

# AUTOMAÇÃO DE PIPELINE PARA VERIFICAÇÃO FORMAL DE CIRCUITOS EM VHDL

PROJETO FINAL  
ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES (2025)

AMANDA DE BRITO BARBOSA, FÁBIO AURÉRIO BARROS ALEXANDRE, JOSÉ CARVALHO NETO

## INTRODUÇÃO E OBJETIVOS



**Desafio:** Crescente complexidade de projetos em VHDL e limitações da simulação tradicional.

**Solução:** Verificação Formal via Model Checking (análise exaustiva de estados).

**Abordagem:** Uso de tecnologias abertas para tradução de hardware para software.

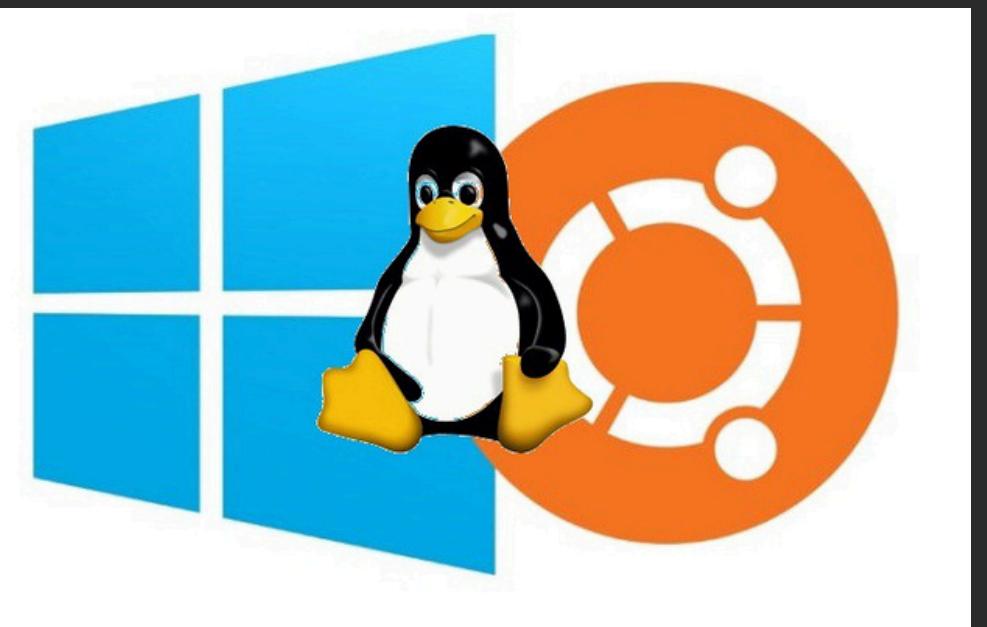
## AMBIENTE



# Yosysho



**Quartus**  
Prime



# TESTE (TÓPICO 1 E 2)

Códigos:

1. teste\_downto
2. teste\_integer
3. teste\_array
4. teste\_clock

```
|  
| THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES  
TIES |  
| WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF  
|  
| MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE  
FOR |  
| ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES  
ES |  
| WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN A  
N |  
| ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT  
OF |  
| OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.  
|  
|  
\\-----/
```

Yosys 0.9 (git sha1 1979e0b)

```
-- Running command 'ghdl teste_downto.vhd -e teste_downto; prep; write_v  
erilog elaborado.v' --  
ERROR: No such command: ghdl (type 'help' for a command overview)  
amanda@DESKTOP-SNJM090:/mnt/d/Semestre 2025.2/Arquitetura_e_Organizacao/  
projeto_final/codigos/teste_downto$
```

# Automação do Pipeline

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity teste_integer is
    port (
        val_in : in integer range 0 to 15;
        val_out : out integer range 0 to 31
    );
end teste_integer;

architecture behavioral of teste_integer is
begin
    process(val_in)
    begin
        -- Operação simples para testar propagação de limites
        val_out <= val_in + 10;
    end process;
end behavioral;
```

Círculo VHDL para teste

# Automação do Pipeline

```
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assertions in step 11..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assumptions in step 12..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assertions in step 12..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assumptions in step 13..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assertions in step 13..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assumptions in step 14..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assertions in step 14..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assumptions in step 15..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assertions in step 15..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assumptions in step 16..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assertions in step 16..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assumptions in step 17..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assertions in step 17..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assumptions in step 18..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assertions in step 18..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assumptions in step 19..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Checking assertions in step 19..  
SBY 23:40:24 [teste_integer] engine_0.basecase: ## 0:00:00 Status: passed  
SBY 23:40:24 [teste_integer] engine_0.basecase: finished (returncode=0)  
SBY 23:40:24 [teste_integer] engine_0.basecase: Status returned by engine for basecase: pass  
SBY 23:40:24 [teste_integer] summary: Elapsed clock time [H:MM:SS (secs)]: 0:00:01 (1)  
SBY 23:40:24 [teste_integer] summary: Elapsed process time [H:MM:SS (secs)]: 0:00:00 (0)  
SBY 23:40:24 [teste_integer] summary: engine_0 (smtbmc) returned pass for basecase  
SBY 23:40:24 [teste_integer] summary: engine_0 (smtbmc) returned pass for induction  
SBY 23:40:24 [teste_integer] summary: engine_0 did not produce any traces  
SBY 23:40:24 [teste_integer] summary: successful proof by k-induction.
```

Análise de Vacuidade

```
module verify_teste_integer (  
    input [3:0] val_in,  
    output [4:0] val_out  
);  
  
    teste_integer dut (  
        .val_in(val_in),  
        .val_out(val_out)  
    );  
  
    always @(*) begin  
    end  
endmodule
```

Arquivo auxiliar gerado pelo script

# Automação do Pipeline

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity teste_integer is
    port (
        val_in : in integer range 0 to 15;
        val_out : out integer range 0 to 31
    );
    -- TAGS DE VERIFICAÇÃO PARA O SCRIPT PYTHON:
    -- O range de entrada é 0-15 (4 bits), restrinjimos a busca a isso:
    -- @c2vhdl:ASSUME val_in <= 15;

    -- A saída deve ser a entrada + 10:
    -- @c2vhdl:ASSERT val_out == val_in + 10;

    -- A saída não deve estourar 31:
    -- @c2vhdl:ASSERT val_out <= 31;
end teste_integer;

architecture behavioral of teste_integer is
begin
    process(val_in)
    begin
        val_out <= val_in + 10;
    end process;
end behavioral;
```

Inserção de regras/tags

# Automação do Pipeline

```
# 3. Extrair Tags @c2vhdl
# Procura por linhas -- @c2vhdl:TYPE regra
lines = content.split('\n')
for line in lines:
    line = line.strip()
    if "-- @c2vhdl:" in line:
        # Separa o tipo (ASSERT/ASSUME) do conteúdo
        parts = line.split("@c2vhdl:", 1)[1].strip().split(" ", 1)
        tag_type = parts[0].upper() # ASSUME ou ASSERT
        rule = parts[1].strip()

        # Limpeza: remove ponto e vírgula final se existir
        if rule.endswith(";"):
            rule = rule[:-1]

        if tag_type == "ASSUME":
            info["assumes"].append(rule)
        elif tag_type == "ASSERT":
            info["asserts"].append(rule)

return info
```

Script que extrai tags dos arquivos em VHDL

```
module verify_teste_integer (
    input [3:0] val_in,
    output [4:0] val_out
);

teste_integer dut (
    .val_in(val_in),
    .val_out(val_out)
);

always @(*) begin
    assume (val_in <= 15);
    assert (val_out == val_in + 10);
    assert (val_out <= 31);
end
endmodule
```

Arquivo auxiliar gerado pelo script

# Automação do Pipeline

```
-- @c2vhdl:ASSUME val_in <- 10;  
  
-- A saída deve ser a entrada + 10:  
-- @c2vhdl:ASSERT val_out == val_in + 10;  
  
-- A saída não deve estourar 31:  
-- @c2vhdl:ASSERT val_out <= 20;  
end teste_integer;  
  
architecture behavioral of teste_integer is  
begin
```

Mudança na regra

# Automação do Pipeline

```
SBY 16:17:46 [teste_integer] engine_0.induction: ## 0:00:00 Writing trace to VCD file: engine_0/trace_induct.vcd
SBY 16:17:46 [teste_integer] engine_0.basecase: finished (returncode=1)
SBY 16:17:46 [teste_integer] engine_0.basecase: Status returned by engine for basecase: FAIL
SBY 16:17:46 [teste_integer] engine_0.induction: terminating process
SBY 16:17:46 [teste_integer] summary: Elapsed clock time [H:MM:SS (secs)]: 0:00:00 (0)
SBY 16:17:46 [teste_integer] summary: Elapsed process time [H:MM:SS (secs)]: 0:00:00 (0)
SBY 16:17:46 [teste_integer] summary: engine_0 (smtbmc) returned FAIL for basecase
SBY 16:17:46 [teste_integer] summary: counterexample trace [basecase]: teste_integer/engine_0/trace.vcd
SBY 16:17:46 [teste_integer] summary: failed assertion verify_teste_integer._witness_.check_assert_verif_teste_integer
_sv_17_7 at verif_teste_integer.sv:17.9-17.31 in step 0
SBY 16:17:46 [teste_integer] summary: counterexample trace [induction]: teste_integer/engine_0/trace_induct.vcd
SBY 16:17:46 [teste_integer] summary: failed assertion verify_teste_integer._witness_.check_assert_verif_teste_integer
_sv_17_7 at verif_teste_integer.sv:17.9-17.31 in step 0
SBY 16:17:46 [teste_integer] DONE (FAIL, rc=2)
```

## Demonstração de falha

# Tradução para C

```
Saída: |-----  
|***** v2c 0.1 *****|  
|***** Verilog to C Translator *****|  
|***** Authors: Rajdeep Mukherjee and Michael Tautschnig *****|  
|*****|  
|***** Erro: file elaborado_clock.v line 17: syntax error before '(*'  
PARSING ERROR  
PARSING ERROR  
  
Dica: v2c pode não suportar todos os tipos do Verilog  
[INFO] Arquivo C 'teste_clock.c' não existe. Pulando ESBMC.
```

```
***** v2c 0.1 *****  
***** Verilog to C Translator *****  
***** Authors: Rajdeep Mukherjee and Michael Tautschnig *****  
***** Developed at University of Oxford *****  
***** Usage: v2c main.v --module <name of top module> main.c *****  
-----  
file elaborado.v: Parsing  
Parsing elaborado.v  
Converting  
Type-checking Verilog::teste_downto  
terminate called after throwing an instance of 'int'  
Aborted (core dumped)
```

## Erro de tradução para C

# Front-end unificador

## Dashboard — TASK 04

Lê task04/results/summary.json e mostra o status por etapa (vhd2vl / yosys / sby / v2c / esbmc) + links para logs e Common AST.

Filtrar por nome do design (ex.: integer, cl)    Etapa (todas)    Status (todos)    Recarregar

Itens: 4 (de 4)

Design	VHDL	Spec	vhd2vl	yosys_prep	sby	v2c	esbmc	Common AST	Notas
teste_array	VHDL	spec.json	SKIP	FAIL	SKIP	SKIP	SKIP	ast.json	vhd2vl not found in PATH (configure/install) • AST generated from VHDL only (no yosys.json).
teste_clock	VHDL	spec.json	SKIP	FAIL	SKIP	SKIP	SKIP	ast.json	vhd2vl not found in PATH (configure/install) • AST generated from VHDL only (no yosys.json).
teste_downto	VHDL	spec.json	SKIP	OK	SKIP	SKIP	SKIP	ast.json	vhd2vl not found in PATH (configure/install)
teste_integer	VHDL	spec.json	SKIP	FAIL	SKIP	SKIP	SKIP	ast.json	vhd2vl not found in PATH (configure/install) • AST generated from VHDL only (no yosys.json).

O que este front-end resolve:

- Visibilidade imediata do status por etapa e por design
- Links diretos para artefatos (spec.json e \*.ast.json)
- Campo “Notas” registra causas de SKIP/FAIL (ex.: tool ausente no PATH)
- Filtros (nome/etapa/status) ajudam a depurar rapidamente

# Front-end unificador

```
end behavioral;
```

```
os_avalio@osavalio-pc:~/ADC_desenvolvimento_AcordoDellaTolleranza_FabrikurelTolleranzo_UFBB_2020$ sed -n '1,20p' tasks04/results/ast/teste_downto.ast.json
```

```
{
```

```
  "schema": "task04-common-ast-v1",
```

```
  "design_name": "teste_downto",
```

```
  "source_vhdl": "task04/inputs/vhdl/teste_downto.vhd",
```

```
  "source_verilog": "task04/generated/yosys_json/teste_downto.json",
```

```
  "ports": [
```

```
    {
```

```
      "name": "input_bus",
```

```
      "direction": "input",
```

```
      "vhdl_type": "",
```

```
      "width": 8
```

```
    },
```

```
    {
```

```
      "name": "output_bus",
```

```
      "direction": "output",
```

```
      "vhdl_type": "",
```

```
      "width": 8
```

```
    }
```

```
  ],
```

```
  "properties": [
```

```
    {
```

```
      "label": "assert",
```

```
      "expr": "input_bus == output_bus",
```

```
      "msg": "",
```

```
      "source_line": 11
```

```
    }
```

```
  ],
```

```
  "wires": [
```

```
    {
```

```
      "name": "input_bus",
```

```
      "width": 8
```

```
    },
```

```
    {
```

```
      "name": "output_bus",
```

```
      "width": 8
```

```
    }
```

```
  ],
```

```
  "cells": {},
```

```
  "stats": {
```

```
    "has_clock": false,
```

```
    "yosys_top": "teste_downto",
```

```
    "cell_count": 0,
```

```
    "wire_count": 2
```

```
  },
```

```
  "notes": []
```

```
}
```

```
os_avalio@osavalio-pc:~/ADC_desenvolvimento_AcordoDellaTolleranza_FabrikurelTolleranzo_UFBB_2020$
```

## Por que um AST comum?

- Padroniza a interface do design (ports + largura)
- Preserva propriedades extraídas das tags (@c2vhdl:ASSERT/ASSUME)
- Centraliza metadados (source\_vhdl, source\_verilog)
- Permite anexar estrutura quando disponível (wires/cells/stats)
- Facilita depuração: o dashboard aponta direto para o \*.ast.json

Print do ast sendo confirmado