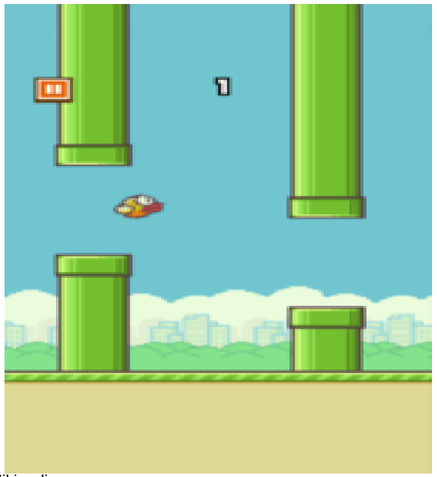
COMPSYS 305: DIGITAL SYSTEM DESIGN I

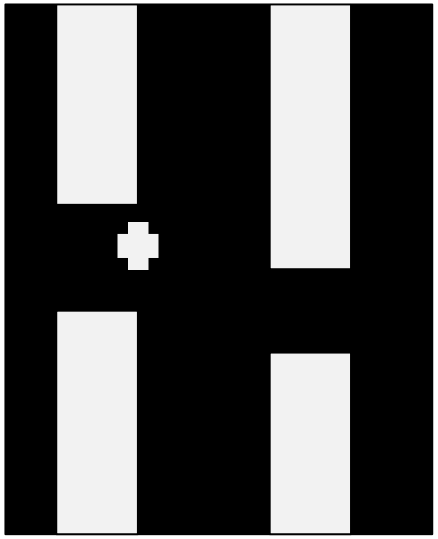
Introduction to Mini Project

Muhammad Nadeem
Dept. of Electrical and Computer Engineering
The University of Auckland
Room: 401. 806, Email: muhammad.nadeem@auckland.ac.nz

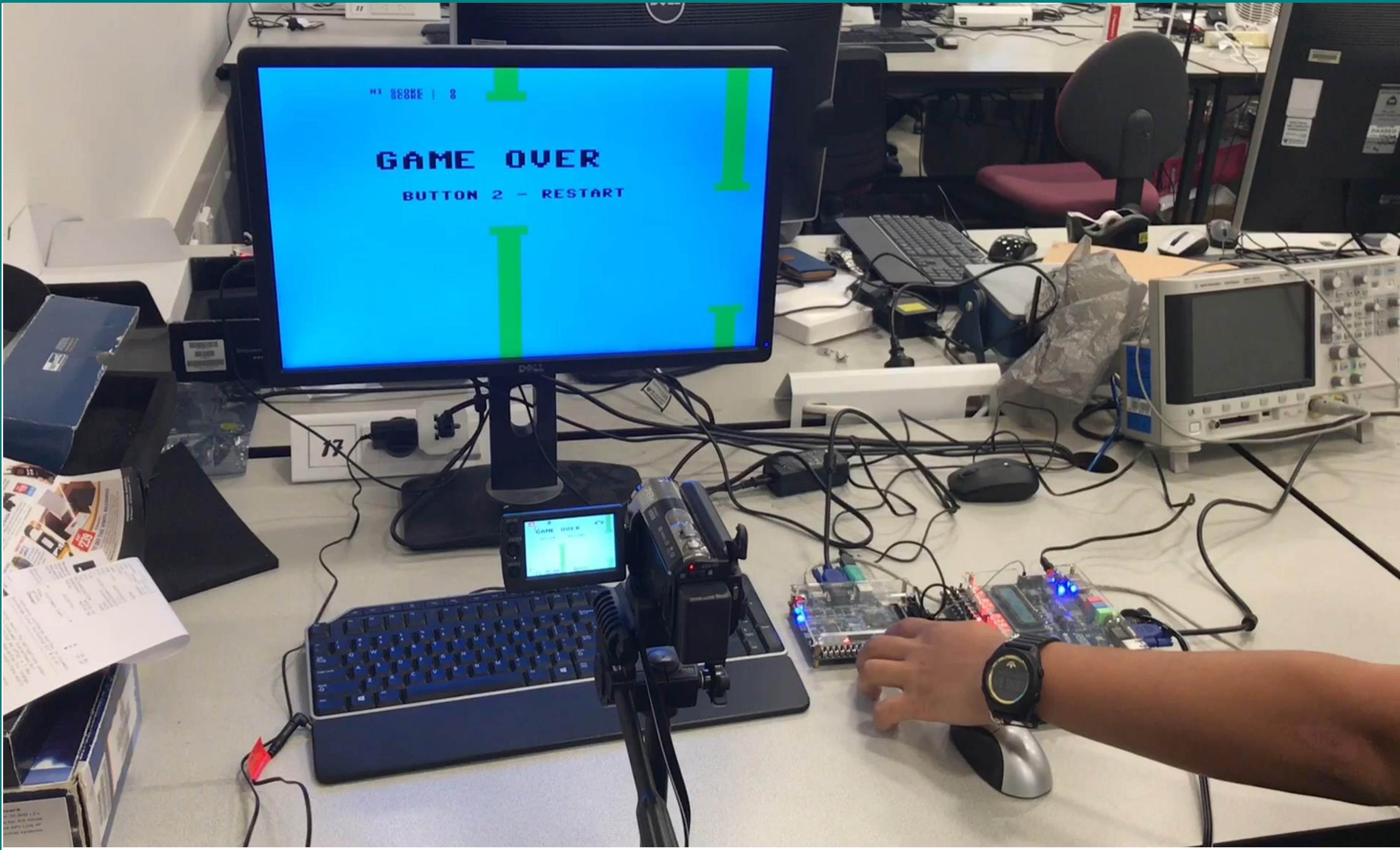
Mini Project

- Design game using hardware components only
- Side-scroller game (moving from left to right)
- Player controls a bird, attempting to fly between rows of pipes without hitting them.
- The bird can move up or down (and backwards/forwards to some extent), controlled using a PS/2 mouse.
- If the bird is not flapping, it will free fall towards the ground and dies if he falls on the ground.
- May have different types of obstacle (e.g. pipes) and/or gifts (e.g. dollars, boxes, special flying abilities).
- The level of difficultness: the horizontal screen motion speed, the types of obstacles, and energy.


Wikipedia



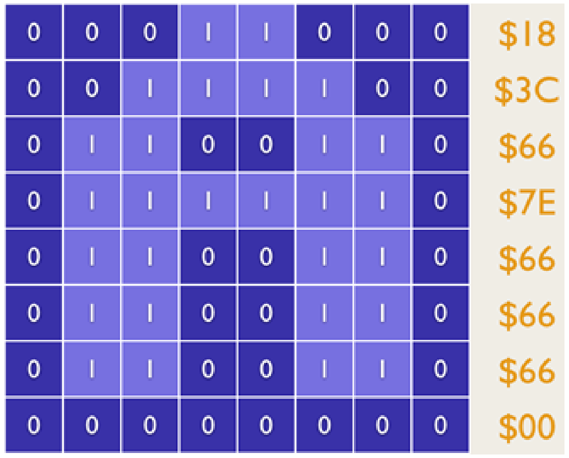
Mini Project



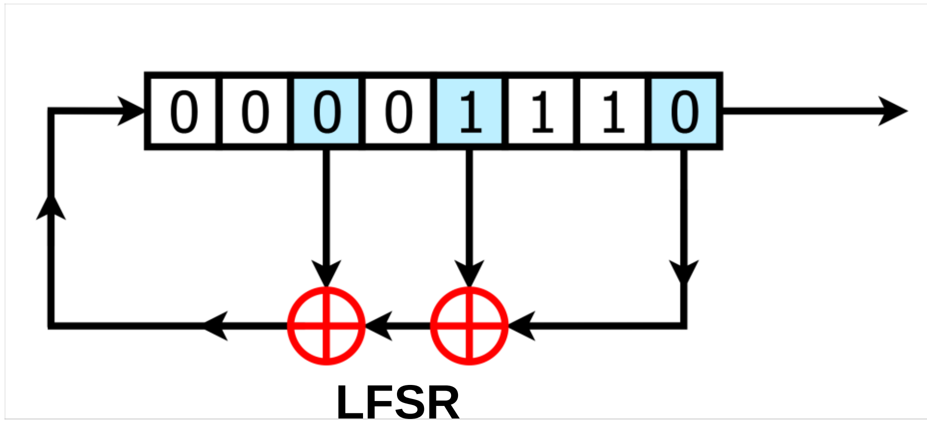
Main Components of Project

VGA Controller

Character ROM



Mouse Interface

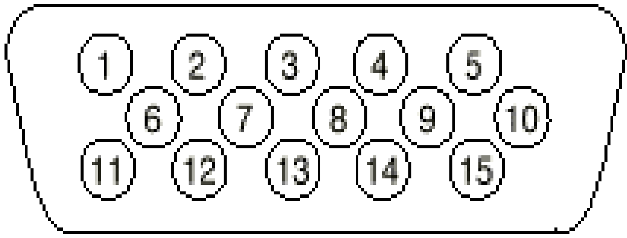


LFSR

VGA (Video Graphic Array) concepts

- VGA is a high-resolution video standard used mostly for computer monitors

PIN	SIGNAL NAME	DESCRIPTION
1	RED	Red video signal
2	GREEN	Green video signal
3	BLUE	Blue video signal
4	MONID(0)	Monitor ID signal 0
5	GND DDC	Return
6, 7, 8	AGND_VID	Analog video ground
9	+5V_IO 5 V	Power for I/O device
10	GND	HSYNC and VSYNC ground
11	VGA_ID	VGA ID signal
12	MONID(2)	Monitor ID signal 2
13	HSYNC	Horizontal synchronization signal
14	VSYNC	Vertical synchronization signal
15	MONID(1)	Monitor ID signal 1

