ADVANCED DIGITAL LOGIC DESIGN WITH VERILOG AND FPGA

EC-551

PROJECT REPORT

THE LAST DRUMMER

Prof Douglas Densmore

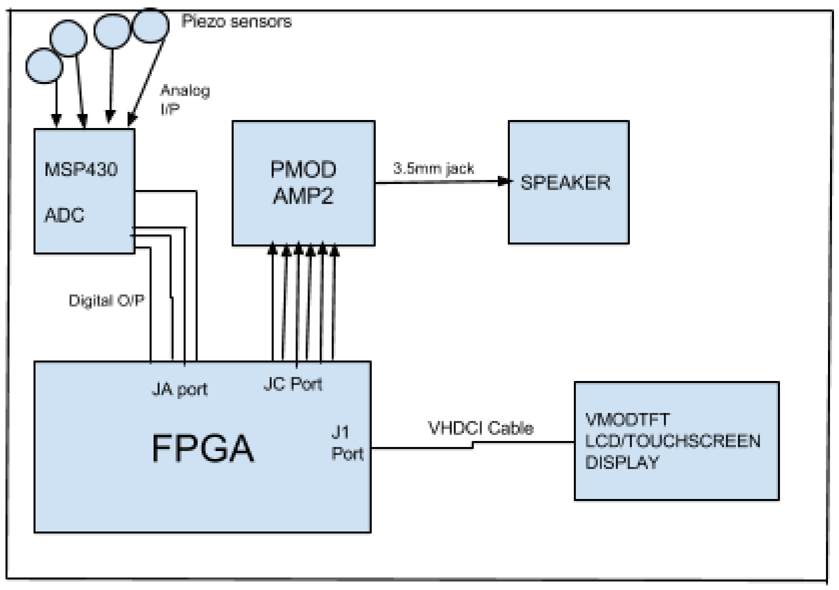
BY:
ANISHA DATLA
DAMINI CHOPRA
GAURAV SUTHAR
MOHAMMED AFZAL SOPARIWALA
SIDDHESH KULKARNI

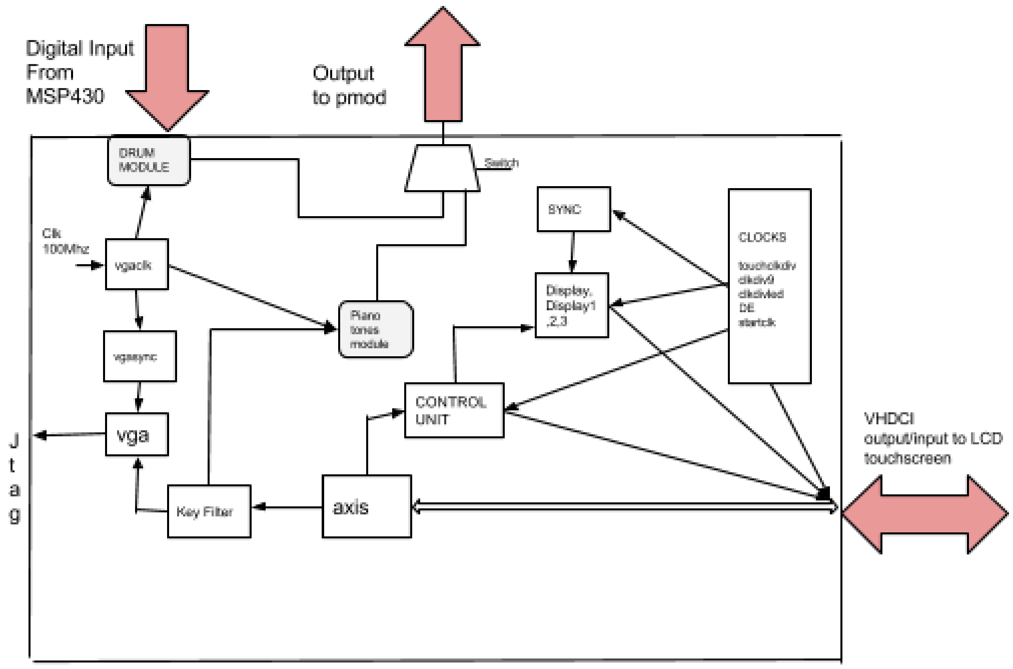
Introduction:

We created a Digital Drum and Piano set using the Nexsys 3 Spartan 6 Board. The Drums are created using piezo sensors, When the drums are hit, the piezo generates a voltage; this is converted to digital signal using an ADC. This digital signal is sent to the FPGA to which plays a different beats depending on which drum is hit.
The piano is created using the LCD touch screen. Similar to the Drum, depending on which key is pressed the FPGA plays its respective note.
The notes of the Piano and the Drums were generated by manipulating with the input clock frequency to get multiple different frequencies.

Block Diagram:

Top Level





Detailed Functioning(Verilog Modules):

Main.v *[module main(input[3:0]drum,input select, sw1,Clk,rst,interrupt,DOUT,busy,output [7:0]data_RED,output [7:0]data_BLUE,output [7:0]data_GREEN, output TP_DCLK, TP_CS, DISP, TFT_en,DE_clk,clk_lcd,backlight,vga_h_sync,vga_v_sync,vga_R, vga_G, vga_B,DIN,output [3:0]pianokey, led,output speaker,gain,shutdown,i);]*

The main module combines all the modules together. It also combines all the RGB values from the different display modules by ORing all the different outputs. It assigns the PMOD-AMP2 either the piano tones or the drum tones depending on the switch value.

touchclkdiv.v *[module touchclkdiv(input clk,DCLK_en,output reg TP_DCLK);]*

TP_DCLK needs to be set between 10khz-2Mhz to drive the touch screen. This module divides the 100 MHz by 2^13 using a counter to get 12 KHz clock. This is enabled when the interrupt goes low initially.

clkdiv9.v *[module clkdiv9(input pixel_en,Clk,output reg clock_lcd);]*

TFT_CLK(clock_lcd) should be driven at 9 MHz to drive the LCD display.This module divides the 100 MHz clock to get the 9 MHz clock. The clock_lcd is high for 6 counts and low for 6 counts of the input clock.

clkdivled.v *[module clkdivled(input clk,led_en,output reg clk_out);]*

LED_on drives the backlight of the lcd. This module divides the 100 MHz by 2^11 to get the pwm clock. The clock is only enabled in the fourth stage of the control unit.

DE.v *[module DE(input de_en,Clk,output reg DE_clk);]*

This module generates the TFT_DE clock to enable scanning on the lcd. It is a combination of hsync and vsync signals. This clock is enabled in the second stage of the control unit.
The clock is high for 272 counts and low for 16 counts.(for vertical scan). Within each of 272 counts the clock is low for 45 periods and then toggles for 480 periods.(for horizontal scan).

vgaclk.v *[module vgaclk(input clk, output reg pixel_clk);]*

pixel_clk needs to be set between 25 MHz to drive the vga display. This module divides the 100 MHz by 4 using a counter to get 25 MHz clock.

vgasync.v *[module vgasync(output reg vga_h_sync, vga_v_sync, inDisplayArea, [9:0]CounterX, [8:0]CounterY,input pixel_clk);]*

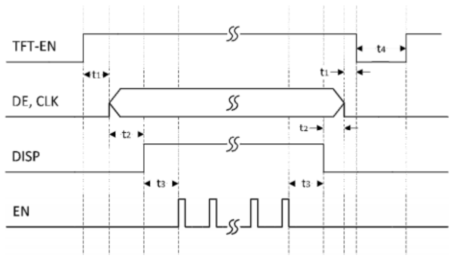
This module drives the hsync and vsync of the vga display.

vga.v *[module vga(output vga_R, vga_G, vga_B,input [3:0] num,input [9:0]offset, input inDisplayArea, input [9:0]CounterX, input [8:0]CounterY, inputpixel_clk);]*

Displays a 7-segment implementation of the input num. Multiple numbers can be displayed by calling the function with different numbers and offsets.

controlunit.v *[module controlunit (input lcdoff,clk_out,rst,output TFT_en,de_en,disp_en, led_en, rgb_en, pixel_en,en_sync);]*

