# INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Licenciatura em Engenharia Informática e de Computadores


Relatório do 4.º Laboratório
de
Lógica e Sistemas Digitais

## Circuitos Dados/Controlo – Tecnologia VHDL

Trabalho realizado por:
Nome: Ana Carolina Pereira Nº49470
Nome: Júlia Oliveira Mota Nº49452

Docente: Mário Vestias

24 de Janeiro de 2022

# ÍNDICE

# 1 Introdução

O quarto trabalho laboratorial teve como principal objetivo a implementação de circuitos com processamento de dados e controlo, descritos em VHDL.
Neste trabalho tivemos que projetar e implementar um circuito de multiplicação dos operadores, M e m, de 4 bits sem sinal (entre 0 e 9) através do algoritmo de somas sucessivas:

$$P = M \times m = \underbrace{M + M + M + \cdots + M}_{m \text{ vezes}}$$

# 2 Análise e Projeto

O circuito de Dados/Controlo foi descrito em VHDL no programa Quartus Prime. Para começar a descrição da unidade aritmética foi criado um novo projeto Quartus com o nome "TLAB4" onde foram feitos dezoito ficheiros, três para fazer o Controlo, treze para o Caminho de Dados, um ficheiro para o FFD que é comum a ambos e um final para juntar todo o projeto.

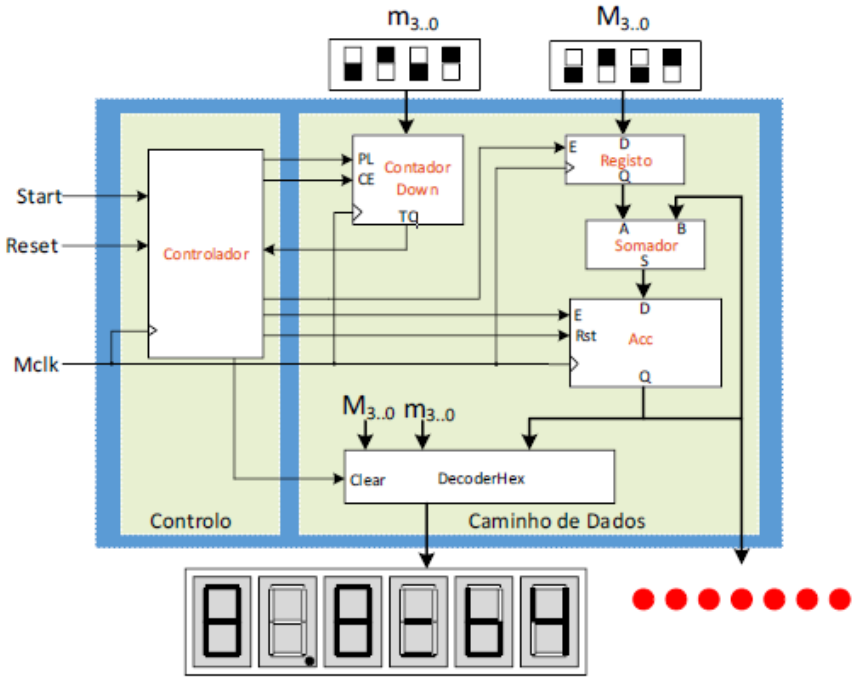



Figura 1 – Diagrama de blocos do multiplicador por somas sucessivas

## 2.1 Caminho de Dados

Para realizar o trabalho laboratorial 4 tivemos que começar por implementar um Caminho de Dados, que contém um Contador Down de 4 bits, um Registo de 4 bits, um Somador de 7 bits, um Acc de 7 bits, e um DecoderHex.

## 2.1.1 Contador Down

Para começar tivemos que implementar um sistema de contagem decrescente, que já tinha sido realizado no trabalho laboratorial 3, que irá realizar a contagem de quantas vezes somar-se-á o valor do M a si mesmo tendo em conta o valor introduzido para o m. Este contém como valor máximo o número 15 e a possibilidade de carregamento paralelo (PL). Para realizar este sistema necessitámos de criar os ficheiros "MuxDown", "FULLADER", "SomadorDown" e "RegistoDown" que dão origem ao contador decrescente, e por último criar o ficheiro final "ContadorDown" que junta todos os ficheiros anteriormente referidos e cria o sistema de contagem decrescente.