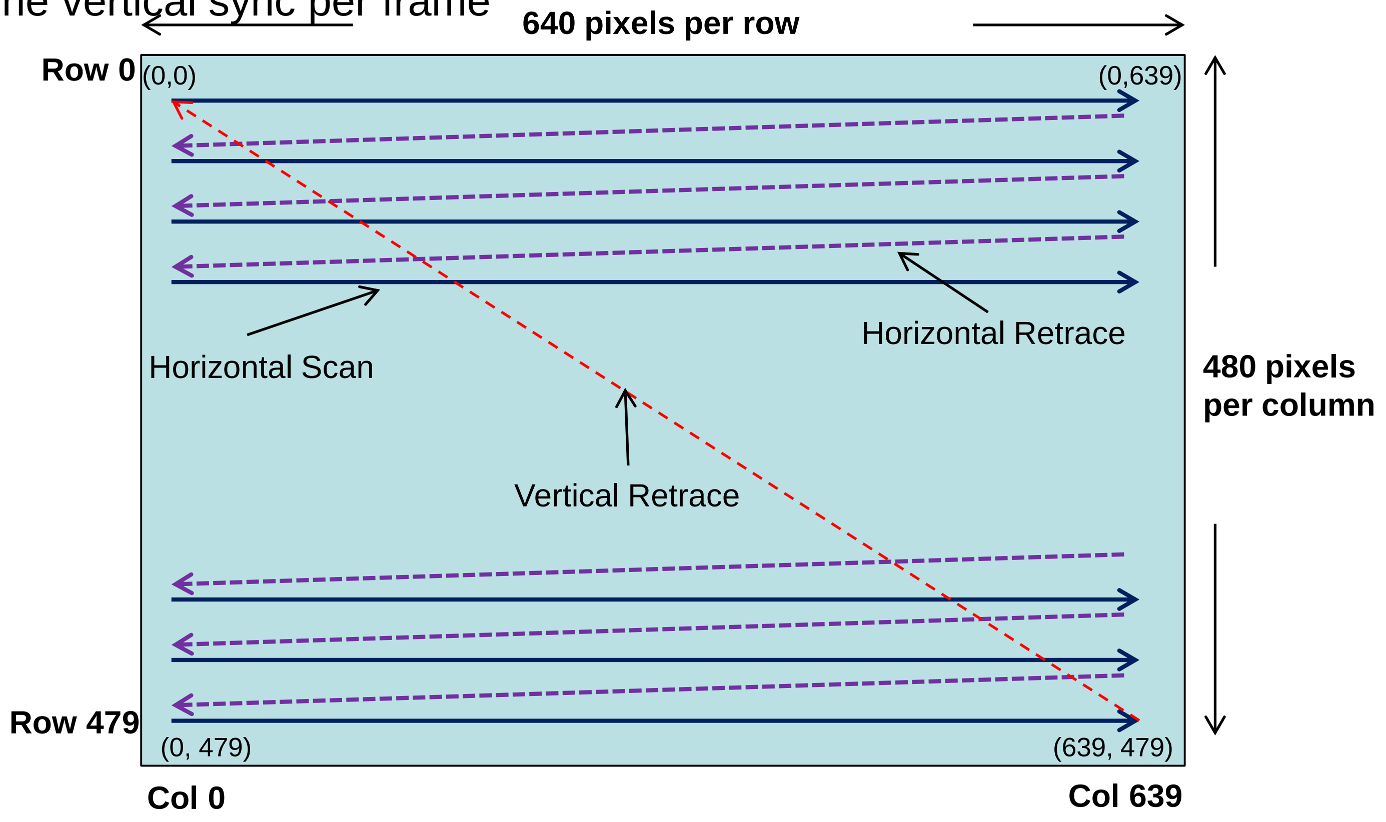


VGA (Video Graphic Array) concepts

- VGA video standard contains 5 active signals:
  - Horizontal and vertical synchronisation signals
  - Three analog signals for red, green and blue (RGB) colours formation
  - By changing the analog voltage levels of the RGB signals, different colours can be produced
  - Depending on the number of bits supported by the development board, different amount of colours can be represented
- Video signal must redraw the entire screen 60 times per sec (60Hz rate) to avoid flickers
  - Human eyes detect flickers at refresh rate less than 30Hz
- We will use the common VGA display standard at 25MHz pixel rate with 640x480 resolution
  - 640x480 is the visible range
  - Each pixel takes 40ns at 25MHz pixel rate

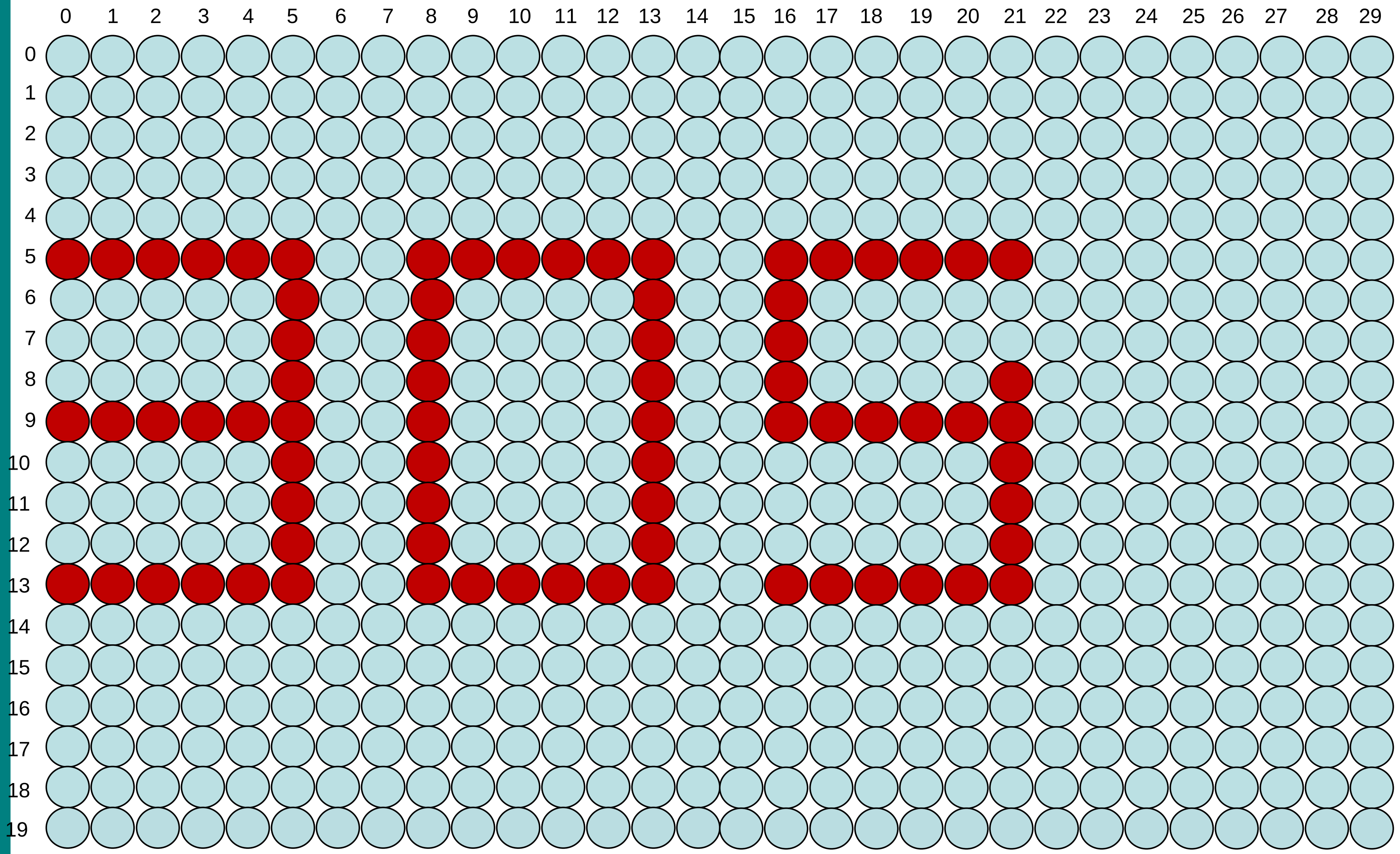
VGA concepts (cont’d)

- Image displayed by turning pixels ONN and OFF
- The screen refresh the display from left to right, top to bottom
- You supply two pulses, hsync and vsync, that let the monitor lock onto timing
  - One horiozonal sync per scan line
  - One vertical sync per frame



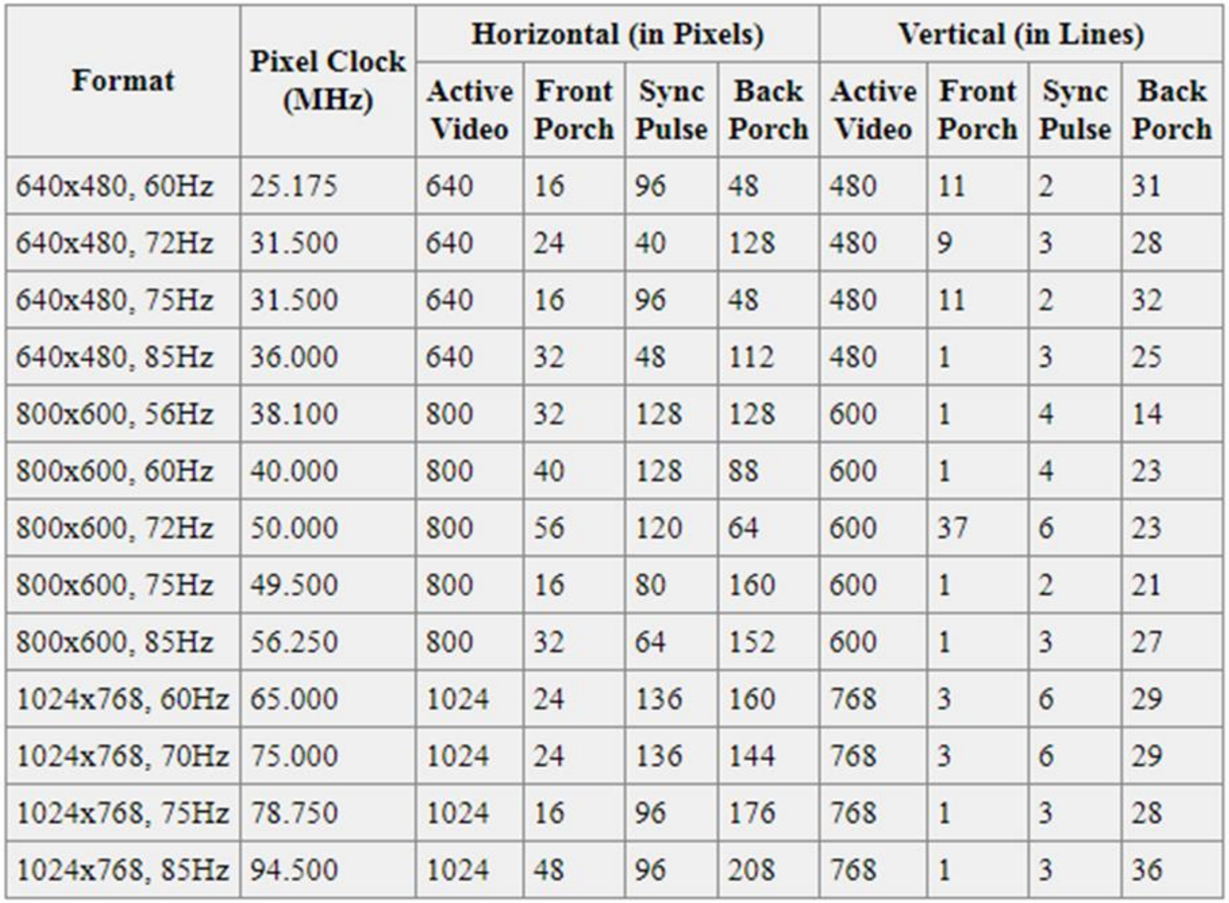
Screen Refresh

Muhammad Nadeem



VGA Timing

- To work with standard monitors and TVs you need to use the correct video timings
- The following table lists timing values for several popular resolutions.

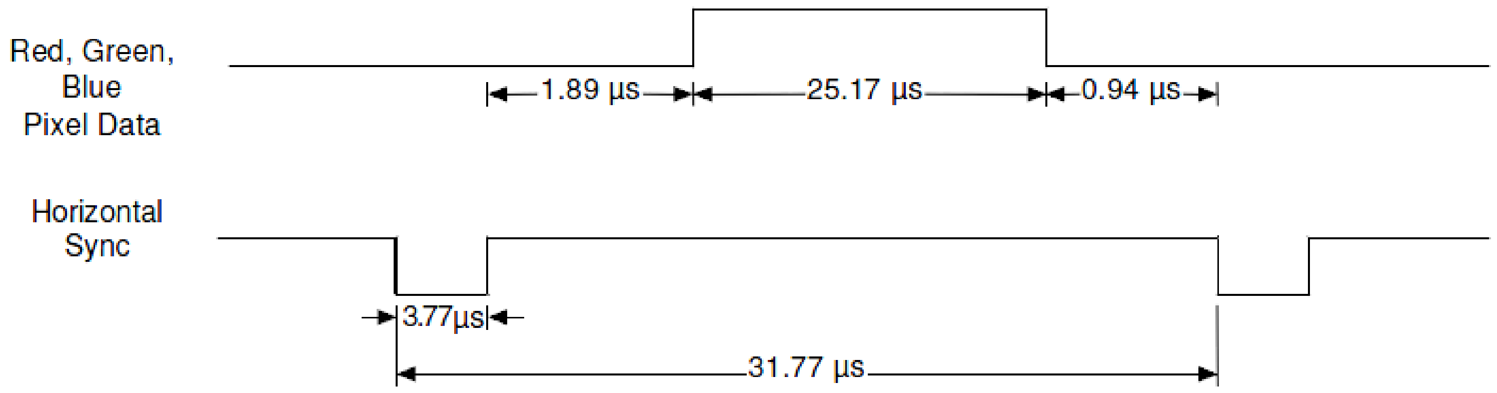


Format	Pixel Clock (MHz)	Horizontal (in Pixels)				Vertical (in Lines)			
		Active Video	Front Porch	Sync Pulse	Back Porch	Active Video	Front Porch	Sync Pulse	Back Porch
640x480, 60Hz	25.175	640	16	96	48	480	11	2	31
640x480, 72Hz	31.500	640	24	40	128	480	9	3	28
640x480, 75Hz	31.500	640	16	96	48	480	11	2	32
640x480, 85Hz	36.000	640	32	48	112	480	1	3	25
800x600, 56Hz	38.100	800	32	128	128	600	1	4	14
800x600, 60Hz	40.000	800	40	128	88	600	1	4	23
800x600, 72Hz	50.000	800	56	120	64	600	37	6	23
800x600, 75Hz	49.500	800	16	80	160	600	1	2	21
800x600, 85Hz	56.250	800	32	64	152	600	1	3	27
1024x768, 60Hz	65.000	1024	24	136	160	768	3	6	29
1024x768, 70Hz	75.000	1024	24	136	144	768	3	6	29
1024x768, 75Hz	78.750	1024	16	96	176	768	1	3	28
1024x768, 85Hz	94.500	1024	48	96	208	768	1	3	36

With analogue VGA monitors you can usually get away with using a 25 MHz pixel clock. However, based on the VESA tolerance of 0.5%, 25 MHz is not acceptable and displays may reject it. Note that 25.2 MHz is considered acceptable by VESA, which gives a 60 Hz refresh rate (rather than 59.940 Hz).

