# ECE441/543 Lab 1 Report:

# Design and Performance of Adder Logic

1st Liu CheukKi
V00937822
Dept. of Electrical of Computer Engineering
University of Victoria
cheukkiliu@uvic.ca

2nd Hou dengyao
V00973596
Dept. of Electrical and Computer Engineering
University of Victoria
dengyaohou@uvic.ca

***Abstract*—**

In this lab, students use the VHDL source files and Constraint files to design and perform adder logic.

***Keywords*—**

*Hierarchical Design of a Full Adder (FA), Ripple Carry Adder, (RCA), Carry Look-ahead (CLA) addition, Parallel Prefix Adder (Kogge-Stone Adder (KSA)), Look-up Tables (LUT), synthesis schematic, RTL schematic*

## I. Introduction

In this experiment, we demonstrate and evaluate several adder structures by using the Digilent NEXYS A7 FPGA board and Vivado. And understand their fundamental design and operation and how adder performance can be improved with a variety of advanced logical structures.

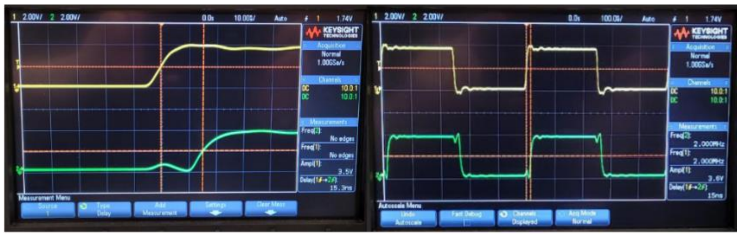
## II. In-lab steps

### A. Step 1 - Hierarchical Design of a Full Adder (FA)

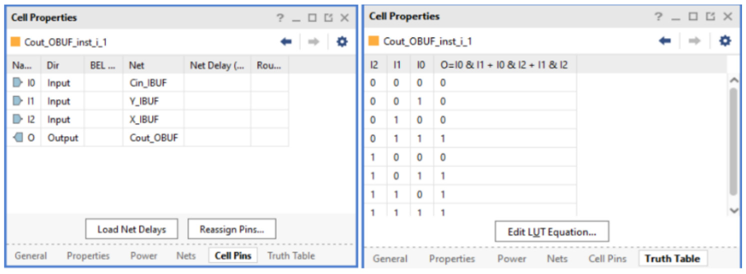
Include truth table and logic equation for COUT_OBUF_inst_i_1 LUT and verify the logic it implements.

Comment on COUT_OBUF_inst_i_1 LUT 's logic.

The logic equation for COUT_OBUF_inst_i_1 LUT is
Cout = XY + (X (xor) Y)Cin
Assume X is 1, Y is 0, and Cin is 1.
Cout equals 1
When X is 0, Y is 0 and Cin is 1
Cout equals 0
Which means the Truth table is correct.



From the oscilloscope, the delay between Cin and Cout is 15.3ns



Cell Properties — Cout_OBUF_inst_i_1 (Cell Pins)

| Na... | Dir | BEL ... | Net | Net Delay (... | Rou... |
|---|---|---|---|---|---|
| I0 | Input | | Cin_IBUF | | |
| I1 | Input | | Y_IBUF | | |
| I2 | Input | | X_IBUF | | |
| O | Output | | Cout_OBUF | | |

Cell Properties — Cout_OBUF_inst_i_1 (Truth Table)

| I2 | I1 | I0 | O=I0 & I1 + I0 & I2 + I1 & I2 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Fig. 1. COUT_OBUF_inst_i_1 LUT

### B. Step 2 – 4-bit Ripple Carry Adder (RCA)

Complete the following table as you test your circuit

TABLE I.

| $A_3$-$A_0$ | $B_3$–$B_0$ | $C_{in}$ | $S_3$-$S_0$ | $C_{out}$ |
|---|---|---|---|---|
| 0000 | 0101 | 0 | 0101 | 0 |
| 0000 | 0101 | 1 | 0110 | 0 |
| 0111 | 1000 | 0 | 1111 | 0 |
| 0111 | 1000 | 1 | 0000 | 1 |
| 1111 | 1111 | 0 | 1110 | 1 |

