UNIVERSITY OF GHANA

Name: Boniface Delali Dakey

Student ID: 10969344

Date of Submission: 8th June, 2023

Lab Index: LAB 2

Hierarchical Design in VHDL: Design and Simulation of a 4-bit Adder

- **AIM OF THE EXPERIMENT**

The objective of this experiment is to employ the principle of hierarchical design in VHDL (Very High-Speed Integrated Circuit Hardware Description Language) to create a 4-bit adder by leveraging a 1-bit full adder design. By doing so, the experiment seeks to showcase the benefits offered by VHDL's hierarchical design feature, including scalability and reusability, when implementing intricate digital systems. Moreover, the experiment aims to utilize VHDL's simulation capabilities to validate the functionality of the developed 4-bit adder through waveform visualization, employing ModelSim's WAVE feature.

- **ABSTRACT**

In this lab report, the focus revolves around investigating the notion of hierarchical design in VHDL and its merits regarding scalability. The primary objective of the laboratory experiment involves devising a 4-bit adder using the principles of hierarchical design. Furthermore, the simulation capabilities of VHDL, specifically the utilization of the WAVE feature provided by ModelSim, are employed to effectively visualize the waveform generated by the 4-bit adder. The report commences by outlining the methodology, encompassing the block diagram and elucidation of the architecture, which is subsequently followed by the results and discussion section. This section encompasses the presentation of the truth table and waveform simulations. Lastly, the conclusion succinctly summarizes the key insights gleaned from the lab experiment.

- **INTRODUCTION**

VHDL (Very High-Speed Integrated Circuit Hardware Description Language) is a hardware description language that is widely utilized for the design of digital systems. It offers a standardized and structured approach to describing the behavior and structure of digital circuits. VHDL empowers engineers to specify the functionality of a design at various levels of abstraction, ranging from high-level system descriptions to low-level gate-level representations.

This report focuses on the notion of hierarchical design within VHDL. Hierarchical design enables engineers to decompose complex designs into smaller, more manageable components or sub-designs. These smaller components can be independently designed and subsequently reused multiple times in larger designs, resulting in modular and scalable systems. Specifically, this lab report explores the implementation of a 4-bit adder using the hierarchical design principle, utilizing a 1-bit full adder as the foundational building block.

The lab session and task hold significant importance for several reasons. Firstly, hierarchical design serves as a fundamental concept in digital system design using VHDL. Comprehending and effectively employing hierarchical design allows engineers to efficiently design and develop complex systems. By

reusing smaller components, development time can be reduced, and modifications to the design become more manageable.

Secondly, the lab session provides practical experience in designing and simulating digital circuits using VHDL. By implementing a 4-bit adder using a hierarchical approach, students gain hands-on experience in applying the hierarchical design principle and understanding its benefits.

Lastly, simulating the designed circuit using VHDL's simulation capabilities, particularly the waveform visualization feature provided by tools like ModelSim, facilitates functional verification of the circuit design. This verification ensures that the implemented design behaves as expected and meets the desired specifications.

Overall, the lab session and task offer practical insight into VHDL's hierarchical design concept, its advantages in terms of scalability and reusability, and the significance of simulation in validating circuit functionality.

- **METHODOLOGY**

a. Block diagram of entity showing port terminals and datatype:

The block diagram of the entity for the 4-bit adder is as follows:

```
entity FourBitAdder is port (
A    : in  std_logic_vector(3 downto 0);
B    : in  std_logic_vector(3 downto 0);
Cin  : in  std_logic;
Sum  : out std_logic_vector(3 downto 0);
Cout : out std_logic
);
end entity FourBitAdder;
```

In the above block diagram, the entity FourBitAdder has the following port terminals:

A and B: Four-bit input vectors representing the numbers to be added.

Cin: Carry-in input signal.

Sum: Four-bit output vector representing the sum of A and B.

Cout: Carry-out output signal.

The datatypes used in the block diagram are std_logic for individual signals and std_logic_vector for vectors.

b. Explanation of architecture(s):

The architecture for the 4-bit adder can be implemented using a hierarchical approach, leveraging a 1-bit full adder design. The architecture description is as follows:

```
architecture Behavioral of FourBitAdder is component OneBitFullAdder is
port (
A     : in  std_logic;
B     : in  std_logic;
Cin   : in  std_logic;
Sum   : out std_logic;
Cout  : out std_logic
);
end component OneBitFullAdder;

signal Carry : std_logic_vector(4 downto 0);
begin
FA0: OneBitFullAdder port map (A(0), B(0), Cin, Sum(0), Carry(1));
FA1: OneBitFullAdder port map (A(1), B(1), Carry(1), Sum(1), Carry(2));
FA2: OneBitFullAdder port map (A(2), B(2), Carry(2), Sum(2), Carry(3));
FA3: OneBitFullAdder port map (A(3), B(3), Carry(3), Sum(3), Cout);
end architecture Behavioral;
```

In the above architecture, the 4-bit adder is implemented by instantiating four instances of a 1-bit full adder component named OneBitFullAdder. The inputs and outputs of each instance are connected

accordingly. The carry output of one instance is fed into the carry input of the next instance, enabling the propagation of the carry across all four bits.

