

Proyecto Final

Laboratorio de Electrónica Digital I

Buffer Memoria-VGA-MEF

Profesor:

Ferney Alberto Beltrán Molina

Autores:

Ana Isabella Goyeneche Fonseca

Diego Andrés Quintero Rois

Oscar Santiago Suarez Aguilar

C O N S I D E R A C I O N E S

Tamaño Máximo de Buffer de Memoria :

$$640 \times 480 \times 12 = 3686400 \text{ bits} / 1024 = 3600 \text{ kbits}$$

Pinout Necesario :

- Red PIN_2
- Green PIN_1
- Blue PIN_144
- VS PIN_143
- HS PIN_142

$$\text{Memoria necesaria máxima: } 640 \times 480 \times 3 = 921600 \text{ bits} / 1024 = \mathbf{900 \text{ kbits}}$$

Estrategia Inicial:

- Resolución de Pantalla = $207 * 1024/3^*$
*width X height = 70656 bits
- width = height = $\text{SQRT}(70656)$
= 265,811 pixels
- Se define width = height = **256 pixeles**

PANTALLA

15 pulgadas TFT LCD visualización 0.297 mm

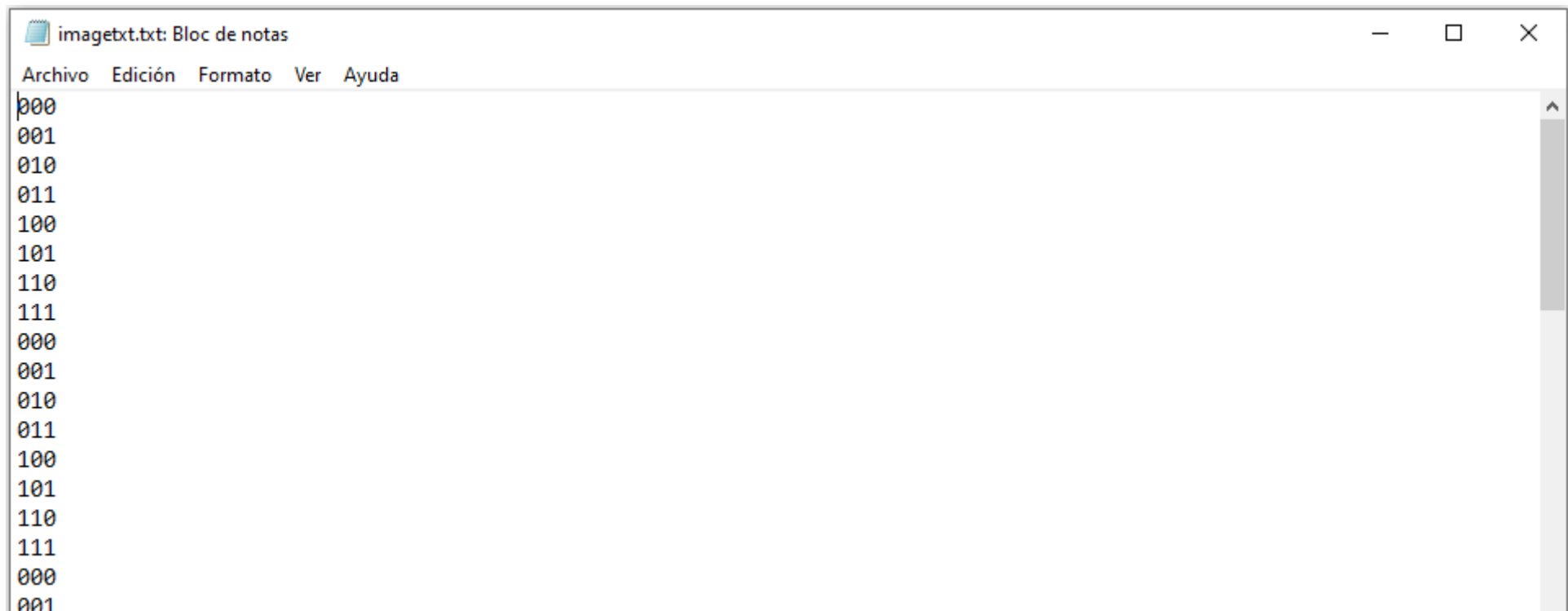
Dot pitch 1024 x 768 @ 60 Hz, con Máxima
resolución

30 – 61 kHz y 56 – 76 Hz de frecuencia vertical y
frecuencia horizontal respectivamente

de contraste de 450: 1.

