**Module: UVM-2**

**Project**

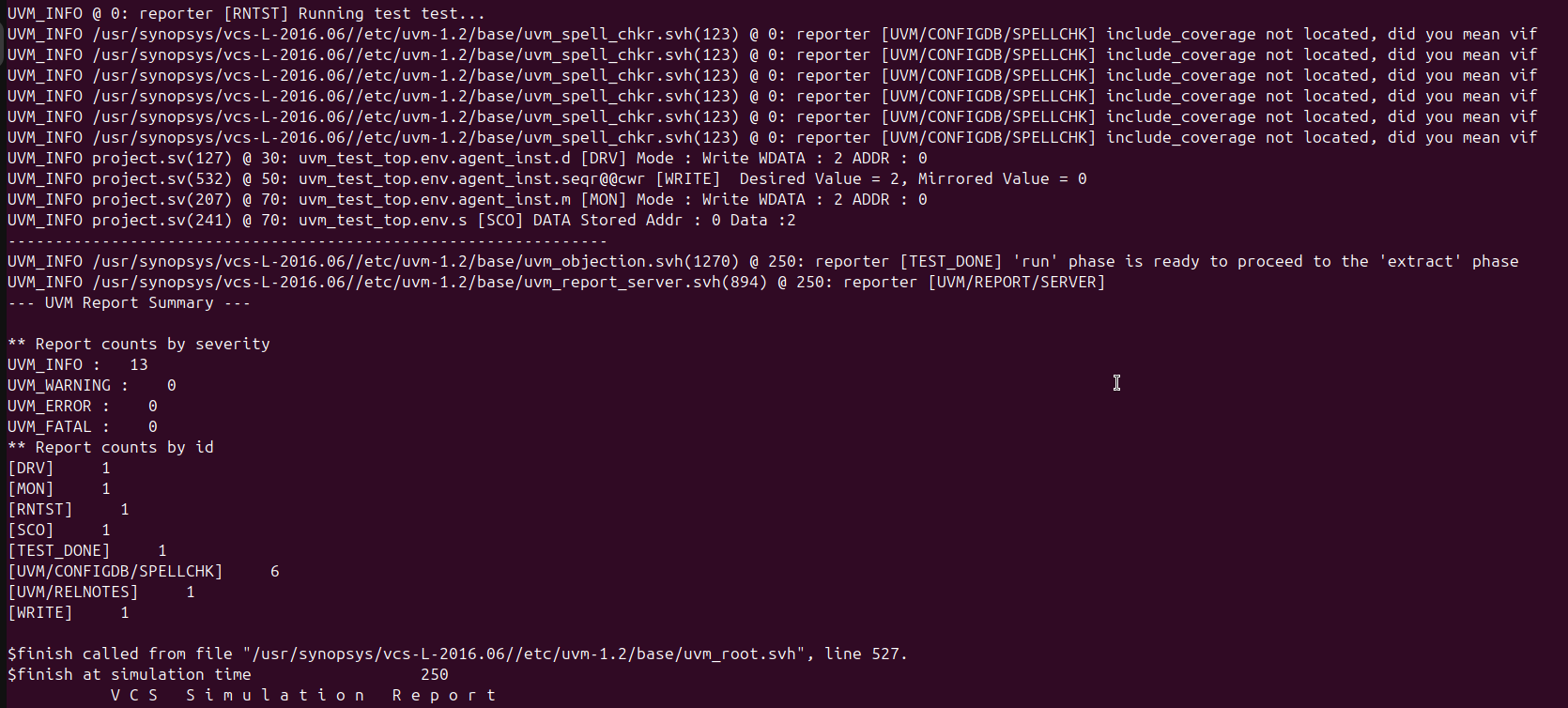
**Project -** [**Source Code**](https://github.com/Noman-10xe/UVM-2/tree/main/Project/code)( V1.1 to V1.2 )

