**Further Digital Electronics VHDL Lab 4 Task B**

**VHDL Code for d type flipflop**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

--D type flipflop

**entity** dff **is**

**Port** **(** clk **:** **in** STD\_LOGIC**;** --clock signal

d **:** **in** STD\_LOGIC**;** --data input

q **:** **out** STD\_LOGIC**;** --data output

rst **:** **in** STD\_LOGIC**);** --reset signal

**end** dff**;**

**architecture** Behavioral **of** dff **is**

**signal** data **:** std\_logic**;**

**begin**

flip\_flop**:** **process** **(**clk**)**

**begin**

--on a rising edge

**if** **rising\_edge(**clk**)** **then**

**if** rst **=** '1' **then**

--if reset is high then set data to 0

data **<=** '0'**;**

**else**

--otherwise, take copy of input

data **<=** d**;**

**end** **if;**

**end** **if;**

**end** **process** flip\_flop**;**

--connect output

q **<=** data**;**

**end** Behavioral**;**

**VHDL code for top level shift register**

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

--Bidirectional load

**entity** bidi\_shift\_reg **is**

**generic** **(**size**:** natural **:=** 16**);**

**Port** **(** clk **:** **in** STD\_LOGIC**;**

shift\_in **:** **in** STD\_LOGIC**;**

data\_in **:** **in** STD\_LOGIC\_VECTOR **(**size**-**1 **downto** 0**);**

data\_out **:** **out** STD\_LOGIC\_VECTOR **(**size**-**1 **downto** 0**);**

ctrl **:** **in** STD\_LOGIC\_VECTOR **(**1 **downto** 0**);**

rst **:** **in** STD\_LOGIC**);**

-- type ctrl\_type is (hold, shift\_right, shift\_left, load);

**end** bidi\_shift\_reg**;**

**architecture** Behavioral **of** bidi\_shift\_reg **is**

**signal** data\_d **:** std\_logic\_vector **(**size**-**1 **downto** 0**);**

**signal** data\_q **:** std\_logic\_vector **(**size**+**1 **downto** 0**);**

**begin**

--loop to generate the shift register flipflops and muxes

shift\_reg**:** **for** i **in** 0 **to** size**-**1 **generate**

one\_dff**:** **entity** work**.**dff

**port** **map** **(**

clk **=>** clk**,**

rst **=>** rst**,**

d **=>** data\_d**(**i**),** --connect unique data\_d line

q **=>** data\_q**(**i**+**1**)** --connect unique data\_q line, leave one spare q at start

**);**

--the control input selects the source for the d inputs of the flipflops

**with** ctrl **select**

data\_d**(**i**)** **<=** data\_q**(**i**+**1**)** **when** "00"**,** --hold: take value from self

data\_in**(**i**)** **when** "11"**,** --load: take value from input bus

data\_q**(**i**)** **when** "10"**,** --shift left: take value from next flipflop right

data\_q**(**i**+**2**)** **when** "01"**,** --shift right: take value from next flipflop left

'U' **when** **others;**

**end** **generate;**

--first signal in data\_q is shift in

data\_q**(**0**)** **<=** shift\_in**;**

--last signal in data\_q is also shift in

data\_q**(**size**+**1**)** **<=** shift\_in**;**

--data out is just data\_q without the signals at the two ends

--(the two ends are the shift in lines)

data\_out **<=** data\_q**(**size **downto** 1**);**

**end** Behavioral**;**

**VHDL test bench for top level shift register**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**ENTITY** bidi\_shift\_reg\_tb **IS**

**END** bidi\_shift\_reg\_tb**;**

**ARCHITECTURE** behavior **OF** bidi\_shift\_reg\_tb **IS**

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

**COMPONENT** bidi\_shift\_reg

**PORT(**

clk **:** **IN** std\_logic**;**

shift\_in **:** **IN** std\_logic**;**

data\_in **:** **IN** std\_logic\_vector**(**15 **downto** 0**);**

data\_out **:** **OUT** std\_logic\_vector**(**15 **downto** 0**);**

ctrl **:** **IN** std\_logic\_vector**(**1 **downto** 0**);**

rst **:** **IN** std\_logic

**);**

**END** **COMPONENT;**

--Inputs

**signal** clk **:** std\_logic **:=** '0'**;**

**signal** shift\_in **:** std\_logic **:=** '0'**;**

**signal** data\_in **:** std\_logic\_vector**(**15 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** ctrl **:** std\_logic\_vector**(**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** rst **:** std\_logic **:=** '0'**;**

