ECE 501

Contemporary Digital Systems

Prof. Eric J. Balster

Homework #3

Aditya Sankaramanchi

101266059

**Contents** **Page**

Introduction 3

Design 3

Results 5

Conclusion 7

Appendix VHDL code

Design file 8

Test bench 13

.tcl script file 18

**Introduction:**

A barrel shifter is a digital circuit that can implement arithmetic shifting, logical shifting and rotatory functions by a specified number of bits in one clock cycle. The functionality is obtained by a sequence of multiplexers connected such that the output of one mux is connected to the input of the next mux in a way that depends on the shift distance. The barrel shifter is used in many digital circuits of digital signal processors and microprocessors.

**Design:**

Barrel shifter is similar to a universal shift register (USR). The digital circuit should shift a data word by a specified number of bits in a single clock cycle.

Consider a barrel shifter with an N-bit input vector. The output is a shifted form of the input. The shift amount is specified by the shift input (from 0 to N-1).

The truth table for the barrel shifter is shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Time | Input | Register | Select | Shift |
| 0 | 0101 1111 | XXXX XXXX | 3 (load) | 3 |
| 1 | 0000 0000 | 0101 1111 | 0 (hold) | 3 |
| 2 | 0000 0000 | 0101 1111 | 2 (sl) | 3 |
| 3 | 0000 0000 | 1111 1010 | 1 (sr) | 4 |
| 4 | 0000 0000 | 1010 1111 | 1 (sr) | 1 |
| 5 | 0000 0000 | 1101 0111 | 2 (sl) | 2 |
| 6 | 0000 0000 | 0101 1111 | 0 | 0 |

Table 1: Truth table for the barrel shifter

The following logic is implemented in the VHDL code:

The main control of the design logic of the barrel shifter depends on select (S). It has the values from 00 to 11. The following table shows the functionality:

|  |  |
| --- | --- |
| S | Value |
| 00 | 0 (hold) |
| 01 | 1 (shift right) |
| 10 | 2 (left shift) |
| 11 | 3 (load) |

Table 2: Main Operation control

The shift specifies the number of bits to be shifted either right or left. It has values from 0 to 4.

A new type of test bench is written where the input data and expected data are read from files input.csv and output.csv. Input.csv will drive the barrel shifter and the output.csv will check for proper output.

For the test bench to read the text files input.csv and output.csv, an ASCII to hex converter is needed. So this is included in a package work.sim\_mem\_init.vhd. This package is used for memory initialization and ASCII to hex conversion.

**Results:**

The simulation result of the barrel shifter is shown below:

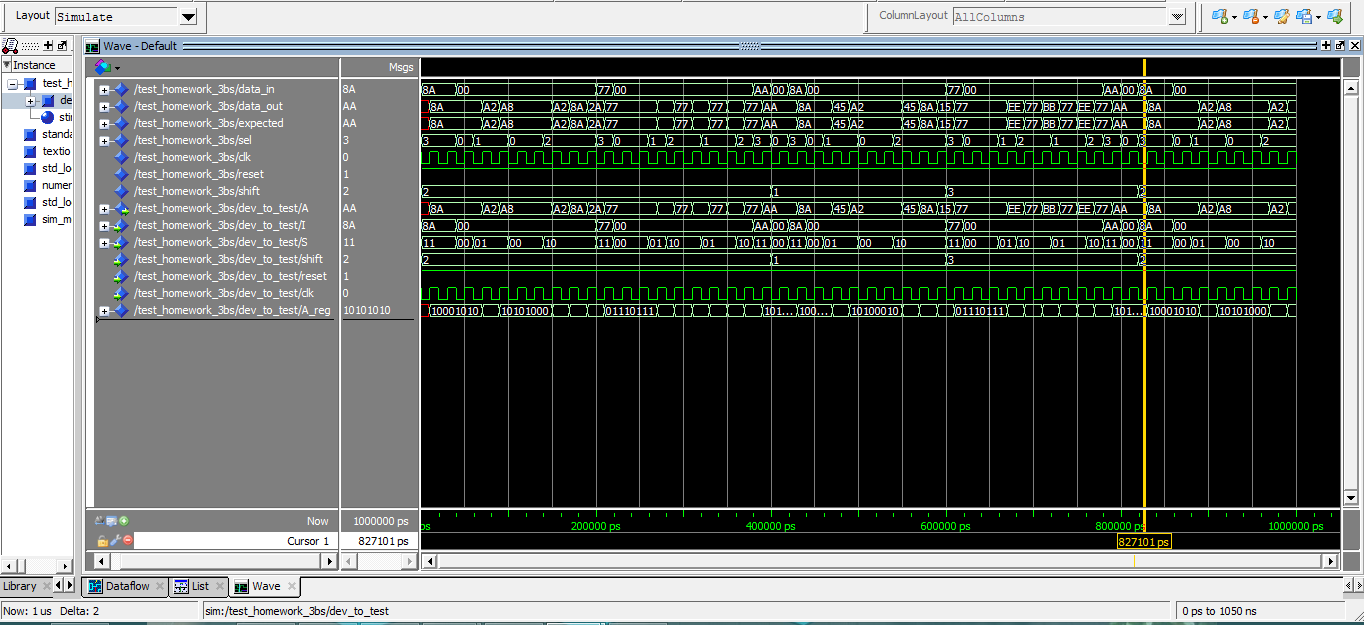


Figure 1: Simulation result for barrel shifter

The transcript window is shown below:

 Figure 2. Transcript window of the ModelSim simulation

Quartus output:

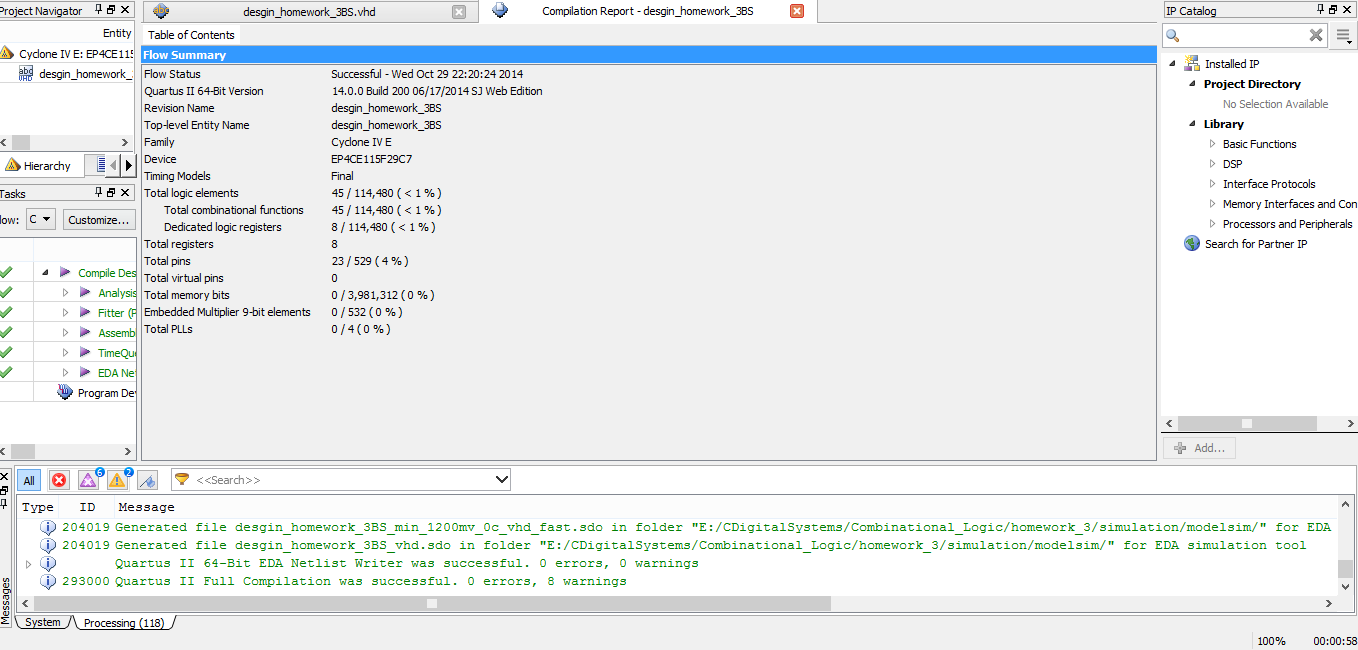


Figure 3: Quartus result of the barrel shifter

Total logic elements = 45

Total combinational functions = 45

Dedicated logic registers = 8

Total registers = 8

Total pins = 23

**Conclusion:**

For the given truth table of the barrel shifter, the VHDL code is written in the behavioral design. The operations like hold, shift right, shift left and load are performed. Maximum of four shifts are implemented for the right and left shifts. A test bench is used to check for the correctness of every shift operation. A new type of test bench is written in such a way that the input data and expected output data are read from files input.csv and output.csv. The sim\_mem\_init.vhd is a package that is included in the test bench which is used for memory initialization and ASCII to hex conversion. A .tcl script file is executed to simplify the compiling and simulation process. The simulation is done using ModelSim and the output waveforms are obtained. The output satisfies the required functionality. Further, Quartus is used to synthesize the design.

**Appendix – VHDL code:**

desgin\_homework\_3BS.vhd

-- Barrel Shifter Homework #3

library IEEE;

use IEEE.STD\_logic\_1164.all;

use IEEE.numeric\_std.all;

entity desgin\_homework\_3BS is

generic (

bit\_depth : integer := 8);

port(

A : out std\_logic\_vector(bit\_depth-1 downto 0);

I : in std\_logic\_vector(bit\_depth-1 downto 0);

S : in std\_logic\_vector(1 downto 0);

shift : in integer range 0 to bit\_depth-1;

reset : in std\_logic;

clk : in std\_logic);

end desgin\_homework\_3BS;

architecture behavior of desgin\_homework\_3BS is

signal A\_reg : std\_logic\_vector(bit\_depth-1 downto 0);

begin

A <= A\_reg;

desgin\_homework\_3BS\_porc: process(clk, reset)

begin

if(rising\_edge(clk)) then

if(reset = '0') then

A\_reg <= (others => '0');

else

case S is

when "00" => -- for hold

A\_reg <= A\_reg;

when "01" => -- for shift right

case shift is

when 0 =>

A\_reg <= A\_reg ;

when 1 =>

A\_reg(bit\_depth-1) <= A\_reg(0);

A\_reg(bit\_depth-2 downto 0) <= A\_reg(bit\_depth-1 downto 1);

when 2 =>

A\_reg(bit\_depth-1 downto bit\_depth-2) <= A\_reg(1 downto 0);

A\_reg(bit\_depth-3 downto 0) <= A\_reg(bit\_depth-1 downto 2);

when 3 =>

A\_reg(bit\_depth-1 downto bit\_depth-3) <= A\_reg(2 downto 0);

A\_reg(bit\_depth-4 downto 0) <= A\_reg(bit\_depth-1 downto 3);

when 4 =>

A\_reg(bit\_depth-1 downto bit\_depth-4) <= A\_reg( 3 downto 0);

A\_reg(bit\_depth-5 downto 0) <= A\_reg(bit\_depth-1 downto 4);

when others =>

A\_reg <= (others => 'X');

end case;

when "10" => -- for left shift

case shift is

when 0 =>

A\_reg <= A\_reg;

when 1 =>

A\_reg(0) <= A\_reg(bit\_depth-1);

A\_reg(bit\_depth-1 downto 1) <= A\_reg(bit\_depth-2 downto 0);

when 2 =>

A\_reg(1 downto 0) <= A\_reg(bit\_depth-1 downto bit\_depth-2);

A\_reg(bit\_depth-1 downto 2) <= A\_reg(bit\_depth-3 downto 0);

when 3 =>

A\_reg(2 downto 0) <= A\_reg(bit\_depth-1 downto bit\_depth-3);

A\_reg(bit\_depth-1 downto 3) <= A\_reg(bit\_depth-4 downto 0);

when 4 =>

A\_reg(3 downto 0) <= A\_reg(bit\_depth-1 downto bit\_depth-4);

