Implementing a Simple GPU on FPGA Card

İbrahim Hakkı Candan
ibrahimhakkicandan@icloud.com

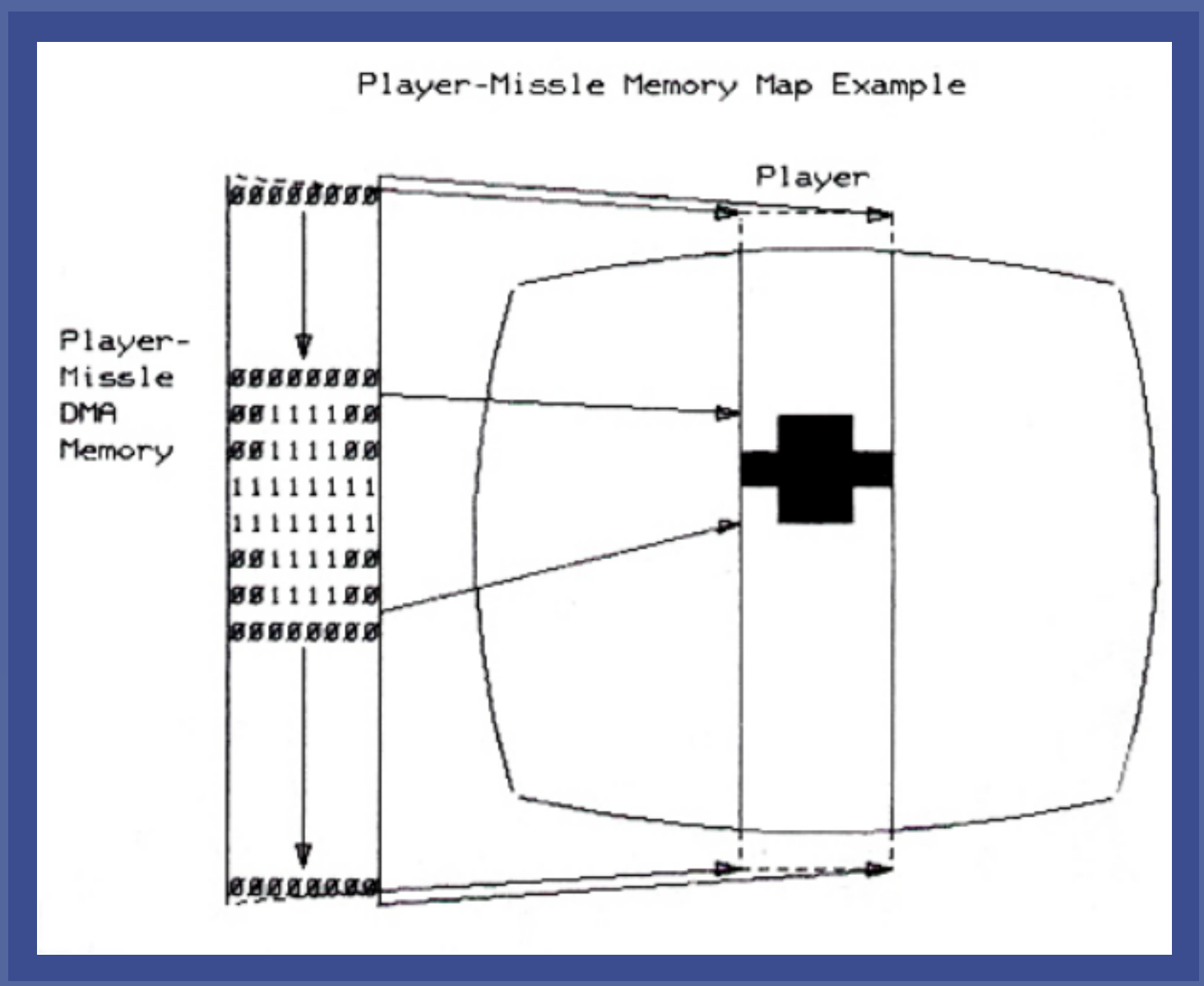
Ramazan Karkin
karkinramazan02@gmail.com

Rumeysa Ulusoy
rumysaulusoy@gmail.com

Advisor: Asst. Prof. Mehmet Kadir Baran

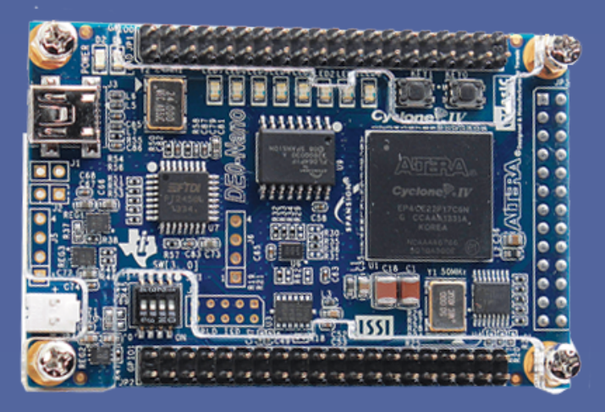
INTRODUCTION

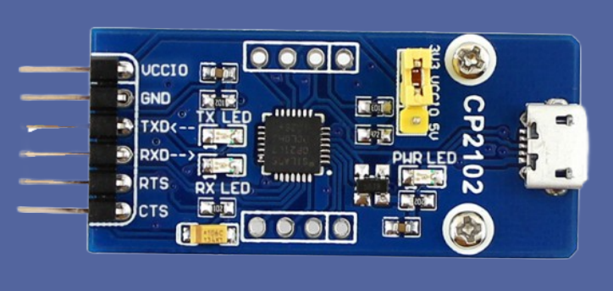
This project aims to develop a straightforward GPU using an FPGA board, with the purpose of generating VGA graphics. The primary task involves establishing data transfer from the FPGA to the SDRAM using the UART protocol. Subsequently, the VGA graphics data is transmitted from the SDRAM to both the CPU and VGA modules, enabling the display of the basic graphics on a monitor. It is important to note that while the CPU has both read and write access to the SDRAM, the VGA module is limited to read-only permissions.



