Session 5 Exercise

fares.sultan9@gmail.com Switch account

Not shared

* Indicates required question

Name *

Choose

Build phase and connect phase consume time while execution * 1 point

- ○ True
- ○ False

Build , connect, and run phase, they all run parallel of each other * 1 point

- ○ True
- ○ False

What are the order of the execution of the following phases * 1 point

- ○ build_phase, run_phase, connect_phase
- ○ connect_phase, build_phase, run_phase
- ○ build_phase, connect_phase, run_phase
- ○ run_phase, connect_phase, build_phase

We use raise_objection to decrement the counter and end testing * 1 point

- ○ True
- ○ False

While reporting, we select verbosity of sim to be high (UVM_DEBUG) to avoid noisy transcript * 1 point

- ○ True
- ○ False

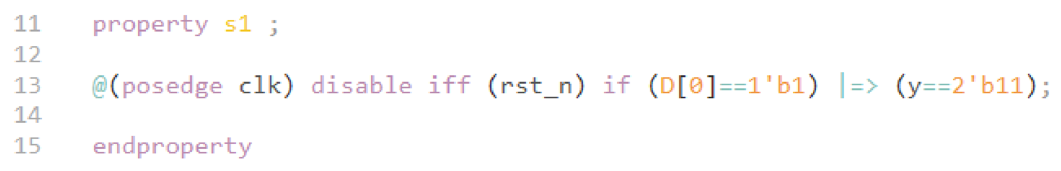
If I set the verbosity of uvm_info to be UVM_NONE, then this info message will be printed no matter the simulation verbosity level * 1 point

- ○ True
- ○ False

If u need to compile this files (ALU_top , ALU_test , ALU_env , ALU_interface , ALU, ALU_config, ALU_driver) , so we can sort them as following * 1 point

- ALU_interface → ALU → ALU_config → ALU_Driver → ALU_env → ALU_test → ALU_top
- ALU_top→ ALU → ALU_config → ALU_Driver → ALU_env → ALU_test → ALU_interface
- ALU_top → ALU_test → ALU_env → ALU_Driver → ALU_ config → ALU → ALU_interface
- no needing for sorting

If I want to disable the asserted reset while checking the following property (active low reset) 1 point



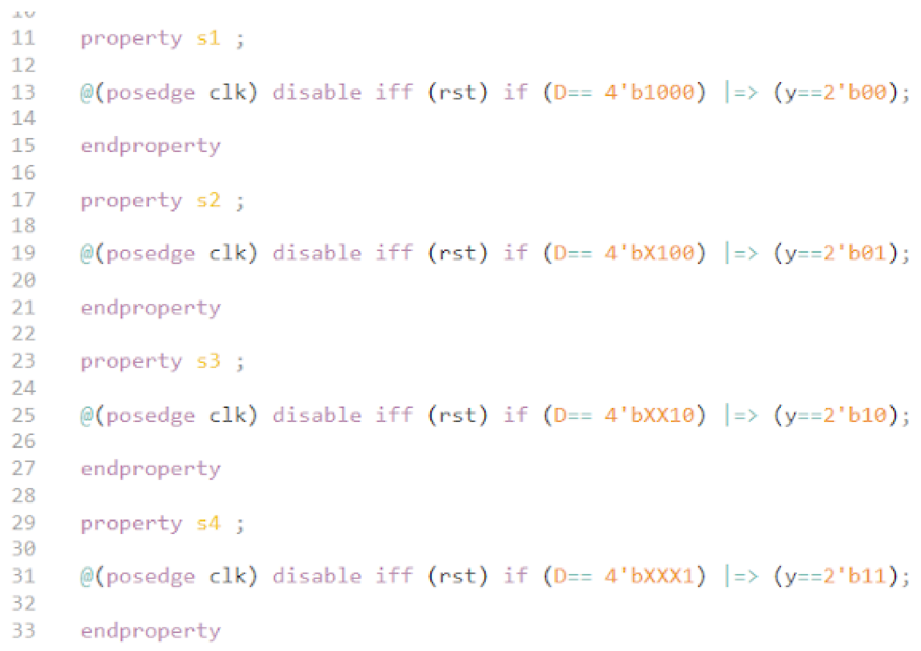
```
property s1 ;

@(posedge clk) disable iff (rst_n) if (D[0]==1'b1) |=> (y==2'b11);

endproperty
```

- True
- False

In the following property if I need to check priorty encoder output I can do this (active high reset) * 1 point



```
property s1 ;

@(posedge clk) disable iff (rst) if (D== 4'b1000) |=> (y==2'b00);

endproperty

property s2 ;

@(posedge clk) disable iff (rst) if (D== 4'bX100) |=> (y==2'b01);

endproperty

property s3 ;

@(posedge clk) disable iff (rst) if (D== 4'bXX10) |=> (y==2'b10);

endproperty

property s4 ;

@(posedge clk) disable iff (rst) if (D== 4'bXXX1) |=> (y==2'b11);

endproperty
```

