Video Graphics Array
VGA

Konstantinos Seraskeris

November 2024

1 Summary

This project involves the **implementation of a VGA controller** in Verilog, that includes: **the VRAM**, **the timing signals (HSYNC, VSYNC)**, and the **integration** of these components to form the complete VGA controller/driver. The VRAM is limited to a resolution of 128×96 pixels, using three 16K×1 BRAMs for the RGB colors. The **HSYNC** and **VSYNC** signals were implemented using **Finite State Machines (FSMs)** for horizontal and vertical scanning, respectively, with timing verification with testbenches. Finally, the VGA controller was completed by combining these components, successfully driving the display and rendering images.

2 Introduction

The objective of this project is to implement a VGA controller in hardware using Verilog. In this context, VRAM was designed and implemented along with the HSYNC and VSYNC synchronization signals and the driver for displaying data on the screen. The project's success was verified through simulations and experiments, ultimately achieving image display on a conventional **640×480 resolution screen**.

Specifically, in **Part A**, the VRAM was implemented using **three BRAMs**—one for each color (RGB)—with a resolution of **128×96**. This reduced resolution ensures that the memory remains within the FPGA's constraints. **Each pixel is repeated five times** both horizontally and vertically.

In **Part B**, the **HSYNC signal** was implemented using a **Moore-type Finite State Machine (FSM)**, which manages the **timing sequence (pulse, front/back porch, display period)**. A pixel counter controls the pixel transitions, displaying each pixel five times.

In **Part C**, the **VSYNC signal** manages the **transition between lines** and is implemented with a corresponding FSM, considering **longer** time intervals than HSYNC.

Finally, the **VGA controller** was completed by integrating **VRAM, HSYNC**

and VSYNC signals, and pixel counters, forming the final architecture for data display on the screen. The circuit's operation was successfully verified, displaying the final output on the screen.

3 Part A - VRAM Implementation

3.1 Implementation:

VRAM is a central component of the VGA controller, as it is continuously accessed by the driver to display its contents on the screen. A full-resolution 640×480 display would require:

$$640 * 480 * 3 \ (RGB \ bits) = 921600 \ bits = 115200 KBytes$$

which exceeds the accessible memory of the FPGA. To address this, the resolution is reduced by a factor of five in each dimension, resulting in 128×96 resolution, where each pixel is displayed five times before moving to the next. Additionally, separate memory blocks are used for each color, requiring:

$$28 * 96 = 12288 \ bits$$

Thus, to meet these requirements, three BRAMs, each of size 16K×1, are used to implement the 128×96 VRAM. Each pixel's color data is stored separately in the corresponding memory addresses.

3.2 Verification:

To verify the circuit's functionality, a testbench was developed to read pixel data from different addresses with various colors, ensuring correct functionality and proper color placement. It is crucial to retain the address of the black pixel for future use. The verification process is illustrated in Figure 1.

4 Part B - HSYNC and Horizontal Pixel Counter Implementation

4.1 Implementation:

The HSYNC timing signal controls the **horizontal synchronization** of the display. A **single line** scan consists of:

1. HSYNC pulse width
2. Display period
3. Front and back porches (before and after the display period, respectively)

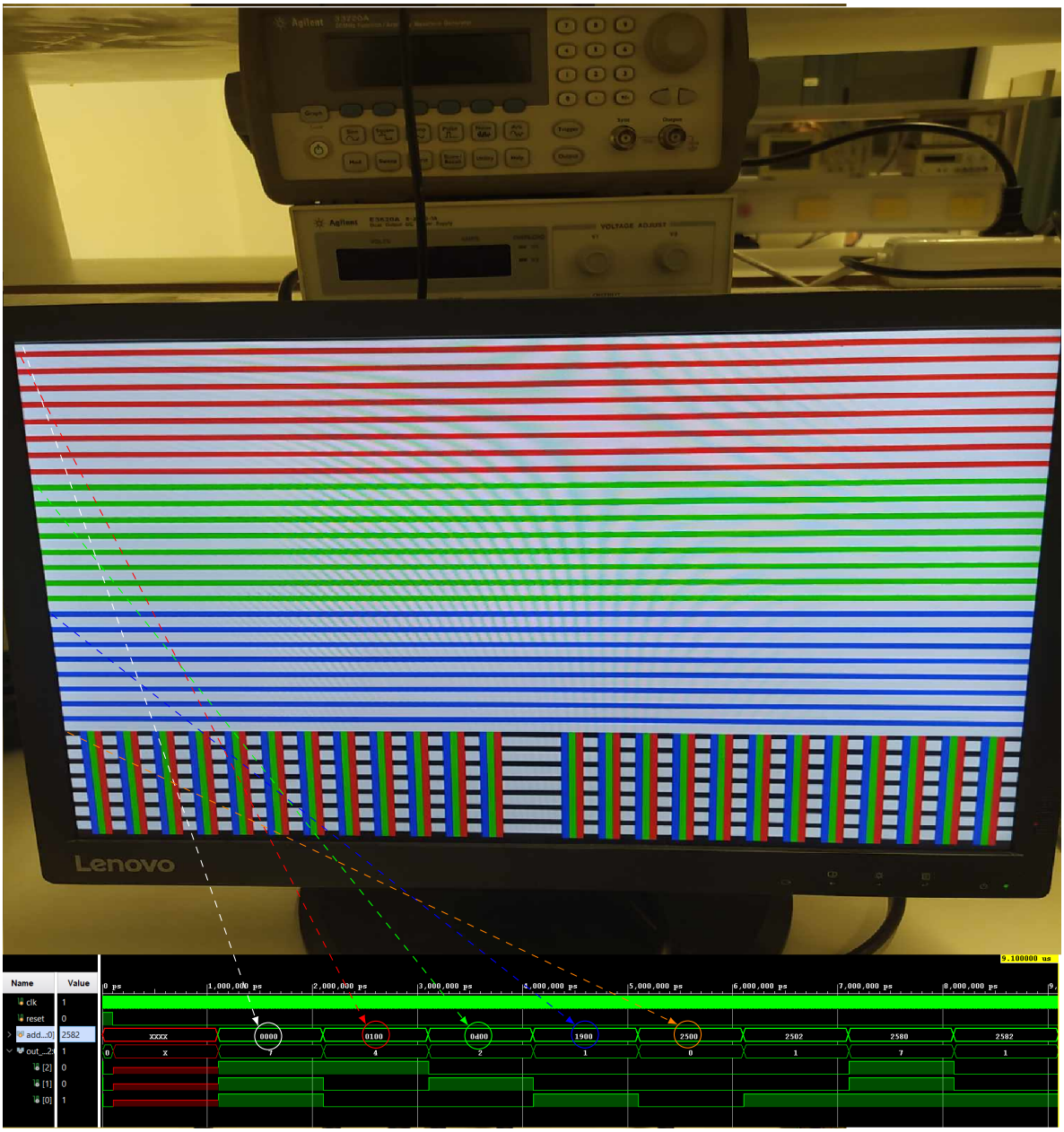


Figure 1: The addresses, colors, and pixel representation on the screen (Out[2]=Red, Out[1]=Green, Out[0]=Blue)

A Finite State Machine (FSM) with states corresponding to these intervals is used to implement the HSYNC signal. The cycle count for each interval is calculated as:

$$Cycles = \frac{t_{interval}}{T_{fpga}} = \frac{t_{interval}}{10 * 10^{-9}}$$

The total cycles for each interval are shown in Table 1.

Symbol	Parameter	Duration	Cycles
A	Scanline Time	32 μsec	3,200
B	HSYNC Pulse Width	3.84 μsec	384
C	Back Porch	1.92 μsec	192
D	Display Time	25.6 μsec	2,560
E	Front Porch	0.064 μsec	64

Table 1: HSYNC timing intervals, their durations, and in FPGA clock cycles

The display time of a single line lasts 25.6µs, which corresponds to 2,560 clock cycles of a 10ns clock. For a 640x480 resolution, the next pixel would be fetched from VRAM every:

$$\frac{2560\ cycles}{640\ pixels/line} = 4\ cycles\ per\ pixel$$

However, since the resolution and memory are reduced by a factor of 5, each pixel will be displayed 5 times. Meaning the next pixel will be fetched every:

$$\frac{2560\ cycles}{128\ pixels/line} = 20\ cycles\ per\ pixel$$

The key components and their functions:

- Counter: A common counter used in all FSM states. Each new state sets a target count, and once it is reached, the FSM transitions to the next state.
- Counter Signal: A counter used only during the image display state. It is activated by the **inverse logic** of the offDisplay_H signal. It counts the horizontal pixels of the line, has a fixed maximum count of 128, and increments when the main clock cycle counter resets (i.e., maximum_cycles → 0, 1, . . .).
- HSYNC FSM: A Moore-type Finite State Machine (FSM) consisting of states and outputs that define the HSYNC signal and its behavior. The outputs include:
  - The HSYNC signal itself.
  - The offDisplay_H signal, which indicates when the screen is not displaying data.

- The number of clock cycles that must pass before transitioning to the next state, as calculated earlier.

This FSM is **continuous** and never remains in one state indefinitely; instead, it cycles through the same states indefinitely.

