieval info: CONSTANT: INVALID_LOCK_MULTIPLIER NUMERIC "5"
// Retrieval info: CONSTANT: LPM_TYPE STRING "altpll"
// Retrieval info: CONSTANT: OPERATION_MODE STRING "NORMAL"
// Retrieval info: CONSTANT: PORT_ACTIVECLOCK STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_ARESET STRING "PORT_USED"
// Retrieval info: CONSTANT: PORT_CLKBAD0 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_CLKBAD1 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_CLKLOSS STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_CLKSWITCH STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_CONFIGUPDATE STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_FBIN STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_INCLK0 STRING "PORT_USED"
```

```
// Retrieval info: CONSTANT: PORT_INCLK1 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_LOCKED STRING "PORT_USED"
// Retrieval info: CONSTANT: PORT_PFDENA STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_PHASECOUNTERSELECT STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_PHASEDONE STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_PHASESTEP STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_PHASEUPDOWN STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_PLLENA STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANACLR STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANCLK STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANCLKENA STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANDATA STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANDATAOUT STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANDONE STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANREAD STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANWRITE STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clk0 STRING "PORT_USED"
// Retrieval info: CONSTANT: PORT_clk1 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clk2 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clk3 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clk4 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clk5 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena0 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena1 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena2 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena3 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena4 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena5 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_extclk0 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_extclk1 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_extclk2 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_extclk3 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: VALID_LOCK_MULTIPLIER NUMERIC "1"
// Retrieval info: USED_PORT: @clk 0 0 6 0 OUTPUT_CLK_EXT VCC "@clk[5..0]"
// Retrieval info: USED_PORT: @extclk 0 0 4 0 OUTPUT_CLK_EXT VCC
"@extclk[3..0]"
// Retrieval info: USED_PORT: areset 0 0 0 0 INPUT GND "areset"
// Retrieval info: USED_PORT: c0 0 0 0 0 OUTPUT_CLK_EXT VCC "c0"
// Retrieval info: USED_PORT: inclk0 0 0 0 0 INPUT_CLK_EXT GND "inclk0"
```

```
// Retrieval info: USED_PORT: locked 0 0 0 0 OUTPUT GND "locked"
// Retrieval info: CONNECT: @areset 0 0 0 0 areset 0 0 0 0
// Retrieval info: CONNECT: @inclk 0 0 1 1 GND 0 0 0 0
// Retrieval info: CONNECT: @inclk 0 0 1 0 inclk0 0 0 0 0
// Retrieval info: CONNECT: c0 0 0 0 0 @clk 0 0 1 0
// Retrieval info: CONNECT: locked 0 0 0 0 @locked 0 0 0 0
// Retrieval info: GEN_FILE: TYPE_NORMAL PLL.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL PLL.ppf TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL PLL.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL PLL.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL PLL.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL PLL_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL PLL_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf
// Retrieval info: CBX_MODULE_PREFIX: ON
```

- Módulo SPI

```
module spi (
        sck_in,
        miso,
        dado,
        new_data,
        start,
        sck,
        mosi,
        cs,
        busy);

input sck_in;
input miso;
input [23:0] dado [7:0];
input new_data;
input start;

output reg sck;
output reg mosi;
output reg cs;
output reg busy;
```

```
//reg [15:0] cont;
//reg [25:0] cont_ck;
reg [7:0] cont_ck1;
reg [4:0] index_bit;
reg [3:0] index_dado;

always@(posedge sck_in)
begin
//      cont_ck = cont_ck+1;
//      if(cont_ck>=2)
//      begin
//              cont_ck=0;
//              cont = cont+1;
//      end

        if(busy)
        begin
                cont_ck1 = cont_ck1 + 1;

                if(cont_ck1 == 2) mosi = dado[index_bit-1][index_dado];
                if(cont_ck1 == 100) sck = sck^1;
                if(cont_ck1 == 200)
                begin
                        cont_ck1 = 0;
                        sck = sck^1;
                        index_bit = index_bit - 1;
                        if(index_bit == 0)
                        begin
                                index_dado = index_dado + 1;
                                index_bit = 24;
                        end

                        if(index_dado>7)
                        begin
                                busy = 0;
                                cont_ck1 = 0;
                                index_bit = 24;
                                index_dado = 0;
                                cs = 1;
                        end
```

```
                end
        end

        if(new_data)
        begin
                busy = 1;
                cs = 0;
        end

        if(start)
        begin
                busy = 0;
                cont_ck1 = 0;
                index_bit = 24;
                index_dado = 0;
        end


end
endmodule
```