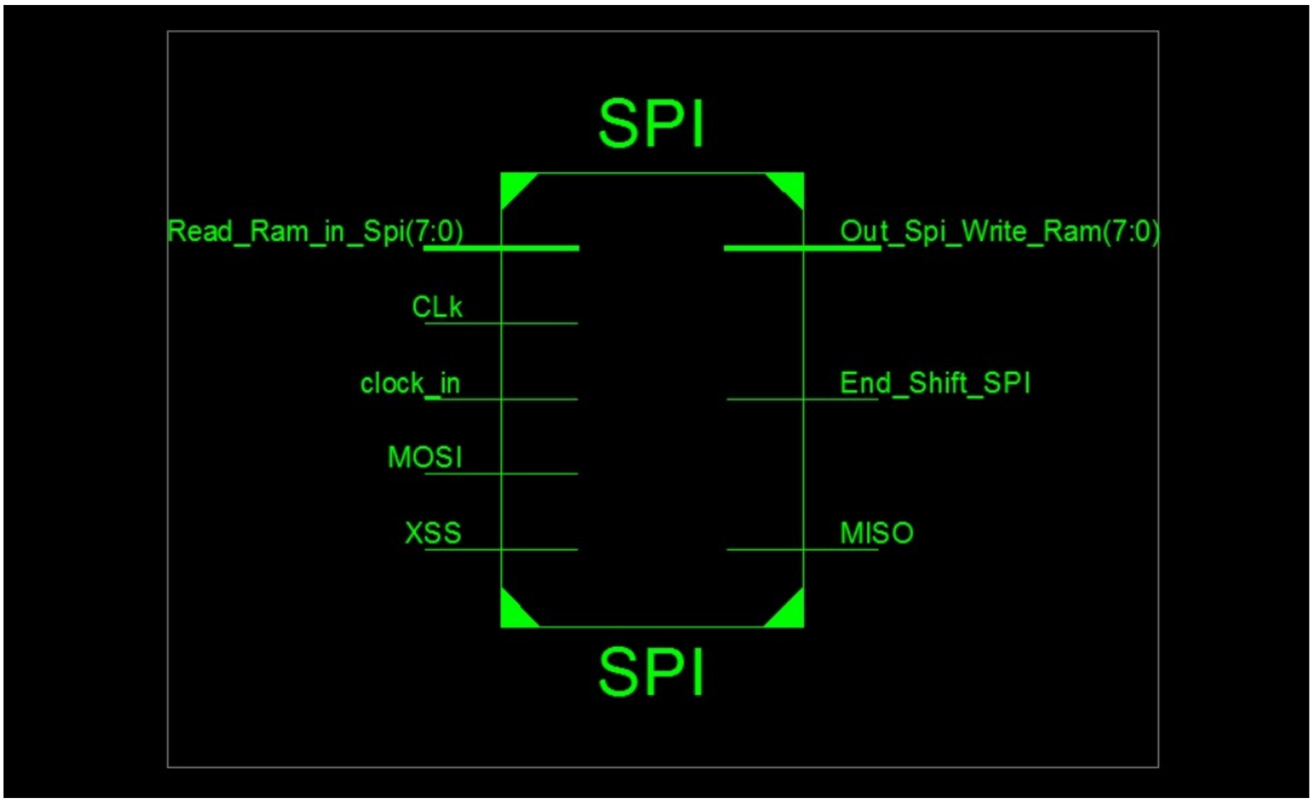
به نام خدا

طراحی و پیاده‌سازی SPI controller



Read_RAM_in_SPI:ورودی ۸ بیتی که از RAMخوانده می‌شود.

Out_SPI_Write_RAM:خروجی ۸ بیتی که به RAM ارسال می‌شود.

