Grupo de Electrónica e Instrumentação

Electrónica Analógica e Digital

Trabalho de Laboratório 3 (Março de 2015)

Objectivo e introdução

O objectivo deste trabalho é finalizar o "curto" curso de desenvolvimento de hardware digital, baseado em metodologias de projecto modernas (HDLs). Para isso vai ser implementado um circuito de média complexidade (um somador) numa placa *Basys2* que contém uma FPGA *Xilinx* Spartan 3E. Recorre-se, de novo, ao ISE/Webpack (ISE) da *Xilinx* para sintetizar o projecto a partir da descrição em Verilog. Pode ser utilizada nesta aventura qualquer versão do ISE posterior à 8.2. (Nalguns computadores apenas existe a versão 6.3; esta pode servir para escrever e depurar o Verilog, mas não suporta a FPGA Spartan 3E)

Material

O hardware utilizado no laboratório consiste da placa **Basys2** da Digilent. Inclui uma FPGA Xilinx Spartan 3E com, aproximadamente, 100 mil portas "equivalentes" (a "porta equivalente" é a "unidade de medida" da capacidade das FPGAs), um controlador USB, vários interruptores de pressão ("pushbuttons"), interruptores de posição ("dip-switches"), 8 leds e quatro *displays* de 7 segmentos (DISP7), realizados com LEDs, que nos permitirão comunicar com a FPGA. A documentação relevante acerca destas placas está disponível no Moodle e na "dropbox de EAD".

Plano de trabalho

Implemente no ISE cada um dos blocos de Verilog abaixo descritos e observe o circuito que o sintetizador gera (clique na linha *"RTL Schematic"* para ver os circuitos). Pode também examinar a sua implantação na FPGA (invocando o *floorplanner* do fluxo de projecto). Implemente os módulos num mesmo ficheiro, na medida em que todos eles são instanciados no módulo de nível superior TOP (na realidade, este descreve o circuito de adição). Após ter sintetizado com sucesso o projecto, poderá descarregá-lo para a placa Basys2 (se não tiver tempo, tente fazê-lo nos horários de dúvidas).

Somador em binário

O resultado da soma de dois números de dimensão n tem, no máximo, a dimensão de $n + 1$. No presente trabalho pretende-se implementar um somador de 4 bits, pelo que o resultado corresponderá a uma palavra de 5 bits.

A parcela máxima neste contexto é 15 ('1111'). O resultado máximo da soma será, portanto, 30 ('11110'). Para visualizar resultados com 5 bits no DISP7 vamos necessitar de dois diplays de 7 segmentos (Nota: para perceber como são utilizados estes displays na Basys2, leia as páginas 4 e 5 do pequeno manual da Basys2 disponível no *moodle* ou na *dropbox* (ou então neste link).

Um diagrama de blocos do somador pode ser visto na fig. 1. Na parte inferior são especificados os três módulos que é necessário escrever. O módulo **TOP** dispõe dos sinais de interface com os interruptores e com o display de 7 segmentos, e instancia um descodificador (tipo **BCD7S** e denominado BCD-to-7Seg) e quatro *full-adders* (FA0,...,FA3) do tipo **FA**.

Vai-se passar a descrever estes módulos em mais detalhe.

Meio somador ("half-adder") e somador completo ("full-adder")

Na fig. 2 são mostrados os diagramas lógicos do meio somador e do somador completo (FA, de *full-adder*). Este último é o bloco básico que permite construir somadores binários de qualquer dimensão, sendo uma das células básicas na aritmética digital. O bloco soma (em binário) as entradas A e B com a carga na

