Instituto Superior Técnico – Universidade de Lisboa

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Ano Lectivo 2014 / 2015

Lisboa, 19 de Setembro de 2014

Projecto de Sistemas Digitais

# PROJECTO DE SISTEMAS DIGITAIS

## PROJECTO 1

## UNIDADE LÓGICO-ARITMÉTICA SIMPLES

Grupo nº21

David Romão Fialho, nº 73530

Ricardo Filipe Amendoeira, nº73373

# 1 ÍNDICE

# Unidade Lógico-Aritmética Simples

## 2 INTRODUÇÃO

Neste trabalho tínhamos como objetivo realizar o projeto de uma calculadora lógico-aritmética capaz de executar um grupo de operações simples. A calculadora é composta por 3 unidades principais a Unidade de Controlo (UC) a Unidade de Dados (UD) e a Unidade de Interface.

Permite fazer adições, subtrações, multiplicações, shift's aritméticos para a direita e ainda uma operação lógica XOR. Tem ainda 2 registos para guardar os valores a utilizar nessas operações.

## 3 ESPECIFICAÇÃO DO FUNCIONAMENTO

A calculadora lógico-aritmética foi implementada com recurso aos 4 botões de pressão, os 8 interruptores e os 4 *displays* de 7 segmentos disponíveis na FPGA.

As operações da calculadora são feitas entre os valores de 2 registos, R1 e R2. O resultado de cada operação é guardado no registo R2, este resultado está limitado ao intervalo [-4095; 4095], caso este limite seja excedido (*overflow*) o registo R2 é reinicializado. O valor a armazenar nos registos é introduzido com recurso aos 7 interruptores mais à direita em formato sinal-módulo, este valor está limitado ao intervalo [-63; +63]:

| I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 |
|---|---|---|---|---|---|---|---|
| Seleção do registo R1/R2 | Sinal | MSB | … | … | … | … | LSB |

*Tabela 1*

O primeiro interruptor permite selecionar qual dos registos é apresentado nos displays de 7 segmentos:

| I8 | Display |
|---|---|
| Ativo | R1 |
| Desativo | R2 |

*Tabela 2*

A seleção da operação a efetuar é feita através dos mesmos interruptores seguindo a seguinte configuração:

| Interruptores | | | | | | | Operação | |
|---|---|---|---|---|---|---|---|---|
| I6 | I5 | I4 | I3 | I2 | I1 | I0 | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | Soma | R2 <- R2 + R1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | Subtração | R2 <- R2 - R1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | Multiplicação | R2 <- R2 * R1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | XOR | R2 <-R2 xor R1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | *Shift-Right* Aritmético | R2 <- R2 sra (R1%8) |

*Tabela 3*

Os 4 botões de pressão funcionam da seguinte forma:

| Botão 3 | Botão 2 | Botão 1 | Botão 0 |
|---|---|---|---|
| Reset | Load R1 | Load R2 | Operação |

*Tabela 4*

Para realizar uma operação entre dois valores são necessários os seguintes passos:

1) Introduzir valor a armazenar em cada registo (Interruptores, tabela 1)
2) Carregar valor para o registo apropriado (Botão 2 e 1, tabela 4)
3) Selecionar operação a realizar (Interruptores, tabela 3)
4) Executar operação especificada (Botão 0, tabela 4)

Quando o botão 3 é pressionado o sistema é reinicializado e os registos são reinicializados a zero.

Optou-se que um botão ao ser premido apenas uma operação é realizada até que este seja libertado. Esta decisão foi tomada porque se quis projetar uma calculadora o mais próximo possível da realidade.

# 4 Unidade de Controlo

A Unidade de Controlo é uma máquina de estados que gere o funcionamento geral da calculadora. Tem como entradas o sinal gerado pelos últimos 3 botões de pressão mais à direita da FPGA, a partir dos mesmos o sinal de saída é gerado. Este sinal indica como a unidade de dados se deve comportar. A máquina de estados tem 4 estados:

- Idle (INI) – em espera que o utilizador pressione um botão de pressão
- Load R1 – guardar entrada no registo R1
- Load R2 – guardar entrada no registo 2
- Operação – executar uma das operações baseando-se nos três últimos interruptores da direita.
- Wait for Release (END) – em espera que o utilizador depressione o botão para evitar fazer operações acidentais devido a manter o botão pressionado.