VGA horizontal sync

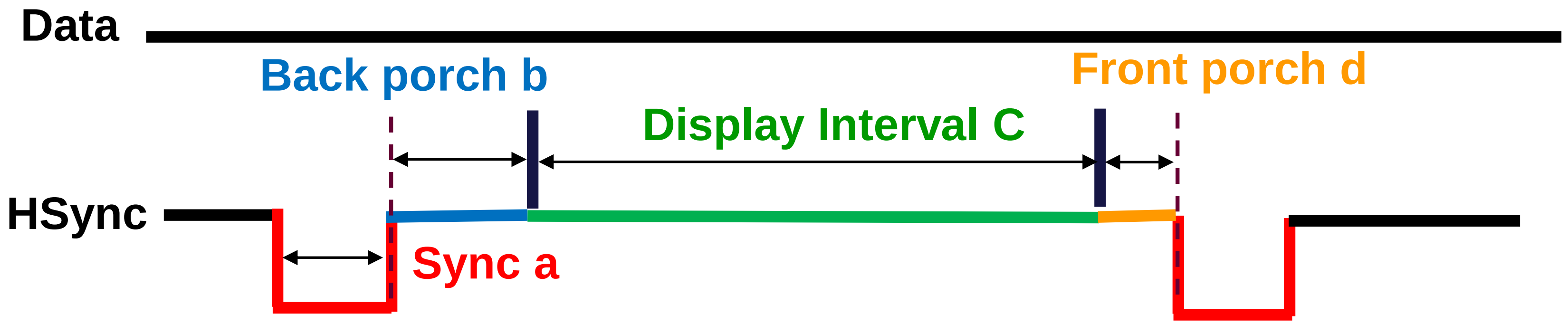
- Horizontal sync signal tells the monitor when to refresh 1 row of pixels (i.e. 640 pixels)
- When completing the refreshing process for 1 row, the beam is returning from the right most position to left most. During the return process, no pixel data is displayed



VGA horizontal sync

Time/Pixel =

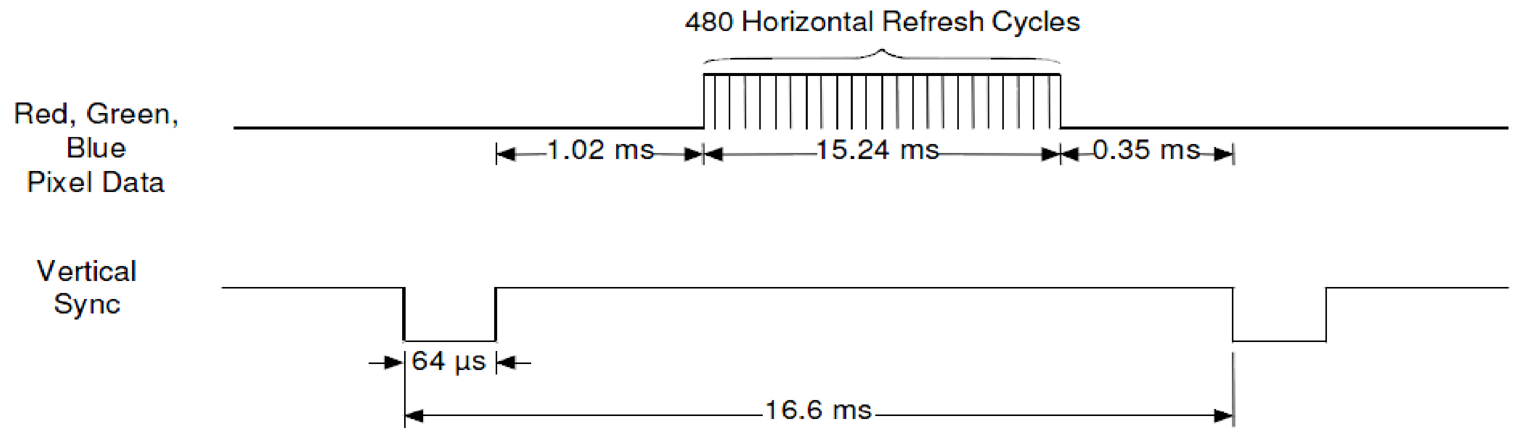
Assuming each pixel is updated on rising edge of the clock, what is the operating frequency of VGA clock?



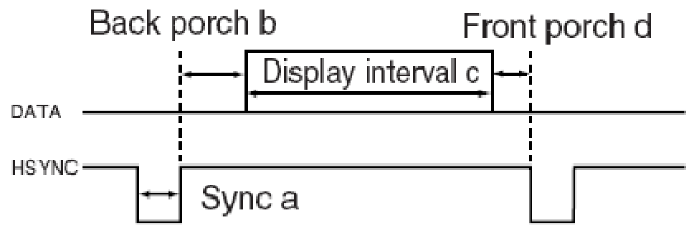
	a	b	c	d
(µs)	3.8	1.9	25.4	0.6
Pixels			640	

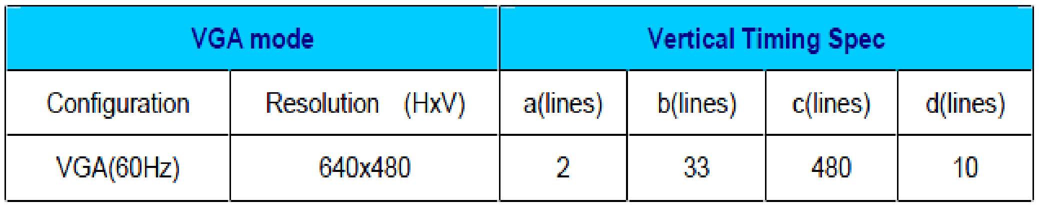
VGA Vertical Sync

- Vertical sync signal tells the monitor when to display a new image/frame
  - It is the time taken to completely refresh 480 lines of pixels



Altera VGA vertical timing specifications

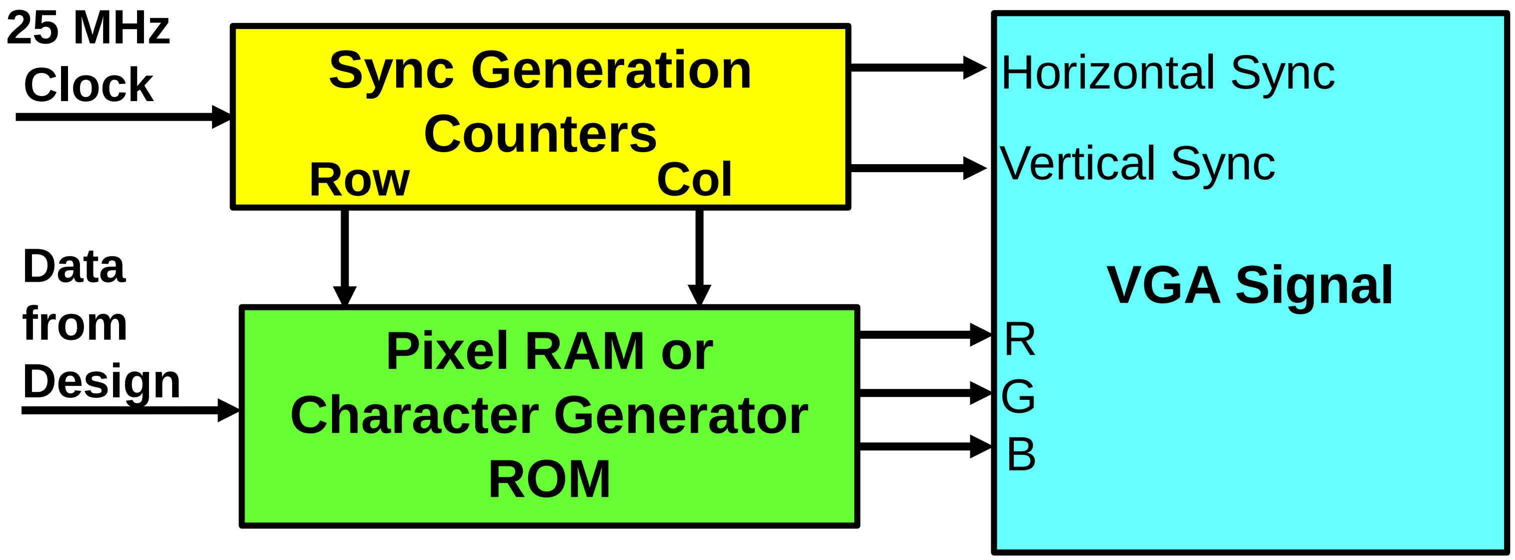




VGA mode		Vertical Timing Spec			
Configuration	Resolution (HxV)	a(lines)	b(lines)	c(lines)	d(lines)
VGA(60Hz)	640x480	2	33	480	10

Using FPGA for VGA display generation

- In order to generate the VGA signal at 25 MHz, the clock signal provided by DE0 (50MHz) needs to be halved
  - The halved clock signal can be used by counters to generate the horizontal and vertical sync signals
  - The counters also represent row and column address of a pixel, which can be used to retrieve pixel information from memory
  - For vertical sync and horizontal sync
  - Your hardware has to drive the control signals to the display and provide pixel values at the right rate

