EXPERIMENT O1
ELEMENTARY ADDERS

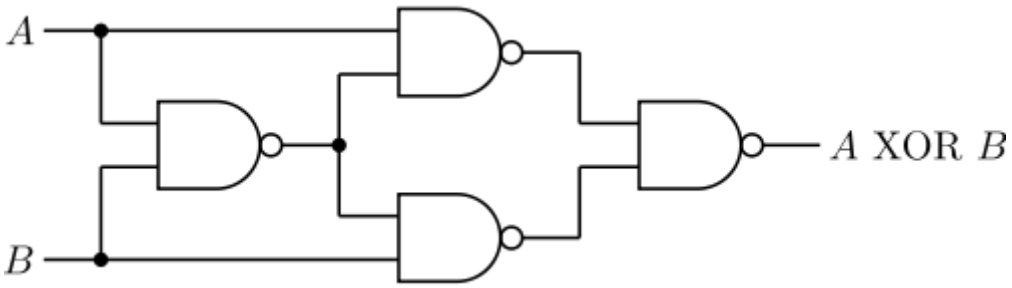
TIME DELAY COMPUTATION

ANALYSING BASIC LOGIC GATES –

Let the time-delay of each of the AND / OR / NAND / NOR gate be δ units of time.

$$T_{AND} = T_{OR} = T_{NAND} = T_{NOR} = \delta \text{ units}$$

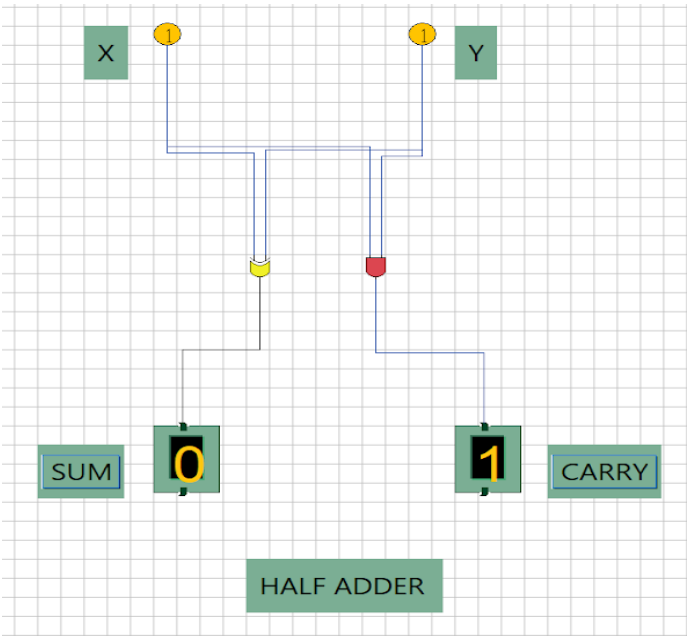
An XOR gate can be implemented as a combination of NAND gates as shown below.



Following the initial NAND gate, the next two NAND gates operate *simultaneously* before the final NAND gate outputs A XOR B. This implies the following time-delay of an XOR gate.

