Desenvolvimento de um circuito aritmético em VHDL

Bruno Monteiro

Resumo **- Este artigo descreve um circuito baseado numa FPGA (*Field Programmable Gate Array*) que implementa duas operações aritméticas (+, -) e que interage com um monitor e com um painel táctil ligados à FPGA. A especificação do circuito foi feita em VHDL. Esta especificação em VHDL foi compilada e convertida num *bitstream* para FPGA no ambiente *Xilinx ISE 5.2i*. O circuito foi testado na FPGA da família Spartan-IIE XC2S300E que é o componente reconfigurável principal da placa TE-XC2SE fornecida pela Trenz Electronics. Este artigo mostra algumas das potencialidades de VHDL e do Xilinx ISE, bem como as facilidades na interacção com os diversos componentes externos.**

Abstract **- This paper describes an FPGA-based circuit that implements two arithmetical operations (+, -) and interacts with a monitor and a touch panel connected to the FPGA. Functionality of the circuit has been described in VHDL. The VHDL specification was compiled and converted in Xilinx ISE 5.2i environment to a bitstream for FPGA. The designed circuit was tested in a FPGA Spartan-IIE XC2S300E, which is a primary reconfigurable component of TE-XC2SE board supplied by Trenz Electronics. The paper demonstrates some capabilities of VHDL and Xilinx ISE, as well as interaction features with external components.**

I . INTRODUÇÃO

O trabalho aqui descrito vem no seguimento da disciplina CR (Computação Reconfigurável) leccionada no plano de estudos da Licenciatura em Engenharia de Computadores e Telemática, no 4.º ano, 2.º semestre, pelo Professor Valery Sklyarov. A matéria da disciplina forneceu as bases necessárias para a linguagem de descrição de hardware VHDL, assim como a utilização do ambiente ISE 5.2 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 FPGA 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); porta série de 9 pinos; porta USB; LED's; DIP *switches*; botões; LCD de 2*16 caracteres; dois barramentos de expansão.

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 programa na linguagem VHDL que permite fazer os cálculos básicos apresentando-os no ecrã de um monitor VGA e no painel táctil [2], como pode ser visto na fig. 1.

Refira-se que todas as operações aqui descritas são efectuadas directamente no painel táctil.

Nas duas janelas principais ilustradas na fig. 1 (em cima, nos extremos), surgem os operandos, que são inteiros não negativos, com valores até 9. Podem ser alterados através dum controlo posicionado no topo de cada uma das janelas que incrementa de uma unidade o valor do operando respectivo.

A operação a realizar é seleccionada através de dois botões posicionados ao centro, um intitulado "ADD" e o outro "SUB", ambos auto-explicativos.

Após a escolha dos operandos e da operação a realizar, a selecção da tecla "Resultado", situado no canto inferior direito, faz com que o resultado surja no monitor VGA e simultaneamente no painel táctil.

Os dois periféricos (monitor e painel táctil) estão ligados às portas correspondentes (VGA e série, de 15 e 9 pinos, respectivamente) existentes na placa Trenz. Nesta placa existe um LCD de 2*16 caracteres, que está, neste caso, a mostrar em tempo real qual o caracter de controlo associado ao botão pressionado no painel táctil (ver fig. 1).

II . ESTRUTURA DO PROGRAMA EM VHDL

A. Estrutura básica do programa

O programa é constituído por quatro módulos principais (descritos em VHDL), um UCF (*User Constraints File*), um esquemático específico e outro global, de acordo com a fig. 2.

O UCF fpga.ucf indica o modo como os pinos devem ser atribuídos, assim como as definições a nível de sinais de relógio.

O esquemático clk_res.sch efectua as divisões necessárias no relógio inicial para este poder ser utilizado nos diversos componentes do programa. Devido à menor velocidade dos componentes utilizados (LCD, painel táctil), estas divisões são necessárias de modo aos resultados poderem ser percepcionados.

