INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO


DISEÑO DE SISTEMAS DIGITALES

Flores Escobar José Antonio

# *Memoria ROM*

Integrantes:

Álvarez Hernández Gabriel Alexander

García Quiroz Gustavo Iván

Huesca Laureano José Alejandro

Muñoz Valdivia Irving Omar

Pedroza Villagomez Emir

## *Introducción*

La memoria de solo lectura (ROM, por sus siglas en inglés) es un tipo crucial de almacenamiento no volátil. La ROM se utiliza para almacenar datos que no cambian con el tiempo y que deben estar disponibles incluso después de que el sistema se apague. En este proyecto se ilustra cómo una ROM puede integrarse y usarse para almacenar y mostrar datos en un display de 7 segmentos, a través de un sistema de módulos interconectados en Verilog.

*¿Qué es una ROM?*

La ROM es un tipo de memoria de solo lectura que se programa generalmente durante el proceso de fabricación del chip. A diferencia de la RAM (memoria de acceso aleatorio), los datos en la ROM no se pierden cuando se apaga la energía. Existen varios tipos de ROM, incluyendo:

Mask ROM: Programada durante la fabricación.

PROM (Programmable ROM): Programable una sola vez por el usuario.

EPROM (Erasable Programmable ROM): Se puede borrar y reprogramar usando luz ultravioleta.

EEPROM (Electrically Erasable Programmable ROM): Se puede borrar y reprogramar eléctricamente.

*Funcionamiento de la ROM*

La ROM almacena datos en una matriz de celdas de memoria, donde cada celda contiene un bit de información. La estructura básica incluye una serie de líneas de dirección y de datos:

Líneas de dirección: Utilizadas para seleccionar la ubicación de memoria que se va a leer.

Líneas de datos: Utilizadas para leer el dato almacenado en la ubicación seleccionada.

Cuando se proporciona una dirección y se habilita la lectura, la ROM coloca el dato correspondiente en las líneas de datos.

## *Desarrollo.*

```
/*
   Proyecto: ROM
   Archivos: ROM.v
   Asignatura: DSD
   Prof: Flores Escobar Jose Antonio
   Equipo:  Álvarez Hernández Gabriel Alexander
            García Quiroz Gustavo Iván
            Huesca Laureano José Alejandro
            Muñoz Valdivia Irving Omar
            Pedroza Villagomez Emir
*/

module ROM #(
    parameter D = 8,
    parameter W = 32
)
(
    input  [D-1:0] addr_i,
    input          rden_i,
    output reg [W-1:0] dato_o
);

    always @(*) begin
        if (rden_i) begin
            case (addr_i)
                8'h0: dato_o = 32'h01234567;
                8'h1: dato_o = 32'h76543210;
                8'h2: dato_o = 32'hABC24681;
                8'h3: dato_o = 32'hCD120201;
                8'h4: dato_o = 32'hCACA2357;
                8'h5: dato_o = 32'hF56AC87F;
                8'h6: dato_o = 32'hDED05BA7;
                8'h7: dato_o = 32'h11111111;
                default: dato_o = 32'h00000000;
            endcase
        end else begin
            dato_o = 32'h00000000;
        end
    end

endmodule

```

ROM.

El módulo ROM almacena una serie de datos de 32 bits que se pueden leer utilizando una dirección de entrada. Al habilitar la lectura (rden_i), el módulo entrega el dato correspondiente a la dirección dada.

```
/*
   Proyecto: ROM
   Archivos: ROM_Board.v
   Asignatura: DSD
   Prof: Flores Escobar Jose Antonio
   Equipo:  Álvarez Hernández Gabriel Alexander
            García Quiroz Gustavo Iván
            Huesca Laureano José Alejandro
            Muñoz Valdivia Irving Omar
            Pedroza Villagomez Emir
*/
module ROM_Board #(
    parameter D = 3,
    parameter W = 32
)
(
    input  wire [D-1:0] addr_i,           // switch 1,2,3
    input  wire         rden_i,           // read enable
    input  wire [D-1:0] byteselector_i,   // byte selector
    output wire [0:6]   display_o,        // display output
    output wire         display_enable
);

    // Sección de definición de señales
    wire [W-1:0] dato_w;
    reg  [3:0]   datodisplay_o;           // Dato que se va a mostrar
    wire         slow_clk;

    // Instancia de la ROM
    ROM #(
        .D (D),
        .W (W)
    ) ROM_U0 (
        .addr_i (addr_i),
        .rden_i (rden_i),
        .dato_o (dato_w)
    );

    // Sección de selección de byte
    always @(*) begin
```

```
40      always @(*) begin
41          case(byteselector_i)
42              3'b000: datodisplay_o = dato_w[3:0];
43              3'b001: datodisplay_o = dato_w[7:4];
44              3'b010: datodisplay_o = dato_w[11:8];
45              3'b011: datodisplay_o = dato_w[15:12];
46              3'b100: datodisplay_o = dato_w[19:16];
47              3'b101: datodisplay_o = dato_w[23:20];
48              3'b110: datodisplay_o = dato_w[27:24];
49              3'b111: datodisplay_o = dato_w[31:28];
50          endcase
51      end
52
53      // Instancia del divisor de frecuencia
54      DivFreq #(
55          .freqdev(10_000_000),     // Frecuencia de desarrollo de 10MHz
56          .freqfinal(4)             // Frecuencia final dividida en 4
57      ) div_freq_instance (
58          .clk_i(clk_i),            // Señal de reloj de entrada
59          .rst_ni(rst_ni),          // Señal de reset
60          .clk_o(slow_clk)          // Salida de reloj dividido
61      );
62
63      // Instancia del display
64      display display_instance (
65          .cuenta_i (datodisplay_o),
66          .enable_i (1'b1),
67          .display_o (display_o),
68          .daenable_o (display_enable)
69      );
70
71  endmodule
```

