Red Sox Final Project Report: 3D Motion Controller

The 3D motion controller project is comprised of two demo projects. One project demonstrates the gyro and accelerometer integrated with ball rocking on a line and a 2D maze. The second demo project displays a cube that rotates in response to the gyro and accelerometer inputs. The hierarchy of the modules have been attached to the Appendix at the end of the report.The Line Drawing module is the fundamental block of the project. The two designs are explained after the line drawing module

clk_divider.v

This module divides the input clock by P_DIV_FACTOR and outputs a lower frequency clock.

Port	Type	Description
i_clk	Input	Input clock
o_clk	Output	Output clock (divided clock based on input clock

Parameter	Default	Description
P_DIV_FACTOR	4	What to divide input clock by

button_debounce.v

This module debounces the input from a push button by using a slow clock to sample the button input.

Port	Type	Description
i_clk	Input	Input clock
i_button	Input	Input from button (raw and non-debounced)
o_button_state	Output	Debounced button output

Parameter	Default	Description
IS_ACTIVE_LEVEL	1	Which level indicates the active level
P_CLK_FREQ	100000000	Frequency of i_clk

counter.v

This module implements a simple counter with reset.

Port	Type	Description
i_clk	Input	Input clock
i_reset	Input	Reset counter
o_count	Output	Counter value

Parameter	Default	Description
P_COUNT_W	16	Width of the counter register

Line Drawing Module

This module has the Verilog files for drawing an arbitrary line

dp_ram.v

This module implements a dual port RAM that the Xilinx tools will infer and place in the BRAM resources.

Port	Type	Description
i_clk	Input	Input clock
i_a_wr	Input	Write strobe
i_a_addr	Input	Address
i_a_data	Input	Input data (for write)
o_a_data	Output	Output data (for read)
i_b_wr	Input	Write strobe
i_b_addr	Input	Address
i_b_data	Input	Input data (for write)
o_b_data	Output	Output data (for read)

Parameter	Default	Description
P_DATA_W		RAM data width
P_LOG2_RAM_DEPTH		Log_2(depth of RAM)

fifo.v

This module implements a FIFO. The memory of the FIFO is implemented using the dp_ram.v module.

Port	Type	Description
i_clk	Input	Input clock
i_reset	Input	Reset
i_data	Input	Write data
i_read	Input	Read strobe
i_write	Input	Write strobe
o_data	Output	Output data
o_empty	Output	Flag to indicate that FIFO is empty
o_full	Output	Flag to indicate that FIFO full (additional writes will be dropped)

Parameter	Default	Description
P_DATA_W	11	RAM data width
P_ADDR_W	8	Address width = Log_2(depth of FIFO)

VGAcontrol.v

This module controls the VGA display.

Port	Type	Description
pixel_clk	Input	Input clock
HS	Output	Horizontal sync
VS	Output	Vertical sync
hcounter	Output	Horizontal counter
vcounter	Output	Vertical counter
blank	Output	Indicates if pixel is blank

Parameter	Default	Description
HMAX	800	maxium value for the horizontal pixel counter
VMAX	525	maxium value for the vertical pixel counter
HLINES	640	total number of visible columns
HFP	648	value for the horizontal counter where front porch ends
HSP	744	value for the horizontal counter where the synch pulse ends
VLINES	480	total number of visible lines
VFP	482	value for the vertical counter where the frone proch ends
VSP	484	value for the vertical counter where the synch pulse ends
SPP	0	

bresenham.v

This module outputs coordinates for points that are illuminate on a line between two points. This component uses a simple state machine to implement the Bresenham algorithm. A pair of x,y coordinates for the points defining a line are loaded into the state machine that outputs a sequence of the coordinates of points that are illuminated along the line.

