Design Document - Sketch Master

CSEE 4840

Embedded System Design

Ajay Vanamali(va2465), Bhoomi Shah(bbs2144), Manish Shankar(mr4264), Rahul Shanbhag(rns2166)

Spring 2023

Contents

1. Introduction

The growth in human-computer interaction has not only been in the quality of interaction, it has also experienced different implementations. From the basic keyboard and mouse, to camera controlled gestures, the ways to interact with devices has never been greater. We wish to showcase this with the use of a drawing tablet.

Our design aims to implement a HID interface between a Wacom Tablet and the FPGA. Using this device, we plan to implement a drawing system that provides users with an image from a randomized set of low-resolution images. The user will be tasked with tracing this image on the surface of the tablet and once completed, the user’s attempt will be graded according to a grading algorithm.

A Wacom Tablet or any other drawing tablet, is usually registered as a HID device and follows the similar USB HID packet format and behaves similar to a USB mouse. We plan to leverage this while designing the drivers.

2. System Design and Block Diagram

A high level overview of our system, includes the kernel module and driver to detect and process the pen strokes. The Wacom tablet USB packets will determine what pixels are modified. The modified pixels are stored in the frame buffer and the frame buffer is used to write into the SDRAM. The FPGA will read from the SDRAM and display on the monitor using the VGA port.

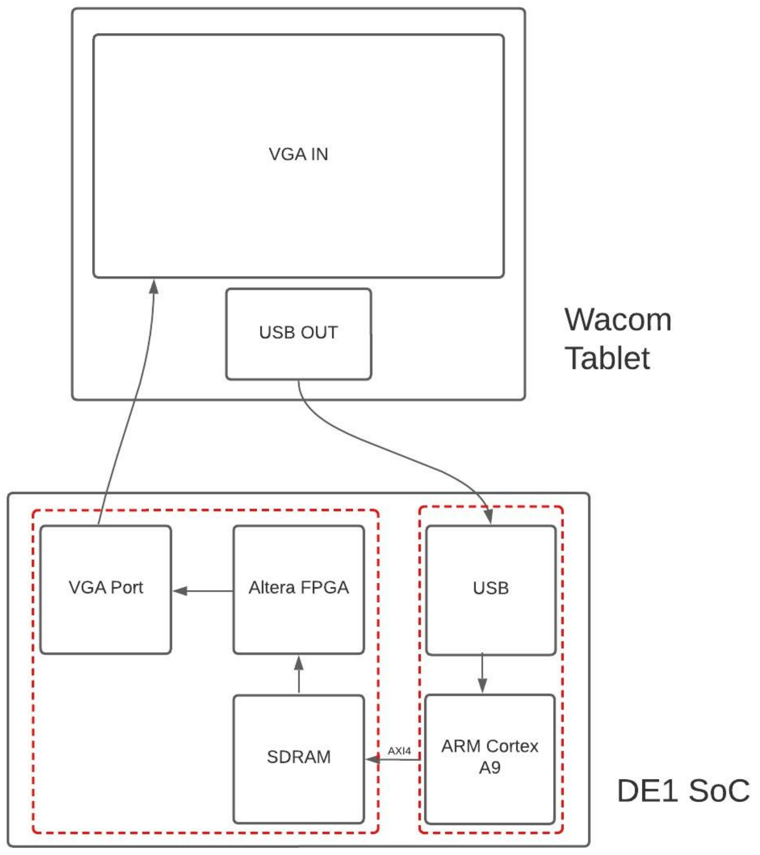


Figure 1: Block diagram of Sketch Master

3. Algorithms

Check Algorithm - Comparing Reference and Live image

Pseudocode

```
1.  Load reference image with its outline
2.  Load student image with its outline
3.  Compute the absolute difference between the binary images of the outlines
4.  Count the number of pixels that exceed a threshold value in the final image
    consisting of superimposed images
5.  Initialize a penalty counter to 0
6.  For each pixel (x,y) in the reference image:
7.  Calculate the Euclidean distance between the corresponding pixel in the student
    image and the reference image
        a.  If the distance is greater than ‘x’ pixels:
        b.  Increment the penalty counter by the difference between the distance and
            pixels
8.  Calculate the final score as the percentage of correct pixels minus the penalty
9.  Assign a grade based on the final score
10. Print the grade (UI / 7 Segment Display)
```

