LSDI 2013/14 – Trabalho laboratorial 2

Projeto de Circuitos Combinacionais

1 - Introdução

Este trabalho laboratorial tem por objetivos desenvolver circuitos combinacionais simples e explorar uma ferramenta computacional para projeto de sistemas digitais que permite o desenvolvimento, implementação e ensaio de circuitos digitais. Para tal, serão usadas técnicas de obtenção de circuitos lógicos a partir da descrição das suas funções e a linguagem de descrição de *hardware* Verilog. Como ambiente de desenvolvimento, será usado o *software* de projeto ISE da Xilinx. O ensaio dos circuitos far-se-á na placa de desenvolvimento S3Board (www.digilentinc.com/Products/Detail.cfm?Prod=S3BOARD), a qual inclui um dispositivo reconfigurável FPGA (*Field Programmable Gate Array*) com capacidade equivalente a 200.000 portas lógicas. O *software* de projeto ISE, utilizado para o desenvolvimento deste e dos próximos trabalhos práticos da unidade curricular de Laboratório de Sistemas Digitais, é disponibilizado gratuitamente pelo fabricante do circuito FPGA, Xilinx (www.xilinx.com). A versão instalada nos laboratórios (ISE 10.1) pode ser obtida em www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools/v10_1.html.

Para simplificar a construção e ensaio dos circuitos em laboratório, será fornecido (na pasta de conteúdos da unidade curricular no Sigarra) um arquivo com o projeto ISE parcialmente construído, LSDI2013_lab2, que já inclui todos os elementos necessários para implementar interfaces com os dispositivos de entrada e saída que a placa S3Board dispõe (LEDs, visualizadores de 7 segmentos, botões de pressão e interruptores).

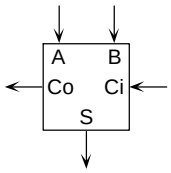
Este trabalho é composto por duas partes: projeto e implementação. A componente de projeto deverá ser realizada previamente pelos estudantes como trabalho de casa. Durante a aula laboratorial serão implementados e ensaiados os circuitos projetados, de acordo com os passos apresentados neste guião laboratorial.

A preparação, como trabalho de casa, é fundamental para o cumprimento dos objetivos dos trabalhos laboratoriais e para a avaliação a realizar posteriormente na forma de fichas laboratoriais.

2 - Projeto

2.1- Projeto de um circuito somador completo

Um somador completo (*full-adder*) é um circuito com três entradas e duas saídas que realiza a operação aritmética adição entre 3 números de 1 bit (cada um só pode ser 0 ou 1), produzindo na saída 2 bits que representam o valor da soma desses 3 bits (podendo ser 0, 1, 2 ou 3). As três entradas podem designar-se por A, B e Ci, e as duas saídas por Co e S, representando Co o bit mais significativo do resultado. A figura 1 mostra o símbolo deste circuito e a respetiva tabela de verdade.



A	B	Ci	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Figura 1 – Somador completo: símbolo e tabela de verdade.

a) Recorrendo a representações em mapas de Karnaugh, obtenha expressões minimizadas na forma soma de produtos ou produto de somas que realizem as funções Co(A,B,Ci) e S(A,B,Ci).

b) Desenhe os circuitos lógicos `sum` e `carryout` que realizam, respetivamente, as funções S(A,B,Ci) e Co(A,B,Ci) encontradas. Para tal, use as portas lógicas `not`, `and` e `or`.

c) Construa agora um circuito, `fulladder`, que implemente o somador completo recorrendo aos símbolos dos circuitos `sum` e `carryout` (figura 2), e respetivas interligações. Mantenha as entradas e as saídas do somador completo com os nomes A, B, Ci e Co, S, respetivamente.

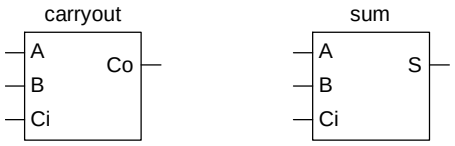


Figura 2 – Símbolos dos circuitos que realizam as funções Co(A,B,Ci) e S(A,B,Ci).

2.2 – Projeto de um circuito somador/subtrator para números de 4 bits

O circuito desenvolvido nas alíneas anteriores é um componente fundamental para a construção de circuitos aritméticos. O circuito mostrado na figura 3 implementa um somador de dois números de n bits, obtido por interligação de n blocos do tipo somador completo. O operando A é representado pelos bits A_{n-1}, …, A_1 e A_0, o operando B pelos bits B_{n-1}, …, B_1 e B_0, e o resultado S pelos bits S_{n-1}, …, S_1 e S_0. Este circuito é designado por *ripple-carry adder* e produz o resultado da adição de forma semelhante à que é seguida quando se efetua a operação manualmente: a saída Co de um *full-adder* representa o bit de transporte para o andar seguinte, que por sua vez é ligado à entrada Ci do *full-adder* seguinte. Note que a entrada Ci do andar menos significativo deverá ser ligada a zero e que a saída Co mais significativa representa o transporte resultante da soma dos bits mais significativos.