O contador decrescente contém como entradas o m, o PL (carregamento paralelo), que quando está ativo o valor do m fica salvo no contador, o reset, o CE (Count Enable), que quando ativado faz com que a contagem decrescente comece, e o CLK e contem como saída o TC (Terminal Countdown), que fica ativo quando a contagem chega ao valor 0. Para fazer a implementação em VHDL precisámos de um somador, um mux e um registo.

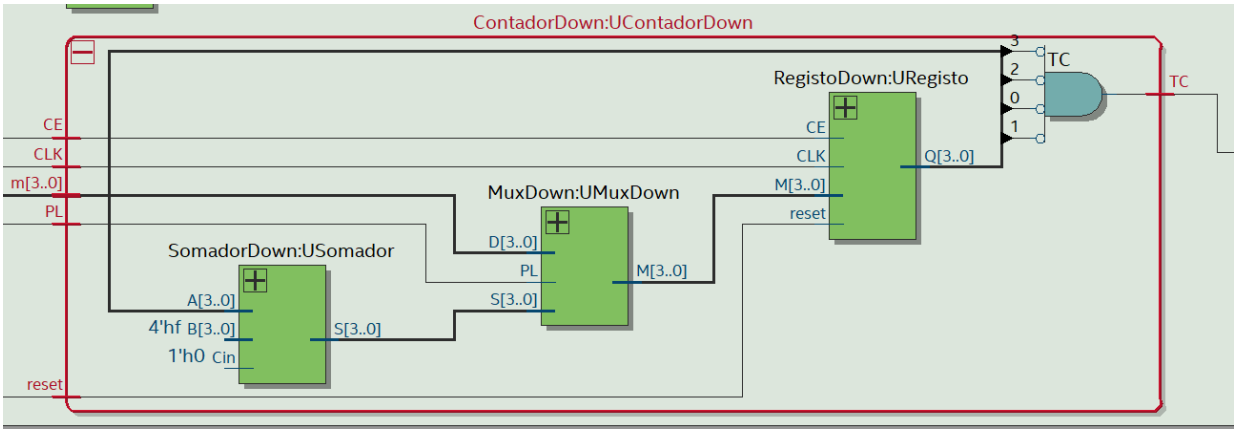



Figura 2 – Netlist Viewer do ContadorDown

### 2.1.1.1 Mux Down

Para programarmos o primeiro ficheiro "MuxDown" necessitámos de declarar duas entradas, a D e a S de 4 bits, uma entrada PL de 1 bit e uma saída M de também 4 bits. Após termos feito as declarações fomos fazer as funções lógicas para a saída M onde colocámos o valor de D a sair quando o PL estivesse ativo e o valor de S a sair quando o PL estivesse inativo.

**M** = (D and PL) or (S and (not PL))

### 2.1.1.2 FULLADER

O segundo ficheiro "FULLADER" foi programado de forma a fazer a soma e a subtração entre 2 números com 1 bit. Para a programação deste foi necessário declarar 3 entradas de um bit cada, a entrada A, a entrada B e a entrada Cin, e 2 saídas também de 1 bit cada, a saída S e a saída Cout. Após a declaração tivemos que fazer as funções aritméticas para a saída S e para a saída Cout.

**Nota:** As funções do S e do Cout são:
**S** = A ⊕ B ⊕ Cin
**Cout** = AB + Cin (A ⊕ B)

| D | S | Cin | M | Cout |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Tabela 1 – Tabela de verdade do Fullader

### 2.1.1.3 Somador

O terceiro ficheiro "Somador" foi programado de forma a fazer a junção de 4 "FULLADER" para se poder fazer a soma e a subtração entre dois números de 4 bits. Para a programação deste ficheiro tivemos que declarar três entradas, a entrada A e a entrada B de 4 bits cada e a entrada Cin de 1 bit, e duas saídas, a saída S de 4 bits e a saída Cout de 1 bit. Após isto estar feito tivemos que fazer um componente do ficheiro "FULLADER" programado anteriormente e declarar os sinais C1, C2 e C3 de 1 bit cada. De seguida tivemos que indicar a ligação entre todas as variáveis de cada um dos quatro FULLADER ligando sempre o bit do A e do B ao bit seguinte, o Cout a ligar-se ao C1, C2, C3 e por fim à própria saída Cout, o Cin a ligar-se ao Cin inicial e de seguida aos C que vinham da saída Cout de cada FULLADER e por último a nossa saída S a ligar-se sempre aos seus bits.

