HW4 + Simulations

FPGA & ASIC

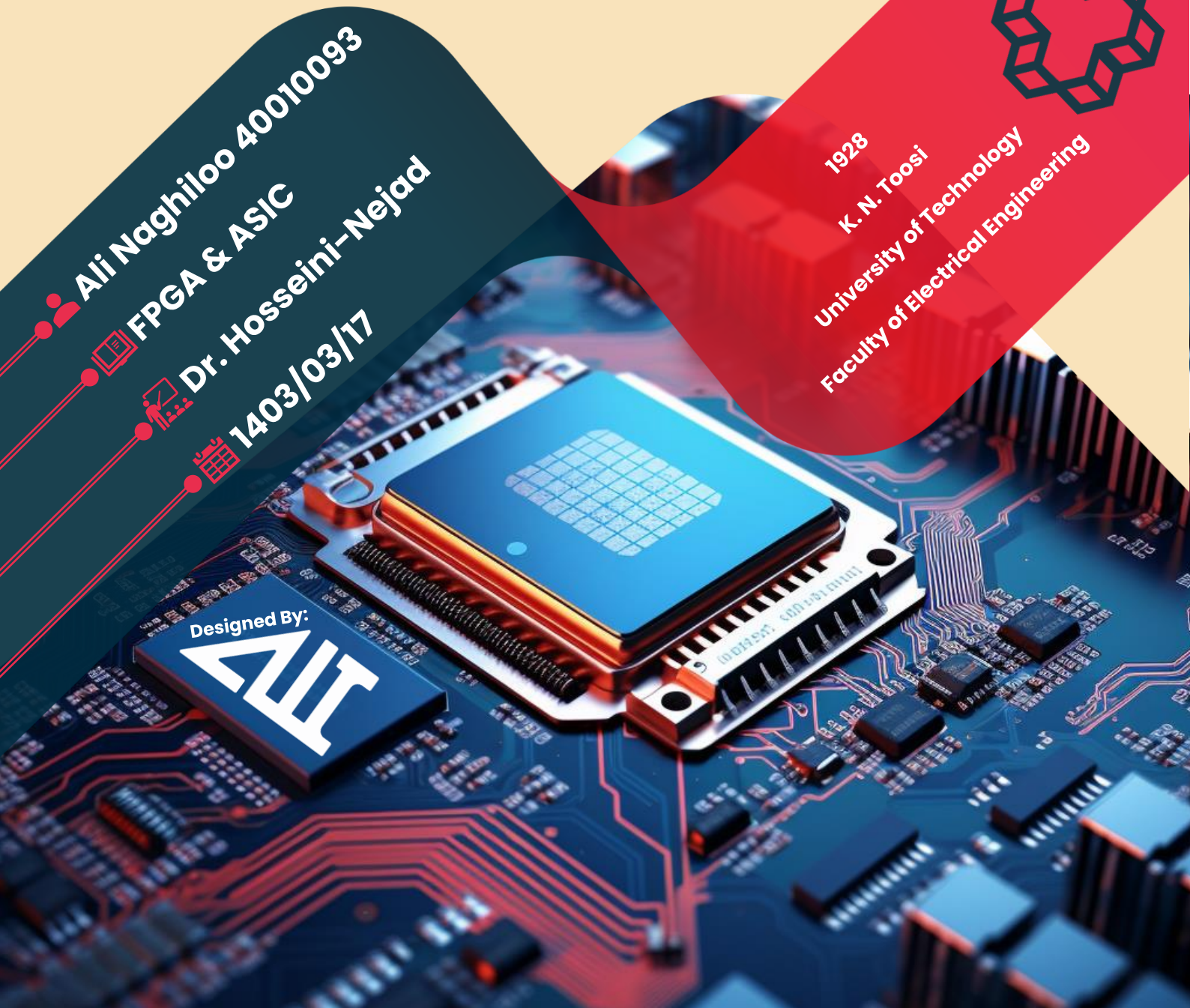
Ali Naghiloo 40010093

FPGA & ASIC

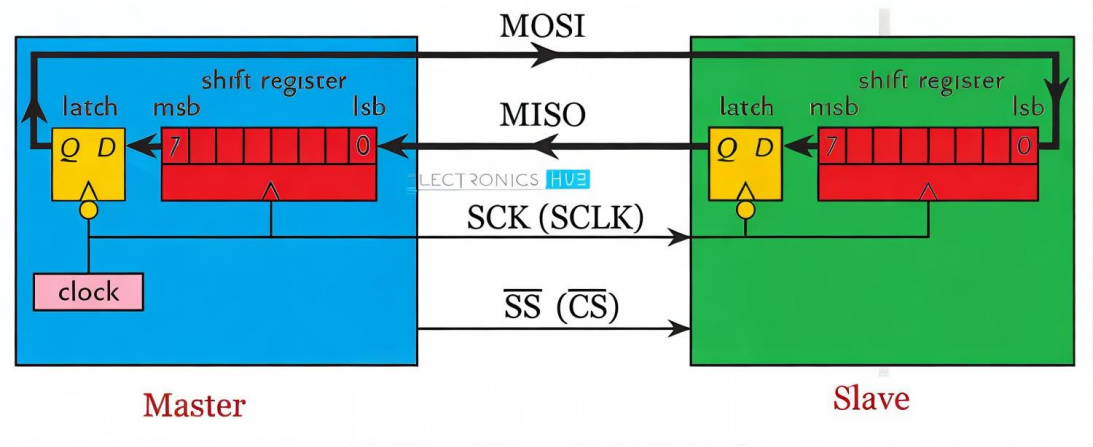
Dr. Hosseini-Nejad

