ECE532: Final Group Report

Computer Vision of the Idioscope

Paul Haist – 994865197
Nahid Hassan - 994877558

Table of Contents

1 Overview

1.1 Background Information

The inspiration for our project came from the Idioscope suggested by Professor Mann [**1**]. The Idioscope is basically a virtual desktop drum set as shown below in Figure 1. A series of shapes drawn on a surface represent the different sounds or notes that can be played. To determine which block is tapped, a camera is mounted in a fixed position above the surface and tracks the position of a wand that has red tape at the end of it.

A tap is detected by a microphone mounted either on the desktop surface or on the wand itself. Audio filtering would determine when a tap has occurred and the video tracking system would be alerted. The coordinates of the red tape would then be used to determine which shape the wand had tapped and the corresponding note that should be played.

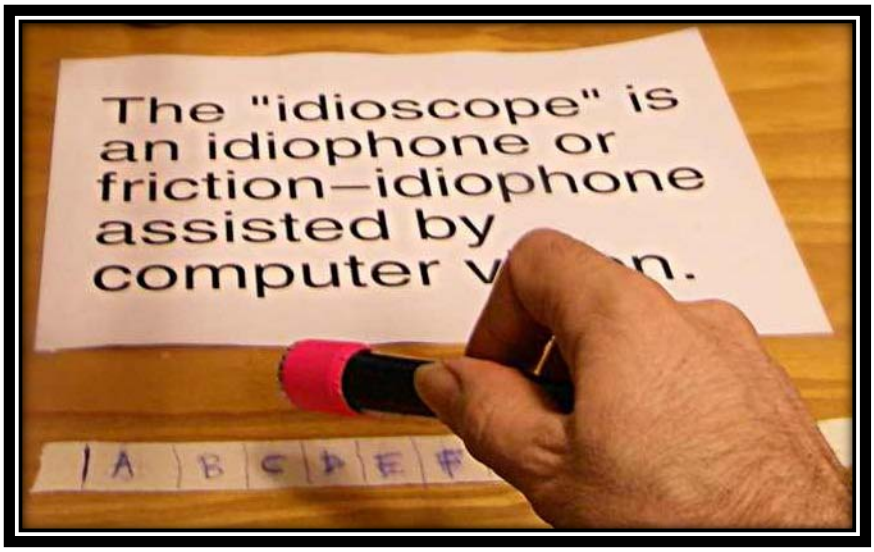

Figure 1: A basic implementation of the Idioscope from Prof Mann's website

1.2 Goals

Our team decided to implement the video side of the Idioscope project. That is, our system must be able to determine the centre point of the tip of a wand in the camera's field of view and determine which shape the wand tip is within. Also, the system must report which note or sound needs to be played.

We decided to implement an extra feature in addition to those mentioned above. In the Idioscope proposed by Prof Mann, the shapes drawn on the surface that represent the possible notes are limited to a set of rectangles with fixed notes. We felt that the user should not be limited to an fixed set of rectangles so we decided to incorporate an initial teaching phase

where arbitrary closed shapes could be drawn on the surface and the notes could then be assigned to these shapes by the user.

1.3 Implementation

Our project was implemented on a Xilinx XUP Virtex-II Pro Development System and used the Diligent Video Decoder Board (VDEC1) to interface with a video camera. A Panasonic DMC-TZ4 digital camera was used to track the position of the wand. We also used a VGA monitor to display the results of the teaching phase and to indicate which note has been played.

For demonstration purposes, a C++ program running on a PC interprets data coming from the XUP Development Board over RS-232 and plays sound files corresponding to the correct notes. The PC also informs the XUP Board that a tap has occurred based on user input.

A block diagram of all the components in our system and how they connect together can be found in Figure 2 on the next page.

