RISHIK SAHA-2022CS11932| PARTH VERMA-2022CS11936

COL215-HARDWARE_2

REPORT

PART-1

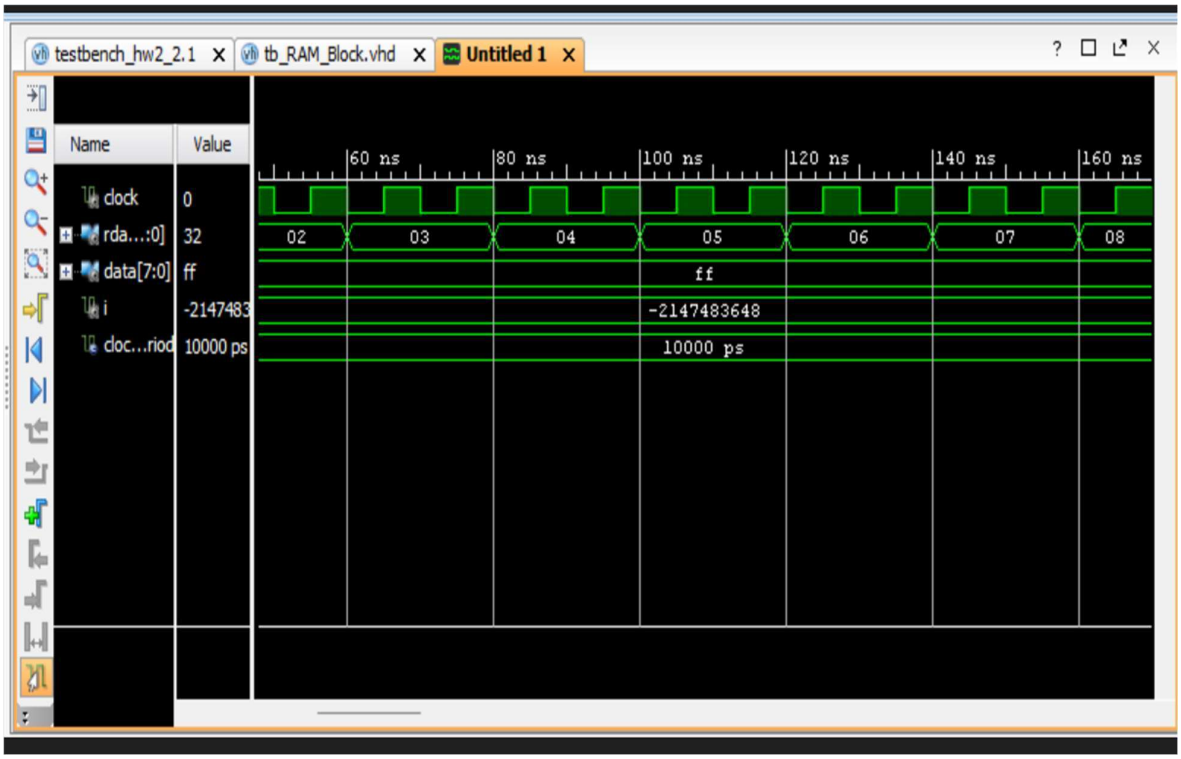
AIM

To generate RAM/ROM using Vivado's memory generator

and to populate it with the input coe file

LAB_WORK AND DESIGN DECISIONS

- Using Vivado's IP Catalog (under Memory and storage elements) ROM was generated.
- The dimensions of ROM were set in accordance with the Coe input file. Depth was set to 65536 and the data width was set to 8 bits. Clock was included, as a port. Under the coefficients file the address of Coe file was mentioned.
- A testbench was created to instantiate the generated ROM and data was represented on the simulation.
- Finally, the data shown on the simulation was crosschecked with the data in the unput Coe file.


testbench_hw2_2.1
tb_RAM_Block.vhd
Untitled 1
Name
Value
clock
0
rda...:0]
32
data[7:0]
ff
i
-2147483
doc...riod
10000 ps
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
02
03
04
05
06
07
08
ff
-2147483648
10000 ps