Clk:کلاک مرجع VPC3

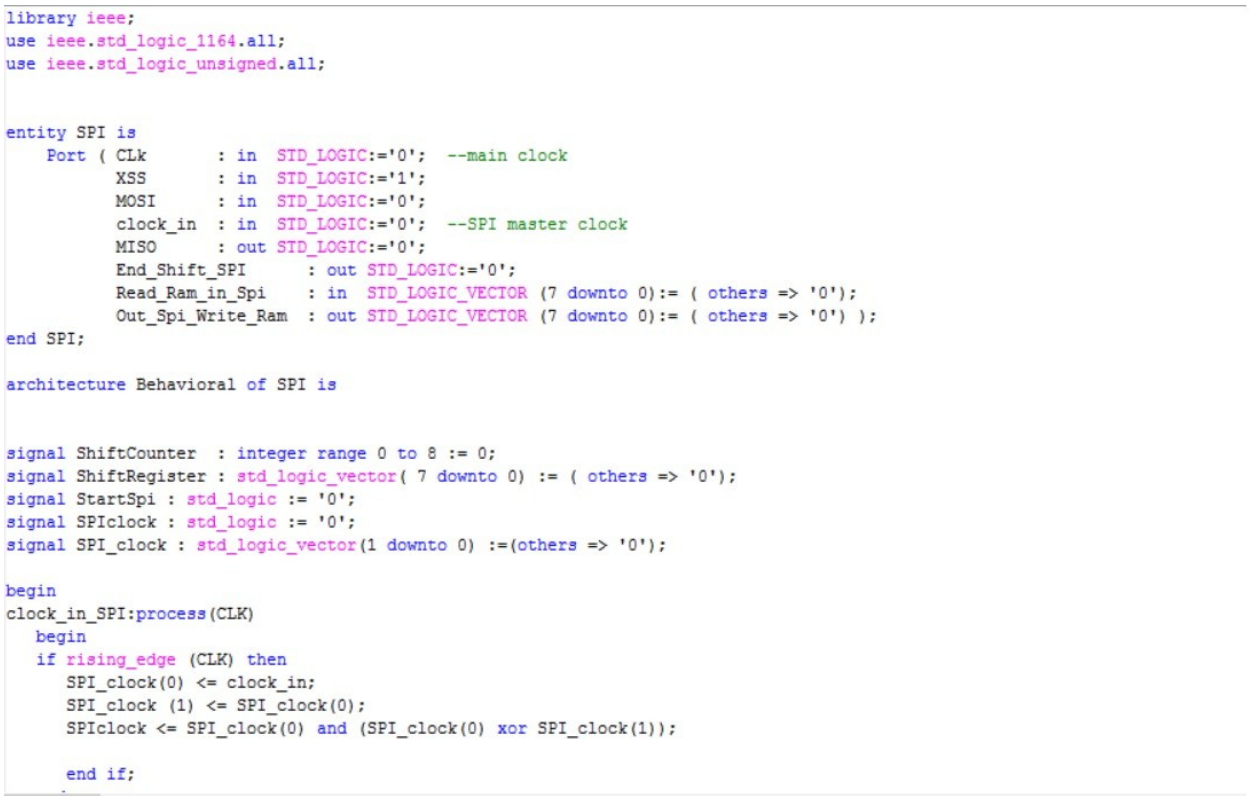
Clock_in: کلاک ورودی از سمت میکروکنترلر

XSS: پایه فعال کننده SPI

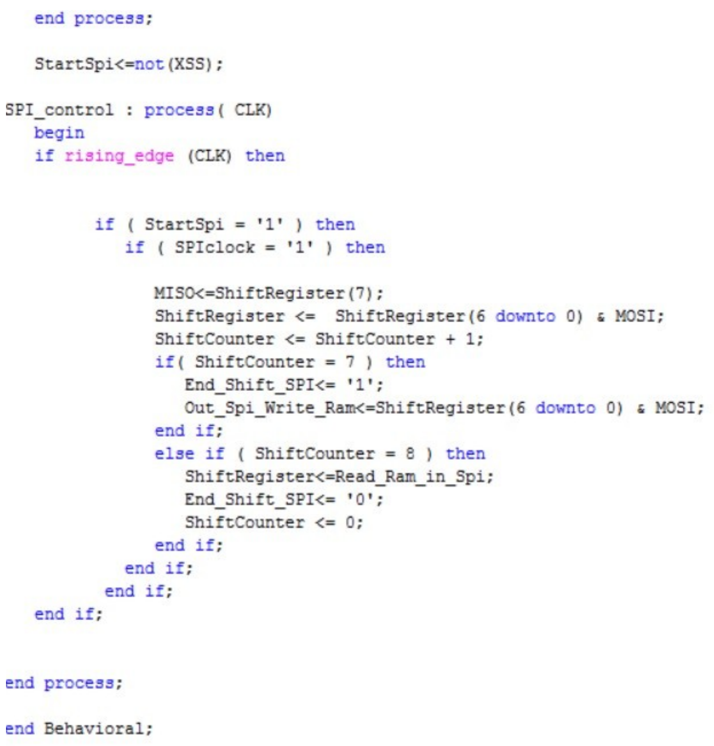
End_Shift_SPI: خروجی یک بیتی بعد از انجام ۸ بار شیفت

MOSI:ورودی شیفت رجیستر از سمت میکرو

MISO:خروجی شیفت رجیستر به سمت میکرو

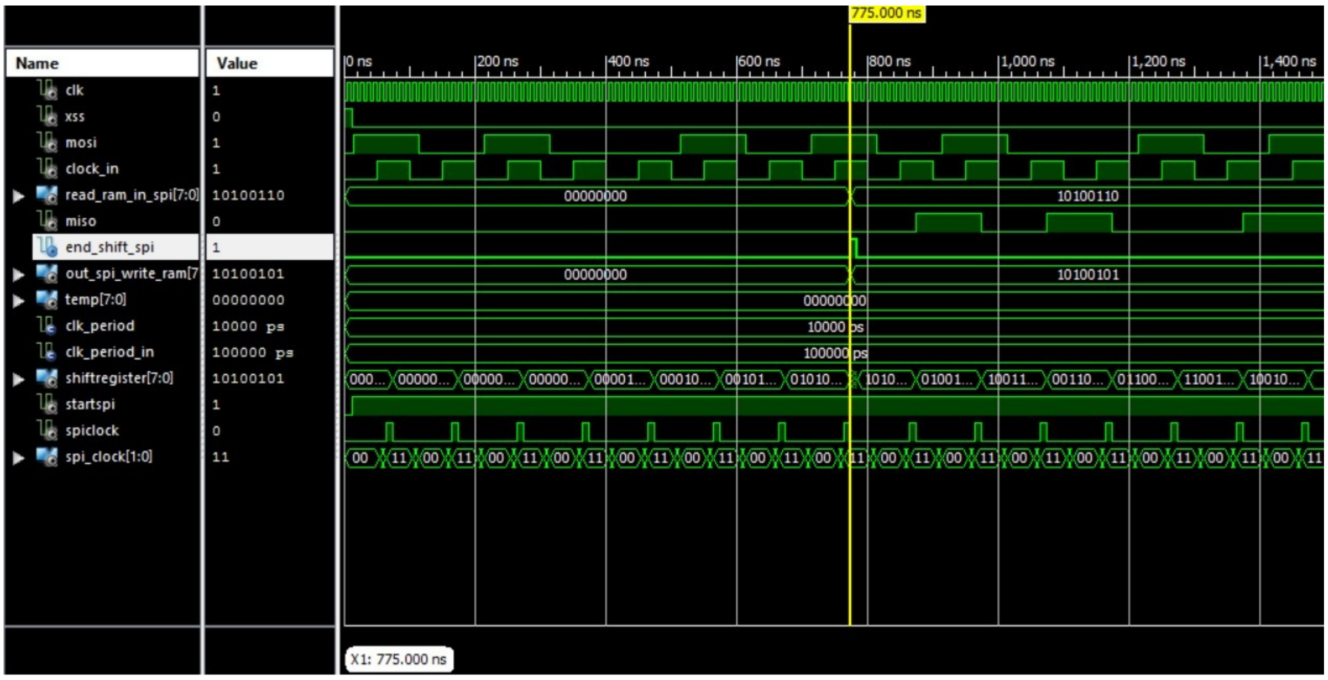


```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;


entity SPI is
    Port ( CLk       : in  STD_LOGIC:='0';  --main clock
           XSS       : in  STD_LOGIC:='1';
           MOSI      : in  STD_LOGIC:='0';
           clock_in  : in  STD_LOGIC:='0';  --SPI master clock
           MISO      : out STD_LOGIC:='0';
           End_Shift_SPI      : out STD_LOGIC:='0';
           Read_Ram_in_Spi    : in  STD_LOGIC_VECTOR (7 downto 0):= ( others => '0');
           Out_Spi_Write_Ram  : out STD_LOGIC_VECTOR (7 downto 0):= ( others => '0') );
end SPI;

architecture Behavioral of SPI is


signal ShiftCounter  : integer range 0 to 8 := 0;
signal ShiftRegister : std_logic_vector( 7 downto 0) := ( others => '0');
signal StartSpi : std_logic := '0';
signal SPIclock : std_logic := '0';
signal SPI_clock : std_logic_vector(1 downto 0) :=(others => '0');

begin
clock_in_SPI:process(CLK)
   begin
   if rising_edge (CLK) then
      SPI_clock(0) <= clock_in;
      SPI_clock (1) <= SPI_clock(0);
      SPIclock <= SPI_clock(0) and (SPI_clock(0) xor SPI_clock(1));

      end if;
```