--Outputs

**signal** data\_out **:** std\_logic\_vector**(**15 **downto** 0**);**

-- Clock period definitions

**constant** clk\_period **:** time **:=** 10 ns**;**

**BEGIN**

-- Instantiate the Unit Under Test (UUT)

uut**:** bidi\_shift\_reg **PORT** **MAP** **(**

clk **=>** clk**,**

shift\_in **=>** shift\_in**,**

data\_in **=>** data\_in**,**

data\_out **=>** data\_out**,**

ctrl **=>** ctrl**,**

rst **=>** rst

**);**

-- Clock process definitions

clk\_process **:process**

**begin**

clk **<=** '0'**;**

**wait** **for** clk\_period**/**2**;**

clk **<=** '1'**;**

**wait** **for** clk\_period**/**2**;**

**end** **process;**

-- Stimulus process

stim\_proc**:** **process**

**begin**

-- hold reset state for 100 ns.

**wait** **for** 100 ns**;**

--initial input values

data\_in **<=** X"0000"**;**

ctrl **<=** "00"**;**

shift\_in **<=** '0'**;**

--reset

rst **<=** '1'**;**

**wait** **for** clk\_period**;**

rst **<=** '0'**;**

**wait** **for** clk\_period**;**

--start with load mode test

ctrl **<=** "11"**;**

data\_in **<=** X"b13f"**;**

**wait** **for** clk\_period**;**

--Self test that load worked

**assert** data\_out **=** X"b13f"

**report** "Load failed"

**severity** error**;**

--Now hold for some clock cycles

ctrl **<=** "00"**;**

data\_in **<=** X"af34"**;**

**wait** **for** clk\_period**\***4**;**

--self test that loaded value is still present

**assert** data\_out **=** X"b13f"

**report** "Hold failed"

**severity** error**;**

--Reset for left shift test

data\_in **<=** X"0000"**;**

ctrl **<=** "00"**;**

shift\_in **<=** '0'**;**

rst **<=** '1'**;**

**wait** **for** clk\_period**;**

rst **<=** '0'**;**

**wait** **for** clk\_period**;**

--Self check that second reset worked

**assert** data\_out **=** X"0000"

**report** "Reset failed"

**severity** error**;**

--Now test shift left mode, start by shifting one in

ctrl **<=** "10"**;**

shift\_in **<=** '1'**;**

**wait** **for** clk\_period**;**

shift\_in **<=** '0'**;**

--Check that the first bit is shifted in correctly

**assert** data\_out **=** "0000000000000001"

**report** "Shift left failed."

**severity** error**;**

--shift left and check another 16 times

**for** i **in** 1 **to** 16 **loop**

**wait** **for** clk\_period**;**

--expected output pattern is generated using "sll" operator (logical shift left)

**assert** unsigned**(**data\_out**)** **=** **(**unsigned'**(**X"0001"**)** sll i**)**

**report** "Shift left test failed. Expected " **&** integer'**image(to\_integer(**unsigned'**(**X"0001"**)** sll i**))**

**&** " but got " **&** integer'**image(to\_integer(**unsigned**(**data\_out**)))**

**&** "."

**severity** error**;**

**end** **loop;**

--Reset for right shift test

data\_in **<=** X"0000"**;**

ctrl **<=** "00"**;**

shift\_in **<=** '0'**;**

rst **<=** '1'**;**

**wait** **for** clk\_period**;**

rst **<=** '0'**;**

**wait** **for** clk\_period**;**

--Check reset again

**assert** data\_out **=** X"0000"

**report** "Reset failed"

**severity** error**;**

--Now test shift right mode, start by shifting one in

ctrl **<=** "01"**;**

shift\_in **<=** '1'**;**

**wait** **for** clk\_period**;**

shift\_in **<=** '0'**;**

--Check that the first bit is shifted in correctly

**assert** data\_out **=** "1000000000000000"

**report** "Shift right failed."

**severity** error**;**

--shift left and check another 16 times

**for** i **in** 1 **to** 16 **loop**

**wait** **for** clk\_period**;**

--expected output pattern is generated using "srl" operator (logical shift right)

**assert** unsigned**(**data\_out**)** **=** **(**unsigned'**(**X"8000"**)** srl i**)**

**report** "Shift right test failed. Expected " **&** integer'**image(to\_integer(**unsigned'**(**X"0001"**)** sll i**))**

