Desenvolvimento de um analisador lógico simples

Luis Gomes

***Resumo* - Este artigo descreve um sistema baseado numa FPGA (*Field Programmable Gate Array*) que implementa circuitos lógicos e sequenciais elementares, e representa as suas saídas através de diagramas temporais, interagindo com um monitor VGA ligado à FPGA. A especificação do sistema foi feita em VHDL. Esta especificação em VHDL foi sintetizada e convertida num *bitstream* para FPGA no ambiente *Xilinx ISE 6.3*. O sistema foi testado na FPGA da família Spartan-IIE XC2S300E que é o componente reconfigurável principal da placa TE-XC2SE fornecida pela Trenz Electronics.**

***Abstract* - This paper describes an FPGA-based system that implements simple logical and sequential circuits and represents their outputs with time diagrams, interacting with a VGA monitor connected to the FPGA. The functionality of the circuit has been described in VHDL. The VHDL specification was synthesized and converted in Xilinx ISE 6.3 environment to a bitstream for FPGA. The designed circuit was tested in an XC2S300E Spartan-IIE FPGA, which is a primary reconfigurable component of TE-XC2SE board supplied by Trenz Electronics.**

I . INTRODUÇÃO

O trabalho detalhado neste artigo foi desenvolvido na disciplina de Computação Reconfigurável (CR), leccionada na Licenciatura em Engenharia de Computadores e Telemática, na Universidade de Aveiro, cujo docente é Valery Sklyarov. Os temas abordados e estudados nesta disciplina permitiram fundar as bases e conhecimentos necessários para a linguagem VHDL, de descrição de hardware, assim como a utilização do ambiente ISE 6.3 da Xilinx.

A. Características principais da placa TE-XC2SE

A placa Trenz Electronics TE-XC2SE tem como componente reconfigurável principal a FPGA XC2S300E [1] da família Spartan-IIE da Xilinx. Esta placa oferece:

- 6912 células lógicas (300.000 portas de sistema, 1536 CLB's)
- 98.304 bits de RAM distribuída;
- 64K bits de RAM dedicada.
- A FPGA tem ligação com oscilador de quartzo de 48 MHz (para USB);
- oscilador de quartzo (utilizado de 25 MHz, para VGA);
- vídeo DAC (VGA 8 bits);
- LED's;
- porta USB;
- porta série de 9 pinos;
- DIP *switches*;
- botões;
- LCD de 2*16 caracteres;
- dois barramentos de expansão;
- SRAM;
- Memória Flash.

A comunicação com a placa é feita através da interface USB, com suporte para controladores UHCI (USB 1.0), OHCI (USB 1.1) e EHCI (USB 2.0), diminuindo os tempos de configuração da placa conforme o controlador usado.

B. Especificação do projecto

O projecto consiste no desenvolvimento de um sistema na linguagem VHDL que permite representar no ecrã os diagramas temporais de alguns blocos lógicos, cujas entradas são fornecidas através dos *Dip Switches* e botões, apresentando-os no ecrã de um monitor VGA, como pode ser visto na Fig. 1.

No ecrã surgem os 10 diagramas temporais das saídas dos vários blocos – uma máquina de estados, operada pelos *switches*; um shift register; um OR, AND e XOR, os 3 operados através dos botões. Além destes, um 10º diagrama temporal representa o estado de um buffer ligado a um botão.

O monitor VGA está ligado à porta correspondente (VGA de 15) existente na placa Trenz.

II . ESTRUTURA DO PROJECTO EM VHDL

A. Estrutura básica do projecto

O projecto foi desenvolvido numa estrutura constituída por 6 módulos e uma *package* em VHDL, conforme ilustrado na Fig. 2, e um *User Constraint File* (UCF).

O UCF (fpga.ucf) contém as definições do relógio, e de mapeamento de pinos.

A *package* Alphabet.vhd define um alfabeto de dígitos de 0-9 e de A-Z, e os 5 símbolos de representação de sinal (0, 1, as 2 respectivas transições, e ainda um símbolo de inêxistencia de sinal (no tempo)). Cada um destes caracteres é codificado numa matriz 16x16.

