

# **INSTITUTO POLITÉCNICO NACIONAL**

## **ESCUELA SUPERIOR DE CÓMPUTO**

***ING. EN SISTEMAS COMPUTACIONALES***



## **Tarea 2: Memoria ROM**

**ASIGNATURA:** ARQUITECTURA DE COMPUTADORAS

**PROFESOR:** MIGUEL ANGEL ALEMAN ARCE

**GRUPO:** 5CV1

**ALUMNO:** ROMERO HERNÁNDEZ OSCAR DAVID

## INTRODUCCION

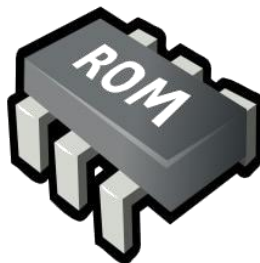
La memoria ROM (Read-Only Memory, en inglés) es un tipo de memoria que se utiliza en sistemas electrónicos para almacenar datos que no se pueden modificar o borrar. A diferencia de la memoria RAM (Random Access Memory), la ROM es una memoria de solo lectura, lo que significa que los datos almacenados en ella no se pierden cuando se apaga el dispositivo.

La memoria ROM se utiliza para almacenar el firmware del sistema, que es el software que controla el funcionamiento básico del hardware de un dispositivo electrónico. Esto incluye las instrucciones para arrancar el sistema y el software de bajo nivel necesario para comunicarse con los dispositivos de entrada y salida.

La ROM se fabrica mediante un proceso conocido como "quemado", en el que se escriben los datos en la memoria mediante un proceso físico de grabación. Una vez que se han grabado los datos en la ROM, estos no se pueden borrar ni modificar, por lo que la información almacenada es permanente y segura.

Consideremos tres tipos distintos de memoria ROM:

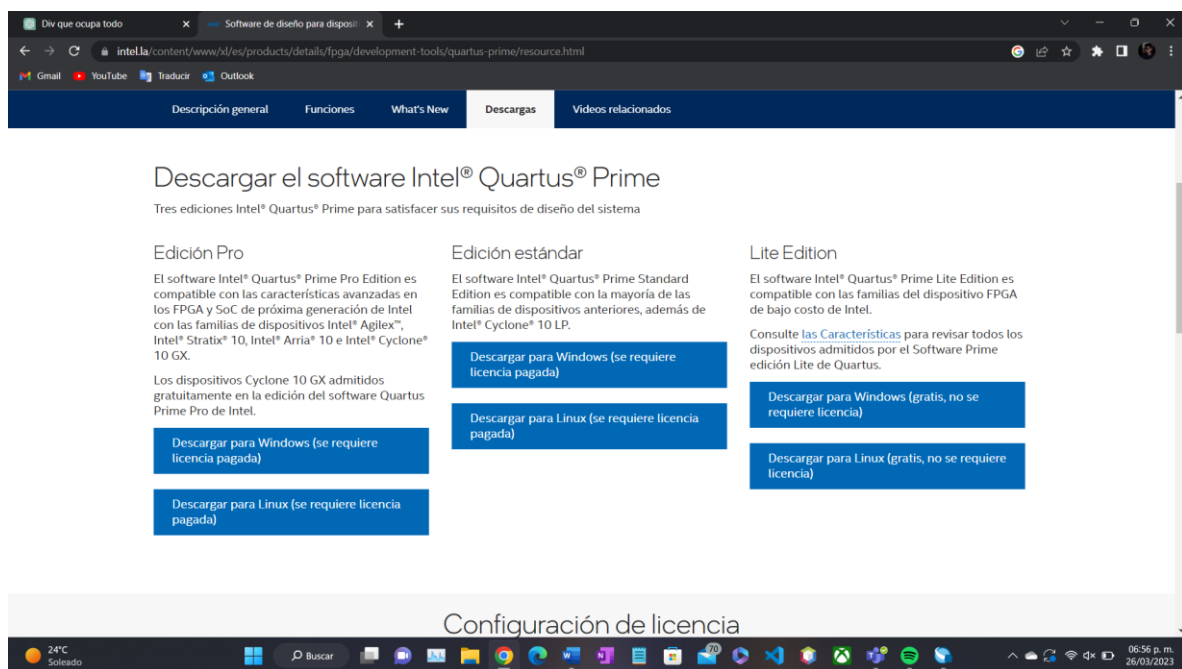
- **PROM.** Acrónimo de *Programmable Read-Only Memory* (Memoria de Sólo Lectura Programable), es de tipo digital y puede ser programada una única vez, ya que cada unidad de memoria depende de un fusible que se quema al hacerlo.
- **EPROM.** Acrónimo de *Erasable Programmable Read-Only Memory* (Memoria de Sólo Lectura Borrable y Programable) es una forma de memoria PROM que puede borrarse al exponerse a luz ultravioleta o altos niveles de voltaje, borrando la información contenida y permitiendo su replazo.
- **EEPROM.** Acrónimo de *Electrically Erasable Programmable Read-Only Memory* (Memoria de Sólo Lectura Borrable y Programable Eléctricamente) es una variante del EPROM que no requiere rayos ultravioletas y puede reprogramarse en el propio circuito, pudiendo acceder a los bits de información de manera individual y no en conjunto.



