Proyecto Paint - FPGA

Kevin Santiago Aldana Muñoz - 1014979769
Emmanuel Bonilla Mitrotti - 1109543118
Ariel Giovanni Cardenas Santisteban - 7494038

Diciembre 2025

Resumen

Sistema de pintado implementado en una FPGA Tang Primer 25K, utilizando un mouse PS/2 conectado a un Arduino que envía datos por UART hacia la FPGA, la cual controla un panel LED RGB de 64x64 píxeles.

Índice

1 Introducción

El proyecto implementa una arquitectura pipeline donde la data del mouse atraviesa tres etapas principales (Adquisición, Procesamiento y Visualización) antes de afectar la pantalla LED.

1.1 Adquisición de Datos (Arduino)

1. **Lectura PS/2:** El firmware del Arduino lee de forma síncrona el mouse PS/2, capturando el estado de los botones y los cambios relativos de posición (ΔX y ΔY).
2. **Encapsulación UART:** El Arduino ensambla estos datos en un paquete serial de **3 bytes** (`[Botones] [Delta X] [Delta Y]`). Este paquete es transmitido continuamente a 9600 baudios a través del pin TX hacia la FPGA.

1.2 Procesamiento Lógico (FPGA)

La FPGA (Módulo `paint.v`) maneja dos submódulos críticos que operan en serie:

- **Receptor de Paquetes UART (`mouse_uart_receiver`):** Este módulo utiliza una FSM para sincronizar la llegada de los 3 bytes del paquete. Solo cuando el tercer byte es recibido, se activa la señal `data_valid`, liberando el paquete completo de movimiento a la lógica de pintado.
- **Lógica de Pintado (`PS2_TO_SCREEN`):** Este es el núcleo del sistema, implementado como una compleja FSM.
  - **Movimiento:** La FSM recibe ΔX y ΔY y los suma a las coordenadas absolutas actuales del cursor (X_{abs}, Y_{abs}). Realiza verificaciones constantes para asegurar que $0 \leq X_{abs} \leq 63$ y $0 \leq Y_{abs} \leq 63$.
  - **Pintado:** Si se detecta un **Clic Izquierdo**, la FSM calcula la dirección exacta de la memoria y activa la señal de escritura (`wr`) en la memoria para sobrescribir el píxel con el color de pintado.
  - **Cursor:** El módulo también gestiona la visualización temporal del cursor, alternando entre el color de fondo y un color de cursor para indicar la posición actual.

1.3 Visualización (FPGA - Controlador LED)

Este proceso opera de manera **independiente y concurrente** a la lógica de pintado:

- **Refresco Constante:** El **Controlador LED** (`panel_controller.v`) implementa la lógica de barrido (scan) de la matriz. Recorre cíclicamente las 32 (o 64) filas del panel.
- **Lectura de Memoria:** Por cada ciclo de reloj y para cada fila, el controlador lee el dato de color de la RAM de video y lo desplaza (shift) hacia los *drivers* del panel LED.

- **Sincronización:** Utiliza las señales `LATCH` (para transferir los datos desplazados a los *buffers* de salida) y `OE` (para el control de brillo y evitar el efecto *ghosting*) para mantener la imagen estable y visible, incluso mientras la lógica de pintado está actualizando píxeles individuales.

2 Conexiones Físicas

El camino de las conexiones físicas se presenta a continuación:

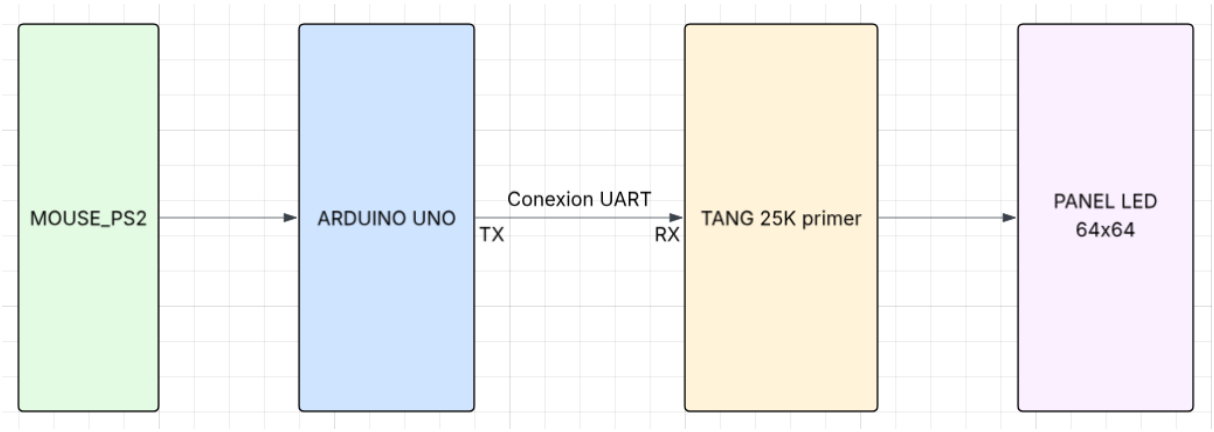


Figura 1: Diagrama de conexiones físicas del sistema

2.1 Conexión PS2 Mouse - Arduino UNO

Esta sección establece la interfaz de comunicación bidireccional entre el mouse PS/2 (utilizando el protocolo PS/2) y el microcontrolador Arduino. El Arduino leerá el movimiento y el estado de los botones del mouse. Para realizar la conexión del Mouse PS2 al Arduino UNO se utilizó el proyecto rucek/arduino-ps2-mouse.

PS2 Pin	Arduino Pin
DATA	5
CLK	6
VCC	5V
GND	GND

Cuadro 1: Conexión PS2 Mouse a Arduino UNO

2.2 Conexión UART: Arduino UNO (TX) a FPGA (RX)

Esta sección define el canal de comunicación serial unidireccional por UART, utilizado para enviar los datos procesados del mouse (movimiento y botones) desde el Arduino (Transmisor) hacia el módulo UART Receptor implementado en la FPGA.

Arduino UNO TX Pin	FPGA RX Pin
7	B2
GND	GND

Cuadro 2: Conexión UART Arduino a FPGA

2.3 Conexión: FPGA a Panel LED

Esta es la interfaz de hardware donde la FPGA actúa como el controlador de video, generando las señales de tiempo y datos para mostrar la imagen de 64x64 píxeles y el cursor del Paint.

FPGA Pin	**Panel LED Pin**	**Descripción**
G10	R0	Fila 0 - Rojo
G11	G0	Fila 0 - Verde
D10	B0	Fila 0 - Azul
B10	R1	Fila 1 - Rojo
B11	G1	Fila 1 - Verde
C10	B1	Fila 1 - Azul
A10	A	Dirección Fila A (Fila Select)
A11	B	Dirección Fila B (Fila Select)
E10	C	Dirección Fila C (Fila Select)
E11	D	Dirección Fila D (Fila Select)
C11	E	Dirección Fila E (Fila Select)
L11	CLK	Señal de Reloj (Shift Clock)
K11	LATCH	Señal Latch (Strobe)
K5	OE	Output Enable (Control de Brillo)
GND	GND	Tierra
GND	N	(GND o Pin sin uso)

Cuadro 3: Conexión FPGA a Panel LED

3 Estructura del Proyecto

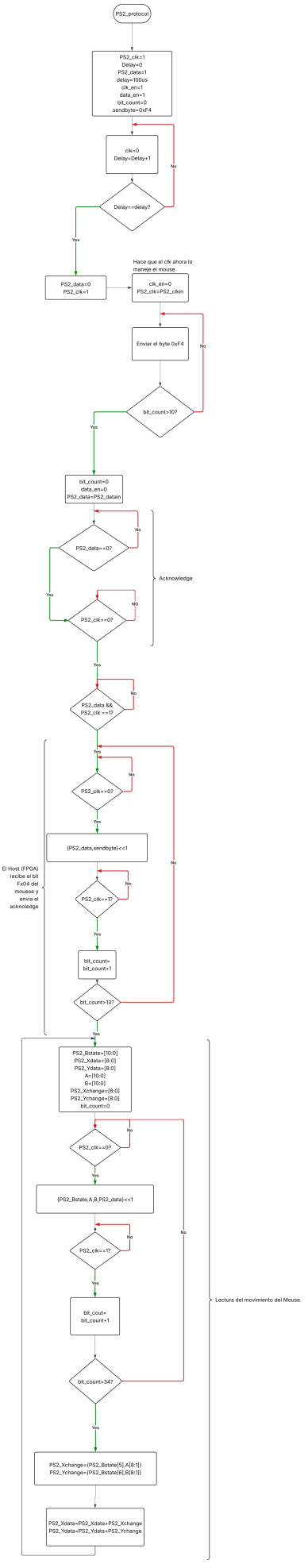
Esta sección detalla el funcionamiento interno de los módulos lógicos y protocolos utilizados en el proyecto.

```
paint_top
 mouse_rx       # Recibe 3 bytes del mouse por UART
    uart       # Módulo UART base
 ctrl_paint     # FSM de pintado (cursor + pintura)
 memory         # Framebuffer dual-port
 ctrl_panel     # FSM del panel LED
     count      # Contadores (fila, columna, delay, index)
     lsr_led    # Registro de desplazamiento para delay
     comp_4k    # Comparador de igualdad
     mux        # Multiplexor de bits RGB
```

3.1 Protocolo PS/2 (Mouse)

El protocolo PS/2 utiliza dos líneas (Clock y Data) para la transmisión serial síncrona de datos desde el dispositivo (Mouse) hacia el host (Arduino). El host lee los datos en el flanco de bajada del reloj.