A especificação esquemática top_vga.sch define o nível de topo do programa, onde se encontram os restantes módulos utilizados.

A divisão dos sinais de relógio, para obtenção de um relógio com o período de 0.5s, é efectuada pelo módulo descrito em clk_div.vhd.

Os módulos que descrevem a máquina de estados e o shift register estão, respectivamente, em state_machine.vhd e shift_reg.vhd. O funcionamento da máquina de estados está ilustrado na Fig. 3.

Nos módulos VGA.vhd e button.vhd estão descritas as interacções com o monitor VGA e com os botões, respectivamente. O módulo symbol_rom utiliza a *package* Alphabet para construir uma memória com os dígitos a representar.

No esquemático de topo surgem ainda outras estruturas: o OR2, AND2 e XOR2, que são os blocos lógicos cujas saídas estamos a representar.

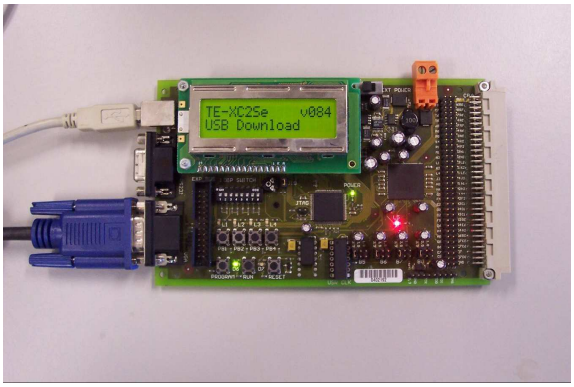

Fig. 1 – Spartan IIE ligada ao interface VGA

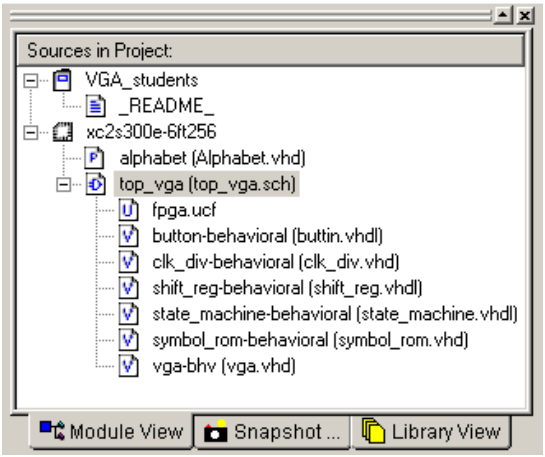

Fig. 2 – Módulos do projecto

III . CONTROLO DOS DISPOSITIVOS

A. Controlo das entradas dos blocos lógicos

A interacção com os blocos lógicos é feita da seguinte forma:

DIP Switches (0,1) – Entradas da Máquina de Estados;

DIP Switches (7-4) – Entrada do Shift Register;

Botão 0 – Load do Shift Register;

Botões (1,2) – Entradas dos blocos OR,AND,XOR;

Botão 3 – Entrada do buffer.

No ecrã surgem os 10 diagramas temporais das saídas dos vários blocos (Fig. 3)– uma máquina de estados (diag. 0,1), operada pelos *switches* (0-1); um shift register (diag. 2-5), onde é lido o valor dos *switches* (7-4) em cada ciclo (virtual); um OR (diag. 6), AND (diag. 7) e XOR (diag. 8), os 3 operados através dos botões (1,2). Além destes, um último diagrama temporal representa o estado de um buffer ligado ao botão 3.