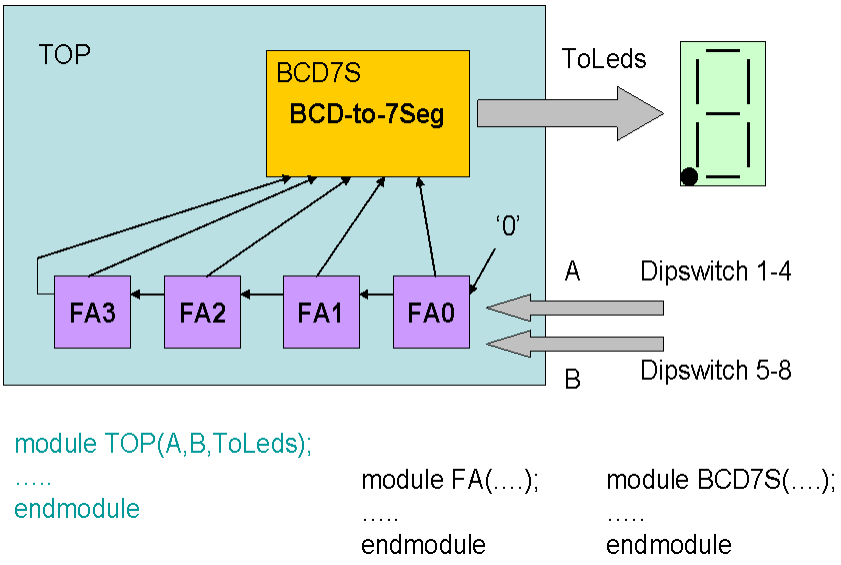

Fig. 1: Diagrama de blocos do somador.

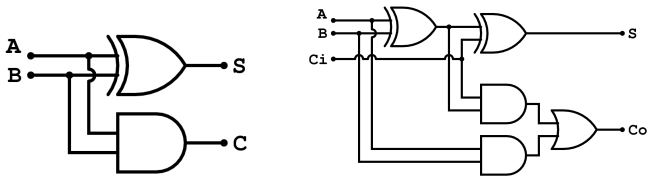

Fig. 2: À esquerda: *half-adder*; à direita: *full-adder*.

entrada, CIN, e apresenta nas saídas o resultado, S, e a carga ("*carry*"), C, que é injectada no próximo bloco. À esquerda da figura 3 mostra-se também (a título de curiosidade) que o *full-adder* é criado com dois *half-adders* e uma porta OR.

À direita nesta mesma figura pode ser visto o esquema lógico do somador de 4 bits que corresponde ao circuito 7483 da família TTL. É este somador em cascata que irá aqui ser descrito em Verilog.

Implemente o somador completo da fig. 2 como um módulo independente, para que possa ser utilizado na realização de um somador de 4 bits. A estrutura deste somador de 4 bits, mostrado na figura 3, consiste na cascata de 4 *full-adders*.

Descodificador de hexadecimal para display de 7 segmentos

Para poder visualizar o resultado da operação aritmética vai necessitar de usar o display de 7 segmentos. Para isso tem de escrever o descodificador que realiza essa conversão de binário para display de 7 segmentos. No apêndice está o mapa que relaciona os 8 LEDs dos displays, e os interruptores, com os pinos da FPGA da Basys2. O nome do ficheiro com essa informação é `Basys2.ucf`.

Para ajudar na escrita do descodificador mostra-se, de seguida, um de muitos exemplos, retirado algures da Internet. Note que este código não contempla o "ponto decimal" que iremos usar para mostrar o bit mais significativo do resultado da adição, e a denominação dos sinais (pelo menos) tem de ser editada para concordar com os nomes do presente projecto.

```
module SevenSegmentDisplayDecoder(ssOut, nIn);
  output [6:0] ssOut;
  reg [6:0] ssOut;
  input [3:0] nIn;
```

