1928

K. N. Toosi University of Technology

1402-1403

Ramin Tavakoli

FPGA

Student Number: 9925063

Dr. Hosseini Nezhad

Homework 3

Department of electrical engineering

*K.N Toosi University of Technology

December 12, 2023

پروتکل ارتباط **SPI** را با استفاده از ماشین حالت را پیاده سازی و شبیه سازی کنید.

در این طراحی ورودی ها به صورت زیر خواهند بود:

- یک ورودی ۸ بیتی داده ورودی
- یک ورودی تک بیتی برای فعالسازی پروتکل **data ready**
- یک ورودی برای فعالسازی دریافت داده **miso in**
- یک ورودی **miso** مربوط به دریافت سریالی در **SPI**
- یک ورودی کلاک سیستم
- یک ورودی کلاک **SPI**

خروجی ها:

- خروجی های **SCK,MOSI,CS** مربوط به ارتباط **SPI**
- یک خروجی ۸ بیتی داده دریافت شده از طریق **MISO**
- یک خروجی **Data ready out** برای تعیین اینکه ۸ بیت با موفقیت دریافت شده است یا نه.

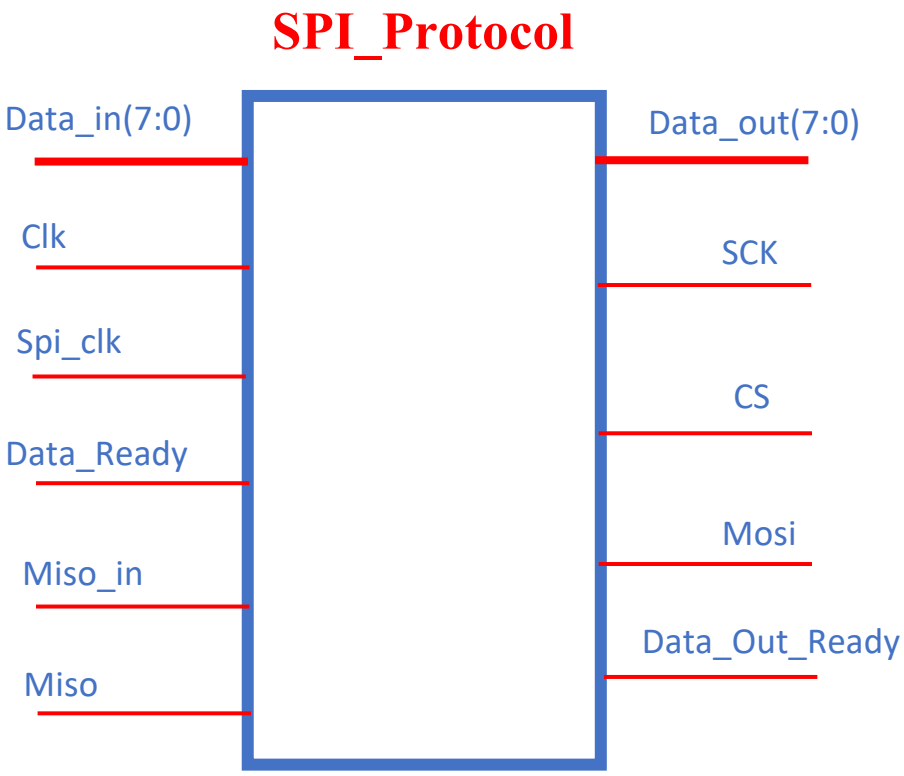
سیستم با کلاک اصلی کار خواهد کرد اما کلاکی که از طریق خروجی **SCK** ارسال می شود و کلاکی که با آن داده ارسال می شود کلاک **SPI** خواهد بود که به صورت یک ورودی دیگر تعریف شده است.

ماشین حالت سه حالت **idle** ، **send** و **tranceive** را خواهد داشت.

