Вычислительная и микропроцессорная техника

M13 СОЗДАНИЕ МНОГОФАЙЛОВЫХ ПРОЕКТОВ. СУММАТОР. ДЕМУЛЬТИПЛЕКСОР

ъыъ.рф/ыУЫе

Цель работы

Познакомиться принципами работы демультиплексора и сумматора. Научиться создавать многофайловые проекты. Получить первоначальные сведения о программировании ПЛИС.

Задание на лабораторную работу

- Создать модуль демультиплексора на языке Verilog.
- Выполнить функциональную симуляцию демультиплексора согласно варианту;
- Создать модуль шинного демультиплексора на языке Verilog.
- Создать модуль одноразрядного полного сумматора на языке Verilog;
- Создать модуль четырехразрядного сумматора на языке Verilog;
- Выполнить функциональную симуляцию сумматора согласно варианту.
- Создать многомодульное устройство.
- Выполнить программирование отладочной платы DE1-SoC.

Выполнение работы

1. Создайте проект. Путь **/VMT/Lab13/** название **Lab13**. При создании **импортируйте** файл driver.v (из работы M12) в меню *Add Files*.

Демультиплексор

2. Создайте файл типа *Verilog HDL File*. Сохраните его под именем *dmx_1_4.v*.
3. Запрограммируйте демультиплексор с двумя адресными входами на языке Verilog. *Допишите код, приведенный ниже, самостоятельно.*

```
module dmx_1_4 (
        input [] addr,
        input data,
        output reg [] a
);

always @(*) case(addr)
```

```
        2'd1:  a = {2'bzz, data, 1'bz};     // конкатенация {}
                                            // z - высокий импеданс

        default: a = 4'bzzzz;
        endcase
    endmodule
```

4. Сделайте файл *dmx_1_4.v* старшим в иерархии файлов. Выполните анализ и синтез проекта.
5. Произведите функциональную симуляцию демультиплексора.

- addr [1:0] - счетчик, начальное значение 0, шаг 20 нс;
- data - случайный, со случайным шагом.

6. Сохраните результаты работы (скриншот временной диаграммы) от 0 до 160 нс.

Шинный демультиплексор

Шинный демультиплексор представляет собой совокупность демультиплексоров с общей адресной шиной и предназначен для коммутации шин, в отличие от обычных, которые коммутируют лишь одноразрядный сигнал.

7. Создайте файл типа *Verilog HDL File*. Сохраните его под именем *dmx7.v*.
8. Запрограммируйте шинный (7) демультиплексор на языке Verilog. *Допишите код, приведенный ниже, самостоятельно.*

```
module dmx7 (
        input [1:0] addr,
        input [7:0] data,
        output [6:0] a0, a1, a2, a3
);
        // подключение модуля dmx_1_4 под именем dmx0
dmx_1_4 dmx0 (.addr(addr), .data(data[0]), .a({a0[0], a1[0], a2[0], a3[0]}));


endmodule
```

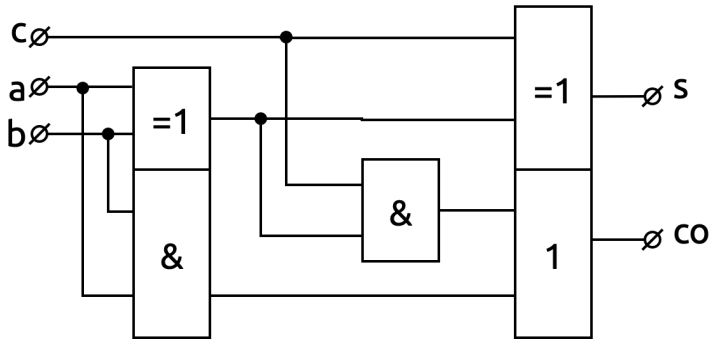
9. Сделайте файл *dmx7.v* старшим в иерархии файлов. Выполните анализ и синтез проекта.
10. Получите RTL-схему модуля dmx7, развернув один из подмодулей.
11. Произведите функциональную симуляцию dmx7, система счисления (Radix) unsigned decimal:

- addr [1:0] - случайный, интервал 30 нс;
- data [6:0] - случайный, со случайным шагом.

12. Сохраните результаты работы (скриншот временной диаграммы) от 0 до 160 нс.