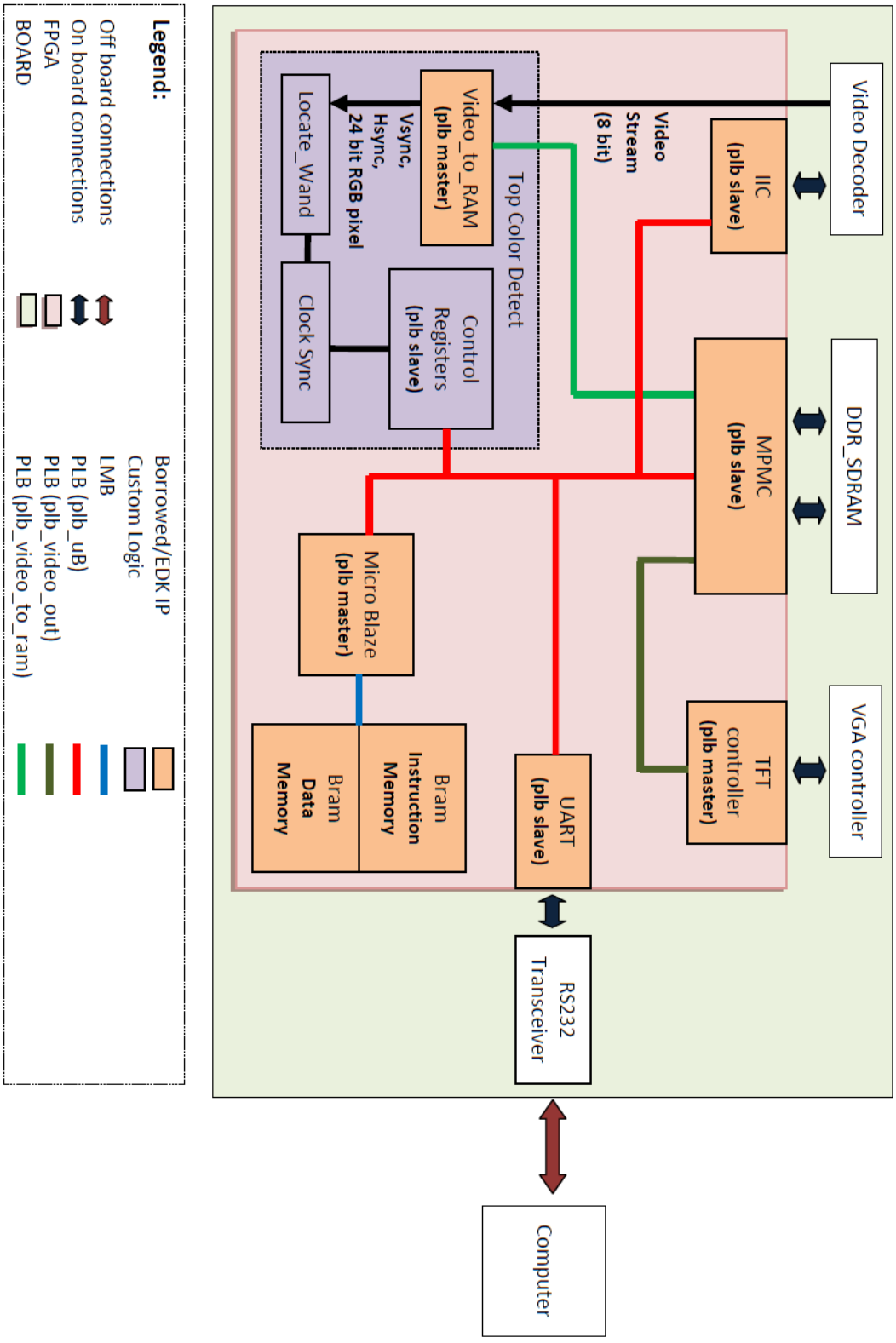

Figure 2: Block diagram of the system

1.4 Brief Description of IP

For each of the blocks in the system block diagram (see Figure 2), a description of its function in the system along with its origin can be found below in Table 1.

Name	Function	Origin
Microblaze + BRAM	A soft processor used to configure the different IP blocks as well as execute the teaching algorithm.	Xilinx IP
LMB	The bus used by the microblaze to communicate with the BRAM containing data and instruction memory	Xilinx IP
PLB	Three high bandwidth buses: 1. Video_to_RAM block writes video frames to the MPMC 2. TFT block reads video frames from the MPMC 3. Microblaze configures the IP blocks and performs reads/writes on MPMC	Xilinx IP
MPMC	Provides multiple interfaces to the DDR SDRAM located on the XUP board. Acts as an arbitrator when simultaneous access of the SDRAM is requested	Xilinx IP
UART	Connected to the microblaze PLB. It is used to send and receive data from the PC using an RS-232 protocol	Xilinx IP
IIC	Bridges communication between the microblaze and the VDEC1 decoder	Xilinx IP
TFT	Reads video frames from the MPMC and generates VGA compatible video signals	Xilinx IP
Video_to_RAM	Reads in the YCbCr video stream from the VDEC1, converts the stream to RGB and continuously writes each frame to the MPMC	Borrowed Custom/Xilinx IP
Locate_Wand	Reads the RGB video stream from the Video_to_RAM block and finds the centre of a coloured object (Either Red, Green, or Blue). Communicates with the microblaze over PLB	Custom IP
Control_Registers	The microblaze configures the Locate_Wand block and reads back the coordinates of the wand through these registers	Custom IP (Generated by Xilinx wizard)
Clock_Synchronizer	Synchronizes communication between the microblaze's 100MHz clock and the 13MHz clock of the video logic	Custom IP
Top_Colour_Detect	A wrapper module for the Video_to_RAM block, the Locate_Wand block, the Control_Registers and the Clock_Synchronizer	Custom IP

