**COL215**

Pratik Pranav 2018CS10368

Sarthak Behera 2018CS0384

BUS/METRO ROUTE DISPLAY

**October 25, 2019**

# Overview

Starting from a simple metro route from IIT Delhi Metro Station to Hauz Khas Metro Station to much more complicated route from Delhi to Bengaluru by Bus, Maps have played significant role navigating through cities, states and even countries sometimes. We have tried to make a user input specific route display system in which the user could give inputs about the starting and the ending location, and the display will show the most efficient path between those two locations.

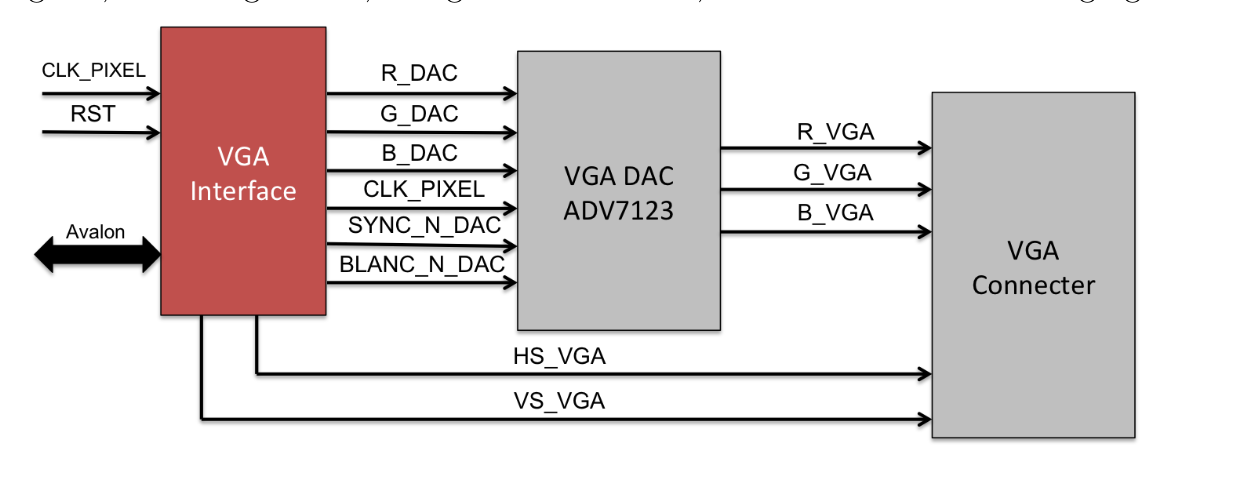
# Requirements

1. **CRT Display:** It should be compatible with VGA and can be transformed to resolution of 640x480 pixels.
2. **4x4 Keypad (PMod):** For taking user input.
3. **Basys3 FPGA Board:** For providing VGA port and basic Input/Output.

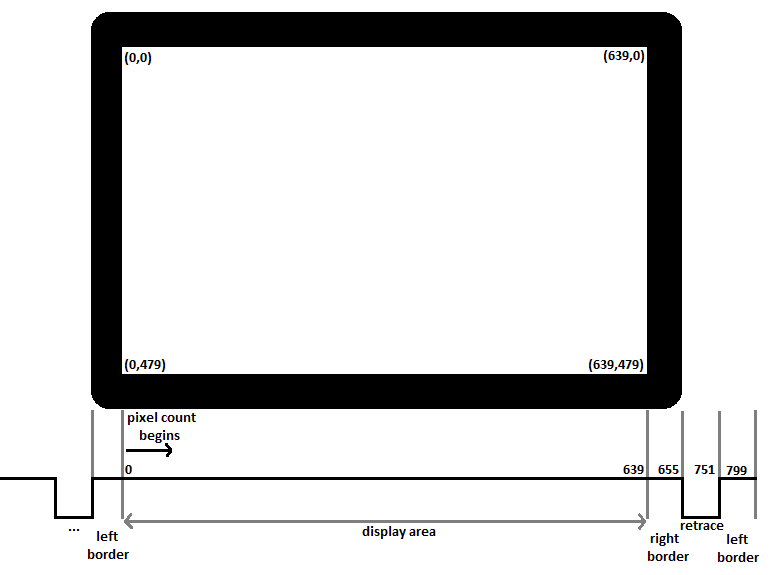
# Main Component:

## VGA

The main display of this project uses a vga to illustrate the bus/metro route of a city. This is facilitated by the vga port of the basys3 board and the user specifies the source and the destination in that particular city. Each pixel on the monitor has a red, green, and blue color component. The red, green, and blue color signals from the VGA port to the monitor are analog and are generated by resistor ladder DACs on the FPGA board that receive input from dedicated FPGA IO lines.The Basys 3 has 12 IO lines to drive color signals, with 4 bits driving each respective RGB color intensity. The VGA ports on both boards use two synchronization signals, one called hsync (horizontal sync) to specify the time taken to scan through a row of pixels, and the other called vsync (vertical sync) to specify the time taken to scan through an entire screen of pixel rows. Properly driving the hsync and vsync signals will be our main focus. Now we will begin to talk about resolution and pixels.The display area on the screen using VGA has a resolution of 640 by 480 pixels. The origin of the display area is (0,0) and is located in the upper left corner. The x dimension increases from left to right, while y increases from top to bottom.



