**Technion**

*Electrical Engineering Department*

High Speed Digital Systems Lab

**Menu Navigation Project**

**Characterization Documentation**

**Student: Tzahi Ezra**

**Advisors: Netanel Yamin**

**Moshe Porian**

**Last updated: 22.04.2014**

**Characterization document:**

1. General scheme

The following scheme includes the Managing Marker block added to the SG TOP block,

Y\_sync

X\_sync

Debouncer

Debouncer

Debouncer

Debouncer

Software synchronization

Right

X, Y

Location

To manager

RIGHT

left

Hor\_out

LEFT

UP

To manager

ver\_out

up

DOWN

down

Navigator block

From Managing Marker block

Ver\_out

Hor\_out

Sym\_col

Sdram\_mux\_out[7:0]



Sym\_row

Ram\_data\_out[12:0]

Cond

Counters

FSM

Fifo\_b\_wr\_en

Fifo\_b\_rd\_en

sdram\_rd\_en\_out

Sdram\_addr\_rd[23:0]

Fifo\_b\_data\_in[7:0]

Ram\_addr\_rd[8:0]

Fifo\_a\_data\_in[7:0]

Fifo\_a\_wr\_en

Fifo\_a\_rd\_en

Ram\_rd\_en\_out

Manager

Symbol inversion

Clk

Reset\_n

Req\_in\_trg

Mng\_en

Sdram\_data[7:0]

Sdram\_data\_valid

Fifo\_a\_empty

Fifo\_b\_empty

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Width (bits)** | **Signal Type** | **Direction** | **Signal name** |
| system clock (100MHz) | 1 | Std logic | In | Clk |
| Count right Enable | 1 | Std logic | In | right |
| Count left Enable | 1 | Std logic | In | left |
| Count up Enable | 1 | Std logic | In | up |
| Count down Enable | 1 | Std logic | In | down |
| Horizontal location value set by the software | hor\_width\_g | Std logic vector( hor\_width\_g-1 downto 0) | In | X\_set |
| Vertical location value set by the software | ver\_width\_g | Std logic vector ( ver\_width\_g-1 downto 0) | In | Y\_set |
| Horizontal location output | hor\_width\_g | Std logic vector( hor\_width\_g-1 downto 0) | out | Hor\_out |
| Vertical location output | ver\_width\_g | Std logic vector( ver\_width\_g-1 downto 0) | out | Ver\_out |
| Horizontal location Synchronized value with the software | hor\_width\_g | Std logic vector( hor\_width\_g-1 downto 0) | out | X\_Sync |
| Vertical location Synchronized value with the software | Ver\_width\_g | Std logic vector( Ver\_width\_g-1 downto 0) | out | Y\_Sync |

|  |  |  |  |
| --- | --- | --- | --- |
| **Generic parameter** | | |  |
| **Name** | **Type** | **description** | **Default value** |
| Time\_till\_trig | natural | defines the time required to wait until an event in which for one cycle the debouncer output turns to '1', after its input is high for a period of Time\_till\_trig msec | 2 msec |
| hor\_width\_g | natural | defines the horizontal width of Hor\_out lines needed to hold the maximum value kept in the counter | 5 |
| ver\_width\_g | natural | defines the vertical width of Ver\_out lines needed to hold the maximum value kept in the counter | 4 |

1. Inverting a symbol:

Marking of a symbol would be responded by inverting the symbol pixels (32\*32).

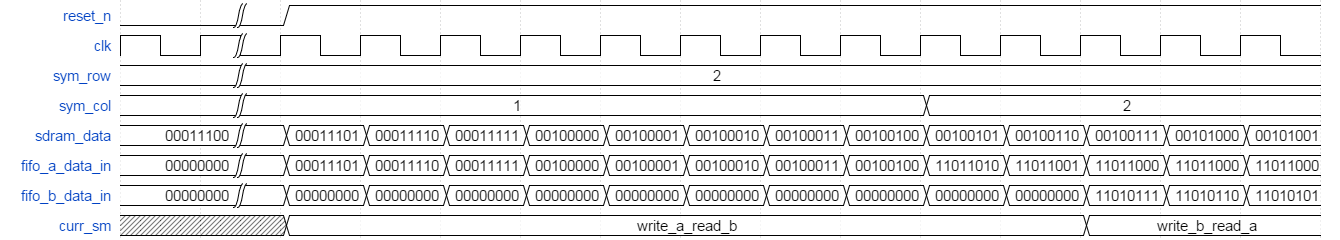
Cond

Pixel (7..0)

Symbol inversion

In the manager block, the information of each pixel of the chosen symbol, being read from the SDRAM is inverted before being written to the fifos and passed to the VESA display. Using a synchronous mux as shown in the figure above. The Cond would be '1' if the internal row and column that are being calculated (represent the displayed symbol) matches the desired marker symbol's location, which would be discussed later on.

