INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Licenciatura em Engenharia Informática e de Computadores

e

Licenciatura em Engenharia Informática, Redes e Telecomunicações

Relatório do 4.º Laboratório

de

Lógica e Sistemas Digitais

Circuito Dados/Controlo

19 de janeiro de 2022

1 Introdução

Este quarto e último laboratório para avaliação de Lógica e Sistemas Digitais, tem como objetivo analisar e projetar circuitos com processamento de dados e controlo e implementar este circuito na placa de desenvolvimento DE10-Lite da Intel.

Este circuito tem como propósito realizar uma multiplicação de M por *m*, ambos números de 4 bits com sinal positivo, entre 0 e 9.

Para realizar esta multiplicação, vamos recorrer à multiplicação por somas sucessivas, ou seja, vamos somar *m* vezes o número M, até obtermos o resultado pretendido.

Este circuito tem também um botão de Start que deve ser ativado após a escolha dos operandos ter sido realizada. Outra operação só ocorre quando o sinal de Start é desativado.

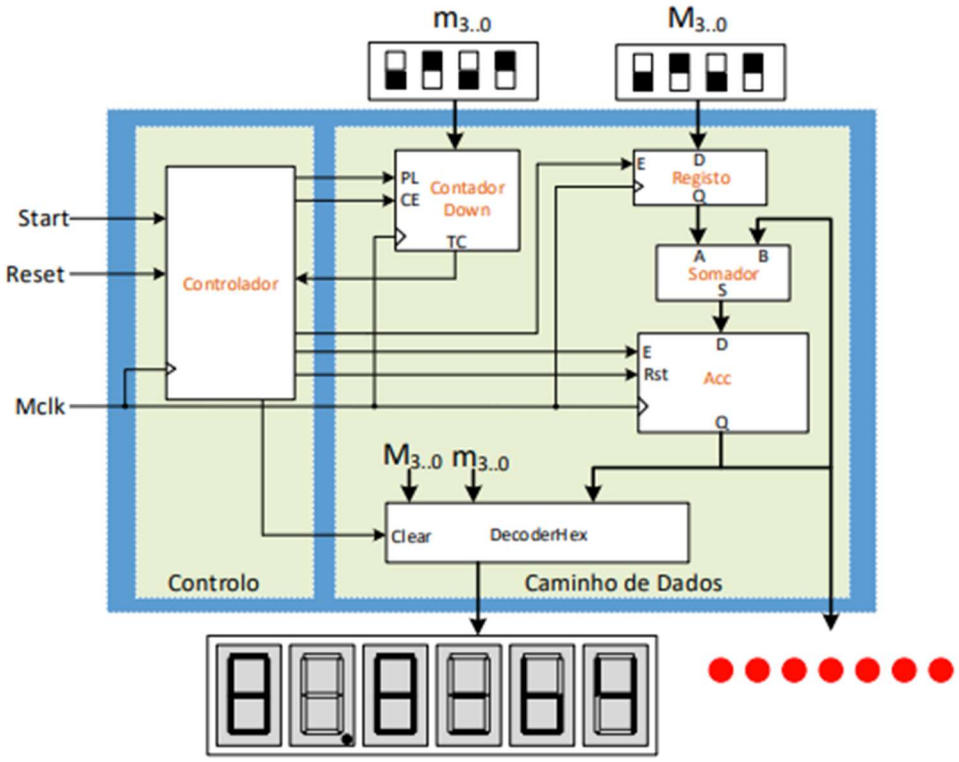

Imagem 1 – Diagrama de blocos do multiplicador por somas sucessivas

2 Desenvolvimento do Trabalho

Este circuito é formado por dois módulos principais, o módulo Caminho de Dados e o módulo controlo.

Primeiro, começámos por desenvolver o módulo Caminho de Dados. O mesmo é constituído por 5 componentes: Um Contador Down, um Registo, um Somador, um Acc e um DecoderHex, que é fornecido pelo docente. Este módulo tem como entrada M e *m* que são os números a serem multiplicados e como saída tem leds que vão apresentar o resultado da multiplicação em binário e o ecrã que vai apresentar M, *m* e o resultado da multiplicação.

