Group #:
G 12

Design Project 2

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 ten design candidates across five distinct adder circuit topologies, each implemented on two different FPGA devices. Specifically, each topology was synthesized on a Cyclone IV E and an Arria II FPGA (models EP4CE115F29C7 and EP2AGX45DF29C6, respectively) to enable timing simulations, as Quartus provides detailed timing and configuration data for these devices. The five adder topologies were designed based on the principles of hierarchical carry-select, carry-propagation, conditional carry-propagation, parallel carry-generation, and parallel-prefix computation. The VHDL entity, which serves as the foundation for all topologies, is provided in Appendix A.3.

2 - Experimental Procedures

In this experiment, five 64-bit adder topologies, the ripple-carry adder (RCA), conditional-sum adder (CSA), look-ahead carry tree adder (LACTA), Brent-Kung adder (BKA), and carry-bypass adder (CBA), were implemented in VHDL and synthesized using Quartus for the two FPGA devices. Each design was analyzed, fitted, and compiled to evaluate hardware synthesis behaviour and verify proper implementation.

Following synthesis, a configuration VHDL file and several **.do** scripts were created, and a comprehensive testbench was developed for both functional and timing simulations. The testbench read test vectors from a **.tvs** file containing predefined inputs and expected outputs, which were applied to the adders to confirm correct operation under various input conditions. The overall experimental flow is illustrated in the figure below.

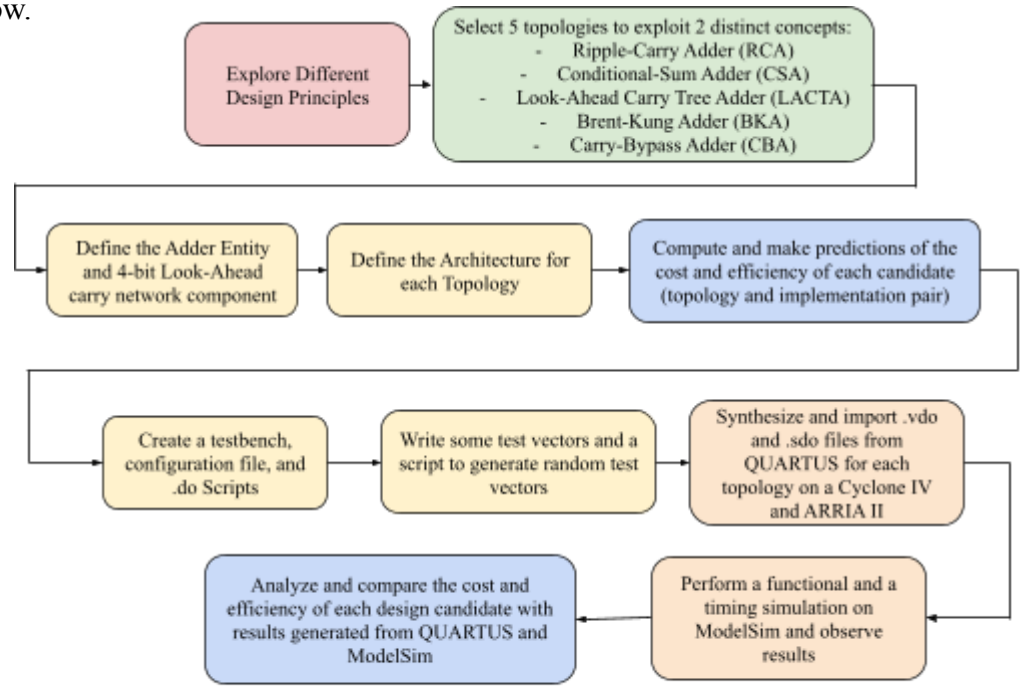


Figure 1: Flow of experimental procedure

3 - Design Candidates

Design Topologies

Ripple-Carry Adder (RCA)

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.

