Implementação de um projecto em VHDL para a execução de várias operações sobre um vector booleano

Rui Santos

Resumo – **Este artigo descreve a estratégia seguida no desenvolvimento de um projecto realizado no âmbito da disciplina de Computação Reconfigurável, do curso de LECT (Licenciatura em Engenharia de Computadores e Telemática).**

O projecto consistiu na especificação de um circuito capaz de executar várias operações sobre vectores booleanos (a ser implementado numa FPGA), através de uma linguagem de descrição de hardware, VHDL. Para este efeito, foi utilizado o ambiente Xilinx ISE 6.x e a placa TE-XC2Se da Trenz Electronic com uma FPGA XC2S300E-6-FT256 da família Spartan-IIE e o programa ModelSim v5.7c da Model Technology.

Abstract – **This paper describes a strategy adopted to accomplish a project within the Reconfigurable Computing discipline taught to Computer and Telematics Engineering undergraduate curriculum in the second semester of 2004/2005.**

The goal of the project was to design, with the aid of VHDL hardware description language, and to implement, in an FPGA, a reconfigurable circuit for executing different operations over Boolean vectors. The following software and hardware tools were used: Xilinx Integrated Software Environment 6.x, TE-XC2Se reconfigurable board of Trenz Electronic containing one XC2S300E-6-FT256 FPGA of Xilinx Spartan-IIE family and starter version 5.7c of ModelSim from Model Technology.

I. INTRODUÇÃO

O problema proposto incidia na especificação de um circuito em VHDL, que disponibilizasse um conjunto de operações sobre um vector booleano. Operações essas passíveis de serem usadas por um utilizador.

Toda a interface com o utilizador seria realizada através do monitor VGA e dos botões B1 a B4 disponíveis na placa.

A. Objectivos

O circuito a desenvolver em VHDL requeria assim a concretização dos seguintes objectivos:

- Permitir a definição e alteração de um vector booleano, utilizando os botões B1, B2 e B3 da placa (ver figuras 1 e 2);
- Disponibilizar um conjunto de operações que produziriam um resultado específico com base nesse mesmo vector booleano;
- Permitir a selecção da operação desejada em cada momento com base no botão B4 da placa (ver figuras 1 e 2);
- Mostrar no monitor VGA todas as acções realizadas.

B. Material e ferramentas utilizadas para o desenvolvimento deste projecto

B1. Placa TE-XC2SE da Trenz Electronic

Para a realização deste projecto recorreu-se à utilização da placa TE-XC2SE da Trenz Electronic (ver figura 1) [1] que tem incorporada uma FPGA XC2S300E-6-FT256. Esta placa possui dois conectores de expansão, um com 26 pinos, alguns dos quais partilhados com o *display* LCD de 2×16 caracteres e o outro conector do tipo VG96 com 96 pinos. A placa utilizada inclui ainda dois osciladores que geram sinais de relógio de 48 MHz e 25 MHz (utilizado neste projecto). Os dados do utilizador podem ser visualizados no *display* LCD e em 5 LEDs individuais, 4 destes ligados ao CPLD e o restante controlável directamente através do pino C15 da FPGA. O CPLD permite também receber os dados de 4 botões (também utilizados no desenvolvimento deste projecto) e 8 interruptores. A placa pode controlar a porta série RS232 e comunicar com um monitor VGA. A porta USB fornece alimentação para a placa, quando o respectivo interruptor estiver ligado, e, para além disso, permite a programação da FPGA carregando o *bitstream* na memória *Flash* da placa, com a ajuda da ferramenta TEprog disponibilizada pela Trenz Electronic.

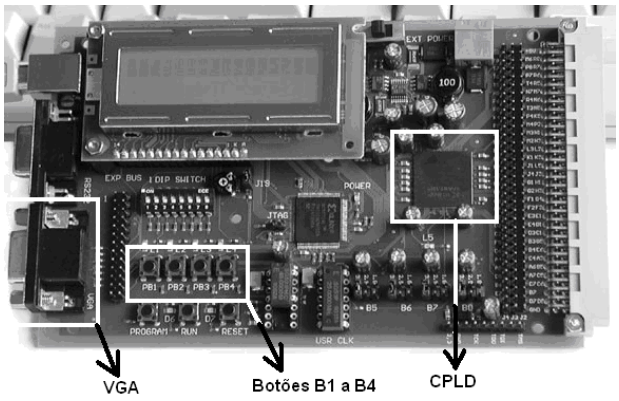

Fig. 1 — Placa TE-XC2SE da Trenz Electronic

B2. Ferramenta ISE 6.x da Xilinx

Esta ferramenta possibilita o desenvolvimento e a implementação de circuitos digitais, com base em FPGAs, disponibilizando para isso a possível realização dos seguintes passos entre outros [2]:

- Projecto baseado em VHDL;
- Projecto esquemático;
- Simulação em *ModelSim.*

B3. Monitor VGA

O monitor VGA utilizado foi um monitor perfeitamente comum de 17".

