XMOS SEMICONDUCTORS PVT LIMITED

Internship Report

May 18, 2015 to July 17, 2015

Submitted By:

KIRAN C PATTANASHETTY
M.Des Electronics Systems
3rd Semester
Indian Institute of Information Technology, Design and Manufacturing
Kancheepuram -600 127

Project 0: Blinking LED

This project implements an example of blinking the xCORE200 eXplorer kit on board LED. The board has 4 LEDs. They are red, green, blue and yellow. Each LED gets blinked in a sequence from Yellow through red with 500ms delay. Different color combination can be achieved by setting different combination of RGB LED.

IDE used: xTIMEcomposer studio 14

Project 1: Controlling LED using Switch

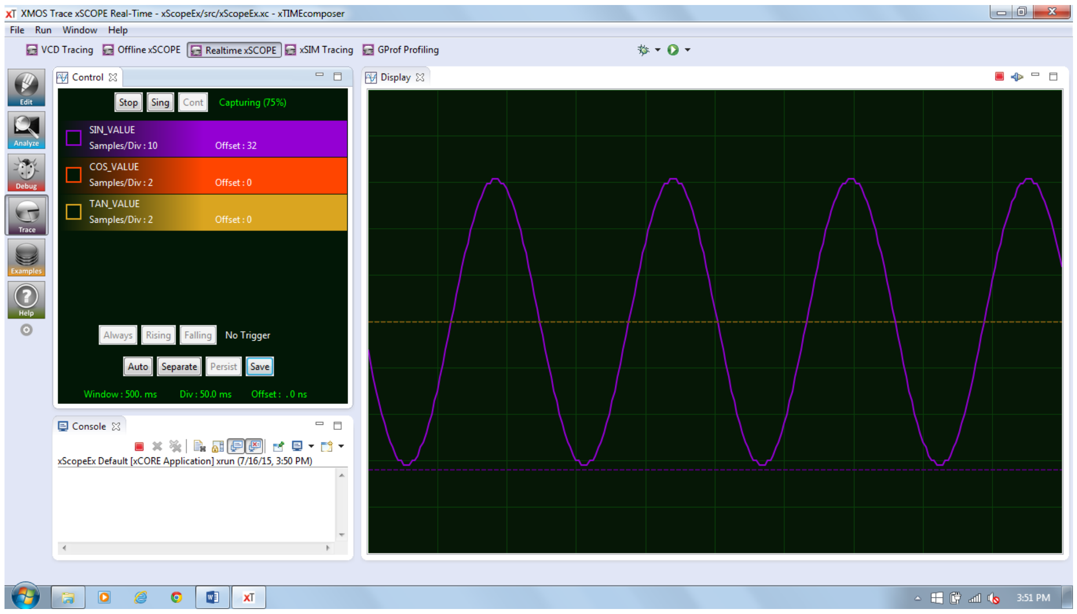
The xCORE200 eXplorer kit comes with two push button switches on board. These are negative logic switches. The project aims at controlling the states of LED from on board push button. When button pressed, all the LEDs toggle in sequence with different colors.

IDE used: xTIMEcomposer studio 14

Project 2: Sine wave display on xSCOPE

In this project a sine wave is generated using 64 sample per period each of 6 bit. The sine values are assigned to a variable, and this variable sends data to host using xSCOPE. Using xSCOPE real-time in xTIMEcomposer studio 14, a sine wave can be observed. By adjusting the Window and Div parameters on xscope display, the sine wave can be observed. The window and Div parameters can be varied by left and right mouse click on the xSCOPE display.

IDE used: xTIMEcomposer studio 14



```
xScopeEx Default [xCORE Application] xrun (7/16/15, 3:50 PM)
```

Fig 1: Sine wave on xSCOPE real-time.

Project 3: Two Sine wave display on xSCOPE by switch control

This project describes the use of real-time xSCOPE display over real-time events. Two sine waves of different frequencies are generated. Each signal is controlled by an onboard push button switch.

When the switch is not pressed, a default sine signal is generated and the same is displayed. When switch SW1 pressed a different frequency sine wave is generated and it can be observed in xSCOPE display.

