8-bit Shift Register implementation using HDL and FPGA

Abstract: This project demonstrates the use of hardware description language (HDL) to model digital circuitry on a field programmable gate array (FPGA).

Hardware and Software used: Atlys Spartan-6 FPGA Trainer Board, ISE Design Suit 14.7, Digilent Adept.

Description

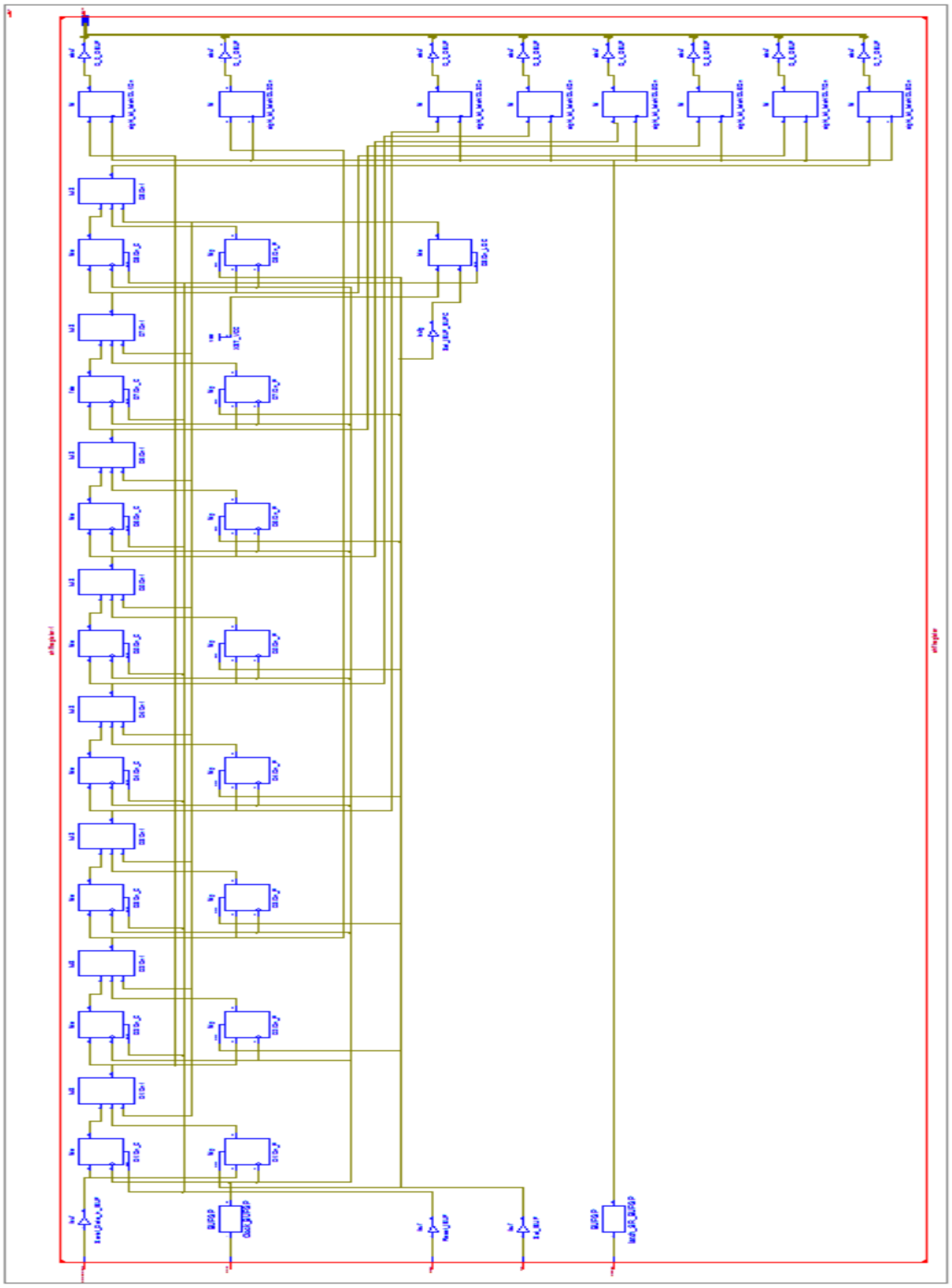
This project makes use of a Xilinx Spartan-6 FPGA to implement an 8-bit SIPO shift register with set, reset and latch functionality.