Verify 1-bit RCA truth tables (lab manual, page 21)

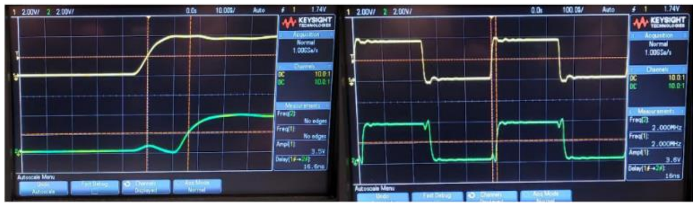
1 bit RCA truth table is the same as the Full adder of Cout.

And the truth table of the sum is X (XOR) Y (XOR) Cin.

Therefore the 1-bit RCA truth table for both sum and Cout is true.

Verify 2-bit RCA truth tables and logic equations (lab manual, page 24)

The truth table of 2-bit RCA truth tables and logic equations is shown in step 3.

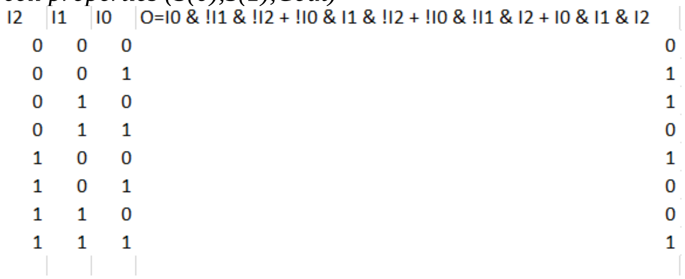


From the oscilloscope, the delay between Cin and Cout is 16.6ns

*C. Step 3 – 2-bit Ripple Carry Adder (RCA)*

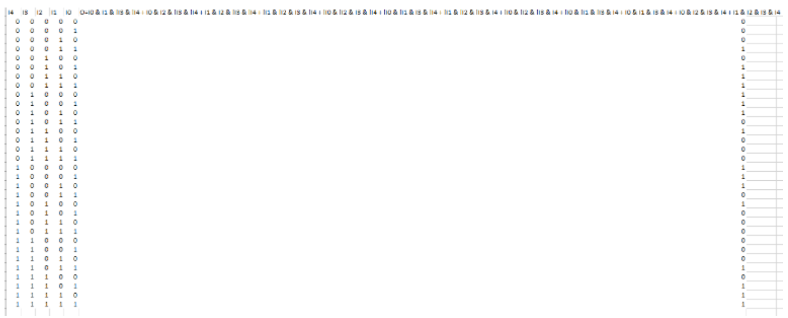
Capture a screen shot of the truth table (and the corresponding logic equations) for each LUT in this circuit for inclusion in your lab report. You will need these for proving the correctness of the circuit using truth tables in your lab report.

*cell properties (S(0),S(1),Cout)*



| I2 | I1 | I0 | O=I0 & !I1 & !I2 + !I0 & I1 & !I2 + !I0 & !I1 & I2 + I0 & I1 & I2 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

*2-bit RCA S0_LUT*



| I4 | I3 | I2 | I1 | I0 | O=I0 & I1 & !I3 & !I4 + I0 & I2 & !I3 & !I4 + I1 & I2 & !I3 & !I4 + !I1 & !I2 & I3 & !I4 + !I0 & !I2 & I3 & !I4 + !I0 & !I1 & I3 & !I4 + !I1 & !I2 & !I3 & I4 + !I0 & !I2 & !I3 & I4 + !I0 & !I1 & !I3 & I4 + I0 & I1 & I3 & I4 + I0 & I2 & I3 & I4 + I1 & I2 & I3 & I4 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

*2-bit RCA S1_LUT*

Equations for 2-bit RCA S1_LUT:

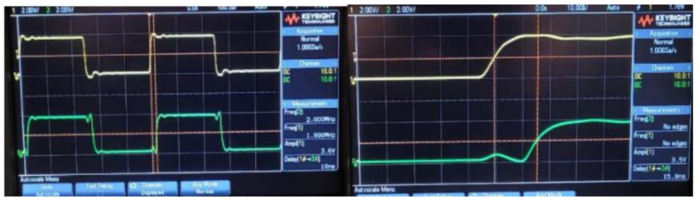
O=I0 & I1 & !I3 & !I4 + I0 & I2 & !I3 & !I4 + I1 & I2 & !I3 & !I4 + !I1 & !I2 & I3 & !I4 + !I0 & !I2 & I3 & !I4 + !I0 & !I1 & I3 & !I4 + !I1 & !I2 & !I3 & I4 + !I0 & !I2 & !I3 & I4 + !I0 & !I1 & !I3 & I4 + I0 & I1 & I3 & I4 + I0 & I2 & I3 & I4 + I1 & I2 & I3 & I4

| I4 | I3 | I2 | I1 | I0 | O=I0 & I1 & I3 + !I1 & I3 & I4 + I1 & I2 & I3 + I0 & I1 & !I3 & I4 + I0 & I2 & I3 + I0 & I2 & !I3 & I4 + I1 & I2 & !I3 & I4 + !I2 & I3 & I4 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

*2-bit RCA Cout_LUT*

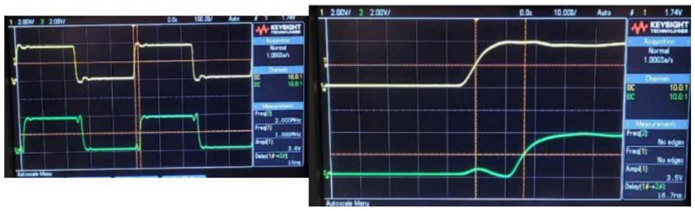
Fig. 2. Screen shots of the truth tables for LUTS(and the corresponding logic equations)

Show that the Vivado® generated truth tables for the 2-bit RCA match the truth table for a 2-bit adder.



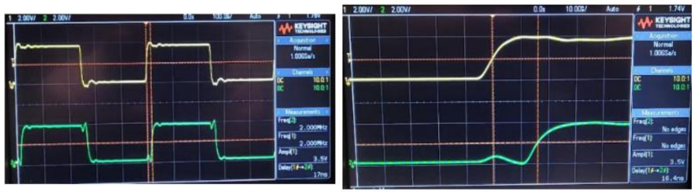
From the oscilloscope, the delay between Cin and Cout is 15.8ns

*D. Step 4 – Carry-Lookahead (CLA) addition*



From the oscilloscope, the delay between Cin and Cout is 16.7ns

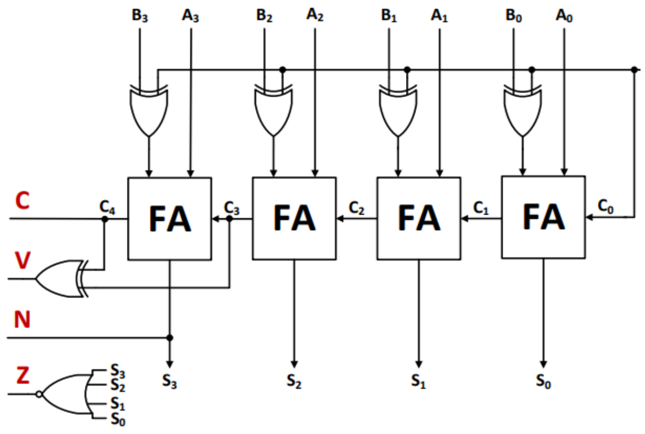
*E. Step 5 – Parallel Prefix Adder (Kogge-Stone Adder)*



From the oscilloscope, the delay between Cin and Cout is 16.4ns

## III. Questions

*A. Show how the 4-bit RCA could be modified to operate as an adder/subtractor (implementation not required).*



From the graph above, when M = 1, it is a subtractor, when M = 0, it is an adder. The reason is the XOR gates (together with M) implement the 2's complement of B.

*B. What is the measured propagation delay from the carry input to the carry output using: 1110 + 0001 with Cin = 0, 1 for each adder structure in Steps 1 – 5 of this lab. Complete the following table using your DSO measurements in the lab and comment on the results.*

TABLE II.

