Group #:
G12

Design Project 1

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

1- Table of Contents

1- Table of Contents **1**
2 - Introduction **2**
3 - Experimental Procedures **2**
4 - Design Candidates **3**
Design Candidate 1 (Baseline) and 2: Ripple Adder Topology 3
Design Candidate 1(Baseline): Ripple Adder on a Cyclone IV FPGA 3
Design Candidate 2: Ripple Adder on an ARRIA II FPGA 5
Design Candidate 3 and 4: Conditional-Sum Adder Topology 6
Design Candidate 3: CSA on a Cyclone IV FPGA 7
Design Candidate 4: CSA on an ARRIA II FPGA 8
5 - Testbenches **9**
6 - Cost Analysis **10**
Theoretical Cost Predictions 10
Design Candidate 1(Baseline): Ripple Adder on a Cyclone IV FPGA 10
Design Candidate 2: Ripple Adder on an ARRIA II FPGA 10
Design Candidate 3: CSA on a Cyclone IV FPGA 11
Design Candidate 4: CSA on an ARRIA II FPGA 11
Experimental Cost Calculations 12
Theoretical VS Experimental Results 12
7 - Results Analysis and Conclusion **13**
8 - Appendix **14**
A.1 Directory Structure 14
A.2 References 17

2 - Introduction

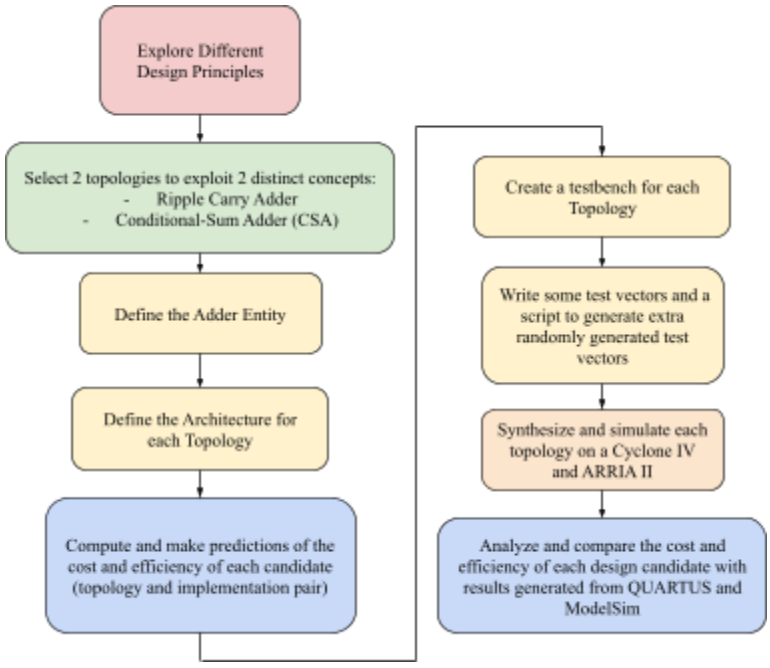
This project explored four different design candidates and two distinct adder topologies, each implemented on two different FPGAs. Specifically, each topology was implemented on a Cyclone IV FPGA and an ARRIA II FPGA (models EP4CE115F29C7 and EP2AGX45DF29C6, respectively). These devices were selected to compare the cost and efficiency of each topology when mapped onto architectures using 4-input logic elements (Cyclone IV) versus 8-input adaptive logic modules (ARRIA II). The two adder topologies were designed based on the hierarchical carry-select principle and the carry-propagation principle. 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

3 - Experimental Procedures

In this experiment, two 64-bit adder topologies, the ripple-carry adder and the conditional-sum adder (CSA), were first implemented in VHDL and synthesized using Quartus for two FPGA devices: the Cyclone IV E and the Arria II. Each design was analyzed, fitted, and compiled to examine hardware synthesis behaviour and ensure proper implementation. Following the synthesis, separate testbenches were developed for both architectures to verify their functionality through simulation. The testbenches read test vectors from a .tvs file containing predefined input and expected output values, which were applied to the adders to confirm correct operation under various input conditions.