### 2.1.1.4 Registo

Para a programação do quarto ficheiro "Registo" foi preciso declarar 4 entradas, a entrada M, de 4 bits, e as entradas CLK, CE e reset, de 1 bit cada, e a saída Q de 4 bits. Após isto estar feito tivemos que fazer um componente do ficheiro "FFD", que foi fornecido pelo professor. De seguida tivemos que indicar a ligação entre todas as variáveis de cada FFD quatro vezes ligando sempre o CLK ao CLK, o RESET ao reset, o D e o Q sempre a cada bit de M e de Q, respetivamente, por exemplo D => M(0) e Q =>Q(0), o EN a ligar-se sempre ao CE e por último o SET a '0'.

#### 2.1.1.5 Contador Down

Por último, para programarmos o ficheiro "ContadorDown" precisámos de declarar cinco entradas, a entrada m de 4 bits e as entradas PL, reset, CE e CLK de 1 bit, e uma saída, a saída TC de 1 bit. Depois de fazermos esta declaração tivemos que criar quatro componentes para os ficheiros "MuxDown", "RegistoDown", "SomadorDown" e "CLKDIV". De seguida declarámos os sinais sSom, sM e sQ de 4 bits cada. Após termos tudo declarado e os componentes feitos tivemos que fazer as ligações. Primeiramente tivemos que ligar o TC à sua fórmula. Depois no MuxDown ligámos o nosso d ao m, o S ao sSom, o PL ao PL e o M ao sM. No Registo ligámos o CE, o CLK e o reset a eles mesmos, o M à sM e por último o Q à sQ. No Somador ligámos o A à sQ, o B ao valor de "1111" para a contagem ocorrer de forma decrescente, o Cin a '0', o Cout como open e o S à sSom.
**Nota:** O TC é dado através da fórmula:

**TC** = Not (A (3) or A (2) or A (1) or A (0))

## 2.1.2 Registo

Após termos o Contador Down feito tivemos que implementar um Registo de 4 bits. Para o realizar necessitámos de criar o ficheiro "Registo" e de utilizar o ficheiro "FFD" fornecido pelo professor.

O Registo contém como entradas o CLK, o Er (Enable do registo) de 1 bit cada e a entrada M, de 4 bits, e contem como saída o Q, de 7 bits. Para fazer a implementação em VHDL precisámos de um FFD.

Para a programação do "Registo" foi preciso declarar as entradas e as saídas enumeradas anteriormente. Após termos feito as declarações tivemos que fazer um componente do ficheiro "FFD", que foi fornecido pelo professor. De seguida tivemos que indicar a ligação entre todas as variáveis de cada FFD quatro vezes ligando sempre o CLK ao CLK, o RESET a '0', o D e o Q sempre a cada bit de M e de Q, respetivamente, por exemplo D => M(0) e Q =>Q(0), o EN a ligar-se sempre ao Er e por último o SET também a '0'.

## 2.1.3 Somador

Após o Contador Down e o Registo estarem elaborados tivemos que implementar um Somador de 7 bits para ser capaz de realizar as sucessivas somas do M, dado que a conta máxima que pode ser realizada é 9x9 e o resultado desta é 81 sendo assim necessário os 7 bits para o representar, até o Contador Down atingir o valor 0.

Para Implementar o Somador tivemos que criar o ficheiro "Somador" e utilizar o ficheiro "FULLADER" já implementado anteriormente. O ficheiro do Somador tem como entradas o A e o B de 7 bit e o Cin de 1 bit e tem como saídas o S de 7 bits e o Cout de 1

bit. Após declarar as entradas e saídas foi necessário fazer um componente do ficheiro "FULLADER" e de declarar os sinais c1, c2, c3, c4, c5, c6. Após termos tudo declarado foi necessário fazermos as ligações entre 7 Fulladers de forma a obtermos um somador de 7 bits. Nestas ligações tanto o A como o B estão sempre ligados aos seus próprios bits, o Cout aos sinais e por último a si mesmo, o Cin primeiramente a si mesmo e de seguida aos sinais que saem do Cout e por último o S também a ligar-se sempre aos seus bits.