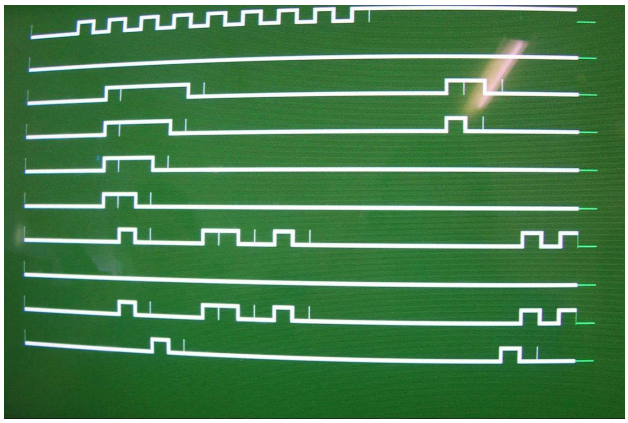

Fig. 3 – Display com o programa em execução.

Uma vez que o relógio dos blocos lógicos tem de período 0.5 segundos, é necessário que uma entrada (e.g. pressionar um botão) dure o tempo suficiente para que esteja activo durante uma transição de relógio, de forma a activar o bloco lógico a que se destina.

B. Descrição do Output no Ecrã

Foi utilizada uma resolução de 640x480 pixels, sendo que o interface com o VGA utilizado se baseia no código ilustrativo da Trenz Electronic.

O ecrã apresenta, como ilustrado na Fig. 3, 10 diagramas temporais, com 32 ciclos de duraçao:

- diagramas [0,1] : output da state machine;
- diagramas [2-5] : output do shift register;
- diagrama [6] : ouput do OR lógico;
- diagrama [7] : output do AND lógico;
- diagrama [8] : output do XOR lógico;
- diagrama [9] : output do buffer para o botão.

D. Descrição detalhada do Projecto

O divisor de relógio foi uma introdução necessária para podermos assegurar, por um lado, a visualização dos resultados em ciclos de 0.5s e, por outro, a continuidade do funcionamento dos outros módulos que requerem um relógio de mais alta frequência, tais como os interfaces com o VGA ou com os botões.

Assim, e para evitar a redundância de código que originaria introduzir esta divisão em cada um dos módulos, optou-se por criar uma entidade separada que se encarrega dessa mesma divisão:

```
entity clk_div is
        Port ( clk : in std_logic;
               rst : in std_logic;
               subclk : out std_logic);
end clk_div;

architecture Behavioral of clk_div is
signal internal: std_logic_vector (23 downto 0);
begin
process(clk, rst)
begin
        if rst = '1' then
        internal <= (others=>'0');
        elsif rising_edge(clk) then
                internal <= internal+1;
        end if;
end process;
subclk <= internal(internal'left);
end Behavioral;
```

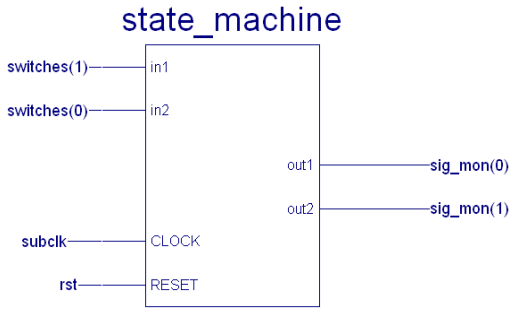

Fig. 4.1 – Símbolo esquemático da máquina de estados

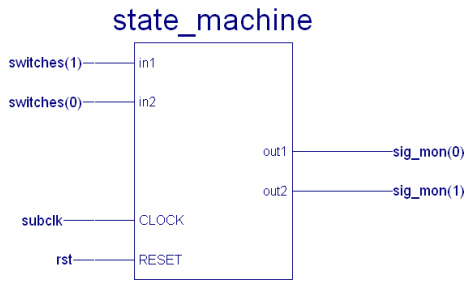

Fig. 4.2 – Máquina de estados utilizada.

A máquina de estados cujas entradas são os *Dip Switches* 0 e 1, tem o funcionamento ilustrado nas Fig. 4.1 e 4.2. A necessidade de utilzação de um relógio com uma frequência mais baixa, para efeitos de visualização, levou-nos a ligar o sinal subclk (proveniente do divisor de relógio) à entrada de relógio desta máquina de estados, como se pode ver no modelo esquemático apresentado na Fig. 4.1.

