SIMULAÇÃO EM VHDL DE CONVERSOR CC-CC *BOOST* EM ARQUITETURA CELULAR PARALELA

ALEXANDRE B. MARCELO[1], MARCELO N. TIROLLI[2]

[1] Mestrando em Engenharia Elétrica, Bolsista CAPES, UNESP, Câmpus Sorocaba, alexandre.marcelo@gmail.com.

[2] Doutorando em Engenharia Elétrica, UNESP, Câmpus Bauru, mtirolli@gmail.com.

Área de conhecimento (Tabela CNPq): 3.04.04.03-7 Conversão e Retificação da Energia Elétrica

Apresentado no
4º Congresso de Pós-Graduação do IFSP
27 e 28 de novembro de 2019 - Sorocaba-SP, Brasil

RESUMO: Este trabalho apresenta uma forma de implementar e simular um conversor CC-CC *Boost* em arquitetura paralela utilizando a linguagem de descrição de hardware, possibilitando uma maior interação física no estudo de controladores em sistemas digitais como o FPGA. A partir das equações diferenciais que descrevem o funcionamento do conversor, as mesmas são discretizadas e em seguidas descritas em VHDL. Para a simulação do conversor em sua arquitetura de operação foi utilizado um software livre, com os resultados então obtidos deste comparados com outros softwares comerciais consolidados, permitindo assim a validação dos resultados.

PALAVRAS-CHAVE: eletrônica de potência; conversor elevador.

VHDL SIMULATION OF DC-DC BOOST CONVERTER IN CELLULAR PARALLEL ARCHITECTURE

ABSTRACT: This work presents a way to implement and simulate a DC-DC Boost converter in parallel architecture using hardware description language, allowing for a greater physical interaction on the study of controllers in digital systems such as FPGA. Starting from the differential equations describing the converter functionalities, they are discretized and later described in VHDL. For the converter simulation in its operating architecture an open software was used, with its then obtained results compaired with other consolidated commercial softwares ,allowing for the validation of the results.

KEYWORDS: power electronics; step-up converter.

INTRODUÇÃO

Conversores CC-CC *Boost* tem por finalidade fornecer uma tensão de saída em corrente contínua superior à tensão de entrada, sendo utilizado em diversas aplicações, como sistemas de geração distribuída, alimentação de sistemas de iluminação por LED, correção do fator de potência, entre outros. O conversor *Boost* pode operar em três modos de condução (contínuo, descontínuo e crítico), sendo o controle de seu chaveamento responsável por fazer o conversor operar num modo específico, fornecendo valores nas condições desejadas.

Sistemas digitais de controle, como DSP (Digital Signal Processor) e FPGA (Field Programmable Gate Array), oferecem grande flexibilidade no estudo de estratégias para controlar o chaveamento de conversores de energia [2, 5]. O FPGA consiste em um arranjo de blocos lógicos que podem ser fisicamente configurados, através de sua programação por linguagens de descrição de hardware como VHDL ou Verilog, de forma a estabelecer os critérios de operação do conversor. É possível implementar algoritmos que executam tarefas paralelas, aumentando significativamente o desempenho do sistema [4]. Os dados de entrada para o sistema de controle provêm de um conversor analógico/digital (ADC), no qual um sinal analógico no tempo contínuo é transformado em uma palavra binária em tempo discreto [1], demandando que o FPGA funcione de forma síncrona.

O comportamento do circuito do conversor boost, bem como o respectivo controle, podem ser implementados e simulados para eventuais estudos em softwares pagos, como por exemplo PSIM, MATLAB/Simulink e Vivado. Porém a linguagem VHDL possibilita o uso de ferramentas livres, como o GHDL. Baseado neste cenário, o objetivo deste trabalho é implementar e simular um conversor boost em arquitetura celular paralela através de programação em VHDL, visando facilitar o desenvolvimento de controladores e estratégias de controle em FPGA.

MODELAGEM DISCRETA

A estrutura do conversor *Boost*, ilustrada na Figura 1 possui um indutor em seu terminal de entrada que, dependendo da situação imposta pela razão cíclica no interruptor semicondutor, armazena energia da fonte de tensão, ou a entrega ao filtro capacitivo de forma a manter um nível de tensão maior em relação ao estágio de entrada, alimentando a carga.

