Symbol Generator Blocks

# Abbreviations

MP= Message Pack (includes [SOF, Type, Address, Data Length ,Payload ,CRC ,EOF].

MPD= Message Pack Data (1 pack of 8 bits that is transmitted in the Payload field. There can be more than 1 MPD in Payload field and the number of MPD is represented by the field Data Length.

OPU= the block Opcode\_Unite.

RM= the block Read\_Manager.

Remarks:

1. we figured that we need only 23 bits to create 1 opcode (equal to 3 MPD).

Calculation:

* command =
* Com\_type – add or remove (1 bit)
* x coordinate (4 bits)
* y coordinate (5 bits)
* com\_add : address of the symbol in SDRAM (11 bits because: SDRAM depth is 2^12, SDRAM width is 2^8, and there are 4 banks represented by 2 more bits. However, we need only the depth of the SDRAM to represent the address of the symbol, because we use all the width of the SDRAM (we use the whole row) therefore 12 bits for address.

However, the symbol is saved in 2 rows in the SDRAM therefore the LSB is known – (even or odd row) , so we actually need 11 bits.

* 2 bits to represent the Bank we use in the SDRAM (total of 4 Banks)

**TOTAL = 23 bits for the command**

1. We want to save 1 symbol that will be the color black to represent the removing of a symbol. This unique symbol will be saved at the first row of the SDRAM (at address "000000000000,00000000,00" = (row,col,bank).

# Conceptual Description – Data Flow

The data flow in the Symbol Generator block is as follows:

1. The command packages (opcodes) from the SW are transferred through the Wishbone protocol and received in the Opcode Unite block. The Opcode Unite block unites the commands, which come in byte packets, and transfers them to the Command FIFO.
2. The Command FIFO receives the united opcodes and stores them until a new video frame is initialized by the VESA Generator block. The reason we want to store the opcodes until the new frame is because we don't want to override the current display which is stored in the RAM. Therefore, we store the opcodes until a new frame is initialized, and then we can transfer the opcodes to the RAM.

When opcodes from Command FIFO are transferred to the RAM, the Command FIFO is flushed, in order to be empty and ready for the next data packets to arrive.

1. The RAM represents the current display. Each row in the RAM indicates a 1 (x,y) block on the display.

The RAM receives the opcodes from the Command FIFO, and stores each opcodes in the relevant row in the RAM (the relevant row is calculated using fields in the Opcode). The RAM stores the addresses in the SDRAM, in which the wanted symbols are saved.

When all the opcodes were updated in the RAM, a signal for the Read Manager block is asserted.

1. When the VESA Generator requests video data, the Read Manager initializes read transaction from the SDRAM through the Wishbone master interface. The addresses of the symbols in the SDRAM are stored in the RAM. The Read Manager manages the read from the SDRAM using calculations of the row and the column of the address in the SDRAM. The data which is received from the SDRAM in the read transaction, is stored in one of the FIFOs in the Read Manager block: FIFO-A or FIFO-B.
2. Video data to the VESA Generator block is transferred from the Read Manager block. The data comes from FIFO-A or FIFO-B, respectively.

To add a block diagram of the conceptual description of the Symbol Generator block!

# Detailed Description

## Opcode Unite - OPU

Goal : OPU unites every 3 packs MPD into 1 opcode by a Finite State Machine (FSM). Writing data to RAM.

Functionality:

When Wbs\_adr\_i [9..0] = OCU\_address, then OCU is being activated.

FSM:

When we reach state MPD3 then we have 1 opcode (24 bit).

Then we:

1. Calculate which row to write to RAM (row\_RAM\_o [8..0]= 20\*x + y). We have 300 rows in total (as the number of (x,y) blocks) ).
2. If com\_type = 0 (remove a symbol) then data\_RAM\_o[0..13]= "0…0"

Else (add a symbol) data\_RAM\_o[0..13]= "com\_add".

1. Valid\_RAM\_o ='1'.

We also implement a counter of received bytes: when the counter reaches Wbs\_tga\_i it means the last change (therefore the last opcode) was transmitted. Then rd\_mng\_valid='1', activating Read\_Manager (RM).