* **Control Sequence:**

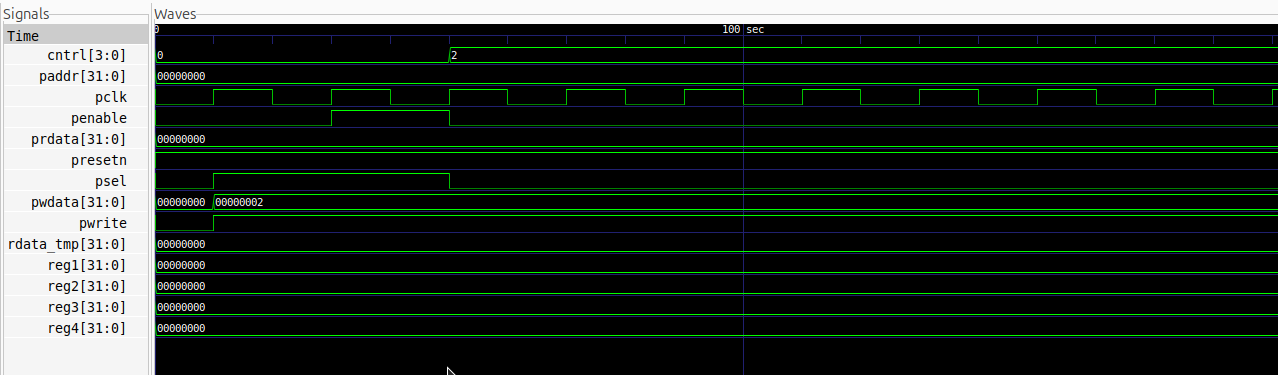
| **ctrl\_wr** |
| --- |

1. **Checking Desired Value with get( ):**

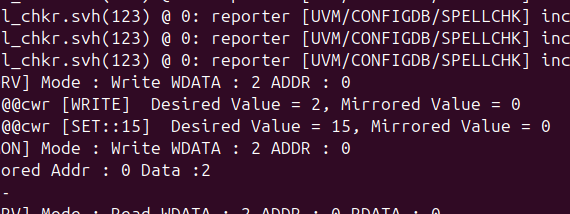
* **Console Output:**

****

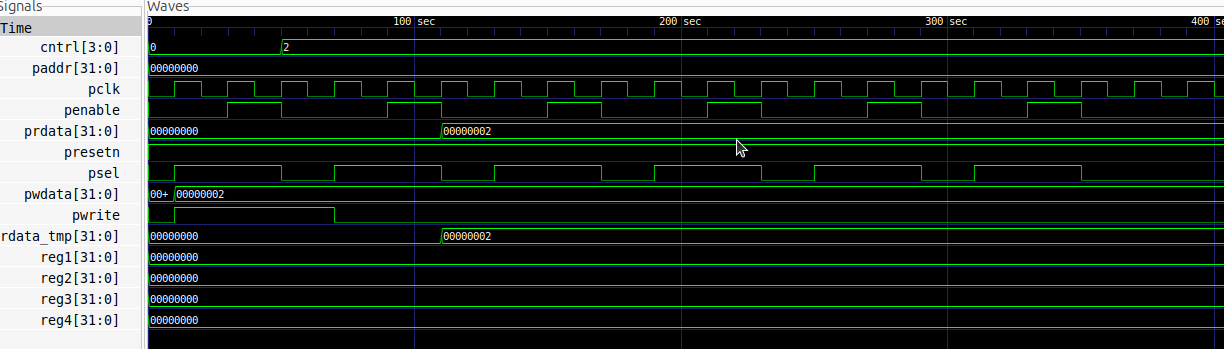
* **Waveform:**

****

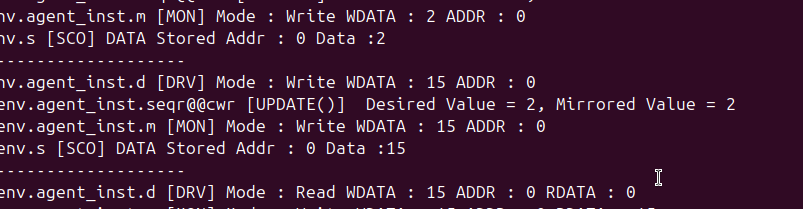
1. **Setting Desired Value with set( ):**
   * **Console Output:**

****

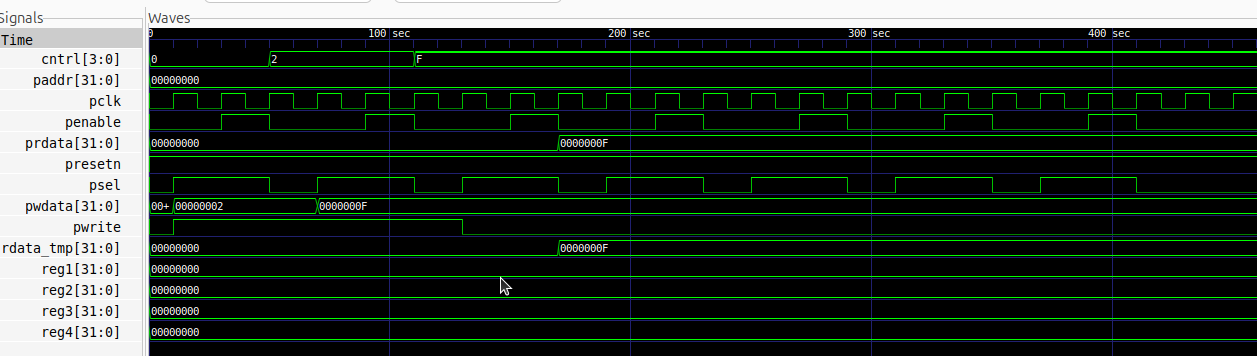
* + **Waveform:**

****

1. **Writing Desired Value to DUT with update( ):**
   * **Console Output:**

****

* + **Waveform:**

****

1. **Purpose of each method:**
   * **get( ):** Retrieves the desired value of the register from the UVM register model. This value reflects what the testbench expects, not necessarily the actual value in the DUT.
   * **set( ):** Sets the desired value field of the register in the UVM model. It does not directly affect the DUT but prepares the model for updates or checks.
   * **update ( ):** Writes the desired value from the UVM model to the DUT register, ensuring the DUT reflects the intended state defined by the testbench.
2. **Code:**

| class ctrl\_wr extends uvm\_sequence;  `uvm\_object\_utils(ctrl\_wr)  reg\_block regmodel;  function new (string name = "ctrl\_wr");  super.new(name);  endfunction  task body;  uvm\_status\_e status;  bit [3:0] wdata;  bit [3:0] dv, mv; // Desired Value & Mirrored Values  wdata = $urandom();    regmodel.cntrl\_inst.write(status, wdata);  // Check 'dv' and 'mv' Values  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("WRITE", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  // Set the Desired value  regmodel.cntrl\_inst.set(15); // Set the Value to be 15  // Check again  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("SET::15", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  // Update the Value to DUT  regmodel.cntrl\_inst.update(status); // Write the Value to DUT  // Check again  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("UPDATE()", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  endtas**k**  **endclass** |
| --- |

* **Predict( ) method:**
  1. **Difference between predict( ) & mirror( ):**

**Predict Method:**

* **Purpose:** Updates the mirrored value of a register in the UVM register model based on observed activity on the DUT.
* **Trigger:** Typically called by monitors or predictors when a register operation (read/write) is detected on the bus.
* **Behavior:** It does not initiate any physical access; it simply reflects the updated state of the register.
* **Use Case:** A predictor observes a write operation to a register and uses **predict** to update the mirrored value accordingly.