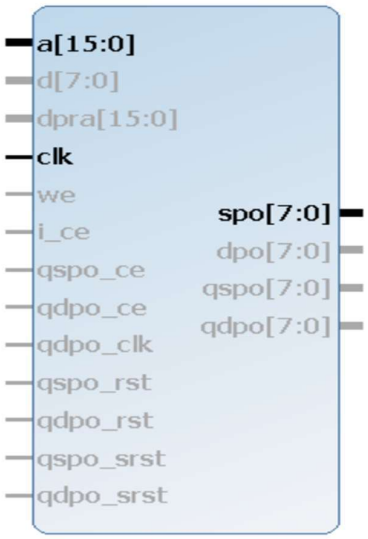
#PART-2

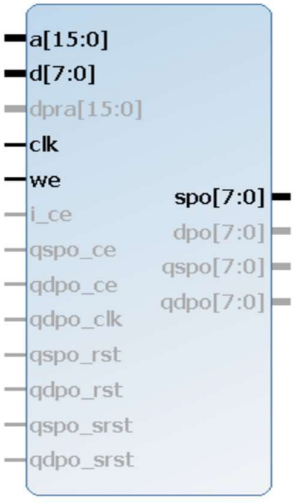
AIM

To carry out image gradient operations using the generated memory

LAB_WORK AND DESIGN DECISIONS

- Using Vivado's IP Catalog under Memory and Storage elements RAM of appropriate dimensions was generated.
- A single Port RAM was generated. Depth of RAM was set to 65536 and data width to 8. The ROM generated in part 1 to read the data was used again.





- A “main.vhd” file was created in which the generated ROM and RAM were instantiated. They were correspondingly Port mapped to their signals.
- A logic for the gradient operations was written which involved iterating through the Coe file with the help of counter signals and correspondingly applying the logic filer.
- This was done using registers for storing 3 input pixels simultaneously in a clock cycle and there after applying the 3X1 filter.
- Simultaneously the output pixels generated after applying the 3X1 filter were stored in the RAM.
- The stored values in the RAM were also read from RAM using the output signal (spo).
- Finally, to observe this process a testbench was generated in which all the signals corresponding to (ram_address, rom_address, ram_data, rom_data)

