Desenvolvimento de uma biblioteca em VHDL para manuseamento de dados a partir de uma porta de série

André Monteiro

Resumo **- Este artigo descreve uma biblioteca desenvolvida em VHDL para uma FPGA (*Field Programmable Gate Array*) que permite recepcionar dados a partir da sua porta de série RS232, guardá-los na sua memória interna e visualizá-los simultaneamente num monitor VGA ligado à placa FPGA e no LCD da própria placa. Para enviar os dados pela porta de série é utilizado o PC, tendo o desenvolvimento da biblioteca sido feito em VHDL. Este artigo ilustra o funcionamento básico das entradas e saídas da placa Spartan IIE XC2S300E, assim como a comunicação de dados em série e a sua difusão num periférico de saída, neste caso o monitor VGA.**

Abstract **– This paper describes a VHDL-based library which allows to input data from serial port RS232, to save in internal memory and to visualize simultaneously data on a VGA monitor connected and LCD. The paper ilustrates the basic functionality of the input/output ports of the FPGA Spartan IIE XC2S300E, serial data comunication and data exchange with periphal devices.**

I . INTRODUÇÃO

A reconfiguração de *hardware* é uma área de particular interesse nos dias de hoje, não só pelo seu potencial mas pela variedade de diferentes soluções para os seus problemas. Deste modo, o seu enquadramento numa disciplina do curso revela-se bastante importante para uma formação inicial e suporte para novos desenvolvimentos.

Este projecto foi realizado na sequência da disciplina Computação Reconfigurável leccionada em Licenciatura em Engenharia de Computadores e Telemática, no 4.º ano, 2.º semestre, pelo Professor Valery Sklyarov [1].

O material 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 [2].

A. Objectivos

Apesar de subentendidos no título do artigo, os objectivos pretendidos com a execução deste trabalho podem-se resumir nos seguintes aspectos:

- adquirir experiência na elaboração de soluções práticas;
- conjugar a programação em VHDL com as limitações do hardware;
- lidar com o funcionamento das entradas e saídas da placa Trenz [3], mais especificamente a entrada RS232 e a saída VGA;
- utilizar e manusear o LCD incluído na placa Trenz [3];
- integrar a realidade do *hardware* com a criação e utilização de uma RAM na memória interna da FPGA.

B. Descrição global do trabalho

O projecto consiste no desenvolvimento na linguagem VHDL de um circuito que permite, num a primeira fase, a recepção de dados a partir da porta de série da placa Trenz. Após a recepção de dados, o caracter recebido é directamente enviado para o LCD, sob a forma de caracter de 8 bit. Simultaneamente, o caracter é guardado na memória Single Port Block RAM de 32 bit, da qual é retirado com um contador sucessivamente o valor. Este valor é convertido em caracter gráfico para o VGA através de uma ROM estática de caracteres. A ROM define números de 0 a 9 e letras de A a F. Caso não seja efectuada nenhuma recepção, é exibido o número 0. A saída VGA está estabelecida para uma resolução de 640 x 480 pixéis, e a recepção de dados é efectuada a uma taxa de 115200 baud.

O utilizador tem que usar um software adicional num PC para enviar dados para a placa, sendo esta a única intervenção que tem que realizar.

A disposição estrutural do hardware é elucidada pela figura 1.

II . ORGANIZAÇÃO ESTRUTURAL DO CÓDIGO EM VHDL

A. Estrutura básica do código

O código é constituído por seis módulos principais, descritos em VHDL, um ficheiro UCF (*User Constraints File*), um esquemático específico e outro global.

O UCF fpga.ucf indica a atribuição dos pinos existentes, assim como as definições a nível de sinais de relógio. Estão definidos dois relógios: um de 48MHz e outro de 25MHz específico para o monitor VGA.

O esquemático clk_res.sch efectua no relógio inicial as divisões necessárias para este poder ser utilizado nos vários componentes do programa. Estas divisões são

indispensáveis devido à menor velocidade dos componentes utilizados, de modo a serem visíveis os resultados.

O char_rom.vhdl é o módulo no qual está definida uma pequena ROM de demonstração que contém apenas 15 caracteres gráficos, cada um definido em 16 x 16 bit.

O lcd.vhd é o módulo que permite visualizar no LCD o caracter introduzido.

O RS_divider é o módulo onde se faz a divisão do relógio de modo a proporcionar uma taxa de recepção de 115200 baud na porta de série.

O RS_control é o módulo onde se efectua a recepção de dados, armazenando-os de 8 em 8 bit em caracteres.

O spblockram.vhd é um módulo que define uma RAM de 32 byte localizada na memória interna e que armazena os caracteres.