The FSM with the states and their outputs is displayed in Figure 2 followed by their explanation below:

- B_PULSE:
  - Represents the HSYNC pulse width.
  - Disables the HSYNC signal and enables the offDisplay_H signal (since no data is displayed).
  - Waits 384 clock cycles before transitioning to the next state.
- C_BACKPORCH:
  - Represents the back porch interval.
  - Enables both the HSYNC and offDisplay_H signals (data still not displayed).
  - Waits 192 clock cycles before transitioning to the next state.
- D_DISPLAYTIME:
  - The state where the image is displayed on the screen.
  - HSYNC remains enabled, but offDisplay_H is disabled (since data is now visible).
  - The deactivation of offDisplay_H also enables the counter signal.
  - The system counts 128 horizontal pixels using the counter, and once it reaches the total pixel count and the clock cycles are completed, it transitions to the next state.
- E_FRONTPORCH:
  - Represents the front porch interval.
  - Enables both the HSYNC and offDisplay_H signals (no data displayed).
  - Waits 64 clock cycles before transitioning back to B_PULSE, restarting the cycle.

The dataflow of the circuit components is shown in Figure 3.

4.2 Verification:

To verify the circuit's functionality, a testbench was developed that allows the HSYNC signal to run for several cycles. The waveforms were then analyzed, timing intervals were measured, and results were compared to the calculated values. The verification results for selected values are shown in Figure 4 and Figure 5. The simulation was performed at the post-implementation timing level, where glitches and slight deviations from expected timing were observed. However, this is normal behavior and does not affect the final output.

