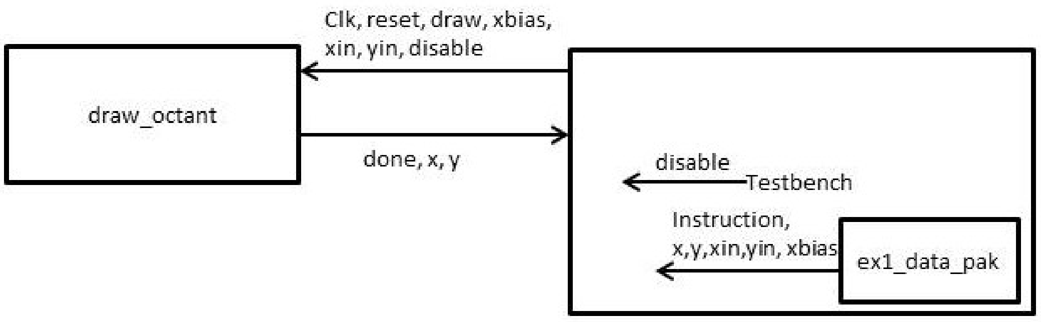
TESTING GUIDE

Understanding the testbench

As you can see from the testbench file, the entity declaration is empty. The testbench does not have input and output ports because it is not hardware. The testbench has a package which contains data to drive the test. Every set of data contains an instruction, values of x, y, xin, yin and xbias.
The testbench generates clock signals, and the clock is connected to *draw_octant*. Every clock cycle, `reset`, `draw`, `xbias`, `xin`, `yin`, `disable` are fed into draw_octant, then `done`, `x` and `y` generated from draw_octant accordingly are fed back into the testbench. Testbench will compare `x`, `y` and `done` generated by *draw_octant* with x, y and instruction in the data set.



The package is generated from a python program which replicates the correct behavior of the hardware. Tests are defined in this python program with starting point, end point and xbias. By calling the function in python with same functionality as the hardware, correct values of x and y in every clock cycle is calculated. If you are interested, you can edit the python file and add your own tests.

Debugging draw_octant

Debug with **ex1_data_pak** which contains the basic tests (which draw lines from from (2,3) to (5,3) and from (5,3) to (9,4) with xbias of 1).

Note that the testbench also generates a `disable` signal. `disable` signal is a control signal. In the testbench it is set to be high in the 3rd, 4th and 5th clock cycle. Your hardware is supposed to be 'frozen' when `disable` signal is high. Check `wave` for this first.

After you have a draw octant hardware that passes basic tests, it's time to test it with corner cases. You are given 9 new packages with different test data, replace the old **ex1_data_pak** with a new package file (from advanced tests folder) and test your hardware with it. You should pass all 9 tests:

Test	**Draw line**		**Main purpose**
	From	To	
1	(10,5)	(12,7)	`xincr` = `yincr` != 0
2	(11,6)	(11,6)	`xincr` = `yincr` = 0
3	(0,0)	(4095,0)	This test is for the case of maximum possible value of 12bits logic vector `xincr`. 4095(decimal) = b1111,1111,1111
4	(0,0)	(4095,4095)	Testing for maximum possible value of `xin` and `yin`.
5	(0,0)	(2048,2048)	2048(decimal) = b1000,0000,0000
6	(0,0)	(2048,0)	Testing for `xincr` = 2048.
7	(0,0)	(2047,30)	Testing the calculation of `err1` and `err2` with large number (number larger or equal to 2048) and also the comparison of `err1` and `err2`.
8	(0,0)	(4000,1)	Same as test 7