O vga.vhd é o módulo que controla o monitor, recebendo o caracter da RAM e enviando para o monitor o respectivo gráfico.

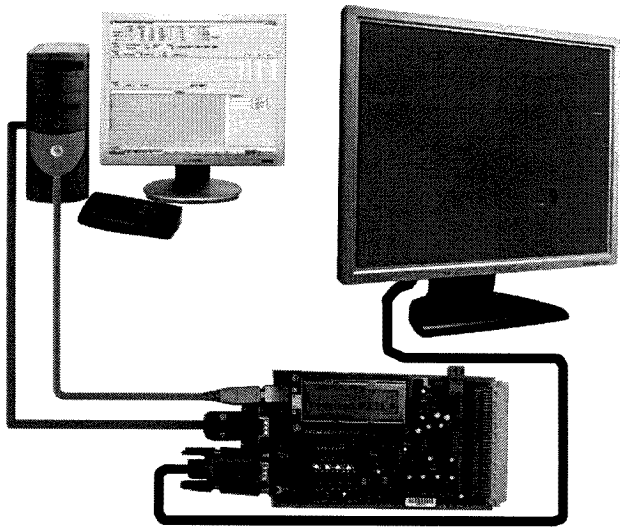

Fig. 1 – Disposição dos periféricos e ligações realizadas

III . ESTRATÉGIA DE SUPORTE

A. Recepção de dados através da porta série

Para a recepção de dados através da porta de série RS232, é necessário um cabo de série directo. O cruzamento dos pinos é necessário para se estabelecerem dois canais de transmissão unidireccionais (placa RxD – PC TxD; placa TxD – PC RxD). Como este cruzamente é feito na FPGA, não é necessário o uso de cabo cruzado.

A comunicação entre o PC e a placa de desenvolvimento é efectuada através de uma adaptação do protocolo standard RS232, já que apenas são utilizados dois pinos, RxD e TxD.

Para corrigir erros de transmissão em comunicações série é utilizado o método da comunicação assíncrona, em que são inseridas marcas no *stream* de bits a enviar. Deste modo, a transmissão tem que ser efectuada à mesma velocidade. É iniciada a sequência com o envio de um bit designado por *start bit* seguido de uma *stream* de 8 bits de dados. O bit menos significativo é enviado no início e o mais significativo no fim da *stream*. Por fim, segue-se o *stop bit* para indicar o fim da sequência. A figura 2 [4] ilustra esse aspecto.

Fig. 2 – Formato dos dados na comunicação rs232

Desta forma, a recepção de dados é efectuada sequencialmente, com se pode constatar no excerto transcrito.

```
process(clk, rst)

variable tmp, ind: integer;

begin
  if rst= '1' then
     tmp:=0; ind :=0;
  elsif falling_edge(clk) then
     if rs232in = '0' then ind := 1;
     end if;
   if (tmp >= 1) then
      if (tmp <= 8) then 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;
    end if;
  end if;
end process;
```

B. Controlo do LCD da placa

Para o controlo do LCD, existe um ficheiro VHDL que especifica a recepção de dados. Este ficheiro contém todos os procedimentos a efectuar sobre o CPLD e especifica também a saída do caracter para o LCD de 2 x 16 caracteres, no qual as duas linhas são preenchidas por constantes, exceptuando o caracter, que varia conforme a entrada. O código VHDL desenvolvido aqui é bastante similar ao [4] e [5].

C. Registo dos dados na RAM

A RAM utilizada é uma Single Port Block RAM, que inclui um relógio, um Write Enable, um endereço, uma entrada de dados e uma saída de dados. Quando o relógio se encontra numa transição ascendente e com o Write Enable activado, é armazenado o caracter recebido. O funcionamento está implícito no código apresentado:

```
architecture syn of spblockram is

 type ram_type is array (31 downto 0) of
std_logic_vector (7 downto 0);
 signal RAM : ram_type;
 signal read_a : std_logic_vector(4 downto 0);
```

```
begin
process (clk)
begin
  if (clk'event and clk = '1') then
       if (we = '1') then
          RAM(conv_integer(a)) <= di;
       end if;
       read_a <= a;
  end if;
end process;

do <= RAM(conv_integer(read_a));

end syn;
```

