Veryl 新しいハードウェア記述言語

PEZY Computing

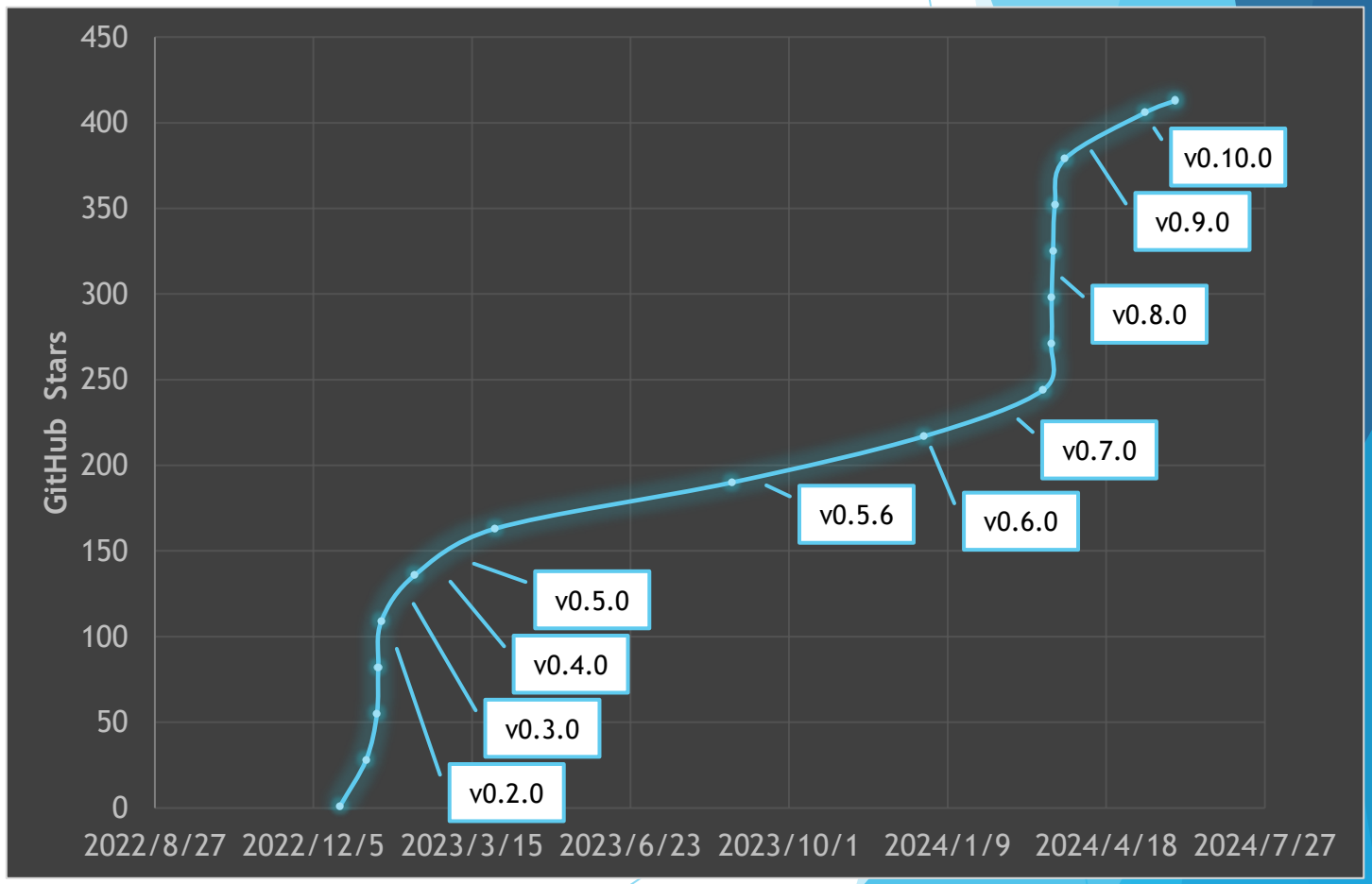
初田 直也

自己紹介

- 名前：初田 直也
  - dalance @ GitHub
- 所属：PEZY Computing
  - スーパーコンピュータ向けのプロセッサLSIの設計
  - SystemVerilogを使用
- OSS活動
  - SystemVerilog向けツール
    - sv-parser/svlint/svls

