International Journal of Engineering Research and General Science Volume 2, Issue 4, June-July, 2014
ISSN 2091-2730

Ram Control Block of Vector Display Processor

N. Agarwala[1]

[1]Lecturer, Department of EEE, School of science and Engineering, Southeast University, Dhaka, Bangladesh

Abstract —Vector Display Processor allows efficient encoding of topology, and as a result more efficient operations that require topological information can be done, e.g. proximity, network analysis. [1] In this work, a Ram control block has been designed which is a part of Vector Display Processor and the block is synthesized with the test bench and the generating simulation waveform was checked for the inputs and outputs.

Keywords —VDP (Vector Display Processor); Ram Control Block; Draw Block.

1. Introduction

Vector Display Processor consists of two blocks. One is Draw Block and another one is RAM Control Block. The RAM Control Block is the interface between the Draw Block and VRAM. At this display, data can be represented at its original resolution and form without generalization. Graphic output is usually more aesthetically pleasing; since, most data, e.g. hard copy maps, is in vector form no data conversion is required and accurate geographic location of data is maintained.[1] Synthesis is the most important design issues which I think everybody should remember while doing the VHDL hardware. The code must be synthesizable. The top level (Vector Display Processor) design integrates several parts and together they behave as a single device. RAM Control Block is one of the parts of VDP where there are many different modules which behave as a single block.

2. Methods

2.1 Design Description:

This is the most important part of my work as all the processes are based on this section. In this part of my project, several things was taking into account, for example, specification, Finite State Machine (FSM), Code description, the Waveform and so on.

2.1.1 Specifications:

Inputs:

- X and Y: These are the 6 bit vectors from which the first 4 bits of each of them are used for the current word address and the other 2 bits are for checking the bit number where to update the data.
- Pen: Pen has 2 bits to give the color which are listed below:

Input	Color
00	Black
01	White
10	Invert
11	Illegal

- Draw pixel: It is single bit input which comes from the Draw Block and informs RAM Control Block for updating the data from the pen inputs and performs the operations of drawing pixels.
- Flush: Flush is a 1 bit input which is used to write the data to RAM from both the data_reg and (stored_ram_word) and address_reg(current_word_addr).
- vdout: This is a 16bit input which comes from the VRAM and it stores the date in the data_reg.

Outputs:

- ack: This is a 1 bit output which tells the Draw Block whether the RAM Control Block is free or not.
- vaddr: This is a 7 bit output from address_reg or x,y which goes to VRAM .

- Vdin: This is a 16 bit output which goes to the VRAM.
- Vwrite: This is the output which goes to the VRAM.

Registers:

- Address_reg (Current_word_addr): This is 8 bit register where the first 4 bit of both the inputs (x,y) come and stores current word address.
- Data_reg (stored_ram_word): This is 16 bit register which stores the data according to the 16 bit pixels.

