Department of Computer and System Engineering

Tallinn University of Technology

IAY0600 Digital System Design Lab

Laboratory Report

Experiment #2: Adder

Written by

Olatayo Olukotun Joshua

172626

Introduction

Task

1. Implement a 2-bit adder using 1-bit full adder and 1-bit half adder as components (Figure1). Describe both components in VHDL in two implementations where components are instantiated in:

a. Block Diagram top-level file.

b. VHDL top-level file

2. Simulate and implement each project on FPGA development board.

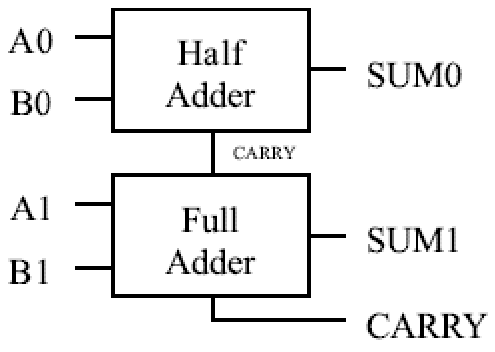


Figure 1: 2-bit Adder

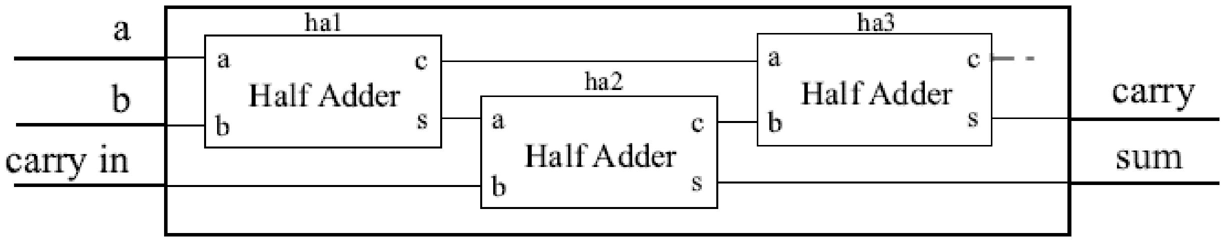


Figure 2: Full Adder Block Diagram

Background

An **adder** is a digital circuit that performs addition of numbers. Adders are combinations of logic gates that combine binary values to obtain a sum. They are classified according to their ability to accept and combine the digits. In this section we will discuss half adders, and full adders.

Half Adder

A half adder is designed to combine two binary digits and produce a carry. Figure 2 shows two ways of constructing a half adder. An AND gate is added in parallel to the quarter adder to generate the carry. The SUM column of the Truth Table represents the output of the quarter adder, and the CARRY column represents the output of the AND gate.

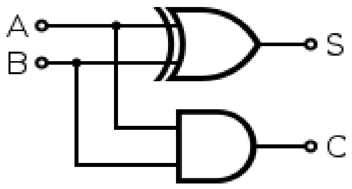
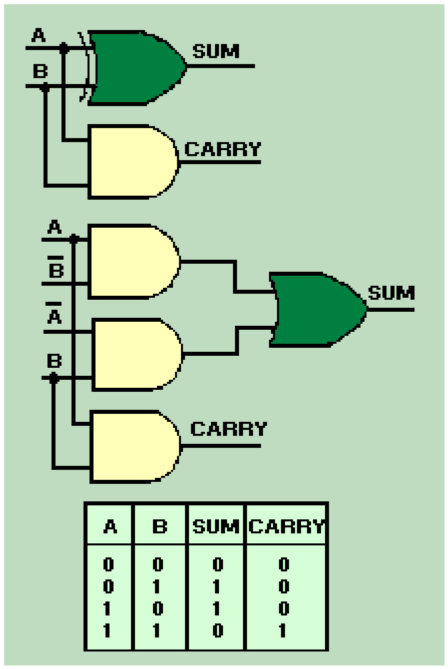


