Instituto Politécnico Nacional

Centro de Investigación en Computación

Lenguajes de descripción de hardware

Práctica 8 - In-System Memory Content Editor

Profesor:

M. en C. Osvaldo Espinosa Sosa

Por:

Ing. Ricardo Aldair Tirado Torres

Ciudad de México, 17 de junio de 2024

Tabla de contenido

1. Objetivos

- Implementar un programa que mande valores, con un determinado retardo, al puerto paralelo de 8 bits del procesador NIOS II, previamente creado.
- Generar un entorno de simulación del procesador NIOS II para imprimir un mensaje en la consola y observar los valores del puerto de salida en el visor de formas de onda.

2. Manejo de una memoria de 32 localidades de 8 bits

Actividad 1

Crear una memoria inicializada de 32x8 con archivo MIF como se vio en clase, habilitar la opción de uso del *In-System Memory Content Editor*. Automatizar el despliegue del contenido de las 32 localidades de memoria, usando un contador a una razón de localidad por segundo. Usar la tarjeta DE2-115.

La visualización RTL de la memoria de 32 localidades de 8 bits se muestra en la Figura 1. De manera interna, la implementación en hardware utiliza una instancia de memoria perteneciente a la herramienta de *IP Catalog* (ver Figura 2). Para inicializar la memoria, se generó un archivo *.mif* cuyos valores se presentan en la Figura 3, siendo estos el valor de la dirección incrementado en uno. Las simulaciones se visualizan en la Figura 4. Se observa que por cada ciclo de reloj se incrementa el valor de la dirección en uno y el valor almacenado en la localidad actual se ve reflejado en la salida Q. Una observación importante es que la señal WE esta en bajo, por lo que la memoria esta en modo lectura. Con respecto a la implementación en la tarjeta DE2-115, se utilizó la herramienta *In-System Memory Content Editor*, como se visualiza en la Figura 6

En los Anexos se localiza la descripción de la memoria de 32 localidades de 8 bits. Al emplear un elemento proveniente de la herramienta de *IP Catalog*, el código se reduce a describir las entradas y salidas y llamar a la instancia dentro del módulo principal.

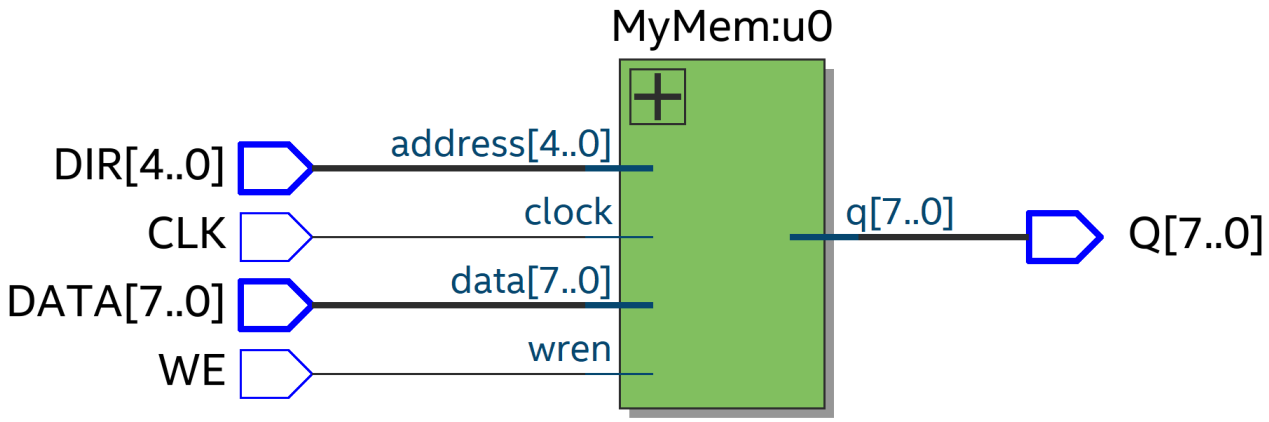

Figura 1: Diagrama RTL de la memoria de 32 localidades de 8 bits, instanciada con un componente de IP Catalog.