## ROM_Board.

El módulo ROM_Board es el módulo superior que interconecta todos los componentes del sistema. Toma entradas de dirección (addr_i), habilitación de lectura (rden_i), y un selector de bytes (byteselector_i). Utiliza estas entradas para leer datos de la ROM y seleccionar el byte específico que se mostrará en el display. Además, incluye un divisor de frecuencia (DivFreq) para ajustar la velocidad del reloj.

```v
/*
    Proyecto: ROM
    Archivos: display.v
    Asignatura: DSD
    Prof: Flores Escobar Jose Antonio
    Equipo:  Álvarez Hernández Gabriel Alexander
             García Quiroz Gustavo Iván
             Huesca Laureano José Alejandro
             Muñoz Valdivia Irving Omar
             Pedroza Villagomez Emir
*/
module display (
    input   [3:0] cuenta_i,
    input         enable_i,
    output  [0:6] display_o,
    output        daenable_o
);

    reg [0:6] display_w;
    assign daenable_o = 1'b1;

    always @(*) begin
        case (cuenta_i)
            4'b0000: display_w = 7'b0111111; // 0
            4'b0001: display_w = 7'b0000110; // 1
            4'b0010: display_w = 7'b1011011; // 2
            4'b0011: display_w = 7'b1001111; // 3
            4'b0100: display_w = 7'b1100110; // 4
            4'b0101: display_w = 7'b1101101; // 5
            4'b0110: display_w = 7'b1111101; // 6
            4'b0111: display_w = 7'b0000111; // 7
            4'b1000: display_w = 7'b1111111; // 8
            4'b1001x: display_w = 7'b1101111; // 9
            4'b1010: display_w = 7'b1110111; // A
            4'b1011: display_w = 7'b0111111; // B
            4'b1100: display_w = 7'b0111001; // C
            4'b1101: display_w = 7'b1011110; // D
            4'b1110: display_w = 7'b1111001; // E
            4'b1111: display_w = 7'b1110001; // F
            default: display_w = 7'b0000000; // Default
        endcase
    end

    assign display_o = (enable_i) ? display_w : ~display_w;

endmodule
```

