The logo features the text "UART project" in a stylized, blue, hatched font. It is centered within a circular graphic composed of three concentric black rings. The background of the entire slide is a light gray with a pattern of binary digits (0s and 1s) arranged in a circular, tunnel-like perspective.

UART project



Victor Moreno Arribas

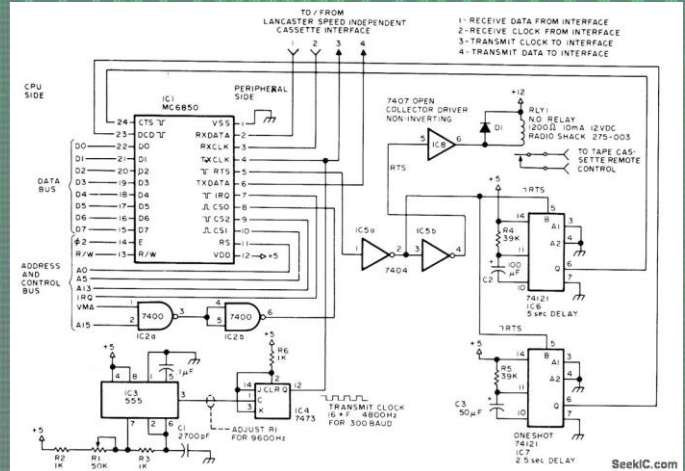
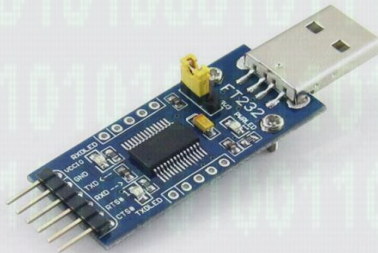
180070

Jorge Albendea Pizarro

190229

MC6850

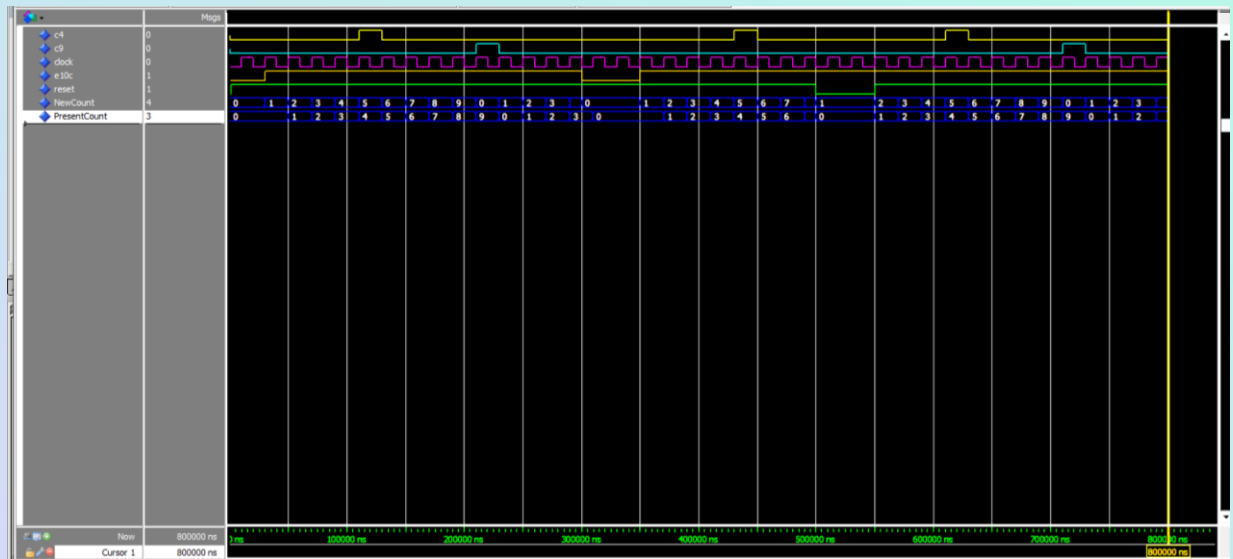
Pin	Function	Pin	Function
1	VSS	24	CT
2	Rx Data	23	DCO
3	Rx CLK	22	D0
4	Tx CLK	21	D1
5	RTS	20	D2
6	Tx Data	19	D3
7	IRO	18	D4
8	CS0	17	D5
9	CS2	16	D6
10	CS1	15	D7
11	RS	14	E
12	VCC	13	R/W



MC6850



Get a screenshot of the whole 800 μ s simulation.



