Aluna: Jaqueline Ferreira de Brito

Data: 30 de Janeiro de 2025

RELATÓRIO VERIFICAÇÃO FORMAL E FUNCIONAL
SOURCE, SINK, TRANS

Descrição: Este relatório apresenta uma variação da primeira simulação UVM explorada em aula. A simulação continha apenas os componentes básicos: fonte, transação e sorvedouro.

Instruções:

- Realizar uma variação da primeira simulação apresentada em aula (fonte, transação, sorvedouro).
- A variação pode ser uma das apresentadas em aula ou uma nova, a critério do aluno.

Entregar um relatório breve (2 a 5 páginas) em formato PDF.

1. Introdução

O ambiente de teste UVM é organizado em pacotes para reutilização de código e mantém a conexão entre componentes por meio de **ports e exports**. A **UVM Factory** permite a criação dinâmica e configuração flexível dos componentes. O fluxo de dados segue as etapas de geração, envio, processamento, recepção e verificação de transações.

2. Testbench UVM

- **Transações (Jaque_a_tr)**
  - Classe derivada de uvm_sequence_item.
  - Variável inteira j para armazenar dados.
  - Registrada na UVM Factory com uvm_object_utils.
  - Utilizada para gerar dados e enviá-los para um driver.
- **Source (Fonte)**
  - Componente uvm_component que gera estímulos.
  - Utiliza put_port para enviar transações.
  - Gera **20 transações** com atraso de **2ns** cada.
  - Usa raise_objection para manter a simulação ativa e drop_objection para finalizá-la.
- **Sink (Sorvedouro)**
  - Componente uvm_component que recebe transações.
  - Utiliza put_imp para processar transações recebidas.
  - Exibe mensagem no log ao receber uma transação.

2. Transações: Definidas no arquivo trans.svh:

Código alterado com minhas especificações:

```
class Jaque_a_tr extends uvm_sequence_item;
       int j;
       //Troca de nome a_tr por Jaque_a_tr
       `uvm_object_utils_begin(Jaque_a_tr)
       `uvm_field_int(j, UVM_ALL_ON | UVM_DEC)
       `uvm_object_utils_end

       function new(string name = "Jaque_a_tr");
       super.new(name);
       endfunction
endclass
```

Explicação:

A classe Jaque_a_tr representa uma transação em um teste UVM, herdando de uvm_sequence_item. Ela contém uma variável inteira j para armazenar dados e é registrada

na factory UVM com uvm_object_utils. O campo j é configurado para ser impresso, comparado e copiado automaticamente. O construtor permite nomear a transação e chama o construtor da classe base para garantir a inicialização correta.

3. Source (Fonte): Componente que gera as transações:

```
class source extends uvm_component;
   `uvm_component_utils (source)

   // creating a put_port which will accept a "Jaque_a_tr" type of data
   uvm_blocking_put_port #(Jaque_a_tr) out;

   function new (string name = "Jaqueline_Brito", uvm_component parent);
       super.new (name, parent);
       out = new ("Saída", this);
   endfunction

   task run_phase (uvm_phase phase);
       Jaque_a_tr  tr;

       phase.raise_objection(this); // avoid simulation to finish immediately
       // Trocando as 10 transações do professor por 20 transações que vão ser
enviadas via put_port
       repeat (20) begin
       #2;                          // Espera 2ns
       tr = new();                  // create transaction instance
       `bvm_begin_tr(tr)            // start transaction recording
       `uvm_info ("SOURCE", "Sending transaction by Jaqueline_Brito", UVM_LOW)
       out.put (tr);                // send transaction to put_port
       end

       #50;                         // allow for some downstream processing time
       phase.drop_objection(this);   // allow simulation to finish
   endtask
endclass
```

A classe source é um componente UVM que gera e envia transações do tipo Jaque_a_tr através de uma put_port. Ela herda de uvm_component, permitindo sua integração no ambiente de verificação. O registro com uvm_component_utils possibilita a criação dinâmica da classe.

No construtor, a porta out é inicializada para comunicação. Durante a run_phase, a classe gera 20 transações, enviando cada uma com um atraso de 2ns. A objeção é levantada para evitar o fim prematuro da simulação e removida após um tempo de processamento adicional de 50ns, garantindo que todas as transações sejam processadas antes do término.

