Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа компьютерных технологий и информационных систем

Отчёт по лабораторной работе № 2_3

Дисциплина: Автоматизация проектирования дискретных устройств (на английском языке).

Выполнил студент гр. 5130901/10101 _____ Д.Л. Симоновский
(подпись)

Руководитель _____ А.А. Федотов
(подпись)

“07” февраля 2024 г.

Санкт-Петербург
2024

Оглавление

1. Список иллюстраций:

2. Задача:

На языке Verilog разработать устройство по следующей схеме:

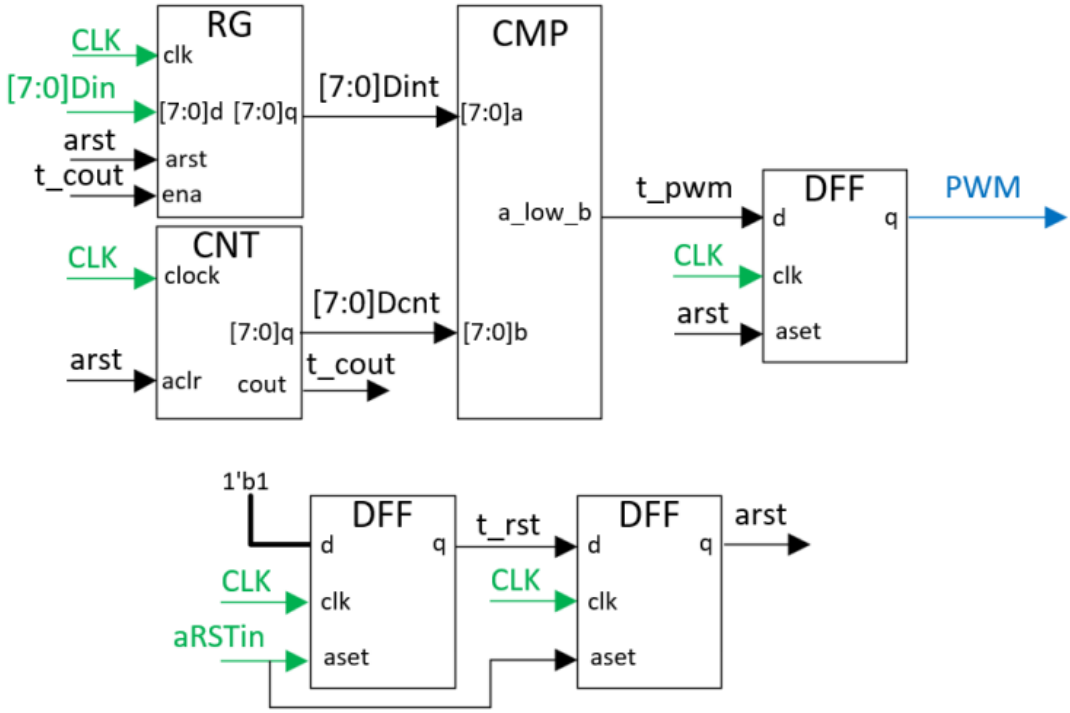

Рис. 2.1. Схема разрабатываемого устройства.

Выводы:

1) ***Входы:***
   a) *CLK* – тактовый сигнал.
   b) *aRSTin* – вход асинхронного сброса (активный уровень для сброса – 1).
   c) *[7:0] Din* – вход данных для управления ШИМ.
2) ***Выходы:***
   a) *PWM* – выход ШИМ.

Модули:

1) ***CNT*** – счетчик, создаваемый с помощью IP модуля LPM_COUNTER:
   a) Разрядность: 8 бит.
   b) Двоичный счетчик на сложение.
   c) Выход переноса (carry_out).
   d) Вход асинхронного сброса (clear).
2) ***CMP*** – модуль сравнения, создаваемый с помощью IP модуля LPM_COMPARE:
   a) Два входа по 8 бит.
   b) a < b
   c) Без знаковый.
   d) Без конвейеризации.
3) ***RG*** – регистр, описываемый на Verilog в файле верхнего уровня:
   a) arst – вход асинхронного сброса (активный уровень – 1) устанавливает 8`d128.
4) ***DFF*** – триггеры, описываемые на Verilog в файле верхнего уровня:
   a) aset – вход асинхронно устанавливает триггер в 1.

3. Решение:

Создадим модуль верхнего уровня на языке Verilog. Его код будет выглядеть следующим образом:

```
module lab2_3 (
  input           CLK,
  input           aRSTin,
  input      [7:0] Din,
  output reg      PWM
);

  // DFF_arst
  reg t_rst = 1'b1, arst = 1'b1;

  always @(posedge CLK or posedge aRSTin)
    if (aRSTin) begin
      t_rst <= 1'b1;
      arst  <= 1'b1;
    end else begin
      t_rst <= 1'b0;
      arst  <= t_rst;
    end

  // RG
  reg [7:0] Dint = 8'd128;
  wire      t_cout;

  always @(posedge CLK or posedge arst)
    if (arst) Dint <= 8'd128;
    else if (t_cout) Dint <= Din;

  // CNT
  wire [7:0] Dcnt;

  cnt cnt_inst (
    .aclr (arst),
    .clock(CLK),
    .cout (t_cout),
    .q    (Dcnt)
  );

  // CMP
  wire t_pwm;

  cmp cmp_inst (
    .dataa(Dint),
    .datab(Dcnt),
    .alb  (t_pwm)
  );

  // DFF_out
  always @(posedge CLK or posedge arst)
    if (arst) PWM <= 1'b1;
    else PWM <= t_pwm;

endmodule
```

Рис. 3.1. Код модуля верхнего уровня lab2_3.