4. Resource Budget

Frame Buffer:
To compute the approximate memory requirement for storing a frame of data, we consider **640** pixels and there are **480** vertical rows. This means the frame size is **640*480** = **307200**.
In this case we would need ~ **307k*8** or **307K** bytes. Where **8** is the **8** bits of color data per pixel.
We use the **4:2:2** format for **RGB** where there are **4** red bits, **2** green bits, and **2** blue bits summing to a total of **8** bits.

5. Hardware/Software Interface

For this project we plan to use the De1-SoC Evaluation Board from Terasic which holds a Cyclone V SoC FPGA device. The FPGA and the ARM core are connected through the Advanced Microcontroller Bus Architecture (AMBA) Advanced eXtensible Interface (AXI) bridges. Although these two components can function entirely independently, communication between the ARM core and FPGA fabric can be a bottleneck for the overall system. That is why they are connected with two high-speed 128bit AMBA AXI bus bridges called HPS to FPGA and FPGA to HPS. The data path width for both bridges can be configured via QSYS to 32, 64, and 128. By having this variable data width, the bridge can be tuned for maximum performance when there is a communication between the FPGA fabric and the HPS.

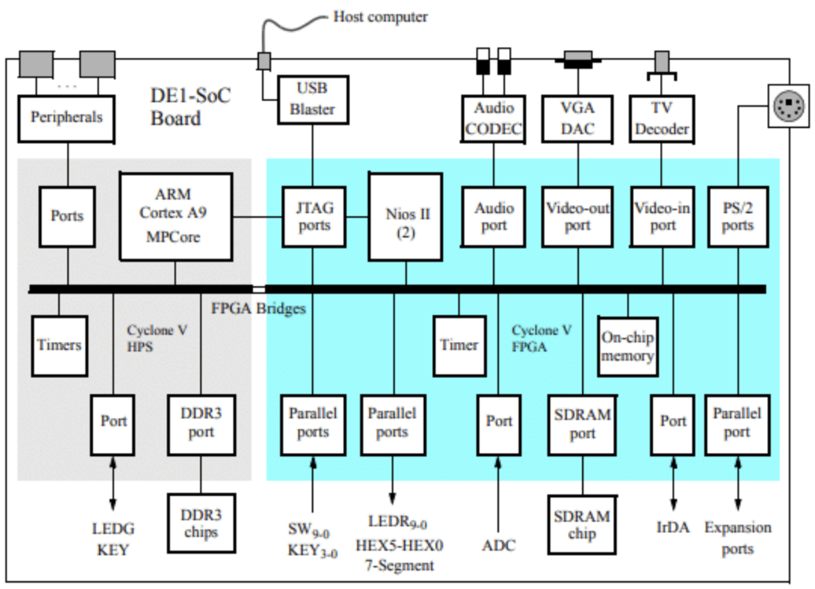


Figure 2: Block Diagram of De1- SoC

We consider the following peripherals for our design:

- VGA Video Output – Displaying the drawing board
- Tablet USB Interface – To handle pen strokes on the tablet
- SDRAM Controller - Storing the framebuffer for VGA

The DE1-SoC has the following specifications:-

1. **Memory**: The DE1-SoC Computer has an SDRAM, as well as two memory modules implemented using the on-chip memory inside the FPGA. These memories are described below.
    a. **SDRAM**: An SDRAM Controller in the FPGA provides an interface to the 64 MB synchronous dynamic RAM (SDRAM) on the DE1-SoC board, which is organized as 32M x 16 bits. It is accessible by the A9 processor using word (32-bit), halfword (16-bit), or byte operations, and is mapped to the address space 0xC0000000 to 0xC3FFFFFF.
    b. **BRAM** (On-Chip Memory): The DE1-SoC Computer includes a 256 KB memory that is implemented inside the FPGA. This memory is organized as 64K x 32 bits, and spans addresses in the range 0xC8000000 to 0xC803FFFF. The memory is used as a pixel buffer for the video-out and video-in ports.

