ECE 368 Digital Design

Spring 2014

Lab 3 – VGA Video and PS2 Keyboard Interface

Dates Performed: Wednesday, February 26th , 2014

Submission Date: Monday, March 3rd , 2014

Team #2 :

Massarrah Tannous ________________________

Daniel Noyes ______________________________

Table of Contents

Problem Statement

For this lab, we were to compete the design of the VGA video controller. Moreover, we had to create an interface that interface PS2 keyboard interface. The PS2 Keyboard interface is mainly based from the previous lab. These interfaces will allow the used to type and display the ASCII character on a monitor of type VGA.

In order to implement this assignment, the Xilinx ISE software tools and the Digilent Spartan-3E Starter Board are to be used along with a PS/2 keyboard. The Xilinx software allows designing the VHDL code, simulating through the test bench and programming the Spartan 3E board.

This lab allowed using the concept of FSM machine and lookup tables apart from the VGA and PS/2 controllers.

RTL Block Diagram (Hand Drawn Design)

The original designs the attached engineer papers which are labeled as “Original Designs”.

VHDL Entities Specification

The specifications of each entity used for the VGA and the PS2 controller are shown in Appendix A.

System design top level entity

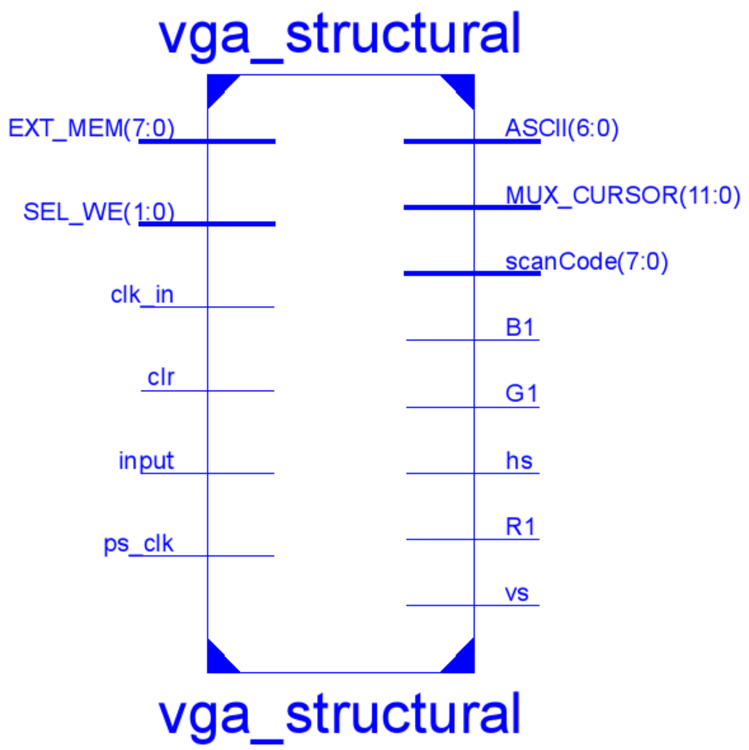


Figure 1: Top level entity for the VGA controller

VHDL Code

The VHDL code for both the VGA has been emailed as a zip file to Dr. Fortier.

Generated RTL Designs with comparisons.

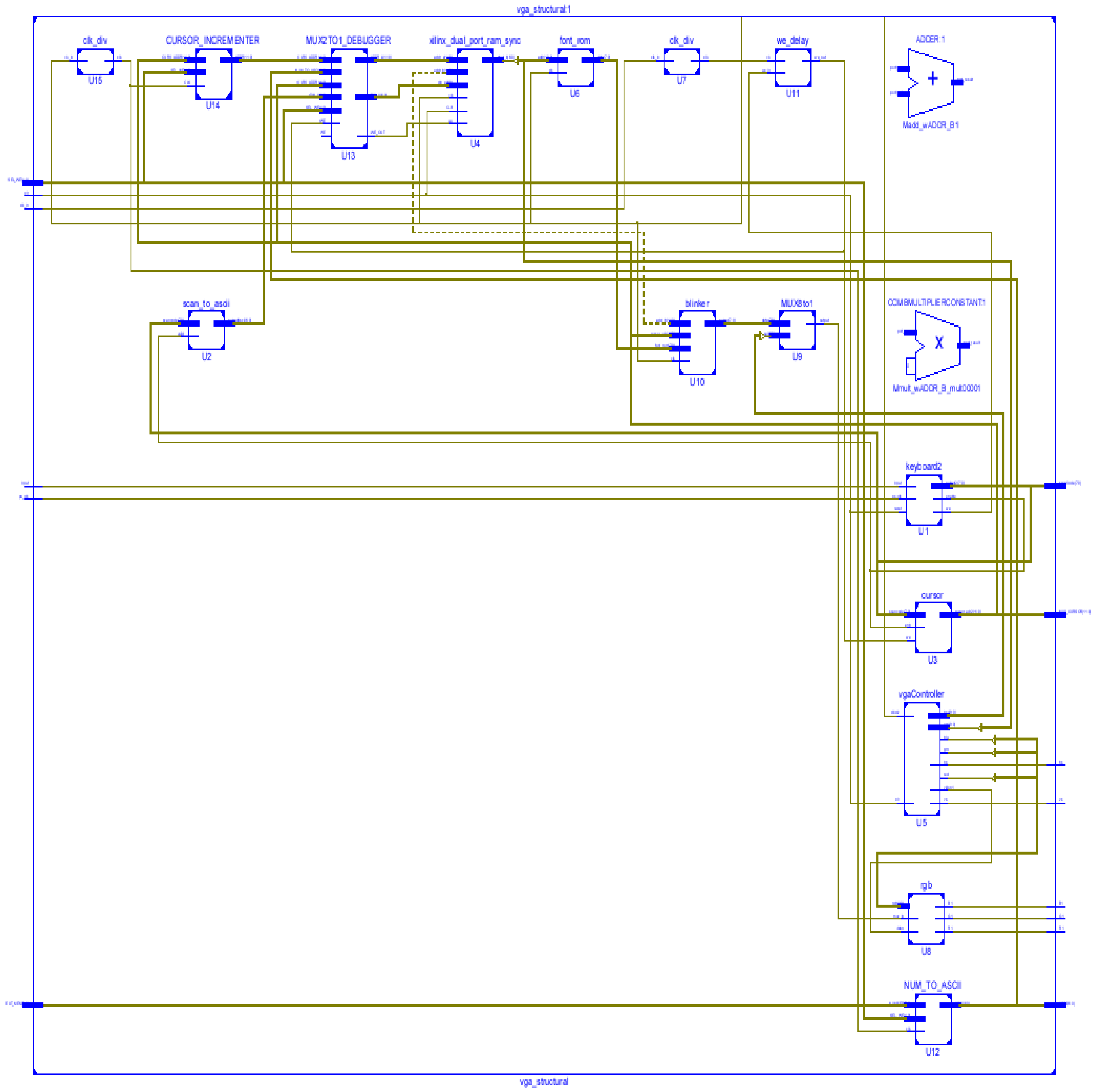


Figure 2: RTL Diagram for the VGA controller

Between the various RTL diagrams we incorporated, there were various types of entities in the lab. A lot of more RTL diagrams were included and various ones were based from lab 2. This Lab was its own entity with just the keyboard controller used from the previous lab. Overall with each RTL digram we used on the lab, except for the keyboard controller and ascii converter. The lab had a whole new segments of code, the first new section of the code was the VGA controller which was originally supplied to us from DR. Fortier for this lab. We then incorporated a overlay converter from key code to ascii and replaced the keyboard driver with a fully tested one

Test Plan

Overall plan (simulation and HW): For the overall plan in the simulation, the VGA was used to act as a test for the inputs from the keyboard to the display.