Funcionamiento del protocolo PS2
Idle
Clock
Idle
Data
0 1 2 3 4 5 6 7 P
ACK
Keyboard
Host


PS2_protocol
PS2_clk=1
Delay=0
PS2_data=1
delay=100us
clk_en=1
data_en=1
bit_count=0
sendbyte=0xF4
clk=0
Delay=Delay+1
No
Delay==delay?
Yes
Hace que el clk ahora lo maneje el mouse
PS2_data=0
PS2_clk=1
clk_en=0
PS2_clk=PS2_clkin
Enviar el byte 0xF4
No
bit_count>10?
Yes
bit_count=0
data_en=0
PS2_data=PS2_datain
PS2_data==0?
No
Yes
Acknowledge
PS2_clk==0?
NO
Yes
PS2_data && PS2_clk ==1?
No
Yes
PS2_clk==0?
No
Yes
{PS2_data,sendbyte}<<1
El Host (FPGA) recibe el bit FxO4 del mouase y envia el acknoledge
PS2_clk==1?
No
Yes
bit_count= bit_count+1
bit_count>13?
No
Yes
PS2_Bstate=[10:0]
PS2_Xdata=[8:0]
PS2_Ydata=[8:0]
A=[10:0]
B=[10:0]
PS2_Xchange=[8:0]
PS2_Ychange=[8:0]
bit_count=0
PS2_clk==0?
No
Yes
{PS2_Bstate,A,B,PS2_data}<<1
PS2_clk==1?
No
Yes
bit_cout= bit_count+1
bit_count>34?
No
Yes
Lectura del movimiento del Mouse.
PS2_Xchange={PS2_Bstate[5],A[8:1]}
PS2_Ychange={PS2_Bstate[6],B[8:1]}
PS2_Xdata=PS2_Xdata+PS2_Xchange
PS2_Ydata=PS2_Ydata+PS2_Ychange

3.2 Protocolo UART (Módulo Genérico)

El módulo UART en la FPGA se encarga de deserializar los datos entrantes. Utiliza un sobremuestreo (16 veces la tasa de baudios) para detectar el bit de inicio y muestrear los datos en el centro del periodo de cada bit, garantizando la integridad de la recepción.

- **Diagrama de Flujo:** Muestra la máquina de estados de recepción (Detección de Start Bit → Muestreo de Bits 0-7 → Stop Bit).
- **Camino de Datos:** Ilustra los registros de desplazamiento y contadores utilizados.

