НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА КОНСТРУЮВАННЯ ЕЛЕКТРОННО-ОБЧИСЛЮВАЛЬНОЇ АПАРАТУРИ

Курсова робота

з курсу:

«Проєктування систем на кристалі»

на тему: «Генератор ШІМ сигналу з керованою частотою та коефіцієнтом заповнення»

Виконав:
Геращенко А.Ю.
студент V курсу ФЕЛ
групи ДК-31мп

Перевірив:
ас. Омелян А. В.
Допущено до захисту:
“___”_____20__ р.
Захищено з оцінкою:
_____

Київ – 2024

Національний Технічний Університет України

"Київський Політехнічний Інститут

імені Ігоря Сікорського„

Кафедра *Конструювання електронно-обчислювальної апаратури*

Дисципліна *Проєктування систем на кристалі*

Спеціальність *172 Електронні комунікації та радіотехніка*

Курс *5* Група *ДК-31мп* Семестр *IX*

ЗАВДАННЯ

до курсового проєкту

Геращенка Артема Юрійовича

(прізвище, ім'я та по батькові)

1. Тема проєкту *Генератор ШІМ сигналу з керованою частотою та коефіцієнтом заповнення*

2. Строк здачі студентом закінченого проєкту (роботи) *03.12.2024*

3. Вихідні дані до проєкту (роботи)

Розробити генератор ШІМ сигналу, з можливістю керування частотою сигналу та коефіцієнтом заповнення.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що розроблюються)

1. ЗАВДАННЯ НА ПРОЄКТУВАННЯ ТА РОЗРОБКА СТРУКТУРНОЇ СХЕМИ

2. РОЗРОБКА ПРИСТРОЮ

3. ДОСЛІДЖЕННЯ ПРОЄКТУ В СЕРЕДОВИЩІ QUARTUS

4. ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ РОЗРОБЛЕНОГО ПРИЛАДУ

6. Дата видачі завдання *16.09.2023*

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів виконання курсового проєкту	Час виконання етапів проєкту
1	Розробка технічного завдання	16.09.2023-20.10.2023
2	Аналіз теоретичних матеріалів	20.10.2023-10.11.2023
3	Створення програмного забезпечення в	10.11.2023-30.12.2023
4	Оформлення пояснювальної записки	30.12.2023-03.01.2024

Студент _____

(підпис)

Керівник _____

(підпис)

«____» _____ 2024

Зміст

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

ШІМ – широтно-імпульсна модуляція

FPGA – Field Prgrammable Gate Array

ПЛІС – програмована логічна інтегральна схема

ARM – Advanced RISC Machine

ВСТУП

Широтно-імпульсна модуляція (ШІМ) - це метод управління потужністю, який полягає в зміні ширини імпульсів при постійній частоті. ШІМ широко використовується в різних галузях електроніки, таких як перетворювачі напруги, регулятори швидкості, драйвери світлодіодів, аудіо- та відео- кодери, комунікаційні пристрої тощо. ШІМ дозволяє ефективно контролювати потужність навантаження.

Одним з основних параметрів ШІМ сигналу є коефіцієнт заповнення, який визначає відношення часу, коли сигнал має високе значення, до періоду сигналу. Коефіцієнт заповнення впливає на середнє значення сигналу та рівень гармонік, які виникають при перетворенні ШІМ сигналу в аналоговий. Зміна коефіцієнта заповнення дозволяє регулювати потужність навантаження без зміни частоти ШІМ сигналу, що зручно для застосувань, які вимагають стабільної частоти, наприклад, аудіо-підсилювачі.

Іншим важливим параметром ШІМ сигналу є частота, яка визначає кількість імпульсів за одиницю часу. Частота впливає на швидкість реакції навантаження, якість перетворення ШІМ сигналу в аналоговий.

Таким чином, можна зробити висновок, що генератор ШІМ сигналу зкерованою частотою та коефіцієнтом заповнення є актуальним та перспективним об'єктом дослідження, який має широке практичне застосування в різних сферах електроніки.

Метою даної курсової роботи є розробка генератора ШІМ сигналу, який може змінювати частоту та коефіцієнт заповнення на базі системи на кристалі, що реалізована на відлагоджувальній платі DE1 SoC.

Практична значимість даної курсової роботи полягає в створенні функціонального прототипу генератора ШІМ сигналу з керованою частотою та коефіцієнтом заповнення, який може бути використаний для різних застосувань, таких як регулювання потужності навантаження, перетворення сигналів, комунікації, модуляції тощо.

РОЗДІЛ 1. ЗАВДАННЯ НА ПРОЄКТУВАННЯ ТА РОЗРОБКА СТРУКТУРНОЇ СХЕМИ ПРИСТРОЮ

1.1 Опис завдання на проєктування

Ціллю проєкту є розробка пристрою який являє собою генератор ШІМ сигналу. Розроблюваний генератор повинен мати можливість керування частотою сигналу та коефіцієнтом заповнення. Пристрій повинен бути компактним для забезпечення мобільності роботи.

1.2 Розробка структурної схеми

Для реалізації пристрою буде використовуватись відлагоджувальна плата DE1 SoC, тому необхідно розглянути які ресурси є у нашому розпорядженні. На рисунку 1 зображено всі доступні на відлагоджувальній платі компоненти. З рисунку 1 видно, що всі компоненти мають з'єднання з мікросхемою 5CSEMA5F31C6N, яка поєднує в собі частину FPGA та частину Hard Processor System, яка представлена двохядерним процесором ARM Cortex A9.

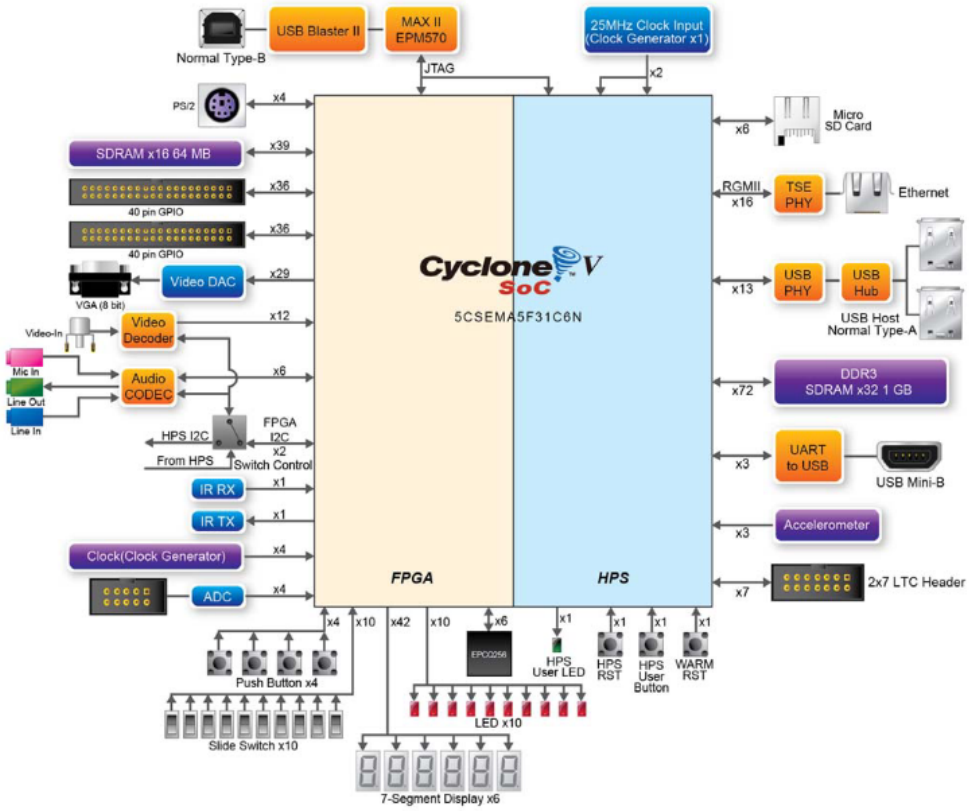

Рис. 1 – Доступні ресурси на платі DE1 SoC

Оскільки у вимогах до курсової роботи було зазначено, що у розроблюваному пристрої необхідно реалізувати зв’язок ARM процесору з FPGA частиною, було вирішено використати процесорне ядро для керування параметрами ШІМ сигналу, а генератор буде реалізований на ПЛІС. Структурна схема пристрою зображена на рисунку 2.