Veryl

- SystemVerilogに代わる新しいHDLを開発中
  - 2022年末に開発開始
  - 400 stars @ GitHub
    - 新しいHDLとしては中堅くらい？



なぜ新しいHDLを作るのか

- SystemVerilogへの不満
  - 構文が複雑すぎる
    - 商用EDAツールですら完全対応はできていない
    - 自作ツールを作るのも大変
- 既存のAlt-HDL（Chiselなど）への不満
  - プログラミング言語の内部DSLがほとんど
    - Chisel on Scala, MyHDL on Python
  - HDLとしては構文が不自然
    - ビット幅指定リテラルや信号方向、クロック・リセットなど
  - 大量のVerilogが生成される
    - 可読性が低くデバッグが困難
    - SystemVerilogと繋がらない
      - interfaceなどをいちいち展開しないといけない

Verylの目標

- 合成可能なRTLに最適化した構文
  - 最近のプログラミング言語（RustやGoなど）の知見を取り入れる
- SystemVerilogとの相互運用性
  - 可読性の高いSystemVerilogを生成する
- 生産性の高い開発環境
  - フォーマッタやLanguage Serverを標準で提供する

Verylの特徴（1 / 3）

- 基本的な構文

SystemVerilog

```
// SystemVerilog code

// Counter
module Counter #(
    parameter WIDTH = 1
)(
    input  logic             i_clk  ,
    input  logic             i_rst_n,
    output logic [WIDTH-1:0] o_cnt
);
    logic [WIDTH-1:0] r_cnt;

    always_ff @ (posedge i_clk or negedge i_rst_n) begin
        if (!i_rst_n) begin
            r_cnt <= 0;
        end else begin
            r_cnt <= r_cnt + 1;
        end
    end

    always_comb begin
        o_cnt = r_cnt;
    end
endmodule
```

Veryl

```
// Veryl code

/// Counter
module Counter #(
    param WIDTH: u32 = 1,
)(
    i_clk: input  clock       ,
    i_rst: input  reset       ,
    o_cnt: output logic<WIDTH>,
){
    var r_cnt: logic<WIDTH>;

    always_ff {
        if_reset {
            r_cnt = 0;
        } else {
            r_cnt += 1;
        }
    }

    always_comb {
        o_cnt = r_cnt;
    }
}
```

ドキュメンテーションコメント

末尾カンマ

ビット幅記法

クロック・リセットの省略

代入演算子の統合

Verylの特徴（2 / 3）

- クロックとリセット
  - SystemVerilogの生成時に極性・同期非同期を指定可能
    - ASICとFPGAでリセットが異なるようなケースを扱える

```
// Veryl code
module ModuleA (
    i_clk_a: input clock           ,
    i_clk_b: input clock_negedge   ,
    i_rst_a: input reset           ,
    i_rst_b: input reset_async_high,
) {
    always_ff (i_clk_a, i_rst_a) {
        if_reset {
        }
    }
    always_ff (i_clk_b, i_rst_b) {
        if_reset {
        }
    }
}
```

clock_type=posedge
reset_type=async_low

```
// Generated SystemVerilog code
always_ff @ (posedge i_clk_a or negedge i_rst_a) begin
    if (!i_rst_a) begin
    end
end
always_ff @ (negedge i_clk_b or posedge i_rst_b) begin
    if (i_rst_b) begin
    end
end
```

clock_type=negedge
reset_type=sync_high

```
// Generated SystemVerilog code
always_ff @ (negedge i_clk_a) begin
    if (i_rst_a) begin
    end
end
always_ff @ (negedge i_clk_b or posedge i_rst_b) begin
    if (i_rst_b) begin
    end
end
```

Verylの特徴（3 / 3）

- ジェネリクス
  - パラメータオーバーライドより強力なコード生成

```
module SramQueue::<T> {
    inst u_sram: T;

    // queue logic
}

module Test {
    // Instantiate a SramQueue by SramVendorA
    inst u0_queue: SramQueue::<SramVendorA>();

    // Instantiate a SramQueue by SramVendorB
    inst u1_queue: SramQueue::<SramVendorB>();
}
```

