Intoduction to Mini Project - Part 1

Maryam Hemmati

Department of Electrical, Computer and Software Engineering
University of Auckland

email: m.hemmati@auckland.ac.nz

COMPSYS 305-Digital Systems Design

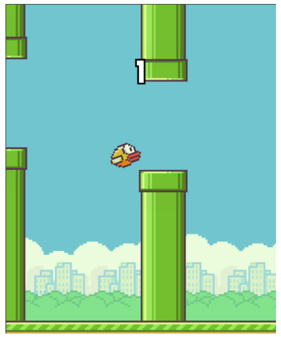
31 March 2022

Content

Mini Project Objective

The goal of the mini project is to design a simple game console.

- The game is Flappy Bird.
- The game is implemented on DE0 board.
- The game is controlled and played using
  - A PS/2 mouse
  - DIP switches on the DE0 board
  - Push-buttons on the DE0 board
- The game is displayed on a VGA screen.
  - with a resolution of **640x480** pixels



Mini Project Objective

Game Description

- The bird can move up or down
  - It is controlled by a PS/2 mouse.
  - If the bird is not flapping, it will free-fall towards the ground.
  - The bird must not touch anything when flying, otherwise, it will lose life points.
- The game may consist of different types of obstacles and gifts
  - pipes
  - dollars, medicine boxes, special flying abilities
- The screen must be kept in motion from the right-hand side to the left-hand side.
  - The speed increases with the game level
- The level of difficulty can be controlled by other criteria
  - The types of obstacles

Mini Project Objective

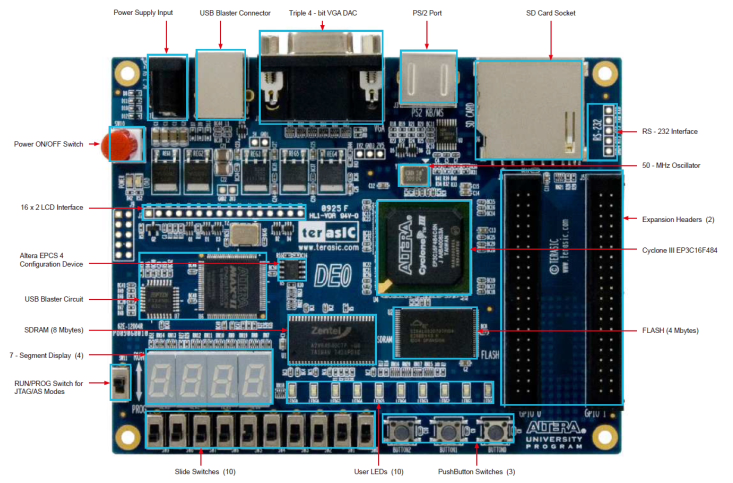
Game Modes

- **Training Mode**
  - Allows the player to practice at the lowest game level.
  - Will continue for a specific time.
- **Single-player Game Mode**
  - The game will proceed to more advanced levels following certain criteria.
  - The time, distance, or the number of obstacles passed could be used as such criteria.

The game mode can be determined by using a DIP switch on the console or through a selection on the welcome screen.

DE0 Board

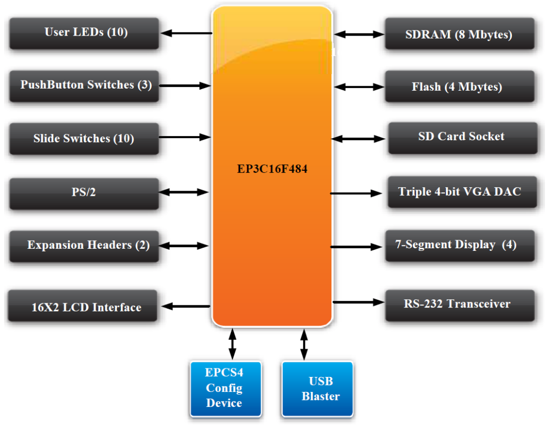
The hardware platform that you will use for implementing the game console is Terasic DE0 board.



DE0 Board

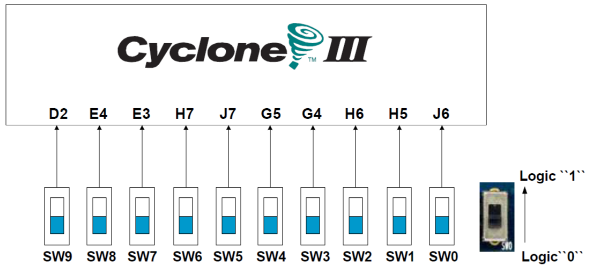
DE0 board includes Altera **Cyclone III 3C16** FPGA device

- All the connections are made through the Cyclone III FPGA device.
  - Gives the flexibility to the user to configure the FPGA to implement any system design.
- The DE0 board includes a **50 MHz** clock signal.



