" بسم الله الرحمن الرحيم "

Faculty of Engineering & Technology

Department of Electrical and Computer Engineering

ENCS3310

Final project report

Name: Razan Abdelrahman.

ID: 1200531.

Section: 1.

Semester: 2021/2022.

Instructor: Dr. Abdullatif Abuissa.

- **Abstract:**

In this project I have been able to implement a 2-Digit BCD adder (8-bits) in two stages. First one will be to build the 2-Digit BCD Adder using Ripple adders, on the second stage I used Carry Look Ahead Adders instead of ripple adders. I get deeper into the difference between concentrated on recognizing the difference between the Bring Look Ahead Adder and the Ripple Adder.

- **Contents:**

- **Brief introduction and overview:**

  ❖ 1-bit full adder.

  A combinational circuit that adds three inputs and generate 2-bits as results, one bit considered as the sum, and the other will be the carry.

  ❖ Four-bits full adder.

  A circuit that consist a four instances from 1-bit full adder, and perform the addition operation on 4-bits.

  ❖ One digit BCD adder.

  A combinational circuit that considered as a special kind of ripple adders.

  But this circuit check if the sum is more than 9, then it adds 6 to the result, if not the sum remain the same.

  ❖ Two digits BCD adder (the whole System).

  This circuit is the whole system, that consist of two 1-digit BCD adder,

  Which mean this circuit adds 8 to 8 bits.

  ❖ Ripple carry adder:

  A ripple carry adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. It is called a ripple carry adder because each carry bit gets rippled into the next stage.

  ❖ Carry look ahead adder:

  Or fast adder is a type of electronics adders used in digital logic. A carry-look ahead adder improves speed by reducing the amount of time required to determine carry bits, by using more complex hardware circuitry.