Picture_01.png

c4
c9
clock
e10c
reset
FutRegValue
RegValue

The pictures on this report had been modified with stylish effects to show the signals easier.

In the folder you can find the modified picture with the name they have underneath and the original screenshot with the name of the file, for example in this case:

"Counter_Simulation.png"

At $t = 500 \mu s$ the reset is activated and it certainly sets the present state of the counter to 0, but why does it have the effect of setting the future state to 1?

A

B

C

D

E

F

G

H

I

J

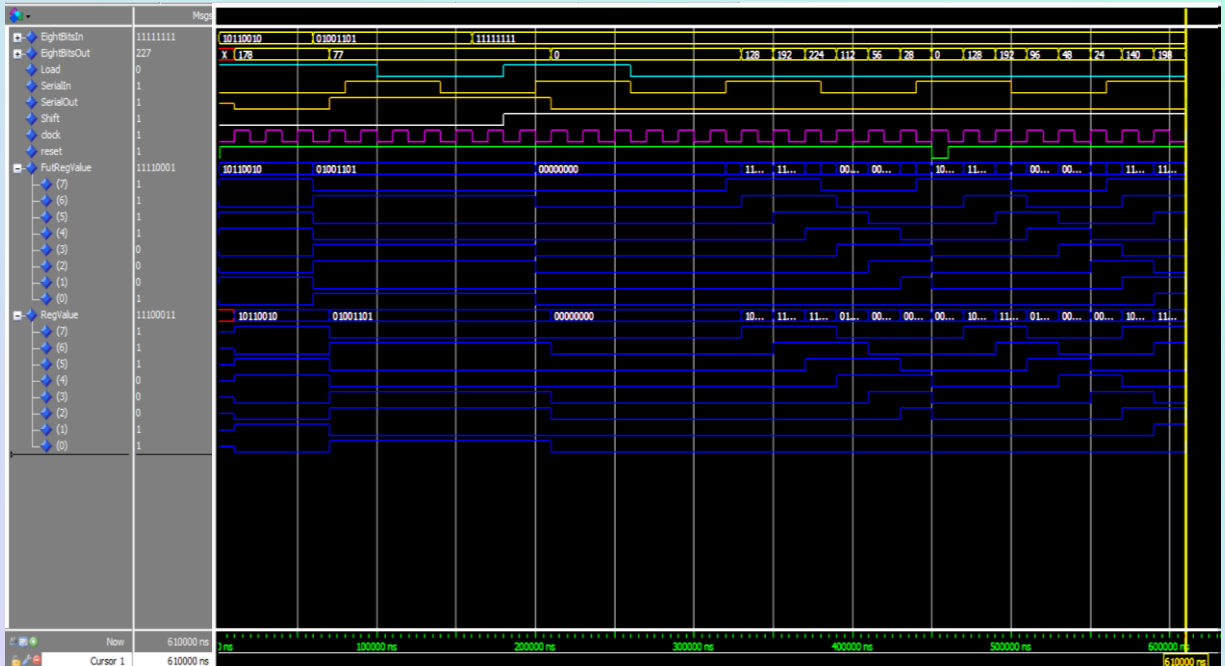
K

The state 0 is the only that has an input signal called reset, the rest of them only have the e10c.

But there is one more thing that only the state 0 has, it only has one output, while the rest of states have two outputs.

In the rest of stest when it receives a 1 it does one thing and when it receives a 0 it does another, instead the state 0 can only do one thing, and is going to the state 1, that why when reset is activated sets the present state to 0, and the next step is e10c with value 0, because it can't have value '1' or at least we don't have anything to do in state 0 when e10c is '1'.

Get a screenshot of the whole 610 μ s simulation.



Picture_02.png

EightBitsIn
 EightBitsOut
 Load
 SerialIn
 SerialOut

Shift
 clock
 reset
 FutRegValue
 RegValue

At $t = 450 \mu s$ the reset is activated and it certainly sets the present state of the shift-register to 0, but why does it have the effect of setting the future state to 128?