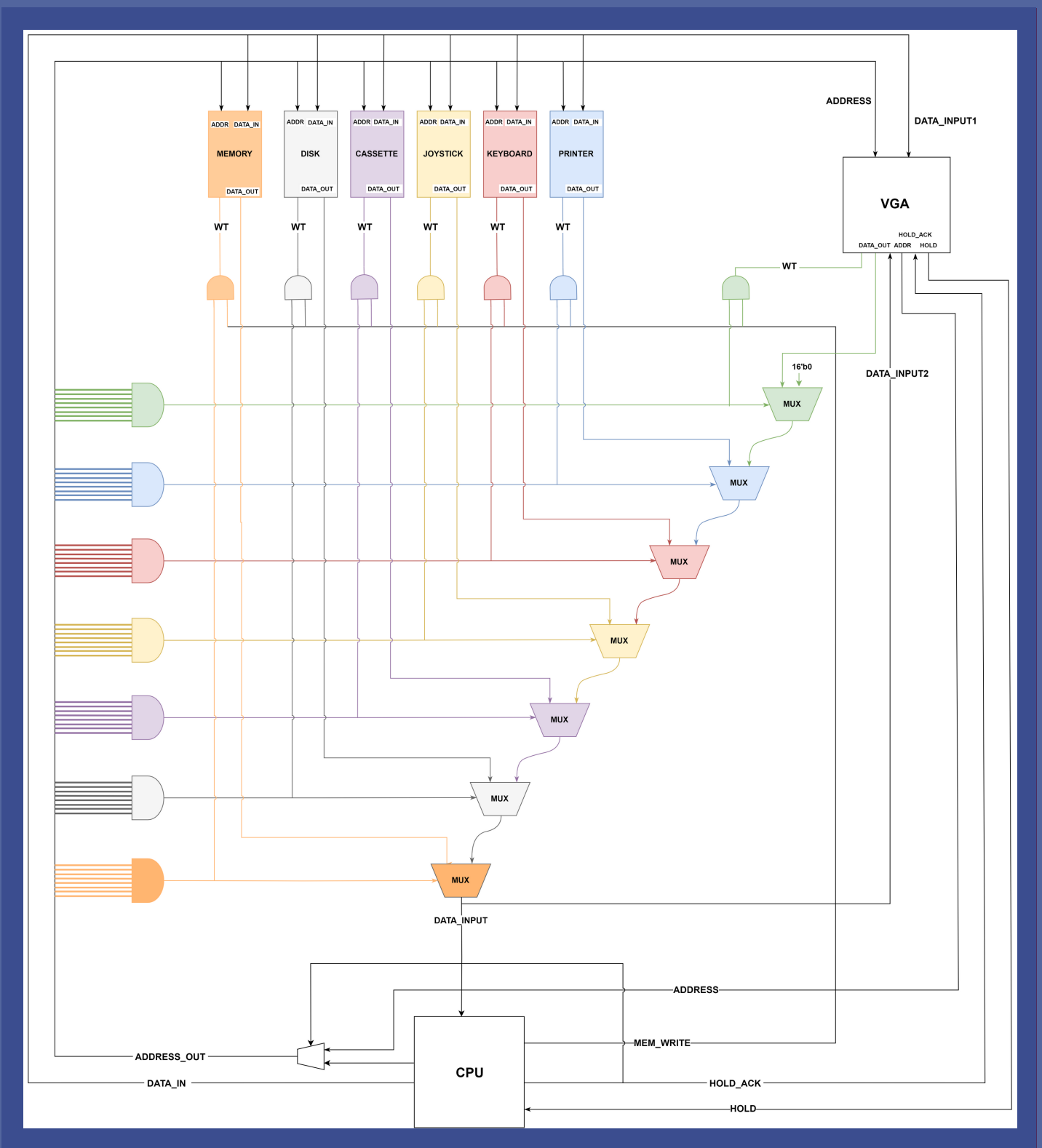
TECHNICAL APPROACH

- FPGA was utilized as a versatile component that served as a CPU, GPU, and memory within the system.
- To ensure optimal memory utilization and effective coordination between the GPU and CPU, we created "hold", "hold_ack", "hold_cpu" signals.


