Desenvolvimento de um "*Counter*" & "*Shifter*" em VHDL

Sérgio Veiga

Resumo **- Este artigo descreve um circuito baseado numa FPGA *(Field Programmable Gate Array)* que se pode comportar como um *shifter* ou como um *counter*. Foi usada uma placa TE-XC2Se da Trenz Electronic, baseada na FPGA XC2S300E da família Spartan-IIE da Xilinx, sendo os resultados visualizados num monitor VGA Toda a descrição do circuito foi feita na linguagem VHDL recorrendo ao ambiente de desenvolvimento Xilinx ISE 6.3.03i. Este artigo foca todo processo de desenvolvimento efectuado de forma a proporcionar uma melhor perspectiva em relação à linguagem VHDL e às potencialidades da ferramenta Xilinx ISE 6.3.03i.**

Abstract **- This paper describes implementation of an FPGA-based circuit which is able to execute either count or shift operation. For the experiments TE-XC2Se prototyping board of Trenz Electronic was employed, which includes one XC2S300E Spartan – IIE FPGA of Xilinx. The results of experiment are displayed on a VGA monitor screen connected to the board. The circuit description was done in VHDL language within the Xilinx ISE 6.3 environment. The focus of this paper is to give a better view of the VHDL language, and Xilinx software through describing all the work done with the *Counter* & *Shifter* project.**

I. INTRODUÇÃO

O trabalho aqui descrito foi efectuado como projecto final da disciplina CR (Computação Reconfigurável), disciplina do curso de Licenciatura em Engenharia de Computadores e Telemática, leccionada no 2º semestre do 4º ano pelo Professor Valery Sklyarov e pela professora Iouliia Skliarova. Todas as bases teóricas e práticas necessárias para o desenvolvimento de todo o trabalho a seguir descrito foram adquiridas no âmbito desta disciplina.

A. Características principais da placa TE-XC2Se

As caracteristicas principais da placa TE-XC2SE (ilustrada na Fig. 1) são as seguintes:[1]

- 8 Mbit (1M * 8 / 512K * 16) de Memória Flash
- 4 Mbit (256K * 16) de RAM estática
- FPGA XC2S300E com 300.000 portas de sistema
- LCD com 2*16 caracteres
- Interface USB (*Universal Serial Bus*)
- LEDs, interruptores e botões
- 2 osciladores que geram sinais de relógio de 48 MHz e 25 Mhz
- porta VGA
- porta RS-232
- CPLD

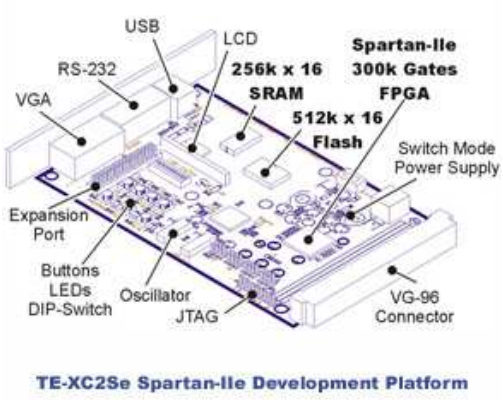

Fig. 1 – Placa TE-XC2Se

B. Especificação do Projecto

O projecto tem como objectivo elaborar um circuito que se comporte como um *shifter* ou como um *counter*. Este circuito é mostrado como um bloco digital virtual no monitor VGA sendo controlado a partir dos botões e dos interruptores da placa. Um dos botões da FPGA permite escolher qual a entidade activa, o *counter* ou o *shifter*, outro botão caso o *counter* esteja seleccionado permite escolher entre uma contagem positiva ou negativa, ou caso seja o *shifter* a estar activo permite escolher o modo de realimentação deste, ou seja se o bit que entra à esquerda é o bit que tinha sido deslocado à direita ou se entra 0. Outro botão funcionaria como clock enable, realizando uma operação de *count* ou *shift* cada vez que é pressionado. Temos ainda a opção de inicializar o estados do bloco digital a zeros usando botão de reset da placa. No monitor VGA é mostrado qual o estado do sistema ou seja qual a entidade activa,

o respectivo modo de funcionamento desta e o seu estado.

II. ESTRUTURA DO PROGRAMA

A. Estrutura do programa

O circuito está dividido em cinco módulos principais (descritos em VHDL), um UCF (*User Constraints File*), um esquemático (ilustrado na Fig. 2) e um pacote.

O UCF fpga.ucf especifica a relação entre os pinos da FPGA e os sinais de forma a podermos usufruir do monitor VGA, dos botões e interruptores da placa.

O bloco *button-behavioral* tem como função disponibilizar o estado dos botões e dos interruptores aos restantes blocos do circuito. Neste bloco tivemos o especial cuidado de usar um processo de *debouncing* de forma a evitar uma indefinição de estado dos botões devido a *contact bouncing*.

O bloco *counter-behavioral* efectua a operação de contagem positiva ou negativa conforme a opção seleccionada.

O bloco *shifter-behavioral* efectua a operação de deslocamento com ou sem realimentação conforme a opção seleccionada.