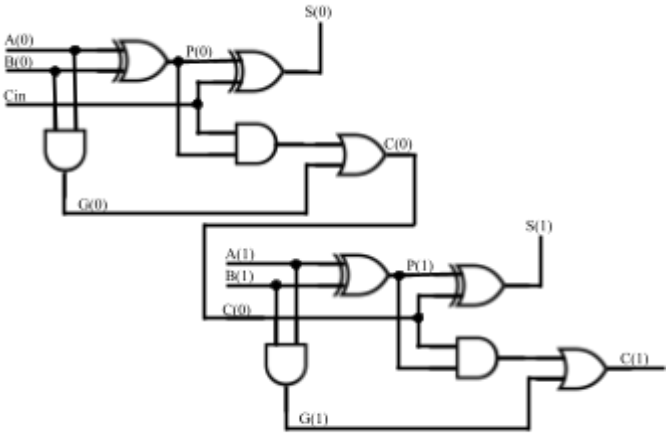


Figure 2: Ripple Adder (case N=2)

Conditional-Sum Adder (CSA)

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.

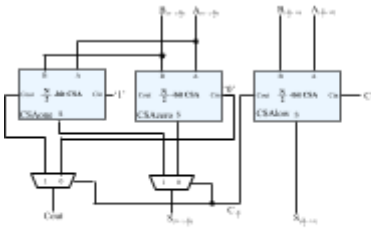


Figure 3: Generalized leaf case for the conditional sum adder

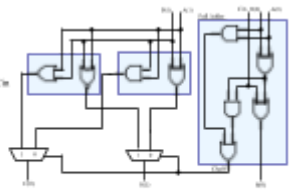


Figure 4: Leaf case (N =2) for the conditional sum adder (more detailed description)

Look-Ahead Carry Tree Adder (LACTA)

The Look Ahead Carry Tree adder utilizes a hierarchical parallel carry generation design principle by using a 4-bit look-ahead carry network component in an inverted tree structure. This architecture leverages the fast carry generation provided by the full-look-ahead principle and minimizes the rippling effect through its hierarchical structure.

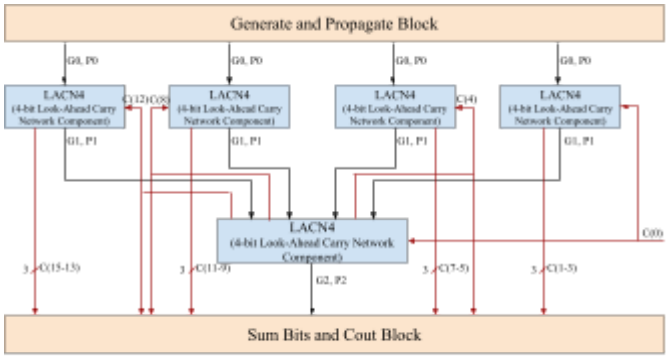


Figure 5: Look-Ahead Carry Tree Adder (case N=16)

Brent-Kung Adder (BKA)

The Brent-Kung adder uses the parallel prefix design principle to minimize computations by performing them in parallel. It arranges, propagates, and generates signals in a recursive pattern that reduces fan-out and limits wiring complexity. This structure computes carries in logarithmic time while using fewer nodes than other prefix adders. As a result, it provides an efficient trade-off between speed and hardware cost, making it well-suited for wide and high-performance arithmetic units.

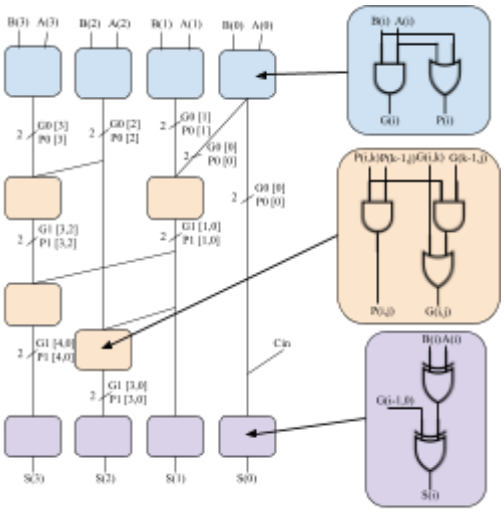


Figure 6: Brent-Kung Adder (case N=4)

Carry-Bypass Adder (CBA)

The Carry-Bypass adder utilizes the look-ahead carry network to produce fast internal carries. The group propagates a signal from each block to a multiplexer (mux) that determines whether the block can be bypassed or not. This bypass mechanism shortens the critical path whenever all bits in a block propagate. This design is beneficial for low-cost implementations that aim to enhance the performance of a ripple adder in most use cases.