Following presented a simulation showing the inversion operation correctness, in the manager block, when the manager FSM, responsible for toggling between fifos a and b, is in state write\_a\_read\_b, or write\_b\_read\_a, and when the internal row and column count are both 2, the corresponding outputs fifo\_a\_data\_in and fifo\_b\_data\_in, receives the inverted pixel input from the sdram, sdram\_data:



1. X and Y registers

The desired marker symbol's location, is saved in registers: X Reg, () and in Y Reg, ((. 4 buttons that are located on the DE2 board, are used to move the marker's location: left, right, up and down. Each button produces an active high signal when it is pressed. In addition, the unit can update the GUI with the desired marker symbol's location, through the software synchronization block.

A managing Marker block is responsible to save the current marker's location. 2 counters are used to achieve that. Counter X, ((, that manages the current marker's horizontal location, and Counter Y, ((, that manages the current marker's vertical location. This block receives the desired marker symbol's location (registers X,Y and the 4 buttons signals). This block holds 4 debouncers responsible to trigger a move only when the user presses the button for more than 2 msec.

The counters are cyclically incremented and decremented, i.e, X is decremented to 19 from 0, and incremented to 0 from 19.

In the special case when counter X output is 19 and the right signal is triggered, then Counter Y is decremented, and when counter X output is 0 and the left signal is triggered, then Counter Y is incremented.

1. Debouncer

**Functionality:**

The debouncer is a counter which counts how long its input is '1', and triggers when it reaches a defined value (in our case, 2msec is used). It works with the system frequency (100 MHz) such that when it reaches the defined value, its output turns from '0' to '1' for 1 period, than return to '0'.

**Block Top Diagram:**

Clk

Dout

Debouncer

Din

**Block implantation:**

The unit uses T-FF modules and AND gates to fulfill its task. Total 15 T-FF, 1 AND2-AND14, 2 AND15. The following diagram represent the block's inner implementation:

2msec

Din

Q0

Clk

Q

T

Dout

Reset\_n

Reset\_n

Q1

Q

T

T

Q

Reset\_n

Debouncer

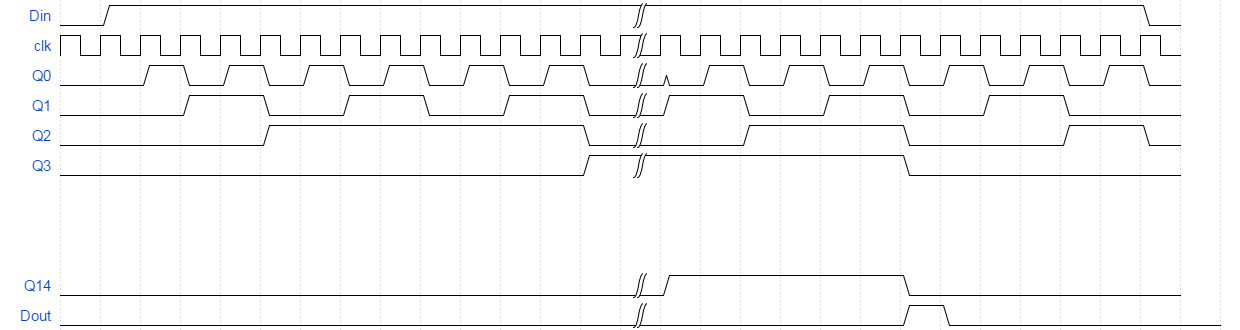
Q\_FF\_required\_g-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Component generics** | | |  |
| **Name** | **Type** | **description** | **Default value** |
| FF\_required\_g | natural | defines the number of the FF needed for counting number of system clock periods that equals to TBD | 15 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Width (bits)** | **Signal Type** | **Direction** | **Signal name** |
| system clock (100MHz) | 1 | Std logic | In | Clk |
| Input from a button on the DE2 board | 1 | Std logic | In | Din |
| Debouncer data output | 1 | Std logic | Out | Dout |

**Debouncer operation example:**

The following diagram shows Debouncer wave form. Special cases where all outputs are '1', output turns to '1', and afterwards TFF's outputs turn to zero. When Din turns to zero (reset\_n is then active low), TFF's outputs turn to zero.