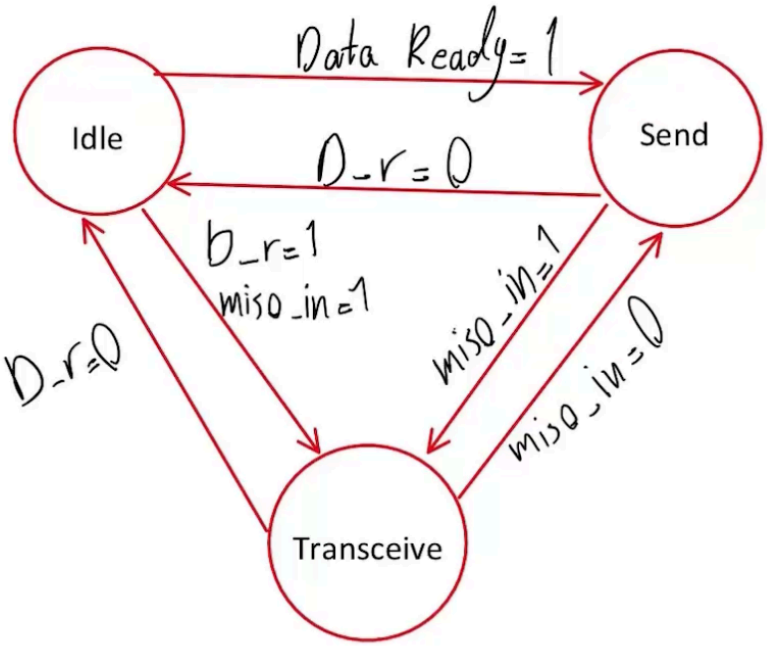
نحوه ی تغییر حالات بین سه حالت با استفاده از ورودی ها **data ready** و **miso in** تعیین می شود.

توضیحات تکمیلی مربوط به این تمرین را در ویدیو آپلود شده مشاهده کنید.

To design the SPI protocol hardware, we need output and input ports as shown below.



We use finite state machine design(FSM) and switch between the following states according to different inputs.