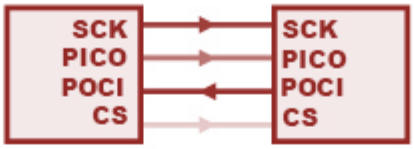
1403/03/17

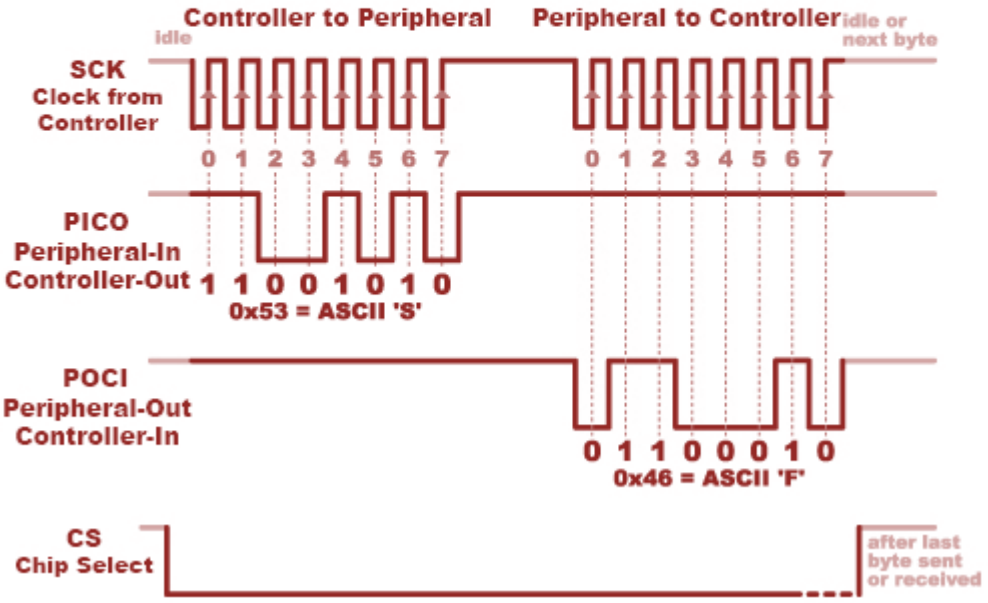
1928
K. N. Toosi
University of Technology
Faculty of Electrical Engineering