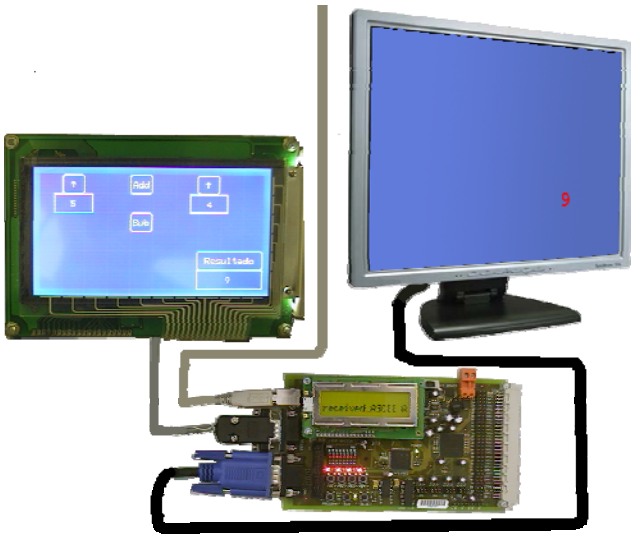

Fig. 1 – Ligação dos periféricos e demonstração de operação ADD

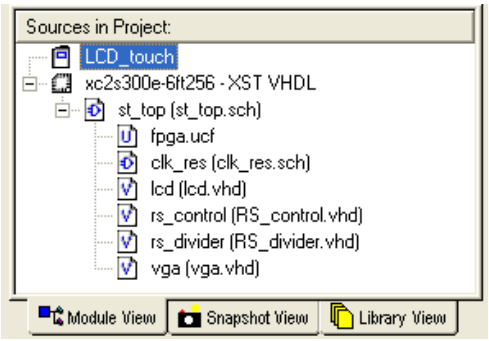

Fig. 2 – Módulos do programa

O módulo lcd.vhd revela como se efectuam as interacções com o LCD presente na placa TE-XC2SE.

O módulo rs_control.vhd é o bloco mais extenso, descrevendo todas as interacções com o painel táctil, bem como a passagem de dados deste para o LCD.

O módulo rs_divider.vhd é um módulo auxiliar, que tem como principal função fornecer um relógio secundário.

O módulo vga.vhd contém todas as funções associadas à saída para o monitor VGA.

III . CONTROLO DOS DISPOSITIVOS

A. Controlo do LCD da placa

Para o controlo do LCD, foi criado um ficheiro VHDL com todas as especificações, desde as constantes como a linha "received_ASCII" (ver fig. 1) até aos processos de controlo de entrada e saída.

B. Controlo do painel táctil

O manual de referência do painel táctil [3] inclui todas as funções suportadas, e é o principal auxílio no desenvolvimento deste módulo.

Aqui, foram usadas primitivas de desenho de rectângulos, definição personalizada de símbolos (setas ↑ e ↓), escrita de texto, sendo também feito um uso extensivo da primitiva que permite a definição de zonas de toque, que depois foram associadas a rectângulos, formando botões que são o meio de interacção com o utilizador.

Algumas das características e possibilidades deste dispositivo encontram-se documentadas em [4], [5].

C. Envio de sinais para o ecrã do monitor

Basicamente, o cinescópio do monitor envia raios catódicos para o ecrã fazendo um varrimento horizontal e vertical. É portanto necessário saber qual a posição exacta (x,y) do *pixel* que está a ser "iluminado" para assim definir qual a cor que se pretende que ele tome.

A resolução usada foi de 640x480 pixeis, sendo que o DAC tem a sua frequência originada no cristal de quartzo de frequência 25MHz, sendo, no entanto, possível utilizar, mediante algumas alterações no módulo respeitante ao monitor, resoluções mais elevadas, como 800x600 ou 1024x768, com uma taxa de varrimento de 60 Hz, em todos os casos (igualmente configurável).

Existe uma demonstração do uso de sinais VGA [6], que pode ser usada para perceber os aspectos primários desta interacção.