1. Cyclic Counter

**Functionality:**

The Counter returns the ng its input is '1', until it reaches a defined value (in our case, 2msec is used). It works in the system frequency (100 MHz) and when it reaches the defined value, its output turns from '0' to '1' for 1 period, than return to '0'.

hor\_width\_g

Clk

X\_out

Up

Down

Right

Reset\_n

X, Y

Location

Left

|  |  |  |  |
| --- | --- | --- | --- |
| **Component generics** | | |  |
| **Name** | **Type** | **description** | **Default value** |
| hor\_width\_g | natural | defines the width of X\_out lines needed to hold the maximum value kept in the counter | 5 |
| ver\_width\_g | natural | defines the width of Y\_out lines needed to hold the maximum value kept in the counter | 4 |

Y\_out

ver\_width\_g

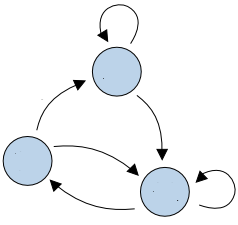
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Width (bits)** | **Signal Type** | **Direction** | **Signal name** |
| system clock (100MHz) | 1 | Std logic | In | Clk |
| The block's resets | 1 | Std logic | In | Reset\_n |
| right Enable | 1 | Std logic | In | Right |
| left Enable | 1 | Std logic | In | Left |
| up Enable | 1 | Std logic | In | Up |
| down Enable | 1 | Std logic | In | Down |
| The current X value | hor\_width\_g | Std logic vector(hor\_width\_g -1 downto 0) | Out | X\_out |
| The current Y value | ver\_width\_g | Std logic vector(ver\_width\_g -1 downto 0) | Out | Y\_out |

**Block implantation:**

The block uses an FSM to manage its operation.

The following diagram represent the block's inner implementation:

Clk

FSM

X\_out

Right

Reset\_n

X

Left

Y

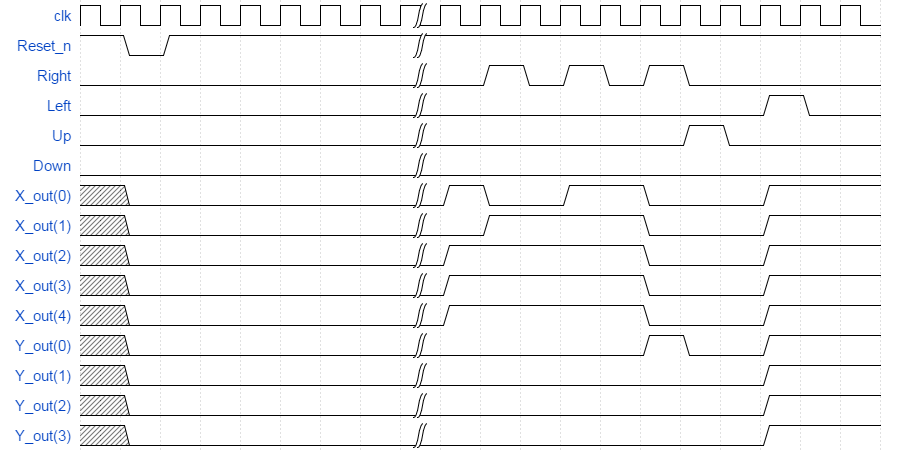
Y\_out

Up

Down

X, Y location

**X, Y location operation example:**

The following diagram shows the block's wave form. Special cases where reset\_n is low, outputs turn to be all zeros. When Right is triggered while X\_out="11111" and Y\_out="0000", they turn to: X\_out="00000" and Y\_out="0001", when Left is triggered while X\_out="00000" and Y\_out="0000", they turn to: X\_out="11111" and Y\_out="11111".