LUND UNIVERSITY

Department of Computer Science

Design of Embedded Systems Advanced Course EDA385 - FPGA Piano Project Report

Submitted by:
Amir Shademani(mas09as7)
Hemanth S Prabhu(mas09hpr)
Sherine Thomas(mas09sth)

Abstract

This project report is submitted as part of the Design of Embedded Systems - Advanced Course. The main intension of the project was to implement a simple low cost electric piano on Digilent Nexys2 development board using Xilinx EDK tool. This may be used to teach kids how to play a Piano. The project was done in both hardware and software. The hardware part was done using VHDL and the software part was done using C programming language. This report discusses in detail the architecture and all other implementation details of the project.

Contents

1 Introduction

FPGA Piano project was chosen since it would help us to learn interfacing most of the important components to the Digilent Nexys2 development board. We tried to use Xilinx IP cores where ever possible. The main components of the project are

- PS2 Keyboard
- PMOD AMP1 Speaker/ Headphone Amplifier
- VGA Monitor
- Speaker/Headphone
- Digilent Nexys 2 Board

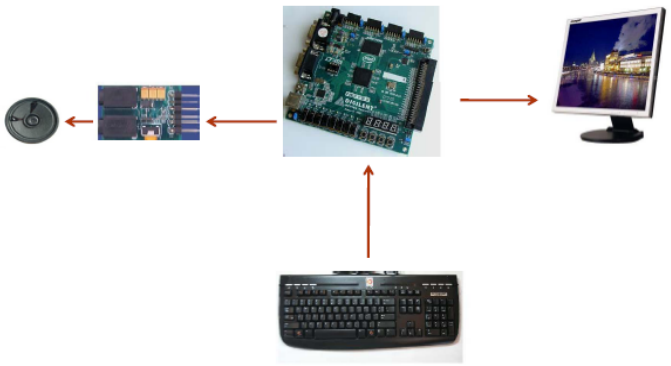

Figure 1: Interface Diagram.

Figure 1 shows the interface diagram of Electric Piano. The PS2 Keyboard is used to read the input keys pressed by the user. The PMOD AMP1 amplifier is used to amplify the PWM output from the FPGA. The VGA monitor is used to display the key pressed and the frequency graph.

We were able to implement almost all the claims made in the proposal. There was an opinion during proposal presentation for adding a feature to handle sound when 2 keys are pressed at the same time. But since PWM input was choosed for PMOD amplifier, we found that handling of multiple frequency at a time is not possible. So we added an additional feature to play songs automatically when the push button switch is pressed. It have 3 pre recorded songs which will be played along with the keys displayed on the VGA monitor. So it will be helpful for kids to start learn playing piano.

1.1 Hardware/Software Partitioning

To control the various components the following methods were used:

- PS2 Keyboard : Xilinx PS2 IP Core v2.00
- PMOD AMP1 Speaker/ Headphone Amplifier : Xilinx Timer IP Core v2.00
- VGA Monitor : Custom IP Core

The VGA part was done using custom IP in the hardware. The decoding of pressed keys were done in the software side. The Software and hardware communication was done using FSL.

1.2 Tools Used

For this project a Digilent Nexys 2 board with a Spartan3E-1200 FPGA is used. Xilinx Platform Studio version 12.2 is used for synthesizing the project. Diligent adept tool was used for downloading the bit file. UART and LED features of the Digilent board were used to debug during the development phase.

2 Architecture

Figure 2 shows the block diagram with all the IP cores and the communication between them. The main part of the project is the MicroBlaze softcore processor. This block is responsible for controlling all the modules and determining the frequency to be played based on the ASCII code from the keyboard controller. Also this block handles the playing of songs automatically. The Xilinx PS2 IP core is responsible for reading the keyboard. It is configured in polling mode. The PS2 IP core is connected to the MicroBlaze using PLB bus. For driving the PMOD amplifier Xilinx timer IP core in PWM mode is used. It is possible to output the PWM signal as an external output. This is then connected to the PMOD amplifier. The VGA was done as a custom IP core, which is then connected to the MicroBlaze processor using FSL bus.