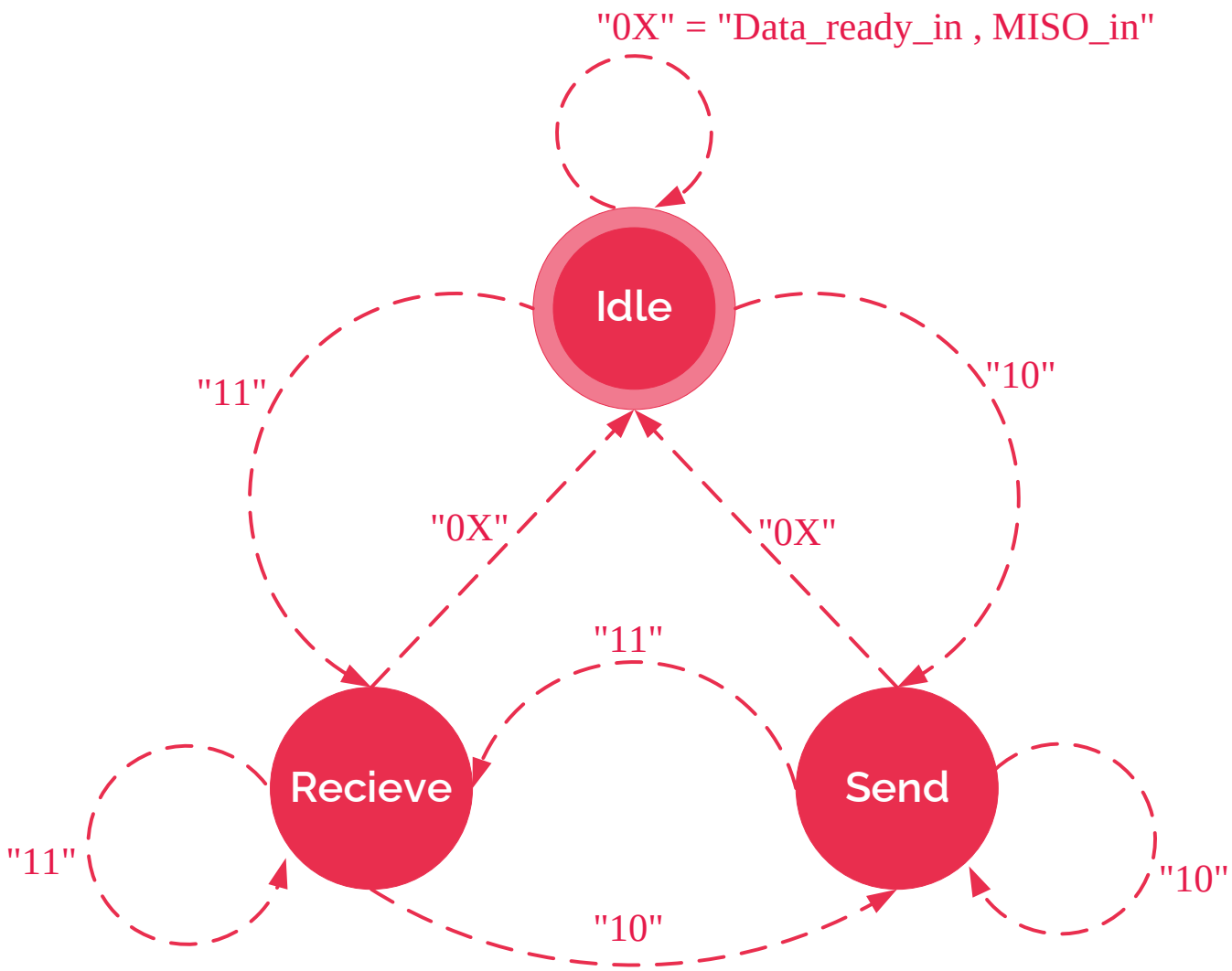
Serial Peripheral Interface Introduction







Short Name	Long Name	Description (historical terms in parens)
$\overline{CS}$	Chip Select	Active-low chip select signal from main (master) to enable communication with a specific sub (slave) device.
SCLK	Serial Clock	Clock signal from main (master) transitions for each serial data bit.
MOSI	Main Out, Sub In (master out, slave in)	Serial data from main (master), highest bit first.
MISO	Main In, Sub Out (master in, slave out)	Serial data from sub (slave), highest bit first.


