1. Testbench.sv
   1. Operations
      1. Create clock signal
      2. Create reset signal
      3. Instantiate a skintone\_datapath
      4. Instantiate a real and virtual io\_interface
2. Test.sv
   1. Operations
      1. Hook up the io\_interface from testbench.sv to this program
      2. Instantiate the environment which takes io\_interface as an input
         1. Env -> generate cfg, build, run, swap up
   2. Questions:
      1. How can the program just use the interface created in a different module… what about locality?
3. Environment.sv
   1. Operations
      1. Create class scoreboard\_driver\_cbs extends io\_xll\_driver\_cbs
         1. Construction sets scoreboard and cfg variables
         2. Implement convert\_ycbcr method with pixel as input
         3. Implement virtual task post\_tx
            1. Make pixel group and save it to scoreboard
      2. Create class scoreboard\_monitor\_cbs extends io\_xll\_monitor\_cbs
         1. Construction sets scoreboard
         2. Implement post\_rx
            1. Check scoreboard
      3. Class environment
         1. Constructor takes in io\_interface
            1. Generate null config
            2. Decide if random seed is needed or not
         2. Method gen\_cfg
         3. Method build
            1. Generate scoreboard, drivers, and such
            2. Not sure why the q
         4. Methon run
            1. Run the cfg and opcode driver, data mon and data drv
            2. Gen the cfg, opcode, data
            3. Set the flag to indicate done
            4. Wait for flag and timeout if exceed certain time
         5. Method wrapup
   2. Questions:
      1. Why does line 71 require “this” but line 55 does not?
      2. On line 145 and 146, the type of data being pushed onto the queue doesn’t match type of data of the queue
         1. I’m not sure why we are pushing them on the score board to begin with
4. io\_xll\_driver.sv
   1. operations
      1. class io\_xll\_driver\_cbs
         1. just have 2 virtual task pre\_tx and post\_tx as call backs
      2. class io\_xll\_driver
         1. constructor takes mailbox, event and io\_driver interface
      3. method run
         1. run deassert function
         2. loop
            1. get pixel group from mailbox
            2. go through pre-transmit callbacks
            3. send pixel group if no drop
            4. go through post-transmit callbacks
            5. remove pixel group from mailbox
            6. execute event to notify that we have consume the pg
      4. deassert method
         1. set tx\_valid and tx\_data to 0
      5. method send
         1. takes pixel group as input
         2. put data on the bus after setting tx\_valid to 1
   2. Questions:
      1. On line 86 and 96, nothing was put in the queue… what exactly are these lines running?
5. Io\_xll\_monitor.sv
   1. Operations
      1. Class io\_xll\_monitor\_cbs
         1. Only have post\_rx as call back
      2. Class io\_xll\_monitor
         1. Constructor takes io\_driver interface as input
         2. Method run
            1. Keep count of received pixel group
            2. Run receive function
            3. Go through all the post-transmit callbacks
         3. Method deassert
            1. Set rx\_ready to 0
         4. Method receive
            1. Take receive count and pg as inputs
            2. Set rx\_ready to 1
            3. Wait for rx\_valid is set
            4. Receive all the bits and create a new pixel group

This new pixel group takes the receive count and probably generate a new id also

* + - * 1. deassert