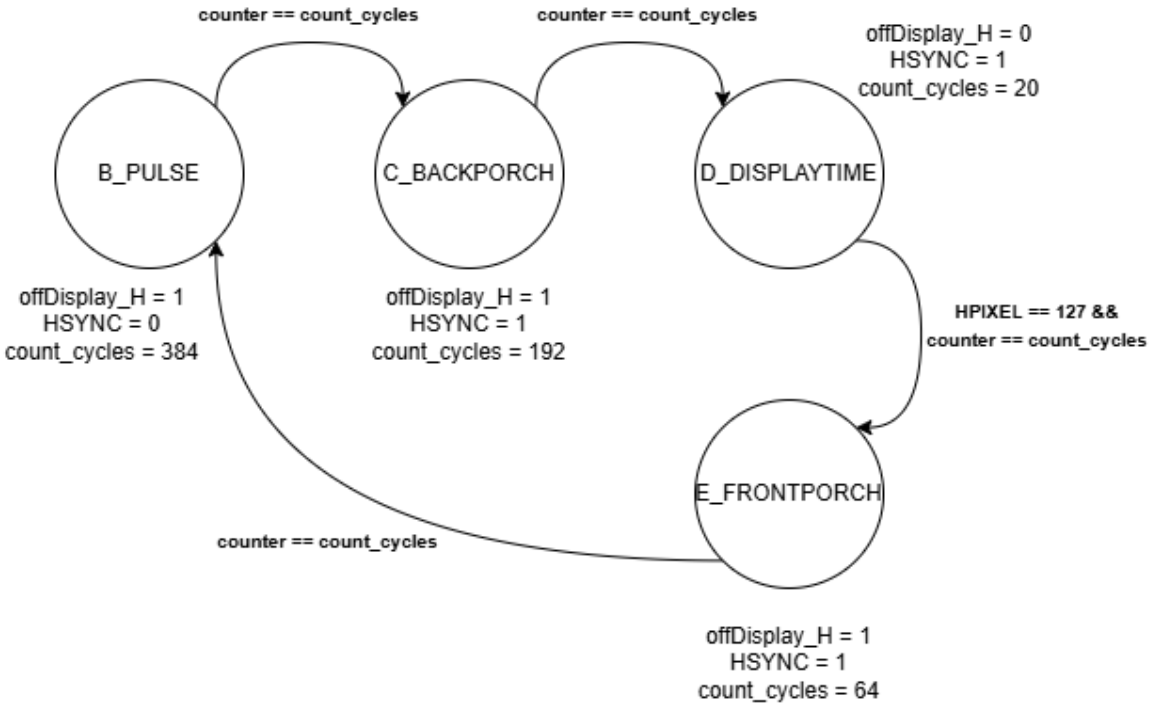


Figure 2: FSM diagram in Part B

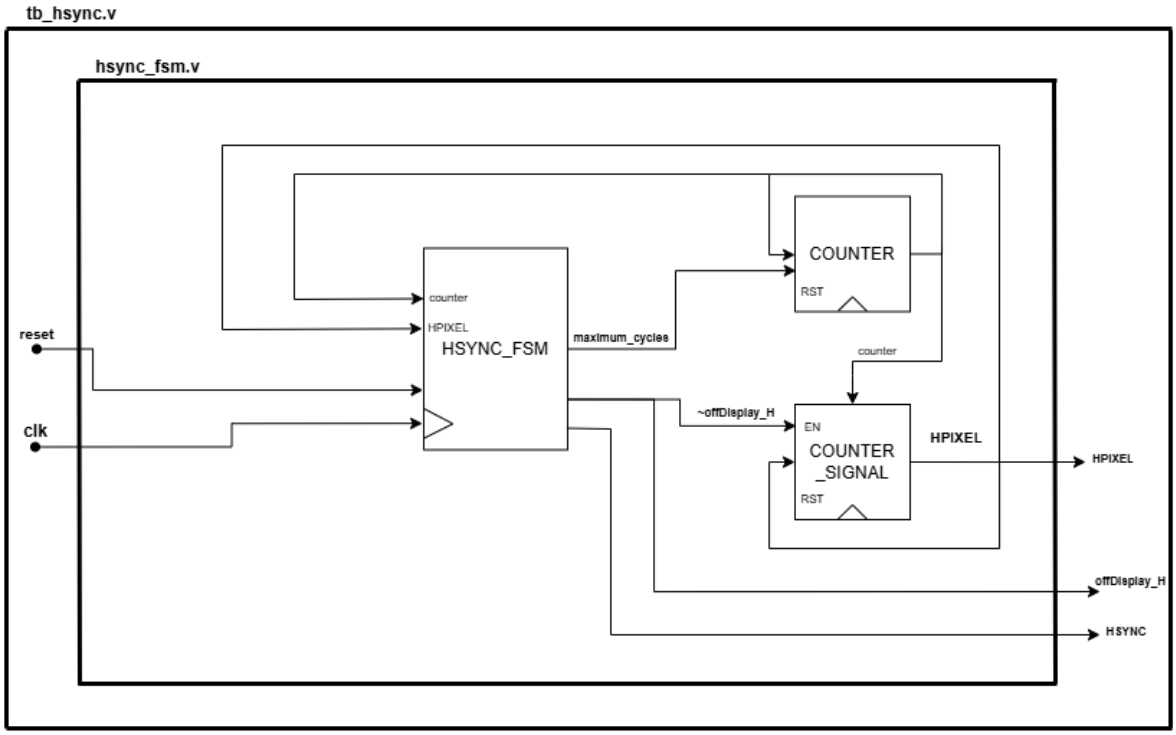


Figure 3: Circuit Dataflow in Part B

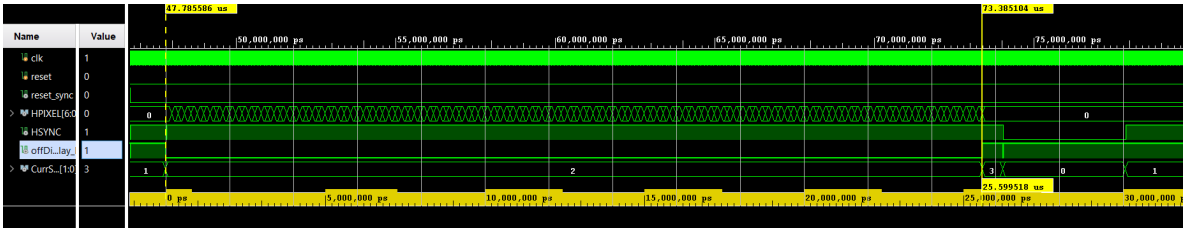


Figure 4: Signal states during the display time (D) and duration measurement

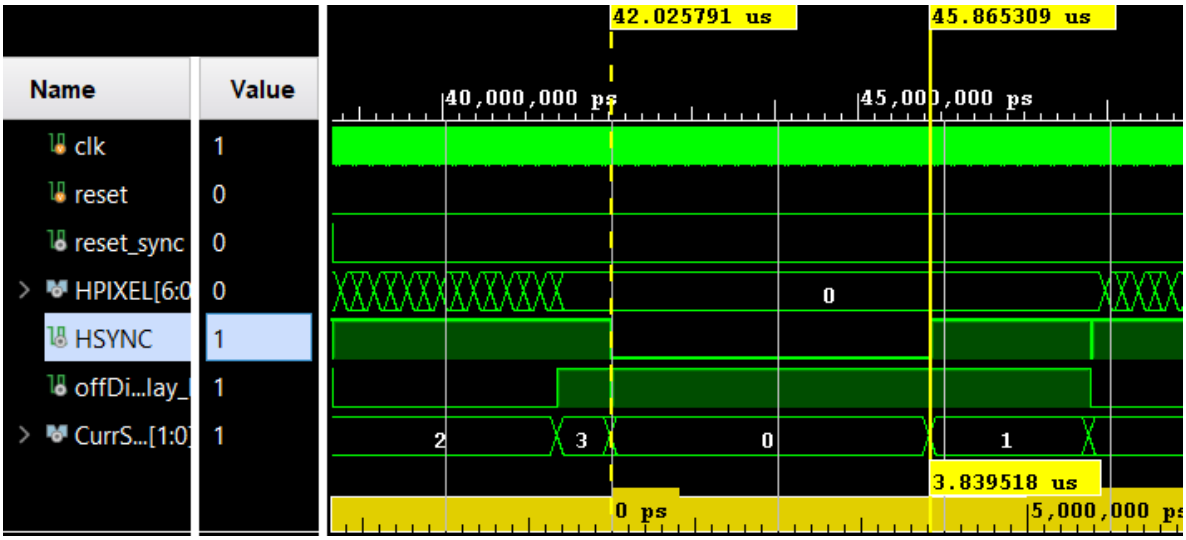


Figure 5: Signal states during the HSYNC pulse width (B) and duration measurement

5 Part C - VSYNC Implementation and Vertical Pixel Counter

5.1 Implementation:

The VSYNC timing signal controls the **vertical synchronization** of the screen. It is implemented **similarly to HSYNC** since it consists of the same timing components, but with **longer** time intervals, as it controls the entire screen, meaning **all horizontal pixel lines**. The corresponding timing intervals and clock cycles are shown in Table 2.

Symbol	Parameter	Duration	Cycles
O	Total Frame Time	16.67 msec	1,667,000
P	VSYNC Pulse Width	64 μsec	6,400
Q	Back Porch	928 μsec	92,800
R	Active Video Time	15.36 msec	1,536,000
S	Front Porch	320 μsec	32,000

Table 2: VSYNC timing intervals, durations, and in FPGA clock cycles

