lab_MS_SV5 задание для самостоятельного выполнения

Описание проекта lab_MS_SV5

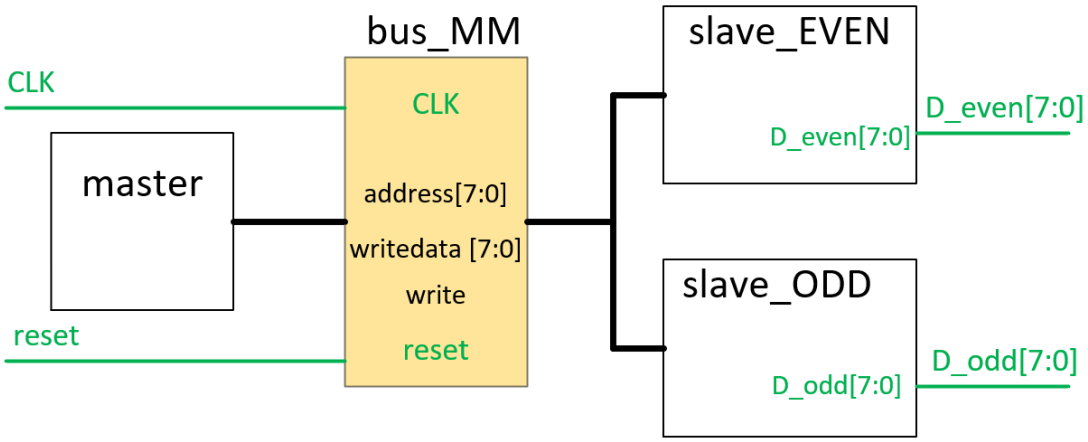
Рабочая папка C:\Intel_trn\Q_MS_SV\lab_MS_SV5.
Имя проекта – lab_ms_sv5.
Имя модуля верхнего уровня – lab_ms_sv5.
Файл с описанием – lab_ms_sv5.sv.

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

Структура разрабатываемого устройства.

Структура разрабатываемого устройства приведена на рисунке ниже.



В состав устройства входят:

- Модуль master – ведущее устройство, формирует обращение к двум ведомым устройствам.
- Модули slave_EVEN и slave_ODD – ведомые устройства, управляемые мастером.
- Модуль bus_MM - экземпляр интерфейса, обеспечивающий подключение мастера и ведомых устройств.

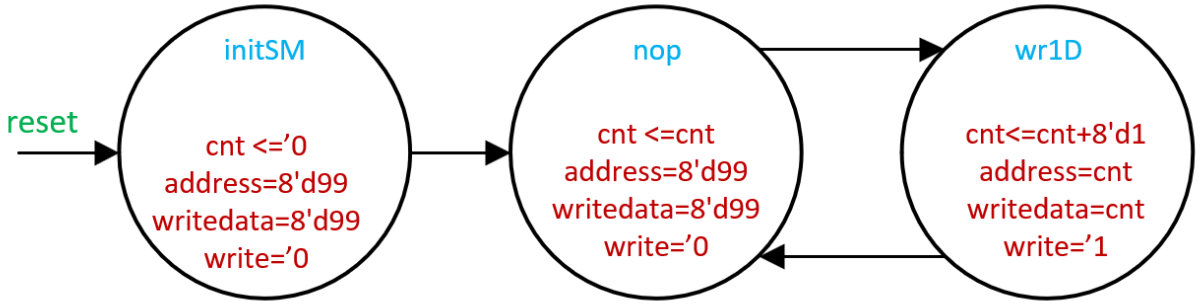
Выводы устройства (выделены зеленым цветом):

- CLK – вход тактового сигнала.
- reset – синхронный сброс всех устройств
- D_odd – восьмиразрядный выход
- D_even – восьмиразрядный выход

Алгоритм работы разрабатываемого устройства

Алгоритм работы разрабатываемого устройства определяется алгоритмами работы его модулей:

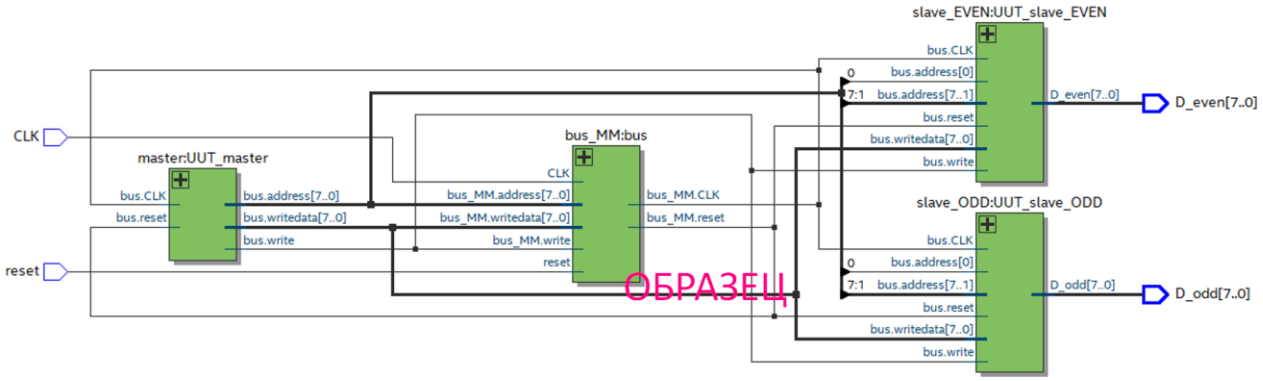
- Модуль master:
  - содержит КА Мура:
    - 3 состояния (граф переходов приведен ниже) - все переходы между состояниями безусловные:
      - начальное – initSM.
      - пустое – nop.
      - записи данных -wr1D.
    - формирует сигналы
      - address - Адреса (8 бит); комбинационный выход.
      - writedata - Данных (8 бит); комбинационный выход;
      - write - Разрешения записи; комбинационный выход;
  - содержит счетчик cnt (счетчик 8-ми разрядный), его значение используется для формирования адреса (address) и данных (writedata).



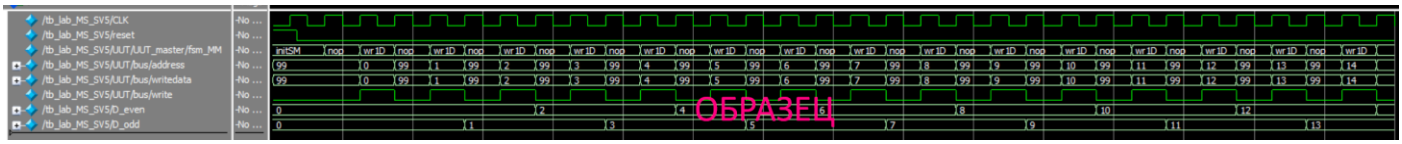
- Модуль slave_EVEN - «ведомый_четный»:
  - анализирует address и сигнал write и если: address четный и write = 1, то записывает в свой внутренний регистр значение шины данных (writedata);
  - на выходе – значение внутреннего регистра;
- Модуль slave_ODD - «ведомый_**не**четный»:
  - анализирует address и сигнал write и если: address **не**четный и write = 1, то записывает в свой внутренний регистр значение шины данных (writedata);
  - на выходе – значение внутреннего регистра;

Программа работы

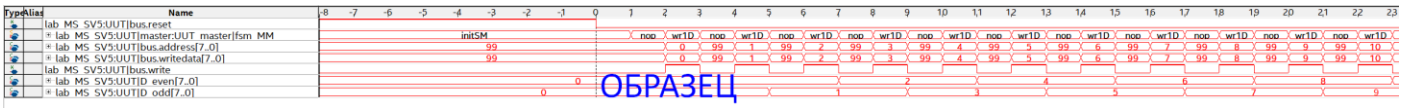
1. Разработайте описание интерфейса (файл lab_MS_SV5_interface.sv), включающего сигналы CLK и reset. *Пример файла приведен в приложении А.*
2. Используя интерфейс разработайте описание «мастера» (файл master.sv, модуль - master). *Пример файла приведен в приложении А.*
3. Используя интерфейс разработайте описание «ведомого_нечетного» (файл slave_ODD.sv, модуль - slave_ODD). *Пример файла приведен в приложении А.*
4. Используя интерфейс разработайте описание «ведомого_четного» (файл slave_EVEN.sv, модуль - slave_EVEN). *Пример файла приведен в приложении А.*
5. Используя интерфейс разработайте описание модуля верхнего уровня, объединяющего «мастера», «ведомого_нечетного» и «ведомого_четного» (файл lab_ms_sv5.sv, модуль - lab_ms_sv5). *Пример файла приведен в приложении А.*
6. Осуществите компиляцию модуля lab_ms_sv5 в пакете Quartus.
7. С помощью RTL Viewer отобразите структуру устройства.
    a. Проверьте правильность полученной структуры
    b. Приведите полученную структуру в отчете



