Лабораторная работа № 6.
Исследование четырехразрядного сумматора.

Цель работы. Изучение функционирования устройства, позволяющего получить арифметическую сумму двух 4-разрядных двоичных чисел.
Постановка задачи. Устройство должно содержать блок 4-разрядного сумматора, блок регистра результата, содержащий также триггер переноса и блок кодопреобразователя, позволяющего получить вывод числа на сегментный индикатор. Структура программы рассмотрена на упражнении, предшествующем лабораторной работе.

Задание на работу в лаборатории.

1. Создать проект под названием **summ4**. При создании проекта **добавить в него файлы coder и latch_rgstr** из работ 3 и 5 соответственно.
2. Открыть файл **latch_rgstr** и переписать его из 4-разрядного в 5-разрядный через изменение параметра. Сохранить измененный файл а проекте.
3. Открыть новый Verilog файл и записать в него текст кода одноразрядного сумматора.

Программа 6.1.

```
module sum
(input wire a,b,cr,
output wire s,crp);
assign s=(a^b)^cr;
assign crp=(a&b) | ((a^b)&cr);
endmodule
```

4. Пользуясь «Приложением 2», получить диаграммы при **интервалах импульса на входе a – 20нс, на входе b – 30нс, на входе cr – 50нс.**

5. Открыть новый Verilog-файл и записать в него **Программу 6.2** для 4-разрядного сумматора. Сохранить файл, установить его старшим в иерархии и откомпилировать.

Программа 6.2

```
module sum_4
(input wire [3:0]a_in,b_in,
input wire cr_in,
output wire [3:0]s_out,
output wire crp_out);
```

```
wire [2:0]crp_n;
sum sum0(.a (a_in[0]),.b (b_in[0]),.cr(cr_in),.s(s_out[0]),.crp(crp_n[0]));
sum sum1(.a (a_in[1]),.b (b_in[1]),.cr(crp_n[0]),.s(s_out[1]),.crp(crp_n[1]));
sum sum2(.a (a_in[2]),.b (b_in[2]),.cr(crp_n[1]),.s(s_out[2]),.crp(crp_n[2]));
sum sum3(.a (a_in[3]),.b (b_in[3]),.cr(crp_n[2]),.s(s_out[3]),.crp(crp_out));
endmodule
```

6. Пользуясь «Приложением 2», получить диаграммы 4-разрядного сумматора. Число **a** установить равным **10** на интервале **30ns** и, затем, на интервале **50ns** равным **3**. Число **b** установить равным **7** на интервале **50ns** и, затем на интервале **50ns** равным **13**. Проанализируйте результат симуляции.

7. Открыть новый Verilog-файл и записать в него **Программу 6.3** для 4-разрядного сумматора с сохранением результата в регистре, сохранением флага переноса в триггере, входящем в состав такого регистра и выводом результата на индикаторы. Сохранить файл под именем проекта, установить его старшим в иерархии и откомпилировать. **Это файл верхнего уровня.**

   Программа 6.3

```
module summ_4
(input wire [3:0]op_a,op_b,
input wire op_c, sync,
output wire led,
output wire [6:0]hex);
wire [3:0]sm;
wire c_y;
wire [4:0]y;
assign led=y[4];
sum_4  block1(.a_in(op_a), .b_in(op_b), .cr_in(op_c), .s_out(sm), .crp_out(c_y));
latch_rgstr  block2(.d_in({c_y,sm[3:0]}), .clk(sync), .d_out(y));
coder  block3(.data(y[3:0]), .seg(hex));
endmodule
```

8. Пользуясь «Приложением 3» произвести разводку выводов схемы для работы в макете таким образом, чтобы ввод **числа “a”** осуществлялся с тумблеров **SW9,SW8,SW7,SW6**(**SW9** – старший разряд), ввод **числа“b”** – с тумблеров **SW5,SW4,SW3,SW2**(**SW5** – старший разряд), вход переноса соединить с тумблером **SW0,** ввод синхронизации - с кнопки **KEY(0)**. Вывод результата производить на **сегментный индикатор 0-й, вывод значения флага переноса на светодиод –**

LEDR 9. После компиляции файла планировщика еще раз откомпилируйте файл верхнего уровня!

9. Пользуясь «Приложением 4» произвести программирование кристалла FPGA макета. На крайнем правом индикаторе должен высветиться «0». Проверить работоспособность схемы. Для этого набирать различные значения чисел **“a”** и **“b”** и нажимать крайнюю правую кнопку, имитируя подачу импульса синхронизации. **Тумблер SW0 установить в 0.**
10. Продемонстрировать работу преподавателю.

Отчет по работе должен содержать тексты программ и диаграммы работы основных блоков устройства.