○ True

○ False

If D input is 4′b1111, is it correct to check the encoder with these assertions ? * 1 point

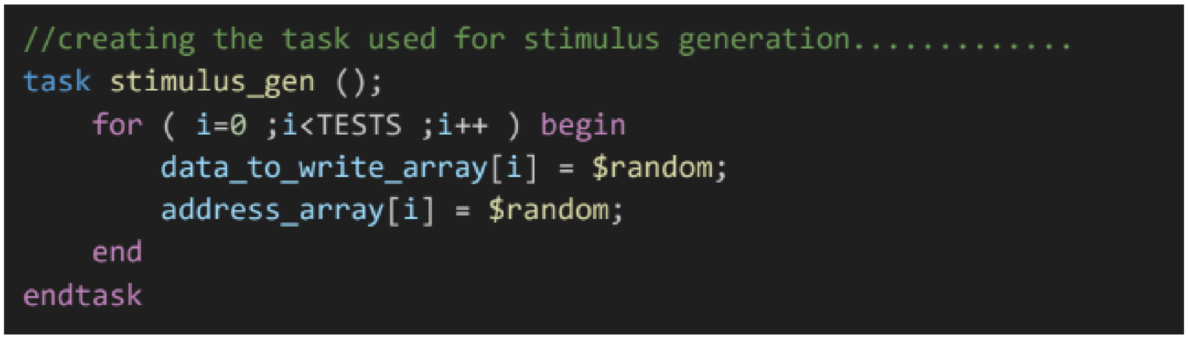
```
property s1 ;
@(posedge clk) disable iff (rst) if (D[3]== 1'b1) |=> (y==2'b00);
endproperty
property s2 ;
@(posedge clk) disable iff (rst) if (D[2]== 1'b1) |=> (y==2'b01);
endproperty
property s3 ;
@(posedge clk) disable iff (rst) if (D[1]== 1'b1) |=> (y==2'b10);
endproperty
property s4 ;
@(posedge clk) disable iff (rst) if (D[0]== 1'b1) |=> (y==2'b11);
endproperty
```

- ○ yes, 3 pass and 1 fail
- ○ no, 3 pass and 1 fail
- ○ no,1 pass and 3 fail
- ○ yes, 4 pass
- ○ no, 4 fail

If i need to calculate the even parity then I should use * 1 point

- ○ ^ (Xor)
- ○ ~^ (Xnor)
- ○ Nand

If i need to get 100 different unique location (addresses) then I can use the following: * 1 point



```
//creating the task used for stimulus generation.............
task stimulus_gen ();
    for ( i=0 ;i<TESTS ;i++ ) begin
        data_to_write_array[i] = $random;
        address_array[i] = $random;
    end
endtask
```

- ○ True, this will get 100 unique addresses at all times
- ○ False

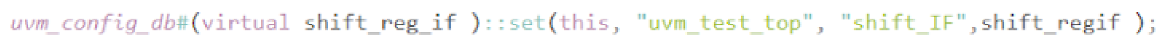
Assertions can be in .... * 1 point

- ○ module binding in testbench
- ○ module binding in top
- ○ the design with macros
- ○ all above

When I select verbosity of the uvm_info to be UVM_DEBUG, uvm_info with a lower verbosity level are filtered away * 1 point

- ○ True
- ○ False

while setting the virtual interface in uvm_config_db from top we use the following command * 1 point



```
uvm_config_db#(virtual shift_reg_if )::set(this, "uvm_test_top", "shift_IF",shift_regif );
```

- ○ True
- ○ False

Which of the following class we must import the package of uvm env * 1 point

- ○ UVM_Driver
- ○ UVM_Config_object
- ○ UVM_Top
- ○ UVM_Test

What is the role of the UVM Factory? * 1 point

- ○ Generate clock signals
- ○ Control the creation of UVM objects and components
- ○ Handle testbench connections
- ○ Generate random stimulus

Which UVM phase is responsible for running the stimulus in a UVM testbench? * 1 point

- ○ build_phase
- ○ run_phase
- ○ connect_phase
- ○ final_phase

What is the main function of the UVM interface? * 1 point

- ○ Connect two drivers
- ○ Specify signal interactions between the DUT and testbench
- ○ Generate stimuli
- ○ Check the response output of the design