Первый блок always описывает два триггера DFF для сигнала асинхронного сброса. Поскольку сброс асинхронный, always реагирует не только на фронт clk, но и на фронт сигнала сброса.

Блок RG реализует регистр на выходе сравнителя. Поскольку регистр должен иметь асинхронный сигнал сброса, поэтому always реагирует не только на фронт clk, но и на фронт сигнала сброса.

Блок CNT передает параметры IP-модулю счетчика, полученного из IP-модулей Quartus Prime.

Блок CMP, аналогично CNT, передает параметры IP-модулю сравнителя.

Блок DFF_out – регистр на выходе сравнителя, реагирует как на clk, так и на сигнал асинхронного сброса, устанавливая значение в 1.

Проверим корректность разработанной схемы, используя RTL Viewer:

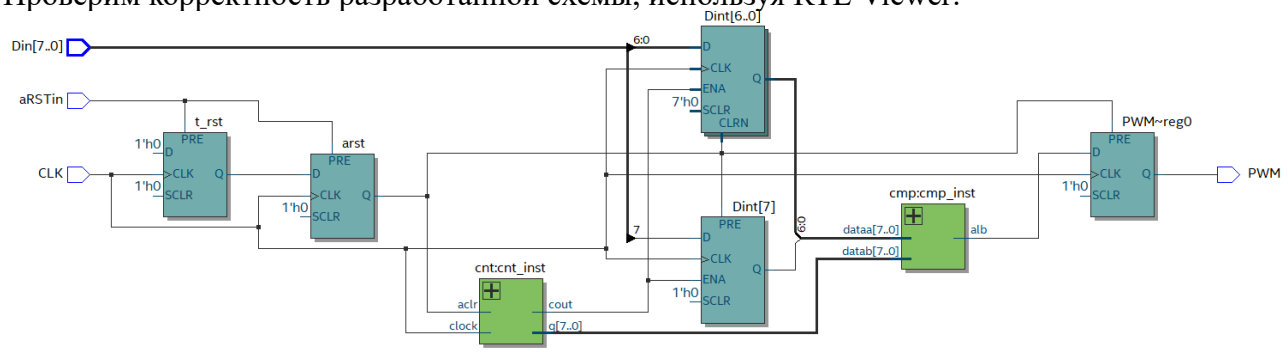

Рис. 3.2. RTL Viewer модуля lab2_3.

Интересным тут является разделение блока регистров Dint на два. Один блок для регистров с 6 по 0, а второй отдельно для седьмого. Если посмотреть на то, что записывается в Dint, видно, что вход D инвертирован. Это свзяно с тем, что при сигнале сброса в Dint должно записываться значение 128, именно поэтому регистр хранит инвертированное значение, чтоб при сбросе автоматически устанавливалось значение 128. В остальном схема соответствует ожиданиям.

Далее разработаем тесты первого класса для разработанного модуля:

```
`timescale 1ns / 1ns
module tb_lab2_3;
  parameter PERIOD = 10;
  reg         CLK = 0;
  reg         aRSTin = 0;
  reg  [7:0] Din = 0;
  wire        PWM;


  initial begin
    forever #(PERIOD / 2) CLK = ~CLK;
  end

  lab2_3 u_lab2_3 (
    .CLK   (CLK),
    .aRSTin(aRSTin),
    .Din   (Din[7:0]),
    .PWM(PWM)
  );

  initial begin
     Din = 8'd64;
     #(PERIOD * (256 + 128));
     Din = 8'd200;
     #(PERIOD * (256 + 128 + 10));
    $stop;
  end

endmodule
```

Рис. 3.3. Код теста первого класса для модуля lab2_3.

В начале устанавливаем значение 64, однако в регистр модуля оно запишется только когда счетчик досчитает до максимума и даст сигнал carry_out. Поэтому подождем 256 тактов. После этого ждем еще 128 тактов и меняем значение на 200, однако запись произойдет еще через 128 тактов. Смотрим на результат 256 + 10 тактов, чтоб увидеть, как выход сбросится опять до 0.
Запустим тестовый файл средствами Quartus Prime.
Результат запуска выглядит следующим образом:

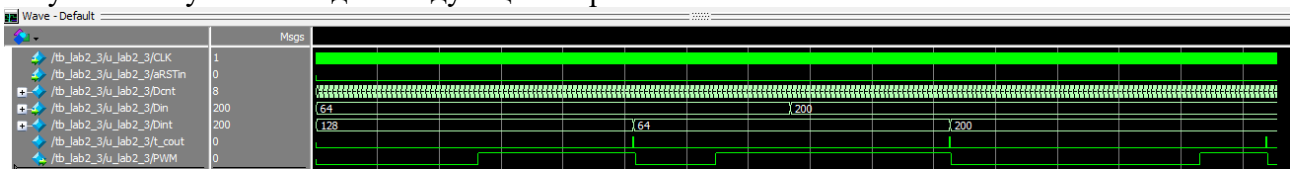

Рис. 3.4. Результат моделирования.

Как видно, получившаяся waveform полностью соответствует ожиданиям и техническому заданию, что свидетельствует о корректно разработанном устройстве.

4. Вывод:

В ходе лабораторной работы было разработано устройство на языке Verilog в соответствии с схемой. Получившееся устройство полностью соответствует техническому заданию. Разработанное устройство – обычный ШИМ (широтно-импульсный модулятор), что видно по выходу PWM. Оно принимает на вход Din величину ШИМ, которая записывается по сигналу t_cout, чтоб не происходило каких-либо помех при смене этого значения в процессе вывода сигнала.
В процессе разработки использовались IP-модули из библиотеки Quartus Prime, стоит отметить, что они сильно ускорили процесс и помогли избавиться от написания стандартных модулей, дав возможность сосредоточиться на основном задании.