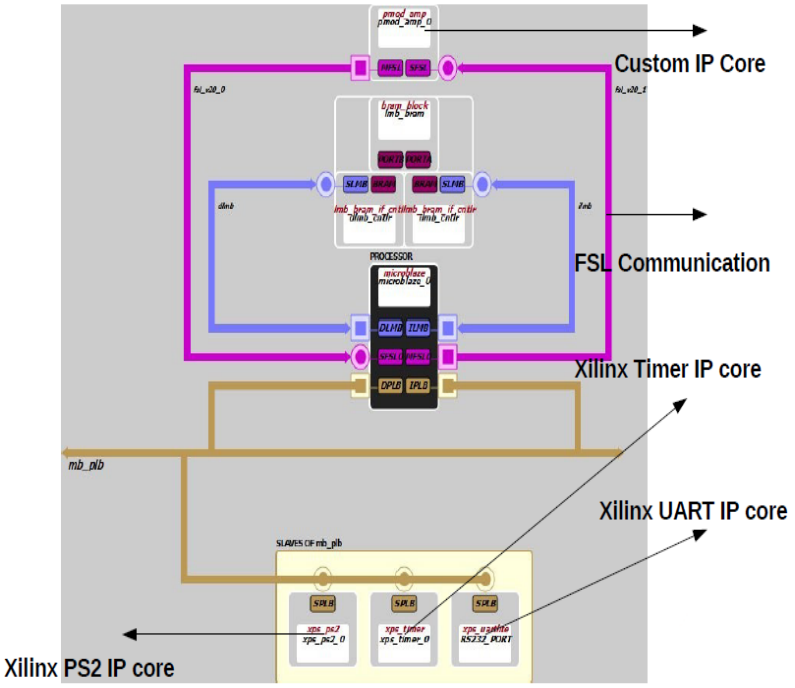

Figure 2: Block Diagram.

The hardware utilization of the project is summarized in Table 1. About 23% of the available slices are utilized in this project.

	Used	Available	Utilization
Number of 4 input LUTs	2632	17344	15%
Number of Occupied Slices	2004	8672	23%
Number of RAMB16s	8	28	28%
Number of MULT18X18SIOs	3	28	10%

Table 1: Device Utilization

3 Hardware

The main parts of the hardware are VGA, PS2 keyboard and PWM output IP cores. VGA is a custom built IP core. Both PS2 and PWM are Xilinx IP cores.

3.1 VGA Controller

The VGA part is responsible to generate synchronizing signals and pixel data and display the FPGA piano keys on the monitor screen. Figure 3 shows the screen shot of VGA.

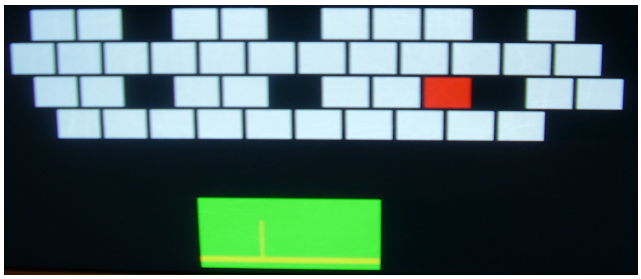

Figure 3: VGA Display.

Every time when a key is pressed on the keyboard, it will be highlighted on the VGA screen with red color. The code for VGA generator is divided into two parts: VGA synchronizer and Pixel generator.

VGA synchronizer

The pixel data sent to the monitor is synchronized by two synchronizing signals, H-sync and V-sync. This standard comes from the CRT display monitors, where the pixels were displayed in a raster scan procedure so the Horizontal raster and Vertical raster were controlled by H-sync and V-sync signals. In our project we use a 640-by-480 VGA mode, which means we should have approximately 25MHz pixel rate. Since we use a 50MHz onboard clock signal, so the clock rate to the VGA unit is divided by two to gain our desired VGA mode.

Apart from sending out the synchronization data to the display monitor, this part of the code also send the location of pixel data to the next part which is pixel generator section. Pixel generation circuit uses the pixel location data and turns the pixel on or off to display the desire keyboard scheme on the screen.