Simulation: The current simulation for the VGA memory controller is the following figure 4. When we incorporate the PS2 driver, it will wait till it grabs a 8 bit code from the keyboard and then was sent through a convert to see the ascii. The testing and observation of the Lab done more in the hardware level instead of simulation. This was due to the testing each keycode to ascii and then display on the VGA.

Testbench and simulation results: Overall the test bench and simulation results we on the mark for each function they had to achieve. The Ascii tables were included in order to view what does the keycode to ascii will output.

Hardware test plan and procedure (what will be tested, how you will do it): For the hardware test plan and procedure's, the first part was to observer the VGA changes with the keyboard inputs. Then observed the changes that the keyboard did upon the cursor input. The cases that are tested are and their results are summarized in table 1.

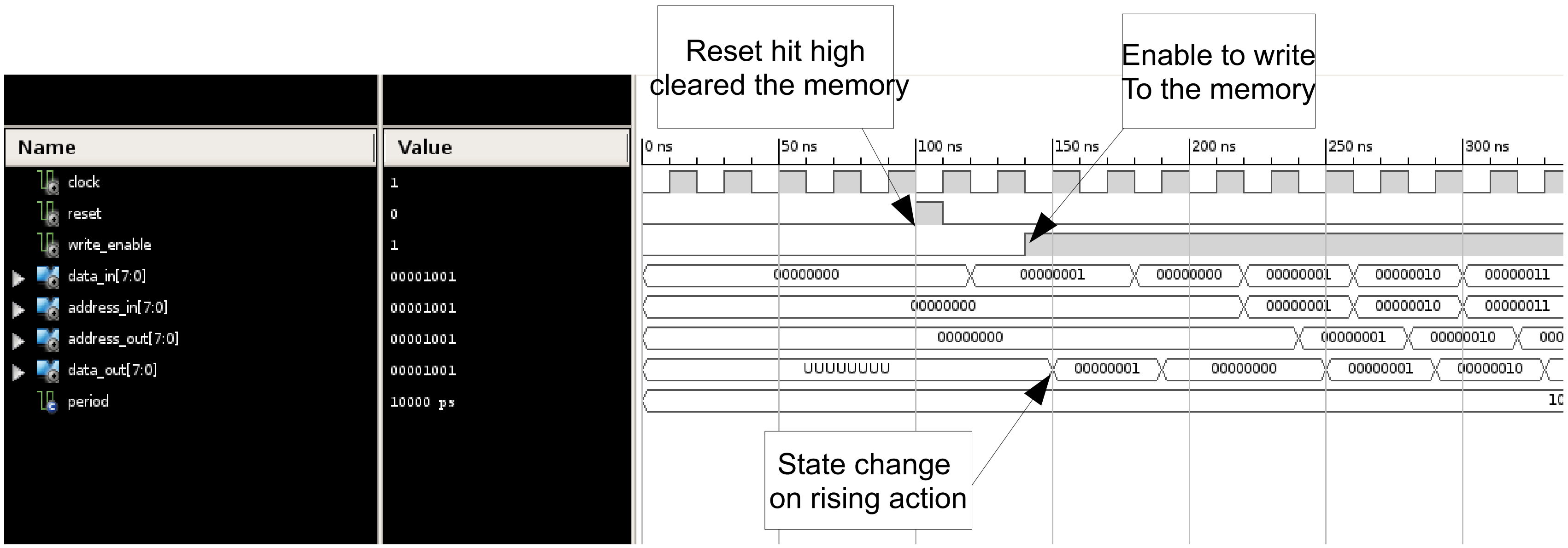


Figure 3: Simulation of writing to memory

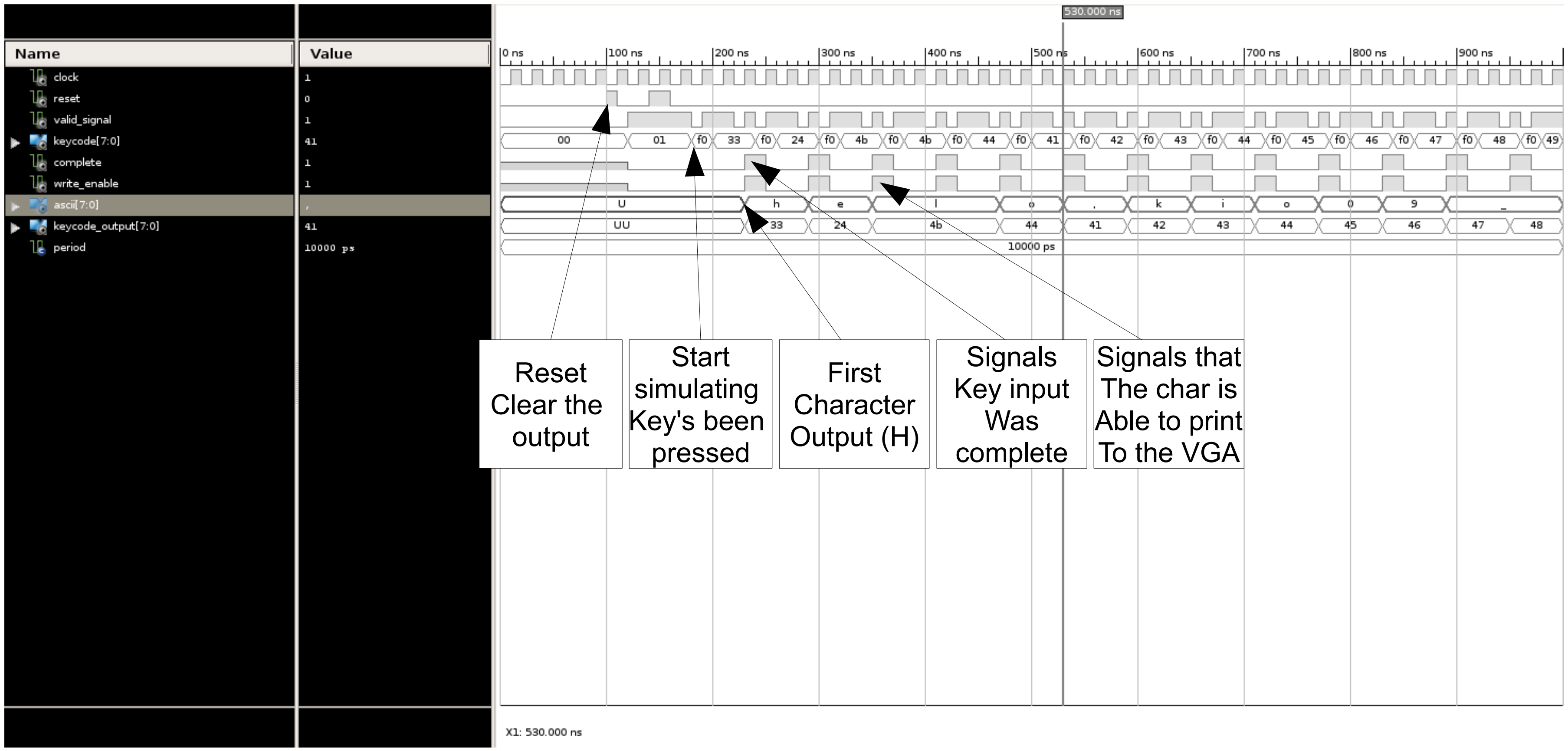


Figure 4: Simulation of Keycode to Ascii output

UPPER CASE					
ASCII CHAR	ASCII OUTPUT	KeyCode Input	ASCII CHAR	ASCII OUTPUT	KeyCode Input
A	41	1C	N	4E	31
B	42	32	O	4F	44
C	43	21	P	50	4D
D	44	23	Q	51	15
E	45	24	R	52	2D
F	46	2B	S	53	1B
G	47	34	T	54	2C
H	48	33	U	55	3C
I	49	43	V	56	2A
J	4A	3B	W	57	1D
K	4B	42	X	58	22
L	4C	4B	Y	59	35
M	4D	3A	Z	5A	1A