2.1.2 The Finite State machine:

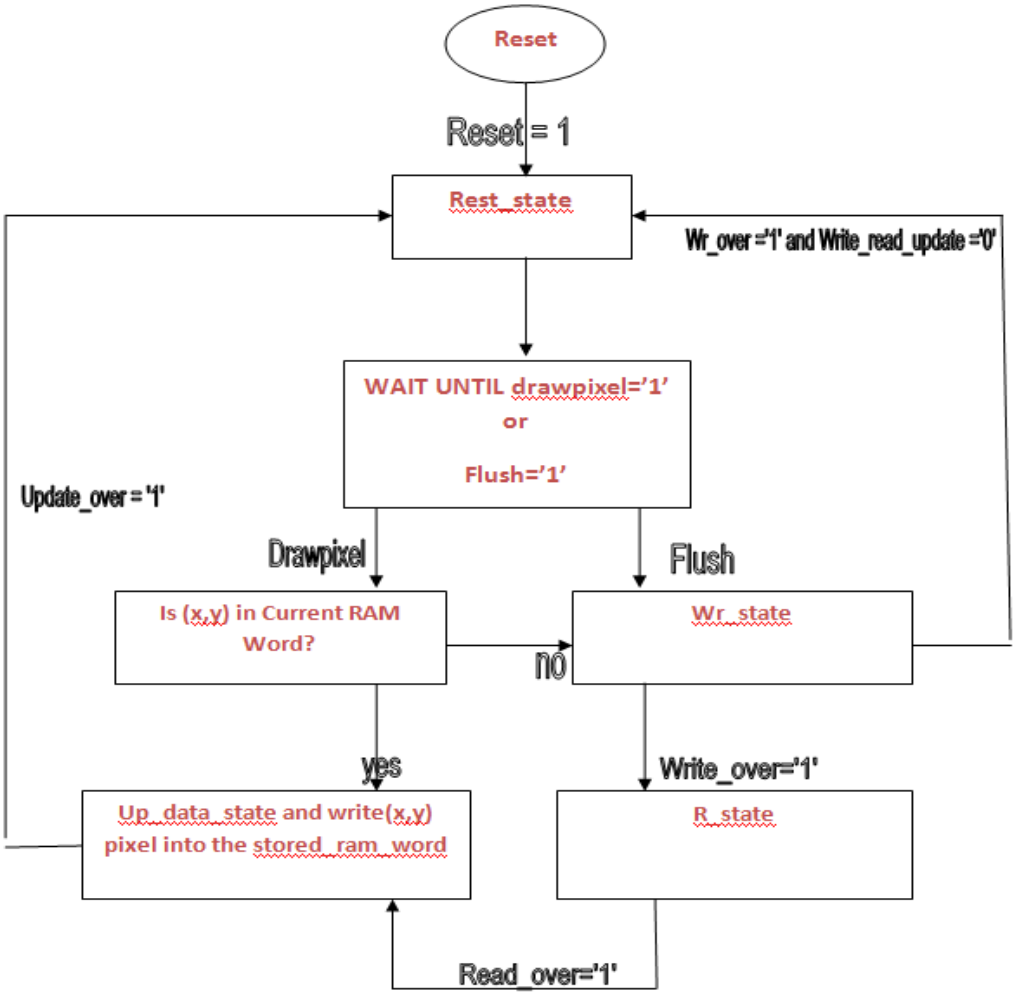

Fig 1: Flow Chart of FSM

The Finite State machine consists of four states which are described below:

- rest_state: This is the idle state in which state RAM Control Block has nothing to work. In this state the Reset = 1, meaning that it will reset all the previous values and also ack = '1' which means that now the RAM Control Block is ready to accept the data from the Draw Block. In the rest state two commands might come, one is Drawpixel = '1' or Flush = '1'. If it will get

the drawpixel =' 1' then it will check one more condition, whether (x,y) = Current RAM Word or not. If the answer is 'yes' then it will directly go to up_data_state and if 'no' then it will go to Wr_state. In the case of Flush, if the command is Flush = '1' then it will immediately go to the write state.

- Wr_state: This is the next state after rest which is used to write data to the RAM. The system can be in Wr-state because of two commands. One when Flush = '1' and another when (x,y) is not same as Current RAM Word. In the case of Flush, it will just flush the present data to the two registers. In another case, it will take the data from the two registers and write the data to the RAM. After performing all the operation if the value of write_over = '1' and write_read_update = '0' then it will go back to the idle state and update the ack = '1'meeaning that RAM Control Block has performs all the operations and waiting for the next operation to do from the Draw Block.

- R_state: This is the next state after the Wr_state when it will get the command like Write_over = '1'. At this state the only work is to read the Data from the RAM. After this state the data read from the Data neeede to be updated. So, whenever the R_state will give a value of '1' it will go to the up_data_state to update the data.

- Update_data_state: This is the nest state after the R_state. After updating the data it will go to the rest state.

2.1.3 Description of Code:

The code consists of one Entity and an Architecture

- ENTITY RAM_CONTROL_BLOCK: In this Entity all the input and output ports are define. Some are for the interfacing with the Draw Block and some are for VRAM.