A

B

C

D

E

F

G

H

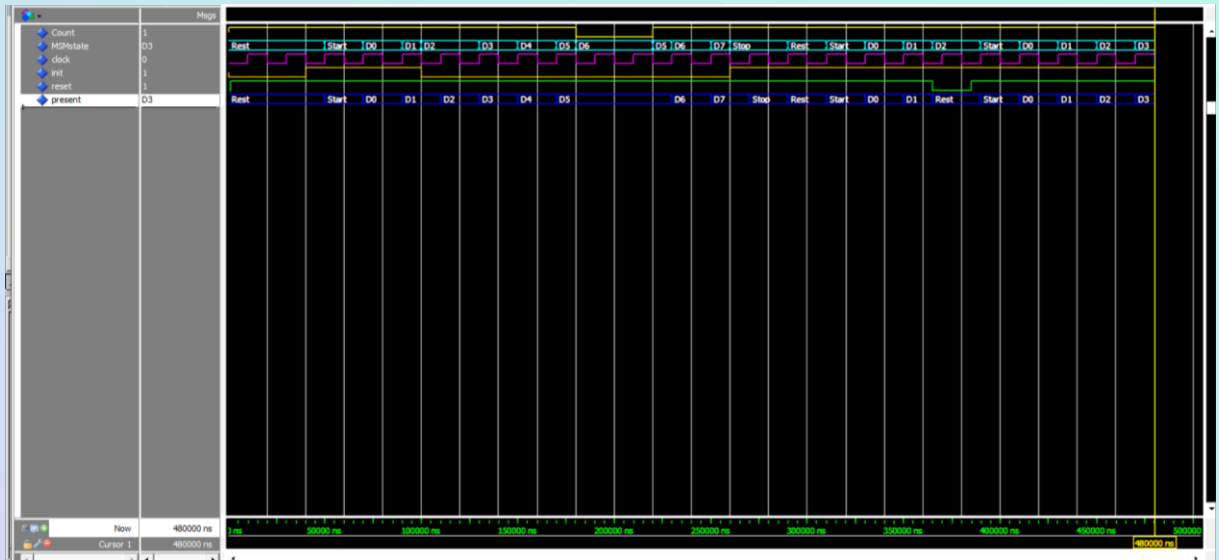
I

J







K

Due to the reset being out putted to 0, and changing the value to all zeros the next stage will always be the parallel one, being that 128.

Get a screenshot of the whole 480 μ s simulation.



Picture_03.png

 count
 MSMstate
 clock
 init
 reset
 present

A**B****C****D****E****F****G****H****I****J****K**

Explain how the function `NextState` works, which means that you should search on the internet what an attribute in VHDL is and what the “succ” attribute does. Put some other example of an attribute (and explain it).

Before the function `NextState` was declared, a new type of value called `MSMstates` had been declared with all possible states. The function `NextState` receives as a input parameter a `State` of type `MSMstates` and returns one of the states declared as type `MSMstates`.

To select the right output of the function `NextState` there are a few conditions, the first one is the exceptional case that the state received is `Stop` so the next state following the diagram is the `Rest` state. The `MSMstates` declaration looks like an array in which we declared all the possible states in the same order as the diagram.

A

B

C

D

E

F

G

H

I

J

K

Now we go to the second condition, in case the state given is not Stop, we go to the “array” MSMstates and choose the next one in the list. In Java it will be like MSMstates[i+1] been “i” the input parameter states. Now we can understand why is the first condition declared first, because **Stop is the end of the “array”** so if the first condition wasn't declared it wouldn't be a loop and it would have errors.

Attributes are used in a lot of programming languages to set additional information about an object.

For example in HTML we have the attribute background-color to set the background color of an object, or the href attribute to set a link or url in an object.

In **VHDL the attributes can be values, functions, types, signals or constants** and we can associated with a name for easier use and addaption to the code.

A

There are two types of attributes:

- Predefined attributes
- User-defined attributes

B

An example of user-defined attributes can be the type attribute *MSMstates* defined at *AuxTypes* or the function attribute *NextState* also declared in the file *AuxTypes*.

C

D

E

F

Some predefined value attributes are *T'LEFT* or *T'RIGHT* the leftmost value of *T* or the rightmost value of *T*.