Pins:

Detailed explanation on pins:

Input:

1. Clk\_133 : the main clock to which all the internal logic of the Symbol Generator block is synchronized.
2. Reset : asynchronous reset of the Symbol Generator block. TBD - where it comes from.
3. Wbs\_adr\_i [9..0] : this pin is sent from the WBS. It indicates the address of the Display Controller Top. With it we can know if our blocks inside Display Controller are requested – TBD with Beeri
4. Wbs\_tga\_i [9..0] : this pin is sent from the WBS. It indicates how many words are there that are transmitted to us (in other words, it indicate the field Data\_Length in the message pack.
5. Wbs\_dat\_i [7..0] : this pin is sent from the WBS. It indicates the data itself from the field Payload in the message pack.
6. Wbs\_cyc\_i : this pin is sent from the WBS. It indicates the signal cycle required by the Wishbone protocol.
7. Wbs\_stb\_i : this pin is sent from the WBS. It indicates the signal strobe required by the Wishbone protocol.

Output:

1. Wbs\_ack\_o : the acknowledge signal required by the Wishbone protocol.
2. Wbs\_stall\_o : the stall signal required by the Wishbone protocol. It indicates that the slave is busy, therefore, the will be repeated until stall is low.
3. Wbs\_err\_o : the error signal required by the Wishbone protocol. TBD – how we use it?
4. RAM\_adr [8..0] : this pin is sent to the RAM. It represents the row to which we want to write to the RAM.

Reminder: each row in RAM represent 1 (x,y) block (total 300 blocks).

1. RAM\_data[13..0] : this pin is sent to the RAM . It represents the data to be send to the RAM (represent the first row in which the symbol is saved in SDRAM).
2. RAM\_wr\_en : this pin is sent to the RAM. It represents the data to RAM is valid, therefore we can write data to the RAM.
3. Rd\_mng\_valid: this pin is sent to the RM (Read\_Manager) . It equals '1' when the last opcode is sent to the RAM, indicating we can now start generating the new frame (initialize read transaction to the SDRAM).

## Command FIFO

Goal: The main purpose of the Command FIFO is to store commands from the Opcode Unite block. The reason we want store the commands before transferring them to the Main RAM is because we don't want to override the current state of the display in the RAM, in case it is the middle of the frame. We will write to the Main RAM only at start of the frame – when VSYNC of the VESA is active.

Functionality: The Command FIFO receives the opcodes from the Opcode Unite block. It stores the commands until the VSYNC of the VESA is active. Then, the opcodes will be written to the respective row in the RAM

Pins:

## Main RAM

Goal: In RAM we save the row in SDRAM, in which the symbol of this block starts. (14 bits)

Functionality: The Command FIFO writes to Main RAM. Read\_Manager (RM) reads from RAM.

Pins:

Detailed explanation on pins:

Input:

1. Clk\_133 : the main clock to which all the internal logic of the Symbol Generator block is synchronized.
2. Reset : asynchronous reset of the Symbol Generator block. TBD - where it comes from.

Signals for write:

1. RAM\_adr\_in[8..0]: the row number of RAM to which we want to write.

Remark: we need 300 rows for each (x,y) block (therefore we need 2^9 bits)

1. RAM\_wr\_en: enables write to RAM.
2. RAM\_data\_in[13..0]: the data to be written to RAM.
3. RAM\_in\_valid : indication that the data on the RAM\_data\_in signal is valid.

Signals for read:

1. RAM\_adr\_rd[8..0] : the row number of RAM from which we want to read.
2. RAM\_rd\_en: enables read from RAM.

Output:

1. RAM\_data\_out[13..0]: the data read from RAM.
2. RAM\_out\_valid: equals 1 when RAM\_data\_out is valid.

## Read Manager - RM

Goal: reading from SDRAM data , sending it to SDRAM through WBM , receiving symbols from SDRAM through WBM, saving it in internal FIFOs, sending it to Vesa according to certain algorithm.

Functinality:

When Valid\_RM\_i ='1' is sent from OPU, RM starts working.

Algorithm:

Not written yet. We want you to answer our questions first about the pins we receive from mem\_mng.

Our main difficulties are:

1. When I want to read from SDRAM, I first need to send the address from which I want to read to reg, so I write the address of this reg in Wbm\_add\_o[9..0].

Beery said its address is "0..010".

First: where does it written in the documentations? why so much bits?,

Second: what I should write in Wbm\_add\_o[9..0] when I want to receive Data from SDRAM ? does it matter?

Third: what should we do when Wbm\_stall\_i = '1' ? Wbm\_err\_i?