2.1 Módulo Caminho de Dados

O módulo caminho de dados tem o seguinte aspeto:

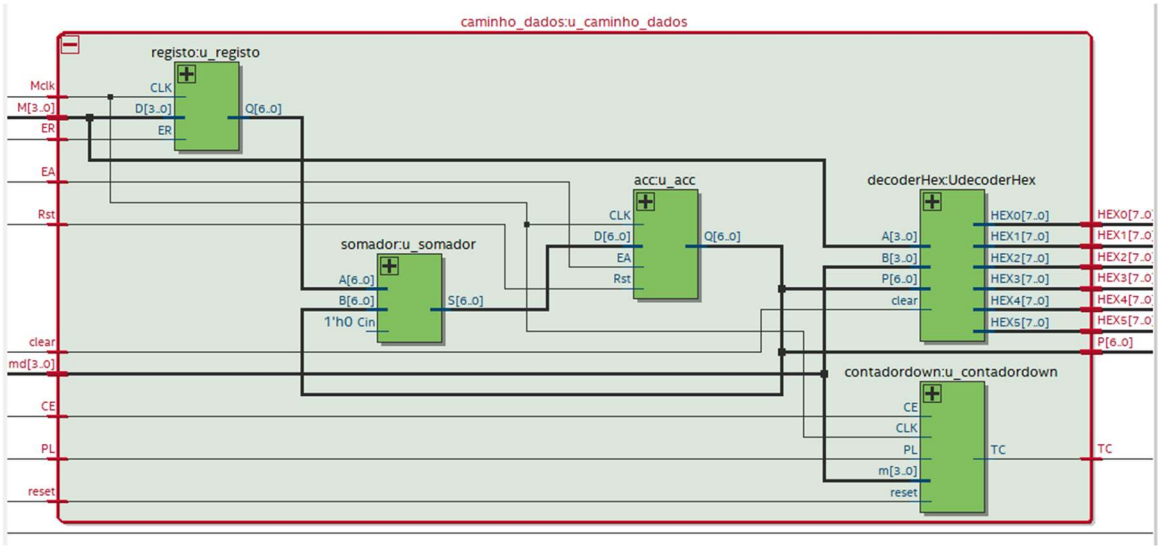

Imagem 3 – Módulo Caminho de Dados

2.1.1 Contador Down

Este componente foi desenvolvido no Lab anterior e tem como objetivo fazer a contagem decrescente, até 0, a partir de *m*. Tem como entrada *m* (um número que queremos multiplicar), PL, CE e um Clock. Tem como saída um Terminal Count, que deverá ficar ativo quando a contagem acabar.

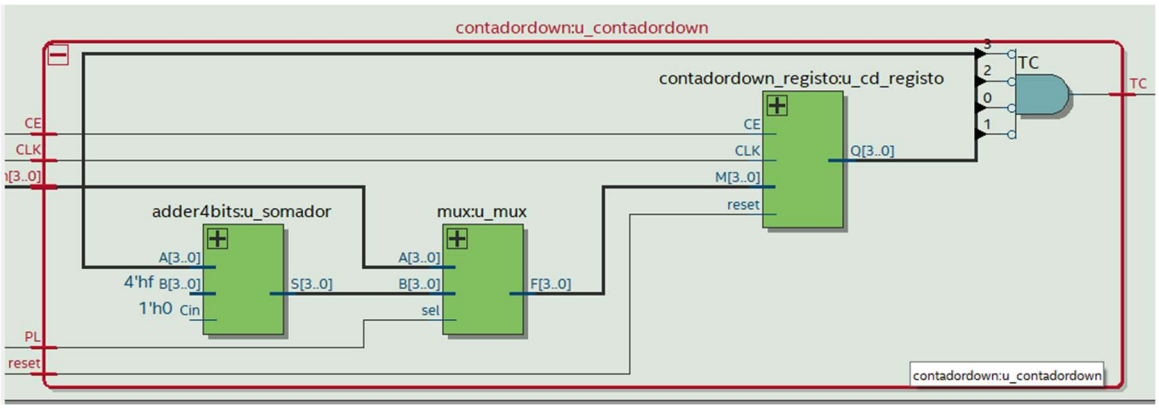

Imagem 2 – Contador down

2.1.2 Registo do caminho de dados

Este componente é composto por 7 flip flops, sendo que a soma vai ser efetuada entre dois números de 7 bits. Recebe como entrada M (número a ser multiplicado por *m*), um Enable e um Clock. Tem como saída um Q que é um número de 7 bits. No registo foi onde foi feita a extensão de sinal de 4 bits para 7. Para tal usamos 7 flip flops em que 3 deles serviram para fazer a extensão de sinal, colocando 3 bits de maior peso a 0 e os restantes bits são os que foram introduzidos pelo utilizador.

