Experiment 0: Four Bit Adder

Prasann Viswanathan Roll Number 190070047
EE-214, WEL, IIT Bombay
January 27, 2021

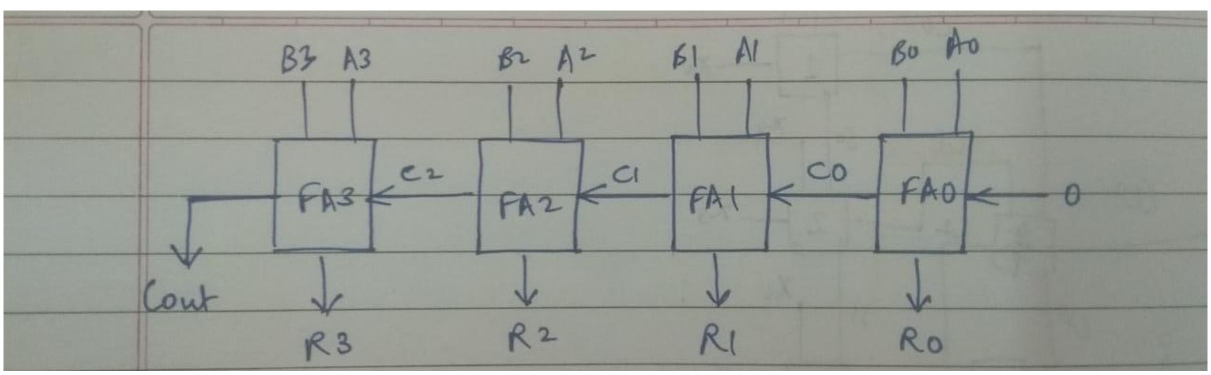
Overview of the experiment:

The purpose of this experiment was to design a logical unit in quartus which enables addition of two 4-bit numbers and gives the correct output, with a carry value. I implemented this using 4 full adder units, which themselves used AND, NOT, OR and XOR gates.

I started by recalling what I had learnt from Digital Systems course about ripple carry adders to perform addition of n-bit numbers and planned to implement the same in Quartus. If the first binary number was represented by A3 A2 A1 A0 and similarly, the second by B3 B2 B1 B0, A0 and B0 would be added using a Full Adder with ‘0’ Carry input and thus ‘ripple’ this effect to the subsequent full adders in sequence, with Cout from previous adder acting as Cin for the next. As a result of this, I needed only one VHDL file to code with logic, named ‘FourBitAdder’. The remaining ‘Gates’ and ‘FullAdder’ remained the same and ‘Testbench’ required a change in only number of inputs and outputs. Then I finished the DUT file to suit my needs. Finally, I made a python script to generate the testcases in ‘TRACEFILE.txt’ which I added to simulations>modelsim. I added Testbench to the simulation and ran RTL as well as Gate Level to see if all cases passed.

The simulations show input waveform of length 8 (corresponding to two 4 bit numbers) and a output vector of length 5 (One Cout and 4 bit summation result)

Approach to the experiment:



Using the logic of a ripple carry adder as learnt in Digital Systems course, I split the problem statement into adding 4 significant bits whilst simultaneously taking care of carries and summing them up as well. In the end I implemented the classic structure we are familiar with of a Ripple Carry adder. Everything has been labelled and is self-explanatory.