"0X" = "Data_ready_in , MISO_in"
Idle
"11"
"10"
"0X"
"0X"
"11"
Recieve
Send
"11"
"10"
"10"

SPI controller Code:

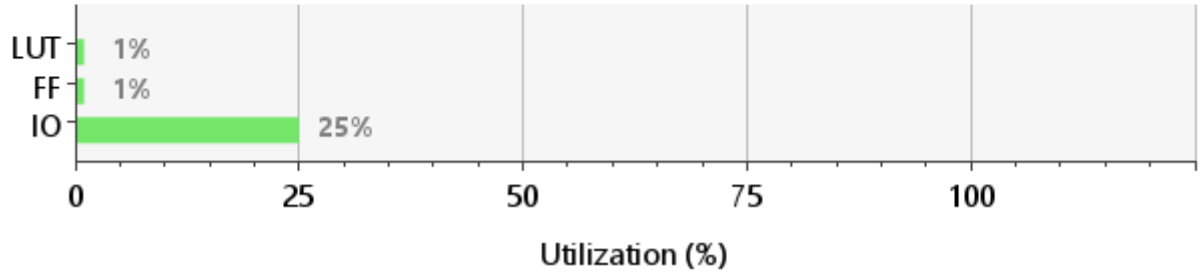
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.ALL;

entity SPI is
    Port ( Parallel_in : in STD_LOGIC_VECTOR (7 downto 0);
           Data_ready_in : in STD_LOGIC;
           MISO_in : in STD_LOGIC;
           CLK : in STD_LOGIC;
           MISO : in STD_LOGIC;
           SPI_CLK : in STD_LOGIC;
           ------------------- in <=> out -------------------------
           CS : out STD_LOGIC;
           SCK : out STD_LOGIC;
           MOSI : out STD_LOGIC;
           Parallel_out : out STD_LOGIC_VECTOR (7 downto 0);
           Data_ready_out : out STD_LOGIC);
end SPI;

architecture Behavioral of SPI is
type state_type is (Idle, Send, Recieve);
signal present_st, next_st : state_type :=Idle;
signal bit_counter : integer range 0 to 7 :=7;
begin
-----------------------------------------------------------
demo_p: process(CLK)
begin
if(CLK'event and CLK='1') then
present_st <= next_st;
end if;
end process;
-----------------------------------------------------------
state_changer_p: process(Data_ready_in, MISO_in, present_st)
begin
--case present_st is
--when Idle    => if(Data_ready_in='0')then next_st<=Idle; elsif(MISO_in='0')then next_st<=Send; else next_st<=Recieve;end if;
--when Send    => if(Data_ready_in='0')then next_st<=Idle; elsif(MISO_in='0')then next_st<=Send; else next_st<=Recieve;end if;
--when Recieve => if(Data_ready_in='0')then next_st<=Idle; elsif(MISO_in='0')then next_st<=Send; else next_st<=Recieve;end if;
--when others  => Idle;
--end case;
if(Data_ready_in='0')then next_st<=Idle; elsif(MISO_in='0')then next_st<=Send; else next_st<=Recieve;end if; -- equall to the upper
code!
end process;
-----------------------------------------------------------
ALL_output_p: process(SPI_CLK)
begin
if(SPI_CLK'event and SPI_CLK='1') then
case present_st is
when Idle    => CS<='0'; SCK<='0'; MOSI<='0'; Parallel_out<=(others=>'0'); Data_ready_out<='0';
when Send    => CS<='1'; SCK<=SPI_CLK;
                bit_counter <= bit_counter - 1;
                MOSI<=Parallel_in(bit_counter); Parallel_out<=Parallel_in;
when Recieve => CS<='1'; SCK<=SPI_CLK;
                bit_counter <= bit_counter - 1;
                MOSI<='0'; Parallel_out(bit_counter)<= MISO;
end case;

if(bit_counter=0)then bit_counter<=7; Data_ready_out<='1'; end if;

