МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КПІ ІМ. ІГОРЯ СІКОРСЬКОГО
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ

КАФЕДРА КОНСТРУЮВАННЯ
ЕЛЕКТРОННО-ОБЧИСЛЮВАЛЬНОЇ АПАРАТУРИ

РОЗРАХУНКОВА ГРАФІЧНА РОБОТА

з кредитного модуля «Обчислювальні та мікропроцесорні засоби в радіоелектронній апаратурі - 1»

на тему «Підключення генератора ШІМ-сигналу»

Керівник: Антонюк Олександр Ігорович
_____
Захищено з оцінкою _____
Дата «____»_____2022 р.

Виконав:
студент гр. ДК-91
Махно Віктор
Залікова книжка № ДК

Київ – 2022

ЗМІСТ

ВСТУП

Метою роботи є створення та підключення генератора ШІМ-сигналу до MIPS ядра описані мовою Verilog та тестування правильності роботи створеного пристрою. Особливість даного об’єднання можливість використання ядра з описаними інструкціями для створення потрібного ШІМ-сигналу.

Прототип буде складатись з декількох функціональних блоків, а саме:

1) Багатотактне ядро MIPS.

2) Лічильник.

3) Генератор ШІМ-сигналу.

Завданням проекту є об’єднання багатотактного ядра з генератором ШІМ вхідним сигналом, що передає цифровий 8-ми розрядний двійковий код рівня заповненості з Data Memory, вхідні сигнали початку роботи, ресету та завантаження значення рівня заповненості. Вихідним сигналом повинен бути ШІМ-сигнал, що має постійну частоту та зміну скважність.

Для створення прототипу буде використане середовище ModelSim. Дане ПЗ надає можливість швидкої перевірки правильності роботи коду та виведення на екран сигналів, що демонструють роботу прототипу.

Реалізація генератора ШІМ-сигналу

Для реалізації генератора ШІМ-сигналу було створено 8-ми розрядний лічильник, що рахує вгору. Вихідний сигнал лічильника подається на вхід ШІМ генератора для порівняння з вхідним значення скважності, таким чином відтворюючи необхідну скважність та частоту вихідного ШІМ-сигналу.

