

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и кибербезопасности  
Высшая школа компьютерных технологий и информационных систем

### **Отчёт по лабораторной работе № 3**

Дисциплина: Автоматизация проектирования дискретных  
устройств (на английском языке).

Выполнил студент гр. 5130901/10101 \_\_\_\_\_ Д.Л. Симоновский  
(подпись)

Руководитель \_\_\_\_\_ А.А. Федотов  
(подпись)

“21” февраля 2024 г.

Санкт-Петербург

2024

## Оглавление

1.	Список иллюстраций: .....	2
2.	Цель упражнения: .....	3
3.	Алгоритм работы проекта: .....	3
4.	Решение: .....	3
5.	Вывод: .....	11

## 1. Список иллюстраций:

Рис. 3.1. Структура разрабатываемого проекта.....	3
Рис. 4.1. RTL Viewer.....	5
Рис. 4.2. Результат моделирования. ....	6
Рис. 4.3. Настройки ISSP. ....	7
Рис. 4.4. Настройки Signal Tap II. ....	8
Рис. 4.5. Значения в ISSP.....	8
Рис. 4.6. Значения в ISSP.....	8
Рис. 4.7. Значения в ISSP.....	9
Рис. 4.8. Значения в ISSP.....	9
Рис. 4.9. Результат измерений в Signal Tap II. ....	9
Рис. 4.10. Новые настройки Signal Tap II.....	10
Рис. 4.11. Значения в ISSP.....	10
Рис. 4.12. Значения в ISSP.....	10
Рис. 4.13. Результат измерений в Signal Tap II. ....	11

## 2. Цель упражнения:

Пройти цикл проектирования в рамках пакетов Quartus и ModelSim, включая следующие этапы:

- Создание проекта.
- Разработка описания модулей с использованием конструкций расширения SystemVerilog.
- Разработка теста на языке SystemVerilog и моделирование.
- Отладка проекта.

## 3. Алгоритм работы проекта:

- Два экземпляра (cntA и cntB) 10-разрядного счетчика (cnt\_10bits) на сложение, по модулю, заданному на входах cntA\_Module и cntB\_Module, формируют:
  - текущие значения: cntA[9:0], cntB[9:0]
  - сигналы переноса (по достижению заданного модуля счета): CoutA, CoutB
- Компаратор (cmp\_eq) на равенство сравнивает значения двух счетчиков и формирует выходной сигнал cntA\_EQ\_cntB.
- Конечный автомат (fsm) анализирует сигналы переноса двух счетчиков и формирует сигналы:
  - AeqB – равное количество сигналов переноса от счетчика A и счетчика B.
  - AmB – сигналы переноса от счетчика A появляются чаще.
  - BmA – сигналы переноса от счетчика B появляются чаще.
- Сигнал асинхронного сброса (rst\_n) обеспечивает сброс всех устройств с памятью.

Структура проекта приведена на рисунке ниже:

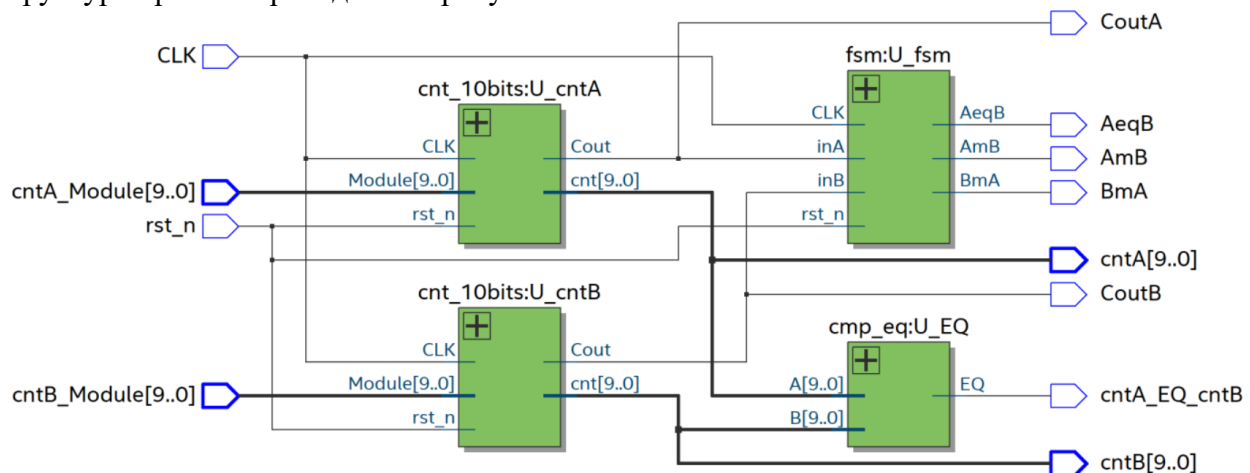


Рис. 3.1. Структура разрабатываемого проекта.