Pixel generator

To make the VGA display simple, we used 1 bit output data for red, blue and green outputs each. The location of keyboard layout to be displayed on the VGA is hard coded. So when the pixel location data is received from the VGA synchronizer, the pixel generator checks whether it is the location to be displayed. If so the pixel generator will generate the pixel color and brightness. Otherwise the output signal from pixel generator will be blank. Also when a key is pressed, the key location is received from the processor. Then Red color will be displayed in the corresponding location. The display monitor receives pixel data in analog signal and one signal line for basic colors: Red, Green and Blue. The Nexys2 FPGA board accepts two bits for each color and has onboard DAC to generate related analog signals for color signals. There are three main categories for coding the VGA pixel generator: Bit-mapped scheme, Tile

mapped scheme and Object-mapped scheme. In this project, because of the simplicity of the display data we used the Object-mapped scheme and turn on/off the pixel according to the position of the pixel scan data. In order to have the compact coding, the pixel generator circuit is implemented using for loop and proper constants are defined to make it easier for further modifications.

3.2 PS2 Controller

Xilinx PS2 IP core can be configured to control keyboard or mouse. In this project it is configured to read data from the keyboard. The slave PS2 controller is connected to the MicroBlaze processor using PLB bus. The data is read from keyboard in polling mode. The functions used to configure PS2 controller are:

- XPs2_LookupConfig()
- XPs2_CfgInitialize()
- XPs2_SelfTest()
- XPs2_Recv()

Functions XPs2_LookupConfig and XPs2_CfgInitialize are used to initialize the PS2 controller. XPs2_SelfTest function is used to do a self test on the PS2 controller for any errors. XPs2_Recv function is used to read the data from the PS2 controller.

3.3 PWM Output Controller

Xilinx timer IP core is used to output PWM signals to the PMOD amplifier. The Xilinx timer IP core is connected to the MicroBlaze processor using PLB bus. Xilinx counter/timer IP core can be configured either in

- Generate Mode
- Capture Mode
- Pulse Width Modulation Mode

In this project the Xilinx counter/timer IP core is configured as Pulse Width Modulation mode. The PWM generated by the IP core is connected to the PMOD amplifier. The functions used to configure Xilinx timer/counter are:

- XTmrCtr_Initialize()
- XTmrCtr_SelfTest()
- XTmrCtr_WriteReg()
- XTmrCtr_SetLoadReg()

XTmrCtr_Initialize() function is used to initialize the XPS timer IP. XTmrCtr_WriteReg function is used to write to the timer register for configuring it in the PWM output mode. XTmrCtr_SetLoadReg function is used to set the duty cycle and period of the PWM output.

3.4 Output Port Configuration

The ucf file was modified to add all the input and output ports for the project. The ports added to the ucf file are:
Keyboard Clock LOC = R12 - This pin is for supplying clock to the PS2 keyboard.
Keyboard DATA LOC = P11 - This pin is for reading data from the keyboard.
PMOD Amplifier Right LOC = L15 - This is the PMOD amplifier right output pin.
PMOD Amplifier Left LOC = L17 - This is the PMOD amplifier left output pin.
Switch Pin 0 LOC = H13 - This is the input switch for playing song 1.
Switch Pin 1 LOC = E18 - This is the input switch for playing song 2.
Switch Pin 2 LOC = D18 - This is the input switch for playing song 3.
VGA RGB Pin 0 LOC = R8 - This is the VGA Red output pin.
VGA RGB Pin 1 LOC = P6 - This is the VGA Green output pin.
VGA RGB Pin 2 LOC = U4 - This is the VGA Blue output pin.
VGA HSync LOC = T4 - This is the VGA output for horizontal synchronization.
VGA VSync LOC = U3 - This is the VGA output for vertical synchronization.

4 Software

The main controlling part was done in the software. All the communication between IP cores were controlled by the MicroBlaze processor. The frequency controller and the module for playing songs were implemented in software.

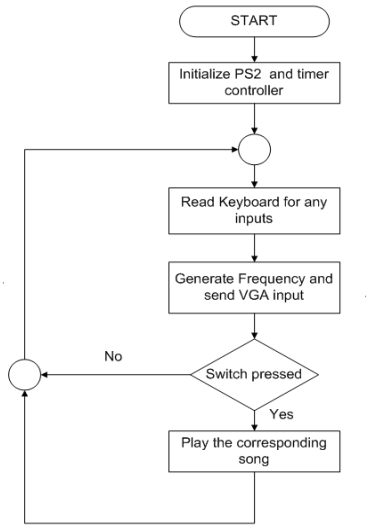

Figure 4: Flow Chart.

