VGA Display System with UART Terminal

Complete 640x480@60Hz Display Controller with Dual-Mode Operation

VHDL Development Project

July 31, 2025

Contents

1 Executive Summary

The VGA Display System is a comprehensive FPGA-based video controller designed for the Basys3 development board. The system implements a complete 640x480@60Hz VGA display interface with dual-mode operation: a test pattern generator and a UART-based text terminal. The design features professional-grade VGA timing generation, 25MHz pixel clock synthesis, real-time character display with scrolling, and switch-controlled configuration.

2 System Overview

2.1 Key Features

- Complete VGA 640x480@60Hz display controller
- Dual-mode operation: Pattern generator + UART text terminal
- 25MHz pixel clock generation from 100MHz system clock
- 80x30 character text display with scrolling and cursor
- 8 test patterns including color bars, gradients, and checkerboard
- Real-time UART integration at 115200 baud
- Switch-controlled mode selection and color configuration
- Professional VGA timing compliance
- 12-bit color depth (4 bits per RGB channel)

2.2 Target Platform

- **Primary:** Digilent Basys3 FPGA Development Board
- **FPGA:** Xilinx Artix-7 XC7A35T-1CPG236C
- **Tools:** Xilinx Vivado 2025.1 or later
- **System Clock:** 100MHz
- **Pixel Clock:** 25MHz (generated)
- **Display:** Standard VGA monitor (640x480@60Hz)

3 System Architecture

3.1 Top-Level Block Diagram

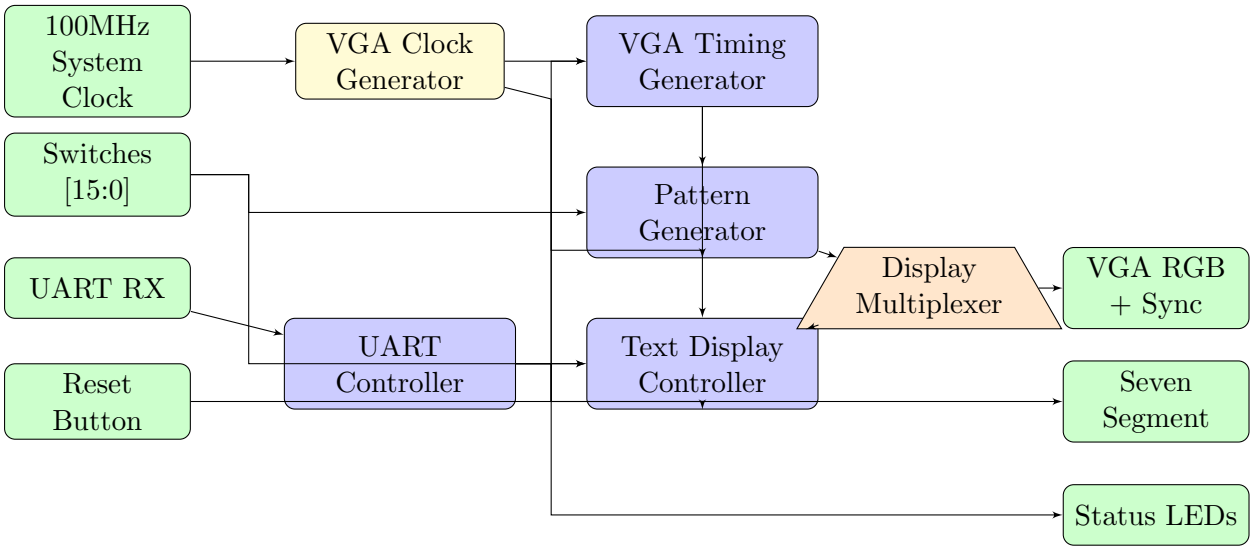

Figure 1: VGA Display System Architecture

3.2 Module Hierarchy

Module	File	Description
vga_top	vga_top.vhd	Top-level system integration
vga_clock	vga_clock.vhd	25MHz pixel clock generator
vga_timing	vga_timing.vhd	VGA sync and timing signals
vga_pattern	vga_pattern.vhd	Test pattern generator
vga_pattern_txt	vga_pattern_txt.vhd	Text display controller
uart_rx	uart_rx.vhd	UART receiver with FIFO
uart_tx	uart_tx.vhd	UART transmitter with FIFO
seven_segment_controller	seven_segment_controller.vhd	Display controller

Table 1: VGA System Module Hierarchy

4 VGA Timing Specifications

4.1 Standard VGA 640x480@60Hz Timing

Parameter	Horizontal	Vertical	Unit	Description
Active Area	640	480	pixels/lines	Visible display area
Front Porch	16	10	pixels/lines	After active, before sync
Sync Pulse	96	2	pixels/lines	Sync signal width
Back Porch	48	33	pixels/lines	After sync, before active
Total Period	800	525	pixels/lines	Complete line/frame
Sync Polarity	Negative	Negative	-	Active low sync signals

Table 2: VGA 640x480@60Hz Timing Parameters

4.2 Timing Diagram

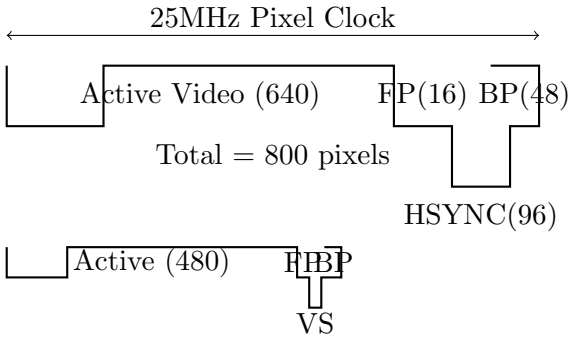

Figure 2: VGA Timing Waveforms (Horizontal and Vertical)

5 Clock Generation System

5.1 VGA Clock Generator (vga_clock.vhd)

The VGA clock generator creates the required 25MHz pixel clock from the 100MHz system clock using integer division.