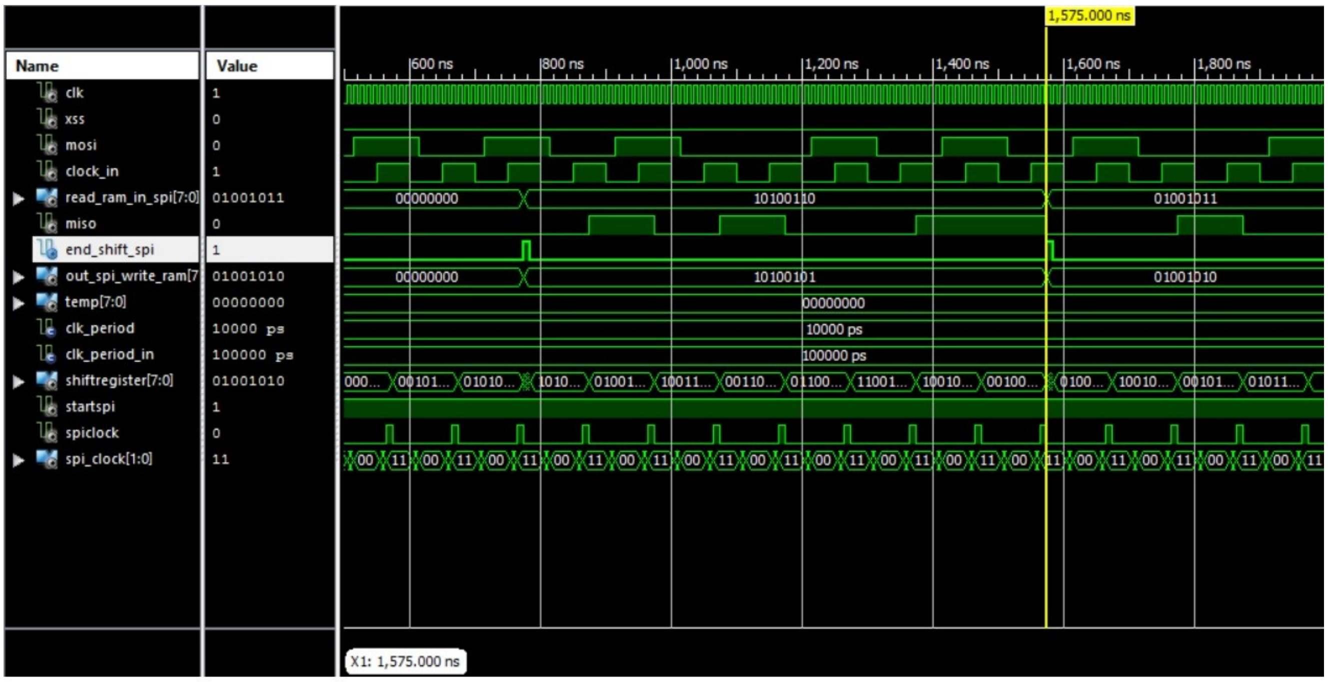


```vhdl
   end process;

   StartSpi<=not(XSS);

SPI_control : process( CLK)
   begin
   if rising_edge (CLK) then


         if ( StartSpi = '1' ) then
            if ( SPIclock = '1' ) then

               MISO<=ShiftRegister(7);
               ShiftRegister <=  ShiftRegister(6 downto 0) & MOSI;
               ShiftCounter <= ShiftCounter + 1;
               if( ShiftCounter = 7 ) then
                  End_Shift_SPI<= '1';
                  Out_Spi_Write_Ram<=ShiftRegister(6 downto 0) & MOSI;
               end if;
               else if ( ShiftCounter = 8 ) then
                  ShiftRegister<=Read_Ram_in_Spi;
                  End_Shift_SPI<= '0';
                  ShiftCounter <= 0;
               end if;
            end if;
          end if;
   end if;


end process;

end Behavioral;
```

