FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Sistemas Digitales (86.41)

Trabajo Práctico Final

Lucas Scheinkerman - Padrón: 100047 - lscheinkerman@fi.uba.ar

Facultad de Ingeniería, UBA
17 de Agosto, 2023

Índice

1. Enunciado

En el presente Trabajo Práctico el alumno desarrollará una arquitectura de rotación de objetos 3D basada en el algoritmo CORDIC. El objetivo principal es desarrollar tanto la unidad aritmética de cálculo como así también el controlador de video asociado.

Para la realización completa del Trabajo Práctico se cargarán en memoria externa las coordenadas correspondientes a un objeto tridimensional predefinido mediante una interfaz serie UART. A partir de los valores de las componentes, se rotará el objeto alrededor de cada uno de los ejes de coordenadas según el valor que adquieran las entradas del sistema, y por último las componentes rotadas serán presentadas en un monitor VGA mediante la aplicación de una proyección plana. En la Figura 1 puede observarse un diagrama en bloques del sistema completo. Deberá determinarse la cantidad mínima de bits de ancho de palabra (bits de precisión) para alcanzar las especificaciones requeridas.

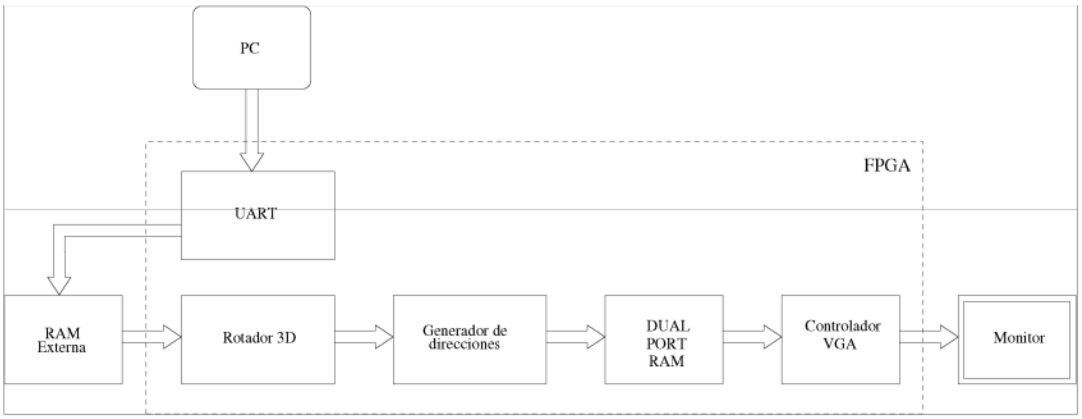

Figura 1.1: Esquema completo del sistema

2. Módulos implementados

2.1. Main

La función de este módulo es implementar los estados que atraviesa el sistema a través de una máquina de estados finita. Mediante este esquema se controla el flujo de información entre todos los módulos del sistema.

