Logo

Description automatically generated**EEDG/CE 6370**

34

**Design and Analysis or Reconfigurable Systems**

**Homework 1**

**Design of Combinational Logic Designs using Schematic entry and Verilog/VHDL using Intel Quartus Prime**

**Student Name: Khoa Diep**

**Date: 8/29/2023**

**Part I: Schematic Entry (1-bit adder)**

a.) Follow the instructions in the homework sheet and design a 1-bit full adder using schematic entry show here the generated schematic (screenshot of the circuit)

(4 marks)

A diagram of a circuit

Description automatically generated

b.) Simulate the 1-bit full adder showing that it works. Include here the test vectors used and the simulation results (waveform). Clearly indicate why it is working.

(4 marks)

A black and green grid

Description automatically generated

Looking at the waveforms, we can see that low means 0 and high means 1. Using binary addition, we do A + B + Cin and the result is Cout is the carry bit (most significant bit) and S is the least significant bit. Looking at the wave form, the previous statement is true for all of the cases (0 + 0 + 0 = 00, 0 + 0 + 1= 01, …. , 1 + 1 + 1 = 11).

c.) What test vectors would you use in the case of having to test a 4-bit adder and an 8-bit adder? Do you foresee any problems verifying larger adders?

(2 marks)

Since 1 bit adder has 3 inputs (A, B, Cin) and 2 outputs (S, Cout), if we scale this up to a 4-bit adder, we will need 9 inputs (A and B for each bit plus the Cin) and 5 outputs (outputs of each bit and then the Cout). Similarly, we would need 17 inputs and 9 outputs for an 8-bit adder. Thus, the test vectors for the 4-bit adder (and the 8-bit adder) would be simply the inputs and outputs mentioned previously. Even if vectors of length 4 (or 8) are used for A, B and S, verifying these larger bit adders via a wave form, we would have to look at ALL cases of ALL the vectors on the wave form and would be a tedious and lengthy process if needed to verify manually.

d.) Report the number of ALMs, ALUTs and critical path of the design from the synthesis report. Compute the maximum frequency.

(2 marks)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ALMS/ALUTs | FFs | Ios | DSPs | BRAM |
| 1-bit adder schematic | 2/3 | 0 | 5 | 0 | 0 |

Fmax: Quartus did not report the timing. Thus, Fmax and critical path is not recorded.

**Part II : RTL Entry (Verilog/VHDL) (1-bit adder)**

1. Re-do the full adder example specifying the 1-bit adder directly in Verilog or VHDL. Enter here the clean source code with comments.

(4 marks)

-- A library clause declares a name as a library. It

-- does not create the library; it simply forward declares

-- it.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

ENTITY fulladder\_rtl IS

PORT (

A: in std\_logic;

B: in std\_logic;

Cin: in std\_logic;

S: out std\_logic;

Cout: out std\_logic);

END fulladder\_rtl;

ARCHITECTURE behav OF fulladder\_rtl IS

BEGIN

-- S (Based on the schematic)

S <= (A XOR B) XOR Cin ;

--Cout (Based on the schematic)

Cout <= (A AND B) OR ((A XOR B) AND Cin) ;

END behav;

Screenshot of code

A computer code with text

Description automatically generated

1. Show the simulation results of the working adder.

(4 marks)

A green line on a black background

Description automatically generated

1. Report the number of ALMs, ALUTs and critical path of the design from the synthesis report. Compute the maximum frequency.

(2 marks)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ALMs/ALUTs | FFs | Ios | DSPs | BRAM |
| 1-bit adder RTL | 2/3 | 0 | 5 | 0 | 0 |

Fmax: Quartus did not report the timing. Thus, Fmax and critical path is not recorded. (Same as earlier)

1. Compare the ALUT usage and critical path delay from the schematic adder and RTL adder do the results match? Yes/no? Explain why?

(2 marks)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ALMs/ALUTs | FFs | Ios | DSPs | BRAM | Delay |
| Schematic | 2/3 | 0 | 5 | 0 | 0 | N/A |
| RTL | 2/3 | 0 | 5 | 0 | 0 | N/A |

The results of the schematic adder and RTL adder matched. This is because the RTL adder is directly using the same logic and gates as the schematic adder. Thus, when synthesizing, the result (the netlist) will be the exact same, since it is the exact same circuit when programming it to the FPGA board.

**Part III : RTL Entry (Verilog/VHDL) (4-bit adder)**

1. Create a 4-bit adder using structural and behavioral RTL (Verilog or VHDL). Annotate their resource usage and critical path in the table below:”

(4 marks)

|  |  |  |
| --- | --- | --- |
|  | ALMs/ALUTs | Critical path |
| Structural | 4/7 | N/A |
| Behavioral | 3/5 | N/A |

1. Discuss if the results obtained are what you expected and why yes or no.

(2 marks)

Before, I expected Behavioral would use more resources than structural, but, the opposite was in fact true. My expectations were that usually the more abstract the design of a circuit (or code in general) would lead to less efficient resource allocation. However, the FPGA board accounts for lots of addition, thus can add more efficiently than our designed full adder.

