

lab_MS_SV1

Цель упражнения

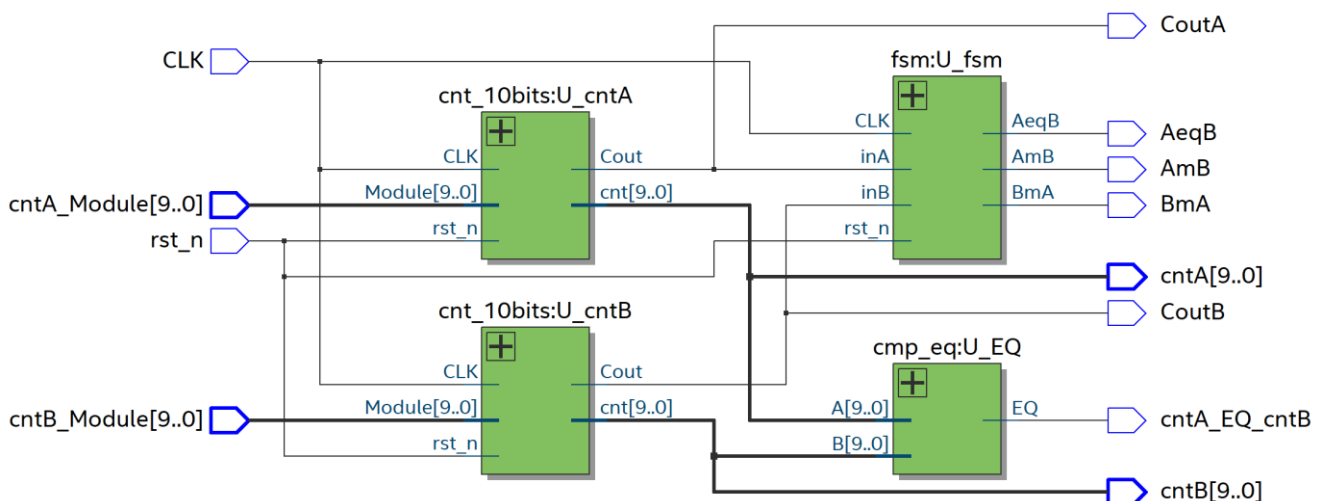
Пройти цикл проектирования в рамках пакетов Quartus и ModelSim, включая следующие этапы:

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

Алгоритм работы проекта

- Два экземпляра(cntA и cntB) 10-ти 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



Часть 1 – Создание проекта

1. Рабочая папка: **C:\Intel_trn\Q_MS_SV\lab_MS_SV1**
2. Имя проекта: **lab_MS_SV1**
3. Имя модуля верхнего уровня: **lab_MS_SV1**
4. Микросхема:
 - **Cyclone IV E**
 - **EP4CE6E22C8** – для платы **MiniDilab-CIV**
 - **EP4CE22F17C6** – для платы **DE0-nano**
 - **MAX10**
 - **10M50DAF484C6GES** – для платы **MAX10 NEEK**
5. Внешнее средство проектирования: **ModelSim Altera, SystemVerilog**

Часть 2 – Разработка описания модулей на языке SystemVerilog

1. На языке SystemVerilog (максимально используя конструкции расширения SystemVerilog) создайте описание 10 разрядного двоичного счетчика - **cnt_10bits** (файл - **cnt_10bits.sv**)

- модуль счета задается на входе Module
- счет на сложение до значения Module-1
- вход асинхронного сброса rst_n – активный уровень 0
- выход счетчика - [9:0] cnt
- выход переноса Cout – формируется логическая единица, когда значение на выходе счетчика = Module-1.

Пример описания счетчика приведен ниже

```

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  always_ff @(posedge CLK, negedge rst_n)
9      if (!rst_n)
10         cnt <= 10'd0;
11     else
12         if (cnt < (Module- 10'd1))
13             cnt <= cnt + 10'd1;
14         else
15             cnt <= 10'd0;
16  assign Cout = cnt==(Module - 10'd1);
17  endmodule

```

2. На языке SystemVerilog (максимально используя конструкции расширения SystemVerilog) создайте описание комбинационного компаратора - **cmp_eq** (файл - **cmp_eq.sv**)

- Компаратор на равенство двух 10 разрядных
- Выход EQ – равен 1 при равенстве двух входных чисел.

Пример описания компаратора приведен ниже

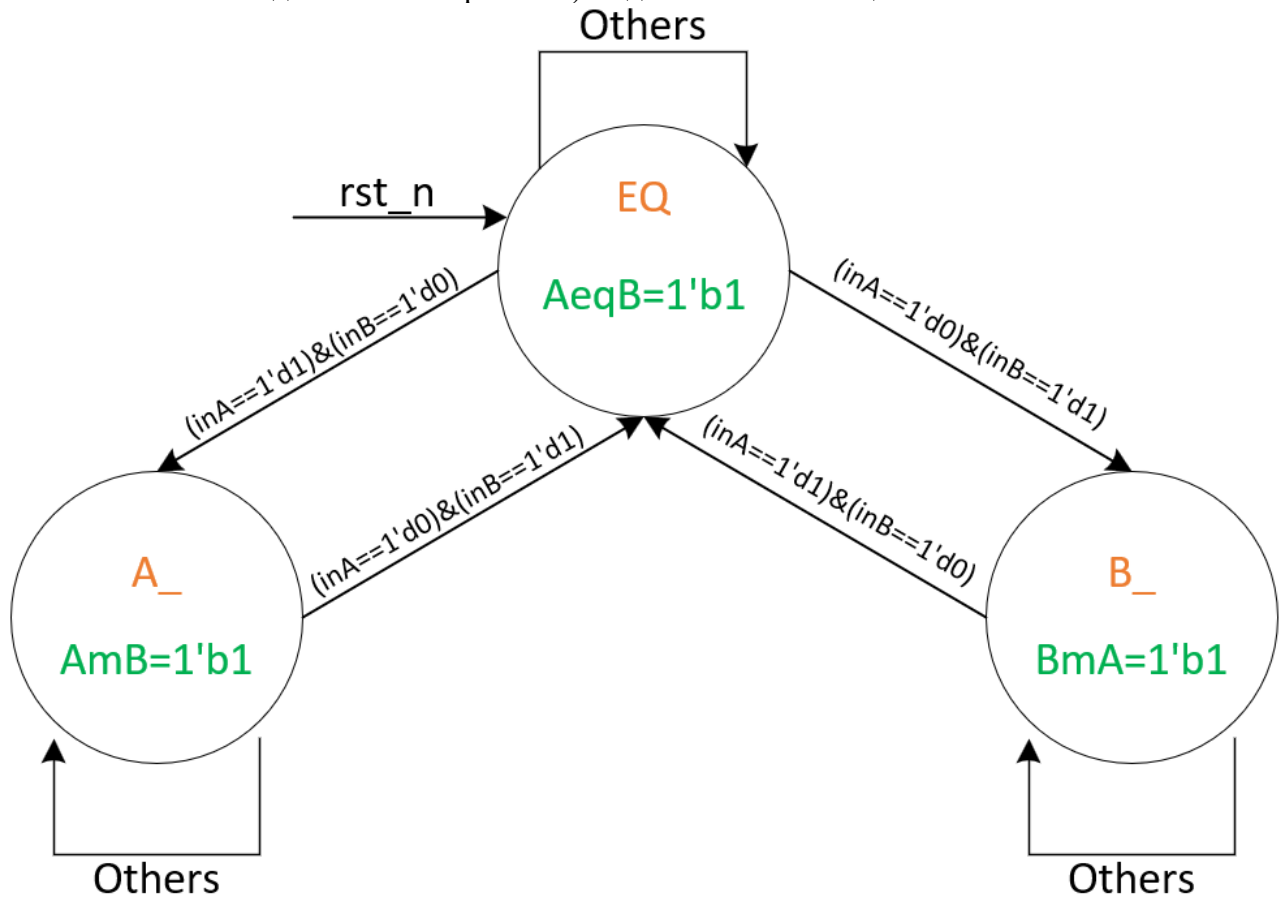
```

1  module cmp_eq (
2      input bit [9:0] A,
3      input bit [9:0] B,
4      output bit EQ
5  );
6  always_comb
7      if (A==B)
8          EQ <= 1'd1;
9      else
10         EQ <= 1'd0;
11  endmodule

```

3. На языке SystemVerilog (максимально используя конструкции расширения SystemVerilog) создайте описание конечного автомата - fsm (файл - fsm.sv)

- Конечный автомат (fsm) анализирует сигналы переноса двух счетчиков и формирует сигналы:
 - i. AeqB – равное количество сигналов переноса от счетчика А и счетчика В.
 - ii. AmB – сигналы переноса от счетчика А появляются чаще.
 - iii. BmA – сигналы переноса от счетчика В появляются чаще.
- Граф переходов приведен ниже
 - i. Состояния выделены оранжевым цветом
 - ii. Выходные сигналы (в тех состояниях где они равны 1, в остальных состояниях они должны быть равны 0) выделены зеленым цветом.