DE0-Nano FPGA Card


UART

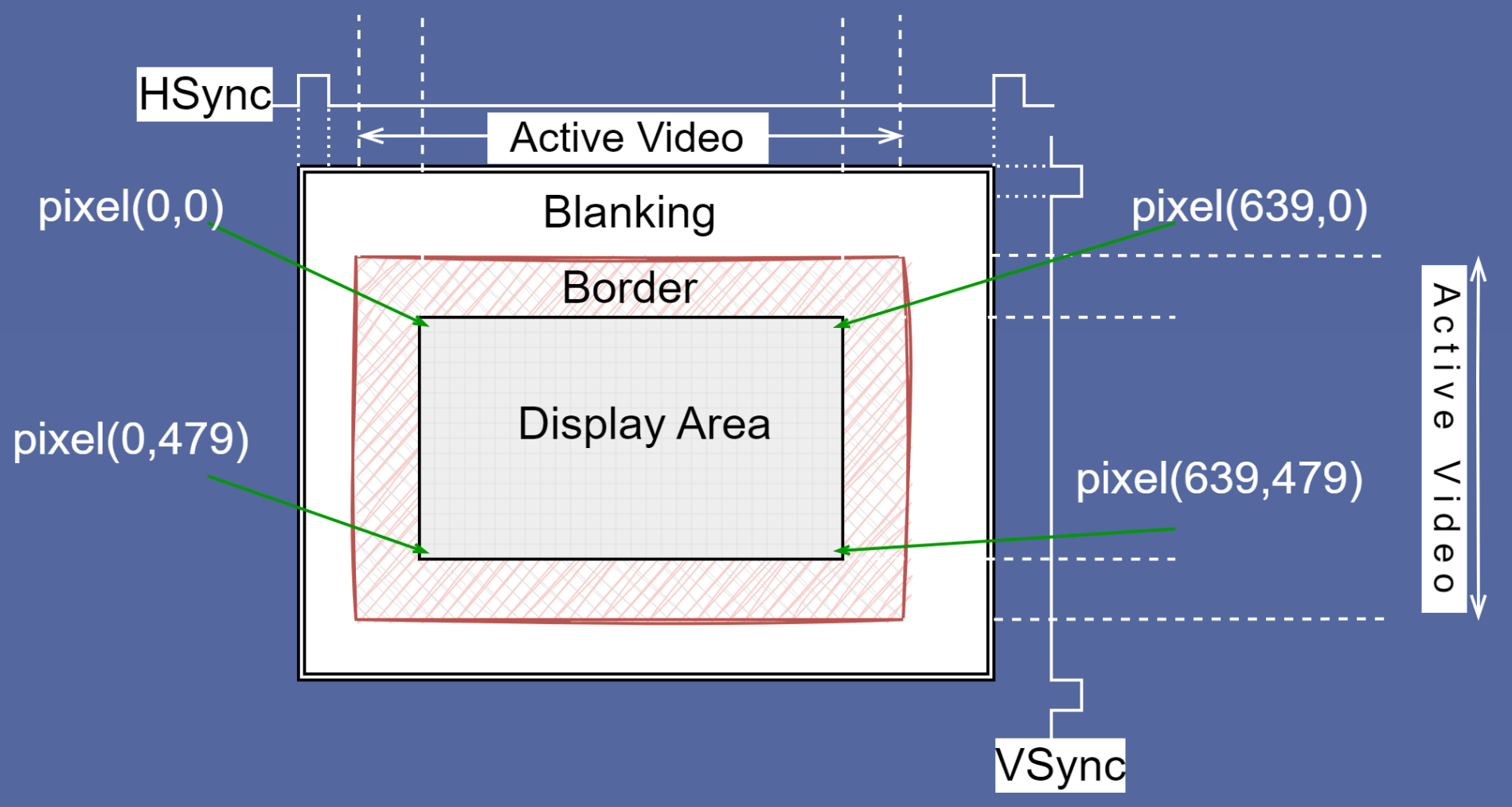
- UART adapter was used to transfer data from external devices to SDRAM of FPGA.