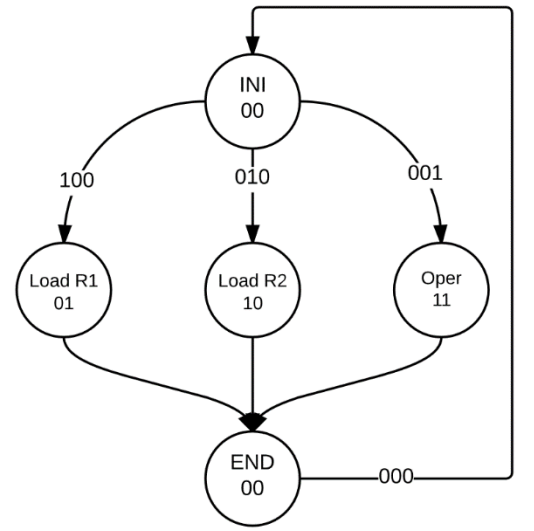
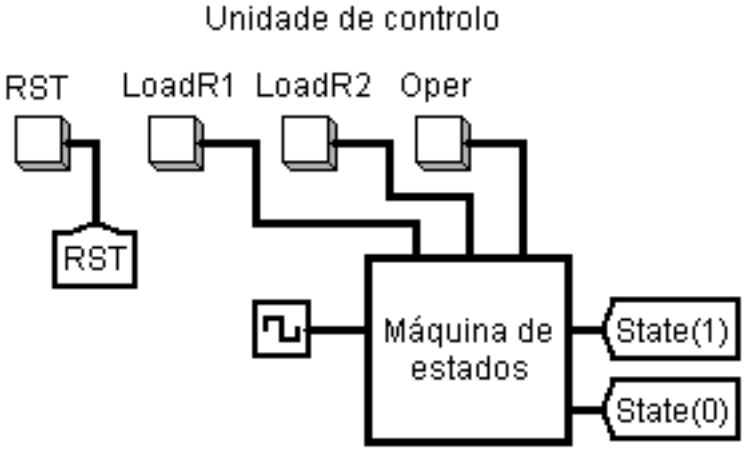



*Figura 1: Máquina de estados*

Abaixo encontra-se uma tabela com a relação entre as entradas e as saídas, bem como uma imagem do circuito da unidade de controlo:

| | Saída | |
|---|---|---|
| Estado | State1 | State0 |
| Operação | 1 | 1 |
| *Load* R2 | 1 | 0 |
| *Load* R1 | 0 | 1 |
| *Idle* | 0 | 0 |
| *End* | 0 | 0 |

*Table 5*



# 5 UNIDADE DE DADOS

A Unidade de Dados é constituída por uma ALU e 2 registos, tem como entradas os sinal gerado pelos 8 interruptores da FPGA, o sinal de reset vindo do botão 0 e os 2 bits de estado fornecidos pela unidade de controlo; na saída é colocado conteúdo do registo R1 ou R2. Esta seleção é feita pelo bit mais significativo do sinal de entrada.

O sinal de entrada é imediatamente convertido de sinal-módulo para complemento para 2 sem qualquer verificação acerca da operação a realizar, pois este valor só pode ter dois significados: ou é para ser armazenado e a conversão tem que ser feita de forma a facilitar as operações aritméticas, ou é utilizado para identificar a operação da ALU a selecionar e nesse caso os valores de entrada devem ser positivos e por isso são representado de igual forma em complemento para dois e em sinal-módulo. A conversão inversa é feita na saída da Unidade de Dados.