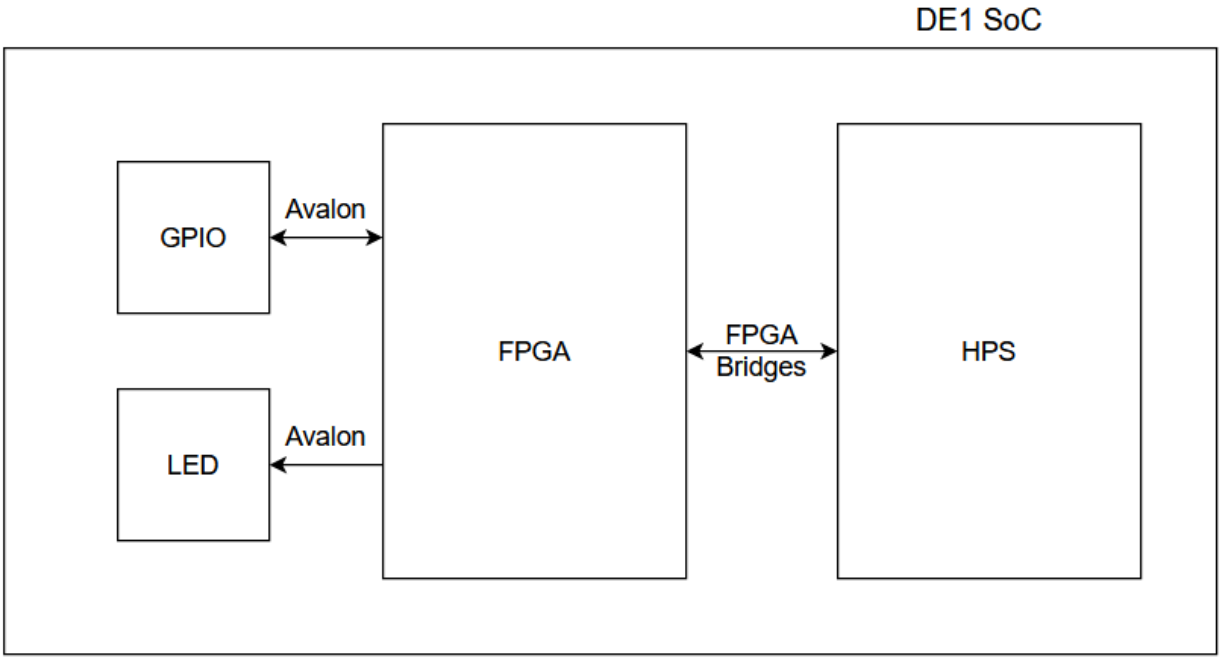

Рис. 2 – Структурна схема пристрою

Зі структурної схеми можна побачити, що для реалізації пристрою буде використовуватись ARM процесор, FPGA, порти вводу/виводу, світлодіоди.

Частина FPGA містить опис всіх з’єднань та модулів з яких складається система на кристалі. Побудову системи на кристалі виконано у Platform Designer з використанням IP бібліотеки Intel та додаванням власного модуля, який реалізує генератор ШІМ сигналу.

Керування побудованою системою забезпечує ARM процесор, він виконує користувацьку програму та надсилає на входи модуля генератора ШІМ сигналу дані для керування його частотою та коефіцієнтом заповнення.

Порти вводу/виводу використовуються для виводу згенерованого ШІМ сигналу, також до виходу генератора під’єднано світлодіод, щоб можна було наглядно побачити його роботу.

Для завантаження програми користувача на плату використовується USB Blaster, повну структурну схему плати зображено на рисунку 3.

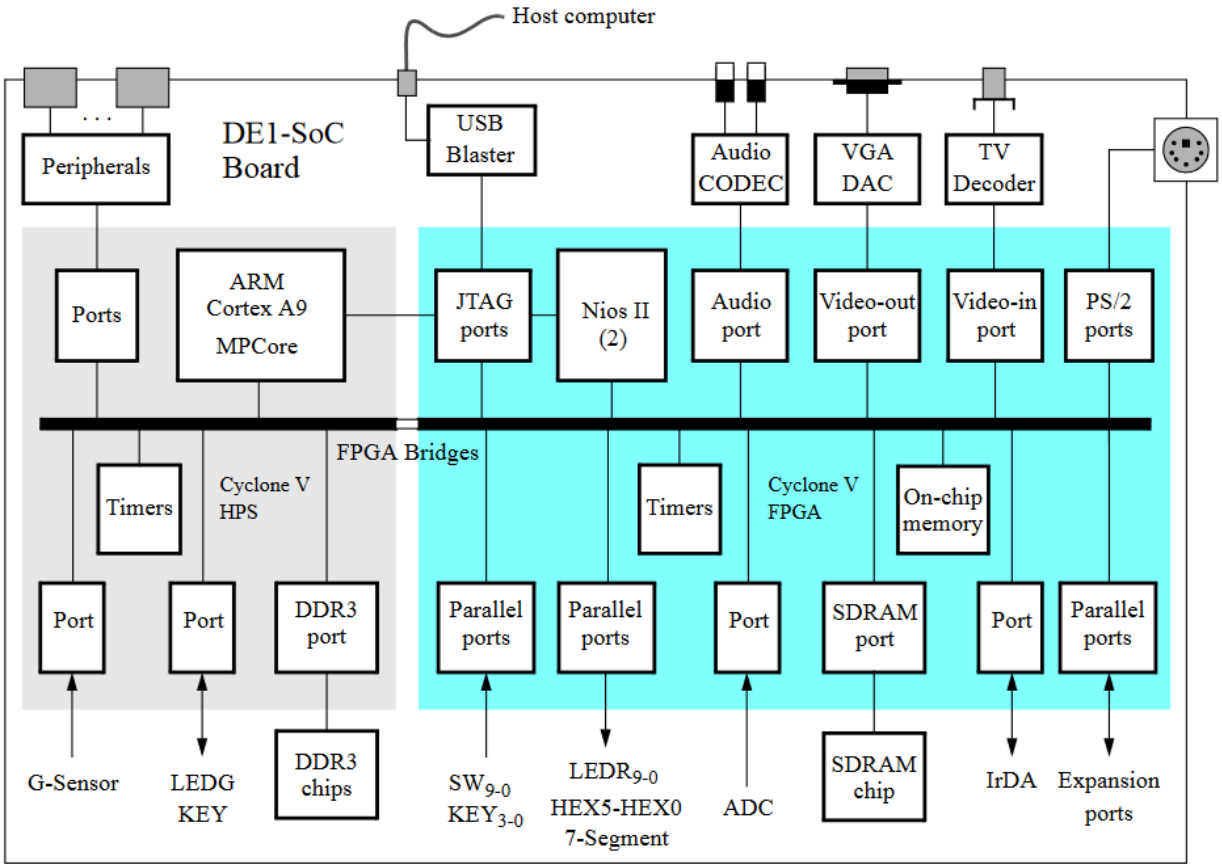

Рис. 3 – Структурна схема DE1 SoC

РОЗДІЛ 2. РОЗРОБКА ПРИСТРОЮ

2.1 Розробка модуля генератора ШІМ сигналу

Модуль генератора ШІМ сигналу було реалізовано на мові опису апаратури Verilog, модуль включає в себе наступні сигнали:

```
module my_pwm(
// signals to connect to an Avalon clock source interface
clk,
reset,
// signals to connect to an Avalon-MM slave interface
slave_address,
slave_read,
slave_write,
slave_readdata,
slave_writedata,
slave_byteenable,
// Non-Avalon Interface IO
led,
my_pwm
);
```

clk – сигнал тактування;

reset – сигнал скидання;

slave_address – сигнал, який визначає адресу взаємодії з Avalon інерфейсом;

slave_read – сигнал, який інформує що відбувається операція читання через Avalon інтерфейс;

slave_write – сигнал, що вказує на те, що відбувається операція запису через Avalon-MM інтерфейс;

slave_readdata – дані, які читаються з інтерфейсу Avalon-MM і передаються для подальшого використання в модулі;

slave_writedata – дані, які записуються через інтерфейс Avalon-MM і використовуються для збереження або обробки модулем;

slave_byteenable – сигнал, який використовується для вказання типу транзакції (байт, півслово, слово) для периферійних операцій читання та запису через інтерфейс Avalon-MM;

led – вихідний сигнал, який керує світлодіодом;

my_pwm – вихідний сигнал, який представляє собою ШІМ сигнал, згенерований модулем.