Port	Type	Description
i_clk	Input	Input clock
i_reset	Input	Reset
i_x0	Input	Input 0 x-coordinate
i_x1	Input	Input 1 x- coordinate
i_y0	Input	Input 0 y-coordinate
i_y1	Input	Input 1 y-coordinate
i_load_vals	Input	Input strobe to load values and initiate Bresenham state machine
o_x_val	Output	Output x-coordinate (will generate a sequence of these)
o_y_val	Output	Output y-coordinate (will generate a sequence of these)
o_vals_rdy	Output	Indicates that the output value is valid
o_waiting	Output	Indicates that the bresenham module is ready to receive another pair of points

Parameter	Default	Description
P_X_COORD_W	11	Width of the x coordinate value
P_Y_COORD_W	11	Width of the y coordinate value

draw_lines.v

This module uses the Bresenham module, 4 FIFOs and a dp_ram buffer to display lines on the VGA display. The FIFOs buffer the points that define multiple lines. On a strobe of i_clear_buffer, the screen buffer is cleared and then the points in the FIFO are each run through the Bresenham module and the points generated by the Bresenham module are loaded into the screen buffer that is implemented using dual port RAM.

Port	Type	Description
i_clk	Input	Input clock
i_reset	Input	Reset
i_x0	Input	Input 0 x-coordinate
i_x1	Input	Input 1 x- coordinate
i_y0	Input	Input 0 y-coordinate
i_y1	Input	Input 1 y-coordinate
i_clear_buffer	Input	Input strobe to refresh the screen buffer with new lines

i_load_fifo	Input	Input strobe to load a value in the FIFOs
o_waiting	Output	Module is not writing lines to the screen buffer and can accept data
o_fifo_full	Output	The FIFOs are full. Additional points written will be dropped.
i_hcounter	Input	Horizontal counter from the VGA controller
i_vcounter	Input	Vertical counter from the VGA controller
o_pixel_on	Output	Indicates if the pixel at (i_hcounter, i_vcounter) is illuminated

Parameter	Default	Description
P_X_COORD_W	11	Width of the x coordinate value
P_Y_COORD_W	11	Width of the y coordinate value
P_SCREEN_W	640	Width of VGA display
P_SCREEN_H	480	Height of VGA display
P_DATA_W	1	Width of screen buffer data
P_LOG2_RAM_DEPTH	19	Log_2(depth of FIFO)

Design 1: Rocking Ball and 2D Maze and Ball Game

This design contains 2 demos:

1. Rocking ball and movi+ng cursor demo- The rocking ball-on-line demonstrates gyroscope interfacing and the moving cursor demonstrates accelerometer interfacing.
2. 2D Maze- The idea is to move a baseball through a 2D maze using the gyroscope attached to the FPGA board.

top.v:

This is the top level module of design 1. There are 5 main components in this module, gyro_interface, acl_interface, display_controller, task1 and task3. Gyro_interface and acl_interface contain the SPI protocol logic to interface with the sensors. The display_controller module displays data to the SSD. Task1 contains the logic for the ball-on-line and moving cursor animations. Task3 contains the logic for the 2D Maze.

Port	Type	Description
sw	Input	**Switch T10** toggles the reset signal **Switch T9** toggles the start signal **Switch V9** toggles the display for the SSD (Bit 0) **Switch M8** toggles the display for the SSD (Bit 1) **Switch N8** toggles between hex or decimal on SSD display
chooseDemo	Input	Toggles between demo 1 and demo 3 **(Switch B8)**
choosePart1Mode	Input	Toggles between ball-on-line and moving cursor parts for demo 1 **(Switch V8)**
clk	Input	Board clock
switchShape	Input	Switches between shape displayed for ball-on-line for demo 1 **(Switch B8)**
mazeMode	Input	Toggles between blinking map and fully displayed map for demo 3 **(Switch U8)**
an	Output	SSD anode
seg	Output	SSD cathode
dp	Output	SSD decimal place
JA	Inout	JA PMOD (Gyro)
Jb	Inout	JB PMOD (ACL)
hsync	Input	VGA hsync
vsync	Output	VGA vsync

red	Output	VGA red
green	Output	VGA green
blue	Output	VGA blue

GYRO_Interface.v

The GYRO_Interface contains a serial port interface controller and spi interface pair, which provides an interface to our gyroscope to retrieve values from the sensor.

