Литературный RTL

Васил Дядов

Среда 18 июля 2018 г.

Содержание

1 Настройка Merlin для проекта

Добавляем все пакеты, что установленны в текущей конфигурации Opam, в файл `.merlin`.

```
opam list | grep -v '#' | awk '{print "PKG ", $1}'
```

2 Кодогенераторы на Hardcaml

2.1 Счётчик по-модулю

```
module ModCounter = struct
  open HardCaml
  open Signal.Comb
  open Signal.Seq

  let gen ~clr ~cntr_modulo ~cntr_width =
    let zero = consti cntr_width 0 in
```

```
    let feedback_fn d =
          mux2 (d >=: (cntr_modulo -:. 1))
            zero
            (d +:. 1)
    in
    reg_fb r_sync (const "1'b1") cntr_width
      (fun d -> mux2 clr zero (feedback_fn d))
end
```

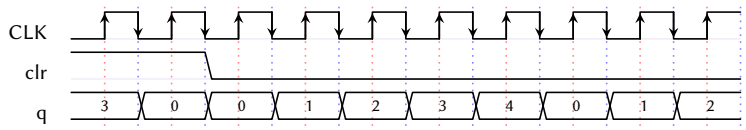
2.1.1 Тестирование

1. Тестовое окружение

```
module ModCounterTest = struct
  open HardCaml
  open Signal.Comb

  module In = struct
    type 'a t = {
        clr: 'a;
        modulo: 'a [@ bits 4]
      } [@@deriving hardcaml]
  end

  module Out = struct
    type 'a t = {
        q: 'a [@bits 4]
      } [@@deriving hardcaml]
  end

  let gen_if i =
    let cntr_width = snd In.(t.modulo) in
    let q = In.(ModCounter.gen
                  ~clr:i.clr
                  ~cntr_modulo:i.modulo
                  ~cntr_width)
    in
    Out.({q})

  module B = Bits.Comb.IntbitsList
  module Builder = Interface.Gen(B)(In)(Out)

  let circuit, sim, i, o, _ =
    Builder.make "ModCounter" gen_if

  module S=Cyclesim.Api
end
```

2. Запуск теста

```
let _ =
  let open ModCounterTest in
  let open In in
  let open Out in
  let module TD = TimingDiagram(B) in
  let signals =
    ref
      [{TD.name = "clr";
        fmt = Hex;
        edge = Rising;
        data = []};
       {TD.name = "q";
        fmt = Hex;
        edge = Rising;
        data = []}] in
  let step () =
    S.cycle sim;
    signals := TD.update_signals
                 !signals
                 [ ("clr", [!(i.clr)]);
                   ("q", [!(o.q)])]
  in
  S.reset sim;
  i.modulo := B.consti 4 5;
  i.clr := B.vdd;
  step (); step ();
  i.clr := B.gnd;
  for _ = 0 to 7 do
    step ();
  done;
  TD.gen_latex !signals
  |> print_string
```



3. Тест экспорта верилога

```
let _ =
  HardCaml.Rtl.Verilog.write
```

```
    print_string
    ModCounterTest.circuit
```

```
module ModCounter (
    clear,
    clock,
    modulo,
    clr,
    q
);

    input clear;
    input clock;
    input [3:0] modulo;
    input clr;
    output [3:0] q;

    /* signal declarations */
    wire _40 = 1'b1;
    wire [3:0] _42 = 4'b0000;
    wire vdd = 1'b1;
    wire [3:0] _43 = 4'b0000;
    wire [3:0] _39 = 4'b0000;
    wire [3:0] _45 = 4'b0001;
    wire [3:0] _46;
    wire [3:0] _47 = 4'b0001;
    wire [3:0] _48;
    wire _49;
    wire _50;
    wire [3:0] _51;
    wire [3:0] _52;
    wire [3:0] _41;
    reg [3:0] _44;

    /* logic */
    assign _46 = _44 + _45;
    assign _48 = modulo - _47;
    assign _49 = _44 < _48;
    assign _50 = ~ _49;
    assign _51 = _50 ? _39 : _46;
    assign _52 = clr ? _39 : _51;
    assign _41 = _52;
    always @(posedge clock) begin
        if (clear)
            _44 <= _42;
        else
            if (_40)
```

```
                    _44 <= _41;
      end

      /* aliases */

      /* output assignments */
      assign q = _44;

  endmodule
  - : unit = ()
```

2.2 Разное тестирование

```
module FreqDivider = struct
  open HardCaml
  open Signal.Comb
  open Signal.Seq

  let gen ~clr ~div_by ~cntr_width =
    let mod_counter = ModCounter.gen
                        ~clr
                        ~cntr_modulo:div_by
                        ~cntr_width
    in
    ()
end
```

7

```
  (* x 10)
```

```
  let () = Printf.printf "\nlet max_value = %d;;\n" max_value;;
```

3 Код verilog

3.1 Общие параметры проекта

3.2 Счетчик по модулю

- Период счетчика и ширина регистра

Параметр	Значение
Частота	100000000
Скорость UART	115200

Таблица 1: Таблица параметров

Параметр	Значение
Модуль	868
Ширина регистра	10

Таблица 2: Параметры счётчика

10

При частотах
Test call:

```
let max_value = 70;;
```