O pacote *alphabet.vhd* define o aspecto das letras a serem desenhadas no ecrã. Por exemplo para desenhar um zero é necessário associar o caracter a um *array* de vectores que contém, por exemplo o seguinte conteúdo:

```
0 =>("0111111111100000",
     "1111111111110000",
     "1100000000110000",
     "1100000000110000",
     "1110000001110000",
     "1100000000110000",
     "1100000000110000",
     "1100000000110000",
     "1100000000110000",
     "1100000000110000",
     "1111111111110000",
     "0111111111100000",
     "0000000000000000",
     "0000000000000000",
     "0000000000000000",
     "0000000000000000")
```

O bloco *symbol_rom-behavioral* funciona como memória para o armazenamento dos caracteres definidos no pacote alphabet.vhd.

O bloco *vga-bhv* é o bloco central do circuito pois não só desenha o circuito no ecrã como é o responsável pela interligação entre os vários blocos descritos, de forma a poder representar o requerido pelo utilizador.

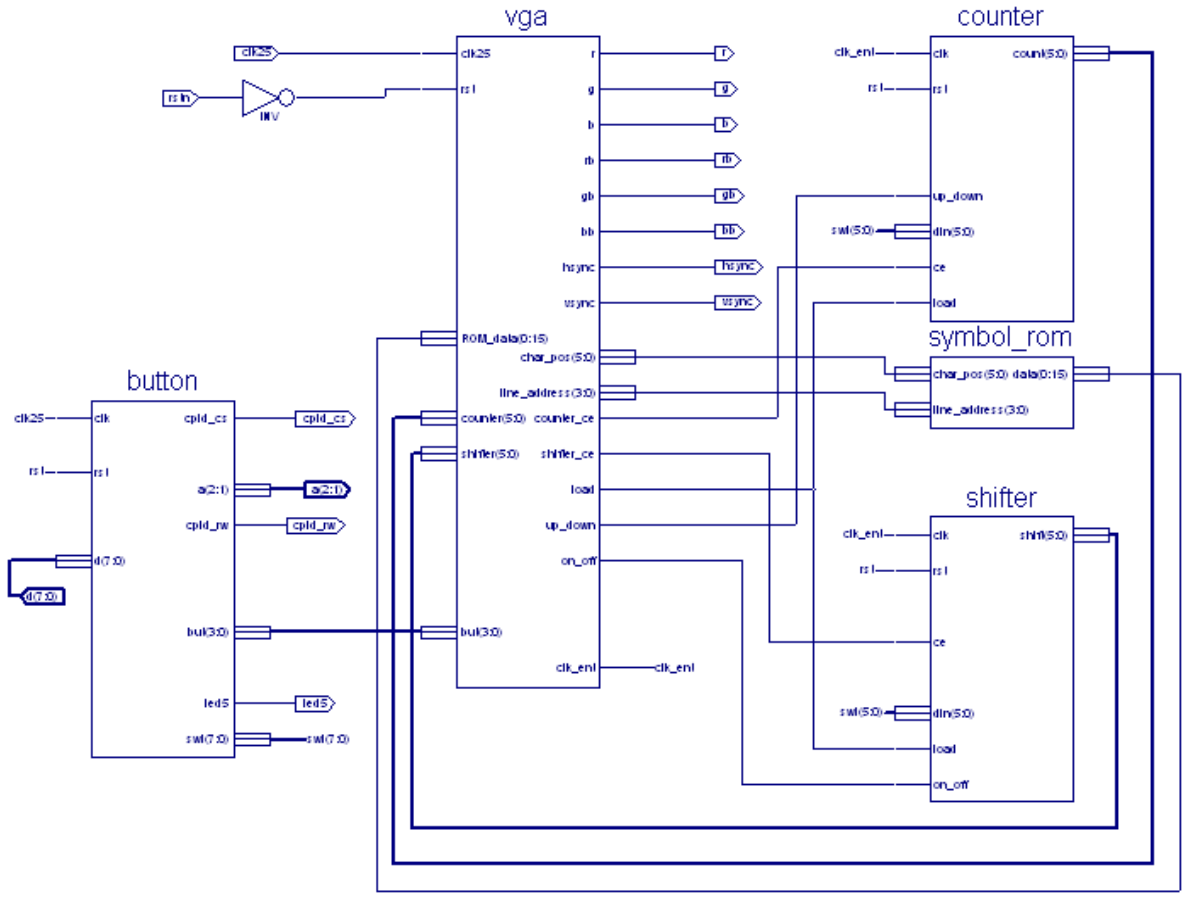

Fig. 2 – Esquemático do projecto

B. Exemplo de código

Para uma melhor compreensão de como a linguagem VHDL permite descrever este género de circuitos é apresentado a seguir o código pertencente ao bloco *counter*:

```
entity counter is
port
(       clk, rst,load,ce,up_down : in std_logic;
        din : in std_logic_vector(5 downto 0);
        count : inout std_logic_vector(5 downto 0)

);
end counter;

architecture Behavioral of counter is
begin
process(clk, rst)
begin
        if (rst = '1') then
                count <= (others => '0');
        elsif rising_edge(clk) then
                if ce = '1' then
                        if load = '1' then
                                count <= din;
                        elsif up_down='1' then
                                count <= count + 1;
                        else
                                count <= count - 1;
                        end if;
                end if;
        end if;
end process;
end Behavioral;
```