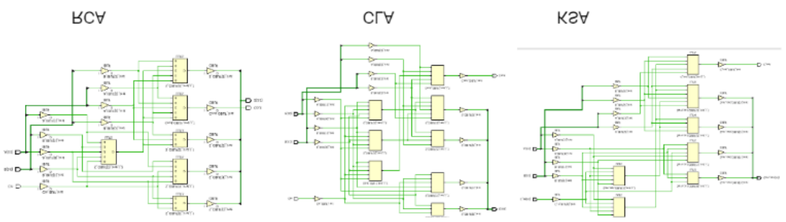
| | *Adder Structure* | *Delay from $C_{in}$ to $C_{out}$* |
|---|---|---|
| Step 1 | Single bit FA | 15.3ns |
| Step 2 | RCA(n=4) | 16.6ns |
| Step 3 | RCA(n=2) | 15.8ns |
| Step 5 | CLA(n=4) | 16.7ns |
| Step 5 | KSA(n=4) | 16.4ns |

From the table above, there are no changes for the time delay with different adders. The reason is the Digilent NEXYS A7 FPGA board and Vivado design suite will optimize all designs and give the fastest performance from the board.

*C. Show and explain the LUT contents programmed in the FPGA for Step 3 and compare to the truth table for the 2-bit RCA given in Step 3.*

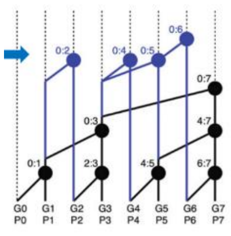
The truth table generated from Vivado is the same as the truth tables for the 2-bit adder.
It looks different from the lab manual because of the representation difference which shows in the cell properties.

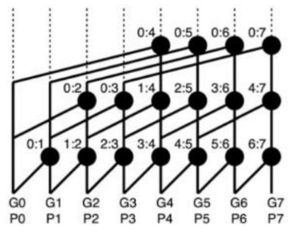
*D. Compare the IMPLEMENTATION schematics generated from Vivado® for each of the 4-bit RCA, CLA, and KSA structures. Explain.*



From the implementation schematic diagram, CLA needs 8 boxes to form a result, KSA needs 7 boxes to form a result, and RCA needs 6 boxes to form a result. The reason is that CLA needs to combine Group Generate and Group Propagate, which requires more processes to get the result. KSA uses the dot operator to generate the result, while RCA just forms the result and passes the carry to the next full adder. Overall, the number of implementation boxes is based on the adder structure.

*E. Briefly comment on the fundamental advantage of the Brent-Kung adder compared to that of the Kogge-Stone. No implementation is required.*





8-bit BKA

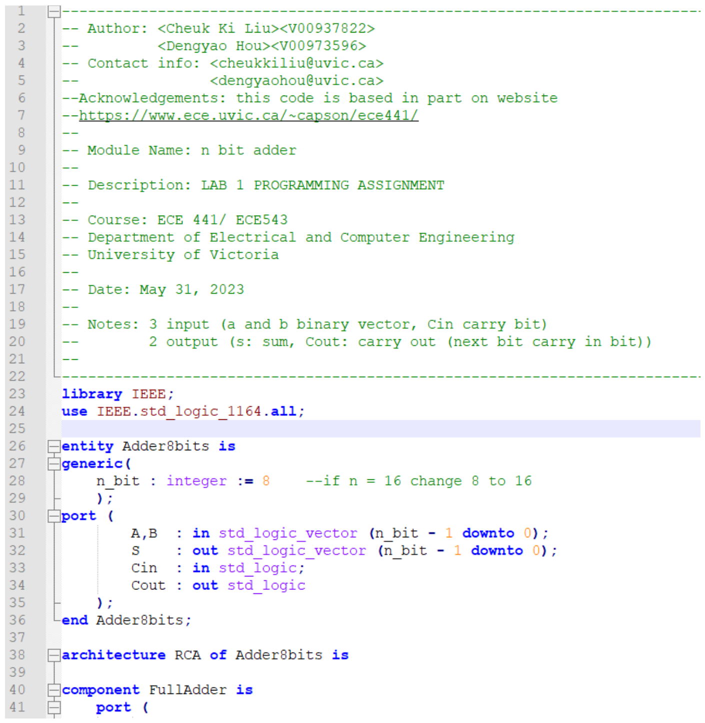
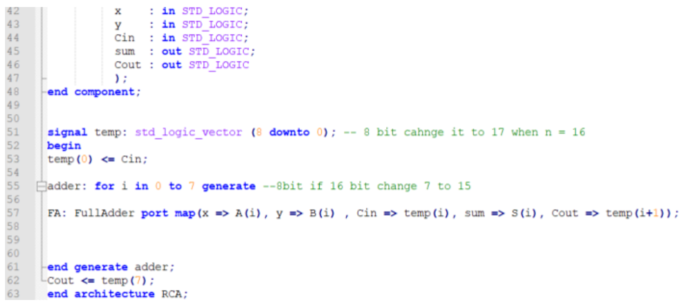
8-bit KSA