Figure 3.1: Half Adder logic diagram

We see below that the output of the quarter adder is HIGH when either input, but not both, is HIGH. It is only when both inputs are HIGH that the AND gate is activated and a carry is produced. The largest sum that can be obtained from a half adder is 10_2 (1_2 plus 1_2). The Boolean expression for the outputs (Sum and Carry) are: $Carry$ = A.B $\mathrm{Sum} = \mathrm{A}\ XO\mathrm{R\ B}$



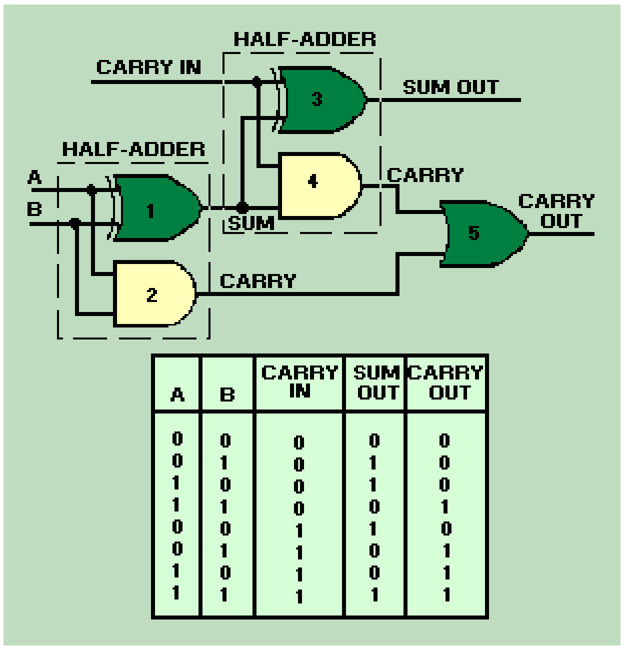
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 3.2: Half Adder and Truth table

Full Adder

The full adder becomes necessary when a carry input must be added to the two binary digits to obtain the correct sum. A half adder has no input for carries from previous circuits.

One method of constructing a full adder is to use two half adders and an OR gate as shown in figure 3. The inputs A and B are applied to gates 1 and 2. These make up one half adder. The sum output of this half adder and the carry-from a previous circuit become the inputs to the second half adder. The carry from each half adder is applied to gate 5 to produce the carry-out for the circuit.



A	B	CARRY IN	SUM OUT	CARRY OUT
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Figure 3.3: Full Adder and Truth table

	00	**01**	**11**	**10**
0			1	
1		1	1	1

Carryout

$CarryOut = A.B + B.Cin + A.\mathrm{Cin}$

	00	**01**	**11**	**10**
0		1		1
1	1		1	

Sum

$$Sum = \bar{A}.\bar{B}.\,Cin + A.\bar{B}.\,.\overline{Cin} + A.\,B.\,Cin + \bar{A}.\,B.\overline{Cin}$$

Workflow

Designing 2-bit adder using block diagram top level file

a. Create a new vivado project file.

b. Create source file for entity “half_adder” and describe its architecture using “AND” and “XOR”.

c. Select “create block diagram” in the “IP Integrator” category and then right click on the workspace and add four instances of the “half_adder” module.

d. Connect three half adders together as shown in figure 2 to form a 1-bit full adder and the connect the full adder to the last adder as shown in figure 1 to form a 2-bit full adder.