Table 1: Description of all IP blocks used in the FPGA

There were several components used in the implementation of our design that were not internal to the FPGA. These components are described below in Table 2.

Name	Description/Function
DDR SDRAM	A 512MB stick of DDR SDRAM located on the XUP Board holds various data for the project including: Input, output, and working video frames as well as a custom stack used by the shape detection algorithm
VGA Monitor	Displays the image in the output frame of the DDR SDRAM. Used to provide visual feedback to the user
VDEC1	The Diligent Video Decoder Board attached to the XUP Board. The analog video stream from the camera is decoded into a digital YCbCr stream and sent to the FPGA
Camera	A Panasonic digital camera with an composite video out is used to track the position of the wand during all phases of the design
PC	A PC is connected to the FPGA over a UART connection. All messages for the user are displayed and input from the user is sent back to the FPGA. The PC also plays sound files whenever the user taps a valid shape.

Table 2: Description of the components external to the FPGA

1.5 Outcome

The final result of our project is a successful prototype. The user first draws any number of arbitrary closed shapes using a black marker on white paper within the field of view of the camera. Then, through the PC, the user starts the teaching phase. Then the user can assign notes in ascending order by 'tapping' within each shape. A 'tap' is triggered by hitting the <ENTER> key on the PC.

Once all shapes have been assigned a note, the user can exit teaching mode and begin to play their newly created virtual instrument. When any of the shapes are 'tapped' a codeword byte is sent to the PC followed by an integer representing the note that should be played. For our demonstration, only 10 unique notes from a piano can be assigned and played.

1.6 Future Work

In the future, an audio processing module would be integrated on the FPGA and perform two main functions. Firstly, a microphone would listen for the wand to be tapped and an audio filter would determine when a tap occurs. Secondly, the notes would be generated by the audio module instead of being a fixed set of audio files. This would allow any note to be played from the instrument of the user's choice.

2 Detailed Description of Hardware Blocks

The following sections outline the details of each block used in our design.

2.1 Microblaze (Ver 7.10d)

This IP provided by Xilinx is a soft-processor with a 32-bit architecture. It uses the Block RAM local to the Virtex-II FPGA to store both instruction and data memory. The microblaze is run at a 100MHz as is the PLB it is connected to.

The microblaze executes all the initialization and configuration code for the VDEC1 board and the Locate_Wand IP block. It then uses the functionality of the Locate_Wand block to run the teaching phase, followed by the main function of the project. This main function involves reading the location of the wand and sending codewords followed by notes over the UART to the PC.

2.2 PLB (plb_v46 Ver 1.03a)

The Processor Local Bus is a high speed data bus with several useful features. It has support for multiple masters, multiple slaves, and has the capability of arbitrating control for masters with different priorities.

Our design uses three separate instantiations of the PLB all using a 32-bit data bus. These three buses have the following functions:

1. **plb_uB:** The microblaze is the single master on this bus. It is used to communicate between the different IP blocks for initial configuration. It is also used to control the Wand_Locator block and read back the result as well as send and receive data from the PC over the UART block
2. **plb_video_out:** The TFT block is the only master on this bus and the MPMC is the only slave. The sole function of this PLB is to transfer data from the output frame in the DDR_SDRAM to the TFT block so it can be displayed on the VGA monitor.
3. **plb_video_to_RAM:** The Video_to_RAM block is the only master on this bus and the MPMC is the only slave. The video_to_RAM block writes RGB video frames to the DDR_SDRAM over this bus.