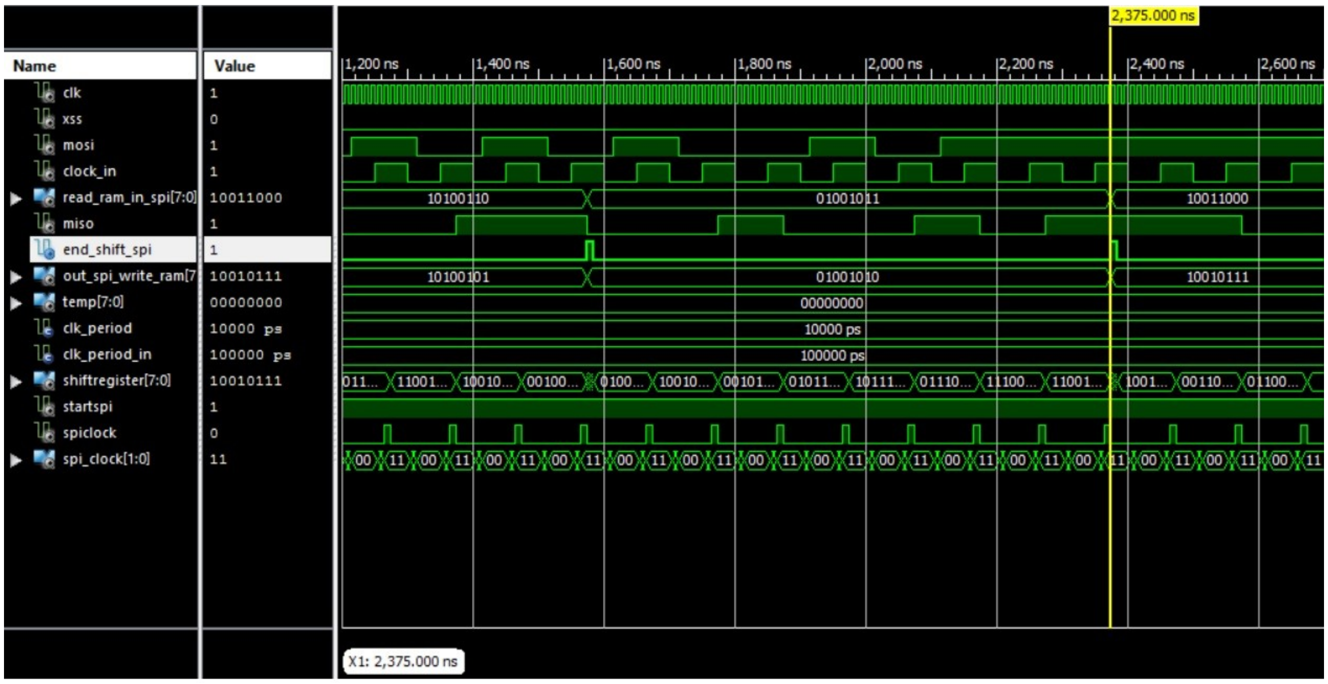
```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
ENTITY tb_SPI_Controller IS
END tb_SPI_Controller;
ARCHITECTURE behavior OF tb_SPI_Controller IS
    COMPONENT SPI
    PORT(
         CLk : IN  std_logic;
         XSS : IN  std_logic;
         MOSI : IN  std_logic;
         clock_in : IN  std_logic;
         MISO : OUT  std_logic;
         End_Shift_SPI : OUT  std_logic;
         Read_Ram_in_Spi : IN  std_logic_vector(7 downto 0);
         Out_Spi_Write_Ram : OUT  std_logic_vector(7 downto 0)
        );
    END COMPONENT;
   signal CLk : std_logic := '0';
   signal XSS : std_logic := '1';
   signal MOSI : std_logic := '0';
   signal clock_in : std_logic := '0';
   signal Read_Ram_in_Spi : std_logic_vector(7 downto 0) := (others => '0');


   signal MISO : std_logic;
   signal End_Shift_SPI : std_logic;
   signal Out_Spi_Write_Ram : std_logic_vector(7 downto 0);
   signal temp : std_logic_vector(7 downto 0) :=(others => '0') ;

   -- Clock period definitions
   constant CLk_period : time := 10 ns;
   constant CLk_period_in : time := 100 ns;


BEGIN
    uut: SPI PORT MAP (
          CLk => CLk,
          XSS => XSS,
          MOSI => MOSI,
          clock_in => clock_in,
          MISO => MISO,
          End_Shift_SPI => End_Shift_SPI,
          Read_Ram_in_Spi => Read_Ram_in_Spi,
          Out_Spi_Write_Ram => Out_Spi_Write_Ram
        );
      ADD_process :process(End_Shift_SPI)
      begin
      if( End_Shift_SPI = '1')then
      Read_Ram_in_Spi <= Out_Spi_Write_Ram+1 ;


      end if;
      end process;

   CLk_process :process
   begin
      CLk <= '0';
      wait for CLk_period/2;
      CLk <= '1';
      wait for CLk_period/2;
   end process;

   CLk_in_process :process
      begin
         clock_in <= '0';
         wait for CLk_period_in/2;
         clock_in <= '1';
         wait for CLk_period_in/2;
```



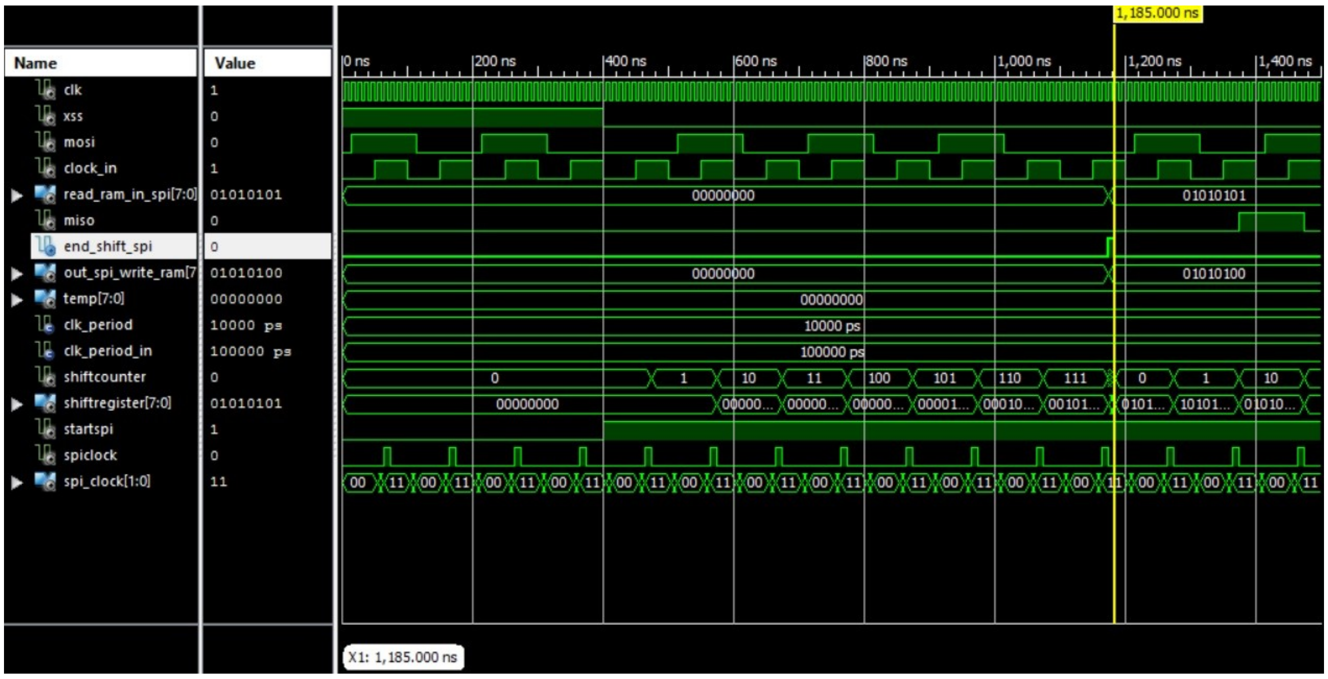
Name	Value
clk	1
xss	0
mosi	1
clock_in	1
read_ram_in_spi[7:0]	10100110
miso	0
end_shift_spi	1
out_spi_write_ram[7	10100101
temp[7:0]	00000000
clk_period	10000 ps
clk_period_in	100000 ps
shiftregister[7:0]	10100101
startspi	1
spiclock	0
spi_clock[1:0]	11



