Библиотека кода для литературного RTL

Васил Дядов

Среда 18 июля 2018 г.

Содержание

1 Инициализация OCaml сессии **1**

2 Установка принтера для сигналов **1**

3 Генератор временных диаграмм **1**

3.1 Тест генератора временных диаграмм 4

1 Инициализация OCaml сессии

```
#use "topfind";;
#require "hardcaml";;
#require "ppx_deriving_hardcaml";;
#require "ppx_hardcaml";;
```

2 Установка принтера для сигналов

```
let print_signal fmt signal =
  Format.fprintf fmt "%s\n"
    (HardCaml.Signal.Comb.to_string signal);;
#install_printer print_signal;;
```

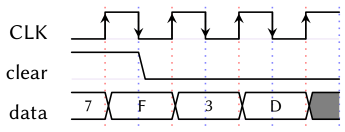
3 Генератор временных диаграмм

```
module TimingDiagram(S : HardCaml.Comb.S) = struct
  let header =
    "\\definecolor{lightlightviolet}" ^
      "{rgb}{0.90,0.85,0.95}\n" ^
        "\\begin{tikztimingtable}[%%\n" ^
          "timing/name/.style=" ^
            "{font=\\sffamily\\scriptsize},\n" ^
              "semithick,  timing/dslope=0.1,\n" ^
                "timing/.style={x=5ex,y=2ex},\n" ^
                  "timing/coldist=1ex, x=5ex, \n" ^
                    "timing/rowdist=3ex,\n" ^
                      "timing/c/dual arrows,\n" ^
                        "timing/c/arrow tip=stealth]\n"

  let footer ~periods =
    "\\extracode\n" ^
      "\\begin{pgfonlayer}{background}\n" ^
        "\\begin{scope}[semitransparent,semithick]\n" ^
          "\\horlines[lightlightviolet]{}\n" ^
            "\\vertlines[red,dotted]{0.5,1.5 ,...," ^
              (string_of_int periods) ^ ".0}\n" ^
                "\\vertlines[blue,dotted]{1.0,2.0 ,...," ^
                  (string_of_int periods) ^ ".5}\n" ^
                    "\\end{scope}\n" ^
                      "\\end{pgfonlayer}\n"^
                        "\\end{tikztimingtable}\n"

  type format = Bin | Dec | Hex
  type edges = Rising | Falling | Both
  type waveform = {name:string;
                   fmt: format;
                   edge: edges;
                   data: S.t list}

  let signal_to_diagram {fmt; edge; data; _} =
    let width = S.width (List.hd data) in
    begin
      match width with
      | 1 -> List.map
               (fun v -> if v = S.vdd
                         then "h"
                         else "l")
               data
      | _ -> List.map
               (fun v ->
                 let v = S.to_int v in
                 match fmt with
                 | Bin -> Printf.sprintf "d{%d}" v
                 | Dec -> Printf.sprintf "d{%d}" v
                 | Hex -> Printf.sprintf "d{%X}" v)
               data
    end
    |> List.mapi
         (fun i x ->
           match edge with
           | Both -> "1" ^ x
           | Rising -> "2" ^ x
           | Falling -> if i = 0
                        then "1" ^ x
                        else "2" ^ x)
    |> (fun l -> match edge with
                 | Falling -> l @ ["1u{}"]
                 | _ -> l)
    |> String.concat " "

  let out_signals ?(clock_name = "CLK") ~signals =
    let periods = match List.hd signals with
      | {data; edge; _} ->
         (List.length data) *
           (match edge with Both -> 1 | _ -> 2)
    in
    clock_name ^ "& " ^
      (string_of_int periods) ^ "{c}\\\\n" ^
        begin
          signals
          |> List.map
               (fun ({name; fmt; edge; data} as signal) ->
                 name ^ "& " ^
                   signal_to_diagram signal ^ "\\\\n")
          |> String.concat ""
        end
        ^ footer ~periods:(periods / 2)


  let gen_latex ?(clock_name = "CLK") ~signals  =
    header ^ out_signals ~clock_name ~signals

  let update_signals ~signals newdata =
    List.map
      (fun ({name; data; _} as s) ->
        let new_data =
          List.find_opt
            (fun (n,d) -> n = name)
            newdata
        in
        match new_data with
```

```
        | None -> s
        | Some (_, d) -> {s with data = data @ d})
      signals
end
```

3.1 Тест генератора временных диаграмм

```
let module B = HardCaml.Bits.Comb.IntbitsList in
    let module TD = TimingDiagram(B) in
    Printf.printf "%s\n" @@
      TD.gen_latex
        [
          {TD.name = "clear";
           fmt  = TD.Hex;
           edge = Rising;
           data = (List.map
                     B.constb
                     ["1"; "0"; "0"; "0"] )};
          {TD.name = "data";
           fmt  = TD.Hex;
           edge = Falling;
           data = (List.map
                     B.constb
                     ["0111"; "1111"; "0011"; "1101"] )}]
```


CLK
clear
data
7
F
3
D