DISPLAY ARCHITECTURE

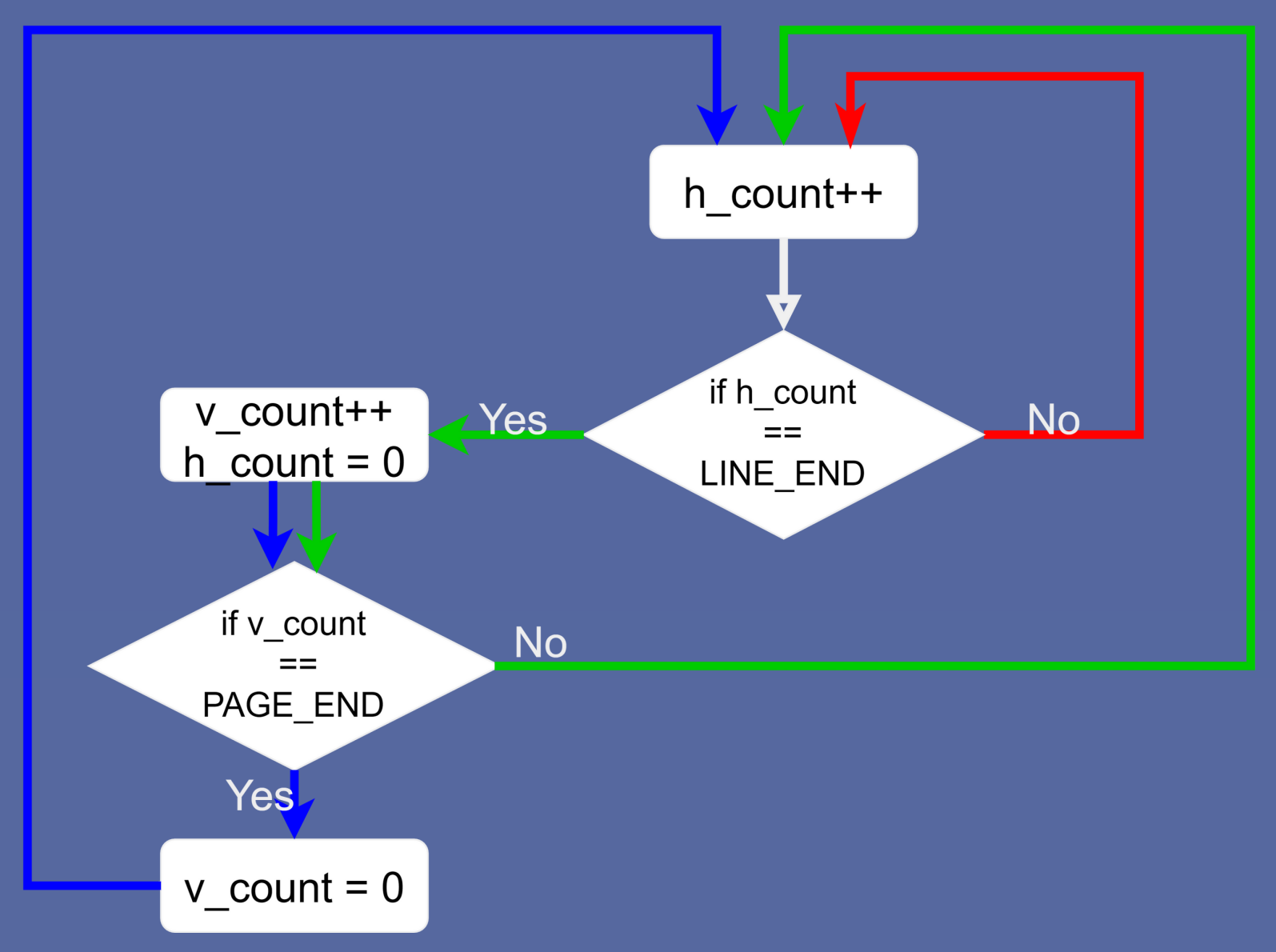
Understanding VGA Regions

- Blanking is necessary for preparing the image for scanning and creating other signals used during the scanning process
- Border lies outside of the image area and covers the inactive parts of the screen.
- The display area is the area where the displayed pixels are located.