To power up the LCD the following sequence has to followed:



This module sets the signals at each state as pictured in the figure. When the lcdoff signal becomes 1, it turns off the signals similarly.

display.v,display1.v,display2.v,display3.v *[module display(offset,rgb_en,clk_lcd, flagv, flagh, data_RED, data_BLUE, data_GREEN, hcount_reg,Vcount_reg);]*

All the display modules have the horizontal count and the vertical count as an input. These three modules are used to display the piano on the LCD.

sync.v *[module sync(disp_en,de_clk,en_sync,clk_lcd,flagh, flagv, hcount_reg, Vcount_reg, DISP);]*

This module generates the synchronization signals for the LCD which is used to display the piano.

startclk.v *[module startclk(input clk,output reg startclk);]*

This module outputs a 12 KHz clock that is used to enable the various signals for the LCD display. Used by the control unit.

axis.v*[module axis(input busy, interrupt, TP_DCLK, DOUT, clk, output reg DCLK_en, lcdoff, DIN, TP_CS, output reg [3:0]key, output reg [11:0]xaxis, yaxis);]*

This module initially enables the TP_CS and DCLK_en when the interrupt occurs.
Then at every negedge of the TP_DCLK it sends data (start bit,axis,mode,power mode) serially through the DIN pin.Once the DIN id sent the touchscreen sets a busy bit for one clock cycle.
After the busy bit the x-axis and y-axis values can be sampled through the DOUT pin serially and stored in a reg once all the 12 bits have been transferred.While sampling we turn off the lcd display to reduce the touchscreen noise.
Finally depending on the xaxis and yaxis the piano key is selected.

keyfilter.v *[module keyfilter(input TP_DCLK,input [3:0]key,output reg[3:0]pianokey);]*

This module filters out the noisy output from the touchscreen. It continously takes the difference of the previous and current key. Only if the difference is zero for at least 64 clock cycles it outputs the key number else it outputs a 0.

pianotones.v *[module music(input sysclk, input [3:0] piano , output speaker, output [3:0] led)]*

This module is used to generate the exact pianotone depending on the 4 bit input which we get from the LCD module. This main module incorporates the modules music_rom and clk_divby12. The value of "note" from the music_rom module which is then divided by 12 in clk_div12 module, we get a pair of scale number and octave number. A particular value is assigned against each scale number and octave number which is called as 'counter_note' and 'octave' respectively. A particular frequency is generated by dividing the clock first by counter_note and then by octave. In this frequency the speaker pin is toggled which leads to generating a particular tone.

music_rom.v *[module music_rom(input [3:0] piano, output [7:0] note)]*

This module is used to assign a particular value for "note" depending on the inputs from the pianotones module. Depending on which key is pressed, a particular value is assigned to the variable "note". This value is assigned in such a way that we get a particular scale sequence of a constant octave.

clk_divby12.v*[module clk_divby12(input [5:0]note, output [2:0]octave, output [3:0]counter_note)]*

This module is basically used to separate out the value of note in terms of counter_note number and octave number. In this module only the first 6 bits of the 8 bit note is taken from the music_rom module. This value is then divided by 12. In order to do so, it is first divided 4 i.e. the bottom 2 bits are attached as it is. Then it is divided by 3 and the value is separated as quotient and remainder. The quotient is the value of octave number and the remainder two bit value along with the two bits separated by dividing by 4 forms the counter_note number value. This values are then used to assign a particular scale and octave.

durm.v [module drum(input clk,input [3:0]drum,output reg [3:0] led,output reg speaker);]

Similar to the piano tones module, this module is used to generate drum tones. Depending on which drum has been played, respective value is assigned to the variable “note” at that instance. The difference between piano tones and drum tones is that, the values assigned to note for piano are sharper whereas those assigned to drums are of lower octaves giving a low frequency elongated tone.

Reference:

1. http://www.digilentinc.com/Data/Products/VMOD-TFT/VmodTFT_rm.pdf
2. http://www.farnell.com/datasheets/1780769.pdf
3. https://www.digilentinc.com/Data/Products/NEXYS3/Nexys3_rm.pdf