G

H

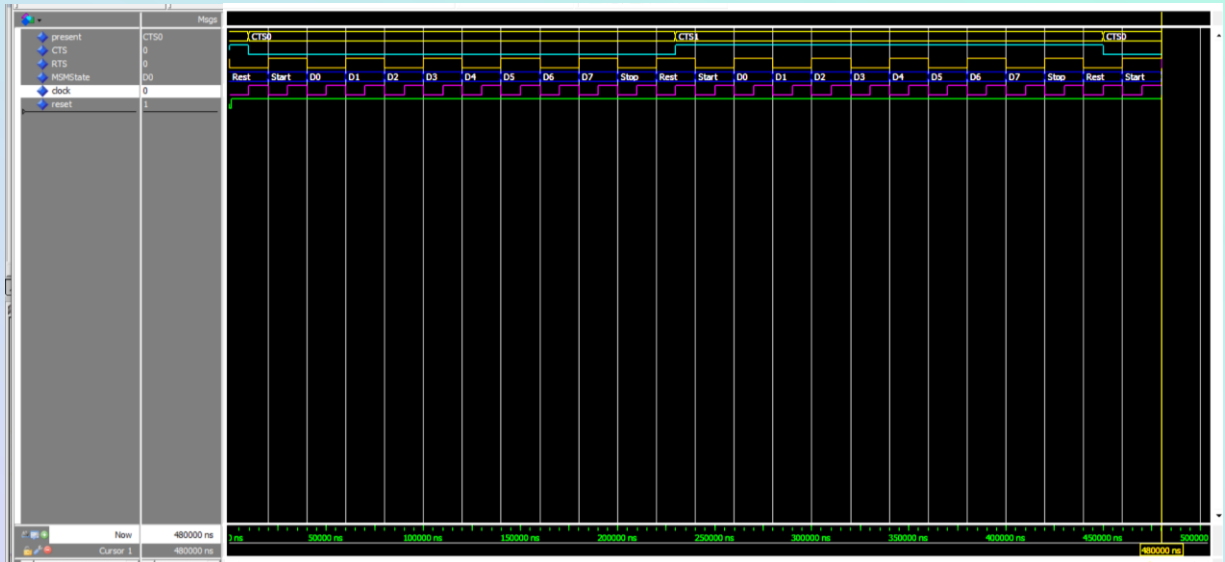
I

J

K

Some predefined function attributes are *T'POS(v)* which Return the position number of the value *v* in the ordered list of values of *T*, or *T'VAL(p)* which Return the value of the type that corresponds to the position *p*, and the King of the party would be ***T'SUCC(v)*** which Returns the value of the parameter whose position is one larger than the position of value *V* in *T*

Get a screenshot of the whole 480 μ s simulation.



Picture_04.png

present
CTS
RTS
clock
reset
MSMstate

Explain in your own words all problems which could happen if you forget to take into account the “Rest” condition in both transitions.

A

B

C

D

E

F

G

H

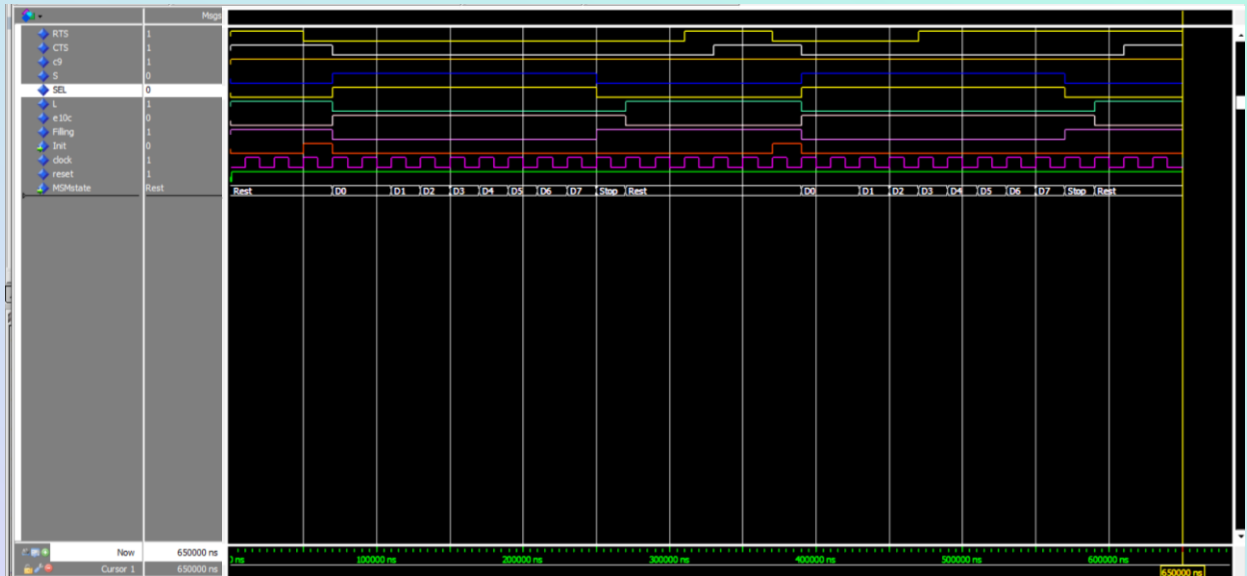
I

J

K

If we forget about the rest state we wouldn't know when the emitter is done transmitting data, so we cannot be Clear To Send (CTS). When we are in the rest state we are the data is already loaded.