In UVM, Which phase runs after the build_phase? * 1 point

- ○ reset_phase
- ○ run_phase
- ○ start_of_simulation_phase
- ○ connect_phase

What does a UVM uvm_config_db object allow you to do? * 1 point

- ○ Store configuration settings across the UVM testbench
- ○ Manage connections between drivers and monitors
- ○ Generate stimuli for the test
- ○ Control testbench termination

Which UVM macro is used to define a component class in UVM? * 1 point

- ○ uvm_component_utils
- ○ uvm_object_utils
- ○ uvm_sequence_utils
- ○ uvm_factory_create

Which UVM function is used to set a string field in the uvm_config_db? * 1 point

- ○ uvm_config_set("path", "key", value)
- ○ uvm_config_db#(string)::set(this, "path", "key", value)
- ○ uvm_set_db_string("path", "key", value)
- ○ uvm_config_db_string::set(this, "key", value)

Who is responsible for writing the assertions? * 1 point

- ○ Only verification engineer writes it because he is responsible for verifying the design
- ○ Only design engineer writes it because he is reponsible for the design itself
- ○ Both verification and design engineers

What is the default behavior of assertions if you did not specify a severity level? 1 point

- ○ They will always terminate the simulation.
- ○ They will log an error and continue the simulation.
- ○ They will ignore all assertions.
- ○ They will cause the simulation to pause.

What is the correct syntax for defining a sequence that follows another sequence for a sequential output? * 1 point

- ○ sequence; A |=> B; endsequence
- ○ sequence; A |-> B; endsequence

What does the operator |-> indicate in a sequence? * 1 point

- ○ It indicates a condition must be true at the next time step
- ○ It creates a loop within a sequence
- ○ It specifies that if the left hand side is true, the right hand side will be true on the same clock tick

What does the operator |=> indicate in a sequence? * 1 point

- ◯ It specifies that if the left hand side is true, the right hand side will be true on the same clock tick
- ◯ It creates a loop within a sequence
- ◯ It specifies that if the left hand side is true, the right hand side will be true on the next clock tick

What does the function $past(a,7) do in SVA 1 point

- ◯ The value of the same clock cycle
- ◯ The value shifted right one cycle
- ◯ The value shifted right 7 cycles
- ◯ The value shifted left 7 cycles

What is the purpose of the "disable" keyword in SVA? 1 point

- ◯ To stop the simulation
- ◯ To a halt a sequence or property from executing
- ◯ To define the scope of an assertion

A practical rule: more code lines more verification efficiency * 1 point

- ◯ True
- ◯ False

You can add a parameterized value to a class but not a parameterized type because data types are to be defined before the compilation * 1 point

○ True

○ False

A struct is the similar to a class but without methods * 1 point

○ True

○ False

class CAPITAL_X extends small_x
this code line means that every function and variable in small_x can be used in CAPITAL_X if it wasn't overwritten * 1 point

○ True

○ False

Super keyword is used to call: * 1 point

○ A function in the parent after it was edited in the child

○ A function in the parent

○ None of the above

Static variables and functions aren't global to all objects of the same class signature * 1 point

- ◯ True
- ◯ False

`define macros are used for: * 1 point

- ◯ defining parameters
- ◯ defining states in FSM
- ◯ replacing a certain line code with multi-line code for the code to be reusable

UVM stands for * 1 point

- ◯ Ultra verification method
- ◯ Universal verification module
- ◯ Universal Verification Methodology

UVM is: * 1 point

- ◯ HDL language
- ◯ Replacement for system verilog assertions
- ◯ It provides a standard reusable methodology for testbenches

Build phase is used for: * 1 point

- build the constraint blocks
- build the class handles
- build the UVM components

The run_phase in the driver is where we: * 1 point

- Connect the driver with the monitor
- Sample data for Coverage groups
- Drive the design interface with the stimulus

Difference between new and type_id::create is * 1 point

- Syntax difference
- The latter creates and register the object in the UVM factory
- All of the above

UVM_NONE in UVM report messages are always printed * 1 point

- True
- False

uvm_config_db can be treated as a parameterized class where we set in shared variables and get them if the scope was defined correctly * 1 point

- True
- False

Which feature allows a SystemVerilog class to be reused with different types? * 1 point

- Interface
- Static class
- Parameterized class

What is the correct way to declare a static method? * 1 point

- static function void display();
- function static void display();
- void display() static;
- function display();

What mechanism is used to end a UVM simulation phase? * 1 point

- ○ raise_finish()
- ○ uvm_stop()
- ○ raise_objection() and drop_objection()
- ○ uvm_end_phase()

Which method is used to raise an objection in UVM? * 1 point

- ○ uvm_test::raise()
- ○ raise_objection(this);
- ○ this.raise();
- ○ drop_objection();