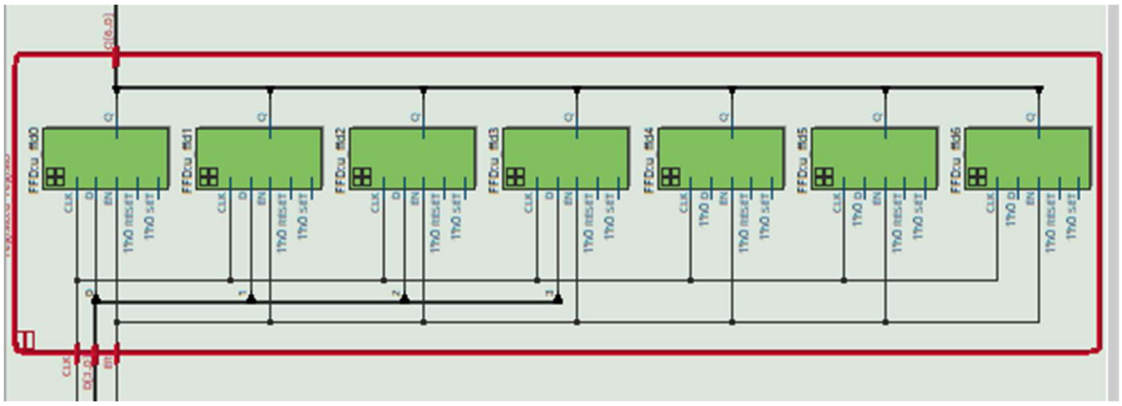

Imagem 4 – Registo

2.1.3 Somador

Este componente recebe A e B como entrada, ambos de 7 bits e é responsável pelas somas consecutivas que vão ser realizadas. As somas vão ser feitas entre dois números de 7 bits sendo que, o resultado máximo da soma possível a realizar é 9 * 9 = 81 e para representar 81 em binário é necessário 7 bits.

Internamente tem 7 full adders.

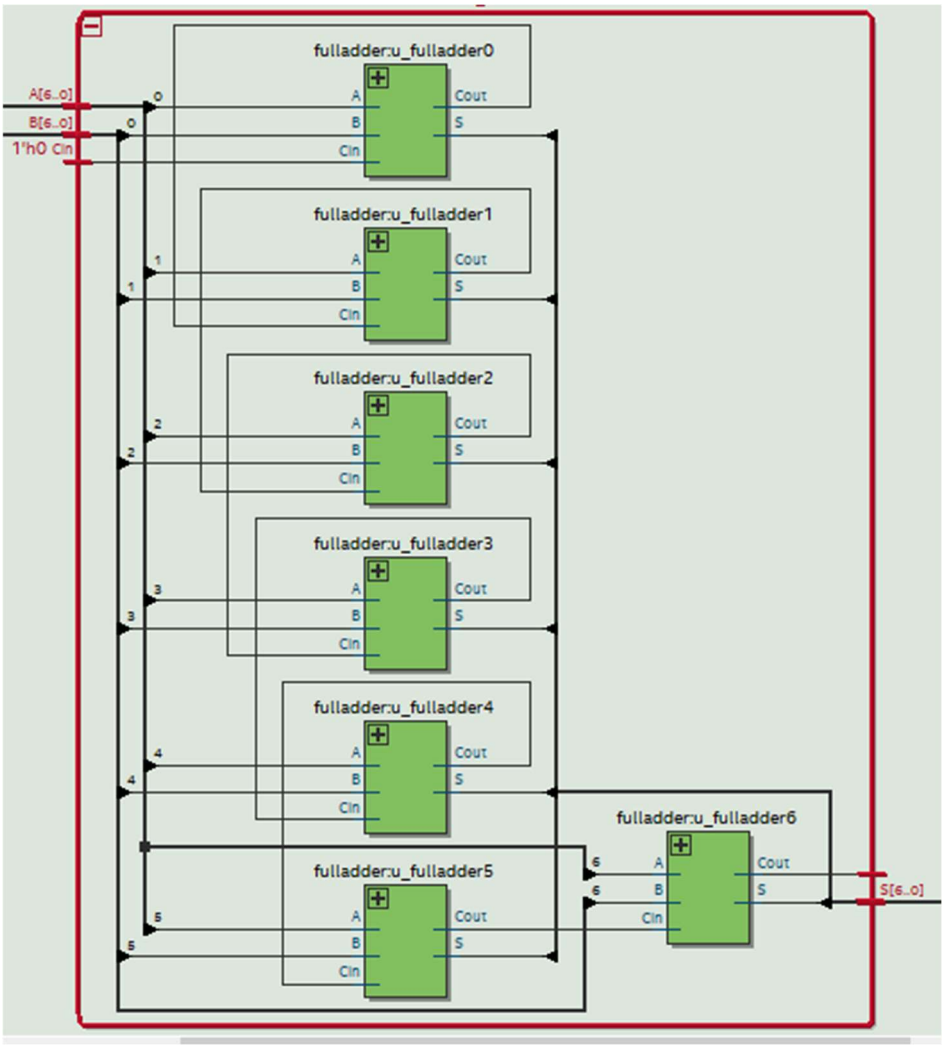

Imagem 5 – Somador

2.1.4 Acc

Este componente guarda a soma atual, por exemplo, na multiplicação 3*3. Primeiro ocorre a soma 3+3=6, é guardada no Acc e volta a entrar no somador, desta vez a soma irá ser 3+6=9. Como a soma já terminou, não volta a entrar no somador novamente.

