Experiment 0: Combinational Circuit to count number of 1's

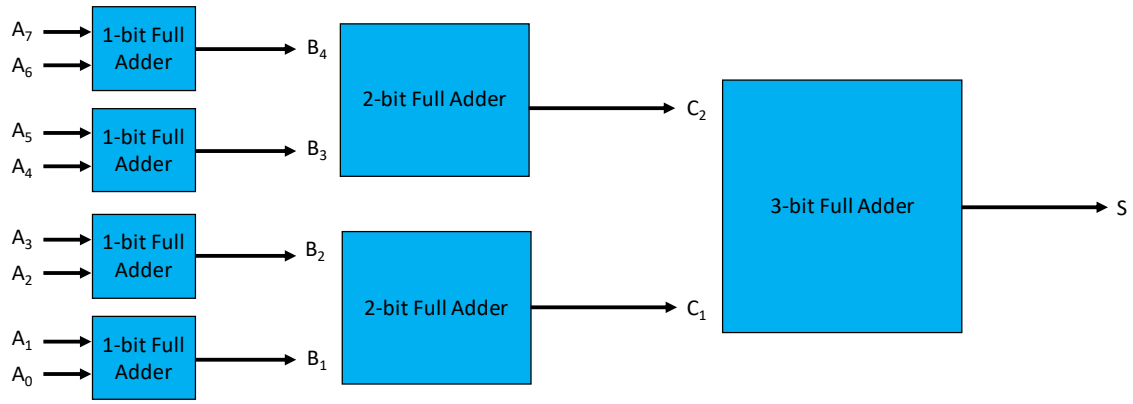
Rathour Param Jitendrakumar Roll Number: 190070049
EE-214, WEL, IIT Bombay
January 29, 2021

Overview of the experiment:

The purpose of this experiment is to design a circuit which counts the number of inputs which are 1 for a given 8-bit input, describe it using VHDL and simulate using the generic testbench using Quartus.
I designed the circuit and generated the trace file for all 256 combinations using Python as well as Julia

This report contains my approach to the experiment, design of circuit with relevant code followed by DUT input-output format, RTL netlist view, verified by output waveforms using RTL and Gate Level Simulation

Approach to the experiment:



My approach is to sum all 8 bits of input $A_7A_6A_5A_4A_3A_2A_1A_0$
This is done by making 4 pairs and adding them to get 2-bit vectors B_1, B_2, B_3 and B_4
Then 2 pairs of these vectors on adding gives 3-bit vectors C_1 and C_2
Finally adding last two gives a 4-bit vector which is the answer $S_3S_2S_1S_0$

Design document and VHDL code if relevant:

This whole design is part of Device Under Test with 8 input bits & 4 output bits which confirm our design.
See DUT Input/Output Format:

I constructed an entity named OneCounter, which follows the above approach to calculate the result.
$A_1 + A_0 = B_1$, $A_3 + A_2 = B_2$, $A_5 + A_4 = B_3$ and $A_7 + A_6 = B_4$
Now, $B_1 + B_2 = C_1$ and $B_3 + B_4 = C_2$
Finally, $C_1 + C_2 = S$ (output)
It uses four 1-bit Full Adders, two 2-bit Full Adders and one 3-bit Full Adder.

I am appending carry to output vector as MSB, hence carry not needed for above full adders. So I use 0.

Architecture of OneCounter

```
architecture Struct of OneCounter is
  signal B1,B2,B3,B4: std_logic_vector(1 downto 0);
  signal C1,C2      : std_logic_vector(2 downto 0);
begin
          -- Goal is to calculate A0 + A1 + A2 + A3 + A4 + A5 + A6 + A7 = S3S2S1S0
          -- Calculate B1,B2,B3,B4 2 bit vectors as pairwise sum of all Ai (use FULL_ADDER)
          -- Calculate C1,C2 3 bit vectors as pairwise sum of all Bi (use FULL_ADDER_2bit)
          -- Calculate S 4 bit result as C1 + C2 (use FULL_ADDER_3bit)
  FA1: FULL_ADDER
    port map (A => A(1), B => A(0), Cin => '0',  S => B1(0), Cout => B1(1)); -- A1 + A0 = B11B10 = B1

  FA2: FULL_ADDER
    port map (A => A(3), B => A(2), Cin => '0',  S => B2(0), Cout => B2(1)); -- A3 + A2 = B21B20 = B2

  FA3: FULL_ADDER
    port map (A => A(5), B => A(4), Cin => '0',  S => B3(0), Cout => B3(1)); -- A5 + A4 = B31B30 = B3

  FA4: FULL_ADDER
    port map (A => A(7), B => A(6), Cin => '0',  S => B4(0), Cout => B4(1)); -- A7 + A6 = B41B40 = B4

  F2A1: FULL_ADDER_2Bit
    port map (A => B1, B => B2, Cin => '0',  S => C1);                      -- B1 + B2 = C1

  F2A2: FULL_ADDER_2Bit
    port map (A => B3, B => B4, Cin => '0',  S => C2);                      -- B3 + B4 = C2

  F3A: FULL_ADDER_3Bit
    port map (A => C1, B => C2, Cin => '0',  S => S);                       -- C1 + C2 = S


end Struct;
```