A comunicação dos dados lidos do painel táctil para o ecrã do monitor é feita através dum processo que é sensível aos sinais de relógio e de *reset*. As operações são efectuadas na transição descendente do relógio (*falling edge*), fazendo-se o teste condicional do símbolo lido do painel táctil, bem como de vários contadores e variáveis de controlo.

Segue-se um excerto do código desse processo:

```
entity RS_control is
     Port (clk : in  std_logic;
           rst : in  std_logic; [...]
end RS_control;
 [...]
 signal cont1 : std_logic_vector (3 downto 0);
 signal cont2 : std_logic_vector (3 downto 0);
 signal resultado: std_logic_vector (4 downto 0);
 [...]
 process(clk, rst)
 variable tmp, ind: integer;
 begin
 if rst= '1' then
       tmp:=0; ind :=0; cont1 <= "0000";
       cont2 <= "0000"; resultado <= "00000";
 elsif falling_edge(clk) then
```

```
          if rs232in = '0' then ind := 1;
          end if;
          if (tmp >= 1) then
             if (tmp <= 8) then
                LCD_symbol(tmp-1) <= rs232in;
             end if;
          end if;
        if ind = 1 then tmp := tmp + 1;
        end if;
        if (tmp >= 9) and (rs232in = '1') then
          tmp := 0; ind := 0;
        if LCD_symbol = x"2b" then
          cont1<=cont1+'1';
        elsif LCD_symbol = x"3e" then
        -- Símbolo ">"
        cont2<=cont2+'1';
 end process;
```

D. Escrita do resultado no ecrã

Usando funções de baixo nível, através de instruções repetitivas, é possível desenhar cada algarismo do resultado no ecrã [7].

Como a gama de resultados varia entre 0 e 18, visto o programa ter sido feito de modo ao resultado dar sempre positivo, e dada a entrada RESULT_IN de 5 bits, definiu-se exaustivamente a atribuição de valores a cada dígito. Caso a gama de resultados fosse mais alargada, justificar-se-ia o uso de uma ROM para efectuar a conversão para código BCD. A definição foi feita do seguinte modo:

```
case RESULT_IN is
 when "00000" => CodNum2<='0'; CodNum1<="0000";
 when "00001" => CodNum2<='0'; CodNum1<="0001";
 when "00010" => CodNum2<='0'; CodNum1<="0010";
 when "00011" => CodNum2<='0'; CodNum1<="0011";
 -- . . .
 when "01111" => CodNum2<='1'; CodNum1<="0101";
 when "10000" => CodNum2<='1'; CodNum1<="0110";
 when "10001" => CodNum2<='1'; CodNum1<="0111";
 when "10010" => CodNum2<='1'; CodNum1<="1000";
 when others => CodNum2<='0'; CodNum1<="0000";
end case;
```

A ressalva para quando o resultado sai fora da gama [0,18] (por exemplo, *underflows*); a saída no monitor indicará 0 nesses casos.

IV. INTERFACE AO NÍVEL DO SOFTWARE

A. Escrita de algarismos no ecrã

Para escrever algarismos no ecrã é necessário construir uma matriz para determinar a forma de cada um. Neste caso, foi construída uma matriz de 16x10 para cada algarismo, sendo cada um deles definido da mesma forma que se pode ver no exemplo, o número “3”, no seguinte código, com as definições explícitas:

```
architecture bhv of vga is
  subtype word10 is STD_LOGIC;
  type s_numero is array (0 to 159) of word10;
  [...]
begin [...]
constant num3 : s_numero := (
        '0','0','1','1','1','1','1','0','0','0',
        '0','1','1','1','1','1','1','1','0','0',
        '1','1','1','0','0','1','1','1','1','0',
        '1','1','0','0','0','0','1','1','1','0',
        '0','0','0','0','0','1','1','1','1','0',
        '0','0','1','1','1','1','1','1','1','0',
        '0','0','1','1','1','1','1','1','0','0',
        '0','0','1','1','1','1','1','1','0','0',
        '0','0','0','0','0','1','1','1','1','0',
        '0','0','0','0','0','0','1','1','1','0',
        '1','1','0','0','0','0','1','1','1','0',
        '1','1','1','0','0','1','1','1','1','0',
        '1','1','1','1','1','1','1','1','1','0',
        '0','1','1','1','1','1','1','1','0','0',
        '0','0','1','1','1','1','1','0','0','0',
        '0','0','0','0','0','0','0','0','0','0'
        );
[...]
```