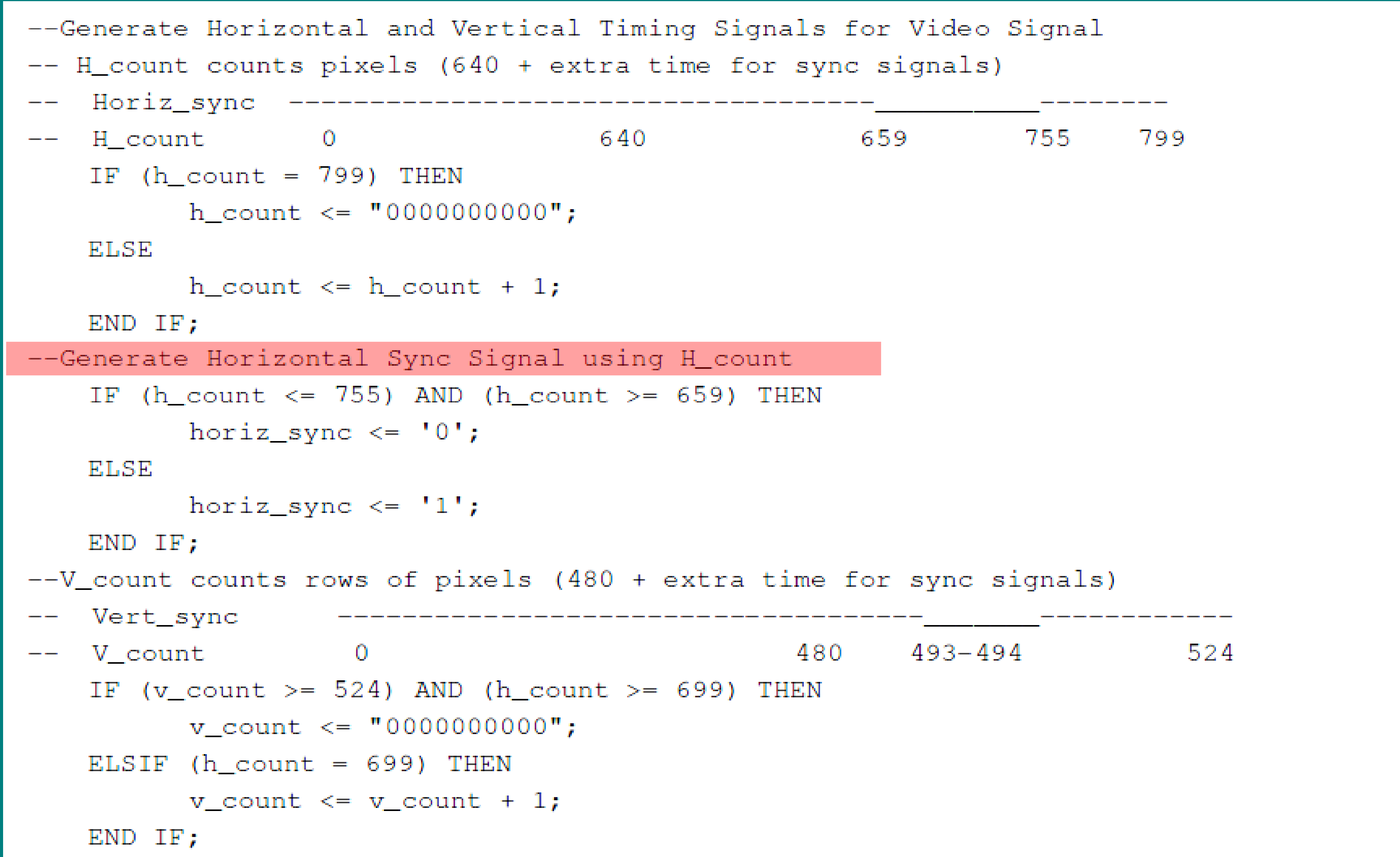


VGA sync example

```
library IEEE;
use  IEEE.STD_LOGIC_1164.all;
use  IEEE.STD_LOGIC_ARITH.all;
use  IEEE.STD_LOGIC_UNSIGNED.all;

ENTITY VGA_SYNC IS
    PORT( clock_25Mhz, red, green, blue : IN      STD_LOGIC;
          red_out, green_out, blue_out, horiz_sync_out, vert_sync_out : OUT STD_LOGIC;
                   pixel_row, pixel_column: OUT STD_LOGIC_VECTOR(9 DOWNTO 0));
END VGA_SYNC;
ARCHITECTURE a OF VGA_SYNC IS
    SIGNAL horiz_sync, vert_sync : STD_LOGIC;
    SIGNAL video_on, video_on_v, video_on_h : STD_LOGIC;
    SIGNAL h_count, v_count :STD_LOGIC_VECTOR(9 DOWNTO 0);

BEGIN
-- video_on is high only when RGB data is displayed
video_on <= video_on_h AND video_on_v;

PROCESS
BEGIN
    WAIT UNTIL(clock_25Mhz'EVENT) AND (clock_25Mhz='1');
```

VGA sync example (cont'd)

Muhammad Nadeem

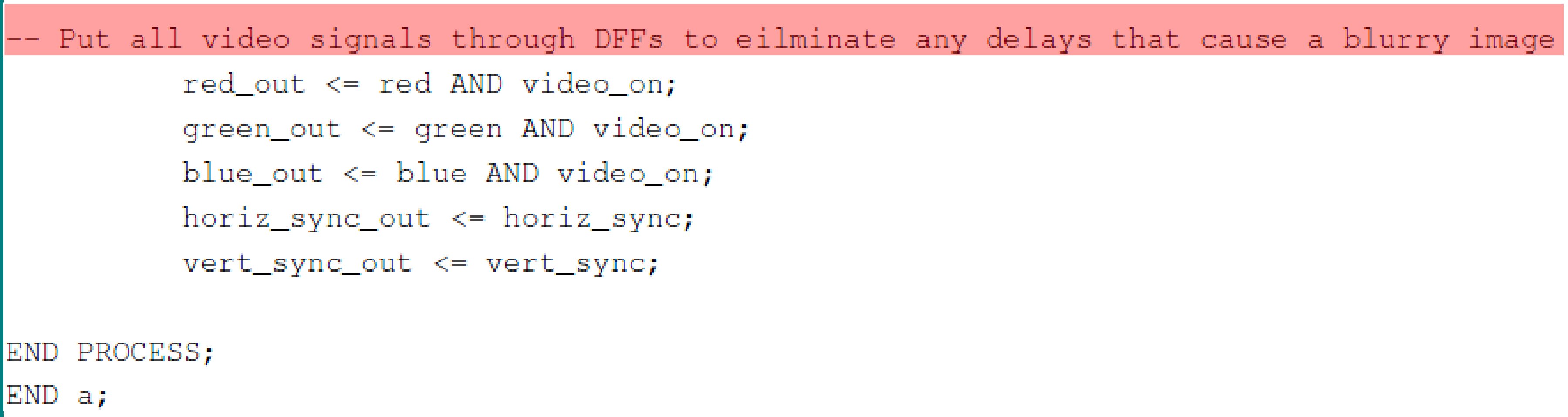


```
--Generate Horizontal and Vertical Timing Signals for Video Signal
-- H_count counts pixels (640 + extra time for sync signals)
--   Horiz_sync  ------------------------------------__________--------
--   H_count       0                640              659       755     799
    IF (h_count = 799) THEN
          h_count <= "0000000000";
    ELSE
          h_count <= h_count + 1;
    END IF;
--Generate Horizontal Sync Signal using H_count
    IF (h_count <= 755) AND (h_count >= 659) THEN
          horiz_sync <= '0';
    ELSE
          horiz_sync <= '1';
    END IF;
--V_count counts rows of pixels (480 + extra time for sync signals)
--   Vert_sync      -----------------------------------_______------------
--   V_count         0                          480    493-494          524
    IF (v_count >= 524) AND (h_count >= 699) THEN
          v_count <= "0000000000";
    ELSIF (h_count = 699) THEN
          v_count <= v_count + 1;
    END IF;
```

VGA sync example (cont'd)

```
-- Generate Vertical Sync Signal using V_count
    IF (v_count <= 494) AND (v_count >= 493) THEN
          vert_sync <= '0';
    ELSE
          vert_sync <= '1';
    END IF;
-- Generate Video on Screen Signals for Pixel Data
    IF (h_count <= 639) THEN
          video_on_h <= '1';
          pixel_column <= h_count;
    ELSE
          video_on_h <= '0';
    END IF;

    IF (v_count <= 479) THEN
          video_on_v <= '1';
          pixel_row <= v_count;
    ELSE
          video_on_v <= '0';
    END IF;
```

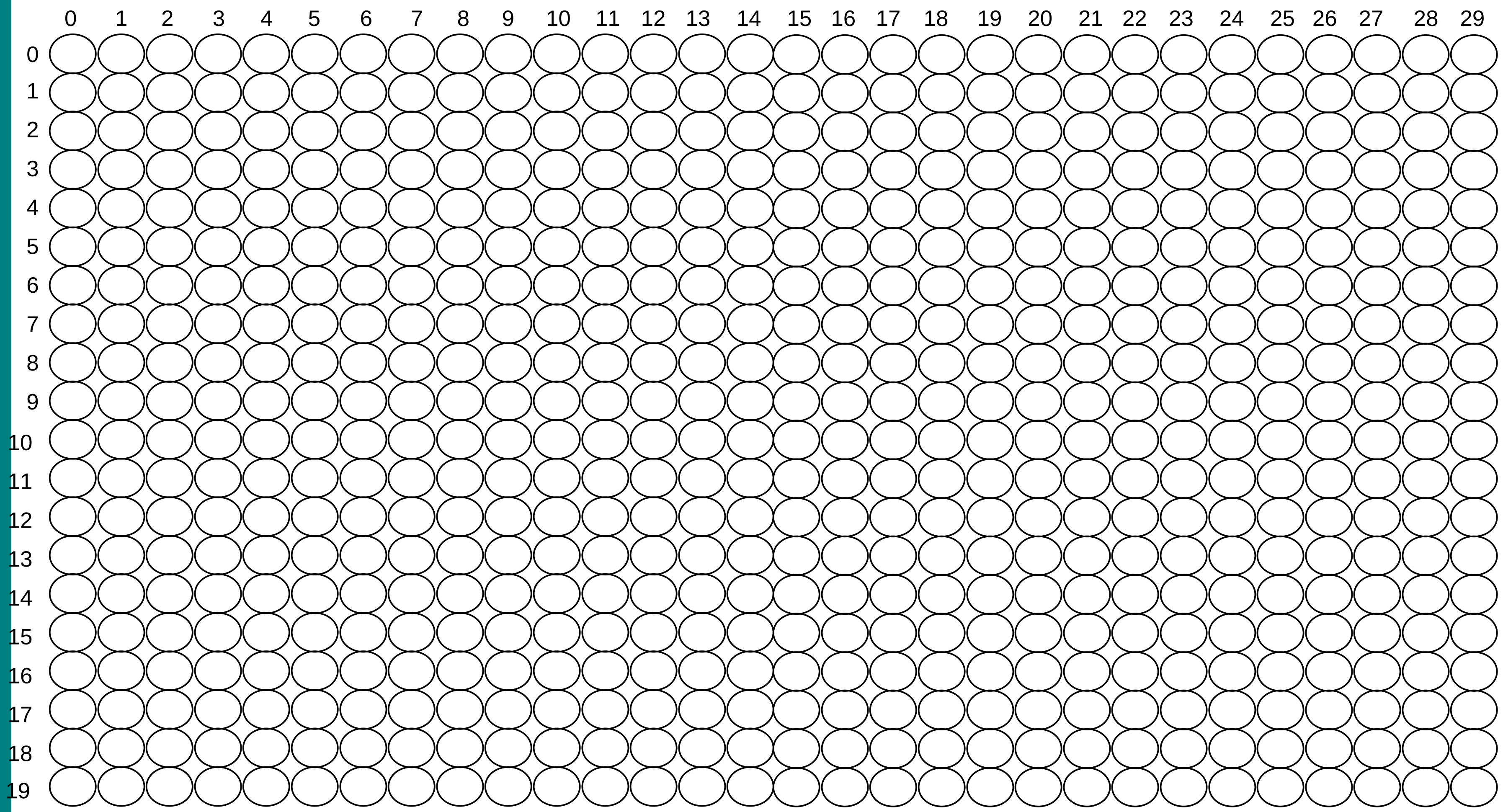
VGA sync example (cont’d)



```
-- Put all video signals through DFFs to eilminate any delays that cause a blurry image
          red_out <= red AND video_on;
          green_out <= green AND video_on;
          blue_out <= blue AND video_on;
          horiz_sync_out <= horiz_sync;
          vert_sync_out <= vert_sync;

END PROCESS;
END a;
```

Which letter will be displayed on VGA by the following VHDL statement?