Este componente tem como entrada D, que corresponde à soma anterior, um Enable que determina quando a multiplicação terminou e um Reset. Como saída tem Q que ou volta para o somador se a multiplicação ainda não tiver terminado ou, quando a multiplicação já tiver terminado, Q vai conter o resultado da multiplicação que vai ser apresentado no display e nos leds.

É composto por 7 flip flops.

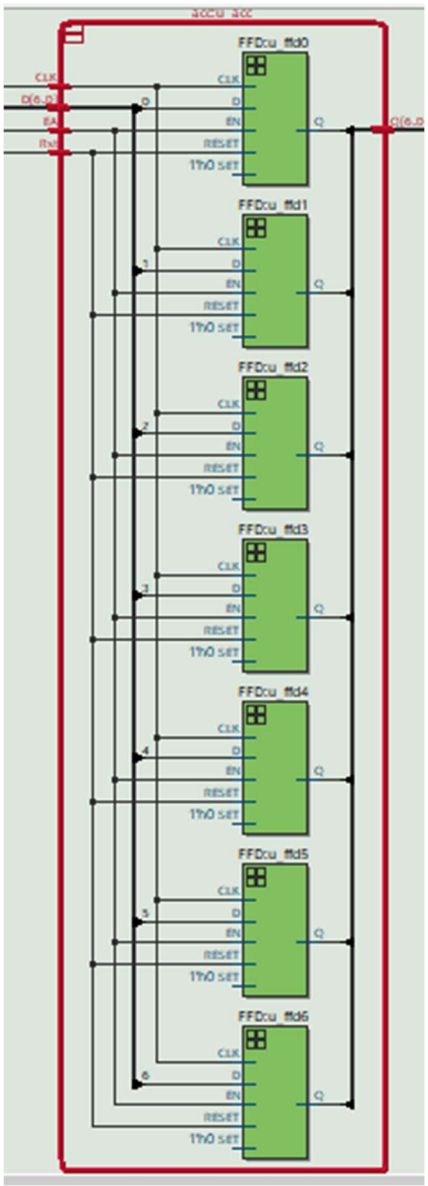

Imagem 6 - Acc

2.1.5 DecoderHex

Este componente é fornecido pelo docente. É responsável pela descodificação dos números binários para o display da placa intel DE-Lite da Intel.

2.2 Componente Controlador

Para começar, determinamos o ASM Chart:

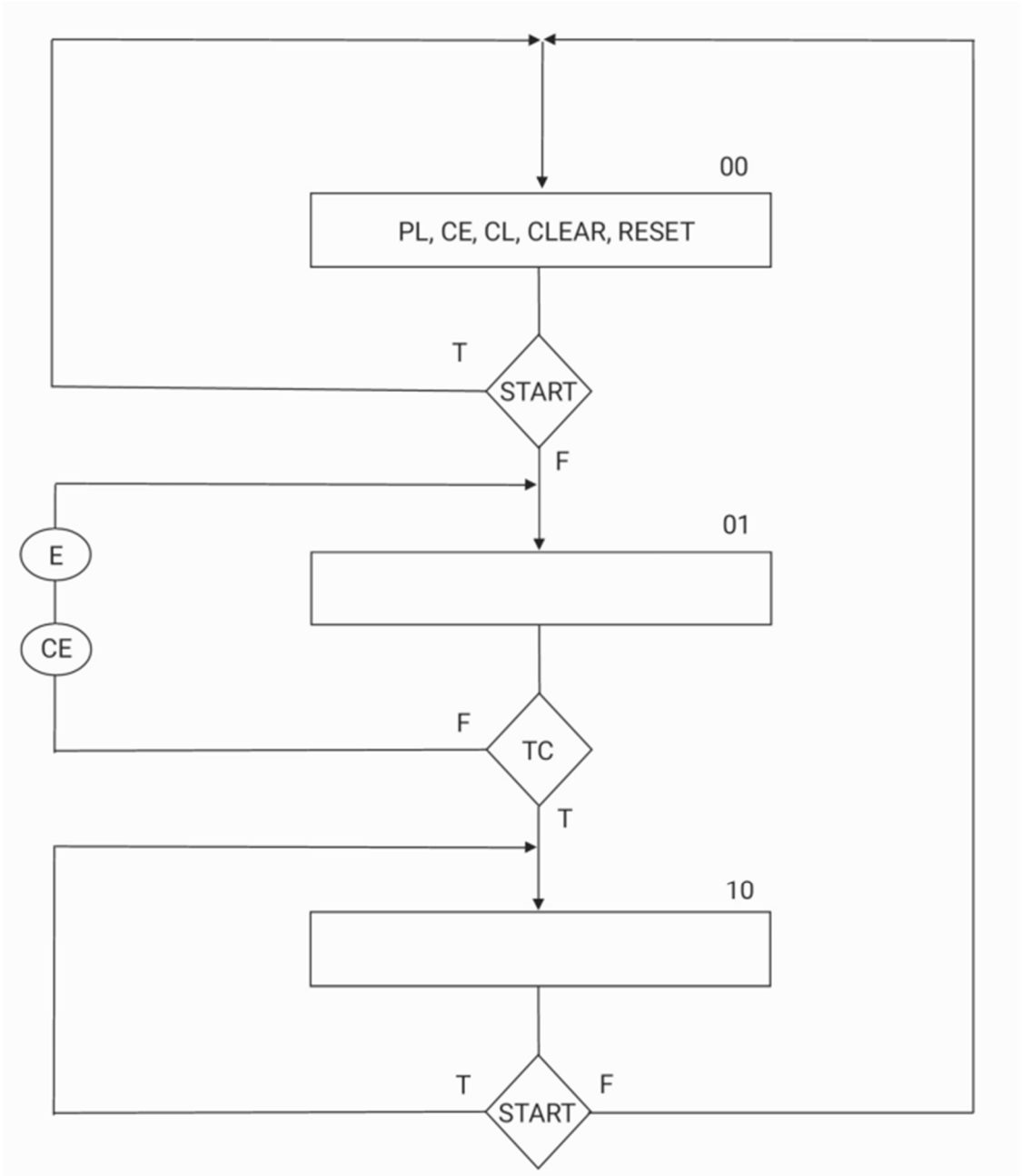

Imagem 7 – ASM Chart

De seguida, determinamos a tabela para posteriormente projetarmos o controlador com controlo microprogramado.

EP	Start	TC	ES	PL	Rst	Clear	CE	EA	ER
00	0	0	01	1	1	1	1	0	1
00	0	1	01	1	1	1	1	0	1
00	1	0	00	1	1	1	1	0	1
00	1	1	00	1	1	1	1	0	1
01	0	0	01	0	0	0	1	1	0
01	0	1	10	0	0	0	0	0	0
01	1	0	01	0	0	0	1	1	0
01	1	1	10	0	0	0	0	0	0
10	0	0	10	0	0	0	0	0	0
10	0	1	10	0	0	0	0	0	0
10	1	0	00	0	0	0	0	0	0
10	1	1	00	0	0	0	0	0	0

Este componente é composta por um registo e pela ROM.