**&** " but got " **&** integer'**image(to\_integer(**unsigned**(**data\_out**)))**

**&** "."

**severity** error**;**

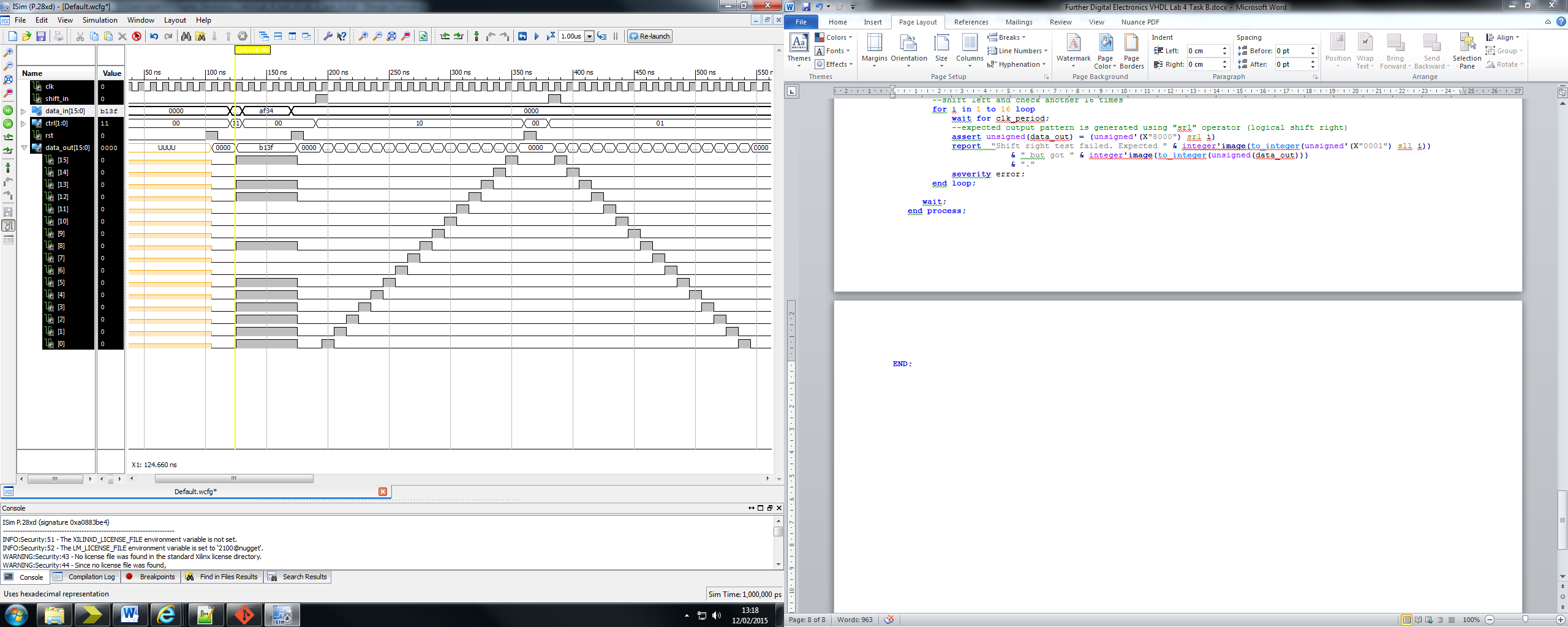
**end** **loop;**

**wait;**

**end** **process;**

**END;**

**Screenshot of testbench**



**HDL Synthesis report**

=========================================================================

\* HDL Synthesis \*

=========================================================================

Synthesizing Unit <bidi\_shift\_reg>.

Related source file is "C:\Users\\*\*\*\*\*\*\*\Digital\_Electronics\_Labs\Lab\_4\_task\_b\bidi\_shift\_reg.vhd".

size = 16

Found 1-bit 4-to-1 multiplexer for signal <data\_d<0>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<1>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<2>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<3>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<4>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<5>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<6>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<7>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<8>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<9>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<10>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<11>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<12>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<13>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<14>> created at line 35.

Found 1-bit 4-to-1 multiplexer for signal <data\_d<15>> created at line 35.

Summary:

inferred 16 Multiplexer(s).

Unit <bidi\_shift\_reg> synthesized.

Synthesizing Unit <dff>.

Related source file is "C:\Users\\*\*\*\*\*\*\*\Digital\_Electronics\_Labs\Lab\_4\_task\_b\dff.vhd".

Found 1-bit register for signal <data>.

Summary:

inferred 1 D-type flip-flop(s).

Unit <dff> synthesized.

=========================================================================

HDL Synthesis Report

Macro Statistics

# Registers : 16

1-bit register : 16

# Multiplexers : 16

1-bit 4-to-1 multiplexer : 16

=========================================================================