- There are a total of 5 inputs – Data, Clock, Latch, Set, and Reset which are connected to onboard switches.
- The outputs of the shift register are serially connected to 8 onboard led's.
- Data can be serially inserted into the shift register using clock and data lines.
- A High signal on 'Reset' sets the output of the shift register to '0' irrespective of the current state of the inputs.
- A High signal on 'Set' sets the output of shift register to '1' if Reset is not High.
- A rising edge on the clock sets the LSB of the output to the data input and right shifts the remaining bits.
- The output of the shift register will be visible only when there is a rising edge of the signal on the 'Latch' input.

Implementation:

- Behavioral VHDL (VHSIC Hardware Description Language) programming is used to describe a D Flip-Flop.
- The D Flip-Flops have a set and a reset terminal.
  1. A High signal on 'Reset' sets the output of the Flip-flop to '0' irrespective of the current state of the inputs.
  2. A High signal on 'Set' sets the output of the Flip-flop to '1' if Reset is not High.
  3. A rising edge on the clock sets the output to the data input if Reset and Set are Low.
- A series of cascading D Flip-Flops is used to create a 8-bit shift register using structural VHDL programming.
- The outputs of the shift register is given to an 8-bit latch. The 8 bit latch is also designed using structural programming. When the 'Latch' faces a rising edge on its input the output of the shift register is given to the led's.

Technology schematic of the shift register



Various levels of RTL schematic.