The diagram shows that BKA takes fewer modules to implement, so the BKS structure is more simple than KSA.

## IV. Programming Assignment

Generalize the VHDL code given for the 4-bit RCA to build an n-bit RCA adder using the VHDL GENERIC and GENERATE statements. Test your design on the NEXYS A7 FPGA board with switches and LEDS for n=8. Show the RTL and SYNTHESIS schematics for n=8 and n=16. Include your code listing in this Lab Report.

Include comments (including a header section) to receive full marks for this part!

```vhdl
--------------------------------------------------------------------------------
-- Author: <Cheuk Ki Liu><V00937822>
--          <Dengyao Hou><V00973596>
-- Contact info: <cheukkiliu@uvic.ca>
--               <dengyaohou@uvic.ca>
--Acknowledgements: this code is based in part on website
--https://www.ece.uvic.ca/~capson/ece441/
--
-- Module Name: n bit adder
--
-- Description: LAB 1 PROGRAMMING ASSIGNMENT
--
-- Course: ECE 441/ ECE543
-- Department of Electrical and Computer Engineering
-- University of Victoria
--
-- Date: May 31, 2023
--
-- Notes: 3 input (a and b binary vector, Cin carry bit)
--         2 output (s: sum, Cout: carry out (next bit carry in bit))
--
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity Adder8bits is
generic(
    n_bit : integer := 8    --if n = 16 change 8 to 16
    );
port (
        A,B   : in std_logic_vector (n_bit - 1 downto 0);
        S     : out std_logic_vector (n_bit - 1 downto 0);
        Cin   : in std_logic;
        Cout  : out std_logic
    );
end Adder8bits;

architecture RCA of Adder8bits is

component FullAdder is
    port (
            x    : in STD_LOGIC;
            y    : in STD_LOGIC;
            Cin  : in STD_LOGIC;
            sum  : out STD_LOGIC;
            Cout : out STD_LOGIC
            );
end component;


signal temp: std_logic_vector (8 downto 0); -- 8 bit cahnge it to 17 when n = 16
begin
temp(0) <= Cin;

adder: for i in 0 to 7 generate --8bit if 16 bit change 7 to 15

FA: FullAdder port map(x => A(i), y => B(i) , Cin => temp(i), sum => S(i), Cout => temp(i+1));



end generate adder;
Cout <= temp(7);
end architecture RCA;
```