**Part IV: Homework reflection**

(4 marks)

What do you think were the objectives of this homework?

The first objective of this homework was to demonstrate the differences between schematic design and HDLs. Another objective is to review/learn HDLs such as Verilog and VHDL. The main objective was to learn how to use tools, such as Quartus and Questa RTL Simulator. From Quartus, we had to learn how to synthesize and read its report, as well as learn how to write a testbench for the designs. From those testbenches, we learned how to simulate on Questa and read/understand the waveforms generated. Finally, the differences between structural and behavioral HDLs were observed.

What conclusions have you extracted from the results obtained?

Schematics designs are much more tedious and complicated than HDLs such as Verilog and VHDL. However, in terms of resource usage, RTL design on HDLs and schematic design are the same (or very similar). Without flip-flops and a clock, we cannot calculate Max Frequency or measure the critical path, since there is no timing. The FPGA accounts for addition and has dedicated architecture built in to accommodate for addition. This means that adding will be made more efficient.

**Attachments**

Include the Verilog/VHDL code here. Make sure it is well formatted and include comments. Marks might get deducted if not.

Fourbit adder VHDL code (structural)

-- A library clause declares a name as a library. It

-- does not create the library; it simply forward declares

-- it.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

ENTITY fourbitadder\_str IS

PORT (

A: in std\_logic\_vector (3 downto 0);

B: in std\_logic\_vector (3 downto 0);

Cin: in std\_logic;

S: out std\_logic\_vector (3 downto 0);

Cout: out std\_logic);

END fourbitadder\_str;

ARCHITECTURE struct OF fourbitadder\_str IS

component fulladder\_rtl IS

port(

A: in std\_logic;

B: in std\_logic;

Cin: in std\_logic;

S: out std\_logic;

Cout: out std\_logic);

END component;

signal C0, C1, C2, C3: std\_logic;

BEGIN

-- Each fulladder’s carryout is the next fulladder’s carryin

F0:fulladder\_rtl port map (A(0), B(0), Cin, S(0), C0);

F1:fulladder\_rtl port map (A(1), B(1), C0, S(1), C1);

F2:fulladder\_rtl port map (A(2), B(2), C1, S(2), C2);

F3:fulladder\_rtl port map (A(3), B(3), C2, S(3), C3);

Cout <= C3;

END struct;

Testbench for fourbit adder (structural)

-- Copyright (C) 2023 Intel Corporation. All rights reserved.

-- Your use of Intel Corporation's design tools, logic functions

-- and other software and tools, and any partner logic

-- functions, and any output files from any of the foregoing

-- (including device programming or simulation files), and any

-- associated documentation or information are expressly subject

-- to the terms and conditions of the Intel Program License

-- Subscription Agreement, the Intel Quartus Prime License Agreement,

-- the Intel FPGA IP License Agreement, or other applicable license

-- agreement, including, without limitation, that your use is for

-- the sole purpose of programming logic devices manufactured by

-- Intel and sold by Intel or its authorized distributors. Please

-- refer to the applicable agreement for further details, at

-- https://fpgasoftware.intel.com/eula.

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- This file contains a Vhdl test bench template that is freely editable to

-- suit user's needs .Comments are provided in each section to help the user

-- fill out necessary details.

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Generated on "09/04/2023 17:33:27"

-- Vhdl Test Bench template for design : fulladder\_rtl

--

-- Simulation tool : Questa Intel FPGA (VHDL)

--

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

ENTITY fulladder\_rtl\_vhd\_tst IS

END fulladder\_rtl\_vhd\_tst;

ARCHITECTURE fulladder\_rtl\_arch OF fulladder\_rtl\_vhd\_tst IS

-- constants

-- signals

SIGNAL A : STD\_LOGIC;

SIGNAL B : STD\_LOGIC;

SIGNAL Cin : STD\_LOGIC;

SIGNAL Cout : STD\_LOGIC;

SIGNAL S : STD\_LOGIC;

COMPONENT fulladder\_rtl

PORT (

A : IN STD\_LOGIC;

B : IN STD\_LOGIC;

Cin : IN STD\_LOGIC;

Cout : OUT STD\_LOGIC;

S : OUT STD\_LOGIC

);

END COMPONENT;

BEGIN

i1 : fulladder\_rtl

PORT MAP (

-- list connections between master ports and signals

A => A,

B => B,

Cin => Cin,

Cout => Cout,

S => S

);

init : PROCESS

-- variable declarations

BEGIN

-- code that executes only once

WAIT;

END PROCESS init;

always : PROCESS

-- optional sensitivity list

-- ( )

-- variable declarations

BEGIN

wait for 1 ns;

A <= '0';

B <= '0';

Cin <= '0';

wait for 1 ns;

A <= '0';

B <= '0';

Cin <= '1';

wait for 1 ns;

A <= '0';

B <= '1';

Cin <= '0';

wait for 1 ns;