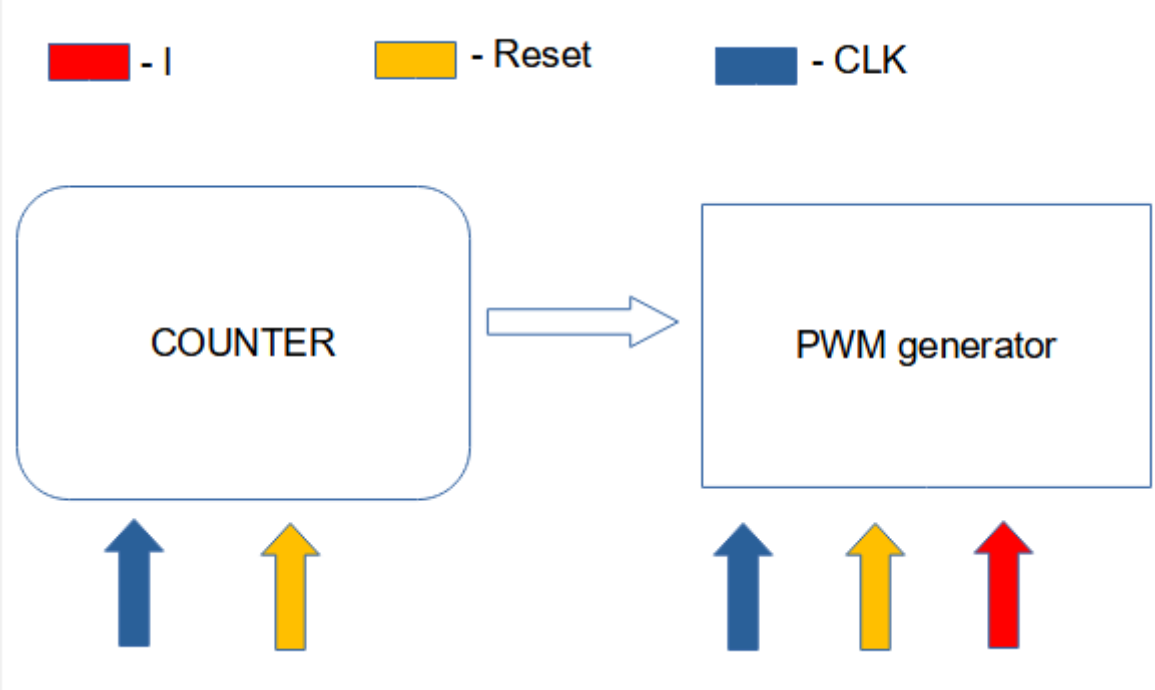

Рис. 1 Структурна схема генератора ШІМ-сигналу

Для тестування частин пристрою використаємо середовище ModelSim в якому будемо запускати ТБ файл з заданими сигналами. Завдяки ТБ файлу ми можемо задати значення вхідних сигналів та слідкувати за реакцією вихідних у вигляді графіку, також ПЗ дає можливість ставити курсори та переконатися у вірності та відповідності таймінгам встановлення сигналів.

РОЗРОБКА ФАЙЛІВ ОПИСУ СКЛАДОВИХ ЧАСТИН ТА ФАЙЛІВ ДЛЯ ЇХ ТЕСТУВАННЯ

Лічильник

Я створив 8-ми розрядний лічильний для контролю частоти ШІМ сигналу на виході контролера. Вихід лічильника **counter** з кожним переднім фронтом **clk** збільшується на 1, таким чином діапазон лічильника складає від 0 до 255, що забезпечує постійну частоту **PWM_out** . При надходженні заднього фронту сигналу **rst_n** значення на виході встановиться 0.

Лістинг коду на мові Verilog опису генератора та ТБ представлені у додатку 1 і 2.

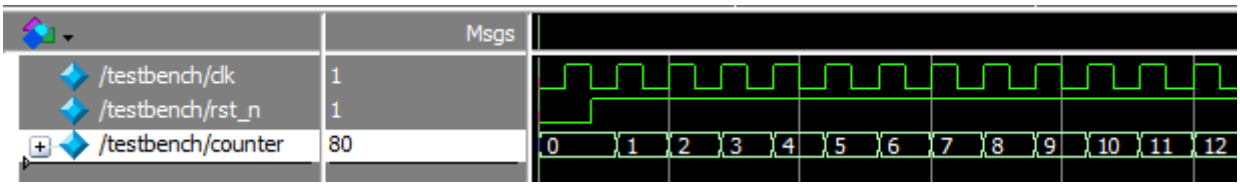

Рис. 2.1 Зміна вихідного сигналу по кожному такту CLK

Для модуляції роботи лічильника я написав ТБ файл в якому задав вхідний сигнал **clk,** котрий зміню кожні 10с або PERIOD/2, де параметр PERIOD = 20нс. Сигнал **rst_n** в нульовий момент часу я задаю рівним 0 та по наступному задньому фронту **clk** переводжу в 1, що відповідає RESET в один такт. Провівши тест я отримав збільшення вихідного сигналу на одиницю по кожному такту тактового генератора (CLK), що показано на Рис.2.1.

Генератор ШІМ сигналу

Для реалізації вихідного сигналу **PWM_out** було вирішено розробити генератор ШІМ сигналу.

Лістинг коду на мові Verilog опису генератора та ТБ представлені у додатку 3 і 4. Як було зазначено раніше, для контролю періоду такту ШІМ сигналу, генератор використовує вхідний сигнал з лічильника. Для контролю скважності сигналу використаємо вхідний 8-ми розрядний сигнал PWM_in та по кожному такту **clk** будемо порівнювати синал отриманий з датчика з сигналом отриманим з лічильника. Якщо сигнал з датчика більший або рівний

сигналу з лічильника, то отримаємо на виході **PWM_out** сигнал низького рівня, в іншому випадку отримаємо сигнал високого рівня, що і буде відповідати ШІМ сигналу зі змінною скважністю та постійним періодом як зазначено в формулі 1.1. Також не слід забувати про те, що генератор постійно очікує задній фронт сигналу **rst_n,** якщо він рівний 0, то **PWM_out** встановиться в високоімпедансний стан, що відповідає таблиці 1.