$$T_{XOR} = 3 * T_{NAND} = 3\delta \text{ units}$$

ANALYSING HALF ADDER –



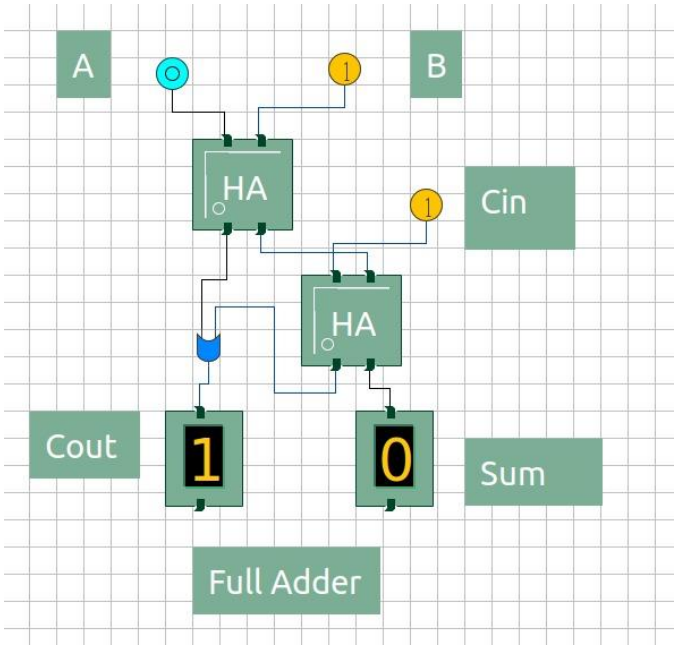
Consider the above configuration of a *half-adder circuit*. The circuit contains an XOR gate and an AND gate connected in parallel. The former gives the *sum* as output and the latter the *carry*. Let T_{HA-sum} be the time taken by the circuit to compute the bit corresponding to the sum, and $T_{HA-carry}$ be the time taken by the circuit to compute the bit corresponding to the carry. Directly using the time-delays of AND and XOR gates, we get the following.

$$T_{HA-sum} = T_{XOR} = 3\delta \text{ units}$$
$$T_{HA-carry} = T_{AND} = \delta \text{ units}$$

Note that since the two gates are connected in parallel, the complete output is obtained in T_{HA} time, where-

$$T_{HA} = \max\left(T_{HA-carry}, T_{HA-sum}\right) = 3\delta \text{ units}$$

ANALYSING FULL ADDER –



Now consider the above diagram of a *full adder circuit*. Let us consider time-delays in computation of the *sum* bit and the *carry* bit separately.

TIME DELAY IN COMPUTING SUM –

For computation of the *sum* bit S, the circuit must go through the following sequence of steps.

- Half adder HA1 takes A and B as input and gives sum S' as output.
- Half adder HA2 takes S' and C_{in} as input and gives sum S as output.

Let T_{FA-sum} be the time taken by the circuit to compute the *sum* bit S. Therefore

$$T_{FA-sum} = 2 * T_{HA-sum} = 6\delta \text{ units}$$

TIME DELAY IN COMPUTING CARRY –

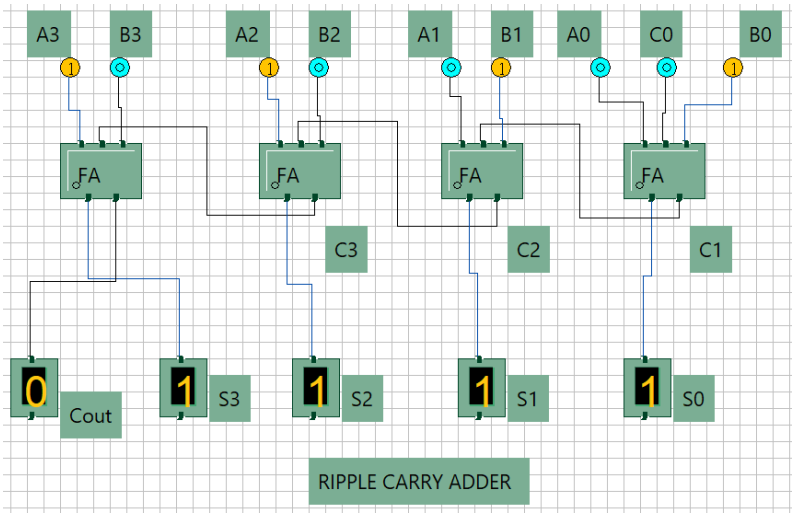
For computation of the *carry* bit C_{out}, the circuit must go through the following sequence of steps.

- Half adder HA1 takes A and B as input and gives carry C and sum S' as output.
- Half adder HA2 takes S' and C_{in} as input and gives carry C' as output.
- OR gate takes C and C' as input and gives C_{out} as output.

Let $T_{FA-carry}$ be the time taken by the circuit to compute the *sum* bit S. Therefore

$$T_{FA-carry} = T_{HA} + T_{HA-carry} + T_{OR} = 5\delta \text{ units}$$

ANALYSING RIPPLE CARRY ADDER –



Now consider the above diagram of a *ripple carry adder circuit* that adds two 4-bit binary strings. The time delay of the ripple carry adder circuit is the time it takes (since the first full-adder receives its input carry), in displaying the most significant bit of the binary output.