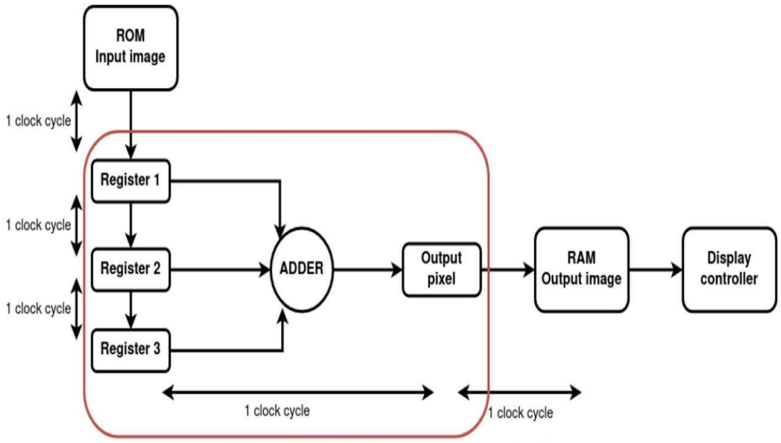


Figure 4: Block diagram component-wise

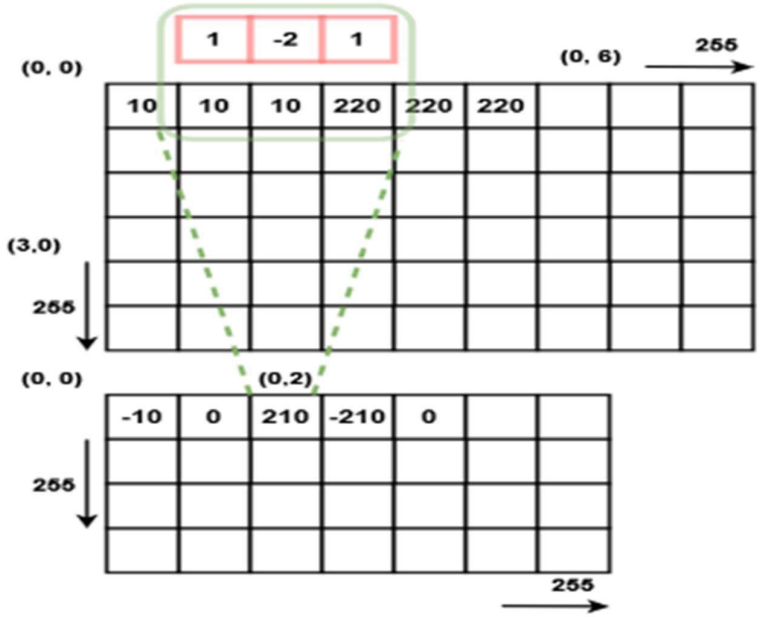
LOGIC FOR GRADIENT OPERATION

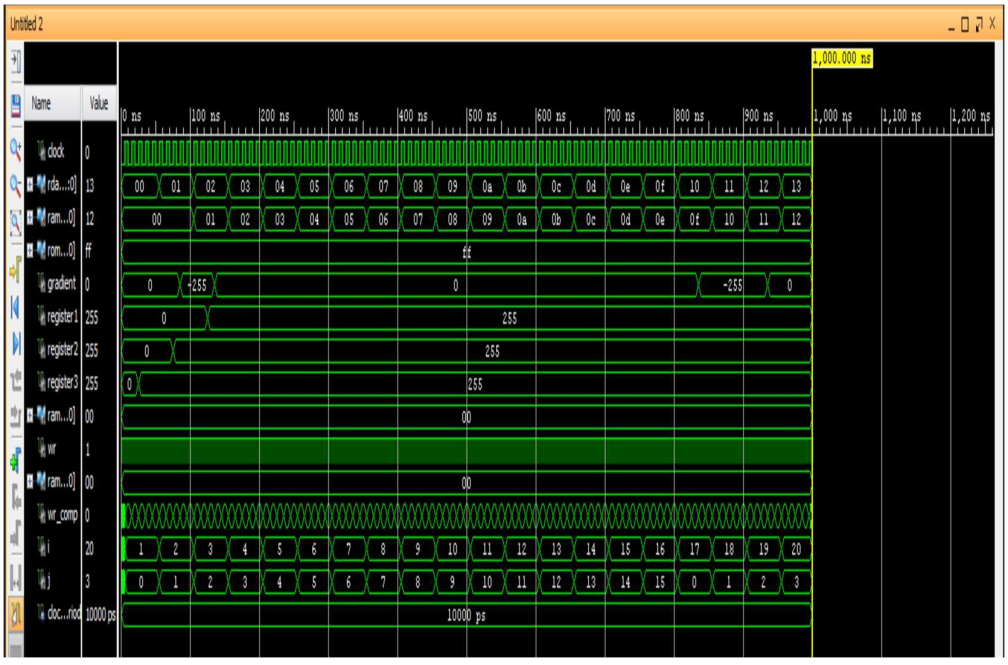
- Basically, we create 2 counters “I”,” j” instantiating them with 0 and -1 respectively.
- We then start a process giving Clk_100MHz as a signal in its sensitivity list.
- We also use a counter “wr_comp” which is helpful in maintaining the required clock cycles gap between different operations.
- ‘I’ maintains the address no. ranging from (0 to 65535) while ‘j’ maintains the column no ranging from (0 to 255).
- Further we apply a 3X1 filter to the input pixels by maintaining 3 neighboring pixels values in 3 different registers.
- This is done by first assigning the value stored in register2 to register 1, the value stored in register 3 to register 2 and finally assigning the Rom data to register 3. And then. We compute the gradient as per the following rule:

$$O(i,j) = 1 * I(i,j-1) - 2 * I(i,j) + 1 * I(i,j+1)$$

Here O (I, j) is the output pixel, while I (I, j) is the input pixel stored at location (I,j).

- Further the if the gradient comes negative, we set it to 0 and if it comes > 255 we set it to 255.
- Finally, we store the value of gradient obtained in RAM.





#PART-3

AIM

To display the image stored in the memory before and after the image gradient operation.

LAB_WORK AND DESIGN DECISIONS

- Firstly, a display controller module was constructed.
- The Display Controller Module involved formation of a clock divider, horizontal counter, a vertical counter, and a video on indicator.
- The clock divider was developed to make the frequency of clock 25 MHz which was necessary frequency for the pixels to get displayed on CRT Monitor.