Get a screenshot of the whole 650 μ s simulation.

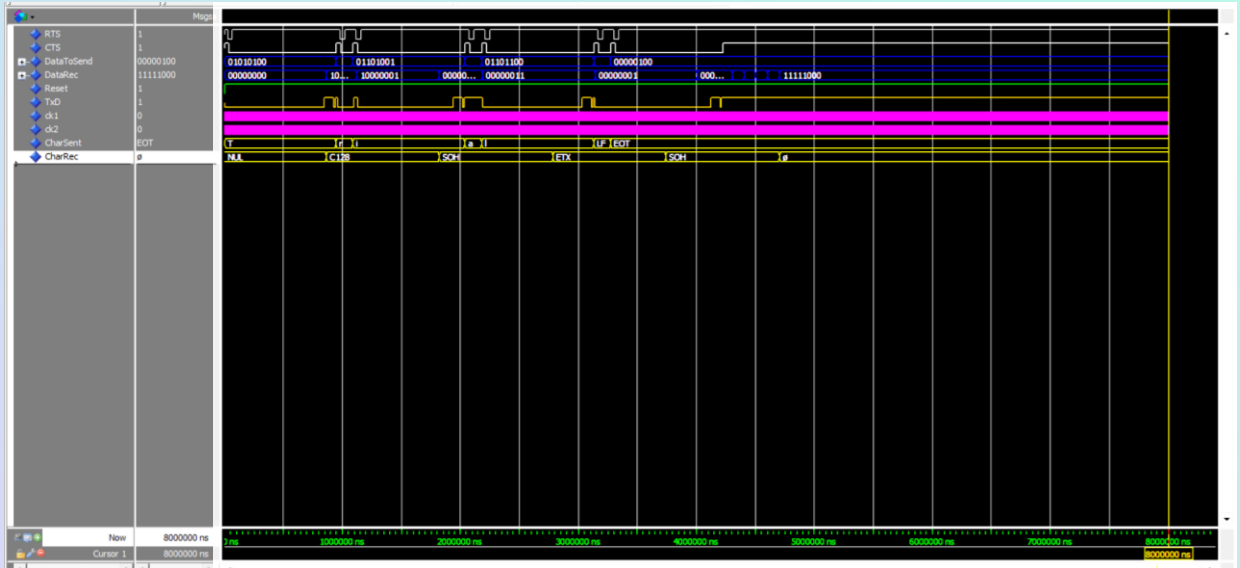


Picture_05.png

RTS
CTS
c9
S
Sel
L

e10c
Filling
init
clock
reset
MSMstate

Get a screenshot of the whole 8000 μ s simulation.



Picture_06.png

RTS
CTS
DataToSend
DataRec
reset

TxD
ck 1
ck 2
CharSent
CharRec

A

B

C

D

E

F

G

H

I

J

K

The reader process asks for the name of the file to be transmitted. However the writing process uses a fixed filename. Read both processes to see how it is done and modify the writing process so that it also asks for the name of the file to be written. You may also need to search on the internet the properties of the functions of the TEXTIO VHDL package, if you want to understand just every single detail of the code.

In the Reader.vhd, first the autor has declared the default question that the program has to ask to know which file to read. As well, the autor has also declared all the possible errors or “exceptions” as we might called in Java. He declared which cases would be an exception and which message to show in the screen in each case with the predefined function WRITELINE. In the other hand, for reading the name of the file we wrote in the console, it uses the predefined function READLINE. Now if there we no errors we may continue analysin the file.