Принцип роботи генератора ШІМ наступний:

Інтерфейс Avalon-MM призначений для зв'язку з периферійними пристроями та зовнішнім середовищем, він приймає дані для запису (slave_write) і читання (slave_read) разом з адресою (slave_address) і даними (slave_writedata, slave_readdata). Також використовується сигнал slave_byteenable для вказання типу транзакції (байт, півслово, слово).

Модуль містить набір регістрів для керування параметрами ШІМ сигналу, такими як період (period_ff), коефіцієнт заповнення (duty_cycle_ff), лічильник (cnt_ff) та сигнал виведення (out_ff). Значення періоду та коефіцієнту заповнення надходять через Avalon шину у вигляді «склеєних» 32 бітних даних, з яких перші 16 біт містять інформацію про період сигналу, останні 16 біт містять інформацію про скважність сигналу.

ШІМ сигнал генерується за допомогою двох основних параметрів: періоду і коефіцієнта заповнення. Кожен такт сигналу clk використовується для ітерації лічильника (cnt_ff). Лічильник порівнюється з параметром періоду (period_ff), якщо лічильник досягає значення періоду, він скидається в нуль, в іншому випадку збільшується на одиницю. Сигнал виведення (out_ff) встановлюється відповідно до того, чи лічильник менший за коефіцієнт заповнення (duty_cycle_ff).

Результат ШІМ сигналу доступний через порт my_pwm. Також результат використовується для керування сигналом світлодіода (led).

2.2 Розробка системи на кристалі

Для розробки цілісної системи було використано інструмент Platform Designer. Для розробки системи було використано проєкт DE1 SoC Computer, який поставляється разом з програмним забезпеченням Quartus Prime в рамках University program. До системи було додано наступні компоненти:

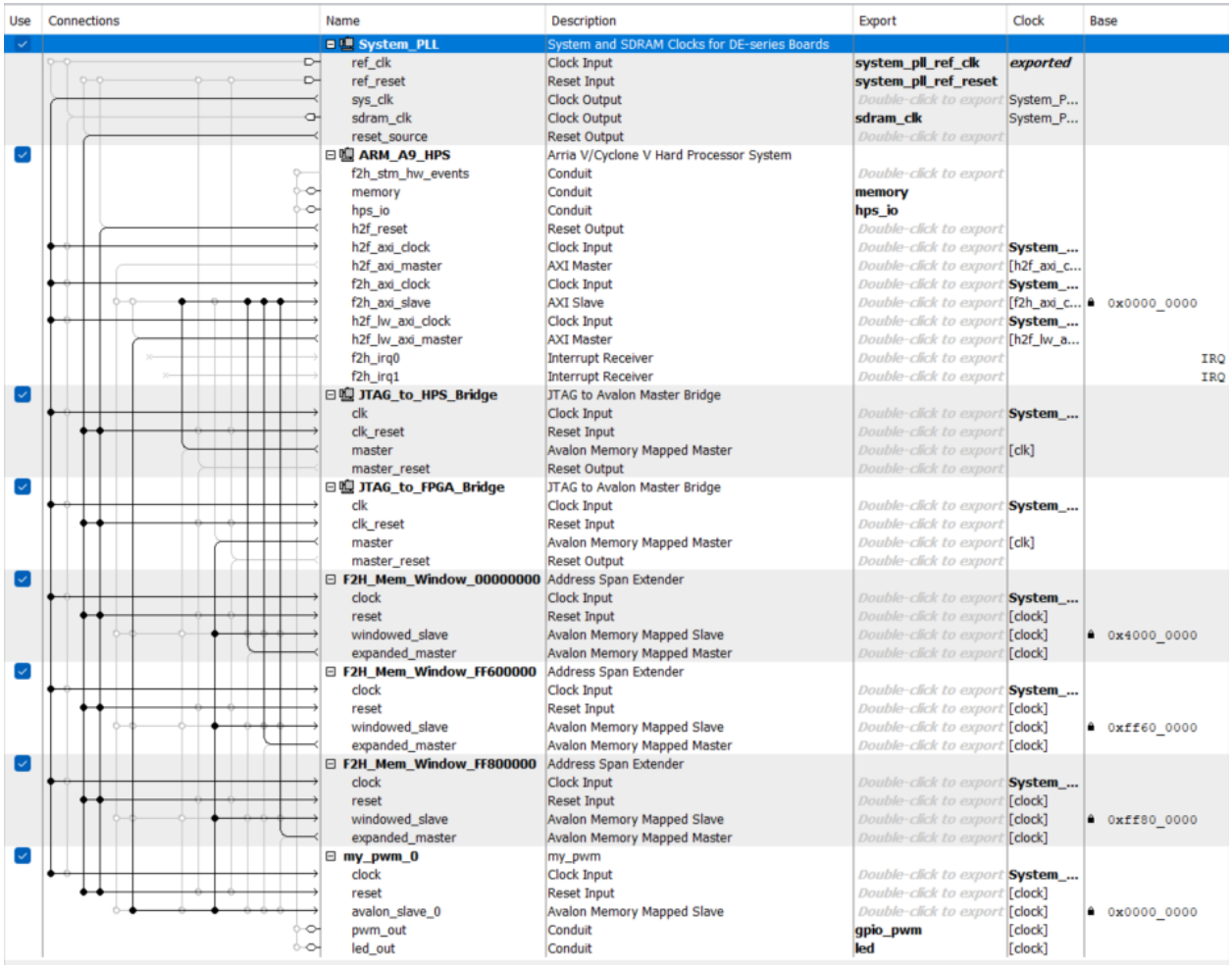

Рис. 4 – Вигляд системи у Platform Designer

Компонент System_PLL являє собою pll блок, який підвищує частоту тактового генератора з 50МГц до 100МГц, від цього компоненту тактується вся система. Налаштування System_PLL показано на рисунку 5.

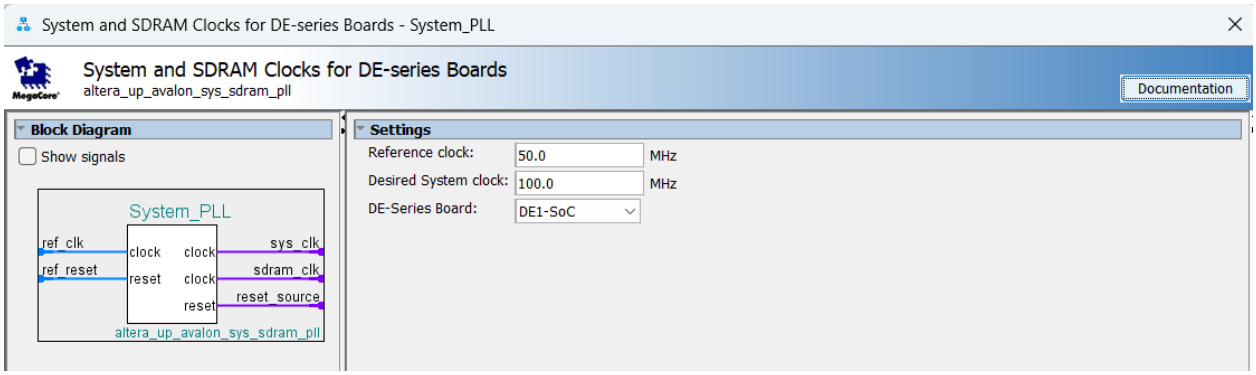

Рис. 5 – Налаштування System_PLL

Компонент ARM_A9_HPS – це процесорне ядро ARM, воно використовується для керування всією системою.