Одноразрядный сумматор

13. Создайте файл типа *Verilog HDL File*. Сохраните его под именем *sum_1.v*.
14. Запрограммируйте одноразрядный сумматор согласно схеме на языке Verilog. *Для справки см. практическое задание М3. Допишите код, приведенный ниже, самостоятельно.*




```
module sum_1 (
       input
       output
);
assign s =
assign co =
endmodule
```

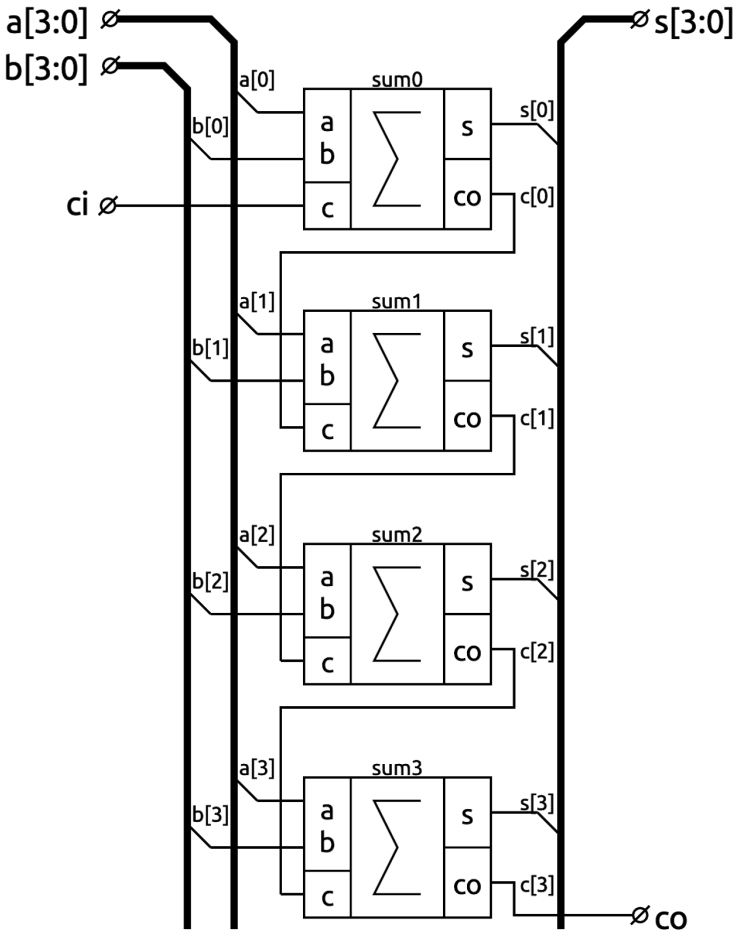
15. Сделайте файл *sum_1.v* старшим в иерархии файлов. Выполните анализ и синтез проекта.
16. Произведите функциональную симуляцию sum_1:

- a - счетчик, шаг 40 нс;
- b - счетчик, шаг 20 нс;
- s - счетчик, шаг 80 нс.

17. Сохраните результаты работы (скриншот временной диаграммы) от 0 до 160 нс.

Четырехразрядный сумматор

18. Создайте файл типа *Verilog HDL File*. Сохраните его под именем *sum_4.v*.
19. Опишите четырехразрядный сумматор согласно схеме на языке Verilog. *Для справки см. практическое задание М3. Допишите код, приведенный ниже, самостоятельного.*



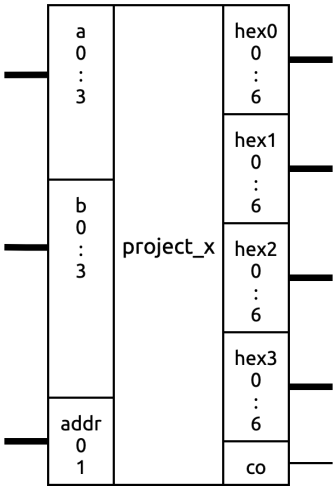

```
module sum_4 (
        input []
        input
        output []
);
wire [2:0] c;  // объявление переменной типа wire (шина)
                    // подключение модулей
sum_1 sum1 (.a(a[0]), .b(b[0]), .c(ci), .s(s[0]), .co(c[0]));
sum_1 sum2 ();


endmodule
```