end if;
end process;
end Behavioral;
```

---------- TB ----------

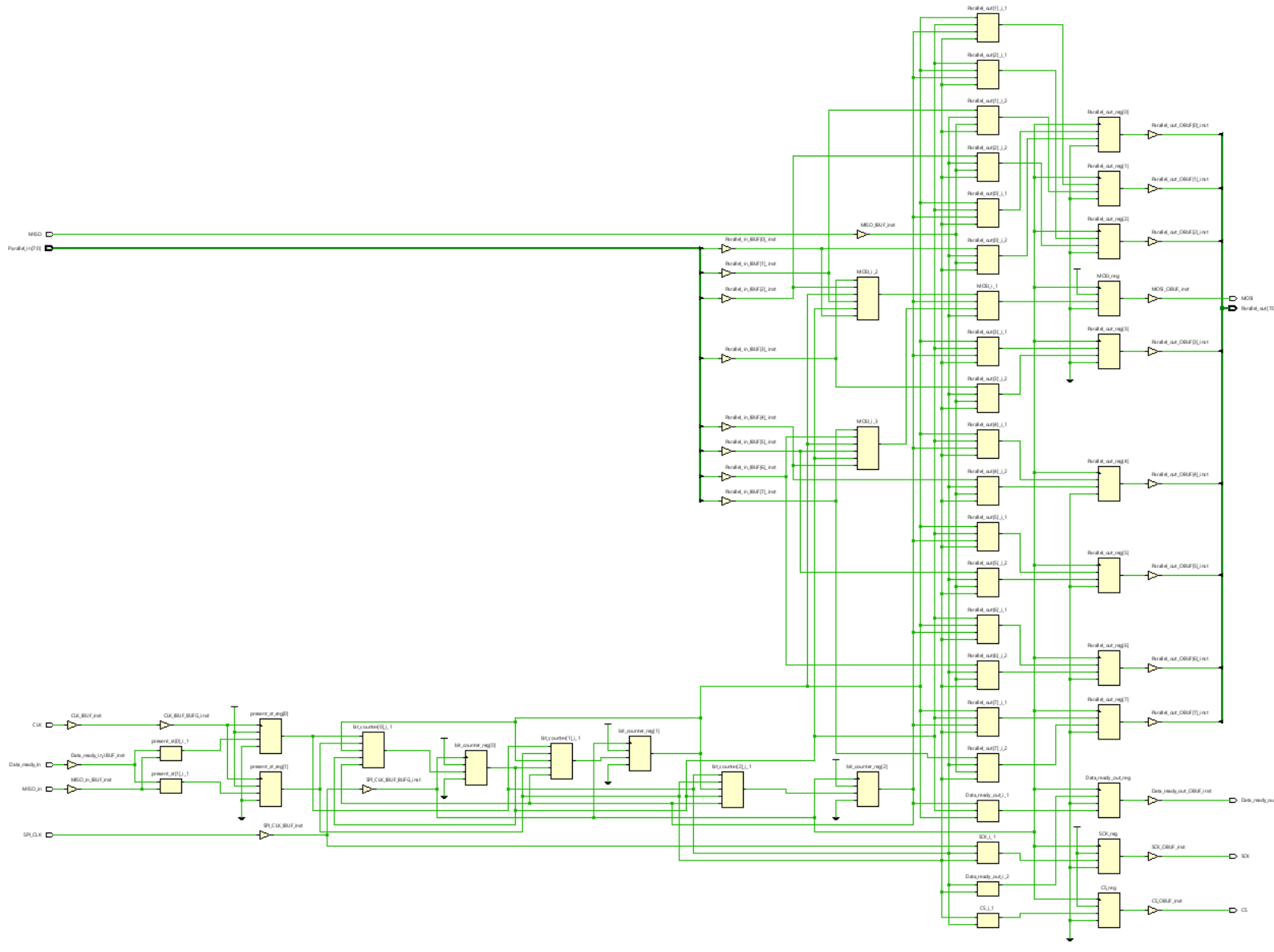
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.ALL;

entity SPI_TB is
--  Port ( );
end SPI_TB;

architecture Behavioral of SPI_TB is
    signal Parallel_in : STD_LOGIC_VECTOR (7 downto 0) :=(others=>'0');
    signal Data_ready_in : STD_LOGIC :='0';
    signal MISO_in : STD_LOGIC :='0';
    signal CLK : STD_LOGIC :='0';
    signal MISO : STD_LOGIC :='0';
    signal SPI_CLK : STD_LOGIC :='0';
    ------------------- in <=> out -------------------------
    signal CS : STD_LOGIC;
    signal SCK : STD_LOGIC;
    signal MOSI : STD_LOGIC;
    signal Parallel_out : STD_LOGIC_VECTOR (7 downto 0);
    signal Data_ready_out : STD_LOGIC;

    constant period : time := 20 ns;
begin

DUT: entity work.SPI port map(Parallel_in,
                              Data_ready_in,
                              MISO_in,
                              CLK,
                              MISO,
                              SPI_CLK,
                              ---------
                              CS,
                              SCK,
                              MOSI,
                              Parallel_out,
                              Data_ready_out);

CLK <= not CLK after (period/2);
SPI_CLK <= not SPI_CLK after (period/2)*1.5;
MISO <= not MISO after (period/2)*2;

Data_ready_in <= '1' after 92 ns, '0' after 352 ns, '1' after 402 ns, '0' after 702 ns, '1' after 752 ns;
MISO_in <= '1' after 407.5 ns, '0' after 956.8 ns, '1' after 1111.85 ns;

Parallel_in <= b"1100_0010" after ((period/2)*1.5)*3;

end Behavioral;
```