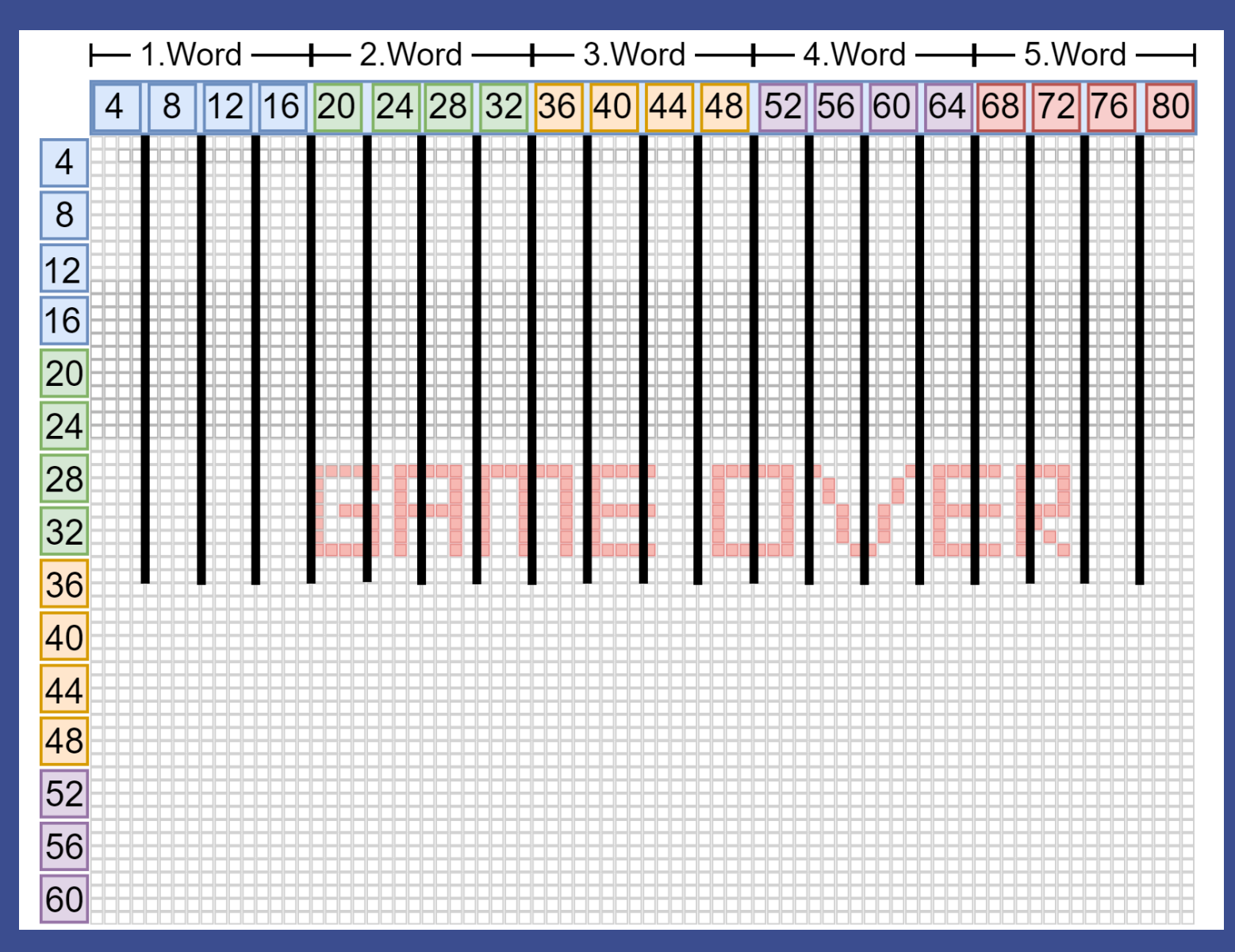
FSM of VGA Scanning

- The blue line represents vertical retrace. For this, the scanning process must end both horizontally and vertically.
- The green line represents horizontal retrace. For this, the scanning process only needs to end horizontally.
- The red line represents horizontal scan. In the FSM, this is shown by h_count being equal to LINE_END.