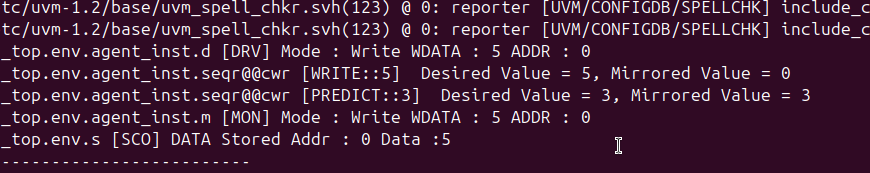
### **Mirror Method:**

* **Purpose:** Compares or synchronizes the mirrored value in the UVM register model with the actual value in the DUT.
* **Trigger:** Can initiate a physical access (read) to retrieve the DUT's current value and update the mirrored value.
* **Behavior:** Provides the option to verify whether the mirrored value and the DUT value are consistent.
* **Use Case:** Verifying that the current state of the register in the DUT matches the expected mirrored value.

### **Summary:**

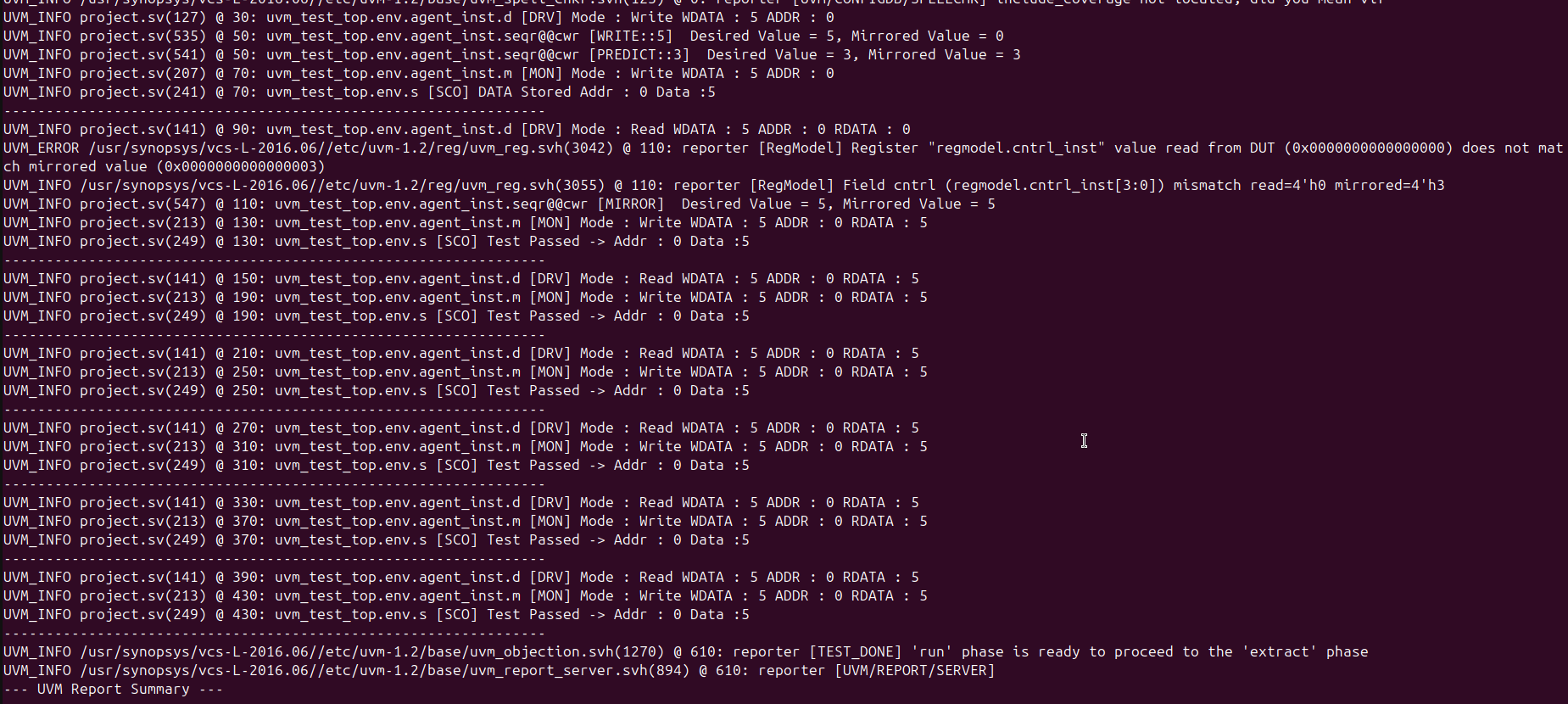
* **predict:** Passive, updates the UVM model without DUT access.
* **mirror:** Active, ensures consistency between the UVM model and the DUT.
  1. **Write(4’h5) and Predict(4’h3):**