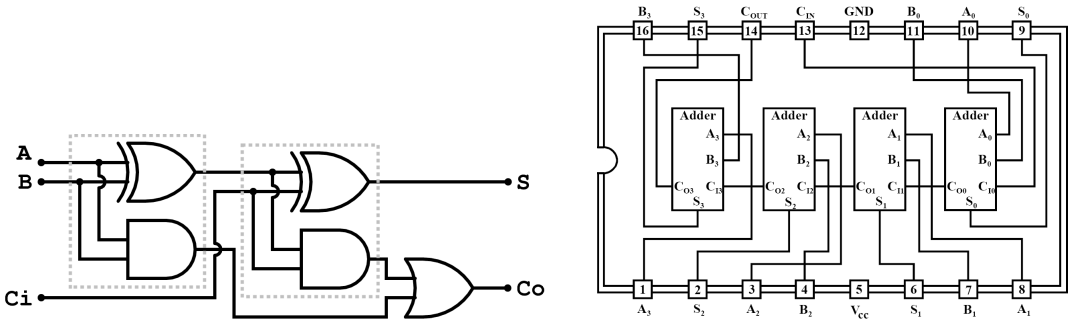

Fig. 3: À esquerda: ilustração de que um *full-adder* é implementado com dois *half-adders* e uma porta OR; à direita: diagrama do circuito integrado 7483 da família TTL que é um somador de dois números de 4 bits realizado com uma cascata de quatro *full-adders*.

```
// ssOut format {g, f, e, d, c, b, a}
always @(nIn)
  case (nIn)
   4'h0: ssOut = 7'b0111111;
   4'h1: ssOut = 7'b0000110;
   4'h2: ssOut = 7'b1011011;
   4'h3: ssOut = 7'b1001111;
   4'h4: ssOut = 7'b1100110;
   4'h5: ssOut = 7'b1101101;
   4'h6: ssOut = 7'b1111101;
   4'h7: ssOut = 7'b0000111;
   4'h8: ssOut = 7'b1111111;
   4'h9: ssOut = 7'b1100111;
   4'hA: ssOut = 7'b1110111;
   4'hB: ssOut = 7'b1111100;
   4'hC: ssOut = 7'b0111001;
   4'hD: ssOut = 7'b1011110;
   4'hE: ssOut = 7'b1111001;
   4'hF: ssOut = 7'b1110001;
  endcase

endmodule
```

Módulos de utilidade geral

Embora não sejam necessários no presente trabalho, não resistimos a apresentar dois módulos que amiúde se revelam úteis em circuitos práticos.

Contador

O contador de 28 bits, já usado no trabalho de laboratório anterior, pode servir para reduzir a frequência de relógio de 50 MHz (gerada na placa Basys2) para poucos Hertz. A redução tem de ser obrigatoriamente feita com uma constante divisora que é potência de 2. Neste exemplo, em que o relógio-saída é retirado do bit de índice 25 do contador, essa frequência será aproximadamente $f_{clockout} = 50 \times 10^6 / 2^{25+1} \approx 0,7$ Hz.

```
module CONTADOR(clkin, clockout);
  input clkin;
  output clockout;
  reg [27:0] bigcount;

  // processo que é sintetizado como um circuito sequencial (contador)
  always @(posedge clkin)
    bigcount <= bigcount+1 ;

  // processo concorrente que é sintetizado para um circuito combinatório
  // (neste caso é apenas uma ligação directa)
  assign clockout = bigcount[25];
endmodule
```

Sincronização de sinais assíncronos externos

É fortemente sugerido (senão, mesmo, obrigatório) que os sinais assíncronos externos lidos na FPGA (como é o caso daqueles provenientes dos interruptores da placa) sejam sincronizados pelo relógio, principalmente se servirem de entradas em circuitos sequenciais. Uma popular técnica para fazer este "debouncing" usa um registo de deslocamento. O sinal externo vai sendo amostrado pelo relógio e só quando todos os bits do registo, à excepção do último, forem positivos é que é enviado um impulso (POUT) "para dentro" da FPGA. A duração do sinal de entrada, PIN, necessária para disparar o pulso interno depende da dimensão N do registo de deslocamento e do período do relógio: será aproximadamente $(N-1) \times T_{CK}$.

```
// 8-bit Shift-Left Register with Positive-Edge Clock,
// Asynchronous Reset, Serial In, and Serial Out
// used for mechanical input debouncing
module pulse_deb (CK, CLR, PIN, POUT);
  input CK,PIN,CLR;
  output POUT;
  reg [7:0] tmp;

  always @(posedge CK or posedge CLR) begin
    if (CLR) tmp <= 8'b00000000;
    else tmp <= {tmp[6:0], PIN};
  end

  // and (POUT,~tmp[7],tmp[6],tmp[5],tmp[4],tmp[3],tmp[2],tmp[1],tmp[0]);
  assign POUT = &tmp[6:0] & ~tmp[7];
endmodule
```

Resta acrescentar que não é necessário utilizar estes "debouncers" no somador, visto ele ser realizado por um circuito integralmente combinatório.