The Figure 4 shows the flow of function calls in the MicroBlaze processor. The first step after power on is to initialize the PS2 controller and PWM output module. The other operations are scheduled in polling mode. So all the other functions will be called in a cyclic manner. The first step is to check the PS2 controller for any input keys. If any input key is pressed, the PS2 controller will send the ASCII code of the corresponding key. Then the ASCII code is processed by the Frequency controller, which will decide the frequency which is to be generated based on the ASCII code. Then based on the ASCII input, data is send to the VGA controller using FSL communication. The custom IP core module will send the status of switch inputs to the MicroBlaze processor. By decoding the key status it is possible to decide whether any key is pressed. If any of the input keys are pressed, the corresponding song will be played. The operation will continue in a cyclic fashion restarting again from the PS2 controller.

4.1 Frequency Controller

The frequency controller will read the ASCII code from the keyboard driver and detects the frequency which needs to be played. For each specific key a specific frequency is allocated. Figure 5 shows the frequency notes of piano, implemented in this project. Only a specific set of notes from the piano frequency notes is taken for implementation due to the restriction in the number of keys which can be implemented using the keyboard. The minimum frequency able to be generated by the FPGA piano is 415.30 Hz and the maximum frequency is 2349.3 Hz.

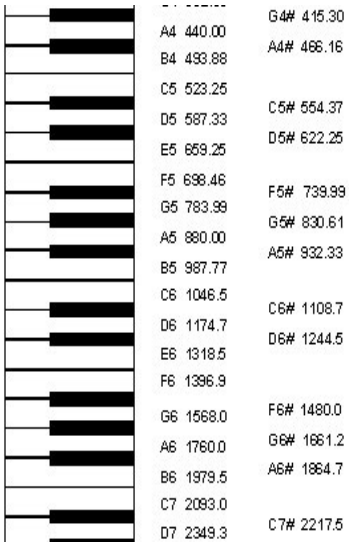

Figure 5: Pinao Notes.

4.2 Play Songs

This module is responsible for playing the songs automatically based on the switch inputs. This project is capable of playing 3 songs based on the keys. A delay function is used to adjust the timing between each note played. When the push button switch is pressed, the song corresponding to that button is played by the piano. When the songs are played all other operations are suspended.

5 User Manual

The following components are required for running the FPGA piano:

- PS2 keyboard
- VGA Monitor
- PMOD Amplifier
- Digilent Nexys2 board

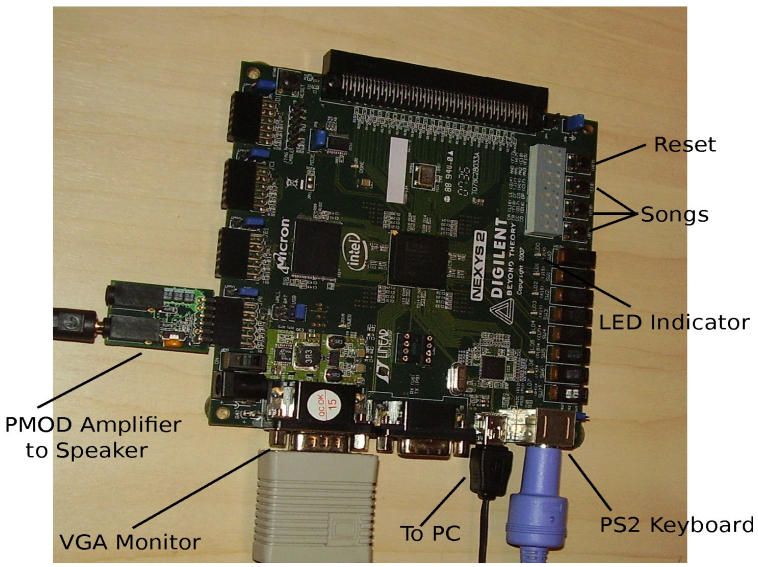

Figure 6: Pheripheral Connections.

