**COA LAB PROJECT**

****

**Submitted by:**

**Ujjawal Mandhani 9918103237**

**Submitted to:**

**Mrs. Rashmi Kushwaha**

**Title Of The Project**

carry lookup adder, carry select and carry save adder circuits

**Problem Statement**

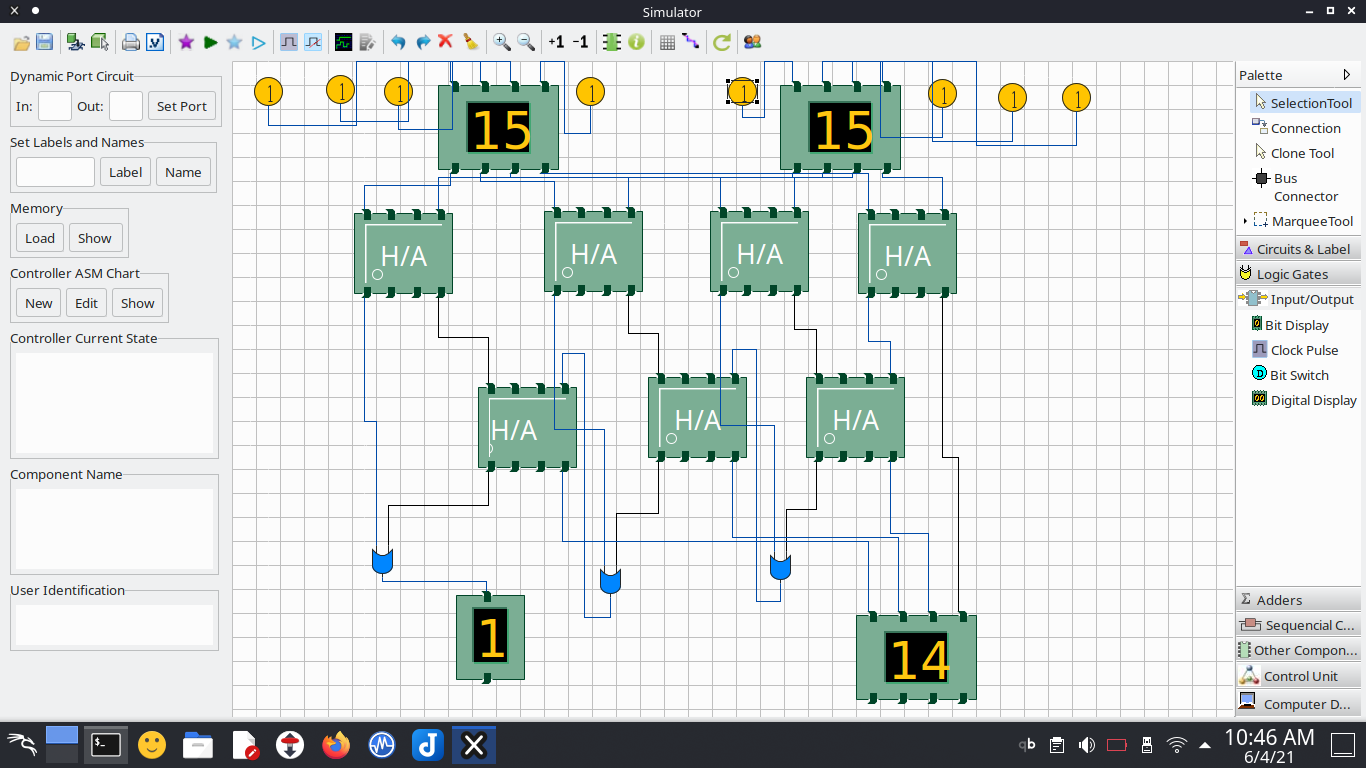
Design the carry lookup adder, carry select and carry save adder circuits by modifying the ripple carry adder logic and analyze their performances.

