**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В. И. Ульянова (Ленина)**

**Кафедра информационной безопасности**

отчет

**по лабораторной работе №2**

**по дисциплине «Электроника и схемотехника»**

Тема: Использование встроенных кнопок и светодиодов платы Tang Nano 9K, создание, симуляция и загрузка собственного проекта

|  |  |  |
| --- | --- | --- |
| Студент гр. 3363 |  | Минко Д. А. |
| Студент гр. 3363 |  | Гончаренко О. Д. |
| Студент гр. 3363 |  | Овсейчик Н. И. |
| Преподаватель |  | Рыбин В. Г. |

**Цель работы**

Ознакомиться с использованием встроенных кнопок и светодиодов платы Tang Nano 9K, изучить процесс создания, симуляции и загрузки собственного проекта в среде Gowin EDA, разработать модуль для работы с LED-светодиодами с применением широтно-импульсной модуляции (PWM), протестировать работу модуля с использованием тестбенча.

**Оборудование и программное обеспечение:**

* Плата Tang Nano 9K
* Среда разработки Gowin EDA
* Язык описания оборудования Verilog

**Ход работы**

1. Создание проекта в Gowin EDA

В Gowin EDA был создан новый проект с выбором ПЛИС GW1NR-LV9QN88PC6/I5 из семейства GW1NR. Название проекта — pwm\_led

1. Создание Verilog-файла

В проект был добавлен файл pwm\_led.v, где реализован модуль для управления светодиодами через ШИМ. Модуль использует один входной тактовый сигнал от осциллятора с частотой 27 МГц и 6 светодиодов в качестве выходов.

1. Определение входов и выходов модуля, промежуточных переменных

Определены входы и выходы:

* input clk; — это объявление входного сигнала clk, который представляет собой тактовый сигнал (clock).
* output [5:0] led; — это объявление выходного порта **led**, который представляет собой 6-битную шину (6 светодиодов). Каждый бит шины соответствует одному светодиоду на плате Tang Nano 9K.
* localparam LED\_COUNT = 6; — это локальный параметр LED\_COUNT, который задаёт количество светодиодов, подключённых к выходному порту led.

В качестве внутренней переменной на данном этапе будет использоваться счётчик. Его предназначением является отсчёт тактов для работы ШИМ. Для начала его максимальным значением будет число светодиодов.

reg [2:0] counter;

1. Описание логики работы ШИМ

Реализован счётчик counter для отсчёта тактов и формирования управляющих сигналов для светодиодов.

always @(posedge clk)

begin

if (counter < LED\_COUNT-1)

counter <= counter + 1'b1;

else

counter <= 0; end

Счётчик работает в цикле от 0 до 5 (количество светодиодов минус один). На каждом тактовом сигнале clk он увеличивается на 1, а когда достигает максимального значения (5), сбрасывается в ноль. Этот счётчик можно использовать для управления яркостью светодиодов с помощью ШИМ: в зависимости от значения счётчика светодиоды будут включаться или выключаться, что создаёт эффект изменения яркости.

1. Логика управления модулем с использованием блока generate

Для каждого светодиода с помощью цикла for была описана логика управления, где каждый светодиод загорается при условии:

assign led[i] = counter > i % LED\_COUNT;

Это условие отвечает за управление каждым светодиодом на основе значения счётчика count, создавая эффект широтно-импульсной модуляции (ШИМ) для изменения яркости светодиодов.

* assign led[i] — оператор assign используется для непрерывного (комбинаторного) присваивания значений. В данном случае он присваивает значение каждому светодиоду led[i]. То есть, для каждого светодиода led[i] будет выполняться условие справа от знака равенства.
* counter > i % LED\_COUNT — это условие сравнивает значение счётчика counter с выражением i % LED\_COUNT:
* i — это индекс текущего светодиода.
* % LED\_COUNT — оператор взятия остатка от деления на LED\_COUNT (6).

counter > i % LED\_COUNT — светодиод led[i] включается, если значение счётчика больше, чем результат этого выражения. В результате светодиоды включаются в определённой последовательности, создавая эффект градиента яркости, который изменяется со временем.

1. Тестирование с использованием тестбенча

Был разработан тестбенч для проверки работы модуля ШИМ. Для корректной симуляции следует добавить обнуление всех внутренних переменных в тестируемый модуль.