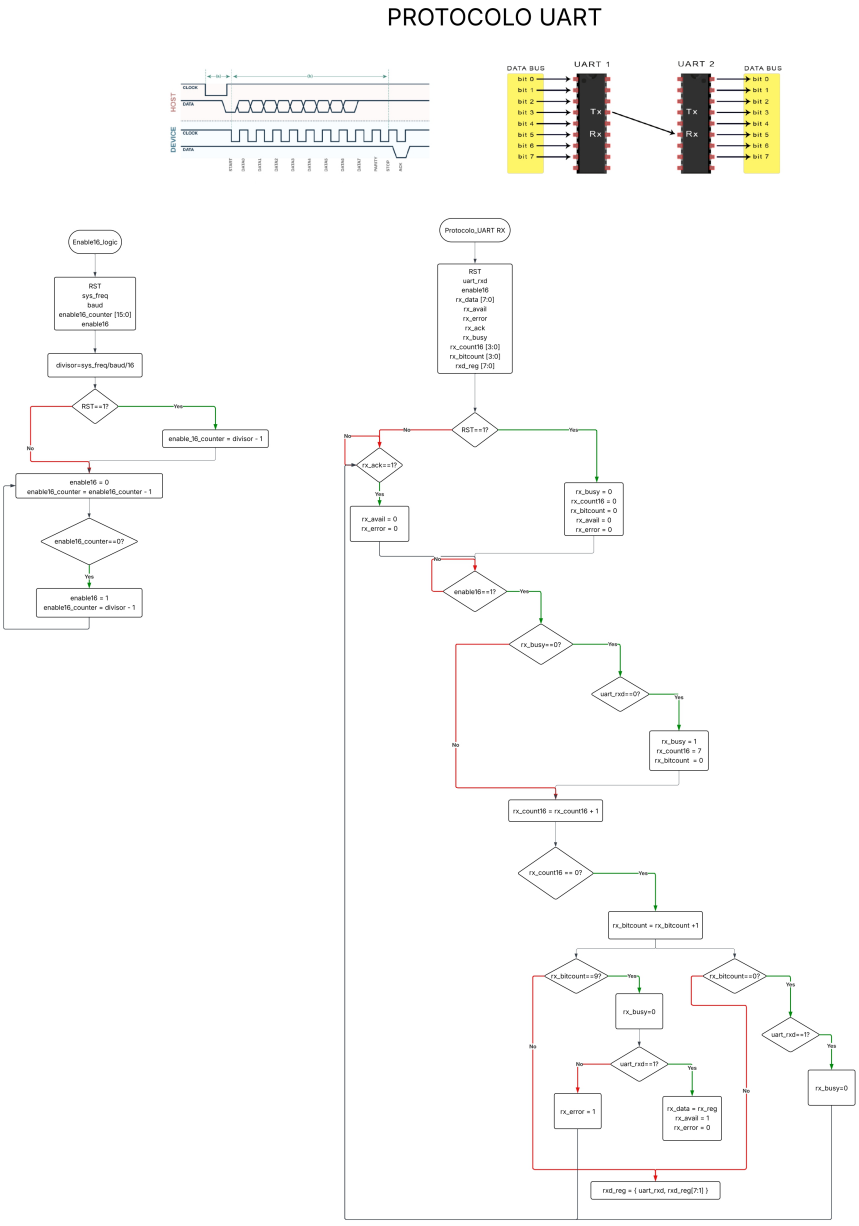


Figura 3: Diagrama de Flujo UART

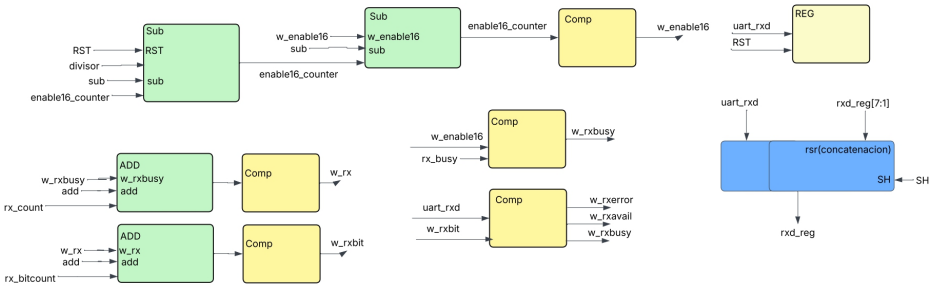


Figura 4: Camino de Datos UART

3.3 Interfaz UART: Arduino a FPGA

Este módulo superior gestiona la recepción de paquetes completos de 3 bytes provenientes del Arduino. La máquina de estados asegura que los datos se interpreten en el orden correcto: `[Byte 1: Botones]` → `[Byte 2: Movimiento X]` → `[Byte 3: Movimiento Y]`.

- **Diagrama de Flujo:** Describe la FSM que espera secuencialmente los 3 bytes y valida la integridad del paquete.
- **Camino de Datos:** Muestra el buffer de 3 posiciones y cómo se asignan a las señales de salida (`btn`, `delta_x`, `delta_y`).