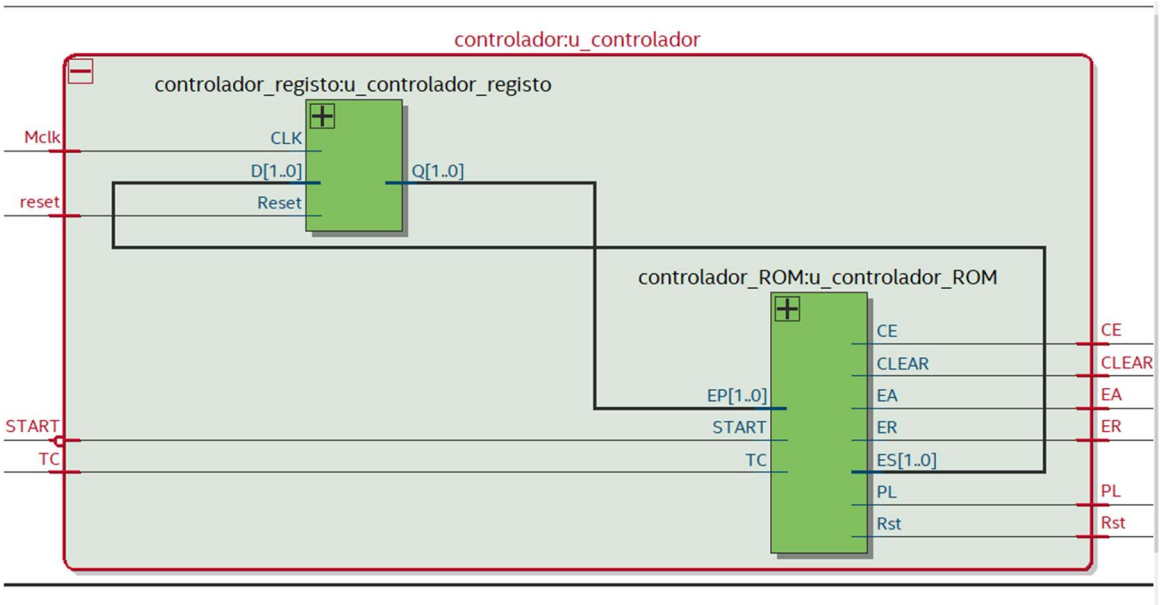

Imagem 8 – Módulo Controlo

2.2.1 Registo

Este componente tem como entrada um Clock, um D de 2 bits, que corresponde ao estado seguinte e um Reset. Como saída tem um Q também de 2 bits que corresponde ao estado atual. É composto por 2 flip flops que guardam os estados.

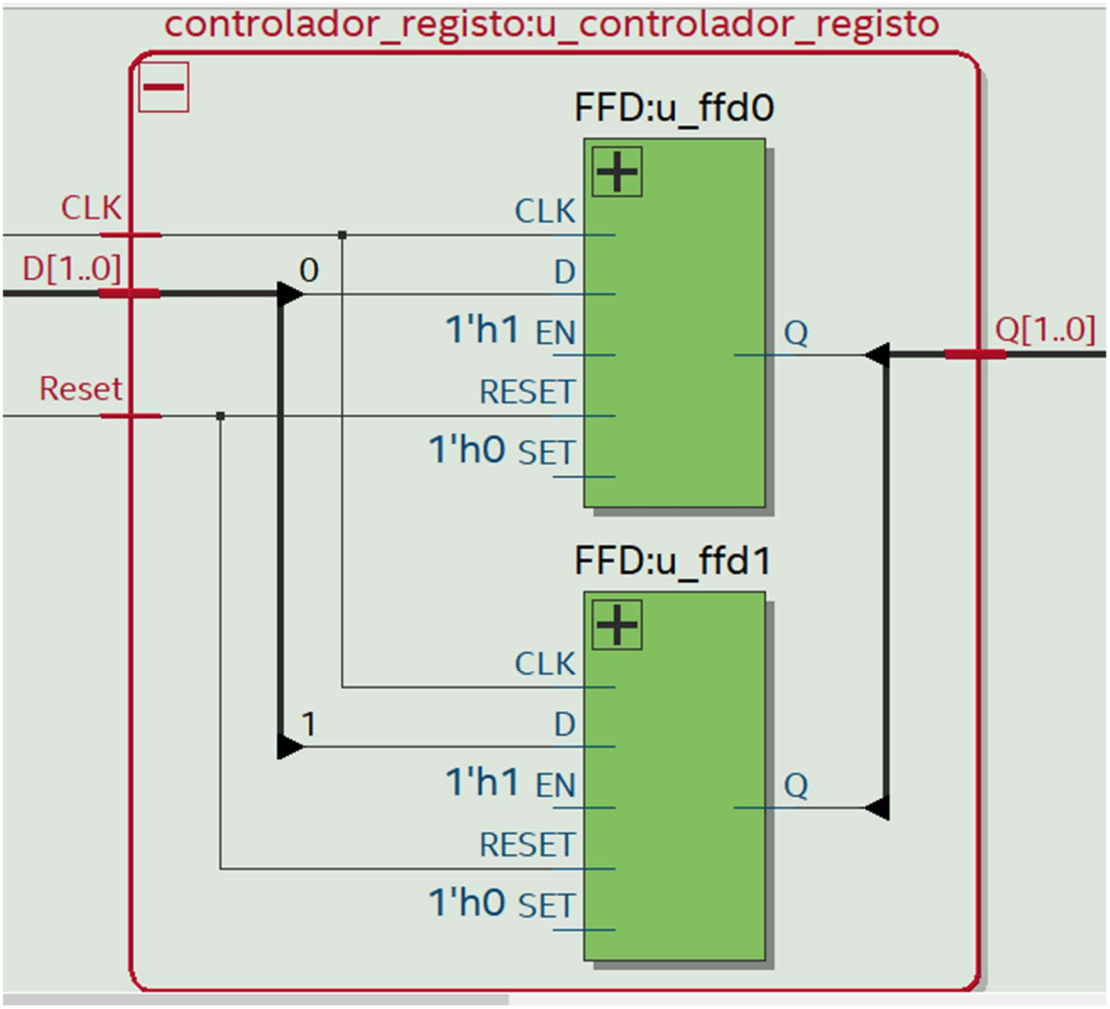

Imagem 9 – Registo do controlador

2.2.2 ROM

O ROM tem como entrada um EP (estado presente) um Start e um TC (terminal count) e tem como saída um CE, um Clear, um EA (enable do Acc), um ER (enable do registo), um ES (estado seguinte), um PL e um Reset. O ROM é responsável pela mudança de estados, ou seja, vai alterar os seus outputs (saídas) consoante o estado presente.