Компоненти JTAG_to_HPS_Bridge та JTAG_to_FPGA_Bridge необхідні для завантаження прошивки на плату.

Компоненти F2H необхідні для реалізації зв'язку між HPS частиною та FPGA частиною, вони були позначені як FPGA Bridges на структурній схемі пристрою.

Компонент my_pwm – це кастомний компонент, який реалізує генератор ШІМ, опис його сигналів та принципу роботи було наведено в попередньому підпункті.

Додавання власного компоненту до системи на кристалі відбувається наступним чином:

1. У вікні IP Catalog необхідно натиснути Add.

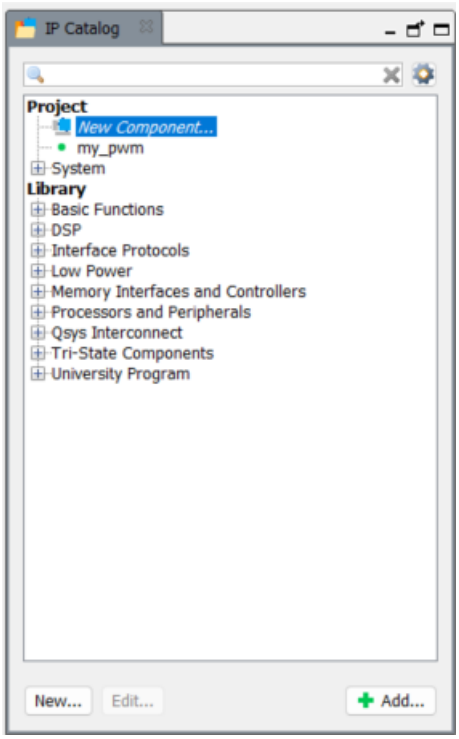

Рис. 6 – Додавання власного компоненту

2. У вікні що відкрилось необхідно заповнити опис компоненту.

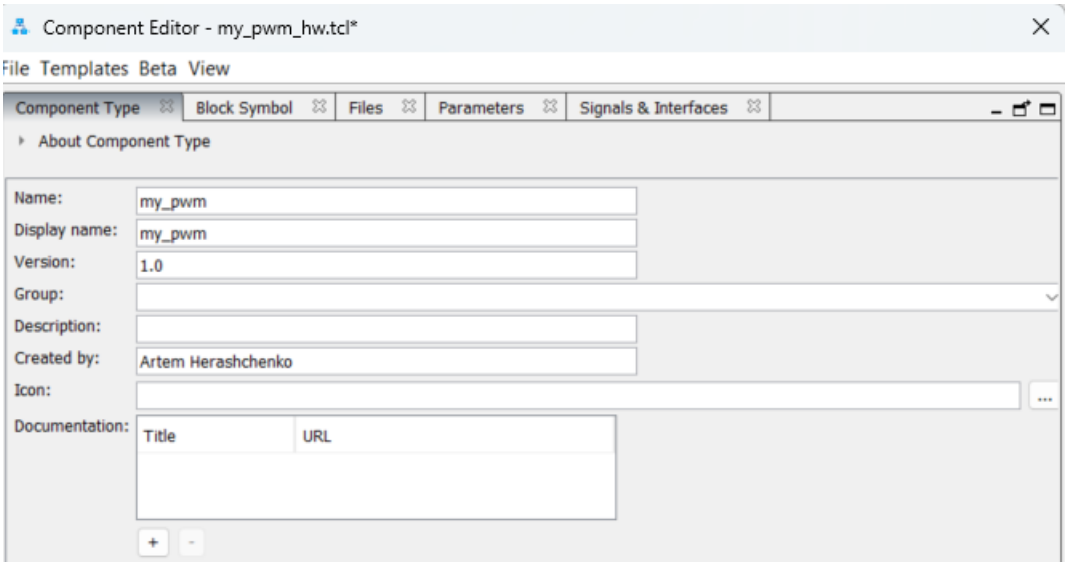

Рис. 7 – Опис компоненту

3. Перейти на вкладку Files та додати Verilog файл з кастомним модулем у поле Synthesis Files, потім натиснути Analyze Synthesis Files. У полі Verilog Simulation Files необхідно натиснути Cope from Synthesis Files.

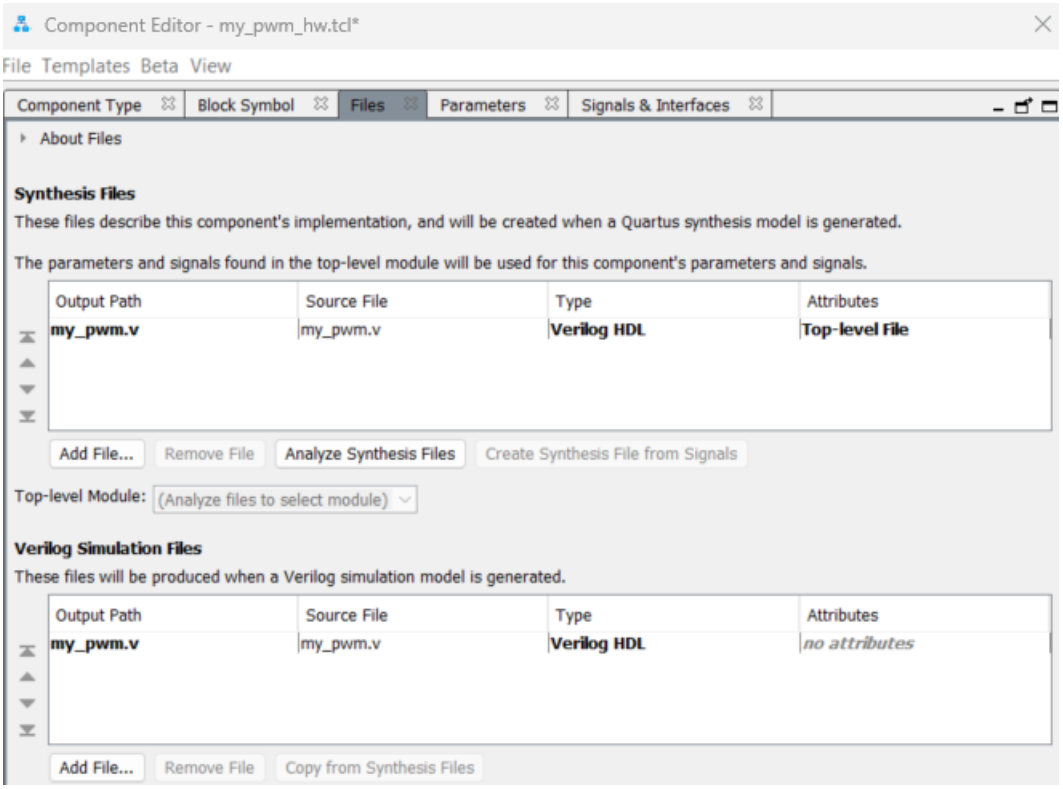

Рис. 8 – Додавання файлів з описом модуля

4. На вкладці Signals & Interfaces налаштувати сигнали власного компоненту. Сигнали, які необхідні для взаємодії з шиною Avalon будуть визначені автоматично, функціонал власних сигналів необхідно визначити автоматично, щоб Platform Designer розумів їх призначення.