BUFFER RAM INIT.

RGB 000 --> Negro 001 --> Azul 010 --> Verde 011 --> Cian 100 --> Rojo 101 --> Violeta 110 --> Amarillo 111 --> Blanco



```

module test_VGA(
    input wire clk,           // board clock: 50 MHz
    input wire rst,          // reset button

    // VGA input/output
    output wire VGA_Hsync_n, // horizontal sync output
    output wire VGA_Vsync_n, // vertical sync output
    output wire VGA_R,       // 1-bit VGA red output
    output wire VGA_G,       // 1-bit VGA green output
    output wire VGA_B,       // 1-bit VGA blue output
    output wire clkout,

    // input/output

    input wire bntr,
    input wire bntl

);

// TAMAÑO DE visualización
parameter CAM_SCREEN_X = 256;
parameter CAM_SCREEN_Y = 256;

localparam AW = 8; // LOG2(CAM_SCREEN_X*CAM_SCREEN_Y)
localparam DW = 3; //Numero de bits RGB

// El color es RGB 111
localparam RED_VGA = 3'b100;
localparam GREEN_VGA = 3'b010;
localparam BLUE_VGA = 3'b001;

```

```

assign VGA_R = data_RGB111[2];
assign VGA_G = data_RGB111[1];
assign VGA_B = data_RGB111[0];

```

```

assign clk50M=clk;
clock75 clk75(
    .inclk0(clk50M),

```

```

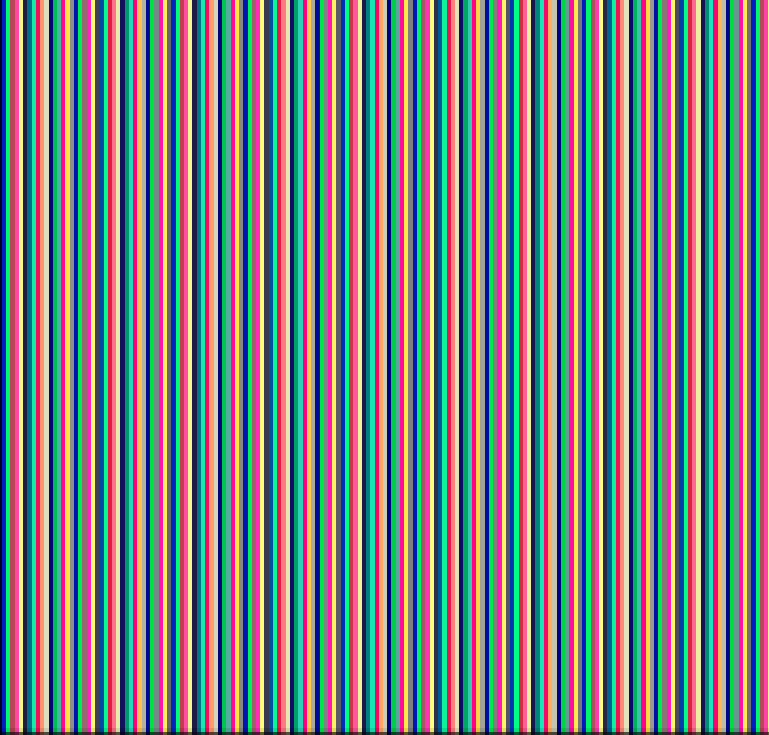
    divisor_de_frecuencia(
        .clk(clk75M),
        .clk_out(clkout)
    );