A <= '0';

B <= '1';

Cin <= '1';

wait for 1 ns;

A <= '1';

B <= '0';

Cin <= '0';

wait for 1 ns;

A <= '1';

B <= '0';

Cin <= '1';

wait for 1 ns;

A <= '1';

B <= '1';

Cin <= '0';

wait for 1 ns;

A <= '1';

B <= '1';

Cin <= '1';

WAIT;

END PROCESS always;

END fulladder\_rtl\_arch;

Waveform for 4bit adder (structural)

A screenshot of a computer

Description automatically generated

Fourbit adder VHDL code (behavioral)

-- A library clause declares a name as a library. It

-- does not create the library; it simply forward declares

-- it.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

--use ieee.std\_logic\_arith.all;

ENTITY fourbitadder\_beh IS

PORT (

-- Difference from full adder is that A, B, S are vectors of 4 bits rather than 1 bit.

A: in std\_logic\_vector(3 downto 0);

B: in std\_logic\_vector(3 downto 0);

Cin: in std\_logic;

S: out std\_logic\_vector(3 downto 0);

Cout: out std\_logic);

END fourbitadder\_beh;

ARCHITECTURE behavioral OF fourbitadder\_beh IS

signal temp: std\_logic\_vector(4 downto 0);

BEGIN

-- Concate 0 to the left of A and B to prevent and record overflow

temp <= ('0' & A) + ('0' & B) + Cin;

-- S is the 3 significant bits (3 bits from right) and Cout is the most significant

S <= temp(3 downto 0);

Cout <= temp(4);

END behavioral;

Testbench for 4bit adder (behavioral)

-- Copyright (C) 2023 Intel Corporation. All rights reserved.

-- Your use of Intel Corporation's design tools, logic functions

-- and other software and tools, and any partner logic

-- functions, and any output files from any of the foregoing

-- (including device programming or simulation files), and any

-- associated documentation or information are expressly subject

-- to the terms and conditions of the Intel Program License

-- Subscription Agreement, the Intel Quartus Prime License Agreement,

-- the Intel FPGA IP License Agreement, or other applicable license

-- agreement, including, without limitation, that your use is for

-- the sole purpose of programming logic devices manufactured by

-- Intel and sold by Intel or its authorized distributors. Please

-- refer to the applicable agreement for further details, at

-- https://fpgasoftware.intel.com/eula.

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- This file contains a Vhdl test bench template that is freely editable to

-- suit user's needs .Comments are provided in each section to help the user

-- fill out necessary details.

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Generated on "09/05/2023 18:21:26"

-- Vhdl Test Bench template for design : fourbitadder\_beh

--

-- Simulation tool : Questa Intel FPGA (VHDL)

--

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

ENTITY fourbitadder\_beh\_vhd\_tst IS

END fourbitadder\_beh\_vhd\_tst;

ARCHITECTURE fourbitadder\_beh\_arch OF fourbitadder\_beh\_vhd\_tst IS

-- constants

-- signals

SIGNAL A : STD\_LOGIC\_VECTOR(3 DOWNTO 0);

SIGNAL B : STD\_LOGIC\_VECTOR(3 DOWNTO 0);

SIGNAL Cin : STD\_LOGIC;

SIGNAL Cout : STD\_LOGIC;

SIGNAL S : STD\_LOGIC\_VECTOR(3 DOWNTO 0);

COMPONENT fourbitadder\_beh

PORT (

A : IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

B : IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

Cin : IN STD\_LOGIC;

Cout : BUFFER STD\_LOGIC;

S : BUFFER STD\_LOGIC\_VECTOR(3 DOWNTO 0)

);

END COMPONENT;

BEGIN

i1 : fourbitadder\_beh

PORT MAP (

-- list connections between master ports and signals

A => A,

B => B,

Cin => Cin,

Cout => Cout,

S => S

);

init : PROCESS

-- variable declarations

BEGIN

-- code that executes only once

WAIT;

END PROCESS init;

always : PROCESS

-- optional sensitivity list

-- ( )

-- variable declarations

BEGIN

-- code executes for every event on sensitivity list

wait for 1 ns;

A <= "0000";

B <= "0000";

Cin <= '0';

wait for 1 ns;

A <= "0000";

B <= "0000";

Cin <= '1';

wait for 1 ns;

A <= "0010";

B <= "0010";

Cin <= '0';

wait for 1 ns;

A <= "0010";

B <= "0010";

Cin <= '1';

wait for 1 ns;

A <= "0010";

B <= "0100";

Cin <= '0';

wait for 1 ns;

A <= "0010";

B <= "0100";

Cin <= '1';

wait for 1 ns;

A <= "1111";

B <= "1111";

Cin <= '0';

wait for 1 ns;

A <= "1111";

B <= "1111";

Cin <= '1';

WAIT;

END PROCESS always;

END fourbitadder\_beh\_arch;

Waveform for 4bit adder (behavioral)

A black screen with colorful lines

Description automatically generated