Port	Type	Description
clk	Input	Input clock
rst	Input	Reset
start	Input	Start signal
miso	Input	Master input, slave output
x_axis_data	Output	X axis gyro data
y_axis_data	Output	Y axis gyro data
z_axis_data	Output	Z axis gyro data
mosi	Output	Master output, slave input
sclk	Output	Serial clock
slaveSelect	Output	Slave select (output from master)

display_controller.v

The display controller takes 4 input data values- 1 8-bit value and 3 16-bit values. Depending on the value of sel, the SSD will display 1 of the 4 input data values.

Port	Type	Description
clk	Input	Input clock
rst	Input	Reset
sel	Input	Selects which axis to display
temp_data	Input	Temperature data
x_axis	Input	X-axis data
y_axis	Input	Y-axis data
z_axis	Input	Z-axis data
display_sel	Input	Hex or decimal display
dp	Output	Decimal point
an	Output	Anode
seg	Output	Cathode

ACL_Interface.v

The ACL Interface contains a SPI master module which controls data transfer to and from the sensor, SPI interface for reading and writing data to sensor, and a slave select module which enables/disables sensor communication with the board.

Port	Type	Description
CLK	Input	Input clock
RST	Input	Reset
SDI	Output	Master input, slave output
SDO	Output	Master output, slave input
SCLK	Output	Serial clock
SS	Output	Slave select (Output from master)

xAxis	Output	X-axis acl data
yAxis	Output	Y-axis acl data
zAxis	Output	Z-axis acl data

task1.v

The task1 module contains the logic for running the ball-on-line demo as well as the moving cursor demo. In terms of user input, the sel input switches between demos, and the switchShape input, when enabled, changes the shape of the "ball" in the ball-on-line demo. We feed in the 3-axis gyro and 3-axis acl data to the respective demos, and output the red, green, blue, hsync, and vsync for the VGA display.

Port	Type	Description
clk	Input	Input clock
rst	Input	Reset
switchShape	Input	User input to switch shape for ball-on-line
sel	Input	Selects either ball-on-line or cursor movement drawing
xAxis_gyro	Input	X-axis gyro data
yAxis_gyro	Input	Y-axis gyro data
zAxis_gyro	Input	Z-axis gyro data
xAxis_acl	Input	X-axis acl data
yAxis_acl	Input	Y-axis acl data
zAxis_acl	Input	Z-axis acl data
red	Output	VGA red
green	Output	VGA green
blue	Output	VGA blue
hsync	Output	VGA hsync
vsync	Output	VGA vsync

vga_controller_640_60.v

This is our vga controller. Given a vga clock signal, we output a hsync, vsync, blank, hcounter, and vcounter.

Port	Type	Description
Pixel_clk	Input	VGA clock
HS	Output	VGA hsync
VS	Output	VGA vsync
Blank	Output	Indicates whether VGA is drawable or not
Hcounter	Output	VGA hcounter
vcounter	Output	VGA vcounter

move_line.v

This is the module contain the sub-modules for the ball-on-line demo. It instantiates the modules getCoords and draw_lines. Module getCoords adjusts the raw gyro values in preparation for the line drawing module, draw_lines. Module draw_lines is where the logic for the demo is contained

Port	Type	Description
clk	Input	Clock
switchShape	Input	User input to switch shape for ball-on-line
x_axis_in	Input	X-axis gyro data

rst	Input	Reset
vga_blank	Input	VGA blank signal
vga_hcounter	Input	VGA hcounter
vga_vcounter	Input	VGA vcounter
vgaRed	Output	VGA Red
vgaGreen	Output	VGA Green
vgaBlue	Output	VGA Blue
Hsync	Output	VGA hsync
Vsync	Output	VGA vsync

getCoords.v

The getCoords module takes the x-axis gyro data as raw values, and adjusts it in preparation for the line drawing module. In order to achieve this, we keep a saved variable incY, which increments the Y value every time we sense a change in the gyroscope. We set a max and min value for incY in order to keep the line's range of motion approximately +- 80 degrees (From horizontal)

