

lab_MS_SV4 задание для самостоятельного выполнения

Платы для аппаратной отладки проекта

MiniDiLaB-CIV	: Микросхема - EP4CE6E22C8,	Вход тактового сигнала (25МГц) – 23
MAX10 NEEK	: Микросхема - 10M50DAF484C6GES,	Вход тактового сигнала (50МГц) – N5
DE0_nano	: Микросхема - EP4CE22F17C6,	Вход тактового сигнала (50МГц) – R8

Описание проекта lab_MS_SV4

Рабочая папка C:\Intel_trn\Q_MS_SV\lab_MS_SV4.

Имя проекта – lab_MS_SV4.

Имя модуля верхнего уровня – lab_MS_SV4.

Файл с описанием – lab_MS_SV4.sv.

Алгоритм работы:

АЛУ (арифметико-логическое устройство) с параметризированной разрядностью (width=8, по умолчанию). Реализуется как комбинационная схема. Выполняет знаковые операции:

- ADD – сложение.
- SUB – вычитание.
- MUL – умножение.
- DIV – деление.
- Операция, соответствующая Вашему номеру в списке группы (см. таблицу ниже). Имя для Вашей дополнительной операции - *op_Ваш номер в списке группы*.

Выводы устройства:

- ops – вход кода операции (**3** бита – в Вашем проекте, в примерах ниже – 2 бита).
- op_a – вход операнда A: знаковый (разрядность width).
- op_b – вход операнда B: знаковый (разрядность width).
- ALU_out – выход результата: знаковый (разрядность width).
- CLK – вход тактового сигнала, для синхронизации ISSPE и SignalTapII.

Программа работы

1. Разработать описание устройства АЛУ – модуль lab_MS_SV4 используя структуру, пакет и другие конструкции SystemVerilog.
 - i. Пакет, с определением структуры, типа данных, параметра, должен быть выполнен отдельным файлом lab_MS_SV4_pack.sv (пример приведен ниже)
 - ii. В основном файле lab_MS_SV4.sv в модуль должна передаваться структура (пример приведен ниже)
2. Разработать тест tb_lab_MS_SV4.sv для проверки АЛУ.
 - a. Тест первого класса – без автоматической проверки (пример приведен ниже).
 - i Значения данных надо выбрать так, чтобы показать правильность работы всех операций, включая дополнительную.
 - b. Результаты теста (должны быть представлены в отчете): временная диаграмма **и** результаты, выводимые в консоль. *Подобные приведенным ниже (включая дополнительную операцию).*

+	/tb_lab_MS_SV4/INST	DIV 10 -5	ADD 10 -5	SUB 10 -5	MUL 10 -5	DIV 10 -5
+	/tb_lab_MS_SV4/ALU_out	-2	5	15	-50	-2

```

#
# instruction=ADD op_a= 10 op_b= -5 ALU_out= 5
#
# instruction=SUB op_a= 10 op_b= -5 ALU_out= 15
#
# instruction=MUL op_a= 10 op_b= -5 ALU_out= -50
#
# instruction=DIV op_a= 10 op_b= -5 ALU_out= -2
#
#
# ** Note: $stop : C:/Intel_trn/Q_MS_SV/lab_MS_SV4/tb_lab_MS_SV4.sv(23)

```

с. По результатам моделирования в ModelSim необходимо доказать работоспособность устройства АЛУ (продемонстрировать выполнение всех операций для набора данных, выбранного вами самостоятельно).

3. Разработать модуль верхнего уровня для отладки (db_lab_MS_SV4), содержащий:
 - i. модуль lab_MS_SV4;
 - ii. модуль SP_unit (его надо создать в пакете Quartus, используя IP: ISSPE) - модуль, обеспечивающий возможность:
 1. задания входных сигналов, синхронизируемых тактовым сигналом CLK, без использования кнопок на плате;
 2. отображения результата.
4. Используя ISSPE, надо: провести анализ работы lab_MS_SV4 и доказать (зафиксировав результаты снимками экрана) работоспособность устройства АЛУ (продемонстрировать выполнение всех операций, включая дополнительную. Данные должны соответствовать данным, используемым при моделировании). Результаты теста, **подобные** приведенным ниже, должны быть представлены в отчете.

Index	Type	Alias	Name	Data	-8	-7	-6	-5	-4	-3	-2	-1	0
P[7..0]			* ALU out	-2									
S[17..16]			* ops	00b				00b					
S[15..8]			* op a	10							15	01b	10b
S[7..0]			* op b	-5							10	-50	11b

5. Создать такие настройки SignalTapII, чтобы получилась временная диаграмма, **аналогичная** приведенной ниже (данные должны соответствовать данным, используемым при моделировании), на которой должна быть отображена соответствующая дополнительная операция (в приведенном примере дополнительная операция – еще одна ADD).