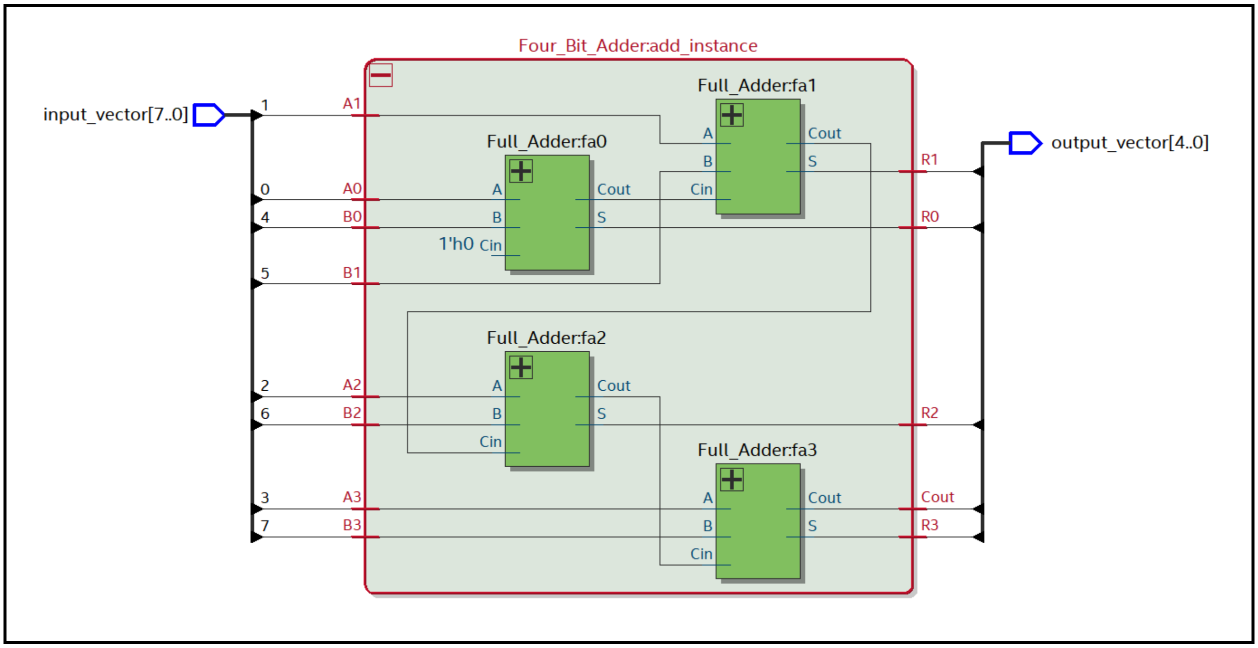
Design document and VHDL code if relevant:

DUT: Wraps the Four_Bit_Adder entity and converts the inputs and outputs to be std logic vectors.
Testbench: Compares outputs of TRACEFILE.txt and output of implementation.
Full_Adder: Acts as a full adder to add 3 bits
Four_Bit_Adder: The main logic of the design which instantiates component Full_adder and gives output as the summation of two 4 bit numbers.

Architecture of Four_Bit_Adder:

```
architecture Struct of Four_Bit_Adder is

        -- Only full adder components are used
        component Full_Adder is
                port (A, B, Cin: in std_logic; S, Cout: out std_logic);
        end component Full_Adder;

        -- Carry signals are intermediate
        signal C1, C2, C3: std_logic;

-- Port maps explicitly depict the connections and logic of the circuit
begin
        -- component instances
        fa0: Full_Adder
                port map (A => A0, B => B0, Cin => '0', S => R0, Cout => C1);

        fa1: Full_Adder
                port map (A => A1, B => B1, Cin => C1, S => R1, Cout => C2);

        fa2: Full_Adder
                port map (A => A2, B => B2, Cin => C2, S => R2, Cout => C3);

        fa3: Full_Adder
                port map (A => A3, B => B3, Cin => C3, S => R3, Cout => Cout);

end Struct;
```

RTL View:



DUT Input/Output Format:

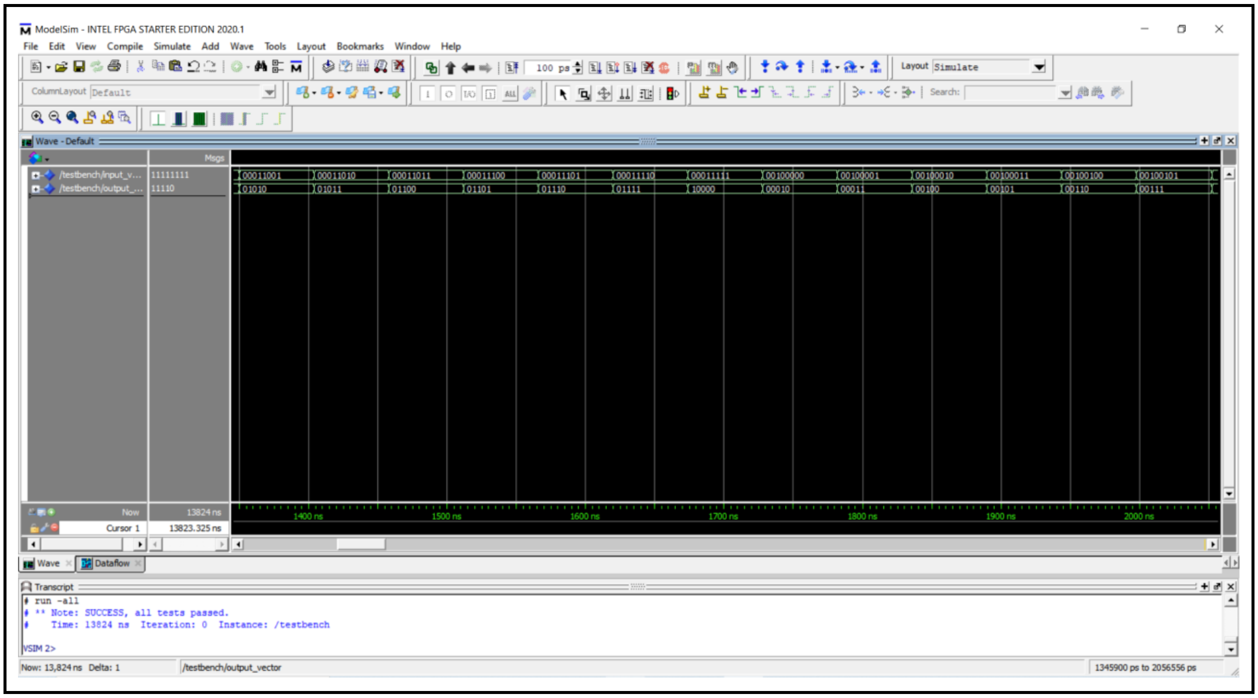
Input is a std logic vector of size 8. The LSB of the input is A0 and the MSB of the input is B3.
As one would expect, the structure is [A0, A1, A2, A3, B0, B1, B2, B3].
Output is a std logic vector of size 5 [R0 R1 R2 R3 Cout].

