LSDI 2013/14 - Laboratório 4

Caminho de Dados

1. Introdução

O circuito implementado no trabalho laboratorial anterior permitiu efetuar cálculos simples. No entanto, para calcular algumas expressões concluiu-se que era necessário memorizar temporariamente resultados de cálculos anteriores. Veja-se o exemplo seguinte:
Pretende-se obter o produto de X (o valor definido nos interruptores) pela constante 10. Este cálculo (10×X) pode ser decomposto em (2+8)×X = 2×X+8×X. Usando linguagem simbólica, a sequência de operações elementares seria então a seguinte:

A ← X;	começamos por colocar o valor de X no registo A
A ← A << 1;	depois deslocamos o conteúdo de A 1 bit para a esquerda, obtendo 2×X

Mas agora como continuar? Se continuamos a efetuar deslocamentos para a esquerda, com o intuito de obter o valor 8×X, perder-se-à o valor 2×X que se encontra de momento no registo A. Uma forma de resolver o problema seria anotar num papel o valor de 2×X e continuar a efetuar as operações seguintes, adicionando-o no final. No entanto, esta solução necessita de algo externo ao próprio circuito para que seja capaz de memorizar este valor intermédio (neste exemplo, uma folha para anotar).

Para resolver este problema e aumentar a capacidade de armazenamento do sistema, vamos acrescentar 3 registos ao circuito que foi ensaiado no laboratório 3 (R1, R2 e R3), resultando a composição mostrada na figura 1.

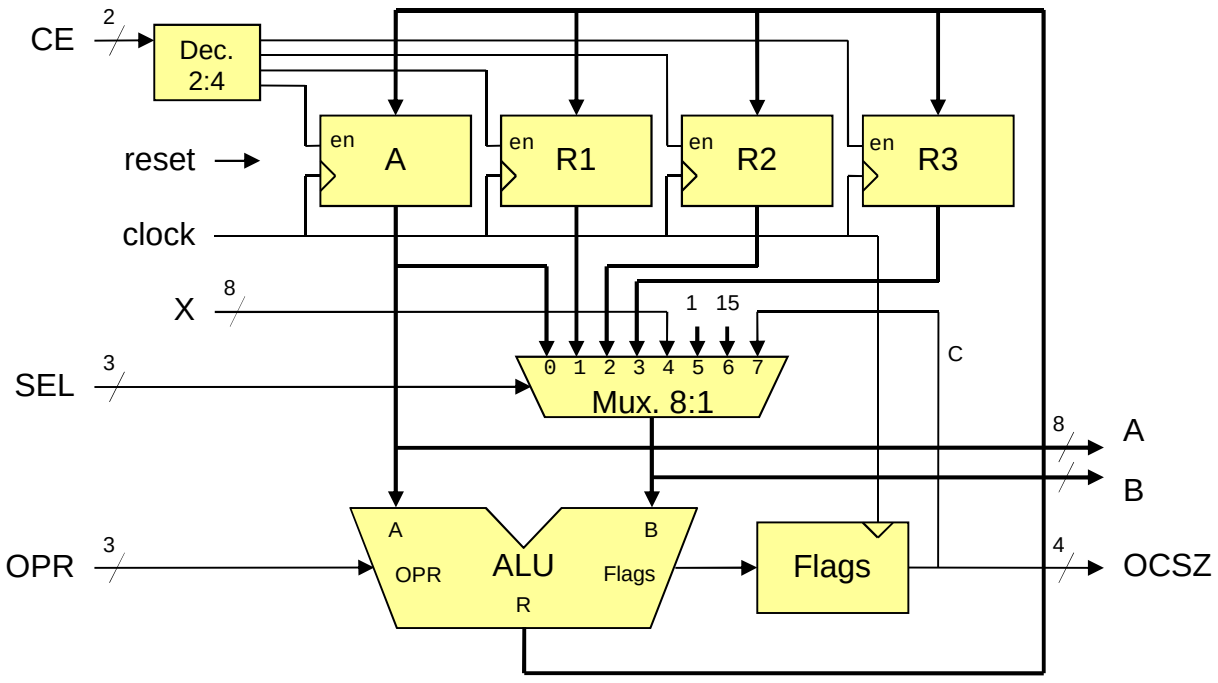

Figura 1 – Novo caminho de dados.

Neste circuito a ALU volta a receber 2 operandos A e B, produzindo um resultado R para uma de 8 operações possíveis (as mesmas do trabalho anterior) definidas por OPR, tal como está indicado na tabela 1.

Tabela 1 – Operações realizadas pela ALU

OPR	Operação
000	R ← B
001	R ← A − B
010	R ← A + B
011	R ← A ^ B
100	R ← A >> 1
101	R ← A << 1
110	R ← A & B
111	R ← A | B

Tal como já acontecia no trabalho anterior, o primeiro operando continua a ser proveniente do registo A. Contudo, agora o operando B não é sempre o valor X definido pela posição dos 8 interruptores como anteriormente. O valor de B pode ser 1 de 8 valores possíveis. Esta escolha é feita pelo multiplexador de 8 para 1 que, dependendo do valor do sinal SEL, coloca na sua saída um dos 8 valores como mostra a tabela 2. As constantes nas entradas 5 e 6 do multiplexador poderão ser úteis para a realização de alguns cálculos, conforme verificará posteriormente. Também C (*flag* CARRY) na entrada 7, será útil, por exemplo, para realizar adições com mais de 8 bits.

Tabela 2 – Valores que podem constituir o operando B da ALU

SEL	Operando B
000	Registo A
001	Registo R1
010	Registo R2
011	Registo R3
100	Valor externo X (definido com os interruptores)
101	Constante de 8 bits igual a 1
110	Constante de 8 bits igual a 15
111	Valor da *flag* de *carry* (zero ou um)

O resultado R produzido pela ALU poderá agora ser encaminhado para 1 de 4 registos possíveis: A, R1, R2 e R3. Uma vez que todos os registos têm a sua entrada ligada à saída da ALU, e como não pretendemos que todos eles guardem em simultâneo esse valor, os registos possuem uma entrada de habilitação 'en' (*enable*) que só permite a memorização do resultado da operação se estiver ativa (en=1). Os 4 sinais que controlam o *enable* dos registos são gerados por um descodificador de 2 para 4, a partir de um sinal CE de 2 bits, selecionando um e um só registo, tal como indicado na tabela 3. O registo selecionado será aquele que irá memorizar o valor presente na sua entrada (o resultado R produzido pela ALU).

Tabela 3 – Registo que guardará o resultado produzido pela ALU

CE	Registo cujo *enable* estará ativo
00	Registo A
01	Registo R1
10	Registo R2
11	Registo R3

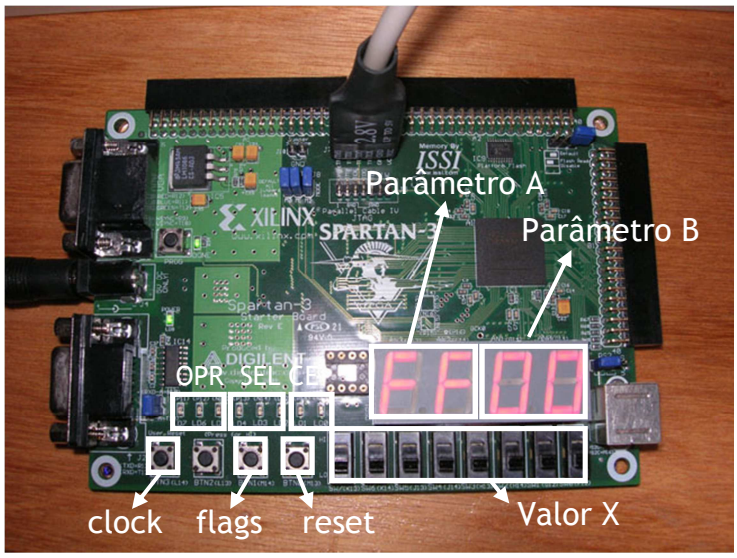

Figura 2 – Entradas e saídas do circuito na placa S3board.

