『Verylで作るCPU』を読んでいる

Kenta Arai

自己紹介

- 名前：Kenta Arai
  - Twitter: @isKenta14
  - Qiita: Kenta11
- 仕事：組込みソフトウェア開発
- 趣味で作ったもの
  - micro-alpha
    - 簡素なマイクロプログラム制御方式CPU
    - Xilinx FPGA 上で実際に動かせました
  - simple_uart：UART の SystemVerilog 実装

これまでにやった自作CPU

- MICRO-1[1]：マイクロプログラム制御方式の簡素なCPU
  - 制御部
    - 制御記憶：1語40ビット、最大4K語の容量
    - アドレス長：12ビット
  - 被制御部
    - 主記憶：1語16ビット、最大64K語の容量
    - 汎用レジスタ：16ビット×8
- SystemVerilog で実装
- Xilinx FPGA 上で動作させることができた
  - PC と FPGA 間を UART で接続
  - 逆ポーランド記法の電卓が動作した

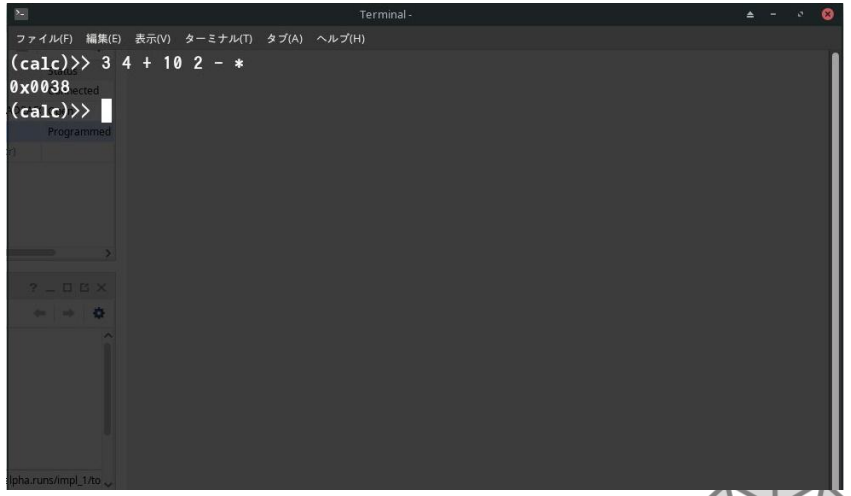


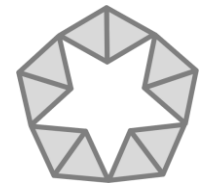
図. デモプログラムが動作する様子

[1] 馬場敬信：ソフトウェア講座（２３）マイクロプログラミング, 昭晃堂, pp. 31-107, 1985

SystemVerilog で書くのは結構ツライ

- Verilog-HDL から続く古典的な構文スタイル
- 手頃なフォーマッタや静的解析ツールが少ない

→Veryl が解決してくれるかも？



Veryl

- SystemVerilog にトランスパイルされる HDL
  - JavaScript にとっての TypeScript みたいな立ち位置
- 文法がモダンで書きやすい
  - 公式も「学習の容易さ、設計プロセスの信頼性と効率の向上、およびコードの記述の容易さが実現されます。」と言っている
- トランスパイラだけでなく、静的解析器やコードフォーマッタ等のツールチェインも公開されている
  - verylup で一通りインストールできる
  - 文法だけでなく、この辺のエコシステムも含めて Rust の影響を受けているっぽい

Veryl の書きやすいところ

- コードブロック
  - SystemVerilog だと begin, end のところ、Veryl では {}
- 複数 bit の信号の宣言

SystemVerilog での宣言

```
logic [31:0] clock_counts;
```

Veryl での宣言

```
var clock_counts : logic<32>;
```

- 列挙型の定義と名前空間

SystemVerilog での列挙型定義

```
typedef enum logic [1:0] {
    STATE_WAIT         = 2'h0,
    STATE_RECEIVE_BITS = 2'h1,
    STATE_WRITE_WORD   = 2'h2
} state_t;
// state <= STATE_WAIT; // 列挙子の参照方法
```

Veryl での列挙型定義

```
enum state_t: logic<2> {
    WAIT,
    RECEIVE_BITS,
    WRITE_WORD,
}
// state = state_t::WAIT; // 列挙子の参照方法
```

Veryl で感心したアイデア

- リセットの性質（極性と同期）をコードから分離できる
  - SystemVerilog では、文法上はリセットはあくまで信号の1つ
  - Veryl ではリセット型を用意し、リセット専用の構文を備えており
    リセットの性質は設定ファイルから変更できる

SystemVerilog でのリセット定義

```
module receiver_axis (
    input rst,
    // 省略
);
```

Veryl でのリセット定義

```
module receiver_axis (
    rst : input reset,
    // 省略
)
```

Veryl.toml に追記する内容

```
[build]
reset_type = "sync_high"
```

Veryl でのリセット参照

```
always_ff (clk) {
    if_reset {
        // リセット時の記述
    } else {
        // 省略
    }
}
```

Veryl で自作 CPU をしたい

- そんなことを考えていたら、すでに素敵なテキストがあった Verylで作るCPU
- Veryl で RISC-V を実装する内容
- 執筆中のようで、目次を読むと、ゆくゆくは Linux を動かす構想もあるみたい

『Verylで作るCPU』を読んでいる

- テキストを読み進めながら RISC-V コアを実装中
- 実装の進捗は Zenn でスクラップにしてます
- テキストで直したら良さそうなところは PR 投げてます

- みなさんも一緒に読みませんか？