4 - Design Candidates

Design Candidate 1 (Baseline) and 2: Ripple Adder Topology

The ripple-carry architecture of the EN_Adder implements a 64-bit ripple-carry adder structure, where each bit stage computes its sum and carry sequentially based on the results of the previous stage. For every bit position, the propagate and generate signals are formed. This causes the carry output of each stage to depend on the carry from the preceding bit. The sum bit of that stage is then determined. This dependency causes the carry signal to “ripple” through all 64 stages from the least significant bit to the most significant bit. This simplifies the VHDL design but introduces a longer carry propagation delay as N increases. The architecture also computes the final carry-out (Cout) from the last stage and the overflow flag (Ovfl) to detect arithmetic overflow for signed operations. The circuit displayed in Figure 2 displays the first two bits of addition. For the 64-bit case, it would continue in the same manner, adding a full adder for each bit.

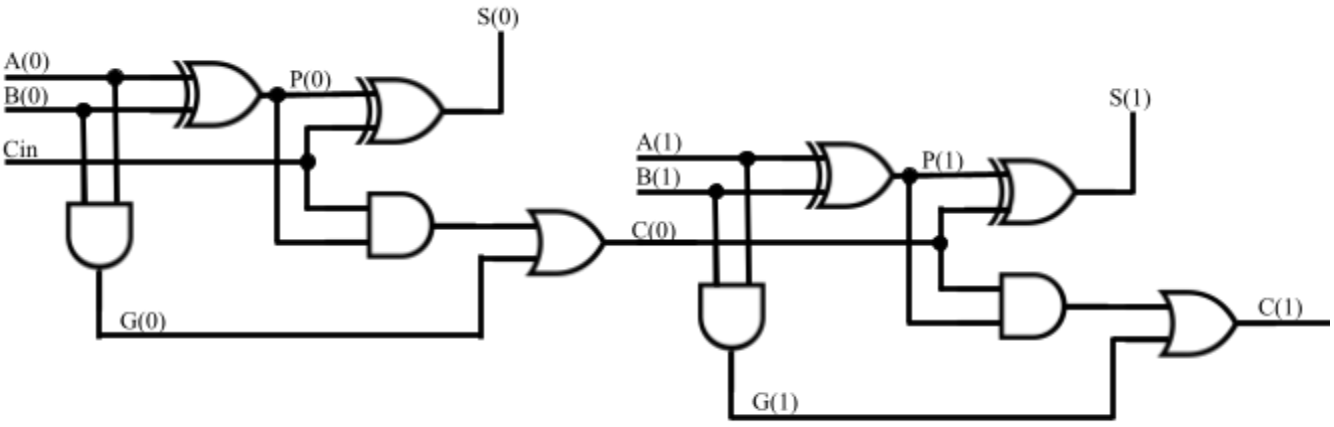


Figure 2: Ripple Adder (case N=2)

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

The cost of this candidate will be used as a baseline when discussing the following design candidates. The total number of logic elements and the price of the device were found online and used to calculate the cost per logic element, as shown below.

Target Device	Total Device LEs	Device Price (CAD)	Cost per LE (CAD)
Cyclone IV (EP4CE115F29C7)	114480	1029.56472 [1]	0.008993403

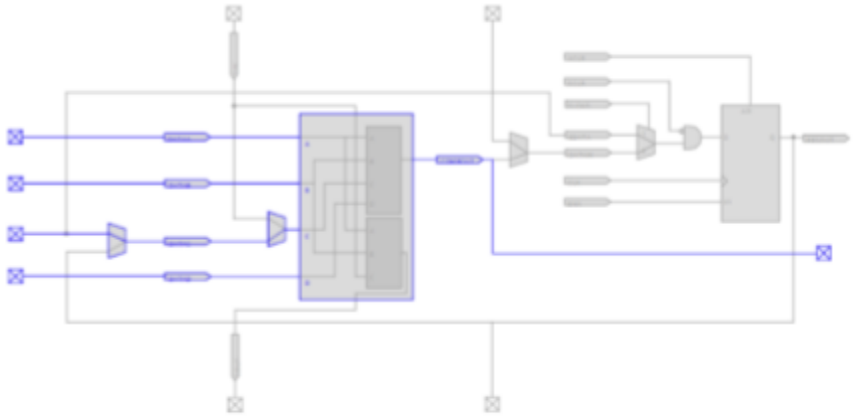


Figure 3: A single logic element on Cyclone IV FPGA (EP4CE115F29C7)

In Figure 3, it is evident that each Cyclone IV FPGA logic element contains a four-input look-up table (LUT). Since both the sum and carry outputs of a full adder can be derived from their respective truth tables, it is estimated that each full adder requires two logic elements on this device. Based on this assumption, the 64 full adders connected in series are expected to utilize approximately 128 logic elements in total. An additional logic element is included to account for the overflow computation, resulting in an estimated total of 129 logic elements and an approximate implementation cost of $1.16 CAD.