2.3 MPMC (Ver 4.03a)

The Multi-Ported Memory Controller is the FPGA's interface to the DDR_SDRAM on the XUP Board. The true strength of the MPMC is the multi-port access it provides. This allows various IP blocks to read and write to the DDR_SDRAM independently and without worrying about collisions.

Our project uses an MPMC with three independent ports configured as PLB slaves. Each of the three PLB instantiations mentioned in Section 2.2 has a connection to a port of the MPMC.

2.4 UART_uB (xps_uartlite Ver 1.00a)

The Universal Asynchronous Receiver/Transmitter block is a slave device on the plb_uB and is the communication link between the FPGA and the PC. The microblaze sends strings of characters to the UART block and these characters are relayed to the PC over the RS-232 protocol.

This link is used for two purposes. Firstly, it is used to communicate messages to and from the user. Secondly, it sends codewords to the PC in order to trigger a sound file to be played.

2.5 IIC (xps_iic Ver 2.00a)

The Inter-Integrated Circuit block is a slave on the plb_uB and is used to communicate with the video processing chip on the VDEC1 Board. The block is used only once at the startup of the system to configure the settings of the video processor.

2.6 TFT (xps_tft Ver 1.00a)

The Thin Film Transistor Controller block is the single master on the plb_video_out bus. Its sole purpose is to read the output frame located on the DDR_SDRAM and generate VGA timing signals.

2.7 Video_to_RAM (Ver 1.00a)

This IP block was borrowed from Jeffrey Goeders. It consists of several Xilinx IP blocks linked together in a custom top-level block. The purpose of this block is to take in the YCbCr video stream from the VDEC1 Board and write RGB pixel data to the input frame of the DDR_SDRAM. The Xilinx blocks within and their purposes are listed below:

- **BUFG:** Clock buffer for the 27MHz input clock and the 13MHz generated clock
- **OFDDRRSE: A** DDR output flip flop to center the clock in the pixel data
- **lf_decode:** Decodes the incoming video stream to extract the vertical and horizontal sync signals as well as determine if the stream is interlaced
- **vp422_444_dup:** Upsamples the YCbCr stream by simply duplicating the Cb and Cr values
- **YCrCb2RGB:** Converts the upsampled YCbCr stream to an RGB stream
- **BRAM LINE_BUFFER:** This stores a line of RGB pixel data. The dual port functionality of the BRAM is used to synchronize data between the 13MHz video clock and the 100MHz plb_video_to_RAM clock.
- **NEG_EDGE_DETECT:** Detects the falling edge of the field bit in the timing reference code to synchronize the output stream with the video source.
- **SPECIAL_SVGA_TIMING_GENERATION:** Handles interlaced video and correctly outputs line numbers.

A few modifications were made to this IP Block based on the requirements of our project. The Locate_Wand block needs the incoming stream encoded in RGB format along with the sync signals to function. Therefore these internal signals were brought out as outputs from this block.

Also, the 13MHz clock generated by this block is used to clock the logic inside the Locate_Wand block as well as the Clock_Synchronizer block.

2.8 Locate_Wand

The purpose of this block is to sniff into the pixel stream generated by the video_to_RAM block to locate the centre coordinates of the wand.

Three possible colours can be used in a wand – Red, Blue or Green. The equation used to determine if a pixel is red from the RGB values is given below:

$$ColorRedValue = red^2 - green^2 - blue^2 - COLOR\_NORM$$

The *ColourRedValue* being greater than zero means that a red colour has been detected. The *COLOUR_NORM* factor enables us to control the measure of pure red that is to be detected. So a higher *COLOUR_NORM* value will detect purer red. Similarly, blue and green can be detected using the same equation just by swapping the red component with the respective colour.

The Locate_Wand traverses through the source video frame keeping track of the uppermost, lowermost, leftmost and rightmost coordinates of the wand coloured pixels in the frame. In order to cope up with corruption in the frame, we disregard small cluster of pixels which might have the same colour as the wand. Then it calculates the mean of these values appropriately to find the X and Y coordinates of the centre of the wand using the following equations:

$$X_{coordinate} = (left + right)/2$$

$$Y_{coordinate} = (up + down)/2$$

When this algorithm was implemented in hardware, several registers were created to store configuration values as well as the results of the algorithm. After the configuration registers are set, the process of locating the wand is stated by setting Go bit within the status register. The hardware waits for the next Vsync to start computations on a new frame. Once the algorithm has completed, it sets the Done bit within the status register on the next Vsync which indicates that the coordinates are ready to be read.

