

Universidade Federal do Rio Grande do Sul
Programa de Pós Graduação em Microeletrônica
MIC61 - Concepção de Circuitos Integrados II

Verificação de uma ULA com diferentes operações matemáticas

Karina Kerne
Julho
2023

1 Introdução

O presente relatório trata da verificação de uma Unidade Lógica Aritmética (ULA/ALU) capaz de realizar 7 operações. O desenvolvimento do ambiente de verificação teve como base um tutorial oferecido por Tulio Pereira Bitencourt e Pedro Paiva.

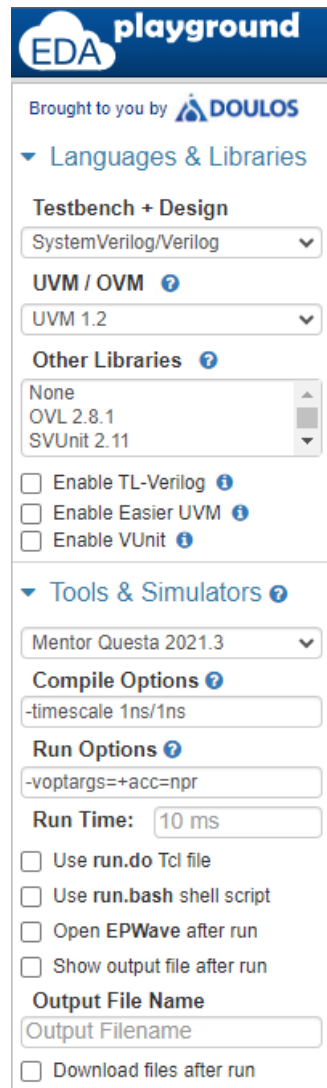
O projeto foi desenvolvido utilizando a ferramenta EDA Playground (que é de uso livre) com base na técnica de *Metric Driven Verification* que possui 4 passos: Planejamento, construção de um ambiente de verificação baseado em *Randomization*, execução dos cenários de teste e medição/análise do processo (*Coverage*). Os ambientes são construídos utilizando a metodologia UVM em linguagem SystemVerilog.

Ao longo deste relatório, são mostradas as etapas do desenvolvimento de um ambiente de verificação e os resultados obtidos para este projeto.

2 Metodologia

2.1 Setup do ambiente

Para desenvolver o projeto, utilizou-se o site EDA Playground com as configurações mostradas na Figura 1.



The image shows the configuration panel of the EDA Playground website. At the top, it says "Brought to you by DOULOS". Below this, there are two main sections: "Languages & Libraries" and "Tools & Simulators".

Languages & Libraries:

- Testbench + Design:** Set to "SystemVerilog/Verilog".
- UVM / OVM:** Set to "UVM 1.2".
- Other Libraries:** A list box containing "None", "OVL 2.8.1", and "SVUnit 2.11".
- Three checkboxes are present: "Enable TL-Verilog", "Enable Easier UVM", and "Enable VUnit", all of which are currently unchecked.

Tools & Simulators:

- Set to "Mentor Questa 2021.3".
- Compile Options:** A text box containing "-timescale 1ns/1ns".
- Run Options:** A text box containing "-voptargs=+acc=npr".
- Run Time:** A text box containing "10 ms".
- Four checkboxes are present: "Use run.do Tcl file", "Use run.bash shell script", "Open EPWave after run", and "Show output file after run", all of which are currently unchecked.
- Output File Name:** A text box containing "Output Filename".
- A checkbox "Download files after run" is at the bottom, which is currently unchecked.

Figura 1: Configurações para executar programa de verificação

Na Figura 2 é mostrada uma visualização completa do site. No lado esquerdo ficam os arquivos que compõe a UVM e no lado direito os arquivos que descrevem o projeto RTL específico. Abaixo é mostrado um terminal com o log para visualização do resultado após execução do programa.