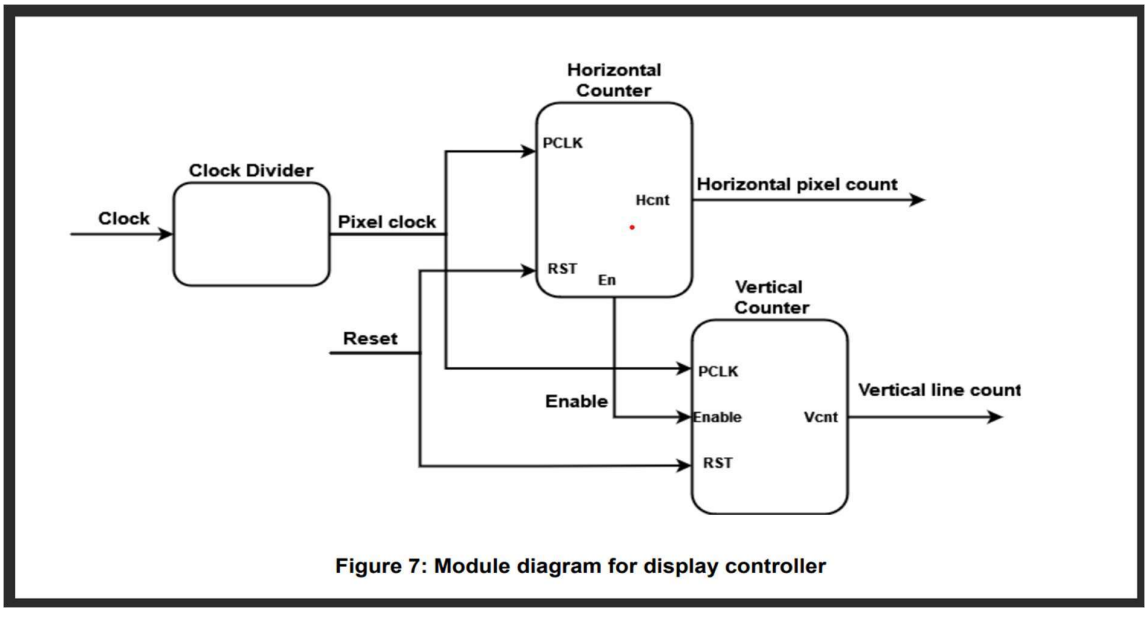
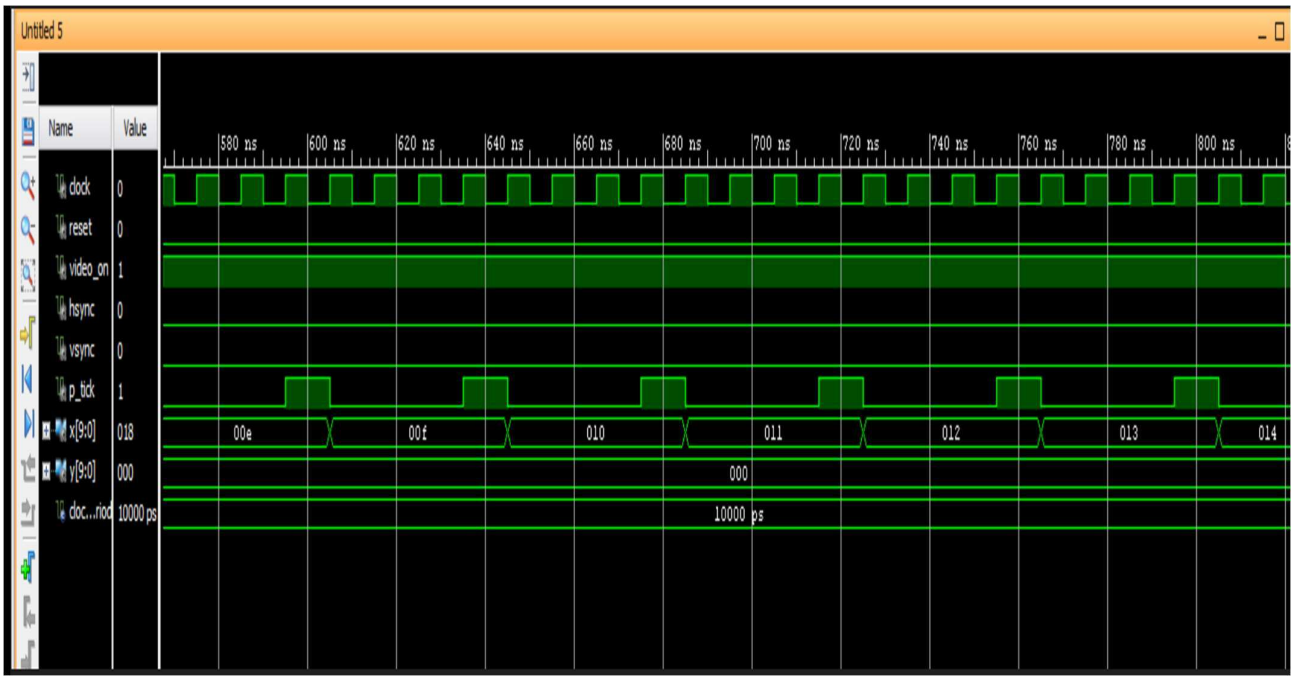


Figure 7: Module diagram for display controller

- A horizontal Counter was made which iterated through the Rows in the output filtered image
- A vertical Counter was made which iterated through the columns in the output filtered image.
- Finally depending on the values of the horizontal and vertical counter, Hsync, Vsync and Video_on signal were set
- Further to test the Display Controller Module another vhd file was constructed which takes 4-bit input rgb value from the BASYS FPGA Board and displays the corresponding pixel on the VGA Monitor.
- A Top-level entity was developed which reads the pixels values stored in the RAM and passes it to Display Controller Module to display the final resultant image on the VGA Monitor.