Table 1: Shows the results for all upper case ASCII characters

lower case					
ASCII CHAR	ASCII OUTPUT	KeyCode Input	ASCII CHAR	ASCII OUTPUT	KeyCode Input
a	61	1C	n	6E	31
b	62	32	o	6F	44
c	63	21	p	70	4D
d	64	23	q	71	15
e	65	24	r	72	2D
f	66	2B	s	73	1B
g	67	34	t	74	2C
h	68	33	u	75	3C
i	69	43	v	76	2A
j	6A	3B	w	77	1D
k	6B	42	x	78	22
l	6C	4B	y	79	35
m	6D	3A	z	7A	1A

Table 2: Shows the results for all lower case ASCII characters

	Keyboard Inputs	Monitor Output	Test Results	Comments
Regular input	hello world, we are team two.	hello world, we are team two.	Passed	With Slow typing: Cursor hiding the last letter
	hello world we are team two	helllooo wld we are teaaamm twoo	Passed- Not favorable	With fast typing
	1234567890	1234567890	Passed	
	'Enter key' 'Enter key' 'Enter key'	Cursor moved	Passed	Moved the cursor to the beginning of the third line below,
	'hello' + 'Backspace key'	hell	Passed	
SHIFT	Shift + 'hello'	HELLO	Passed	
	Shift + 'h ' 'ello'	HE||o	Passed -Not favorable	'H' was only supposed to uppercase, it treated the following letter as lower case.
Arrow keys	Up	Cursor moved	Passed	In some senario the key must pressed more than once. Conversely, some scenarios the it moves the cursor as if the key was pressed more than once.
	Down	Cursor moved	Passed	
	Left	Cursor moved	Passed	
	Right	Cursor moved	Passed	

Table 3: Shows the test results of all test based on various inputs.

UCF File

The UCF for the VGA controller has been emailed to Dr. Fortier.

Conclusion

Through this lab we learned a lot more about using the VHDL syntax and the XILINX software. Furthermore, we learned how to improve our programing skills and interact with hardware such much more by using the keyboard and the VGA Displayer.

We implemented the VGA displayer, where the results were as expected in Hardware. This was done by having data pushed on a duel port ram which will then be displayed on the screen. We were able to implement simulations for various components of the VGA in order to debug and isolate issues in the code.

Reflection

This Lab overall felt harder then the previous two in a logarithmic scale. At first we tested the code Dr. Fortier gave us and it took a while to debug. After going over the methods of how each component operates, it brought us closer to fully understanding the overall structure of the code. This lab had us go into the code more than both previous labs combined. Once we completed the lab, I(Daniel) went and started with coding a dumper for the ALU for the extra credit but as the more I tried to get a dumper working the more I wanted to clean up the code and fix small bugs in the overall project. This one small Idea which lead me to clean out all the code and separate them into smaller components.

After Cleaning up the code for version 1.0, The ALU was then initialized, unfortunately due to time constraints. The extra credit was halted in production due to output errors of acquired timing delays in the keyboard to VGA display. The code was looked through and the problem is still hidden in plain sight. Later on when free time is allocated, maybe the bug will be found. But for now, version 0.2 of the code is designed to meet the bare specification of the lab.

With all the progress of the Lab, One of the highlights was to re-engineer all the code in a improved fashion but was never finished. Another highlight was the recreating of the keyboard and ascii controls for the VGA displayer interface.

With the daunting time constraints of the lab getting smaller and smaller. I noticed how we must be on top of each lab in order to properly finished it and move to the next lab. This semester feels so short and with the short timeline for each of the labs, The scary part is that March 7th is half way of the semester. This means that have of the semester has already went by in a flash.

This lab allowed us to dive deeper in VHDL code and helped us understand what could be done in the FPGA world. Apart from the PS2 protocol, we understood the VGA video interface. Yes, now we can type through a key board and display the output via VGA monitor! Also, the user can have the some control over the display by controlling the cursor via Enter, Up, Down, Left and Right keys. This lab added a lot to our knowledge by understanding the design needed to display on VGA monitor. The design involved, a memory that acts RAM where the characters behind each position on the VGA are stored. It also involved 'Front ROM' that represent ROM where the pixel by pixel representation of each ASCII character are stored. Moreover, this process involves timing consideration.

For this lab, we were given most of the code to get this lab done. With limited time we were given, we did well by understanding the operation of the VGA protocol, understand the functionality of the given code, interfacing it with PS2 controller, and adding features into the keyboard-cursor controls. On the other hand, what didn't well is understanding the cause of certain behavior. For example, the interface crashed after uncertain number of inputs where it doesn't accept on taking any key board input. Moreover, we noticed some timing issues where key pressed gets displayed pressed, This needs to debugged by outputting the key code and the data that the VGA controller is using to display the characters.

ECE 368
Digital Design
Spring 2014

Lab 3 – VGA Video and PS2 Keyboard Interface

Design Layout Section

VGA Toplevel – VGA_Structural

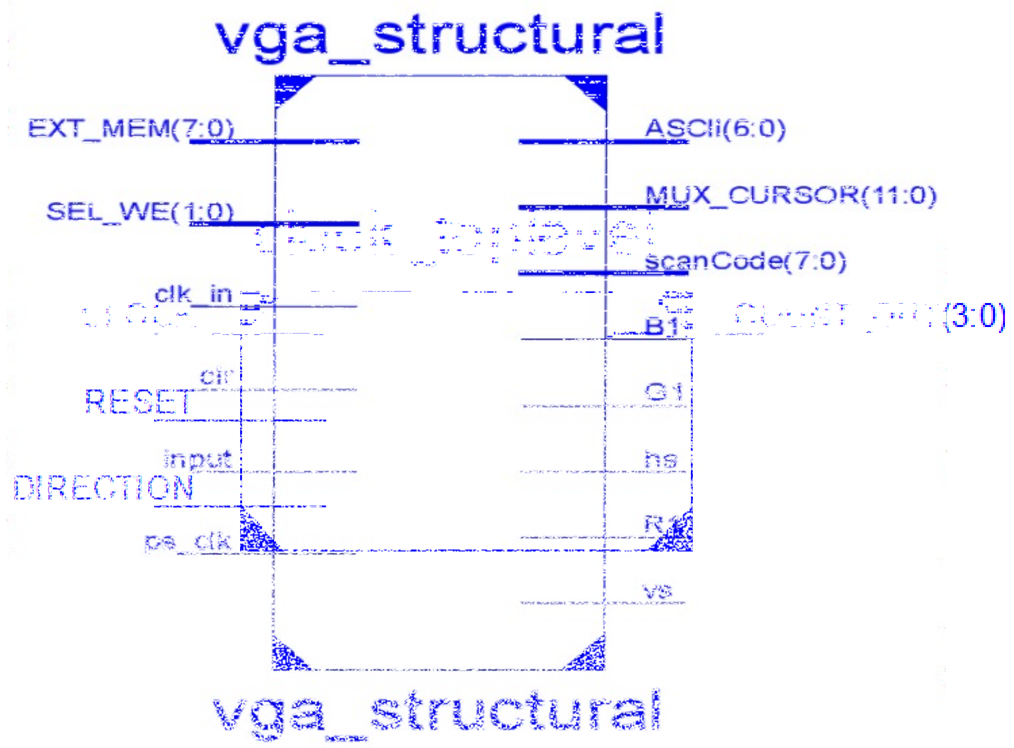
Inputs:

Clk_in, ps_clk, input, clr <STD_LOGIC>,
EXT_MEM <STD_LOGIC_VECTOR (7 downto 0) >,
SEL_WE <STD_LOGIC_VECTOR (1 downto 0) >,