Let us take only the first full adder. This adder has nothing to do with the other adders because its inputs are not received from any of them. Therefore, its outputs are obtained in the typical time delays of a full adder. That is, its *sum* bit will be computed in $T_{FA-sum} = 6\delta$ units of time and the *carry* bit is computed in $T_{FA-carry} = 5\delta$ units of time. While the first full adder is performing some

computation, the rest of the full adders are also performing some computations. This is because, two of the three inputs are already available to those adders and by the design of the full-adder implemented in this case, the first half-adder component inside it does not need any input carry to compute the output. Therefore, before a full adder (except the first one) receives its input carry from the previous full adder, it has already completed "one half-adder worth of computation", i.e., $T_{saved} = T_{HA} = 3\delta$ units of time.

Due to this phenomenon, when the second adder accepts the input carry from the first adder, it takes from that moment only 3δ additional units of time to compute the *sum* bit ($T_{FA-sum} - T_{saved} = 3\delta$). Similarly, it takes only 2δ additional units of time to compute the *carry* bit ($T_{FA-carry} - T_{saved} = 2\delta$). Therefore, if the first adder receives its input carry at t=0, then the second adder outputs its sum at t = $3\delta + 5\delta = 8\delta$ and outputs the carry at t = $5\delta + 2\delta = 7\delta$. Let us chart down a timeline to understand the computations of the next few adders. Note that $t_{FAi-sum}$ is the absolute time at which the i-th full adder (from the right) outputs its *sum* bit and $t_{FAi-carry}$ is the absolute time at which the i-th full adder (from the right) outputs its *carry* bit.

$t_{start} = 0$

$t_{FA1-sum} = 6\delta$

$t_{FA1-carry} = 5\delta$

$t_{FA2-sum} = 8\delta$

$t_{FA2-carry} = 7\delta$

$t_{FA3-sum} = t_{FA2-carry} + T_{FA-sum} - T_{saved} = 10\delta$

$t_{FA3-carry} = t_{FA2-carry} + T_{FA-carry} - T_{saved} = 9\delta$

Now consider the value of the general terms, $t_{FAn-sum}$ and $t_{FAn-carry}$. They are observed to form an intertwined recursive structure that can be deduced by the following argument.

$t_{FAn-sum} = t_{FA(n-1)-carry} + T_{FA-sum} - T_{saved} = t_{FA(n-1)-carry} + 3\delta$

$t_{FAn-carry} = t_{FA(n-1)-carry} + T_{FA-carry} - T_{saved} = t_{FA(n-1)-carry} + 2\delta$

The following two observations are prominent from the above two recursive formulas.

- The recursion structure of $t_{FAn-carry}$ shows that it is an arithmetic progression with the common difference of 2δ.
- The entity $t_{FAn-sum} - t_{FAn-carry}$ is an invariant in this case with a fixed value of δ.

Using the formula of the nth term of an AP,

$t_{FAn-carry} = 5\delta + (n-1) * 2\delta = (2n+3)\delta$ units of time.

Using the second observation,

$t_{FAn-sum} = t_{FAn-carry} + \delta = (2n+4)\delta$ units of time.

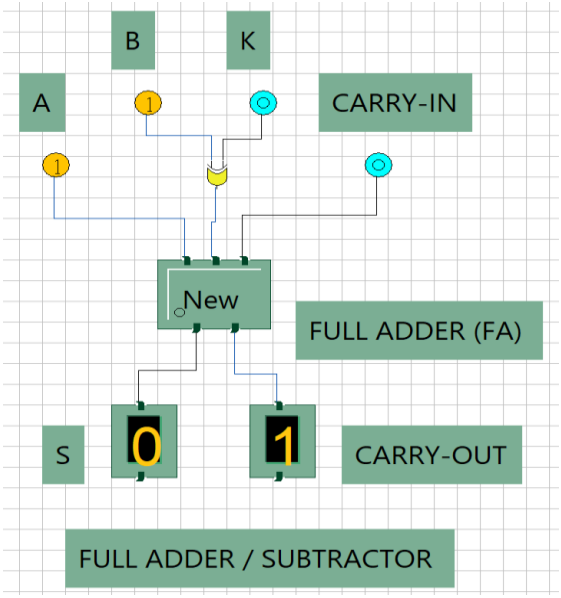
Therefore,

$$T_{RCA-sum} = (2n+4)\delta \text{ units} = O(n)\delta$$

$$T_{RCA-carry} = (2n+3)\delta \text{ units} = O(n)\delta$$

Now, as particularly implemented in this experiment, for 4-bit input vectors, the delay of the ripple carry adder will be 12δ units of time.