2.9 Control_Registers

The Locate_Wand block uses several configuration values that are set by software running on the microblaze. Also, the resulting coordinates must be read from the block by the microblaze. The simplest way to accomplish this communication is over the plb_uB bus.

Five 32-bit registers were created for the Locate_Wand block. The names and descriptions of these registers can be found below:

STATUS_NORM

Bit	31	30	29:16	15:0
Name	Go	Done	Reserved	Colour_Norm
Read/write	R0/W	R	R0	R/W

- **Bit 31 – Go**: Setting this bit to one starts the process of detecting the wand based on the configuration values. This bit will always read as a zero.
- **Bit 30 – Done**: This bit will go high once the location of the wand has been found and the coordinates have been loaded into CENTRE register. The bit will be cleared by hardware when a one is written to Go. Writing to this bit has no effect.
- **Bits 29:16 – Reserved:** These bits will always read as zero.
- **Bits 15:0 – Colour_Norm:** The 16 bit normalization value for the Locate_Wand algorithm. Details on this value can be found in section 2.8.

WAND_IGNORE

Bit	31:18	17:16	15:0
Name	Reserved	Wand_Colour	Ignore_Pixels
Read/write	R0	R/W	R/W

- **Bits 29:16 – Reserved:** These bits will always read as zero.
- **Bits 17:16 – Wand_Colour**: These bits correspond to the colour of the wand. There are three possible values that can be stored in these bits:
  - **0x00**: Red
  - **0x01**: Green
  - **0x02**: Blue
- **Bit s 15:0 – Ignore_Pixels**: To deal with stray pixels that are the same colour as the wand any cluster of pixels less than the value Ignore_Pixels will be ignored.

LEFT_RIGHT

Bit	31:16	15:0
Name	Left	Right
Read/write	R	R

- **Bits 31:16 – Left**: Once the Done bit in the STATUS_NORM register has gone high, the horizontal coordinate of the leftmost pixel can be read here.
- **Bit 15:0 – Right**: Once the Done bit in the STATUS_NORM register has gone high, the horizontal coordinate of the rightmost pixel can be read here.

TOP_BOTTOM

Bit	31:16	15:0
Name	Top	Bottom
Read/write	R	R

- **Bits 31:16 – Top**: Once the Done bit in the STATUS_NORM register has gone high, the vertical coordinate of the uppermost pixel can be read here.
- **Bit 15:0 – Bottom**: Once the Done bit in the STATUS_NORM register has gone high, the vertical coordinate of the lowermost pixel can be read here.

Centre

Bit	31:16	15:0
Name	X	Y
Read/write	R	R

- **Bits 31:16 – X**: Once the Done bit in the STATUS_NORM register has gone high, the horizontal coordinate of the centre of the wand can be read here.
- **Bit 15:0 – Y**: Once the Done bit in the STATUS_NORM register has gone high, the vertical coordinate of the centre of the wand can be read here.

2.10 Clock_Synchronizer

This block is used to synchronize the data between the Control_Registers which is running at the plb_uB clock of 100 MHz and the Locate_Wand which is running at the pixel clock of 13 MHz. The mux enable scheme is used for clock domain synchronization [**2**]. The Go bit is the enable bit from PLB clock domain and the Done bit is the enable bit from pixel clock domain.

2.11 Top_Colour_Detect

The Top_Colour_Detect is the top level wrapper for the video_to_RAM, Clock_Synchronizer, Wand_Locator and the Control_Registers blocks. It brings out the PLB interfaces for both the slave Control_Registers and the master video_to_RAM. It also takes in the video signals from the video decoder.

3 Microblaze Software

There are two main sections in the microblaze software – the teaching phase and the playing phase. During system startup, the user needs to train the device to indentify the notes associated with different shapes. This is done by constructing a lookup table containing integers that represent the notes of the instrument. In the playing phase, the system determines the coordinate of wand during a tap and simply looks up the integer from the lookup table and sends it out via UART.

3.1 Teaching phase

The teaching phase implements a look up table containing the integers that represent the notes of the instrument. There are two main tasks involved in the creation of this lookup table. Firstly, an image of the shapes drawn by the user is stored in the DDR_SDRAM and floored. The result of the flooring is an image where the borders of the shapes and the background have a significant contrast. Secondly, the region enclosed by each shape is identified and a note is assigned by the user.