Пример описания автомата приведен ниже

```

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     typedef enum bit[1:0] {EQ, A_, B_} fsm_states;
11     fsm_states states;
12     always_ff @(posedge CLK, negedge rst_n)
13         if(~rst_n)
14             states = EQ;
15         else
16             case (states)
17                 EQ      :   if      ((inA==1'd1)&(inB==1'd0)) states<= A_;
18                             else if ((inA==1'd0)&(inB==1'd1)) states<= B_;
19                 A_      :   if      ((inA==1'd0)&(inB==1'd1)) states<= EQ;
20                 B_      :   if      ((inA==1'd1)&(inB==1'd0)) states<= EQ;
21             endcase
22     always_comb begin:outTable
23         AeqB    = 1'd0;
24         AmB     = 1'd0;
25         BmA     = 1'd0;
26         case (states)
27             EQ      :   AeqB    = 1'd1;
28             A_      :   AmB     = 1'd1;
29             B_      :   BmA     = 1'd1;
30         endcase
31     end : outTable
32 endmodule

```

4. На языке SystemVerilog (максимально используя конструкции расширения SystemVerilog) создайте описание модуля верхнего уровня - **lab_MS_SV1** (файл - **lab_MS_SV1.sv**)

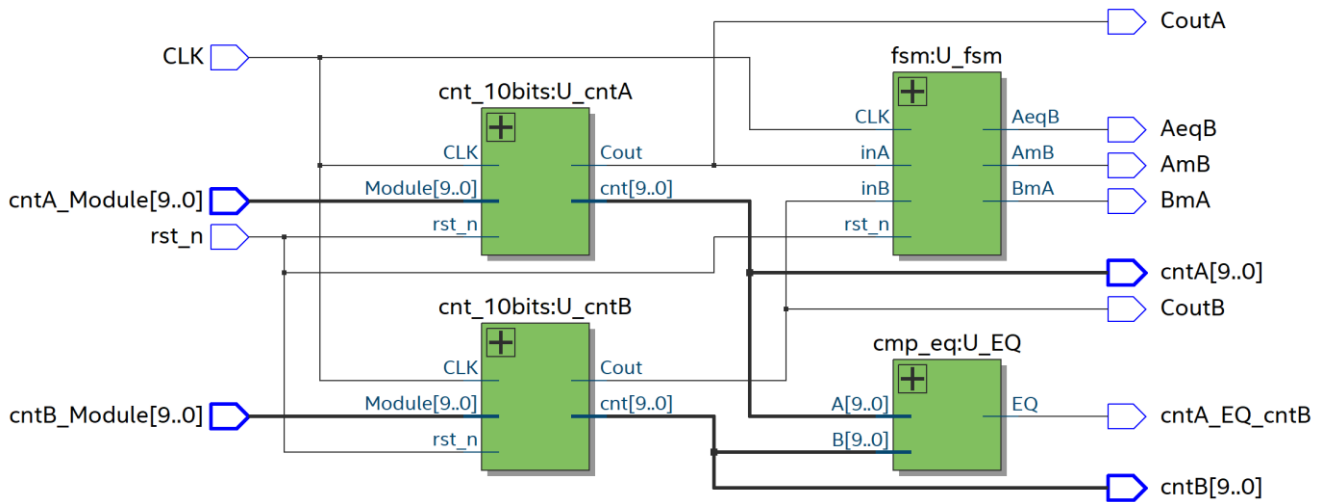
Пример описания модуля верхнего уровня приведен ниже

```

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, AmB, BmA
12 );
13 cnt_10bits U_cntA(
14     .CLK      (CLK),
15     .Module   (cntA_Module),
16     .rst_n    (rst_n),
17     .Cout     (CoutA),
18     .cnt      (cntA)
19 );
20 cnt_10bits U_cntB(
21     .CLK      (CLK),
22     .Module   (cntB_Module),
23     .rst_n    (rst_n),
24     .Cout     (CoutB),
25     .cnt      (cntB)
26 );
27 cmp_eq      U_EQ(
28     .A        (cntA),
29     .B        (cntB),
30     .EQ       (cntA_EQ_cntB)
31 );
32 fsm         U_fsm(
33     .CLK      (CLK),
34     .rst_n    (rst_n),
35     .inA      (CoutA),
36     .inB      (CoutB),
37     .AeqB     (AeqB),
38     .AmB      (AmB),
39     .BmA      (BmA)
40 );
41 endmodule

```

5. Задайте файл lab_MS_SV1.sv файлом верхнего уровня в иерархии описаний.
6. Выполните компиляцию в режиме Analysis & Synthesis.
7. Компиляция должна завершиться без ошибок и предупреждений.
8. При появлении ошибок и предупреждений – исправьте созданные описания блоков.
9. С помощью RTLViewer проверьте структуру проекта, она должна соответствовать структуре, приведенной ниже



Часть 3 - Разработка теста на языке SystemVerilog и моделирование

1. На языке SystemVerilog (максимально используя конструкции расширения SystemVerilog) создайте описание теста **-tb_lab_MS_SV1** (файл – **tb_lab_MS_SV1.sv**)

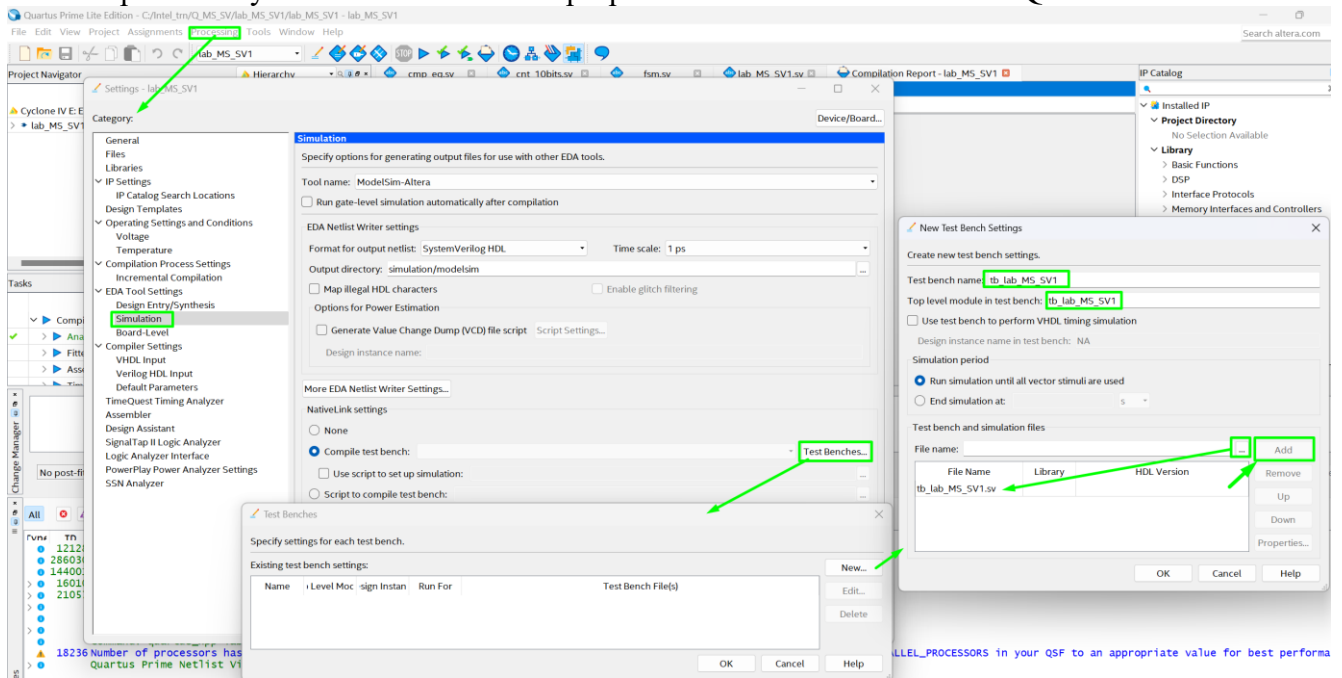
Пример описания теста приведен ниже

```

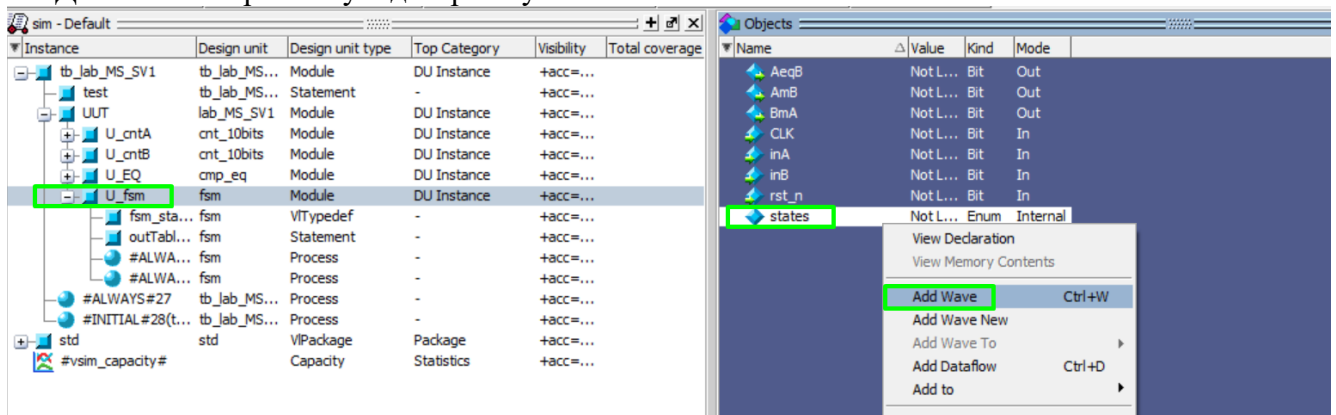
1  `timescale 1ns/1ns
2  module tb_lab_MS_SV1 ();
3      bit CLK;
4      bit rst_n;
5      bit CAout, CBout;
6      bit AeqB, AmB, BmA;
7      bit CouA, CoutB;
8      bit cntA_EQ_cntB;
9      bit [9:0] cntA_Module, cntB_Module;
10     bit [9:0] cntA, cntB;
11     Lab_MS_SV1 UUT(
12         .CLK          (CLK),
13         .cntA_Module  (cntA_Module),
14         .cntB_Module  (cntB_Module),
15         .rst_n        (rst_n),
16         .CoutA        (CoutA),
17         .CoutB        (CoutB),
18         .cntA         (cntA),
19         .cntB         (cntB),
20         .cntA_EQ_cntB (cntA_EQ_cntB),
21         .AeqB         (AeqB),
22         .AmB          (AmB),
23         .BmA          (BmA)
24     );
25     always #10 CLK = ~CLK;
26     initial begin : test
27         rst_n    = 1'd0;
28         #(20*4);
29         rst_n    = 1'd1;
30         cntA_Module = 10'd10;
31         cntB_Module = 10'd10;
32         #(28*20);
33
34         rst_n    = 1'd0;
35         #(20*4);
36         rst_n    = 1'd1;
37         cntA_Module = 10'd5;
38         cntB_Module = 10'd10;
39         #(28*20);
40
41         rst_n    = 1'd0;
42         #(20*4);
43         rst_n    = 1'd1;
44         cntA_Module = 10'd10;
45         cntB_Module = 10'd5;
46         #(28*20);
47
48         rst_n    = 1'd0;
49         #(20*4);
50         rst_n    = 1'd1;
51         cntA_Module = 10'd5;
52         cntB_Module = 10'd5;
53         #(28*20);
54         $stop;
55     end : test
56 endmodule