**Output:**

****

* 1. **Mirror( UVM\_CHECK ):**

**Output:**

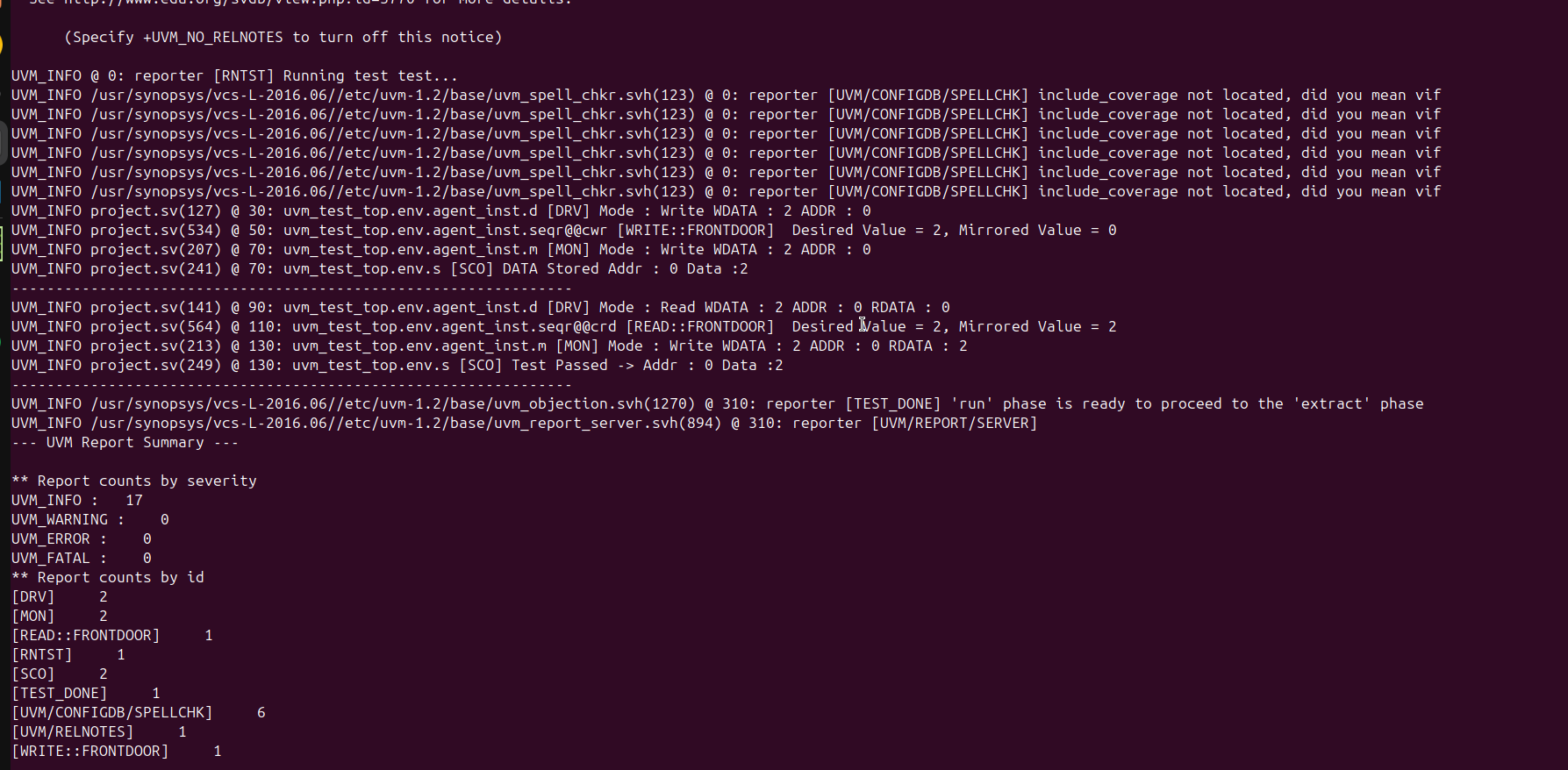
****

The error occurs because the predicted value in the UVM model was set to **4'h3**, without reflecting the actual state of the register inside the DUT. When the **mirror()** method is called, it reads the current value from the DUT's register and compares it to the mirrored value in the UVM model using the **UVM\_CHECK** option. If a mismatch is detected between the DUT value and the mirrored value, **UVM\_CHECK** reports an error. After reporting the error, the mirrored value in the UVM model is updated to match the DUT's register value.

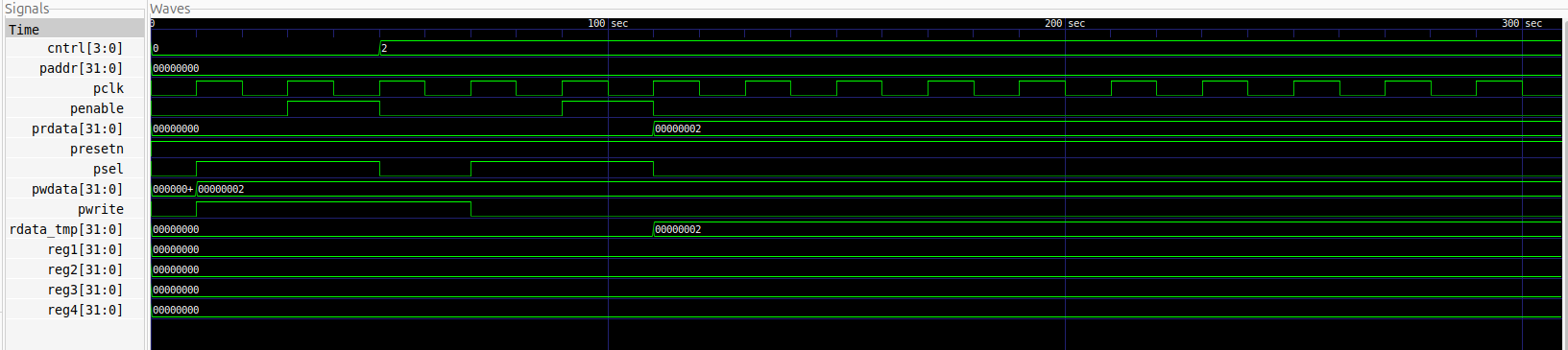
* 1. **Observation:**

First, a constant value of **4'h5** is written to the DUT using the **write()** method. Next, the predicted value in the UVM model is set to **4'h3**, which updates the mirrored value field in the model. When the **mirror()** method is invoked with the **UVM\_CHECK** option, it detects a mismatch between the mirrored value in the model and the actual value in the DUT. This mismatch triggers an error report. After reporting the error, the **mirror()** method updates the mirrored value in the UVM model to match the actual value in the DUT.