Utilization-synth & synthesis report is as follows:

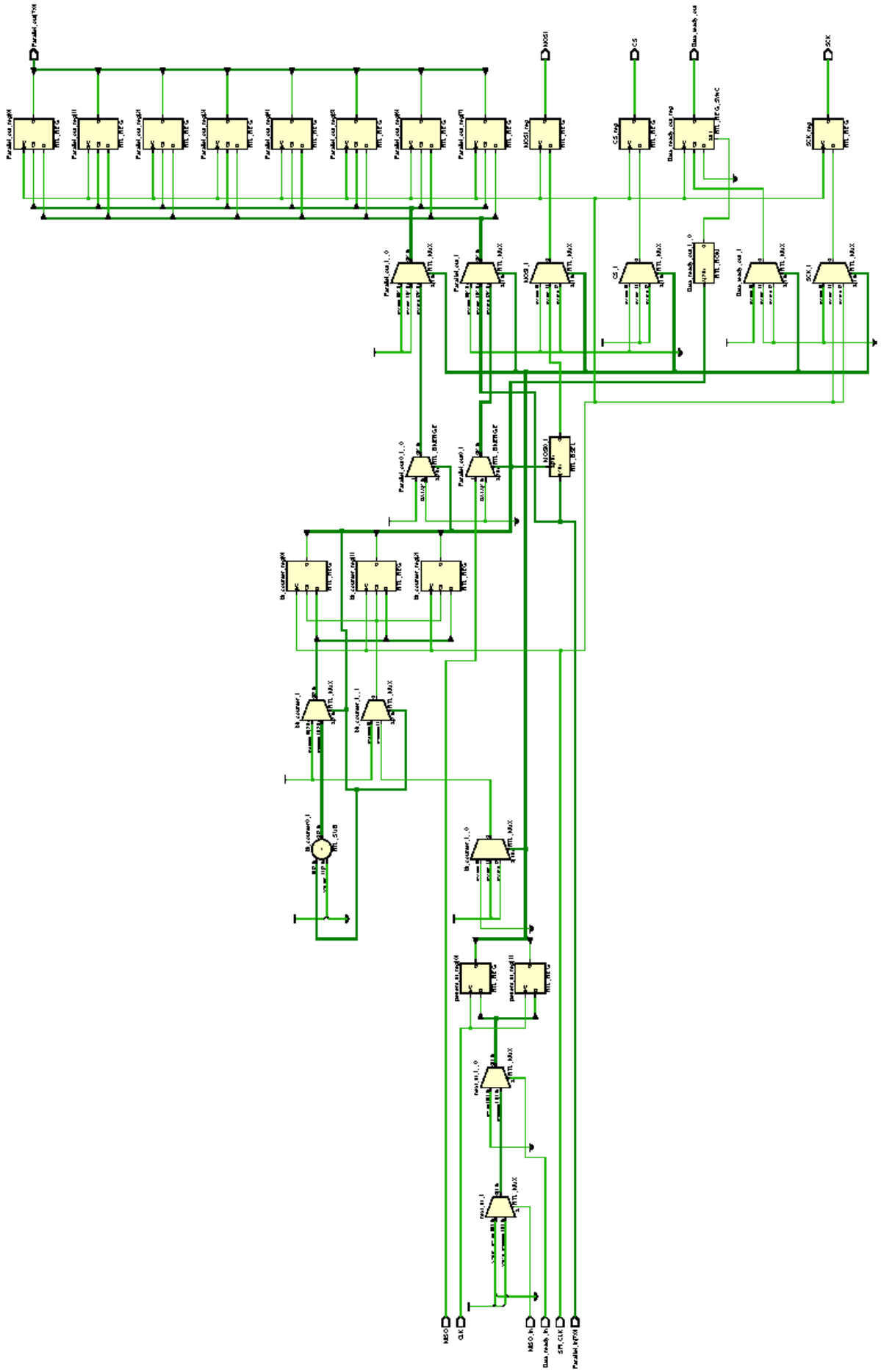
Resource	Utilization	Available	Utilization %
LUT	25	3750	0.67
FF	17	7500	0.23
IO	25	100	25.00