IDE used: xTIMEcomposer studio14

Project 4: Hue Filter

This project implements Integer approximation of Hue Filter. The integer approximation of this filter is described in OpenCV documents. The code generates sample red, green, and blue values and sends it to the HSV calculator function. This function returns the calculated HSV values to the main. The functions are implemented on different threads on XCORE, and runs in parallel.

IDE used: xTIMEcomposer studio 14

Project 5: Image transfer and reception through RS232 Serial Communication

The project implements image transfer between PC and XCORE through RS232 serial communication. The on board USB is used in this project. The USB communication class CDC example of XMOS was programmed on XCORE200. This results in turning USB to Virtual COM port. The code has several options to interact with the board such as turning on LED, reading accelerometer values. The default mode of COM is echo mode or ping back mode. Using Python 2.7 and pySerial package, communication between host and computer is established. By using OpenCV the image is read by python and the code transfers image data through serial communication to host and receives back.

The data transfer rate is at 128kbit baud rate, and is slow for image reconstruction.

Tools used: xTIMEcomposer studio 14, Python IDLE 2.7, pySerial, OpenCV

Project 6: Video over USB

The project implements video over USB using video class. The example code from XMOS was available and the same is ported to XCORE200 eXplorer kit. Once the code is programmed to the board, the sample video can be viewed on VLC medial player. Under VLC, select capture device and then select XMOS USB and then click play. The sample video generated by the device with stripes appears streaming on the PC.

Tools used: xTIMEcomposer studio 14, VLC Media player.

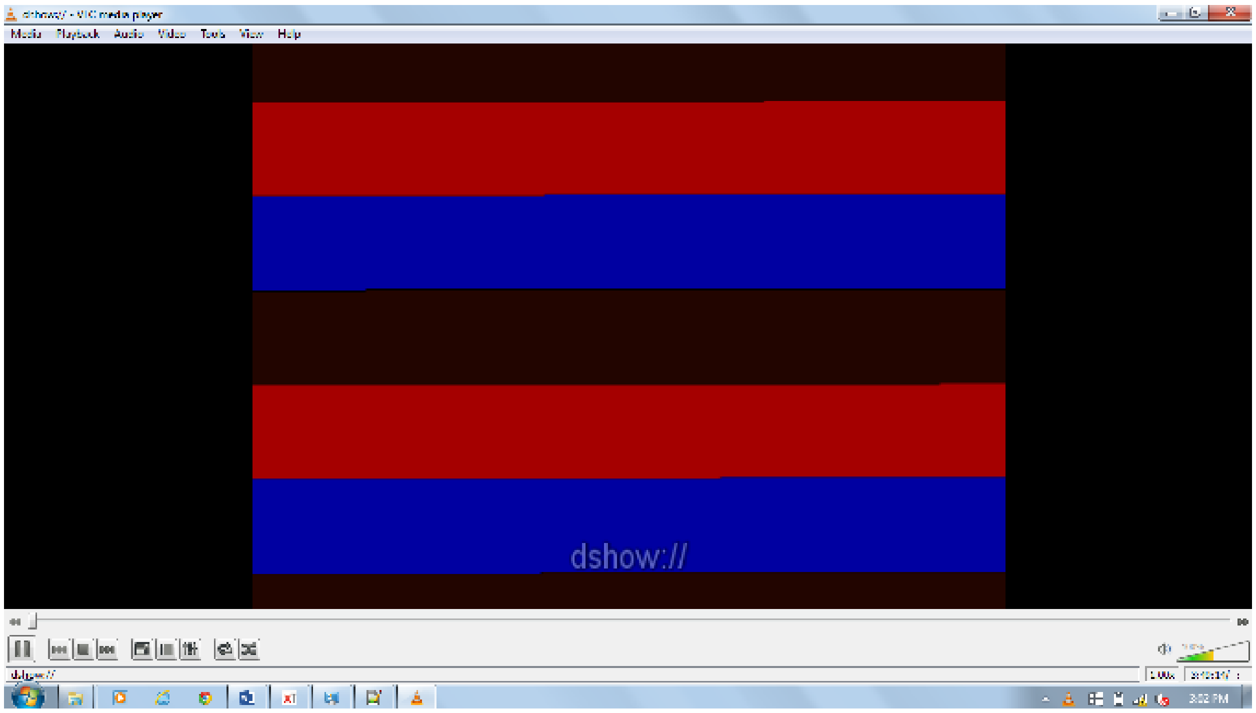

dshow:// - VLC media player
Media Playback Audio Video Tools View Help
dshow://

Fig 2: Sample Video display on VLC

Project 7: USB bulk transfer

This project implements USB bulk transfer on XCORE200. The example code from XMOS, USB Vendor Specific project was modified to ping back the data. On the host side, the code to transfer and receive the data is implemented on python 2.7 with pyUSB package. By setting the USB vendor ID and product ID, USB connection can be claimed. Python forms an USB object and then uses this object to send and receive the data to the device. The default write type in pyUSB is bulk transfer. Isochronous transfer is not implemented on python.