* **FRONT\_DOOR Access:**
  1. **FRONTDOOR Wr/Rd using ctrl\_wr and ctrl\_rd Sequences:**
     + **Output:**

****

* + - **Waveform:**

****

* + - **Code:**

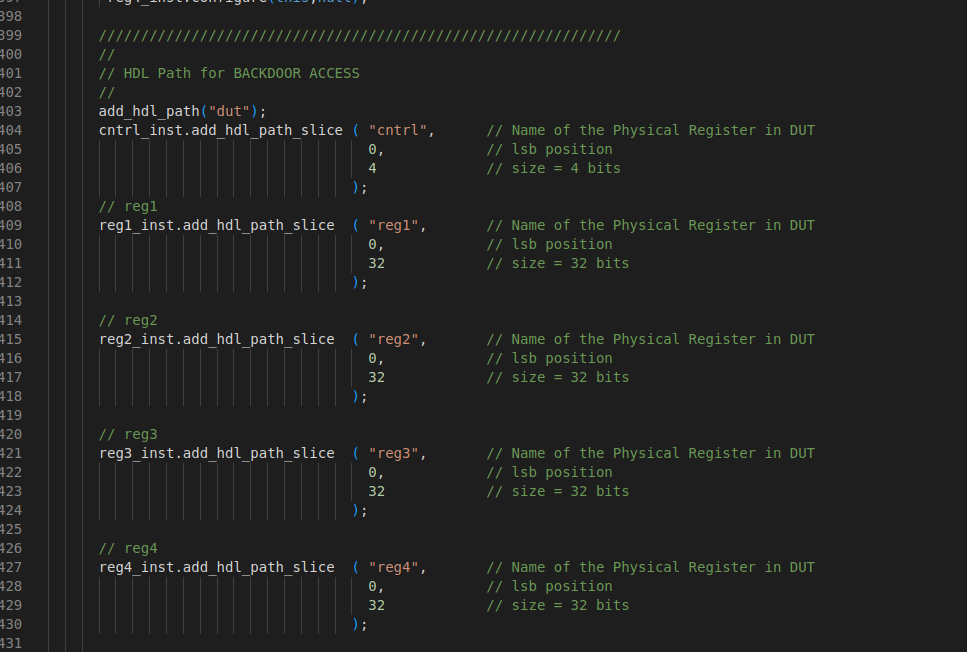
| ///////////////////// WRITE Sequence ///////////////////////////  class ctrl\_wr extends uvm\_sequence;  `uvm\_object\_utils(ctrl\_wr)  reg\_block regmodel;  function new (string name = "ctrl\_wr");  super.new(name);  endfunction  task body;  uvm\_status\_e status;  bit [3:0] wdata;  bit [3:0] dv, mv; // Desired Value & Mirrored Values  wdata = $urandom();  regmodel.cntrl\_inst.write(status, wdata, UVM\_FRONTDOOR);  //Check 'dv' and 'mv' Values  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("WRITE::FRONTDOOR", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  endtask  endclass  ///////////////////// READ Sequence ///////////////////////////  class ctrl\_rd extends uvm\_sequence;  `uvm\_object\_utils(ctrl\_rd)  reg\_block regmodel;  function new (string name = "ctrl\_rd");  super.new(name);  endfunction  task body;  uvm\_status\_e status;  bit [3:0] rdata;  bit [3:0] dv, mv; // Desired Value & Mirrored Values    //////working with control  regmodel.cntrl\_inst.read(status, rdata, UVM\_FRONTDOOR);  //Check 'dv' and 'mv' Values  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("READ::FRONTDOOR", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  endtask  endclass |
| --- |

* **BACKDOOR Access:**
  1. **Backdoor Access in UVM:**

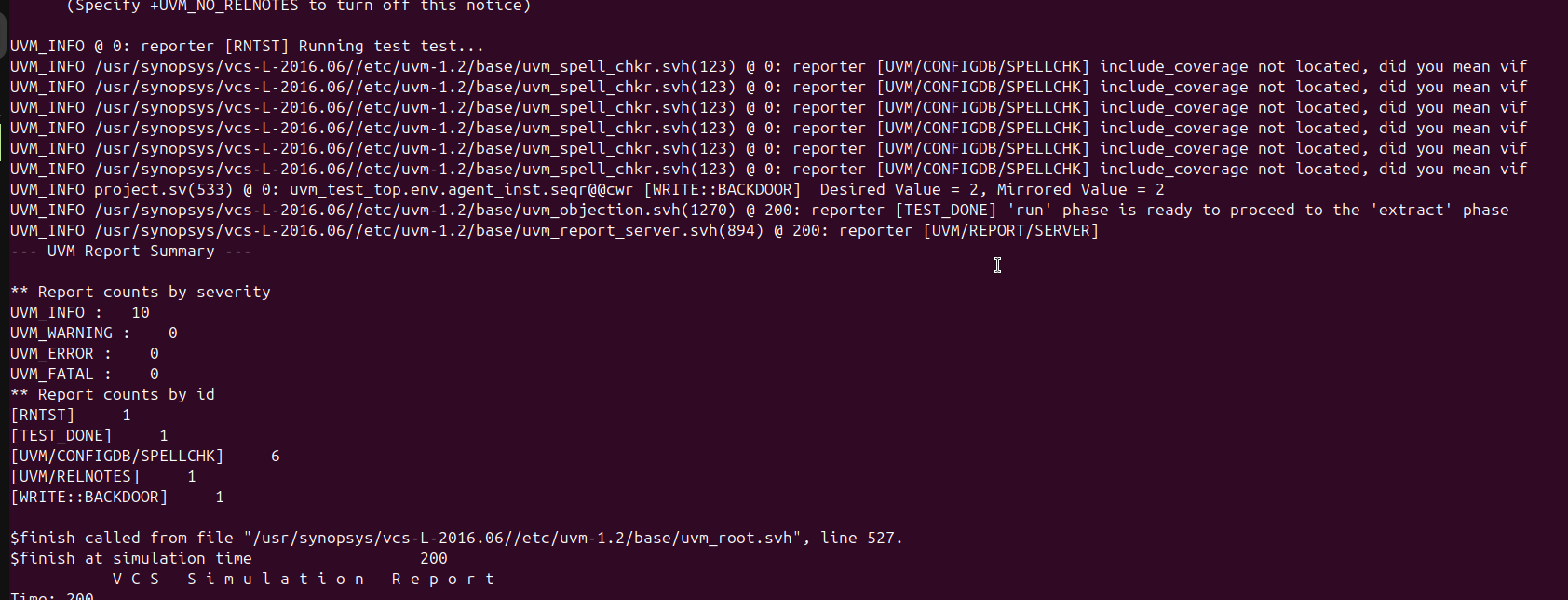
Backdoor access allows direct interaction with DUT signals using the simulator's database, bypassing the bus protocol. Write operations deposit values directly onto register signals, and read operations sample their current values. While this method avoids triggering control logic connected to the bus, it enables efficient signal manipulation without interfering with design transitions.