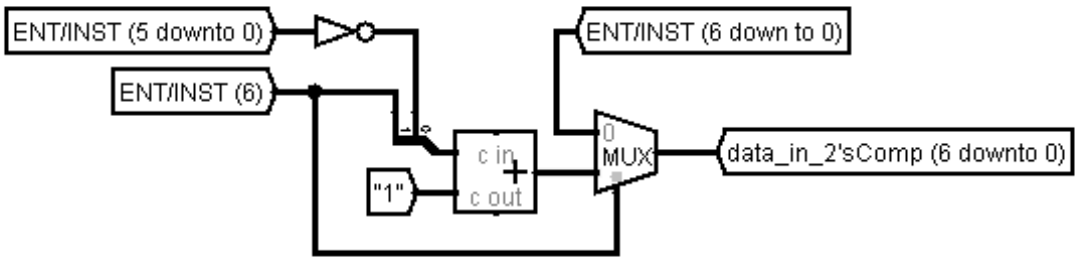
Complemento para 2 -> Sinal-módulo:

$$\text{input\_CP2} <= \text{data\_in when data\_in}(6) = '0' \text{ else } (('1'\&(\text{not data\_in}(5 \text{ downto } 0))) + 1)$$

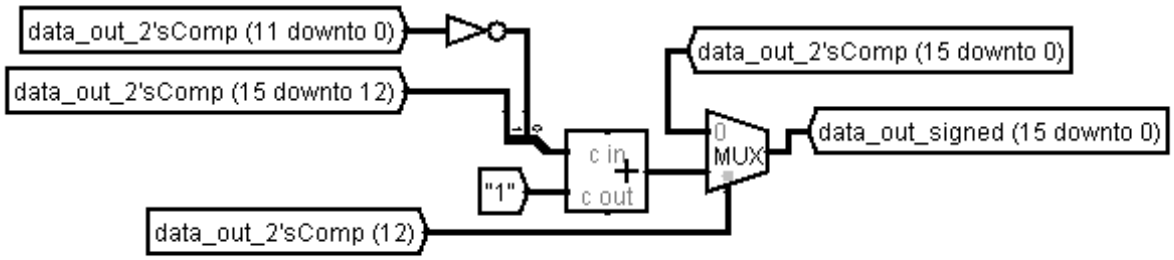
Sinal-módulo -> Complemento para 2:

$$data\_out <= ('1'\&(not(out\_CP2(11\ downto\ 0)))) + 1\ when\ out\_CP2(12) = '1'\ else\ out\_CP2$$