## 2.1.4 Acc

Após ambos os somadores e o registo estarem implementados tivemos que implementar um outro Registo, o Acc, também de 7 bits de forma que este fosse capaz de armazenar o resultado das sucessivas somas do M até estas acabarem de forma a conseguirmos assim obter o resultado final.

Para implementar o Acc foi necessário declararmos a entrada D, de 7 bits, as entradas CLK, Eacc e Rst, de 1 bit cada, e a saída Q, de 7 bits. De seguida tivemos que fazer um componente do ficheiro FFD já fornecido pelo professor e tivemos também que fazer as ligações todas entre sete FFD ligando sempre o EN ao Eacc, o RESET ao Rst, o SET a '0', o CLK a si mesmo e o D e o Q a ligarem-se aos seus próprios bits.

## 2.1.5 CaminhoDados

Por fim depois de termos todos os ficheiros implementados e de termos guardado no projeto o ficheiro "decoderHex" que representa o DecoderHex, tivémos que criar o ficheiro final que une todos estes ficheiros anteriormente explicados e realiza as ligações entre eles. Para realizar este ficheiro necessitámos de primeiramente declarar todas entradas e saídas e de fazer um componente de todos os ficheiros finais explicados anteriormente, de seguida declarar os sinais sA, sD e sQ, de 7 bits cada, e por último tivemos que realizar as ligações entre cada componente e entre os LEDS ao sQ finalizando assim o Caminho de Dados.

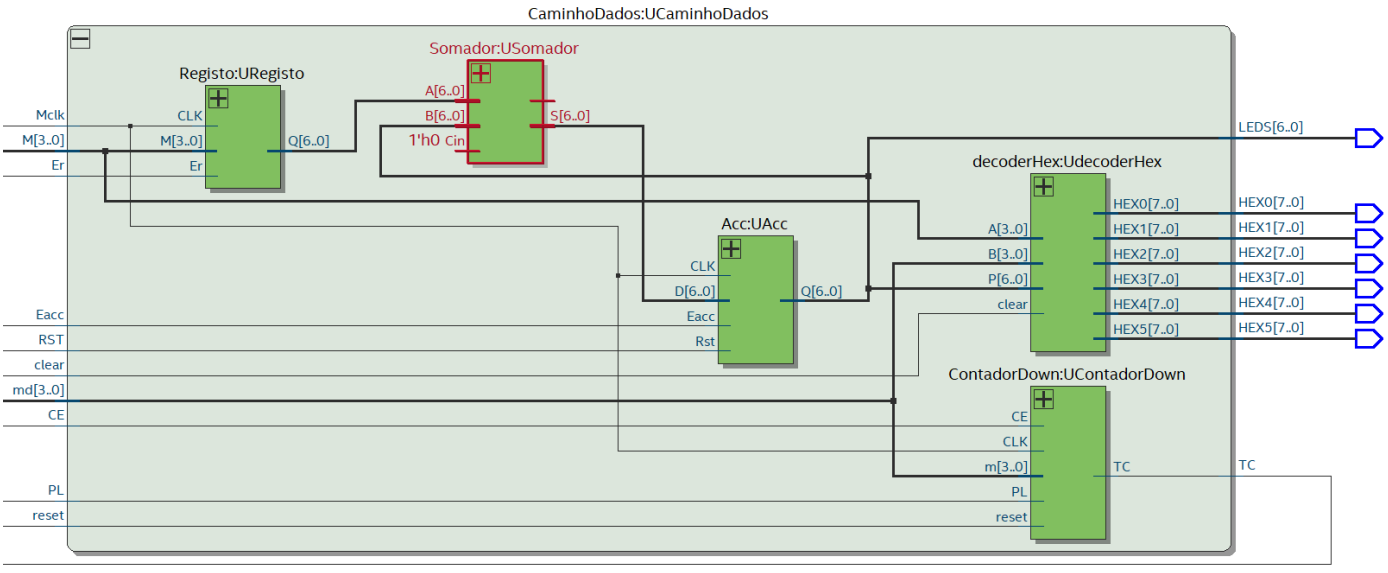



Figura 3 – Netlist Viewer do CaminhoDados

## 2.2 Controlo

Para realizar o trabalho laboratorial 4 tivemos também que implementar um Controlo que é constituído por um registo e uma máquina de estados.