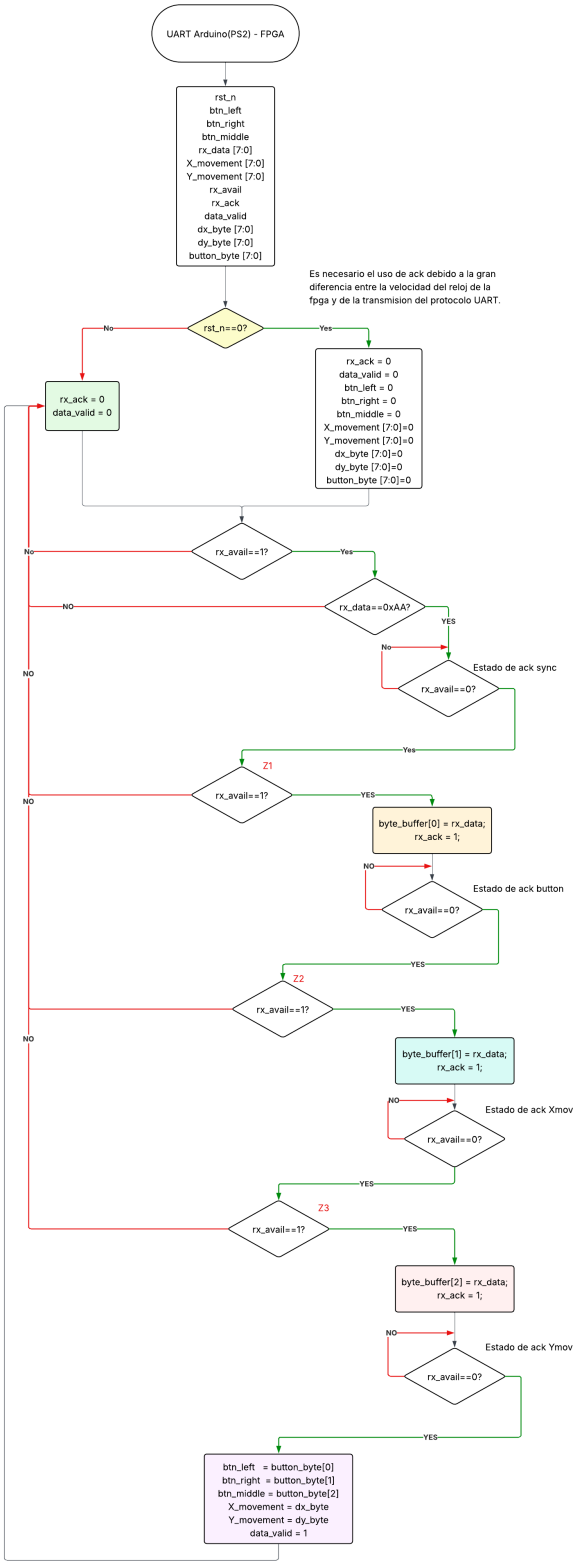


Figura 5: Flujo Arduino-FPGA

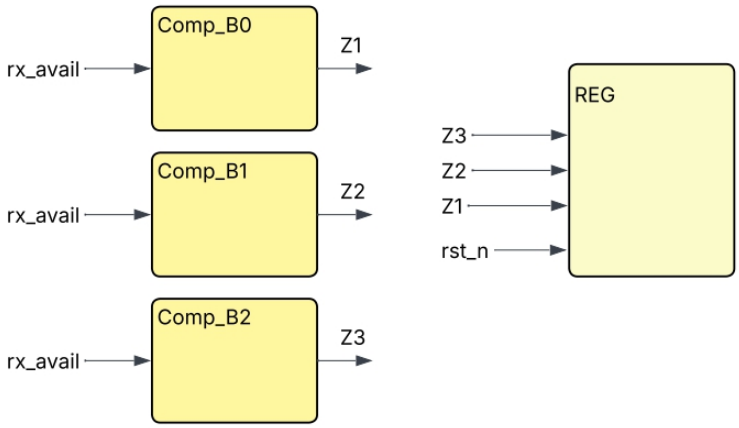


Figura 6: Datapath Arduino-FPGA

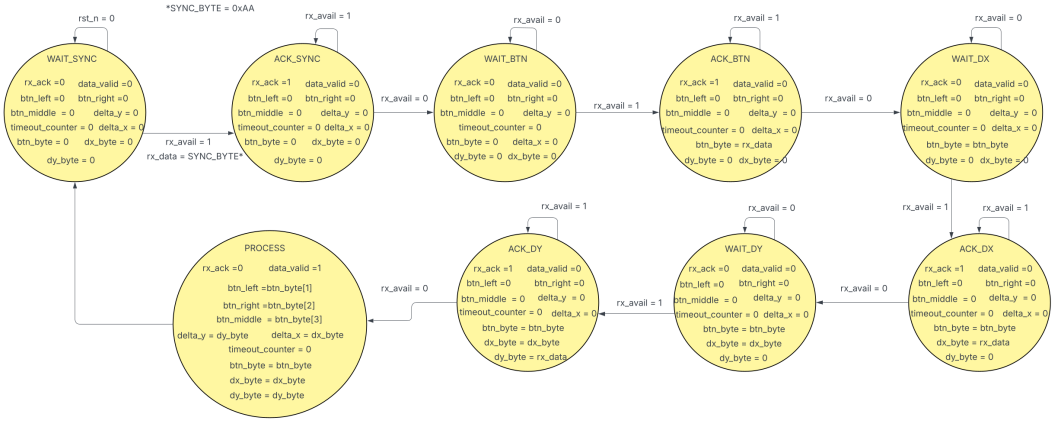


Figura 7: Diagrama de estados Arduino-FPGA

3.4 Controlador FPGA a Pantalla (Lógica de Pintado)

Este es el núcleo del proyecto (`PS2_TO_SCREEN`). Recibe las coordenadas del mouse, calcula la posición de memoria correspondiente en la matriz de 64x64, y actualiza el color del píxel si se detecta un clic (Pintar). También maneja la lógica de lectura de memoria para refrescar el panel LED continuamente.

- **Diagrama de Flujo:** Detalla el algoritmo para limitar las coordenadas (0-63), calcular la dirección de memoria (`Address = Y*64 + X`) y la lógica de escritura/-lectura.
- **Camino de Datos:** Muestra los comparadores (para límites de pantalla), sumadores (para movimiento relativo) y la interfaz con la memoria de video.

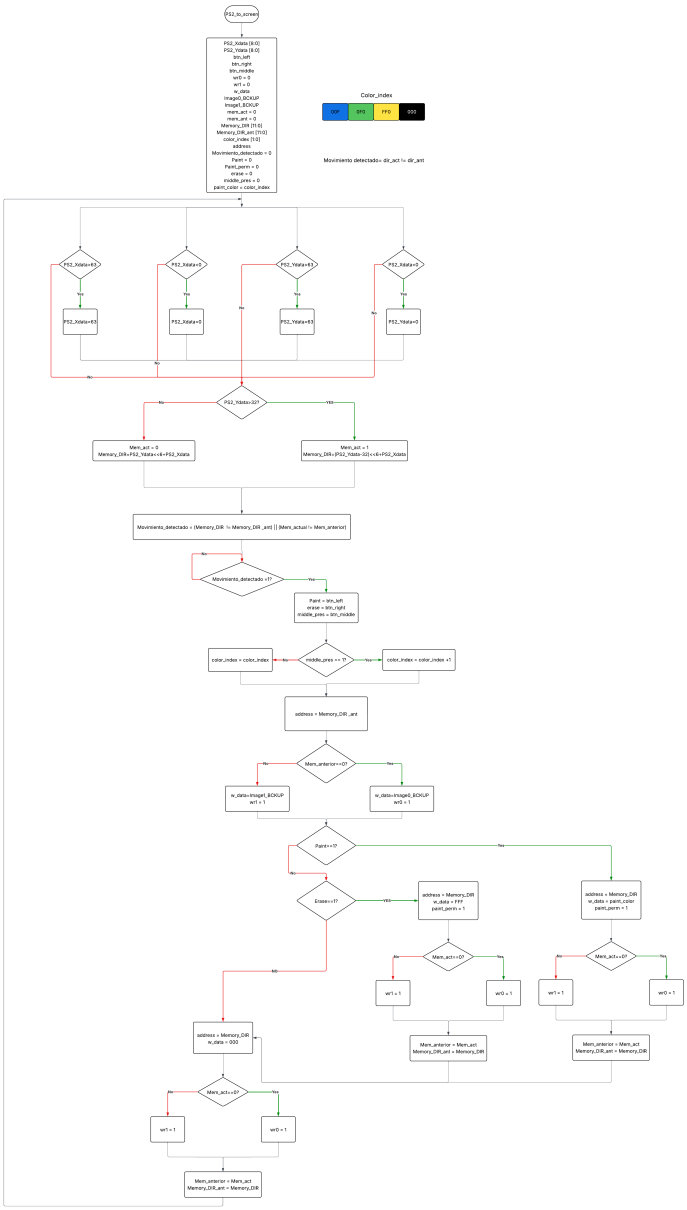


Figura 8: Flujo Lógica de Pantalla

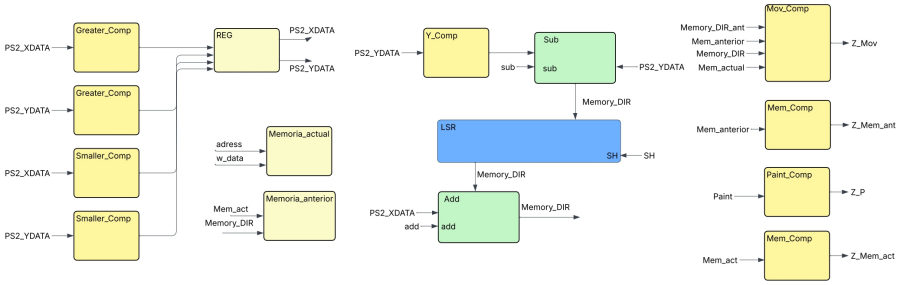


Figura 9: Datapath Lógica de Pantalla

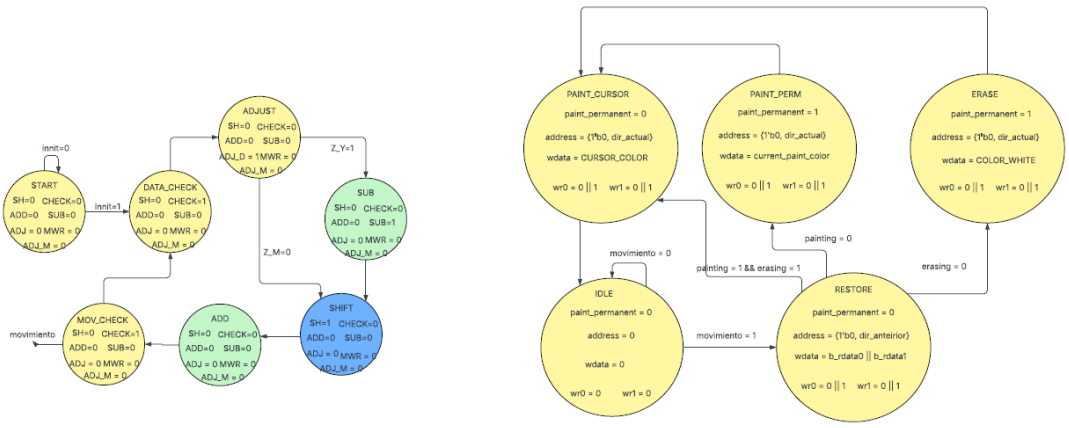


Figura 10: Diagrama de estados Lógica de Pantalla

3.5 Conexión FPGA a Pantalla (Lectura de Memoria)

Este módulo permite que la pantalla lea la memoria ya modificada por la lógica de pintado, y muestra la imagen en el panel LED.