```

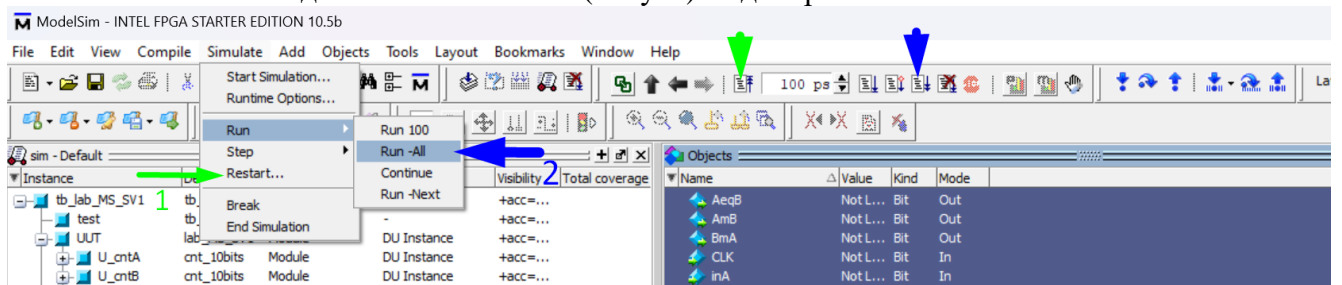
2. Настройте запуск пакета ModelSim с разработанным тестом из пакета Quartus



3. Выполните компиляцию в режиме Analysis & Synthesis.
4. Компиляция должна завершиться без ошибок и предупреждений
5. Выполните команду Tools=>RTL Simulation Tool => RTL Simulation
6. Запуск ModelSim, загрузка проекта для моделирования и моделирование должны пройти без ошибок. При возникновении ошибок их необходимо исправить.
7. Добавьте на временную диаграмму конечный автомат



8. Выполните команды Restart и Run -all (Запуск) моделирования

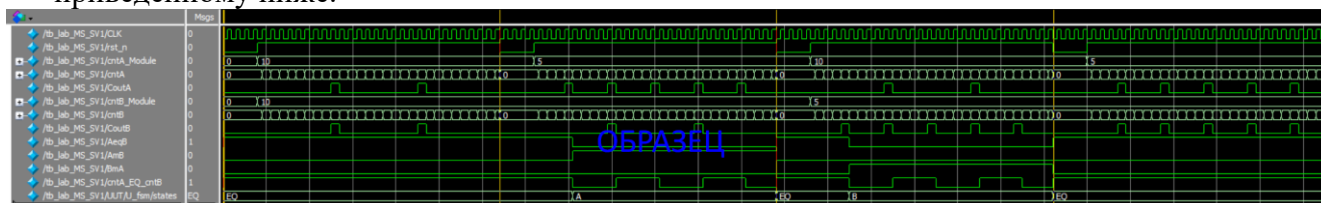


9. В окне временных диаграмм установите для указанных шин систему счисления Unsigned

	Msgs
/tb_lab_MS_SV1/CLK	0
/tb_lab_MS_SV1/rst_n	0
+ /tb_lab_MS_SV1/cntA_Module	0
+ /tb_lab_MS_SV1/cntA	0
/tb_lab_MS_SV1/CoutA	0
+ /tb_lab_MS_SV1/cntB_Module	0
+ /tb_lab_MS_SV1/cntB	0
/tb_lab_MS_SV1/CoutB	0
/tb_lab_MS_SV1/AeqB	1
/tb_lab_MS_SV1/AmB	0
/tb_lab_MS_SV1/BmA	0
/tb_lab_MS_SV1/cntA_EQ_cntB	1
/tb_lab_MS_SV1/UUT/U_fsm/states	-No ...

ОБРАТИТЕ ВНИМАНИЕ: на вашей временной диаграмме порядок сигналов **должен** соответствовать приведенному выше.

10. **Проверьте:** полученная у Вас временная диаграмма должна соответствовать образцу, приведенному ниже.



ЗАДАНИЕ: проверьте правильность работы устройства. И поясните преподавателю полученную временную диаграмму.

Проверьте:

- Значения счетчиков в моменты появления сигналов переноса
- Правильность формирования состояний конечного автомата (сравните с сигналами *AeqB* *AmB* *BmA* и графом переходов автомата)
- Правильность формирования сигнала *cntA_EQ_cntB*

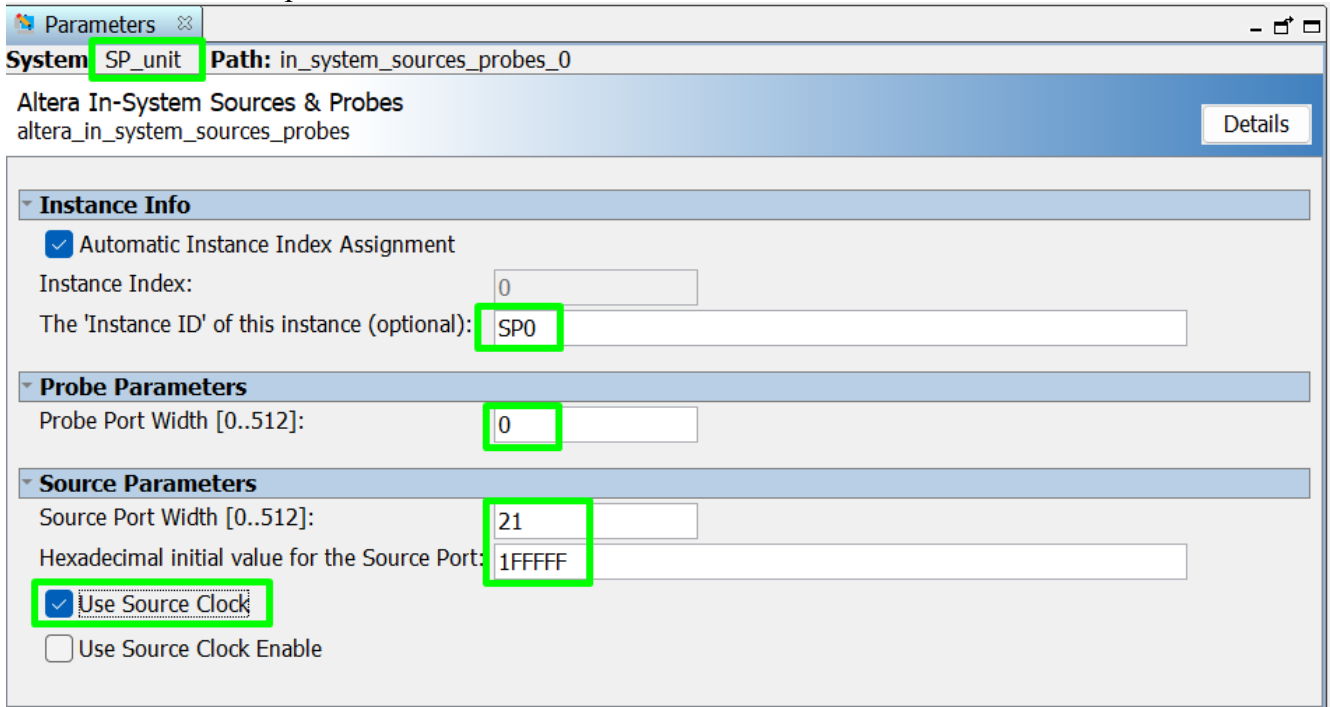
ЗАДАНИЕ: сохраните полученную временную диаграмму.