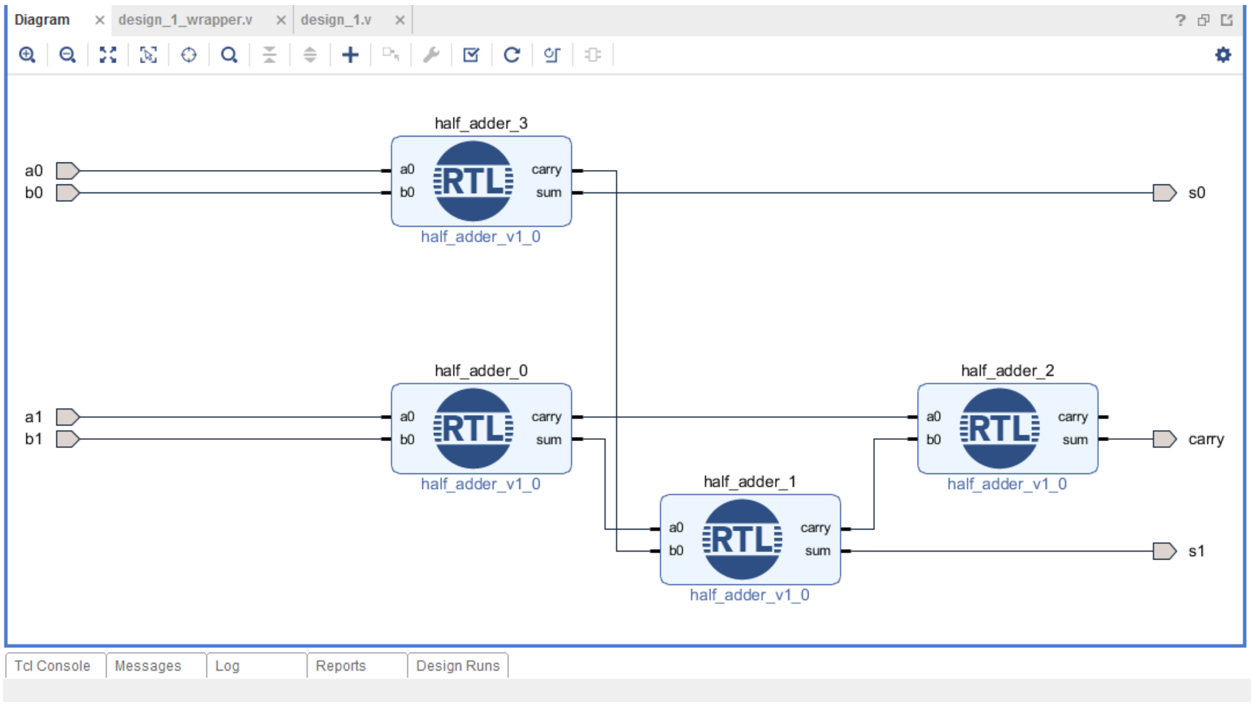


Figure 4.1: Block diagram of 2-bit adder showing the four-connected half adders

e. The RTL schematic which is the same as the Implementation schematic is shown below;

