Hardware Security Assignment I

Sudarshan Sharma

February 9, 2019

1 Problem Statement

In this assignment, you have to create adder circuits in Verilog to add n, 4-bit numbers. For convenience, let us assume a specific case where n = 128. You have to do it in the two following ways:

1. Sequentially using one single adder.

2. Using a network of cascaded adders as discussed in class. The adder network is of depth log(n).

In this assignment, you have to submit:

a) Verilog codes for both the designs.

b) A report on the time taken for both implementations for n=128. There should be some observable timing differences.

2 Solution

Ripple carry adder circuits are used to realise the adder circuit in verilog for the comparison between a sequential addition and cascaded adders in the form of log(n) levels in the experiment. For the experiment the verilog codes for both the design is generated using the python script depending on the number of the elements in the array, It is assumed that all the elements in the array are 4-bits long. The verilog file is generated according to the number n from the python script.

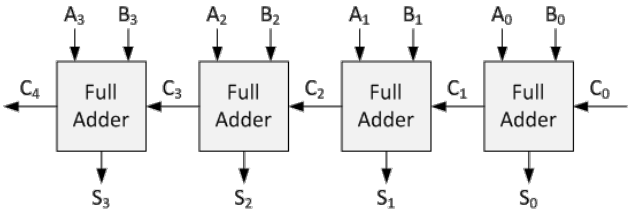

Figure 1: 4 bit Ripple Carry Adder Block Diagram.

2.1 Sequential using single Adder

In this part we design a single ripple carry adder similar to the Fig: 1, and the circuit's schematic generated from Xilinx ISE is shown in Fig:2. It can be seen as an accumulator like structure, for the sake of simplicity and considering the capabilities of verilog it is assumed that the 128 inputs are few to the module at every clock edge for addition as it is not possible to generalise a 2D bit array in verilog.

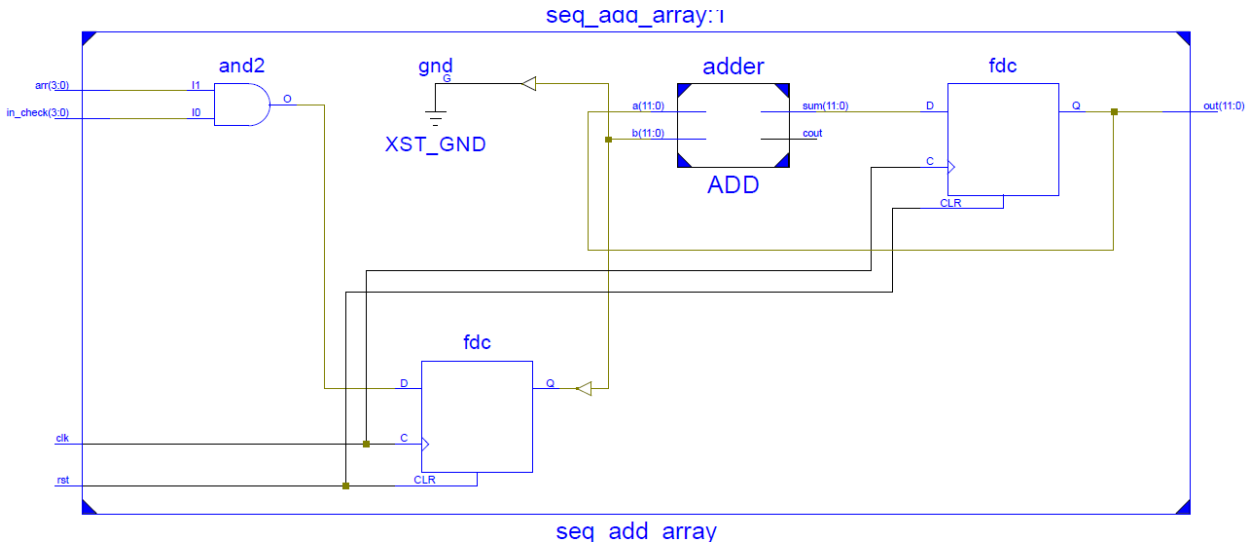

Figure 2: 4 bit Ripple Carry Adder Block Diagram.