Existe um contador para os endereços da RAM no módulo do monitor VGA, que permite retirar da memória o caracter correcto. Esta memória permite armazenar 32 caracteres antes de ser reescrita pois uma vez chegado a 32 valores, o contador é reiniciado a zero. O código pode ser visto de seguida:

```
-- RAM address
process(clk25, rst)
  begin
       if rst= '1' then
        RAM_count <= "00000";
       elsif rising_edge(clk25)
         then   if   RAM_count   ="11111"   then
RAM_count <="00000";
       else RAM_count <= RAM_count + 1;
           end if;
       end if;
    RAM_address <= RAM_count;
  end process;

-- RAM data to respective rom address
process(result, rst)
  begin

    if rst = '1' then count <= "0000";
    else
     case result is
      when "00110000" = >count <= "0000";   -- 0
      when "00110001" => count <= "0001";   -- 1
      when "00110010" => count <= "0010";   -- 2
      when "00110011" => count <= "0011";   -- 3
      when "00110100" => count <= "0100";   -- 4
      when "00110101" => count <= "0101";   -- 5
      when "00110110" => count <= "0110";   -- 6
      when "00110111" => count <= "0111";   -- 7
      when "00111000" => count <= "1000";   -- 8
      when "00111001" => count <= "1001";   -- 9
      when "01000001" => count <= "1010";   -- A
      when "01000010" => count <= "1011";   -- B
      when "01000011" => count <= "1100";   -- C
      when "01000100" => count <= "1101";   -- D
      when "01000101" => count <= "1110";   -- E
      when "01000110" => count <= "1111";   -- F
      when others => count <= "0000";
     end case;
    end if;
end process;
```

D. Envio de caracteres para o monitor VGA

Como já foi referido, resolução usada foi de 640x480 pixéis, que serve para ilustrar o funcionamento do projecto. Também é possível utilizar resoluções mais elevadas, com uma taxa de varrimento de 60 Hz, mediante algumas alterações no módulo respeitante ao monitor.

A apresentação de caracteres no monitor requer uma transformação, já que o monitor apenas apresenta pixéis. Desta forma, é necessário converter cada caracter num conjunto de pixéis que simbolizem o caracter pretendido. Esta conversão é feita pelo módulo char_ROM, para o qual é enviado o caracter e devolvido o gráfico correspondente. A impressão no monitor é efectuada posteriormente e na posição escolhida. A posição inicial do caracter é escolhida com o referencial de linhas e colunas. Neste caso e nesta resolução a coluna 12 e a linha 14 indicam aproximadamente o centro do ecrã.

Também é possível definir uma cor para o caracter. Dentro das seis saídas (R, G, B, RB, GB, BB) do módulo basta definir uma combinação de cores. O excerto de código em baixo refere essa mesma potencialidade:

```
process(clk25)
 begin
       blank  <= h_blank or v_blank;

       if blank ='1' then
              r  <= '0';
              rb <= '0';
              g  <= '0';
              gb <= '0';
              b  <= '0';
              bb <= '0';
       else
              if (text_col < 15) then
                     r  <= pixel;
                     rb <= pixel;
                     g  <= '1';
                     gb <= '1';
                     b  <= '0';
                     bb <= '0';
              else
                     r  <= '1';
                     rb <= '1';
                     g  <= pixel;
                     gb <= pixel;
                     b  <= '0';
                     bb <= '0';
              end if;
       end if;
end process;
```

IV. INTERFACE AO NÍVEL DO SOFTWARE

A. Envio de dados através do software Terminal v1.9b

O programa Terminal [6] tem um funcionamento simples. Uma vez ligados os cabos, é necessário escolher a taxa de 115200 baud, 8 bit de dados, sem paridade, 1 stop bit, sem *handshake* e efectuar o "Connect". De seguida basta escrever o caracter pretendido, dentro da gama disponível, e carregar em "Send", como ilustra a figura 3. O caracter é enviado em 8 bit e exibido no LCD e monitor VGA simultaneamente.

Quando se efectua a ligação, é recebido o caracter "0", pois a saída rs232out foi definida estaticamente para esse valor. O projecto apenas está concebido para receber um caracter de cada vez, pelo que é desnecessário enviar mais que um, pois apenas o último é recepcionado. Qualquer

caracter fora da gama referida irá proporcionar um 0 no monitor VGA.