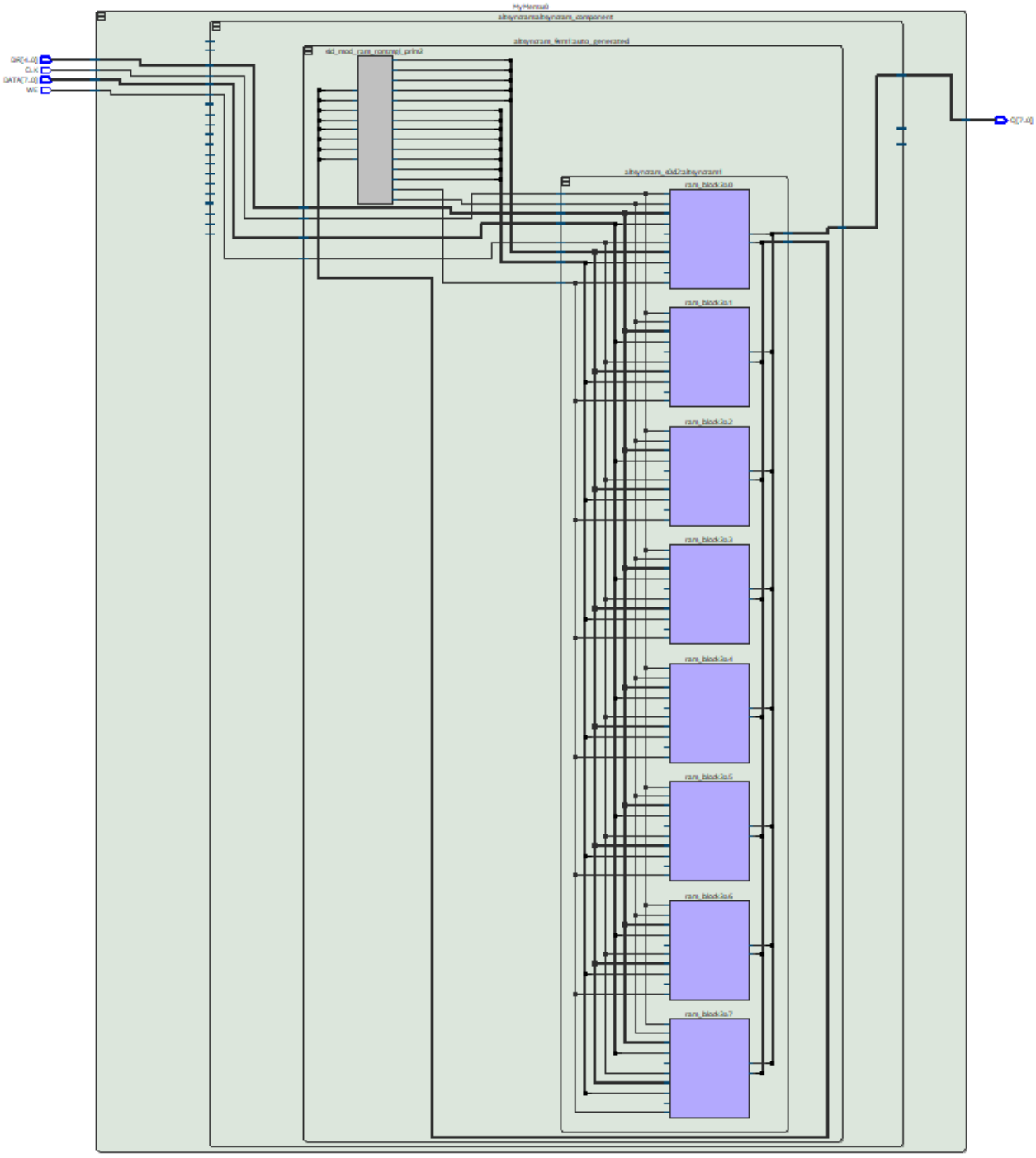

Figura 2: Diagrama RTL de la memoria de 32 localidades de 8 bits, instanciada con un componente de IP Catalog (vista interna).