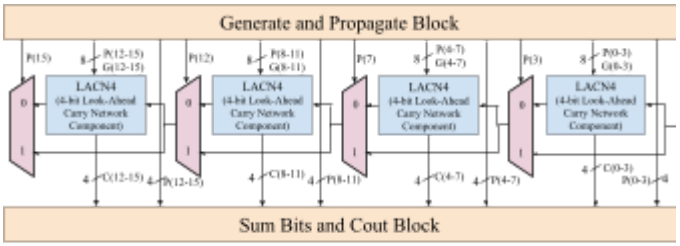


Figure 7: Carry-Bypass Adder (case N=16)

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 is mapped and optimized differently onto FPGA families with different primitive elements.

Candidate Estimations

Cost

The estimated cost for the RCA topology was 130 LEs on the Cyclone IV E and 66 ALMs on the Arria II, calculated using the following formula:

$$Total\ LE/ALMs\ = (LE/ALMs\ for\ N\ =\ 2\ Addition)\ *\ N/2\ +\ Ovfl\ +\ Cout$$

For the CSA topology, the estimated costs were 960 LEs on the Cyclone IV E and 667 ALMs on the Arria II. The calculation for the CSA on the Arria II, based on the N = 4 case, is shown below:

$$C_4 = 5$$
$$C_8 = 3C_4 + 2(2) = 19$$
$$C_{16} = 3C_8 + 4(2) = 65$$
$$C_{32} = 3C_{16} + 8(2) = 211$$
$$C_{64} = 3C_{32} + 16(2) = 665$$
$$C_{64} + (Ovfl\ \&\ Cout) = 665 + 2 = 667$$

The cost estimations for the LACTA and BKA topologies were performed in a similar manner. The LACTA topology yielded 909 LEs on the Cyclone IV E and 443 ALMs on the Arria II, while the BKA topology yielded 578 LEs on the Cyclone IV E and 322 ALMs on the Arria II.

Finally, the estimated cost for the CBA topology was 368 LEs on the Cyclone IV E and 240 ALMs on the Arria II, computed using the following formula:

$$Total\ LE/ALMs\ = (LE/ALMs\ for\ LACN4)\ *\ N/4\ +\ (LE/ALMs\ needed\ for\ G\ \&\ P\ \&\ Sum)\ *\ N\ +\ (LE/ALMs\ for\ muxs)\ *\ N$$

Timing

The worst-case timing delay of the RCA was found to be 66 levels of LEs/ALMs, 64 for the carry network, 1 for the generate-and-propagate block, and 1 for the sum-generation block.

The worst-case timing delays of the CSA, LACTA, and BKA were calculated recursively. The results were 9 levels of LEs/ALMs for the CSA on both devices, 13 levels of LEs for the LACTA on the Cyclone IV E and 8 levels of ALMs on the Arria II, and 15 levels of LEs/ALMs for the BKA.

A sample calculation for the CSA on the Cyclone IV E is shown below:

$$T_4 = 2$$
$$T_8 = T_4 + 2 = 4$$
$$T_{16} = T_8 + 2 = 6$$
$$T_{32} = T_{16} + 2 = 8$$
$$T_{64} = T_{32} + 2 = 10$$
$$C_{64} + (propagate\ +\ generate\ +\ sum\ blocks) = 10 + 3 = 13$$

The worst-case timing delay of the CBA was estimated to be 50 levels of LEs on the Cyclone IV E and 34 levels of ALMs on the Arria II, using the following formula:

$$Max\ Delay\ = (Delay\ of\ LACN4)\ *\ N/4\ +\ Delay\ of\ G\ \&\ P\ \&\ Sum\ blocks$$

