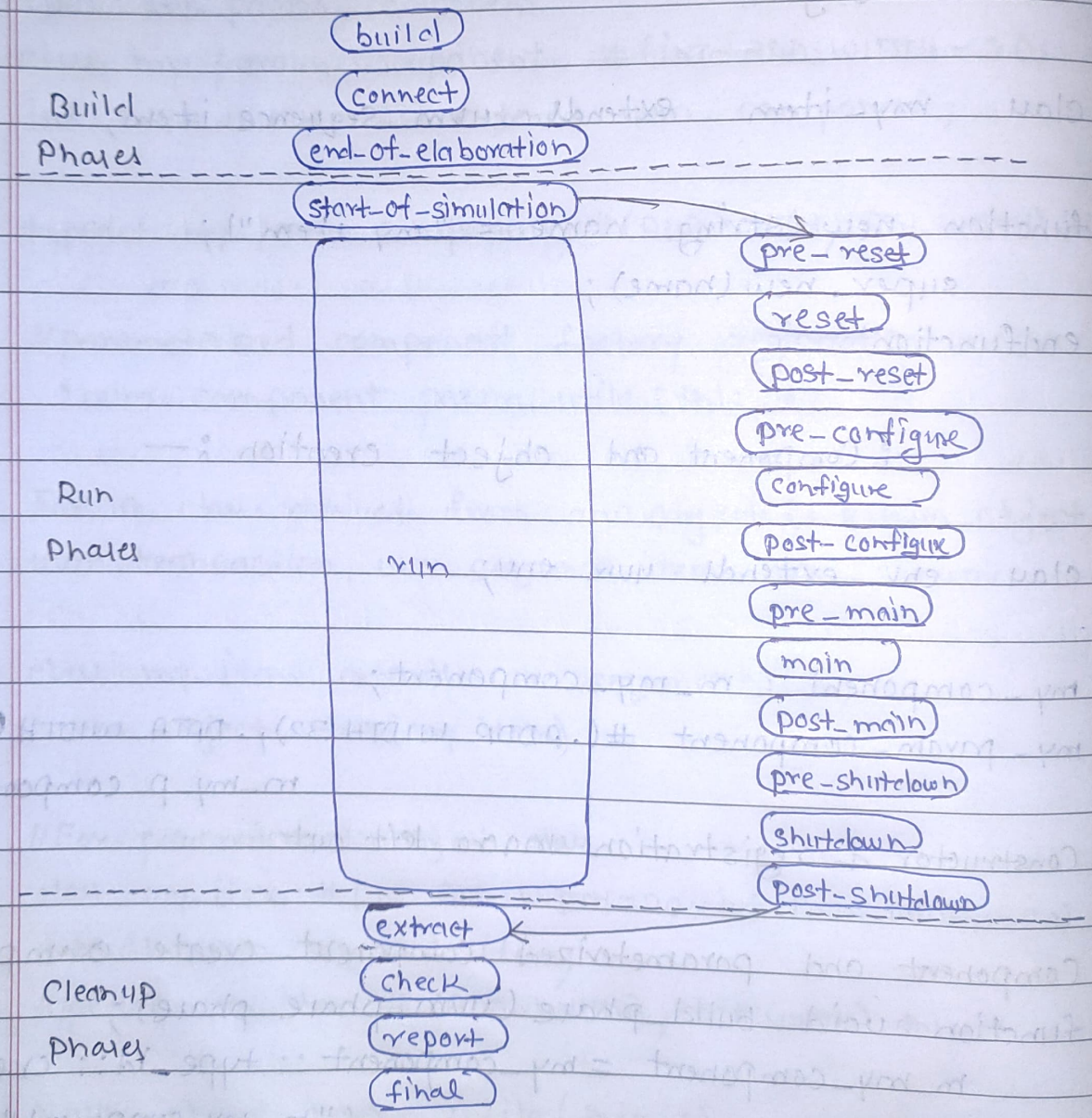
—: Standard UVM Phases :—



- only run phase is task and other all is function.

Q/ why connect is bottom up while build is top bottom.

Q/ when you call run phase ~~which~~ phase start first and how invoke test.

1. Build phase - where the testbench is configured and constructed.
2. Run_time phase - where time is consumed in running the testcase on the testbench.
3. Clean up phase - where the results of the testcase are collected and reported.