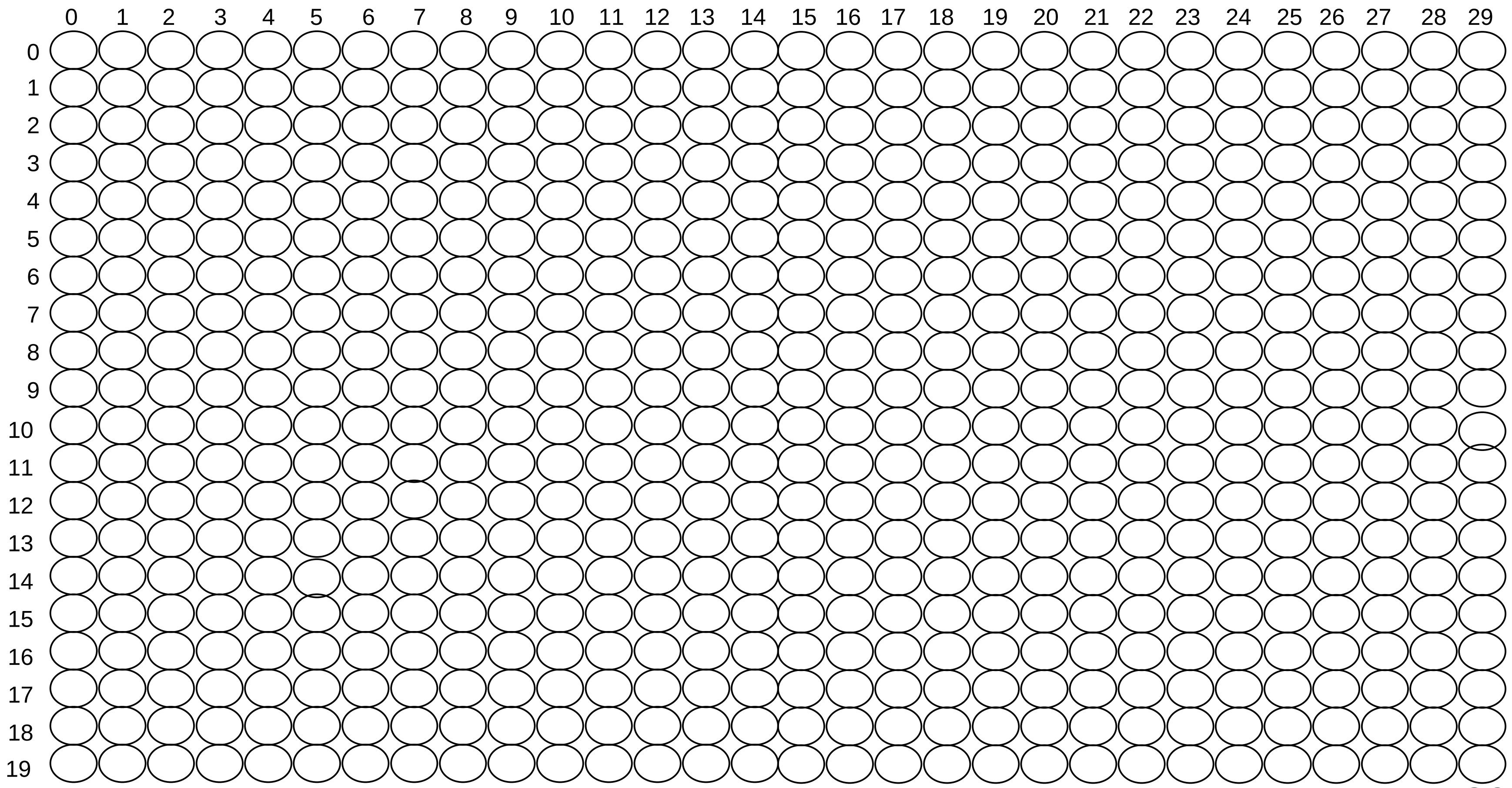
```
if (((8<row<18) and (col = 8)) or ((8<row<18) and (col = 13)) or
((row=13) and (8<col<13))) then red <= '1';  else red <= '0';
end if;
```



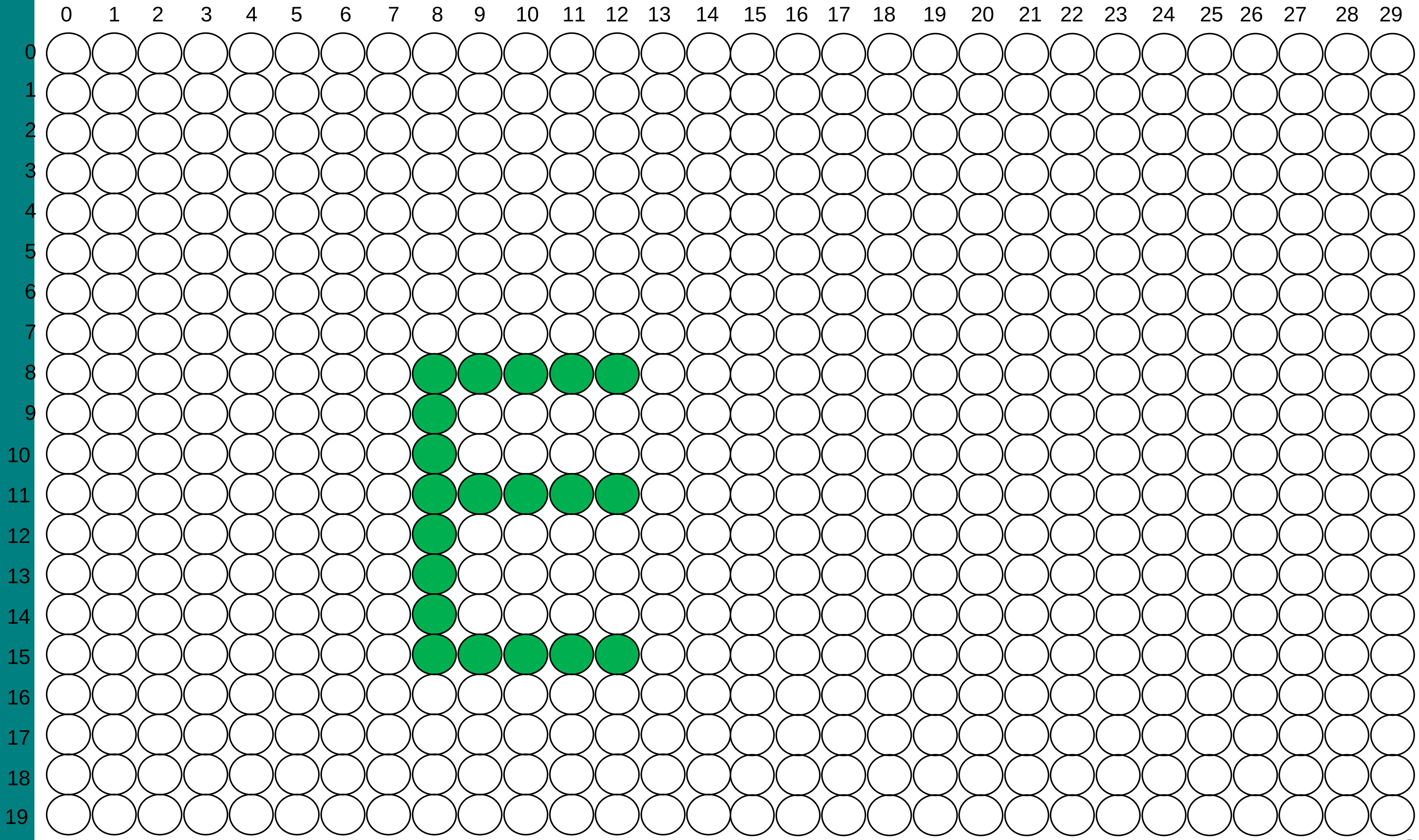
Display Pipe for two frames:

--current_position = 0; speed= 5;

```
Process(Vert_sync)          -- Vert_Sync generated at the start of new frame
        if Vert_sync'event and Vert_sync = '1' then
                current_pos  <= new_pos; end if;
end process;
new_pos <= current_pos + speed;
red <='1' when ((0 < row < 6) and (current_pos <col< current_pos+3)) else '0';
```

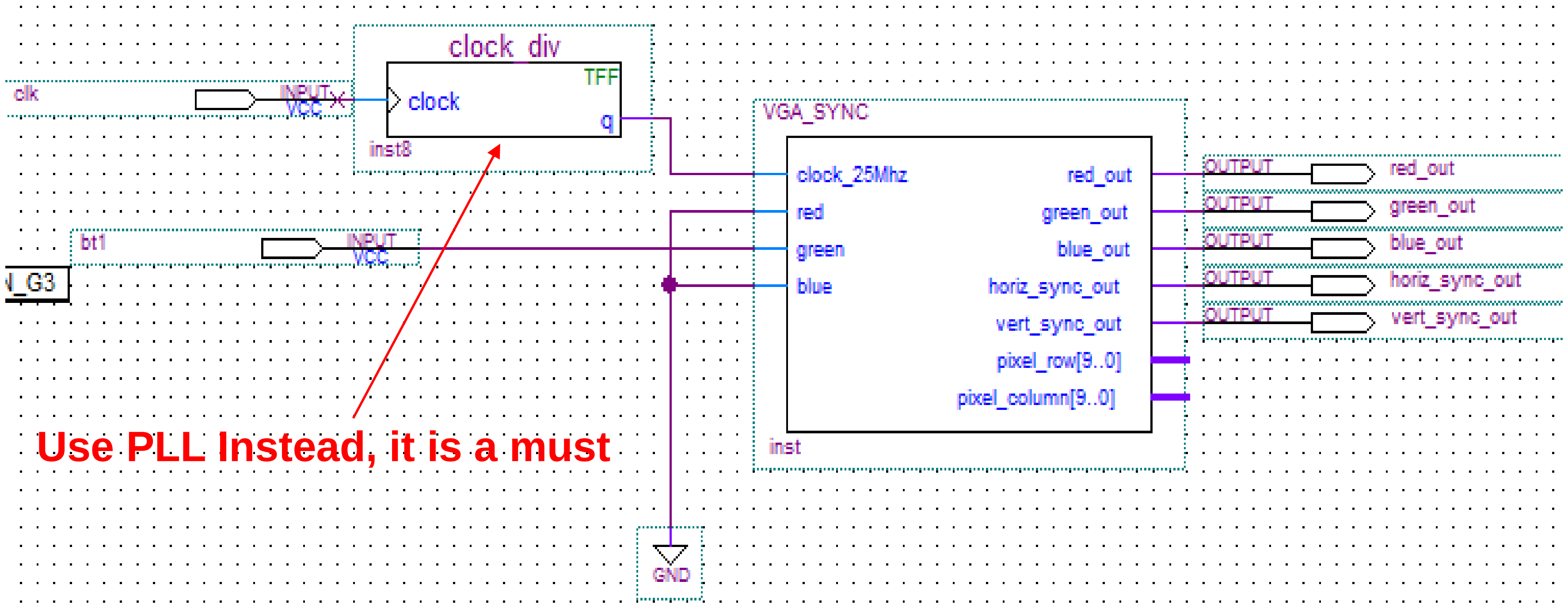


VGA Practice – Write VHDL code for displaying letter ‘E’ on the screen



VGA sync example (cont'd)