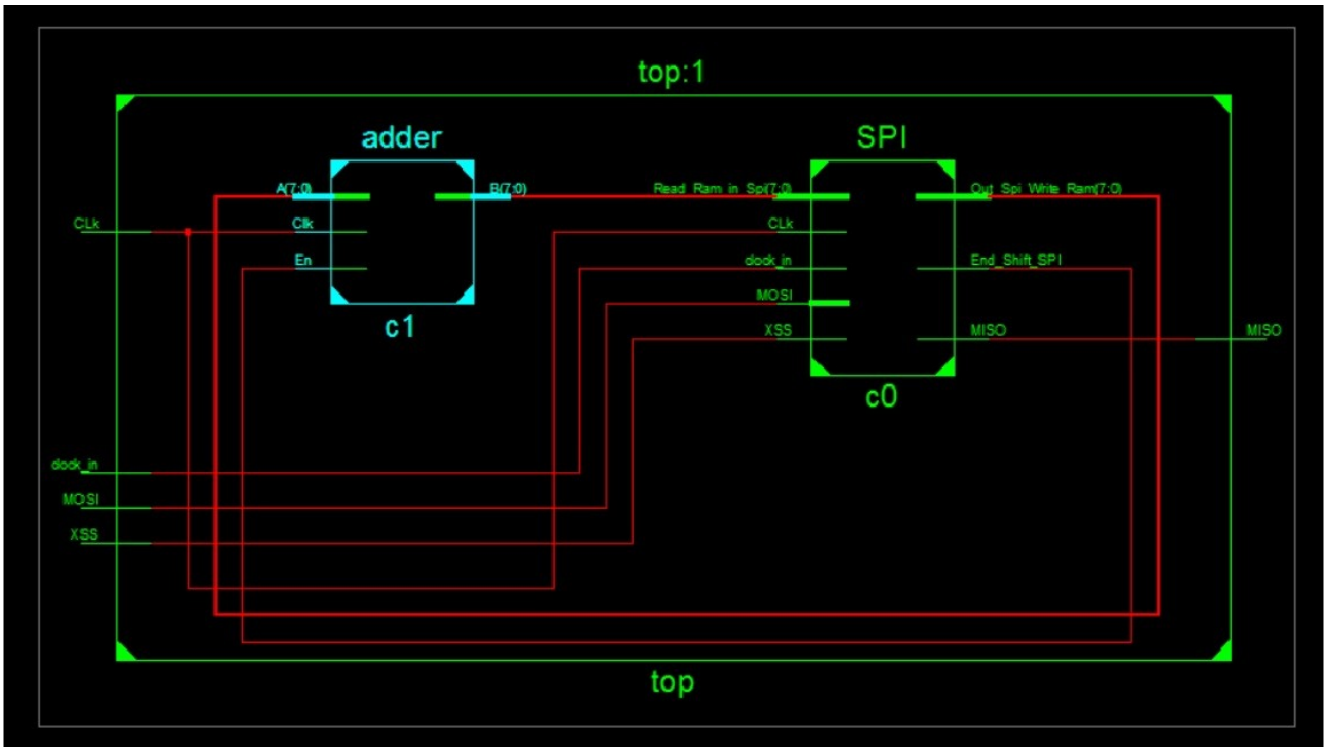
Name	Value
clk	1
xss	0
mosi	0
clock_in	1
read_ram_in_spi[7:0]	01001011
miso	0
end_shift_spi	1
out_spi_write_ram[7	01001010
temp[7:0]	00000000
clk_period	10000 ps
clk_period_in	100000 ps
shiftregister[7:0]	01001010
startspi	1
spiclock	0
spi_clock[1:0]	11

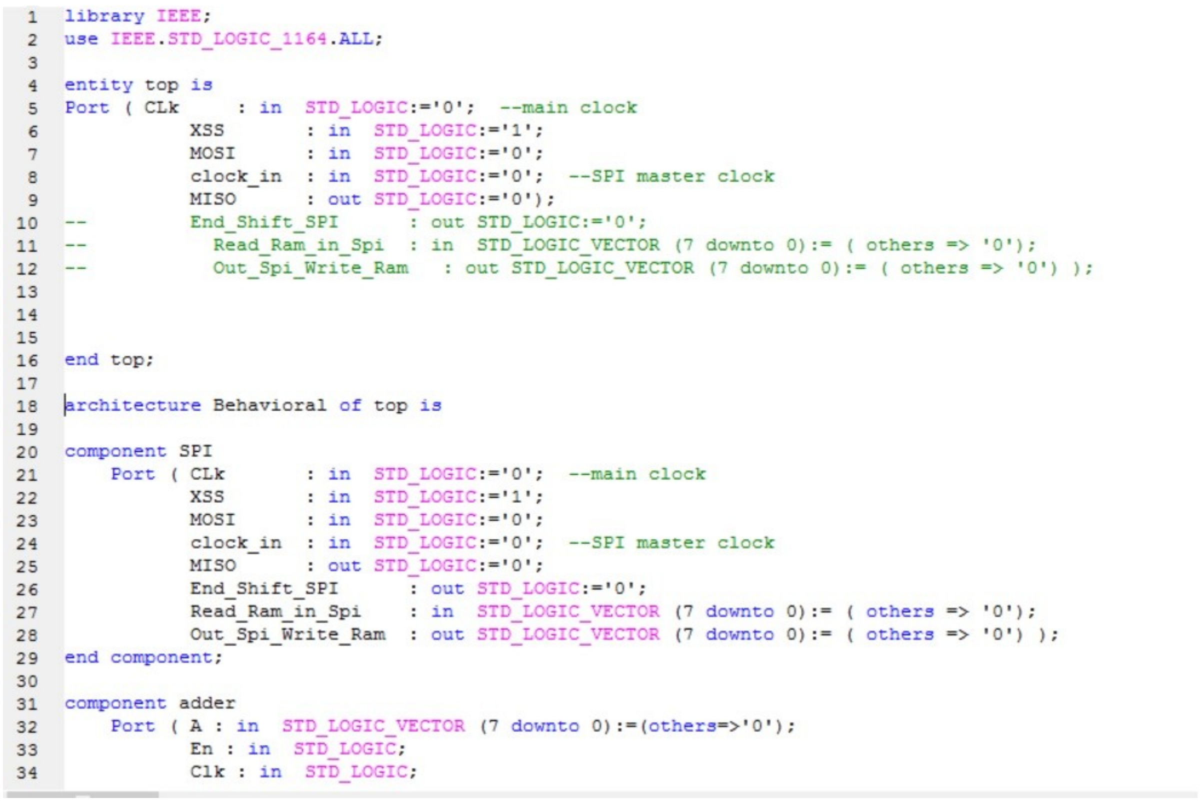


Name	Value
clk	1
xss	0
mosi	1
clock_in	1
read_ram_in_spi[7:0]	10011000
miso	1
end_shift_spi	1
out_spi_write_ram[7	10010111
temp[7:0]	00000000
clk_period	10000 ps
clk_period_in	100000 ps
shiftregister[7:0]	10010111
startspi	1
spiclock	0
spi_clock[1:0]	11