—: Starting UVM Phase Execution :—

- To start a UVM testbench, the run_test() method has to be called from the static part of the testbench. It is usually called from within an initial block in the top level module of the testbench.
- calling run_test() construct the UVM environment root component and then initiates the UVM phasing.
- The run_test() method can be passed a string argument to define the default type name of an uvm_component derived class which is used as the root node of the testbench hierarchy.
- However, the run_test() method checks for a command line plusarg called UVM_TESTNAME and uses that plusarg string to lookup a factory registered uvm_component, overriding any default type name.

```
vsim tb_top +UVM_TESTNAME = my_test
```

1. Build Phases :—

The build phases are executed at the start of the UVM testbench simulation and their overall purpose is to construct, configuration, and connect the testbench component hierarchy.

- All the build phase methods are functions and therefore execute in zero simulation time.

Build :—

Once the UVM testbench root node component is constructed, the build phase starts to execute.

- It constructs the testbench component hierarchy from the top downwards.
- The construction of each component is deferred so that each layer in the component hierarchy can be configured by the level above.
- During the build phase uvm_components are indirectly constructed using the UVM factory.

Connect :—

The connect phase is used to make TLM connections b/w components or to assign handles to testbench resources.

- It has to occur after the build method has put the testbench component hierarchy in place and works from the bottom of the hierarchy upwards.

end_of_elaboration :—

The end_of_elaboration phase is used to make any final adjustments to the structure, configuration or connectivity of the testbench before simulation starts.

- Its implementation can assume that the testbench component hierarchy and inter-connectivity is in place.
- This phase executes bottom up.

2. Run Time Phases :—

The testbench stimulus is generated and executed during the run time phases which follow the build phases.

- After the start_of_simulation phase, the UVM executes the run phases and the phases pre_reset through to post_shutdown in parallel.
- Run phase is a phase that transaction will use.
- The other phases were added to the UVM to give finer runtime phase granularity for tests, scoreboard and other similar components.
- It is expected that most testbenches will only use reset, configure, main and shutdown and not their pre and post variants.

Start_of_simulation :—

The start_of_simulation phase is a function which occurs before the time consuming part of the testbench begins.

- It is intended to be used for displaying banners; testbench topology; or configuration information.
- It is called in bottom up order.

run :—

The run phase occurs after the start of simulation phase and is used for the stimulus generation and checking activities of the testbench.

- The run phase is implemented as a task, all uvm_component run_phase() task are executed in parallel.
- Transaction such as drivers and monitor will nearly always use this phase.

Parallel Run-Time Phases :—

The following run-time phases execute in-order, in parallel with the run_phase phase.

- These phases should only be called from the test and the env to start sequences.
- Drivers, monitors and other components should not implement these phases.

(i) pre_reset :—

The pre_reset phase starts at all same time as the run phase. Its purpose is to take care of any activity that should occur before reset, such as waiting for a power-good signal to go active.

2. reset :—

The reset phase is reserved for DUT or Interface specific reset behaviour. eg; the phase would be used to generate a reset and to put an interface into its default state.

3. post_reset :—

The post_reset phase is intended for any activity required immediately following reset. This might include training or rate negotiation behaviour.

4. pre_configure :—

The pre_configure phase is intended for anything that is required to prepare for the DUT's configuration process after reset is completed, such as waiting for components (e.g. drivers) required for configuration to complete training, and/or rate negotiation.

- It may also be used to program the DUT and any memories in the testbench so that it is ready.

- It may also be used to program the DUT and any as a last chance to modify the information described by the test/environment to be uploaded to the DUT.

5. Configure :-

The configure phase is used to program the DUT and any memories in the testbench so that it is ready for the start of the test case. It can also be used to set signals to a state ready for the test case start.

6. post_configure :-

The post_configure is used to wait for the effect of configuration to propagate through the DUT, or for it to reach a state where it is ready to start the main test stimulus.

7. main :-

This is the where stimulus specified by the test case is generated and applied to DUT. It complete when either all stimulus is exhausted or a timeout occurs. Most data throughout will be handled by sequences started in this phase.

8. Shutdown :-

The shutdown phase is used to ensure that the effect of the stimulus generated during the main phase have propagated through the DUT and that any resultant data has drained away.

It might also be used to execute time consuming sequences that read status registers.

3. Clean Up Phase :—

extract :—

The extract phase is used to retrieve and process information from scoreboards and functional coverage monitors.

This may include the calculation of statistical information used by the report phase. this phase is usually used by analysis components.

check :—

The check phase is used to check that the DUT behaved correctly and to identify any errors that may have occured during the execution of the testbench. This phase is usually used by analysis components.

report :—

The report phase is used to display the results of the simulation or to write the results to file. This phase is usually used by analysis components.

final :—

The final phase is used to complete any other outstanding actions that the testbench has not already completed.