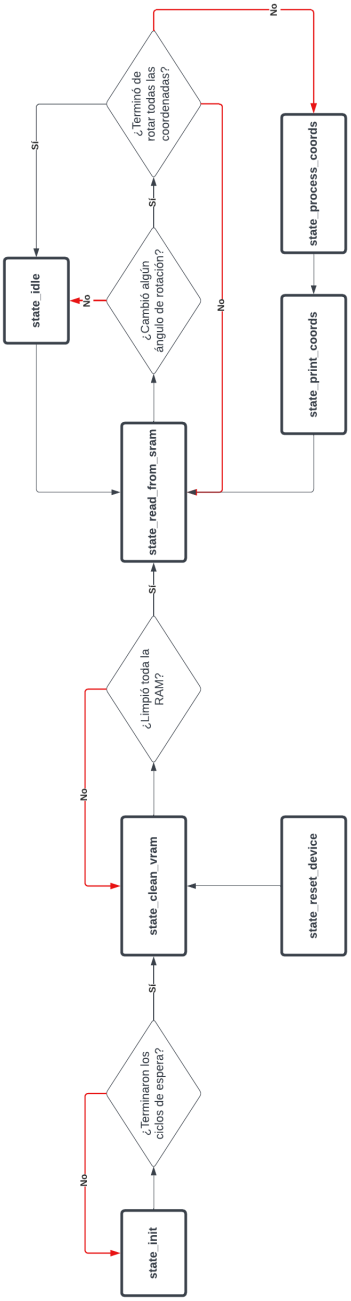

Figura 2.1: Esquema de la máquina de estados implementada.

```
entity main is
    generic (
        -- RAM constants
        constant RAM_DATA_WIDTH            : integer := 8;
        constant RAM_ADDRESS_WIDTH  : integer := 15; --32 kBytes de RAM
        constant BYTES_TO_RECEIVE   : natural := 32700;
        constant CYCLES_TO_WAIT            : integer := 100000;
        -- CORDIC Constants
        constant COORDS_WIDTH: integer := 8;
        constant ANGLE_WIDTH: integer := 10;
        constant CORDIC_STAGES: integer := 8;
        constant CORDIC_WIDTH: integer := 12;
        constant CORDIC_OFFSET: integer := 4;
        constant ANGLE_STEP_INITIAL: natural := 1;
        constant CYCLES_TO_WAIT_TO_CORDIC_TO_FINISH: natural := 10;
        -- VRAM constants
        constant VRAM_ADDR_BITS: natural         := 16; -- 8 KBytes
        constant VRAM_DATA_BITS_WIDTH: natural := 1);
    port (
        -- Clks and ctrl signals
        clk          : in std_logic;
        clk_wiz      : in std_logic;
        rst          : in std_logic;
        -- Buttons
        btn_x1       : in std_logic;
        btn_x2       : in std_logic;
        btn_y1       : in std_logic;
        btn_y2       : in std_logic;
        btn_z1       : in std_logic;
        -- Feedback signal to clk_wizard. Used to stabilize clock
        wiz_rst      : out std_logic;
        -- Zedboard-specific probe signals
        JA1          : out std_logic;
        JA2          : out std_logic;
        JA3          : out std_logic;
        JA4          : out std_logic;
        -- VGA signals
        hs_main, vs_main : out std_logic;
        red_out_main     : out std_logic_vector(2 downto 0);
        grn_out_main     : out std_logic_vector(2 downto 0);
        blu_out_main     : out std_logic_vector(1 downto 0));
end entity;
```

2.2. CORDIC

2.2.1. Rotator

Este módulo es el punto de entrada a la rotación de coordenadas. Instancia cada uno de los tres módulos rotores de forma sincronizada.

```
entity rotator is
    generic (
        COORDS_WIDTH         : integer := 10;
        ANGLES_INTEGER_WIDTH : integer := 10;
        STAGES               : integer := 16);
    port (
        clk                        :   in std_logic;
        X_in, Y_in, Z_in           :   in signed(COORDS_WIDTH-1 downto 0);
        angle_X, angle_Y, angle_Z  :   in signed(ANGLES_INTEGER_WIDTH-1 downto 0);
        X_out, Y_out, Z_out        :   out signed(COORDS_WIDTH-1 downto 0));
end entity rotator;
```

2.2.2. CORDIC

Este submódulo inicializa la ROM utilizadad para el algoritmo e instancia la cantidad de etapas necesarias para el algoritmo.

```
entity cordic is
    generic (
        COORDS_WIDTH            : integer := 10;
        ANGLES_INTEGER_WIDTH    : integer := 8;
        STAGES                  : integer := 16);
    port (
        X_in, Y_in              : in signed(COORDS_WIDTH-1 downto 0);
        angle                   : in signed(ANGLES_INTEGER_WIDTH-1 downto 0);
        X_out, Y_out            : out signed(COORDS_WIDTH-1 downto 0));
end entity cordic;
architecture behavioral of cordic is
```

2.2.3. CODIC stage

Este submódulo computa el resultado de las ecuaciones básicas que componen al algoritmo CORDIC.

```
entity cordic_stage is
    generic (
        COORDS_WIDTH : integer := 10;
        ANGLE_WIDTH  : integer := 22;
        STEP_WIDTH   : integer := 4);
    port (
        X_in, Y_in   : in signed(COORDS_WIDTH-1 downto 0);
        Z_in         : in signed(ANGLE_WIDTH-1 downto 0);
        atan         : in signed(ANGLE_WIDTH-1 downto 0);
        step         : in unsigned(STEP_WIDTH-1 downto 0);
```

```
        X_out, Y_out : out signed(COORDS_WIDTH-1 downto 0);
        Z_out        : out signed(ANGLE_WIDTH-1 downto 0));
end entity cordic_stage;
```