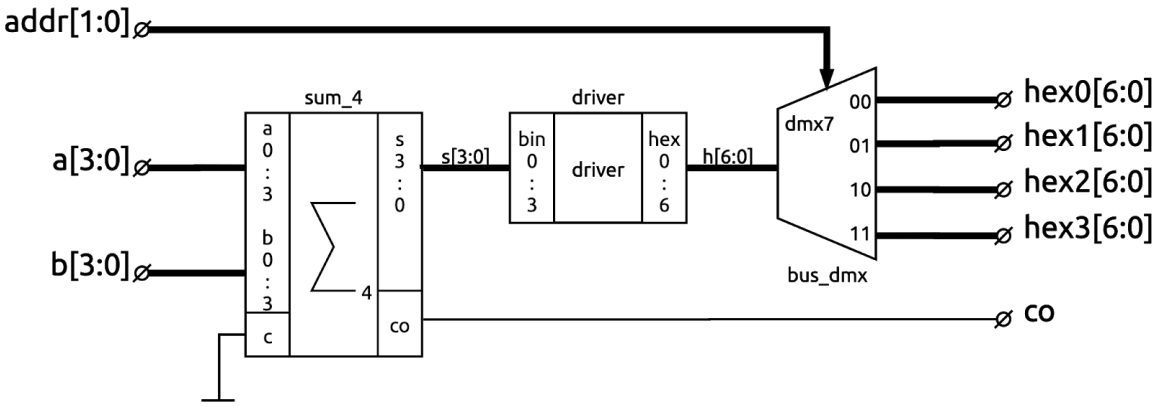
20. Сделайте файл *sum_4.v* старшим в иерархии файлов. Выполните анализ и синтез проекта.
21. Получите RTL-схему модуля sum_4, развернув один из подмодулей.
22. Произведите функциональную симуляцию sum_4 согласно варианту.
23. Сохраните результаты работы (скриншот временной диаграммы) от 0 до 160 нс.

Многомодульное устройство

24. Создайте файл типа Verilog HDL File. Сохраните его под именем *project_x.v*.
25. Опишите модуль на языке Verilog согласно схеме.



Модуль project_x



Содержимое модуля project_x

26. Сделайте файл *project_x.v* старшим в иерархии файлов. Выполните анализ и синтез проекта.
27. Получите RTL-схему модуля *project_x*.
28. Выполните планирование входов и выходов (по указанию преподавателя):
    a. Откройте планировщик входов и выходов (Ctrl-Shift-N).
    b. Убедитесь, что выбрано верное устройство (5CSEMA5F31C6). Выберите, если оно не выбрано.
    c. Заполните столбцы *Location* и *I/O Standard* необходимых сигналов в нижней части рабочего поля в соответствии с номерами контактных ножек и стандартом I/O. Адреса контактных ножек следует взять из таблицы.

- a[3:0] - тумблеры sw[9:6];
- b[3:0] - тумблеры sw[5:2];

- addr[1:0] - тумблеры sw[1:0];
- hex0[6:0] - семисегментный индикатор hex0;
- hex1[6:0] - семисегментный индикатор hex1;
- hex2[6:0] - семисегментный индикатор hex2;
- hex3[6:0] - семисегментный индикатор hex3;
- co - любой светодиод.

d. Запустите анализ назначений входов/выходов. Закройте окно планировщика по окончанию анализа.

29. Скомпилируйте проект (Ctrl+L).
30. Выполните программирование отладочной платы DE1-SoC (по указанию преподавателя).

Содержание отчета

- Схемы устройств;
- Таблицы функционирования устройств;
- Программы устройств на языке Verilog.
- Временные диаграммы функционирования устройств.
- Анализ данных и выводы по работе (заключение).

Варианты симуляции (сумматор)

Вариант 1			Вариант 2			Вариант 3		
t, нс	a	b	t, нс	a	b	t, нс	a	b
0 - 20	101_bin	5_hex	0 - 10	4_dec	100_bin	0 - 30	0_hex	0_bin
20 - 30	6_oct	6_dec	10 - 60	7_hex	111_bin	30 - 40	E_hex	1110_bin
30 - 80	5_hex	101_bin	60 - 70	1_hex	1_dec	40 - 80	11_oct	9_dec
80 - 160	13_dec	15_oct	70 - 160	4_dec	4_oct	80 - 160	13_oct	11_dec

Вариант 4			Вариант 5			Вариант 6		
t, нс	a	b	t, нс	a	b	t, нс	a	b
0 - 20	6_hex	110_bin	0 - 20	1001_bin	9_hex	0 - 20	6_oct	6_dec
20 - 50	100_bin	4_hex	20 - 30	10_oct	8_hex	20 - 50	0_hex	0_bin
50 - 100	5_dec	5_oct	30 - 60	6_hex	6_oct	50 - 80	1011_bin	13_oct
100 - 160	1_oct	1_bin	60 - 160	10_dec	12_oct	80 - 160	2_oct	2_dec

Вариант 7			Вариант 8			Вариант 9		
t, нс	a	b	t, нс	a	b	t, нс	a	b
0 - 20	11_dec	13_oct	0 - 10	0_oct	0_bin	0 - 20	5_hex	101_bin
20 - 30	1010_bin	A_hex	10 - 20	101_bin	5_hex	20 - 30	D_hex	15_oct