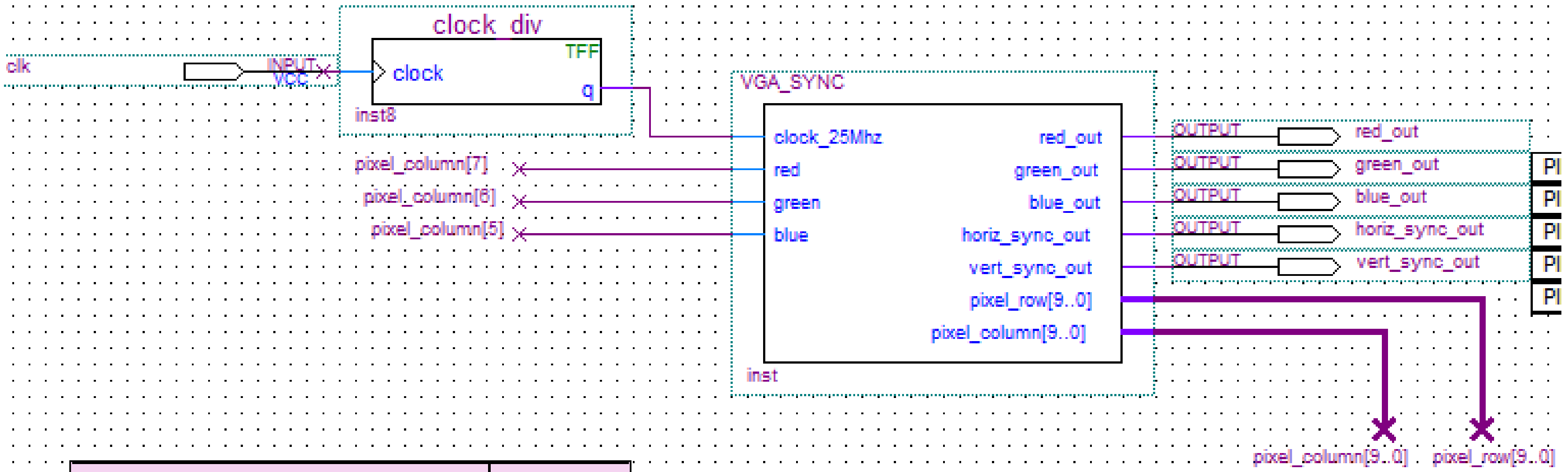
- Try the **vga_sync.vhd** file provided with the following configuration but note:
  - You need to generate a 25MHz clock signal
  - You need to do DE0 Pin assignment. A pin list miniproject.csv file is provided for you to import
  - Try this example out by connecting your DE0 board with the monitor



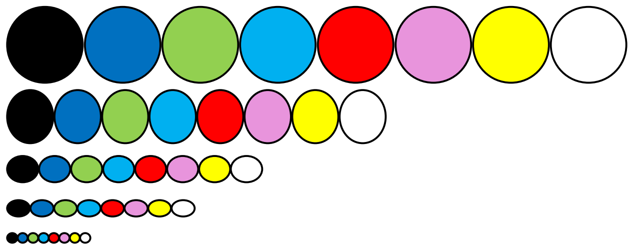
Use PLL Instead, it is a must

Colour Bar Example

- Try the vga_sync.vhd file with the following configuration
  - Since **pixel_column** bits 0-4 are not connected, the pixel colour will change every 32 pixels across the screen and it generates a sequence of colour bars in the VGA output
  - The colour bars display 8 different colours on RGB outputs

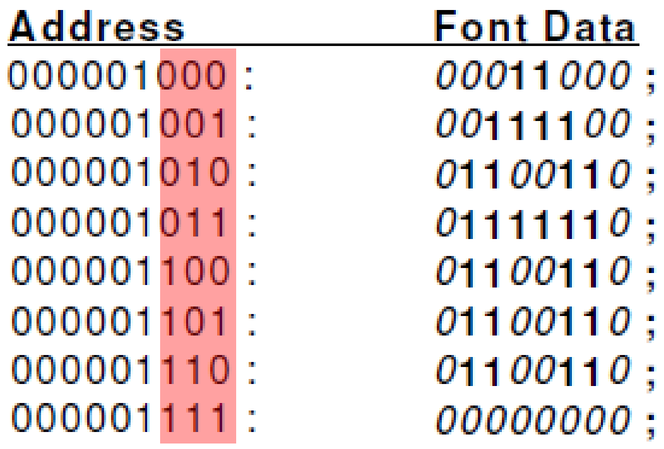


Red	Green	Blue	Colour
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White



Displaying Text Through VGA

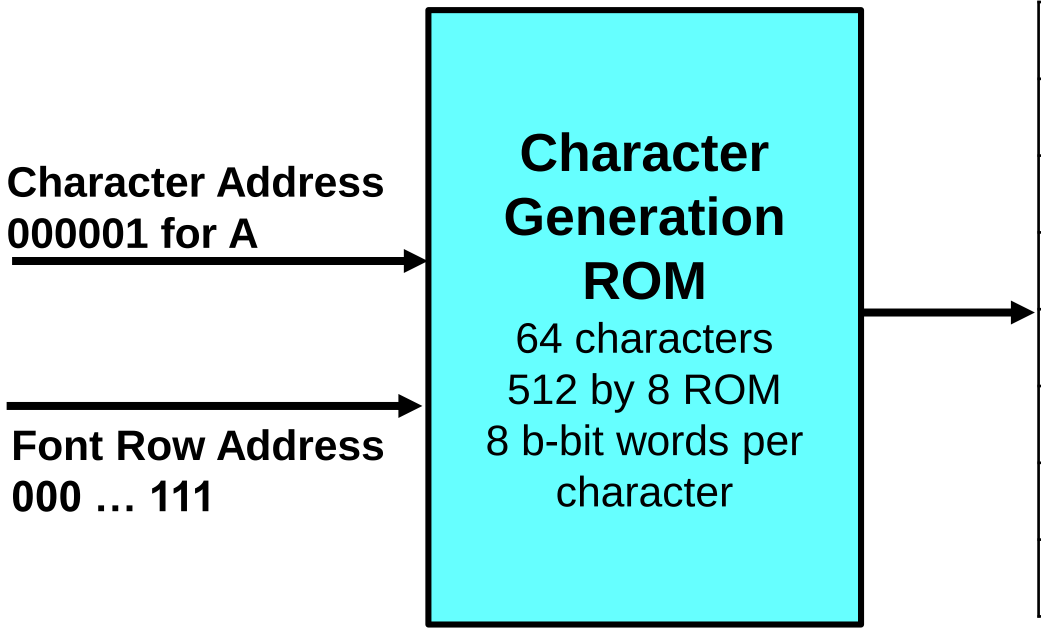
- Each different character is stored in memory as a pixel pattern (or font, a group of pixels)
- A group of character fonts that are needed are stored in ROM (or RAM) in the FPGA (can be initialised by *.mif file)
- Assuming that each character uses an 8x8 pixel array, which can be organised in 8 consecutive bytes, e.g. character A is as follows



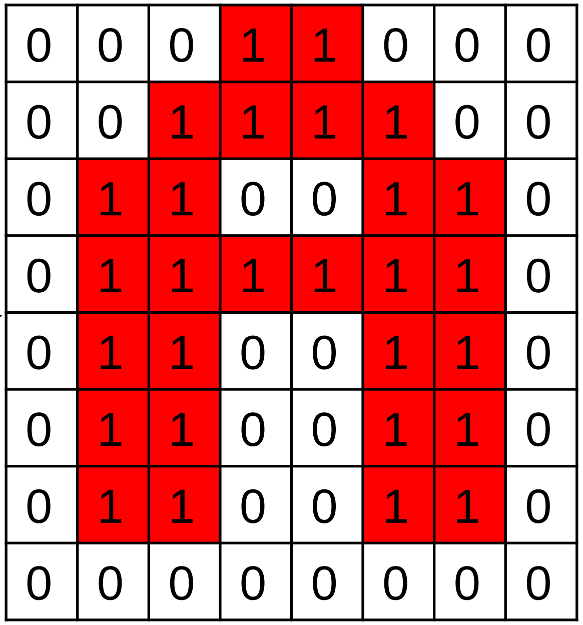
Address	Font Data
000001000 :	00011000 ;
000001001 :	00111100 ;
000001010 :	01100110 ;
000001011 :	01111110 ;
000001100 :	01100110 ;
000001101 :	01100110 ;
000001110 :	01100110 ;
000001111 :	00000000 ;

Scaling Characters

- In the previous example, each character is represented by 8 bytes
  - Character address: 64 characters require 6 bits to represent all addresses
  - Font row address: each font has 8 bytes, hence require 3 bits
- To make characters larger, each dot in the font maps to 2x2 pixel block (i.e. a single character now requires 16x16 pixels)
  - Achieved by dividing the row and column counter by 2, i.e. truncating the least significant bit