**Requirement Specification**

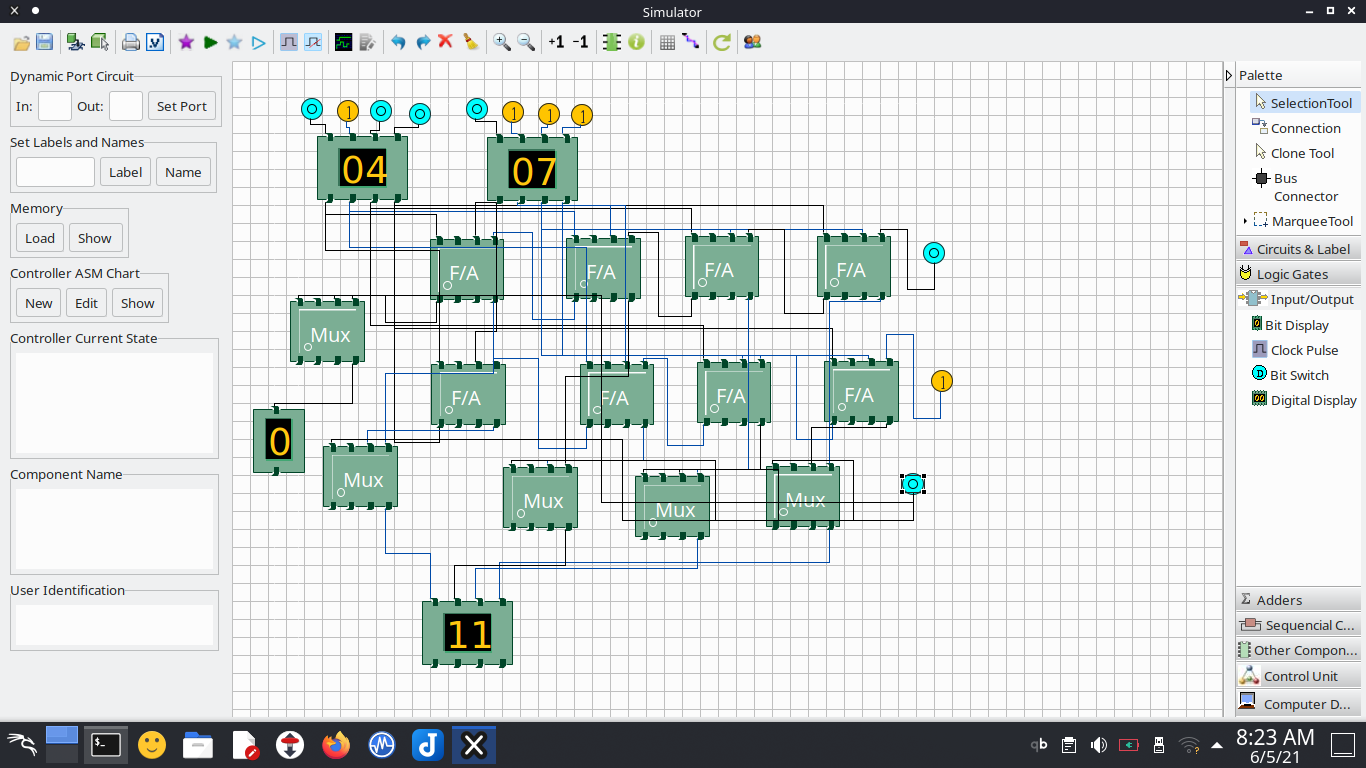
Vlab Simulator

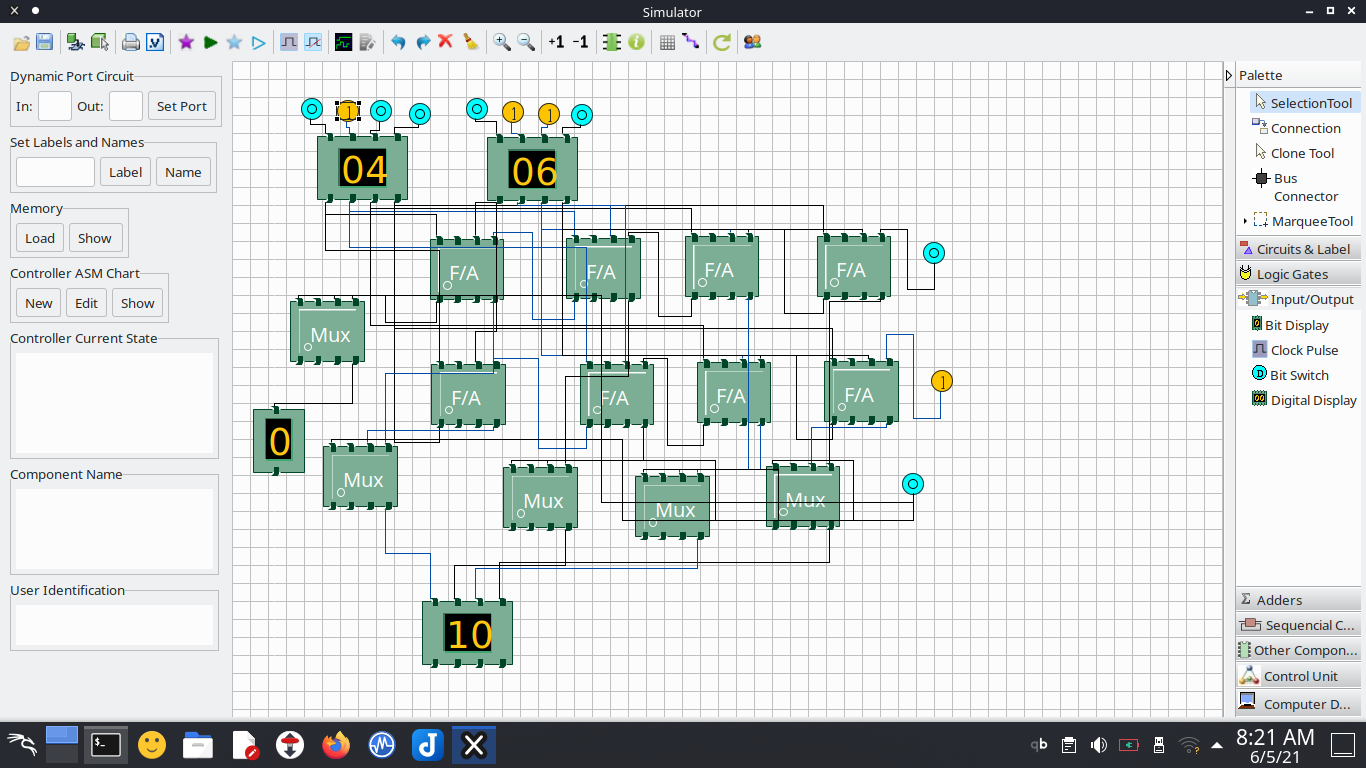
**Design Prototype/ Implementation Code**

