Desenvolvimento de uma calculadora baseada numa FPGA e num touch panel

Rui Costa, João Limas, Inês Oliveira

Resumo - **Este artigo descreve uma calculadora baseada numa FPGA (*Field Programmable Gate Array*). Esta calculadora realiza quatro operações aritméticas (+, -, *, /) com três operandos de dois algarismos cada, e interage com um *touch panel* ligado à FPGA. A especificação da calculadora foi realizada em VHDL e, para a sua implementação, fez-se uso do ambiente *Xilinx ISE 5.1* [1,2]. Foi utilizada a placa TE-XC2Se da *Trenz Electronic*, baseada na FPGA** XC2S300E **da família *Spartan-IIE* da *Xilinx*. As principais vantagens desta FPGA são o baixo custo e a arquitectura bem definida de componentes primários. Esta FPGA possui uma arquitectura semelhante à das FPGAs da família *Virtex*.**

Abstract - **This paper describes an FPGA-based calculator, which executes four arithmetical operations (+, -, *, /) over three operands (two digits each), and interacts with a touch panel, connected to the FPGA. The calculator functionality has been described in VHDL and the design was done in the *ISE 5.1* software of Xilinx [1,2]. The circuit was implemented in FPGA** XC2S300E **of *Spartan-IIE* family available on the prototyping board *TE-XC2Se* of *Trenz Electronic*. The major advantages of this FPGA are the low cost and a well defined architecture of primary components that are similar to FPGAs of *Virtex* family.**

I. Introdução

O trabalho descrito neste artigo foi realizado na sequência da disciplina de Computação Reconfigurável, leccionada à Licenciatura de Engenharia dos Computadores e Telemática, no Departamento de Electrónica e Telecomunicações da Universidade de Aveiro.

A. Objectivos

O objectivo do trabalho em causa foi criar uma calculadora que possibilitasse a realização de operações sobre três operandos, de dois algarismos cada. As operações consideradas foram: a soma, a subtracção, a multiplicação e a divisão inteira. Os operandos e o resultado obtido são representados em notação decimal. Os operandos, assim como as operações, são introduzidos na FPGA por interacção com o *touch panel*, que apresenta uma interface com o desenho da calculadora. Os operandos, operações e resultado são apresentados no *display* (2 x 16) da placa (Fig. 1).

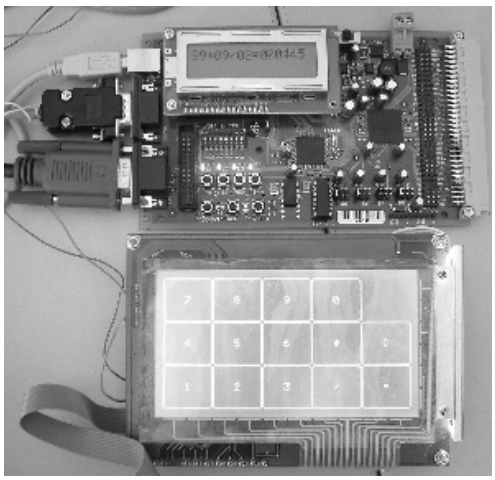

Fig. 1 — Placa de desenvolvimento e *touch panel*

B. Características principais da placa TE-XC2Se

A placa TE-XC2Se [3] tem como componente reconfigurável a FPGA XC2S300E-6-FT256.

A placa possui dois conectores de expansão:

- um com 26 pinos, alguns dos quais são partilhados com o *display* LCD de 2x16 caracteres;
- outro com 96 pinos, que é do tipo VG96.

A placa inclui dois osciladores que geram sinais de relógio de:

- 48 MHz (este é necessário para a porta USB — *Universal Serial Bus*, e pode servir de relógio para o utilizador);
- 25 MHz (este oscilador é substituível).

Os dados do utilizador podem ser visualizados num *display* LCD de 2x16 caracteres e em 5 LEDs individuais.

A porta USB serve para fornecer a alimentação para a placa quando o interruptor respectivo está ligado (o seu estado é reflectido no LED de alimentação). Através da porta USB também se pode programar a FPGA, carregando a configuração na memória *Flash* indestrutível.