11. Сохраните временную диаграмму – файл:

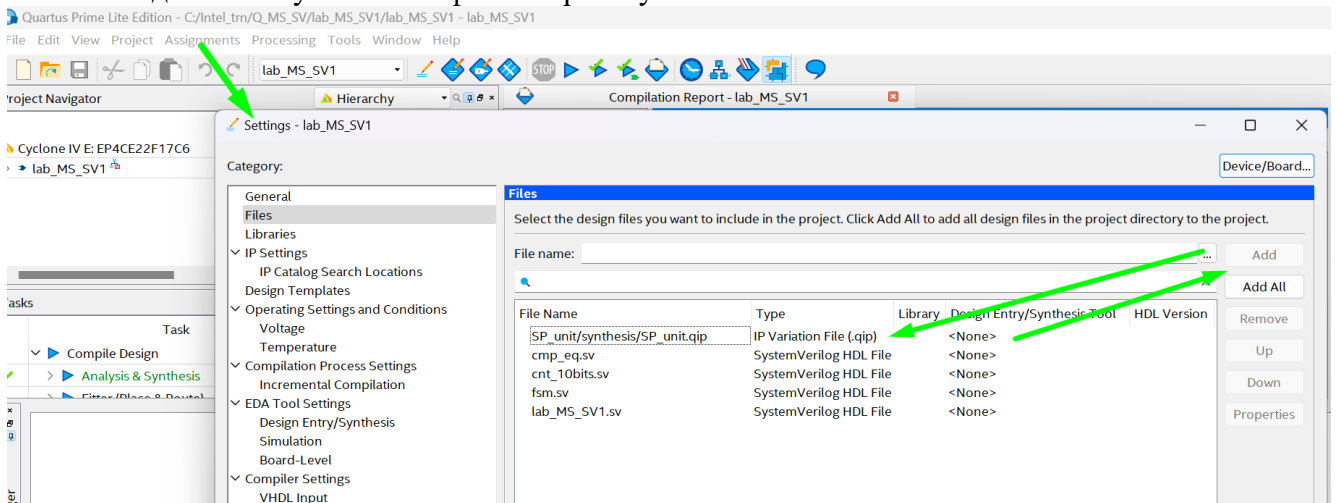
C:/Intel_trn/Q_MS_SV/lab_MS_SV1/simulation/modelsim/lab.do

Часть 4 - Отладка проекта

1. Для отладки проекта следует создать и настроить экземпляр модуля ISSPE
 - 1.1. В окне IP Catalog (если его нет на экране, то нажмите Alt+7), в поле поиска введите PROBE, найдите и дважды щелкните **Altera In System Source and Probe**.
 - 1.2. В появившемся окне задайте имя создаваемого экземпляра - **SP_unit**.
 - 1.2.1. Нажмите кнопку ОК.
 - 1.3. Задайте настройки так, как показано ниже



- 1.4. Нажмите кнопку Finish, далее Close, далее Yes, далее Generate.
- 1.5. Далее нажмите Finish и затем Close.
- 1.6. Появится окно с напоминанием о том, что надо подключить к проекту файл C:\Intel_trn\Q_MS_SV\lab_MS_SV1\SP_unit\synthesis\ **SP_unit.qip**
- 1.7. Подключите указанный файл к проекту



2. Создайте описание верхнего уровня для отладки проекта на плате

2.1. Модуль – **db_lab_MS_SV1** (файл **db_lab_MS_SV1.sv**)

Пример описания модуля приведен ниже (вывод тактового сигнала задайте в соответствии с той платой, которая будет использована в лаборатории)

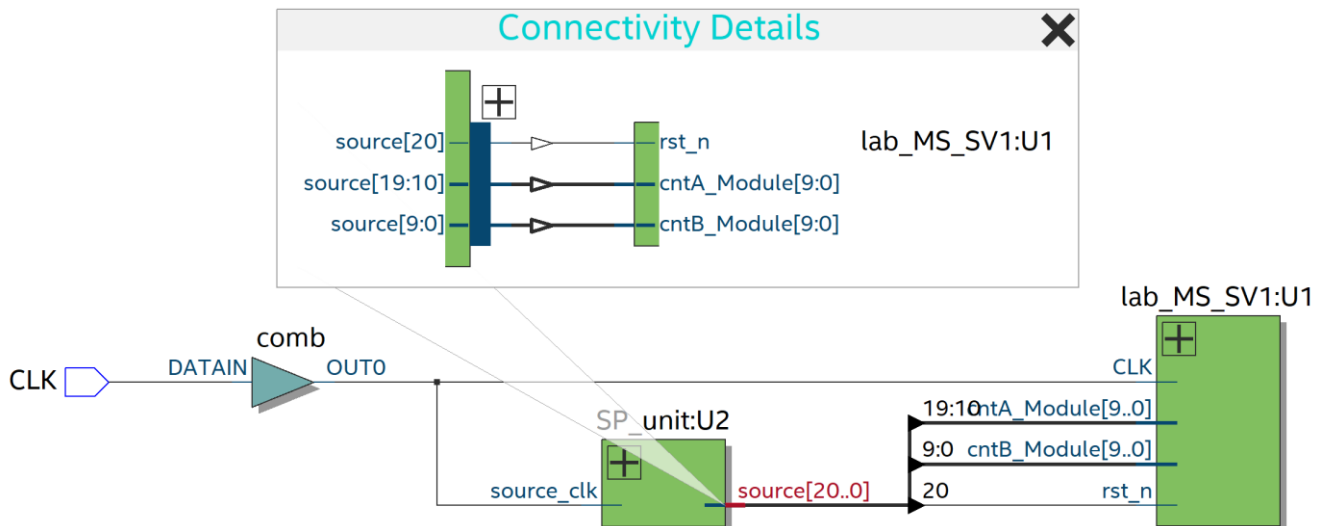
```

1  module db_lab_MS_SV1 (
2      (* altera_attribute = "-name IO_STANDARD \"3.3-V LVC MOS\"", chip_pin = "R8" *)
3      //"23" for miniDilab-CIV
4      //"R8" for DE0_nano
5      //"N5" for MAX10 NEEK
6      input CLK
7  );
8      bit rst_n;
9      bit AeqB, AmB, BmA;
10     bit CoutA, CoutB;
11     bit cntA_EQ_cntB;
12     bit [9:0] cntA_Module, cntB_Module;
13     bit [9:0] cntA, cntB;
14     Lab_MS_SV1 U1 (
15         .CLK          (CLK),
16         .cntA_Module  (cntA_Module),
17         .cntB_Module  (cntB_Module),
18         .rst_n        (rst_n),
19         .CoutA         (CoutA),
20         .CoutB         (CoutB),
21         .cntA          (cntA),
22         .cntB          (cntB),
23         .cntA_EQ_cntB  (cntA_EQ_cntB),
24         .AeqB          (AeqB),
25         .AmB           (AmB),
26         .BmA           (BmA)
27     );
28     SP_unit U2 (
29         .source        ({rst_n, cntA_Module, cntB_Module} ),
30         .source_clk     (CLK
31     );
32     endmodule

```

3. Задайте файл **db_lab_MS_SV1.sv** файлом верхнего уровня в иерархии описаний.
4. Выполните компиляцию в режиме Analysis & Synthesis.
5. Компиляция должна завершиться без ошибок и предупреждений.
6. При появлении ошибок и предупреждений – исправьте созданные описания блоков.

7. С помощью RTLViewer проверьте структуру проекта, она должна соответствовать структуре, приведенной ниже



8. В рабочей папке проекта создайте файл с требованиями к тактовому сигналу – файл **db_lab_MS_SV1.sdc**
- 8.1. Команда File=>New=> Synopsys Design Constraints File
 - 8.2. Задайте в нем требования к тактовому сигналу.

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

9. Запустите полную компиляцию проекта.
- 9.1. Компиляция должна завершиться без ошибок
 - 9.2. Проверьте выполнение требований к временным параметрам

Compilation Report - lab_MS_SV1

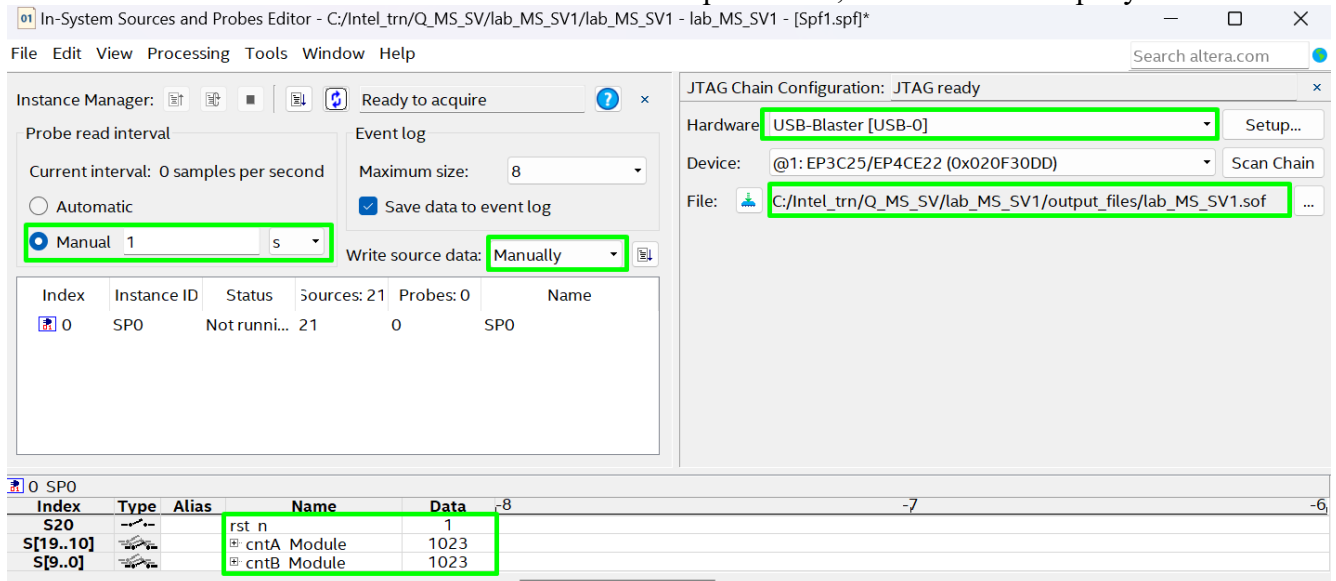
Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- > Analysis & Synthesis
- > Fitter
- > Assembler
- > TimeQuest Timing Analyzer
 - Summary
 - Parallel Compilation
 - SDC File List
 - Clocks
 - Slow 1200mV 85C Model
 - Fmax Summary
 - Timing Closure Recommendations

Slow 1200mV 85C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name
1	135.83 MHz	135.83 MHz	altera_reserved_tck
2	1113.59 MHz	250.0 MHz	CLK

10. Запустите приложение Programmer (команда Tools=>Programmer)
 - 10.1. Сконфигурируйте FPGA на плате.
 - 10.2. Закройте приложение Programmer (без сохранения настроек)
11. Запустите ISSPE (команда Tools=>In System Source and Probe Editor)
 - 11.1. В появившемся окне выполните настройки так, как показано на рисунке ниже.