## 4. Решение:

Выполним создание первого модуля – cnt\_10bits, используя средства System Verilog:

```
1 module cnt_10bits (  
2     input bit      CLK,  
3     input bit [9:0] Module,  
4     input bit      rst_n,  
5     output bit     Cout,  
6     output bit [9:0] cnt  
7 );  
8  
9     always_ff @(posedge CLK, negedge rst_n)  
10         if (!rst_n) cnt <= 10'd0;  
11         else cnt <= cnt < (Module - 10'd1) ? cnt + 10'd1 : 10'd0;  
12  
13     assign Cout = cnt == (Module - 10'd1);  
14  
15 endmodule
```

Это счетчик делитель с делителем, подаваемым на вход Module.

Далее реализуем модуль `cmp_eq`, как комбинационную схему средствами System Verilog:

```
1 module cmp_eq (  
2     input bit [9:0] A,  
3     input bit [9:0] B,  
4     output bit EQ  
5 );  
6  
7 always_comb EQ = A == B;  
8  
9 endmodule
```

И последний модуль, необходимый для реализации – конечный автомат для фиксации, какие сигналы переноса появляются чаще – со счетчика А или В:

```
1 module fsm (  
2     input bit CLK,  
3     input bit rst_n,  
4     input bit inA,  
5     input bit inB,  
6     output bit AeqB,  
7     output bit AmB,  
8     output bit BmA  
9 );  
10  
11 enum bit [1:0] {EQ, A_, B_} states;  
12  
13 always_ff @(posedge CLK, negedge rst_n) begin  
14     if (~rst_n) states <= EQ;  
15     else  
16         case (states)  
17             EQ: if (inA & ~inB) states <= A_;  
18                 else if (~inA & inB) states <= B_;  
19             A_: if (~inA & inB) states <= EQ;  
20             B_: if (inA & ~inB) states <= EQ;  
21         endcase  
22     end  
23  
24 always_comb begin  
25     AeqB = 1'd0;  
26     AmB = 1'd0;  
27     BmA = 1'd0;  
28     case (states)  
29         EQ: AeqB = 1'd1;  
30         A_: AmB = 1'd1;  
31         B_: BmA = 1'd1;  
32     endcase  
33 end  
34  
35 endmodule
```

Теперь создадим модуль верхнего уровня, собрав все модули вместе:

```

1 module lab_MS_SV1 (
2     input bit      CLK,
3     input bit [9:0] cntA_Module,
4     input bit [9:0] cntB_Module,
5     input bit      rst_n,
6     output bit     CoutA,
7     output bit     CoutB,
8     output bit [9:0] cntA,
9     output bit [9:0] cntB,
10    output bit     cntA_EQ_cntB,
11    output bit     AeqB,
12    output bit     AmB,
13    output bit     BmA
14 );
15
16 cnt_10bits U_cntA (
17     .CLK (CLK),
18     .Module(cntA_Module),
19     .rst_n (rst_n),
20     .Cout (CoutA),
21     .cnt (cntA)
22 );
23 cnt_10bits U_cntB (
24     .CLK (CLK),
25     .Module(cntB_Module),
26     .rst_n (rst_n),
27     .Cout (CoutB),
28     .cnt (cntB)
29 );
30 cmp_eq U_EQ (
31     .A (cntA),
32     .B (cntB),
33     .EQ(cntA_EQ_cntB)
34 );
35 fsm U_fsm (
36     .CLK (CLK),
37     .rst_n(rst_n),
38     .inA (CoutA),
39     .inB (CoutB),
40     .AeqB (AeqB),
41     .AmB (AmB),
42     .BmA (BmA)
43 );
44
45 endmodule

```

Выполним полную компиляцию и проверим, что получившаяся схема соответствует поставленному заданию:

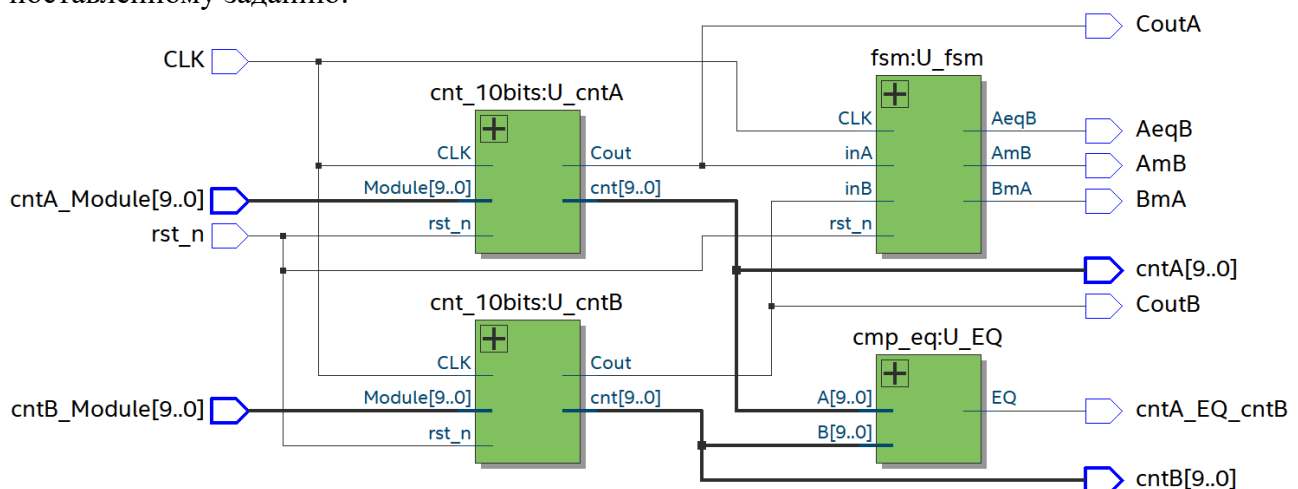


Рис. 4.1. RTL Viewer.

Как мы видим, получившаяся схема полностью идентична поставленной задаче, далее необходимо выполнить тестирование.

Разработаем тест первого уровня:

```

1  `timescale 1ns / 1ns
2  module tb_lab_MS_SV1 ();
3      bit CLK = 1'b0;
4      bit rst_n;
5      bit CAout, CBout;
6      bit AeqB, AmB, BmA;
7      bit CoutA, CoutB;
8      bit cntA_EQ_cntB;
9      bit [9:0] cntA_Module, cntB_Module;
10     bit [9:0] cntA, cntB;
11
12     lab_MS_SV1 u_lab_MS_SV1 (
13         .CLK          (CLK),
14         .cntA_Module  (cntA_Module),
15         .cntB_Module  (cntB_Module),
16         .rst_n        (rst_n),
17         .CoutA        (CoutA),
18         .CoutB        (CoutB),
19         .cntA         (cntA),
20         .cntB         (cntB),
21         .cntA_EQ_cntB (cntA_EQ_cntB),
22         .AeqB         (AeqB),
23         .AmB          (AmB),
24         .BmA          (BmA)
25     );
26
27     localparam PERIOD = 20;
28
29     initial forever #(PERIOD / 2) CLK = ~CLK;
30
31     initial begin
32         rst_n = 1'd0;
33         #(PERIOD * 4);
34         rst_n = 1'd1;
35         cntA_Module = 10'd10;
36         cntB_Module = 10'd10;
37         #(PERIOD * 28);
38
39         rst_n = 1'd0;
40         #(PERIOD * 4);
41         rst_n = 1'd1;
42         cntA_Module = 10'd5;
43         cntB_Module = 10'd10;
44         #(PERIOD * 28);
45
46         rst_n = 1'd0;
47         #(PERIOD * 4);
48         rst_n = 1'd1;
49         cntA_Module = 10'd10;
50         cntB_Module = 10'd5;
51         #(PERIOD * 28);
52
53         rst_n = 1'd0;
54         #(PERIOD * 4);
55         rst_n = 1'd1;
56         cntA_Module = 10'd5;
57         cntB_Module = 10'd5;
58         #(PERIOD * 28);
59         $stop;
60     end
61 end
62
63
64 endmodule