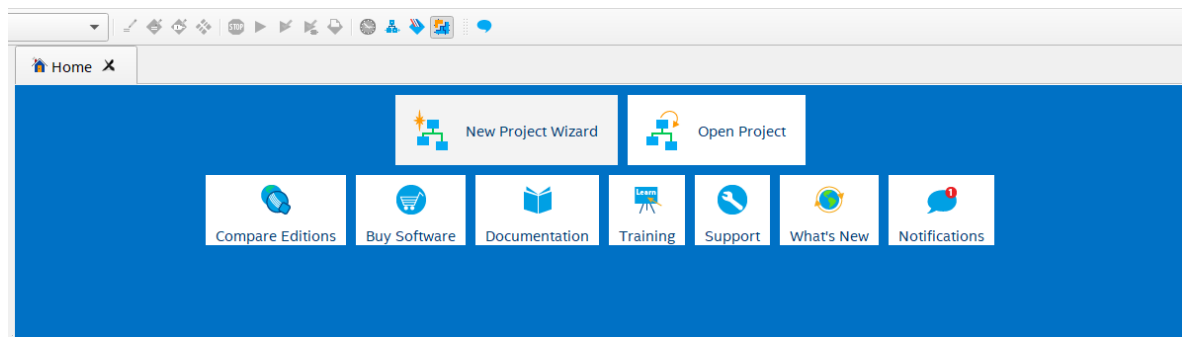
# IMPLEMENTACIÓN DE MEMORIA ROM EN VERILOG USANDO QUARTUS

Para implementar y simular una memoria ROM en verilog se usará el software "Quartus" el cual es un software de diseño de circuitos integrados y programación de dispositivos lógicos programables (FPGA, por sus siglas en inglés) desarrollado por Intel. A continuación, se detallan los pasos para usar Quartus y realizar una compilación y simulación de la Memoria ROM:

**1.- Descargar e instalar Quartus:** Para empezar, se debe descargar Quartus desde la página web de Intel y seguir las instrucciones de instalación. Nota Usar como máximo la versión 20.1



**2.- Crear un nuevo proyecto:** Después de abrir Quartus, selecciona la opción "New Project Wizard" desde el menú "File". En la ventana que aparece, ingresa un nombre y una ubicación para el proyecto y hacer clic en "Next".



New Project Wizard

### Directory, Name, Top-Level Entity

What is the working directory for this project?

C:/Users/osvid/Documents/rom

What is the name of this project?

rom

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

rom

Use Existing Project Settings...

< Back Next > Finish Cancel Help

**3.- Seleccionar dispositivo:** En la siguiente ventana, seleccionar el dispositivo que utilizarás en el proyecto. También se puede agregar uno si no aparece en la lista.

New Project Wizard

### Family, Device & Board Settings

Device Board

Select the family and device you want to target for compilation.  
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family

Family: Cyclone V (E/GX/GT/SX/SE/ST)

Device: All

Target device

☐ Auto device selected by the Fitter

☒ Specific device selected in 'Available devices' list

☐ Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

Name filter:

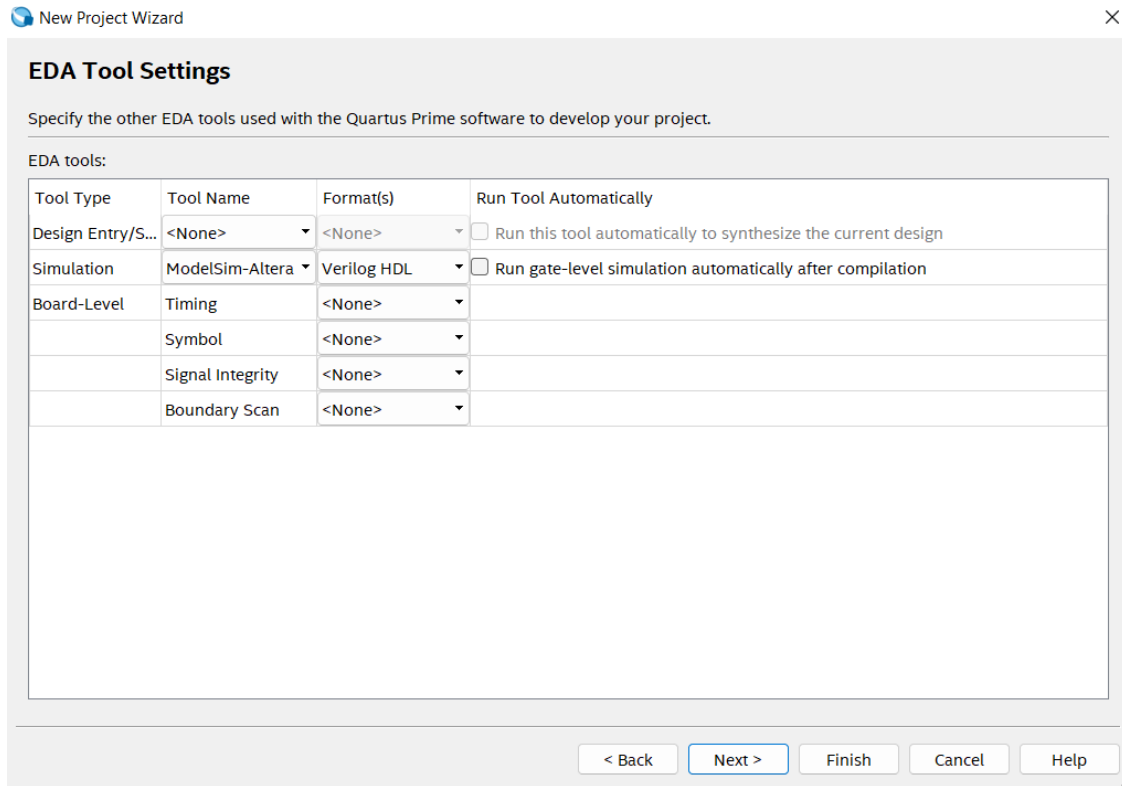
☒ Show advanced devices

Available devices:

Name	Core Voltage	ALMs	Total I/Os	GPIOs	GXB Channel PMA	GXB Channel PCS	PC
5CGXFC7C6F23I7	1.1V	56480	268	240	6	6	1

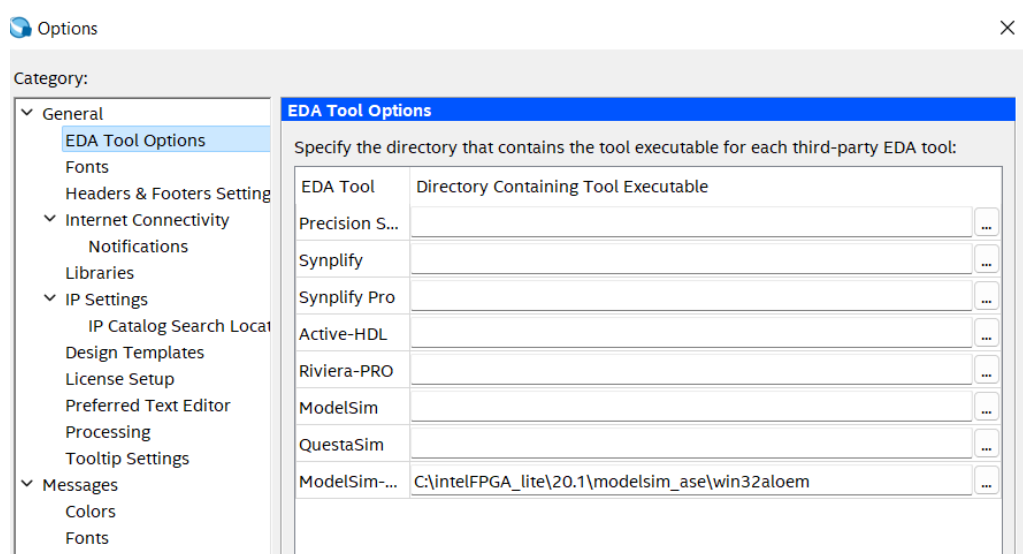
Help < Back Next > Finish Cancel

**4.- Configurar fuentes de diseño:** En la siguiente ventana, selecciona las fuentes de diseño para tu proyecto. Puedes agregar archivos VHDL, Verilog, archivos de texto y otros. *Nota: en este caso para la simulación se debe seleccionar “ModelSim-Altera” y “Verilog HDL”*



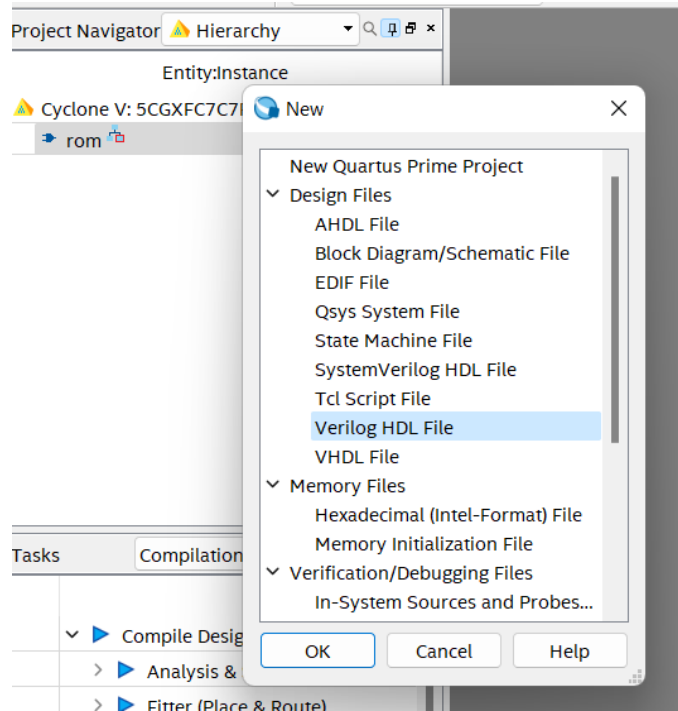
Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/S...	<None>	<None>	<input type="checkbox"/> Run this tool automatically to synthesize the current design
Simulation	ModelSim-Altera	Verilog HDL	<input type="checkbox"/> Run gate-level simulation automatically after compilation
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

**5.- Configurar el PATH del simulador:** Para configurar el path del simulador debemos entrar a Tools > Options > General > EDA tool Options. Y colocar la ruta del simulador.