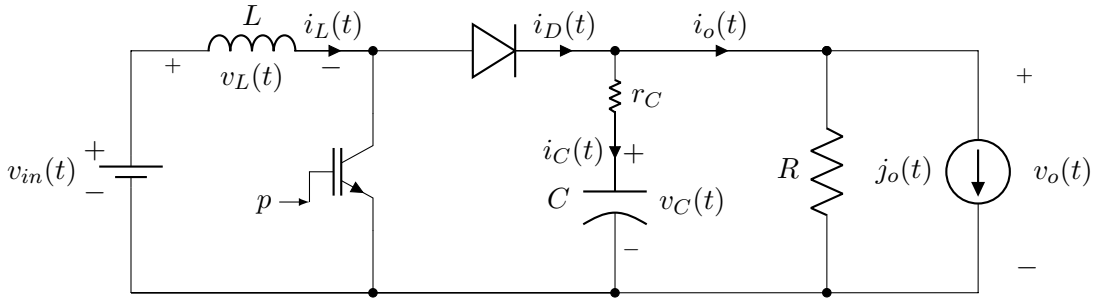


FIGURA 1 – Conversor CC-CC *Boost*

A energia armazenada no indutor é proporcional à variação de sua corrente $i_L(t)$, onde tal variação produz uma diferença de potencial $v_L(t)$ entre seus terminais, conforme a eq. (1). O filtro capacitivo por sua vez resiste a variações em sua tensão $v_C(t)$, consumindo uma corrente proporcional $i_C(t)$ conforme a eq. (2). Adicionalmente, o filtro capacitivo será modelado considerando sua resistência equivalente série r_C. A carga será modelada com uma resistência R, representando a potência ativa nominal do conversor, e uma fonte de corrente $j_o(t)$ em paralelo, permitindo a descrição de uma carga arbitrária através da variação instantânea da corrente total consumida $i_o(t)$, conforme a eq. (3).

$$v_L(t) = L\frac{di_L(t)}{dt} \quad (1) \qquad i_C(t) = C\frac{dv_C(t)}{dt} \quad (2) \qquad i_o(t) = \frac{v_o(t)}{R} + j_o(t) \quad (3)$$

Discretizando as equações diferenciais (1) e (2) através da regra dos trapézios, encontram-se as equações diferenças (4) e (5).

$$i_L[n+1] = i_L[n] + \frac{h}{2L}\left(v_L[n] + v_L[n+1]\right) \quad (4) \qquad v_C[n+1] = v_C[n] + \frac{h}{2C}\left(i_C[n] + i_C[n+1]\right) \quad (5)$$

O funcionamento do conversor consiste em até três etapas: o interruptor está comandado para a condução; o interruptor está comandado para o bloqueio, mas o indutor ainda possui energia para polarizar diretamente o diodo, permanecendo em condução; interruptor está comandado para o bloqueio, e o indutor não está em condução.

Enquanto o conversor opera com apenas uma célula, a tensão de saída $v_o(t)$ depende exclusivamente da tensão de entrada $v_{in}(t)$ e da condição imposta pelo sinal de comando p da chave semicondutora. Porém, para a arquitetura celular em configuração paralela, a tensão de saída $v_o(t)$ depende da condição de todas as células, devendo seu valor ser realimentado para o cálculo considerar a influência de células adicionais na resposta da célula modelada.

METODOLOGIA ESTRUTURADA EM VHDL

A implementação do código em VHDL empregou o método proposto por Gaisler [3], definido pela declaração dos sinais dentro de estruturas do tipo *record* e utilizando apenas dois processos por entidade, um processo combinatório lidando com a lógica assíncrona e um processo sequencial gerenciando o uso de registradores. Vale notar que, apesar da lógica combinatória ser assíncrona, implementá-la dentro de um processo a torna sequencial, eliminando erros de atribuições concorrentes para um mesmo sinal.