**Carry Lookup**

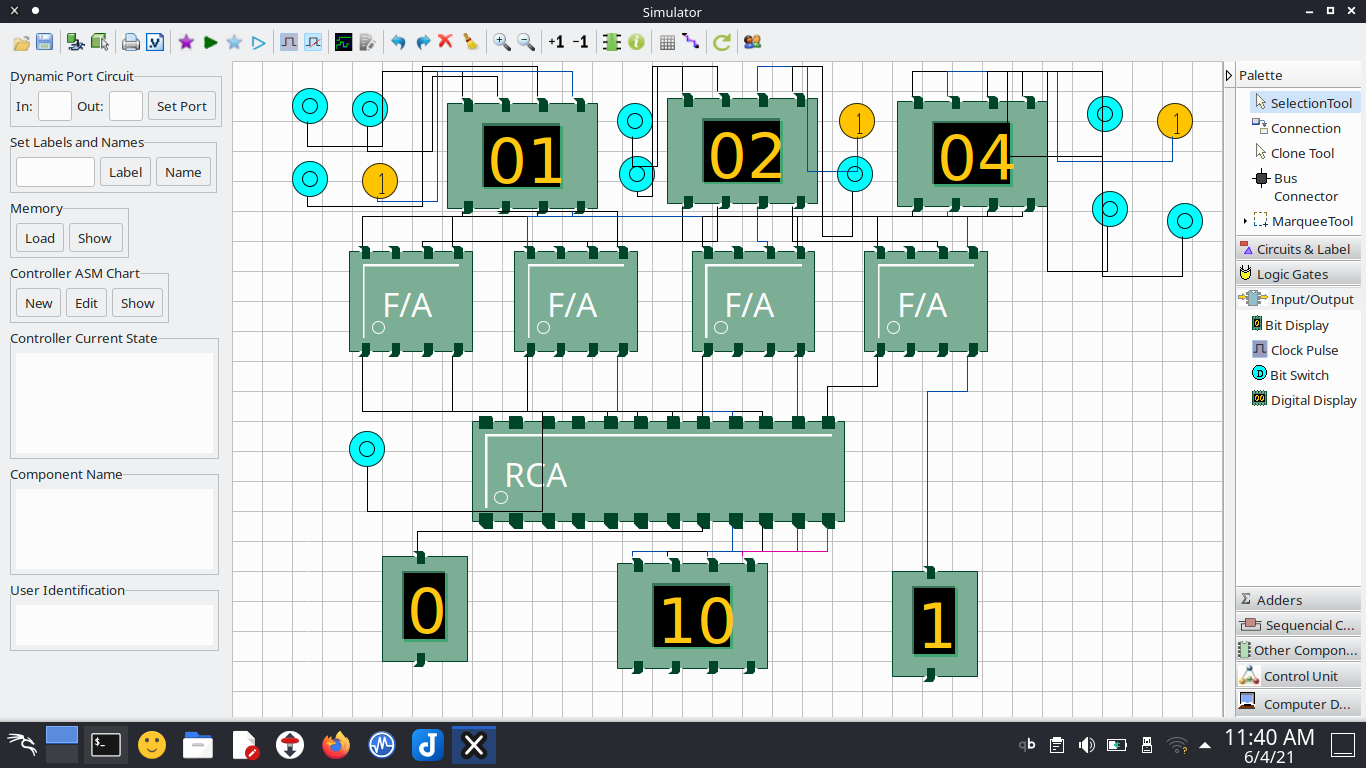
****

**Carry Select Adder**

****

****

**Carry Save**

****

**Output with Description**

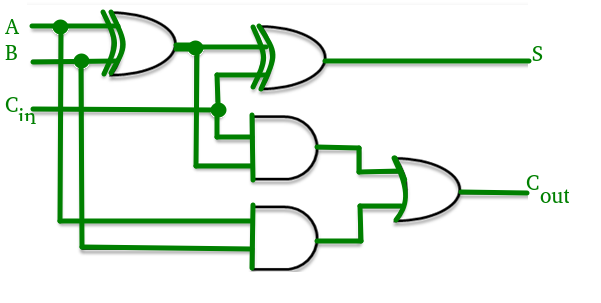
**Carry lookup Adder**

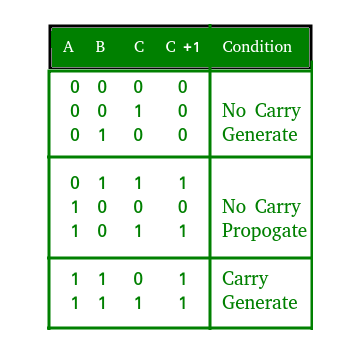
In ripple carry adders, for each adder block, the two bits that are to be added are available instantly. However, each adder block waits for the carry to arrive from its previous block. So, it is not possible to generate the sum and carry of any block until the input carry is known. The ith block waits for the block to produce its carry. So there will be a considerable time delay which carries propagation delay.

Consider the above 4-bit ripple carry adder. The sum S1 is produced by the corresponding full adder as soon as the input signals are applied to it. But the carry input C1 is not available on its final steady state value until carry C3 is available at its steady state value. Similarly C3 depends on C2 and C2 on C1. Therefore, though the carry must propagate to all the stages in order that output S3 and carry C1 settle their final steady-state value.