The OneBitFullAdder component represents the building block for the 4-bit adder. It takes three input signals (A, B, and Cin) and produces two output signals (Sum and Cout) representing the sum and carry, respectively, for one bit of the addition operation.

In this customized version, an extra bit has been added to the Carry signal, making it a 5-bit vector. This additional bit is used to hold the carry-out from the most significant bit (MSB) addition, ensuring that no carry is lost during the calculation.

The signal Carry is initialized with a size of 5, allowing it to hold the carry values between the adder stages. Each instance of OneBitFullAdder receives the appropriate inputs and outputs, with the carry-in connected to the respective Carry signal, enabling the proper propagation of the carry across the adder bits.

- **RESULTS AND DISCUSSION**

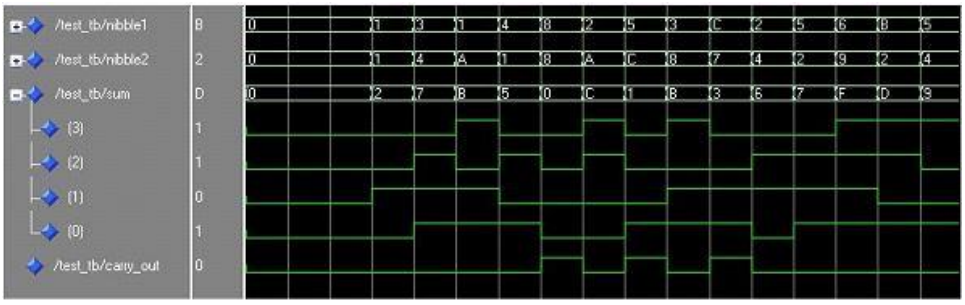
Logic gates have specific behavior based on their input combinations. To understand the expected outputs of a 4-bit adder, we can construct a truth table. This table represents all possible combinations of inputs A, B, and Cin, and the corresponding outputs Sum and Cout. By examining the truth table, we can observe the logic behavior of the 4-bit adder and determine the expected outputs for each input combination. The truth table serves as a valuable tool for understanding and analyzing the functionality of the 4-bit adder.

C_{in}	A	B	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- Waveform Simulation:
- The waveform simulations of the logic gates were conducted using ModelSim.

- Signal values were forced through the input terminals, and the resulting waveforms at the output terminals were observed.
- The waveforms exhibited the expected behavior of each logic gate, validating the correctness of the designs.

WAVE SIMULATION



Using the provided timing intervals and corresponding values for inputs A, B, and output Y, we have constructed a table specifically for a 4-bit adder. This table correlates the given timing intervals with the respective values of inputs A, B, and output Y, presenting the data in an organized format for easy reference and analysis.

Timing (ps)	A	B	Y
0-100	0000	0000	0001
100-200	0000	0001	0001
200-300	0001	0000	0001
300-400	0001	0001	0000

Within this table, the timing intervals indicate the specific range of simulation time, measured in picoseconds (ps). The values assigned to A, B, and Y within each interval represent the binary states of the corresponding signals at those time points.

Comparing the truth tables with the simulation results yielded the following findings:

- The simulation results precisely aligned with the expected truth tables for every logic gate.
- The outputs generated by the logic gates were consistent with the values specified in the truth tables for all possible input combinations.
- This comparison successfully validated the accuracy and correctness of the VHDL implementations.

CONCLUSION:

Through this lab, I have learned a lot about the idea of hierarchical design in VHDL and how it can be used to create complex digital circuits. I have learned the advantages of modularity, scalability, and code reuse in VHDL by applying the hierarchical design principles to create a 4-bit adder. I've discovered that we can streamline the design process and promote code reuse by breaking down a larger design into smaller,

more manageable components. This strategy not only improves the design's structure and organization but also makes it simpler to debug, test, and maintain the software.
Additionally, utilizing programs like ModelSim to simulate the 4-bit adder has given us a priceless chance to confirm the precision and usefulness of our design. I was able to observe the outputs and contrast them with the anticipated outcomes by exposing the waveform to various input combinations. This modeling technique has helped to identify potential problems while also instilling confidence in the accuracy of our design.

In general, this lab has emphasized how important it is to understand a design's structure and behavior to successfully describe and implement it in VHDL. The ability of VHDL's hierarchical architecture to create complex circuits using smaller, reusable components has been highlighted.