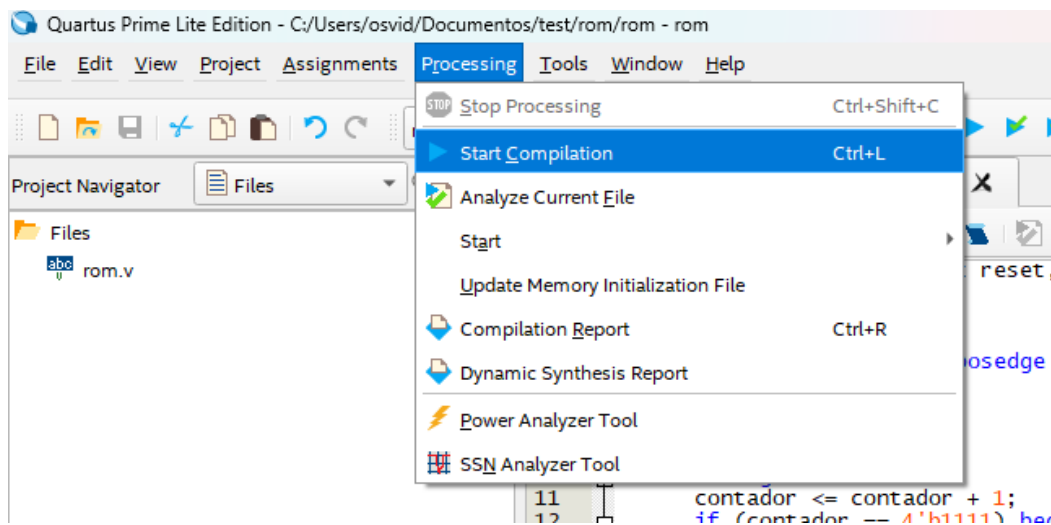
EDA Tool	Directory Containing Tool Executable
Precision S...	
Synplify	
Synplify Pro	
Active-HDL	
Riviera-PRO	
ModelSim	
QuestaSim	
ModelSim-...	C:\intelFPGA_lite\20.1\modelsim_ase\win32aloem

**6.- Crear el código de la memoria ROM:** Se procede a crear el código de la memoria ROM, para poder ser simulada (el funcionamiento del código se explicará más adelante). Donde en primera instancia se procede a crear un nuevo archivo verilog para el proyecto:

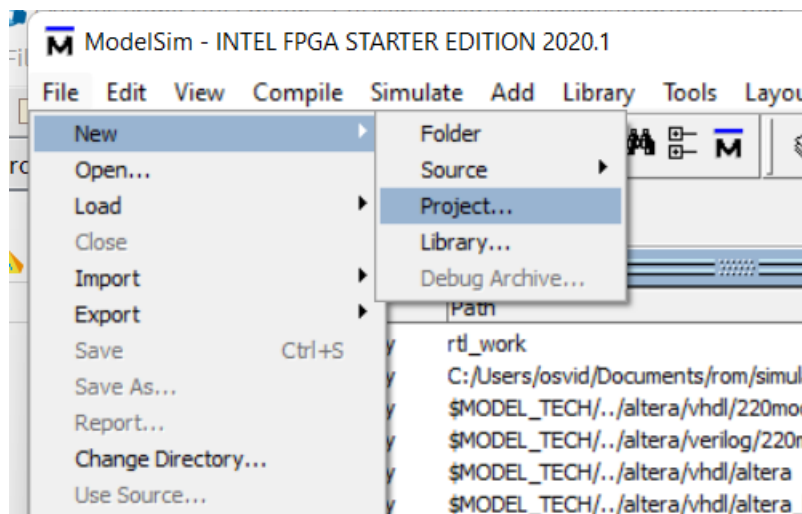
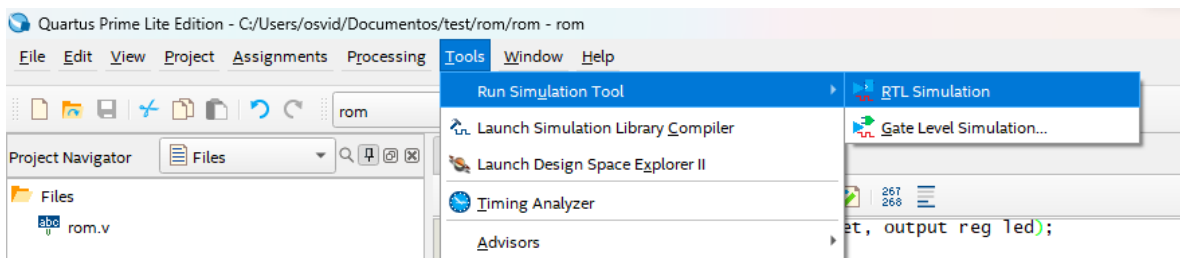


```
1 //Behavioral Model of a 4x4 Synchronous Read Only Memory in Verilog
2 // Brock J. LaMeres, Introduction to Logic Circuits & Logic Design with Verilog,
3 // Springer, 1st Edition, USA, 2017. pp 353
4
5 //modulo descriptivo
6 module rom (
7     input clk,
8     input [1:0] address,
9     output reg [3:0] data_out
10 );
11
12     reg [3:0] ROM [0:3];
13
14     initial
15     begin
16         ROM[0] = 4'b1110; //E
17         ROM[1] = 4'b0010; //2
18         ROM[2] = 4'b1111; //F
19         ROM[3] = 4'b0100; //4
20     end
21
22     always @(posedge clk)
23         data_out = ROM[address];
24
25 endmodule
```