II. ESPECIFICAÇÃO

Para permitir a interacção do utilizador com as funcionalidades disponíveis, são utilizados os botões B1 a B4 da placa (ver figura 2), o que possibilita que todas as acções realizadas sobre estes se reflictam no monitor VGA [3, 4].

As operações que o utilizador pode realizar, produzindo um resultado baseado no vector booleano, são as seguintes:

- Encontrar N "uns" consecutivos no vector booleano;
- Contar o número de "uns" presentes no vector booleano;
- Calcular a soma de N bits + N bits do vector booleano, com N a variar de 0 a 10.

Deste modo, os botões B1 e B2 permitem movimentar o cursor de modificação entre os bits do vector booleano e o número N de bits. Por sua vez, o botão B3 permite definir o vector booleano, bem como o número N de bits enquanto que o botão B4 serve para seleccionar a operação activa em cada momento (ver interacção presente na figura 3).

De forma a possibilitar um fácil e rápido desenvolvimento do circuito proposto, este foi dividido em várias partes, que por sua vez deram origem a algumas entidades que facilitam agora uma melhor interpretação da ideia seguida.

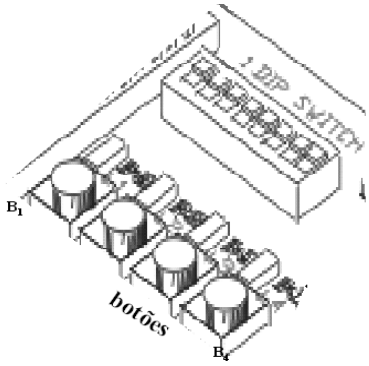

Fig. 2 — Botões B1..B4 da placa TE-XC2SE

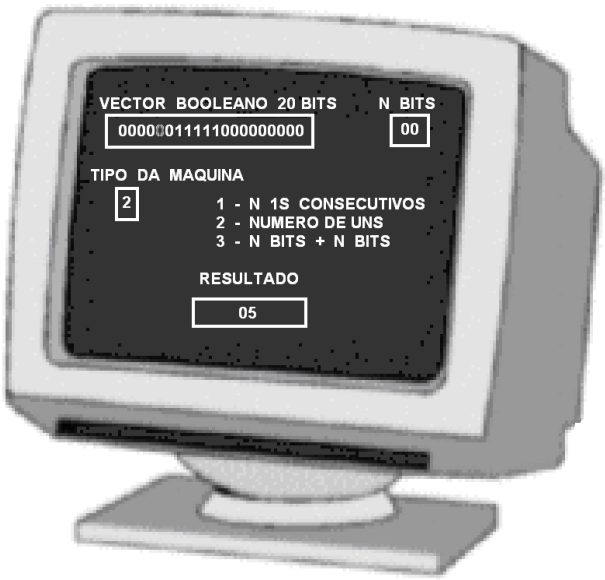

Fig. 3 — Resultado final (exemplo)

A. Descrição das entidades que constituem o projecto

As entidades que constituem o projecto são as seguintes (ver figura 4):

- Entidade *cpld_trabalho*: esta entidade permite a interacção do utilizador com os botões B1 a B4 da placa através do CPLD. Os sinais indicando o estado (pressionado ou não pressionado) dos botões são enviados à entidade *deboucing_trabalho*;
- Entidade *deboucing_trabalho*: esta entidade permite fazer o *debouncing* dos botões, isto é, permite eliminar as oscilações que causam indefinição no estado do botão quando pressionado.
- Entidade *memory_trabalho*: esta entidade armazena num bloco de memória RAM embutida na FPGA todos os caracteres estáticos que serão mostrados no monitor VGA. Entende-se por caracteres estáticos todos aqueles que no monitor VGA não são passíveis de ser alterados pelo utilizador.
- Entidade *symbol_rom_trabalho*: esta entidade permite efectuar a tradução de um algarismo pelos dados de um caracter. Estes dados permitem que o caracter correspondente fosse desenhado no monitor VGA.
- Entidade *vga_trabalho*: nesta entidade procede-se à apresentação dos caracteres estáticos e dinâmicos no monitor VGA. Esta entidade encarrega-se ainda de efectuar o cálculo e apresentação do resultado correspondente à tarefa seleccionada.