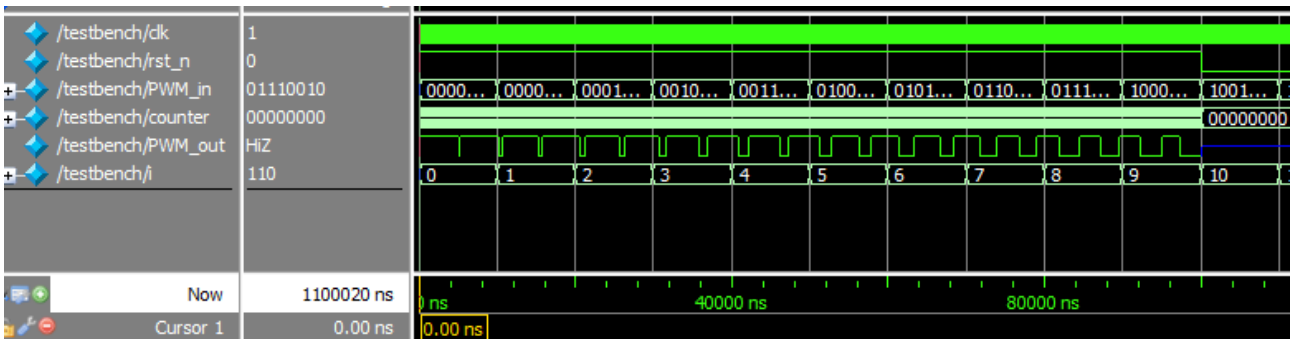

Рис. 2.2 Зміна сигналу PWM_out від зміни значення PWM_in

В створеному ТБ файлі я задав вхідний сигнал **clk,** котрий зміню кожні 10с або PERIOD/2, де параметр PERIOD = 20нс. Сигнал **rst_n** в нульовий момент часу я задаю рівним 1 та по наступному задньому фронту **clk** переводжу в 1, що відповідає RESET в один такт і ще **rst_n** встановлюю в 0 на 600000 нс, відповідно отримуємо RESET та вихід PWM_out встрановлюється у високоімпедансний стан, що підтверджує описаний принцип роботи вище. Також я збільшую на 15 значення з вхоту PWM_in, що призводить до зменшення скважності високого рівна на кожному періоді вихідного сигналу PWM_out.

Спосіб підключення генератора до ядра та тестування результатів

Для отримання сигналу зміни скважності було обрано підключення до блоку Data Memory. Вихідний сигнал з блоку пам’яті Data_out був підключений через мультиплексор з сигналом конролю входу PWM_load для завантаження нового значення скважності в генератор через регістр RegPWM мета якого, полягає в за пам’ятовуванні значення скважності.

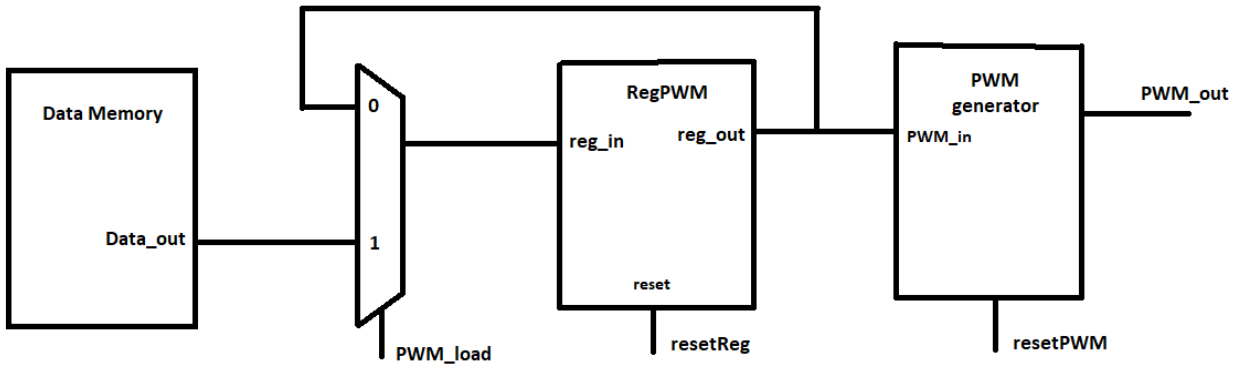

Рис. 3.1 Схема підключення ШІМ-генератора

Головною проблемою підключення генератора є те, що при кожній зміні значення з виходу Data_out скважність також би змінювалась, тому і був підключений мультиплексор з контролем входження PWM_load, задача якого полягає в контролі зміни значення скважності у потрібний момент часу.

Для тестування в ТБ, я створив вхідні та вихідні сигнали відповідно нашому контролеру. Вхідний сигнал **clk** я зміню кожні 10с або PERIOD/2, де параметр PERIOD = 20нс. Всі сигнали **RESET** в нульовий момент часу я задаю рівними 0 та через 2нс переводжу в 1. Сигнал PWM_load встановлюю в 1 в різні проміжки часу для завантаження різних значень скважності котрі задані завдяки інструкцій в ROM па’яті ядра. Результат симуляції цього моменту часу продемонстрований на рис. 3.2.