8. Разработайте описание теста (файл tb_lab_MS_SV5.sv, модуль - tb_lab_MS_SV5).
    a. Тест первого класса – без автоматической проверки (пример приведен ниже).
    b. Результаты теста (должны быть представлены в отчете): временная диаграмма
    c. *Пример файла с описание теста приведен в приложении А.*



    d. По результатам моделирования в ModelSim необходимо доказать работоспособность устройства.
9. Разработайте модуль верхнего уровня для отладки (файл db_lab_MS_SV5.sv, модуль - db_lab_MS_SV5), содержащий:
    i. модуль lab_ms_sv5;
    ii. модуль SP_unit (его надо создать в пакете Quartus, используя IP: ISSPE) - модуль, обеспечивающий возможность задания сигнала reset, синхронизируемого тактовым сигналом CLK;
10. Создайте файл lab_ms_sv5.stp с такими настройки SignalTapII, чтобы получилась временная диаграмма, аналогичная приведенной ниже (на рисунке показана только часть диаграммы).
    a. Проверьте ее соответствие с временной диаграммой, полученной при моделировании.

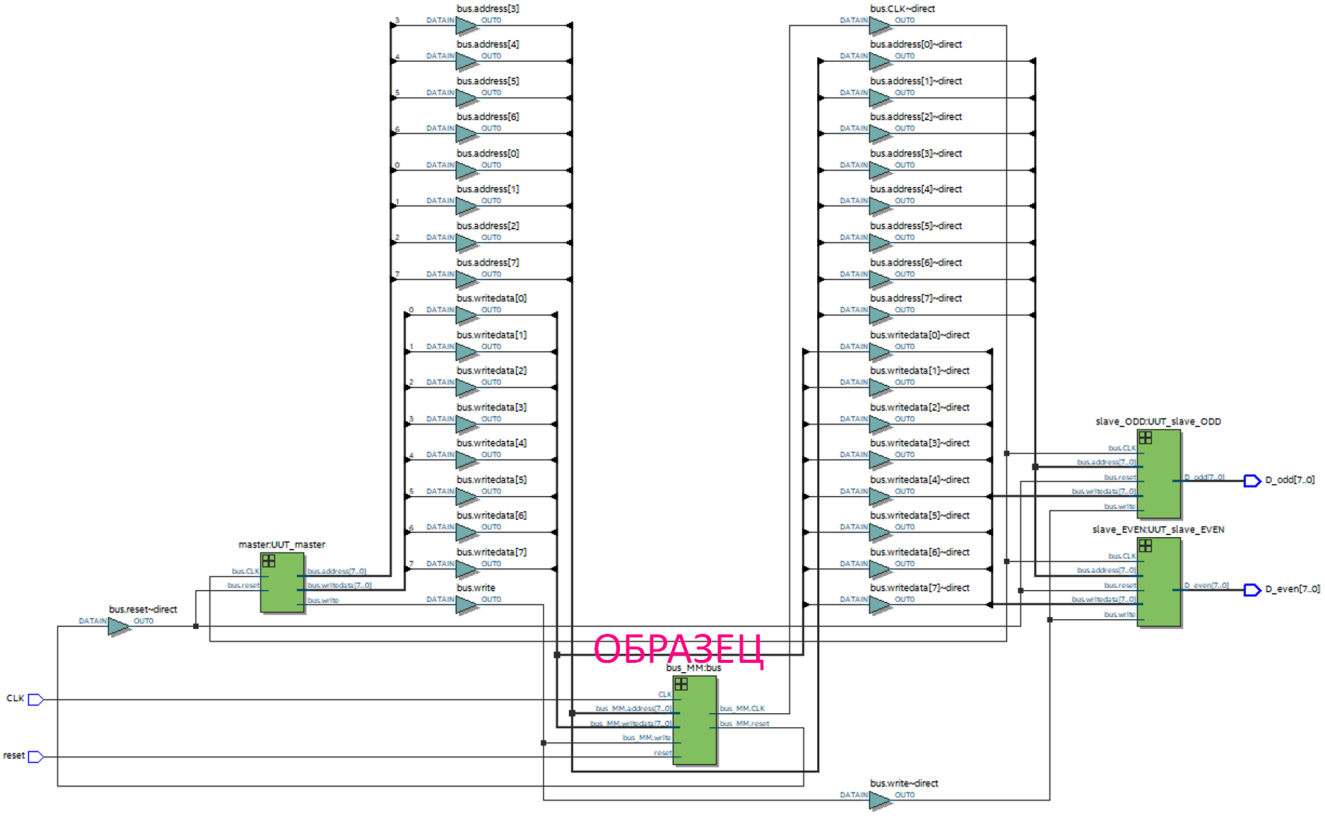


Дополнительное задание

1. Измените описание интерфейса (lab_MS_SV5_interface.sv):
   a. Добавьте modport для master
   b. Добавьте modport для slave

   Пример в приложении В

2. Соответственно измените описание модулей **master, slave_ODD, slave_EVEN**
   Примеры файлов в приложении В.
3. Осуществите компиляцию модуля lab_ms_sv5 в пакете Quartus.
4. С помощью RTL Viewer отобразите структуру устройства.
   b. Проверьте правильность полученной структуры
   c. Приведите полученную структуру в отчете



5. На основе теста из первой части работы осуществите моделирование и убедитесь в правильности работы модифицированного устройства.