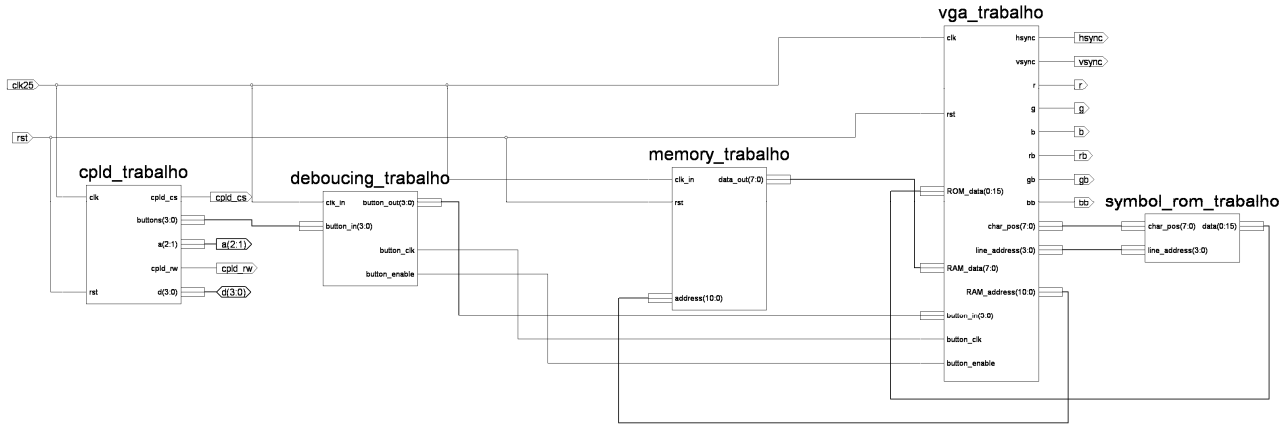

Fig. 4 — Interligação das entidades pertencentes ao projecto

III. ESTRATÉGIA SEGUIDA

O cálculo do resultado das três tarefas propostas no enunciado do projecto é realizado na entidade *vga_trabalho*.

O resultado da primeira tarefa, que consiste em verificar existência de N "uns" consecutivos no vector booleano, é calculado de forma sequencial através de um processo pertencente à entidade mencionada atrás. É apresentado a seguir o código em VHDL referente a esta primeira tarefa.

```
.....
-- entidade
entity vga_trabalho is
   -- portas relevantes para o processo em questão
   port (
      clk : in std_logic;
      rst : in std_logic;
      .....
         );
end vga_trabalho;

-- arquitectura
architecture bhv of vga_trabalho is
   .....
   -- vector booleano sobre o qual se vai realizar a operação
   signal vector_booleano : std_logic_vector(0 to 19);
   -- número de uns consecutivos a procurar
   signal nbits : natural range 0 to 10;
   -- resultado -  resposta positiva (sim)  ou  negativa (não)
   -- apresentada graficamente
   signal func1_resultado : type_array;
   -- índice – usado para percorrer todas as posições do vector
   signal func1_index : natural range 0 to 21;
   -- sinal auxiliar para determinar o resultado
   signal func1_result_temp : std_logic;
   -- contador – armazena o número de uns consecutivos existentes
   signal func1_cont : natural range 0 to 20;
   .....
begin
   .....
   -- processo em questão
   process(rst, clk)
      -- variável auxiliar
      variable aux : natural range 0 to 20;
begin
  -- se ocorrer um reset
  if rst = '0' then
     func1_index <= 0;
     func1_result_temp <= '0';
     -- resultado negativo (não)
     func1_resultado <= (10,10,10,10,10,30,31,10,10,10,10);
     func1_cont <= 0;
  -- na transição ascendente do relógio
  elsif rising_edge(clk) then
     if func1_index = 21 then
        if func1_result_temp = '1' then
           -- resultado positivo (sim)
           func1_resultado <= (10,10,10,10,41,21,35,10,10,10,10);
        else
           -- resultado negativo (não)
           func1_resultado <= (10,10,10,10,10,30,31,10,10,10,10);
        end if;
        func1_index <= 0;
        func1_cont <= 0;
        func1_result_temp <= '0';
     elsif func1_index = 20 then
        if func1_cont = nbits then
           func1_result_temp <= '1';
        end if;
        func1_index <= func1_index + 1;
     else
        -- se for um bit igual a '1'
        if vector_booleano(func1_index) = '1' then
           aux := func1_cont;
           aux := aux + 1;
           if aux = nbits then
              func1_result_temp <= '1';
           end if;
           func1_cont <= aux;
        else
           aux := 0;
           if aux = nbits then
              func1_result_temp <= '1';
           end if;
           func1_cont <= aux;
```

```
        end if;
        func1_index <= func1_index + 1;
      end if;
    end if;
  end process;
  .....
end bhv;
```