```

```

always @ (VGA_posX, VGA_posY) begin
    if ((VGA_posX>CAM_SCREEN_X-1) || (VGA_posY>CAM_SCREEN_Y-1))
        DP_RAM_addr_out=0;
    else
        DP_RAM_addr_out=VGA_posX;
end

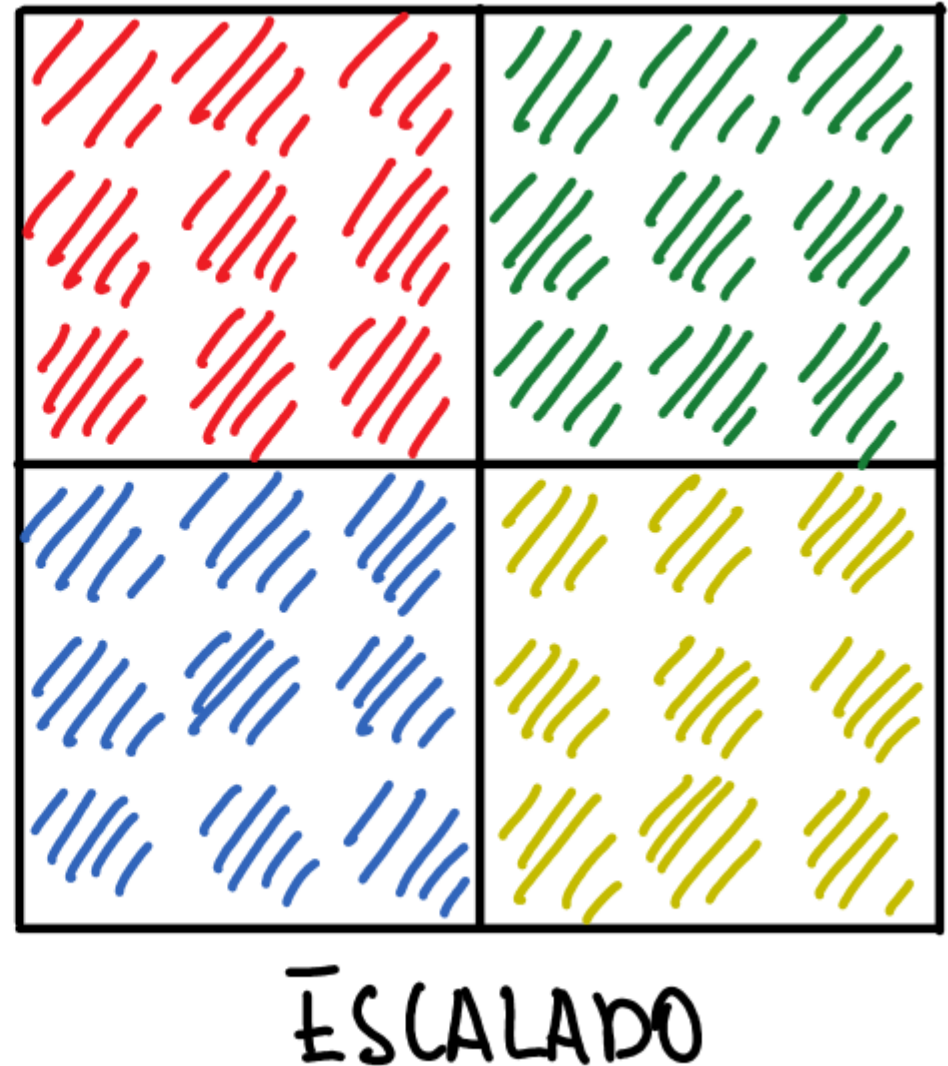
```




