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

Отчёт по лабораторной работе № 2_2

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:

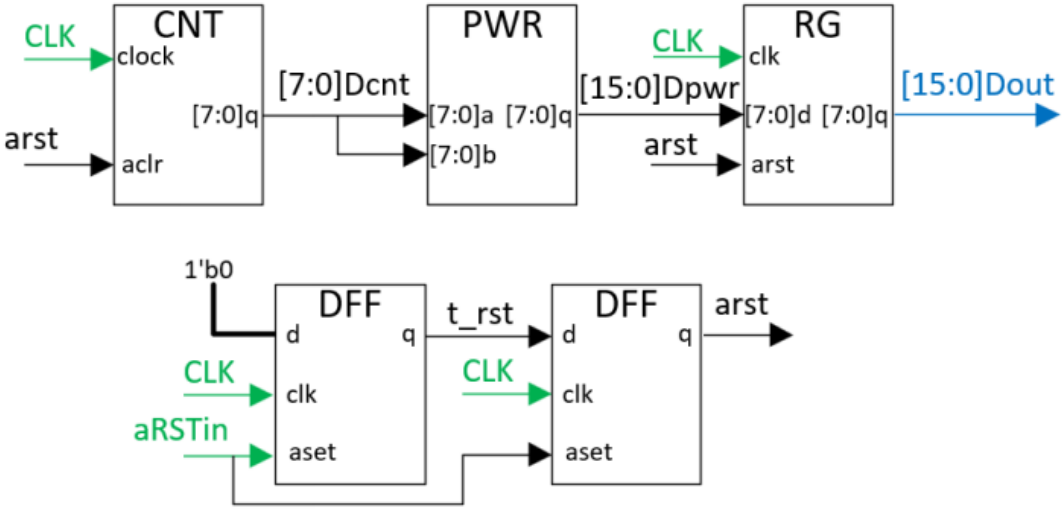

Рис. 2.1. Схема разрабатываемого устройства.

Выводы:

1) ***Входы:***
   a) *CLK* – тактовый сигнал.
   b) *aRSTin* – вход асинхронного сброса (активный уровень для сброса – 1).
2) ***Выходы:***
   a) *[15:0] Dout* – выход.

Модули:

1) ***CNT*** – счетчик, создаваемый с помощью IP модуля LPM_COUNTER:
   a) Разрядность: 8 бит.
   b) Двоичный счетчик на сложение.
   c) Вход асинхронного сброса (clear).
2) ***PWR*** – модуль возведения в степень 2, создаваемый с помощью IP модуля LPM_MULT:
   a) Два входа по 8 бит.
   b) Без знаковый.
   c) Без конвейеризации.
3) ***RG*** – регистр, описываемый на Verilog в файле верхнего уровня:
   a) arst – вход асинхронного сброса (активный уровень – 1).
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_2 (
  input             CLK,
  input             aRSTin,
  output reg [15:0] Dout
);

  // DFF
  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

  // CNT
  wire [7:0] Dcnt;

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

  // PWR
  wire [15:0] Dpwr;

  pwr pwr_inst (
    .dataa (Dcnt),
    .datab (Dcnt),
    .result(Dpwr)
  );

  // RG
  always @(posedge CLK or posedge arst)
    if (arst) Dout <= 1'b0;
    else Dout <= Dpwr;

endmodule

```

Рис. 3.1. Код модуля верхнего уровня lab2_2.

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 CNT передает параметры IP-модулю счетчика, полученного из IP-модулей Quartus Prime.
Блок PWR, аналогично CNT передает модули IP-модулю умножителя.
Блок RG реализует регистр на выходе умножителя. Поскольку регистр должен иметь асинхронный сигнал сброса, поэтому always реагирует не только на фронт clk, но и на фронт сигнала сброса.
Проверим корректность разработанной схемы, используя RTL Viewer:

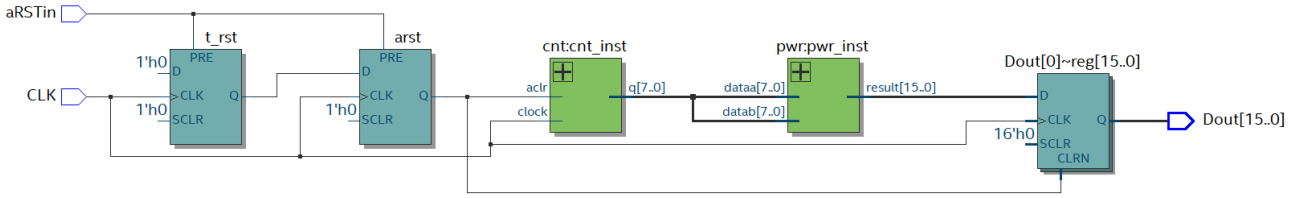

Рис. 3.2. RTL Viewer модуля lab2_2.

Как мы видим, разработанное устроство полностью совпадает с примером.
Далее разработаем тесты первого класса для разработанного модуля:

```
1  `timescale 1ns / 1ns
2  module tb_lab2_2;
3
4    parameter PERIOD = 10;
5    reg         CLK = 0;
6    reg         aRSTin = 0;
7    wire [15:0] Dout;
8
9
10   initial begin
11     forever #(PERIOD / 2) CLK = ~CLK;
12   end
13
14   lab2_2 lab2_2_inst (
15     .CLK   (CLK),
16     .aRSTin(aRSTin),
17     .Dout  (Dout[15:0])
18   );
19
20   initial begin
21     #(PERIOD * 256 * 2);
22     $stop;
23   end
24
25 endmodule
26
```

Рис. 3.3. Код теста первого класса для модуля lab2_2.

Поскольку на один полный цикл счетчика (от 0 до 255 (т.к. раззрядность счетчика 8)) необходимо 256 тактов, для двух полных циклов необходимо $PERIOD * 256 * 2$ времени. Именно столько мы и ждем перед остановкой.

Для запуска тестового модуля созададим следующий .do файл:

```
1  vlib work
2  vlog cnt.v
3  vlog lab2_2.v
4  vlog pwr.v
5  vlog tb_lab2_2.v
6  vsim -L lpm_ver work.tb_lab2_2
7  add wave /tb_lab2_2/lab2_2_inst/CLK
8  add wave /tb_lab2_2/lab2_2_inst/aRSTin
9  add wave -format Analog-Step -height 128 -max 255.0 -radix unsigned /tb_lab2_2/lab2_2_inst/Dcnt
10 add wave -format Analog-Step -height 512 -max 65025.0 -radix unsigned /tb_lab2_2/lab2_2_inst/Dout
11 run -all
```

Рис. 3.4. .do файл для запуска тестового модуля.

В строках с 1 по 5 мы создаем проект и выполняем компиляцию всех требуемых модулей, далее с 6 по 10 мы запускам симуляцию и добавлем все необходимо на waveform, после чего мы запускаем проект.

Результат запуска выглядит следующим образом:

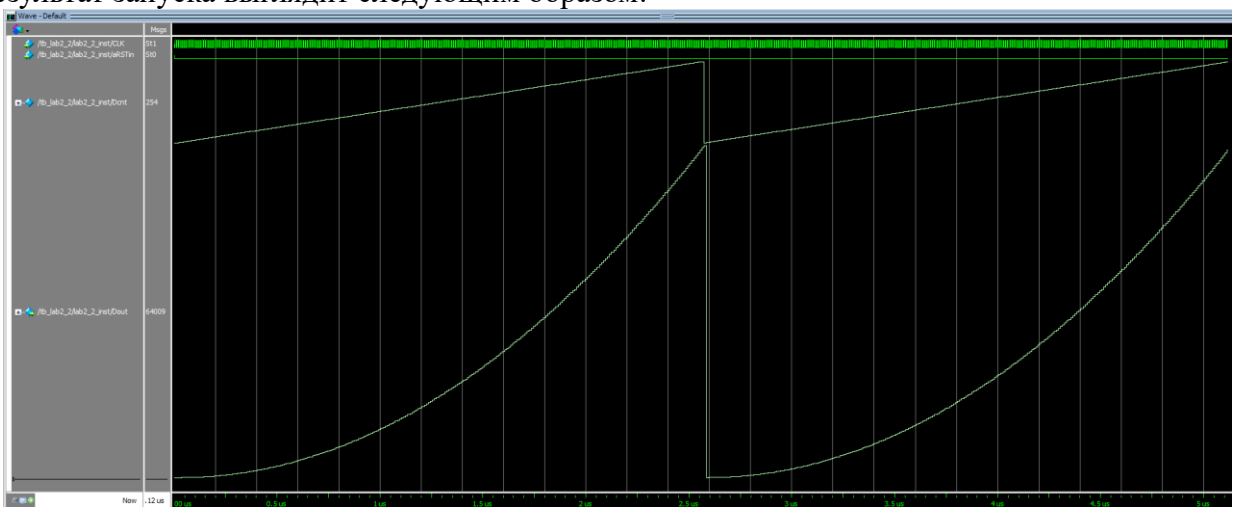

Рис. 3.5. Результат моделирования.

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.
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.