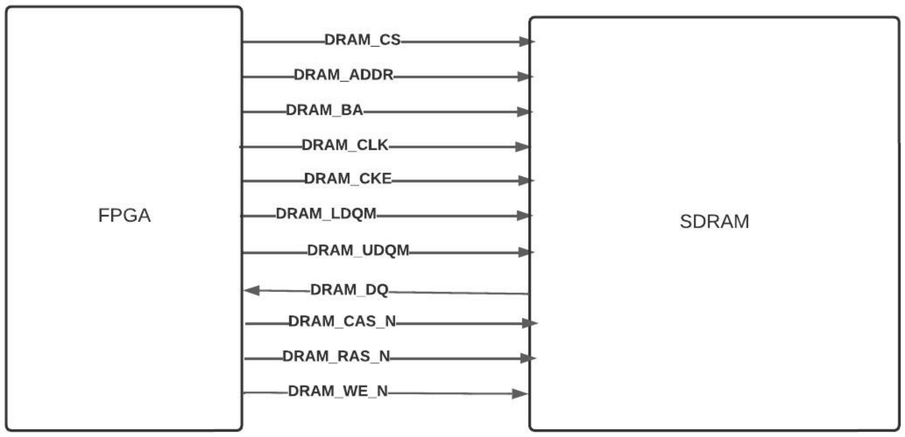
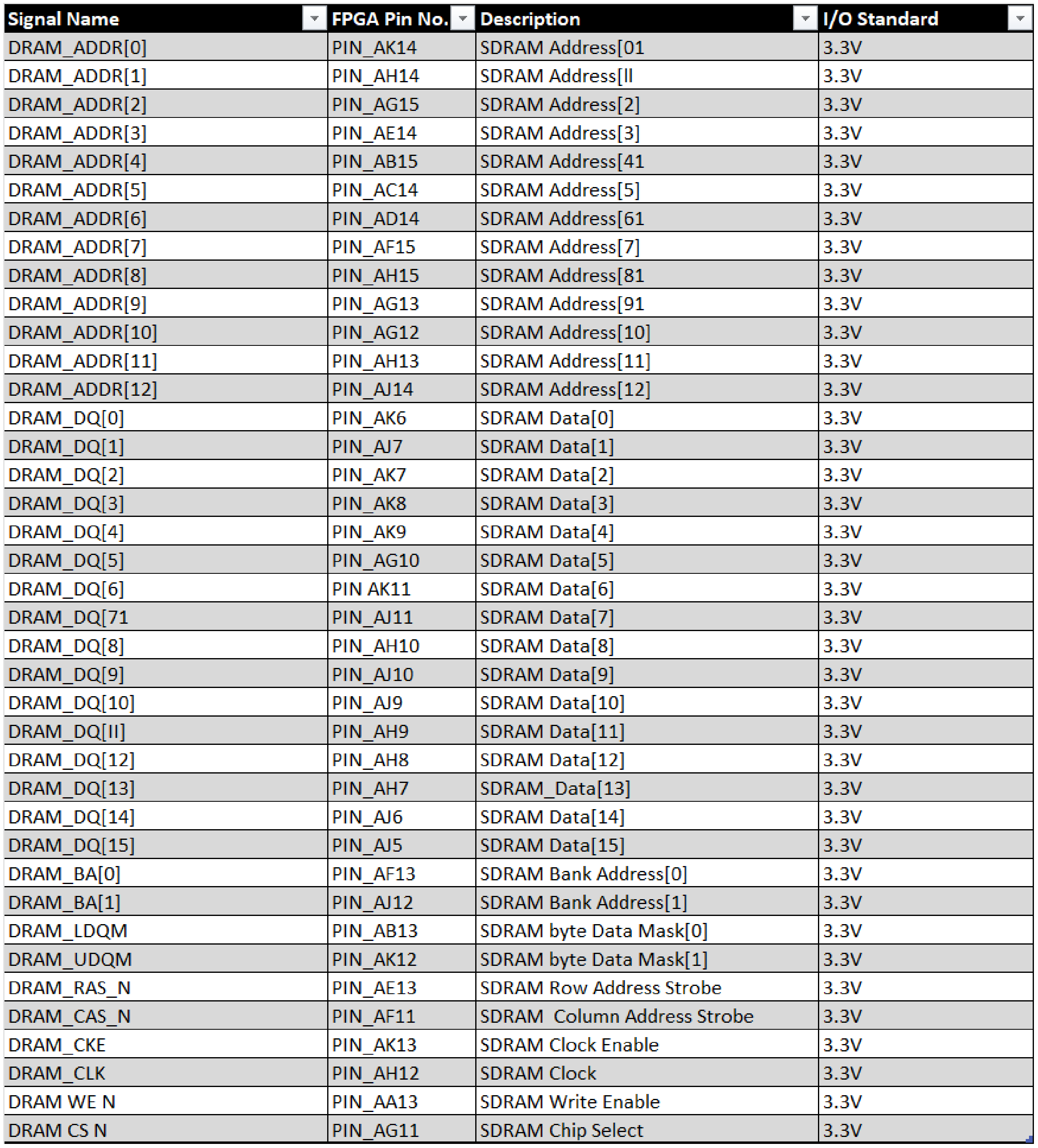