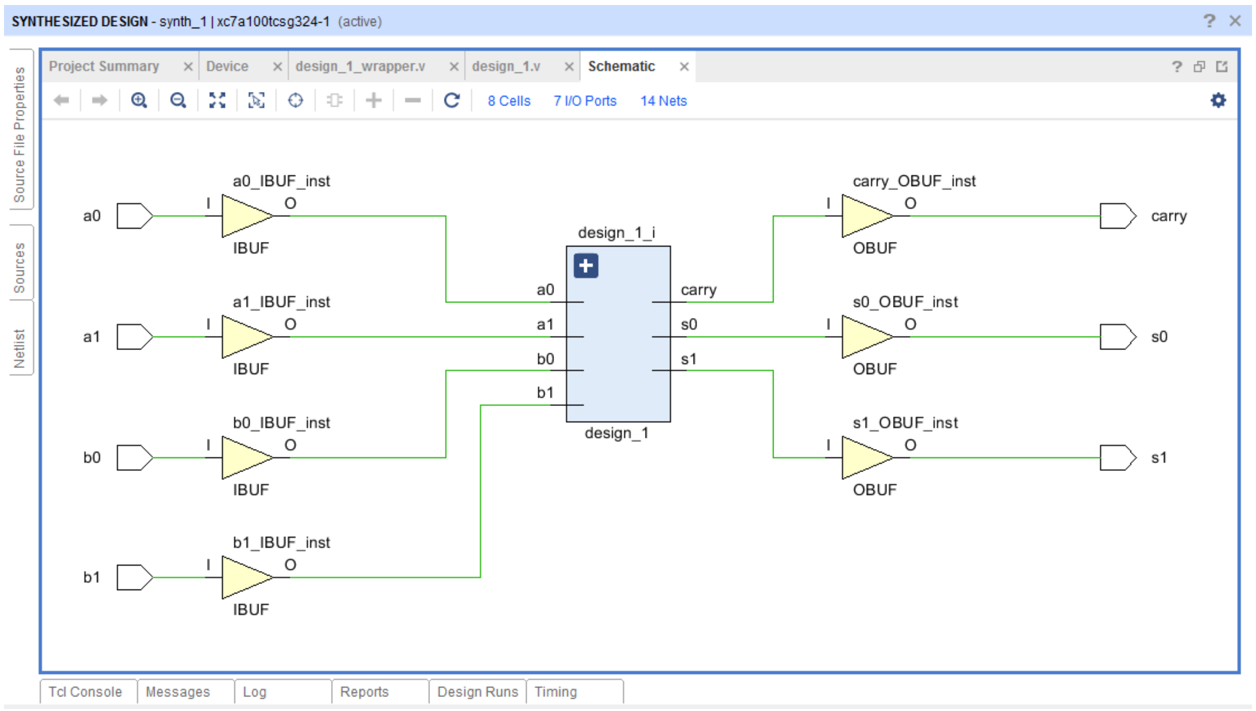


Figure 4.2: Implementation design of a 2-bit adder using a block diagram top level file

f. Functional simulation was performed in order to verify correctness of the received VHDL descriptions, test bench was used for applying stimulus to the design entity during simulation. A new source file was added to the project and VHDL test bench file was created. The truth table for a 2-bit adder and the result of the simulation are shown below.

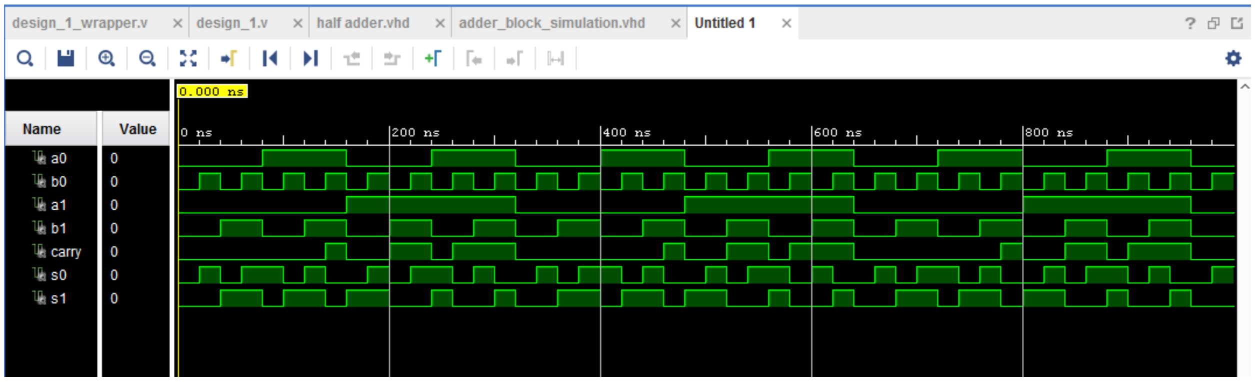


Figure 4.3: Simulation of a 2-bit Adder

Truth table of a 2-bit Adder

a0	a1	b0	b1	carry	s0	s1
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	0	0
0	1	1	0	0	1	1
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	0	0

g. Inputs/outputs ports to device pin mapping were specified in the I/O ports tab of the Result Window area in the RTL Analysis flow section layout. Where inputs a0, a1, b0 and b1 were assigned to M17, M18, P17 and P18 switches of FPGA device package pins respectively. The output carry, s0 and s1 were assigned H17, K15 and J13 LED of the FPGA board respectively. Then I set I/O Std to “LVCMOS33” in every case.

h. A bit file was created which stores all configuration data that defines the internal logic and interconnections in the target device. The bit file downloaded into the FPGA device's memory cells. The FPGA board was connected to the PC and then powered ON for testing.