Port	Type	Description
clk_in	Input	Clock
rst_in	Input	Reset
x_axis_in	Input	X-axis gyro data
incY	Output	The Y incrementor to update ball-on-line
slowClk	Output	A slower clock used to sample gyro data

move_circle.v

The move circle module contains the logic for the moving cursor demo. This module contains a slowed down clock signal called circle_clk. At each posedge of circle_clk, we update and save the x and y coordinates of the cursor. Given these coordinates, we print to the screen the cursor image at the location of the x and y coordinates.

Port	Type	Description
CLK	Input	Input clock
RST	Input	Reset
xAxisG	Input	X-Axis gyro data
yAxisG	Input	Y-Axis gyro data
vga_blank	Input	VGA blank signal
vga_hcounter	Input	VGA hcounter
vga_vcounter	Input	VGA vcounter
vgaRed	Output	VGA Red
vgaGreen	Output	VGA Green
vgaBlue	Output	VGA Blue
Hsync	Output	VGA Hsync
Vsync	Output	VGA Vsync

sel_data.v:

Converts axis data from 2's compliment to its magnitude representation. Based upon the input switches SW(1) and SW(0) either the x-axis, y-axis, or z-axis data will be output on the DOUT output.

spicomponent.v:

This module has three sub modules :

spimaster.v :

Selects data to be transmitted and stores all data from the PmodACL. It is then made available to the rest of the design on xAxis , yAxis, zAxis outputs

Port	Type	Description
CLK	Input	100MHz Onboard system clock
RST	Input	Main Reset Controller
START	Input	Signal to initialize data transfer
SDI	Input	Serial data in
SDO	Output	Serial data out
SS	Output	Save Select
xAxis	Output	X axis data received from PmodACL
yAxis	Output	Y axis data received from PmodACL
zAxis	Output	Z axis data received from PmodACL

spiinterface.v:

controls the SPI interface with three processes including reception and transmission of data and finally production of serial data clock used in the SPI interface.

Port	Type	Description
transmit	Input	Signal from SPI master
txbuffer	Input	Transmission buffer, holds signal from SPI master
rst	Input	User input reset
clk	Input	100MHz clock signal
sdi	Input	Serial data in from ACL
rxbuffer	Output	Receive Buffer , holds signal from sdi
done_out	Output	Signal at the end of Transmission
sdo	Output	Serial data out to ACL
sclk	Output	Serial clock out to ACL

slaveselect.v:

Involves the switching of slave select line during transmission and idle states.

Port	Type	Description
rst	Input	User input reset
transmit	Input	Signal from SPI master causing ss line to go low
done	Input	Signal from SPI interface causing ss line to go high
ss	Output	Output to the ACL

CLKDiv_5Hz:

Converts 100MHz clock signal to 5MHz clock signal

Task3.v:

This is the top module . It displays the 2D maze and ball on the VGA. The maze changes colors and has a blinking feature. The blinking feature has been applied by a **counter**. Whenever the counter is within a particular range of values then a portion of the maze corresponding to those set of values gets displayed. This makes it difficult for the ball to

complete the maze and hence keeps the player entertained. The baseball is a bitmap that has been implemented in the form of a sprite on screen. The module has three sub modules:

clk_divider.v:

The VGA operates at a clock frequency of 25MHz. The Nexys 3 board clock is at 100 MHz. A counter **clk_counter** is used to slow the clock to 25MHz for display.

vga_controller_640_60.v: This module takes the **pixel_clk** (output of clk_divider.v module) as the input and generates the **Hcounter** and **Vcounter** which are the horizontal and vertical counters respectively. It also has the **blank** as an output signal that determines whether the vedio is enabled or disabled.