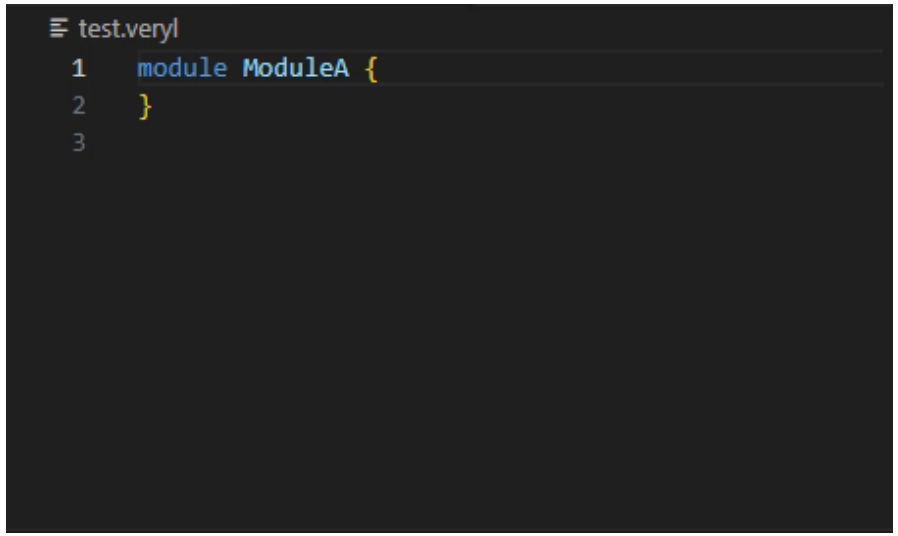
モジュール内でインスタンスする
モジュール名をここで指定できる

Verylの開発環境

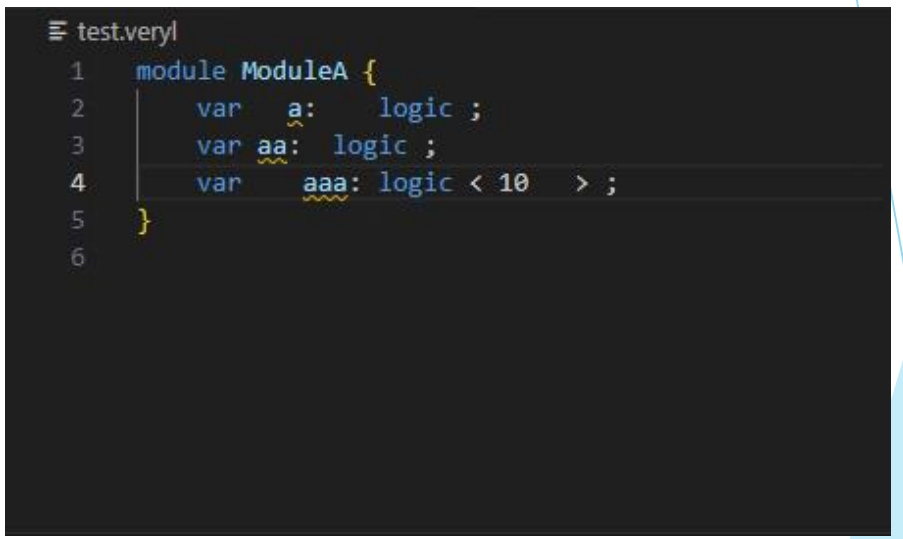
- 言語標準で提供されるもの
  - 組み込みユニットテスト
    - Verilator/VCSなどサポート
  - パッケージマネージャ
    - Gitリポジトリからの依存関係解決
  - ドキュメント生成
  - Language Server
    - VSCode/Vim/EmacsなどLSP対応エディタと連携

Language Serverの動作例

- リアルタイム診断



- フォーマット



まとめ

- SystemVerilogに代わる新しいHDLを開発中
  - https://veryl-lang.org
  - https://github.com/veryl-lang/veryl
  - 日本語ドキュメントもあり

- お願い
  - Verylのコードを書いたらGitHubにコミットしてほしい
    - GitHubのシンタックスハイライト対応条件が2000ファイル以上
    - 現在140ファイルくらい