Partindo dessa visão geral, cada configuração instantânea da operação de uma célula do conversor será o sistema de estados composto pelas características de funcionamento dos elementos $v_L[n]$,$i_L[n]$,$v_C[n]$ e $i_C[n]$, unidas pela aplicação das leis de Kirchhoff conforme cada etapa no código abaixo:

```
type boost_in_type is record
  v_in  : real;           -- Input Voltage
  p     : std_logic;      -- Switching Pulse
  j0    : real;           -- Variable Output Current
  v_out : real;           -- Output Voltage Feedback
end record;

type boost_out_type is record
  v_out   : real;         -- Output Voltage
  iL_out  : real;         -- Input Current
  iL_zero : std_logic;    -- Null Current State, for Boundary conditions
end record;

type boost_reg_type is record
  vC  : real;             -- Capacitor Voltage
  iC  : real;             -- Capacitor Current
  vL  : real;             -- Inductor Voltage
  iL  : real;             -- Inductor Current
  iL_zero : std_logic;    -- Null Current State
end record;

entity boost is
  generic (
    L     : real;         -- Input Inductance in uH
    R     : real;         -- Load Resistance in ohm
    C     : real;         -- Output Capacitance in uF
    rC    : real;         -- Capacitor ESR in ohm
    sf    : real);        -- Sampling Frequency in MHz
  port (
    clk   : in std_logic; -- Must have the same frequency as sf
    b_in  : in boost_in_type;
    b_out : out boost_out_type);
  end;

architecture behavioral of boost is
  signal b_reg, b_reg_in : boost_reg_type; -- State Registers

begin
  combinatorial  : process(b_in, b_reg)
    variable b_var : boost_reg_type; -- Variable State Register, avoids latches
    variable iD    : real;           -- Diode Current
  begin
    b_var := b_reg;          -- Default Variable State Register to Previous State

    if b_var.iL <= 0.0 then  -- Inductor not conducting
      b_var.iL_zero := '1';  -- Set Null Current State
      b_var.iL := 0.0;       -- Correct Inductor Current to non-negative
      b_var.vL := 0.0;       -- Inductor behaves as a Short Circuit
    else                     -- Inductor is conducting
      b_var.iL_zero := '0';  -- Reset Null Current State
      b_var.vL := b_in.v_in - b_in.v_out; -- Inductor Voltage
    end if;

    if b_in.p = '1' then     -- Switch commanded to conduct
      b_var.vL := b_in.v_in; -- Inductor Voltage is only Input Voltage
      iD := 0.0;             -- Diode is blocked
    else
      iD := b_var.iL;        -- Diode is conducting
    end if;

    b_var.iC := -b_in.j0 + iD - (b_reg.vC + rC*b_reg.iC)/R;  -- Kirchhoff
    b_var.iL := b_var.iL + (b_var.vL + b_reg.vL)/(2.0*L*sf);
                  -- Update Inductor Current via Discrete Trapezoidal Rule
    b_var.vC := b_var.vC + (b_var.iC + b_reg.iC)/(2.0*C*sf);
                  -- Update Capacitor Voltage via Discrete Trapezoidal Rule

    b_reg_in <= b_var;                    -- Update Next State Register
    b_out.v_out <= b_reg.vC + rC*b_reg.iC; -- Update Output Voltage
```

```
    b_out.iL_out <= b_reg.iL;              -- Update Input Current
    b_out.iL_zero <= b_reg.iL_zero;        -- Update Null Current State
  end process;

  sequential : process(clk)
  begin                       -- Synchronously Update Current State Register
    if rising_edge(clk) then b_reg <= b_reg_in; end if;
  end process;
end behavioral;
```

RESULTADOS E DISCUSSÃO

Foram simulados dois conversores em arquitetura celular paralela com o seguinte código em VHDL:

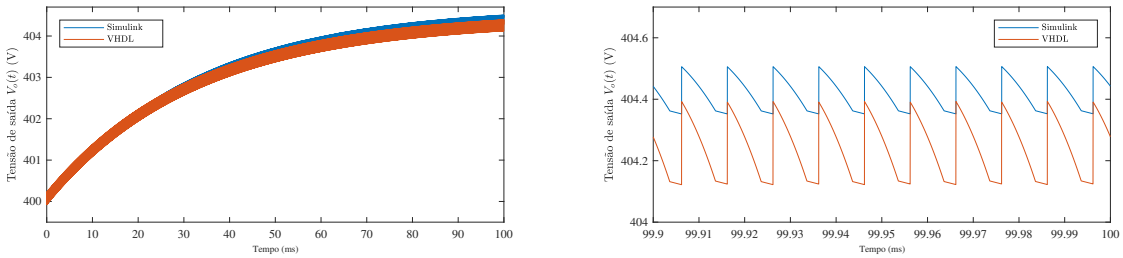
```
architecture behavioral of system_sim is
  signal clk            : std_logic;
  signal b_in1,b_in2    : boost_in_type;
  signal b_out1,b_out2  : boost_out_type;
  signal pulse,pulse2   : std_logic;
  signal v_in,j0,v_out  : real;

begin
  clk <= not clk after 5 ns;
  pulse <= '0' after 6.2 us when pulse= '1' else '1' after 13.8 us;
  pulse2 <= transport pulse after 10 us;
  b_in1 <= (v_in,pulse,j0,v_out);
  b_in2 <= (v_in,pulse2,j0,v_out);

  B1: boost generic map ( L => 200.0,
                          R => 320.0,
                          C => 330.0,
                          rC => 0.045)
              port map ( clk => clk,
                         b_in => b_in1,
                         b_out => b_out1);

  B2: boost generic map ( L => 200.0,
                          R => 320.0,
                          C => 330.0,
                          rC => 0.045)
              port map ( clk => clk,
                         b_in => b_in2,
                         b_out => b_out2);

  process(b_out1,b_out2)
  begin
    if b_out1.v_out >= b_out2.v_out then
      v_out <= b_out1.v_out;
    else
      v_out <= b_out2.v_out;
    end if;
  end process;
end behavioral;
```