```

Выполним запуск этого теста, средствами Quartus и получим следующий результат:

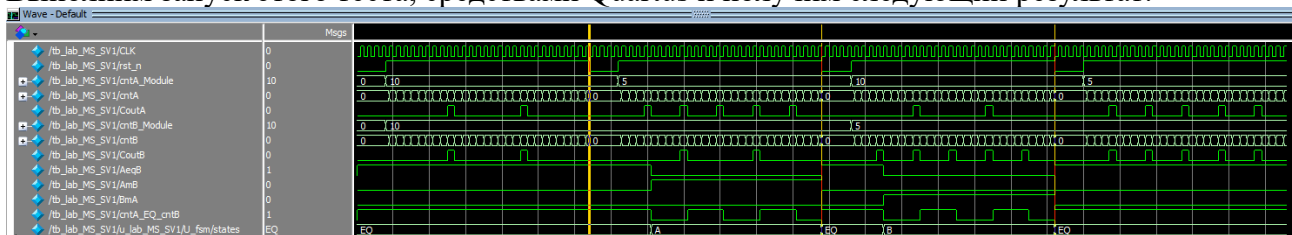


Рис. 4.2. Результат моделирования.

Видим, что результат моделирования полностью соответствует ожиданиям, тогда перейдем к моделированию на плате, для этого создадим следующий модуль верхнего уровня:

```

1 module db_lab_MS_SV1 (
2     (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMS\" chip_pin = \"23\" *)
3     input CLK
4 );
5
6 bit rst_n;
7 bit AeqB, AmB, BmA;
8 bit CoutA, CoutB;
9 bit cntA_EQ_cntB;
10 bit [9:0] cntA_Module, cntB_Module;
11 bit [9:0] cntA, cntB;
12
13 lab_MS_SV1 U1 (
14     .CLK      (CLK),
15     .cntA_Module (cntA_Module),
16     .cntB_Module (cntB_Module),
17     .rst_n     (rst_n),
18     .CoutA     (CoutA),
19     .CoutB     (CoutB),
20     .cntA      (cntA),
21     .cntB      (cntB),
22     .cntA_EQ_cntB(cntA_EQ_cntB),
23     .AeqB      (AeqB),
24     .AmB       (AmB),
25     .BmA       (BmA)
26 );
27
28 SP_unit U2 (
29     .source     ({rst_n, cntA_Module, cntB_Module}),
30     .source_clk (CLK)
31 );
32
33 endmodule

```

Добавим SDC файл, чтоб задать требования тактовому сигналу, со следующим содержанием:

```

1 create_clock -name {CLK} -period 20.000 -waveform { 0.000 10.000 } [get_ports {CLK}]
2 derive_clock_uncertainty

```

Для отладки на устройстве будем использовать ISSP, с помощью которого будем подавать сигнал сброса и значения для счетчиков и Signal Tap II, которым будем отслеживать результаты. ISSP будет иметь следующие настройки:



Рис. 4.3. Настройки ISSP.

А настройки Signal Tap II выглядят следующим образом:



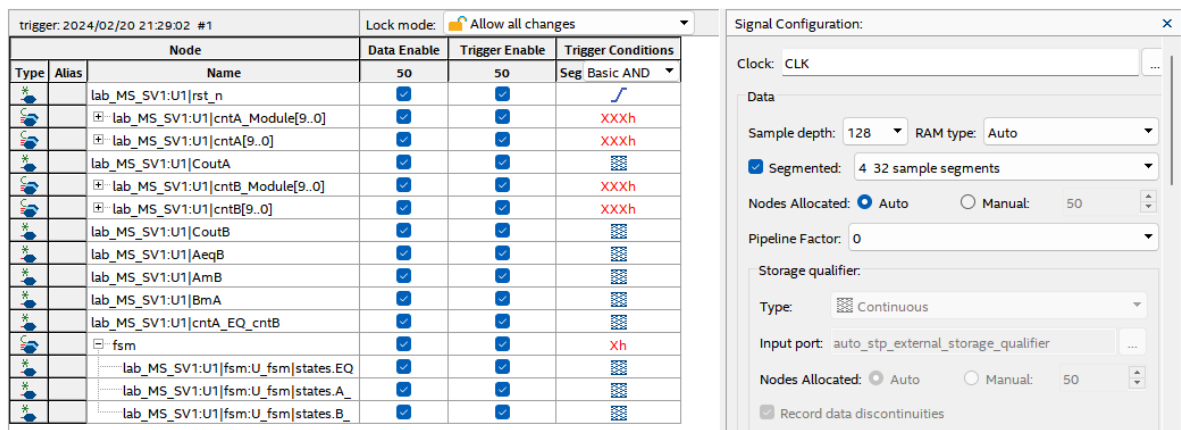


Рис. 4.4. Настройки Signal Tap II.