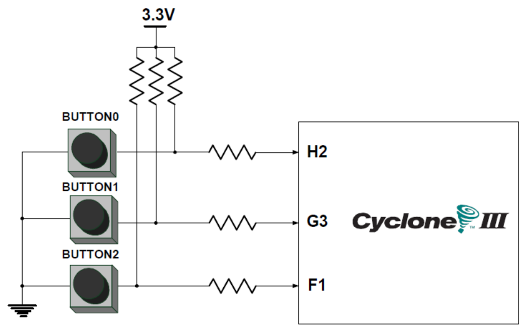
DE0 Board - Switches

- There are 10 slide switches (sliders) on the DE0 board.
- These switches are used as level-sensitive data inputs to a circuit.
  - When a switch is in the **DOWN** position it provides a **low logic level**.
  - When the switch is in the **UP** position it provides a **high logic level**.
- The FPGA pins connected to these switches (and any other I/O) should be defined in the project through pin assignment.



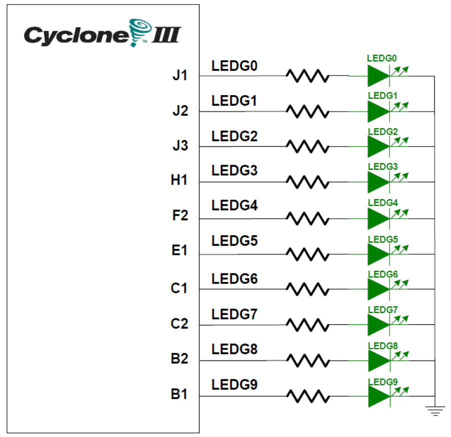
DE0 Board - Pushbuttons

- The DE0 board provides three pushbutton switches.
- BUTTON0, BUTTON1, and BUTTON2 are connected directly to the Cyclone III FPGA as an input.
  - Each button provides a **high** logic level when it is **not pressed**.
  - The button provides a **low** logic level when it is **pressed**.



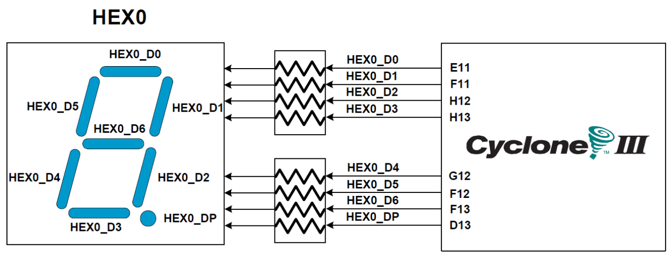
DE0 Board - LEDs

- There are 10 user-controllable LEDs on the DE0 board.
- Each LED is driven directly by a pin on the Cyclone III FPGA.
  - Driving associated pin to a **high** logic level turns the LED **on**.
  - Driving the pin **low** turns it **off**.



DE0 Board - Seven Segments

- The DE0 board has four 7-segment displays.
- They are connected to pins on the Cyclone III FPGA.
  - Applying a **low** logic level to a segment causes it to **light up**.
  - Applying a **high** logic level turns it **off**.
- Each segment in a display is identified by an index from 0 to 6.



DE0 Board - FPGA Configuration

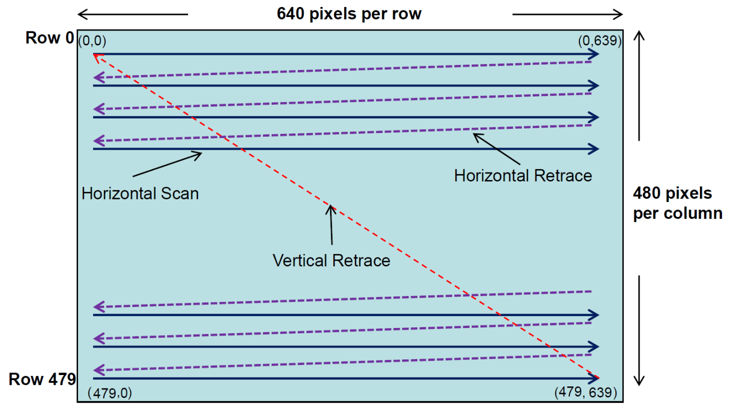
There are two different modes for configuring the **Cyclone III FPGA** on DE0 board.