Tools used: xTIMEcomposer studio 14, Python 2.7 IDLE, pyUSB.

Project 8: Image transmission and reception using USB bulk transfer

The project implements image transfer and reception from host to the XCORE device through USB bulk transfer protocol. A sample image of 160 X 120 was used to transmit and receive it through USB.

The host code is implemented in python 2.7 with OpenCV and pyUSB. OpenCV implements image reading and displaying methods, while pyUSb implements data transmission and reception using USB bulk transfer protocol. The total time to transmit and receive the image for 160 X 120 image was around 1 second. The code was tested for different image sizes and the delays were proportional to the image sizes. The image is sent by sending red, green, blue values of each pixel by forming a packet. This packet consists of data in RGB format with fixed length. Once the data is received, it is decomposed to R, G and separately and then stores in corresponding pixel location. The image is displayed once all image data for the pixels are received.

Tools used: xTIMEcomposer studio 14, Python 2.7 IDLE, pyUSB, OpenCV

Project 9: Sample image display on host through xSCOPE

This project implements sample image display on the host PC through xSCOPE. XSCOPE sends data from XCORE to host through JTAG. The connection between the host and XCORE can be established by setting the address of the board in host and configuring the same on XCORE board. The code generates a sample image of four vertical stripes with RED, GREEN, WHITE and YELLOW. The code sends the values of RGB simultaneously according to pixel location. The host receives these values of RGB and then puts into RGB matrix using OpenCV. OpenCV displays the image in real-time. The image size used was 200 X 200 size. The project is developed on Linux OS.

Tools used: xTIMEcomposer studio 14, xSCOPE, Eclipse C++, OpenCV

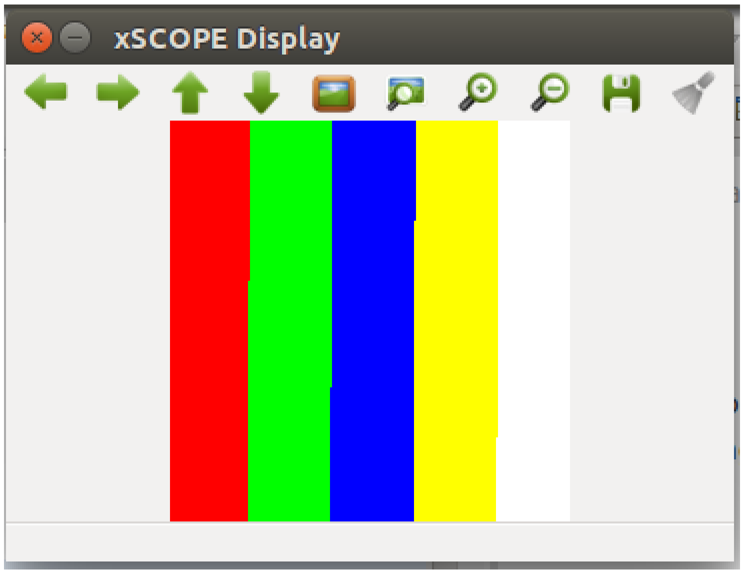


Fig 3: Image display on host.

Project 10: Sample video display on host through xSCOPE

The project implements sample video display on the host PC through xSCOPE. XSCOPE sends the data from XCORE to the host through JTAG. The code sends RGB values to the xSCOPE in real-time. All the values of RGB are sent to the host simultaneously. The code generates vertical stripe of red, green, yellow and white sequentially and sends it to the host. The host uses OpenCV to construct the image received from the XCORE device and forms image. This image is displayed in real-time with a 27ms of delay. The size of the video displayed was 200 x 200 in size. The project is developed on Linux OS.