5.1.1 Clock Division Calculation

$$\text{Divisor} = \frac{\text{System Clock}}{\text{Pixel Clock}} = \frac{100\text{MHz}}{25\text{MHz}} = 4 \tag{1}$$

$$\text{Half Divisor} = \frac{\text{Divisor}}{2} = 2 \tag{2}$$

$$\text{Quarter Divisor} = \frac{\text{Divisor}}{4} = 1 \tag{3}$$

5.1.2 Clock Generation Logic

```
clock_division_process : process(sys_clk)
begin
    if rising_edge(sys_clk) then
        if reset = '1' then
            clk_counter <= 0;
            pixel_clk_reg <= '0';
        else
            if clk_counter = DIVISOR-1 then
                clk_counter <= 0;
            else
                clk_counter <= clk_counter + 1;
            end if;

            -- Generate 25MHz pixel clock
            if clk_counter = HALF_DIVISOR-1 then
                pixel_clk_reg <= '1';
            elsif clk_counter = DIVISOR-1 then
                pixel_clk_reg <= '0';
            end if;
        end if;
    end if;
end process;
```

Listing 1: Clock Division Process

6 VGA Timing Generator

6.1 Timing Generator Architecture

The VGA timing generator implements dual counters for horizontal and vertical timing generation with proper sync signal polarities.

6.1.1 Counter Implementation

```
timing_counters : process(clk)
begin
    if rising_edge(clk) then
        if reset = '1' then
            h_count <= 0;
            v_count <= 0;
        else
            if h_count = H_TOTAL-1 then
                h_count <= 0;
                if v_count = V_TOTAL-1 then
                    v_count <= 0;
                else
                    v_count <= v_count + 1;
                end if;
            else
                h_count <= h_count + 1;
            end if;
        end if;
    end if;
end process;
```

Listing 2: VGA Timing Counters

6.2 Display Enable Generation

The display enable signal defines the active video region:

```
display_enable_generation : process(clk)
begin
    if rising_edge(clk) then
        if (h_count < H_ACTIVE) and (v_count < V_ACTIVE) then
            display_en <= '1';
        else
            display_en <= '0';
        end if;
    end if;
end process;
```

Listing 3: Display Enable Logic

7 Pattern Generator

7.1 Test Pattern Specifications

Pattern ID	Name	Description
0	Black Screen	All pixels off
1	Solid Red	Full red color field
2	Solid Green	Full green color field
3	Solid Blue	Full blue color field
4	Checkerboard	32x32 pixel black/white squares
5	Color Bars	8 vertical RGB color bars
6	Border Test	White border with black center
7	Gradient	Horizontal red, vertical green gradient

Table 3: Test Pattern Definitions

7.2 Color Bar Implementation

The color bar pattern implements the standard RGB truth table:

Bar	Red	Green	Blue
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Table 4: RGB Color Bar Truth Table

8 Text Display Controller

8.1 Text Display Specifications

Parameter	Value
Character Resolution	80 x 30 characters
Character Size	8 x 16 pixels
Font Type	Simple bitmap font
Character Buffer	2400 characters (80x30)
Cursor Type	Blinking block cursor
Cursor Blink Rate	1 Hz (25M cycles)
Scroll Direction	Vertical (upward)
Character Set	ASCII 0x20-0x7E

Table 5: Text Display Parameters

8.2 Character Buffer Management

8.2.1 Buffer Organization

The character buffer is organized as a linear array representing the 80x30 character grid:

```
type char_buffer_type is array (0 to BUFFER_SIZE-1) of
    std_logic_vector(7 downto 0);
signal char_buffer : char_buffer_type := (others => x"20");

-- Buffer addressing
buffer_addr := cursor_y * TEXT_COLS + cursor_x;
```

Listing 4: Character Buffer Type

8.2.2 Scrolling Implementation

```
-- Scroll screen up when at bottom line
if cursor_y = TEXT_ROWS-1 then
    -- Move all lines up by one
    for i in 0 to BUFFER_SIZE-TEXT_COLS-1 loop
        char_buffer(i) <= char_buffer(i + TEXT_COLS);
    end loop;
    -- Clear last line
    for i in BUFFER_SIZE-TEXT_COLS to BUFFER_SIZE-1 loop
        char_buffer(i) <= x"20";  -- Space character
    end loop;
end if;
```

Listing 5: Screen Scrolling Logic

8.3 Font ROM Implementation

The system includes a simple 8x16 bitmap font stored in FPGA block RAM:

```
function init_font_rom return font_rom_type is
    variable rom : font_rom_type := (others => (others => '0'));
begin
    -- Example: Character 'A' (0x41)
    rom(65 * CHAR_HEIGHT + 0)  := "00000000";
    rom(65 * CHAR_HEIGHT + 1)  := "00010000";
    rom(65 * CHAR_HEIGHT + 2)  := "00111000";
    rom(65 * CHAR_HEIGHT + 3)  := "01101100";
    rom(65 * CHAR_HEIGHT + 4)  := "11000110";
    rom(65 * CHAR_HEIGHT + 5)  := "11111110";
    -- ... additional font data
    return rom;
end function;
```

Listing 6: Font ROM Initialization