ANALYSING FULL ADDER / SUBTRACTOR –

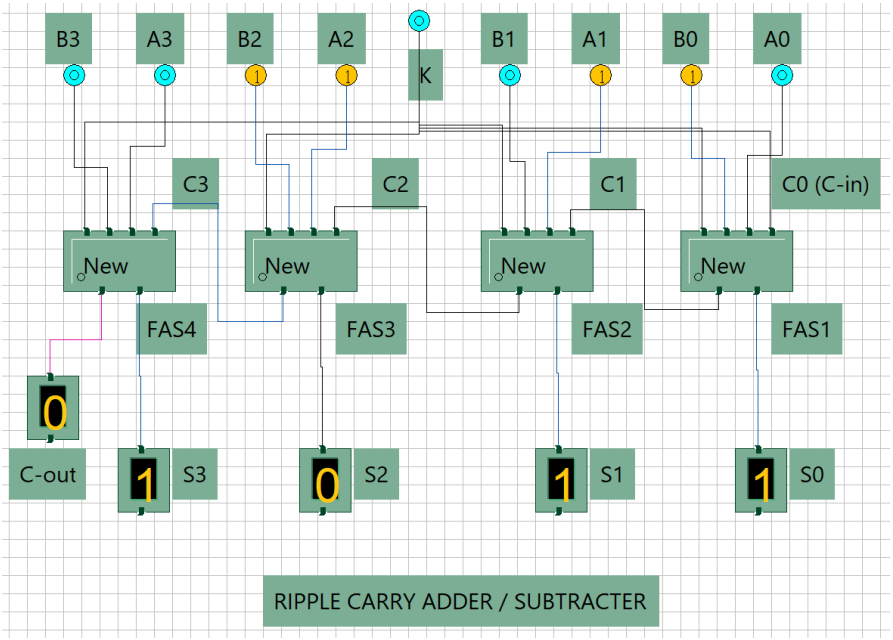


Consider the above diagram for a *full adder/subtractor circuit*. The only addition to this circuit compared to the full adder circuit is an XOR gate at the top. The time taken by the XOR gate in computing B XOR K is reflected equally in the time-delays for computations of both the *sum* and the *carry* bits because this XOR operation is added to the sequence of steps for both the computations. Therefore

$$T_{FAS-sum} = T_{FA-sum} + T_{XOR} = 9\delta \text{ units}$$

$$T_{FAS-carry} = T_{FA-carry} + T_{XOR} = 8\delta \text{ units}$$

ANALYSING RIPPLE CARRY ADDER / SUBTRACTOR –



Now consider the above diagram of a *ripple carry adder/subtracter circuit* that adds two 4-bit binary strings. The time delay of the ripple carry adder/subtracter circuit is the time it takes (since the first full-adder receives its input carry), in displaying the most significant bit of the binary output.

Let us take only the first full adder/subtracter. This adder/subtracter has nothing to do with the other adders/subtracters because its inputs are not received from any of them. Therefore, its outputs are obtained in the typical time delays of a full adder/subtracter. That is, its *sum* bit will be computed in $T_{FAS-sum} = 9\delta$ units of time and the *carry* bit is computed in $T_{FAS-carry} = 8\delta$ units of time. While the first full adder/subtracter is performing some computation, the rest of the full adders/subtracters are also performing some computations. This is because, three of the four inputs are already available to those components and by the design of the full- adder/subtracter implemented in this case, the first half-adder component and the XOR gate before that, present inside it do not need any input carry to compute the output. Therefore, before a full adder/subtracter (except the first one) receives its input carry from the previous full adder/subtracter, it has already completed "one half-adder plus one XOR gate worth of computation", i.e., $T_{saved} = T_{HA} + T_{XOR} = 6\delta$ units of time.