### **Benefits**

* Backdoor access takes zero simulation time, as it avoids bus transactions.
* Useful for quickly verifying or initializing specific register states.
* Speeds up verification by complementing the standard frontdoor protocol checks.
  1. **HDL Paths:**
     + **Code:**

****

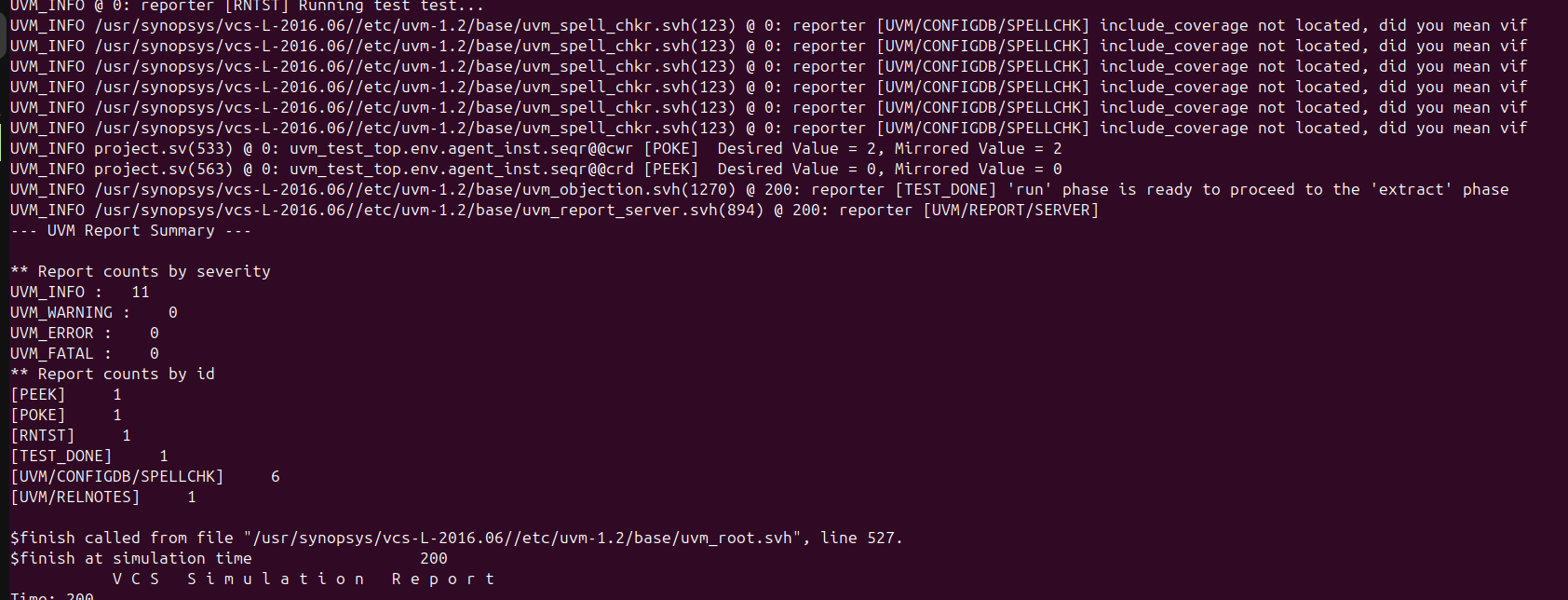
* 1. **BACKDOOR WRITE:**
     + **Output:**

****

* + - **Code:**

| class ctrl\_wr extends uvm\_sequence;  `uvm\_object\_utils(ctrl\_wr)  reg\_block regmodel;  function new (string name = "ctrl\_wr");  super.new(name);  endfunction  task body;  uvm\_status\_e status;  bit [3:0] wdata;  bit [3:0] dv, mv; // Desired Value & Mirrored Values  wdata = $urandom();  regmodel.cntrl\_inst.write(status, wdata, UVM\_BACKDOOR);  //Check 'dv' and 'mv' Values  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("WRITE::BACKDOOR", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  endtask  endclass |
| --- |