Figure 3: FPGA and SDRAM Interface



Signal Name	FPGA Pin No.	Description	I/O Standard
DRAM_ADDR[0]	PIN_AK14	SDRAM Address[01	3.3V
DRAM_ADDR[1]	PIN_AH14	SDRAM Address[Il	3.3V
DRAM_ADDR[2]	PIN_AG15	SDRAM Address[2]	3.3V
DRAM_ADDR[3]	PIN_AE14	SDRAM Address[3]	3.3V
DRAM_ADDR[4]	PIN_AB15	SDRAM Address[41	3.3V
DRAM_ADDR[5]	PIN_AC14	SDRAM Address[5]	3.3V
DRAM_ADDR[6]	PIN_AD14	SDRAM Address[61	3.3V
DRAM_ADDR[7]	PIN_AF15	SDRAM Address[7]	3.3V
DRAM_ADDR[8]	PIN_AH15	SDRAM Address[81	3.3V
DRAM_ADDR[9]	PIN_AG13	SDRAM Address[91	3.3V
DRAM_ADDR[10]	PIN_AG12	SDRAM Address[10]	3.3V
DRAM_ADDR[11]	PIN_AH13	SDRAM Address[11]	3.3V
DRAM_ADDR[12]	PIN_AJ14	SDRAM Address[12]	3.3V
DRAM_DQ[0]	PIN_AK6	SDRAM Data[0]	3.3V
DRAM_DQ[1]	PIN_AJ7	SDRAM Data[1]	3.3V
DRAM_DQ[2]	PIN_AK7	SDRAM Data[2]	3.3V
DRAM_DQ[3]	PIN_AK8	SDRAM Data[3]	3.3V
DRAM_DQ[4]	PIN_AK9	SDRAM Data[4]	3.3V
DRAM_DQ[5]	PIN_AG10	SDRAM Data[5]	3.3V
DRAM_DQ[6]	PIN AK11	SDRAM Data[6]	3.3V
DRAM_DQ[71	PIN_AJ11	SDRAM Data[7]	3.3V
DRAM_DQ[8]	PIN_AH10	SDRAM Data[8]	3.3V
DRAM_DQ[9]	PIN_AJ10	SDRAM Data[9]	3.3V
DRAM_DQ[10]	PIN_AJ9	SDRAM Data[10]	3.3V
DRAM_DQ[II]	PIN_AH9	SDRAM Data[11]	3.3V
DRAM_DQ[12]	PIN_AH8	SDRAM Data[12]	3.3V
DRAM_DQ[13]	PIN_AH7	SDRAM_Data[13]	3.3V
DRAM_DQ[14]	PIN_AJ6	SDRAM Data[14]	3.3V
DRAM_DQ[15]	PIN_AJ5	SDRAM Data[15]	3.3V
DRAM_BA[0]	PIN_AF13	SDRAM Bank Address[0]	3.3V
DRAM_BA[1]	PIN_AJ12	SDRAM Bank Address[1]	3.3V
DRAM_LDQM	PIN_AB13	SDRAM byte Data Mask[0]	3.3V
DRAM_UDQM	PIN_AK12	SDRAM byte Data Mask[1]	3.3V
DRAM_RAS_N	PIN_AE13	SDRAM Row Address Strobe	3.3V
DRAM_CAS_N	PIN_AF11	SDRAM Column Address Strobe	3.3V
DRAM_CKE	PIN_AK13	SDRAM Clock Enable	3.3V
DRAM_CLK	PIN_AH12	SDRAM Clock	3.3V
DRAM WE N	PIN_AA13	SDRAM Write Enable	3.3V
DRAM CS N	PIN_AG11	SDRAM Chip Select	3.3V