2.1.1 Design Utilisation and Timing Summary

Slice Logic Utilization:
Number of Slice Registers: 16 out of 12480 0%
Number of Slice LUTs: 22 out of 12480 0%
Number used as Logic: 22 out of 12480 0%

Slice Logic Distribution:
Number of LUT Flip Flop pairs used: 22
Number with an unused Flip Flop: 6 out of 22 27%
Number with an unused LUT: 0 out of 22 0%
Number of fully used LUT-FF pairs: 16 out of 22 72%
Number of unique control sets: 1

IO Utilization:
Number of IOs: 22
Number of bonded IOBs: 22 out of 172 12%

Specific Feature Utilization:
Number of BUFG/BUFGCTRLs: 1 out of 32 3%

Maximum Clock Delay(i.e Time Required for 128 Additions)=1.965*128= **251.52ns**

2.2 Cascaded Adder Structure

According to the problem statement, we need to add the elements in the array in cascaded form i.e in the form of levels with number of level being $log_2 n$ which in our case is 7 for 128. The schematic in Fig. 3 describes the architecture of the design, at every level the bit length of the adder input and output increases.

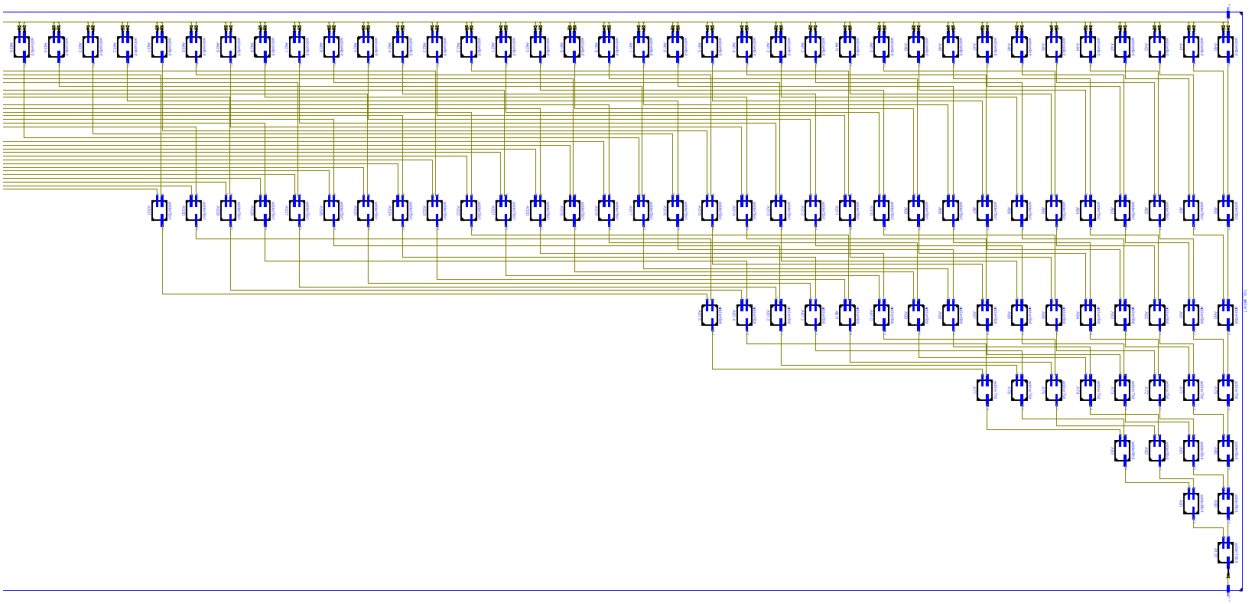

Figure 3: Snippet of Cascaded Array Structure for Array Element Addition

2.2.1 Design Utilisation and Timing Summary

Slice Logic Utilization:
Number of Slice LUTs: 857 out of 204000 0%
Number used as Logic: 857 out of 204000 0%

Slice Logic Distribution:
Number of LUT Flip Flop pairs used: 857
Number with an unused Flip Flop: 857 out of 857 100%
Number with an unused LUT: 0 out of 857 0%
Number of fully used LUT-FF pairs: 0 out of 857 0%
Number of unique control sets: 0

IO Utilization:
Number of IOs: 524

Number of bonded IOBs: 523 out of 600 87%

Maximum Clock Delay(i.e Time Required for 128 Additions)=**7.584 ns**

3 Conclusions

We can observe that the cascaded design is purely combinatorial to add the elements thus the time required is less compared to the sequential design.