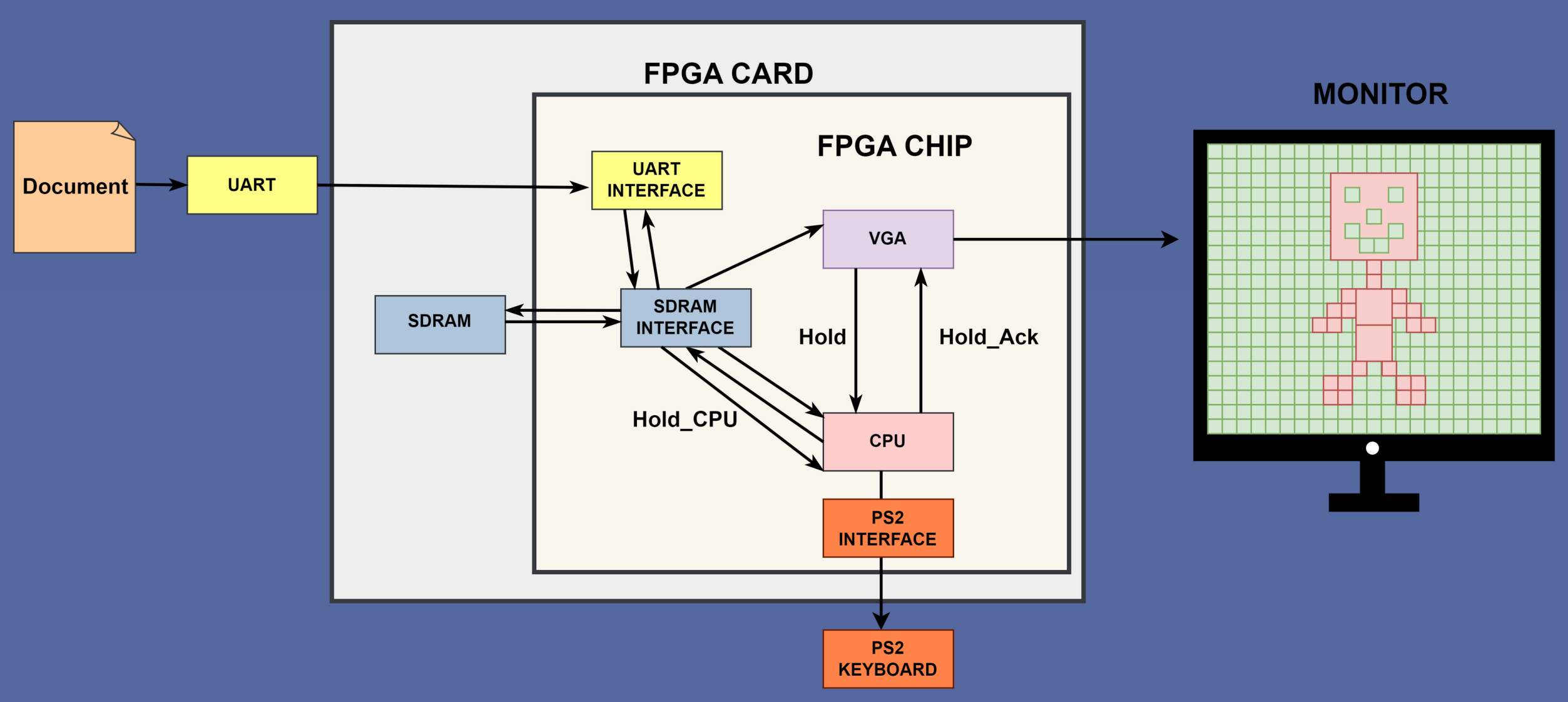


Resolution Adaptation and Pixel Screen

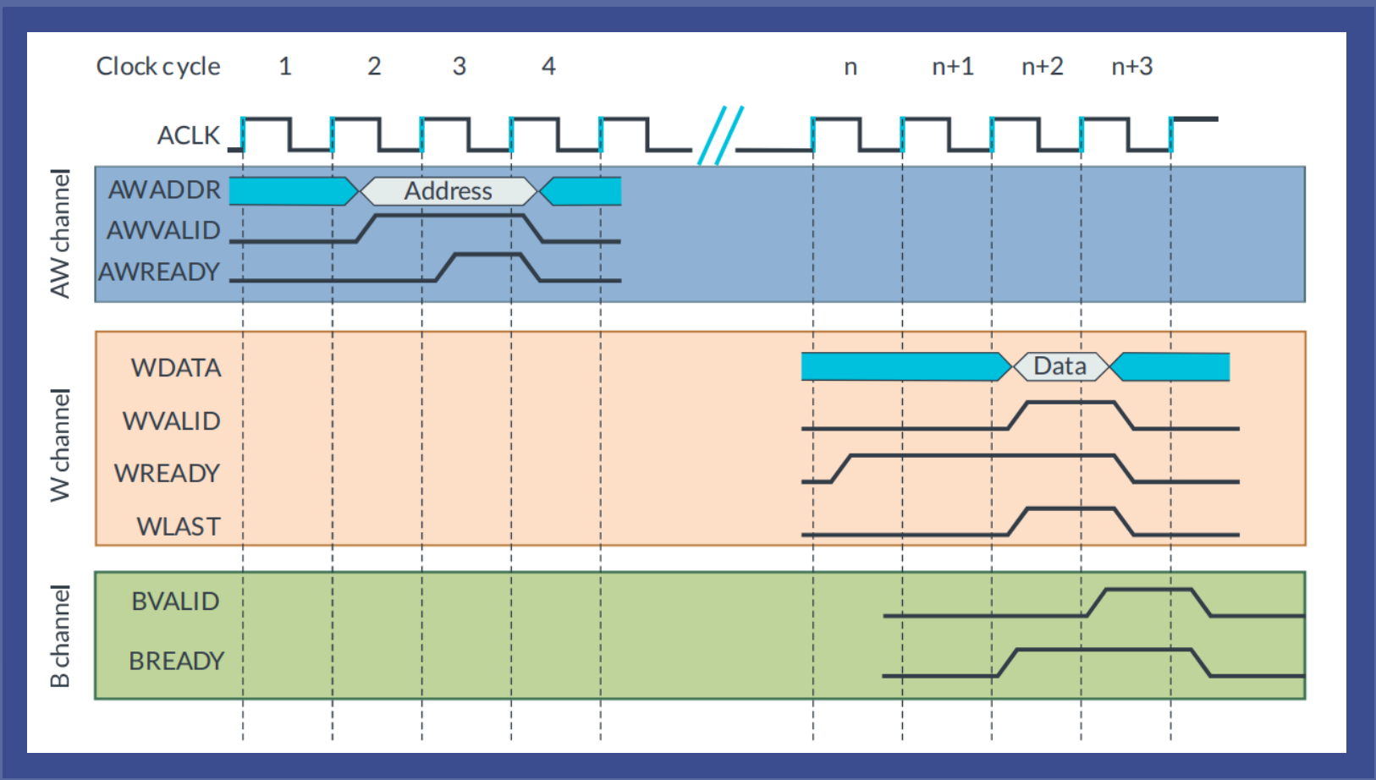
- Due to the insufficient amount of memory on the FPGA board, we had to lower the resolution of the current screen and decided to create an 80 x 60 pixel screen.
- We merged the existing pixels to create new 8x8 pixels and treated these newly formed 8x8 pixels as a single pixel.