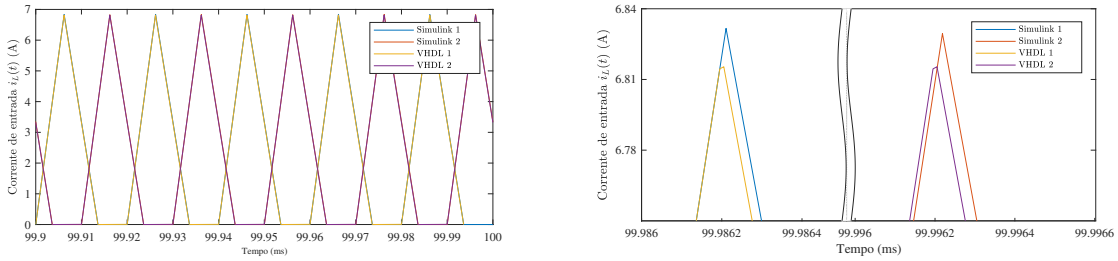
(a) (b)

FIGURA 2 – Tensão de saída

Um circuito equivalente foi implementado no Simulink com o auxílio do *toolbox* SimPowerSystems e simulado nas mesmas condições de operação, notavelmente no modo descontínuo, em malha aberta e utilizando a estratégia de entrelaçamento entre as células, de modo a permitir uma comparação entre os dois ambientes de simulação. A Figura 2a ilustra o comportamento da

tensão de saída, com o detalhe das ondulações de chaveamento apresentadas na Figura 2b. O erro relativo médio se manteve abaixo de $0,05\%$.

As correntes nos indutores, ilustrando na Figura 3a a defasagem característica da técnica de entrelaçamento, apresentaram erro relativo médio em torno de $0,2\%$, com erro máximo menor que $0,45\%$, conforme detalha a Figura 3b.



(a) (b)

FIGURA 3 – Correntes nos Indutores

Os resultados demonstram que o uso de linguagem de *hardware* em *software* livre reproduz resultados compatíveis com os adquiridos em ambientes com licenças comerciais.

CONCLUSÕES

O objetivo de utilizar a linguagem VHDL para simular o comportamento de um conversor *Boost* em arquitetura celular paralela foi atingido, com resultados demonstrando uma precisão similar ao atingido por ferramentas mais consolidadas porém menos acessíveis, e permitindo uma maior interação com a implementação física de sistemas experimentais de controle.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, através da Pró-Reitoria de Pós-Graduação da Universidade Estadual Paulista "Júlio de Mesquita Filho" – UNESP.

REFERÊNCIAS

[1] Barragán, L., Urriza, I., Navarro, D., Artigas, J., Acero, J., and Burdio, J. Comparing simulation alternatives of FPGA-based controllers for switching converters. In *2007 IEEE International Symposium on Industrial Electronics* (2007), IEEE, pp. 419–424.

[2] Chithra, S., and Maheswaran, V. D. Design and implementation of digital controller for DC-DC buck boost converter. *International Journal of Scientific & Engineering Research 5*, 4 (2014), 2229–5518.

[3] Gaisler, J. A structured VHDL design method. *Fault-tolerant microprocessors for space applications* (2011), 41–50.

[4] Sadek, U., Sarjaš, A., Svečko, R., and Chowdhury, A. FPGA-based control of a DC-DC boost converter. *IFAC-PapersOnLine 48*, 10 (2015), 22–27.

[5] Urriza, I., Barragán, L., Artigas, J., Navarro, D., and Lucía, O. FPGA implementation of a digital controller for a dc-dc converter using floating point arithmetic. In *2009 35th Annual Conference of IEEE Industrial Electronics* (2009), IEEE, pp. 2913–2918.