A placa possui também memória estática ligada à FPGA.

Pode-se carregar o *bitstream* para a memória *Flash* da placa com a ajuda da ferramenta *TEprog* disponível da *Trenz electronic* [3].

C. Características principais do kit que inclui o touch panel

O *touch panel* [4] utilizado faz parte do *kit* EA KIT240-7. Este *kit* é uma unidade de controlo e de operação que integra diferentes funções.

Inclui uma larga variedade de fontes disponíveis a apresentar e funciona, normalmente, com uma tensão de +5V.

A comunicação entre o *kit* que inclui o *touch panel* e o *display lcd* da placa de desenvolvimento pode ser efectuada quer através do protocolo standard "RS232" (renomeado no principio da década de 90 como "EIA232 Standard" pela *Electronic Industries Association*), quer através do interface RS422.

Para corrigir erros de transmissão em comunicações série são usados dois métodos: comunicação síncrona (os extremos da comunicação estão sincronizados através do uso de um relógio) e a comunicação assíncrona (são inseridas marcas no *stream* de bits a enviar e a transmissão tem que ser efectuada à mesma velocidade). A transferência de dados usada entre o *kit* do *touch panel* e a placa do *display lcd* é efectuada através da comunicação assíncrona (Fig. 2). É enviado 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* (o *kit* utilizado não contempla o uso de um bit de paridade - *parity bit*).

Fig. 2 — Formato dos dados, na comunicação entre o *kit* que inclui o *touch panel* e a placa de desenvolvimento

A taxa de transmissão pode variar entre 1200 e 115200 baud e pode ser escolhida através de *DIP switchers.*

No que diz respeito ao *software*, o *kit* é programável através de comandos para desenhar linhas ou figuras geométricas. O texto pode ser escrito e posicionado com uma precisão ao nível do pixel. Os gráficos podem ser combinados com texto. Existe também a possibilidade de serem apresentados menus do tipo *pull-down*.

O *touch panel* possui iluminação CFL, *blue negative* e uma resolução de 240x128 pixels. Podem-se definir 10x6 campos diferentes. Quando as teclas do *touch panel* são pressionadas, as suas cores podem ser invertidas e um som pode ser produzido.

II. ESPECIFICAÇÃO

Por forma a possibilitar a interacção do utilizador com a calculadora implementada, foi criada uma interface no *touch panel* que se pretendeu aproximar o mais possível das calculadoras comuns (considerando apenas as operações já referidas). O modelo apresentado corresponde ao de uma calculadora simples.

A. Descrição por blocos

Com o objectivo de facilitar a compreensão do trabalho realizado, ir-se-á de seguida dividir a especificação do mesmo por blocos, que por sua vez integram os diferentes módulos utilizados na implementação da calculadora. A descrição por blocos tende a ser mais genérica, mas servirá para introduzir de forma mais explícita uma explicação que, de outra forma, poderia tornar-se algo confusa. Deste modo, são apresentados os quatro blocos constituintes do trabalho em causa:

1. O primeiro bloco tem como finalidade garantir a sincronização dos diferentes componentes, passando pela criação de um relógio que estabelece uma frequência própria. Nesta parte do trabalho evidenciam-se os módulos *rs_divider.vhd e clk_res.sch*.
2. O segundo bloco tem como propósito apresentar a calculadora no *touch panel* e ler e validar os operandos e as operações, obtidos pela interacção do utilizador com o *touch panel*. Nesta parte do trabalho evidencia-se o módulo *rs_control.vhd*.
3. O terceiro bloco tem como objectivo receber os operandos ou operações (validados no módulo *rs_control*) e apresentá-los no *lcd* 2 x 16. A sua função passa, ainda, pelo cálculo do resultado das operações e pela sua apresentação no *lcd*. Nesta parte do trabalho evidencia-se o módulo *lcd.vhd*.
4. Finalmente, o quarto bloco tem como finalidade converter o resultado para valores decimais, isto porque os cálculos são na realidade efectuados em binário, sendo depois o resultado convertido para decimal para apresentação ao utilizador, no lcd. Nesta parte do trabalho evidencia-se o módulo *sch_bcd2int.sch*.