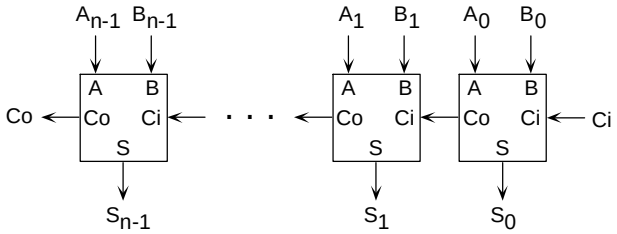


Figura 3 – Somador de 2 números de n bits utilizando n blocos do tipo *full-adder*.

a) Com base na figura 3, desenhe o circuito somador de 4 bits, `adder4b`, por interligação de 4 circuitos somadores completos. Considere como interface as entradas A3, A2, A1, A0, B3, B2, B1, B0 e Ci, e as saídas S3, S2, S1, S0 e Co, tal como no símbolo do circuito apresentado na figura 4.

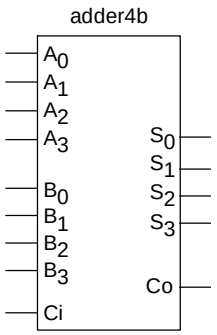


Figura 4 – Símbolo do somador de 4 bits.

b) A operação de subtração A − B pode ser realizada como A + (−B), onde (−B) é o complemento para 2 de B. Com base no circuito `adder4b` desenvolvido na alínea anterior, é possível realizar adições ou subtrações,

consoante o valor lógico que é colocado numa entrada adicional add_sub que permitirá escolher a operação a realizar. Se add_sub for 0, as saídas deverão apresentar o resultado da adição, A+B; se add_sub for 1, o valor apresentado nas saídas deverá ser A−B.
Comece por desenhar este circuito (`addsub4b`) capaz de realizar adições e subtrações de 4 bits, usando o somador de 4 bits representado na figura 4 e mais alguma lógica. Seguidamente, escreva um módulo Verilog que implemente este somador/subtrator de 4 bits, por interligação das portas lógicas e do bloco `adder4b` do circuito obtido. Caso utilize portas lógicas XOR, considere que possuem um tempo de propagação de 9 ns. O módulo deverá ter a seguinte interface:

```
module addsub4b(add_sub, A3, A2, A1, A0, B3, B2, B1, B0, Co, R3, R2, R1, R0);
  input add_sub;  // Se add_sub=0 faz A+B; se add_sub=1 faz A-B
  input A3, A2, A1, A0;  // Operando A
  input B3, B2, B1, B0;  // Operando B
  output Co;
  output R3, R2, R1, R0; // Resultado R
```

3 - Implementação e ensaio

Após o projeto dos circuitos somadores, resultante da descrição feita na secção anterior, esta secção descreve a respetiva implementação e ensaio a realizar nas aulas laboratoriais. Para tal, irá utilizar o *software* ISE da Xilinx, para desenvolver os circuitos e implementá-los num dispositivo reconfigurável contido na placa S3board, podendo assim ensaiá-los.

Para simplificar o processo de desenvolvimento é fornecido o projeto LSDI2013_Lab2 do ISE com os circuitos e o módulo Verilog que projetou na primeira parte do trabalho parcialmente construídos: relativamente aos circuitos estão desenhadas as entradas e as saídas a utilizar; para o módulo Verilog é fornecida a interface. Os ficheiros com estes conteúdos deverão ser editados e completados com os respetivos circuitos, de acordo com o projeto realizado. A tabela 1 identifica estes conteúdos. No projeto estão também incluídos outros módulos, que fazem a interface com a placa S3board, não devendo alterá-los.

Ficheiro	Módulo ou circuito implementado	Usado por
sum.sch	saída S do somador completo	`fulladder`
carryout.sch	saída Co do somador completo	`fulladder`
fulladder.sch	somador completo	`adder4b`
Fulladder_tb.v	simulação do somador completo	-
adder4b.sch	somador de 4 bits	`adder4b_tb` e `addsub4b`
adder4b_tb.v	simulação do somador de 4 bits	-
addsub4b.v	somador/subtrator de 4 bits	`addsub4b_tb` e `s3board_toplevel`
addsub4b_tb.v	simulação do somador/subtrator de 4 bits	-

Tabela 1 – Conteúdo do projeto ISE LSDI2013_Lab2.

Após invocar o ISE abra o projeto LSDI2013_Lab2 (usando a opção "*Open project* …"), execute os passos a seguir descritos e responda às questões colocadas.

