**07.11.2011**

SDRAM:

1. The size of the SDRAM is 2^20X256 = bitXwords, each word has 16 bits. The SDRAM contains a number of symbols (for now let's say 10), each symbol the size of 32X32=1024 pixels. Each symbol is grayed, meaning each pixel has 8 bits. Therefore, each symbol has 1024X8=8192 bits.
2. The symbols are saved "Mefuchpachim", meaning they are saved row after row in the SDRAM. Each row in the SDRAM has 256X16=4096 bits, so we'll need to save each symbol in 2 rows in the SDRAM.

This way, we can use the Burst Read operation in the SDRAM and save time, since we burst the whole row and not just part of it.

1. All the components in the structure work in the same frequency (133MHz). That includes the FIFOs, SDRAM, RAM and VESA controller (But not the VESA itself which work at 60MHz). Therefore, we need to use a FIFO DC (Dual Clock) so we "feed" it from the SDRAM in 133MHz rate and the Vesa read from it at 60MHz
2. The chosen VESA is 640X480 60MHz.
3. The frame size is 640X480 pixels. The frame is divided into 20X15 blocks; each block is the size of a symbol (32X32 pixels). The blocks are aligned, meaning they don't "slide" on one another, but fixed in their place. Each block is referred to by a dot (x,y) the user enter in GUI.

Remark: 640/32 = 15, 480/32=20.

1. The FPGA contains a RAM that saves for each block (x,y) in the frame the following format code:

[ 0/1 , address of the first bit of the desired icon sits in the SDRAM] . The first bit in the code indicates whether or not the block will show a symbol ( '1' = showing, '0'=not showing). The other bits are the address of the first bit of the desired symbol in the SDRAM.

1. The RAM size is 20X15=300 rows, and the number of columns depends on how many symbols there are.
2. The GUI figure:

Choose symbol: entering a number which refer to a specific symbol

Maybe we will introduce a chart of symbols and their referred numbers.

Location: X=\_\_\_ , Y=\_\_\_ (choosing the block).

Add/Remove: a symbol.

GENERATE button.

1. After pushing the GENERATE button MATLAB generate a command of the following structure:

Address in SDRAM (where the first bit of the symbol is sitting)/ add or remove symbol / row / column. Maybe something else will be more efficient.

1. The command itself is wrapped with a startbit, stopbit, CRC, and other bits from Beeri project (but we don't implement the wrapping process).
2. Need to implement a "command decoder".

FIFOs:

1. We need FIFOs to read from the SDRAM the symbols and feed the Vesa.

We find it efficient to use 2 line of FIFOs, so whenever one line reads from SDRAM, the other writes to the Vesa. We have several options to do this:

1. We can use 2 "Big" FIFOs to do this toggling, or we can use 2 lines of FIFOs. The 2 options demand the same memory size, however, we are not sure what will be easier to debug.
2. We decided that each symbol has 32 rows of pixels. So, if we number each row of every symbol, the frame looks like this:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 0 | | 1 | | … | | 31 | | |  | | --- | | 32 | | 33 | | … | | 63 | | |  | | --- | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |
| |  | | --- | |  | |  | |  | |  | | |  | | --- | |  | |  | |  | |  | |  |  |  |
|  |  |  |  |  |
|  | Block (i,j) |  |  |  |

We have 2 options of filling the FIFOs:

1. Each FIFO will contain 1 symbol:

FIFO 0 FIFO 1 … FIFO N

|  |
| --- |
| 32 |
| 33 |
| … |
| 63 |

|  |
| --- |
| 0 |
| 1 |
| … |
| 31 |

1. Each FIFO will contain 1 line of the frame, so each part of every symbol will be saved in another FIFO:

FIFO 0 FIFO 1 … FIFO N

|  |
| --- |
| 1 |
| 33 |
| … |
| … |

|  |
| --- |
| 0 |
| 32 |
| … |
| … |

This way, in one burst of reading one row from the SDRAM, we fill N FIFOs, but the Vesa reads one FIFO per line in the frame.

We think the second option is more convenient.

Let's say from now on that we chose the second option:

1. In order to utilize the burst reading function in the SDRAM optimally, we need 16 FIFO (because in one row sits 16 pixels one symbol = half a symbol).

This way, if we have 2 lines of 16 FIFOs each, then while one line writes to the Vesa (end filling the frame with 16 rows = first half of symbols) the other line reads from the SDRAM the other half of symbols.