Sample Post Fitting Analysis

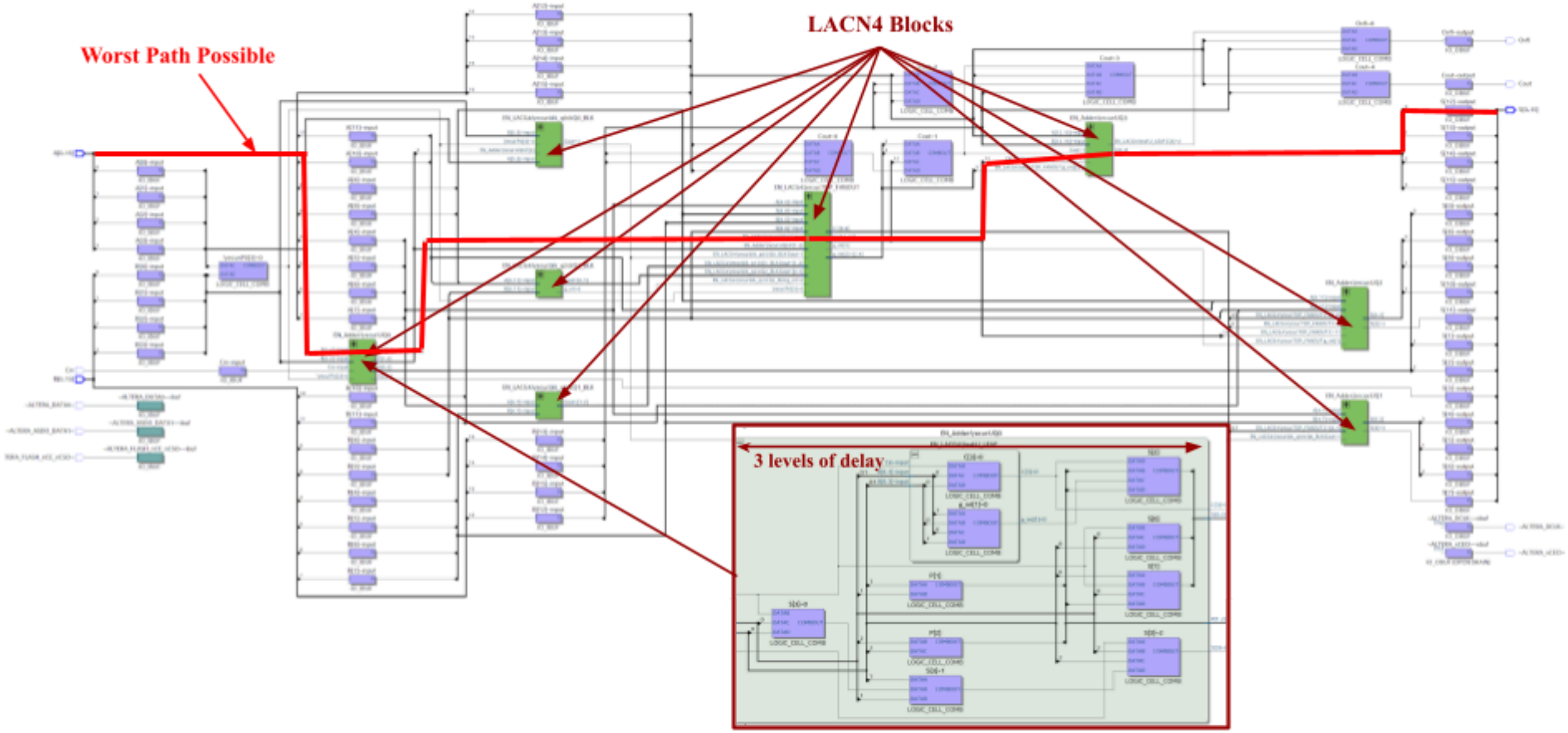
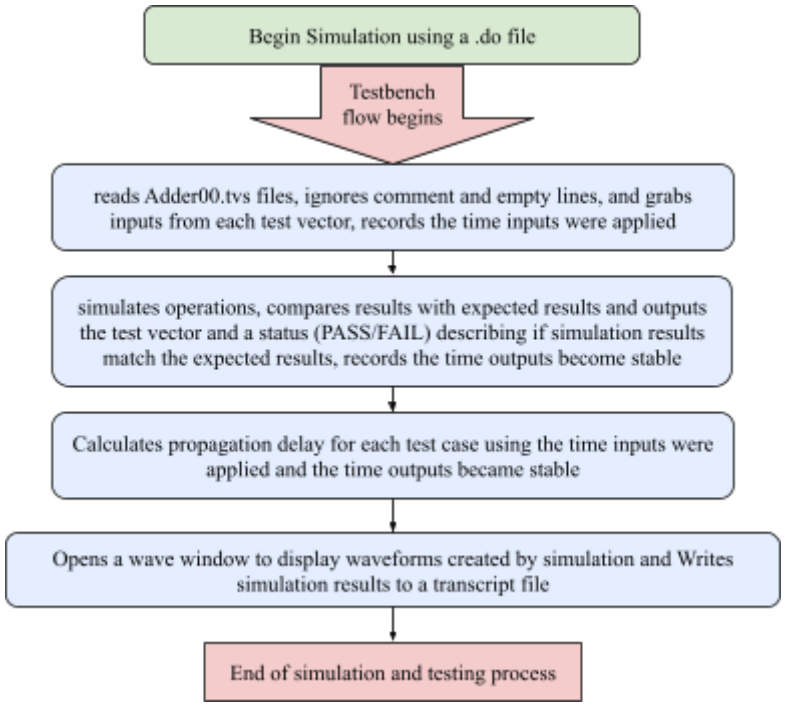