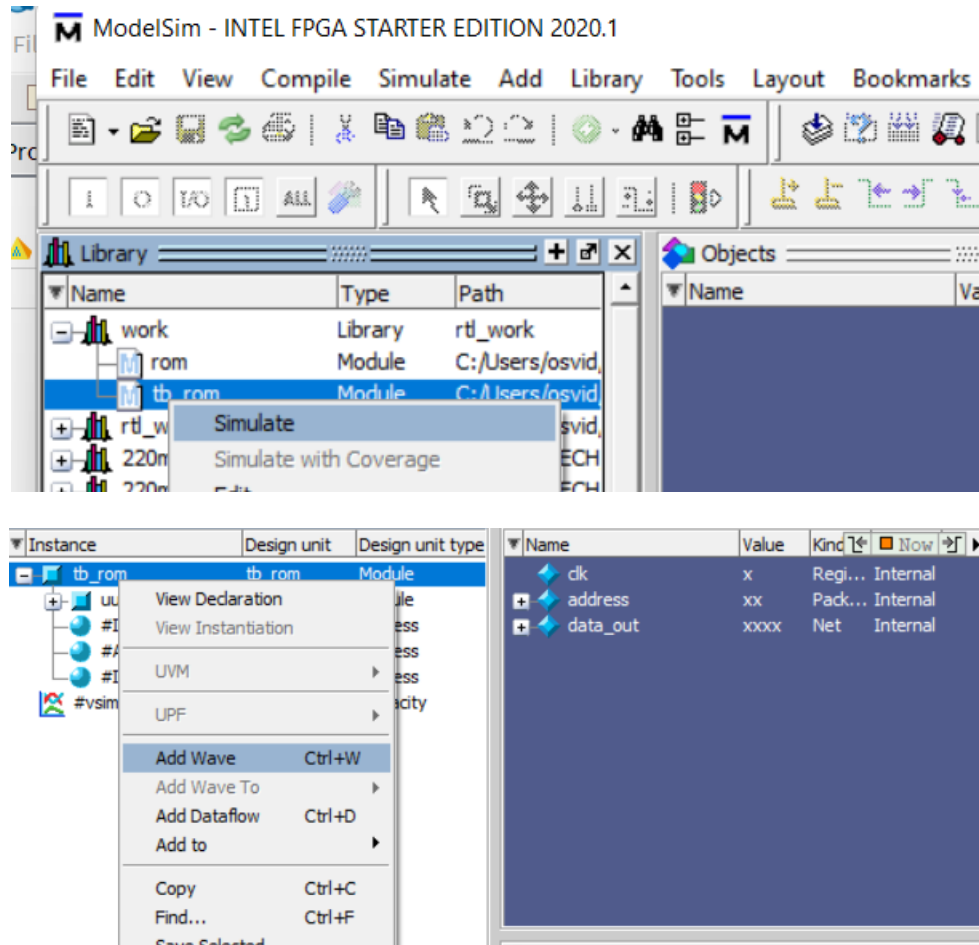
**7.- Compilar diseño:** Una vez que hayas configurado tu proyecto, se debe compilar el diseño haciendo clic en "Compile Design" desde el menú "Processing". El proceso de compilación puede tardar unos minutos.



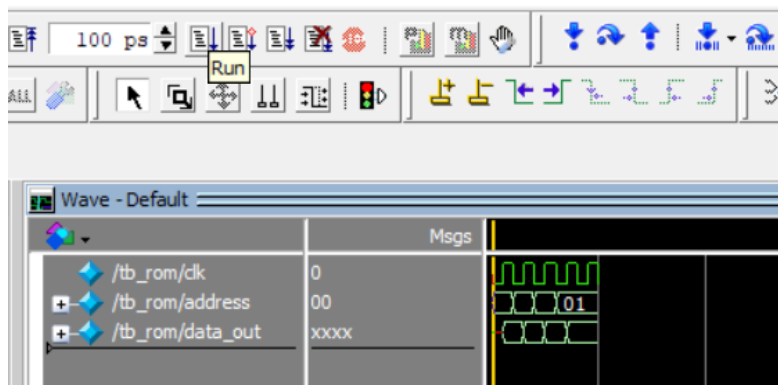
**8.- Simulación:** Para realizar una simulación, haz clic en "Simulate" desde el menú "Tools". En la ventana que aparece, selecciona "RTL Simulation". Y dentro del ModelSim creamos un nuevo proyecto.



**9.- Generar la simulación de la ROM:** Dentro del simulador ModelSim, para empezar a simular se debe seleccionar la Librería work y dentro estará nuestro proyecto, debemos seleccionar el testbench y darle a simular. Posteriormente debemos agregar nuestras entradas y salidas al wave



