Mid-Semester Examination
BitCounter using the Krypton CPLD

Dhruv Ilesh Shah — 150070016

March 18, 2017

Overview

In this experiment, we were expected to design a combinational circuit which calculates the number of input bits that are high, in a 16-bit input.

The code was compiled on Quartus Prime, and simulated using ModelSim. GHDL was also used for simulation purposes, at a low level. This was then uploaded to the *Krypton v1.1* 5M1270ZT144C5 CPLD-based board.

The algorithm and setup has been covered in section 1. We implement the counter in a parallel fashion, significantly reducing gate delays. The VHDL codes have been kept modular and as generic as possible, for reusability and code clarity. Section 2 presents the simulation observations and miscellaneous results. Section 3 presents the observations after running the scan-chain test on the board.

1 Setup & Algorithm

The task is pretty trivial, and using a serial adder to add the 16 bits, although straight-forward, can cause a large amount of delay in the computation. It is also inefficient, as are adding the bits *one at a time*. Instead, I used the following version of *Divide & Conquer*, specific to this case.

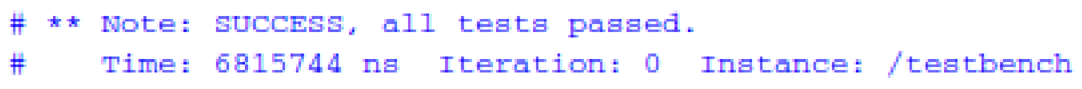
- Group the bits into four groups, of four bits each.
  - `b15 ... b12` → `int0`
  - `b11 ... b8` → `int0`
  - `b7 ... b4` → `int0`
  - `b3 ... b0` → `int0`
- Add the four bits in each group *individually* using the entity `four_bit_linear` which generates a 4-bit output. This takes time equivalent to 4 serial additions instead of 16. The factor of 4 is compensated in memory.
- Take these 4 groups of 4-bit sums, and pool them together using a `FourBitAdder` which is basically a full adder.

The advantage of the above algorithm is that it causes significantly lower gate delays, as 4 computations are now occurring in a parallel setup.

2 Observations

Here, I present the results of running RTL & Gate-Level Simulation on the design.

2.1 RTL Simulation



```
# ** Note: SUCCESS, all tests passed.
#    Time: 6815744 ns  Iteration: 0  Instance: /testbench
```

Figure 1: RTL Simulation Result for the BitCounter

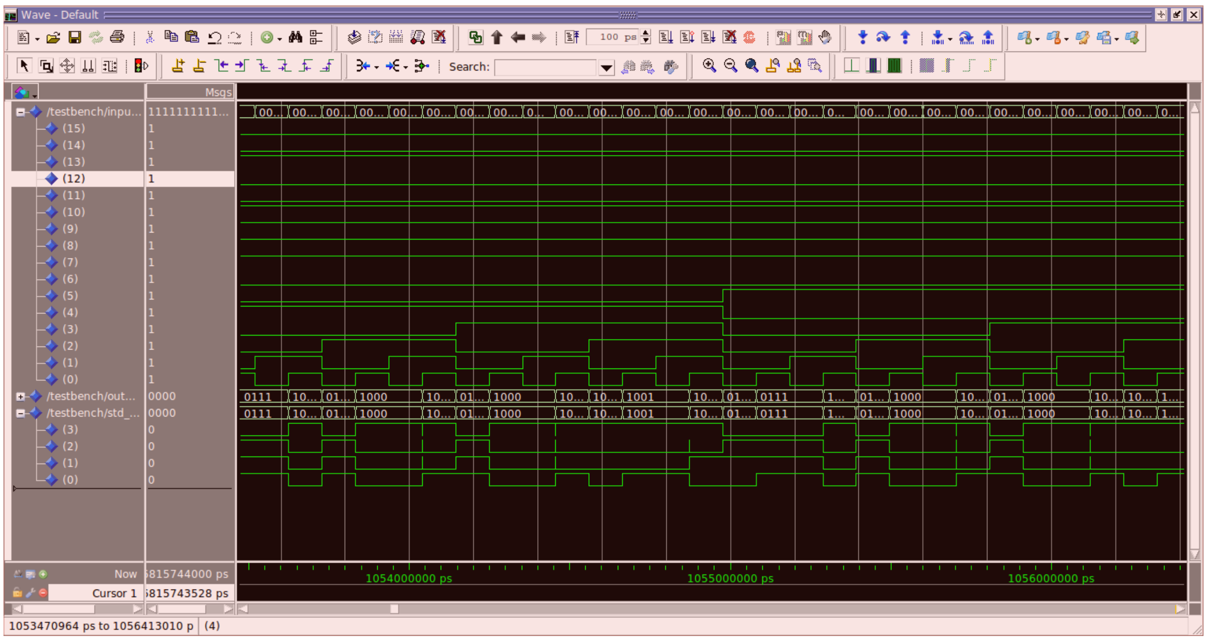
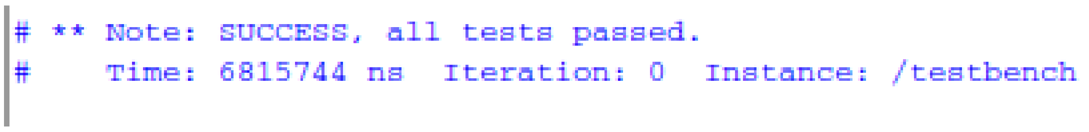