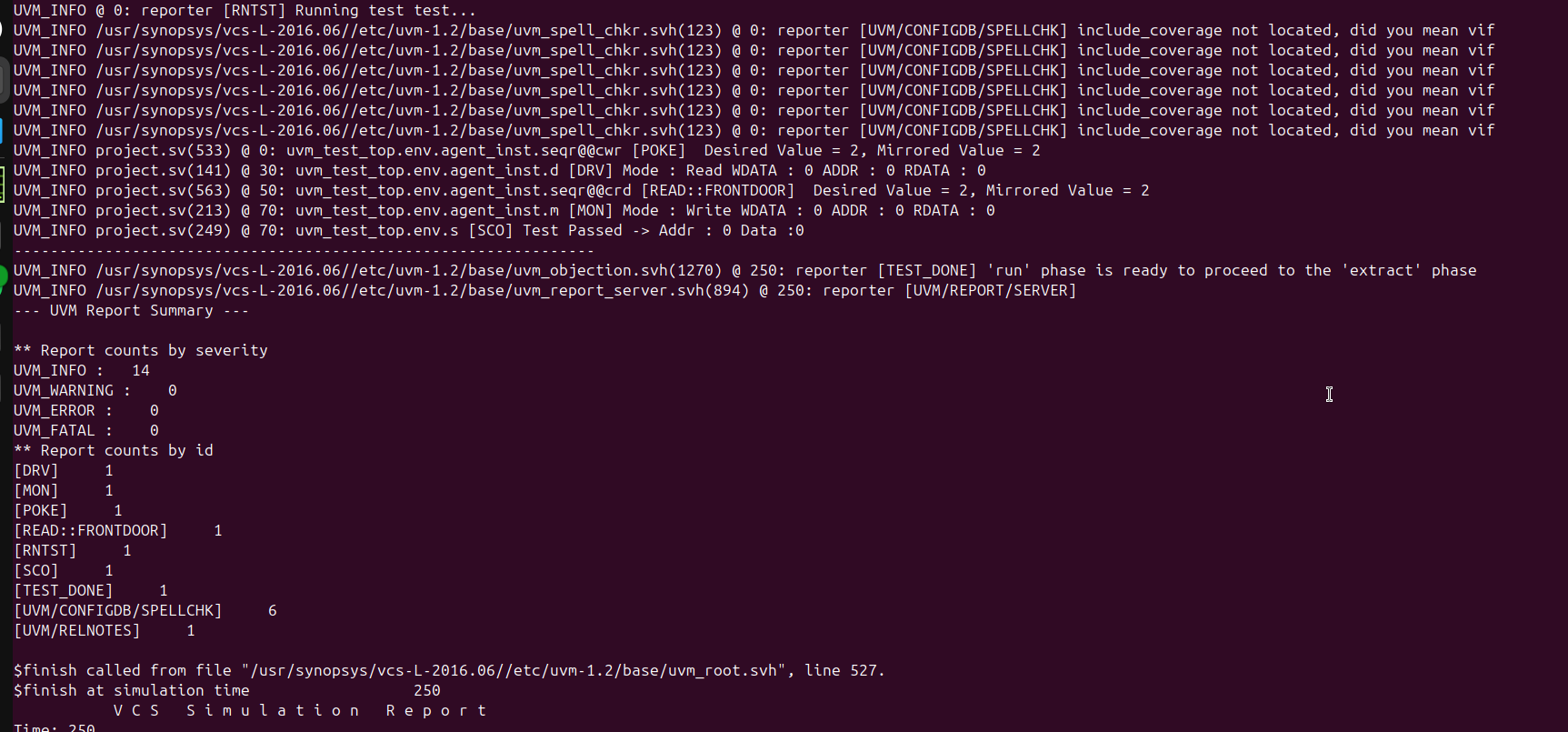
* 1. **Peek/Poke Methods:**
     + **Output:**

****

* + - **Code:**

| class ctrl\_wr extends uvm\_sequence;  `uvm\_object\_utils(ctrl\_wr)  reg\_block regmodel;  function new (string name = "ctrl\_wr");  super.new(name);  endfunction  task body;  uvm\_status\_e status;  bit [3:0] wdata;  bit [3:0] dv, mv; // Desired Value & Mirrored Values  wdata = $urandom();  regmodel.cntrl\_inst.poke(status, wdata);  //Check 'dv' and 'mv' Values  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("POKE", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  endtask  endclass  /////////////////////////////////////////////////////////////////////////  //////////////////read data from control reg      class ctrl\_rd extends uvm\_sequence;  `uvm\_object\_utils(ctrl\_rd)  reg\_block regmodel;  function new (string name = "ctrl\_rd");  super.new(name);  endfunction  task body;  uvm\_status\_e status;  bit [3:0] rdata;  bit [3:0] dv, mv; // Desired Value & Mirrored Values    //////working with control  regmodel.cntrl\_inst.peek(status, rdata);  //Check 'dv' and 'mv' Values  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("PEEK", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  endtask  endclass |
| --- |

