



# Objetivo do Projeto



## Estrutura da Linguagem

Criar uma linguagem com variáveis, condicionais e loops.



## Ferramentas

Utilizar EBNF para estruturar a linguagem e Flex/Bison para análise.



## Saída

Gerar código Verilog como alternativa ao LLVM.



# Motivação

## Mercado Financeiro

O high frequency trading (HFT) depende de regras simples, expressivas e rápidas.

## Proposta

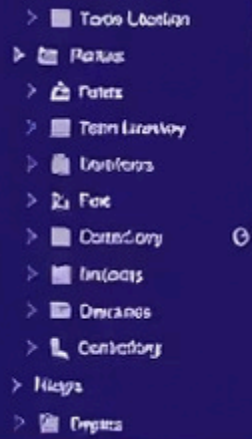
Permitir que programadores descrevam estratégias de trading em linguagem intuitiva.

## Validação

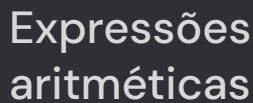
Tradução automática gera código Verilog para simulação em plataformas como EDA Playground.

# EBNF Final da Linguagem

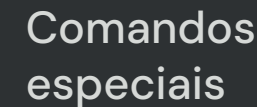
```
PROGRAM = { STATEMENT } ;  
STATEMENT = VAR_DECL | CONDITIONAL | LOOP | EMIT_STMT ;  
VAR_DECL = "LET", IDENTIFIER, "=", EXPR, ";" ;  
CONDITIONAL = "IF", "(", CONDITION, ")", "{", { STATEMENT }, "}" ;  
LOOP = "LOOP", EXPR, "TIMES", "{", { STATEMENT }, "}" ;  
EMIT_STMT = "EMIT", ACTION, EXPR, ";" ;  
CONDITION = IDENTIFIER, COMPARISON_OPERATOR, EXPR ;  
EXPR = TERM, { ("+" | "-"), TERM } ; TERM = FACTOR, { ("*" | "/"), FACTOR } ;  
FACTOR = NUMBER | IDENTIFIER ;  
COMPARISON_OPERATOR = ">" | "<" | "==" ;  
ACTION = "BUY" | "SELL" ;
```

 $\sqrt{\times}$ 

Suporte a variáveis inteiras com palavra-chave LET.



## Comparação de variáveis com operadores >, < e ==.



EMIT BUY e EMIT  
SELL traduzem  
diretamente para  
Verilog.

# Como funciona



## Código de entrada (.pv)

Programador escreve estratégia em Pivota



## Processamento

Analizador léxico e sintático processa o código



## Código Verilog

Geração automática de módulo Verilog equivalente

# Código Gerado

## Entrada Pivota

Código com variáveis, condicionais e loops

## Validação

Simulação em ambiente EDA



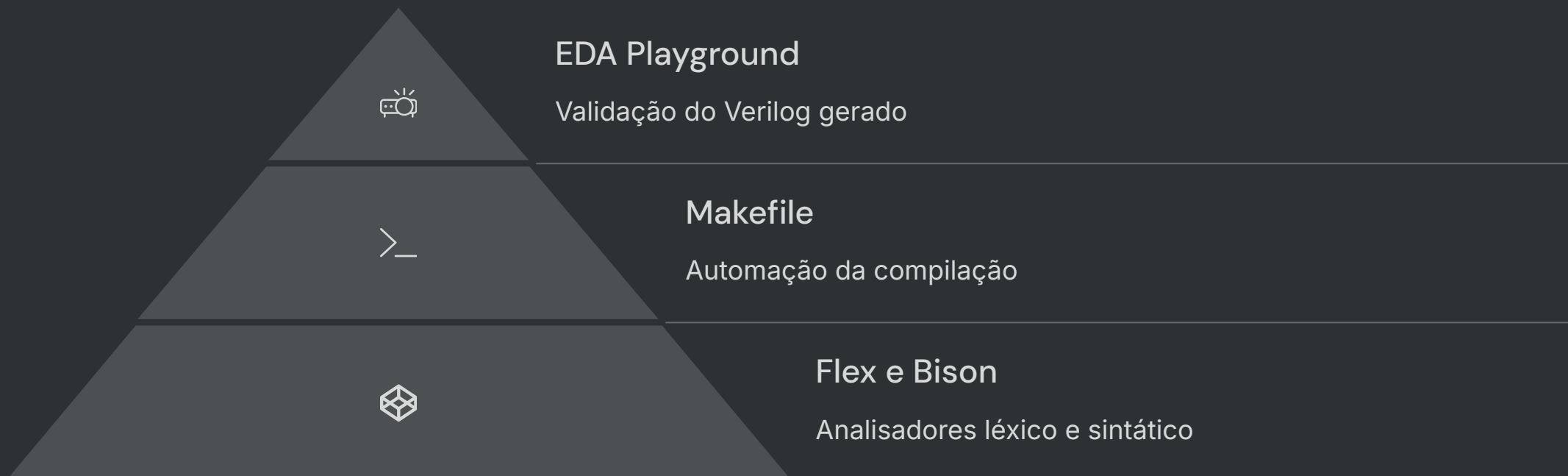
## Tradução

Conversão automática para Verilog

## Saída Verilog

Módulo com registradores e lógica equivalente

# Ferramentas utilizadas





# Testes no EDA Playground

## Interface de Simulação

Código da esquerda: módulo Verilog gerado pela nossa linguagem.

## Testbench

Código da direita: simula a execução e imprime a última ordem executada.

## Resultados

Log mostra os valores finais de order e qty definidos.

playground

NewRunSaveCopy

KnowHow

Essential SystemVerilog Assertions (Reading, UK)

FIND OUT MORE

ResourcesCommunityHelpPlaygroundsProfile

Brought to you by DOULOS

Languages & Libraries

Testbench + Design

SystemVerilog/Verilog

UVM / OVM

None

Other Libraries

NoneOVL SVUnit

Tools & Simulators

Icarus Verilog 12.0

Compile Options

-Wall -g2012

Run Options

Run Options

Use run.bash shell script

Open EPWave after run

Show output file after run

Output File Name

teste

Download files after run

Examples

using EDA Playground

VHDL

Verilog/SystemVerilog

UVM

EasierUVM

SVAUnit

SVUnit

VUnit (Verilog/SV)

VUnit (VHDL)

TL-Verilog

testbench.sv

1 module pivota\_strategy(output reg [3:0] dummy);  
2 reg [3:0] order\_list [0:255];  
3 reg [3:0] qty\_list [0:255];  
4 integer i;  
5 reg [31:0] price, volume, threshold, total, correction, reps, sum, diff;  
6 initial begin  
7 i = 0;  
8 price = 120;  
9 volume = 15;  
10 threshold = 100;  
11 total = 1800;  
12 correction = 1000;  
13 reps = 4;  
14 sum = 135;  
15 diff = 115;  
16 if (1800 > 100) begin  
17 order\_list[i] = 1; qty\_list[i] = 15; i = i + 1;  
18 end  
19 if (1000 < 150) begin  
20 order\_list[i] = 2; qty\_list[i] = 3; i = i + 1;  
21 end  
22 repeat(3) begin  
23 order\_list[i] = 1; qty\_list[i] = 1; i = i + 1;  
24 end  
25 repeat(4) begin  
26 order\_list[i] = 2; qty\_list[i] = 4; i = i + 1;  
27 end  
28 if (115 == 115) begin  
29 order\_list[i] = 1; qty\_list[i] = 115; i = i + 1;  
30 end  
31 end  
32 endmodule

design.sv

1 module test;  
2 wire [3:0] dummy;  
3 pivota\_strategy uut (.dummy(dummy));  
4  
5 initial begin  
6 #1;  
7 for (int j = 0; j < uut.i; j = j + 1) begin  
8 \$display("Order: %d Qty: %d", uut.order\_list[j], uut.qty\_list[j]);  
9 end  
10 \$finish;  
11 end  
12 endmodule  
13

LogShare

[2025-06-11 01:31:14 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out  
Order: 1 Qty: 15  
Order: 1 Qty: 1  
Order: 1 Qty: 1  
Order: 1 Qty: 1  
Order: 2 Qty: 4  
Order: 2 Qty: 4  
Order: 2 Qty: 4  
Order: 2 Qty: 4  
Order: 1 Qty: 3  
design.sv:10: \$finish called at 1 (1s)  
Finding user-specified file...  
File not found. User-specified file will not open. You requested to open the file 'teste' but your code does not create a file with this name.

By using our website, you agree to the usage of cookies.

Hide

Made with GAMMA