Figure 2: A snapshot RTL Simulation waveform on the BitCounter

2.2 Gate-Level Simulation



```
# ** Note: SUCCESS, all tests passed.
#    Time: 6815744 ns  Iteration: 0  Instance: /testbench
```

Figure 3: Gate-Level Simulation Result for the BitCounter

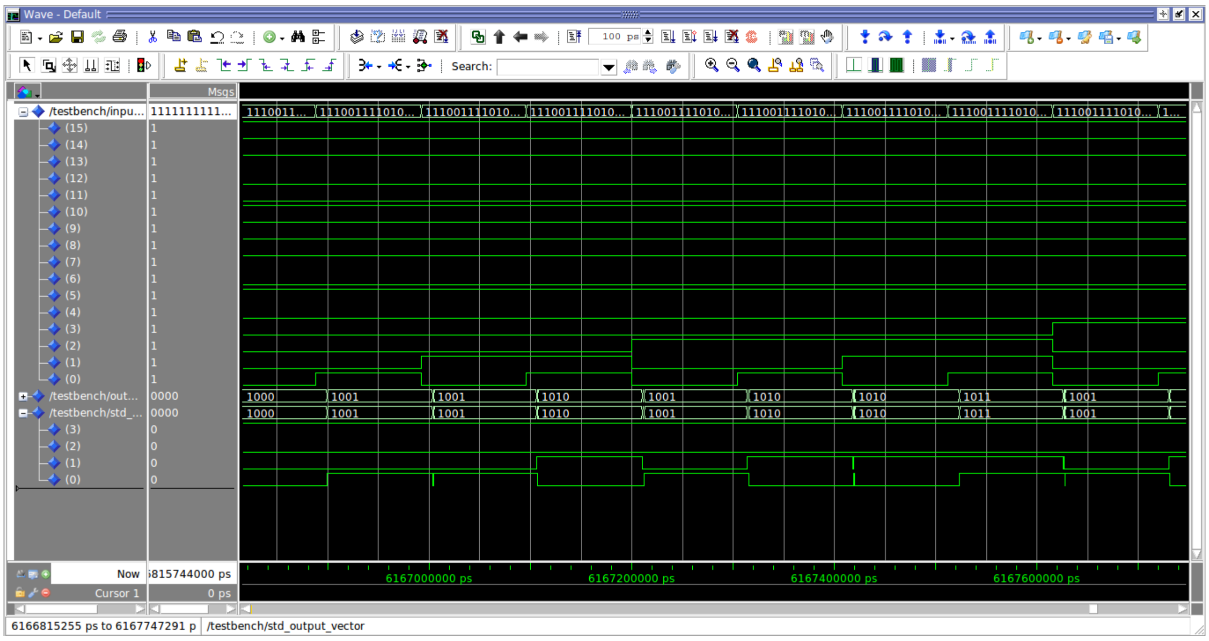
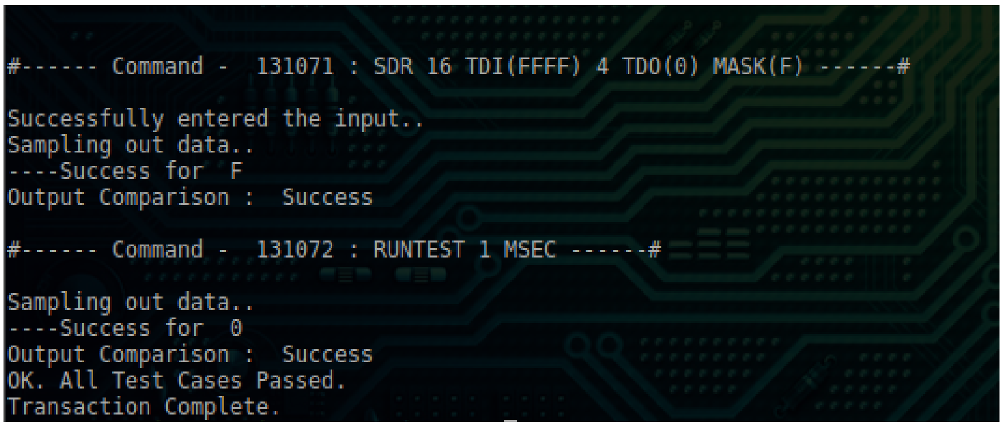