The propagation time is equal to the propagation delay of each adder block, multiplied by the number of adder blocks in the circuit. For example, if each full adder stage has a propagation delay of 20 nanoseconds, then S3 will reach its final correct value after 60 (20 × 3) nanoseconds. The situation gets worse if we extend the number of stages for adding more bits.

A carry look-ahead adder reduces the propagation delay by introducing more complex hardware. In this design, the ripple carry design is suitably transformed such that the carry logic over fixed groups of bits of the adder is reduced to two-level logic. Let us discuss the design in detail.

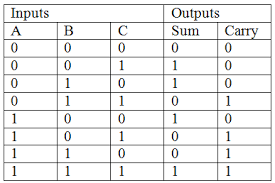


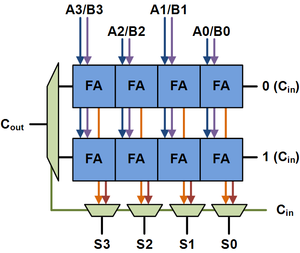


**Carry Select Adder**

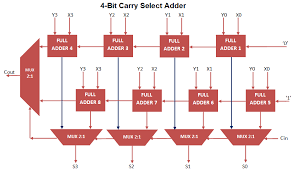
The carry-select adder generally consists of [ripple-carry adders](https://en.wikipedia.org/wiki/Ripple-carry_adder) and a [multiplexer](https://en.wikipedia.org/wiki/Multiplexer). Adding two n-bit numbers with a carry-select adder is done with two adders (therefore two ripple-carry adders), in order to perform the calculation twice, one time with the assumption of the carry-in being zero and the other assuming it will be one. After the two results are calculated, the correct sum, as well as the correct carry-out, is then selected with the multiplexer once the correct carry-in is known.