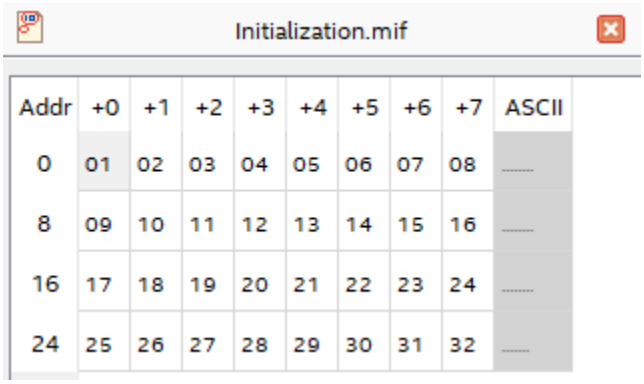

Figura 3: Visualización del archivo de inicialización de la memoria de 32 localidades de 8 bits.

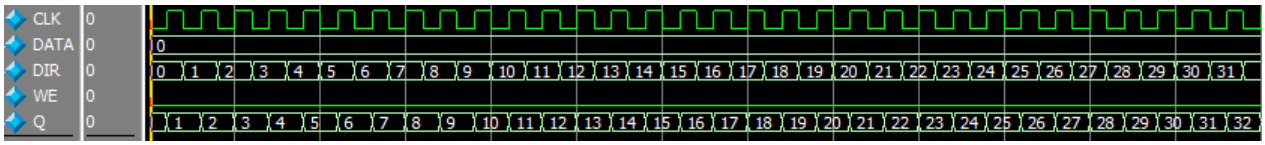

Figura 4: Simulación de la memoria de 32 localidades de 8 bits, en el visor de formas de onda de ModelSim.

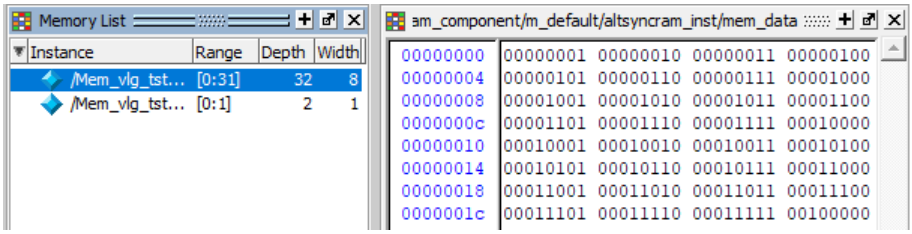

Figura 5: Simulación de la memoria de 32 localidades de 8 bits, en el visor de formas de onda de ModelSim.