- **JTAG programming**
  - The configuration bit stream is downloaded **directly** into the **Cyclone III FPGA**.
  - The FPGA will retain this configuration as long as power is applied to the board.
  - The configuration is lost when the power is turned off.
- **Active Serial programming**
  - The configuration bit stream is downloaded into the Altera **EPCS4 serial EEPROM chip**.
  - It provides non-volatile storage of the bit stream.
  - When the board is turned on, the configuration data in the EPCS4 device is automatically loaded into the Cyclone III FPGA.

Configuration bit stream is downloaded into the board through the USB Blaster.

VGA Interface

- **VGA (Video Graphics Array)** is a popular display standard developed by IBM and introduced in 1987.
- The resolution of the VGA screen can vary but a standard default size is **640x480 pixels**.
- The screen refreshes the display from left to right, top to bottom.



VGA Interface

Image on VGA screen is displayed by turning the pixels ON and OFF.

- Video signal must redraw the entire screen 60 times per sec (**60Hz**) 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.
  - Each pixel takes 40ns at 25MHz pixel rate.

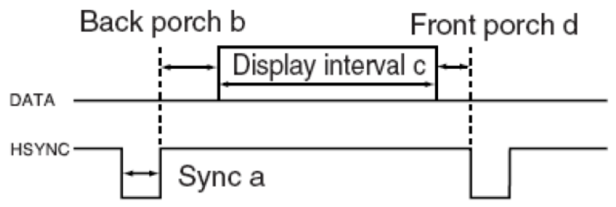
VGA Interface

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.

VGA Interface - Horizontal Synchronisation

- **Horizontal sync** signifies the end of one row of data (i.e. 640 pixels) and the start of the next.
  - The data (RGB) inputs on the monitor must be off for a time period called the **back porch** (b) after the hsync pulse occurs.
  - During the data **display interval** (c) the **RGB** data drives each pixel in turn across the row being displayed.
    - When data display interval is finished, the beam returns from the right most position to left most. During the return process, no pixel data is displayed.
  - **Front porch** (d) is a time period where the RGB signals must again be off before the next hsync pulse can occur.



VGA Interface - Horizontal Synchronisation

- **Horizontal sync** (a) corresponds to 96 pixels.
- **Back porch** (b) corresponds to 48 pixels.
- **Display interval** (c) corresponds to 640 pixels.
- **Front porch** (d) corresponds to 16 pixels.

VGA horizontal timing specification

VGA mode		Horizontal Timing Spec				
Configuration	Resolution(HxV)	a(us)	b(us)	c(us)	d(us)	Pixel clock(Mhz)
VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25 (640/c)

VGA Interface - Vertical Synchronisation

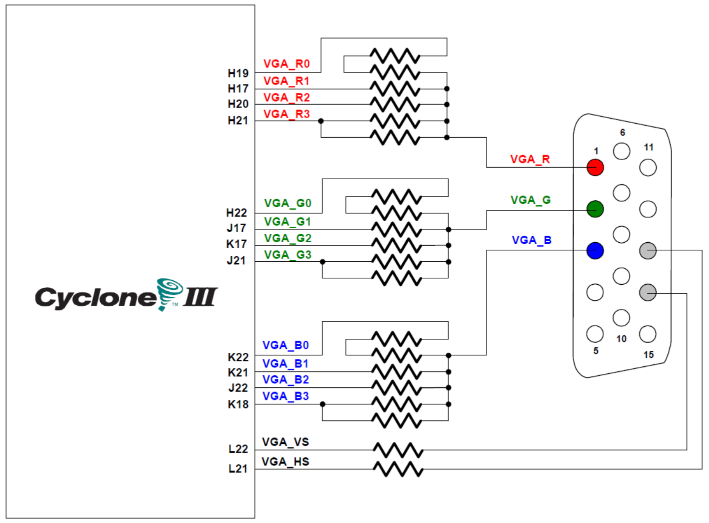
Vertical sync pulse signifies the end of one frame and the start of the next, and the data refers to the set of rows in the frame.

- **Vertical sync** (a) corresponds to 2 lines.
- **Back porch** (b) corresponds to 33 lines.
- **Display interval** (c) corresponds to 480 lines.
- **Front porch** (d) corresponds to 10 lines.