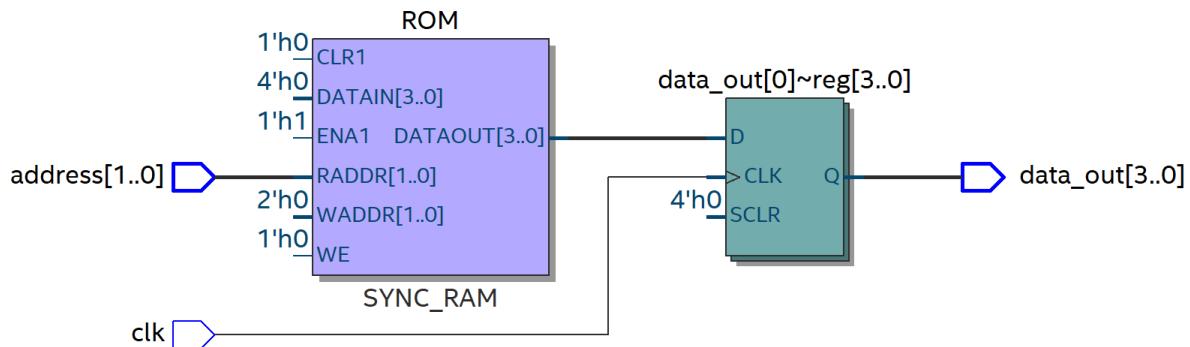
**10.- Visualizar la simulación en el wave:** Por ultimo debemos darle "run" para correr la simulación y listo, tendremos corriendo nuestra simulación de la memoria ROM.





## CODIGO DE MEMORIA ROM EN VERILOG

Este código Verilog implementa un módulo de memoria ROM (Read-Only Memory) de 4x4.



El módulo tiene dos entradas: clk, que es una señal de reloj, y address, que es la dirección de memoria de entrada. También tiene una salida data\_out, que es la salida de datos de la memoria.

El módulo utiliza una matriz de memoria ROM de 4x4, donde cada elemento es una palabra de 4 bits. La matriz ROM es inicializada en el bloque inicial, donde se definen los valores de cada palabra de memoria.

El módulo también tiene un bloque always que se activa en cada flanco positivo de la señal de reloj clk. En este bloque, la salida data\_out se actualiza con el valor de la palabra de memoria correspondiente a la dirección de entrada address.

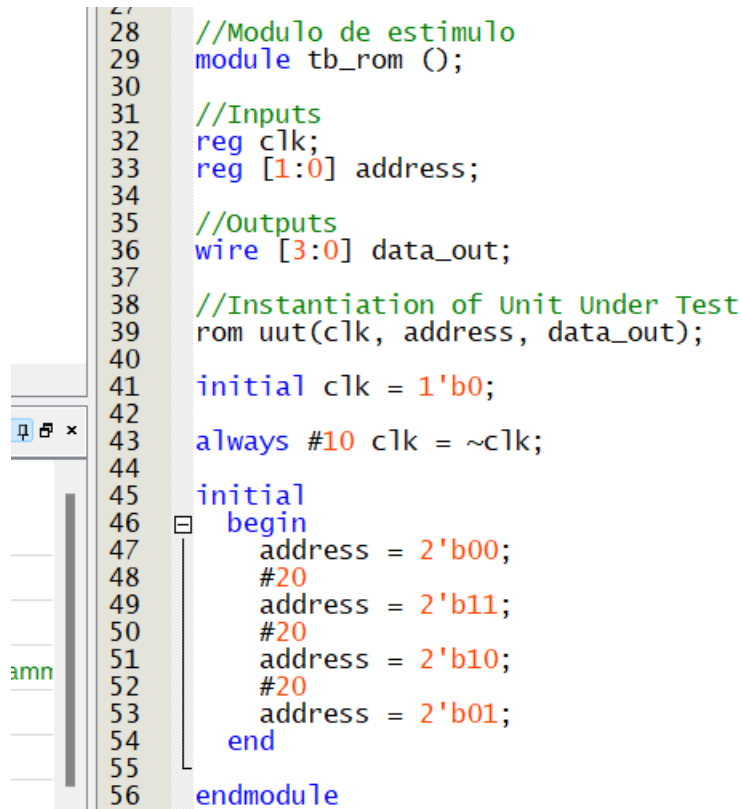
```

5 //modulo descriptivo
6 module rom (
7     input clk,
8     input [1:0] address,
9     output reg [3:0] data_out
10 );
11
12     reg [3:0] ROM [0:3];
13
14     initial
15     begin
16         ROM[0] = 4'b1110; //E
17         ROM[1] = 4'b0010; //2
18         ROM[2] = 4'b1111; //F
19         ROM[3] = 4'b0100; //4
20     end
21
22     always @(posedge clk)
23         data_out = ROM[address];
24
25 endmodule
26

```

El segundo módulo, llamado tb\_rom, es un banco de pruebas para el módulo de memoria ROM. Tiene las mismas entradas y salidas que el módulo de memoria ROM y se encarga de estimular las entradas para probar el funcionamiento del módulo.

El bloque initial dentro del módulo de estímulo configura la dirección de entrada address en secuencia, cada vez que se activa un delay de 20 unidades de tiempo. El bloque always dentro del módulo de estímulo también genera una señal de reloj clk con un periodo de 20 unidades de tiempo.