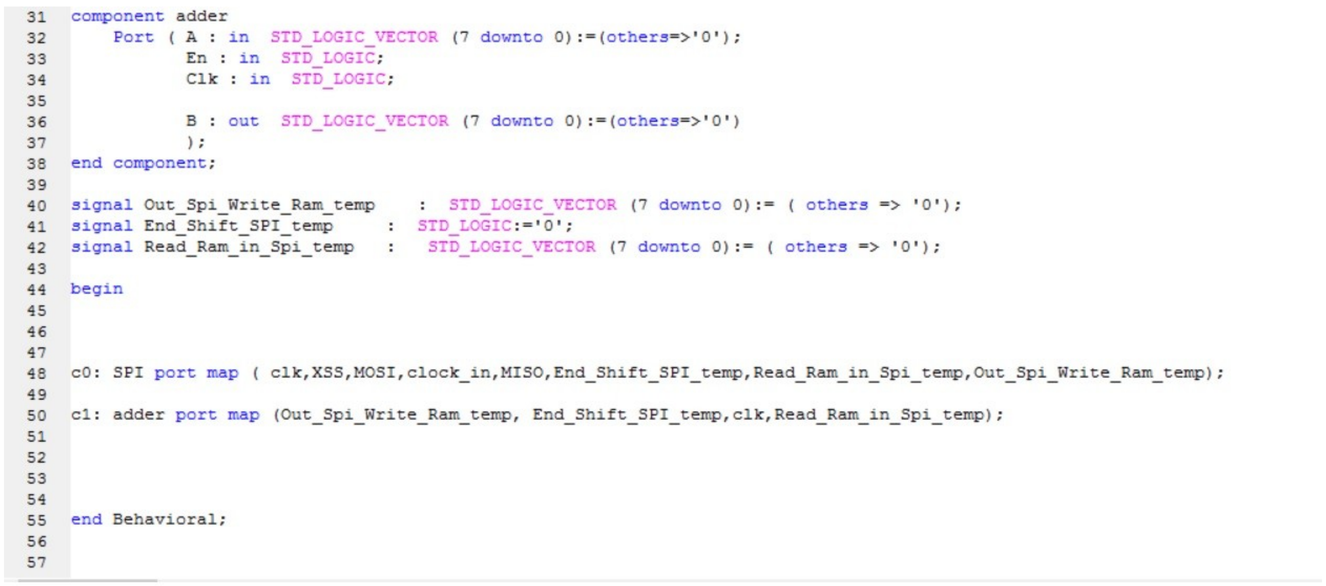


Name	Value
clk	1
xss	0
mosi	0
clock_in	1
read_ram_in_spi[7:0]	01010101
miso	0
end_shift_spi	0
out_spi_write_ram[7	01010100
temp[7:0]	00000000
clk_period	10000 ps
clk_period_in	100000 ps
shiftcounter	0
shiftregister[7:0]	01010101
startspi	1
spiclock	0
spi_clock[1:0]	11



