Aluna: Jaqueline Ferreira de Brito
Data: 17 de Fevereiro de 2025

RELATÓRIO VERIFICAÇÃO FORMAL E FUNCIONAL
Gravação de transações, Randomização e Usando uvm_test

Este relatório apresenta uma evolução da primeira simulação UVM explorada em aula. A simulação inicial era composta apenas pelos componentes fundamentais: fonte (source), transação (transaction) e sorvedouro (sink). Nesta versão, esses componentes foram reorganizados e integrados à classe test, tornando o ambiente de teste mais modular e flexível. Além disso, foram implementadas variações para incorporar conceitos como ambientes de teste (env), pacotes (package), filas FIFO (fifo) e modelos de referência (reference model), aprimorando a estrutura e a eficiência da simulação.

Instruções:
Realizar variações das simulações abordadas nas seguintes lições: env, package, fifo e reference model.

Entrega:
O relatório deve ter entre 2 e 5 páginas e ser entregue em formato PDF.

1. Introdução

Usando a classe uvm_env dentro do teste. Um teste pode conter vários ambientes, um ambiente para cada DUT. Aqui só temos um único DUT, então só temos um único ambiente. Usar a fábrica também para transações, no lugar do new(). Retiramos o código da classe test feito anteriormente na atividade 2.

2. Alterações

2.1 Env

A classe env contém a mesma funcionalidade que a classe test anteriormente, não usa mais new mas a fábrica de UVM, usando a função create que é própria desse elemento. A fábrica funciona através de chamadas *classe*::type_id::create. Os argumentos da chamada são os mesmos do new. Isso permite maior flexibilidade na criação do testbench, pois pode-se testar vários DUT juntos, cada um com seu ambiente de teste.

```
class env extends uvm_env;
   `uvm_component_utils(env)

   source source_h;
   sink sink_h;

   function new(string name, uvm_component parent);
       super.new(name, parent);
   endfunction

   function void build_phase(uvm_phase phase);
       super.build_phase(phase);
       source_h = source::type_id::create("source_h_JaqueB.", this); // a diferença esta
aqui
       sink_h = sink::type_id::create("sink_h_JaqueB.", this);
   endfunction

   function void connect_phase(uvm_phase phase);
       source_h.out.connect (sink_h.in);
   endfunction

endclass
```

2.2 Alteração no source

```
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 6 transações que vão ser enviadas
via put_port
       repeat (6) begin
```

```
        #12;                          // Espera 12ns
        tr = Jaque_a_tr::type_id::create("tr",this); // Aqui esta a alteração de NEW para
CREATE, referente a fabrica UVM
        assert(tr.randomize ());
        `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

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

2.3 Remoção da source e sink da classe test, e apenas env é instanciado:

```
class test extends uvm_test;
    `uvm_component_utils(test)

    env env_h;

    function new(string name, uvm_component parent);
        super.new(name, parent);
    endfunction

    function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        env_h = env::type_id::create("env_JaquelineB.", this);
    endfunction
endclass
```

2.4 e por fim incluímos no módulo top também precisa incluir o env

```
`include "uvm_macros.svh"
import uvm_pkg::*;
`include "bvm_macros.svh" // macros created by Brazil-IP / UFCG

`include "trans.svh"
`include "source.svh"
`include "sink.svh"
`include "env.svh"    // adicionando o env ao top agora
`include "test.svh"

module top;

    initial begin
        run_test("test");
    end
endmodule
```

Para evitar conflitos de nomes e deixar mais flexível criamos um pacote que vai encapsular as classes onde duas classes do mesmo nome que estão em pacotes diferentes podem ser usados na mesma simulação. A biblioteca UVM também é um pacote. Ele precisa ser importado dentro do nosso pacote para podermos usar as classes UVM.

2.5 Podemos apagar agora os `include do arquivo top.sv e adicionar eles em um test_pack.sv

```
`include "uvm_macros.svh"
`include "bvm_macros.svh" // macros created by Brazil-IP / UFCG

package test_pkg;

  import uvm_pkg::*; //packege criado

  `include "trans.svh"
  `include "source.svh"
  `include "sink.svh"
  `include "env.svh" // classe env instanciada
  `include "test.svh"
endpackage
```

2.6 Agora o módulo top só fica os comandos do import:

```
module top;
   import uvm_pkg::*;
   import test_pkg::*;

   initial begin
       run_test("test");
   end
endmodule
```

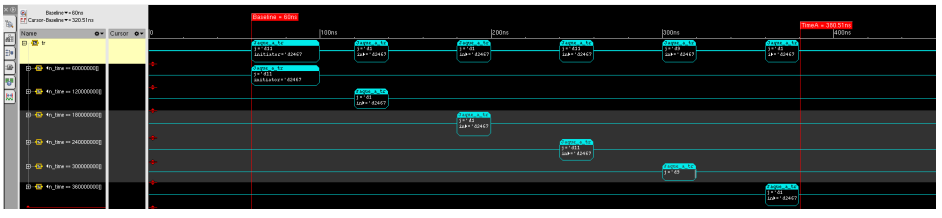
2.7 Substituição do source por drain

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

   uvm_get_port #(Jaque_a_tr) in; // porta de entrada que vai requisitar e receber novas
transações;

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

   task  run_phase (uvm_phase phase);
       Jaque_a_tr tr;

       #60; //alteração de 30 para 60

       // laço infinito que só termina quando toda a simulação for encerrada
       forever begin
       #20; // alteração de 10 para 20
       in.get(tr); // get para solicitar uma nova transação, caso não entregue um nova
transação, ficará esperando, consumindo tempo de simulação, até que a transação esteja
disponível
       `uvm_info("DRAIN, by Jaqueline Brito ", "Transaction received", UVM_MEDIUM)
       #20; // alteração de 10 para 20
       `bvm_end_tr(tr);
       end

   endtask

endclass
```

O sorvedouro foi aprimorado para requisitar uma transação no momento em que deseja, ao invés de apenas receber uma transação que chega. Agora, tanto a fonte quanto o

sorvedouro podem decidir quando enviar ou receber, o que exige um mecanismo para evitar a perda de transações. Esse mecanismo é uma FIFO entre a fonte e o sorvedouro. O novo sorvedouro possui uma entrada uvm_get_port e uma run_phase com um laço infinito, que não significa que o sorvedouro nunca parará, já que a fonte controla o término da simulação. Dentro desse laço, a transação é recebida através do método get da porta de entrada e armazenada. A FIFO garante que as transações sejam armazenadas corretamente, permitindo que a fonte e o sorvedouro trabalhem independentemente, mas mantendo a sequência das transações.

```
UVM_INFO drain.svh(20) @ 80: uvm_test_top.env_JaquelineB..drain_h [DRAIN, by Brito Jaqueline ]
Transaction received

UVM_INFO source.svh(23) @ 120: uvm_test_top.env_JaquelineB..source_h [SOURCE] Sending transaction
by Jaqueline_Brito

UVM_INFO drain.svh(20) @ 120: uvm_test_top.env_JaquelineB..drain_h [DRAIN, by Brito Jaqueline ]
Transaction received

UVM_INFO source.svh(23) @ 180: uvm_test_top.env_JaquelineB..source_h [SOURCE] Sending transaction
by Jaqueline_Brito

UVM_INFO drain.svh(20) @ 180: uvm_test_top.env_JaquelineB..drain_h [DRAIN, by Brito Jaqueline ]
Transaction received

UVM_INFO source.svh(23) @ 240: uvm_test_top.env_JaquelineB..source_h [SOURCE] Sending transaction
by Jaqueline_Brito

UVM_INFO drain.svh(20) @ 240: uvm_test_top.env_JaquelineB..drain_h [DRAIN, by Brito Jaqueline ]
Transaction received

UVM_INFO source.svh(23) @ 300: uvm_test_top.env_JaquelineB..source_h [SOURCE] Sending transaction
by Jaqueline_Brito

UVM_INFO drain.svh(20) @ 300: uvm_test_top.env_JaquelineB..drain_h [DRAIN, by Brito Jaqueline ]
Transaction received

UVM_INFO source.svh(23) @ 360: uvm_test_top.env_JaquelineB..source_h [SOURCE] Sending transaction
by Jaqueline_Brito

UVM_INFO drain.svh(20) @ 360: uvm_test_top.env_JaquelineB..drain_h [DRAIN, by Brito Jaqueline ]
Transaction received

UVM_INFO
/usr/local/cds/XCELIUM2409/tools/methodology/UVM/CDNS-1.1d/sv/src/base/uvm_objection.svh(1268)
@ 460: 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 :   15
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[DRAIN, by Brito Jaqueline ]     6
[RNTST]     1
[SOURCE]     6
[TEST_DONE]     1
[UVM_CMDLINE_PROC]     1
```