VGA vertical timing specification

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

VGA Interface - DE0 Board

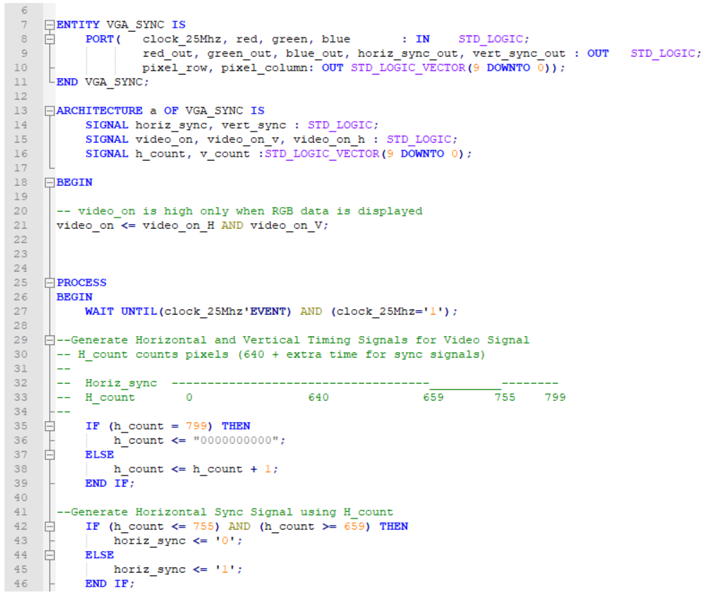


VGA Interface - VGA_Sync Component

We need a component to drive the control signals to the display and provide pixel values at the right rate.

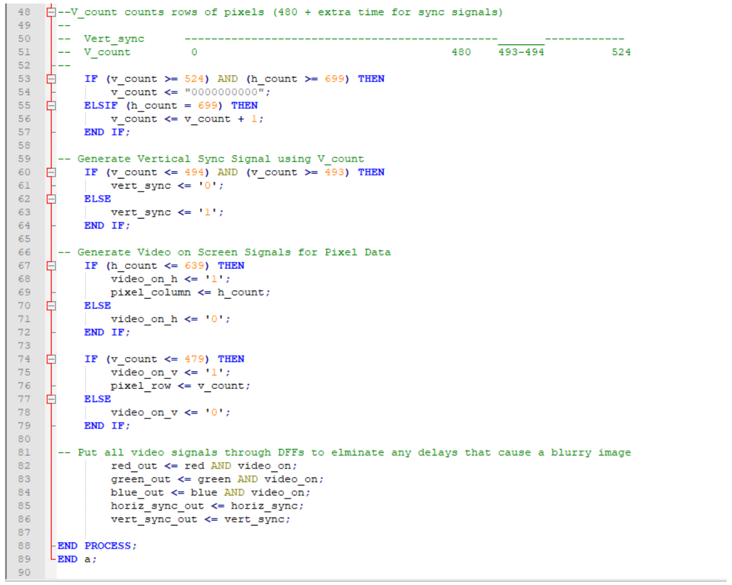
- In order to generate the VGA signal at 25 MHz, the clock signal provided by DE0 (50MHz) needs to be halved.
- 25MHz clock signal can be used by counters to generate the horizontal and vertical sync signals.
- The counters also represent row and column address of a pixel, which can be used by other components to retrieve pixel information.

VGA Interface - VGA_Sync Component



```vhdl

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');

--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;
```

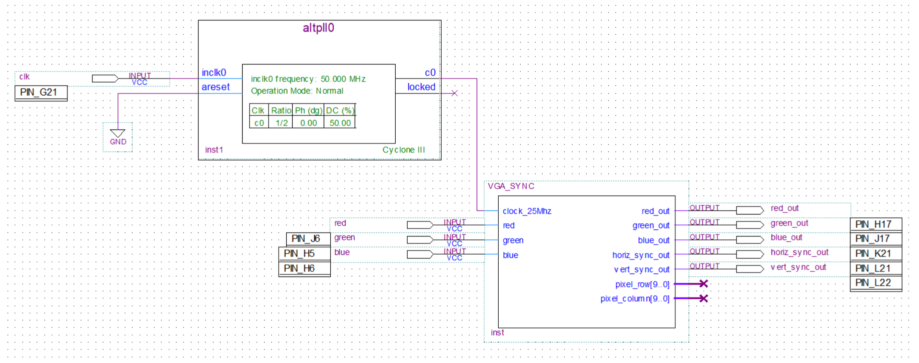
VGA Interface - VGA_Sync Component



```vhdl
--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;

-- 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;

-- Put all video signals through DFFs to elminate 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;

```

VGA Interface - Example

Try this simple example and see how you can change the background colour on your VGA screen by using three switches on the DE0 board:



Summary

- We discussed mini project and its objective.
- We talked about DE0 board and its interfaces.
- We looked at VGA interface, its signals, and the component that drives these signals from FPGA side.

Acknowledgment

- Some figures/notes are taken from or inspired by the
  - CS305 Lecture notes by Muhammad Nadeem, 2019