6. Создайте файл lab_MS_SV5_dop.stp с такими настройки SignalTapII, чтобы захват данных (для модуля db_lab_MS_SV5.sv, созданного в первой части работы) осуществлялся при проявлении на выходе D_even (или D_odd) данных, соответствующих:
   Ваш номер в списке группы + 5.
   a. Приведите файл с настройками SignalTapII и временную диаграмму (в области срабатыванию условия захвата данных) в отчете.

Приложение А

Файл lab_MS_SV5_interface.sv

```
interface bus_MM (input bit CLK, input bit reset);
  bit [7:0]   address;
  bit [7:0]   writedata;
  bit         write;
endinterface
```

Файл master.sv

```
`timescale 1 ns / 1 ns
module master  (bus_MM  bus);
    enum bit[1:0] {initSM, nop, wr1D} fsm_MM;
    bit[7:0] cnt;
always_ff @ (posedge bus.CLK)
if (bus.reset)  begin
                    fsm_MM  <= initSM;
                    cnt     <= '0;
                end
else
    case (fsm_MM)
        initSM  :       fsm_MM <= nop;
        nop     :       fsm_MM <= wr1D;
        wr1D    :   begin
                        fsm_MM <= nop;
                        cnt <= cnt + 8'd1;
                    end
    endcase
always_comb
begin
    case (fsm_MM)
        wr1D:
            begin
                bus.address     = cnt;
                bus.write       = '1;
                bus.writedata   = cnt;
            end
        default
            begin
                bus.address     = 8'd99;
                bus.write       = 1'd0;
                bus.writedata   = 8'd99;
            end
    endcase
end
endmodule
```

Файл slave_EVEN.sv

```
`timescale 1 ns / 1 ns
module slave_EVEN  (
    bus_MM  bus,
    output bit[7:0] D_even
    );
always_ff @ (posedge bus.CLK)
if (bus.reset)  D_even <= '0;
else
    if ((bus.address ==? 8'b???????0) & (bus.write == '1))
        D_even <= bus.writedata;
endmodule
```