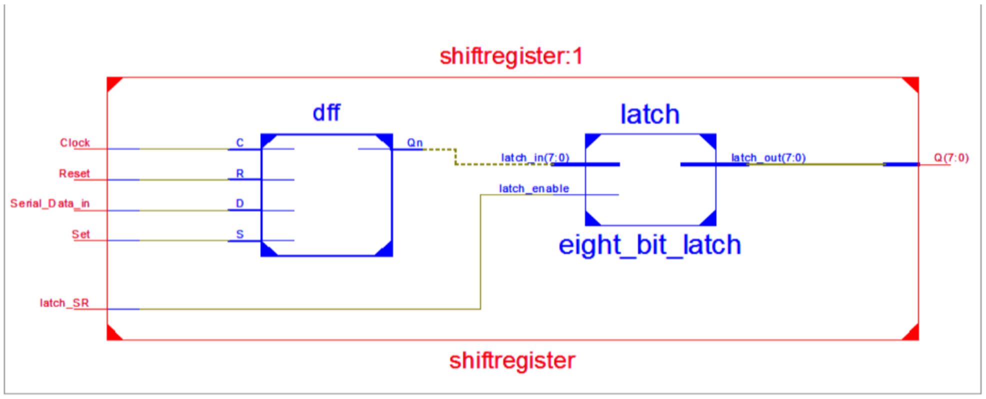


Figure 1

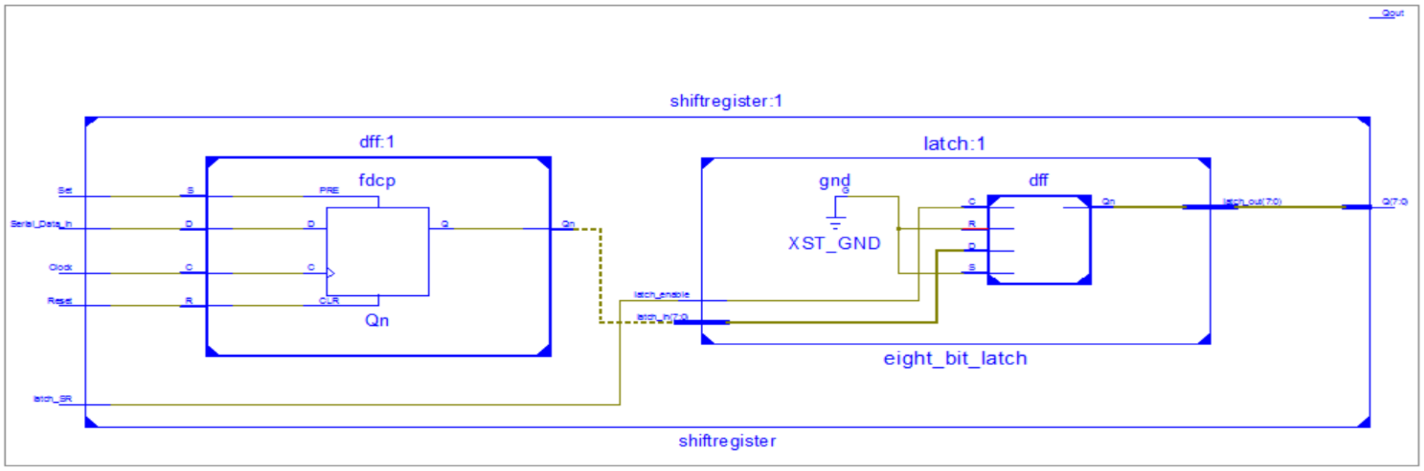


Figure 2

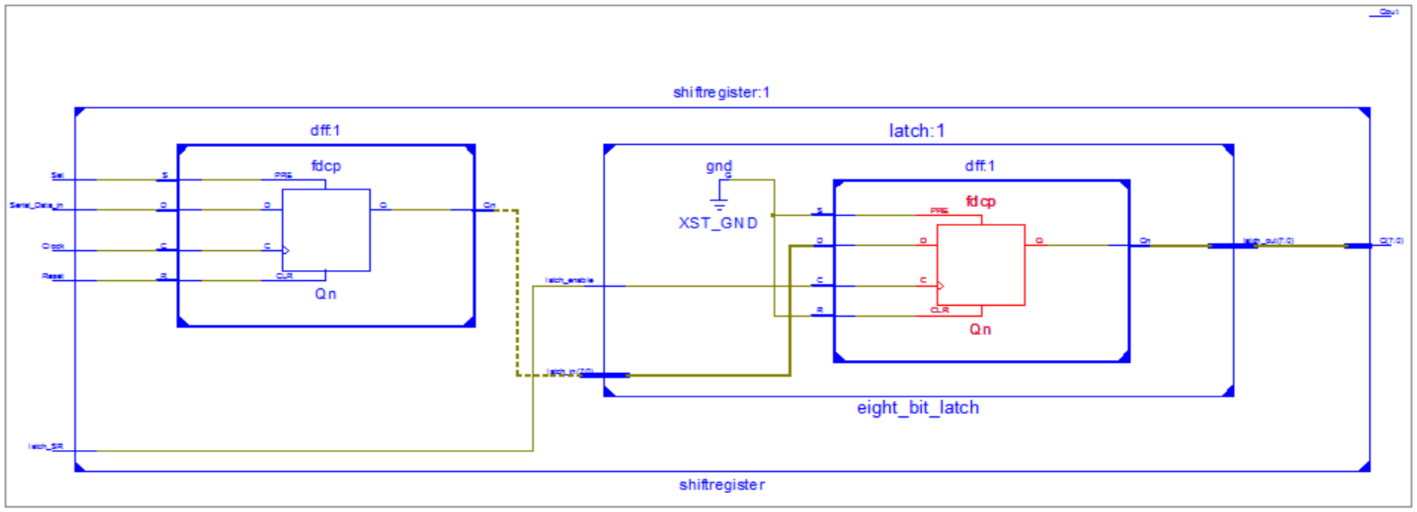
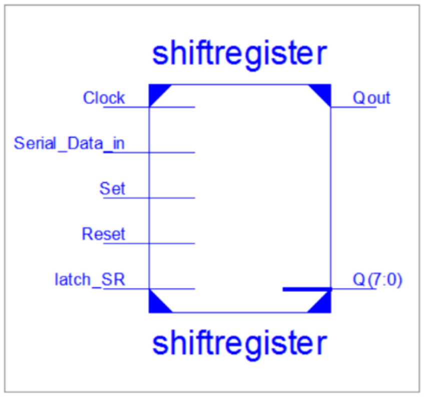