2.8 Adicionando o modelo de referência

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

   uvm_get_port #(Jaque_a_tr) in;          // Solicitar nova transação, que pode ser
conectada na saida de uma FIFO
```

```
   uvm_blocking_put_port #(Jaque_a_tr) out;

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

   task run_phase (uvm_phase phase);

       Jaque_a_tr tr_in, tr_out; // variáveis para transações que entram e saem

       forever begin

       // laço infinito
       in.get(tr_in); // requisição da transação, se tiver disponível, vai retornar
tr_in

       #30;
       `bvm_end_tr(tr_in); // Aqui é marcado finalização da transação

       tr_out = Jaque_a_tr::type_id::create("Jaque_tr_out", this);
       tr_out.j = tr_in.j + 250; // alterando 100 para 250
       `bvm_begin_tr(tr_out)
       #30;
       out.put(tr_out); // Envia a transação para o modelo de referência
       end

   endtask

endclass
```

2.9 Alterando a classe env

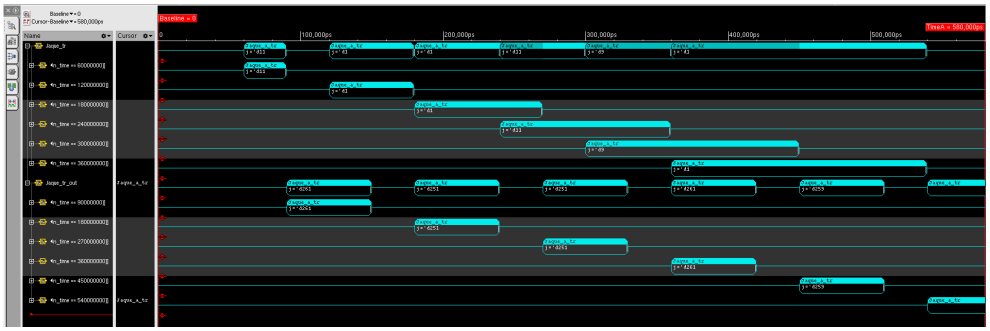
```
class env extends uvm_env;
   `uvm_component_utils(env)

   source source_h;
   uvm_tlm_fifo #(Jaque_a_tr) source_refmod; // adicionando FIFO para armazernar as
transações, entre o momento que o source disponibiliza, e que o drain que solicitar
   refmod refmod_jaque;
   sink sink_jaque;

   function new(string name, uvm_component parent);
       super.new(name, parent);
   endfunction

   function void build_phase(uvm_phase phase);
       super.build_phase(phase);
       source_h = source::type_id::create("source_h", this);
       source_refmod = new("source_refmod_by_Jaqueline", this,5);
       refmod_jaque = refmod::type_id::create("refmod_by_Jaqueline", this);
       sink_jaque = sink::type_id::create("sink_by_Jaqueline", this);
   endfunction

   function void connect_phase(uvm_phase phase);

       source_h.out.connect (source_refmod.put_export);
       refmod_jaque.in.connect( source_refmod.get_export );
       refmod_jaque.out.connect( sink_jaque.in );
   endfunction

endclass
```

2.10 Alterando test_pkg

```
`timescale 1ns/1ps
`include "uvm_macros.svh"
`include "bvm_macros.svh" // macros created by Brazil-IP / UFCG

package test_pkg;

  import uvm_pkg::*;