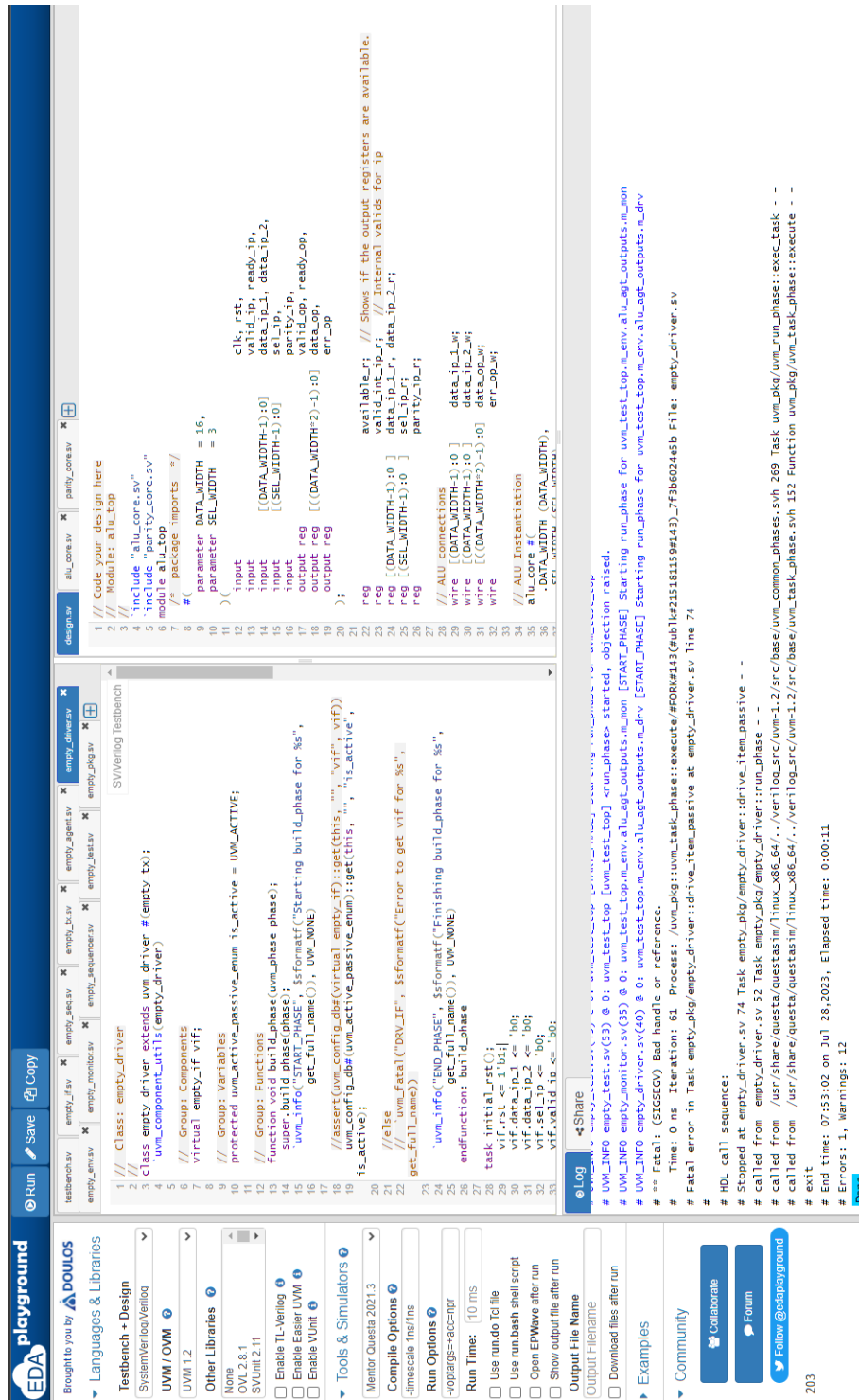


Figura 2: Ambiente completo do site EDA playground

Para a criação do ambiente UVM foram adicionados os arquivos:

- testbench.sv

- empty_if.sv
- empty_seq.sv
- empty_tx.sv
- empty_agent.sv
- empty_driver.sv
- empty_env.sv
- empty_monitor.sv
- empty_sequencer.sv
- empty_test.sv
- empty_pkg.sv

Que são as partes que compõe a estrutura de teste conforme mostrada na Figura 3.