3.1 – Implementação e simulação do somador completo

Edite os ficheiros sum.sch, carryout.sch e fulladder.sch, e complete-os com a implementação dos respetivos circuitos. Para tal, na janela "*Sources*", faça um duplo clique sobre o nome de cada ficheiro a abrir. Na edição dos vários circuitos, utilize as portas lógicas NOT, AND e OR disponíveis em "*Categories, <C:/ … /LSDI2013_lab2>"*. O atraso de propagação destas portas lógicas é 2, 4 e 3 ns, respetivamente.

Para verificar se o somador completo funciona corretamente deve proceder à sua simulação. A simulação consiste na aplicação de valores às entradas do circuito e observação das correspondentes saídas. Comparando as saídas com os valores esperados verifica-se se o circuito apresenta o comportamento desejado. O ficheiro fulladder_tb.v fornecido, já completo, descreve uma sequência de valores a aplicar às entradas A, B e Ci do somador completo, as quais determinam os valores das saídas Co e S.

Proceda então à simulação do somador completo: selecione o ficheiro fulladder_tb.v e, na janela "*Processes"* (visualizada na figura 5), invoque o simulador ModelSim fazendo duplo clique em "*Simulate Behavioral Model*".

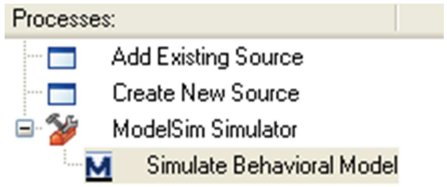


Figura 5 – Conteúdo da janela "*Processes*" para a simulação.

Expanda a janela de visualização das formas de onda ("*Wave*") referentes às entradas e saídas do circuito e verifique se as saídas apresentam os valores esperados. Observe e analise os tempos de propagação com que as saídas vêm afetadas, não esquecendo que os atrasos considerados atrás se referem à versão de duas entradas do AND e do OR.

Questão 1: Considere t=214 ns. Que valores apresentam Co e S? Estes valores são os esperados quando A=0, B=0 e Ci=1? E em t=202 ns?

Questão 2: Qual o tempo de propagação máximo verificado?

3.2 – Implementação e simulação do somador de 4 bits

Edite o ficheiro adder4b.sch e complete-o com o circuito que obteve.

O ficheiro adder4b_tb.v permite efetuar uma simulação lógica do somador de 4 bits. Nele é descrita uma sequência de valores a aplicar às entradas $A=A_3A_2A_1A_0$, $B=B_3B_2B_1B_0$ e Ci do somador, as quais determinam os valores das saídas Co e $S=S_3S_2S_1S_0$. Selecione o ficheiro adder4b_tb.v e invoque o simulador fazendo duplo clique em "*Simulate Behavioral Model*". Expanda a janela de visualização das formas de onda referentes às entradas e saídas do circuito somador e verifique, atentamente, se as saídas apresentam os valores esperados. Observe e analise os tempos de propagação com que as saídas vêm afetadas.

Questão 3: Quando $A_3A_2A_1A_0=0010$ e $B_3B_2B_1B_0=0011$ a soma é $S_3S_2S_1S_0=0101$. Por que razão $S_1=0$ surge primeiro que $S_3=0$?

Questão 4: Altere o ficheiro adder4b_tb.v de modo a poder simular a operação 1111+0000 com Ci=1. Que valores observa em Co e $S=S_3S_2S_1S_0$?

3.3 – Implementação e simulação do somador/subtrator de 4 bits

Relativamente ao somador/subtrator de 4 bits proceda de forma semelhante à praticada com os módulos anteriores, tendo em consideração que este módulo vai ser descrito em Verilog. Após ter completado a edição do código Verilog deste módulo, `addsub4b()`, selecione o respetivo ficheiro de simulação (addsub4b_tb.v) e execute a simulação nele descrita invocando o simulador.

Analise atentamente os resultados da simulação observando as respetivas formas de onda, verificando se os resultados correspondem aos esperados.

Questão 5: Por que razão se usou um intervalo de tempo de 100 ns entre atribuições sucessivas de valores às entradas e não de, por exemplo, 10 ns?

3.4 – Ensaio do somador/subtrator de 4 bits

Implemente e experimente o circuito somador/subtrator de 4 bits, programando a placa S3board. Na lista de itens do projeto, o ficheiro s3board_toplevel.v usa o módulo somador/subtrator de 4 bits, assim como outros módulos, para interface com os interruptores e visualizadores de 7 segmentos, permitindo a sua implementação na placa. Selecione s3board_toplevel.v e execute o comando *"Generate Programming File"*, a partir da janela "*Processes*", que permite obter o ficheiro de configuração do dispositivo reconfigurável contido na placa. A seguir execute "*Configure Target Device*", o que permite transferir o ficheiro de configuração gerado, do computador para a placa, e programar o dispositivo FPGA com o circuito projetado. Realize estas operações de acordo com as indicações dadas pelo docente.

A figura 6 identifica os recursos da placa usados pelo circuito. O operando A é definido pelo estado dos interruptores SW7 a SW4, isto é, os quatro da esquerda, e o operando B é definido por SW3 a SW0. O valor dos operandos A e B é apresentado em hexadecimal nos visualizadores de 7 segmentos (B no visualizador da direita e A no seguinte). O bit de seleção da operação (add_sub) é definido pelo botão de pressão btn0 assinalado. O valor do resultado R, A+B ou A−B, é apresentado em hexadecimal no 3º visualizador e a saída com o bit de transporte Co é apresentada no visualizador mais à esquerda.

Ensaie o circuito definindo vários valores para os operandos A e B através dos respetivos interruptores. Experimente e verifique a realização de operações de adição e subtração.

Questão 6: Supondo que não pressiona o botão add_sub, como deverão estar posicionados os interruptores que definem A de modo a resultar no conjunto dos visualizadores o conteúdo "0F…9"?

Questão 7: Considere que $A_3A_2A_1A_0$=1001 e $B_3B_2B_1B_0$=0011 estão representados em complemento para 2. Recorrendo ao circuito implementado, indique o valor decimal de A+B e A−B, justificando se ocorre *overflow*.

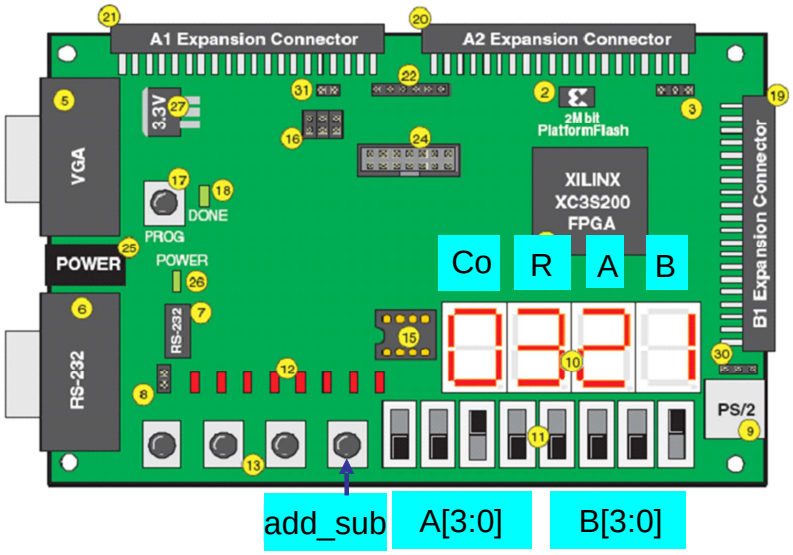


Figura 6 – Identificação das entradas e saídas do somador/subtrator de 4 bits na placa S3board.

Tarefa complementar: Detetor de *overflow* na adição e na subtração

Nota: Esta etapa vem na sequência do trabalho descrito nas seções anteriores, destinando-se aos grupos que o terminaram mais cedo. Deverá portanto ser realizada só depois de terem sido concluídos todos os passos anteriores e de ter verificado com sucesso o funcionamento do circuito na placa de prototipagem.

Pretende-se que o funcionamento do detetor de *overflow* que a seguir se descreve seja verificado através de simulação lógica antes de o implementar e ensaiar na placa de prototipagem.

Tal como estudou, a deteção da situação de *overflow* na adição ou subtração binárias em complemento para 2 pode ser realizada analisando os bits de sinal dos dois operandos e do resultado. Essa situação ocorre na adição binária quando os dois operandos têm o mesmo sinal, mas o resultado da sua adição tem sinal oposto (note que quando se somam operandos com sinais opostos nunca ocorre *overflow*). Como no circuito realizado a subtração é realizada adicionando ao diminuendo o simétrico do diminuidor, a situação de *overflow* acontece quando os dois operandos têm sinais opostos e o sinal do resultado é diferente do sinal do diminuendo.

Construa um circuito capaz de detetar a situação de *overflow* em adições e subtrações, e verifique o seu funcionamento ligado ao circuito `addsub4b` que construiu. O detetor de *overflow* deverá ser escrito em Verilog e ter a interface mostrada a seguir. A saída que identifica *overflow* deverá ser ligada a um dos 8 LEDs disponíveis na placa (sinais `ld0` a `ld7`), solicitando para isso o apoio do docente.

```
module detetor_overflow( signA, signB, signR, addsub, OVF);
input signA, signB, signR;  // bits de sinal dos dois operandos e do resultado
input addsub;               // sinal que indica adição (0) ou subtração (1)
output OVF;                 // saída que identifica com 1 a ocorrência de overflow
```