EE314 Term Project – Spring 2021
FPGA Based Point of Sale Terminal

Introduction

During our daily life, we are doing shopping at markets or we are eating at restaurants. As some of you already noticed that cashiers or waiters are using point of sale (POS) terminals to list down the order lists. While cashiers at groceries use barcode scanners or simply write down the item numbers to make lists through POS terminals, waiters generally use hand-held touch screen devices. Though restaurants or groceries are just two examples of usage of POS terminals, incidence can be increased. Term project of this year was inspired from a common POS terminal:

You are going to design and implement a POS terminal whose screenshot will be similar to the one seen in the Figure 1.

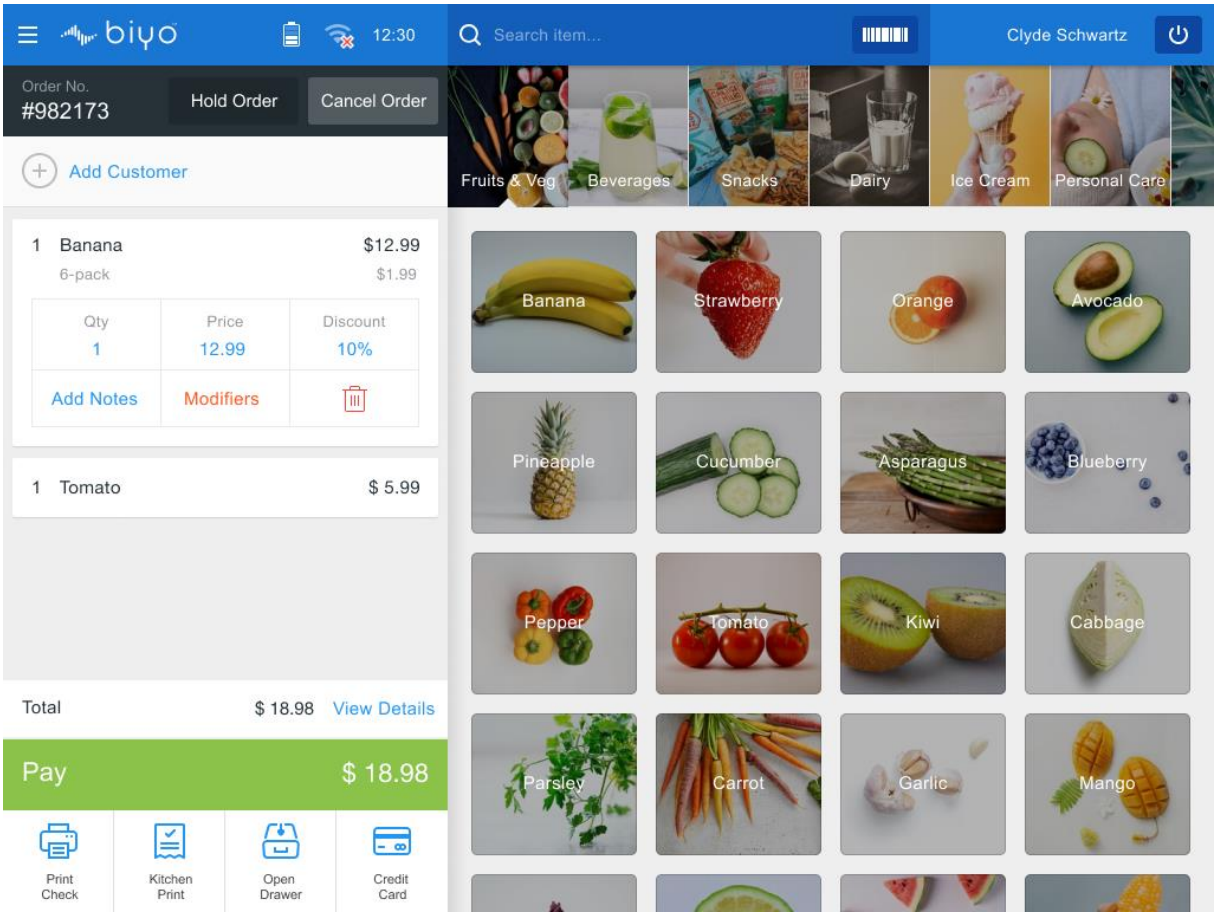

Figure 1: Example Grocery POS Terminal Screenshot

Project Definition

You are going to implement your project on the FPGA boards that you will borrow from us. To define the requirements one by one as a list:

1. You are going to design a main screen using the VGA interface. Probably, 640 × 480 resolution will be enough. Any higher resolution is welcomed of course, but not a must. In your main screen, you must put items appearing in the list below. As long as your screen contains all of the list, layout of your screen is not important.
    a. First, you must design a **logo** for your project group, and you must put it to your screen. Of course, no professional or extensive effort is expected. Just writing your group number (*group name if applicable*) in an italic, colorful etc. way with MS Paint is enough.
    b. Secondly, **images of items you are selling**. Your POS screen must include 12 items being half of them already defined as **banana**, **potato**, **tomato**, **peach**, **apple**, **pineapple**. You are free to choose the image resembling your item. For instance, you might show a green or red apple to resemble apple. Furthermore, you are free to either show high quality images or colored hand drawn bitmap images like in the Figure 2. Remaining six items are to be defined by yourself. Any items to be sold in a grocery like fruits, vegetables etc. are welcomed. However, please do **not** include alcoholic beverages, tobacco products, pork products, any illegal, inappropriate, or disputed item in your project to avoid an undesired sociological problem. Doing so will be **penalized strictly**.

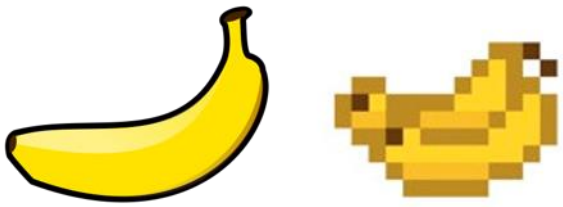

Figure 2: Bananas

    c. **A shopping list with prices**. All items, of course, have a price. No one gives bananas for free, unfortunately. Once a guest buys items through your POS, **bought items** should appear in the list with **prices and quantities of each**. For sake of simplicity, one guest can buy up to six different items. Once the guest finishes shopping, a general **sum** should be calculated and written at a **different line**. Prices of predefined six items can be seen from the Table 1. Prices for items defined by you also should be defined by yourself.

Table 1: Prices of Predefined Items

Banana	Potato	Tomato	Peach	Apple	Pineapple
2.50 TL	0.50 TL	0.75 TL	2.00 TL	1.00 TL	9.95 TL

2. Going to the operator side, input to the system should be done as follows:
    a. Once operator welcomes a new guest, it should switch the device to the listing mode with the help of **buttons** of FPGA board. Each item has a **4-**

digit number (*you can treat it like a very short barcode number*). The operator enters code of a desired item by using buttons in sequential order. Codes of predefined items can be seen at the Table 2.

Table 2: Identification Codes of Predefined Items

Banana	Potato	Tomato	Peach	Apple	Pineapple
3124	**4132**	**4133**	**3121**	**3133**	**3214**

b. While the operator entering numbers to the FPGA board, an auto-complete like operation should be performed as well. This can be done by highlighting related items with a color frame etc. To give an example, if the operator puts **`3-2-1-4;`** banana, peach, apple, pineapple should be highlighted after **`3`**. Then after **`2`**, only pineapple should remain highlighted.
c. After four digits are entered, operator puts the **quantity** again with the help of **buttons**. For the sake of simplicity, one customer can buy up to 4 pieces from each item.
d. Finally, the operator must **finish** the shopping progress by using a button combination. Once shopping is finished, total price should appear at a new line.

The overall procedure can be summarized as in the Figure 3.