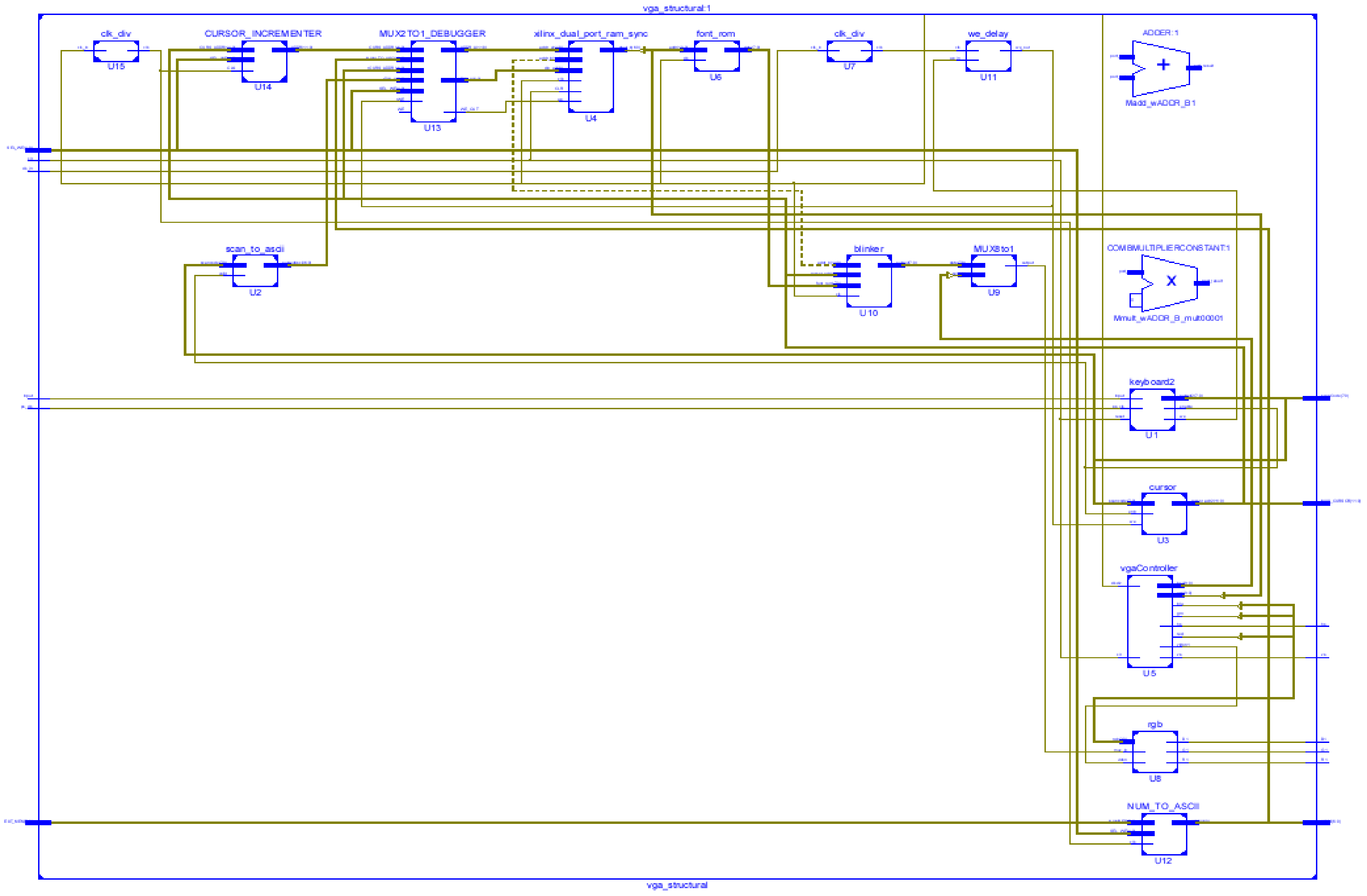
Outputs:

B1, R1, G1, hs, vs <STD_LOGIC>,
ScanCode <STD_LOGIC_VECTOR (7 downto 0) >,
Mux_Cursor <STD_LOGIC_VECTOR (3 downto 0) >,
ASCII <STD_LOGIC_VECTOR (6 downto 0) >

Functionality: This is the top level for the VGA that interfaces the the VGA controller with the PS2 controller.



VGA Toplevel – VGA_Structural

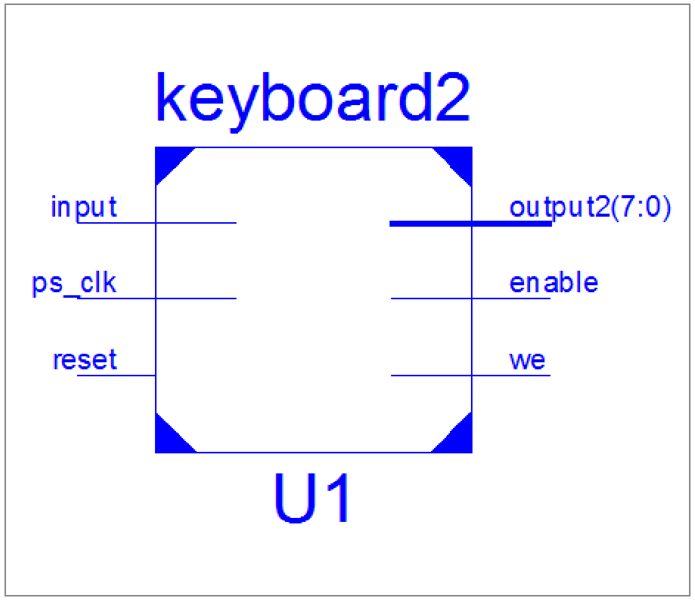


Keyboard Controller: U1 – keyboard2

Inputs:
Input, ps_clk,reset <STD_LOGIC> ,

Outputs:
output<STD_LOGIC_VECTOR (7 downto 0) >,
we, enable <STD_LOGIC>

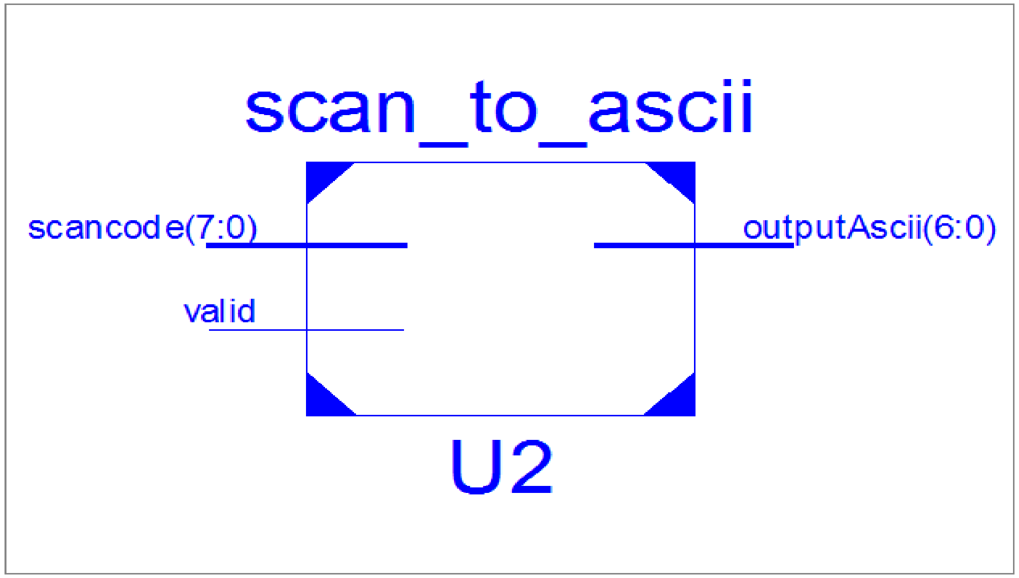
Functionality: It will convert ps2 protocol keyboard input the system and output a keycode value. It output a signal that indicates the write enable.