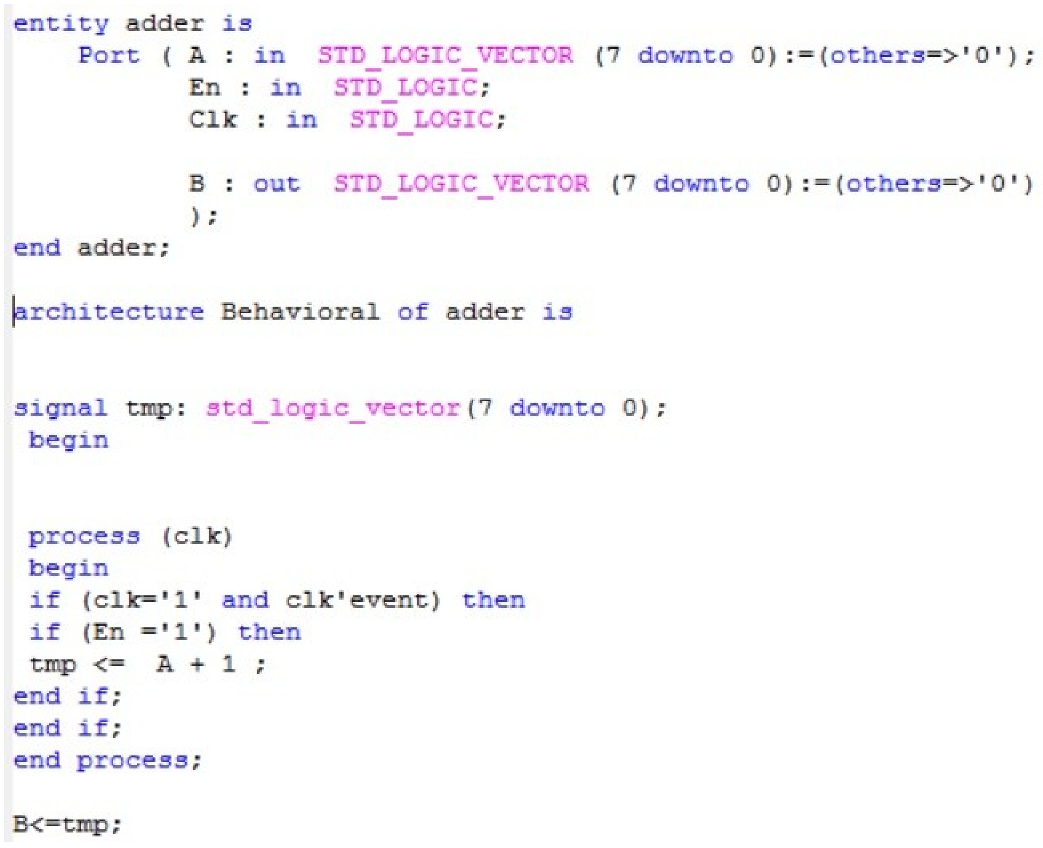


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity top is
Port ( CLk      : in  STD_LOGIC:='0';  --main clock
           XSS       : in  STD_LOGIC:='1';
           MOSI      : in  STD_LOGIC:='0';
           clock_in  : in  STD_LOGIC:='0';  --SPI master clock
           MISO      : out STD_LOGIC:='0');
--         End_Shift_SPI      : out STD_LOGIC:='0';
--           Read_Ram_in_Spi   : in  STD_LOGIC_VECTOR (7 downto 0):= ( others => '0');
--           Out_Spi_Write_Ram   : out STD_LOGIC_VECTOR (7 downto 0):= ( others => '0') );



end top;

architecture Behavioral of top is

component SPI
    Port ( CLk       : in  STD_LOGIC:='0';  --main clock
           XSS       : in  STD_LOGIC:='1';
           MOSI      : in  STD_LOGIC:='0';
           clock_in  : in  STD_LOGIC:='0';  --SPI master clock
           MISO      : out STD_LOGIC:='0';
           End_Shift_SPI      : out STD_LOGIC:='0';
           Read_Ram_in_Spi    : in  STD_LOGIC_VECTOR (7 downto 0):= ( others => '0');
           Out_Spi_Write_Ram  : out STD_LOGIC_VECTOR (7 downto 0):= ( others => '0') );
end component;

component adder
    Port ( A : in  STD_LOGIC_VECTOR (7 downto 0):=(others=>'0');
           En : in  STD_LOGIC;
           Clk : in  STD_LOGIC;
```

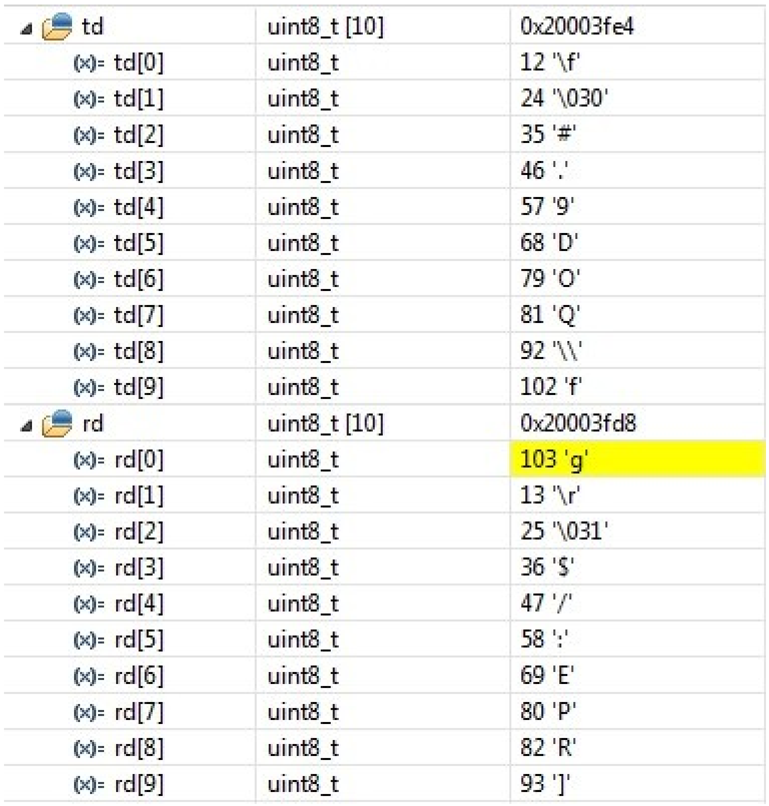


```
component adder
    Port ( A : in  STD_LOGIC_VECTOR (7 downto 0):=(others=>'0');
           En : in  STD_LOGIC;
           Clk : in  STD_LOGIC;

           B : out  STD_LOGIC_VECTOR (7 downto 0):=(others=>'0')
           );
end component;

signal Out_Spi_Write_Ram_temp    :  STD_LOGIC_VECTOR (7 downto 0):= ( others => '0');
signal End_Shift_SPI_temp     :  STD_LOGIC:='0';
signal Read_Ram_in_Spi_temp   :   STD_LOGIC_VECTOR (7 downto 0):= ( others => '0');

begin



c0: SPI port map ( clk,XSS,MOSI,clock_in,MISO,End_Shift_SPI_temp,Read_Ram_in_Spi_temp,Out_Spi_Write_Ram_temp);

c1: adder port map (Out_Spi_Write_Ram_temp, End_Shift_SPI_temp,clk,Read_Ram_in_Spi_temp);