Figure 8: LACTA architecture post-mapping on an Cyclone IV E FPGA for N = 16

From the post fitting of the LACTA for 16 bits, we can see that each Look-Ahead Carry Generator has 10 Logic Elements and 3 levels of delay in the worst case. Following the worst possible path, the bits must propagate through three Look-Ahead Carry Networks, as well as the Propagate, Generate, and Sum blocks.

4 - Testbenches



In this experiment, a singular test bench was used to verify functional correctness and perform timing simulations. The testbench utilizes 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, worst case propagation delay cases, and several randomly generated test vectors. 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. A configuration file was created to allow for easy specification of the architecture. A .do file script was created for each design candidate to run the necessary commands to configure and execute the required tests. The Figure below shows an example of a timing simulation of the carry-bypass adder on the ARRIA II.

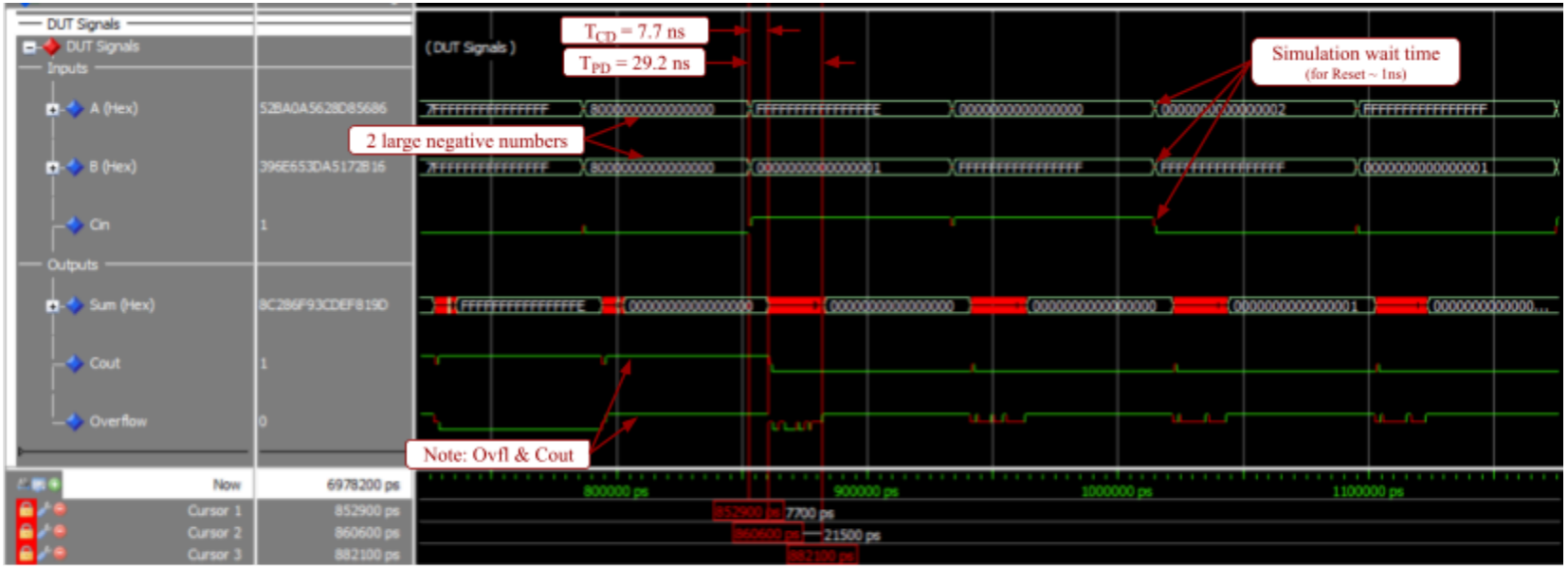


Figure 9: Sample outputs Waveform from the timing simulation of the CBA topology on an ARRIA II device

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 takes into account the total resources available to the board and the fraction of those resources utilized 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 [1][2]. This inflates the price and makes cost calculations inaccurate. To account for these inaccuracies, the Cyclone IV Nano [3] and the ARRIA 10 GX 480 [4] were considered. These devices are comparable to those used in synthesis, however, they have more accurate pricing because they are currently listed 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 Na	22320	$169.49 [3]	$0.00759184 [3]	1
Arria 10 GX 480	181790	$3937.25 [4]	$0.02165823 [4]	2.85