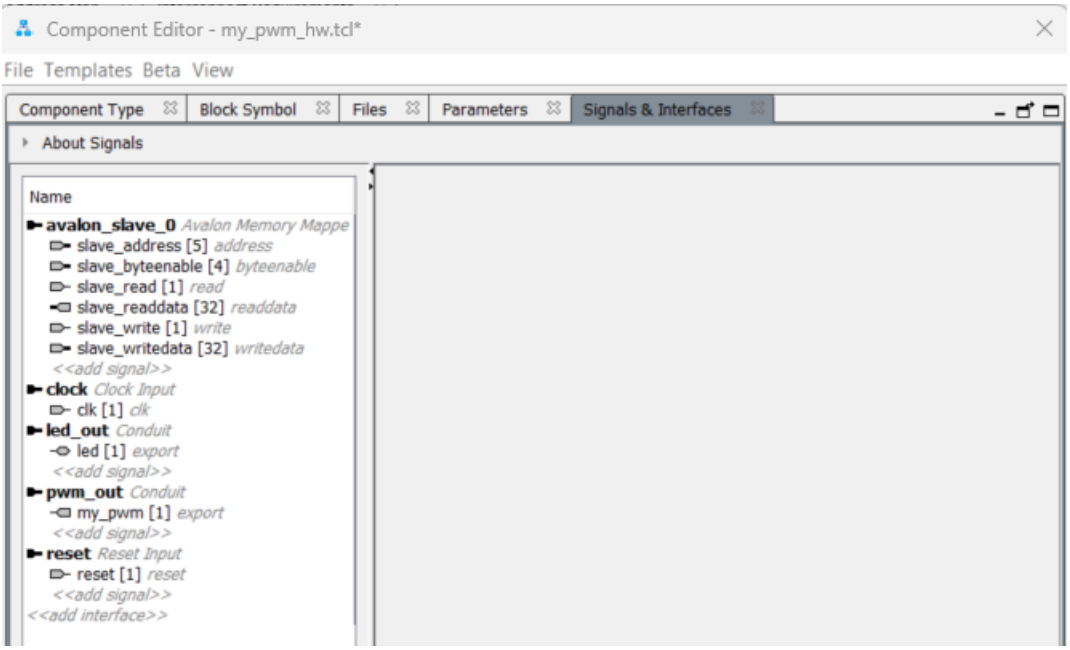

Рис. 9 – Налаштування сигналів

Після того, як всі зв’язки між компонентами системи налаштовано, необхідно призначити базові адреси для всіх доданих компонентів та додати вихідні сигнали у стовбці Export.

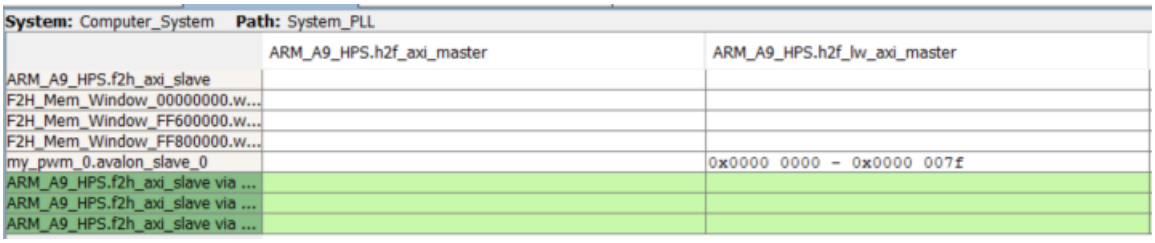

Рис. 10 – Базова адреса для взаємодії з генератором ШІМ

РОЗДІЛ 3. ДОСЛІДЖЕННЯ ПРОЄКТУ В СЕРЕДОВИЩІ QUARTUS

3.1 Схема системи на кристалі в RTL Viewer

Проєкт реалізовано у програмному середовищі Quartus Prime 18.1. В якості плати для розробки було обрано DE1 SoC на базі мікросхеми Cyclone V CSEMA5F31C6N. До проєкту було додано створену у Platform Designer систему на кристалі, після компіляції проєкту було одержано структурну схему у RTL Viewer, яка зображена на рисунку 11.

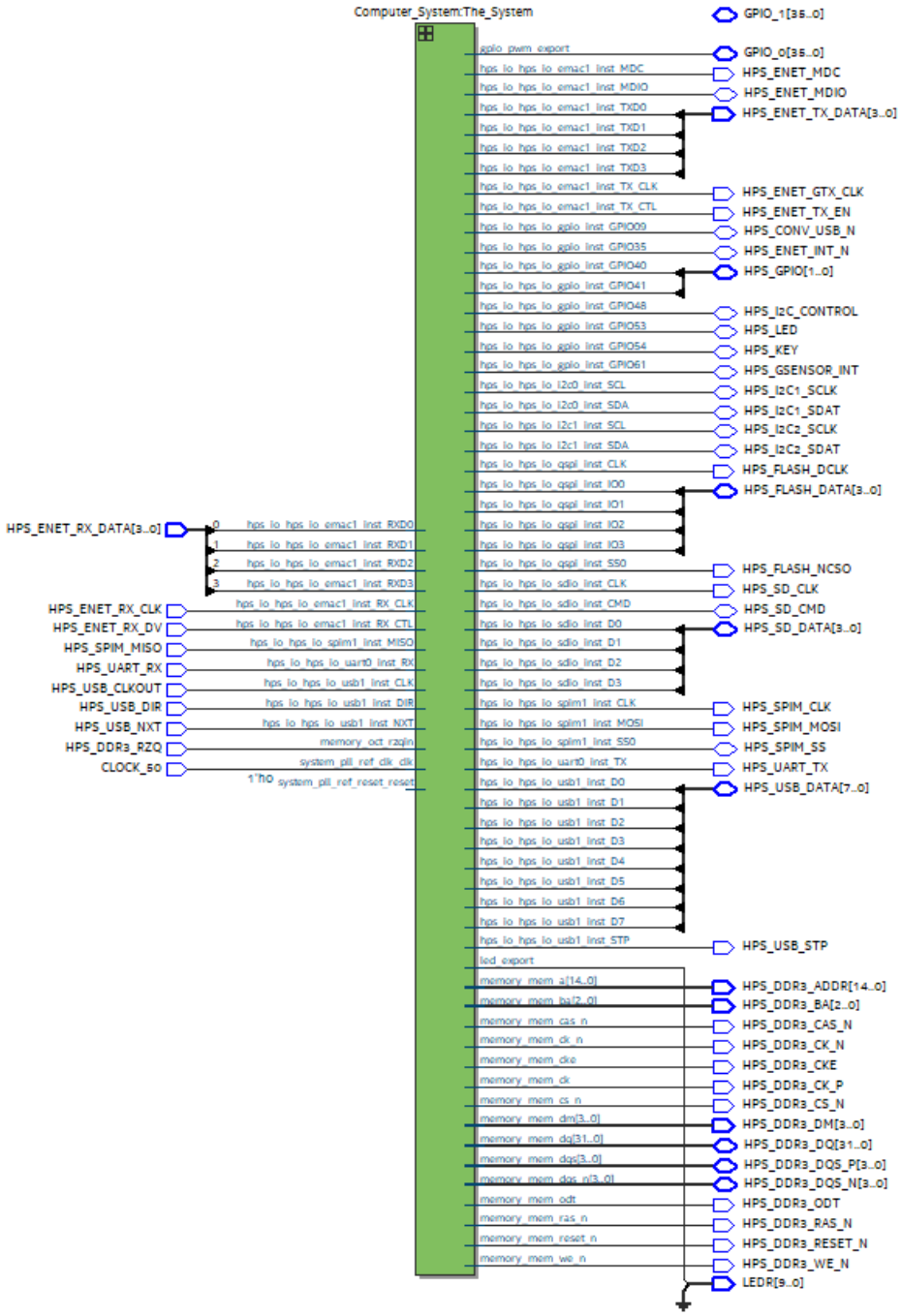

Рис. 11 – Структурна схема проєкту

3.2 Призначення контактів системи в Pin Planer

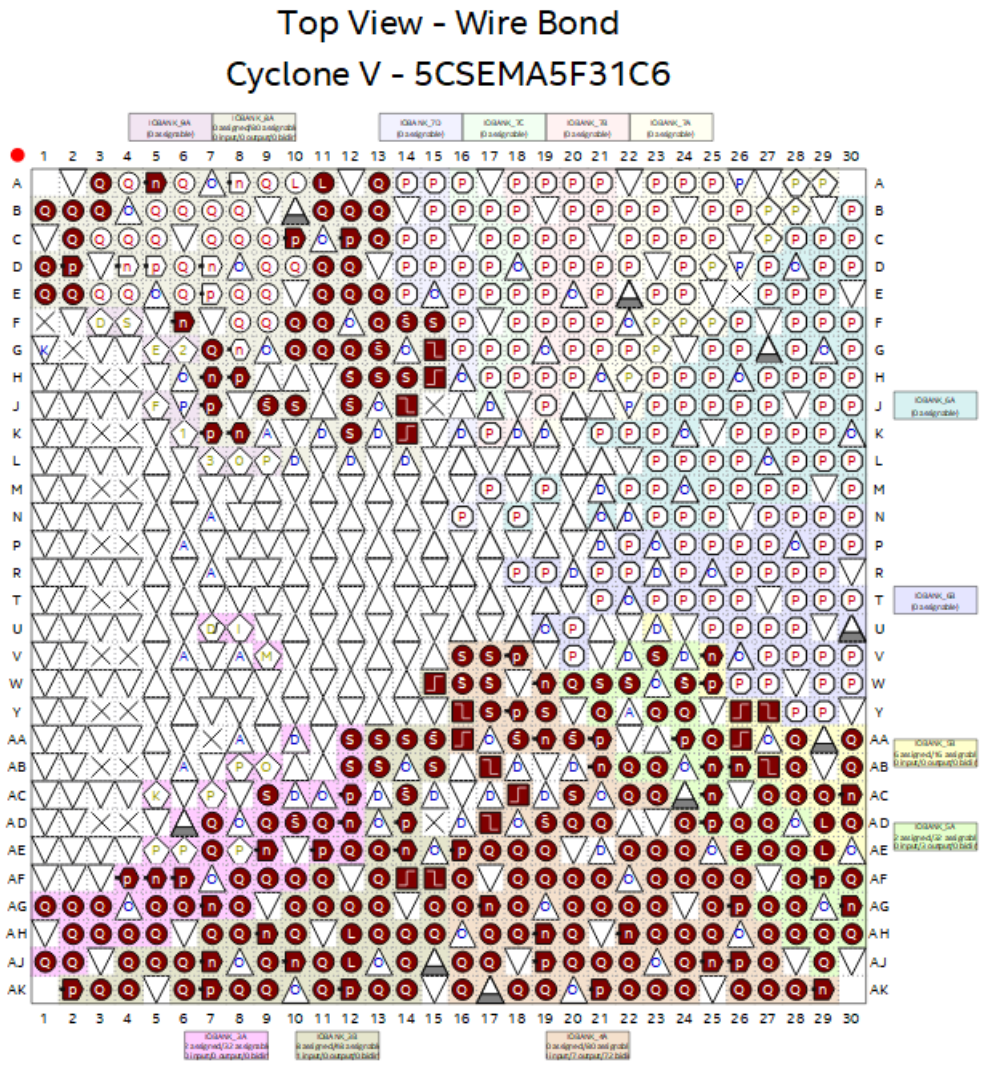

Рис. 12 – Призначення контактів системи на кристалі

3.3 Схема проєкту в Technology Map Viewer

Схема дизайну проєкту на рівні комірок після оптимізації зв’язків між блоками (Post-fitting) та після розміщення елементів на карті комірок FPGA (Post-mapping).

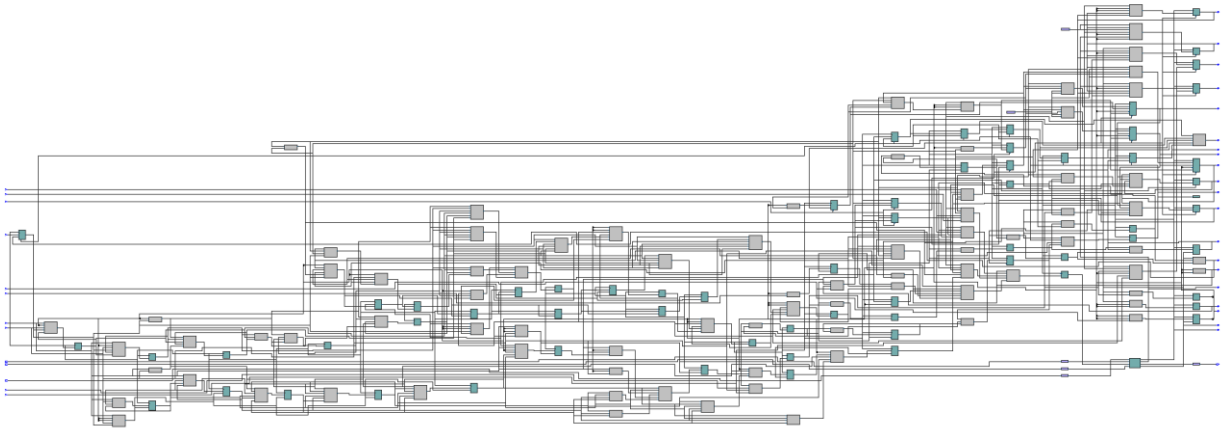

Рис. 13 – Схема Post-Mapping

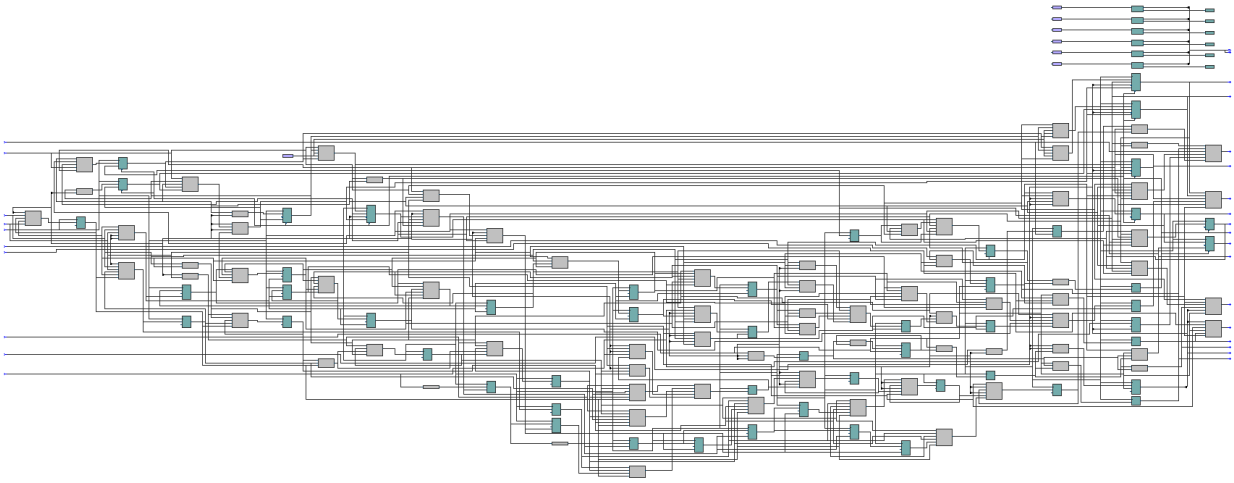

Рис. 14 – Схема Post-Fitting

Як видно зі схем, проєкт було скомпоновано більш оптимально після Post-Mapping.

3.4 Загальна кількість ресурсів, що витрачена на реалізацію проєкту

Після успішної компіляції проєкту у вікні Flow Summary можна побачити загальну кількість ресурсів, які були використані для реалізації проєкту.

Flow Status	Successful - Wed Jan 03 22:03:29 2024
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	DE1_SoC_Computer
Top-level Entity Name	DE1_SoC_Computer
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	1,986 / 32,070 (6 %)
Total registers	2902
Total pins	211 / 457 (46 %)
Total virtual pins	0
Total block memory bits	1,024 / 4,065,280 (< 1 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1 / 6 (17 %)
Total DLLs	1 / 4 (25 %)

Рис. 15 – Загальна кількість витрачених ресурсів

РОЗДІЛ 4. ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ РОЗРОБЛЕНОГО ПРИЛАДУ

4.1 Тестування Verilog модуля ШІМ генератору

Для тестування працездатності Verilog модуля ШІМ генератора було використано програмне забезпечення Questa. Для зручності тестування з модуля було вилучено сигнали, які використовуються інтерфейсом Avalon. Результат тестування для найбільшого значення періоду сигналу (0xFFFF) та найменшого коефіцієнту заповнення (0x0001).

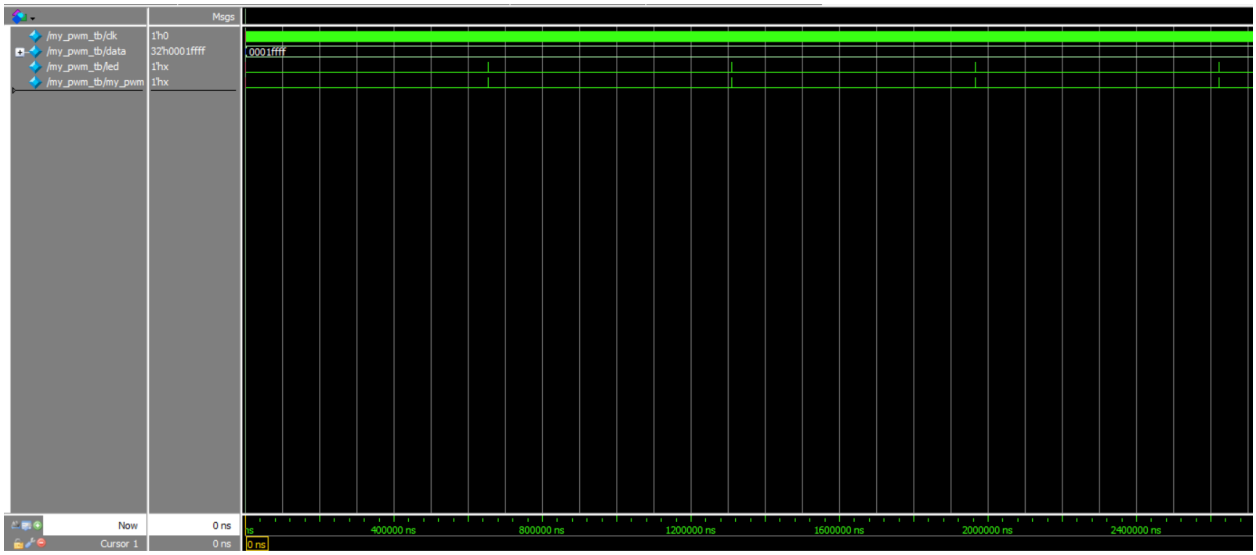

Рис. 16 – Період (0xFFFF) та коефіцієнт заповнення (0x0001)

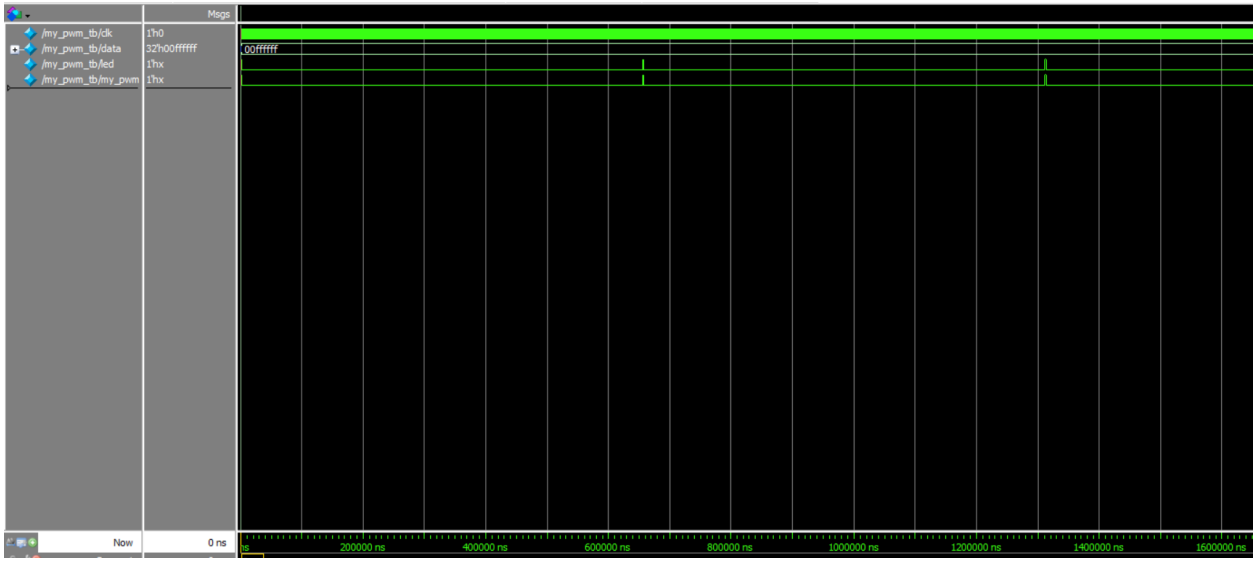

Рис. 17 – Період (0xFFFF) та коефіцієнт заповнення (0x00FF)

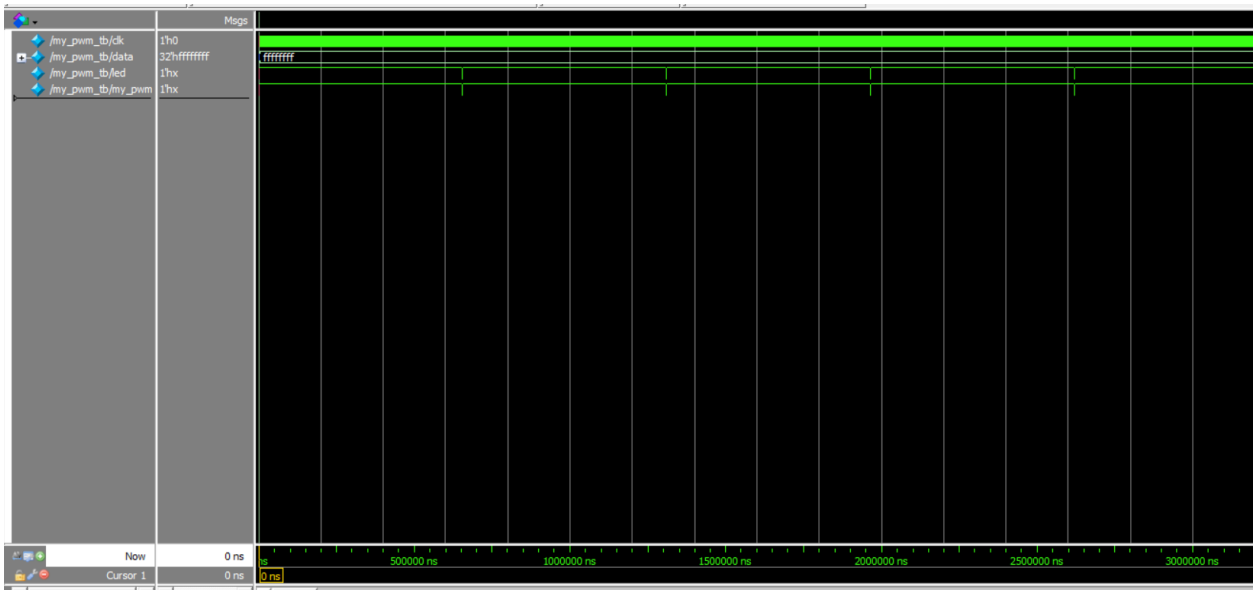

Рис. 18 – Період (0xFFFF) та коефіцієнт заповнення (0xFFFF)

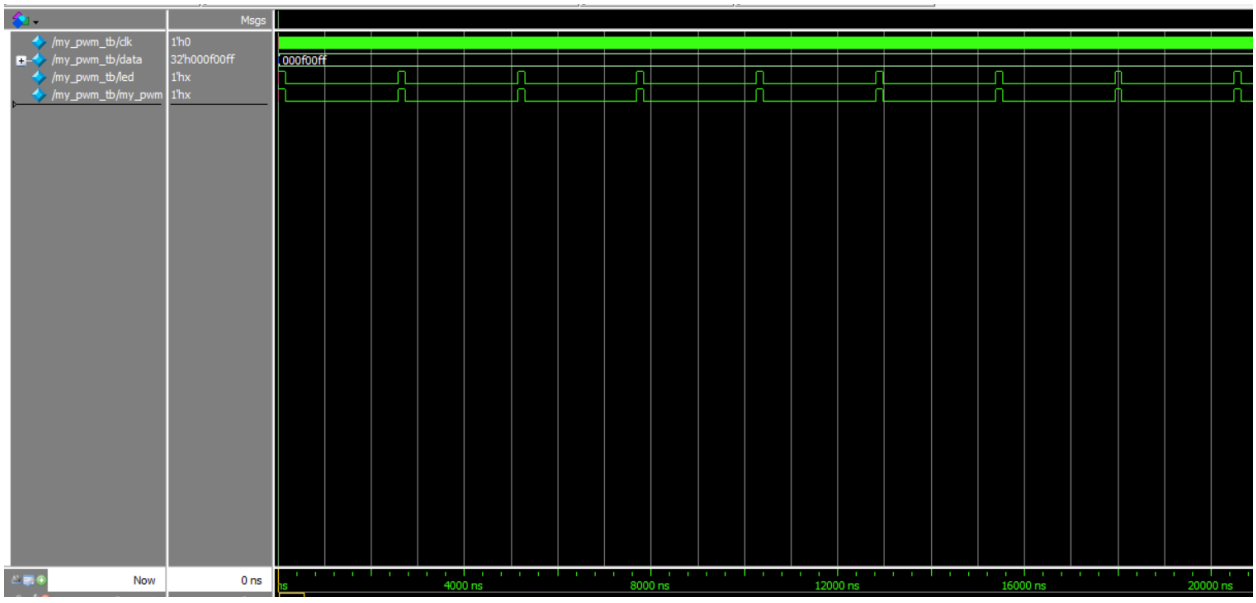

Рис. 19 – Період (0x00FF) та коефіцієнт заповнення (0x000F)

4.2 Тестування пристрою на відлагоджувальній платі DE1 SoC

Для тестування пристрою на відлагоджувальній платі було написано програму для процесорного ядра ARM на мові С, код програми можна побачити у додатках. Дана програма реалізує передачу 32 бітного числа за допомогою шини H2F яка дозволяє процесорному ядру взаємодіяти з периферійними модулями, що розташовані на FPGA.

Принцип роботи програми наступний, користувач вводить значення для періоду і коефіцієнту заповнення PWM через стандартний ввід, використовуючи функцію scanf. Після перевірки на коректність введених значень програма формує дані для запису до регістра керування PWM.

Значення data_to_write формується шляхом об'єднання значень duty_cycle і period. У цьому випадку, duty_cycle зсувається на 16 біт вліво, а потім об'єднується зі значенням period за допомогою операції побітового злиття (|).

Після формування даних для запису програма увійде в нескінченний цикл while(1), де вказівник address використовується для запису значення data_to_write у регістр керування PWM. Після запису значення до регістру програма виводить це значення на екран.

Програма завантажувалась до процесорного ядра та відлагоджувалась за допомогою програмного забезпечення Intel Monitor Program.

Результат роботи ШІМ генератора спостерігався на світлодіоді LEDR[0].

Рис. 20 – Період (0xFFFF) та коефіцієнт заповнення (0x0001)

Рис. 21 – Період (0xFFFF) та коефіцієнт заповнення (0xFFFF)

ВИСНОВКИ

В ході виконання курсового проєкту було розроблено пристрій, який являє собою генератор ШІМ сигналу з керованою частотою та коефіцієнтом заповнення.

В якості апаратної частини було використано відлагоджувальну плату DE1 SoC, яка містить в собі процесорне ядро ARM, що керує генератором та FPGA частину Cyclone V на якій реалізовано генератор, та порти вводу/виводу на яких формується вихідний сигнал генератору.

З програмної частини було розроблено тестову програму на мові С, яка реалізує керування частотою та коефіцієнтом заповнення. Для відлагодження та тестування програмної частини було використано Intel Monitor Program.

Підсумовуючи результати, отримані в ході виконання курсової роботи, можна стверджувати, що розроблений пристрій повністю задовольняє вимогам технічного завдання.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DE1-SoC User manual [Електронний ресурс]. Режим доступу: http://surl.li/owcbs

2. DE1-SoC Getting Started_Guide [Електронний ресурс]. Режим доступу: http://surl.li/owcid

3. DE1-SoC Computer ARM [Електронний ресурс]. Режим доступу: http://surl.li/owcii

4. Embedded Systems Design [Електронний ресурс]. Режим доступу: http://surl.li/owcim

5. Linux_On_DE_Series_Boards [Електронний ресурс]. Режим доступу: http://surl.li/owcip

ДОДАТОК А. Лістинг програм

pwm.c

```c
#include <stdio.h>
#include <stdlib.h>
#define PWM_BASE               0xFF200000

int main(void){

    volatile unsigned int *address = (volatile unsigned int *)PWM_BASE;

    unsigned int duty_cyle = 0x1;     // duty cycle
    unsigned int period = 0xFFFF;   //period
    unsigned int data_to_write;

    printf("Period pwm in hex:\n");
    scanf("%x", &period);

    printf("Duty cycle pwm in hex:\n");
    scanf("%x", &duty_cyle);

    if (period < 0 || period > 0xFFFF || duty_cyle < 0 || duty_cyle > 0xFFFF)
    {
        printf("The period and duty cycle values must be non-negative and no more
than 0xFFFF!");
        return -1;
    }

    data_to_write = (duty_cyle << 16) | period;

    printf("!!!ENTERING THE LOOP!!!");

    while(1){

        *address = data_to_write;

        printf("DATA: 0x%X\n", data_to_write);
    }

    return 0;
}
```

my_pwm.v

```
/*

  Author:  Artem Herashchenko

*/


module my_pwm(
// signals to connect to an Avalon clock source interface
clk,
reset,
// signals to connect to an Avalon-MM slave interface
slave_address,
slave_read,
slave_write,
slave_readdata,
slave_writedata,
slave_byteenable,
// Non-Avalon Interface IO
led,
my_pwm
);

//************************************************************************
// Module Interface
//************************************************************************
input clk;
input reset;

// slave interface
input [4:0] slave_address;
input slave_read;
input slave_write;
output wire [31:0] slave_readdata;
input [31:0] slave_writedata;
input [3:0] slave_byteenable;

//-----------------------------------------------------------------------
output led;           // output signal to LEDR[0]
output my_pwm;        // output signal to GPIO_0[1]
//-----------------------------------------------------------------------

//************************************************************************
// Register Addresses
//************************************************************************
localparam  GPIO_OUT_ADDR   = 5'b0000;
```

```
//*****************************************************************
// Register Set
//*****************************************************************
reg     [31:0] gpio_out_r;
reg     [15:0] cnt_ff  = 0;                     // counter reg
reg     [15:0] duty_cyle_ff = 0;                // duty cycle reg
reg     [15:0] period_ff = 0;                   // period reg
reg            out_ff  = 1'b0;                 // output signal reg

//*****************************************************************
// Wires/Reg
//*****************************************************************
wire  [31:0] gpio_out;
wire  [15:0] i_period;
wire  [15:0] i_duty_cyle;

//*****************************************************************
// Output Assignments
//*****************************************************************

// Input signals for registers
assign gpio_out       = ( (slave_address == GPIO_OUT_ADDR )   && slave_write ) ?
slave_writedata : gpio_out_r;
assign i_period = gpio_out [15:0];
assign i_duty_cyle = gpio_out [31:16];

always @(posedge clk) begin
    gpio_out_r  <= gpio_out;
end

always @(posedge clk) begin
    period_ff    <= i_period;
    duty_cyle_ff <= i_duty_cyle;
end

always @(posedge clk)
    if(cnt_ff == period_ff)
        cnt_ff <= 0;
    else
        cnt_ff <= cnt_ff + 1'b1;

always @(posedge clk)
    out_ff <= (cnt_ff < duty_cyle_ff);

assign my_pwm = out_ff;
assign led = out_ff;

endmodule
```