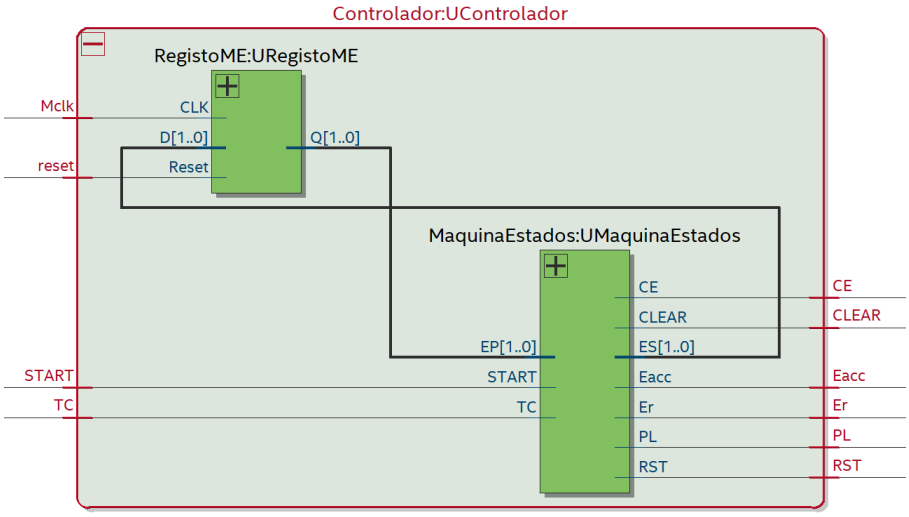



Figura 4 – Netlist Viewer do Controlador

### 2.2.1 Registo

Para implementar o Registo, de 2 bits, do Controlo necessitámos de criar o ficheiro "RegistoME" e de utilizar o ficheiro "FFD" fornecido pelo professor. O Registo contém como entradas o D, de 2 bits, o CLK e o Reset de 1 bit cada e contém como saída o Q de também 2 bits.

Para a programação do "RegistoME" foi preciso declarar as entradas e as saídas enumeradas anteriormente. Depois foi necessário fazer um componente do ficheiro "FFD" e de fazer as suas respetivas ligações duas vezes, ligando assim o CLK e o Reset sempre a si mesmos, o SET a '0', o EN '1' e o D e o Q aos seus respetivos bits.

### 2.2.2 Máquina de Estados

Para realizar a máquina de estados tivemos que fazer 3 estados de forma que a sequência de operações a realizar fossem feitas:

```
Acc = 0
while (m ≠ 0){
      m = m - 1
      Acc = M + Acc
}
```

Depois de termos criado os nossos estados tivemos que elaborar o ASM-chart da nossa máquina de estados, onde o primeiro estado continha o RAcc, pois o acumulador tem de começar a 0, o PL, que é necessário para o Contador Down, o CE, para a contagem poder começar, o Enr e o clear, depois foi necessário fazer a entrada Start pois o algoritmo apenas começa a contar quando o Start encontra-se a 1, após o Start estar a 1 passa-se para o estado seguinte onde queremos colocar o while a funcionar apenas se o valor de m for diferente de 0 e por isso tivemos que introduzir a entrada TC, que apenas permite que a multiplicação comece se este estiver  0 e tivemos também de introduzir CE e o ENAcc, de forma a que a contagem ocorra e que o valor da mesma seja armazenado. Depois da multiplicação ter sido realizada o TC passa para 1 e avançamos assim para o estado seguinte que irá avaliar outra vez o Start pois apenas é permitido voltar-se a fazer uma multiplicação depois de o Start ter sido desativado até lá apenas fica representado a multiplicação atual.