- 11.2. Сохраните файл под именем **lab.spf** в папке **C:\Intel_trn\Q_MS_SV\lab_MS_SV1**

12. Запустите и настройте SignalTapII (выполните команду Tools=>SignalTapII Logic Analyzer)

12.1. Для анализа выберите указанные на рисунке цепи проекта

Node Finder

Named: *

Options

Filter: **SignalTap II: pre-synthesis** Customize...

Look in: |db_lab_MS_SV1| ... ☒ Include subentities ☒ Hierarchy view















Matching Nodes:

Name	Assignments
db_lab_MS_SV1	
CLK	Pin_R8
lab_MS_SV1:U1	
AeqB	Unassigned
AmB	Unassigned
BmA	Unassigned
CLK	Unassigned
CoutA	Unassigned
CoutB	Unassigned
cntA_EQ_cntB	Unassigned
rst_n	Unassigned
cmp_eq:U_EQ	
cnt_10bits:U_cntA	
cnt_10bits:U_cntB	
fsm:U_fsm	
AeqB	Unassigned
AmB	Unassigned
BmA	Unassigned
CLK	Unassigned
inA	Unassigned
inB	Unassigned
rst_n	Unassigned
states.A_	Unassigned
states.B_	Unassigned
states.EQ	Unassigned
cntA	Unassigned
cntA_Module	Unassigned
cntB	Unassigned
cntB_Module	Unassigned
SP_unit:U2	
source_clk	Unassigned
altsource_probe_t...sources_probes_0	
source	Unassigned




Nodes Found:

Name	Assignments
lab_MS_SV1:U1 rst_n	Unassigned
lab_MS_SV1:U1 cntA_Module	Unassigned
lab_MS_SV1:U1 cntA	Unassigned
lab_MS_SV1:U1 CoutA	Unassigned
lab_MS_SV1:U1 cntB_Module	Unassigned
lab_MS_SV1:U1 cntB	Unassigned
lab_MS_SV1:U1 CoutB	Unassigned
lab_MS_SV1:U1 AeqB	Unassigned
lab_MS_SV1:U1 AmB	Unassigned
lab_MS_SV1:U1 BmA	Unassigned
lab_MS_SV1:U1 cntA_EQ_cntB	Unassigned
lab_MS_SV1:U1 fsm:U_fsm states.A_	Unassigned
lab_MS_SV1:U1 fsm:U_fsm states.B_	Unassigned
lab_MS_SV1:U1 fsm:U_fsm states.EQ	Unassigned






- 12.2. Переместите цепи так, чтобы порядок цепей соответствовал порядку, приведенному на рисунке ниже

Node			
Type	Alias	Name	
		lab MS SV1:U1 rst n	
		lab MS SV1:U1 cntA Module[9..0]	
		lab MS SV1:U1 cntA[9..0]	
		lab MS SV1:U1 CoutA	
		lab MS SV1:U1 cntB Module[9..0]	
		lab MS SV1:U1 cntB[9..0]	
		lab MS SV1:U1 CoutB	
		lab MS SV1:U1 AeqB	
		lab MS SV1:U1 AmB	
		lab MS SV1:U1 BmA	
		lab MS SV1:U1 cntA EQ cntB	
		lab MS SV1:U1 fsm:U fsm states.EQ	
		lab MS SV1:U1 fsm:U fsm states.A	
		lab MS SV1:U1 fsm:U fsm states.B	

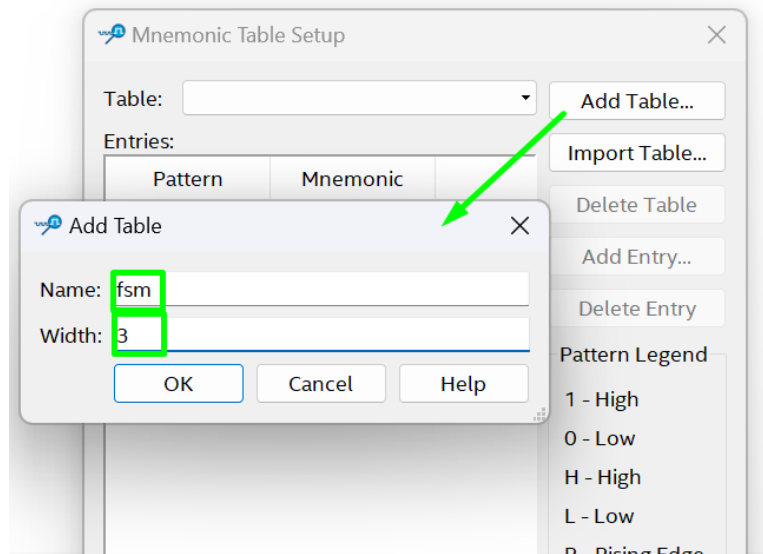
- 12.3. Сгруппируйте указанные цепи

		lab MS SV1:U1 fsm:U fsm states.EQ
		lab MS SV1:U1 fsm:U fsm states.A
		lab MS SV1:U1 fsm:U fsm states.B

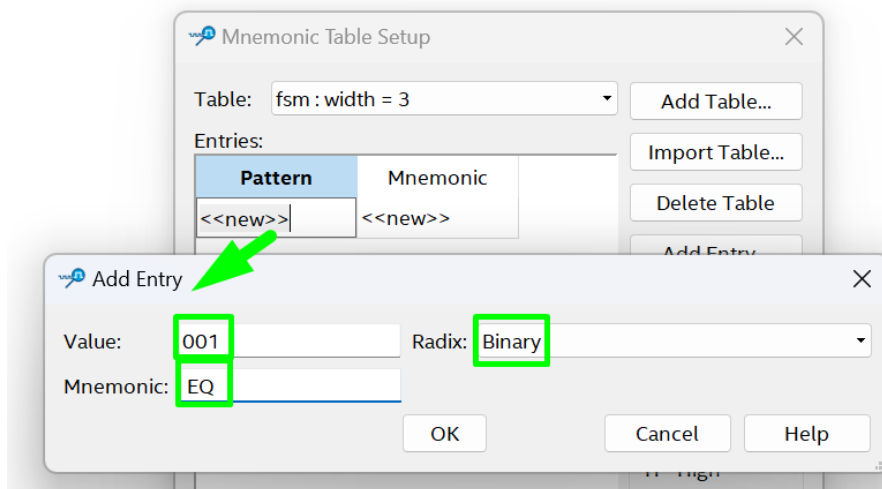
- 12.4. Группу назовите fsm

		 fsm
		lab MS SV1:U1 fsm:U fsm states.EQ
		lab MS SV1:U1 fsm:U fsm states.A
		lab MS SV1:U1 fsm:U fsm states.B

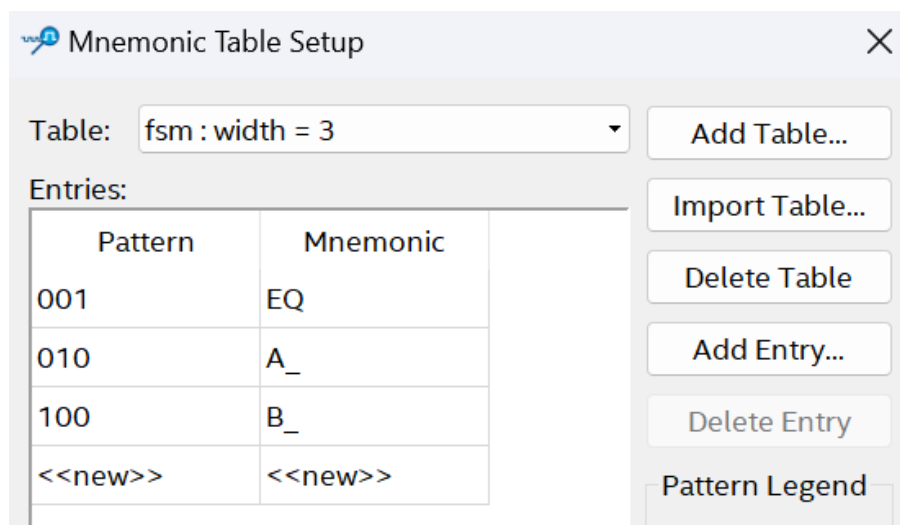
- 12.5. Щелкните правой клавишей мыши в поле под заданными цепями и в появившемся контекстно-зависимом меню выполните команду **Mnemonic Table Setup**
- 12.6. В окне Mnemonic Table Setup задайте мнемоническую таблицу для конечного автомата
- 12.6.1. Имя таблицы fsm, разрядность данных 3.