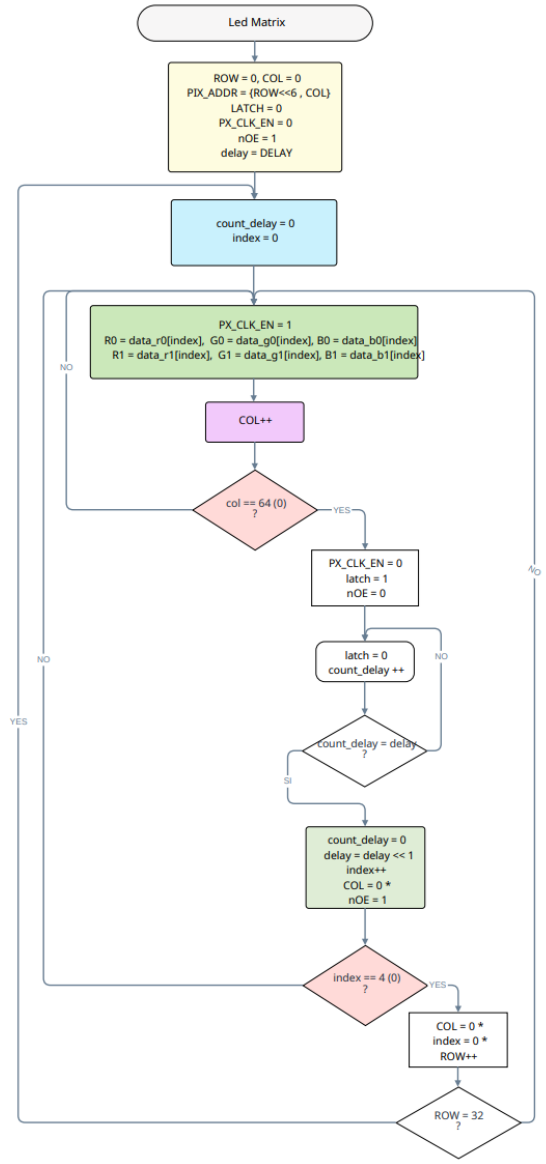


Figura 11: Diagrama de Flujo - Lectura de Memoria

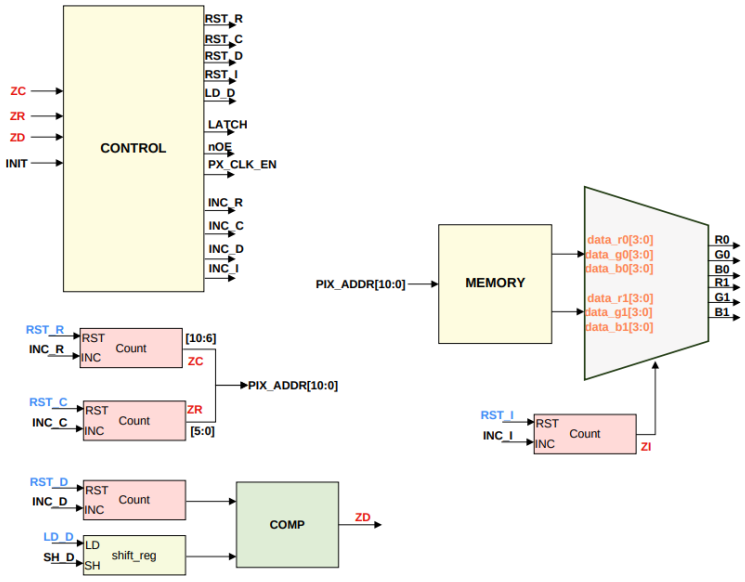


Figura 12: Datapath - Lectura de Memoria

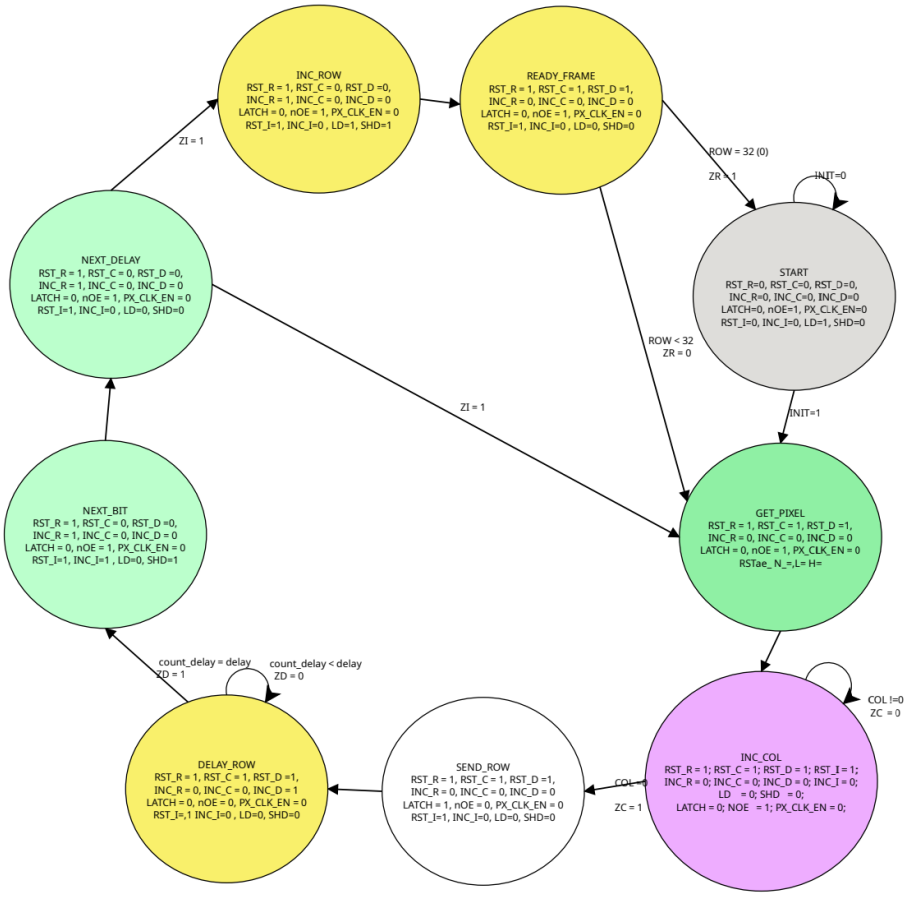


Figura 13: Diagrama de estados - Lectura de Memoria

4 Guía de Uso

Para desplegar y usar el proyecto, se deben seguir estos pasos:

4.1 Realizar Conexiones Físicas

1. Conectar el Mouse PS/2 al Arduino (usando los pines **CLK=6** y **DATA=5**).
2. Conectar el Arduino **TX** (**pin 7**) al UART **RX** de la FPGA (**pin B2**).
3. Conectar el **Panel LED** a los pines de la FPGA según se detalla en la sección **Conexión: FPGA a Panel LED**.

4.2 Programar el Arduino (Firmware)

Cargar el sketch ubicado en `arduino/mouse_uart.ino` en el Arduino UNO. Este firmware se encarga de leer el mouse PS/2 y encapsular los datos de movimiento/clic en paquetes de 3 bytes que se envían por UART.

4.3 Programar la FPGA (Hardware Lógico)

1. Sintetizar el diseño Verilog con los archivos de la carpeta `main/` (el módulo de nivel superior es `paint.v`). Asegurarse de que los archivos de *constraints* estén configurados correctamente para la placa Tang Primer 25K.
2. Cargar el *bitstream* resultante en la FPGA.

4.4 Operación del Sistema

1. Al encender el sistema, el panel LED mostrará la imagen de fondo inicial cargada desde la memoria de la FPGA.
2. Mover el mouse para desplazar el cursor sobre la pantalla.
3. Presionar el **Clic Izquierdo** del mouse para pintar o cambiar el color del píxel actual.
4. El sistema mostrará los trazos en tiempo real gracias a la alta velocidad de refresco del controlador LED.

5 Funcionamiento Resumido

- El cursor se mueve siguiendo el mouse.
- Click izquierdo: pinta en el color seleccionado.
- Click derecho: borra (pinta en blanco).
- Click central: cambia el color de pintado.
- El panel muestra una imagen de fondo cargada desde archivos `.hex`.