A\_reg(bit\_depth-1 downto 4) <= A\_reg(bit\_depth-5 downto 0);

when others =>

A\_reg <= (others => 'X');

end case;

when "11" =>

A\_reg <= I;

when others =>

A\_reg <= (others => 'X');

end case;

end if;

end if;

end process desgin\_homework\_3BS\_porc;

end behavior;

test\_homework\_3BS.vhd

-- test for hw3 barrel shifter

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

use std.textio.all;

use IEEE.std\_logic\_textio.all;

use work.sim\_mem\_init.all;

entity test\_homework\_3BS is

end;

architecture test of test\_homework\_3BS is

component desgin\_homework\_3BS

generic (

bit\_depth : integer := 8);

port(

A : out std\_logic\_vector(bit\_depth-1 downto 0);

I : in std\_logic\_vector(bit\_depth-1 downto 0);

S : in std\_logic\_vector(1 downto 0);

shift : in integer range 0 to bit\_depth-1;

reset : in std\_logic;

clk : in std\_logic);

end component;

constant data\_width : integer :=8;

signal data\_in : std\_logic\_vector(data\_width-1 downto 0);

signal data\_out : std\_logic\_vector(data\_width-1 downto 0);

signal expected : std\_logic\_vector(data\_width-1 downto 0);

signal sel : std\_logic\_vector(1 downto 0);

signal clk : std\_logic := '1';

signal reset : std\_logic := '1';

signal shift : integer range 0 to data\_width-1;

constant in\_fname : string := "input.csv";

constant out\_fname : string := "output.csv";

file input\_file : text;

file output\_file : text;

begin

dev\_to\_test: desgin\_homework\_3BS

generic map(data\_width)

port map(data\_out, data\_in, sel, shift, reset, clk);

stimulus: process

variable input\_line : line;

variable WriteBuf : line;

variable in\_char : character;

variable in\_slv : std\_logic\_vector(7 downto 0);

variable out\_slv :std\_logic\_vector(7 downto 0);

variable ErrCnt : integer := 0;

begin

file\_open(input\_file, in\_fname, read\_mode);

file\_open(output\_file, out\_fname, read\_mode);

while not(endfile(input\_file)) loop

readline(input\_file, input\_line);

for i in 0 to 8 loop --

read(input\_line,in\_char);

in\_slv := std\_logic\_vector(to\_unsigned(character'pos(in\_char),8));

if( i = 3) then

data\_in(7 downto 4) <= ASCII\_to\_hex(in\_slv);

elsif(i = 4) then

data\_in(3 downto 0) <= ASCII\_to\_hex(in\_slv);

elsif (i = 6) then

sel <= in\_slv(1 downto 0);

elsif( i = 8) then

shift <= to\_integer(signed(in\_slv(2 downto 0)));

end if;

end loop;

readline(output\_file,input\_line);

clk <= '0';

wait for 10 ns;

for i in 0 to 4 loop

read(input\_line, in\_char);

out\_slv := std\_logic\_vector(to\_unsigned(character'pos(in\_char),8));

if (i =3) then

expected(7 downto 4) <= ASCII\_to\_hex(out\_slv);

elsif( i =4) then

expected(3 downto 0) <= ASCII\_to\_hex(out\_slv);

end if;

end loop;

clk <= '1';

wait for 10 ns;

if (expected /= data\_out) then

write(WriteBuf, string'(" ERROR barrel shifter failed "));

write(WriteBuf, string'(" expected = "));

write(WriteBuf, expected);

write(WriteBuf, string'(", data\_out = "));

write(WriteBuf, data\_out);

writeline(Output, WriteBuf);

ErrCnt := ErrCnt+1;

end if;

end loop;

file\_close(input\_file);

file\_close(output\_file);

if(ErrCnt = 0) then

report "Success. The barrel shifter test is completed";

else

report "The barrel shifter is broken" severity warning;

end if;

end process stimulus;

end test;

.tcl script:

-- .tcl script file for barrel shifter

transcript off

vcom sim\_mem\_init.vhd

vcom desgin\_homework\_3BS.vhd

vcom test\_homework\_3BS.vhd

vsim test\_homework\_3BS

add wave sim:/test\_homework\_3BS/dev\_to\_test/\*

run 1000 ns