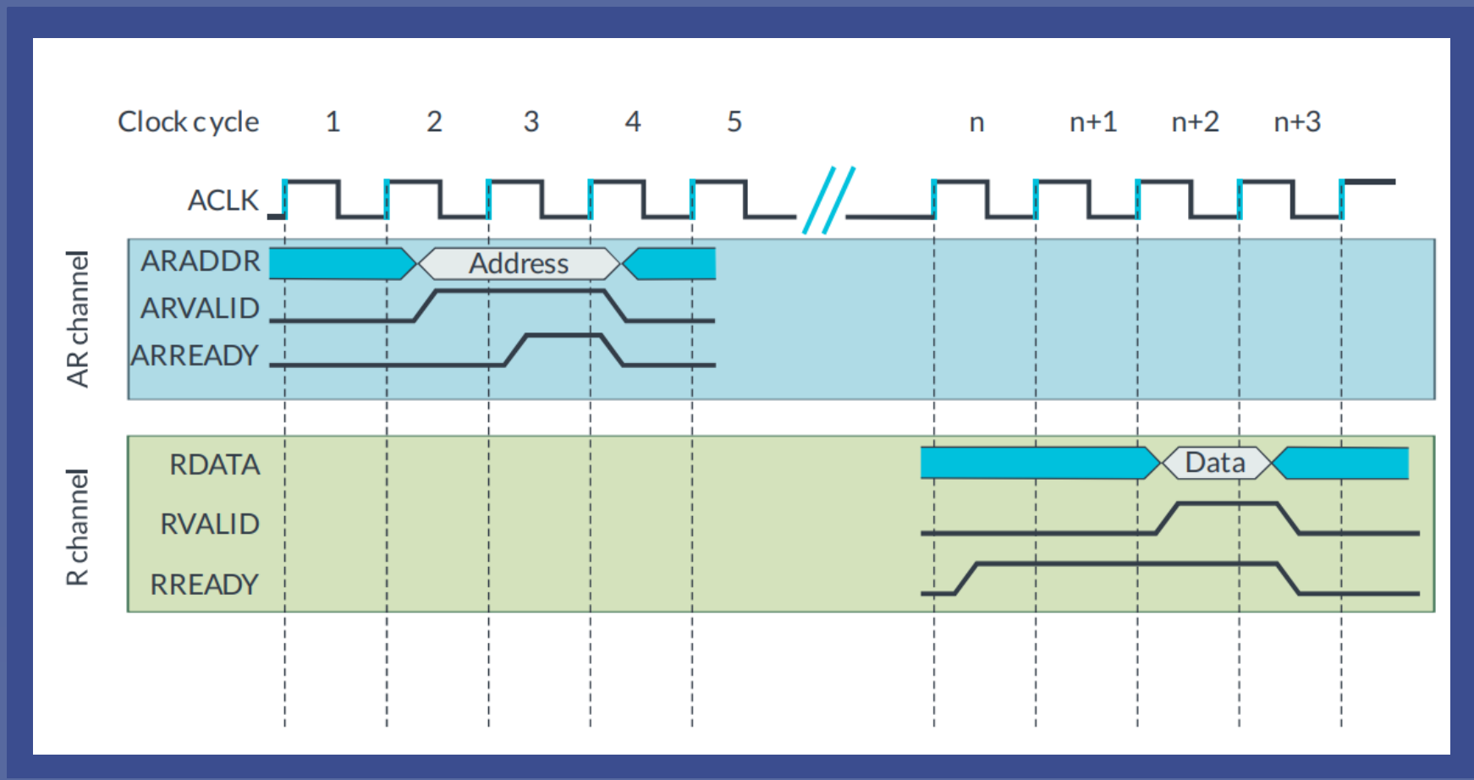


METHODOLOGY



- **Data Transfer with UART:** We transfer a graphic file from the computer to SDRAM using the UART controller and the HyperTerminal software via serial communication protocol.
- **Writing Data to the SDRAM:** UART controller data is written to SDRAM based on a defined protocol, allowing storage of up to 32MB of data independent of the FPGA's chip memory.
- **Hold Signal:** During the data writing process in VGA, a hold signal is sent from VGA to the CPU, which responds with a hold_ack signal. This signal prevents the CPU from altering the memory while the VGA is displaying an image, ensuring image integrity.
- **Data Transfer to the VGA:** Data from the SDRAM can be directly transmitted to the VGA module by the SDRAM controller for display on the screen.
- **Writing Data by the CPU:** The CPU writes data retrieved from the SDRAM to a specific address via the SDRAM interface, following the AXI4 protocol for SDRAM reading and writing.





Write Address Channel

- The master places the address on the AWADDR line and sets AWVALID to indicate the validity of the address.
- The slave sets AWREADY high to signal to indicate its ability to receive the address value.
- The handshake process concludes on the rising edge of clock cycle 4.
- In clock cycle n, the slave waits for data and sets WREADY high.

Write Data Channel

- In clock cycle n, the slave waits for data and sets WREADY high.
- In clock cycle n+2, the master places the data on the WDATA line and sets WVALID to indicate the validity of the data.
- The handshake process concludes on the rising edge of clock cycle n+3.

Write Response Channel

- The master asserts BREADY. The slave sets BVALID high to indicate the success of the write transaction. The handshake process concludes