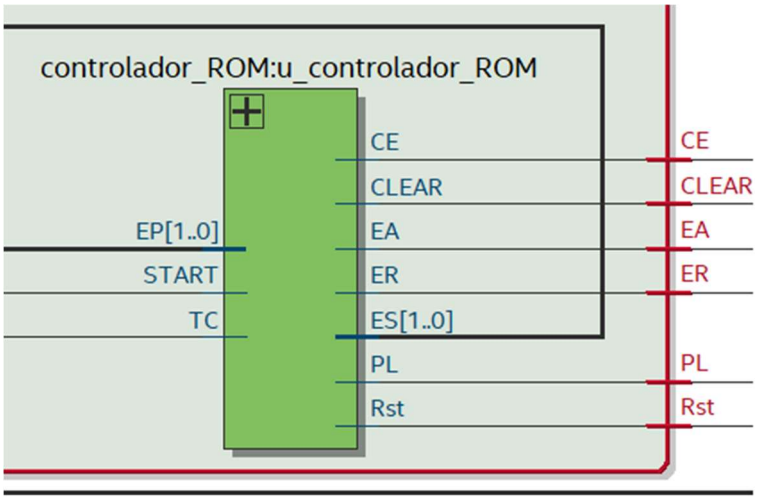

Imagem 10 – ROM

```
19
20      address <= EP & START & TC;
21
22      ES <= data(7 DOWNTO 6);
23      PL <= data(5);
24      Rst <= data(4);
25      CLEAR <= data(3);
26      CE <= data(2);
27      EA <= data(1);
28      ER <= data(0);
29
30      data <= "01111101" WHEN address = "0000" ELSE
31          "01111101" WHEN address = "0001" ELSE
32          "00111101" WHEN address = "0010" ELSE
33          "00111101" WHEN address = "0011" ELSE
34          "01000110" WHEN address = "0100" ELSE
35          "10000000" WHEN address = "0101" ELSE
36          "01000110" WHEN address = "0110" ELSE
37          "10000000" WHEN address = "0111" ELSE
38          "10000000" WHEN address = "1000" ELSE
39          "10000000" WHEN address = "1001" ELSE
40          "00000000" WHEN address = "1010" ELSE
41          "00000000";
42
43  END logicFunction;
```

Imagem 11 – função lógica do ROM

Esta função lógica do ROM foi criada a partir da tabela, que por sua vez foi criada a partir do ASM Chart. Os valores da saída dependem do estado atual e são alterados consoante os valores de EP (estado presente), Start e TC (terminal Count).

3 Montagem e teste laboratorial

Após desenvolvermos os módulos necessários à realização deste Lab, passámos à montagem da entidade de topo, onde fizemos as ligações necessárias entre os dois módulos principais, o módulo controlo e o módulo caminho de dados de modo a obtermos um circuito funcional e com o objetivo de ser testado na placa de desenvolvimento Intel DE-Lite da Intel.

A entidade de topo tem como entradas M e *m*, que são números de 4 bits escolhidos pelo utilizador para a multiplicação, um Start, um Reset e um Clock. Como saídas a entidade de topo tem 6 HEX (de 0 a 5) que irá apresentar no display digital da placa, os números M e *m* e o resultado da sua multiplicação. Por fim também tem uma saída P, que é um número de 7 bits correspondente ao resultado da multiplicação representado nos leds da placa.