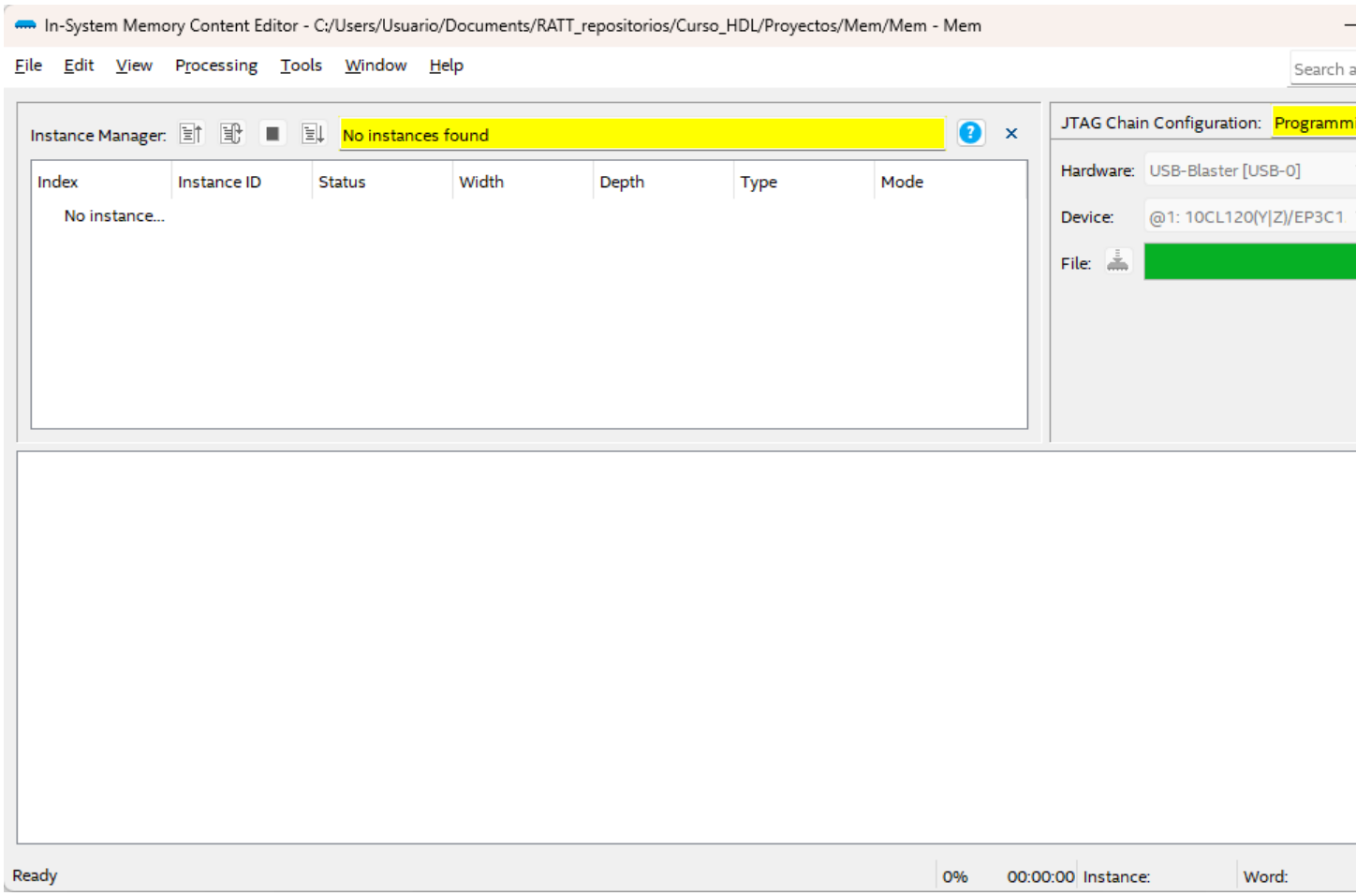

Figura 6: Simulación de la memoria de 32 localidades de 8 bits, en el visor de formas de onda de ModelSim.

