

[illegible]

Uma linguagem de domínio específico (DSL) para estratégias de negociação automatizadas com geração de código Verilog.

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

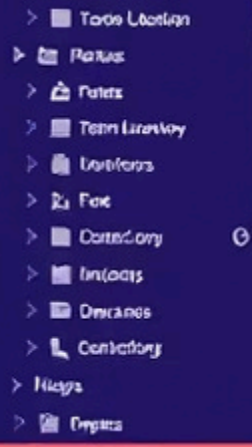
Permitir que programadores descrevam estratégias de trading em linguagem intuitiva.

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 | "(", EXPR, ")" ;  
COMPARISON_OPERATOR = ">" | "<" | "==" ;  
ACTION = "BUY" | "SELL" ;  
IDENTIFIER = LETTER, { LETTER | DIGIT | "_" } ;  
NUMBER = DIGIT, { DIGIT } ;  
LETTER = "a" | ... | "z" | "A" | ... | "Z" ;  
DIGIT = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ;
```



Características da Linguagem



Declaração de variáveis

Suporte a variáveis inteiras com palavra-chave LET.



Expressões aritméticas

Operações com +, -, *, / e parênteses.



Conditionais

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



Comandos especiais

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



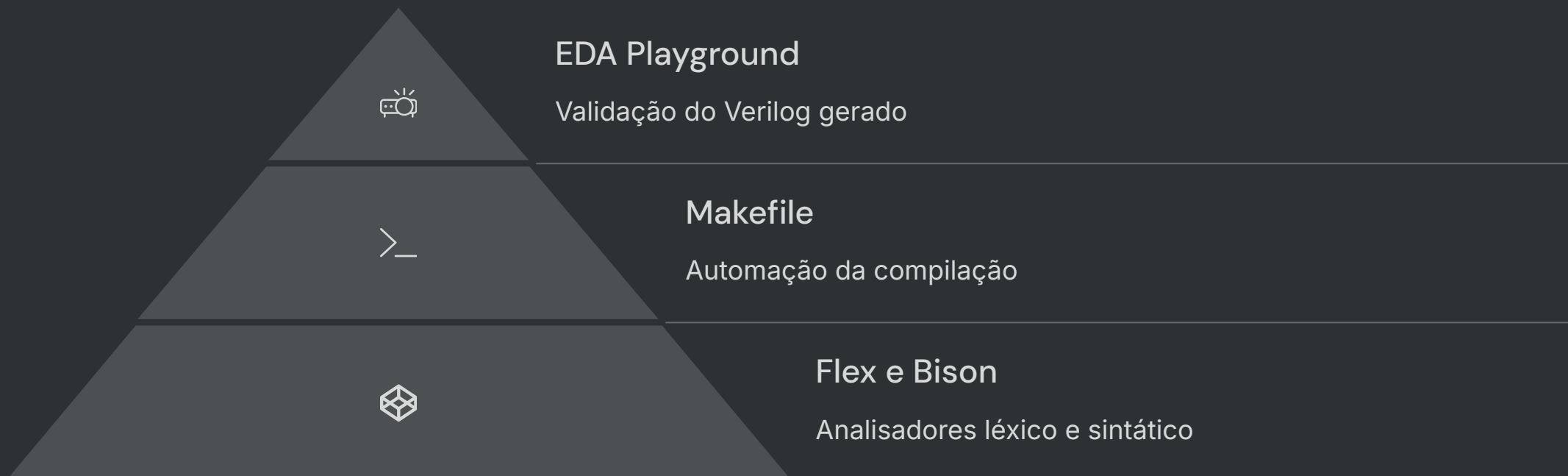
Tradução

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.

The screenshot displays the EDA Playground web interface. The top navigation bar includes the EDA Playground logo, a 'New' button, and buttons for 'Run', 'Save*', and 'Copy*'. A banner for 'KnowHow WORKSHOPS' promotes an 'Essential SystemVerilog Assertions' event in Reading, UK, on June 30, 2025, with a 'FIND OUT MORE' link. The main interface is divided into three sections: a left sidebar, a central code editor, and a right code editor.

Left Sidebar: Contains a 'Brought to you by DOULOS' notice, a 'Languages & Libraries' section with 'Testbench + Design' (SystemVerilog/Verilog) and 'UVM / OVM' (None) dropdowns, and 'Other Libraries' (None, OVL, SVUnit). Below this is 'Tools & Simulators' (Icarus Verilog 12.0) and 'Compile Options' (-Wall -g2012). 'Run Options' include checkboxes for 'Use run.bash shell script', 'Open EPWave after run', and 'Show output file after run' (checked). The 'Output File Name' is 'saida'. At the bottom, there is a 'Download files after run' checkbox and an 'Examples' link.

Central Code Editor (testbench.v): Contains the following Verilog code:

```
1 module pivota_strategy(output reg [3:0] order, output reg [3:0] qty);
2 reg [31:0] price, volume, threshold, total, correction, i;
3 initial begin
4     price = 120;
5     volume = 15;
6     threshold = 100;
7     total = 1800;
8     correction = 1720;
9     i = 0;
10    if (1800 > 100) begin
11        order = 1; qty = 15;
12    end
13    if (1720 < 50) begin
14        order = 2; qty = 3;
15    end
16    repeat(3) begin
17        order = 1; qty = 1;
18    end
19    repeat(4) begin
20        order = 2; qty = 1;
21    end
22 end
23 endmodule
```

Right Code Editor (design.v): Contains the following Verilog code:

```
1 module test;
2     wire [3:0] order;
3     wire [3:0] qty;
4
5     pivota_strategy uut (
6         .order(order),
7         .qty(qty)
8     );
9
10    initial begin
11        #1;
12        $display("Final Order: %d, Qty: %d", order, qty);
13        $finish;
14    end
15 endmodule
```

Log Output: At the bottom, a log entry shows the command: `[2025-06-10 06:20:05 UTC] iverilog '-Wall' '-g2012' design.v testbench.v && unbuffer vvp a.out` and the output: `Final Order: 2, Qty: 1`.