U2 – scan_to_ascii

Inputs:
scanCode <STD_LOGIC_VECTOR (7 downto 0)>
Valid <STD_LOGIC>
Outputs:
outputAscii <STD_LOGIC_VECTOR (6 downto 0)>

Functionality: This converts a keycode into an ASCII code, It considers shift and arrow keys.



U3 -Cursor

Inputs:

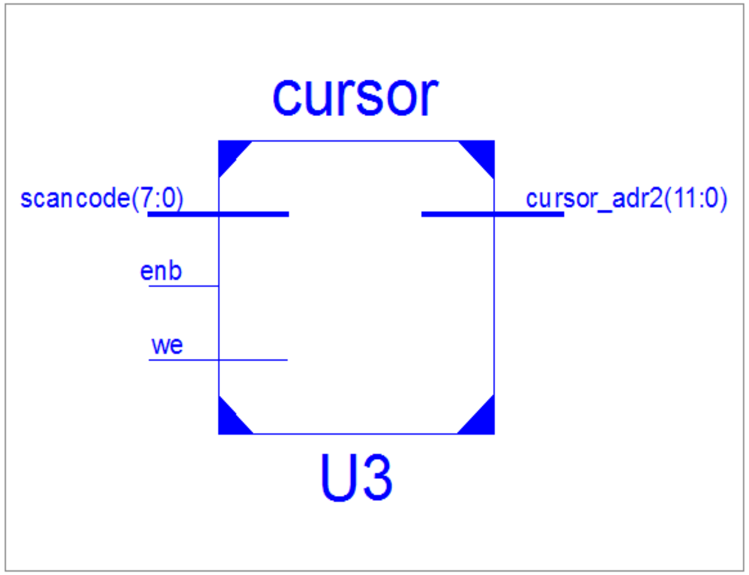
enb, we <STD_LOGIC>

scancode <STD_LOGIC_VECTOR (7 downto 0)>

Outputs:

cursor <STD_LOGIC_VECTOR (11 downto 0)>

Functionality: Based on the write enable input the cursor get shifted to a specific address. This address is the output, The scancode is needed to know when to shift and to consider arrow keys and any other special keys.



U4- xilinx_dual_port_ram

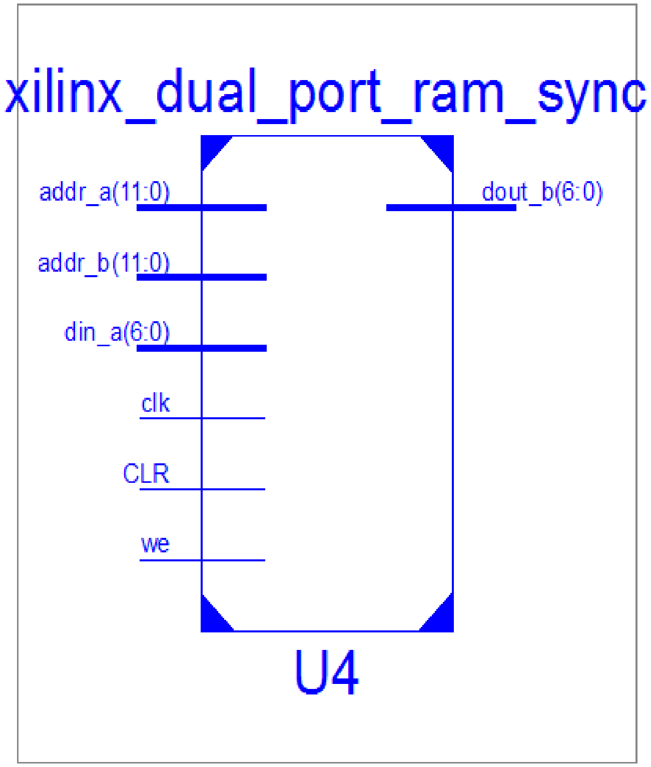
Inputs:

addr_a <STD_LOGIC_VECTOR (11 downto 0) >
addr_b <STD_LOGIC_VECTOR (11 downto 0) >
din_a <STD_LOGIC_VECTOR (6 downto 0) >
clk, CLR, we <STD_LOGIC **>**

Outputs:

dout <STD_LOGIC_VECTOR (6 downto 0) >
RA, RB <STD_LOGIC_VECTOR (23 downto 0) >

Functionality: . This component acts like a memory where the data gets stored at a specific address.

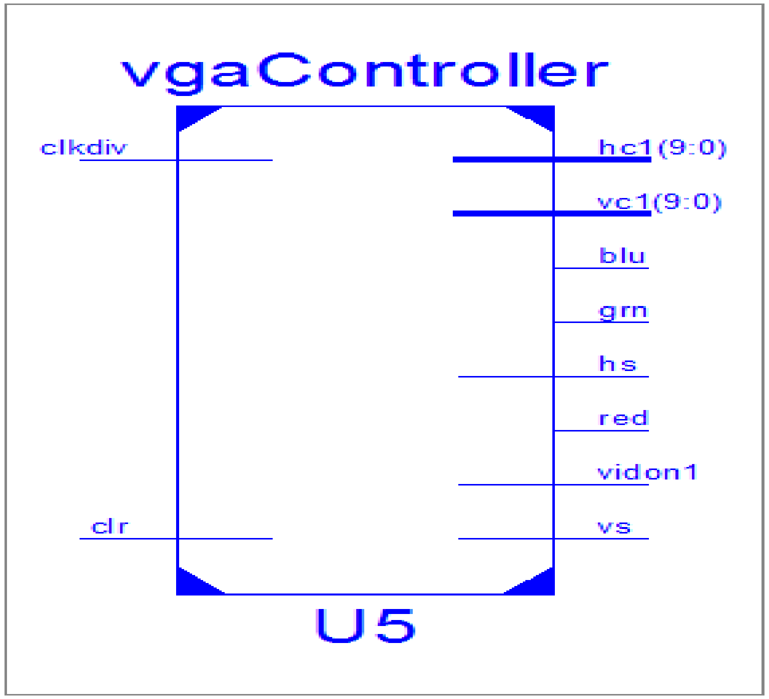


U5- VGA Controller

Inputs:
Clk, clr <STD_VECTOR>

Outputs:
Blu, Red, Grn, hs, vs <STD_LOGIC>,
hc1,vc1 <STD_LOGIC_VECTOR (9 downto 0) >.

Functionality: This component control the communication to the VGA monitor. This output the pixel by pixel data.

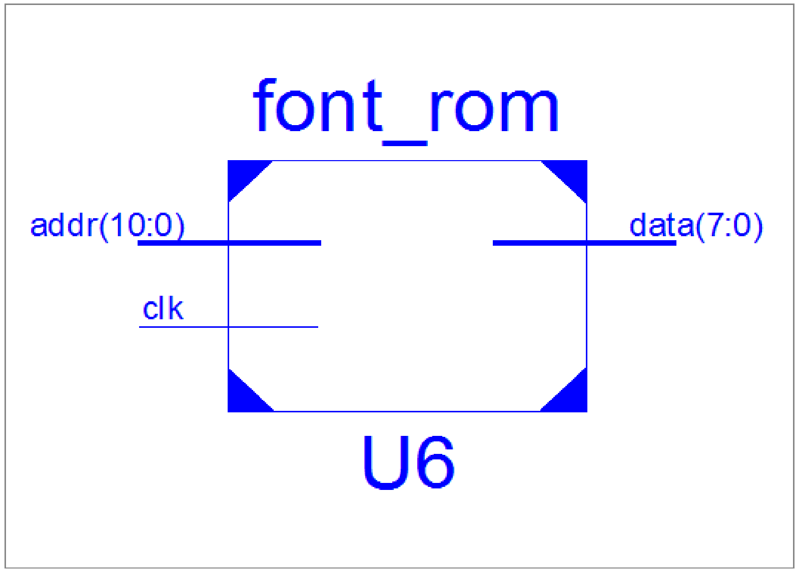


U6 - Front Rom

Inputs:
addr <STD_LOGIC_VECTOR (10downto 0) >
Clk <STD_LOGIC>

Outputs:
data <STD_LOGIC_VECTOR (7 downto 0) >

Functionality: This component acts like ROM where all the pixel by pixel information for ASCII character are stored.



