第3部　Appendix1

Pythonで回路を記述する

高位合成ツールPolyphony

片岡 啓明，鈴木 量三朗

Polyphonyはオープン・ソースの高位合成ツールです．Pythonで書いた処理を論理合成可能なVerilog HDLに変換します．Polyphonyは，ソフトウェア技術者向けの高位合成ツールと言えます．

● インストール手順

Pythonが使える環境で次のコマンドを実行します．

```
$ pip3 install polyphony
```

● Polyphonyを使った作業の流れ

PolyphonyによるFPGA開発の流れは次の通りです．

1. Pythonでアルゴリズムを書く
2. Pythonコードで検証する
3. 高位合成（ここでPolyphonyを使用）
4. 出力後のHDLで検証
5. 論理合成，配置配線（ベンダ・ツールの使用を想定）

リスト1　フィボナッチ数をPolyphonyで高位合成する

```
def fibonacchi(x):
    a, b = 0, 1
    for i in range(x):
        a, b = b, a + b

    return b
```

（a）Pythonコード

Pythonを使うということ以外は他の高位合成ツールと，特に際立った違いはありません．

● Pythonで書いたコードが回路記述に変わる

リスト1（a）にPythonで書かれた関数を示します．これをPolyphonyで合成して得たVerilog HDLのコードを**リスト1**（b）に示します．

このようにPythonのコードでFPGAの回路を書けます．コード上には，ハードウェア的な概念（信号やタイミング）は出てきません．それらを意識せずに，アルゴリズム記述に専念できます．

逆の見方をすれば，クロック単位での細かい調整をするような回路は書けないということです．現在開発中の次期バージョン（timed）ではクロックを意識した処理にも対応します．興味のある方はGitHubのPolyphonyのリポジトリ(1)を参照してください．

複雑な状態遷移が必要な処理など，HDLで書くのが面倒で単純なミスをしやすいようなところでは効果的に使えるのではないでしょうか．

処理速度や省リソースなどの性能については，HDLを使う方が良いものができるでしょう．

早く動くものを実装して結果を確認することが大事な場合もあります．生産性が高いという点において，Pythonを使うことはメリットとなるでしょう．

```
module func01
  (
    input wire clk,
    input wire rst,
    input wire func01_ready,
    input wire func01_accept,
    output reg func01_valid,
    input wire signed [31:0] func01_in_x,
    output reg signed [31:0] func01_out_0
  );

  //localparams
  localparam func01_b1_INIT = 0;
  localparam func01_b1_S0 = 1;
  localparam func01_b1_FINISH = 2;

  //signals:
  reg        [1:0] func01_state;
  wire signed [31:0] inner_result1;
  reg signed [31:0] x;
  //combinations:
  assign inner_result1 = (x + 1);

  always @(posedge clk) begin
    if (rst) begin
      func01_out_0 <= 0;
      x <= 0;
      func01_state <= func01_b1_INIT;
    end else begin //if (rst)
      case(func01_state)
      func01_b1_INIT: begin
        func01_valid <= 0;
        if (func01_ready == 1) begin
          x <= func01_in_x;
          func01_state <= func01_b1_FINISH;
        end
      end
      func01_b1_FINISH: begin
        func01_valid <= 1;
        if (func01_accept == 1) begin
          func01_state <= func01_b1_INIT;
        end
        /* inner_result1 <= (x + 1); */
        func01_out_0 <= inner_result1;
      end
      endcase
    end
  end
```

（b）（a）をPolyphonyで合成したVerilog HDLコード