Due to this phenomenon, when the second adder/subtracter accepts the input carry from the first one, it takes from that moment only 3δ additional units of time to compute the *sum* bit ($T_{FAS-sum} - T_{saved} = 3\delta$). Similarly, it takes only 2δ additional units of time to compute the *carry* bit ($T_{FAS-carry} - T_{saved} = 2\delta$). Therefore, if the first adder/subtracter receives its input carry at t=0, then the second adder/subtracter outputs its sum at t = $3\delta + 8\delta = 11\delta$ and outputs the carry at t = $8\delta + 2\delta = 10\delta$. Let us chart down a timeline to understand the computations of the next few adders. Note that $t_{FASi-sum}$ is the absolute time at which the i-th full adder/subtracter (from the right) outputs its *sum* bit and $t_{FASi-carry}$ is the absolute time at which the i-th full adder/subtracter (from the right) outputs its *carry* bit.

$t_{start} = 0$

$t_{FAS1-sum} = 9\delta$

$t_{FAS1-carry} = 8\delta$

$t_{FAS2-sum} = 11\delta$

$t_{FAS2-carry} = 10\delta$

$t_{FAS3-sum} = t_{FAS2-carry} + T_{FAS-sum} - T_{saved} = 13\delta$

$t_{FAS3-carry} = t_{FAS2-carry} + T_{FAS-carry} - T_{saved} = 12\delta$

Now consider the value of the general terms, $t_{FASn-sum}$ and $t_{FASn-carry}$. They are observed to form an intertwined recursive structure that can be deduced by the following argument.

$t_{FASn-sum} = t_{FAS(n-1)-carry} + T_{FAS-sum} - T_{saved} = t_{FAS(n-1)-carry} + 3\delta$

$t_{FASn-carry} = t_{FAS(n-1)-carry} + T_{FAS-carry} - T_{saved} = t_{FAS(n-1)-carry} + 2\delta$

The following two observations are prominent from the above two recursive formulas.

- The recursion structure of $t_{FASn-carry}$ shows that it is an arithmetic progression with the common difference of 2δ.
- The entity $t_{FASn-sum} - t_{FASn-carry}$ is an invariant in this case with a fixed value of δ.

Using the formula of the nth term of an AP,

$t_{FASn-carry} = 8\delta + (\mathrm{n} - 1) * 2\delta = (2\mathrm{n} + 6)\delta$ units of time.

Using the second observation,

$t_{FASn-sum} = t_{FASn-carry} + \delta = (2n + 7)\delta$ units of time.

Therefore,

$$T_{RCAS-sum} = (2n + 7)\delta \text{ units} = O(n)\delta$$

$$T_{RCAS-carry} = (2n + 6)\delta \text{ units} = O(n)\delta$$

Now, as particularly implemented in this experiment, for 4-bit input vectors, the delay of the ripple carry adder will be 15δ units of time.

SUMMARY

	Time Delay		Time Delay
T_{AND} / T_{NAND}	δ units	$T_{FA-carry}$	5δ units
T_{OR}/T_{NOR}	δ units	$T_{RCA}/T_{RCA-sum}$	$(2n + 4)\delta$ units
T_{XOR}	3δ units	$T_{RCA-carry}$	$(2n + 3)\delta$ units
T_{HA-sum}	3δ units	$T_{FAS-sum}$	9δ units
$T_{HA-carry}$	δ units	$T_{FAS-carry}$	8δ units
T_{HA}	3δ units	$T_{RCAS}/T_{RCAS-sum}$	$(2n + 7)\delta$ units
T_{FA-sum}	6δ units	$T_{RCAS-carry}$	$(2n + 6)\delta$ units

GROUP MEMBERS

Group ID	Roll No.	Name
09	19CS10011	Anish Sofat
09	19CS10031	Abhishek Gandhi
09	19CS10044	Nakul Aggarwal
09	19CS10051	Sajal Chhamunya
09	19CS10053	Satvik Bansal