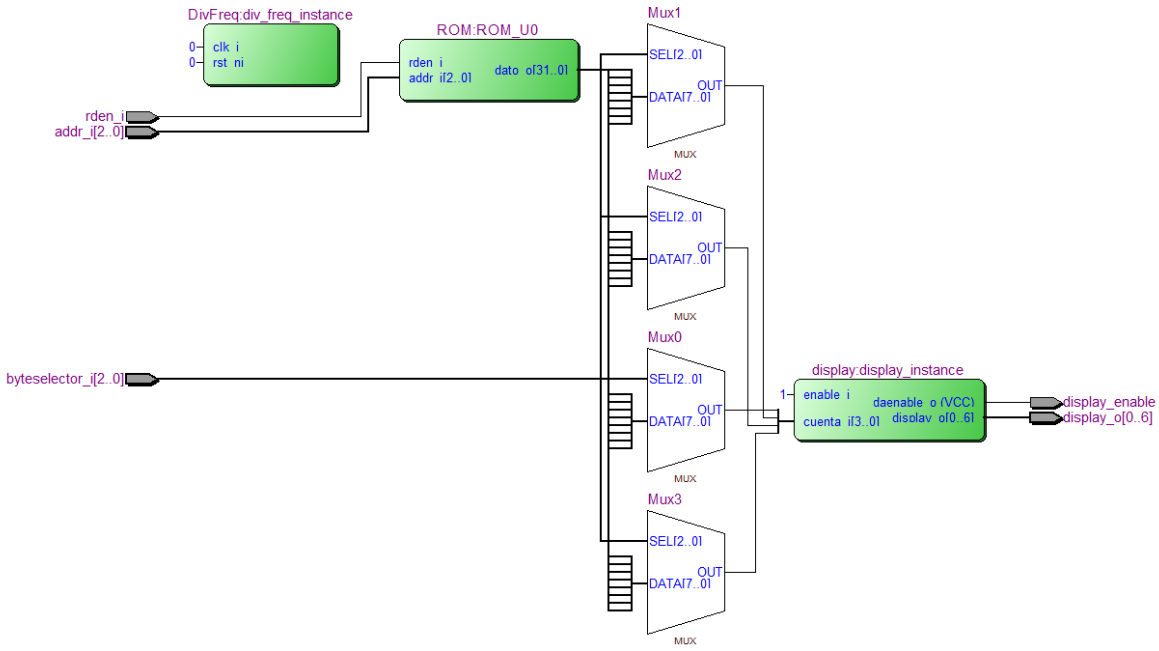
## display

El módulo display convierte un valor de 4 bits en la codificación correspondiente para un display de 7 segmentos. Dependiendo del valor de entrada, activa el patrón de segmentos adecuado para representar números hexadecimales (0-9, A-F).

```
/*
   Proyecto: ROM
   Archivos: DivFreq.v
   Asignatura: DSD
   Prof: Flores Escobar Jose Antonio
   Equipo:  Álvarez Hernández Gabriel Alexander
            García Quiroz Gustavo Iván
            Huesca Laureano José Alejandro
            Muñoz Valdivia Irving Omar
            Pedroza Villagomez Emir
*/
module DivFreq #(
    parameter freqdev = 10_000_000, // 10MHz
    parameter freqfinal = 4 // freqdev Dividido en 4
)(
    input  clk_i,
    input  rst_ni,
    output reg clk_o
);

    reg [31:0] counter_r;

    always @(posedge clk_i or posedge rst_ni) begin
        if (rst_ni) begin
            counter_r <= 32'b0;
            clk_o <= 1'b0; // Comenzamos con el reloj en bajo
        end else begin
            if (counter_r >= (freqdev / freqfinal) - 1) begin
                counter_r <= 32'b0;
                clk_o <= ~clk_o;
            end else begin
                counter_r <= counter_r + 32'b1;
            end
        end
    end

endmodule

```

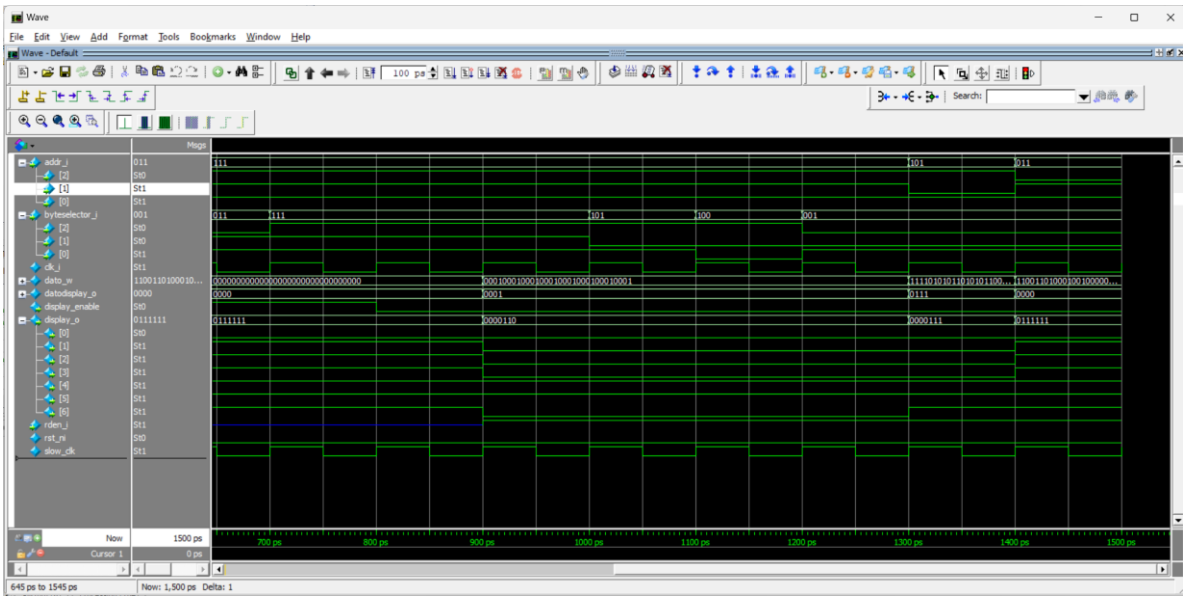
## DivFreq

El módulo DivFreq reduce la frecuencia del reloj de entrada para producir una señal de reloj más lenta. Esto es útil para asegurar que el display funcione a una velocidad visible y estable, adecuada para la visualización humana.