- ARCHITECTURE RAM_CONTROL_BLOCK_BEHAVIOUR: This Architecture consists of FIVE processes which are described below:

- PSTATE_PROC: This is a short process which depends on the clock. When the value of clock will be '1'it will go to next_state from the present_state. When the reset will be '1' it will make the next_state as the rest_state.

- TIME: In this process all the necessary variables and constants are declared as integer and some of them assigned the fixed values. Here, different formulas are used to calculate the time to get the frequency measurements. Here, the operation and the number of clock cycles for both the read and write has been done for all fours states.

- STATE_TRANSITION: This is process is all about the transition of the states and depends on the value of data_bit_signal which is the combination of drawpixel and flush. If the data_but_gis ="00" it will be in the rest_state. If the data_bit_sig ="01", it will go to the wr_state. If the data_bit_sig ="10", then it will check whether the word_sig = addr_reg. If the value is same then it will go to the up_data_state otherwise it will go to the wr_state. After performing the operation in the wr_state it wil go to either the rest_stae (if all other operations is finished) or will go to the r_state. In the r_state, it will read the data and after that it will go to the up_data_state and after updating the data it will go to the rest_state.

- DATA_RAM_WORD: In this process it always update the value of data_reg depending on the value of x1 which is being calculated every state.

- ADDRESS_REG: In this process for loop is used to get the value of word_sig. In this process the value of vaddr is updated with the value of data_reg.

3. Results:

3.1.1 Waveform:

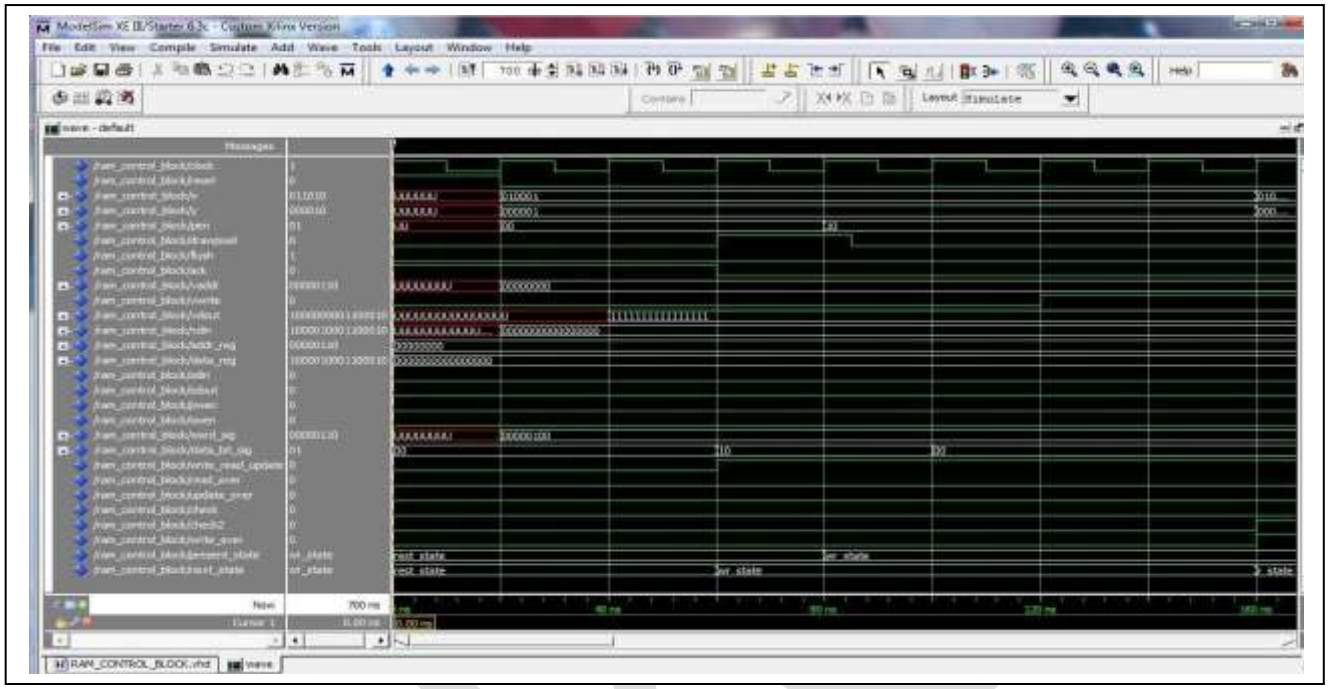

Fig 2: Waveform of Ram Control Block

In this waveform we can see the values of all the inputs and outputs. Here, the inputs values are given differently to check the expected outputs in all cases.