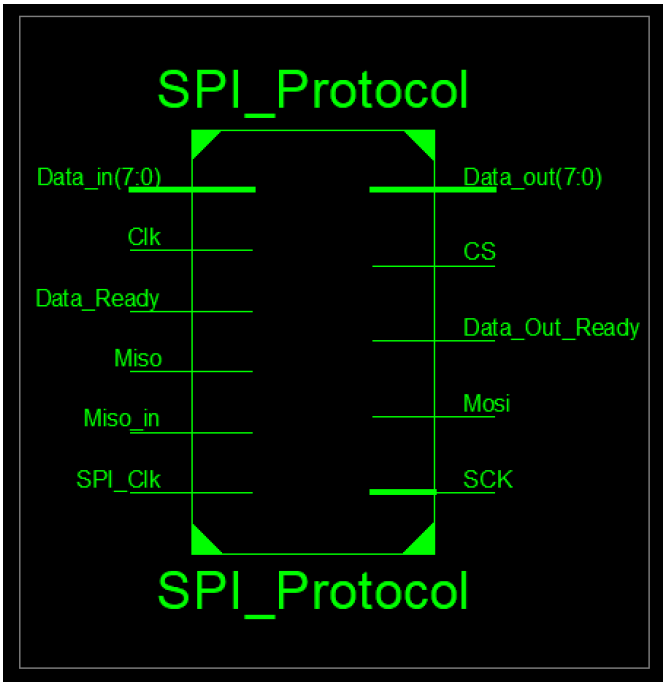


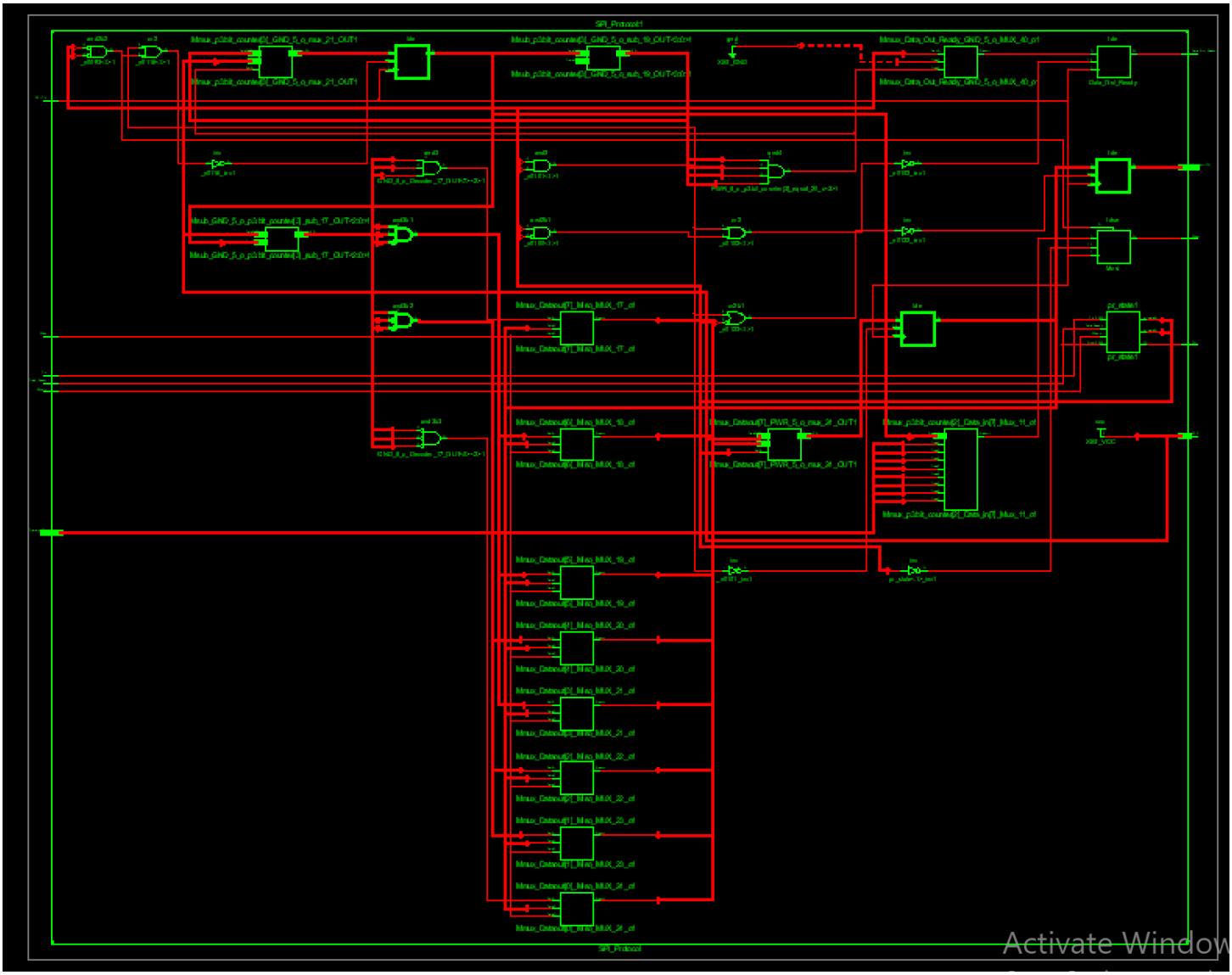
Circuit synthesis code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use ieee.std_logic_unsigned.all;

entity SPI_Protocol is
port(
      Clk, SPI_Clk: in std_logic;
      Data_Ready, Miso_in, Miso: in std_logic;
      Data_in: in std_logic_vector(7 downto 0);
      Data_out: out std_logic_vector(7 downto 0);
      CS, SCK, Mosi, Data_Out_Ready: out std_logic
);
end SPI_Protocol;

architecture Behavioral of SPI_Protocol is

type state is (idle,send,transceive);
signal pr_state,nx_state: state:=idle;
--signal bit_counter : integer range -1 to 7 := 7;  -- counter for bits sent and received

signal Datain: std_logic_vector(7 downto 0);
signal Dataout: std_logic_vector(7 downto 0);

begin

p1: process(clk)
begin

if(clk'event and clk='1') then
   pr_state <= nx_state;
   --Datain <= Data_in;
end if;
end process;
p2:process(Data_Ready, pr_state, Miso_in, miso, clk)
   begin
      case pr_state is
      when idle =>
         if (Data_Ready = '1' and Miso_in = '0') then
            nx_state <= send;
         elsif (Data_Ready = '1' and Miso_in = '1') then
            nx_state <= transceive;
         else
            nx_state <= idle;
         end if;
         CS <= '1';
         --SCK <= '1';

      when send =>
         if (Data_Ready = '0') then
            nx_state <= idle;
         elsif (Miso_in = '1') then
            nx_state <= transceive;
         else
            nx_state <= send;
         end if;
         CS <= '0';
         --SCK <= SPI_CLK;