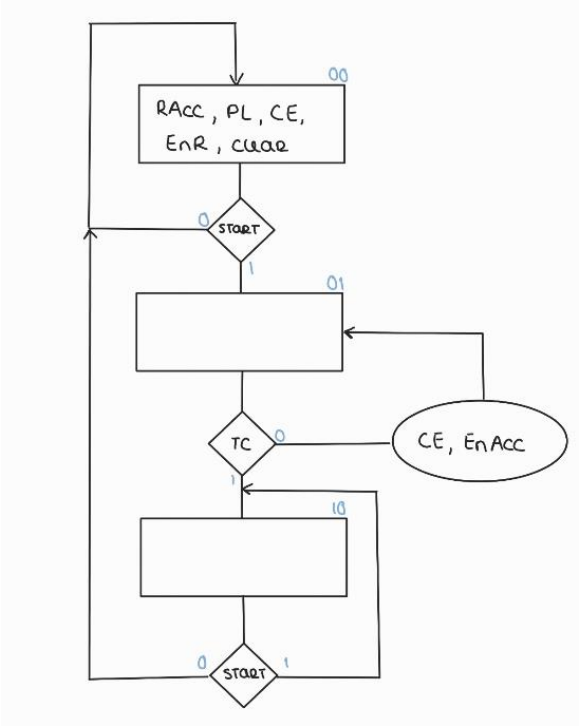



Figura 5 – ASM Chart

A seguir a termos o ASM-Chart elaborado tivemos que criar uma tabela de transição de estados de forma que conseguíssemos saber quais eram os nossos estados atuais, estados seguintes e que entradas e saídas estavam ligadas.

| EP | Start | TC | ES | PL | CE | Reset | EA | ER | Clear |
|---|---|---|---|---|---|---|---|---|---|
| 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 | 10 | 0 | 0 | 0 | 0 | 0 | 0 |

Tabela 2 – Tabela de transição de Estados

Por fim, através da tabela de transições e juntamente com o ASM-chart, foi possível implementar a máquina de estados em VHDL. Para isso tivemos que adicionar ao projeto o ficheiro já fornecido pelo professor e depois alterar nele a fórmula do adress que passou a ser **adress <= EP & START & TC**, alterar as nossas saídas e os seus bits e por fim alterar as fórmulas do data tendo em conta a nossa tabela de transições de estados.

### 2.2.3 Controlo

Após termos o nosso Registo de 2 bits e a nossa máquina de estados implementados tivemos que criar o ficheiro final que cria o controlador. Para realizarmos este ficheiro final tivemos que declarar as entradas START, reset, Mclk e TC de 1 bit cada e as saídas PL, CE, CLEAR, RST, Eacc, Er de 1 bit cada. Depois de termos as entradas e saídas declaradas tivemos que fazer o componente tanto do registo como da maquina de estados, declarar os sinais sD e sQ, de 1 bit cada, e por último fazer as ligações entre o Registo e a Maquina de Estados.

## 2.3 Multiplier

Finalmente depois de termos o Caminho de Dados e o Controlador feitos criámos o ficheiro final “Multiplier” que junta estes dois ficheiros criando assim o multiplier que realiza multiplicações através de sucessivas somas.

Para fazer a programação deste ficheiro tivemos que declarar as entradas M e md de 4 bits, as entradas reset, Mclk e start de 1 bit cada, a saída P de 7 bits e as saídas do HEX0 ao HEX5 de 8 bits cada. Depois de termos isto feito foi preciso fazer os componentes dos ficheiros “CaminhoDados” e “Controlador” e declarar os sinais sPL, sCE, sEacc, sTC,

sRST, sClear de 1 bit cada. Por último tivemos que fazer as ligações entre a "MaquinaEstados" e o "Controlador".