A

B

C

D

E

F

G

H

I

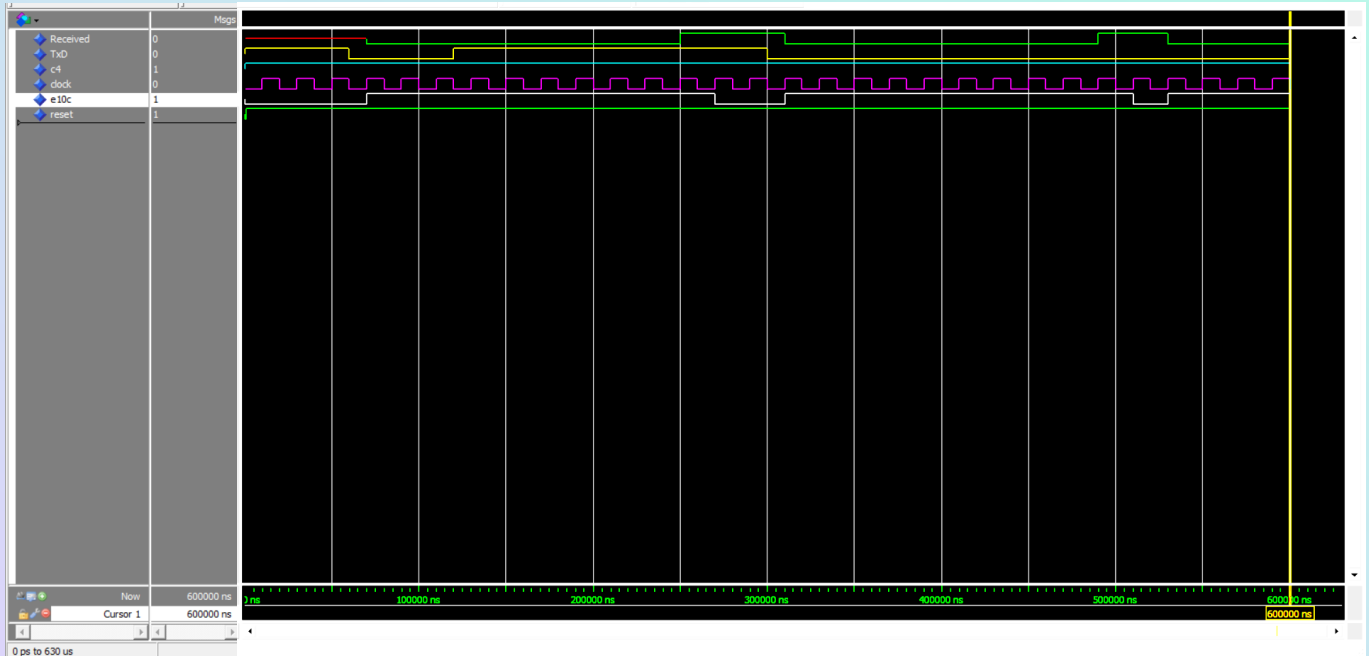
J

K

A “while” loop is declared for Reading each of the characters that compose the ASCII text which is written in the specified file, until the last carácter. After finding it, we have to send the EOT character (End Of Transmission), as we can see is a quite similar carácter as the one we saw in the one line reader example, that one we said that stops a printer from printing.

In the Writer file we have declared a UartOutBox file which is the one we would Return as the result and is just again another “while” loop that is writing the same text as the Reader file readed. In this file we can also observed how some exceptions had been declared, luckily some of them didn't appeared to us, yet. The curious thing in this file, if the OneChart variable which storages the character value of the integer from the binary code in the DataRec. Hopefully, our brain could do it so easy too.

FSMReceiver



 **Received**
 **TxD**
 **c4**
 **clock**
 **e10c**
 **reset**

Picture_06.png

This feels AWESOME!!!

```
# Compile of AuxTypes.vhd was successful.  
# Compile of Reader.vhd was successful.  
# Compile of Receiver.vhd was successful.  
# Compile of TestControlUnit.vhd was successful.  
# Compile of TestCounter.vhd was successful.  
# Compile of TestFSMEmitter.vhd was successful.  
# Compile of TestFSMReceiver.vhd was successful.  
# Compile of TestManyStateMachine.vhd was successful.  
# Compile of TestShiftRegister.vhd was successful.  
# Compile of Writer.vhd was successful.  
# Compile of FinalTest.vhd was successful.  
# Compile of TestTwoStateMachine.vhd was successful.  
# Compile of Counter.vhd was successful.  
# Compile of ShiftRegister.vhd was successful.  
# Compile of ManyStateMachine.vhd was successful.  
# Compile of TwoStateMachine.vhd was successful.  
# Compile of ControlUnit.vhd was successful.  
# Compile of Emitter.vhd was successful.  
# Compile of FSMReceiver.vhd was successful.
```




POLITÉCNICA

“Ingeniamos el futuro”

CAMPUS
DE EXCELENCIA
INTERNACIONAL



E.T.S. de Ingenieros
Informaticos

dat si

Departamento de Arquitectura y
Tecnología de Sistemas Informáticos