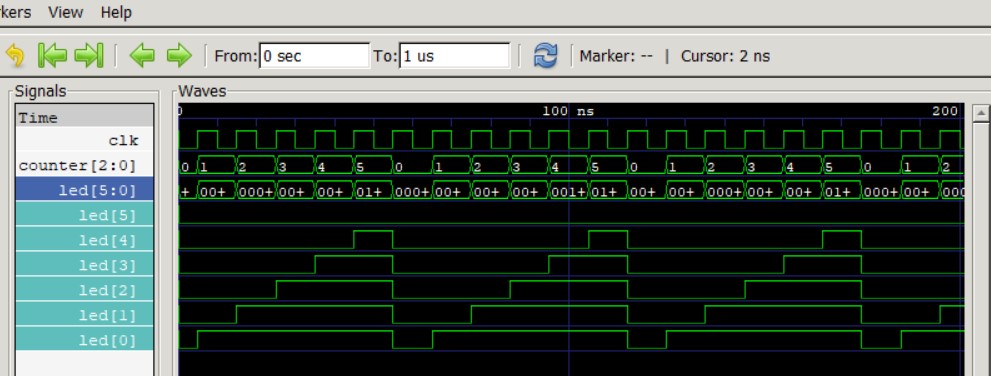
initial

begin

counter <= 0;

end

Выполнив это, модуль ШИМ симуляции выдаст следующий результат (рис. 1):

Рисунок 1 — Реализация модуля ШИМ симуляции

1. Добавление управления с кнопок

Кнопки сдвига градиента светодиодов задаются как входы модуля

input up

input down

В модуль добавляется счётчик сдвига светодиодов (от 0 до 5) и счётчик отката, ограничивающий частоту осуществления сдвига при зажатии кнопки. Разрядность последнего счётчика (локальный параметр COOLDOWN в примере кода) должна быть небольшой (например, 3 разряда) для симуляции и большой (21 разряд) для прошивки:

localparam COOLDOWN = 3;

reg [2:0] shift;

reg [COOLDOWN-1:0] cooldown;

Добавим работу сдвигов при нажатии кнопок:

always @(posedge clk)

begin

if (!cooldown)

begin

if (!up)

begin

cooldown <= 2\*\*COOLDOWN-1;

if (shift < LED\_COUNT-1)

shift <= shift + 1'b1;

else

shift <= 0;

end

else if (!down)

begin

cooldown <= 2\*\*COOLDOWN-1;

if (shift > 0)

shift <= shift - 1'b1;

else

shift <= LED\_COUNT-1;

end

end

else

cooldown <= cooldown - 1'b1;

end

Если счётчик отката имеет значение 0, то проверяется нажатие кнопок (активный уровень – 0). Если кнопка нажата, то счётчик отката устанавливается в максимальное значение и производится сдвиг – увеличение или уменьшение значения на счётчике сдвига. Если сдвиг выходит за диапазон допустимых значений, то значение возвращается обратно с другой стороны. Если же счётчик не равен нулю, то на каждом такте его значение уменьшается на 1.

Тажке модернизируем блок generate с помощью сдвига shift:

assign led[i] = counter > (i + shift) % LED\_COUNT;

shift — это сдвиг, который изменяет позицию светодиодов. Он используется для сдвига градиента яркости по линейке светодиодов.

counter > (i + shift) % LED\_COUNT — светодиод led[i] включается, если значение счётчика больше, чем результат этого выражения. В результате светодиоды включаются в определённой последовательности, создавая эффект градиента яркости, который изменяется со временем.

Итоговая разводка с кнопками будет иметь следующий вид (рис. 2):

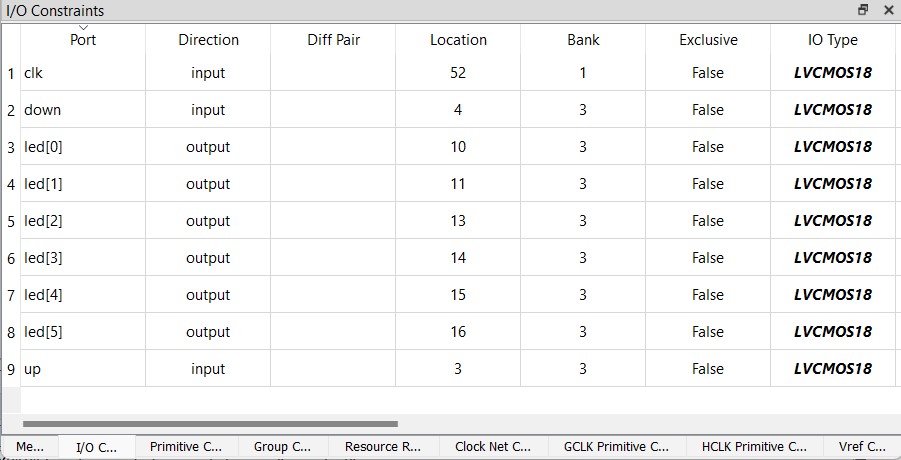


Рисунок 2 — Итоговая разводка с кнопками

Также подключим кнопки к нашей ПЛИС в панели Floor Panel (рис. 3):

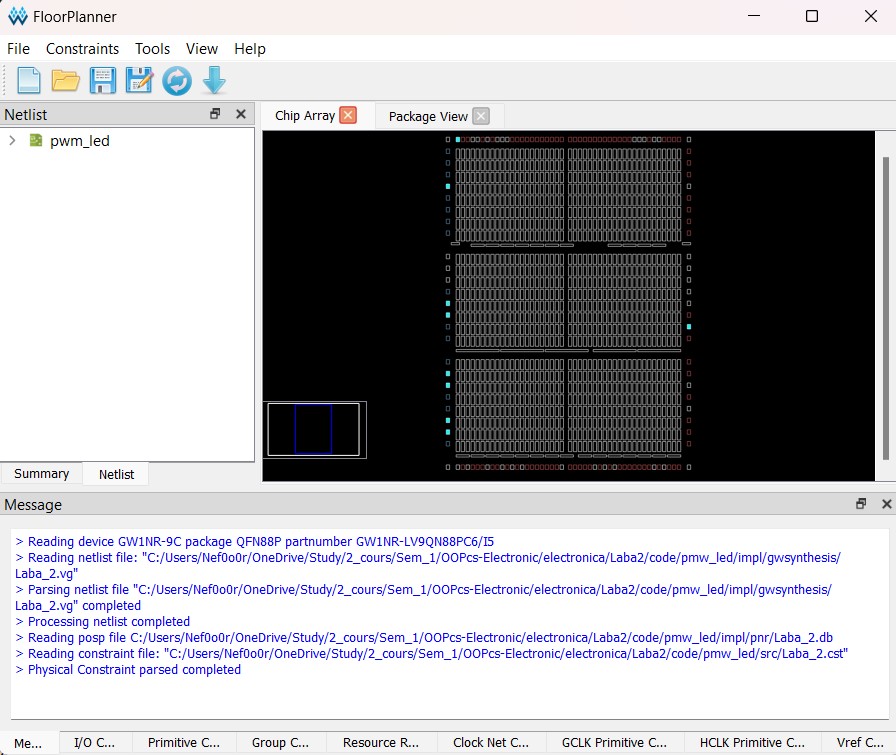


Рисунок 3 — Итоговая разводка с кнопками, подключенная к ПЛИС в панели Floor Panel

В тестбенче сделаем генерацию тактового сигнала и симулирование нажатия кнопок up и down для сдвига градиента яркости светодиодов:

initial

begin

up <= 1;

down <= 1;

#20 up <= 0;

#180 up <= 1;

#200 down <= 0;

#200 down <= 1;

end

Выполнив это, модуль ШИМ симуляции выдаст следующий результат (рис. 4):

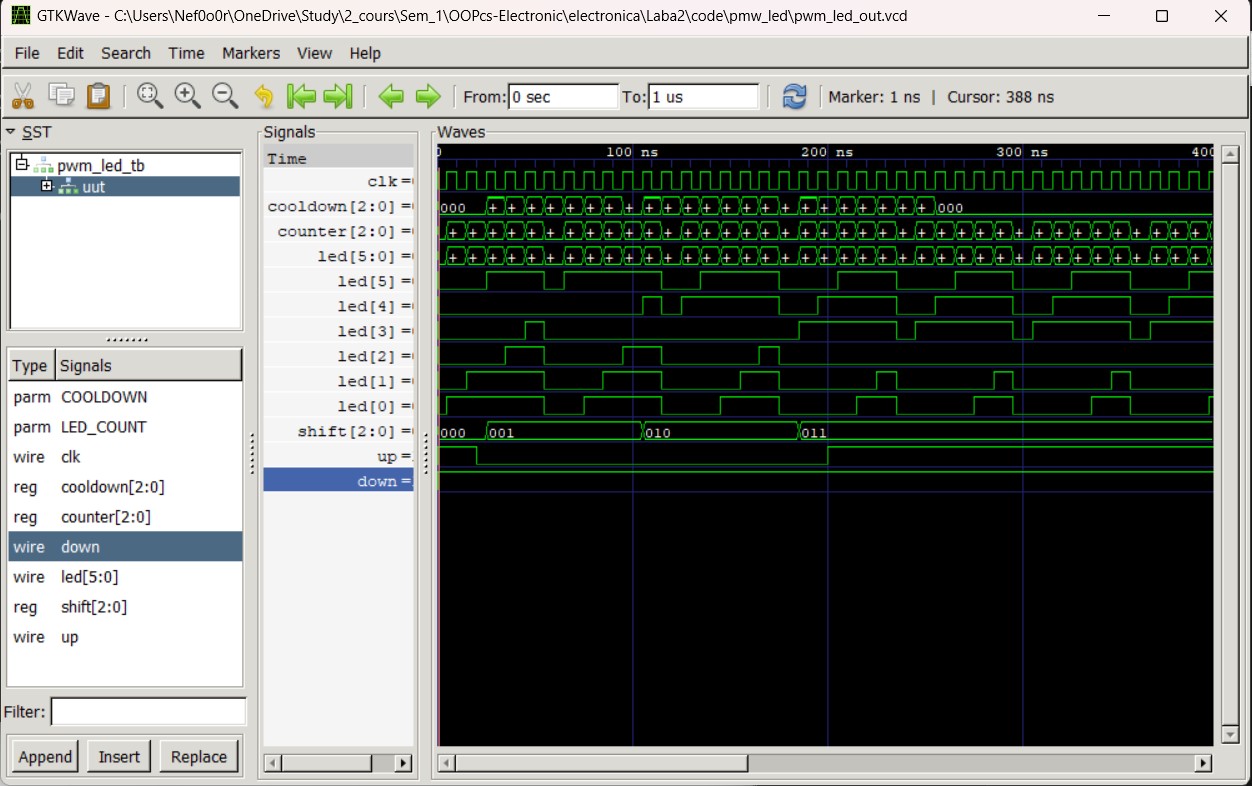


Рисунок 4 — модуль ШИМ симуляции с кнопками

1. Прошивка платы и отладка работы модуля

После успешных синтеза и разводки для прошивки платы используем Programmer и получаем результат, что на плате загорелись светодиоды, образуя градиент яркостей (рис. 5):

Рисунок 5 — Образование градиента светодиодами

Сделана блок-схема алгоритма работы данной программы (рис. X).

Рисунок X – Блок-схема кода

**Вывод**

В результате лабораторной работы был изучен процесс создания проекта в Gowin EDA, разработан и протестирован модуль для управления светодиодами с использованием ШИМ. Проект был успешно загружен на плату, и его работа проверена с помощью физических кнопок управления.

ИСХОДНЫЙ КОД

Исходный код программы:

module pwm\_led

#(parameter LED\_COUNT = 6, parameter COOLDOWN = 21) // для симуляции COOLDOWN = 3, для запуска 21

(

input clk,

input up,

input down,

output [LED\_COUNT-1:0] led

);

initial

begin

counter <= 0;

cooldown <= 0;

shift <= 0;

end

reg [2:0] shift;

reg [COOLDOWN-1:0] cooldown;

reg [2:0] counter;

always @(posedge clk)

begin

if (counter < LED\_COUNT-1)

counter <= counter + 1'b1;

else

counter <= 0;

end

always @(posedge clk)

begin

if (!cooldown)

begin

if (!up)

begin

cooldown <= 2\*\*COOLDOWN-1;

if (shift < LED\_COUNT-1)

shift <= shift + 1'b1;

else

shift <= 0;

end

else if (!down)

begin

cooldown <= 2\*\*COOLDOWN-1;

if (shift > 0)

shift <= shift - 1'b1;

else

shift <= LED\_COUNT-1;

end

end

else

cooldown <= cooldown - 1'b1;

end

genvar i;

generate

for (i = 0; i < LED\_COUNT; i = i + 1)

begin : led\_block

assign led[i] = counter > (i + shift) % LED\_COUNT;

end

endgenerate

endmodule

Исходный код тестбенча:

`timescale 1ns / 1ns

module pwm\_led\_tb();

reg clk; // Тактовый сигнал

reg up;

reg down;

wire [5:0] led; // Светодиоды

// Экземпляр тестируемого модуля

pwm\_led uut(.clk(clk), .up(up), .down(down), .led(led));

// Генерация тактового сигнала с периодом 10 нс (частота 100 МГц)

initial begin

clk = 0;

forever #(5) clk = ~clk; // Период 10 нс

end

initial

begin

up <= 1;

down <= 1;

#20 up <= 0;

#180 up <= 1;

#200 down <= 0;

#200 down <= 1;

end

// Завершение симуляции через 1000 нс

initial begin

#1000 $finish;

end

// Сохранение результата симуляции в файл VCD для анализа

initial begin

$dumpfile("pwm\_led\_out.vcd");

$dumpvars(0, pwm\_led\_tb);

end

endmodule