B. Descrição por módulos

De seguida ir-se-á descrever os diferentes módulos constituintes da implementação da calculadora. Esta descrição é mais específica, procurando entrar um pouco mais no pormenor da implementação. Deste modo, são apresentados os módulos constituintes do trabalho em causa:

1. MÓDULO RS_CONTROL.VHD: Neste módulo é utilizado um processo para envio de informação para o *touch panel*, desenhando a interface de interacção com o utilizador. Um outro processo é utilizado para a recepção de informação do *touch panel*; aqui, à medida que são lidos, os valores são validados, para que no módulo *lcd* sejam apresentados os valores correctos (por exemplo: não é possível escolher um operador sem antes ter sido introduzido um operando);
2. MÓDULO LCD.VHD: Neste módulo são guardados, num *array*, os caracteres validados pelo módulo

rs_control, provenientes do *touch panel*, para apresentação no *lcd*. São, ainda, realizadas as operações aritméticas e apresentados os dados no *lcd*. Este módulo utiliza o módulo *sch_bcd2int.sch* com o propósito de converter o resultado de binário para decimal;

3. MÓDULO SCH_BCD2INT.SCH: Neste módulo o resultado dos cálculos efectuados sobre os operandos é convertido de binário para decimal por forma a poder ser apresentado ao utilizador. Para esse efeito, este módulo usa os módulos *mod_bcd2int.vhd* e *mod_division.vhd*;
4. MÓDULO MOD_BCD2INT.VHD: Neste módulo são utilizados os valores obtidos através do módulo *mod_division.vhd*. Cada um destes valores corresponde a um dígito do resultado final. Estes dígitos são combinados (com diferentes pesos) por forma a constituírem o resultado das operações. Assim, o resultado é convertido de binário para decimal;
5. MÓDULO MOD_DIVISION.VHD: Neste módulo é realizada a divisão inteira em notação binária, obtendo-se como resultado quer o quociente, quer o resto. A sua utilidade estende-se, para além da utilização no módulo de conversão de notação, ao cálculo da divisão inteira no módulo *lcd.vhd*. A especificação do algoritmo da divisão presente neste módulo será fornecida em pormenor na seccção II.C.

C. Descrição de processos

Esta descrição ir-se-á cingir a um único processo, considerado o de maior relevância: processo do algoritmo da divisão inteira em notação binária. Por forma a simplificar a sua descrição é introduzido, na Fig. 3, um esquema exemplificativo.

Portanto,

1. Considere-se o dividendo e o divisor;
2. Se o dividendo for menor que o divisor, o quociente toma o valor “0” e o resto é igual ao dividendo;
3. Caso o divisor seja igual a “0”, tem-se uma situação de “ERRO”;
4. Caso os pontos 2 e 3 não se tenham verificado, *shifta-se* o divisor para a esquerda até que o primeiro “1” do divisor coincida com o primeiro “1” do dividendo;
5. Compara-se o dividendo com o valor do divisor obtido em 4;
6. Caso o dividendo seja maior ou igual que o divisor, então o *bit* correspondente do quociente passa a ser “1” e o novo dividendo, assim como o resto, tomam o valor correspondente ao resultado da subtracção do divisor ao dividendo;
7. Caso o dividendo seja menor que o divisor, então o *bit* correspondente do quociente passa a ser “0” e o resto é igual ao dividendo;
8. No caso da situação verificada em 6, *shifta-se* o divisor um *bit* à direita e compara-se o resultado com o valor do dividendo obtido em 6;
9. No caso da situação verificada em 7, *shifta-se* o divisor um *bit* à direita e compara-se o resultado com o dividendo;
10. Os passos considerados são executados sucessivamente até que o divisor final seja igual ao divisor original.