Файл slave_ODD.sv

```
`timescale 1 ns / 1 ns
module slave_ODD  (
    bus_MM  bus,
    output bit[7:0] D_odd
    );
always_ff @ (posedge bus.CLK)
if (bus.reset)  D_odd <= '0;
else
    if ((bus.address ==? 8'b???????1) & (bus.write == '1))
        D_odd <= bus.writedata;
endmodule
```

Файл lab_ms_sv5.sv

```
`timescale 1 ns / 1 ns
module lab_ms_sv5  (
input bit CLK,
input bit reset,
output bit[7:0]D_even,
output bit[7:0]D_odd
);
    bus_MM      bus             (.*);
    master      UUT_master      (.*);
    slave_EVEN  UUT_slave_EVEN  (.*);
    slave_ODD   UUT_slave_ODD   (.*);
endmodule
```

Файл tb_lab_MS_SV5.sv

```
`timescale 1 ns / 1 ns
module tb_lab_MS_SV5  ();
    bit CLK;
    bit reset ='1;
    bit[7:0]D_even;
    bit[7:0]D_odd;
lab_MS_SV5  UUT (.*);
initial
    forever #5 CLK = ~CLK;
initial begin
    #7;
    reset = '0;
    repeat (32) @(negedge CLK);
    $stop;
end
endmodule
```

Файл db_lab_MS_SV5.sv

```
module db_lab_MS_SV5 (
  (* altera_attribute = "-name IO_STANDARD \"3.3-V LVCMOS\"", chip_pin = "R8"  *)
  //"23" for miniDilab-CIV
  //"R8" for DE0_nano
  //"N5" for MAX10 NEEK
  input bit CLK
);
  bit reset;
  bit [7:0] D_even;
  bit [7:0] D_odd;
lab_MS_SV5 UUT (.*);
SP_unit SP_ (
  .source     (reset),// sources.source
  .source_clk (CLK)   // source_clk.clk
);
endmodule
```

Приложение В

Файл lab_MS_SV5_interface.sv

```
interface bus_MM (input bit CLK, input bit reset);
  bit [7:0]   address;
  bit [7:0]   writedata;
  bit         write;

modport master (
  input CLK, reset,
  output address, writedata, write);

modport slave (
  input CLK, reset,
  input address, writedata, write);
endinterface
```

Файл master.sv

```
`timescale 1 ns / 1 ns
module master  (bus_MM.master  bus);
    enum bit[1:0] {initSM, nop, wr1D} fsm_MM;
    bit[7:0] cnt;
always_ff @ (posedge bus.CLK)
if (bus.reset)              fsm_MM <= initSM;
else
    case (fsm_MM)
        initSM  :       fsm_MM <= nop;
        nop     :       fsm_MM <= wr1D;
        wr1D    :   begin
                        fsm_MM <= nop;
                        cnt <= cnt + 8'd1;
                    end
    endcase
always_comb
begin
    case (fsm_MM)
        wr1D:
            begin
                bus.address     = cnt;
                bus.write       = '1;
                bus.writedata   = cnt;
            end
        default
            begin
                bus.address     = 8'd99;
                bus.write       = 1'd0;
                bus.writedata   = 8'd99;
            end
    endcase
end
endmodule
```

Файл slave_EVEN.sv

```
1   `timescale 1 ns / 1 ns
2   module slave_EVEN
3       bus_MM.slave  bus,
4       output bit[7:0] D_even
5       );
6   always_ff @ (posedge bus.CLK)
7   if (bus.reset)  D_even <= '0;
8   else
9       if ((bus.address ==? 8'b???????0) & (bus.write == '1))
10          D_even <= bus.writedata;
11  endmodule
```

Файл slave_ODD.sv

```
1   `timescale 1 ns / 1 ns
2   module slave_ODD
3       bus_MM.slave  bus,
4       output bit[7:0] D_odd
5       );
6   always_ff @ (posedge bus.CLK)
7   if (bus.reset)  D_odd <= '0;
8   else
9       if ((bus.address ==? 8'b???????1) & (bus.write == '1))
10          D_odd <= bus.writedata;
11  endmodule
```