A figura 2 mostra como este novo circuito pode ser operado na placa S3board. O valor de X continua a ser introduzido nos 8 interruptores. Tal como no projeto anterior, os visualizadores de 7 segmentos apresentam o valor hexadecimal dos operandos A e B da ALU (note-se que agora o operando B não é necessariamente o valor X). O botão btn3 continua a servir de sinal de relógio (clock). Para limpar o conteúdo de todos os registos existe agora um sinal, 'reset', controlado pelo botão btn0, que liga a todos os registos (estas ligações não foram representadas na figura 1 apenas por questões de simplicidade da mesma).
Porém, faltam-nos botões para definir os restantes sinais: 3 bits para OPR, 3 bits para SEL e 2 bits para CE. Estes 8 bits serão por isso introduzidos recorrendo a uma aplicação a executar no PC (LSDI_Lab4.exe), que encaminhará os valores lógicos até à placa através de um cabo próprio. A figura 3 mostra a interface desta aplicação, onde 8 botões permitem definir o valor dos referidos sinais. Estes 8 bits são também visualizados nos 8 LEDs da placa como a figura 2 mostra, pelo que agora, para se saber o valor das *flags* O, C, S e Z (nos 4 LEDs da direita tal como acontecia no trabalho anterior) teremos que premir o botão btn1.

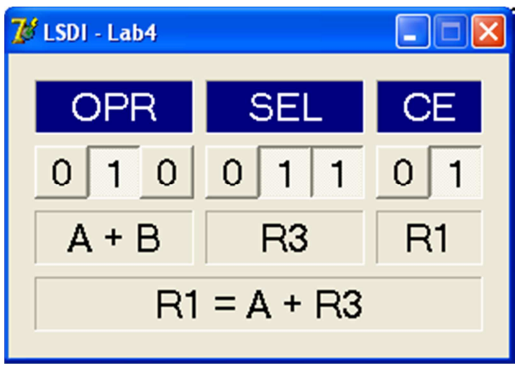

Figura 3 – Interface da aplicação para definição dos sinais OPR, SEL e CE; a última linha indica a operação que será realizada com os bits selecionados para OPR, SEL e CE. Neste exemplo, quando o sinal de relógio passar de zero para 1 (pressionando o botão btn3) o registo R1 será carregado com o resultado da adição do valor que está no registo A com o valor que está no registo R3.

2. Projeto

Em termos do conteúdo do projeto, a sua intervenção neste trabalho laboratorial é mínima: na aula, apenas lhe será indicado que complete o esquema referente à figura 1. Para tal, deverá preparar-se no sentido de entender o objetivo e a função dos sinais e dos blocos presentes.
Porém, ao nível da utilização do circuito, é fundamental que prepare a forma de realizar os cálculos descritos na última secção deste guião, consistindo em criar sequências de operações elementares para a execução das tarefas definidas (tabela 4).

3. Implementação

Ao iniciar a aula laboratorial, abra o projeto LSDI_Lab4.ise e comece por editar o módulo datapath.sch. O esquema nele contido contém todos os blocos necessários, assim como todas as entradas e saídas (cujos nomes não deve alterar). Pretende-se apenas que o complete com as ligações em falta.

Seguidamente, implemente o circuito na placa S3board, seguindo os mesmos passos dos trabalhos anteriores. Não esqueça que nem todas as entradas são definidas diretamente na placa, pois como foi descrito, alguns sinais são definidos com o auxílio da aplicação LSDI_Lab4.exe a executar no PC.

4. Ensaio

A tabela 3 mostra um conjunto de tarefas a efetuar, devendo o estudante indicar, para cada uma delas, a sequência de operações elementares a realizar na ALU. Estas operações deverão ser expressas através de linguagem simbólica mas acompanhadas agora pelos 8 bits correspondentes aos sinais OPR, SEL e CE usados em cada uma delas. Por exemplo, "colocar no registo R1 o dobro do valor X", resulta em:

```
Operação            OPR SEL  CE
A  ← X;             000 100 00
R1 ← A << 1;        101 000 01
```

Esta tarefa necessita pois de duas operações. De notar que na segunda, o valor de SEL podia ser de facto qualquer já que a operação de deslocamento não usa o operando B.

Tabela 4 – Tarefas a realizar

Tarefa	Descrição
1	Incrementar em uma unidade o valor de R3
2	Colocar em R1 o valor de R3
3	Colocar em A o valor 0 (sem usar o X!)
4	Colocar em R2 o simétrico de R1
5	Colocar em R3 a negação de R2
6	Colocar em R1 o valor de R2−A
7	Colocar em R1 o valor de A−R2
8	Colocar em R2 o valor do nibble[1] mais significativo de R3
9	Colocar em R3 o produto de X por 10
10	Colocar nos registos {A, R1} a soma dos valores de 16 bits guardados nos registos {A, R1} e {R2, R3}

[1] Conjunto de 4 bits.