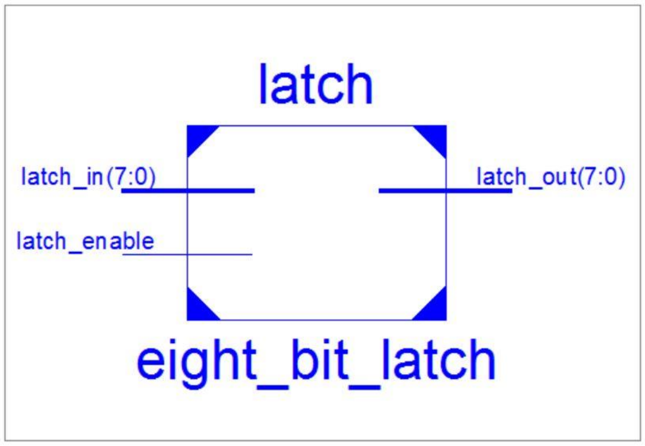
Figure 3

Code for the Shift Register.



```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use ieee.numeric_std.all;

****************************
entity shiftregister is

port( Q: inout std_logic_vector(7 downto 0);
                    Clock: in std_logic;
                    Serial_Data_in: in std_logic;
                    Set: in std_logic;
                    Reset: in std_logic;
                    Qout: out std_logic;
                    latch_SR: in std_logic);
end shiftregister;


***************************
architecture Behavioral of shiftregister is
-------------------------------
 component dff
 port(Qn: out std_logic;
        C,R,D,S: in std_logic);
 end component;


 component latch
 port(latch_enable: in std_logic;
       latch_in: in std_logic_vector(7 downto 0);
       latch_out: out std_logic_vector(7 downto 0));
 end component;
-------------------------------

signal Sig: std_logic_vector(7 downto 0);


begin
 D1:dff port map (Qn=>Sig(0), C=>Clock, R=>reset,D=>Serial_Data_in,S=>Set);
 D2:dff port map (Qn=>Sig(1), C=>Clock, R=>reset,D=>Sig(0),S=>Set);
 D3:dff port map (Qn=>Sig(2), C=>Clock, R=>reset,D=>Sig(1),S=>Set);
 D4:dff port map (Qn=>Sig(3), C=>Clock, R=>reset,D=>Sig(2),S=>Set);
 D5:dff port map (Qn=>Sig(4), C=>Clock, R=>reset,D=>Sig(3),S=>Set);
 D6:dff port map (Qn=>Sig(5), C=>Clock, R=>reset,D=>Sig(4),S=>Set);
 D7:dff port map (Qn=>Sig(6), C=>Clock, R=>reset,D=>Sig(5),S=>Set);
 D8:dff port map (Qn=>Sig(7), C=>Clock, R=>reset,D=>Sig(6),S=>Set);

 eight_bit_latch: latch port map
 (latch_enable=>latch_SR, latch_in=>Sig, latch_out=>Q);

 --D9:dff port map (Qn=>Qout, C=>Clock, R=>reset,D=>Q(7),S=>Set);

end Behavioral;
```

Code for the 8-bit Latch.

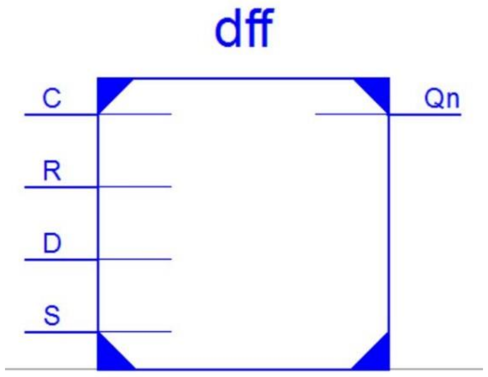


It makes use of the component dff which is a positive edge triggered D flip-flop.

1. It makes use of 8 D Flip-Flops.
2. It has 8 bit input data bus and an 8 bit output data bus.
3. When the input ‘latch_enable’ faces a rising edge the input is latched to the output.