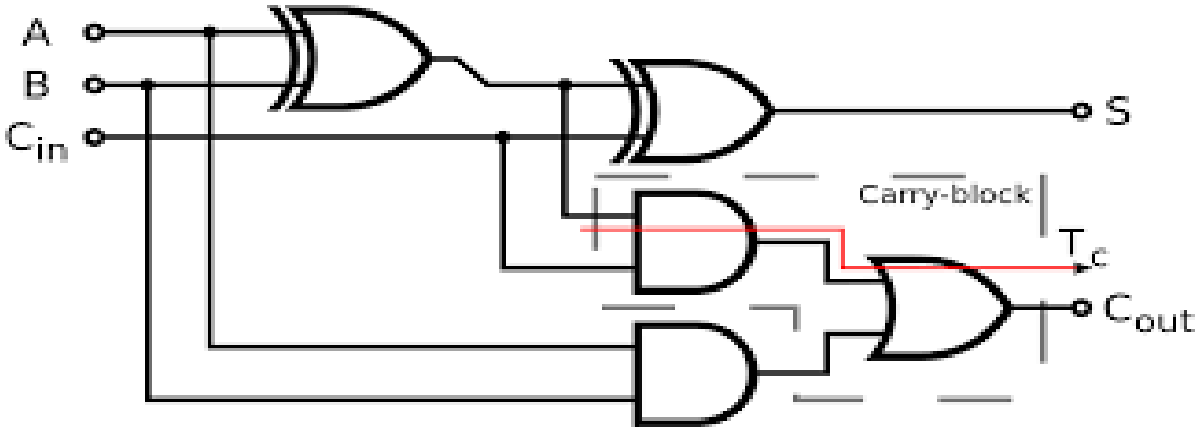


Figure 1 1-bit Full adder

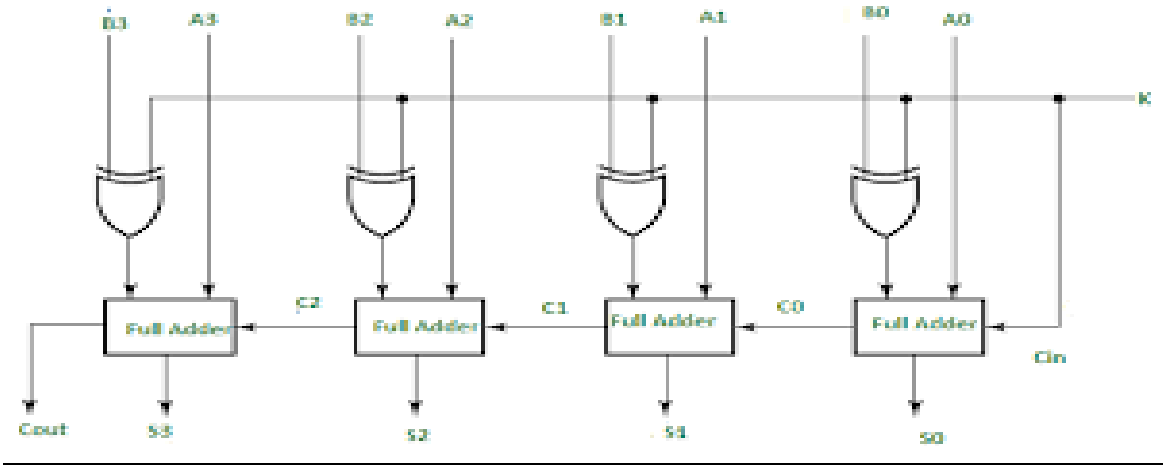


Figure 2 4-bits full adder

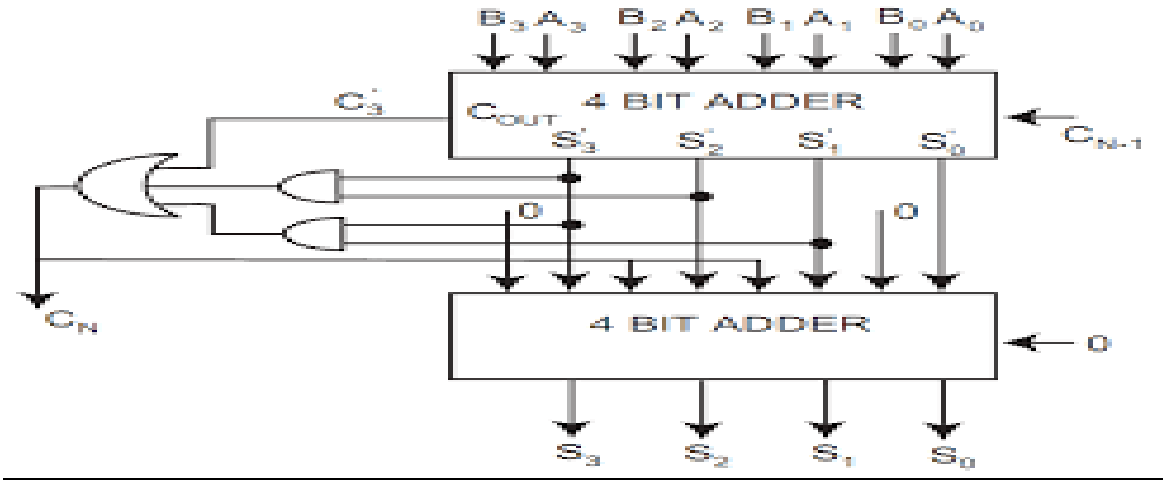


Figure 3 1-digit BCD adder

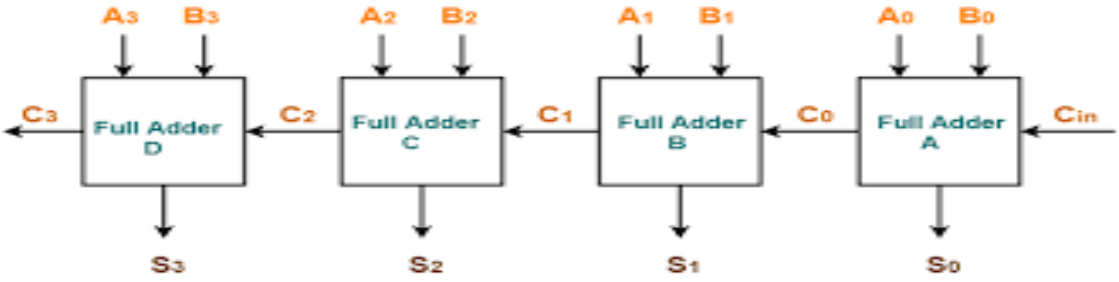


Figure 4 Ripple carry adder

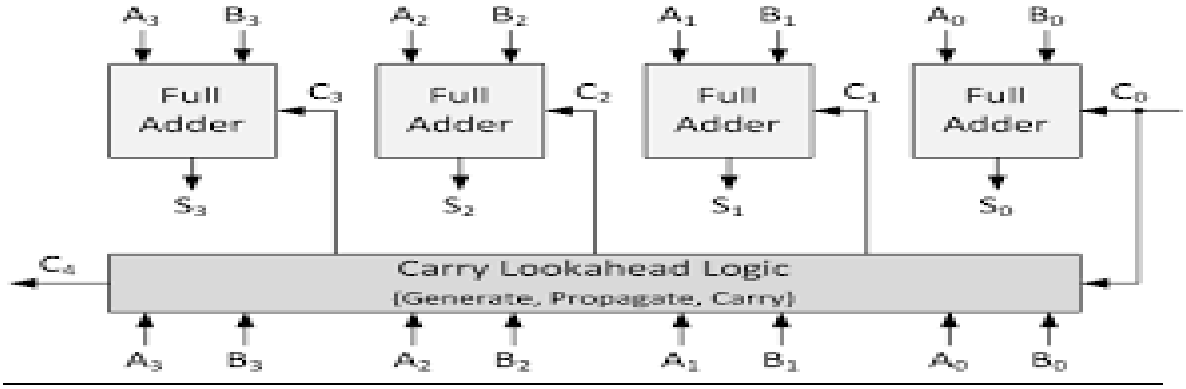


Figure 5 Carry look ahead adde

- **Design philosophy:**

  - Stage 1 (Ripple carry adder):

In the beginning my goal was to create a (2 BCD adder) , so at first I built (2-bit full adder) which consist of (2 XOR gates , 3 AND gates) Which I have used it to create (4-bit full adder) by used generic statement in addition to 5 bit (0-4 bit) signals to carry.

And then I started building the (2-BCD adder) by used (2 of 4-bit adder), As the mechanism for doing this (2-BCD adder) is that if the sum of the greater than 9, he is working to add the 6 to sum.

After that I built (4 bits 2- BCD adder structural) such that the input carry for the first digit carry (output carry (0) and output carry (3) =0) but the second input carry (output carry (1) and output carry (2) =carry from the first digit).

After that I designed the verification entity which consists of (4 bit 2 BCD adders) each one of them working on adding the first two digits, and then examining if their sum is greater than 9 adding to the number 6 and I give carry the value 1 while the sum of the first digits is less than 9, the collection process is done in the usual manner and I give carry the value 0, and then the second digit, is collected and the same mechanism mentioned above is implemented. After that, I called Circuit entity in verification entity, this entity takes different variables, in addition to that I added (4 for loop) the goal of them is to obtain all possible possibilities from the numbers that will be entered in BCD adder and then BCD adder out two variable first variable (8-bits) is sum of adder and second variable (1-bit) is a carry that will be used in Analyzer entity. Now, as I mentioned in the past, I need (Analyzer entity). This Entity takes different variables and this (Entity) is programmed to compare between the expected result and the outcome result. The error will be printed and Specify the location and time in which the error occurred, and it will be printed on the screen.

- Stage2 (Look ahead adder):

In the beginning my goal was to create a (2 carry lock a head adder) , so at first I built (4-bits carry lock a head adder) which consisted of (2 XOR gates ,11 AND gates , 4 OR gates) such that I used AND gate with 2 ,3 and 5 input all of them are the same delay time (8-ns) and I used OR gate with 2,3,4,5 input all of them are the same delay time (8-ns) and XOR gate with 1 and 2 input all of them are the same delay time (12-ns) .