Tools used: xTIMEcomposer studio 14, xSCOPE, Eclipse C++, OpenCV

Project 11: Displaying image captured by Camera slice on host

This projects implements interfacing of camera slice card with XCORE200 slice kit. The Camera module has Aptina MT9V034 CMOS sensors. This module is connected to XCORE200 slice kit on circle slot. The camera registers can be configured and accessed using I2C protocol. The camera has several modes of operation. The camera is configured in slave mode. The camera has control signals such as EXPOSURE, STLN_OUT, and STFRM_OUT. EXPOSURE is used to capture the image. By setting the EXPOSURE signal to high, the camera captures one frame. After the integration time, STFRM_OUT pulse is applied to initiate frame read. STL_OUT pulses are applied to read one row. The pulse generates LINE_VALID signal, and outputs each pixel data for every PIX_CLK pulses. The data sheet can be referred for detail timing and waveform analysis. The program reads one row of the data and stores in a buffer. After complete row readout the buffer value is sent to the host through xSCOPE. The image data by the camera is in bayer format. The host receives the data and stores in a 2D matrix. OpenCV is used to construct the image matrix. The image is stored in a bayer format after a frame is read out, the OpenCV function converts bayer to BGR format. OpenCV uses BGR format for displaying images. The transformed image is displayed on host. The project is developed on Linux OS.

The project is not completed, there are some issues on the host side when the image is getting constructed.

Tools used: xTIMEcomposer studio 14, xSCOPE, Eclipse C++, OpenCV

Project 12: Real time object detection and tracking.

The project implements real time object detection and tracking using HSV filter in OpenCV. The object is detected based on its color. The image is captured by using laptop webcam. The image is converted to HSV color space. A control window is designed to set the threshold value of HSV for the object. The HSV color space image is then processed for opening and closing operation. After these operation, the area moment of the object is calculated. A threshold is set for area moment and if the value is greater than the threshold, then object is detected. The position of the object is determined by its centroid. A green dot is placed on the centroid when the image is reconstructed and displayed on screen. As the program reads second frame, it joins the previous centroid point with the current centroid and thus tracks the image in real time. The program is developed in C++ on Linux OS.

Tools used: Eclipse C++, OpenCV

Project 13: Object detection in image

This project implements object detection in images. The image is captured using laptop's webcam. The image is the converted to HSV color space. The image is the detected by setting threshold on Hue and saturation components. Pixel location that fall under the threshold region are counted. The mean of these pixel location is calculated. This gives the image centroid. A cross mark is applied with the center on the image centroid. The detected image is displayed on the screen. The project is developed on MATLAB

Tools used: MATLAB.

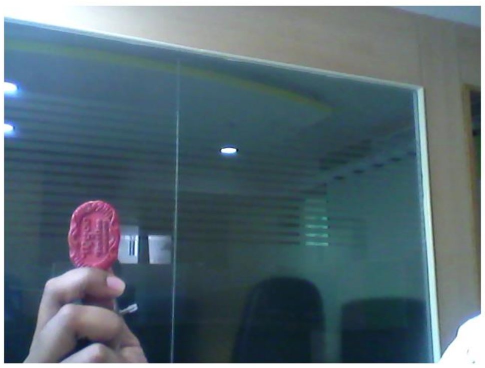
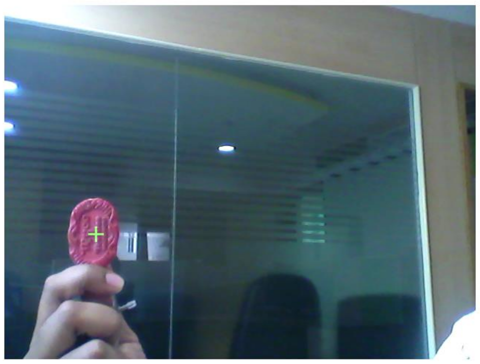



Fig 3: a) Red keychain is object b) Object detected with green cross.

Project 14: Skin pixel detection:

This project implements human face detection using skin pixel detection technique. A sample image is read in the program and it is converted to HSV color space. The skin pixel have a unique Hue and saturation component .In images, face is exposed more than any other parts. The pixels whose Hue and saturation value under this threshold are detected. The detected pixels is represented in green on the display. The project is developed on MATLAB.

Tools used: MATLAB.