Group #:
G12

Design Project 1
Revised

ENSC 350 1257

Project Submitted in Partial Fulfillment of the
Requirements for Ensc 350
Towards a Bachelor Degree in Engineering Science.

Last Name:	Last Name:	Last Name:
Saghafi	Schaufele	Singh
SID: 30145	**SID:** 30145	**SID:** 30139
5814	4255	4671

Table of Contents

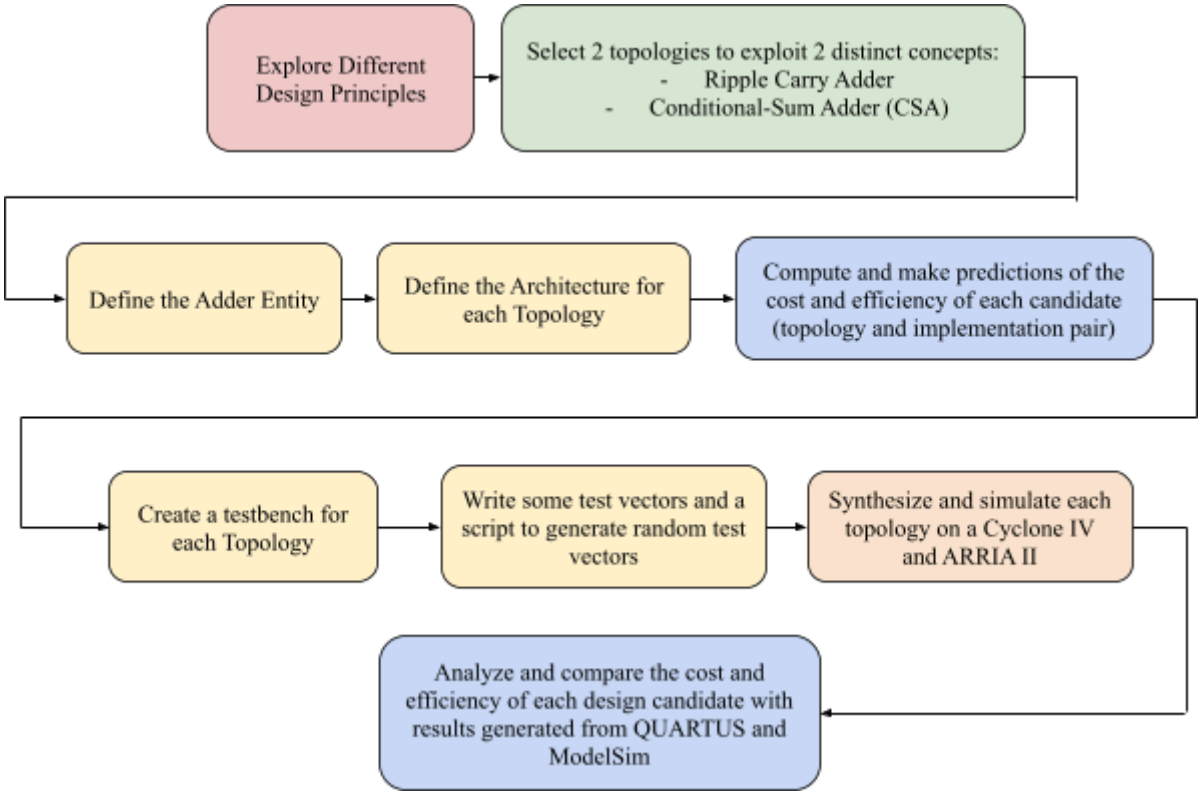
1 - Introduction

This project explored four different design candidates and two distinct adder circuit structures (topologies), each implemented on two different FPGAs. Specifically, each topology was implemented on a Cyclone IV FPGA and an ARRIA II FPGA (models EP4CE115F29C7 and EP2AGX45DF29C6, respectively). The two adder topologies were designed based on the hierarchical carry-select and carry-propagation principles. The adder entity used as the foundation for both topologies is shown in Figure 1.

```
entity EN_Adder is
    generic (N: natural := 64);
    port (
        A, B  : in std_logic_vector (N-1 downto 0);
        S : out std_logic_vector (N-1 downto 0);
        Cin : in std_logic;
        Cout, Ovfl : out std_logic
    );
end EN_Adder;
```

Figure 1: Adder Entity Definition within EN_Adder.vhd

2 - Experimental Procedures



3 - Design Candidates

Design Topologies

Ripple-Carry Adder

The ripple-carry adder topology is based on the design principle of carry propagation, where each full adder waits for the carry-out from the previous bit before producing its own sum and carry. The circuit shown in Figure 2 illustrates the first two bits of the addition; the pattern extends identically for all 64 bits. This design, implemented on a Cyclone IV, was treated as a baseline to establish functional correctness and cost benchmarks, representing the most straightforward approach.

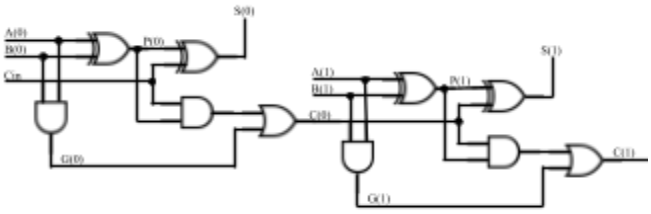


Figure 2: Ripple Adder (case N=2)

Conditional-Sum Adder

The conditional-sum adder (CSA) topology applies the hierarchical carry-select principle to reduce carry propagation delay. The N-bit addition is divided recursively into two halves until a 2-bit base case (leaf) is reached. Figures 3 and 4 show the recursive case and leaf-level carry-select operation, respectively.

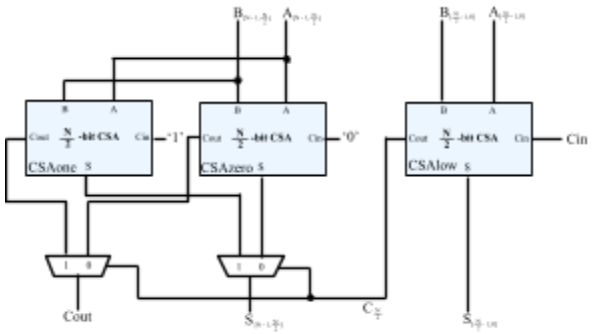


Figure 3: Generalized leaf case for the conditional sum adder

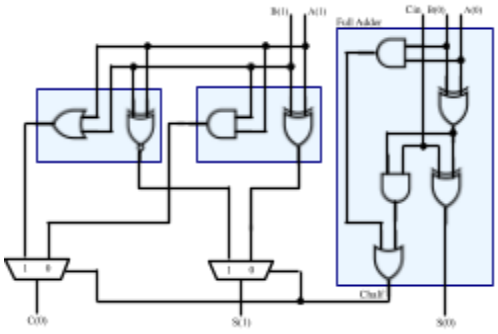


Figure 4: Leaf case (N =2) for the conditional sum adder (more detailed description)

Design Implementations

Each adder topology was implemented on two FPGA devices using structural VHDL to evaluate how architectural differences affect resource utilization and cost. The Cyclone IV uses 4-input 1-output Logic Elements (LEs), which serve as the device's fundamental building blocks for combinational and sequential logic. In contrast, the Arria II uses 8-input 2-output Adaptive Logic Modules (ALMs). Both primitive elements act as look-up tables, utilizing the architecture's Boolean expressions to map the circuit structure. Comparing designs across these devices highlights how the same circuit structure maps and is optimized differently onto FPGA families with different primitive elements.

Fitting and Cost of Candidates

Design Candidate 1(Baseline): Ripple Adder on a Cyclone IV FPGA

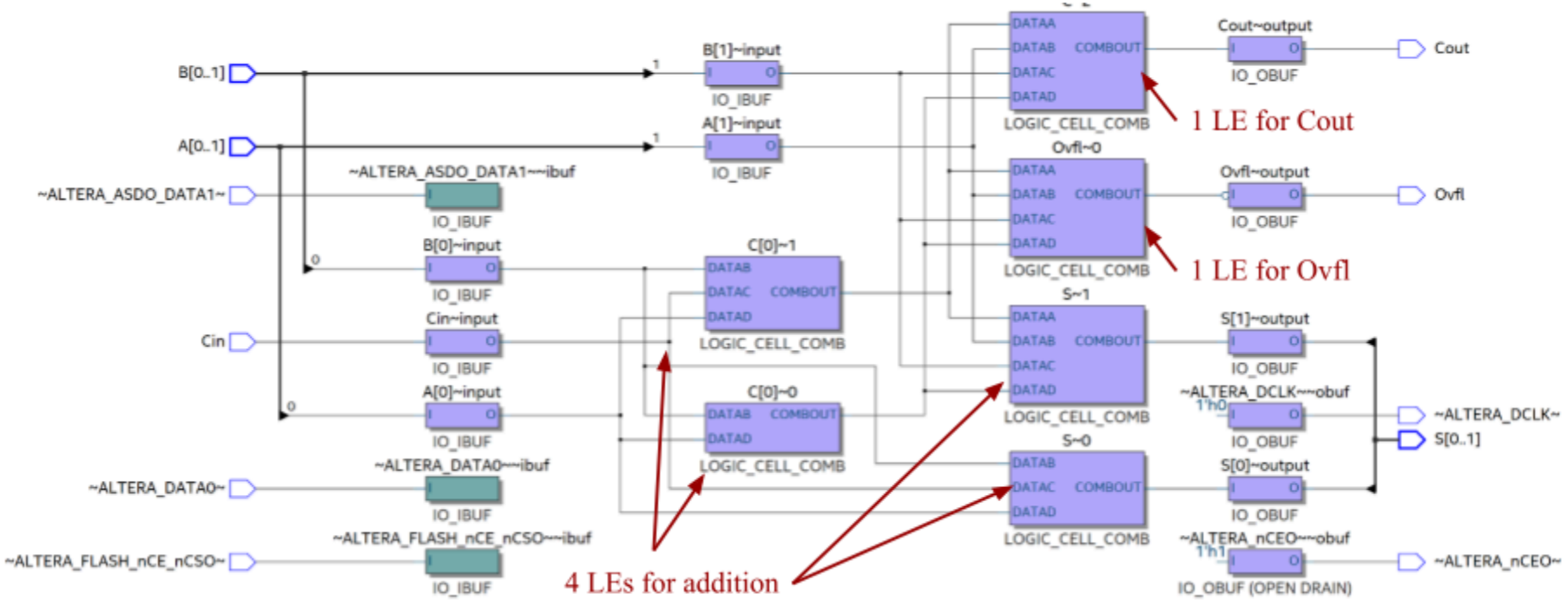


Figure 5: Ripple-Carry architecture post-fitting on a Cyclone IV FPGA for N = 2

The cost was estimated based on the Quartus fitting of the two-bit case. The two-bit case uses four logic elements (LEs) for addition, one for overflow, and one for carry-out. Extrapolating four LEs per four bits to a 64-bit implementation, while keeping the overflow and carry-out LEs constant, gives:

$$(LEs\ for\ Addition)\ *\ 32\ +\ Ovfl\ +\ Cout\ =\ 4\ *\ 32\ +\ 1\ +\ 1\ =\ 130$$

Design Candidate 2: Ripple Adder on an ARRIA II FPGA

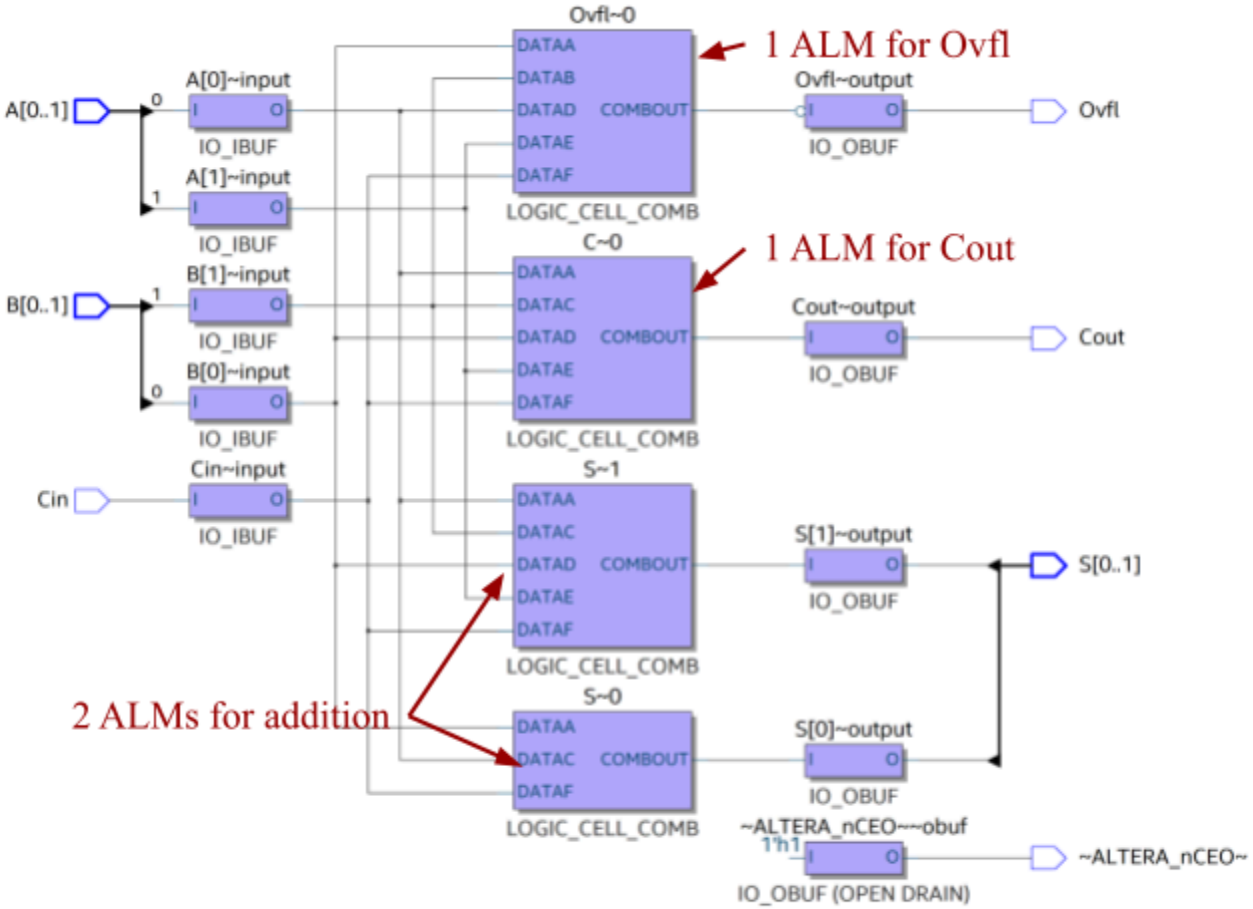


Figure 6: Ripple-Carry architecture post-fitting on an ARRIA II FPGA for N = 2

The cost was estimated based on the Quartus fitting of the two-bit case. The two-bit case uses two ALMs for addition, one for overflow, and one for carry-out. Extrapolating two ALMs per four bits to a 64-bit implementation, while keeping the overflow and carry-out ALMs constant, gives:

$$(ALMs\ for\ Addition)\ *\ 32\ +\ Ovfl\ +\ Cout\ =\ 2\ *\ 32\ +\ 1\ +\ 1\ =\ 66$$

Design Candidate 3: CSA on a Cyclone IV FPGA

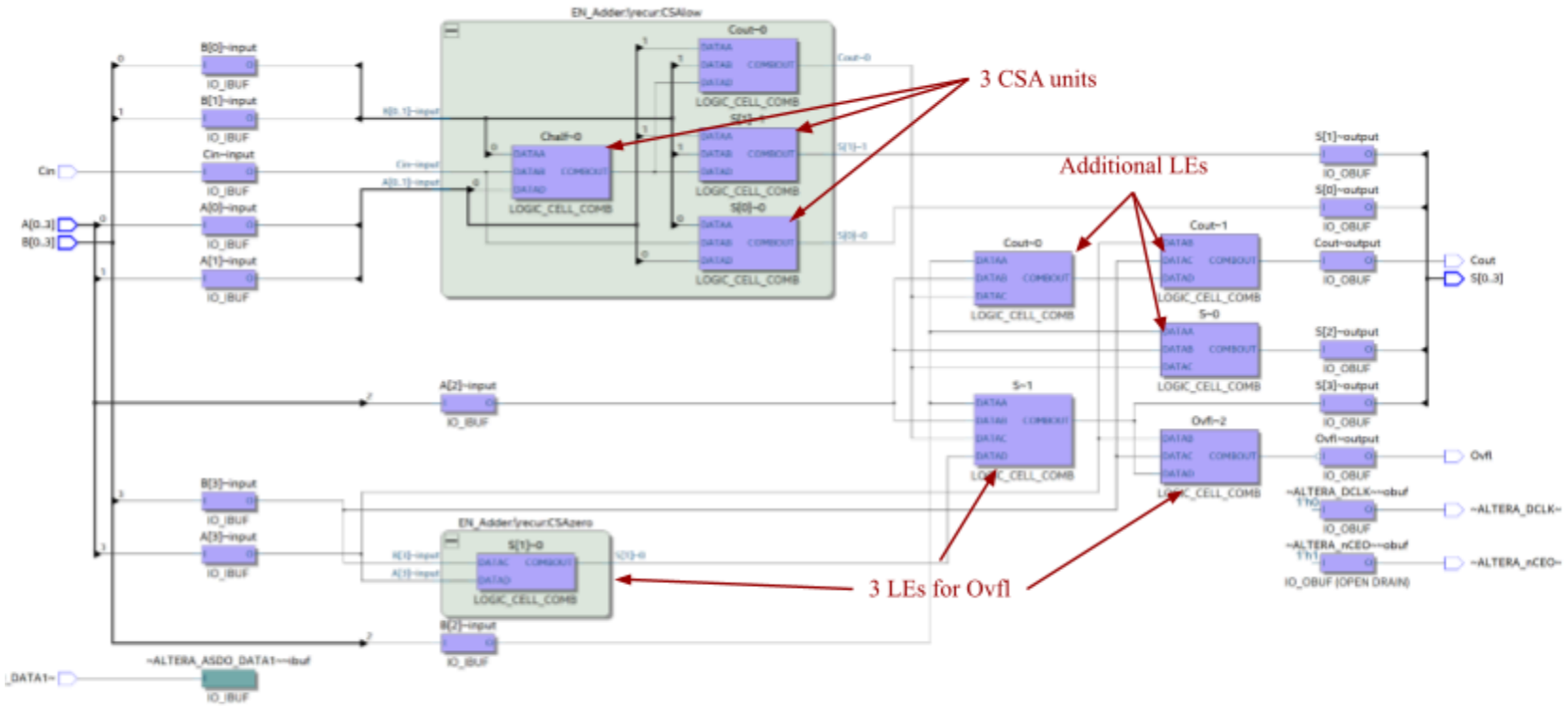


Figure 7: CSA architecture post-fitting on an Cyclone IV FPGA for N = 4

The cost was estimated based on the Quartus fitting of the four-bit case. Each level includes three CSA units from the previous level. Elements outside the CSA unit were extrapolated using the four-bit, which uses three LEs for addition. The LEs for overflow were kept constant at three. The calculations are shown below:

$$C_4 = 7$$

$$C_8 = 3C_4 + 2(3) = 27$$

$$C_{16} = 3C_8 + 4(3) = 93$$

$$C_{32} = 3C_{16} + 8(3) = 303$$

$$C_{64} = 3C_{32} + 16(3) = 957$$

$$C_{64} + Ovfl = +3 = 960$$

Design Candidate 4: CSA on an ARRIA II FPGA

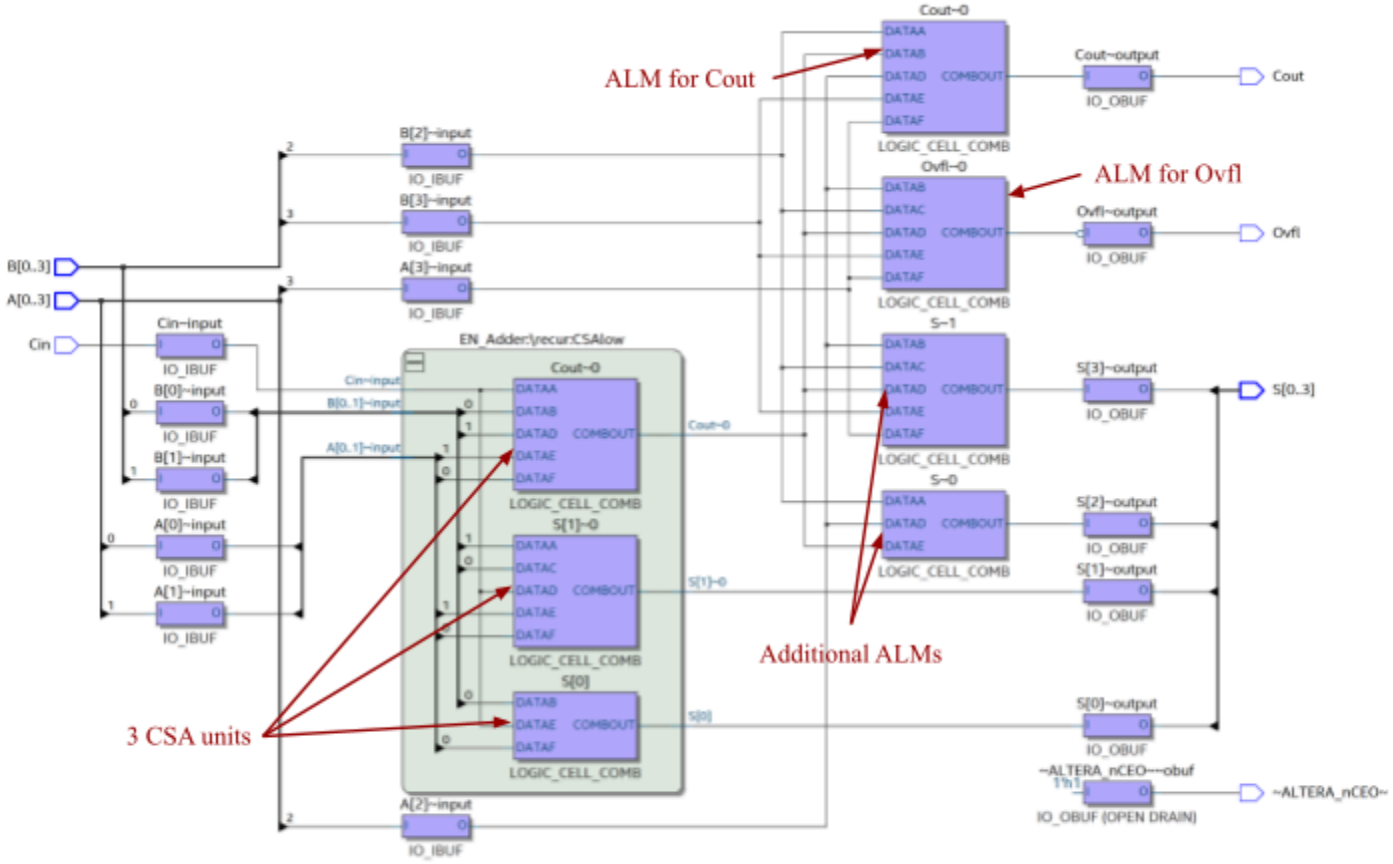


Figure 8: CSA architecture post-fitting on an ARRIA II FPGA for N = 4

The cost was estimated based on the Quartus fitting of the four-bit case. Each level includes three CSA units from the previous level. Elements outside the CSA unit were extrapolated from the four-bit case, which uses two ALMs for addition. The ALMs for overflow and Cout were kept constant at two. The calculations are shown below:

$$C_4 = 5$$

$$C_8 = 3C_4 + 2(2) = 19$$

$$C_{16} = 3C_8 + 4(2) = 65$$

$$C_{32} = 3C_{16} + 8(2) = 211$$

$$C_{64} = 3C_{32} + 16(2) = 665$$

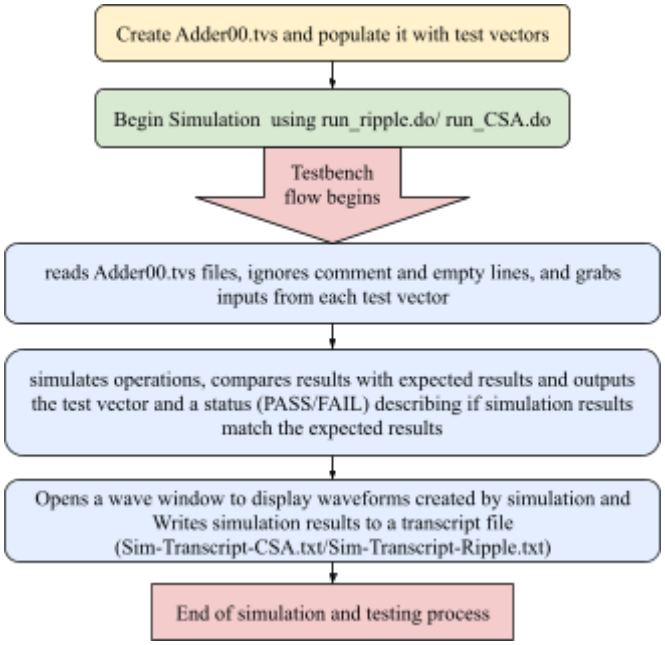
$$C_{64} + (Ovfl \,\&\, Cout) = 665 + 2 = 667$$

4 - Testbenches

A dedicated testbench was developed for each adder topology to verify functional correctness and consistency. Both testbenches utilize a standard test vector file, Adder00.tvs. The test vector file contains a comprehensive set of test cases, including: Base cases for standard addition, Carry-out (unsigned overflow) cases, Two's-complement (signed overflow) cases, and several randomly generated test vectors to validate robustness. Each test vector occupies a single line in the file and follows the format:

A, B, Cin, Sum, Cout, Overflow,

where A and B represent operands in hexadecimal, Cin is the input carry, and Sum, Cout, and Overflow denote the expected outputs. Sample waveforms from the CSA testbench are seen in Figure 9; the ripple-carry adder waveforms were identical.



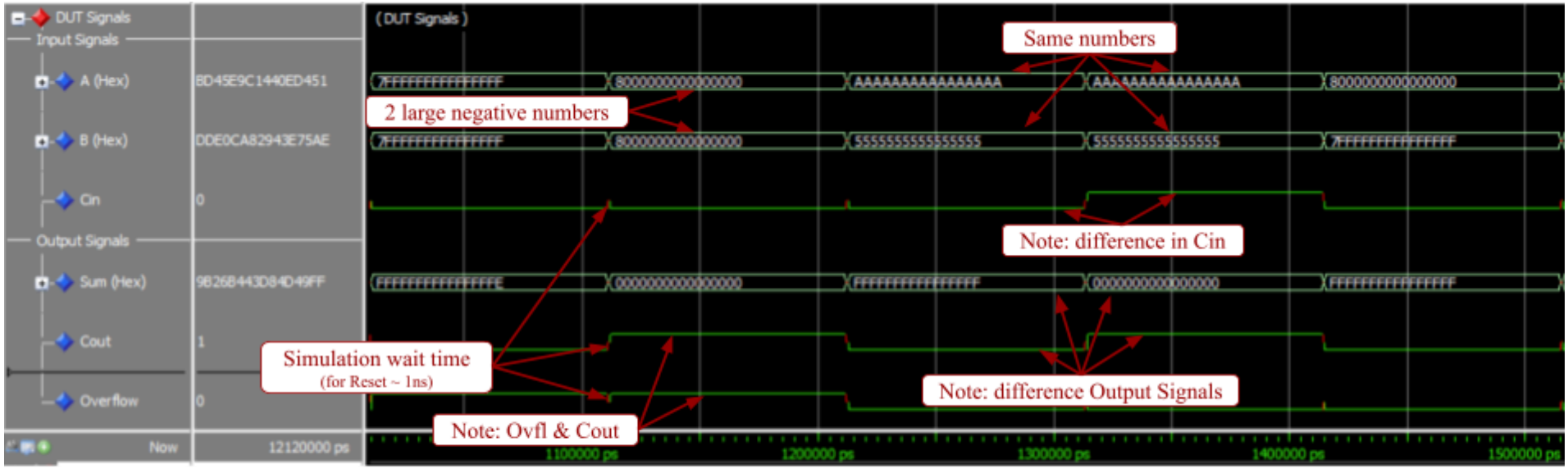


Figure 9: Sample outputs Waveform from CSA Testbench

5 - Cost Analysis

The costs for the design candidates were calculated using this formula:

$$Cost\ per\ LE/ALM\ =\ Device\ price\ \div\ Total\ Device\ LEs/ALMs$$

This method of calculating the cost factors accounts for the total resources available to the board and the fraction of those resources used by the circuit. Resource data was obtained from the Quartus synthesis reports. LEs were selected as the unit of cost, and all ALMs were converted to LEs using a conversion factor. The normalized cost is obtained by dividing each candidate cost by the baseline cost. The boards used to synthesize the architectures are currently obsolete. This inflates the price and makes cost calculations inaccurate. To account for these inaccuracies, the Cyclone IV Nano and the ARRIA 10 GX 480 were considered. These are comparable to the devices used in synthesis, but they have more accurate pricing because they are currently active on DigiKey, indicating they are still widely used and in production. The following calculations assume that the LEs and ALMs on the available devices cost the same as the original board's elements.

Device	Total Device LEs/ALMs	Device Price (CAD)	Cost per LE/ALM (CAD)	Normalized Cost in terms of LEs
Cyclone IV Nano	22320	169.49[3]	$0.00759184	1
Arria 10 GX 480	181790	3937.25[4]	$0.02165823	2.85

Design Candidate	Target Device	Resource Usage Predications (LEs/ALMs)	Resources Used (LEs/ALMs)	Predicted Cost (CAD)	Candidate Cost (CAD)	Cost (LEs)	Normalized Cost (W.R.T. Baseline LEs)
1 (Baseline)	Cyclone IV	130	161	$0.99	$1.22	161	1
2 (Ripple)	ARRIA II	66	147	$1.43	$3.18	418.95	2.60
3 (CSA)	Cyclone IV	960	271	$7.29	$2.06	271	1.68
4 (CSA)	ARRIA II	667	188	$14.45	$4.07	535.8	3.33

Figure 9: Cost Calculations of Design Candidates

The conversion factor from LEs to ALMs was 2.85, slightly more than double. The predicted costs (see the Design Candidates section for calculations) differed significantly from the experimental results. The reasons for this will be discussed in the Results and Analysis Section.

6 - Results Analysis

The synthesis results show clear cost differences between the two FPGA platforms. The Cyclone IV was consistently more cost-effective, with normalized costs of 1.00 and 1.68 for the ripple-carry and conditional-sum adders, compared to the Arria II's 2.60 and 3.33. As expected, the ripple-carry adder used the fewest resources (161 LEs on Cyclone IV, 147 ALMs on Arria II). The conditional-sum adder required significantly more area: 68% more on Cyclone IV and 28% more on Arria II. This demonstrates the area penalty of parallel architecture, as it must duplicate circuitry to calculate both possible Cin values.

The ripple-carry adder's prediction cost was underestimated, suggesting that the VHDL code was not optimized for minimum resource consumption. On the other hand, the conditional-sum adder’s prediction costs were overestimated, likely due to Quartus' optimization and its efficient modelling of recursive circuitry.

In conclusion, for cost-sensitive designs, the ripple-carry adder on the Cyclone IV FPGA provides the best value. The conditional-sum adder trades area for performance, making it suitable for speed-critical applications where the extra cost is justified.

8 - Appendix

A.1 Directory Structure

(folder) Documentation

- (folder) ActivityLogs
  - a folder including each group member's activity log worksheets
- (folder) Images
  - a folder containing all the images used in the report
- (folder) OutputFiles
  - (file) Quartus-Summaries.text
    - Includes summary reports from the synthesis of each design candidate
  - (file) Sim-Transcript-CSA.text
    - Includes the transcript from the conditional sum adder testbench, displays the test vector index, as well as the test vectors, and the status of each tested test vector
  - (file) Sim-Transcript-Ripple.text
    - Includes the transcript from the ripple-carry adder testbench, displays the test vector index, as well as the test vectors, and the status of each tested test vector
- (file) VHDLSrc_EN_Adder.pdf
  - A PDF listing of the VHDL code for the Adder Entity and Architectures
- (file) VHDLSrc_TB_Adder_CSA.pdf
  - A PDF listing of the VHDL code for the testbench written for the conditional sum adder topology
- (file) VHDLSrc_TB_Adder_RIP.pdf
  - A PDF listing of the VHDL code for the testbench written for the ripple adder topology
- (file) DP1-Report-G12-350-1257.pdf
  - Project Report, including cost and efficiency analysis (this document)
- (file) DP1-Summary-G12-350-1257.pdf
  - A summary of tasks completed in this project, as well as a brief overview of the cost and efficiency analysis

(folder) Simulation

- (folder) questa
  - folder created by modelsim, includes pre-compiled netlists
- (folder) TestVectors
  - Adder00.tvs
    - File including all the test vectors used by the testbenches
  - gen_testVec.py
    - a python script to append randomly generated test vectors to the Adder00.tvs file
- (files) DP1.cr.mti / DP1.mpf
  - Modelsim project files for simulation and testing
- (file) run_CSA.do
  - Script to compile, add waves, run simulation and create a simulation transcript for the conditional-sum adder topology testbench
- (file) run_ripple.do
  - Script to compile, add waves, run simulation and create a simulation transcript for the ripple-carry adder topology testbench
- (file) TB_Adder_CSA.vhd
  - Conditional-sum adder testbench written in VHDL
- (file) TB_Adder_RIP.vhd
  - Ripple-Carry adder testbench written in VHDL

(folder) SourceCode

- (file) EN_Adder.vhd
  - Defines the adder entity as well as a ripple adder, fast ripple adder, and conditional sum adder architectures in VHDL

(file) DP1.qpf / DP1.qsf

- Quartus project files for synthesis and analysis of the digital circuits

A.2 References

[1] "DigiKey," 2 November 2009. [Online]. Available: https://www.digikey.ca/en/products/detail/altera/EP4CE115F29C7/2260452. [Accessed 12 October 2025].

[2] "DigiKey," 2 February 2009. [Online]. Available: https://www.digikey.ca/en/products/detail/altera/EP2AGX45DF29C6/2349475. [Accessed 12 October 2025].

[3] "DigiKey," 28 October 2014. [Online]. Available: https://www.digikey.ca/en/products/detail/terasic-inc/P0082/2625112. [Accessed 25 October 2025].

[4] "DigiKey," 29 September 2023. [Online]. Available: https://www.digikey.ca/en/products/detail/iwave-global/IW-G24D-CU0F-4D004G-S008G-NCI/15780313. [Accessed 25 October 2025].