end Behavioral;
```

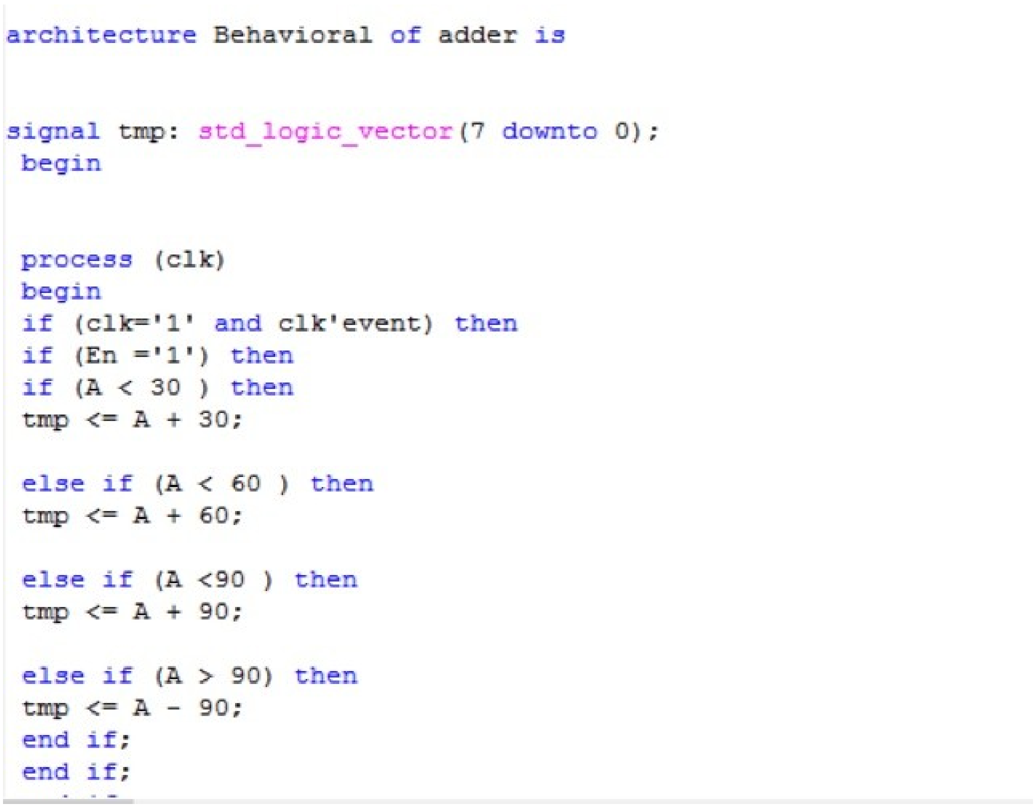
تست عملکرد SPI controller و ارتباط FPGA با میکرو MKV



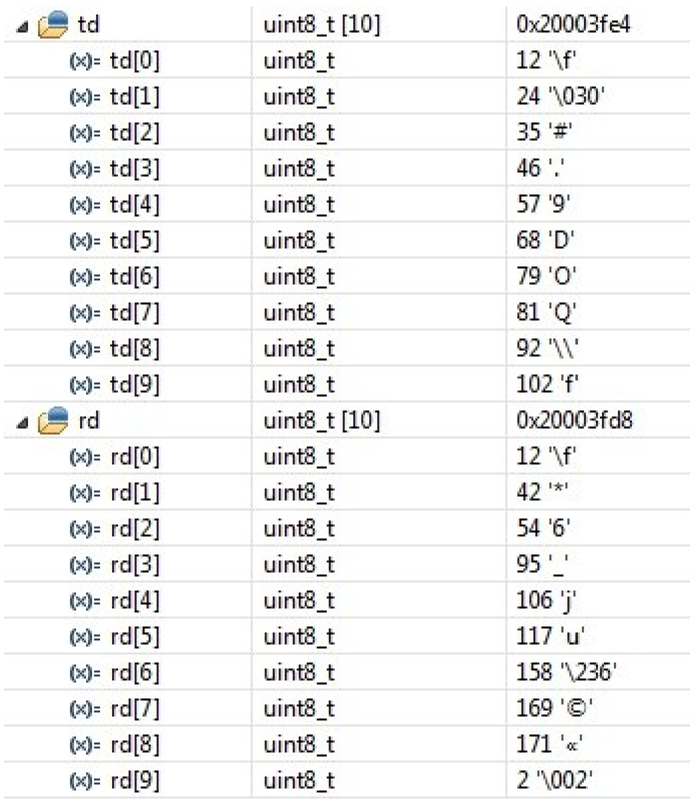
```
entity adder is
    Port ( A : in  STD_LOGIC_VECTOR (7 downto 0):=(others=>'0');
           En : in  STD_LOGIC;
           Clk : in  STD_LOGIC;

           B : out  STD_LOGIC_VECTOR (7 downto 0):=(others=>'0')
           );
end adder;

architecture Behavioral of adder is


signal tmp: std_logic_vector(7 downto 0);
 begin


 process (clk)
 begin
 if (clk='1' and clk'event) then
 if (En ='1') then
 tmp <=  A + 1 ;
end if;
end if;
end process;

B<=tmp;
```



td	uint8_t [10]	0x20003fe4
(x)= td[0]	uint8_t	12 '\f'
(x)= td[1]	uint8_t	24 '\030'
(x)= td[2]	uint8_t	35 '#'
(x)= td[3]	uint8_t	46 '.'
(x)= td[4]	uint8_t	57 '9'
(x)= td[5]	uint8_t	68 'D'
(x)= td[6]	uint8_t	79 'O'
(x)= td[7]	uint8_t	81 'Q'
(x)= td[8]	uint8_t	92 '\\'
(x)= td[9]	uint8_t	102 'f'
rd	uint8_t [10]	0x20003fd8
(x)= rd[0]	uint8_t	103 'g'
(x)= rd[1]	uint8_t	13 '\r'
(x)= rd[2]	uint8_t	25 '\031'
(x)= rd[3]	uint8_t	36 '$'
(x)= rd[4]	uint8_t	47 '/'
(x)= rd[5]	uint8_t	58 ':'
(x)= rd[6]	uint8_t	69 'E'
(x)= rd[7]	uint8_t	80 'P'
(x)= rd[8]	uint8_t	82 'R'
(x)= rd[9]	uint8_t	93 ']'



```
architecture Behavioral of adder is


signal tmp: std_logic_vector(7 downto 0);
 begin


 process (clk)
 begin
 if (clk='1' and clk'event) then
 if (En ='1') then
 if (A < 30 ) then
 tmp <= A + 30;

 else if (A < 60 ) then
 tmp <= A + 60;

 else if (A <90 ) then
 tmp <= A + 90;

 else if (A > 90) then
 tmp <= A - 90;
 end if;
 end if;
```



```
td          uint8_t [10]    0x20003fe4
   td[0]    uint8_t         12 '\f'
   td[1]    uint8_t         24 '\030'
   td[2]    uint8_t         35 '#'
   td[3]    uint8_t         46 '.'
   td[4]    uint8_t         57 '9'
   td[5]    uint8_t         68 'D'
   td[6]    uint8_t         79 'O'
   td[7]    uint8_t         81 'Q'
   td[8]    uint8_t         92 '\\'
   td[9]    uint8_t         102 'f'
rd          uint8_t [10]    0x20003fd8
   rd[0]    uint8_t         12 '\f'
   rd[1]    uint8_t         42 '*'
   rd[2]    uint8_t         54 '6'
   rd[3]    uint8_t         95 '_'
   rd[4]    uint8_t         106 'j'
   rd[5]    uint8_t         117 'u'
   rd[6]    uint8_t         158 '\236'
   rd[7]    uint8_t         169 '©'
   rd[8]    uint8_t         171 '«'
   rd[9]    uint8_t         2 '\002'
```