Figure 4: A snapshot gate-Level Simulation waveform on the BitCounter

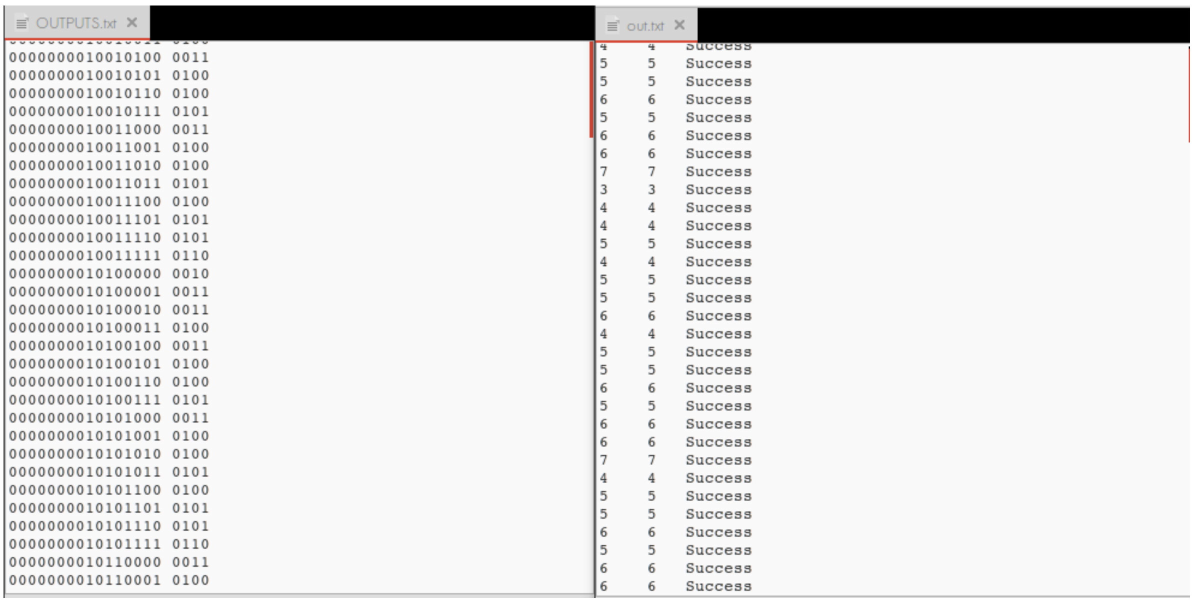
3 Scan-Chain Tests



```
#------ Command -  131071 : SDR 16 TDI(FFFF) 4 TDO(0) MASK(F) ------#

Successfully entered the input..
Sampling out data..
----Success for  F
Output Comparison :  Success

#------ Command -  131072 : RUNTEST 1 MSEC ------#

Sampling out data..
----Success for  0
Output Comparison :  Success
OK. All Test Cases Passed.
Transaction Complete.
```

Figure 5: Result of running an exhaustive Scan Chain test on the board.

For the sake of completeness, Figure 6 shows output screens after running the Gate-Level (left) and Scan-Chain (right) tests.


OUTPUTS.txt

```
0000000010010100 0011
0000000010010101 0100
0000000010010110 0100
0000000010010111 0101
0000000010011000 0011
0000000010011001 0100
0000000010011010 0100
0000000010011011 0101
0000000010011100 0100
0000000010011101 0101
0000000010011110 0101
0000000010011111 0110
0000000010100000 0010
0000000010100001 0011
0000000010100010 0011
0000000010100011 0100
0000000010100100 0011
0000000010100101 0100
0000000010100110 0100
0000000010100111 0101
0000000010101000 0011
0000000010101001 0100
0000000010101010 0100
0000000010101011 0101
0000000010101100 0100
0000000010101101 0101
0000000010101110 0101
0000000010101111 0110
0000000010110000 0011
0000000010110001 0100
```

out.txt

```
5    5    Success
5    5    Success
6    6    Success
5    5    Success
6    6    Success
6    6    Success
7    7    Success
3    3    Success
4    4    Success
4    4    Success
5    5    Success
4    4    Success
5    5    Success
5    5    Success
6    6    Success
4    4    Success
5    5    Success
5    5    Success
6    6    Success
5    5    Success
6    6    Success
6    6    Success
7    7    Success
4    4    Success
5    5    Success
5    5    Success
6    6    Success
5    5    Success
6    6    Success
6    6    Success
```

Figure 6: Output screens after running the Gate-Level (left) and Scan-Chain (right) test.

Conclusion

Starting from the very scratch, in this report, I have presented the logic and code for a combinational implementation of a bit counter. The logic was tested using RTL simulation, followed by the gate-level simulation for delay analysis and emulating the CPLD. This was followed by an actual rigorous test on the CPLD board after burning the code on it, using the *TIVA-C* microcontroller.

All the cases passed successfully at all stages and hence the complete bit counter can be used in hardware, as required.