Node	0	1	2	3	4
Type/alias	Name	Name	Name	Name	Name
	lab_MS_SV4:UUT[INST.opc[1..0]	lab_MS_SV4:UUT[INST.opc[1..0]	lab_MS_SV4:UUT[INST.opc[1..0]	lab_MS_SV4:UUT[INST.opc[1..0]	lab_MS_SV4:UUT[INST.opc[1..0]
	ADD	SUB	MUL	DIV	ADD
	5	15	-50	-2	5

Подсказка: нужно настроить 5 сегментов, условие захвата – любое изменение кода операции. Для отображения кода операции надо создать mnemonic таблицу. Изменением кода операции можно управлять из ISSPE.

Дополнительная операция, соответствующая Вашему номеру в списке группы

Номер в списке группы	Добавляемая операция
1	$ALU_out = op_a++$
2	$ALU_out = op_b--$
3	$ALU_out = \text{минимальное из } op_a \text{ и } op_b$
4	$ALU_out = \text{максимальное из } op_a \text{ и } op_b$
5	$ALU_out = \text{максимальное положительное число}$
6	$ALU_out = op_a \% op_b$
7	$ALU_out = (op_a - op_b) \setminus 2$
8	$ALU_out = - op_b$
9	$ALU_out = 0$
10	$ALU_out = \text{минимальное отрицательное число}$
11	$ALU_out = (op_a^{**2} - op_b^{**2}) / 2$
12	$ALU_out = op_a--$
13	$ALU_out = op_b++$
14	$ALU_out = op_a * 13 - op_b$
15	$ALU_out = (op_a * op_b) \setminus 2$
16	$ALU_out = op_b \% op_a$
17	$ALU_out = op_b^{**3}$
18	$ALU_out = (op_a + op_b) \setminus 2$
19	$ALU_out = op_a * 3 - op_b$
20	$ALU_out = op_b^{**2}$
21	$ALU_out = (op_a^{**2} + op_b^{**2}) / 2$
22	$ALU_out = op_a + op_b * 2$
23	$ALU_out = (op_a * 8 + op_b * 4) / 2$
24	$ALU_out = op_a * 5 + op_b$
25	$ALU_out = - op_a$
26	$ALU_out = op_a^{**2}$
27	$ALU_out = op_a * 2 + op_b$
28	$ALU_out = \text{максимальное из } op_a + op_a \text{ и } op_b - op_b$
29	$ALU_out = \text{минимальное из } op_a + op_a \text{ и } op_b - op_b$

Примеры

Файл с описанием пакета *lab_MS_SV4_pack.sv*

```
1 package lab_MS_SV4_pack;
2   parameter width = 8;
3   typedef enum bit[1:0] {ADD='0', SUB, MUL, DIV} opcode_t;
4   typedef bit signed [width-1:0] data_y;
5
6   typedef struct packed {
7     opcode_t opc;
8     data_y op_a;
9     data_y op_b;
10  } INST_t;
11 endpackage : lab_MS_SV4_pack
```

Файл с описанием основного модуля *lab_MS_SV4.sv*

```
1 `timescale 1ns/1ns
2
3 import Lab_MS_SV4_pack::*;
4
5 module lab_MS_SV4 (
6   input INST_t INST,
7   output data_y ALU_out
8 );
9   always_comb begin
10     case (INST.opc)
11       ADD : ALU_out = INST.op_a + INST.op_b;
12       SUB : ALU_out = INST.op_a - INST.op_b;
13       MUL : ALU_out = INST.op_a * INST.op_b;
14       DIV : ALU_out = INST.op_a / INST.op_b;
15     endcase
16   end
17 endmodule
```

Файл с описанием теста *tb_lab_MS_SV4.sv*

```
1 `timescale 1ns/1ns
2
3 import Lab_MS_SV4_pack::*;
4
5 module tb_lab_MS_SV4();
6   INST_t INST;
7   data_y ALU_out;
8
9   Lab_MS_SV4 uut (.ALU_out, .INST);
10
11   initial begin
12     INST.opc = INST.opc.first();
13     do begin
14       INST.op_a = 10;
15       INST.op_b = -5;
16       #10;
17       INST.opc = INST.opc.next();
18     end
19     while (INST.opc != INST.opc.Last());
20
21     #10;
22     $display("\n");
23     $stop;
24   end
25   initial
26     $monitor("\n \tinstruction=%p \top_a=%d \top_b=%d \tALU_out=%d",
27             INST.opc, INST.op_a, INST.op_b, ALU_out);
28 endmodule
```

Файл с описанием модуля отладки *db_lab_MS_SV4.sv*

```
1  import Lab_MS_SV4_pack::*;
2
3  module db_lab_MS_SV4
4  (
5      (* altera_attribute = "-name IO_STANDARD \"3.3-V LVC MOS\"", chip_pin = "R8" *)
6      //"23" for miniDilab-CIV
7      //"R8" for DE0_nano
8      //"N5" for MAX10 NEEK
9      input CLK
10 );
11
12     INST_t INST;
13     data_y ALU_out;
14
15     Lab_MS_SV4 UUT (.ALU_out, .INST);
16
17     SP_unit SP_ (
18         .source      (INST),      // sources.source
19         .probe       (ALU_out),   // probes.probe
20         .source_clk  (CLK)        // source_clk.clk
21     );
22 endmodule
```