1. We don't understand what does it mean that the Vesa requests pixels from us.

We don't want it to request from us anything , we just want it to accept what we are sending to it when it is comfortable to us. Can we do it?

1. Is Vesa\_controller who sits inside Display controller is the Vesa Generator?

Why in Display Controller Documentation we can't find him and its pins?

1. In Vesa Generator Documentation page 5 it is said that someone from the outside should supply the pixel-clk. Who exactly? How we should implement this in our blocks? Is pixel-clk = 40 MHz (the Vesa clk) ?
2. We have trouble understanding why some pins related to Veas, like Blanking, Hsync,… come out from Display Controller Top, while others, like 40 MHz , req\_In\_trig … are internal? Why is this mixed? It seems as if some part of the Veas is Inside Display Controller and some part is exernal. Can you explain to us this?
3. Regarding the total Message Pack:

Where does it written what is the functionality of the fields Type, Address?

For example, Beery commented us in the mail you send him that:

* + תזכורת: כתיבה לרגיטסרים תתבצע כאשר ביט מספר 7 (הMSB) בשדה הtype בהודעת UART הינו '1'.
  + כתיבת הכתובת הרצוייה, כאשר הType מציין מוד Debug, כלומר ביט '0' (ה-LSB) הינו '1'.

Where is this written?

Moreover, if you can tell us where the functionally of each field is written, it will be great.

Pins :

Detailed explanation on pins:

Remark: each time I write "<XXX> is sent to/from SDRAM" I actually mean <XXX> is sent to/from WBM and eventually <XXX> is sent/received by SDRAM.

Input:

1. Valid\_RM\_i: this pin is sent from the OCU. It equals 1 when the last opcode is sent to the RAM, indicating we can now start generating the new frame.
2. Data\_RAM[0..13]\_i : this pin is sent from the RAM . It represents the data readed from the RAM (represent the first row in which the symbol is saved in SDRAM).
3. Data\_val\_RAM\_i: this pin is sent from the RAM .It equals 1 when Data\_RAM[0..13]\_i is valid.
4. Wbm\_dat\_i[7..0]\_i: this pin is sent from the SDARM. This is the data from SDRAM.
5. Wbm\_stall\_i :this pin is sent from the SDARM. It equals 1 when SDRAM is not ready.
6. Wbm\_ack\_i : this pin is sent from the SDARM. It equals 1 when Wbm\_dat\_i[7..0]\_i is valid.
7. Wbm\_err\_i : this pin is sent from the SDARM. It equals 1 when an error occurd.

Output:

1. En\_read\_RAM\_o: this pin is sent to the RAM. It enables reading from RAM.
2. Row\_RAM\_o[8..0]: this pin is sent to the RAM. It indicates the row's number of RAM.
3. Wbs\_dat\_o[7..0]: this pin is sent to the reg inside SDRAM. It represents the address we want to read from SDRAM. Q: Am I right?
4. Wbs\_ack\_o: this pin is sent to the SDRAM. It equals 1 when Wbs\_dat\_o[7..0] is valid/ when Wbm\_dat\_i[7..0]\_i is received.
5. Wbm\_add\_o[9..0] : this pin is sent to the SDRAM. It represents the address of the reg inside mem\_mng. When we send Wbs\_dat\_o[7..0], this line equals "000000010" (address of reg inside SDRAM). Q: Am I right? If so, why so much bits?, when I want to receive data from SDRAM, what should I write in this line? Does it matter?
6. Wbm\_tga\_o[9..0] : this pin is sent to the SDRAM. It represents the reading burst length. It suppose to be the size of 16 at a time. (we read 32 pix = 16 words at a time).
7. Wbm\_cyc\_o: this pin is sent to WBM. It indicates the signal cycle required by the Wishbone protocol.
8. Wbm\_std\_o: this pin is sent to WBM. It indicates the signal strobe required by the Wishbone protocol.
9. R\_out [9..0]: red value to Vesa\_ Generator.
10. B\_out[9..0] : blue value to Vesa\_ Generator.
11. G\_out[9..0] : green value to Vesa\_ Generator.
12. Blanck\_out[9..0] : Blanking signal to Vesa\_ Generator.
13. Hsync\_out[9..0] : Horizntal Sync to Vesa\_ Generator.
14. Vsync\_out[9..0] : Vertial Sync to Vesa\_ Generator.