Как можно заметить, захват происходит по фронту сигнала сброса. Используя ISSP, запишем на устройство следующие значения:



Рис. 4.5. Значения в ISSP.

После чего включим счетчики, подав 1 на rst\_n. Теперь изменим значения в ISSP:



Рис. 4.6. Значения в ISSP.

После этого опять подадим 1 на rst\_n, тем самым включим счетчики и чтоб Signal Tap II выполнил захват данных. Опять поменяем значения в ISSP:



*Рис. 4.7. Значения в ISSP.*

Опять переключаем значение `rst_n` на 1, после чего повторяем изменение данных в ISSP:



*Рис. 4.8. Значения в ISSP.*

После чего в 4 раз переключаем `rst_n` на 1, тем самым получив 4 сегмента на Signal Tap II:



*Рис. 4.9. Результат измерений в Signal Tap II.*

Как можно заметить, *Рис. 4.2* и *Рис. 4.9* полностью совпадают, что свидетельствует о корректности работы устройства на ПЛИС.

После этого поменяем настройки Signal Tap II следующим образом:

trigger: 2024/02/21 07:14:52 #0

Lock mode: Allow all changes

Type	Alias	Node	Data Enable	Trigger Enable	Trigger Conditions
		lab_MS_SV1:U1 rst_n	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Seg Basic AND
		lab_MS_SV1:U1 cntA_Module[9..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXXXXXXXb
		lab_MS_SV1:U1 cntA[9..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXXXXXXXb
		lab_MS_SV1:U1 CoutA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		lab_MS_SV1:U1 cntB_Module[9..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXXXXXXXb
		lab_MS_SV1:U1 cntB[9..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXXXXXXXb
		lab_MS_SV1:U1 CoutB	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		lab_MS_SV1:U1 AeqB	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		lab_MS_SV1:U1 AmB	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		lab_MS_SV1:U1 BmA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		lab_MS_SV1:U1 cntA_EQ_cntB	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		fsm	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Xh
		lab_MS_SV1:U1 fsm:U_fsm states.EQ	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		lab_MS_SV1:U1 fsm:U_fsm states.A	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		lab_MS_SV1:U1 fsm:U_fsm states.B	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Signal Configuration:

Clock: CLK

Data

Sample depth: 128 RAM type: Auto

☒ Segmented: 2 64 sample segments

Nodes Allocated: ☒ Auto ☐ Manual: 50

Pipeline Factor: 0

Storage qualifier:

Type: Continuous

Input port: auto\_stp\_external\_storage\_qualifier

Nodes Allocated: ☒ Auto ☐ Manual: 50

☒ Record data discontinuities

Рис. 4.10. Новые настройки Signal Tap II.

Таким образом будет выполняться захват 2 сегментов, вместо 4, как было ранее.

Запишем этот проект на плату и используя ISSP подадим следующие значения (в соответствии с вариантом 20 и 40):

Рис. 4.11. Значения в ISSP.

Включаем устройство, переключением rst\_n в единицу, после чего опять меняем значения в ISSP:

Рис. 4.12. Значения в ISSP.

На Signal Tap II получаем следующую диаграмму:

10



*Рис. 4.13. Результат измерений в Signal Tap II.*

Как мы видим, она полностью соответствует ожиданиям. Сначала при подаче комбинации 40; 20 мы получаем В, а после изменения их местами, получаем А.

## **5. Вывод:**

В ходе лабораторной работы успешно пройден цикл проектирования, начиная с создания проекта и разработки модулей с использованием расширений SystemVerilog. Использование SystemVerilog предоставило широкий спектр новых возможностей по сравнению с Verilog, облегчая процесс разработки и улучшая читаемость кода.

Отладка проекта осуществлялась с помощью инструментов In-System Sources and Probes Editor и SignalTap II, что значительно повысило эффективность процесса. Эти инструменты позволили быстро выявить и исправить ошибки, что является ключевым аспектом при работе с любым проектом.