Conversão de sinal-módulo para complemento para 2

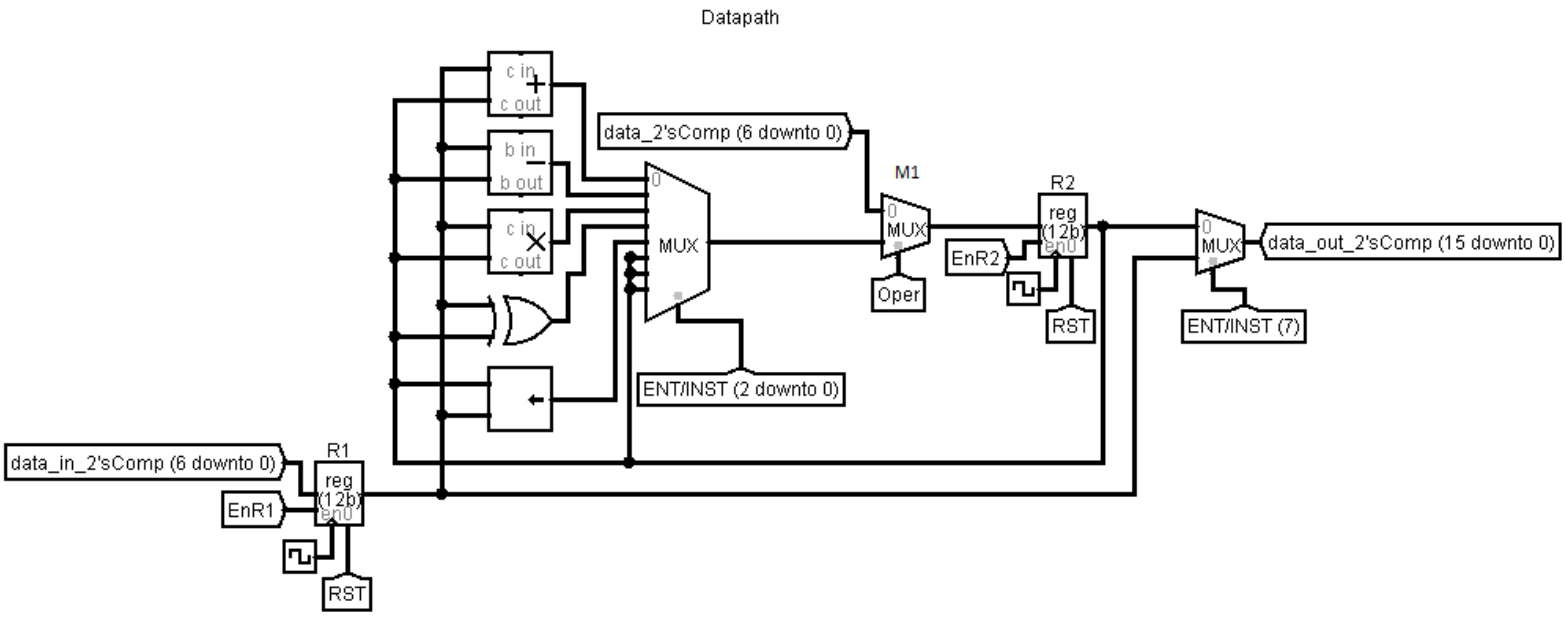



Conversão de complemento para 2 para sinal-módulo



O registo R1 armazena o valor da entrada limitada ao intervalo [-63; +63], este intervalo de valores pode ser representado com recurso a 7 bits logo essa será a dimensão do registo R1. No caso de R2, este pode armazenar o valor de entrada mas também armazena o resultado das operações, este resultado tem uma resolução máxima de [-4095; +4095] sendo necessário que este registo seja capaz de armazenar 13 bits. Quando o sinal de entrada é armazenado em R2 o bit de sinal é estendido para que tenhamos o mesmo valor representado em 13 bits em vez de 7 bits.

Abaixo é apresentado um esquema geral do circuito da unidade de dados:



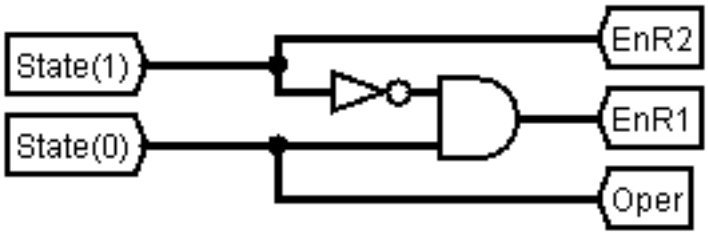

Os dois bits de estado fornecidos pela UC configuram os *enables* dos registos e o bit de seleção do multiplexer M1, de forma a obter a o funcionamento desejado. Quando se pretende guardar a entrada num dos registos o *enable* do respetivo registo deve ser o único ativado e no caso do registo R2 o seletor do multiplexer M1 deve estar colocado a '0' de forma a selecionar o sinal de entrada como o sinal a armazenar no registo. Quando se pretende fazer uma operação o seletor do multiplexer M1 de estar a '1' para selecionar a saída da ALU como entrada do registo R2 e o respetivo *enable* ativado. Esta configuração encontra-se na tabela abaixo:

| Descrição | State1 | State0 | *Enable* R1 | *Enable* R2 | Selector M1 |
|---|---|---|---|---|---|
| Operação | 1 | 1 | 0 | 1 | 1 |
| *Load* R2 | 1 | 0 | 0 | 1 | 0 |
| *Load* R1 | 0 | 1 | 1 | 0 | X |
| *Idle/End* | 0 | 0 | 0 | 0 | X |

A partir desta tabela da verdade é possível retirar as expressões lógicas para os *enables* e o seletor que requerem menos *hardware*:

$$EnR1 = \overline{State_1}.State_0$$

$$EnR2 = State_1$$
$$S1\ (Oper) = State_0$$




A unidade aritmética e lógica (ALU) é a parte central da unidade de dados pois trata-se do bloco responsável pela realização de todas as operações lógico-aritméticas, contém um somador, um subtrator, um multiplicador, uma porta XOR e um *shifter*.