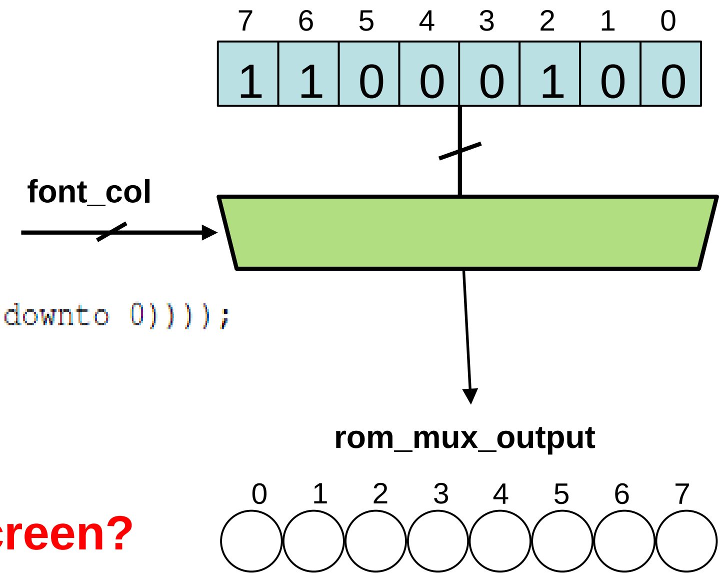
8 x 8 Font Pixel Data



0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0

Displaying text example

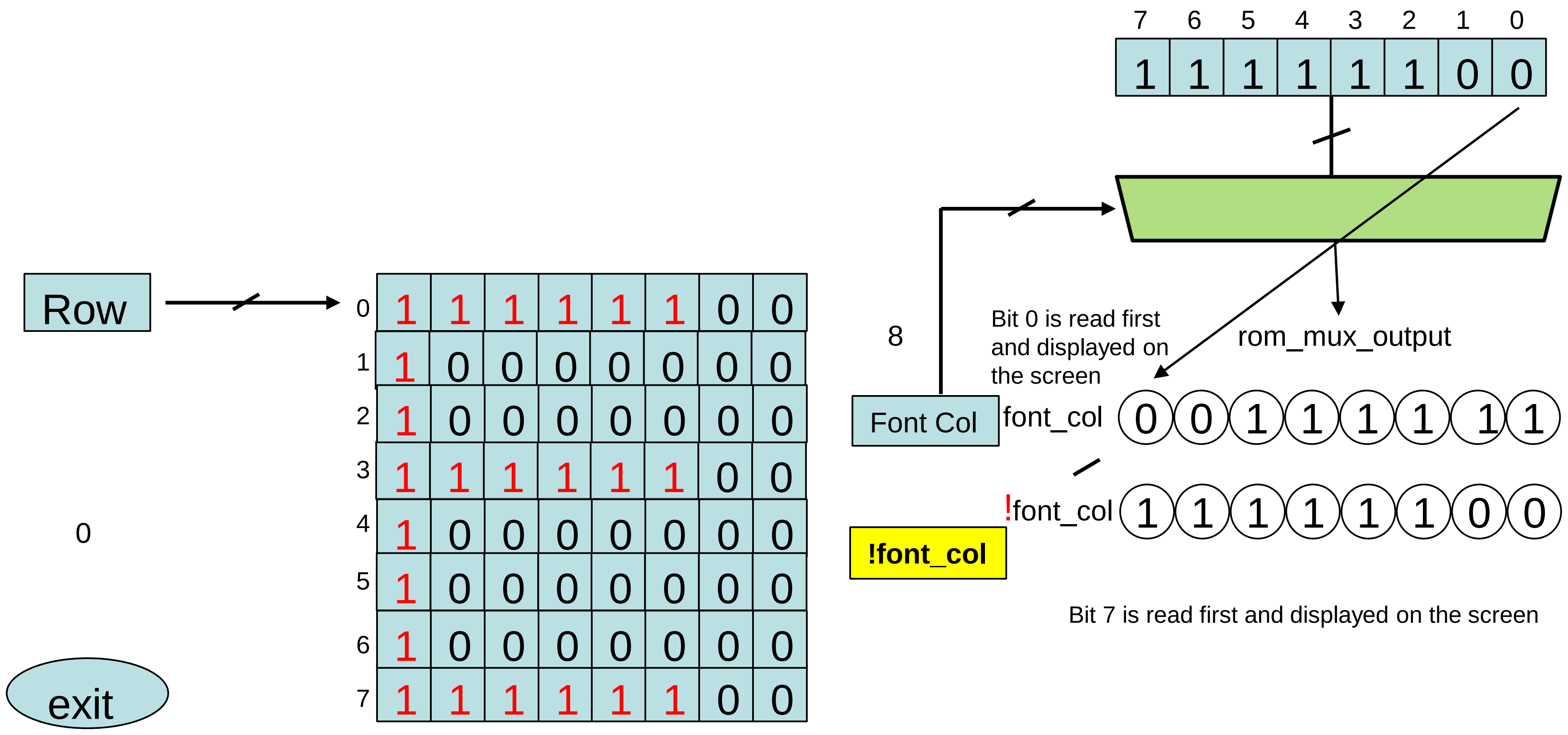
```
ARCHITECTURE a OF Char_ROM IS
   SIGNAL         rom_data: STD_LOGIC_VECTOR(7 DOWNTO 0);
   SIGNAL         rom_address: STD_LOGIC_VECTOR(8 DOWNTO 0);
BEGIN
-- Small 8 by 8 Character Generator ROM for Video Display
-- Each character is 8 8-bits words of pixel data
 char_gen_rom: lpm_rom
      GENERIC MAP ( lpm_widthad => 9, lpm_numwords => 512,
        lpm_outdata => "UNREGISTERED", lpm_address_control => "UNREGISTERED",
-- Reads in mif file for character generator font data
         lpm_file => "tcgrom.mif", lpm_width => 8)
      PORT MAP ( address => rom_address, q => rom_data);
rom_address <= character_address & font_row;
-- Mux to pick off correct rom data bit from 8-bit word
-- for on screen character generation
rom_mux_output <= rom_data ( (CONV_INTEGER(NOT font_col(2 downto 0))));

END a;
```



Which bit should be displayed first on VGA screen?

Which bit is displayed first on VGA screen?

Text Display

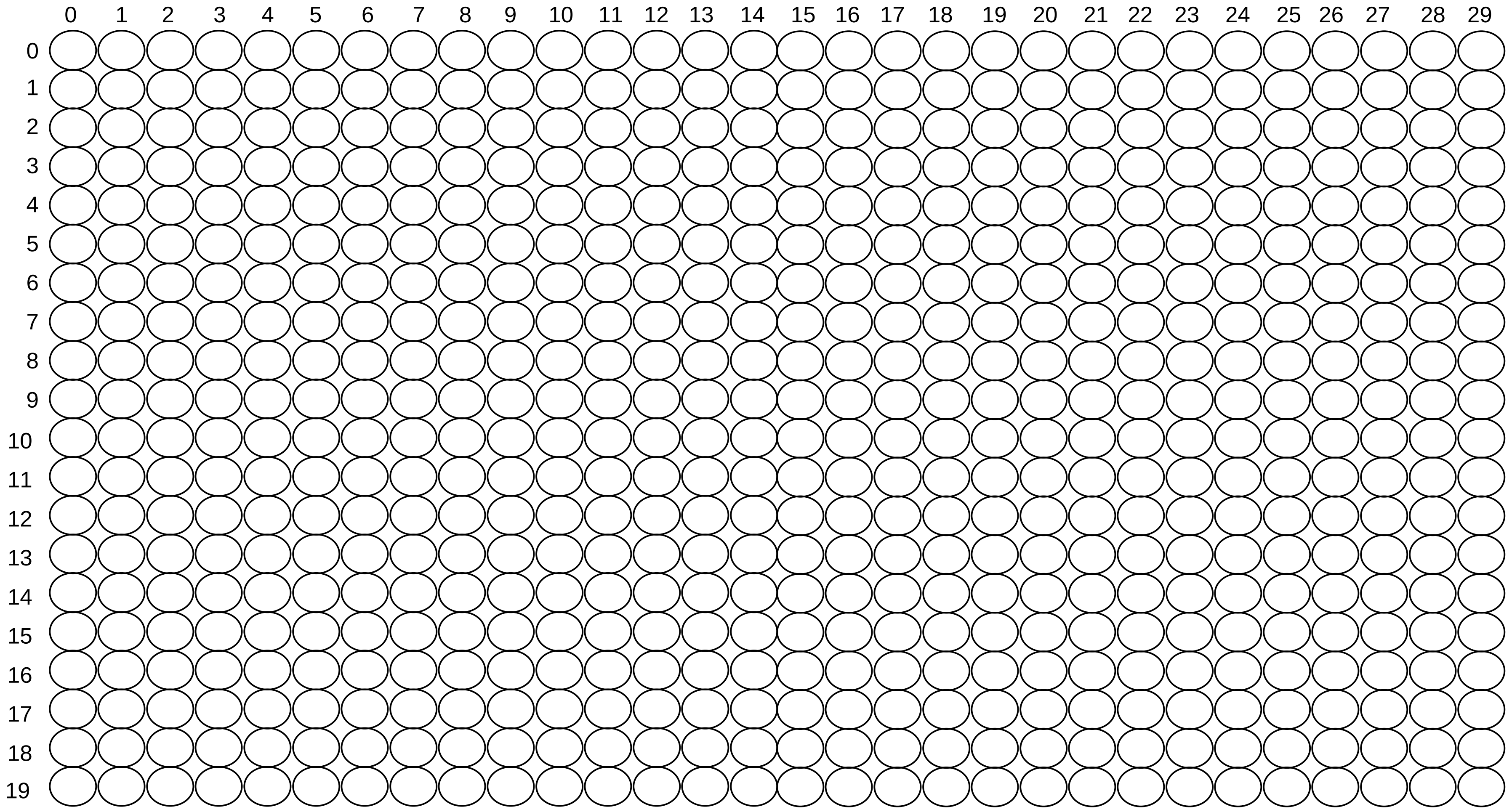


0	1	1	1	1	1	1	0	0
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0
3	1	1	1	1	1	1	0	0
4	1	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0
7	1	1	1	1	1	1	0	0

Display the letter on the following screen assuming that font_row is connected to pixel_row [2..0] and font_col is connected to pixel_col [3..1].

font_row [2..0] = pixel_row[2..0]=v_count[2..0] - - v_count is 9-bit

font_col [2..0] = pixel_col[3..1]=h_count[3..1] - - h_count is 10-bit



Character ROM address map

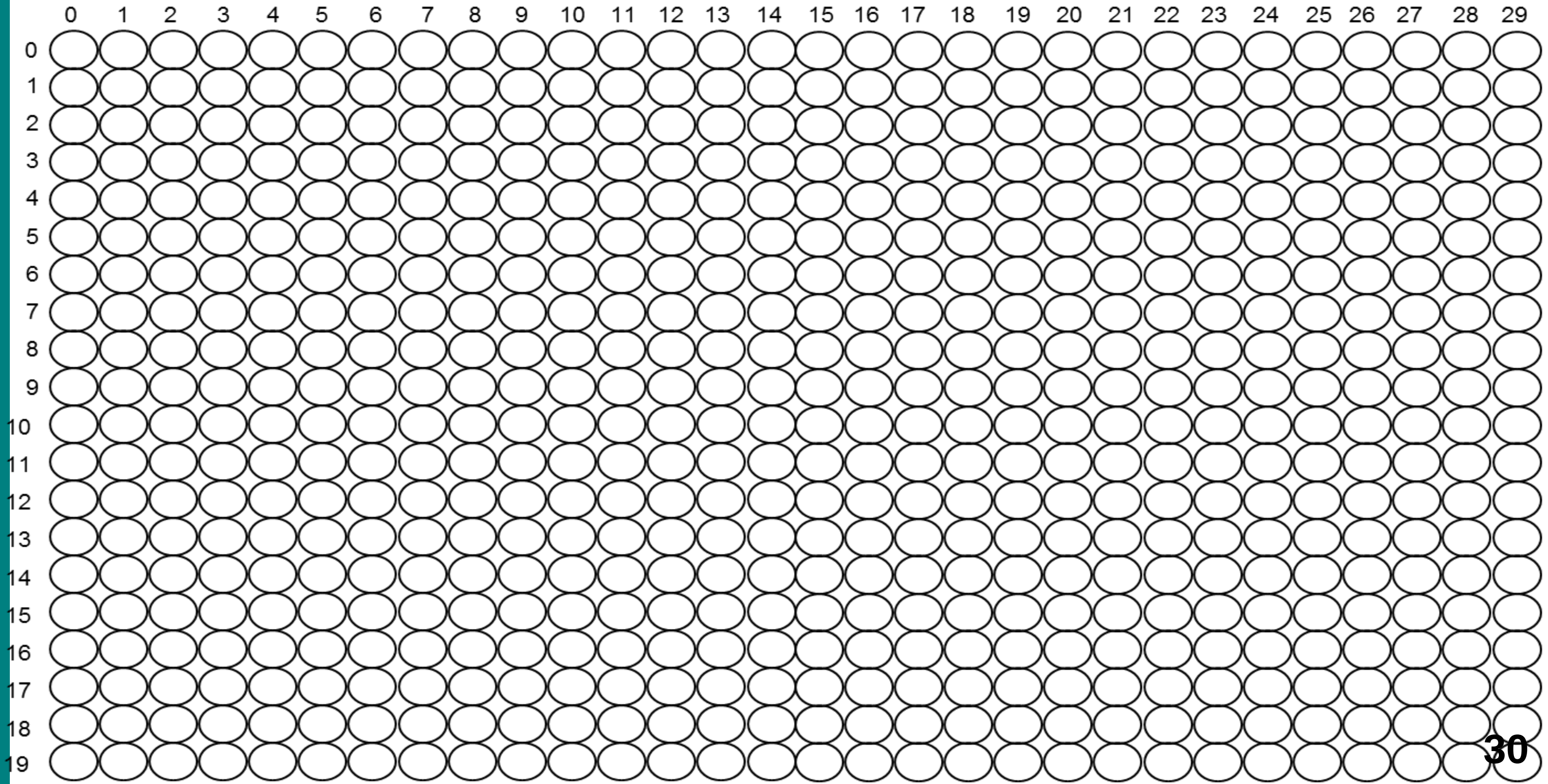
CHAR	ADDRESS	CHAR	ADDRESS	CHAR	ADDRESS	CHAR	ADDRESS
@	*00*	P	*20*	Space	*40*	0	*60*
A	*01*	Q	*21*	!	*41*	1	*61*
B	*02*	R	*22*	"	*42*	2	*62*
C	*03*	S	*23*	#	*43*	3	*63*
D	*04*	T	*24*	$	*44*	4	*64*
E	*05*	U	*25*	%	*45*	5	*65*
F	*06*	V	*26*	&	*46*	6	*66*
G	*07*	W	*27*	'	*47*	7	*67*
H	*10*	X	*30*	(	*50*	8	*70*
I	*11*	Y	*31*	)	*51*	9	*71*
J	*12*	Z	*32*	*	*52*	A	*72*
K	*13*	[	*33*	+	*53*	B	*73*
L	*14*	Dn Arrow	*34*	,	*54*	C	*74*
M	*15*	]	*35*	-	*55*	D	*75*
N	*16*	Up Arrow	*36*	.	*56*	E	*76*
O	*17*	Lft Arrow	*37*	/	*57*	F	*77*