Figure 6 shows the interface of different pheripherals with the Digilent Nexys 2 board. Make sure to make the connections as shown in the figure especially the PMOD amplifier board connection. The current project is built using Xilinx XPS 12.2. To update the bit stream the project needs to be opened in xilinx XPS 12.2.

Follow the following steps to run the FPGA piano project.

1. Connect all the hardware components.
2. Download the bit file using Digilent Adept tool.
3. To play the piano refer to the key mapping given in Figure 7 and Figure 8.
4. To play songs automatically press one of the first 3 push button switches. Take care the 4th push button is the reset button.
5. You can learn how to play by automatically playing a song. The keys to be pressed for playing the song will be highlighted in the VGA. Then try to play the same song by yourself.

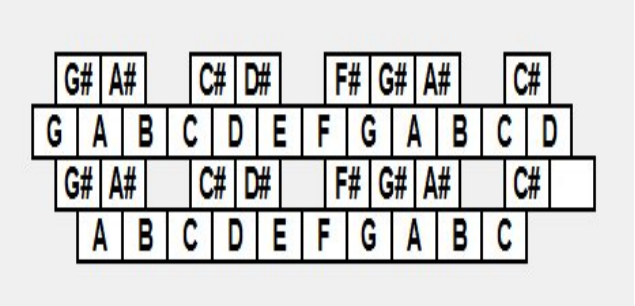

Figure 7: Piano Note Layout.

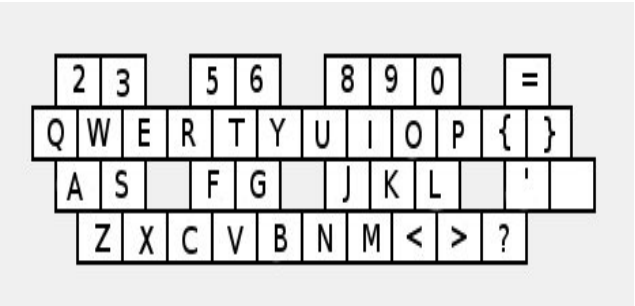

Figure 8: Keyboard Layout.

Figure 7 shows the mapping of piano notes to the keyboard. The lower layer of keys are mapped to low frequency and the upper layer of keys are mapped to the higher frequency. Figure 8 shows the keyboard mapping. The key mapping have been adapted similar to that of piano for easy usage.

6 Problems Encountered and Lessons Learned

We found it difficult to use xilinx IP core due to the lack of good data sheet. The documentation available is not so easy to understand. There are not enough examples which shows how to configure the IP core. From the blogs we found some examples of configuring the PS2 controller. But it was very difficult in getting the details for configuring the timer core in PWM mode.

There was also problems in handling project created with different versions of Xilinx XPS. We did the project initially in Xilinx XPS 10.1 version. Later when we tried to migrate to Xilinx XPS 12.2 version, we got lot of errors. The IP core versions available in the new version was not matching with the current version. Even after migrating to the new version using the Migration tool we were getting errors regarding the IP core version. Later with the help of Flavius, we were able to sort out the issue. We had to manually edit the system.mhs and system.mss files with the new version number of the IP core for successfully synthesizing the project.

7 Contributions

The project was split mainly into 3 parts, VGA, keyboard & PMOD Amplifier and PWM output. The VGA part was done by Amir. The configuration of keyboard and PMOD amplifier were done by Sherine. The PWM output part was done by Hemanth. The report and the presentation was done by all the 3 group members.

8 Conclusion

Even we were familiar with the usage of Xilinx XPS from the EDAN15 course, we didnt use lot of features which were available in the tool. Even though some features of the Xilinx XPS tool are not much user friendly, we were able to learn more features. By doing this FPGA Piano project, we were able to learn interfacing most of the important hardware components. This was a great learning experience in the terms of issues that can happen during actual implementation. Also we found that the IP cores provided by Xilinx really useful. Even though we found slight difficulty in configuring the IP cores initially, it is a very useful thing for reducing the design time of the project. Even though the sound produced by the FPGA Piano is not exactly similar to original piano, we were able to learn the basics of piano. It will be a nice starter tool for kids.