Listing 1: Code listing for n-bit RCA

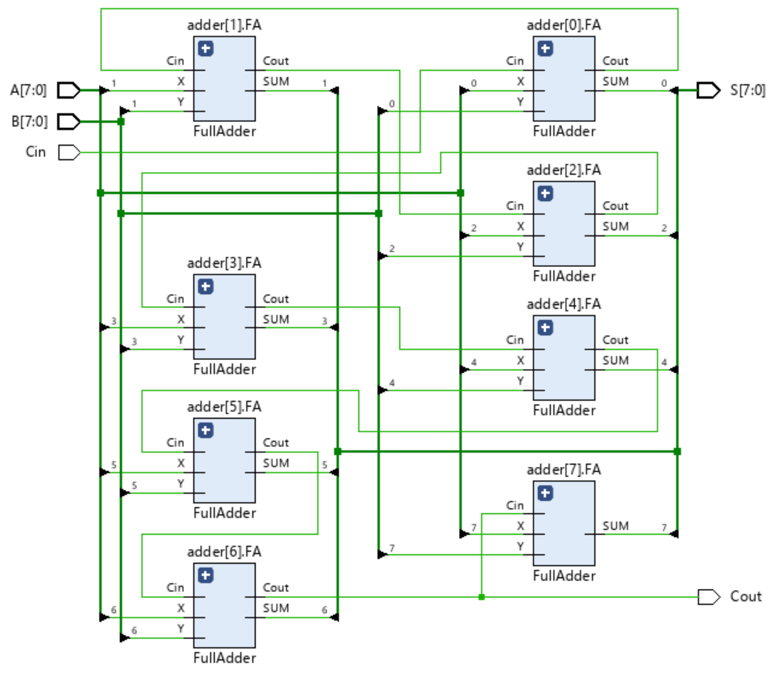


Fig. 3. RTL schematics for n=8

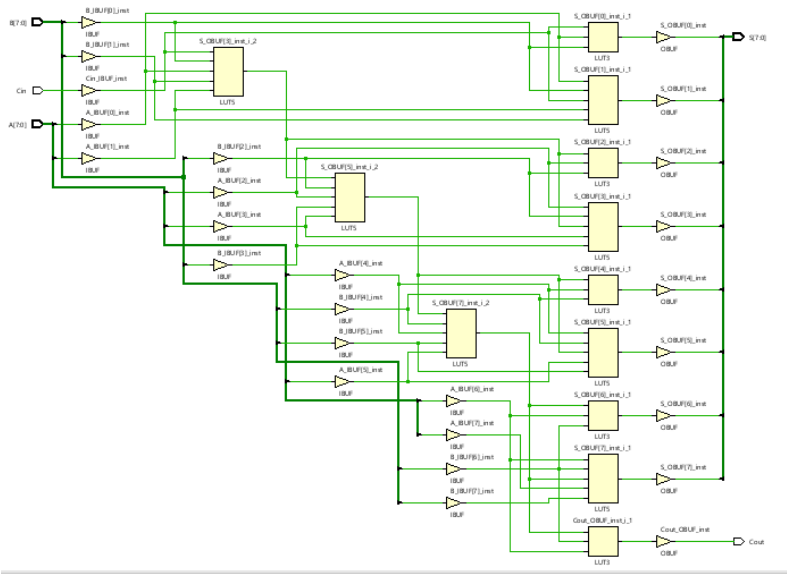


Fig. 4. STYNTHESIS schematics for n=8

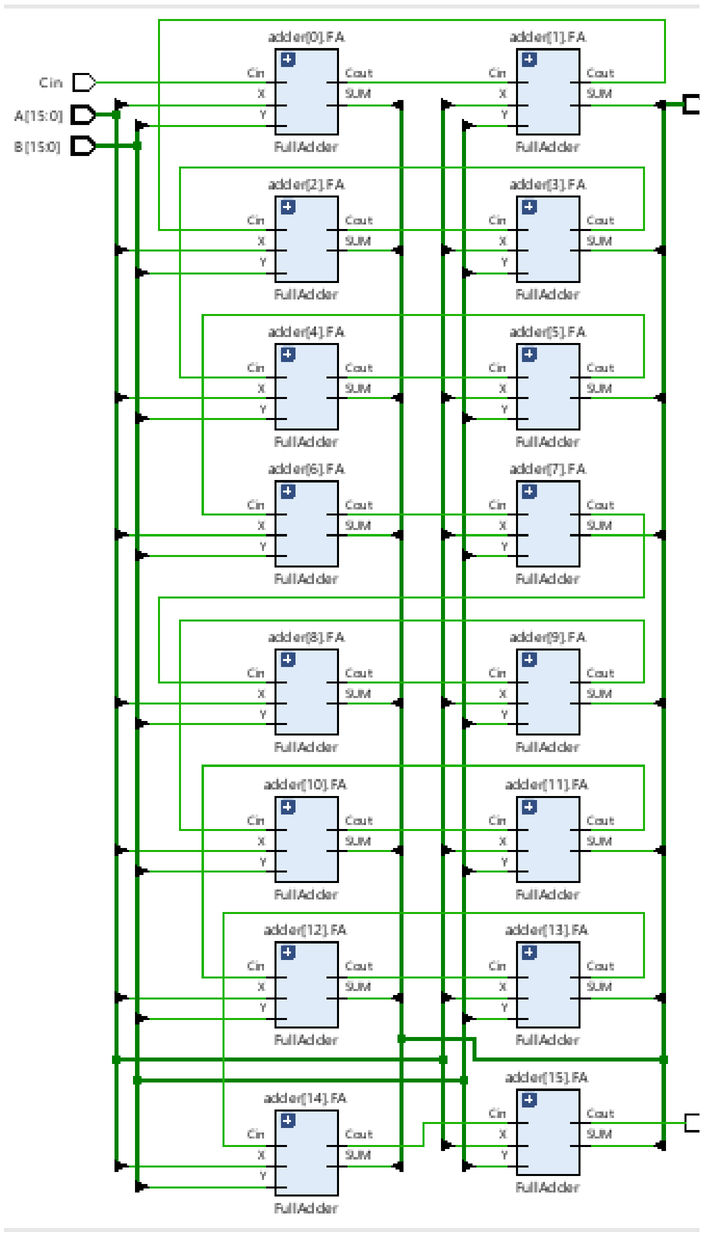


Fig. 5. RTL schematics for n=16

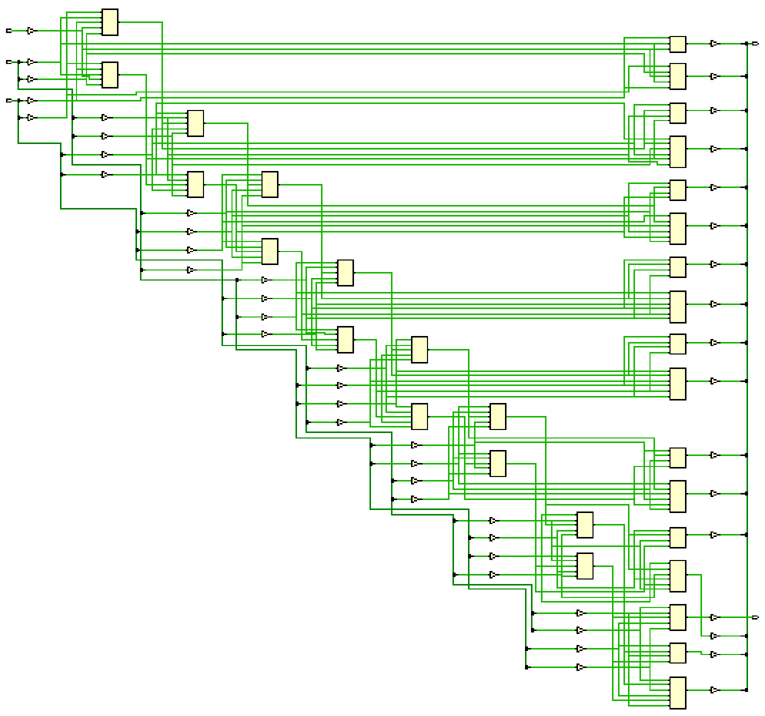


Fig. 6. SYNTHESIS schematics for n=16

## V. Conclusions

Summary of the lab and results.

In lab 1, Ripple Carry Adder, (RCA), Carry Look-ahead (CLA) addition, Parallel Prefix Adder (Kogge-Stone Adder (KSA)) are built successfully by using Vivado with VHDL source files (vhd) and Constraints files (xdc). Although the result generated from different kinds of adders is the same, the structures are different. The delay from Cin to Cout is similar due to optimized on Vivado.

## References

1. *Dr. Capson" Lab#1 Design and Performance of Adder Logic" May 23, 2023*
2. *Dr. Capson "ECE441 / ECE543 Design of Digital and VLSI" Online: [https://www.ece.uvic.ca/~capson/ece441/](https://www.ece.uvic.ca/~capson/ece441/)" 2023*