Design Candidate 2: Ripple Adder on an ARRIA II FPGA

The total number of logic elements and the price of the device were found online and used to calculate the cost per logic element, as shown below.

Target Device	Total Device ALMs	Device Price (CAD)	Cost per ALM (CAD)
ARRIA II (EP2AGX45DF29C6)	36100	1273.03889 [2]	0.035264235

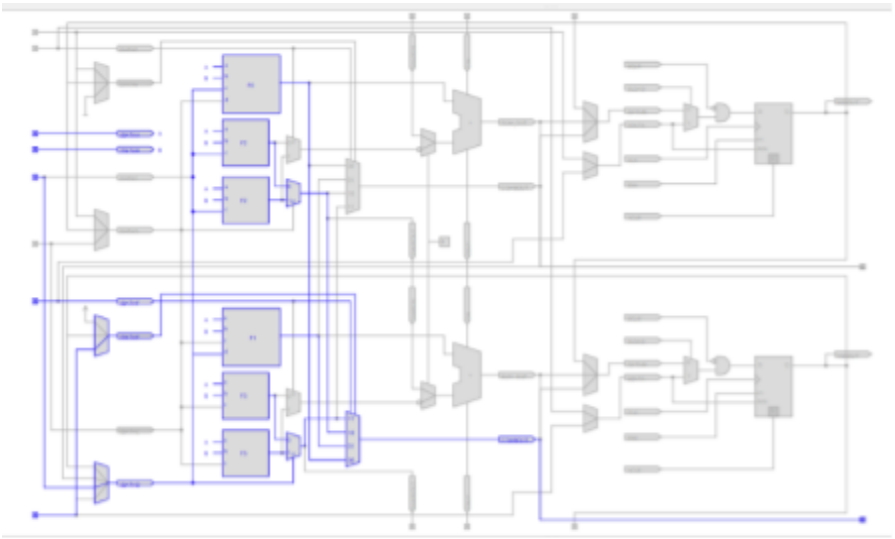


Figure 4: A single adaptive logic module on ARRIA II FPGA (EP2AGX45DF29C6)

A single Adaptive Logic Module (ALM) on the Arria II FPGA can accommodate up to eight inputs and provide up to two outputs, as shown in Figure 4. Under ideal conditions, this makes each ALM approximately equivalent to two Logic Elements (LEs) on the Cyclone IV. Based on this estimation, the design would require around 64 ALMs to implement the same topology. An additional ALM is included to account for the overflow computation, resulting in an estimated total of 65 ALMs and an approximate implementation cost of $2.29 CAD. This is about 2 times higher than the baseline Cyclone IV device, reflecting the greater cost of ALMs relative to LEs.

.

Design Candidate 3 and 4: Conditional-Sum Adder Topology

The conditional-sum adder (CSA) architecture of our adder entity is implemented recursively. The N-bit addition is repeatedly divided into two halves until the base case (leaf) of a 2-bit adder is reached. At each recursive level, the lower half of the adder computes its sum and an intermediate carry. Simultaneously, two upper-half adders operate in parallel, assuming Cin = 0 and Cin = 1, as shown in Figure 3. Once the correct carry from the lower section is known, the correct precomputed upper sum and carry are selected using multiplexers. The leaf stage handles the case when the adder reaches the recursive end condition, N = 2. At this stage, it uses the carry select principle to complete a 2-bit addition, as shown in Figure 4.

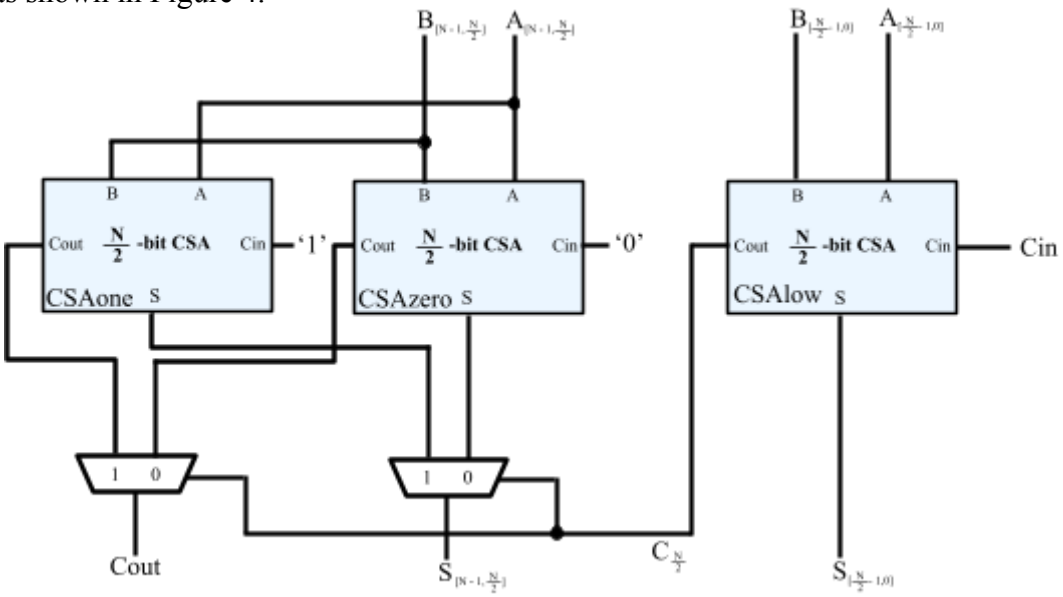


Figure 3: Generalized leaf case for the conditional sum adder

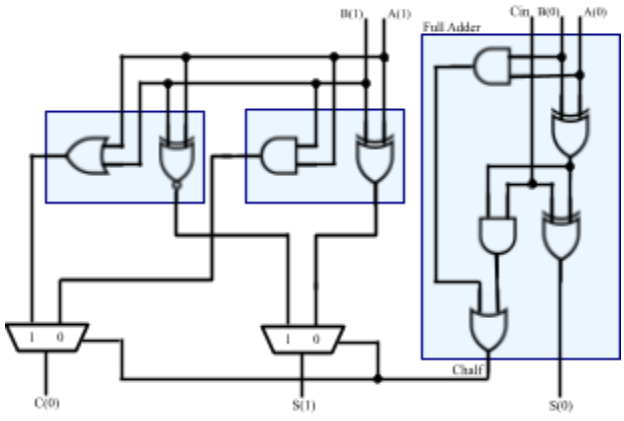


Figure 4: Leaf case (N =2) for the conditional sum adder (more detailed description)

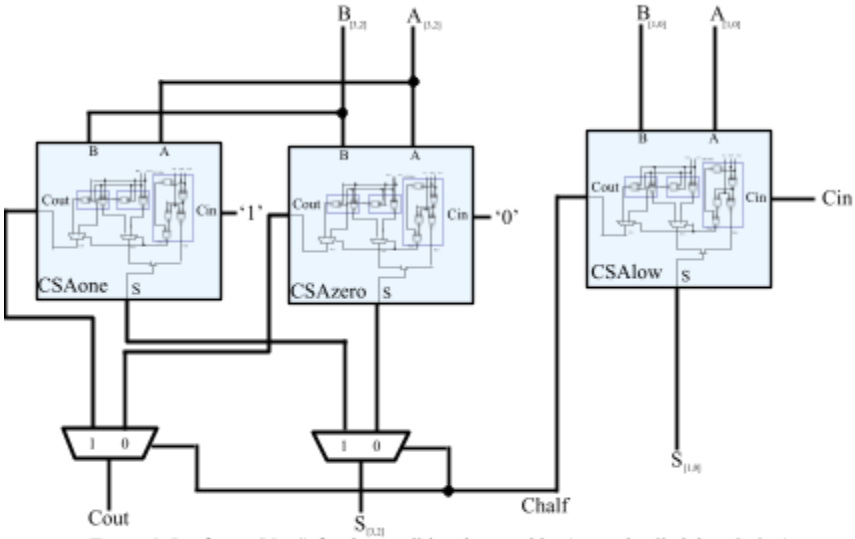


Figure 5: Leaf case (N =4) for the conditional sum adder (more detailed description)

Design Candidate 3: CSA on a Cyclone IV FPGA

The total number of logic elements and the price of the device were found online and used to calculate the cost per logic element, as shown below.

Target Device	Total Device LEs	Device Price (CAD)	Cost per LE (CAD)
Cyclone IV (EP4CE115F29C7)	114480	1029.56472 [1]	0.008993403

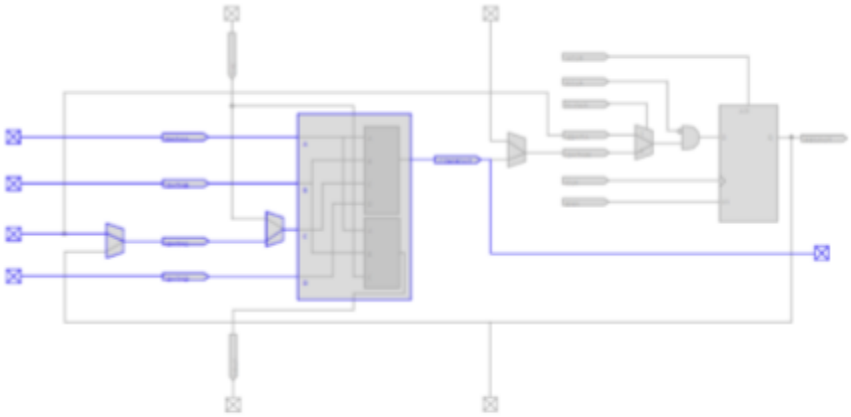


Figure 6: A single logic element on Cyclone IV FPGA (EP4CE115F29C7)

Under the assumption that each full adder requires two logic elements on the Cyclone IV, and that one logic element is used for the sum of the upper half and another for the carry of the upper half, we estimate that the two-bit leaf case uses four logic elements. Using this as the base case, the number of logic elements required for larger adders can be calculated recursively as follows:

$$C_2 = 4$$

$$C_4 = 3C_2 + 2(2) = 16$$

$$C_8 = 3C_4 + 2(4) = 56$$

$$C_{16} = 3C_8 + 2(8) = 184$$

$$C_{32} = 3C_{16} + 2(16) = 584$$

$$C_{64} = 3C_{32} + 2(32) = 1816$$

An additional logic element is included to account for the overflow computation, resulting in an estimated total of 1,816 logic elements, with an approximate implementation cost of $16.33 CAD. This represents about 14 times higher resource usage compared to the baseline topology.

Design Candidate 4: CSA on an ARRIA II FPGA

The total number of logic elements and the price of the device were found online and used to calculate the cost per logic element, as shown below.

Target Device	Total Device ALMs	Device Price (CAD)	Cost per ALM (CAD)
ARRIA II (EP2AGX45DF29C6)	36100	1273.03889 [2]	0.035264235

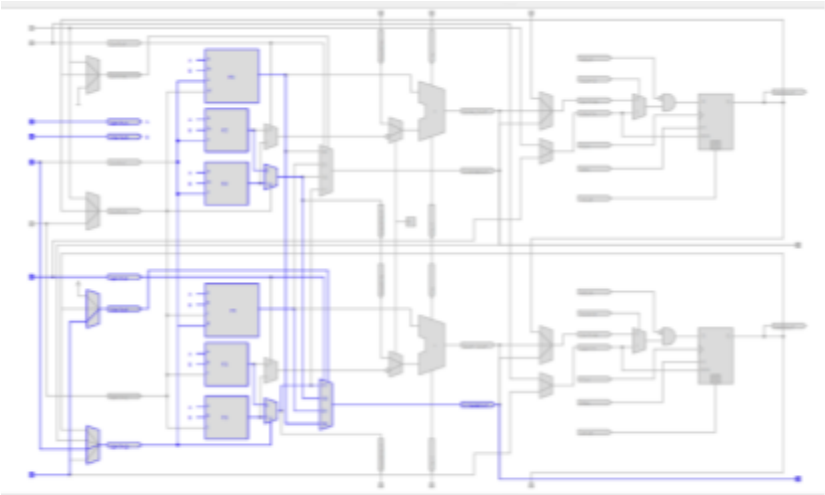


Figure 7: A single adaptive logic module on ARRIA II FPGA (EP2AGX45DF29C6)

Under the assumption that each full adder requires one adaptive logic module (ALM) on the Arria II, and that one ALM is used for the sum and carry of the upper half, we estimate that the two-bit leaf case uses two ALMs. Using this as the base case, the number of logic elements required for larger adders can be calculated recursively as follows:

$$C_2 = 2$$

$$C_4 = 3C_2 + 1 = 7$$

$$C_8 = 3C_4 + 2 = 23$$

$$C_{16} = 3C_8 + 4 = 73$$

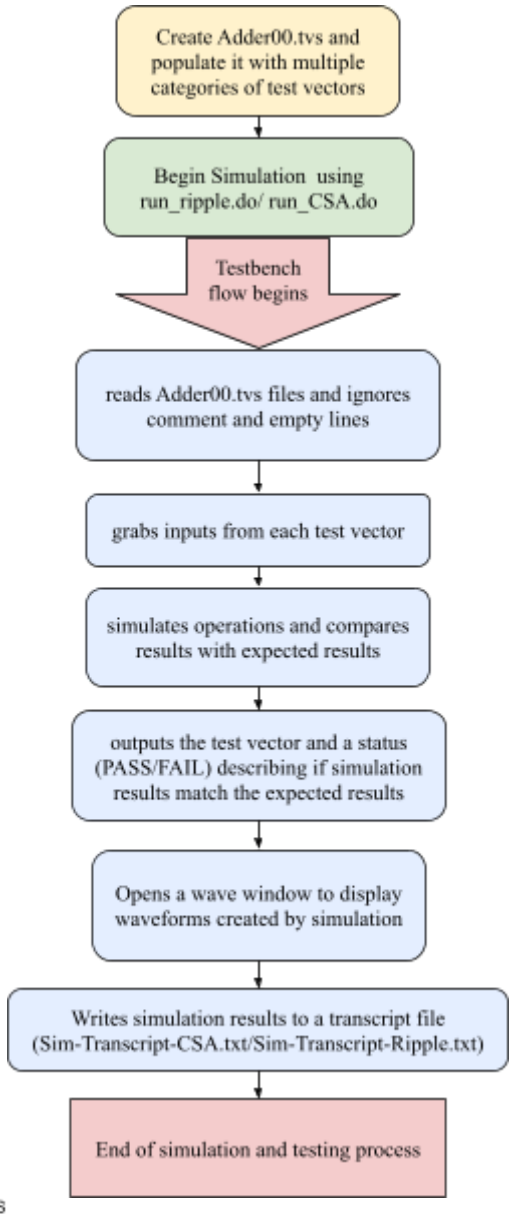
$$C_{32} = 3C_{16} + 8 = 227$$

$$C_{64} = 3C_{32} + 16 = 697$$

An additional logic element is included to account for the overflow computation, resulting in an estimated total of 697 logic elements, with an approximate implementation cost of $24.58 CAD. This represents about 21 times higher resource usage compared to the baseline topology.

5 - Testbenches

A testbench was created for each topology (Ripple-Carry and Conditional-Sum). Each testbench utilized the TEXTIO VHDL package to read test vectors from the file Adder00.tvs, apply the input values, and compare the simulation results to the expected outputs contained within the same file. In Adder00.tvs, each test vector occupies a single line in the following format: A (first operand in hexadecimal), B (second operand in hexadecimal), Cin (input carry as a signal), Sum (expected sum in hexadecimal), Cout (expected output carry as a signal), and Overflow (expected overflow as a signal). The file includes a combination of base cases, carry-out (unsigned overflow) cases, and two's complement (signed overflow) cases, along with several randomly generated test vectors produced by a Python script. These groups of test vectors are separated by comments, which the testbench was programmed to ignore. Sample output from the Ripple-Carry Adder testbench is shown below in Figure 5; the output from the Conditional-Sum Adder testbench followed a similar format.




```
# ** Note: Using test vectors from file: TestVectors/Adder00.tvs
#    Time: 0 ps  Iteration: 0  Instance: /tb_adder_rip
# 0 A=0000000000000000 B=0000000000000000 Cin=0  | S=0000000000000000 Cout=0 Ovfl=0  status=PASS
# 1 A=0000000000000001 B=0000000000000000 Cin=0  | S=0000000000000001 Cout=0 Ovfl=0  status=PASS
# 2 A=0000000000000001 B=0000000000000001 Cin=0  | S=0000000000000002 Cout=0 Ovfl=0  status=PASS
# 3 A=0000000000000001 B=0000000000000001 Cin=1  | S=0000000000000003 Cout=0 Ovfl=0  status=PASS
# 4 A=FFFFFFFFFFFFFFFF B=0000000000000000 Cin=0  | S=FFFFFFFFFFFFFFFF Cout=0 Ovfl=0  status=PASS
```

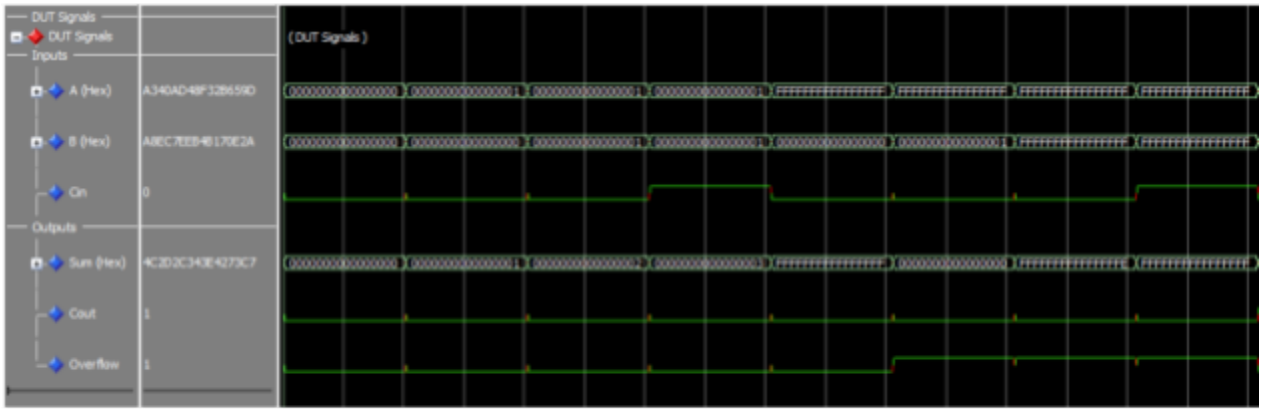


Figure 8: Sample outputs from Ripple Adder Testbench

6 - Cost Analysis

Theoretical Cost Predictions

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

Each Cyclone IV FPGA logic element contains a four-input look-up table (LUT). Since both the sum and carry outputs of a full adder can be derived from their respective truth tables, it is estimated that each full adder requires two logic elements on this device. Based on this assumption, the 64 full adders connected in series are expected to utilize approximately 128 logic elements in total. An additional logic element is included to account for the overflow computation, resulting in an estimated total of 129 logic elements and an approximate implementation cost of $1.16 CAD.

Design Candidate 2: Ripple Adder on an ARRIA II FPGA

A single Adaptive Logic Module (ALM) on the Arria II FPGA can accommodate up to eight inputs and provide up to two outputs. Under ideal conditions, this makes each ALM approximately equivalent to two Logic Elements (LEs) on the Cyclone IV. Based on this estimation, the design would require around 64 ALMs to implement the same topology. An additional ALM is included to account for the overflow computation, resulting in an estimated total of 65 ALMs and an approximate implementation cost of $2.29 CAD. This is about 2 times higher than the baseline Cyclone IV device, reflecting the greater cost of ALMs relative to LEs.

Design Candidate 3: CSA on a Cyclone IV FPGA

Under the assumption that each full adder requires two logic elements on the Cyclone IV, and that one logic element is used for the sum of the upper half and another for the carry of the upper half, we estimate that the two-bit leaf case uses four logic elements. Using this as the base case, the number of logic elements required for larger adders can be calculated recursively as follows:

$$C_2 = 4$$

$$C_4 = 3C_2 + 2(2) = 16$$

$$C_8 = 3C_4 + 2(4) = 56$$

$$C_{16} = 3C_8 + 2(8) = 184$$

$$C_{32} = 3C_{16} + 2(16) = 584$$

$$C_{64} = 3C_{32} + 2(32) = 1816$$

An additional logic element is included to account for the overflow computation, resulting in an estimated total of 1,816 logic elements, with an approximate implementation cost of $16.33 CAD. This represents about 14 times higher resource usage compared to the baseline topology.

Design Candidate 4: CSA on an ARRIA II FPGA

Under the assumption that each full adder requires one adaptive logic module (ALM) on the Arria II, and that one ALM is used for the sum and carry of the upper half, we estimate that the two-bit leaf case uses two ALMs. Using this as the base case, the number of logic elements required for larger adders can be calculated recursively as follows:

$$C_2 = 2$$

$$C_4 = 3C_2 + 1 = 7$$

$$C_8 = 3C_4 + 2 = 23$$

$$C_{16} = 3C_8 + 4 = 73$$

$$C_{32} = 3C_{16} + 8 = 227$$

$$C_{64} = 3C_{32} + 16 = 697$$

An additional logic element is included to account for the overflow computation, resulting in an estimated total of 698 ALMs, with an approximate implementation cost of $24.58 CAD. This represents about 21 times higher resource usage compared to the baseline candidate.