O código do módulo MOD_DIVISION.VHD é o seguinte:

```
--- ENTITY >>> ------------------------------
entity mod_division is
  Port (
    clk: in std_logic;
    rst: in std_logic;
    dividend: in std_logic_vector(19 downto 0);
    divisor: in std_logic_vector(19 downto 0);
    quotient: out std_logic_vector(19 downto 0);
    remainder: out std_logic_vector(19 downto 0)
  );
end mod_division;
--- <<< ENTITY ------------------------------

--- ARCHITECTURE >>> -------------------------
architecture Behavioral_mod_division of
   mod_division is
     type STATE_TYPE is (start, one, two,
three, final);
     signal CS, NS: STATE_TYPE;
     signal divid, divis,
rmd:std_logic_vector(19 downto 0);
     signal quot: std_logic_vector(20 downto
1);

   begin
       -------------------------------------
   process (clk, rst)
     begin
       if rst = '1' then
         CS <= start;
       elsif (clk'event and clk = '1') then
         CS <= NS;
       end if;
   end process;
       -------------------------------------
       -------------------------------------
   process (CS, clk)
     variable cont_divid, cont_divis: integer;
     begin
      if rst = '1' then
      elsif (clk'event and clk = '0') then
       case CS is
         when start =>
          if (dividend < divisor) then
            quot <= (others => '0');
            rmd <= dividend;
            NS <= final;
            elsif (divisor =
              "00000000000000000000") then
             quot <= (others => '1');-- error
                                     -- message
            rmd <= (others => '1');
            NS <= final;
           else
            divid <= dividend;
            divis <= divisor;
            quot <= (others => '0');
            cont_divid := 19;
            cont_divis := 1;
            NS <= one;
           end if;
          when one =>
            if (divid(cont_divid) = '1') then
              NS <= two;
            else
              cont_divid:=cont_divid-1;
```

```
            NS <= one;
          end if;
       when two =>
          if (divis(cont_divid) = '1') then
              NS <= three;
          else
               divis <= divis + divis;
               cont_divis:=cont_divis+1;
               NS <= two;
           end if;

      when three  =>
        if (cont_divis < 1) then
          NS <= final;
       elsif (divid >= divis) then
          quot(cont_divis) <= '1';
          divid <= divid - divis;
          rmd <= divid - divis;
          divis <= '0' & divis(19 downto 1);
          cont_divis := cont_divis - 1;
          NS <= three;
       else
          quot(cont_divis) <= '0';
         divis <= '0' & divis(19 downto 1);
         cont_divis := cont_divis - 1;
         NS <= three;
       end if;

       when final =>
         remainder <= rmd;
         quotient <= quot;
         NS <= start;
      end case;

    end if;
end process;
     -------------------------------------

end Behavioral_mod_division ;

--- <<< ARCHITECTURE --------------------------
```

D. Implementação

Os códigos relativos às três versões da calculadora implementadas estão disponíveis na *WebCT* [5].

III. ARQUITECTURA

Por forma a fornecer uma visualização clara dos diferentes módulos implementados, das relações entre eles e da hierarquia estabelecida, apresenta-se, na Fig.4, a estrutura gráfica ilustrativa.

A descrição detalhada dos módulos foi considerada na seccção II. O código respectivo pode ser consultado na *WebCT* [5].

O modelo de arquitectura apresentado é um dos possíveis modelos de hierarquia implementados. Para a mesma calculadora pode encontrar-se, disponível na *WebCT* [5], outro modelo que corresponde a um raciocínio diferente, mas igualmente válido.

IV. USABILIDADE

A. Interacção com a calculadora

A calculadora implementada é uma calculadora simples que possibilita uma interacção rápida e clara com o utilizador. A interface é apresentada no *touch panel*, sendo o contacto realizado directamente no mesmo. O *touch panel* é um elemento de utilização simples, na medida em que o utilizador pode interagir directamente com o mesmo, sem ser necessário fornecer formação extra para que qualquer pessoa possa facilmente compreender como o utilizar.