- 12.6.2. Задайте обозначения для состояний конечного автомата



- 12.6.3. Общий вид таблицы после заполнения



12.7. Задайте указанные на рисунке ниже настройки Логического анализатора

Signal Configuration:

Clock: CLK

Data

Sample depth: 128 RAM type: Auto

☒ Segmented: 4 32 sample segments

Nodes Allocated: ☒ Auto ☐ Manual: 50

Pipeline Factor: 0

Storage qualifier:

Type: Continuous

Input port:

Nodes Allocated: ☒ Auto ☐ Manual: 50

☒ Record data discontinuities

☐ Disable storage qualifier

Trigger

Nodes Allocated: ☒ Auto ☐ Manual: 50

Trigger flow control: Sequential

Trigger position: Pre trigger position

Trigger conditions: 1

☐ Trigger in

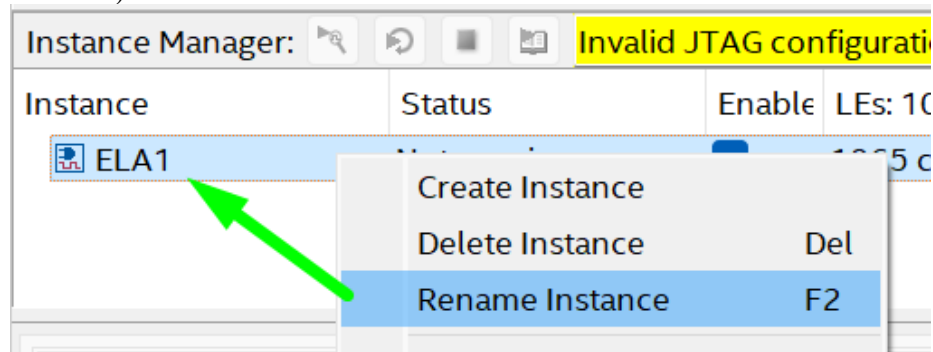
☐ Pin:

☒ Node:

12.8. Настройте условие останова логического анализатора так, как показано на рисунке ниже

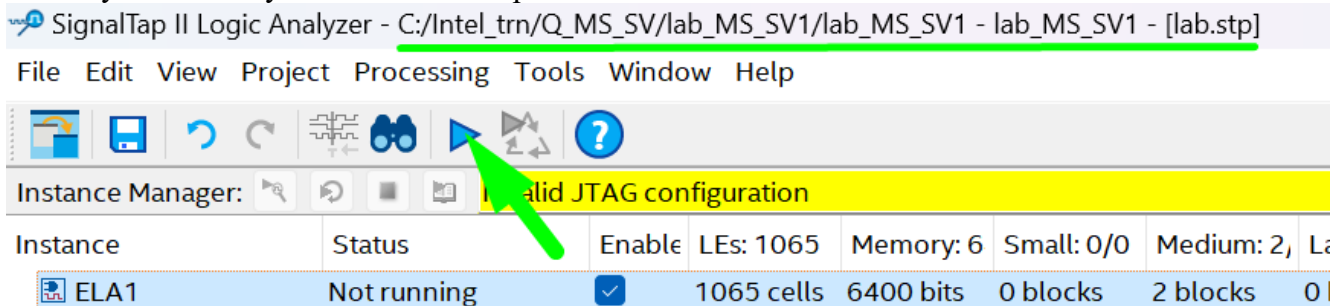
Type	Alias	Node Name	Data Enable	Trigger Enable	Trigger Conditions
		lab MS SV1:U1 rst_n	50	50	Se Basic AND
		lab MS SV1:U1 cntA Module[9..0]	50	50	XXXh
		lab MS SV1:U1 cntA[9..0]	50	50	XXXh
		lab MS SV1:U1 CntA	50	50	
		lab MS SV1:U1 cntB Module[9..0]	50	50	XXXh
		lab MS SV1:U1 cntB[9..0]	50	50	XXXh
		lab MS SV1:U1 CntB	50	50	
		lab MS SV1:U1 AeqB	50	50	
		lab MS SV1:U1 AmB	50	50	
		lab MS SV1:U1 BmA	50	50	
		lab MS SV1:U1 cntA EQ cntB	50	50	
		fsm	50	50	Xh
		lab MS SV1:U1 fsm:U fsm states.EQ	50	50	
		lab MS SV1:U1 fsm:U fsm states.A	50	50	
		lab MS SV1:U1 fsm:U fsm states.B	50	50	

- 12.9. Задайте имя экземпляру логического анализатора ELA1 (используя контекстно-зависимое меню)

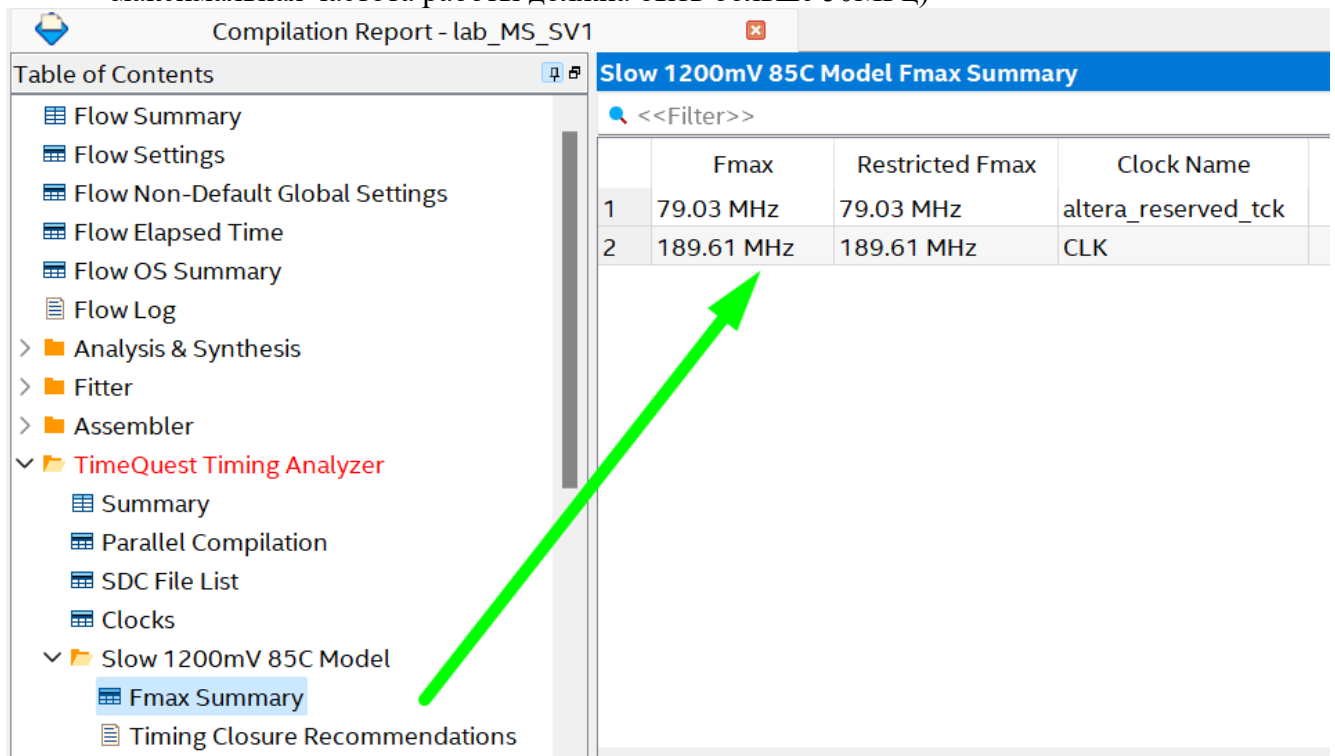


- 12.10. Сохраните файл с настройками логического анализатора под именем lab.stp в папке C:\Intel_trn\Q_MS_SV\lab_MS_SV1 и согласитесь с предложением подключить файл с логическим анализатором к текущему проекту

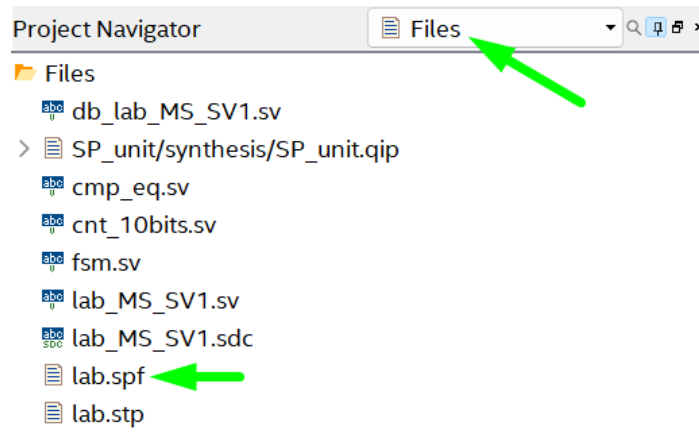
13. Запустите полную компиляцию проекта



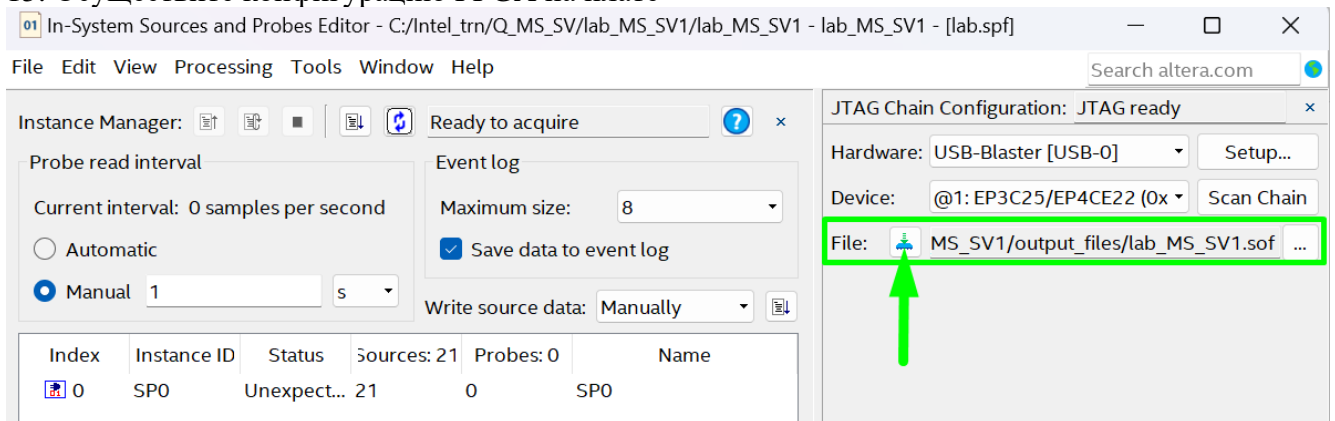
- 13.1. Проверьте выполнение требований к временным параметрам (полученная максимальная частота работы должна быть больше 50МГц)