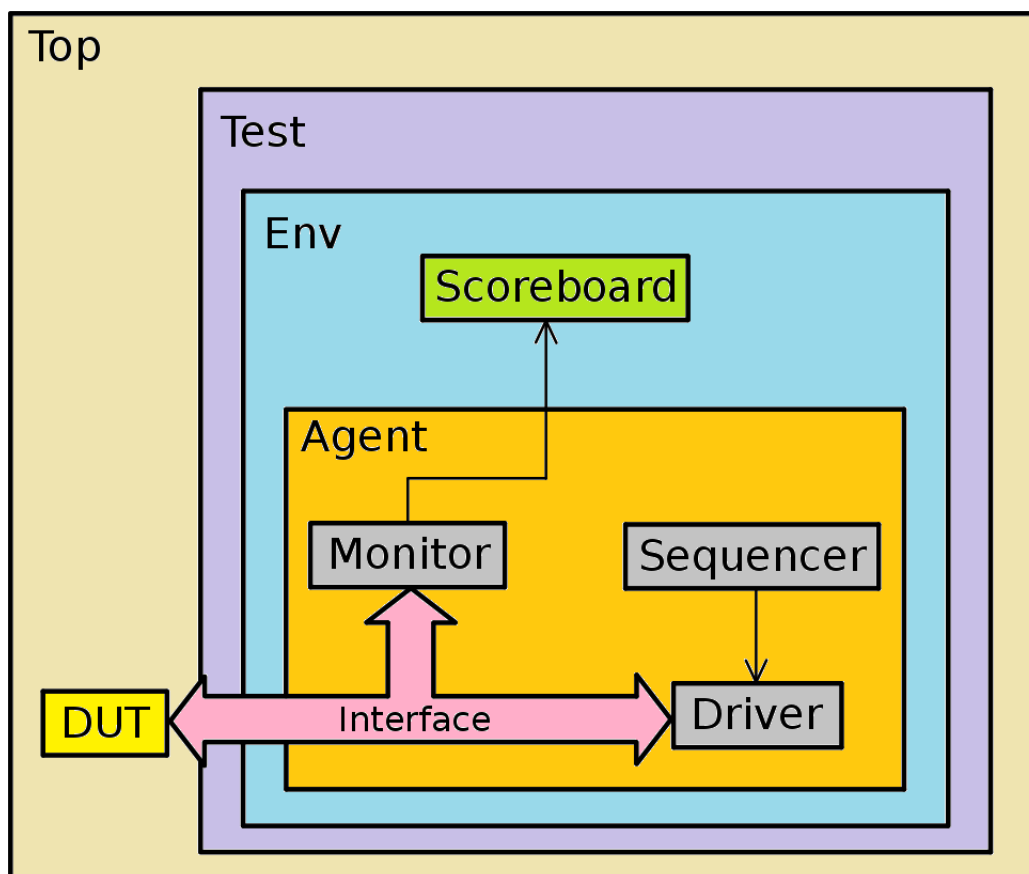


Figura 3: Estrutura de teste UVM

E para descrição do core a ser testado, utilizou-se os arquivos:

- design.sv
- alu_core.sv
- parity_core.sv

É relevante destacar que os engenheiros de verificação tentam reutilizar IPs(*intellectual properties*) e arquivos de restrição para criar o ambiente de verificação que cubra o plano de verificação proposto.

Durante a execução de um programa de verificação, testes de regressão são executados para cada revisão do projeto e métricas como *Code coverage* e *Functional coverage* são obtidas. *Code coverage* indica quanto do código RTL foi usado durante a simulação e é uma métrica analisada automaticamente pelo simulador. *Functional coverage* indica quanto das funcionalidades pré definidas foram abordadas. A *Functional coverage* é planejada pelos engenheiros de verificação.

3 Resultados e Discussões

O resultado da execução do programa é mostrado nas Figuras 4, 5 e 6. Pode-se observar que a execução foi satisfatória demonstrando que o projeto está corretamente desenvolvido. O projeto completo também pode ser conferido pelo link: [EDA Playground - Verificação](#).

```
# ** Note: $finish      : /usr/share/questa/questasim/linux_x86_64/./verilog_src/uvm-1.2/src/base/uvm_root.svh(517)
#   Time: 162 ns  Iteration: 75  Instance: /testbench
# End time: 15:25:39 on Aug 15,2023, Elapsed time: 0:00:07
# Errors: 0, Warnings: 12
Finding user-specified file...
File not found. User-specified file will not open. You requested to open the file 'trab_verif_karina' but your code does not cr
Done
```

Figura 4: Resultados obtidos

```
# -----
# Name                                Type                                Size  Value
# -----
# uvm_test_top                        empty_test                          -    @360
#   m_env                             empty_env                           -    @374
#     alu_agt_inputs                   empty_agent                          -    @389
#       m_drv                          empty_driver                         -    @540
#         rsp_port                      uvm_analysis_port                   -    @557
#           seq_item_port               uvm_seq_item_pull_port              -    @548
#             m_mon                     empty_monitor                        -    @566
#               mon_analysis_port        uvm_analysis_port                   -    @589
#                 m_seqr                 uvm_sequencer                       -    @417
#                   rsp_export            uvm_analysis_export                 -    @425
#                     seq_item_export     uvm_seq_item_pull_imp               -    @531
#                       arbitration_queue array                               0      -
#                         lock_queue      array                               0      -
#                           num_last_reqs integral                          32    'd1
#                             num_last_rsps integral                          32    'd1
#                               alu_agt_outputs empty_agent                          -    @397
#                                 m_drv      empty_driver                         -    @610
#                                   rsp_port  uvm_analysis_port                   -    @627
#                                     seq_item_port uvm_seq_item_pull_port              -    @618
#                                       m_mon    empty_monitor                        -    @636
#                                         mon_analysis_port uvm_analysis_port                   -    @659
# -----
```

Figura 5: Resultados obtidos

```
..
# ** Report counts by severity
# UVM_INFO :    46
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [END_PHASE]    12
# [PRINT_ITEM]   10
# [Questa UVM]   2
# [RNTST]        1
# [START_PHASE]  16
# [TEST_DONE]    1
# [UVM/RELNOTES]  1
# [UVMTOP]       1
# [uvm_test_top]  2
#
```

Figura 6: Resultados obtidos

4 Conclusão

A verificação garante que o projeto atenda as especificações exigidas do produto e opere corretamente. Isso reduz os riscos tomados ao enviar um circuito para fabricação.

O uso da metodologia UVM proporciona o reaproveitamento de técnicas de programação orientada a objetos para ajudar na reutilização de código.

O desenvolvimento deste projeto proporcionou uma visão inicial de como funciona a verificação funcional de um circuito e introduziu conhecimentos essenciais para a atuação no campo da microeletrônica.