Each 2-bit Full adder is made of 2 1-bit full adders and carry bit is added as MSB of output

```
architecture Struct of FULL_ADDER_2Bit is
  signal C: std_logic;
begin
                                        -- A1A0 + B1B0 = S2S1S0
  FA1: FULL_ADDER
    port map (A => A(0), B => B(0), Cin => Cin,  S => S(0), Cout => C); -- A0 + B0 = CS0

  FA2: FULL_ADDER
  port map (A => A(1), B => B(1), Cin => C,  S => S(1), Cout => S(2));-- A1 + B1 + C = S2S1

end Struct;
```

Each 3-bit Full adder is made of 3 1-bit full adders and carry bit is added as MSB of output

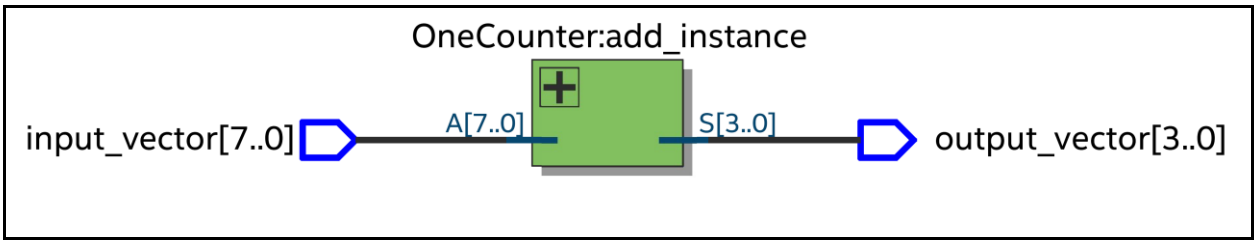
```
architecture Struct of FULL_ADDER_3Bit is
  signal C: std_logic_vector(1 downto 0);
begin
                                                                -- A2A1A0 + B2B1B0 = S3S2S1S0
  FA1: FULL_ADDER
    port map (A => A(0), B => B(0), Cin => Cin,  S => S(0), Cout => C(0));  -- A0 + B0 = C0S0

  FA2: FULL_ADDER
    port map (A => A(1), B => B(1), Cin => C(0),  S => S(1), Cout => C(1)); -- A1 + B1 + C0 = C1S1

  FA3: FULL_ADDER
    port map (A => A(2), B => B(2), Cin => C(1),  S => S(2), Cout => S(3)); -- A2 + B2 + C1 = S2S3

end Struct;
```