Após a configuração ter sido carregada na placa e a interface da calculadora ter sido apresentada no *touch panel*, a sua utilização é equivalente a utilização das calculadoras comuns:

1. Introduzem-se os dois dígitos do primeiro operando no *lcd* da placa, tocando nos números de interesse apresentados no *touch panel*;
2. Introduz-se o operador (+, -, * ou /) no *lcd* da placa, tocando no caracter de interesse apresentado no *touch panel*;
3. Introduzem-se os dois dígitos do segundo operando no *lcd* da placa, tocando nos números de interesse apresentados no *touch panel*;
4. Introduz-se o operador (+, -, * ou /) no *lcd* da placa, tocando no caracter de interesse apresentado no *touch panel*;
5. Introduzem-se os dois dígitos do terceiro operando no *lcd* da placa, tocando nos números de interesse apresentados no *touch panel*;
6. Introduz-se o caracter "=" no *lcd* da placa, indicando que as operações irão ser processadas e o resultado será apresentado, tocando no caracter correspondente apresentado no *touch panel*;
7. Observa-se o resultado no *lcd* da placa.

Um aspecto específico da calculadora implementada é o facto de não realizar cálculos que envolvam números negativos. Esta foi uma limitação imposta por forma a simplificar o trabalho em causa.

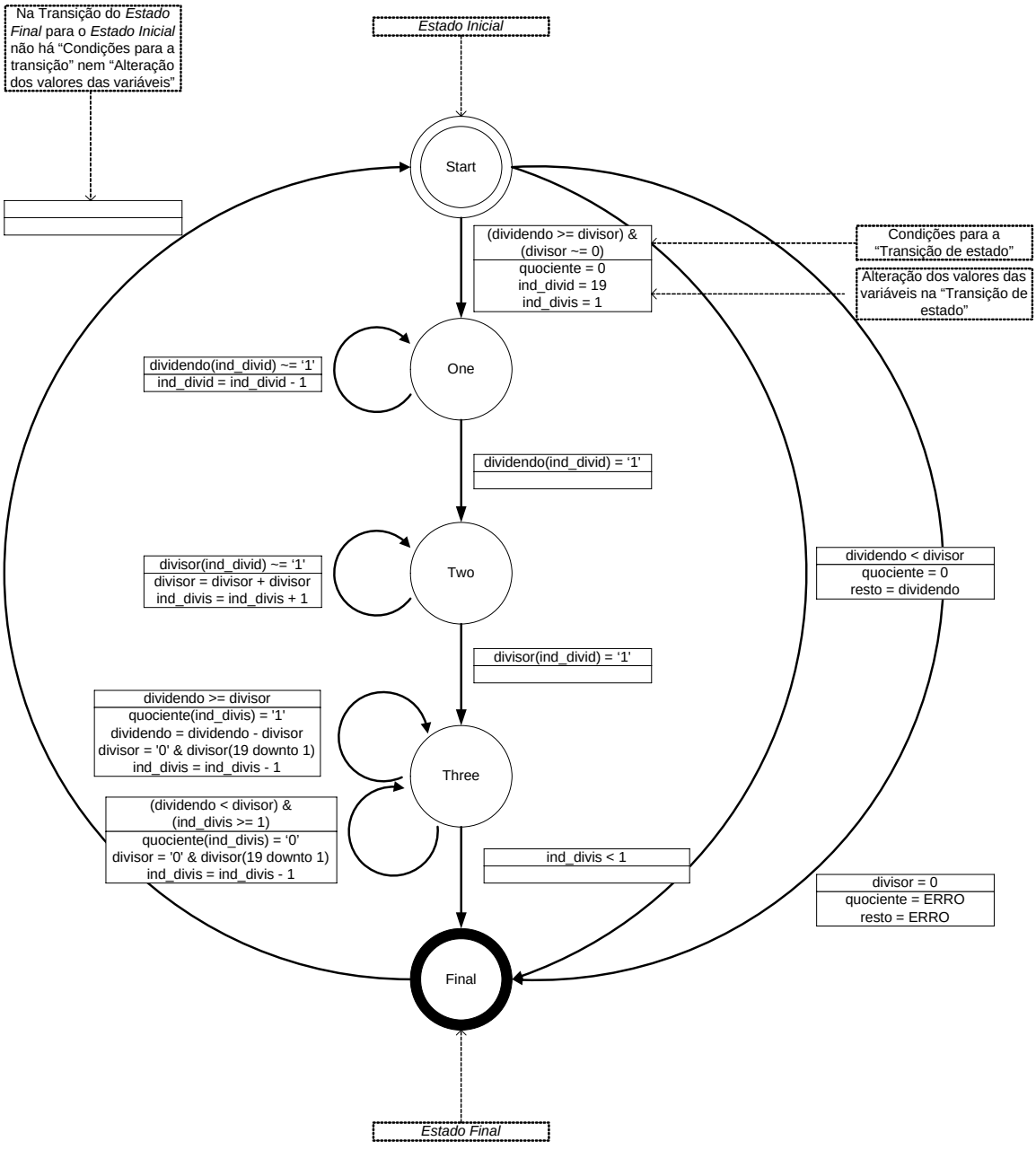

Fig. 3 — Diagrama de fluxos do *algoritmo da divisão inteira em notação binária*

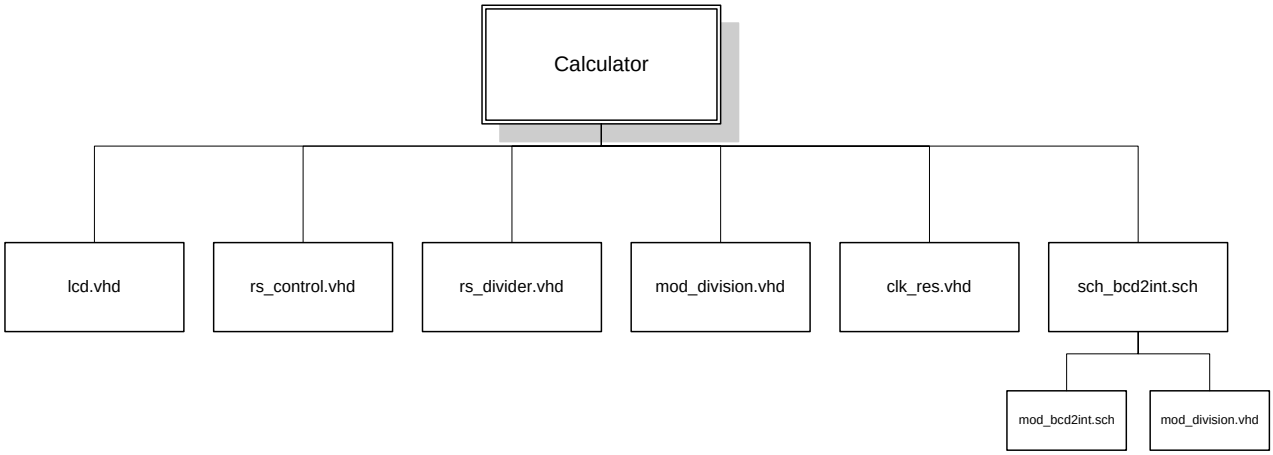

Fig. 4 — Estrutura gráfica do modelo da calculadora implementado

V. CONCLUSÃO