VIDEO DE PRUEBA

<https://youtu.be/CAzSEY2RYaQ>

ESTRATEGIA FINAL





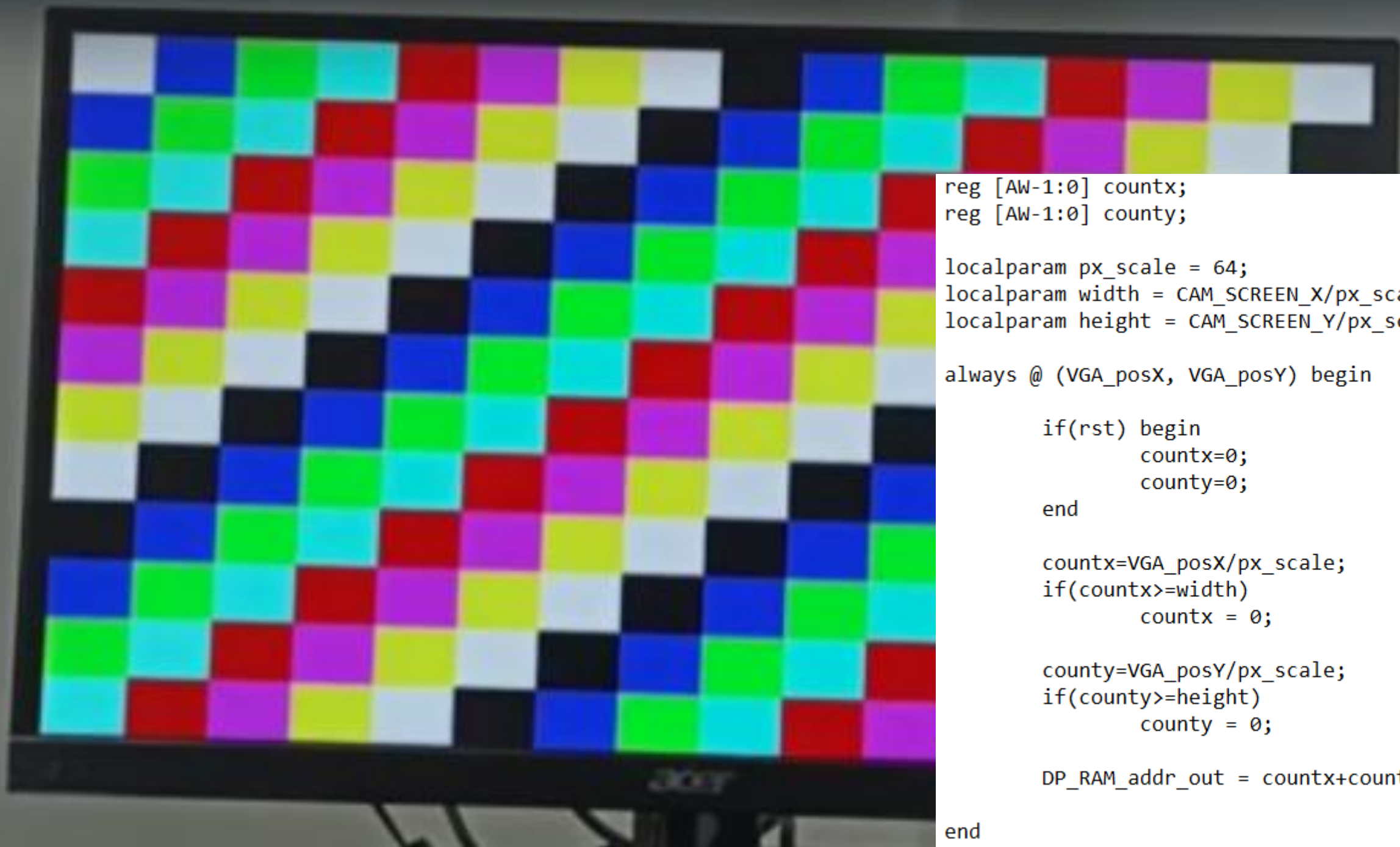
```
always @ (VGA_posX, VGA_posY) begin
```

```
    if(VGA_posX % px_scale == px_scale-1)
        countx = countx >= width ? 0 : countx + 1;
```

```
    if(VGA_posY % px_scale == px_scale-1)
        county = county >= height ? 0 : county + 1;
```

```
    DP_RAM_addr_out = countx + county*width;
```

```
end
```

```
reg [AW-1:0] countx;  
reg [AW-1:0] county;  
  
localparam px_scale = 64;  
localparam width = CAM_SCREEN_X/px_scale;  
localparam height = CAM_SCREEN_Y/px_scale;  
  
always @ (VGA_posX, VGA_posY) begin  
  
    if(rst) begin  
        countx=0;  
        county=0;  
    end  
  
    countx=VGA_posX/px_scale;  
    if(countx>=width)  
        countx = 0;  
  
    county=VGA_posY/px_scale;  
    if(county>=height)  
        county = 0;  
  
    DP_RAM_addr_out = countx+county*width;  
  
end
```

```

module FSM_game #(
    parameter AW = 8, // Cantidad de bits de la direccion
    parameter DW = 3 // cantidad de Bits de los datos
)(
    input clk,
    input rst,
    input clr, //Limpia pizarra
    input in1, //Botón right
    input in2, //Botón left
    input [DW-1:0] switch,

    output [AW-1:0] mem_px_addr,
    output [AW-1:0] mem_px_data,
    output px_wr
);

reg [20:0] count;
reg [AW:0] addr;
reg [AW:0] data;
reg write=0;
reg blocker;

assign mem_px_addr = addr;
assign mem_px_data = data;
assign px_wr = write;

wire gameclk;