Apêndice

O mapa com as ligações entre os pinos da FPGA, o clock, os interruptores da placa Basys2 e o display de 7 segmentos é apresentado na lista que se segue.

```
# This file is a general .ucf for Basys2 rev C board
# To use it in a project:
# - remove or comment the lines corresponding to unused pins
# - rename the used signals according to the project
# clock pin for Basys2 Board; freq. of mclk is 50 MHz
NET "mclk" LOC = "B8"; # Bank = 0, Signal name = MCLK
NET "mclk" CLOCK_DEDICATED_ROUTE = FALSE;

# 7 segment pinout.
# The anode shall be at '0' to light the corresponding display
# Pin assignment for DispCtl
# Connected to Basys2 onBoard 7seg display
NET "seg<0>" LOC = "L14"; # Bank = 1, Signal name = CA
NET "seg<1>" LOC = "H12"; # Bank = 1, Signal name = CB
NET "seg<2>" LOC = "N14"; # Bank = 1, Signal name = CC
NET "seg<3>" LOC = "N11"; # Bank = 2, Signal name = CD
NET "seg<4>" LOC = "P12"; # Bank = 2, Signal name = CE
NET "seg<5>" LOC = "L13"; # Bank = 1, Signal name = CF
NET "seg<6>" LOC = "M12"; # Bank = 1, Signal name = CG
NET "dp" LOC = "N13"; # Bank = 1, Signal name = DP

NET "an<3>" LOC = "K14"; # Bank = 1, Signal name = AN3
NET "an<2>" LOC = "M13"; # Bank = 1, Signal name = AN2
NET "an<1>" LOC = "J12"; # Bank = 1, Signal name = AN1
NET "an<0>" LOC = "F12"; # Bank = 1, Signal name = AN0

# Pin assignment for LEDs
NET "Led<7>" LOC = "G1" ; # Bank = 3, Signal name = LD7
NET "Led<6>" LOC = "P4" ; # Bank = 2, Signal name = LD6
NET "Led<5>" LOC = "N4" ;  # Bank = 2, Signal name = LD5
NET "Led<4>" LOC = "N5" ;  # Bank = 2, Signal name = LD4
NET "Led<3>" LOC = "P6" ; # Bank = 2, Signal name = LD3
NET "Led<2>" LOC = "P7" ; # Bank = 3, Signal name = LD2
NET "Led<1>" LOC = "M11" ; # Bank = 2, Signal name = LD1
NET "Led<0>" LOC = "M5" ;  # Bank = 2, Signal name = LD0

# Pin assignment for SWs and buttons # 8 lock switches
NET "sw<7>" LOC = "N3";  # Bank = 2, Signal name = SW7
NET "sw<6>" LOC = "E2";  # Bank = 3, Signal name = SW6
NET "sw<5>" LOC = "F3";  # Bank = 3, Signal name = SW5
NET "sw<4>" LOC = "G3";  # Bank = 3, Signal name = SW4
NET "sw<3>" LOC = "B4";  # Bank = 3, Signal name = SW3
NET "sw<2>" LOC = "K3";  # Bank = 3, Signal name = SW2
NET "sw<1>" LOC = "L3";  # Bank = 3, Signal name = SW1
NET "sw<0>" LOC = "P11";  # Bank = 2, Signal name = SW0

# 4 buttons
NET "btn<3>" LOC = "A7";  # Bank = 1, Signal name = BTN3
NET "btn<2>" LOC = "M4";  # Bank = 0, Signal name = BTN2
NET "btn<1>" LOC = "C11"; # Bank = 2, Signal name = BTN1
NET "btn<0>" LOC = "G12"; # Bank = 0, Signal name = BTN0
```