No processo de escrita no ecrã, foram definidas duas variáveis, *var1* e *var2*, com a função de controlarem o posicionamento vertical e horizontal de cada um dos dois dígitos no ecrã:

```
var1 := (TO_INTEGER(hcnt)-412 + (TO_INTEGER(vcnt)-300)*10);
var2 := (TO_INTEGER(hcnt)-400 + (TO_INTEGER(vcnt)-300)*10);
```

A variável *var1* indica que o primeiro dígito tem a sua origem horizontal na coordenada x=412 e a origem vertical na coordenada y=300, sendo a multiplicação por 10 necessária de modo a se escrever todas as 10 colunas de cada dígito; de modo análogo com *var2*.

Quanto à escrita, propriamente dita, e visto que uma variável só podia assumir 10 valores e a outra 2, resumiu-se a 12 condições, em que, seguindo o varrimento do monitor, se escreve a vermelho caso a condição se verifique ou a cor de fundo (azul) em todos os outros casos.

As cores foram definidas usando seis parâmetros, três de cor (vr, vg, vb) e três de brilho (vbr, vbg, vbb), todos eles binários, resultando numa gama de 2^6 valores de cor possível, 64 cores. O valor mais escuro (preto) equivale às seis variáveis a 0 e o mais claro (branco) equivale às mesmas a 1. Existem outros 2 bits no sinal VGA, para as sincronizações horizontal e vertical, mas que não influem na gama de cores.

Resumidamente, temos,

```
 if (hcnt>=412 and hcnt <= 421 and vcnt>=300 and vcnt
<=315) then
   case CodNum1 is -- números da direita
when "0000" => if num0(var1)='1' then vr<='0';vg<=
'0';vb<='0';vbr<='1';vbg<='0';vbb<= '0';
        else vr<= '0'; vg<= '0'; vb<= '1'; vbr<= '0';
        vbg<= '0'; vbb<= '1'; end if;
```

```
 when "0001" => if num1(var1)='1' then vr<='0';vg<=
'0';vb<='0';vbr<='1';vbg<='0';vbb<='0';

        else vr<= '0'; vg<= '0'; vb<= '1'; vbr<= '0';
        vbg<= '0'; vbb<= '1'; end if;

 ... [análogo  para os restantes algarismos]

 elsif (hcnt>=400 and hcnt <= 409 and vcnt>=300 and
vcnt <=315)   then if CodNum2 = '1' and num1(var2)='1'
then   vr<= '0'; vg<= '0'; vb<= '0'; vbr<= '1'; vbg<=
'0'; vbb<= '0';

      else vr<= '0'; vg<='0'; vb<='1'; vbr<='0'; vbg<=
'0'; vbb<= '1';

      end if;

 else     vr<= '0'; vg<= '0'; vb<= '1'; vbr<= '0';
vbg<= '0'; vbb<= '1';

 end if;
```

B. Processamento dos dados

Para o cálculo dos resultados, existe uma variável (*op* com o tamanho de 1 bit) que é actualizada quando o utilizador selecciona no painel táctil a operação. Dependendo da operação pretendida, a variável *op* toma os seguintes valores:

- 0 – Subtracção
- 1 – Adição

Fazendo um teste à variável *op*, é escolhida a operação a efectuar com os operandos.

No caso da subtracção, o programa foi implementado de forma o resultado ser sempre positivo, sendo que a subtracção é sempre efectuada da forma *operando1* – *operando2*. Quando o *operando2* é maior que o *operando1* e o resultado seria negativo, é apresentado o valor '0'. Assim, na subtracção, o resultado varia entre 0 e 9.