U8 – rgb

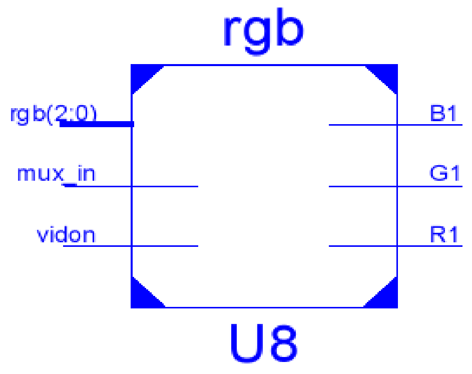
Inputs:

rgb STD_LOGIC_VECTOR (2 downto 0),
mux_in STD_LOGIC,
vidon STD_LOGIC

Outputs:

R1 STD_LOGIC,
G1 STD_LOGIC,
B1 STD_LOGIC

Functionality: This component will output the RGB color spectrum to the monitor. The values will only be outputing when the vidon switch is an active low. If the vidon is high then the mux_in is inputed through to the RGB output.

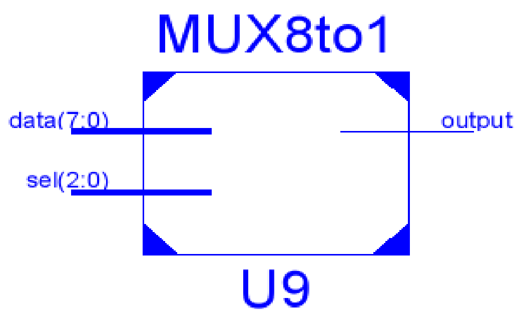


U9 – MUX8to1

Inputs:
    sel STD_LOGIC_VECTOR (2 downto 0),
    data STD_LOGIC_VECTOR (7 downto 0)

Outputs:
    output STD_LOGIC

Functionality: This component will output one of 8 outputs from the data signal. The output is selected based on the 3 bit(8 positions) select input line.



U10 – blinker

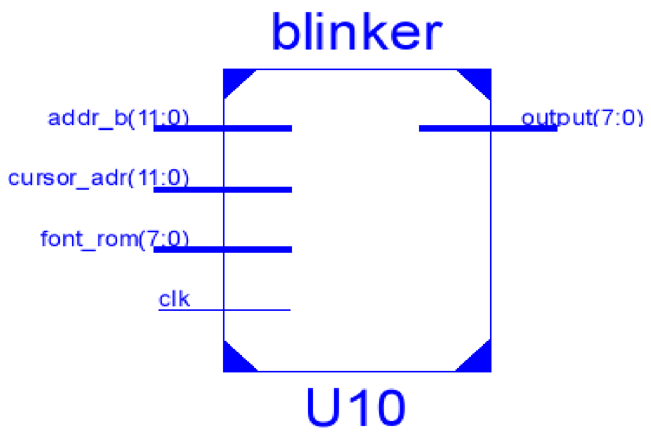
Inputs:

addr_b STD_LOGIC_VECTOR (11 downto 0),
cursor_adr STD_LOGIC_VECTOR (11 downto 0),
clk STD_LOGIC,
font_rom : STD_LOGIC_VECTOR (7 downto 0)

Outputs:

output STD_LOGIC_VECTOR (7 downto 0)

Functionality: This component will simulate a flashing character to represent a blinking cursor on the screen.

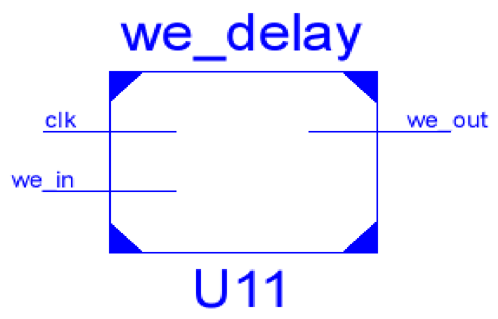


U11 – we_delay

Inputs:
we_in STD_LOGIC,
clk STD_LOGIC

Outputs:
we_out STD_LOGIC

Functionality: This device will output a delay for write enable.



U12 – NUM_TO_ASCII

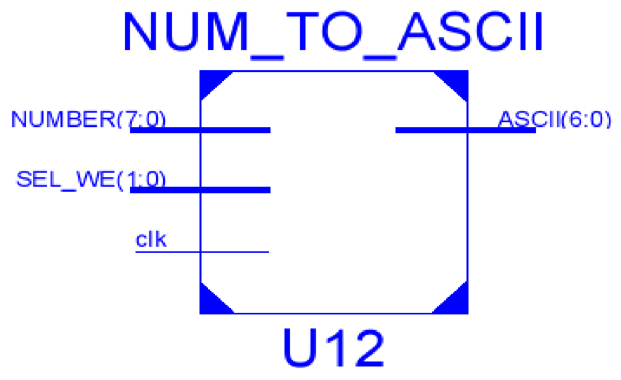
Inputs:

clk STD_LOGIC,
SEL_WE STD_LOGIC_VECTOR (1 downto 0),
NUMBER STD_LOGIC_VECTOR (7 downto 0)

Outputs:

ASCII STD_LOGIC_VECTOR (6 downto 0)

Functionality: This component will convert a given number into its ASCII represenation on the screen.



U13 – MUX2TO1_DEBUGGER

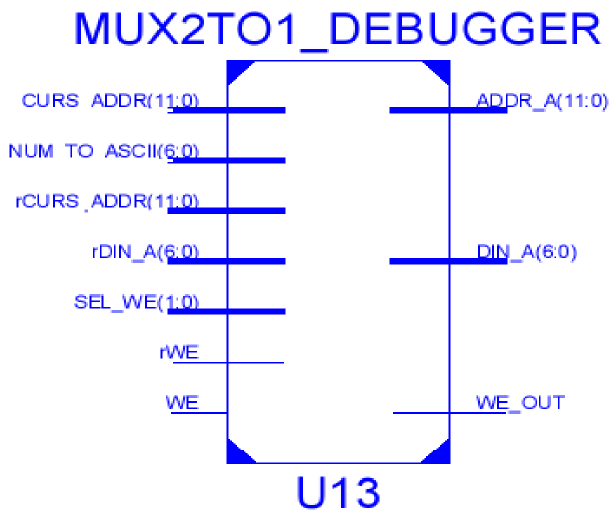
Inputs:

WE STD_LOGIC,
rWE STD_LOGIC,
SEL_WE STD_LOGIC_VECTOR (1 downto 0),
NUM_TO_ASCII STD_LOGIC_VECTOR (6 downto 0),
rDIN_A STD_LOGIC_VECTOR (6 downto 0),
CURS_ADDR STD_LOGIC_VECTOR (11 downto 0),
rCURS_ADDR STD_LOGIC_VECTOR (11 downto 0)

Outputs:

WE_OUT STD_LOGIC,
DIN_A STD_LOGIC_VECTOR (6 downto 0),
ADDR_A STD_LOGIC_VECTOR (11 downto 0)

Functionality: The MUX2TO1 is a selector switch that will output a address, data, and write enable from two sources.



U14 – CURSOR_INCREMENTER

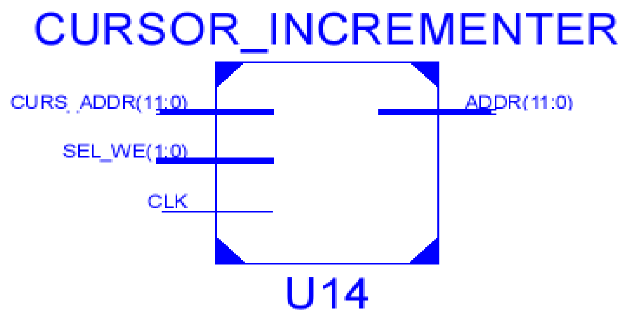
Inputs:

CURS_ADDR STD_LOGIC_VECTOR (11 downto 0),
SEL_WE STD_LOGIC_VECTOR (1 downto 0),
CLK STD_LOGIC

Outputs:

ADDR STD_LOGIC_VECTOR (11 downto 0)

Functionality: This comonent will auto increment the position of the cursor to where to write data to the memory which will display to the screen.

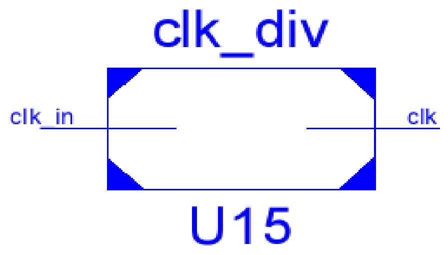


U15 – clk_div

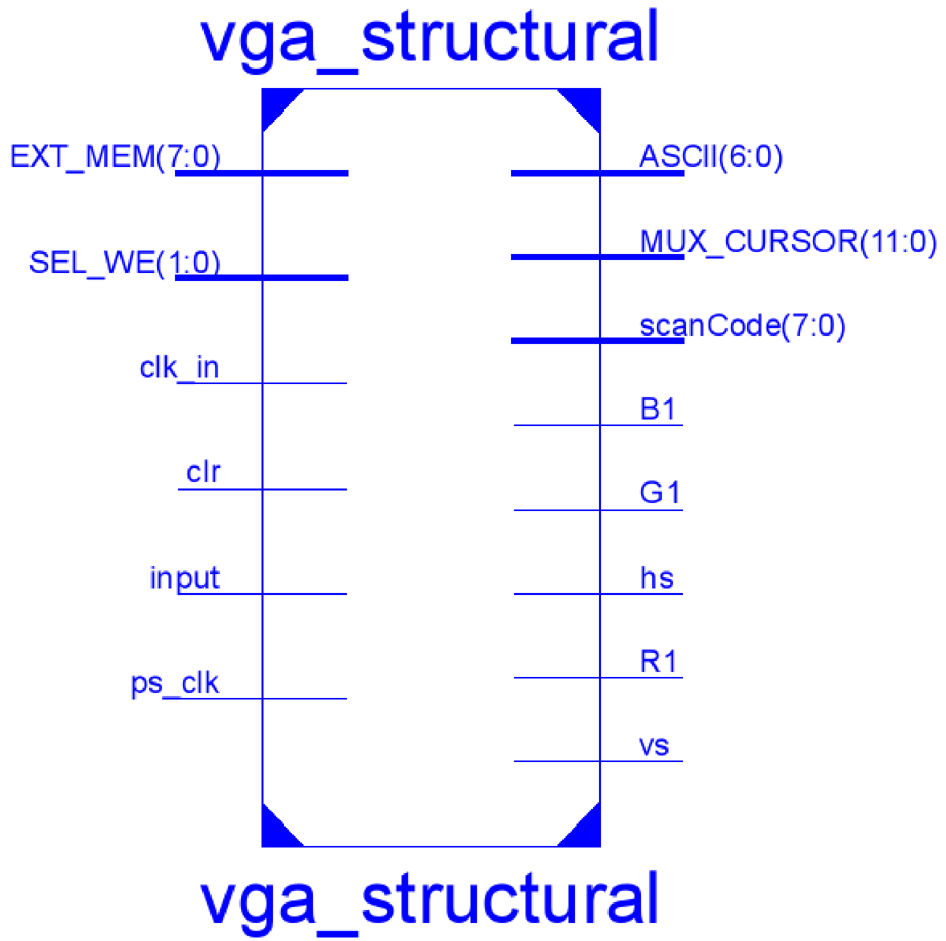
Inputs:
clk_in STD_LOGIC

Outputs:
clk STD_LOGIC

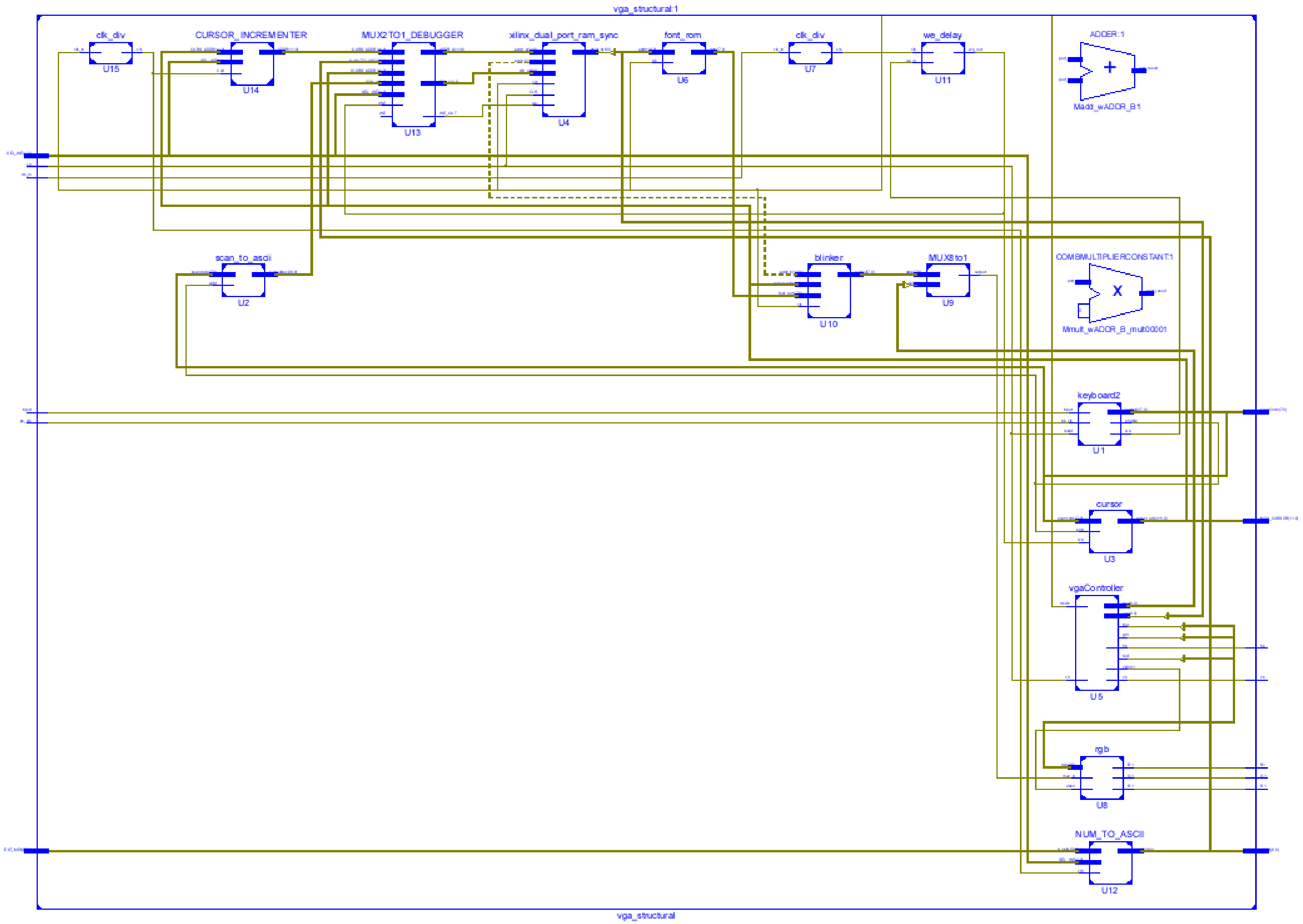
Functionality: divide the clock in half



VGA – vga_structural



VGA – vga_structural Indept



ECE 368
Digital Design
Spring 2014

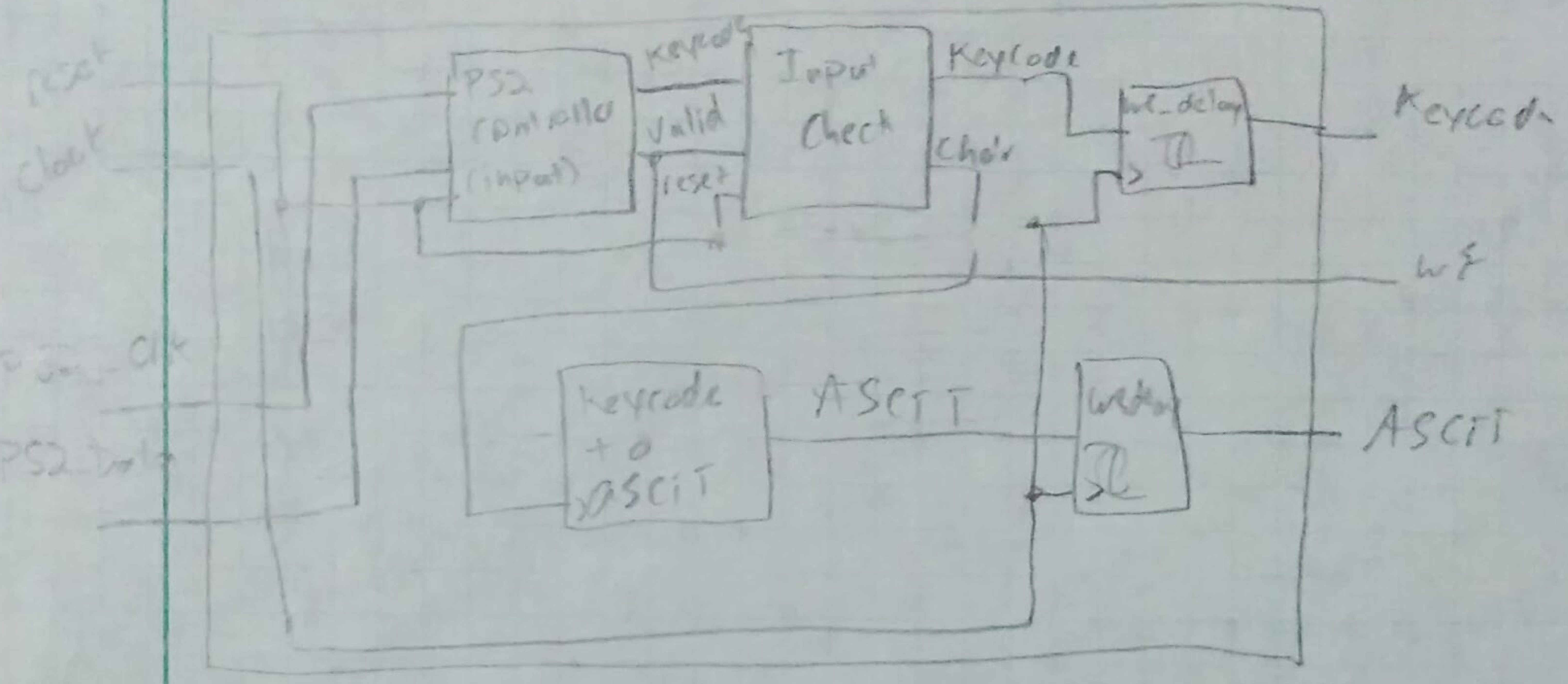
Lab 3 – VGA Video and PS2 Keyboard Interface

Original Design
Section

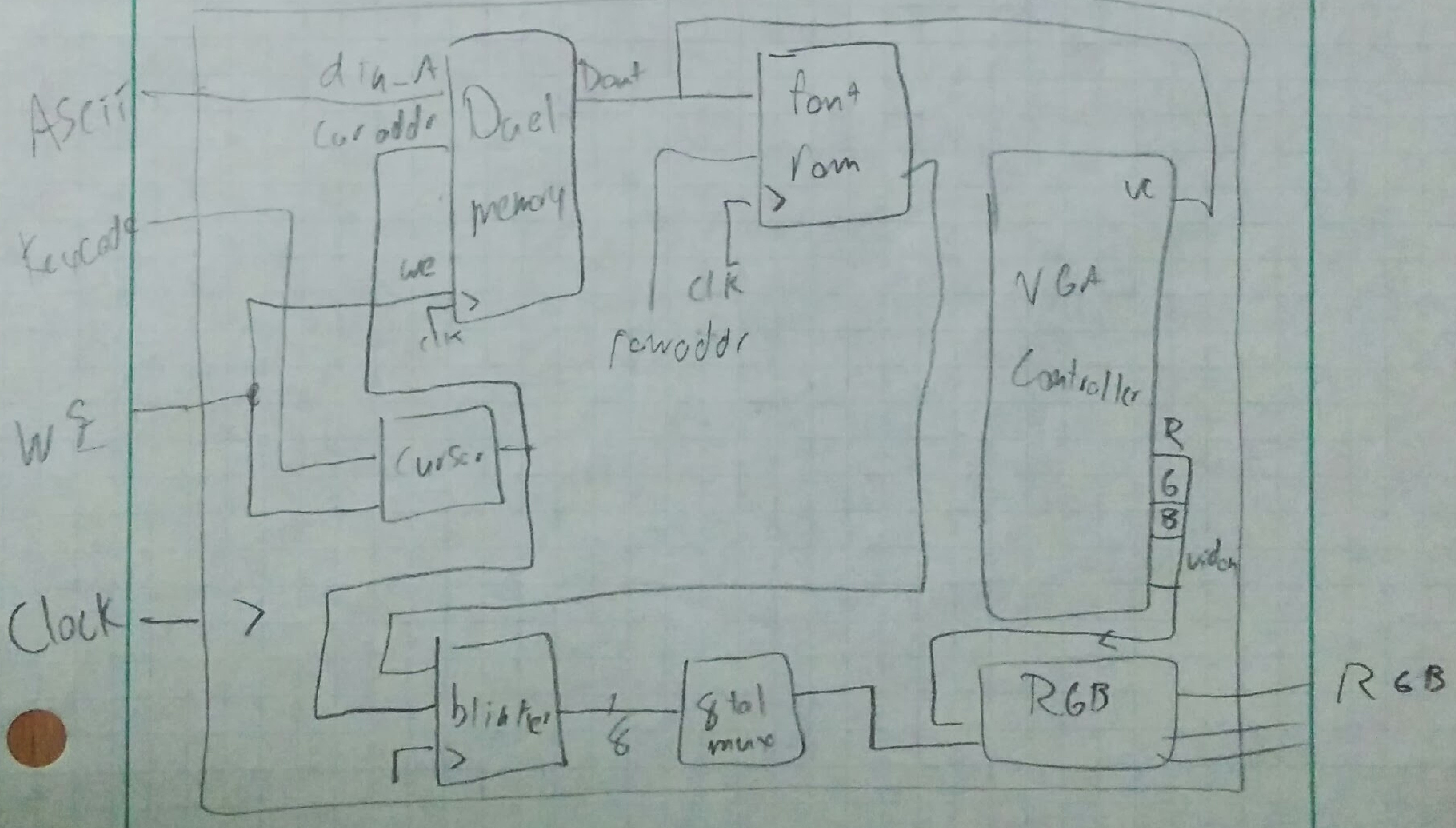
Original Design

* Dr. Fortier Provided the design (RTL) layout for this Lab.
* Separate to two Drivers

PS2_Controller (v1.0.5)



VGA_Driver


ASCII
Keycode
WE
Clock
din_A
Curaddr
Dual memory
we
clk
Dout
font rom
clk
rowaddr
VGA Controller
vc
R
G
B
video
Cursor
blinker
8
8 to 1 mux
RGB
RGB