The steps involved in creating a floored image are outlined below:

1. Copy a video frame of the incoming video source to a separate memory location known as the working frame.
2. Compute the magnitude of the pixels to provide a higher contrast between dark and light regions using the equation below, where the colour components are 8 bit unsigned values:

$$ImageMagnitude = (red^2 + green^2 + blue^2)$$

3. Scale the magnitude of each pixel in the image so that the *ImageMagnitude* is limited to a 8 bit unsigned value:

$$ScaledImageMagnitude = ImageMagnitude/IMAGE\_SCALE\_FACTOR$$

4. Floor each pixel to either white (0xFFFFFF) or black (0x0) based on whether it is above or below the threshold factor called *TEACH_FLOOR_THRESH*

After the creation of the floored image, possible pixel values of the resulting image are either pure black or pure white. This makes it easier to distinguish between a light background and a dark border.

To identify a closed region, the pixels in that region are changed to an integer value other than black or white. This is done using a recursive algorithm which is described in the code below. In the code, `pixel_value` is the integer number that will be written to every pixel within the shape and the `pixel_coordinate` is the algorithm starting location.

```
depth_first_search(pixel_value, pixel_coordinate)

    IF (coordinate outside image OR shape border OR pixel
    visited before)

        DO nothing

    Else

        Assign this pixel with pixel_value

        CALL depth_first_search… on LEFT pixel

        CALL depth_first_search… on RIGHT pixel

        CALL depth_first_search… on UP pixel

        CALL depth_first_search… on DOWN pixel
```

When this algorithm was implemented on the microblaze, there was not enough data memory for the stack to execute the recursive function as many times as required. Therefore, the function was rewritten as an iterative function and a block of the DDR_SDRAM was used as an extended stack.

The different shapes are assigned a unique pixel value which refers to a note. Once all the shapes have been identified by the user, the teaching phase is complete and then the system moves to the playing phase.

3.2 Playing Phase

In the playing phase, the microblaze waits for a tap of the wand which in the case of our demo is any byte received on the UART. When there is a tap, the Go bit in the top_colour_detect status registers is set which starts the Locate_Wand algorithm. Then the CPU polls for the Done bit. When the Done signal is received, the coordinates of the centre of the wand are read. These coordinates are used to look up the integer representing a note from the look up table

generated in the teaching phase. Then a codeword byte followed by the integer is sent out via UART to the PC. This process repeats until the system is restarted by the user.

4 Computer Software

The computer is running a C++ program which notifies the microblaze of a tap as well as plays the note corresponding to the integer. There are two threads running in this program which take care of the read and write of the serial port separately. The write thread notifies the microblaze of a tap of the wand by writing a byte to the serial port when the <Enter> key is hit on the keyboard. The read thread reads bytes sent by the microblaze which contains user interface messages and note information. All the user interface messages are relayed to the console of the computer. When a codeword is received from microblaze indicating a note, it treats the next byte as an integer and plays the associated note.

This C++ program is used only for the purposes of the demo and would not be needed if the work outlined in Section 1.6 were completed.

5 Design Tree

The following is a description of the files located in our project's directory tree:

- **Project_files** – root directory
  - **hw –** all of the pcores used in this project
    - **top_colour_detect_v1_00_a –** this core contains the video stream decoding logic, the colour detection (used to detect the wand), the control registers for the colour detection and finally the synchronizer between the plb's 100MHz clock and the colour detector's 13MHz clock
  - **sw –** all of the drivers and application software used in this project
    - **main.c –** The main application code for the project.
    - **teaching.c –** The depth first algorithm along with the image processing.
    - **colour_detect.c –** Functions that initialize the Locate_Wand core and run the colour detection algorithm
    - **video_setup.c –** A single function along with a custom structure to initialize the decoder chip on the VDEC1 Board over I^2C.
    - **video_RAM.c –** Functions that can read or write a single 'pixel' in the DDR_SDRAM and a function to copy an entire video frame from one location to another.
    - **UART.c –** Functions to access to low level registers of the uartlite module. Used to send special codewords and receive characters from the PC

- **Full_project.zip –** Our project in its entirety without any generated hardware or compiled software. If this project is opened and a bitstream is generated, the project will work as described in this document.

6 Bibliography

[1] http://wearcam.org/idioscope/ece532.htm

[2] http://w2.cadence.com/whitepapers/cdc_wp.pdf