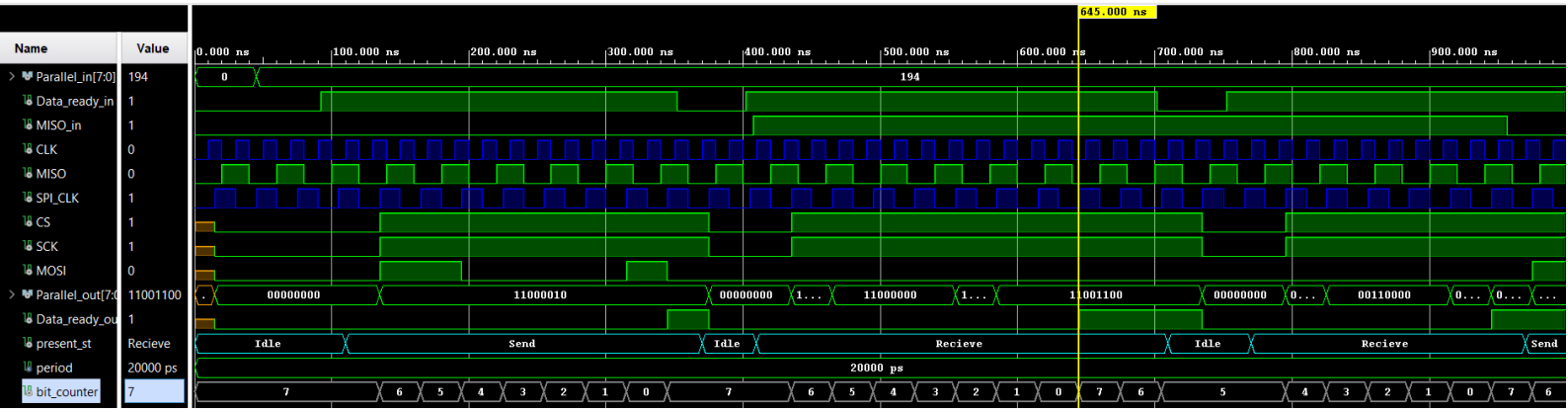
LUT Schematic:



RTL Schematic:



Waveform


Name
Value
Parallel_in[7:0]
194
Data_ready_in
1
MISO_in
1
CLK
0
MISO
0
SPI_CLK
1
CS
1
SCK
1
MOSI
0
Parallel_out[7:0]
11001100
Data_ready_out
1
present_st
Recieve
period
20000 ps
bit_counter
7
645.000 ns
0.000 ns
100.000 ns
200.000 ns
300.000 ns
400.000 ns
500.000 ns
600.000 ns
700.000 ns
800.000 ns
900.000 ns
0
194
00000000
11000010
00000000
11000000
11001100
00000000
00110000
Idle
Send
Idle
Recieve
Idle
Recieve
Send
20000 ps