The active display time lasts 15.36ms, which corresponds to 1,536,000 clock cycles with the 10ns clock. For a 640x480 resolution, each horizontal pixel line would be fetched every:

$$\frac{1536000\ cycles}{480\ lines} = 3200\ cycles\ per\ line$$

However, since the resolution and memory are 5 times smaller, each line will be displayed 5 times, so the next line will be fetched every:

$$\frac{2560\ cycles}{96\ lines} = 16000\ cycles\ per\ line$$

This circuit is implemented exactly like HSYNC, with the only difference being the number of cycles required per state and the cycles needed before increasing the vertical pixel counter (VPIXEL), which are 16,000 cycles for VSYNC compared to 20 cycles for HPIXEL.

The FSM states with the states and their outputs are shown in Figure 6, while the dataflow of the circuit components is shown in Figure 7.

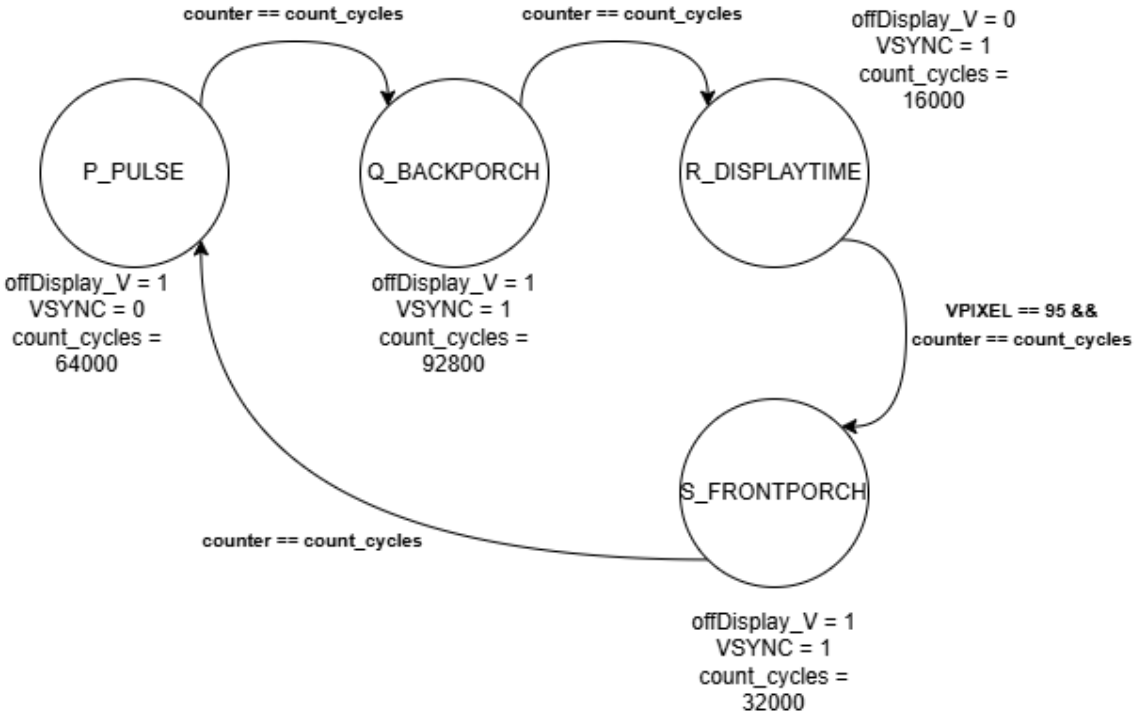


Figure 6: FSM diagram in Part C

5.2 Verification:

To verify the circuit's operation, a testbench was developed that allows the HSYNC signal to run for several cycles. The waveforms were then analyzed, timing intervals were measured, and the results were compared with the calculated values.

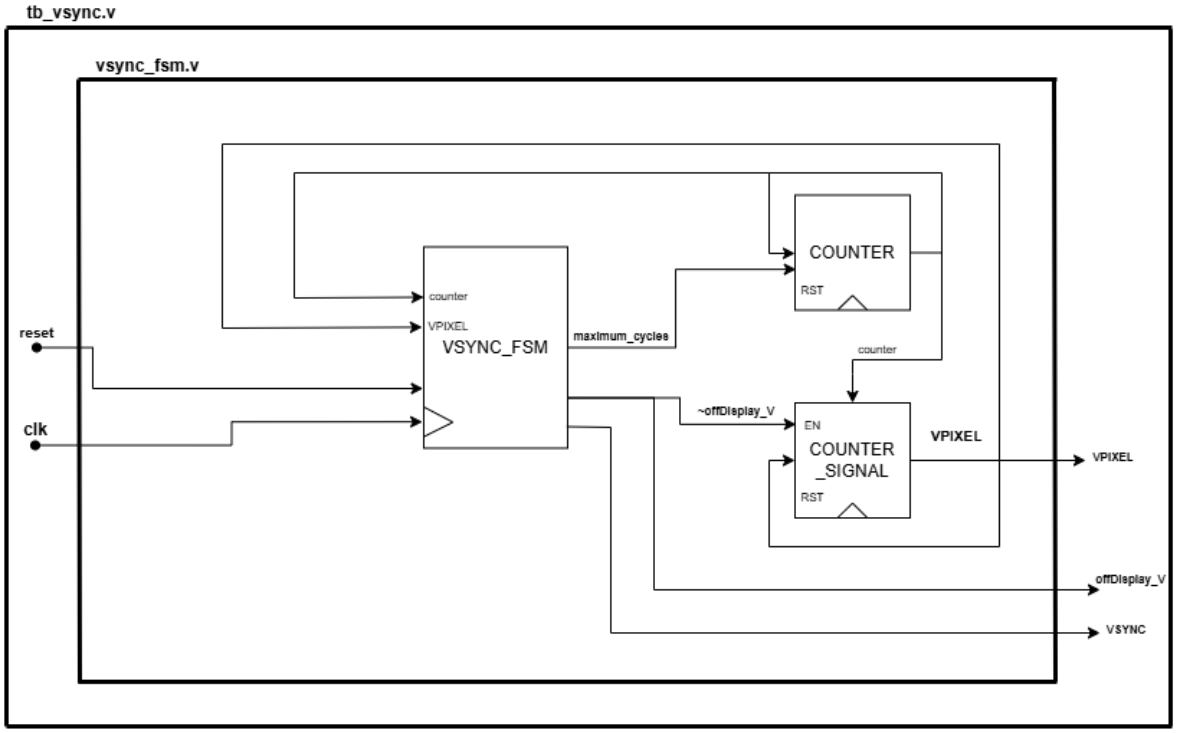


Figure 7: Circuit Dataflow in Part C

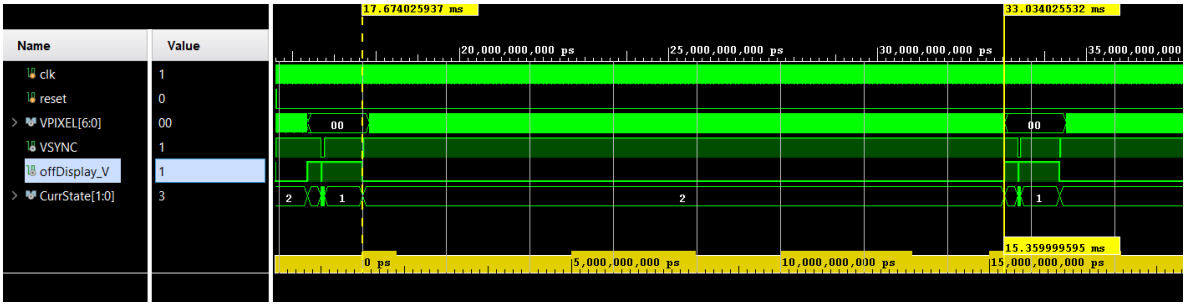


Figure 8: Signal states during the active display time (R) and duration measurement

Verification results for selected values are shown in Figure 8 and Figure 9.The simulation was performed at the post-implementation timing level, where glitches and slight deviations from expected timing were observed. However, this is normal behavior and does not affect the final output.

6 Completion of VGA Controller/Driver

Finally, the VRAM, along with the two timing signals (HSYNC and VSYNC), is combined to drive the display and render the image on the screen. The dataflow of the circuit components is shown in Figure 10.

The distinction between display and non-display intervals, as well as the desired color representation, is achieved as follows:

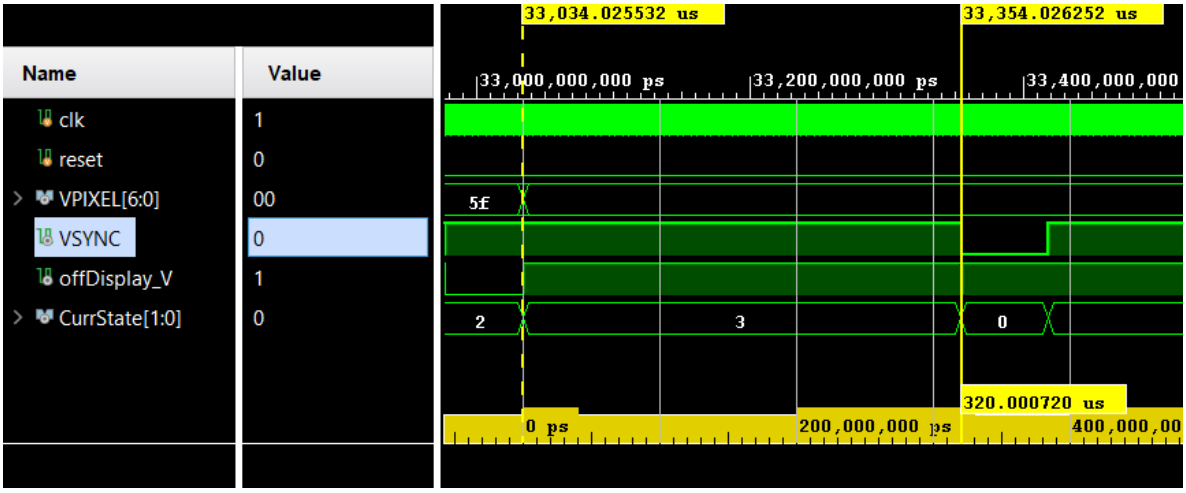


Figure 9: Signal states during the front porch interval (S) and duration measurement

```
1 //addr of VRAM based on display or not
2 //                                                  black pixel Part A
3 assign vga_addr = (offDisplay_V || offDisplay_H)? 14'b10010100000000:
4                   {VPIXEL, HPIXEL};
```

Here, the offDisplay_H and offDisplay_V signals are used to distinguish between display and non-display intervals.

- During non-display intervals, a black pixel from Part A (ii) is used.
- Otherwise, the address is determined by combining the vertical and horizontal pixel vectors.

Some example images of the final circuit running on the FPGA board and displayed on the screen are shown in Example 1, Example 2 and Example 3.

7 Conclusions

The implementation of the VGA controller in Verilog was a challenging yet rewarding experience. During the design process, the efficient management of FPGA resources, such as reducing resolution and using three BRAMs for VRAM, was crucial.

The implementation of the HSYNC and VSYNC timing circuits required precision in timing calculations and FSM organization.

Verification was performed using testbenches, and while glitches and timing variations appeared in simulations, they were deemed normal, as real-world signals and timing intervals are never perfectly precise. Ultimately, these minor variations did not affect the final functionality.