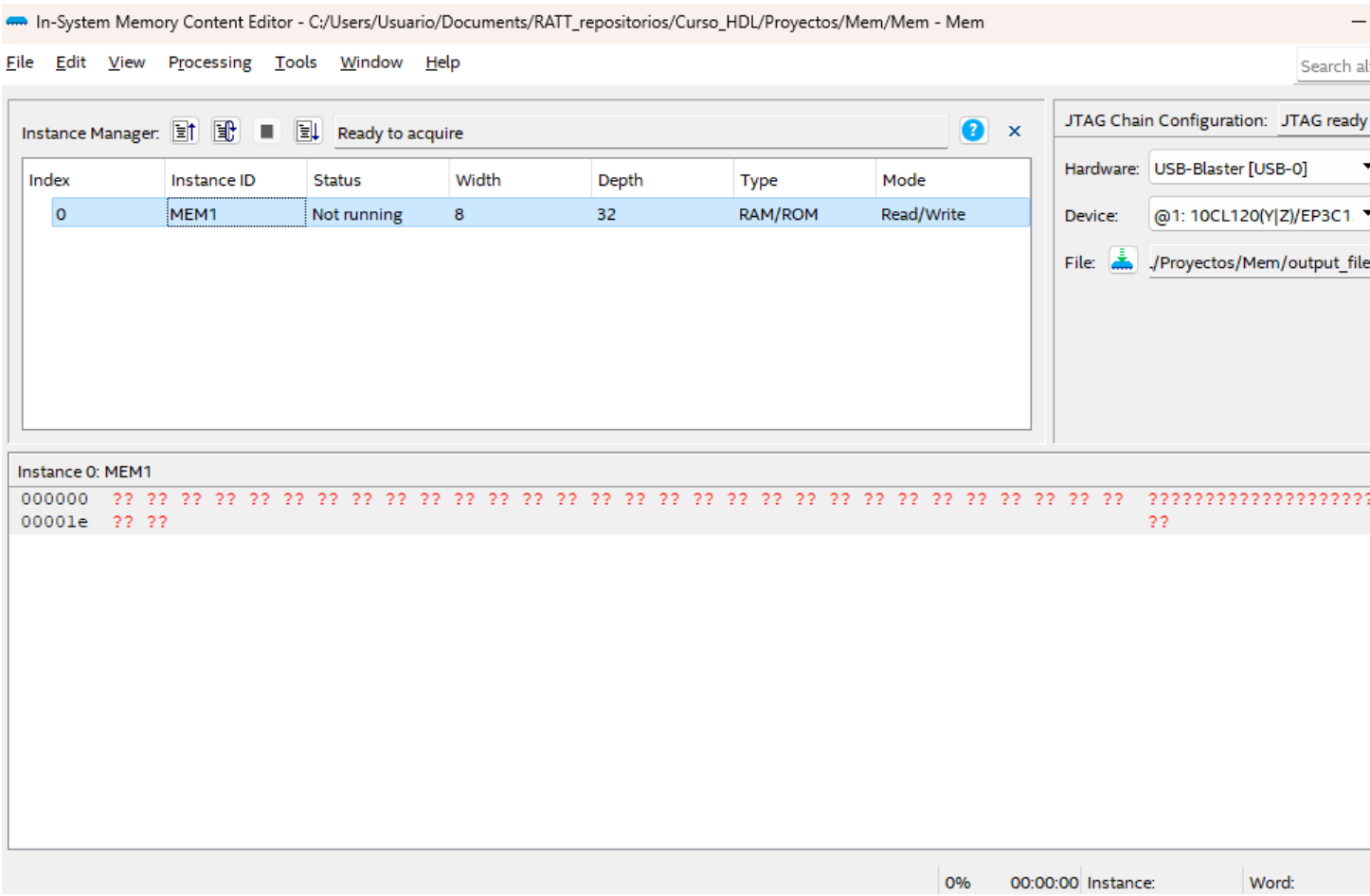

Figura 7: Simulación de la memoria de 32 localidades de 8 bits, en el visor de formas de onda de ModelSim.