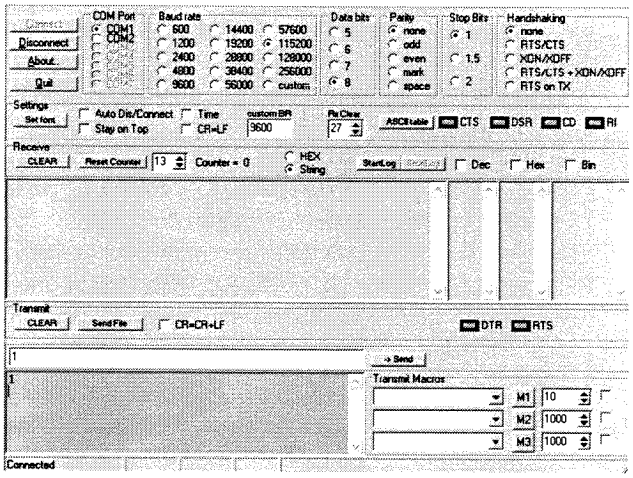

Fig. 3 – *Software* Terminal v1.9b: envio de dados pela porta série

B. Compilação VHDL e geração de bitstream

O código escrito que foi desenvolvido em VHDL tem que ser processado e sintetizado no Xilinx ISE [2] antes para poder ser utilizado na placa Trenz. Assim, é criado o *bitstream*, que através do programa TEProg.exe fornecido pela Trenz Electronics [3] é enviado directamente para a FPGA. O esquemático completo do circuito que apresenta a estrutura final do projecto pode ser visto na figura 4.

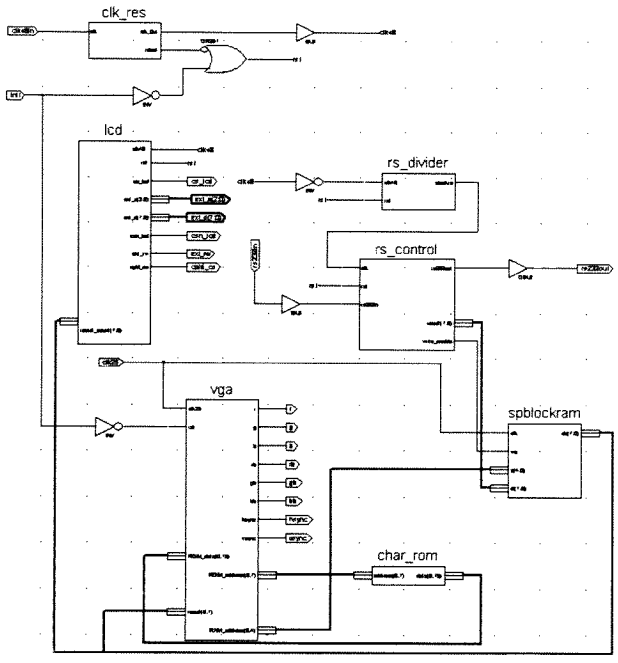

Fig. 4 – Esquemático completo do projecto

V. CONCLUSÕES

A realização deste trabalho prático tinha como ponto chave a aquisição de conhecimentos e princípios de funcionamento de um sistema de hardware reconfigurável.

Mais especificamente, foi estudada a interacção entre os vários componentes existentes e as potencialidades da placa Trenz [3], aliada ao desenvolvimento de código na linguagem VHDL.

As estratégias utilizadas no desenvolvimento deste projecto foram abordadas e clarificadas.

Uma das prioridades foi a optimização do funcionamento do programa, evitando operações demasiado complexas e reutilizando expressões para a minimização da utilização de recursos da FPGA.

O processo de comunicação RS232 foi o mais complicado, não pela dificuldade subjacente mas pelo facto de se difícil encontrar *software* ideal à realização do projecto, uma vez que nos dias de hoje a norma não é muito usada.

Um factor de sucesso foi a organização do projecto em 3 fases. A primeira fase consistiu em receber os dados da porta de série e mostrá-los no LCD da placa. A fase seguinte teve como objectivo visualizar esses mesmos dados e exibi-los no monitor VGA. Por fim, a última fase foi a integração de uma RAM no circuito já construído e funcional.

Ainda que os conhecimentos tivessem que ser aprofundados relativamente aos adquiridos na disciplina do âmbito do projecto, a aprendizagem foi progressiva e sustentada nas referências indicadas, revelando-se proveitosa. Todas as especificações do projecto foram cumpridas e de uma forma optimizada, pelo que o projecto foi bastante bom para aumentar o conhecimento e experiência em sistemas reconfiguráveis e interagir com a placa Spartan IIE da Trenz [3].

O projecto que inclui todas as partes apresentadas acima está disponível na WebCT [1].

AGRADECIMENTOS

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

REFERÊNCIAS

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

[2] http://www.xilinx.com, ISE 5.2, Xilinx series FPGA.

[3] http://www.trenz-electronic.com.

[4] R. Costa, J. Limas, I. Oliveira, "Desenvolvimento de uma calculadora baseada numa FPGA e num *touch panel*", Electrónica e Telecomunicações, vol. 3, n. 9, Setembro 2003.

[5] B. Monteiro, "Desenvolvimento de um circuito aritmético em VHDL", Electrónica e Telecomunicações, Janeiro 2004.

[6] http://bray.velenje.cx/avr/terminal, Terminal v1.9b