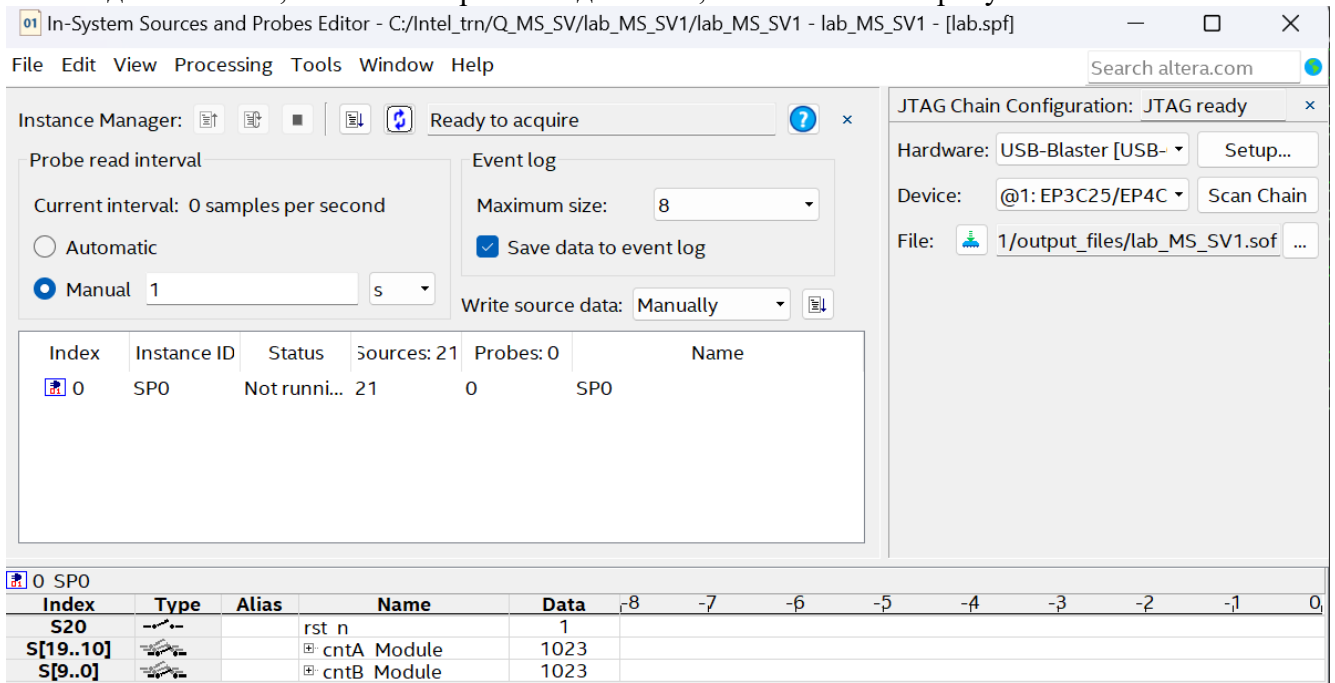
14. Откройте (если он был закрыт) файл lab.spf



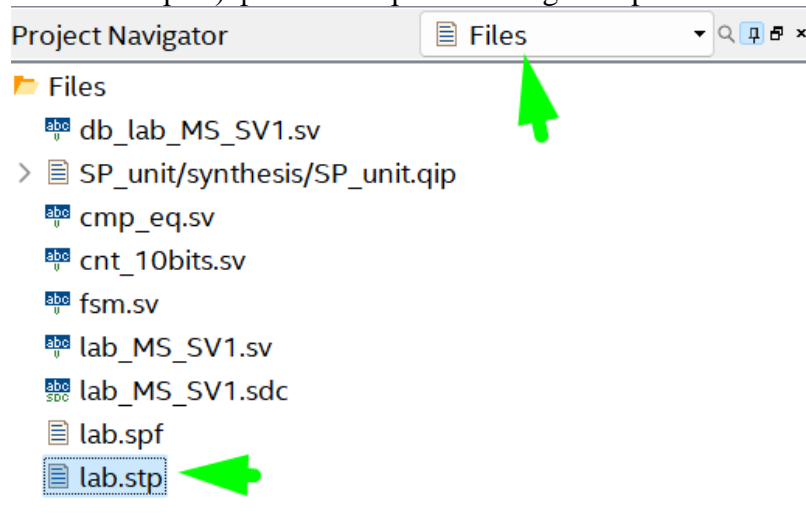
15. Осуществите конфигурацию FPGA на плате



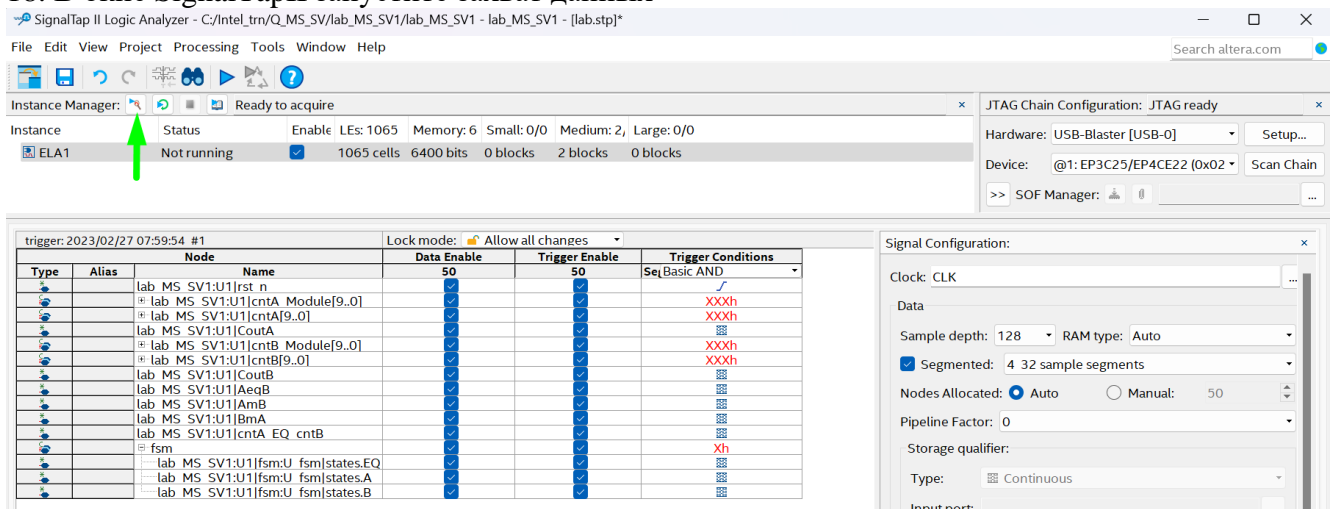
16. Убедитесь в том, что все настройки заданы так, как показано на рисунке ниже



17. Откройте (если он был закрыт) файл с настройками SignalTapII



18. В окне SignalTapII запустите захват данных



ЗАДАНИЕ: поясните преподавателю логику работы (захвата данных) логического анализатора, на основе анализа установленных для логического анализатора режимов и настроек.

19. Задание режимов работы для захвата данных в 4-х сегментах логического анализатора.

19.1. Откройте окно ISSPE

19.2. Установите указанные на рисунке ниже значения, затем запишите их в FPGA

Manual 1 s Write source data: Manually

Index	Instance ID	Status	Sources: 21	Probes: 0	Name
0	SP0	Not runni...	21	0	SP0

Index	Type	Alias	Name	Data
S20			rst n	0
S[19..10]			cntA Module	10
S[9..0]			cntB Module	10

19.3. Установите указанные на рисунке ниже значения, затем запишите их в FPGA

Manual 1 s Write source data: Manually

Index	Instance ID	Status	Sources: 21	Probes: 0	Name
0	SP0	Not runni...	21	0	SP0

Index	Type	Alias	Name	Data
S20			rst n	1
S[19..10]			cntA Module	10
S[9..0]			cntB Module	10

19.4. Установите указанные на рисунке ниже значения, затем запишите их в FPGA

Manual 1 s Write source data: Manually

Index	Instance ID	Status	Sources: 21	Probes: 0	Name
0	SP0	Not runni...	21	0	SP0

0 SP0

Index	Type	Alias	Name	Data
S20			rst n	0
S[19..10]			cntA Module	5
S[9..0]			cntB Module	10

19.5. Установите указанные на рисунке ниже значения, затем запишите их в FPGA

Manual 1 s Write source data: Manually

Index	Instance ID	Status	Sources: 21	Probes: 0	Name
0	SP0	Not runni...	21	0	SP0

0 SP0

Index	Type	Alias	Name	Data
S20			rst n	1
S[19..10]			cntA Module	5
S[9..0]			cntB Module	10

19.6. Установите указанные на рисунке ниже значения, затем запишите их в FPGA

Manual 1 s Write source data: Manually

Index	Instance ID	Status	Sources: 21	Probes: 0	Name
0	SP0	Not runni...	21	0	SP0

0 SP0

Index	Type	Alias	Name	Data
S20			rst n	0
S[19..10]			cntA Module	10
S[9..0]			cntB Module	5

19.7. Установите указанные на рисунке ниже значения, затем запишите их в FPGA

Manual 1 s Write source data: Manually

Index	Instance ID	Status	Sources: 21	Probes: 0	Name
0	SP0	Not runni...	21	0	SP0

0 SP0

Index	Type	Alias	Name	Data
S20			rst n	1
S[19..10]			cntA Module	10
S[9..0]			cntB Module	5

19.8. Установите указанные на рисунке ниже значения, затем запишите их в FPGA

Manual 1 s Write source data: Manually

Index	Instance ID	Status	Sources: 21	Probes: 0	Name
0	SP0	Not runni...	21	0	SP0

0 SP0

Index	Type	Alias	Name	Data
S20			rst n	0
S[19..10]			cntA Module	5
S[9..0]			cntB Module	5

20.