A implementação da calculadora simples foi um trabalho interessante que possibilitou a descoberta e o contacto com algumas das potencialidades da linguagem VHDL.

O processo que envolveu todo o trabalho em causa foi moroso e constituiu um mecanismo de aprendizagem progressivo que, todavia, se revelou bastante compensatório.

Os conceitos utilizados, apesar de não serem desconhecidos, envolveram um aprofundamento algo exigente, na medida em que a cadeira no âmbito da qual este trabalho foi concretizado (*Computação Reconfigurável*) engloba informação menos explorada ou mesmo nova.

Apesar da implementação da calculadora não ter sido executada em grupo, na medida em que os três autores do artigo se envolveram em trabalhos semelhantes, este artigo reflecte o estudo realizado individualmente por cada um dos elementos identificados. É portanto possível encontrar três implementações diferentes da calculadora simples, com três raciocínios personalizados, que podem, no entanto, espelhar-se, de certa forma, na informação apresentada no artigo.

Os objectivos dos trabalhos foram alcançados, pois todas as metas foram atingidas no tempo limite estabelecido.

REFERÊNCIAS

[1] Xilinx 5 Software Manuals, Xilinx Inc, 2002.

[2] Xilinx, The programmable logic data book, Xilinx, 2000;

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

[4] Electronic Assembly, Control panel with fonts, graphics commands and macros, 2000.

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

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