A operação de adição, foi efectuada da forma usual, podendo o resultado variar entre 0 e 18.

Ambas as operações são apresentadas de seguida:

```
...
elsif (LCD_symbol = x"52") then -- RESULTADO (R)
  if cont1= "00000" then resultado(3 downto 0)<=cont2;
  elsif cont2 = "00000" then resultado(3 downto 0)<=cont1;
  elsif op='1' then
          resultado(3 downto 0) <= ((cont1-'1')+(cont2-'1'));
  elsif op='0' then
          resultado(3 downto 0) <= ((cont1-'1') - (cont2-'1'));
  end if;
...
```

Não existe qualquer protecção para o facto de os operandos poderem ultrapassar o valor 9, não sendo o valor mostrado no painel táctil. No entanto, o resultado continua salvaguardado, de acordo com as convenções descritas.

V. Compilação de VHDL para o bitstream

O programas em VHDL têm de ser sintetizados no Xilinx ISE [8] para a criação do *bitstream*, o qual é enviado directamente para a FPGA.

O esquemático completo do circuito apresentado consta na figura 3.

VI. Conclusões

Deste artigo, pode-se verificar que é relativamente fácil a implementação de circuitos simples numa FPGA utilizando a linguagem VHDL, apesar de projectos mais complexos exigirem bastante tempo e cuidado na sua concepção, sendo preferível optar por uma linguagem de mais alto nível, como por exemplo, a linguagem Handel-C, prioritizando-se o desenvolvimento de funções avançadas em detrimento de tentar resolver problemas de baixo nível.

Os objectivos primários, que se centravam primordialmente no uso e interacção do painel táctil e do monitor VGA, foram plenamente atingidos. Há, no entanto, espaço para melhoramentos a diversos níveis, desde o aumento do tamanho dos operandos, e a diversidade de operações disponível, passando por aspectos de interacção no painel táctil. A diminuição do tempo de compilação através da optimização de código também é possível e conveniente.

O projecto que inclui todas as partes apresentadas acima está disponível na WebCT [9]. Adicionalmente, tutoriais de iniciação ao VHDL e outros exemplos estão também disponíveis.

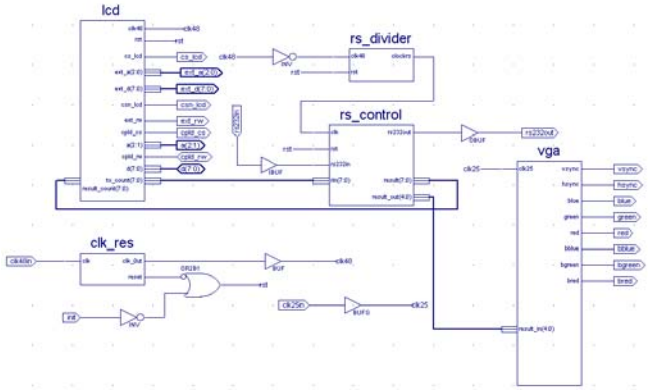

Fig. 3 – Esquemático completo do projecto

Agradecimentos

O autor agradece ao Professor Valery Sklyarov e a Iouliia Skliarova pela ajuda prestada na elaboração deste artigo.

Referências

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

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

[3] EA KIT240-7 Reference Manual, Electronic Assembly, 2000.

[4] B. Pereira, "Desenvolvimento de um circuito para operações sobre vectores booleanos e ternários", E&T, vol. 3, n.º 9, pp. 141-144,Set. 2003.

[5] R. Costa, J. Limas, I. Oliveira, "Desenvolvimento de uma calculadora baseada numa FPGA e num *touch panel*", E&T, vol. 3, n. 9, pp. 145-150, Set. 2003.

[6] www.trenz-electronic.de, "TE-XC2Se Demo Overview".

[7] "Xilinx 5 Software Manuals", Xilinx Inc, 2002.

[8] ISE 5.2, Xilinx series FPGA. Disponível em : http://www.xilinx.com/

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