4. Sink (Sorvedouro): Componente que recebe as transações:

```
class sink extends uvm_component;
       `uvm_component_utils(sink)

       uvm_blocking_put_imp #(Jaque_a_tr, sink) in;

       function new(string name, uvm_component parent);
       super.new(name, parent);
       in = new("Entrada", this);
       endfunction

       task put(Jaque_a_tr tr);
       `uvm_info("SINK", "Receiving transaction by Jaqueline_Brito", UVM_LOW)
       endtask
endclass
```

A classe sink é um componente UVM responsável por receber transações do tipo Jaque_a_tr. Ela herda de uvm_component, permitindo sua integração no ambiente de verificação. O registro com uvm_component_utils possibilita sua criação dinâmica via factory.

A porta de entrada in é do tipo uvm_blocking_put_imp, permitindo a recepção de transações. No construtor, a porta é inicializada com o nome "Entrada". O método put processa as transações recebidas e exibe uma mensagem no log de simulação (uvm_info), indicando que a transação foi recebida com o nível de log UVM_LOW.

5. Top Module: Módulo principal que conecta todos os componentes:

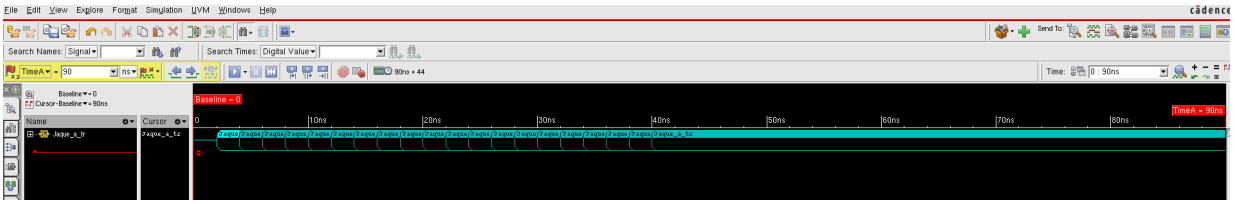
```
`include "uvm_macros.svh"
import uvm_pkg::*;
`include "trans.svh"
`include "source.svh"
`include "sink.svh"

module top;
    source source_h;
    sink sink_h;

    initial begin
        source_h = new("source_h", null);
        sink_h = new("sink_h", null);
        source_h.out.connect(sink_h.in);
        run_test();
    end
endmodule
```

O módulo top é o principal ponto de entrada da simulação UVM, responsável por instanciar e conectar os componentes de geração (source) e recepção (sink) de transações.

- **Inclusão de Arquivos:** Importa macros (uvm_macros.svh), o pacote UVM (uvm_pkg::*), e os arquivos que definem a transação (trans.svh), o gerador (source.svh) e o receptor (sink.svh).
- **Instanciação de Componentes:** Cria source_h (gerador de estímulos) e sink_h (receptor de transações).
- **Conexão de Portas:** Liga out de source_h (put_port) à in de sink_h (get_port), permitindo a transferência de transações.
- **Execução do Teste:** O run_test(); inicia a simulação, acionando a interação entre os componentes.

Esse módulo organiza a comunicação entre os elementos e garante que a simulação ocorra conforme esperado.

6. Simulação



```
UVM_INFO source.svh(22) @ 14: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 14: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 16: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 16: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 18: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 18: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 20: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 20: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 22: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 22: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 24: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 24: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 26: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 26: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 28: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 28: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 30: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 30: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 32: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 32: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 34: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 34: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 36: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 36: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 38: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 38: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO source.svh(22) @ 40: source_h [SOURCE] Sending transaction by Jaqueline_Brito
UVM_INFO sink.svh(12) @ 40: sink_h [SINK] Receiving transaction by Jaqueline_Brito
UVM_INFO
/usr/local/cds/XCELIUM2409/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_objection.svh(1268)
@ 90: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase


--- UVM Report catcher Summary ---


Number of demoted UVM_FATAL reports  :    0
Number of demoted UVM_ERROR reports  :    0
Number of demoted UVM_WARNING reports:    0
Number of caught UVM_FATAL reports   :    0
Number of caught UVM_ERROR reports   :    0
Number of caught UVM_WARNING reports :    0

--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   43
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[SINK]    20
[SOURCE]    20
[TEST_DONE]     1
[UVM_CMDLINE_PROC]     1
Simulation complete via $finish(1) at time 90 NS + 44
/usr/local/cds/XCELIUM2409/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_root.svh:457     $finish;
```

Conclusão

A simulação demonstrou a comunicação correta entre os componentes **source e sink**, garantindo a transferência de transações no ambiente de teste UVM. O código pode ser expandido para incluir verificações e análises mais complexas.