The synthesis results in Figure 9 show clear cost differences between the two FPGA platforms. The Cyclone IV remained the more cost-effective device, with normalized costs of 1.00 for the baseline ripple-carry adder and 1.55 for the carry-bypass adder, compared to the Arria II's much higher values of 2.60 and 3.89 for the same designs. As expected, the ripple-carry adder used the least area on both devices (161 LEs on Cyclone IV), while more advanced parallel architectures showed higher area demands. For example, the conditional-sum adder occupied 271 LEs on Cyclone IV, and the Brent-Kung adder was even more expensive, reaching a normalized cost of 3.57.

These results reinforce the area penalty of parallel-prefix and conditional architectures, which replicate logic to accelerate computations. For area or cost-sensitive designs, the ripple-carry and carry-bypass adders on Cyclone IV offer the best value. The least cost-effective design candidates were the LACTA and BKA on the ARRIA II. Though the predicted logic element usage was inaccurate for these, the trend of using more area for faster computations was expected.

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 (RCA)	ARRIA II	66	147	$1.43	$3.18	419	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	536	3.33
5 (LACTA)	Cyclone IV	909	296	$6.90	$2.25	296	1.84
6 (LACTA)	ARRIA II	443	231	$9.59	$5.00	659	4.09
7 (BKA)	Cyclone IV	578	574	$4.39	$4.36	574	3.57
8 (BKA)	ARRIA II	322	543	$6.97	$11.76	1548	9.61
9 (CBA)	Cyclone IV	368	249	$2.79	$1.89	249	1.55
10 (CBA)	ARRIA II	240	220	$5.20	$4.76	627	3.89

Figure 10: Cost Calculations of Design Candidates

The conversion factor from LEs to ALMs was 2.85, slightly more than double.

6 - Timing Analysis

For timing simulation and analysis, the .vho and .sdo files created by Quartus for each design candidate were used, along with ModelSim's timing capabilities, to simulate and calculate the propagation delays.

From the device datasheets, the worst-case timing delays of a single LE/ALM were determined based on the slowest elements available on each device for timing estimation purposes.