```

```

divisor_de_frecuencia #(75000000,7) GameClk(
    .clk(clk),
    .clk_out(gameclk)
);

always @ (posedge gameclk) begin
    if(in1 && ~blocker) begin
        blocker=1;
        addr = count;
        count = count>=192 ? 0 : count+1;
        case(switch)
            0: data = 3'b000;
            1: data = 3'b001;
            2: data = 3'b010;
            3: data = 3'b011;
            4: data = 3'b100;
            5: data = 3'b101;
            6: data = 3'b110;
            7: data = 3'b111;
        endcase
        write=1;
    end else if(in2 && ~blocker) begin
        blocker=1;
        count = count>=192 ? 0 : count+1;
    end else if(clr) begin
        addr = count;
        count = count>=192 ? 0 : count+1;
        data = 3'b111;
        write=1;
    end else
        write=0;

    if(~in1 && ~in2)
        blocker=0;

    if(rst) begin
        count<=0;
        write<=0;
    end
end

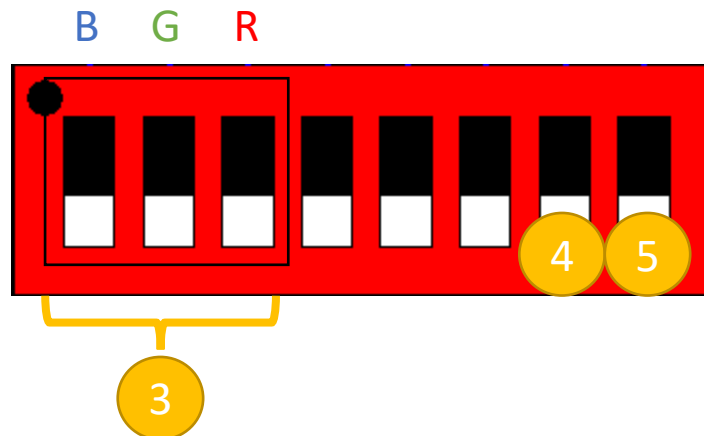
```

**F
S
M

G
A
M
E**



1. **Cursor (in2)**: Avanza al pixel siguiente sin pintarlo.
2. **Acción de Pintar (in1)**: Pinta el pixel donde se encuentra con el color dado en "3. Color" y avanza al siguiente.
3. **Color (switch)**: Selección del color a pintar en "2. Acción de pintar".
4. **Borrador (clr)**: Pinta de blanco los pixeles siguientes de manera automática.
5. **Reset (rst)**: Ubica el cursor en el inicio, parte superior derecha.





VIDEO DE PRUEBA

<https://youtu.be/MevZhtsOevw>