      when transceive =>
         if (Data_Ready = '0') then
            nx_state <= idle;
         elsif (Miso_in = '0') then
            nx_state <= send;
         else
            nx_state <= transceive;
         end if;
         CS <= '0';
         --SCK <= SPI_CLK;
      when others=>
         nx_state<=idle;
      end case;

end process;
```

```
77  p3: process(SPI_CLK)
78  variable bit_counter : integer range -1 to 7 := 7;  -- counter for bits sent and received
79     begin
80        case pr_state is
81        when idle =>
82           SCK <= '1';
83        when send =>
84           SCK <= SPI_CLK;
85        when transceive =>
86           SCK <= SPI_CLK;
87        when others=>
88           SCK <= SPI_CLK;
89        end case;
90        if (SPI_CLK'event and SPI_clk='1') then
91           if(pr_state = idle)then
92              --CS <= '1';
93              --SCK <= '1';
94              Mosi <= '1';
95              Data_Out_Ready <= '0';
96              Dataout <= X"FF";
97           elsif(pr_state = send)then
98              --SCK <= SPI_CLK;
99              --CS <= '0';
100             Mosi <= Data_in(bit_counter);
101             Dataout <= X"FF";
102             bit_counter := bit_counter - 1; -- Decrement bit_counter after shifting out each bit
103             if( bit_counter = -1) then
104                bit_counter := 7;
105             end if;
106          elsif(pr_state = transceive) then
107             --SCK <= SPI_CLK;
108             Dataout(7 - bit_counter) <= Miso;
109             Data_out <= "UUUUUUUU";
110             bit_counter := bit_counter - 1;
111             --CS <= '0';
112             if( bit_counter = -1 ) then
113                Data_Out_Ready <= '1';
114                Data_out <= Dataout;
115                bit_counter := 7;
                      _
116             else
117             Data_Out_Ready <= '0';
118             end if;
119          end if;
120
121       end if;
122 end process;
123 end Behavioral;
124
```

Result of RTL Schematic:





The result and review of the circuit synthesis and extracted elements:

```
=========================================================================
*                            HDL Synthesis                              *
=========================================================================

Synthesizing Unit <SPI_Protocol>.
    Related source file is "C:\xilinix\ise-project\FPGA_class\HW4\HW4\SPI_Protocol.vhd".
    Found 1-bit register for signal <Mosi>.
    Found 1-bit register for signal <Data_Out_Ready>.
    Found 8-bit register for signal <Dataout>.
    Found 4-bit register for signal <p3.bit_counter>.
    Found 8-bit register for signal <Data_out>.
    Found 2-bit register for signal <pr_state>.
    Found finite state machine <FSM_0> for signal <pr_state>.
    -----------------------------------------------------------------------
    | States             | 3                                              |
    | Transitions        | 9                                              |
    | Inputs             | 2                                              |
    | Outputs            | 3                                              |
    | Clock              | Clk (rising_edge)                              |
    | Reset              | Data_Ready (negative)                          |
    | Reset type         | synchronous                                    |
    | Reset State        | idle                                           |
    | Power Up State     | idle                                           |
    | Encoding           | auto                                           |
    | Implementation     | LUT                                            |
    -----------------------------------------------------------------------
    Found 3-bit subtractor for signal <GND_5_o_p3.bit_counter[3]_sub_17_OUT<2:0>> created at line 95.
    Found 4-bit subtractor for signal <p3.bit_counter[3]_GND_5_o_sub_19_OUT<3:0>> created at line 97.
    Found 1-bit 8-to-1 multiplexer for signal <p3.bit_counter[2]_Data_in[7]_Mux_11_o> created at line 87.
    Summary:
        inferred   2 Adder/Subtractor(s).
        inferred  22 D-type flip-flop(s).
        inferred  12 Multiplexer(s).
        inferred   1 Finite State Machine(s).
Unit <SPI_Protocol> synthesized.
```

```
=========================================================================
HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                   : 2
 3-bit subtractor                                      : 1
 4-bit subtractor                                      : 1
# Registers                                            : 5
 1-bit register                                        : 2
 4-bit register                                        : 1
 8-bit register                                        : 2
# Multiplexers                                         : 12
 1-bit 2-to-1 multiplexer                              : 9
 1-bit 8-to-1 multiplexer                              : 1
 4-bit 2-to-1 multiplexer                              : 1
 8-bit 2-to-1 multiplexer                              : 1
# FSMs                                                 : 1
```