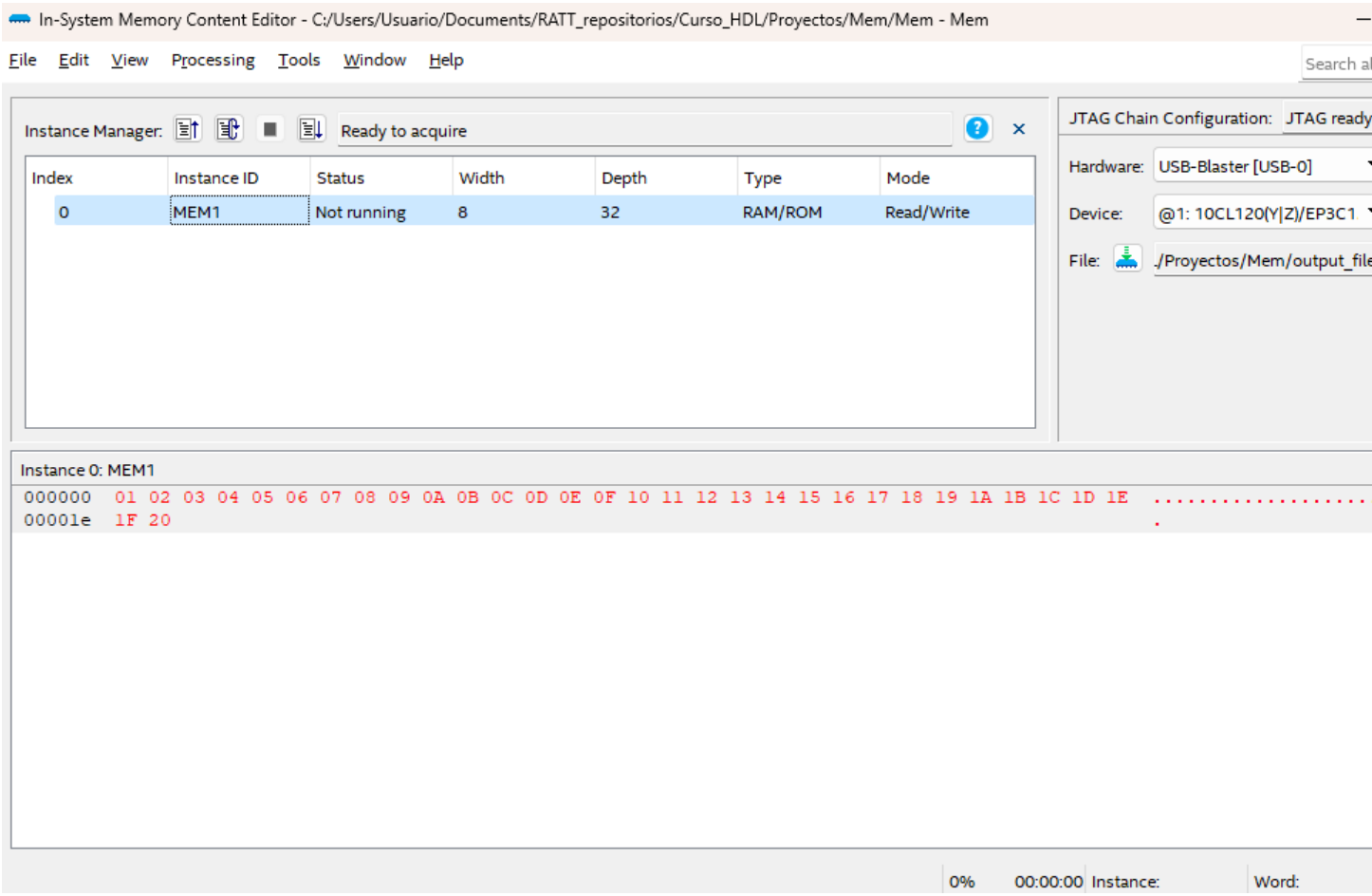

Figura 8: Simulación de la memoria de 32 localidades de 8 bits, en el visor de formas de onda de ModelSim.