2.3. VGA control

Este módulo genera las señales de control de color y posición para el protocolo VGA. Para definir si el pixel que está dibujando en pantalla está en uno o cero, se fija si el pixel está en el rango legible y si está escrito en la VRAM como uno o cero. La información de la VRAM le llega a través de los puertos *red_i*, *grn_i* y *blu_i*.

```
entity vga_ctrl is
    port (
                mclk      : in std_logic;
                red_i     : in std_logic;
                grn_i     : in std_logic;
                blu_i     : in std_logic;
                hs        : out std_logic;
                vs        : out std_logic;
                red_o     : out std_logic_vector(2 downto 0);
                grn_o     : out std_logic_vector(2 downto 0);
                blu_o     : out std_logic_vector(1 downto 0);
                pixel_row : out std_logic_vector(9 downto 0);
                pixel_col : out std_logic_vector(9 downto 0));
end vga_ctrl;
```

2.4. Video driver

Este módulo se encarga de proveerle información al controlador VGA a partir de la VRAM. El controlador de VGA le informa a este módulo, a partir de los puertos *pixel_x* y *pixel_y*, donde quiere leer. Entonces el driver genera la dirección deseada a partir de estas coordenadas, y para el siguiente ciclo de clock le provee al controlador el valor del pixel actual.

```
entity video_driver is
        generic (
                -- A word's width, measured in bits
                VRAM_BITS_WIDTH : natural := 1;
                -- Number of bits for addresses
                VRAM_ADDR_BITS  : natural := 16);
    port (
                rst        : in std_logic;
                clk        : in std_logic;
                pixel_x    : in unsigned(9 downto 0);
                pixel_y    : in unsigned(9 downto 0);
                data_rd    : in std_logic_vector(VRAM_BITS_WIDTH-1 downto 0);
                red_en_o   : out std_logic;
                green_en_o : out std_logic;
                blue_en_o  : out std_logic;
```

```
                addr_rd     : out std_logic_vector(VRAM_ADDR_BITS-1 downto 0));
end video_driver;
```

2.5. VRAM

Este módulo materializa la RAM que se utilizará para guardar la información a enviar por el puerto VGA. Lo que tiene de diferente a la RAM utilizada para guardar las coordenadas, es que es una Dual Port RAM. Es decir, permite lectura y escritura a la vez. Por lo tanto, se puede escribirla mientras el controlador de VGA la lee.

```
entity vram is
        generic (
                VRAM_BITS_WIDTH : natural := 1;
                VRAM_ADDR_BITS  : natural := 16);
        port (
                rst     : in std_logic;
                clk     : in std_logic;
                data_wr : in std_logic_vector(VRAM_BITS_WIDTH-1 downto 0);
                addr_wr : in std_logic_vector(VRAM_ADDR_BITS-1 downto 0);
                ena_wr  : in std_logic;
                addr_rd : in std_logic_vector(VRAM_ADDR_BITS-1 downto 0);
                data_rd : out std_logic_vector(VRAM_BITS_WIDTH-1 downto 0));
end vram;
```

2.6. SRAM

Este módulo materializa la RAM que se utilizará para guardar las coordenadas de la figura a rotar.

Consta de un espacio de 32 kB, para poder almacenar todas las coordenadas entregadas por la cátedra.

```
component sram_internal is
    port(
        clka  : in std_logic;
        wea   : in std_logic_vector(0 downto 0);
        addra : in std_logic_vector(RAM_ADDRESS_WIDTH-1 downto 0);
        dina  : in std_logic_vector(RAM_DATA_WIDTH-1 downto 0);
        douta : out std_logic_vector(RAM_DATA_WIDTH-1 downto 0)
    );
end component;
```