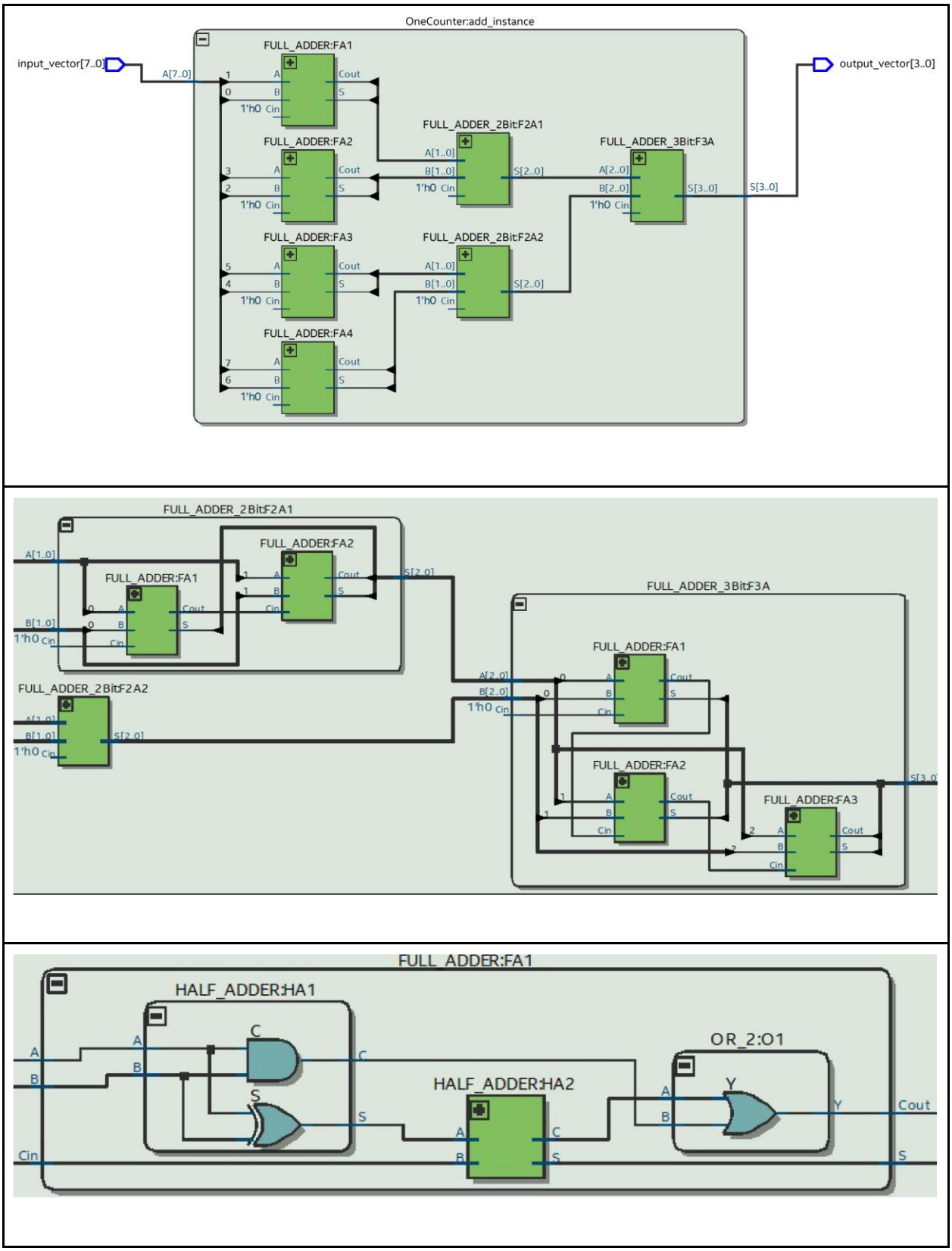
Each 1-bit Full adder is made of 2 half adders and a OR gate.
Each Half adder is made of a XOR gate and a AND gate.

Python code used for generating trace file

```python
import numpy as np

n = 8
Max = 2**n

numberOfOutputBits = int(np.floor(np.log2(n)+1))

def generateBinaryStrings(n, Max):
    return [bin(i)[2:].zfill(n) for i in range(Max)]
BinaryStrings = generateBinaryStrings(n, Max)

def OneCounter(string):
    count = 0
    for i in string:
        if (i == '1'):
          count += 1
    return count

with open("Tracefile.txt", "w") as file:
    for i in range(Max):
        I = BinaryStrings[i]
        output = OneCounter(I)
        mask = '1'*numberOfOutputBits
        O = bin(output)[2:].zfill(numberOfOutputBits)
        str = "{} {} {}\n".format(I, O, mask)
        file.write(str)
```

RTL View:



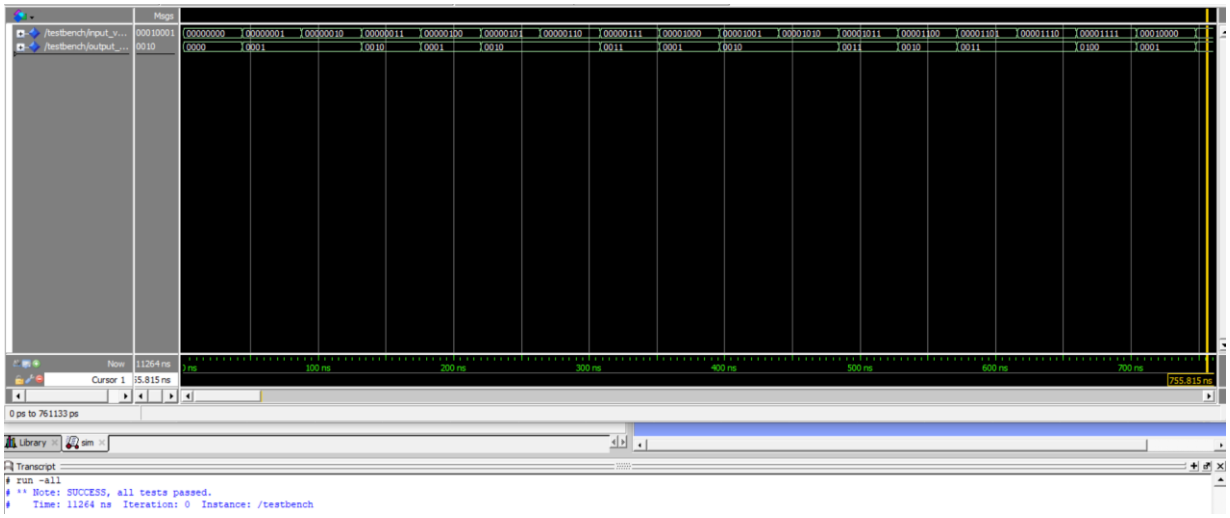

OneCounter:add_instance
input_vector[7..0]
A[7..0]
output_vector[3..0]
FULL_ADDER:FA1
FULL_ADDER:FA2
FULL_ADDER:FA3
FULL_ADDER:FA4
FULL_ADDER_2Bit:F2A1
FULL_ADDER_2Bit:F2A2
FULL_ADDER_3Bit:F3A
Cout
S
A
B
Cin
1'h0
A[1..0]
B[1..0]
S[2..0]
A[2..0]
B[2..0]
S[3..0]
HALF_ADDER:HA1
HALF_ADDER:HA2
OR_2:O1
C
Y

DUT Input/Output Format:

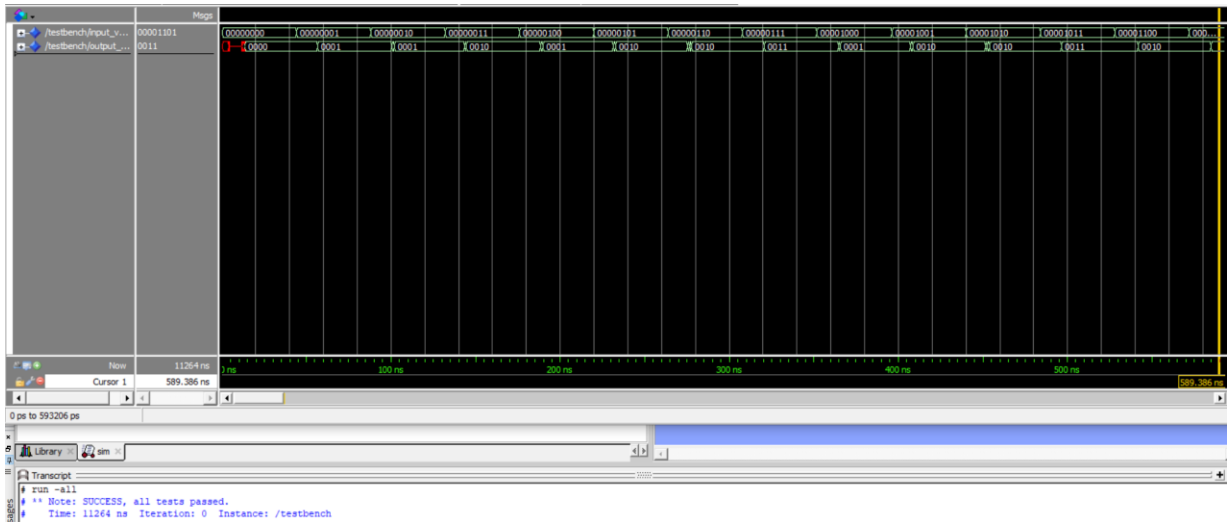
Input Format: $A_7A_6A_5A_4A_3A_2A_1A_0$	(A (8-bit A_7 is MSB A_0 is LSB))
Output Format: $S_3S_2S_1S_0$	(S (4-bit S_3 is MSB S_0 is LSB))

00000000 0000 1111
00010000 0001 1111
00100001 0010 1111
00110100 0011 1111
01000111 0100 1111
01010111 0101 1111
10011111 0110 1111
11111101 0111 1111
11111111 1000 1111

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:

Tertulien Ndjountche *Digital Electronics 2 Sequential and Arithmetic Logic Circuits* Wiley

* To be submitted after the tutorial on ”Using Krypton.