We will run our VGA circuit at a 25 MHz pixel rate, meaning that 25 million pixels will be scanned each second. The monitors usually include small black borders surrounding the display area. We will divide each horizontal scan row into 800 pixels, starting at pixel 0 and ending at pixel 799. We will scan through each horizontal row of pixels at 25 MHz. The pixel count begins at the start of the display area, and runs through all 640 display pixels, followed by a 16 pixel right border. Next is a 96 “pixel” retrace, which is a delay artifact of CRT monitors that we must include, and finally a 48 pixel left border. We will divide the vertical space into 525 horizontal rows, with the scanning through of all rows counting as one refresh of the screen. The vertical pixel count starts in the display area and ends at pixel 479, continues into a 10 pixel bottom border, then a 2 pixel retrace, and finally ends after a 33 pixel top border. The pixels in the border and retrace areas should be black, so we will generate a video\_on signal that is asserted when the current pixel is in the horizontal and vertical display region. This signal will turn on our RGB color signals for the display pixels only. To avoid noticeable flickering from the screen, a goal of a 60 Hz refresh rate is generally acceptable. If the VGA circuit is clocked at 25 MHz, then each pixel scan will have a period of 1/25 Mhz = 40 ns. If we multiply that by 800 pixels per row, and 525 rows per screen, we get around 1/60 seconds per screen refresh, which meets our goal.



## 4x4 Key-pad (PMod)

Functional Description:

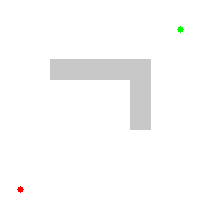
The PmodKYPD utilizes 4 rows and columns to create an array of 16 momentary push-buttons. By driving the column lines to a logic level low voltage one at a time, users may read the corresponding logic level voltage on each of the rows to determine which button, if any, is currently being pressed. Simultaneous button presses can also be recorded, although it is still required to step through each row and column separately in order to ensure that the pressed buttons do not interfere with each measurement. By design, multiple simultaneous button presses within the same row cannot be individually detected.

Interfacing with the Pmod:

The PmodKYPD communicates with the host board via the GPIO protocol. Each button is placed within a simple voltage divider circuit. When a button is not pressed, a large pull-up resistor maintains a logic level high voltage on each of the row pins. When a column pin is driven to a logic level low voltage and a corresponding button is pressed, completing the voltage divider circuit, the row pin will then read a logic level low voltage instead.

## Dijkstra's Algorithm:

We will create an array d[]d[] where for each vertex vv we store the current length of the shortest path from ss to vv in d[v]d[v]. Initially d[s]=0d[s]=0, and for all other vertices this length equals infinity. In the implementation a sufficiently large number (which is guaranteed to be greater than any possible path length) is chosen as infinity. d[v]=∞, v≠s d[v]=∞, v≠s In addition, we maintain a Boolean array u[]u[] which stores for each vertex vv whether it's marked. Initially all vertices are unmarked: u[v]=false u[v]=false The Dijkstra's algorithm runs for nn iterations. At each iteration a vertex vv is chosen as unmarked vertex which has the least value d[v]d[v]: Evidently, in the first iteration the starting vertex ss will be selected. The selected vertex vv is marked. Next, from vertex vv relaxations are performed: all edges of the form (v,to)(v,to) are considered, and for each vertex toto the algorithm tries to improve the value d[to]d[to]. If the length of the current edge equals lenlen, the code for relaxation is: d[to]=min(d[to],d[v]+len) d[to]=min(d[to],d[v]+len) After all such edges are considered, the current iteration ends. Finally, after n iterations, all vertices will be marked, and the algorithm terminates. We claim that the found values d[v]d[v] are the lengths of shortest paths from ss to all vertices vv. Note that if some vertices are unreachable from the starting vertex ss, the values d[v]d[v] for them will remain infinite. Obviously, the last few iterations of the algorithm will choose those vertices, but no useful work will be done for them. Therefore, the algorithm can be stopped as soon as the selected vertex has infinite distance to it.



By Subh83 - Own work, CC BY 3.0,

# 

# 

# Design:

1. The user defines starting and ending locations through the 4x4 Key-pad.
2. There are some predefined path through which user can traverse. This in real life represents the roads.
3. The most efficient path between the starting and the ending point is found through Dijkstra's Algorithm and the path is shown on the monitor.
4. The distance and time of the path will be shown on the leds and seven segment display of the basys3 fpga board respectively.

# Scope of Improvement

There could be some scope of improvement of this project starting from outputting all the locations which are to be covered while going from initial position to the end position. Time output does not consider the traffic on the roads.