Designing a 2-bit adder using the VHDL top level file

a. Create a new vivado project file.
b. Create a source file for entity "half_adder" and describe its architecture using "AND" and "XOR".
c. Create a source file for entity "full_adder" describe its architecture.

Listing 4.1: The architecture part of the VHDL description of a 1-bit full adder

```
architecture Behavioral of full_adder is

begin

carry_out <= (a1 AND b1) OR (a1 AND carry_in) OR (b1 AND carry_in);

sum1 <= (NOT a1 AND NOT b1 AND carry_in) OR (a1 AND NOT b1 AND NOT carry_in) OR (NOT a1 AND
b1 AND NOT carry_in) OR (a1 AND b1 AND carry_in);

end Behavioral;
```

d. Create a source file for entity "two_bit_adder", add components "half adder" and "full_adder" and map to the "two_bit _adder" entity.
e. Functional simulation was performed in order to verify correctness of the received VHDL descriptions, test bench was used for applying stimulus to the design entity during simulation. A new source file was added to the project and VHDL test bench file was created. The truth table for a 2-bit adder and the result of the simulation are shown below. The simulation is same as figure 4.3.
f. The RTL schematic and implementation schematic were generated and shown in figure 4.4 and figure 4.5.
g. Constraint source file was added to the design using package pins H17, K15 and J13 as output ports for s0, s1 and carry respectively, while package pins M17, M18, P17 and P18 were used for inputs a0, a1, b0 and b1 respectively.
h. A bit file was created which stores all configuration data that defines the internal logic and interconnections in the target device. The bit file downloaded into the FPGA device's memory cells. The FPGA board was connected to the PC and then powered ON for testing.