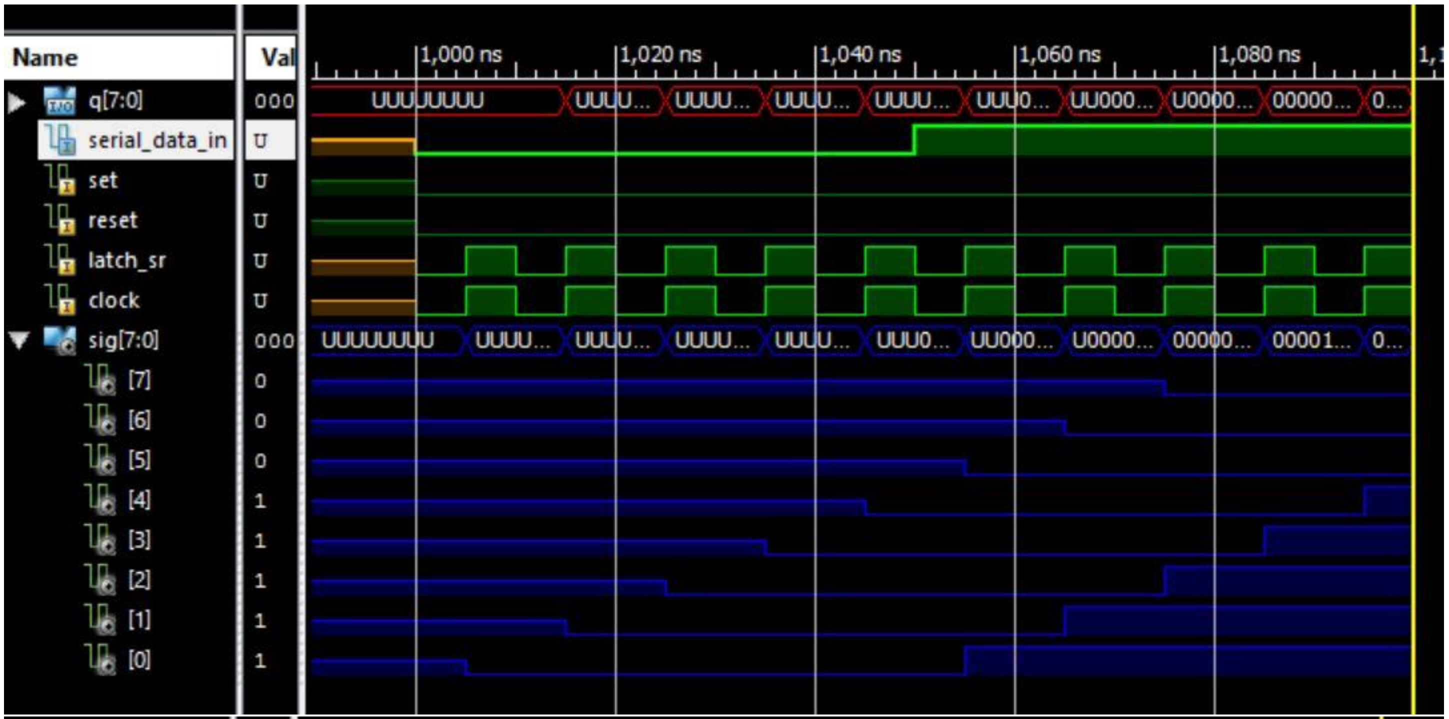
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

****************************
entity latch is
 port (  latch_enable: in std_logic;
         latch_in: in std_logic_vector(7 downto 0);
         latch_out: out std_logic_vector(7 downto 0)
         );
end latch;


****************************
architecture Behavioral of latch is
------------------------------
 component dff
 port(Qn: out std_logic;
                C,R,D,S: in std_logic);
 end component;
-------------------------------