STRUCTURE OF VHDL CODE

- At the very first we have created a vhd file named “Clock_div” which is basically responsible for generating 25 Mhz clock from the standard 100MHz clock available on the Basys FPGA Board.

- Further a file named vga_controller is created which is responsible for getting pixel values as input and displaying them on the VGA Monitor. The module involves various processes like horizontal counter, vertical counter, register control.

- Further we constructed a fie named “vga_test” which serves as a top-level Module. The module instantiates ROM and RAM in it. It contains the process of taking in the input pixels from the ROM filtering them and storing them in RAM. Further the module contains instantiation of vga_controller and takes in then filtered pixel values from RAM and display them to the VGA Monitor.

- We have created testbenches explicitly for the ROM to RAM transfer process to check the correctness of logic

- A testbench for vga_controller named “tb_vga_controller” is created to ensure the correctness of the Display Controller Module.

SYNTHESIS REPORT

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	240	0	0	20800	1.15
LUT as Logic	240	0	0	20800	1.15
LUT as Memory	0	0	0	9600	0.00
Slice Registers	232	0	0	41600	0.56
Register as Flip Flop	232	0	0	41600	0.56
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
0	Yes	-	Set
64	Yes	-	Reset
31	Yes	Set	-
137	Yes	Reset	-

2. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	50	0.00
RAMB36/FIFO*	0	0	0	50	0.00
RAMB18	0	0	0	100	0.00

3. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	0	0	0	90	0.00

4. IO and GT Specific

Site Type	Used	Fixed	Prohibited	Available	Util%
Bonded IOB	16	0	0	106	15.09
Bonded IPADs	0	0	0	10	0.00
Bonded OPADs	0	0	0	4	0.00
PHY_CONTROL	0	0	0	5	0.00
PHASER_REF	0	0	0	5	0.00
OUT_FIFO	0	0	0	20	0.00
IN_FIFO	0	0	0	20	0.00
IDELAYCTRL	0	0	0	5	0.00
IBUFDS	0	0	0	104	0.00
GTPE2_CHANNEL	0	0	0	2	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	0	20	0.00
PHASER_IN/PHASER_IN_PHY	0	0	0	20	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	0	250	0.00
IBUFDS_GTE2	0	0	0	2	0.00
ILOGIC	0	0	0	106	0.00
OLOGIC	0	0	0	106	0.00

5. Clocking

Site Type	Used	Fixed	Prohibited	Available	Util%
BUFGCTRL	1	0	0	32	3.13
BUFIO	0	0	0	20	0.00
MMCME2_ADV	0	0	0	5	0.00
PLLE2_ADV	0	0	0	5	0.00
BUFMRCE	0	0	0	10	0.00
BUFHCE	0	0	0	72	0.00
BUFR	0	0	0	20	0.00

6. Specific Feature

Site Type	Used	Fixed	Prohibited	Available	Util%
BSCANE2	0	0	0	4	0.00
CAPTUREE2	0	0	0	1	0.00
DNA_PORT	0	0	0	1	0.00
EFUSE_USR	0	0	0	1	0.00
FRAME_ECCE2	0	0	0	1	0.00
ICAPE2	0	0	0	2	0.00
PCIE_2_1	0	0	0	1	0.00
STARTUPE2	0	0	0	1	0.00
XADC	0	0	0	1	0.00

7. Primitives

Ref Name	Used	Functional Category
LUT2	185	LUT
FDRE	137	Flop & Latch
FDCE	64	Flop & Latch
CARRY4	56	CarryLogic
LUT3	55	LUT
LUT6	31	LUT
FDSE	31	Flop & Latch
LUT4	28	LUT
LUT1	22	LUT
LUT5	19	LUT
OBUF	14	IO
IBUF	2	IO
BUFG	1	Clock

8. Black Boxes

Ref Name	Used
dist_mem_gen_1	1
dist_mem_gen_0	1

9. Instantiated Netlists

Ref Name	Used

TESTING IMAGES