Table 1 : SDRAM Pin interface

2. **Avalon bus**: Avalon-MM interface that supports read and write transfers with agent-controlled wait requests. The agent can stall the interconnect for as many cycles as required by asserting the waitrequest signal. If an agent uses waitrequest for either read or write transfers, the agent must use waitrequest for both. An agent typically receives address, byteenable, read or write, and write data after the rising edge of the clock. An agent asserts waitrequest before the rising

clock edge to hold off transfers. When the agent asserts waitrequest, the transfer is delayed. While waitrequest is asserted, the address and other control signals are held constant. Transfers complete on the rising edge of the first clk after the agent interface de-asserts the wait request. There is no limit on how long an agent interface can stall.

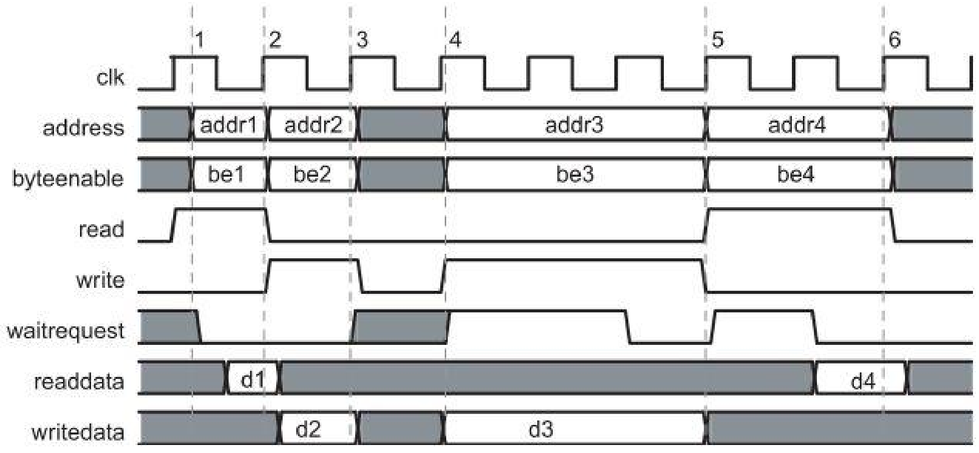


Figure 4: Avalon Bus Timing Bus

3. **VGA Port**: The DE1-SoC board has a 15-pin D-SUB connector populated for VGA output. The VGA synchronization signals are generated directly from the Cyclone V SoC FPGA, and the Analog Devices ADV7123 triple 10-bit high-speed video DAC (only the higher 8-bits are used) transforms signals from digital to analog to represent three fundamental colors (red, green, and blue). It can support up to SXGA standard (1280*1024) with signals transmitted at 100MHz.