maze_and_ball.v:

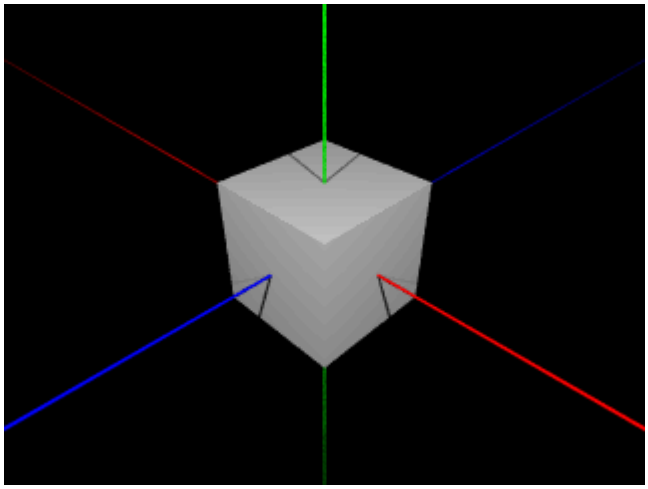
This module has 6 sub modules:

portion_# .v : The maze has been divided into 6 portions. The **Hcounter** and **Vcounter** are the inputs from the vga_controller.v module are implemented as coordinates to display each block of the maze. The **enable** signal is the output that determines which portion is to be displayed. Each portion also implements the collision detection algorithm. The **x_ball** , **y_ball** are the x and y axes coordinates of the ball that are taken as inputs from the gyroscope. These dimensions are compared with the respective coordinates of each portion to generate output signals **stop_right** , **stop_left** , **stop_up** , and **stop_down** that restrict the motion of the ball within the maze.

Design 2:Wireframe Cube

Draw_Lines_Top.v	Toplevel level module that syncs with Lines drawing algorithm. This module also controls the refresh rates. Pull rates of different angels into the OneAngle.V module to number crunch. This module pushes each new calculated line from the One angle output into a buffer. Then after a certain point calls the clear buffer trigger to enable Besenham's algorithm to render the lines on screen
OneAngle.V	OneAngle module stores and number crunches points of the cube. Each set of points from the cube are iterated through. There are 8 points and 12 lines to calculate. For each input angle into this module. The module will continually cycle through the 12 lines stored and spit out each recalculated line based on angles X,Y,Z one at a time. Each line calculation goes through 4 clock cycles and states that cascade down from one another. The calculations were broken down because the current FPGA could not fit all the calculation slices into one clk cycle. This resulted in timing errors. For more power FPGAs feel free to try to squeeze all the calculations in one cycle. This module also spends an additional state/clk cycle switching from line to line(Ex. From line 1 to line 2). This was also done because of constraints in fpga lack of resources. The same algorithm can be applied to all 12 lines/8 points at once if done on an FPGA with more resources.

Calculations are executes with a 4 fold rational axis model in mind.



The matrix where all the calculations are derived from:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Cos2.v	Cosine look up table. This design spec was put in to insure async call of multiple cos values based on angles X,Y,Z setup. No clk cycles are needed to number crunch a result
Sin2.v	Sine look up table. This design spec was put in to insure async call of multiple cos values based on angles X,Y,Z setup. No clk cycles are needed to number crunch a result

OneAngle.V module

Header	Type	Description
Clk	Input	Clock track for calculations
Reset	Input	Defaults values to 0
RW	Input	Functionality reserved for Nonconcurrent read/writes. Held low for current functionality
LineNum	Input	1-12 lines of the cube you want to calculate and display Line(AB)
thedaZ	Input	Theda Z offset from rest
thedaY	Input	Theda Y offset from rest
thedaX	Input	Theda X offset from rest
outX	Output	X of Point A
outY	Output	Y of Point A
outX2	Output	X of Point B
outY2	Output	Y of Point B
outZ	Output	Z of Point A
outZ2	Output	Z of Point B

Cos2.v Module

Header	Type	Description
degreesX	Input	Raw input of X offset
degreesY	Input	Raw input of Y offset
degreesZ	Input	Raw input of Z offset
outX	Output	Output of Cos(X)
outY	Output	Output of Cos(Y)
outZ	Output	Output of Cos(Z)

Sine2.v Module

Header	Type	Description
degreesX	Input	Raw input of X offset
degreesY	Input	Raw input of Y offset

degreesZ	Input	Raw input of Z offset
outX	Output	Output of Sin(X)
outY	Output	Output of Sin(Y)
outZ	Output	Output of Sin(Z)

Appendix:

As the project is divided into two demos, the Fig:1 gives the hierarchy of the Line drawing and the 2D Maze and Ball Game modules. Fig:2 gives the flow of the Wireframe Cube part of the project.

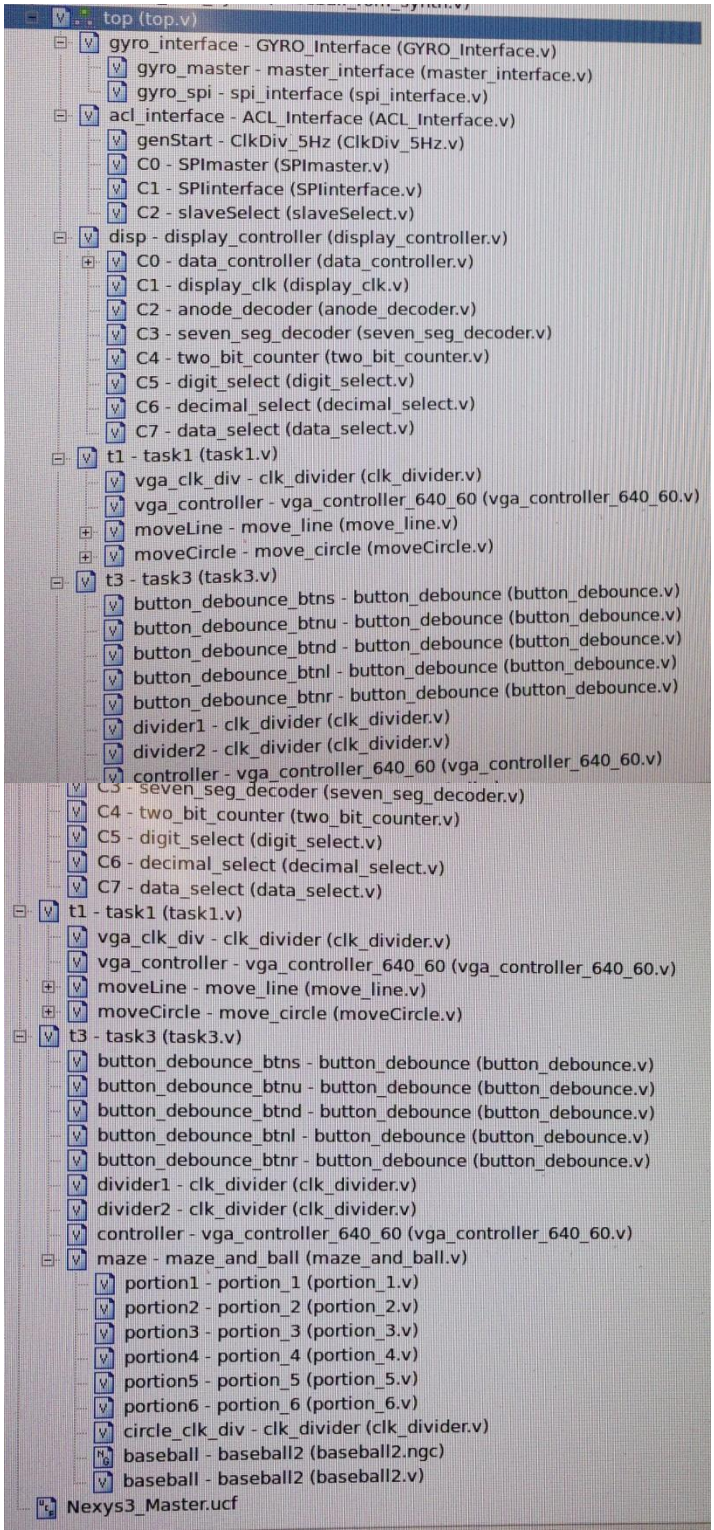


Fig 1

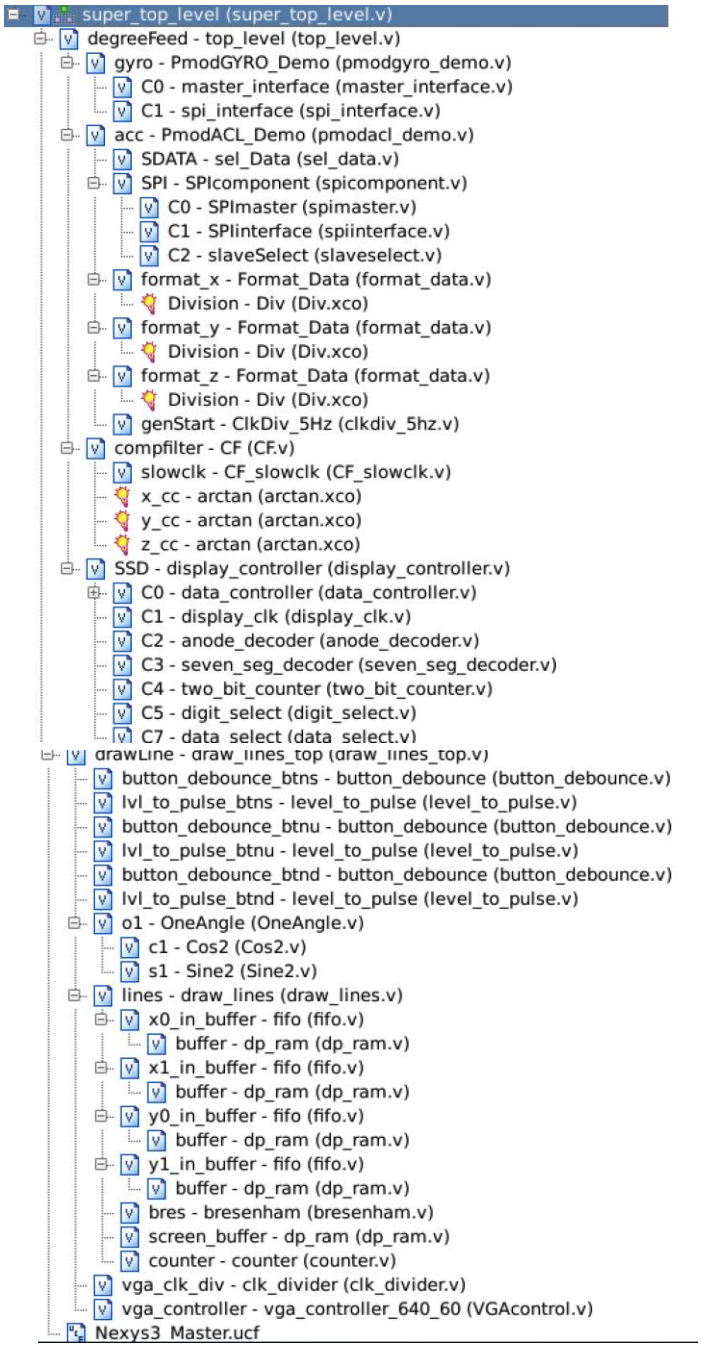


Fig 2