A operação de *Shift-Right* Aritmético nesta calculadora corresponde a serem feitos tantos *Shift-Right* Aritméticos quantos o resto da divisão do valor em R1 ditar. O resto da divisão por 8 corresponde ao número obtido (sem sinal) pelos 3 bits menos significativos de R1. A implementação do SRA é feita tendo em conta todos os casos possíveis [0; 7] *shifts*, posteriormente um multiplexer é utilizado para selecionar o resultado que se pretende com base no valor do resto.

Após cada operação aritmética é feito um teste ao resultado para verificar se ocorreu *overflow*, caso esta situação se verifique o registo R2 é reinicializado a zero. Este teste é feito no caso da soma e da subtração com recurso a um bit adicional, sendo que o resultado dessas operações um sinal de 14 bits, caso os 2 bits mais significativos sejam diferentes isto significa que ocorreu *overflow*. No caso da multiplicação como é feita entre um sinal de 7 bits e outro de 13 bits o resultado é um sinal de 20 bits. Para testar a ocorrência de overflow verifica-se se os 9 bits mais significativos são todos iguais, isto significa que não ocorreu *overflow* e por isso o resultado deve ser armazenado em R2. O resultado da multiplicação tem sempre de ser truncado a 13 bits antes de ser armazenado.

# 6 SIMULAÇÕES EXEMPLO

Foram efetuadas duas simulações para demonstrar o funcionamento do circuito, uma soma simples e uma série de multiplicações que causam um overflow do registo de resultado R2.

Na simulação 1 é feita uma soma. A ordem de eventos é a seguinte:

1. O Register Select é posto a high para ver o conteúdo de R1 na Data Out

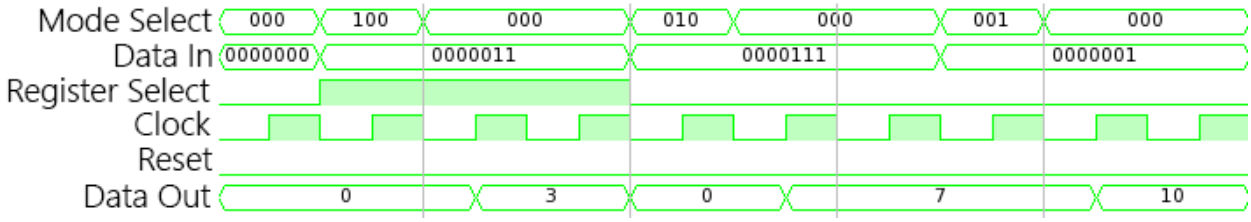
2. É carregado o valor “3” para R1, colocando o valor na Data In e pressionando o botão da esquerda

3. O Register Select é posto a low para ver o conteúdo de R2 na Data Out

4. É carregado o valor “7” para R2, colocando o valor na Data In e pressionando o botão do meio

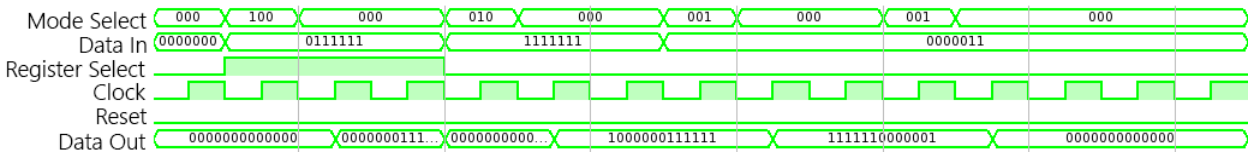
5. É efectuada a operação de soma colocando o valor “1” na Data In e pressionando o botão da direita

6. O resultado da soma, “10”, é guardado em R2




Na simulação 2 são feitas várias multiplicações até obter um overflow, altura em que se faz reset a R2:

1. É carregado o valor “63” para R1, colocando o valor na Data In e pressionando o botão da esquerda
2. É carregado o valor “-63” para R2, colocando o valor na Data In e pressionando o botão do meio
3. É efectuada a operação de multiplicação colocando o valor “3” na Data In e pressionando o botão da direita
4. O resultado, “-3969”, é guardado em R2
5. É efectuada a operação de multiplicação colocando o valor “3” na Data In e pressionando o botão da direita
6. O resultado, “-250047”, não cabe em R2, causando um overflow. É feito um reset automático a R2



# 7 ANEXO: DIAGRAMA DE BLOCOS