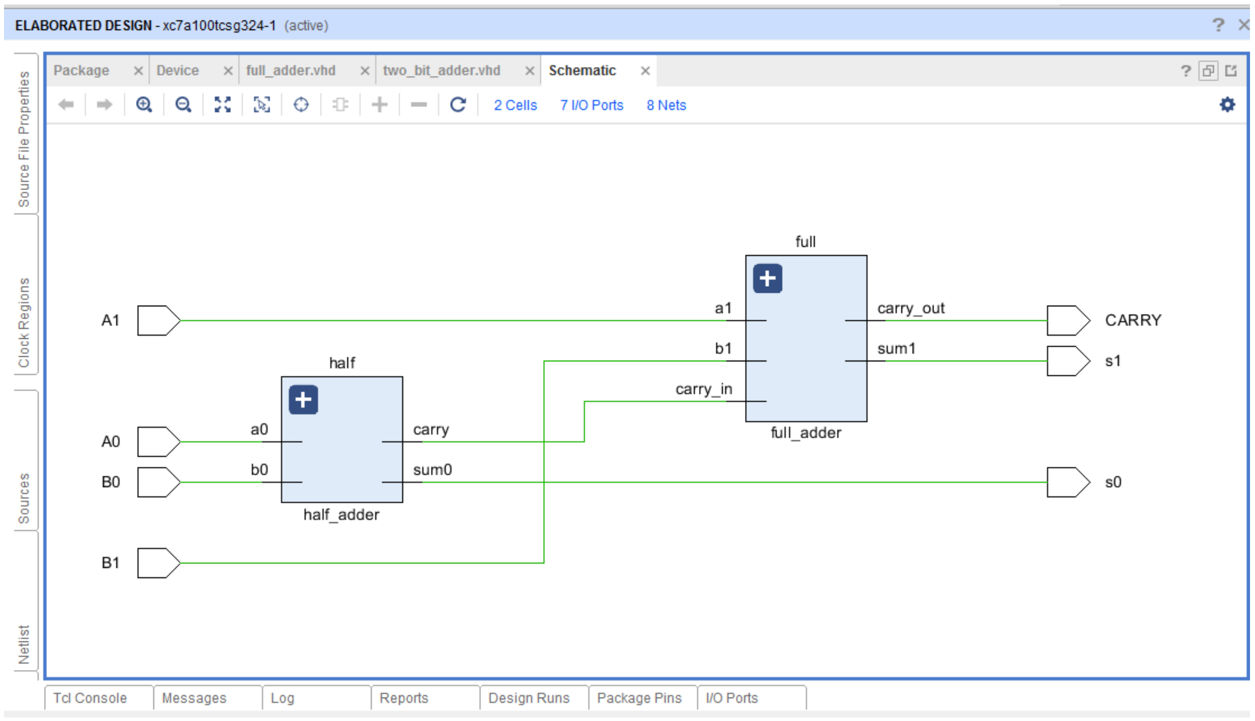


Figure 4.4: RTL Schematic of a 2-bit adder using VHDL top level file

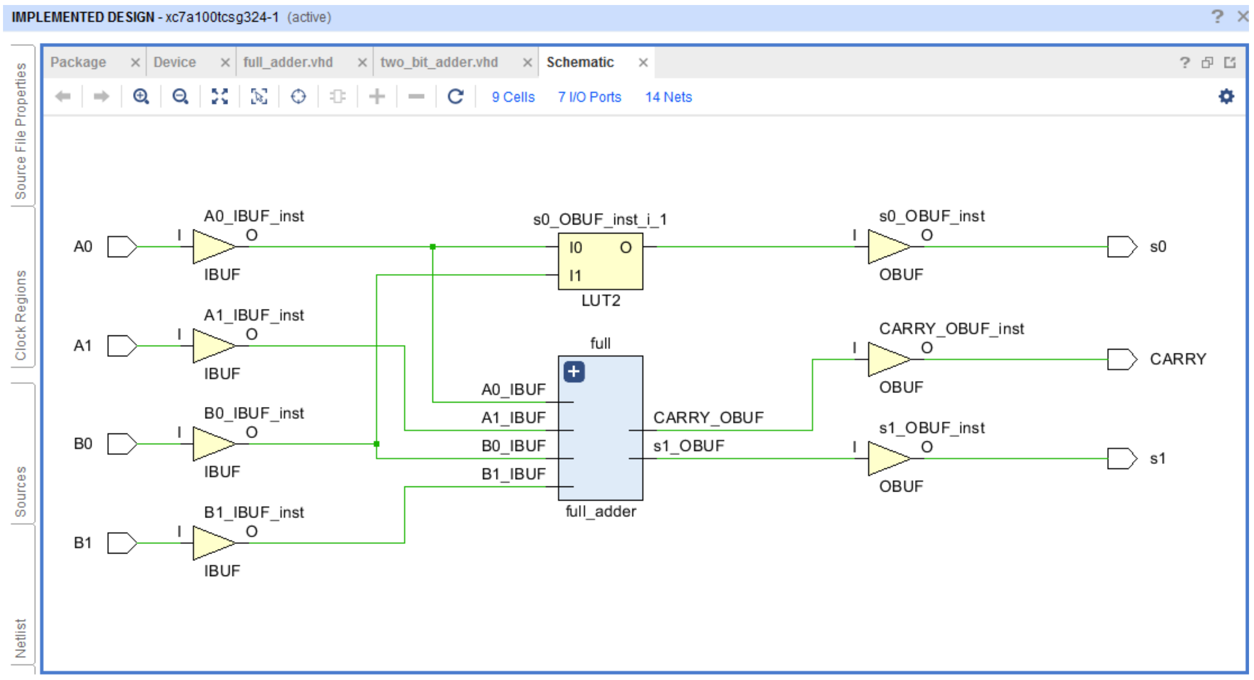


Figure 4.5: Implementation design of a 2-bit adder using VHDL top level file

Conclusion

A system can be represented as a collection of components and their interconnections using structural design approach. In this way, the hardware is described as schematic or netlist, where internal structure of the interconnected components is hidden.

The 2-bit comparator were design using block diagram and VHDL top level file. The components can either be regular structures (like logic gates, adder, comparator, registers) or previously defined sources which are used within another higher-level design to create hierarchy and a practical understanding of hierarchy was achieved.