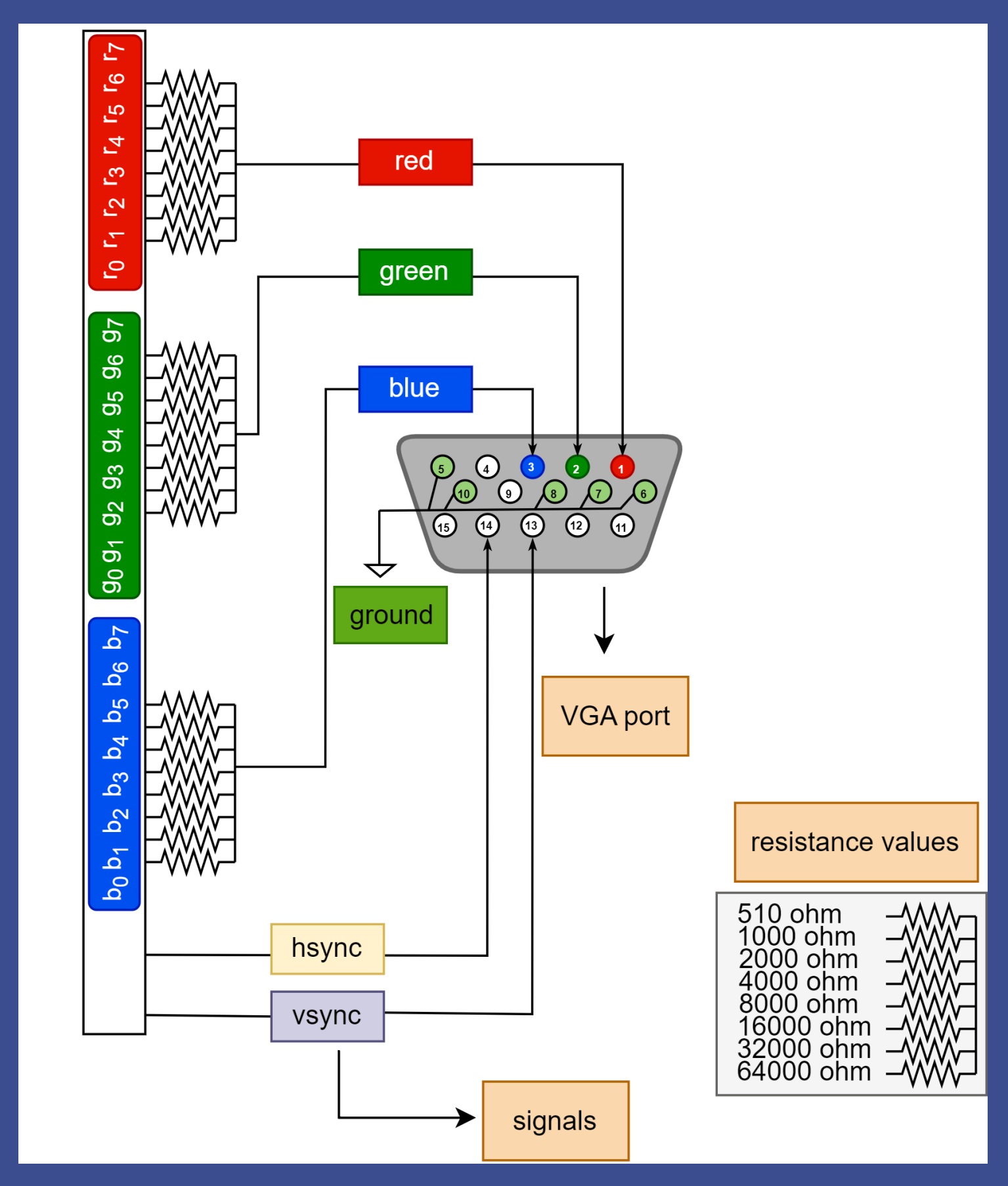
Read Address Channel

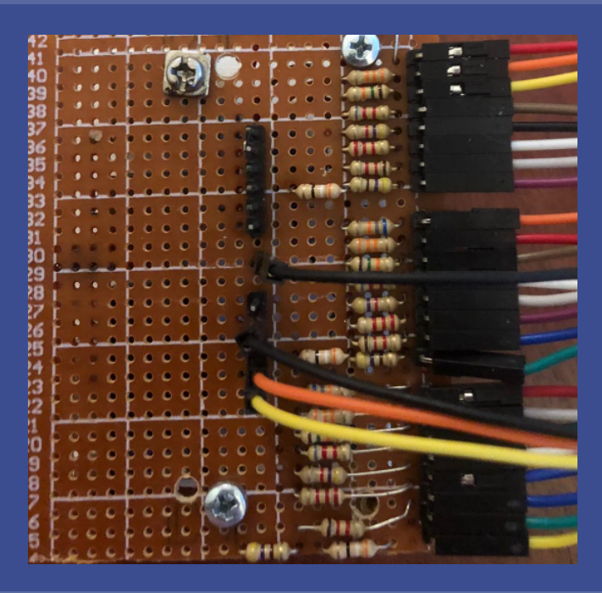
- The master sends the read address to the slave (ARADDR) and indicates the validity of the address
- The slave acknowledges its readiness to receive the address by asserting ARREADY

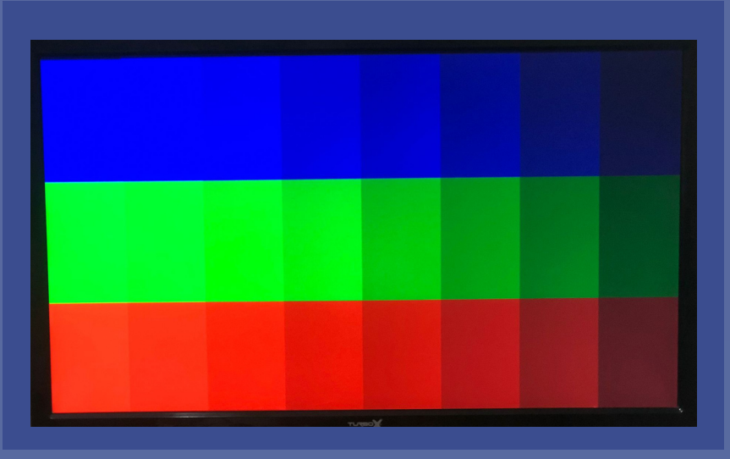
Read Data Channel

- The master indicates that it is waiting to receive the data by asserting RREADY.
- The slave retrieves the data and places it on RDATA then slave asserts RVALID.
- RREADY is already asserted by the master, and the handshake completes.

COLOR GENERATION







In FPGA, we could only output one tone of each of the RGB colors to the VGA monitor. The VGA card was running RGB colors with a single color bit. By connecting 8 parallel resistors for each color to the VGA card, we reduced the voltage value coming to the VGA card over the FPGA with these resistors, and we were able to get different voltage values. With these different voltage values, we were able to produce 16 million of RGB colors.

CONCLUSION

- By writing and reading data to the SDRAM chip on the FPGA board, we were able to expand the available memory space of the FPGA itself, utilizing the SDRAM as an extended storage area.
- By modifying the VGA card, we achieved the ability to represent RGB colors, originally represented by a single bit, with each color component expanded to 8 bits. This enhancement allowed us to obtain a broader range of shades and tones in our graphics display.

REFERENCES

[1] Arm Limited. Company. (2023, March). AMBA AXI Protocol Specification. Documentation – arm developer. https://developer.arm.com/documentation/ihi0022/latest/

[2] Chu, P. P. (2011). FPGA prototyping by Verilog examples: Xilinx spartan-3 version. Wiley.

TECHNOLOGIES USED

ARM SV ALTERA QUARTUS II ATARI 8 bit DEV C++