



| |
|------------------------------|
| Experiment No. 5 |
| Implement ripple carry adder |
| Name: Yash Mayekar |
| Roll Number: 29 |
| Date of Performance: |
| Date of Submission: |

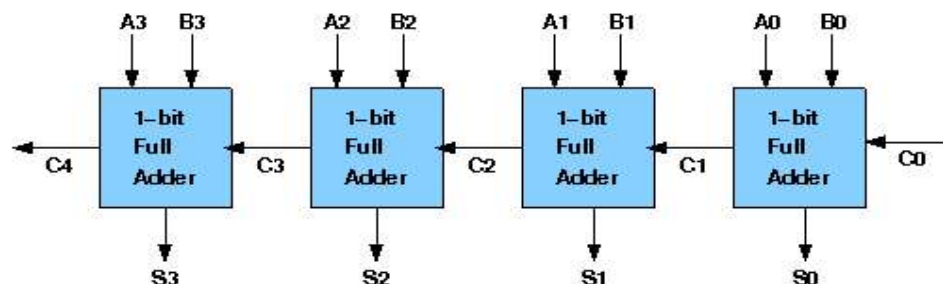
Aim: To implement ripple carry adder.

Objective: To understand the operation of a ripple carry adder, specifically how the carry ripples through the adder.

1. examining the behaviour of the working module to understand how the carry ripples through the adder stages
2. to design a ripple carry adder using full adders to mimic the behaviour of the working module
3. the adder will add two 4 bit numbers

Theory: Arithmetic operations like addition, subtraction, multiplication, division are basic operations to be implemented in digital computers using basic gates like AND, OR, NOR, NAND etc. Among all the arithmetic operations if we can implement addition then it is easy to perform multiplication (by repeated addition), subtraction (by negating one operand) or division (repeated subtraction).

Half Adders can be used to add two one bit binary numbers. It is also possible to create a logical circuit using multiple full adders to add N-bit binary numbers. Each full adder inputs a C_{in} , which is the C_{out} of the previous adder. This kind of adder is a Ripple Carry Adder, since each carries bit "ripples" to the next full adder. The first (and only the first) full adder may be replaced by a half adder. The block diagram of 4-bit Ripple Carry Adder is shown here below -



The layout of ripple carry adder is simple, which allows for fast design time; however, the ripple carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder. The gate delay can easily be calculated by inspection of the full adder circuit. Each full adder requires three levels of logic. In a 32-bit [ripple carry] adder, there are 32 full adders, so the critical path (worst case) delay is $31 * 2$ (for carry)



propagation) + 3(for sum) = 65 gate delays.

Design Issues:

The corresponding Boolean expressions are given here to construct a ripple carry adder. In the half adder circuit the sum and carry bits are defined as

$$\text{sum} = A \oplus B$$

$$\text{carry} = AB$$

In the full adder circuit the the Sum and Carry output is defined by inputs A, B and Carryin as

$$\text{Sum} = ABC + \bar{A}BC + A\bar{B}C + AB\bar{C}$$

$$\text{Carry} = ABC + \bar{A}BC + AB\bar{C} + A\bar{B}C$$

Having these we could design the circuit. But, we first check to see if there are any logically equivalent statements that would lead to a more structured equivalent circuit.

With a little algebraic manipulation, one can see that

$$\text{Sum} = ABC + \bar{A}BC + A\bar{B}C + AB\bar{C}$$

$$= (AB + \bar{A}B)C + (A\bar{B} + AB)\bar{C}$$

$$= (A \oplus B)C + (A \oplus B)\bar{C}$$

$$= A \oplus B \oplus C$$

$$\text{Carry} = ABC + \bar{A}BC + AB\bar{C} + A\bar{B}C$$

$$= AB + (\bar{A} + A)BC$$

$$= AB + (A \oplus B)C$$

Procedure:

Procedure to perform the experiment: Design of Ripple Carry Adders

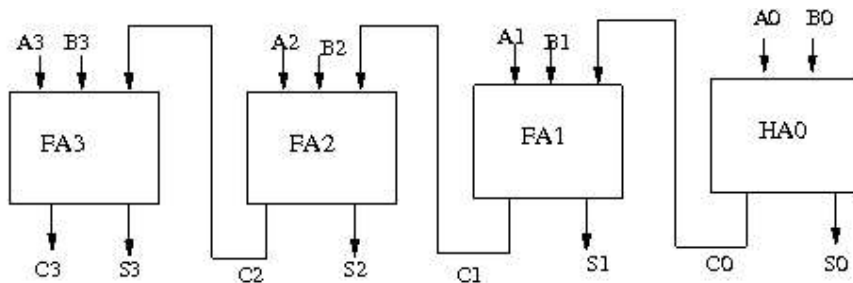
- 1) Start the simulator as directed. This simulator supports 5-valued logic.
- 2) To design the circuit we need 3 full adder, 1 half adder, 8 Bit switch(to give input), 3 Digital display(2 for seeing input and 1 for seeing output sum), 1 Bit display(to see the carry output), and wires.
- 3) The pin configuration of a component is shown whenever the mouse is hovered on any canned component of the palette or presses the 'show pin config' button. Pin numbering starts from 1 and from the bottom left corner (indicating with the circle) and increases anticlockwise.
- 4) For half adder input is in pin-5,8 output sum is in pin-4 and carry is pin-1, For full adder input is in pin-5,6,8 output sum is in pin-4 and carry is pin-1
- 5) Click on the half adder component(in the Adder drawer in the pallet) and then click on the position of the editor window where you want to add the component(no drag and drop, simple click will serve the purpose), likewise add 3 full adders(from the Adder drawer in the pallet), 8 Bit switches, 3 digital display and 1 bit Displays(from Display and Input drawer of the pallet, if it is not seen scroll down in the drawer)
- 6) To connect any two components select the Connection menu of Palette, and then click on the Source terminal and click on the target terminal. According to the circuit diagram, connect all the components, connect 4 bit switches to the 4 terminals of a digital display and another set of 4 bit switches to the 4 terminals of another digital



display. connect the pin-1 of the full adder which will give the final carry output. connect the sum(pin-4) of all the adders to the terminals of the third digital display(according to the circuit diagram shown in screenshot). After the connection is over click the selection tool in the pallet.

- 7) To see the circuit working, click on the Selection tool in the pallet then give input by double clicking on the bit switch, (let it be 0011(3) and 0111(7)) you will see the output on the output(10) digital display as sum and 0 as carry in bit display.

Circuit diagram of Ripple Carry Adder:



Components required:

The components needed to create 4 bit ripple carry adder is listed here -

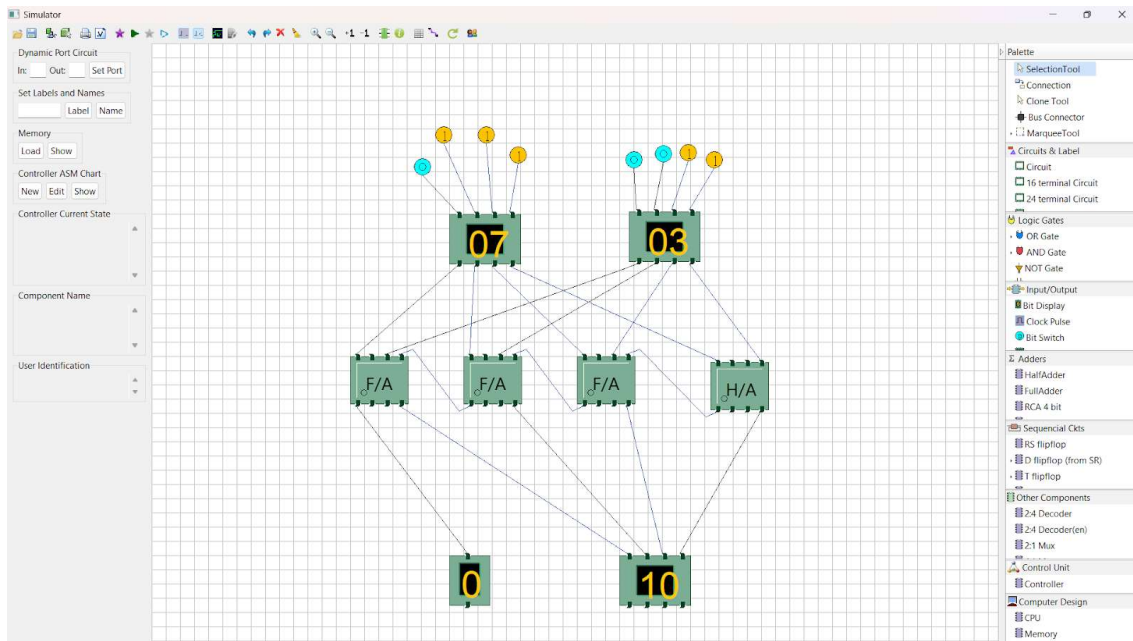
- 4 full-adders
- wires to connect
- LED display to obtain the output

OR we can use

- 3 full-adders
- 1 half adder
- wires to connect
- LED display to obtain the output



Screenshots of Ripple Carry Adder:



Conclusion:

In conclusion, Experiment 5 provided a comprehensive understanding of ripple carry adders, a crucial component in digital arithmetic operations. By simulating the behaviour of a ripple carry adder using full adders and half adders, we gained insight into how carry bits ripple through the adder stages, enabling the addition of two 4-bit binary numbers. This experiment emphasised the importance of efficient circuit design, even though the ripple carry adder has limitations due to carry propagation delays. By analysing the gate delay and logical expressions involved in building the adder, we acquired valuable insights into the practical implementation of complex arithmetic operations in digital computers. This knowledge is foundational for more advanced digital logic and computer architecture concepts.