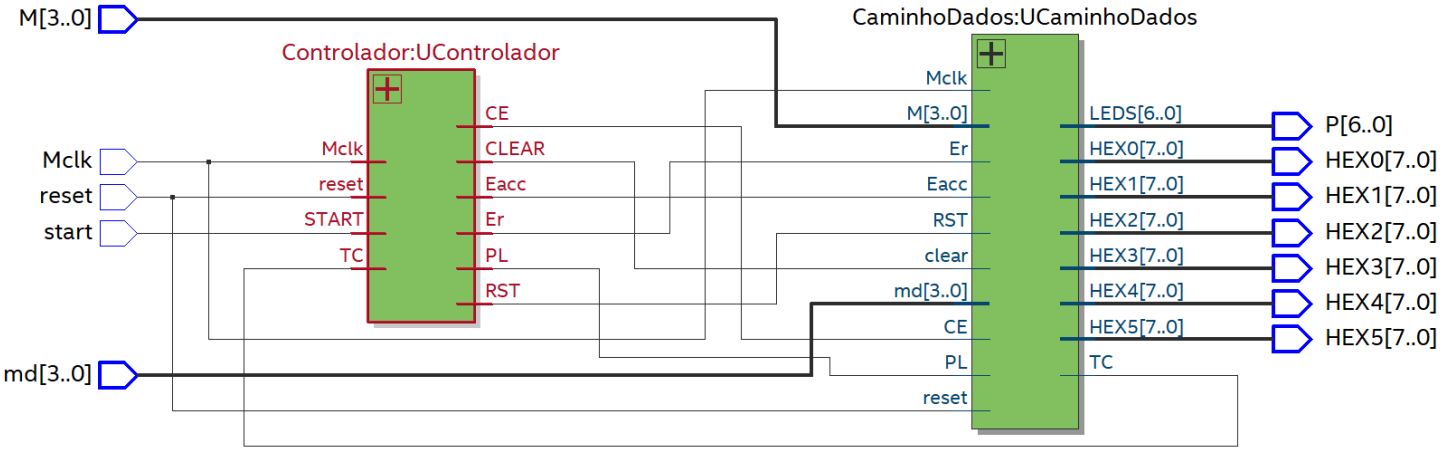
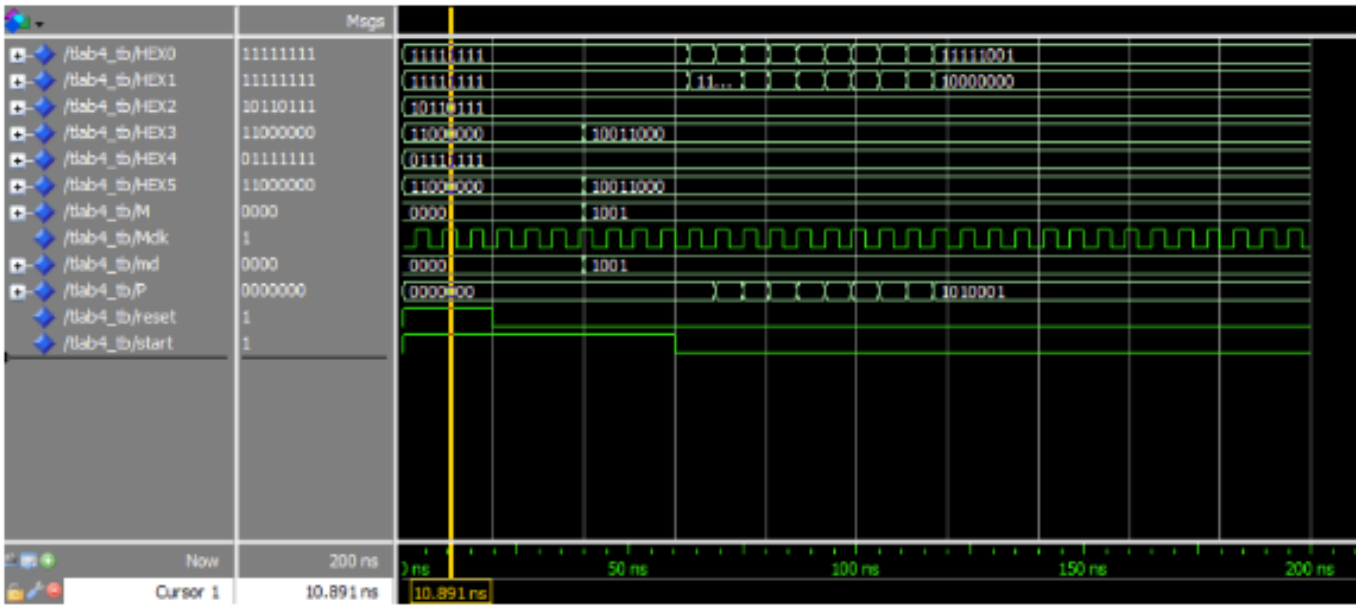



Figura 6 – Netlist Viewer do TLAB4

## 3- Montagem laboratorial:

Após o circuito de dados/controlo ter sido descrito em VHDL estrutural este foi simulado de maneira a verificar que todas as funções estavam a ser bem executadas. Primeiramente testamos o nosso multiplier no simulador onde pelas imagens conseguimos, através de movimentar a reta amarela pela linha temporal, analisar os cálculos a serem efetuados, a ativação do botão start e o resultado final da multiplicação para conseguirmos ter a certeza que o circuito está a funcionar como o suposto.