For the Cyclone IV E, the worst-case timing delay was found to be 9.94 ns:

- 4.146 ns input delay from pin to internal cells
- 4.374 ns input delay from pin to input register
- 1.420 ns delay from the output register to the output pin

$$Total\ Delay\ (LE) = 4.146 + 4.374 + 1.420 = 9.94\ ns$$

For the Arria II, the worst-case timing delay was 7.65 ns:

- 0.873 ns output enable pin delay
- 0.722 ns delay from output register to output pin
- 2.944 ns input delay from pin to internal cell
- 2.944 ns input delay from pin to input register

$$Total\ Delay\ (ALM) = 0.873 + 0.722 + 2.944 + 2.944 = 7.65\ ns$$

Design Candidate	**Target Device**	**Levels of Delay**	**Estimated Max Propagation Delay (ns)**	**Max Propagation Delay (ns)**	**Performance** (W.R.T. Baseline Simulation)
1 (Baseline)	Cyclone IV	66	656.04	137.4	1
2 (RCA)	ARRIA II	66	504.9	117.9	1.275
3 (CSA)	Cyclone IV	9	89.46	96.3	1.834
4 (CSA)	ARRIA II	9	68.85	79.6	2.773
5 (LACTA)	Cyclone IV	13	129.22	86.3	2.300
6 (LACTA)	ARRIA II	8	61.2	72.4	3.559
7 (BKA)	Cyclone IV	15	149.1	71.7	3.660
8 (BKA)	ARRIA II	15	114.75	70.5	3.847
9 (CBA)	Cyclone IV	50	497	126.1	1.143
10 (CBA)	ARRIA II	34	260.1	79.7	2.765

Figure 10: Propagation Delay Calculations of Design Candidates

As expected, all candidates demonstrate superior performance compared to the baseline, with the recursive architectures achieving more than two to three times the performance improvement. The Brent–Kung Adder (BKA), when implemented on the ARRIA II, delivers a nearly 290% increase in performance. Likewise, implementations on the ARRIA II device consistently outperform those on the Cyclone IV, enhancing performance by approximately 5% to 150% depending on the topology.

7 - Conclusion

Design Candidate	**Target Device**	**Normalized Cost** (W.R.T. Baseline LEs)	**Performance** (W.R.T. Baseline Simulation)	**Performance to Cost Ratio**
1 (Baseline)	Cyclone IV	1	1	1
2 (RCA)	ARRIA II	2.60	1.275	0.490
3 (CSA)	Cyclone IV	1.68	1.834	1.092
4 (CSA)	ARRIA II	3.33	2.773	0.833
5 (LACTA)	Cyclone IV	1.84	2.300	1.250
6 (LACTA)	ARRIA II	4.09	3.559	0.870
7 (BKA)	Cyclone IV	3.57	3.660	1.025
8 (BKA)	ARRIA II	9.61	3.847	0.400
9 (CBA)	Cyclone IV	1.55	1.143	0.737
10 (CBA)	ARRIA II	3.89	2.765	0.711

Figure 11: Cost and Performance Comparison of Design Candidates

The Cyclone IV will always be the cheaper option for any adder architecture. The ARRIA II, although more expensive, consistently delivers better performance across all topologies. For a purely cost-optimized solution, the baseline candidate is the best option. For a strictly performance-optimized solution, the best option is the BKA on the ARRIA II. This solution achieves an impressive performance, nearly 290% faster than the baseline, but has a significant cost increase of 861%. The option with the best performance-to-cost ratio is the LACTA on the Cyclone IV, with a performance ratio of 1.25. This design candidate achieved a 130% performance increase, but incurred only an 84% increase in cost.

These findings underscore that advanced adder architectures provide diminishing returns as complexity increases, with parallel-prefix designs, such as the Brent-Kung design, offering marginal performance gains at substantial cost penalties. The hierarchical inverted tree structure of the LACTA strikes an effective balance for practical implementations where both performance and cost constraints must be considered.