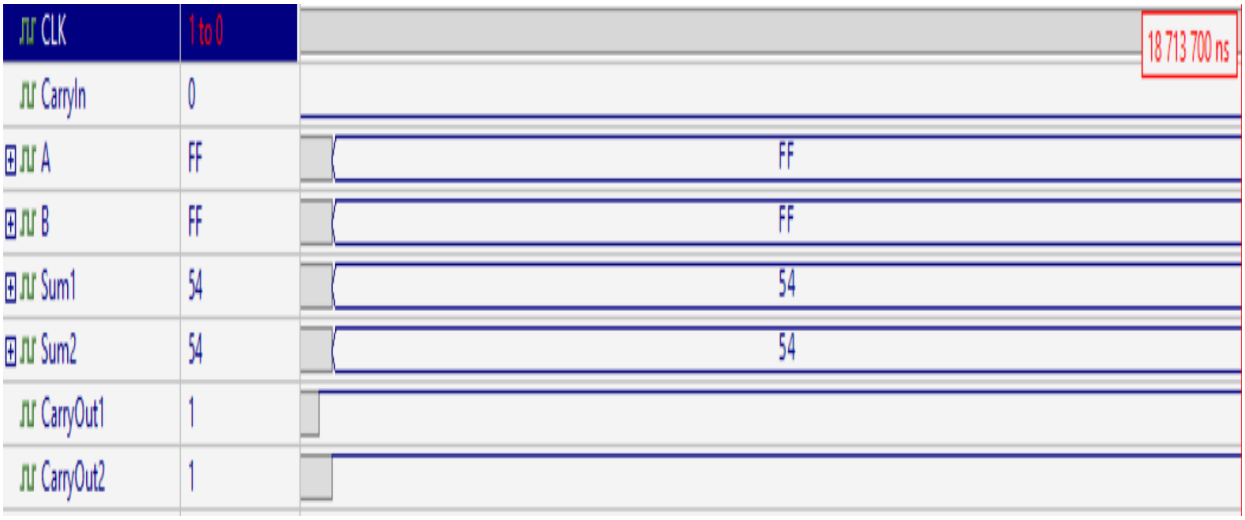
After that I designed one digit carry look ahead BCD adder by using (2 of 4-bits carry look ahead adder) and I want to make it clear that I resorted to this type of adder because it is better than (carry Ripple Adder) in terms of (Delay Time), so we have resorted to using gate have little time delay like (AND , OR gate) ,the adder working on adding the two digits and then examining if their sum is greater than 9 adding to the number 6, the collection process is done in the usual manner.

Now I need to use 2-one digit carry look ahead BCD adder to build (2-digit carry look ahead BCD adder structural) such that the input carry for the first one digit carry look ahead BCD adder =0 but the input carry for the second = the output from the first. As for now, I will use the verification entity that we used in Stage 1, but instead of using carry Ripple Adder I will use carry look ahead BCD adder and do the same previous steps after that I need use the Analyzer entity that I used in the past Stage to check the match between results, if the results aren't match The error will be printed to the screen. The inequality is the result of an unequal Both of the Test Generator and the analyzer where both of them must be connected to the same clock, and in this case we tried more than one value in order to arrive at the final value (CLK = 137) to suit the worst-case delay in 2 Digit Carry Look Ahead BCD Adder .

- **Results:**

These are some screenshots for the simulation shows the results and the difference between the behavioral output and the actual output.

❖ Stage1:



And when the result was not the same between both adders, this was printed to the screen:

- **Conclusion and future works:**

The results that were obtained from the previous procedures agree with the theoretical results. Moreover, I conclude that I can do huge system constructing with smaller ones. I successfully design a 2-digit BCD adder (8 bits), and then complete a code for functional verification .I learned a lot about types of adders, also I noticed the difference between the ripple full adder and the carry look ahead adder in our system. The adder with look ahead carry more faster than the ripple one because the carry of each 1-bit full adder does not depend on the previous carries except the first one, so for example the second full adder of the second bit need to wait until the input carry results from the first full adder, but in the adder with look ahead carry doesn't need to wait, all full adders work in parallel. I became more familiar with Verilog-HDL and how to write commands like how to print an error, make a delay, testing the systems and building entities in behavioral and structural logics, also I used the ALDEC-active HDL to simulate our project and see the signals of the entity that we worked on it.