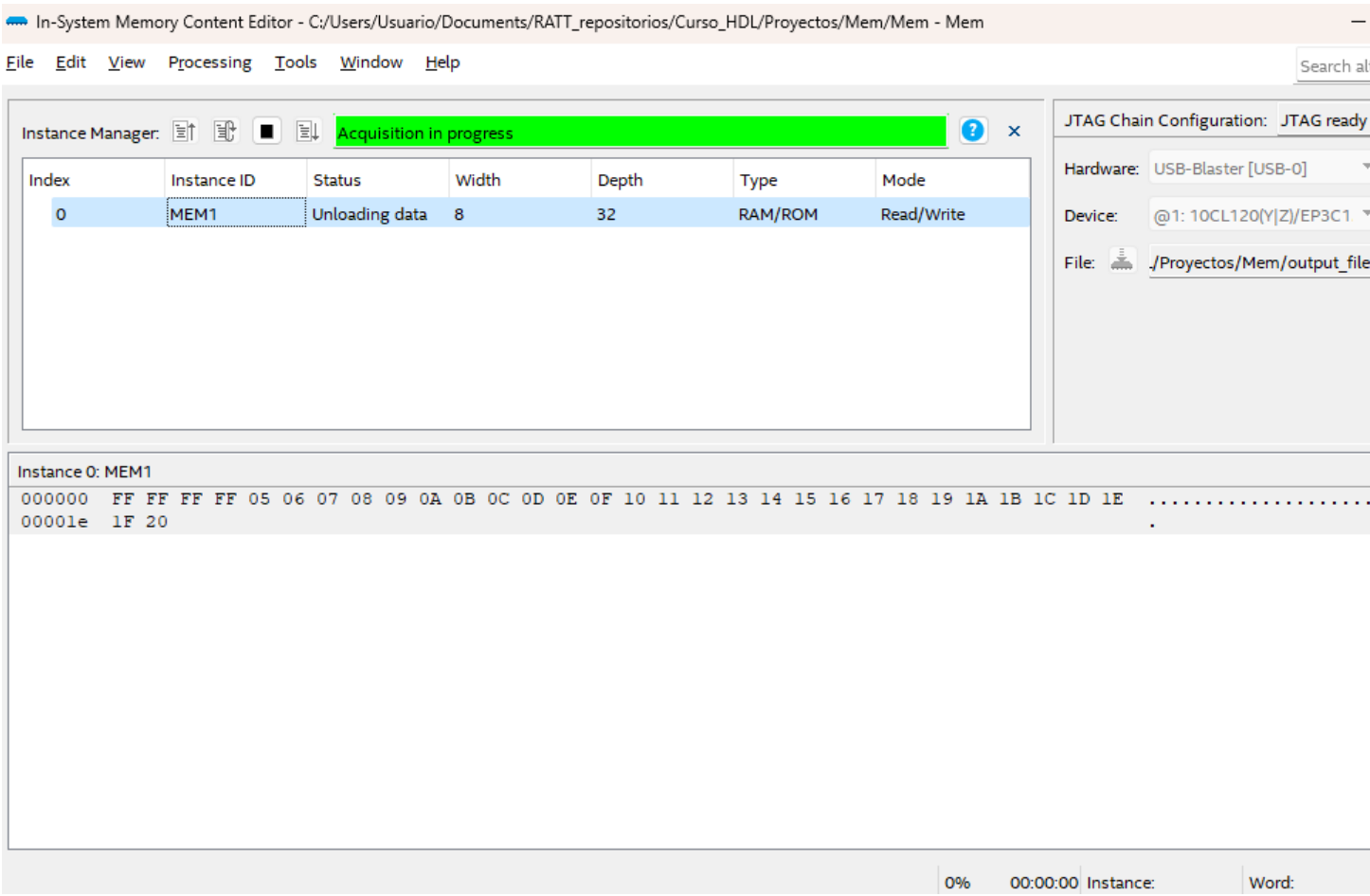

Figura 9: Simulación de la memoria de 32 localidades de 8 bits, en el visor de formas de onda de ModelSim.

3. Conclusiones

En conclusión, se implementó la simulación del programa de asignación de valores en la salida del procesador NIOS II de manera exitosa.

Utilizando el procedimiento de la presentación vista en clase para simular un NIOS II, se escribieron los comandos necesarios para simular la impresión de una cadena de texto en la consola de ModelSim y en el visor de formas de onda se observó el tiempo requerido para cargar el mensaje y para mandar valores al puerto de salida del procesador. Se empleó la herramienta de *Eclipse* para variar el retardo del programa y con la herramienta de ModelSim se generó el archivo *.do* para los comandos.

En los Anexos se pueden encontrar los códigos implementados.

4. Anexos

4.1. Descripciones del hardware

```
module Mem(
input      CLK, WE,
input  [4:0]  DIR,
input  [7:0]  DATA,
output [7:0]  Q
);

MyMem u0(DIR, CLK, DATA, WE, Q);

endmodule
```

Programa 1: Descripción en Verilog de la memoria de 32 localidades de 8 bits, utilizando una instancia de *IP Catalog*.

4.2. Bancos de pruebas (*test benches*)

```
`timescale 1 ns/ 1 ps
module Mem_vlg_tst();
 reg      CLK;
 reg  [7:0]  DATA;
 reg  [4:0]  DIR;
 reg      WE;
 wire  [7:0]  Q;

 Mem i1 (
  .CLK(CLK),
  .DATA(DATA),
  .DIR(DIR),
  .Q(Q),
  .WE(WE)
 );

 initial
 begin
  $display("Running testbench at CIC");
  DIR = 0; WE = 0; DATA = 0; CLK = 0;
 end

 always
```

```
24  begin
25   #10; CLK = ~CLK;
26   #10; CLK = ~CLK;
27   DIR = DIR + 1;
28  end
29
30 endmodule
```

Programa 2: Banco de prueba para el Programa 1.