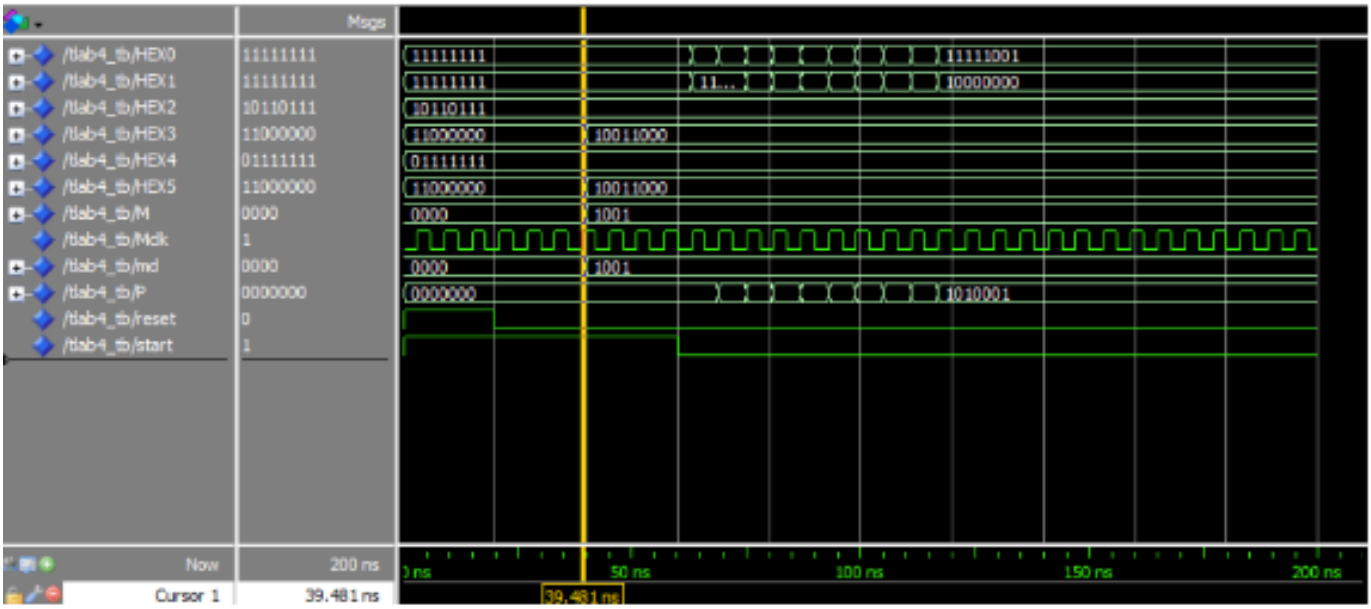



Figura 7 e 8 – Simulação do projeto

Através destas observações e de várias operações realizadas na placa MAX DE-10 LITE podemos concluir que o nosso sistema está a realizar as diversas operações corretamente.

## 4- Conclusão

O trabalho consistiu na implementação de um circuito de dados e controlo, através de tecnologia VHDL estrutural, para criar um sistema que efetua a multiplicação entre dois números através de somas sucessivas.

Com a realização deste trabalho é possível concluir que os sistemas de contagem podem contar de uma forma crescente ou decrescente, que para conseguirmos fazer uma máquina de estados temos que primeiramente dividir as caracteristicas pedidas em estados, criar um ASM-chart e por último realizar uma tabela de transição de estados e que para realizarmos uma multiplicação conseguimos a fazer através de repetidas somas.

Por fim, após termos testado o circuito através da simulação e da placa MAX DE-10 LITE conseguimos concluir que a tabela de transição de estados retirada através do ASM Chart que elaborámos, que a programação e que as ligações do nosso circuito estão corretas pois todos os valores esperados foram obtidos através da análise da simulação do nosso circuito.