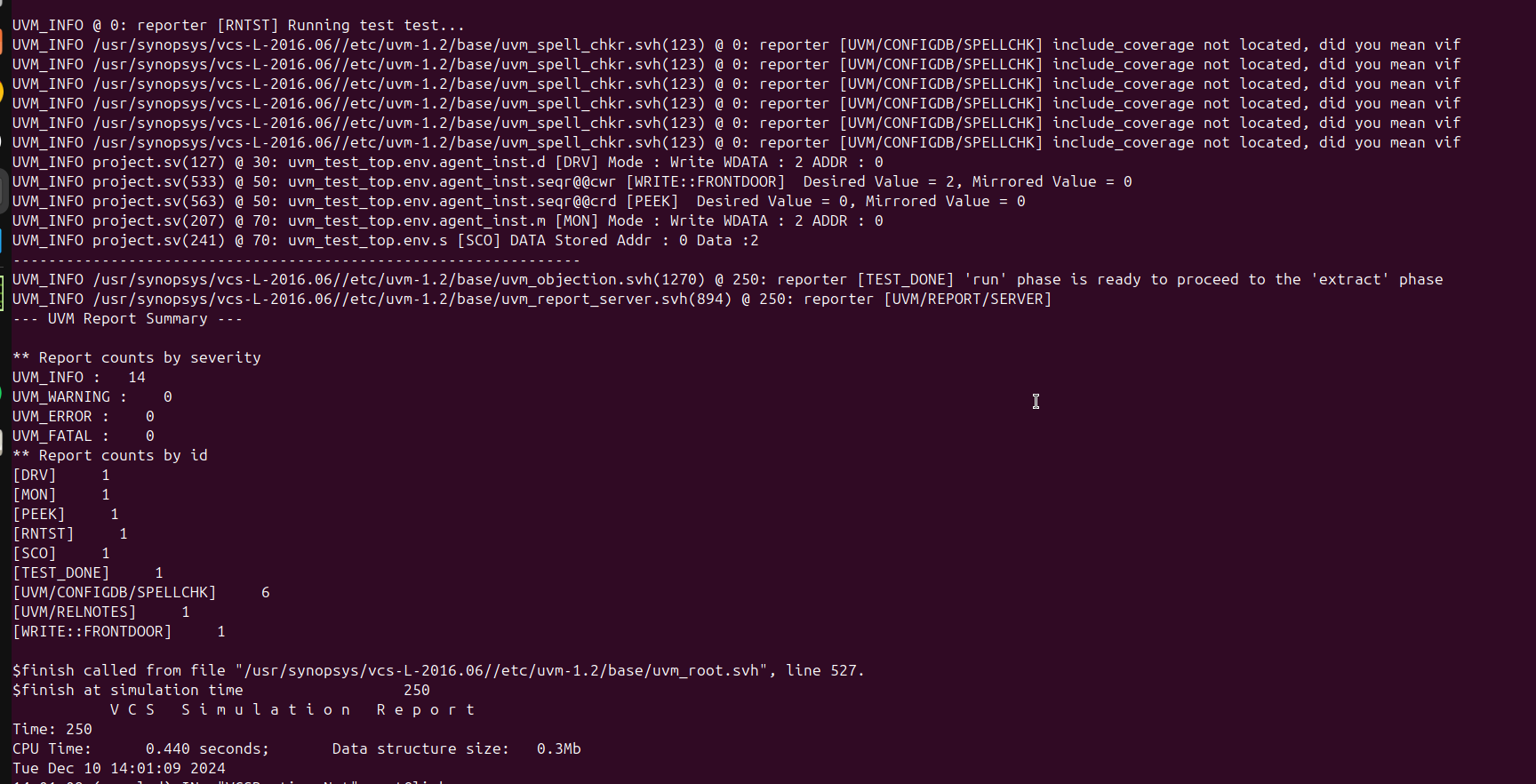
* 1. **Poke & FRONTDOOR READ:**
     + **Output:**

****

* + - **Code:**

| class ctrl\_wr extends uvm\_sequence;  `uvm\_object\_utils(ctrl\_wr)  reg\_block regmodel;  function new (string name = "ctrl\_wr");  super.new(name);  endfunction  task body;  uvm\_status\_e status;  bit [3:0] wdata;  bit [3:0] dv, mv; // Desired Value & Mirrored Values  wdata = $urandom();  regmodel.cntrl\_inst.poke(status, wdata);  //Check 'dv' and 'mv' Values  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("POKE", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  endtask  endclass  /////////////////////////////////////////////////////////////////////////  //////////////////read data from control reg      class ctrl\_rd extends uvm\_sequence;  `uvm\_object\_utils(ctrl\_rd)  reg\_block regmodel;  function new (string name = "ctrl\_rd");  super.new(name);  endfunction  task body;  uvm\_status\_e status;  bit [3:0] rdata;  bit [3:0] dv, mv; // Desired Value & Mirrored Values    //////working with control  regmodel.cntrl\_inst.read(status, rdata, UVM\_FRONTDOOR);  //Check 'dv' and 'mv' Values  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("READ::FRONTDOOR", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  endtask  endclass |
| --- |

* 1. **FRONTDOOR Write & PEEK:**
     + **Output:**

****

* + - **Code:**

| class ctrl\_wr extends uvm\_sequence;  `uvm\_object\_utils(ctrl\_wr)  reg\_block regmodel;  function new (string name = "ctrl\_wr");  super.new(name);  endfunction  task body;  uvm\_status\_e status;  bit [3:0] wdata;  bit [3:0] dv, mv; // Desired Value & Mirrored Values  wdata = $urandom();  regmodel.cntrl\_inst.write(status, wdata, UVM\_FRONTDOOR);  //Check 'dv' and 'mv' Values  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("WRITE::FRONTDOOR", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  endtask  endclass  /////////////////////////////////////////////////////////////////////////  //////////////////read data from control reg      class ctrl\_rd extends uvm\_sequence;  `uvm\_object\_utils(ctrl\_rd)  reg\_block regmodel;  function new (string name = "ctrl\_rd");  super.new(name);  endfunction  task body;  uvm\_status\_e status;  bit [3:0] rdata;  bit [3:0] dv, mv; // Desired Value & Mirrored Values    //////working with control  regmodel.cntrl\_inst.peek(status, rdata);  //Check 'dv' and 'mv' Values  dv = regmodel.cntrl\_inst.get(); // Get Desired Value  mv = regmodel.cntrl\_inst.get\_mirrored\_value();  `uvm\_info("PEEK", $sformatf(" Desired Value = %0d, Mirrored Value = %0d ", dv, mv), UVM\_NONE)  endtask  endclass |
| --- |