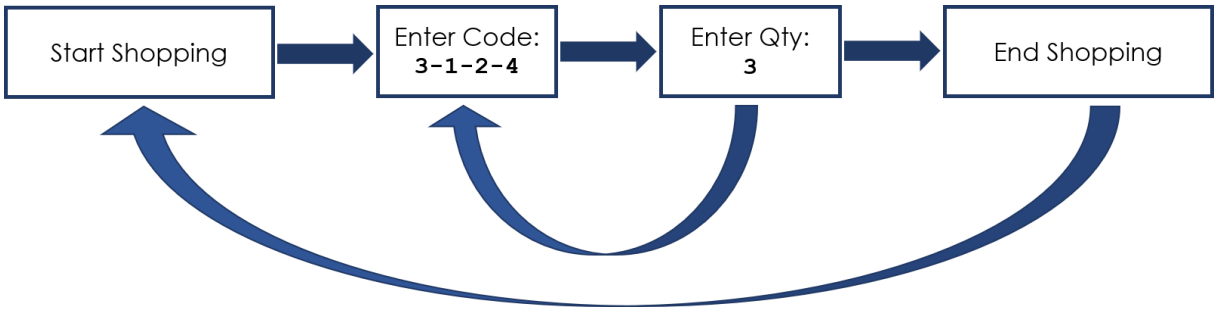

Figure 3: Example Operation Flow

3. Device should include some interactive modes; each can be selected by using switches:
   a. If **`SW-1`** is set **`HIGH`**, selecting item to be sold is done interactively navigating above images on the screen. Operator or guest can navigate between items by using four **buttons**, each of them assigned a direction. While navigating the shopping list, an indicator should be visible on currently hovered item. Then clicking the assigned select button carries POS to the quantity select operation.
   b. If **`SW-2`** is set **`HIGH`**, navigating through to the shopping list becomes possible. While navigating the shopping list, an indicator should be visible on currently hovered item. Then clicking the assigned **cancel** button deletes the selected item from current shopping list. Regardless of state

of the **SW-1**, navigation should be activated through to the list if **SW-2** is set **HIGH**.

Final Words

This is **not** a last-night project. You must put an effort on it to be successful. On the other hand, do not worry. This is a straight-forward project in general.

You are going to submit one **final reports**. In the report you are going to explain your conceptual design or solutions to the requirements for the project. State diagrams defining your logic scheme, theoretical understanding of VGA and buttons, and finally some simulation results should be included as well. Grading will be announced soon as well.

Please keep in mind that we have limited resources and we must benefit wisely so that new students can also benefit. FPGA boards are sensitive, fragile, and expensive boards. Now, they are in perfect condition. For the duration you borrowed them from us, you will be responsible for taking care of them. Once time to give FPGA boards back reached, you must give your borrowed FPGA back in a perfect condition as well. We need to openly declare that **failing** to give back the FPGA board will result in a **harsh penalty**.

As you can notice also, some details regarding the project are not crystal clear, some parts are left to you. For some parts, there are nearly infinite options. This was actually done on a purpose: If you cheat (*copy and paste some parts or the whole project*) on your project, it will be crystal clear. (*How likely to select banana image, group logo, six different undefined items, their IDs, and their prices exactly the same?*) We would like to remind that **cheating** is serious issue, which will be strictly **penalized** as well.

We hope that all of you will do creative and successful project demonstrations. Unless there will be no unexpected situations due to pandemic or any other unexpected thing, we will be together during the demonstrations.

Enjoy.

Appendix: VGA Interface

VGA is a widely used standard in video industry for the transmission of video signals from a computer or microprocessor into a monitor or TV. Each 640x480 image is called a 'frame' and each frame contains 480 lines which are made up of 640 pixels.

The monitor starts displaying each frame by beginning from the first line and then the first pixel of this line. In each line, the display order is from left to right; and each frame is written in an order from top to bottom. So, your first pixel is always at the top left corner, while the last pixel at the bottom right.

You will need to generate an image buffer with at least 640x480 = 307200 bits to store each line and frame in order to form a coherent image; however you will also need to adjust two synchronization signals called **`HSync`** (Horizontal Synchronization) and **`VSync`** (Vertical Synchronization) in order to see a video. These signals tell the monitor when a line or frame is finished, and the monitor should start from the next line or frame.