  `include "trans.svh"
  `include "source.svh"
  `include "refmod.svh"
  `include "sink.svh"
  `include "drain.svh"
  `include "env.svh"
  `include "test.svh"

endpackage
```

2.11 Simulação do refmod

```
UVM_INFO source.svh(23) @ 120000: uvm_test_top.env_JaquelineB..source_h [SOURCE] Sending
transaction by Jaqueline_Brito

UVM_INFO sink.svh(13) @ 120000: uvm_test_top.env_JaquelineB..sink_by_Jaqueline [SINK] Receiving
transaction by Jaqueline_Brito

UVM_INFO source.svh(23) @ 180000: uvm_test_top.env_JaquelineB..source_h [SOURCE] Sending
transaction by Jaqueline_Brito

UVM_INFO sink.svh(13) @ 210000: uvm_test_top.env_JaquelineB..sink_by_Jaqueline [SINK] Receiving
transaction by Jaqueline_Brito

UVM_INFO source.svh(23) @ 240000: uvm_test_top.env_JaquelineB..source_h [SOURCE] Sending
transaction by Jaqueline_Brito

UVM_INFO source.svh(23) @ 300000: uvm_test_top.env_JaquelineB..source_h [SOURCE] Sending
transaction by Jaqueline_Brito

UVM_INFO sink.svh(13) @ 300000: uvm_test_top.env_JaquelineB..sink_by_Jaqueline [SINK] Receiving
transaction by Jaqueline_Brito

UVM_INFO source.svh(23) @ 360000: uvm_test_top.env_JaquelineB..source_h [SOURCE] Sending
transaction by Jaqueline_Brito

UVM_INFO sink.svh(13) @ 390000: uvm_test_top.env_JaquelineB..sink_by_Jaqueline [SINK] Receiving
transaction by Jaqueline_Brito

UVM_INFO sink.svh(13) @ 480000: uvm_test_top.env_JaquelineB..sink_by_Jaqueline [SINK] Receiving
transaction by Jaqueline_Brito

UVM_INFO sink.svh(13) @ 570000: uvm_test_top.env_JaquelineB..sink_by_Jaqueline [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)
@ 580000: 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 :   15
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[RNTST]     1
[SINK]     6
[SOURCE]     6
[TEST_DONE]     1
[UVM_CMDLINE_PROC]     1
```



3. Conclusão

O relatório apresentou a continuação de uma simulação UVM inicial, que continha apenas os componentes básicos (source, transaction e sink, transações e test), para um ambiente de teste mais modular e flexível.

As principais mudanças incluem:

1. Modularização e Uso da Classe env
   - A classe test foi simplificada, deixando a responsabilidade da configuração do ambiente para a classe env.
   - env agora usa a fábrica UVM (type_id::create) em vez de new, permitindo maior flexibilidade na criação dos componentes.
2. Alterações no source
   - O source passou a gerar seis transações, enviadas via put_port, em vez das dez transações originais.
   - Foi implementado o uso da fábrica UVM para criar transações dinamicamente.
3. Introdução da FIFO e Novo Sorvedouro (drain)
   - Um mecanismo FIFO foi adicionado entre source e drain, permitindo que fonte e sorvedouro operem de forma independente sem perder a sequência das transações.
   - O drain solicita transações via uvm_get_port em vez de simplesmente recebê-las.
4. Criação do Modelo de Referência (refmod)
   - Foi implementado um modelo de referência (refmod) que recebe transações, processa os dados e os encaminha ao sink.
   - Uma FIFO foi adicionada para armazenar transações entre o source e o refmod.
5. Ajustes na Classe env
   - Conexões foram atualizadas para incluir a FIFO entre source e refmod, além da conexão do refmod ao sink.
6. Reorganização do test_pkg
   - O código foi modularizado, encapsulando todas as definições de classes dentro de um pacote (test_pkg).
   - O top.sv foi simplificado, contendo apenas os comandos de importação e execução do teste.
7. Simulação do refmod
   - A simulação foi ajustada para considerar o fluxo atualizado de transações através do modelo de referência e FIFO.