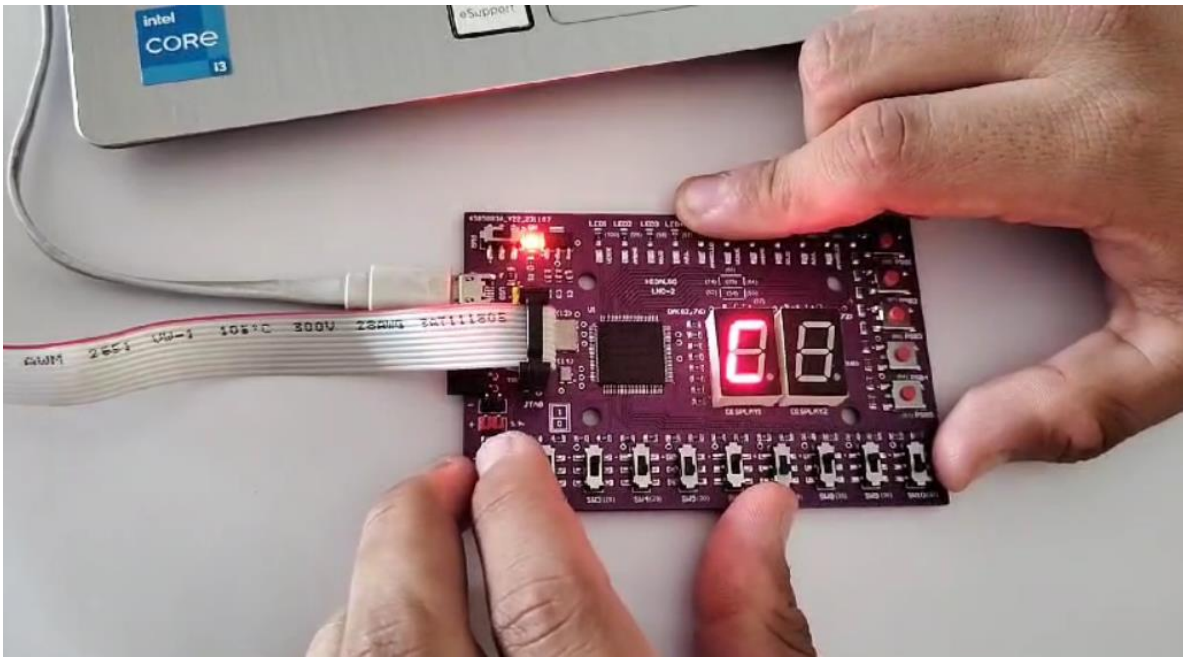

RTLViewer.

Podemos observar cómo están conectados los módulos y componentes del proyecto, así como los datos a procesar, las entradas y salidas que vamos a obtener al simularlo, o programarlo en la FPGA.




## Simulación.

Observamos como funciona en la simulación el proyecto generado, teniendo como entrada el selector de bit y la dirección en la memoria. Obteniendo en la salida del display, el dato almacenado, logrando movernos entre la misma dirección de memoria, solo en diferente bit.



## *Conclusiones.*

El proyecto presentado demuestra una aplicación práctica de la memoria ROM en sistemas digitales, ilustrando cómo los datos almacenados de manera no volátil pueden ser recuperados y mostrados en un display de 7 segmentos. La ROM se utiliza aquí para almacenar valores predefinidos que se leen y procesan.

El módulo `ROM_Board` coordina la lectura de datos desde la ROM y la selección del byte específico que se mostrará en el display. El módulo `display` se encarga de traducir los valores de datos en señales específicas para un display de 7 segmentos, permitiendo la representación visual de números hexadecimales.

Este diseño no solo resalta la funcionalidad de la ROM, sino también la importancia de los componentes auxiliares que facilitan la interacción con el usuario. A través de este proyecto, se destaca la relevancia de la memoria de solo lectura en aplicaciones que requieren almacenamiento persistente de datos y su posterior visualización, un concepto esencial en la electrónica digital.

## *Bibliografía.*


- Demarco, S. (2024). *Memoria ROM: Tipos, usos y características a tener en cuenta*. SabDemarco. Recuperado de: https://sabdemarco.com/memoria-rom-tipos-usos-y-caracteristicas

- NewEsc. (2024). *Diferencias entre memorias RAM y ROM: Conceptos básicos*. Recuperado de: https://newesc.com/diferencias-entre-memorias-ram-y-rom-conceptos-basicos/