Few Testcases:

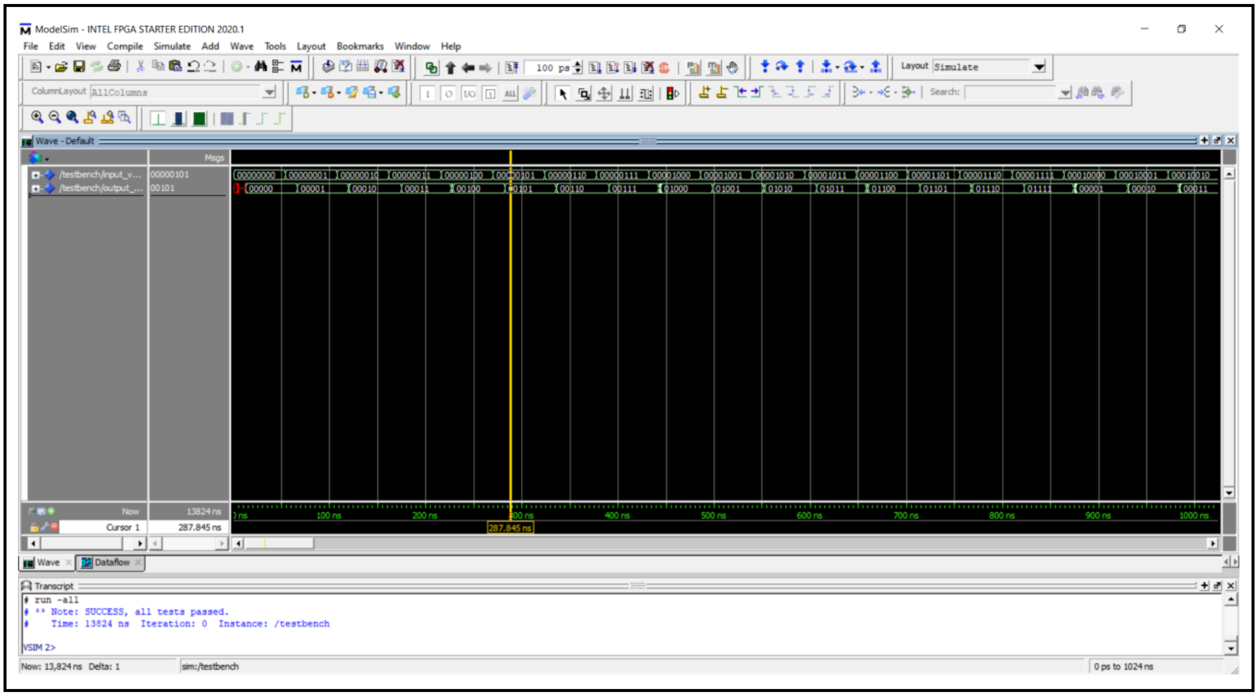
01000010 00110 11111
01000011 00111 11111
01000100 01000 11111
01000101 01001 11111
01000110 01010 11111
01000111 01011 11111
01001000 01100 11111
01001001 01101 11111

(Taken directly from TRACEFILE.txt)

RTL Simulation:



Gate-level Simulation:



Krypton board*:

Map the logic circuit to the Krypton board and attach the images of the pin assignment and output observed on the board (switches/LEDs).

Observations*:

You must summarize your observations, either in words, using figures and/or tables.

References:

NA

* To be submitted after the tutorial on Using Krypton.