begin
 DL1:dff port map (Qn=>latch_out(0), C=>latch_enable, R=>'0',D=>latch_in(0),S=>'0');
 DL2:dff port map (Qn=>latch_out(1), C=>latch_enable, R=>'0',D=>latch_in(1),S=>'0');
 DL3:dff port map (Qn=>latch_out(2), C=>latch_enable, R=>'0',D=>latch_in(2),S=>'0');
 DL4:dff port map (Qn=>latch_out(3), C=>latch_enable, R=>'0',D=>latch_in(3),S=>'0');
 DL5:dff port map (Qn=>latch_out(4), C=>latch_enable, R=>'0',D=>latch_in(4),S=>'0');
 DL6:dff port map (Qn=>latch_out(5), C=>latch_enable, R=>'0',D=>latch_in(5),S=>'0');
 DL7:dff port map (Qn=>latch_out(6), C=>latch_enable, R=>'0',D=>latch_in(6),S=>'0');
 DL8:dff port map (Qn=>latch_out(7), C=>latch_enable, R=>'0',D=>latch_in(7),S=>'0');
end Behavioral;
```

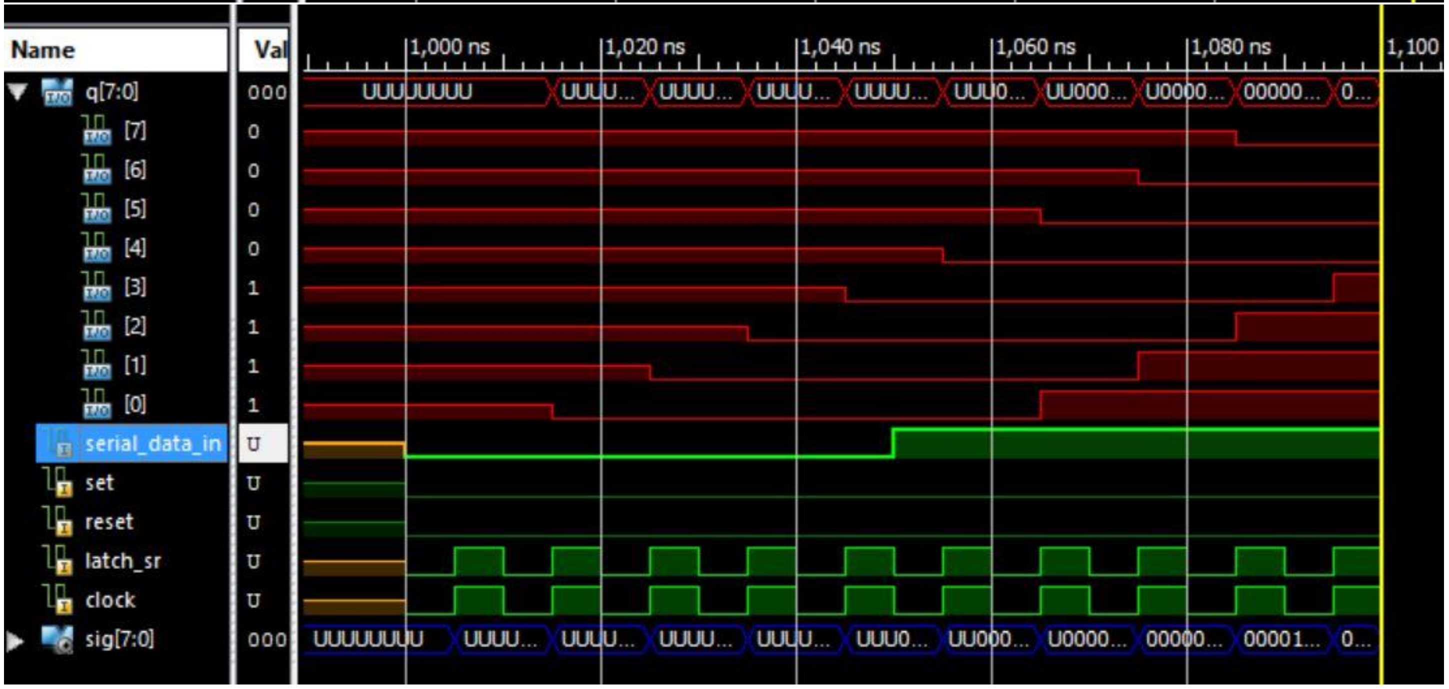
Code for D Flip-Flop.



```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity dff is

port (

Qn:out std_logic;
C: in std_logic;
R: in std_logic;
D: in std_logic;
S: in std_logic);
end dff;


architecture Behavioral of dff is

begin
process (R, C, S)
begin
            if(R = '1') then
            Qn <= '0';

            else if (S = '1') then
   Qn <= '1';

            else if(rising_edge(C)) then
    Qn <= D;
            end if;
            end if;
            end if;
end process;
end Behavioral;
```

Simulation:


Name
Val
1,000 ns
1,020 ns
1,040 ns
1,060 ns
1,080 ns
q[7:0] 000
serial_data_in U
set U
reset U
latch_sr U
clock U
sig[7:0] 000
[7] 0
[6] 0
[5] 0
[4] 1
[3] 1
[2] 1
[1] 1
[0] 1
UUUUUUUU
UUUU...
UUU0...
UU000...
U0000...
00000...
00001...
0...


Name
Val
1,000 ns
1,020 ns
1,040 ns
1,060 ns
1,080 ns
1,100
q[7:0] 000
[7] 0
[6] 0
[5] 0
[4] 0
[3] 1
[2] 1
[1] 1
[0] 1
serial_data_in U
set U
reset U
latch_sr U
clock U
sig[7:0] 000
UUUUUUUU
UUUU...
UUU0...
UU000...
U0000...
00000...
00001...
0...

Learning Experience:

1. We learnt the function and use of structural programming to implement digital circuit designs.
2. How to configure the Spartan 6 board from a USB drive and the ROM of the Atlys board.
3. The working and design of a SIPO shift register.
4. Some of the hardware restrictions of the FPGA and how to effectively debug the code and the circuit.
5. How to test your circuit by running simulations.