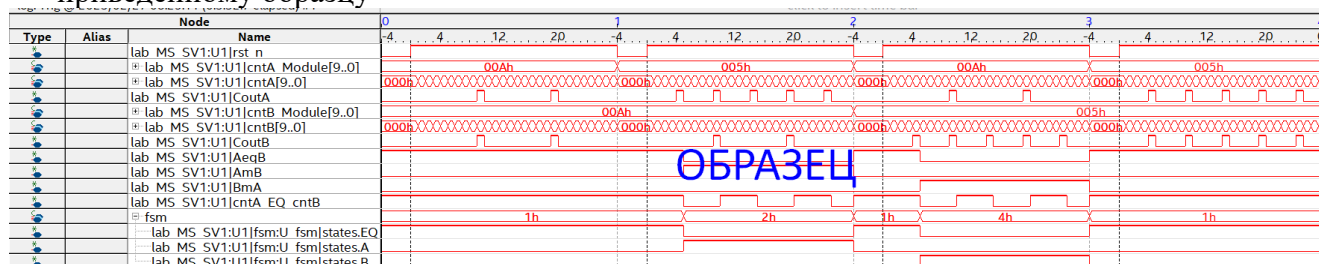
Manual 1 s Write source data: Manually

Index	Instance ID	Status	Sources: 21	Probes: 0	Name
0	SP0	Not runni...	21	0	SP0

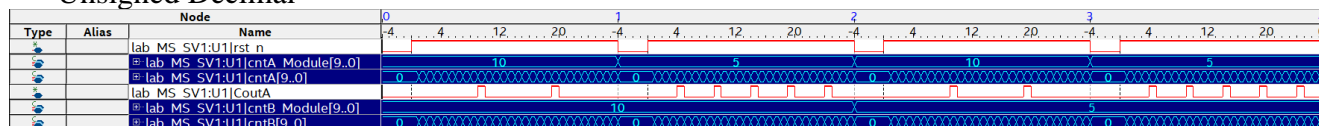
0 SP0

Index	Type	Alias	Name	Data
S20			rst n	1
S[19..10]			cntA Module	5
S[9..0]			cntB Module	5

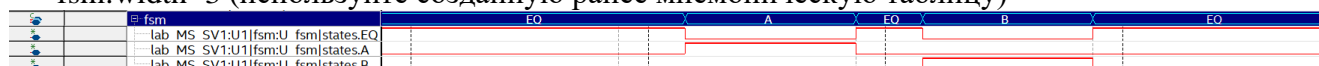
21. Откройте окно SignalTapII – в нем должна отображаться временная диаграмма аналогичная приведенному образцу



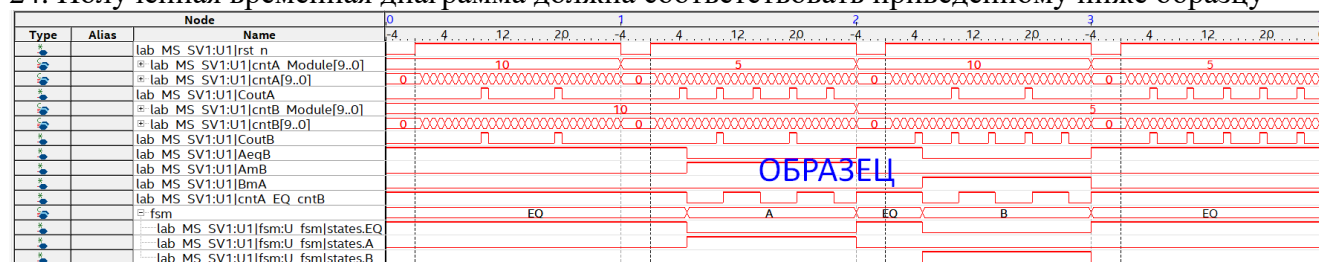
22. Для указанных на рисунке ниже цепей выберите режим отображения: Bus Display Format => Unsigned Decimal



23. Для указанной на рисунке ниже цепи выберите режим отображения: Bus Display Format => fsm:width=3 (используйте созданную ранее мнемоническую таблицу)



24. Полученная временная диаграмма должна соответствовать приведенному ниже образцу

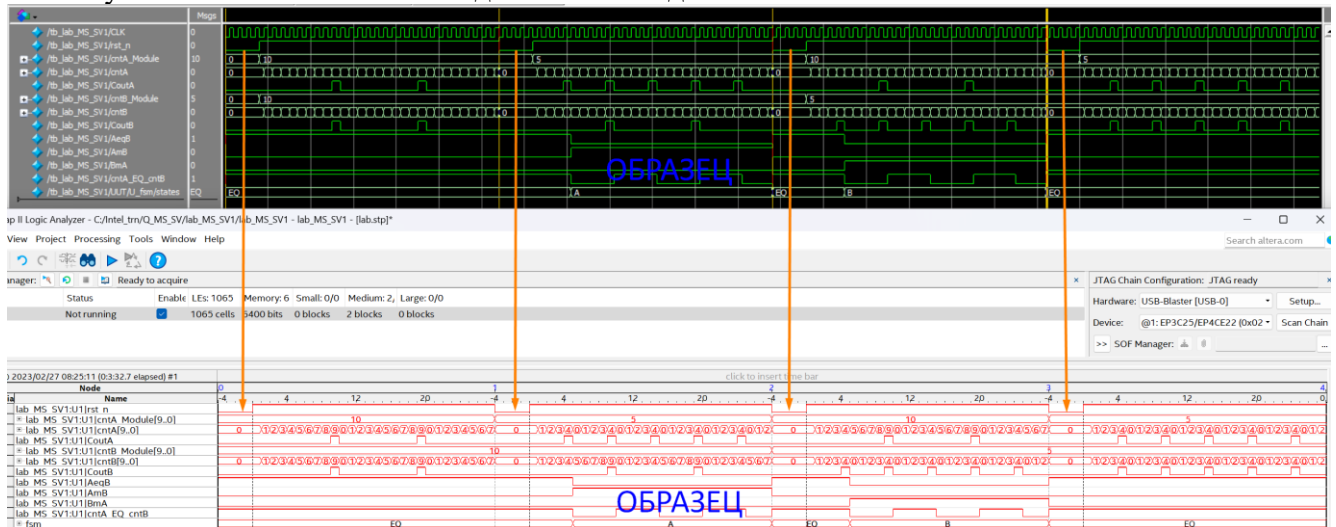


ЗАДАНИЕ: Проверьте:

- Значения счетчиков в моменты появления сигналов переноса
- Правильность формирования состояний конечного автомата (сравните с сигналами AeqB AmB BmA и графом переходов автомата)
- Правильность формирования сигнала cntA_EQ_cntB

ЗАДАНИЕ: сохраните полученную временную диаграмму.

25. Сравните полученную в SignalTapII временную диаграмму с временной диаграммой, полученной в ModelSim – они должны совпадать



26. Сохраните файл lab.stp.

27. Сохраните файл lab.spf.

Часть 5 – Дополнительное задание по отладке

1. Установите: два сегмента для логического анализатора

Type	Node	Name	Data Enable	Trigger Enable	Trigger Conditions
lab MS SV1:U1 Inst n	lab MS SV1:U1 cntA Module[9..0]		50	50	XXXXXXXXXXb
lab MS SV1:U1 cntA Module[9..0]	lab MS SV1:U1 cntA[9..0]				XXXXXXXXXXb
lab MS SV1:U1 CntA	lab MS SV1:U1 cntB Module[9..0]				XXXXXXXXXXb
lab MS SV1:U1 cntB Module[9..0]	lab MS SV1:U1 cntB[9..0]				XXXXXXXXXXb
lab MS SV1:U1 CntB	lab MS SV1:U1 AeB				
lab MS SV1:U1 AeB	lab MS SV1:U1 AmB				
lab MS SV1:U1 BmA	lab MS SV1:U1 cntA EQ cntB				Xh
fsm					

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:

2. Осуществите захват данных в Signal TapII - с помощью ISSPE сформируйте два набора значений модуля счета

2.1. Набор для сегмента 1

2.1.1. $\text{cntA_Module} = (4 + \text{ваш номер в списке группы} * 2)$

2.1.2. $\text{cntB_Module} = (4 + \text{ваш номер в списке группы} * 2) / 2$

2.2. Набор для сегмента 2

2.2.1. $\text{cntA_Module} = (4 + \text{ваш номер в списке группы} * 2) / 2$

2.2.2. $\text{cntB_Module} = (4 + \text{ваш номер в списке группы} * 2)$

3. Зафиксируйте, проверьте правильность и поясните преподавателю полученные временные диаграммы

4. Сохраните файл логического анализатора под именем lab_my.stp в папке C:\Intel_trn\Q_MS_SV\lab_MS_SV1

Лабораторная работа завершена.