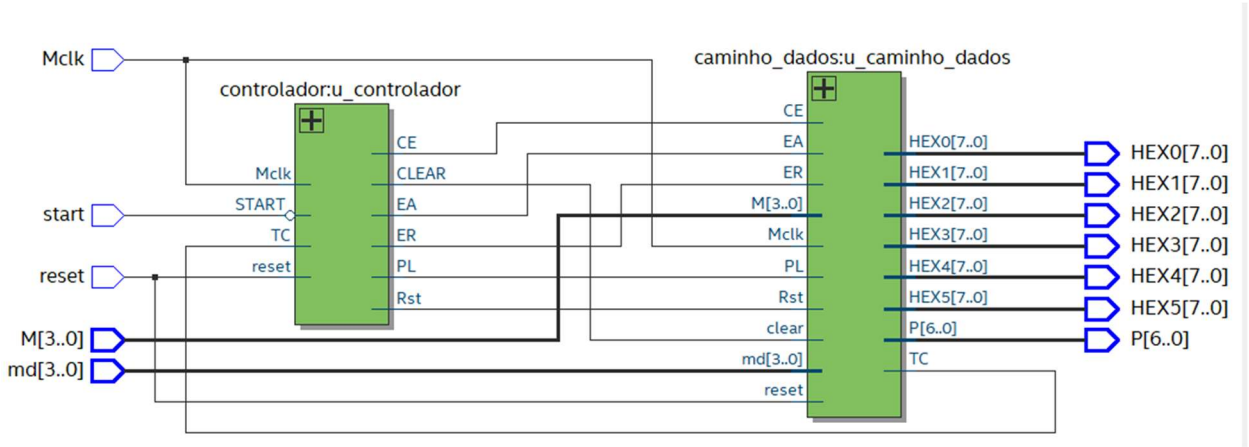

Imagem 12 – Entidade de topo

Para podermos testar este circuito na placa, fizemos a seguinte atribuição de pinos:

```
#input
set_location_assignment PIN_C10 -to M[0]
set_location_assignment PIN_C11 -to M[1]
set_location_assignment PIN_D12 -to M[2]
set_location_assignment PIN_C12 -to M[3]

set_location_assignment PIN_A12 -to md[0]
set_location_assignment PIN_B12 -to md[1]
set_location_assignment PIN_A13 -to md[2]
set_location_assignment PIN_A14 -to md[3]

set_location_assignment PIN_B14 -to start
set_location_assignment PIN_F15 -to reset

set_location_assignment PIN_P11 -to Mclk #CLOCK

#leds
set_location_assignment PIN_A8 -to P[0]
set_location_assignment PIN_A9 -to P[1]
set_location_assignment PIN_A10 -to P[2]
set_location_assignment PIN_B10 -to P[3]
set_location_assignment PIN_D13 -to P[4]
set_location_assignment PIN_C13 -to P[5]
set_location_assignment PIN_E14 -to P[6]
```

Imagem 13 – Atribuição de pinos

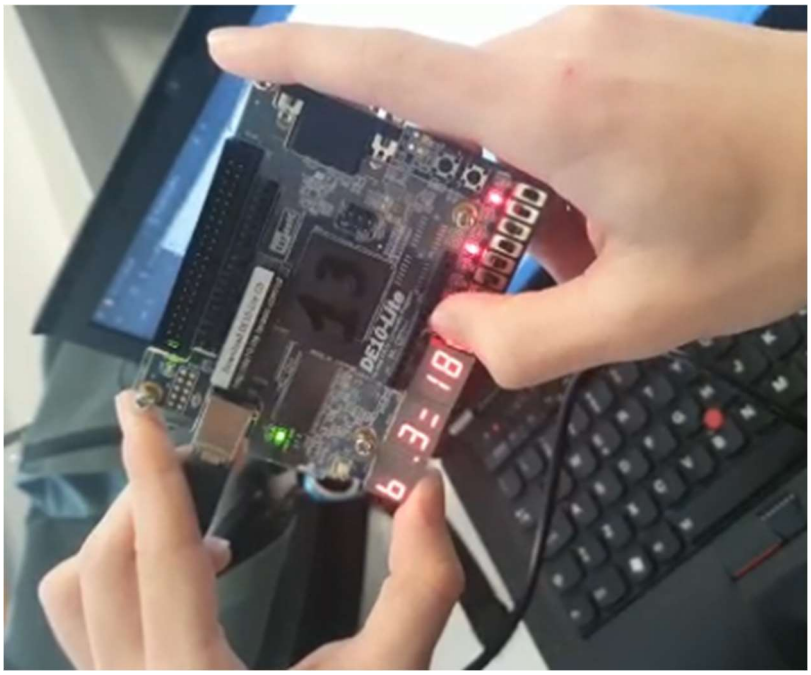

Imagem 14 – Testagem na placa Intel DE-Lite da Intel

4 Conclusão

Concluindo o trabalho, conseguimos aprofundar os nossos conhecimentos de circuitos de dados/controlo utilizando VHDL. Através do mesmo conseguiu-se implementar e aplicar diferentes conceitos aprendidos na cadeira como por exemplo a utilização de máquinas de estados, implementação de controlos microprogramados, elaboração de ASM-chart e também a simulação do circuito no programa Quartus Prime e verificar os resultados obtidos.