One FIFO saves 20 lines of 20 symbols. Each line contains 32 pix, and each pix has 8 bit. Therefore, one FIFO saves 20X32X8=5120 bits, meaning we need 2 M4K for 1 FIFO. We have 32 FIFOs, so the FIFOs need 64 M4K from the FPGA. To this we need to add the RAM and maybe other things we haven't think of, while the FPGA contains 105 M4K.

Another option is to have only 2 toggled FIFOs. While one FIFO write to the Vesa only one row of frame, the other reads the next row from the SDRAM. This way, we'll need only 4 M4K for each FIFO, more than enough. However, we can't use the burst reading function of the SDRAM efficiently, since we need to read 1 pixel from a row of the SDRAM per reading and not the whole row.

We calculated timing for the different option we represented above and found:

Calculating timing:

clk = 133 MHZ , pixel\_clk = 25.175 MHz (frame\_rate = 60 Hz)

1. Writing half of the symbol (16 rows at once)

Reading half of the symbol from the SDRAM:

* Approx. 9 clk for the control signals – from RAS to data on the line
* 256 clk for reading half of the symbol (1 row in the SDRAM)
* Total = approx. 270 clk (round up)

We need to read 20 symbols from the SDRAM

* 270 clk x 20 symbols = 5400 clk = 40.6 usec

To VESA Gen.:

In the NOT-active-top of the frame (time before the active frame starts) :

* V sync = 2 lines
* V back porch = 25 lines
* V top border = 8 lines
* Total = 35 lines

And in the active part of the frame we use 16 lines

Therefore, the limiting part is the active part of the frame (16 lines)

Total = 16 lines x 800 pixels (including H blanking) = … = 508.16 usec

* Reading 20 rows of the SDRAM is much faster than writing them to the VESA Gen.

Pros:

1. When reading from SDRAM, the CAS is always zero (starting from the head of the row) – less complicated logic

Cons:

1. “wasting” a lot of the FPGA memory (64 M4K blocks)
2. Worst case: Writing 1 row of the symbol at once

Reading 1 row of the symbol from the SDRAM:

* Approx. 9 clk for the control signals – from RAS to data on the line
* 16 clk for reading 1 row of the symbol
* Total = approx. 30 clk (round up)

We need to read 20 symbols from the SDRAM

* 30 clk x 20 symbols = 600 clk = 4.511usec

To VESA Gen.:

In the NOT-active-top of the frame (time before the active frame starts) :

* V sync = 2 lines
* V back porch = 25 lines
* V top border = 8 lines
* Total = 35 lines

And in the active part of the frame we use 1 line

Therefore, the limiting part is the active part of the frame (1 line)

Total = 1 line x 800 pixels (including H blanking) = … = 31.76 usec

Reading 20 rows of the SDRAM is much faster than writing them to the VESA Gen.

Pros:

1. Using less FPGA memory
2. The debug is easier?

Cons:

1. When reading from SDRAM, the CAS is increased by 16 each read - complicated logic (using counters)

**Conclusion:**

Maybe we need to choose a middle option – not 1 row, and not 16 rows

Middle option:

To read 1/4 of the symbol (8 rows), it takes 32 M4K blocks, and the logic is not very complicated (for counting the CAS signal in the SDRAM – values: 0 or 128)

**Principle of operation:**

GUI:

Add/Remove: \_\_\_\_\_\_\_\_

Location: X=\_\_, Y=\_\_

Symbol num: \_\_\_\_

GENERATE button

Generating the command + "command decoder"

In VHDL

In MATLAB

Active Frame

15X20 symbols

|  |  |  |  |
| --- | --- | --- | --- |
| 1,20) | ... | (1,2) | (1,1) |
|  |  |  | (2,1) |
|  |  |  | … |
| (15,20) | … | … | (15,1) |
|  |  |  |  |

RAM:

[0/1 , symbol address in SDRAM] for block (x,y)=(1,1).

[0/1 , symbol address in SDRAM] for block (x,y)=(1,2).

. . .

[0/1 , symbol address in SDRAM] for block (x,y)=(1,20).

[0/1 , symbol address in SDRAM] for block (x,y)=(2,1).

[0/1 , symbol address in SDRAM] for block (x,y)=(2,2).

. . .

[0/1 , symbol address in SDRAM] for block (x,y)=(15,20).

SDRAM:

|  |  |  |  |
| --- | --- | --- | --- |
| Pix 0 of sym k | Pix 1 of sym k | … | Pix 15 of sym k |
| Pix 16 of sym k | Pix 17 of sym k |  | Pix 31 of sym k |

0..15 pixels of symbol 1

16..31 pixels of symbol 1

0..15 pixels of symbol 2

16..31 pixels of symbol 2

16..31 pixels of symbol N

0..15 pixels of symbol N