Depois de escolher um nome para dar à entidade, que neste caso se chama counter, definimos a interface externa do bloco através da syntax: port{ porta : tipo}, ou seja definimos as portas de entrada e saída do bloco bem como o tipo destas. De seguida é especificado o comportamento do bloco, ou seja a função que o bloco vai desempenhar. Dentro deste bloco podemos criar processos que estão associados às portas especificadas anteriormente que estão ligadas a sinais externos ao bloco. Sempre que um dos sinais que pertence à lista de sensibilidade do processo muda de valor o processo é executado. Assim sendo no caso do nosso counter, sempre que os sinais que estão associados às portas clk, ou rst, mudam de valor o processo é executado.

III MODO DE UTILIZAÇÃO

A placa TE-XC3Se dispõe de 8 interruptores e 4 botões. Foram usados 6 interruptores (indicado na Fig. 3) para especificar qual o vector a partir do qual quer efectuar o processo de contagem ou deslocamento. Ou seja o utilizador especifica o vector e carrega no botão "Load/WE" de forma a colocar o sistema em modo de LOAD, depois carrega no botão "clk" que vai funcionar como clock enable do sistema, ou seja mostra o vector seleccionado através dos interruptores no ecrã. Para accionar qualquer dos blocos o utilizador coloca o sistema em modo Write Enable (WE) e carrega no botão de "clk" para accionar o sistema. Agora o utilizador pode escolher o tipo de operação que deseja, se contagem ou deslocamento através do botão "Tipo de Circuito". Depois de escolher a operação o utilizador pode escolher como pretende efectuar a contagem ou o deslocamento. Para isso basta pressionar o botão "tipo de operação". Para colocar o vector todo a 0, basta pressionar o botão reset da placa.

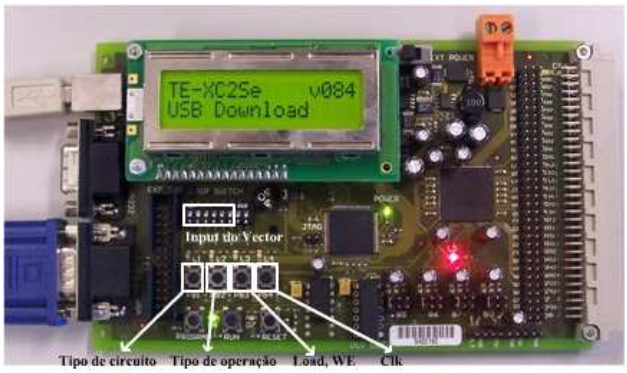

Fig. 3 – Modo de utilização da placa TE-XC2Se

O resultado aparece no ecrã na seguinte forma:

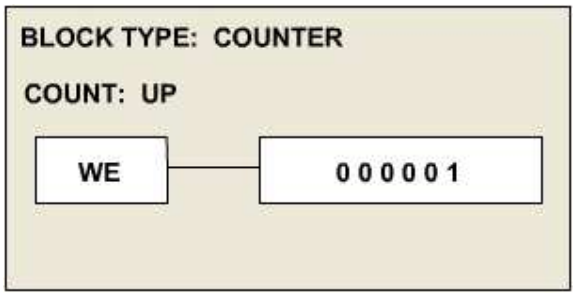

Fig. 4 – Visionamento no ecrã do Bloco digital Virtual

IV CONCLUSÃO

Este projecto foi bastante interessante quer do ponto de vista da tecnologia usada como da parte educativa inerente à realização do mesmo. O circuito só por si é bastante simples sendo ideal para uma introdução à linguagem VHDL e ao ambiente Xilinx, não deixando também de elucidar todos, para as grandes potencialidades que proporciona. Na minha percepção este trabalho tornou-se bastante enriquecedor sobre todos os pontos de vista. De salientar que todo este projecto só foi possível de realizar ao estudar os tutoriais disponíveis e a matéria dada nas aulas de Computação Reconfigurável leccionadas pelo Professor Valery Sklyarov [2] e pela professora Iouliia Skliarova [3]. As principais dificuldades sentidas na realização deste trabalho foram proporcionadas pelo

ambiente de desenvolvimento da Xilinx que em algumas situações não produz um feedback adequado para resolver alguns dos erros que foram aparecendo ao longo do projecto. Em conclusão o trabalho realizado corresponde aos requisitos predefinidos inicialmente deixando ainda espaço para melhoramentos quer ao nível das funcionalidades como no que respeita à optimização de código. O projecto que inclui todas as partes apresentadas acima está disponível para download no site do BlackBoard. [2]

REFERÊNCIAS

[1] http://www.trenz-electronic.de/prod/proden12.htm.

[2] http://elearning.ua.pt, "2º Semestre", disciplina "Computação Reconfigurável".

[3] http://www.ieeta.pt/~iouliia/courses/CR/.