30 - 70	11_bin	3_dec	20 - 40	3_hex	11_bin	30 - 50	B_hex	11_dec
70 - 160	11_bin	3_hex	40 - 160	11_bin	3_hex	50 - 160	14_oct	1100_bin

Вариант 10			Вариант 11			Вариант 12		
t, нс	a	b	t, нс	a	b	t, нс	a	b
0 - 20	1000_bin	8_dec	0 - 30	1011_bin	13_oct	0 - 10	10_bin	2_oct
20 - 30	14_dec	1110_bin	30 - 50	9_hex	1001_bin	10 - 20	13_oct	B_hex
30 - 40	14_dec	16_oct	50 - 110	2_oct	2_hex	20 - 80	2_hex	2_oct
40 - 160	1_dec	1_hex	110 - 160	1_hex	1_bin	80 - 160	6_dec	6_oct

Вариант 13			Вариант 14			Вариант 15		
t, нс	a	b	t, нс	a	b	t, нс	a	b
0 - 10	0_bin	0_hex	0 - 20	1101_bin	15_oct	0 - 10	5_dec	5_oct
10 - 20	C_hex	1100_bin	20 - 40	0_oct	0_hex	10 - 20	111_bin	7_oct
20 - 60	F_hex	1111_bin	40 - 70	14_oct	1100_bin	20 - 30	E_hex	1110_bin
60 - 160	2_oct	2_dec	70 - 160	7_dec	7_oct	30 - 160	11_bin	3_hex

Вариант 16			Вариант 17			Вариант 18		
t, нс	a	b	t, нс	a	b	t, нс	a	b
0 - 10	7_oct	7_dec	0 - 30	12_oct	1010_bin	0 - 10	3_hex	11_bin
10 - 70	2_dec	2_oct	30 - 50	6_dec	110_bin	10 - 20	10_bin	2_oct
70 - 80	14_dec	E_hex	50 - 60	3_hex	3_oct	20 - 70	C_hex	1100_bin
80 - 160	7_dec	7_oct	60 - 160	1111_bin	F_hex	70 - 160	7_dec	7_oct

Вариант 19			Вариант 20			Вариант 21		
t, нс	a	b	t, нс	a	b	t, нс	a	b
0 - 10	E_hex	1110_bin	0 - 10	1010_bin	12_oct	0 - 10	15_dec	1111_bin
10 - 40	1000_bin	8_hex	10 - 20	1111_bin	F_hex	10 - 20	12_oct	A_hex
40 - 60	11_oct	9_dec	20 - 60	13_dec	1101_bin	20 - 60	10_bin	2_oct
60 - 160	8_hex	1000_bin	60 - 160	2_oct	2_hex	60 - 160	4_hex	100_bin

Вариант 22			Вариант 23			Вариант 24		
t, нс	a	b	t, нс	a	b	t, нс	a	b
0 - 10	10_oct	8_dec	0 - 10	0_dec	0_hex	0 - 10	110_bin	6_dec
10 - 50	2_oct	2_hex	10 - 40	15_oct	D_hex	10 - 20	13_dec	15_oct

50 - 70	11_oct	9_dec	40 - 50	10_dec	12_oct	20 - 70	6_hex	6_dec
70 - 160	7_dec	7_hex	50 - 160	11_oct	9_dec	70 - 160	100_bin	4_dec

Программирование отладочной платы DE1-SoC

1. Подключите **DE1-SoC**:
- Включите кабель питания в сеть электропитания (розетку),
- Подключите кабель питания в соответствующее гнездо на DE1-SoC,
- Подключите USB-кабель в USB-порт компьютера,
- Подключите USB-кабель в соответствующее гнездо на DE1-SoC.
2. Включите DE1-SoC, нажав **красную кнопку**.
3. Откройте программатор.
4. В окне *Programmer* нажмите **Hardware Setup**. Дважды щелкните ЛКМ на DE1-SoC в таблице *Available hardware items* окна *Hardware Setup*. Нажмите **Close**.
5. Нажмите **Auto Detect**. Выберите 5CSEMA5. Нажмите **OK**.
6. В нижнем рабочем поле окна Programmer выберите правую из двух появившихся схем. Нажмите **Change File…**.
7. Выберите файл с расширением *.sof* в папке *output_files* данного проекта. Имя файла совпадает с названием проекта. Нажмите **Open**.
8. Установите галочку в столбце *Program/Configure* верхнего рабочего поля окна *Programmer* напротив выбранного файла *sof*.
9. Нажмите **Start**.
10. По окончанию работы отключите DE1-SoC, нажав **красную кнопку** и отключив кабели сначала от DE1-SoC, затем из гнезда USB и розетки.