As shown in Figure 4, VGA interface is actually very simple, and you will only need to make 3 connections, namely R-G-B. For example, for a white pixel all three inputs should be high, and for a black pixel the inputs should be low. The FPGA cards in the laboratory already have a VGA output port with color outputs, so you will only need to supply the R-G-B data digitally to the VGA port. Necessary pins for these assignments can be found in the user manual.

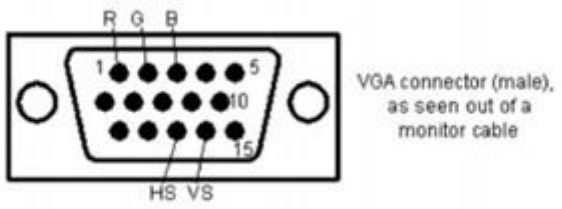

Figure 4: VGA interface.

`HSync` and **`VSync`**: **`HSync`** and **`VSync`** are necessary in order to tell the monitor to 'start' or 'stop' writing a line or frame. You will need to build the necessary digital blocks in order to correctly form these two signals. These blocks are basically counters with some modifications and are very easy to implement in Verilog. You can see the horizontal and vertical synchronization signals in Figure 5 with the corresponding timing in Table 3.

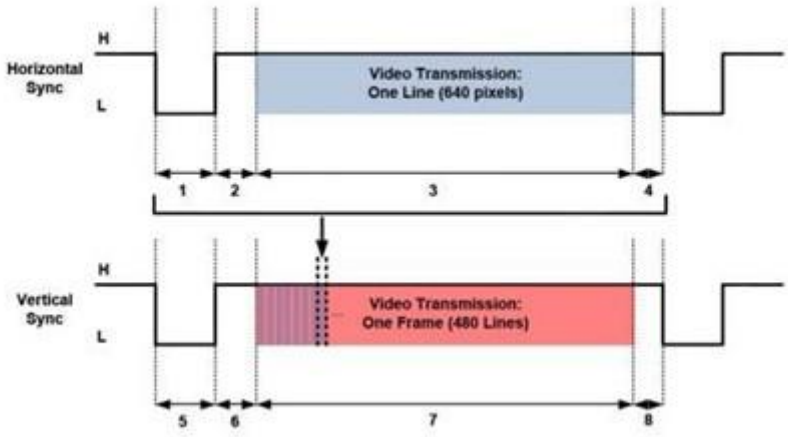

Figure 5: HS and VS.

Timeline # on Fig. 1	Name	Duration	Clock Count
1	H. Sync	3.84 µs	96
2	Back Porch (H)	1.92 µs	48
3	Video Signal (One Line)	25.6 µs	640
4	Front Porch (H)	0.64 µs	16
5	V. Sync	0.064 ms	2
6	Back Porch (V)	1.056 ms	33
7	Video Signal (One Frame)	15.36 ms	480
8	Front Porch (V)	0.32 ms	10

Table 3: Timing.

By observing Figure 5 and Table 3, we can understand that the **HSync** signal is used to synchronize one line in a frame, while **VSync** is used to synchronize each frame. Basically, when **HSync** or **VSync** is low, the monitor understands that it needs to switch from one line or frame to the next. Back and front porch are idle stages where the monitor is getting ready to write the next pixel or line. They also include 8 pixel and line over scan or 'border' pixel/lines outside our standard view of the monitor.

IMPORTANT NOTE: The video input signals (R, G, B) of a VGA monitor should be off (or black) during H. or V. Sync stages, and front/back porch stages. The video input signals should only be active during an active video transmission stage.

In order to construct these **HSync** and **VSync** signals and to achieve transmission of each line/pixel, you will need a 25 MHz clock signal. This will also mean that each pixel will be transmitted at 25 MHz to the monitor during active video stages.

Internal clock information about ALTERA can be found under the Clock Circuitry part of the user manual.

http://www.epanorama.net/documents/pc/vga_timing.html
http://martin.hinner.info/vga/640x480_60.html