3. Diseño de bloques

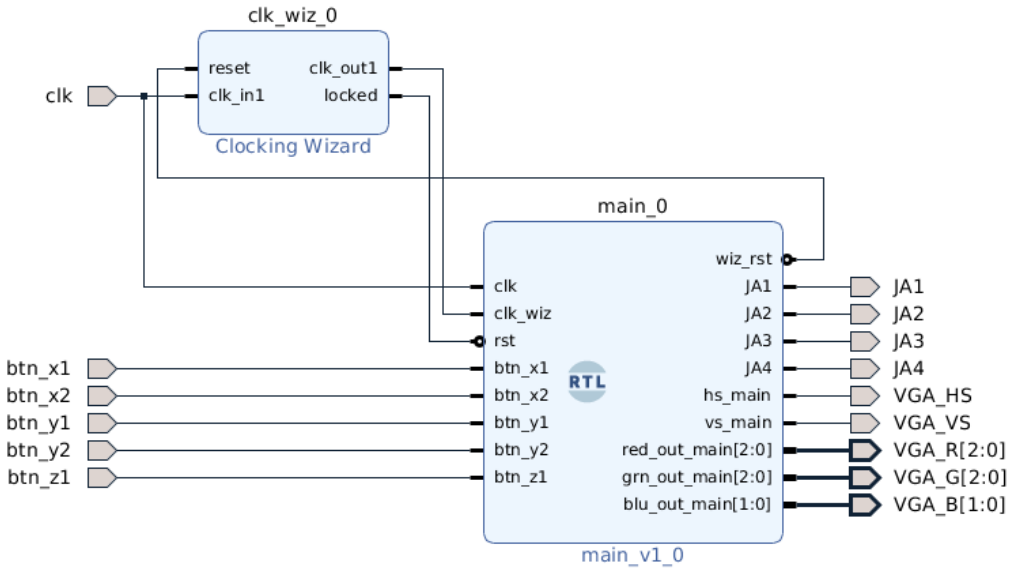

Figura 3.1: Diseño de bloques realizado en Vivado.

4. Métricas de utilización

Resource	Utilization	Available	Utilization %
LUT	1616	53200	3.04
FF	427	106400	0.40
BRAM	10	140	7.14
DSP	6	220	2.73
IO	24	200	12.00
MMCM	1	4	25.00

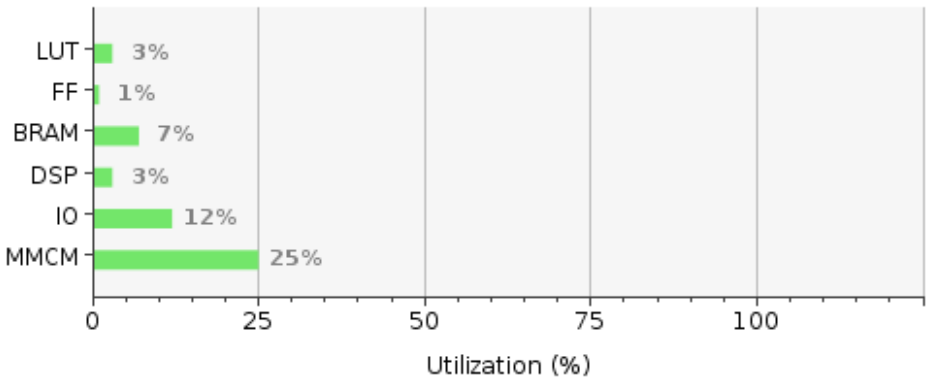

Figura 4.1: Métricas de utilización arrojadas por Vivado

5. Métricas de tiempo

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	12.431 ns	Worst Hold Slack (WHS):	0.061 ns	Worst Pulse Width Slack (WPWS):	3.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	814	Total Number of Endpoints:	814	Total Number of Endpoints:	349

Figura 5.1: Métricas de utilización arrojadas por Vivado

6. Conclusiones

Se logró implementar el sistema pedido en el enunciado, pero sin incluír la parte de comunicación UART con una RAM externa. En su lugar, se inicializó una RAM interna con las coordenadas iniciales, y a partir de éstas se calcularon el resto de los cómputos.

En el proceso también se logró familiarizar con el entorno de Vivado con mayor profunidad. Además, se ganó experiencia en programar FPGAs reales, fuera de la simulación, lo cual proveyó desafíos no encontrados durante la cursada.