8 - Appendix

A.1 Directory Structure

(folder) Documentation

- (files) ProjectLog-G12-xxxx-350-1257.pdf
  - Each group member's activity log worksheet (xxxx corresponds to the last 4 digits of the group member's student ID)
- (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
  - (files) FS-xxx-Transcript.text
    - Transcripts from the functional simulation of each topology (xxx corresponds to the abbreviation of each topology)
  - (files) TS-CYC-xxx-Transcript.text
    - Transcripts from the timing simulation of each topology on the Cyclone IV device (xxx corresponds to the abbreviation of each topology)
  - (files) TS-ARRIA-xxx-Transcript.text
    - Transcripts from the timing simulation of each topology on the ARRIA II device (xxx corresponds to the abbreviation of each topology)
- (file) VHDLSrc_EN_Adder.pdf
  - A PDF listing of the VHDL code of the Adder Entity and Architectures
- (file) VHDLSrc_EN_LACN4.pdf
  - A PDF listing of the VHDL code of the 4 bit Look-Ahead Carry network entity and its architecture
- (file) VHDLSrc_TB_Adder.pdf
  - A PDF listing of the VHDL code of the testbench
- (file) VHDLSrc_TB_Config_Adder.pdf
  - A PDF listing of the VHDL code of the different configurations needed for running the simulations

- (file) DP2-Report-G12-350-1257.pdf
    - Project Report, including cost and efficiency analysis (this document)
- (file) DP2-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) ModelSim/ Questa
    - Includes the .vho and .sdo files for each candidate
- (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) DP2.cr.mti / DP2.mpf
    - Modelsim project files for simulation and testing
- (file) wave.do
    - Script to set up the waveforms for a simulation
- (files) FS_xxx.do
    - Scripts to compile, add waves, run simulation and create a simulation transcript for the function simulation of each topology (xxx corresponds to the abbreviation of each topology)
- (files) TS_CSA_xxx.do / TS_ARRIA_xxx.do
    - Scripts to compile, add waves, run simulation and create a simulation transcript for the timing simulation of each candidate (xxx corresponds to the abbreviation of each topology)
- (file) FS_run.do
    - Script to compile, add waves, run simulation and create a simulation transcript for the functional simulation of every topology
- (file) TS_run.do
    - Script to compile, add waves, run simulation and create a simulation transcript for the timing simulation of every candidate
- (file) runall.do
    - Script to compile, add waves, run simulation and create a simulation transcript for all of the simulations
- (file) TB_Adder.vhd

- General testbench written in VHDL

➢ (file) Config_Adder.vhd

- Contains all the configurations needed for running the simulations written in VHDL

(folder) SourceCode

➢ (file) EN_Adder.vhd

- Defines the adder entity as well as a ripple-carry adder, fast ripple adder, and conditional sum adder, look-ahead carry tree adder, Brent-Kung adder, and carry-bypass adder architectures in VHDL

(file) DP2.qpf / DP2.qsf

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

A.2 References

[1]
"EP4CE115F29C7 Altera | Integrated Circuits (ICs) | DigiKey." Accessed: Oct. 12, 2025. [Online]. Available: https://www.digikey.ca/en/products/detail/altera/EP4CE115F29C7/2260452

[2]
"EP2AGX45DF29C6 Altera | Integrated Circuits (ICs) | DigiKey." Accessed: Oct. 12, 2025. [Online]. Available: https://www.digikey.ca/en/products/detail/altera/EP2AGX45DF29C6/2349475

[3]
"P0082 Terasic Inc. | Development Boards, Kits, Programmers | DigiKey." Accessed: Oct. 25, 2025. [Online]. Available: https://www.digikey.ca/en/products/detail/terasic-inc/P0082/2625112

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

[5]
"Cyclone IV Device Handbook, Volume 3." Dec. 2016.

[6]
"Device Interfaces and Integration for the Arria® II Device Handbook." Accessed: Nov. 15, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/content-details/654011/device-interfaces-and-integration-for-the-arria-ii-device-handbook.html

A.3 VHDL Entities

Adder Entity Definition within EN_Adder.vhd

```
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;
```

Look-Ahead Carry Generator Block Entity Definition within EN_LACN4.vhd

```
entity EN_LACG4 is
    generic (N: natural := 4);
    port (
        Gin, Pin  : in std_logic_vector (N-1 downto 0);
        Gout, Pout  : out std_logic;
        Cin : in std_logic;
        C : out std_logic_vector (N-1 downto 1)
    );
end EN_LACG4;
```