The number of bits in each carry select block can be uniform, or variable. In the uniform case, the optimal delay occurs for a block size of ⌊ n ⌋ {\displaystyle \lfloor {\sqrt {n}}\rfloor } under root n. When variable, the block size should have a delay, from addition inputs A and B to the carry out, equal to that of the multiplexer chain leading into it, so that the carry out is calculated just in time. The O ( n ) {\displaystyle O({\sqrt {n}})} O(under root n) delay is derived from uniform sizing, where the ideal number of full-adder elements per block is equal to the square root of the number of bits being added, since that will yield an equal number of MUX delays.



****

Above is the basic building block of a carry-select adder, where the block size is 4. Two 4-bit ripple-carry adders are multiplexed together, where the resulting carry and sum bits are selected by the carry-in. Since one ripple-carry adder assumes a carry-in of 0, and the other assumes a carry-in of 1, selecting which adder had the correct assumption via the actual carry-in yields the desired result.



**Carry Save Adder**

A carry-save adder is a type of digital adder, used to efficiently compute the sum of three or more binary numbers. It differs from other digital adders in that it outputs two (or more) numbers, and the answer of the original summation can be achieved by adding these outputs together. A carry save adder is typically used in a binary multiplier, since a binary multiplier involves addition of more than two binary numbers after multiplication. A big adder implemented using this technique will usually be much faster than conventional addition of those numbers.

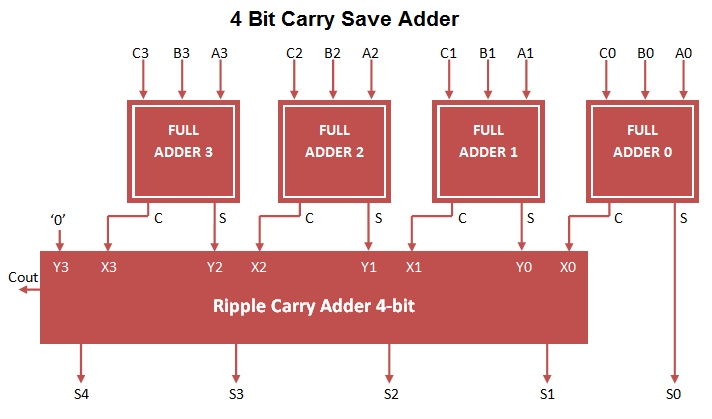
Using basic arithmetic, we calculate right to left, "8 + 2 = 0, carry 1", "7 + 2 + 1 = 0, carry 1", "6 + 3 + 1 = 0, carry 1", and so on to the end of the sum. Although we know the last digit of the result at once, we cannot know the first digit until we have gone through every digit in the calculation, passing the carry from each digit to the one on its left. Thus adding two *n*-digit numbers has to take a time proportional to *n*, even if the machinery we are using would otherwise be capable of performing many calculations simultaneously.

In electronic terms, using bits (binary digits), this means that even if we have *n* one-bit adders at our disposal, we still have to allow a time proportional to *n* to allow a possible carry to propagate from one end of the number to the other. Until we have done this,

1. We do not know the result of the addition.
2. We do not know whether the result of the addition is larger or smaller than a given number (for instance, we do not know whether it is positive or negative).

A carry look-ahead adder can reduce the delay. In principle the delay can be reduced so that it is proportional to log *n*, but for large numbers this is no longer the case, because even when carry look-ahead is implemented, the distances that signals have to travel on the chip increase in proportion to *n*, and propagation delays increase at the same rate. Once we get to the 512-bit to 2048-bit number sizes that are required in public-key cryptography, carry look-ahead is not of much help.

## 



**References**

* <https://en.wikipedia.org/wiki/Carry-select_adder>
* <https://www.google.com/search?q=carry+select+adder&client=firefox-b-d&tbm=isch&source=iu&ictx=1&fir=BKvYoHRibCjFQM%252CLGDffzZ26GYsDM%252C_&vet=1&usg=AI4_-kTJEvfrwffLaYHX2KgsYRFU20DXPQ&sa=X&ved=2ahUKEwjz7uS90qbxAhVMxjgGHWeHBUUQ9QF6BAgnEAE&biw=1366&bih=578#imgrc=BKvYoHRibCjFQM>
* <https://www.google.com/search?q=carry+save+adder&client=firefox-b-d&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjeooW406bxAhUn4jgGHSIZCRoQ_AUoAXoECAEQAw&biw=1366&bih=578#imgrc=4Z5ywu0dgO3_rM>
* <https://en.wikipedia.org/wiki/Carry-save_adder>
* <https://www.geeksforgeeks.org/carry-look-ahead-adder/>