3.1.2. The Synthesis Result:

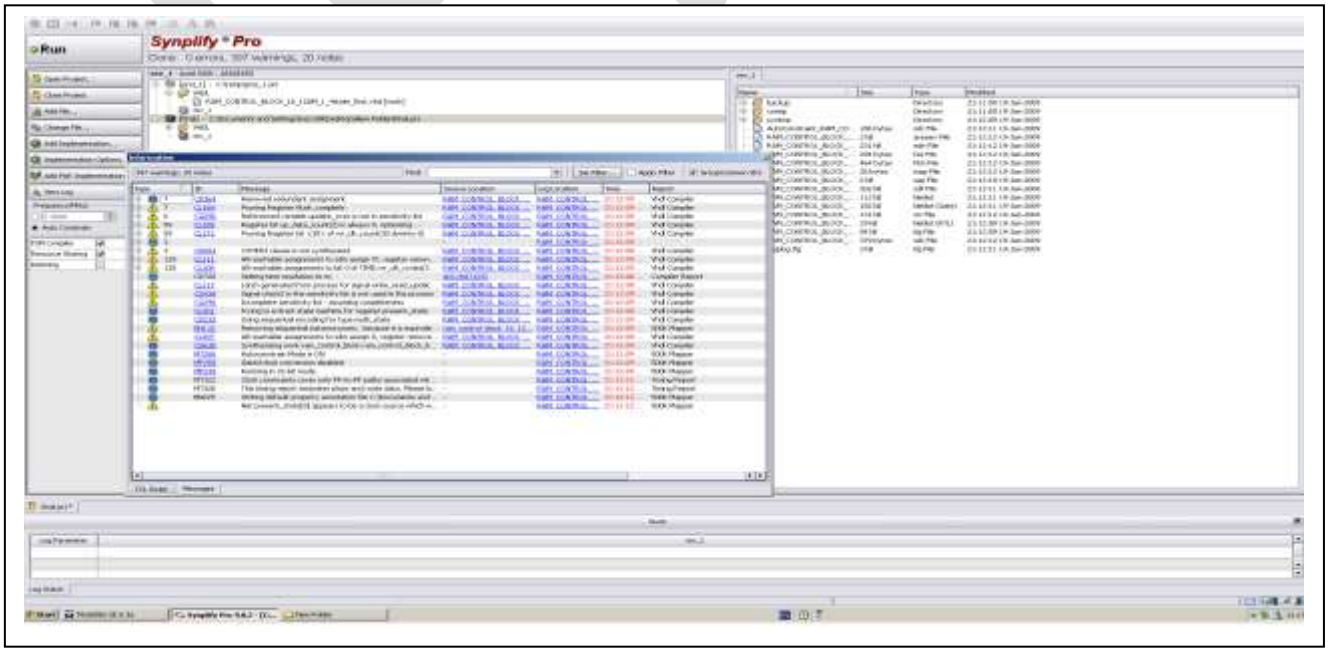

Fig 2: Synthesis Result of Ram Control Block

The RAM CONTROL BLOCK is synthesized.

4. Conclusion

I have tested the RAM Control Block by using the test bench created by me and check the inputs and outputs in the simulation waveform. To complete the total Vector Display Process, my next step will be making the Draw Block to integrate with my one. I hope, I will able to finish my work very soon.

ACKNOWLEDGEMENT:

I would like to thank Dr. Tom Clarke for his enormous support while doing this work. I also want to thank Raj, Kim for their help to check my result with their test bench.

REFERENCE:

[1] Buckey, D.J., "bgis introduction to GIS", BGIS-SANBI