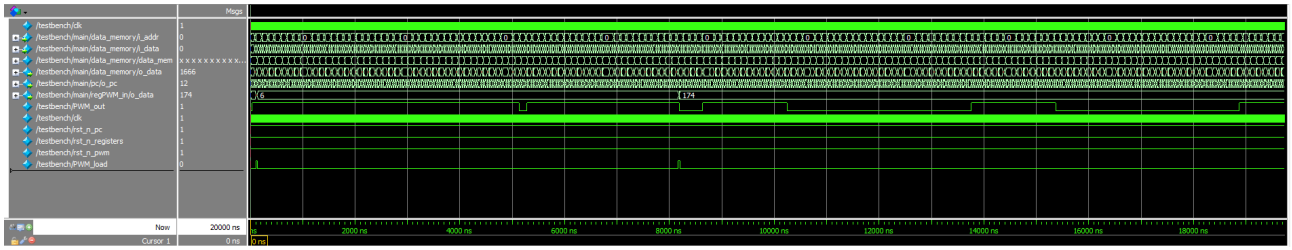

Рис. 3.2 Сигнали симуляції

З рисунку 4.2 видно, що при встановленні сигналу PWM_load в 1 звантажується значення скважності та на виході ШІМ-генератора створюється сигнал відповідний вхідному значенню котрий не змінюється до моменту входження нового значення скважності

ВИСНОВОК

При виконанні розрахункової графічної роботи я навчився створювати та достліджувати переферійні схеми та підключати їх до власного MIPS ядра.

У розрахунковій роботі були описанні результати роботи основних блоків прототипу підключення ШІМ-генератора. Симуляція у програмному забезпеченні ModelSim показує основні принципи роботи проекту, та дозволяє вести розробку без застосування реальних кристалів FPGA. Такий підхід допомагає продуктивно вести розробку командами розробників з економивши достатньо великі суми на закупівлі тестових радіо-компонентів.

ДОДАТКИ

Додаток 1

```
module counter_8bit (
    input clk,
    input rst_n,
    output [7:0] counter);

reg [7:0] counter_r = 0;

assign counter = counter_r;

always @(posedge clk, negedge rst_n) begin
    if (~rst_n)
    counter_r <= 8'b0;
    else
    counter_r <= counter_r + 1;
end
endmodule
```

Додаток 2

```
`timescale 1ns / 1ps

module testbench;

parameter PERIOD = 20;
reg  clk, rst_n;
wire [7:0] counter;

counter_8bit counter_8bit_inst(.clk(clk),
                               .rst_n(rst_n),
                               .counter(counter)
                               );



initial begin
    clk = 0;
    forever #(PERIOD/2) clk = ~clk;
end

initial begin
    rst_n = 1'b0;

    @(negedge clk) rst_n = 1;

    repeat (30) @(negedge clk);

    #1000 $finish;
end


endmodule
```

Додаток 3

```
module PWM_generator (PWM_in, PWM_out , clk , rst_n, counter);

input clk, rst_n;
input [7:0] PWM_in;
input [7:0] counter;
output  PWM_out;


reg PWM_out_r;


always @ (posedge clk, negedge rst_n) begin
    if(~rst_n) PWM_out_r <= 0'bz;
    else begin
        if (PWM_in >= counter) PWM_out_r <= 0;
        else PWM_out_r <= 1;
    end
end

counter_8bit counter_8bit_inst(.clk(clk),
                               .rst_n(rst_n),
                               .counter(counter)
                               );

assign PWM_out = PWM_out_r;

endmodule
```

Додаток 4

tb_pwm_generator.v

```
1   `timescale 1ns / 1ps
2
3   module testbench;
4
5   parameter PERIOD = 20;
6   reg  clk, rst_n;
7   reg [7:0] PWM_in;
8   wire [7:0] counter;
9   wire PWM_out;
10
11  integer i;
12
13  PWM_generator PWM_generator_inst(.PWM_in(PWM_in),
14                                   .PWM_out(PWM_out),
15                                   .clk(clk),
16                                   .rst_n(rst_n),
17                                   .counter(counter)
18                                   );
19
20  initial begin
21      PWM_in = 0;
22      for(i = 0; i < 8'b11111111; i = i + 1) #(10000) PWM_in = PWM_in + 5;
23  end
24
25  initial begin
26      clk = 0;
27      forever #(PERIOD/2) clk = ~clk;
28  end
29
30  initial begin
31      rst_n = 1'b0;
32
33      @(negedge clk) rst_n = 1;
34      #600000 rst_n = 0;
35
36      #1000000 $finish;
37  end
38
39
40  endmodule
```