Experimental Cost Calculations

The costs for the design candidates were calculated by the formula

Cost = (Resources Used) / (Total Available Resources) ✕ (Device Price).

This method of calculating the costs factors in the total resources available to the board and the fraction of the board used by the circuit. Resource data was obtained from the Quartus synthesis reports. The normalized cost is obtained from the ratio of the candidate cost to the baseline cost. The limited resources on the Arria EP2AGX45DF29C6 boards lead to higher costs because a larger portion of the board is in use, despite the board's reduced resource usage.

Design Candidate	**Target Device**	**Total Device LEs/ALMs**	**Device Price (CAD)**	**Resources Used (LEs/ALMs)**	**Cost per LE/ALM (CAD)**	**Candidate Cost (CAD)**	**Normalized Cost**
1 (Baseline)	Cyclone IV (EP4CE115F29C7)	114480	1029.5647 [1]	161	0.008993403	$1.45	1
2 (Ripple)	ARRIA II (EP2AGX45DF29C6)	36100	1273.0389 [2]	147	0.035264235	$5.18	3.58015555
3 (CSA)	Cyclone IV (EP4CE115F29C7)	114480	1029.5647 [1]	271	0.008993403	$2.44	1.68322981
4 (CSA)	ARRIA II (EP2AGX45DF29C6)	36100	1273.0389 [2]	188	0.035264235	$6.63	4.57870234

Figure 9: Cost Calculations of Design Candidates

Theoretical VS Experimental Results

The predicted cost varied significantly from the experimental cost, especially for the conditional-sum adders (candidates 3 and 4). While the ripple adder predictions were reasonably close, the theoretical model overestimated the CSA resource usage by over 80%. This discrepancy is attributed to the optimizations performed by the Quartus synthesis tool. Quartus synthesis has the ability to simplify Boolean expressions to a great degree, especially for recursive circuits. This eliminated the need for many LEs and ALMs, leading to a large cost discrepancy.

Design Candidate	Target Device	Resource Usage Predications (LEs/ALMs)	Resources Used (LEs/ALMs)	Predicted Cost (CAD)	Candidate Cost (CAD)	Predicted Normalized Cost	Normalized Cost
1 (Baseline)	Cyclone IV (EP4CE115F29C7)	129	161	$1.16	$1.45	1	1
2 (Ripple)	ARRIA II (EP2AGX45DF29C6)	81	147	$2.29	$5.18	1.9760	3.58015555
3 (CSA)	Cyclone IV (EP4CE115F29C7)	1395	271	$16.33	$2.44	14.0793	1.68322981
4 (CSA)	ARRIA II (EP2AGX45DF29C6)	698	188	$24.58	$6.63	21.1889	4.57870234

Figure 10: Cost Calculations of Design Candidates

7 - Results Analysis and Conclusion

The synthesis results clearly demonstrate the performance-cost trade-off between the two FPGA architectures. The Cyclone IV device consistently delivered a more cost-effective implementation for both adder topologies. This is shown by its lower normalized costs (1.0 and 1.68). In contrast, the Arria II platform, while using fewer ALMs than the Cyclone IV, used LEs for the CSA, resulting in a significantly higher cost (normalized costs of 3.58 and 4.58). This confirms that the higher-performing Arria II is a more expensive technology platform overall. Furthermore, the data validates the expected area to performance trade-off between the two adder topologies. The ripple-carry adder, as predicted, is the most area-efficient design. The conditional-sum adder consumed more resources on both FPGA families, with a nearly 70% increase in LEs on the Cyclone IV and a close to 30% increase in ALMs on the Arria II. This increased resource usage is the direct cost paid for the increased speed of the CSA architecture.

This project successfully compared the implementation efficiency of two 64-bit adder topologies across different FPGAs. The key conclusion is that the Cyclone IV offers a superior cost-to-performance ratio. Furthermore, the selection of an adder topology depends on the requirements of the customer: The ripple-carry adder is optimal for cost-efficient, low-speed systems, while the conditional-sum adder is optimal for performance-focused applications. Ultimately, this analysis highlights the importance of considering both the logical design and the target's physical architecture to make informed engineering decisions in digital systems design.

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].