What is the default verbosity level of UVM reporting? * 1 point

- ○ UVM_NONE
- ○ UVM_LOW
- ○ UVM_MEDIUM
- ○ UVM_INFO

Which macro is used to print a UVM error message? * 1 point

- ○ uvm_error
- ○ uvm_report_error
- ○ uvm_log_error
- ○ uvm_fatal

What is a virtual interface used for in UVM? * 1 point

- ○ To instantiate interfaces inside classes
- ○ To define interface timing
- ○ To connect classes to DUT interfaces
- ○ To build sequences dynamically

How is a virtual interface typically passed to a UVM component? * 1 point

- ○ Through a file I/O operation
- ○ Using uvm_config_db::set/get()
- ○ Through constructor arguments
- ○ By using a static variable

What does uvm_config_db::set() do? * 1 point

- ◯ Registers the UVM component
- ◯ Stores configuration information for retrieval
- ◯ Declares a virtual interface
- ◯ Adds a component to the simulation hierarchy

What does uvm_config_db::get() do? * 1 point

- ◯ Retrieves stored configuration data
- ◯ Accesses global variables
- ◯ Retrieves DUT signals
- ◯ Overrides default test names

What is a UVM configuration object? * 1 point

- ◯ A generic object used to transfer configuration data
- ◯ A component that generates random data
- ◯ A class that stores sequence items
- ◯ A built-in UVM macro

Which class is responsible for driving stimulus to the DUT? * 1 point

- ○ uvm_monitor
- ○ uvm_test
- ○ uvm_driver
- ○ uvm_env

What is the entry point of a testbench in UVM? * 1 point

- ○ uvm_top
- ○ uvm_test
- ○ run_phase
- ○ main()

Which method is used to start a test in UVM? * 1 point

- ○ uvm_run()
- ○ run_test()
- ○ start_test()
- ○ begin_test()

Which method is called first during the simulation lifecycle in UVM? * 1 point

- ○ run_phase
- ○ build_phase
- ○ report_phase
- ○ connect_phase

Which of the following is true about UVM test classes? * 1 point

- ○ They must extend uvm_test
- ○ They cannot access virtual interfaces
- ○ They are not registered with the factory
- ○ They do not use phases

How are user-defined configuration objects usually passed? * 1 point

- ○ By command-line arguments
- ○ Using global variables
- ○ Through uvm_config_db
- ○ Using macros

What does uvm_info do? * 1 point

- ◯ Terminates simulation
- ◯ Displays simulation warnings
- ◯ Prints a message at a specified verbosity level
- ◯ Sets the simulation time

Which UVM phase ends only after all objections are dropped? * 1 point

- ◯ build_phase
- ◯ run_phase
- ◯ connect_phase
- ◯ start_of_simulation_phase

What is the primary role of the Top module in UVM? * 1 point

- ◯ Define test scenarios and sequences
- ◯ Instantiate the DUT and connect it to the testbench via interfaces
- ◯ Implement UVM phases like build_phase and run_phase
- ◯ Generate coverage reports

Which of the following is typically *not* part of the Top module? * 1 point

- ◯ Physical signals of the DUT
- ◯ Virtual interface handles
- ◯ UVM environment class
- ◯ Clock generation logic

How does the Test class interact with the UVM Environment? * 1 point

- ◯ The Test class directly drives the DUT using virtual interfaces
- ◯ The Test class instantiates and configures the Environment
- ◯ The Test class replaces the Environment in runtime
- ◯ The Test class is a sub-component of the Environment

What does a UVM Environment typically contain? * 1 point

- ◯ DUT instances and physical interfaces
- ◯ Agents, scoreboards, and coverage collectors
- ◯ Configuration objects for the Test class
- ◯ Clock and reset generation logic

How is the UVM Environment configured? *

1 point

- ○ Using uvm_config_db from the Test class
- ○ By modifying global variables in the Top module
- ○ Through direct method calls to the Driver
- ○ Automatically during the build_phase

What is the role of the UVM Driver? *

1 point

- ○ Collect functional coverage from the DUT
- ○ Translate transactions into pin-level signals via a virtual interface
- ○ Generate random sequences for testing
- ○ Check the correctness of DUT outputs

Why are virtual interfaces used in UVM? *

1 point

- ○ To replace physical interfaces in the DUT
- ○ To provide an abstract handle for accessing DUT signals in class-based testbenches
- ○ To generate clock and reset signals
- ○ To store configuration parameters

How is a virtual interface passed to UVM components like the Driver? * 1 point

- ○ Using uvm_resource_db
- ○ Through constructor arguments
- ○ Via uvm_config_db from the Top module
- ○ By declaring it as a global variable

Which phase is common to both Test and Environment classes? * 1 point

- ○ run_phase
- ○ build_phase
- ○ connect_phase
- ○ All the above

Submit Page 1 of 1 Clear form