```
=========================================================================
*                      Advanced HDL Synthesis                           *
=========================================================================


=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                   : 2
 3-bit subtractor                                      : 1
 4-bit subtractor                                      : 1
# Registers                                            : 22
 Flip-Flops                                            : 22
# Multiplexers                                         : 12
 1-bit 2-to-1 multiplexer                              : 9
 1-bit 8-to-1 multiplexer                              : 1
 4-bit 2-to-1 multiplexer                              : 1
 8-bit 2-to-1 multiplexer                              : 1
# FSMs                                                 : 1
```

```
=========================================================================
*                            Design Summary                             *
=========================================================================

Top Level Output File Name         : SPI_Protocol.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 22
#      INV                         : 2
#      LUT2                        : 6
#      LUT3                        : 1
#      LUT6                        : 11
#      MUXF7                       : 1
#      VCC                         : 1
# FlipFlops/Latches                : 23
#      FD                          : 3
#      FDE                         : 19
#      FDSE                        : 1
# Clock Buffers                    : 2
#      BUFGP                       : 2
# IO Buffers                       : 23
#      IBUF                        : 11
#      OBUF                        : 12

Device utilization summary:
---------------------------

Selected Device : 6slx4tqg144-3


Slice Logic Utilization:
 Number of Slice Registers:             23  out of   4800     0%
 Number of Slice LUTs:                  20  out of   2400     0%
    Number used as Logic:               20  out of   2400     0%
```

```
Slice Logic Utilization:
 Number of Slice Registers:              23  out of   4800     0%
 Number of Slice LUTs:                   20  out of   2400     0%
    Number used as Logic:                20  out of   2400     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:     26
   Number with an unused Flip Flop:       3  out of     26    11%
   Number with an unused LUT:             6  out of     26    23%
   Number of fully used LUT-FF pairs:    17  out of     26    65%
   Number of unique control sets:         6

IO Utilization:
 Number of IOs:                          25
 Number of bonded IOBs:                  25  out of    102    24%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                2  out of     16    12%
```

Circuit Test Bench code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--USE ieee.numeric_std.ALL;

ENTITY TB_SPI_Porotocol IS
END TB_SPI_Porotocol;

ARCHITECTURE behavior OF TB_SPI_Porotocol IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT SPI_Protocol
    PORT(
         Clk : IN  std_logic;
         SPI_Clk : IN  std_logic;
         Data_Ready : IN  std_logic;
         Miso_in : IN  std_logic;
         Miso : IN  std_logic;
         Data_in : IN  std_logic_vector(7 downto 0);
         Data_out : OUT  std_logic_vector(7 downto 0);
         CS : OUT  std_logic;
         SCK : OUT  std_logic;
         Mosi : OUT  std_logic;
         Data_Out_Ready : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal Clk : std_logic := '0';
   signal SPI_Clk : std_logic := '0';
   signal Data_Ready : std_logic := '0';
   signal Miso_in : std_logic := '0';
   signal Miso : std_logic := '0';
   signal Data_in : std_logic_vector(7 downto 0) := (others => '0');

   --Outputs
   signal Data_out : std_logic_vector(7 downto 0);
   signal CS : std_logic;
   signal SCK : std_logic;
   signal Mosi : std_logic;
   signal Data_Out_Ready : std_logic;

   -- Clock period definitions
   --constant Clk_period : time := 10 ns;
   --constant SPI_Clk_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: SPI_Protocol PORT MAP (
          Clk => Clk,
          SPI_Clk => SPI_Clk,
          Data_Ready => Data_Ready,
          Miso_in => Miso_in,
          Miso => Miso,
          Data_in => Data_in,
          Data_out => Data_out,
          CS => CS,
          SCK => SCK,
          Mosi => Mosi,
          Data_Out_Ready => Data_Out_Ready
        );

```

```
65  CLK <= not CLK after 10 ns;
66  SPI_CLK <= not SPI_CLK after 15 ns;
67  Data_Ready <= '1' after 90 ns,'0' after 350 ns,'1' after 400 ns,'0' after 700 ns,'1' after 750 ns;
68  Miso_in <= '1' after 400 ns, '0' after 950 ns, '1' after 1100 ns;
69  Miso <= not Miso after 20 ns;
70  Data_in <= x"79";
71
72  END;
```

Simulation result: