### VHDL Team Project

### Vector Display Processor

### 

Figure 1 - Hardware entities and packages

### Hardware Files

* **Config\_pack.vhd**. Package of user-configurable constants. You may alter this e.g. to change testbench clock frequency. However don’t put your own constants in here, this package will be replaced when your code is tested for assessment.
* **Project\_pack.vhd**. Use this for your own constants, types, functions, procedures. The given record **db\_2\_rcd** is used for part of the DB->RCD interface. You must use this to test your blocks in isolation - since all my behavioural code assumes this. See Configuration below.
* **Vdp.vhd**. Skeleton VDP entity & architecture . Replace the architecture with your own code but DON’T ALTER the entity. You will need to write (joint responsibility between team members) code that connects together DB and RCB blocks, each with its own adapter, as shown in Figure 1.
* **Db.vhd**. Skeleton for draw block. DONT CHANGE the entity.
* **Rcb.vhd**. Skeleton for ram control block. DONT CHANGE the entity.
* **(Db\_adapter, rcb\_adapter)**. DONT USE THESE EXCEPT POST-SYNTHESIS. These are blocks that interface post-synthesis DB and RCB (with std\_logic\_vector I/Os) to your vdp that will of course use the db\_2\_rcb bus record.

**Testbench Files**

* **Utility\_pack.vhd**. DON’T ALTER. Conversion function definitions. DONT use this in synthesisable code.
* **Vdp\_pack.vhd.** DON’T ALTER. Example definitions for a VDP testbench which you are free to use in your own code, or to copy and modify in your own package as required.
* **Vdp\_testbench.vhd**. Working testbench for the VDP. DON’T ALTER
* **Vram.vhd**. Behavioural video ram entity used in vdp\_testbench. DON’T ALTER this entity: use it as required in your testbenches.
* **Full\_commands.txt**. Text file containing sequence of draw commands that perform a simple test of the VDP commands. You should change/add to these as necessary, but keep the original for final performance tests.
* **Vtrun.do**. Modelsim "do" script file to automate startup of VHDL simulation. See Modelsim tutorial

**Simulated (behavioural) Hardware**

* **vdp\_behav.vhd**. DON’T ALTER. This entity is a working behavioural version of the VDP, for testing purposes. Your VDP should replace this but use less RAM cycles (because of ram\_control\_block) and be fully synthesisable. You don’t need to examine this file to do the project – it was written to allow the testbench components to be tested. This entity will not form part of your submitted code and is included only for information and so you can test the testbench.
* **db\_behav.vhd**. Behavioural code which simulates working DB. Use this with the structural VDP you need to write anyway and the testbench to test your own RCB independently of your partner's DB
* **rb\_behav.vhd**. Behavioural code which simulates working RCB. Use this with the structural VDP you need to write anyway and the testbench to test your own DB independently of your partner's RCB
* **db\_nocs\_behav**.vhd. As **db\_behav** but send clearscreen commands to RCB instead of implementing them itself.

**Using the Testbench**

NB - ensure vdb\_pack, vdp\_testbench, vdp\_behav, db\_behav, rcb\_behav are all compiling with 1993 VHDL. (right click on file under project windows->properties->VHDL-> 1993). If you do not do this they will not compile!

To test the testbench

* Compile: utility\_pack, config\_pack, project\_pack, vdp\_pack, vram, vdp\_behav, vdp\_testbench
* Ensure that the required testbench control command file full\_commands.txt (or other as determined by config\_pack) is put in your Modelsim project directory.
* Run simulation of vdp\_testbench. This should print out a correct result and terminate (failure ASSERT) saying that all tests have been passed.

To test a full design

* Replace vdp\_behav vdp entity with your own top-level vdp entity and also any necessary sub-entities (e.g. db, rcb, ex1-ex4) and your project\_pack.
* Make very sure you have compiled the new version of vdp, otherwise vdp\_behav will still be simulated and pass anyway.
* Run the testbench simulation again. A correct VDP should pass the test as above.
* The testbench will print out error message if problems are encoutered:
  + VRAM address/data timing errors will result in ASSERT failures from the vram component.
  + Pixel draw errors will result in a non-zero error count. You can see what is wrong from the vram contents screen printout.

**Modifying the testbench**

You will want to modify the testbench to test your code:

* Alter **full\_commands.txt** to make your own tests - the testbench will work with any file. Syntax: each line represents a host bus command, or if started with - a comment. See lines 121-144 of vdp\_testbench which read the first two characters in each line and determine what command to send, for the meaning of the first two chars. The rest of the line specifies X,Y coordinates for the command, if required. You can run your command file under a different name by altering config\_pack.
* Test your new command file by running the testbench with vdp\_behav to see what should be printed out for a given set of commands.
* Use config.vhd to adjust the clock frequency (this should make no difference to correctness) as you wish. For post-synthesis testing you may need a lower clock frequency to ensure post-synthesis code delays do not cause problems.
* Adjust the RAM parameters to make RAM slower if you have implemented RCB which can cope with this. Your RCB should (in principle) use the RAM parameters to determine number of cycles needed for RAM access. However this is not required in the basic deliverable.

**Configuration**

Your RCB must have a constant output **dbb\_rcbclear** that says whether it implements clearscreen commands or not. This will be used by test code to drive it with the correct commands from a testing DB. If it does NOT implement clearscreen ('0') the behavioural DB will output clearscreen commands as a sequence of single pixel writes. Note that although this output will be compile time constant VHDL will not know this because it is sent out through a port.