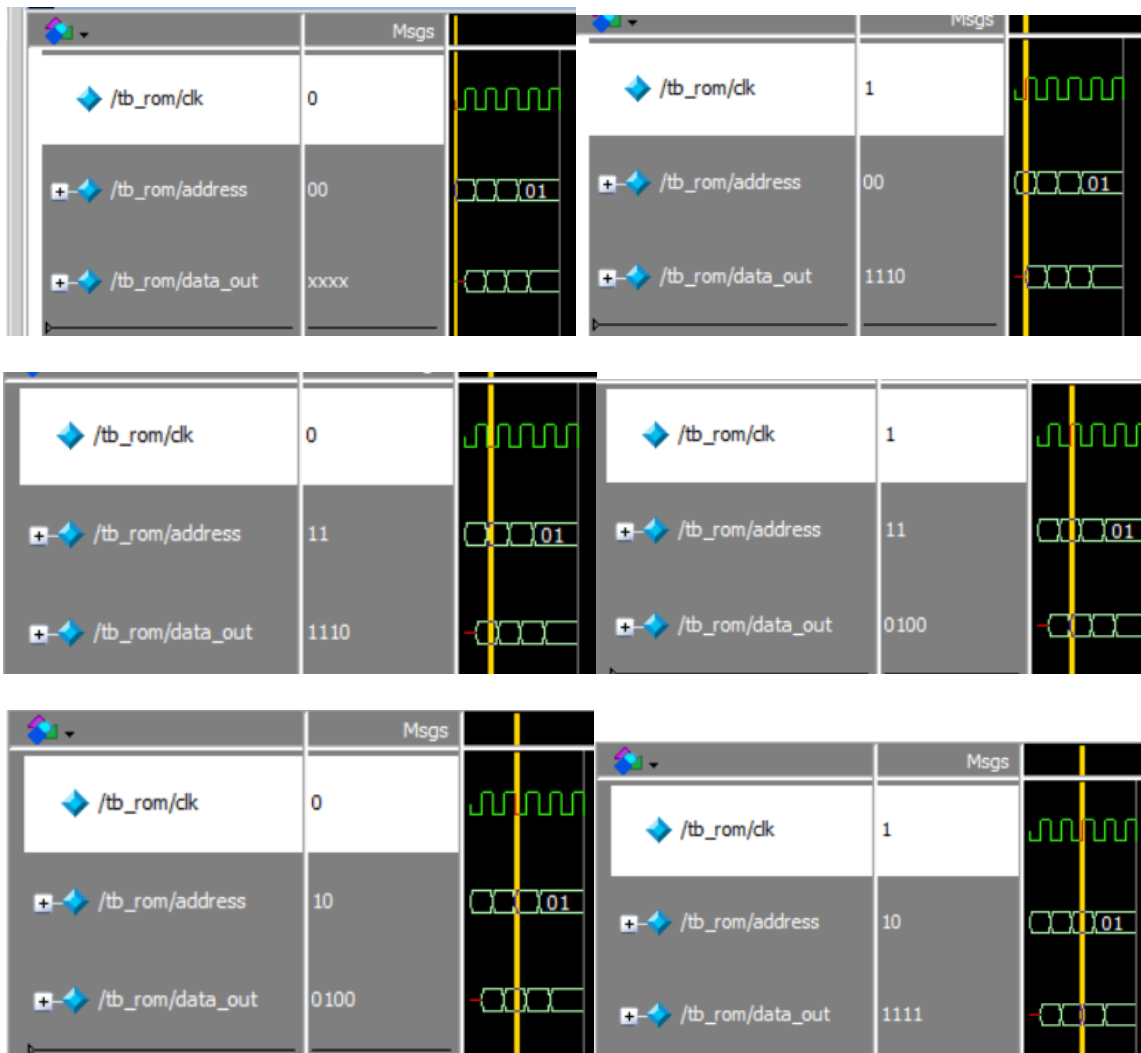


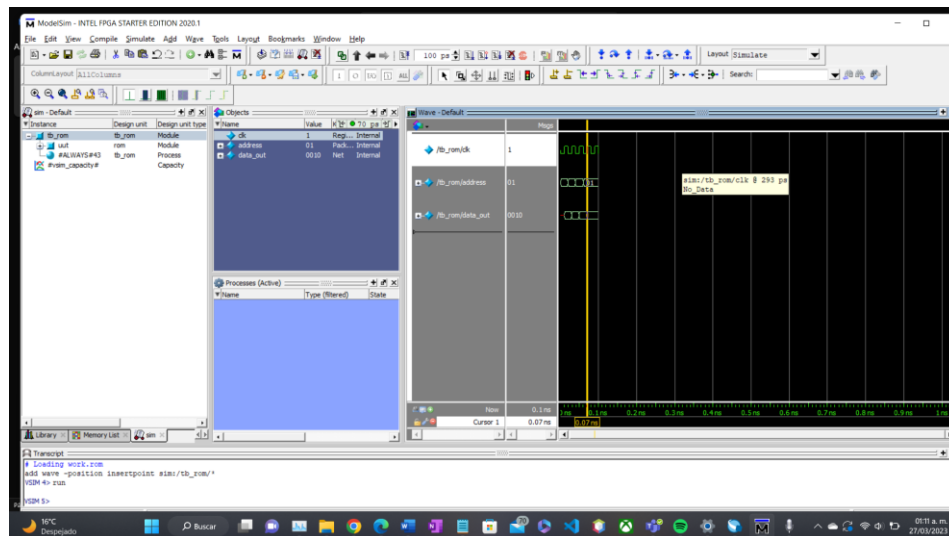
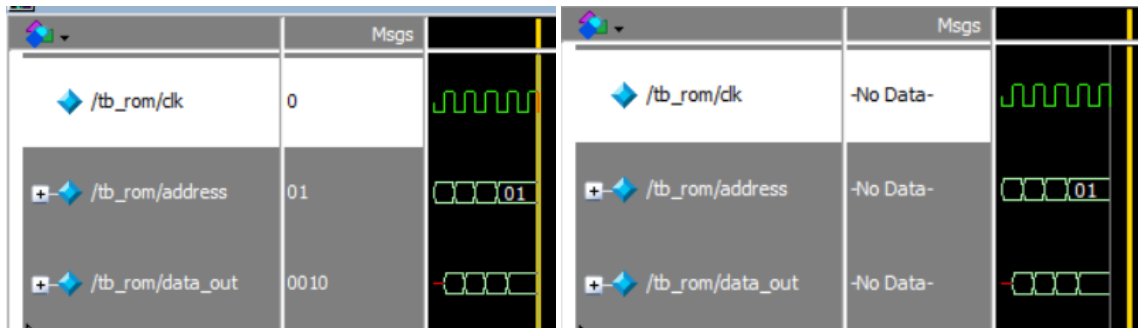
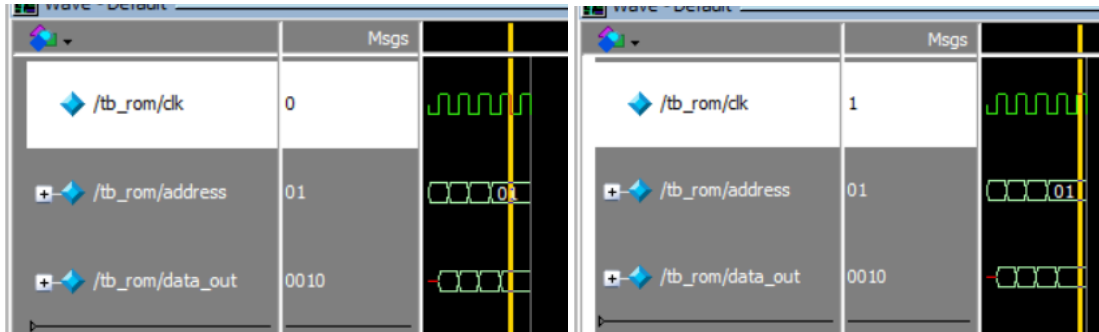
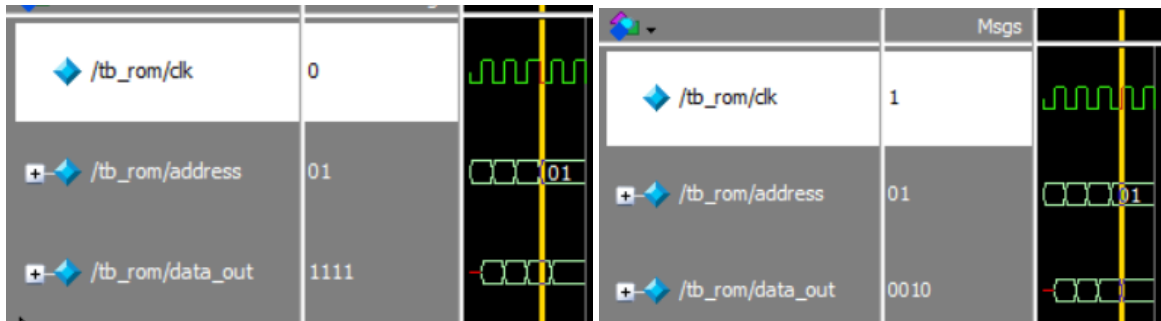
```
28 //Modulo de estimulo
29 module tb_rom ();
30
31 //Inputs
32 reg clk;
33 reg [1:0] address;
34
35 //Outputs
36 wire [3:0] data_out;
37
38 //Instantiation of Unit Under Test
39 rom uut(clk, address, data_out);
40
41 initial clk = 1'b0;
42
43 always #10 clk = ~clk;
44
45 initial
46 begin
47     address = 2'b00;
48     #20
49     address = 2'b11;
50     #20
51     address = 2'b10;
52     #20
53     address = 2'b01;
54 end
55
56 endmodule
```

En resumen, este código Verilog implementa una memoria ROM de 4x4 y un banco de pruebas que se encarga de estimular las entradas para verificar el correcto funcionamiento de la memoria.

## SIMULACIÓN DE MEMORIA ROM EN VERILOG

Como se puede observar en las simulaciones, se tienen 3 señales, la primera es el pulso de reloj definida como clk, la siguiente llamada “address” la cual define la dirección de memoria a la cual se esta accediendo, por ultimo “data\_out” son los datos que tiene almacenado la ROM, como podemos observar en la simulación se accede a cada una de las direcciones de memoria y se despliega correctamente el dato almacenado en la ROM





## **CONCLUSIONES**

En conclusión, la memoria ROM (Read-Only Memory) es un tipo de memoria no volátil que permite almacenar datos de manera permanente. En su implementación en Verilog, se puede utilizar un módulo descriptivo para modelar la memoria ROM, donde se define la cantidad de bits de entrada y salida, la cantidad de direcciones de memoria y el contenido de la memoria.

En el código realizado, se muestra un módulo descriptivo que implementa una memoria ROM de 4 direcciones con 4 bits de salida. Además, se utiliza un módulo de estímulo para probar el comportamiento de la memoria ROM mediante la selección de las direcciones de memoria y observando la salida correspondiente.

La implementación de una memoria ROM en Verilog es importante en la síntesis de circuitos digitales, ya que permite la programación de la memoria con datos de manera permanente, lo que es especialmente útil en aplicaciones como microcontroladores y sistemas embebidos.