Which character is displayed by the address $(0111000100101)_2$

Muhammad Nadeem

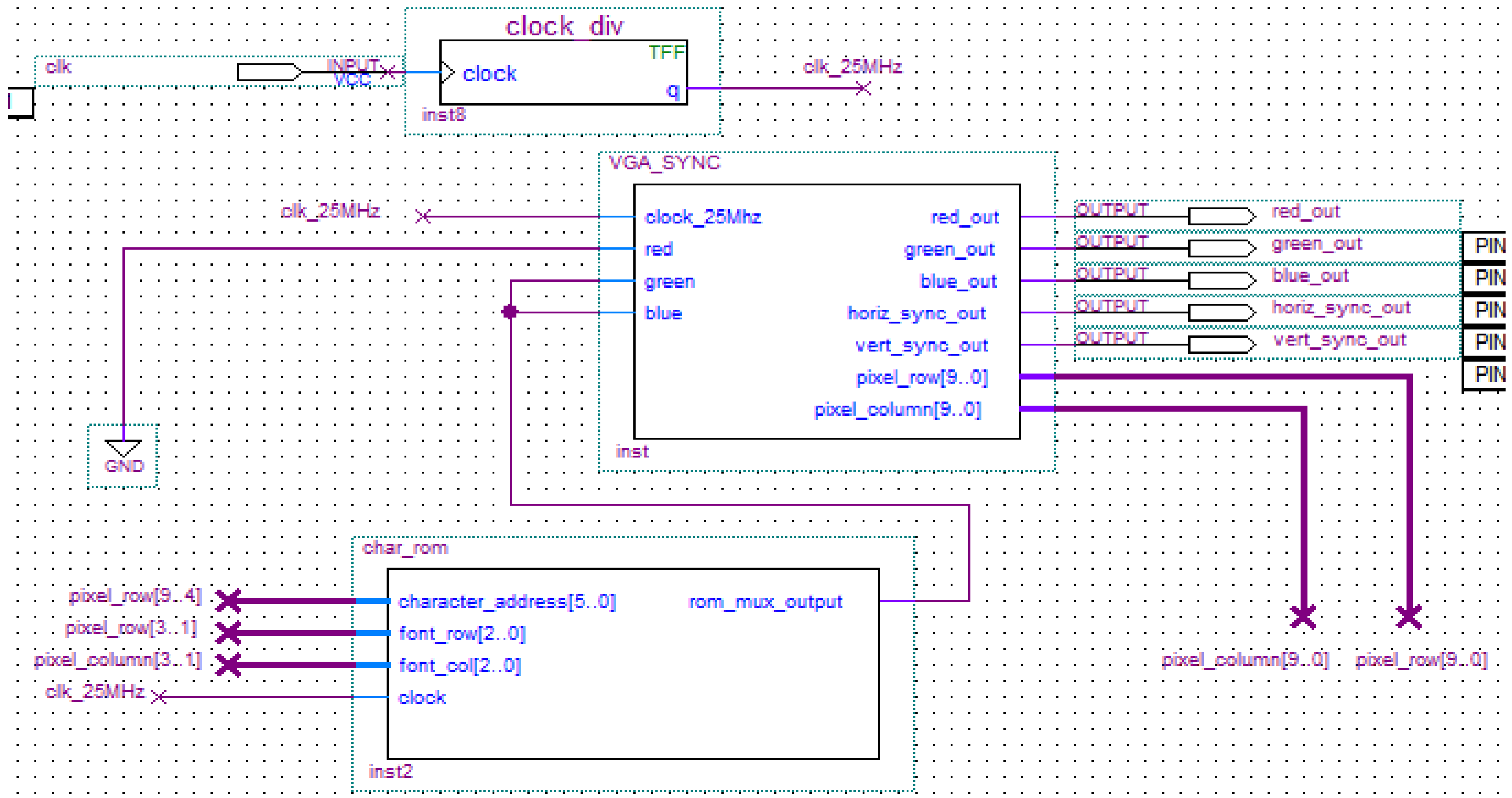
Class Activity

Following is a part of character rom showing a character stored in it starting from address 8. Consider a VGA display with pixels being scanned from left to right and rows from top to bottom. The scanning starts from top left corner with row=0 and col=0 and letter is written at the same location.
font_row [3..1] and font_col [2..0] are used.



Text Display Example

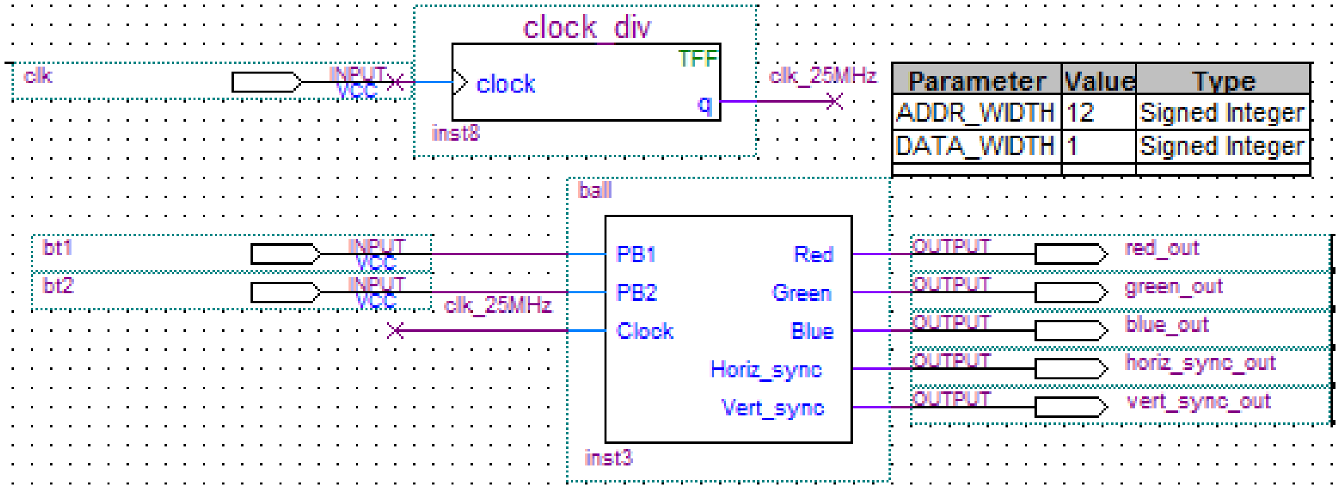
- Try the *vga_sync.vhd* and char_rom.vhd provided together with the memory initialisation file, *TCGROM.mif*, in the following configuration



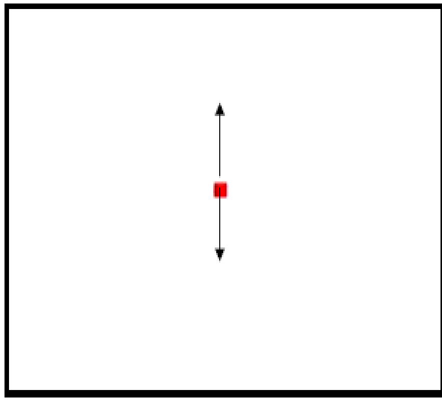
Bouncing Ball Example

Muhammad Nadeem

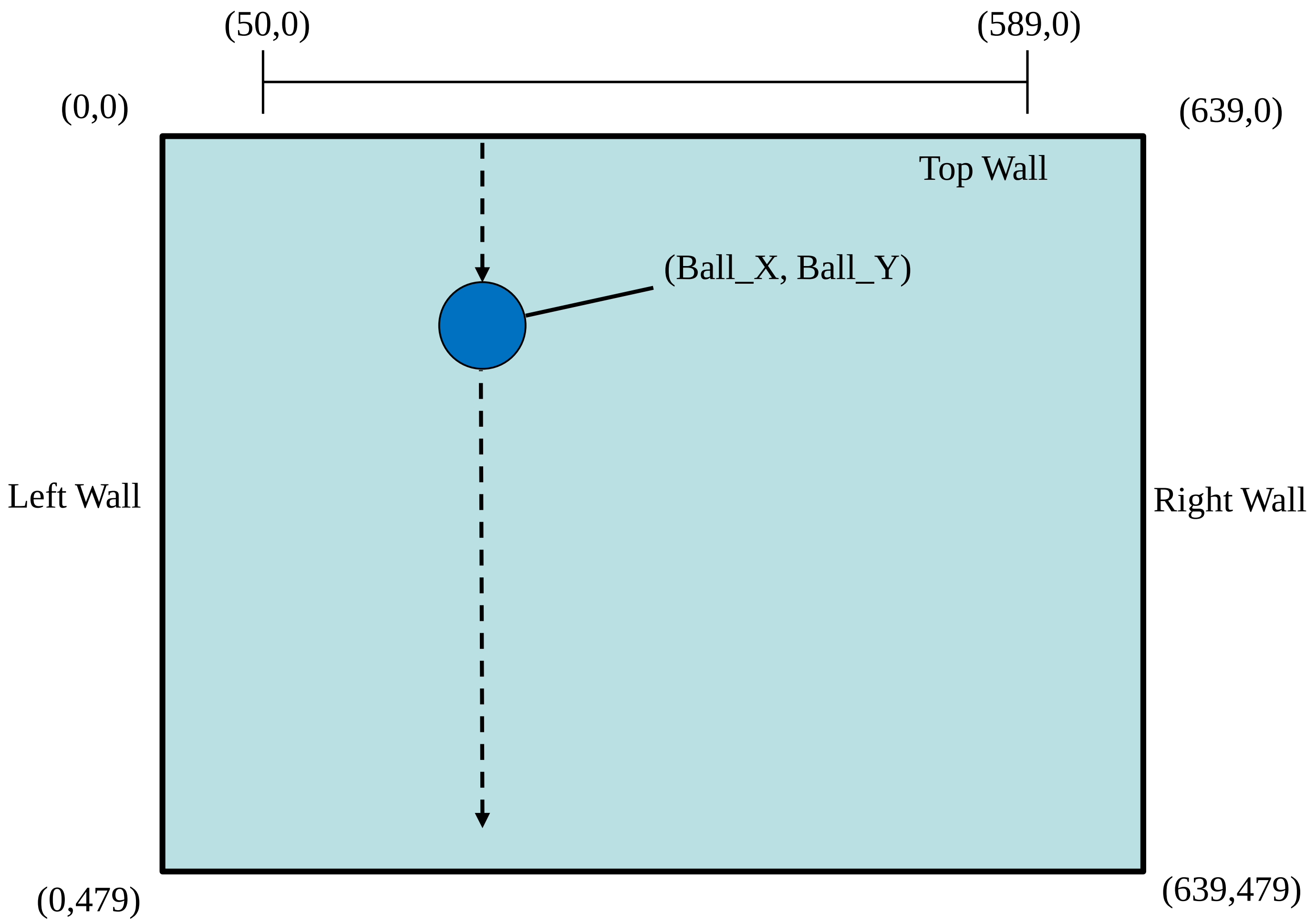
- Try *the ball.vhd* that displays a small square ball bounces up and down on the screen
- *vga_sync* is included to generate sync signal and pixel address info
- RGB_Display process generates a red ball on the white background with a Ball_on signal
- Ball_X_pos and Ball_Y_pos are the current address of the centre of the ball
- Move_Ball process moves the ball a few pixels and checks for bouncing effect of the wall



Parameter	Value	Type
ADDR_WIDTH	12	Signed Integer
DATA_WIDTH	1	Signed Integer



Bouncing Ball Example


(50,0)
(589,0)
(0,0)
(639,0)
Top Wall
(Ball_X, Ball_Y)
Left Wall
Right Wall
(0,479)
(639,479)