Para obtenção do resultado da segunda tarefa, que tem como objectivo calcular o número de "uns" no vector booleano, é utilizado um procedimento auxiliar. O código VHDL referente a esta tarefa é apresentado de seguida.

```
  -- procedimento que recebe como entrada um algarismo de 0 a
  -- 15 e retorna o número de  uns que constituem  esse algarismo em
  -- binário
  procedure conta_aux(signal rin : in std_logic_vector;
                      signal resultado : out natural range 0 to 4) is
   type type_rom is array (0 to 15) of natural range 0 to 4;
   constant trad_num : type_rom := (0 => 0, 1 => 1, 2 => 1,
                      3 => 2, 4 => 1, 5 => 2, 6 => 2, 7 => 3,
                      8 => 1, 9 => 2, 10 => 2, 11 => 3, 12 => 2,
                      13 => 3, 14 => 3, 15 => 4);
 begin
   resultado <= trad_num(conv_integer(rin));
end procedure conta_aux;
....
-- entidade.
entity vga_trabalho is
   -- nenhum porto relevante para esta tarefa
   port(
      .....
       );
end vga_trabalho;

-- arquitectura
architecture bhv of vga_trabalho is
   .....
   -- vector booleano
   signal vector_booleano : std_logic_vector(0 to 19);
   -- sinais auxiliares
   signal temp1, temp2, temp3,
          temp4, temp5 : natural range 0 to 4;
   .....
begin
   .....
   -- divisão do vector booleano em 5  partes de 4 bits cada
   conta_aux(vector_booleano(0 to 3),temp1);
   conta_aux(vector_booleano(4 to 7),temp2);
   conta_aux(vector_booleano(8 to 11),temp3);
   conta_aux(vector_booleano(12 to 15), temp4);
   conta_aux(vector_booleano(16 to 19), temp5);
   -- soma do resultado de cada uma das 5 partes
   temp6 <= temp1 + temp2 + temp3 + temp4 + temp5;
   .....
end bhv;
```

O resultado da terceira tarefa que consiste na soma de N bits com N bits é obtido através da soma directa dos bits pretendidos. Este resultado é apresentado em 11 bits pois a variável N varia entre 0 e 10 bits. O código VHDL seguinte corresponde a esta tarefa.

```
-- entidade
entity vga_trabalho is
   -- nenhuma porta relevante para esta tarefa
   port(
      .....
       );
end vga_trabalho;

-- arquitectura
architecture bhv of vga_trabalho is
   .....
   -- vector booleano e vector auxiliar
   signal vector_booleano : std_logic_vector(0 to 19);
   signal vector_booleano_auxiliar : std_logic_vector(19 downto 0);
   -- N bits e resultado da soma
   signal nbits : natural range 0 to 10;
   signal func3_resultado : std_logic_vector(10 downto 0);
   .....
begin
   .....
   vector_booleano_auxiliar(19 downto 0) <=
              vector_booleano(0 to 19);
   func3_resultado <= "00000000000" when nbits = 0 else
              vector_booleano_auxiliar(nbits-1 downto 0) +
              vector_booleano_auxiliar((2*nbits)-1 downto nbits);
   .....
end bhv:
```

IV. CONCLUSÃO

Neste artigo é descrito o material, as ferramentas e a abordagem utilizada para a concretização com sucesso do projecto apresentado na disciplina de Computação Reconfigurável.

Os objectivos propostos foram plenamente atingidos, podendo certos pontos ter abordagens diferentes, ponderando talvez um pouco mais em termos de questões de eficiência.

Este projecto proporcionou um contacto mais próximo com a linguagem VHDL, onde foi necessário aplicar os conhecimentos adquiridos durante as aulas teóricas, bem como outros conhecimentos adquiridos a título individual.

V. AGRADECIMENTOS

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

REFERÊNCIAS

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

[2] Xilinx Web Site, http://www.xilinx.com.

[3] Página http://www.ieeta.pt/~iouliia/Courses/CR/index.html, a disciplina "Computação Reconfigurável".

[4] Sklyarov, V. e Skliarova, I., Ferramentas para desenvolvimento de sistemas digitais reconfiguráveis, Electrónica e Telecomunicações, Vol. 3, Nº 8, Jan. 2003, pp.743-764.