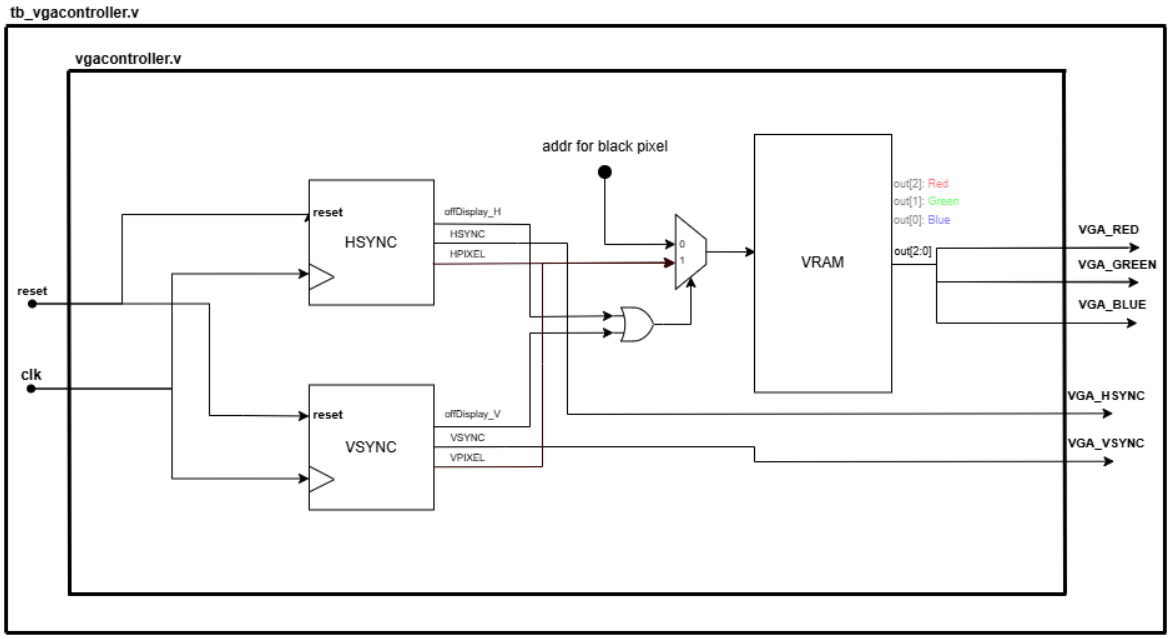
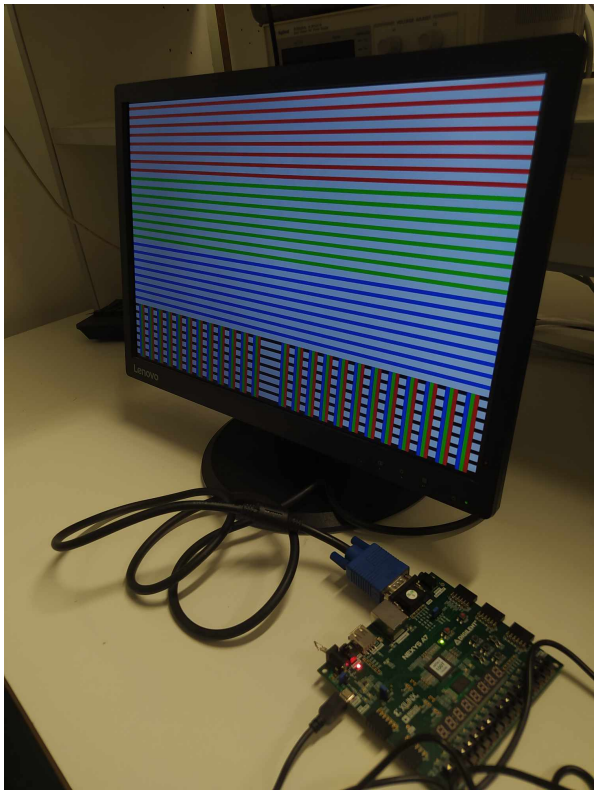
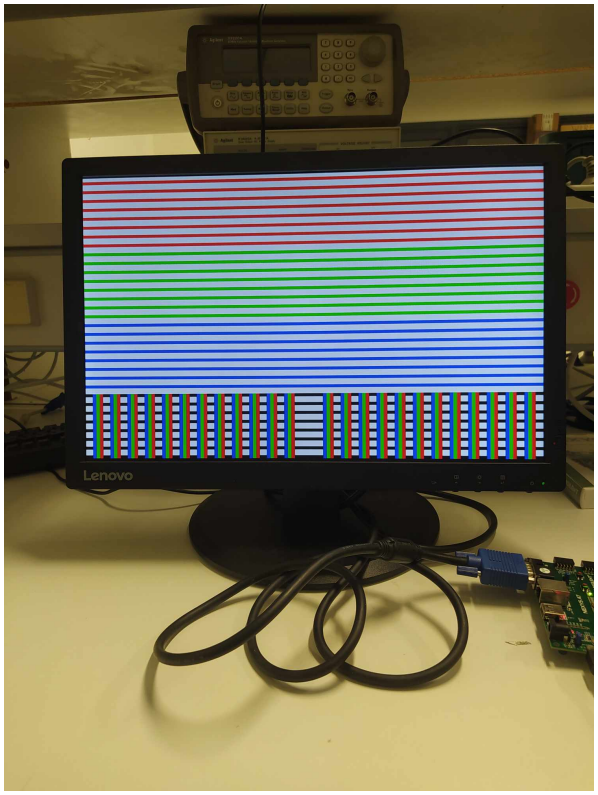


Figure 10: Circuit Dataflow of final top module



Example 1: FPGA board connection and image display on the screen

Example 2: FPGA board connection and image display on the screen



Example 3: FPGA board connection and image display on the screen