D3. Interface com o VGA

O módulo de interface (ver Fig. 5) apresenta algumas modificações relativamente à versão utilizada nas aulas práticas da disciplina [2]. Em primeiro, foi introduzida uma entrada para o relógio dividido *(subclk)*, para representação dos sinais no ecrã segundo a frequência pretendida. As entradas destes sinais são o array *sig_mon*.

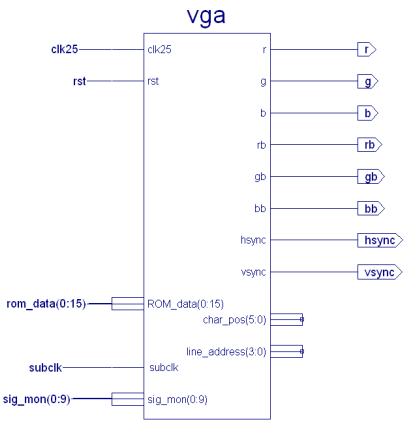

Fig. 5. Modelo esquemático do módulo de interface com o VGA

Foi criado, em primeiro, um processo interno para o desenho dos vários diagramas temporais, tal como as estruturas de dados necessárias: uma matriz para o armazenamento do histórico de 32 ciclos dos sinais a representar, e um contador que indica o ciclo actual.

O processo que lida com o desenho do ecrã, assim, quando está a "varrer" uma linha em que é necessário desenhar um sinal, converte esse número de linha (removendo o offset das linhas iniciais em que não há sinais, e dividindo-o por 2) para o índice da linha da matriz do histórico a que o sinal diz respeito. O mesmo acontece com as colunas, que são igualmente convertidas para indíces das colunas dessa matriz, como ilustrado no excerto de código seguinte.

```
case text_row is
when 2|4|6|8|10|12|14|16|18|20 =>
        row := (conv_integer(text_row)/2)-1;
        col := conv_integer(text_col)-4;
```

De seguida, verifica-se qual o ciclo actual (coluna) e, se este existir (i.e., já se encontra actualizado na matriz), é desenhado. Caso contrário o sinal é representado como símbolo de inêxistencia de sinal, como ilustrado na Fig. 3. O processo de desenho do valor do sinal em determinado ciclo não é trivial, já que não se trata apenas de desenhar

um “0” ou um “1”. É necessário ter em conta o valor anterior do sinal para que a transição seja correctamente representada: um “0” precedido de um “0” representa-se por “_”, mas um “0” precedido de um “1” é representado por “|_”.

```
seq <= matrix(col-1)(row) & matrix(col)(row);
case seq is -- checks current and previous value
            -- for edge detection
        when "00" => char_pos <= "100100"; -- “_”
        when "10" => char_pos <= "100110"; --”|_”
        when "01" => char_pos <= "100111"; -- “¯”
        when "11" => char_pos <= "100101"; --”|¯”
        when others => char_pos <= "000110";
end case;
pixel <= ROM_data(char_col);
```

No excerto de código anterior, podemos ver ilustrado este processo de decisão.

É também desenhada uma “grelha” de background para os valores que não conhecemos, i.e., os ciclos que ainda não aconteceram:

```
elsif (col < 32) then
        char_pos <= "101000";
        grid <= '1';-- used for background grid
        pixel <= ROM_data(char_col);
```

O valor auxiliar grid tem como função alterar a cor no processo de desenho do pixel, para que seja representada a referida grelha com cor diferente.

IV. CONCLUSÕES

Este artigo demonstra a facilidade de implementar circuitos de baixa complexidade numa FPGA, utilizando VHDL. O objectivo deste projecto foi a demonstração da utilização do monitor VGA com a FPGA.

Aperfeiçoamentos seguintes no projecto poderiam centrar-se no desenvolvimento de um histórico dos sinais (ao invés dos actuais 32 ciclos), que pudesse ser percorrido com os botões. Outro melhoramento possível seria o da introdução de uma opção de pausar e retomar o circuito, de forma a visualizar os sinais de uma forma estática.

REFERÊNCIAS

[1] “Spartan-IIE FPGA Family”, 2003: http://www.xilinx.com/spartan2e3.

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