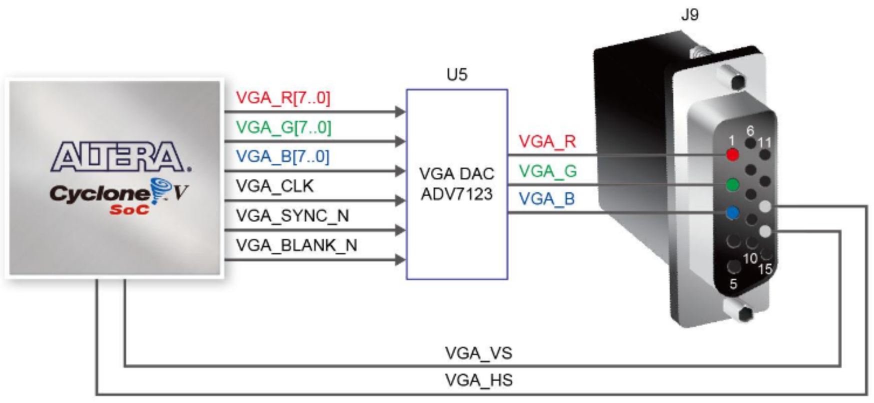


Figure 5: VGA Port Interface

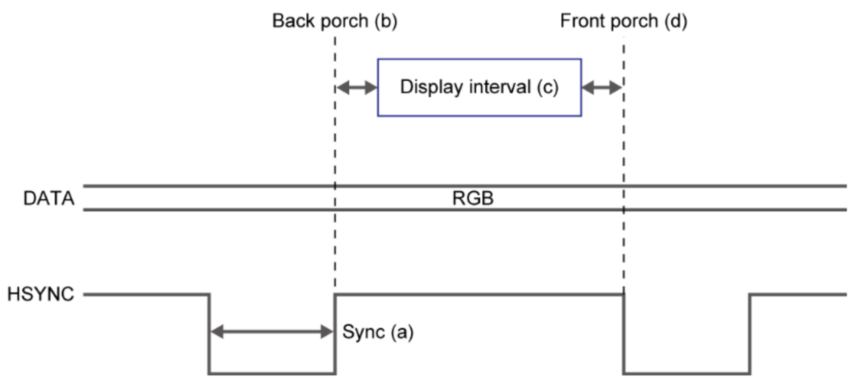


Figure 6: VGA Horizontal Timing Diagram

4. **VGA Display** : We will be using the Raster Scan Algorithm for the VGA Display. Raster scan algorithm is a popular method for generating images on a computer screen. The process of displaying using raster scan algorithm follows these steps :

- Setting up the VGA interface: Configure the FPGA to output the VGA signal, including the horizontal and vertical synchronization signals and the pixel clock. This involves setting up the timing parameters for the VGA interface, such as the resolution and refresh rate.
- Initialize the frame buffer: Allocate memory in the FPGA for a frame buffer that will hold the image data. This buffer should be large enough to store the entire image that you want to display.
- Begin drawing: Start drawing the image by iterating through each pixel on the screen using a nested loop. Begin at the top left corner of the screen and move horizontally across each row, then move down to the next row and repeat until the entire screen is filled.
- Determine the color of each pixel: To determine the color of each pixel, use a mathematical formula that maps the pixel's location on the screen to its corresponding location in the frame buffer. This formula will take into account the resolution and refresh rate of the VGA interface.
- Write pixel data to the frame buffer: Once the color of each pixel has been determined, write the pixel data to the frame buffer in the appropriate location. This involves using the memory address of the frame buffer to store the color data for each pixel.
- Output the frame buffer to the VGA interface: Once the frame buffer has been fully populated with pixel data, output the frame buffer to the VGA interface. This involves sending the pixel data to the VGA controller at the appropriate times based on the VGA timing parameters.
- Repeat: Once the first frame has been output to the VGA interface, repeat the process to generate subsequent frames for animation or video playback.

6. Milestones

Tentative Date	Milestone planned
3/31	Driver Implementation - Interfacing Wacom Tablet with a Linux System
4/7	Implement VGA Controller and Analyze the Wacom packets
4/14	Implement SDRAM Controller and interface it with VGA
4/21	Implement Sketch Check Algorithm
4/28	Create UI for Sketch Board
5/5	Final Testing
5/5	Final Presentation

7. References

- DE-1 SoC Manual and Datasheet: https://www.intel.com/content/dam/altera-www/global/en_US/portal/dsn/42/doc-us-dsnbk-42-1004282204-de1-soc-user-manual.pdf
- Altera Memory System Design - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/edh_ed51008.pdf
- FPGA based Virtual Drawing System: https://apiar.org.au/wp-content/uploads/2017/07/22_APJCECT_v3i2_ICT-259-267.pdf
- VGA and SDRAM Controller: https://hsel.co.uk/2016/05/10/fpga-vga-and-sdram/