Adders

1. Carry skip adders
2. Carry select adders
3. Carry save adders

Carry skip adders:

A carry-skip adder consists of a simple ripple carry-adder with a special speed up carry chain called a **skip chain**. This chain defines the distribution of ripple carry blocks, which compose the skip adder. The addition of two binary digits at stage i, where i is not equal to 0, of the ripple carry adder depends on the carry in, ***Ci*** , which in reality is the carry out, ***Ci-1***, of the previous stage. Therefore, in order to calculate the sum and the carry out, ***Ci+1*** , of stage i, it is imperative that the carry in, ***Ci***, be known in advance. It is interesting to note that in some cases ***Ci+1*** can be calculated without knowledge of ***Ci***.

Boolean Equations of a Full Adder:

Pi = Ai Bi Equ. 1 --carry propagate of ith stage

Si = Pi Ci Equ. 2 --sum of ith stage

Ci+1 = AiBi + PiCi Equ. 3 --carry out of ith stage

Supposing that ***Ai*** = ***Bi***, then ***Pi*** in equation 1 would become zero (equation 4). This would make ***Ci+1*** to depend only on the inputs ***Ai*** and ***Bi***, without needing to know the value of ***Ci***. **Ai = Bi ; Pi = 0** Equ. 4 --from #Equation 1

If **Ai = Bi = 0 ; Ci+1 = AiBi = 0** --from equation 3

If **Ai = Bi = 1; Ci+1 = AiBi = 1** --from equation 3

Therefore, if Equation 4 is true then the carry out, ***Ci+1***, will be one if ***Ai*** = ***Bi = 1*** or zero if ***Ai***= ***Bi = 0***. Hence the output can be computed with the carry out at any stage of the addition provided equation 4 holds. These would enable to build an adder whose average time of computation would be proportional to the longest chains of zeros and of different digits of A and B.

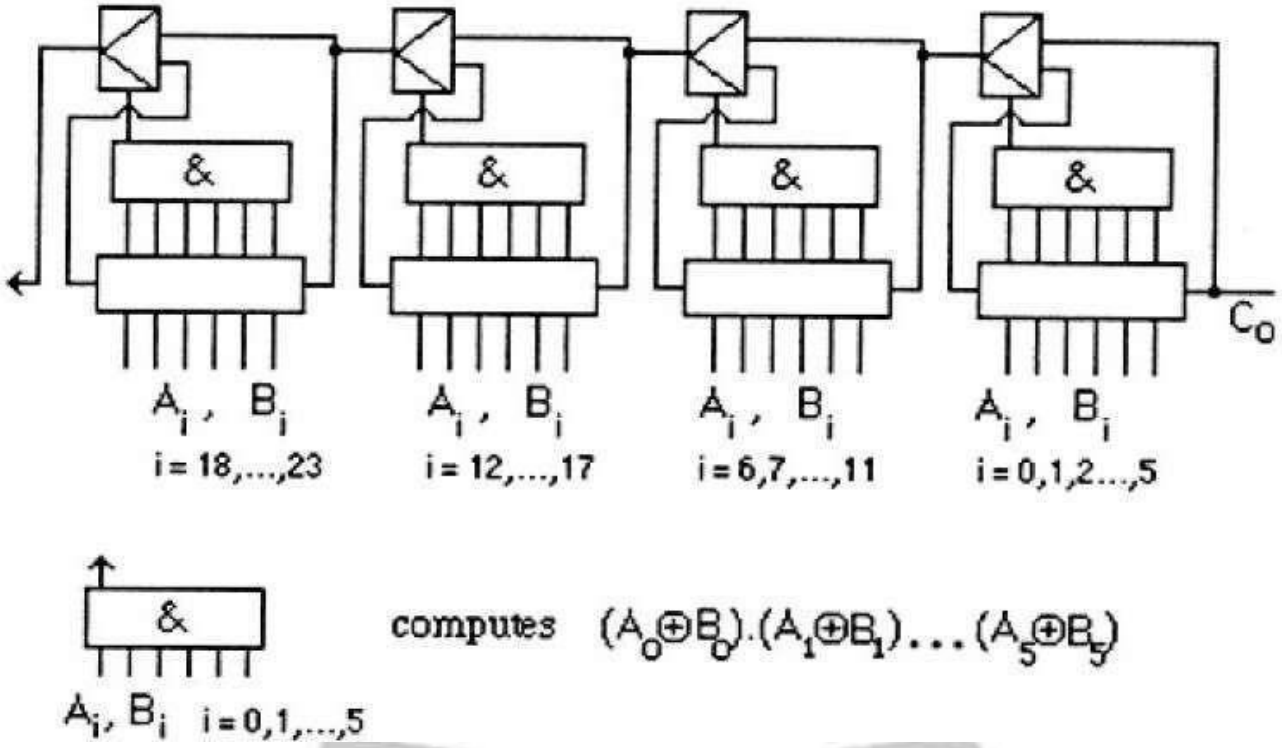

Fig4.1.1: Carry skip Chain

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

Carry select adders:

The concept of the carry-select adder is to compute alternative results in

parallel and subsequently selecting the correct result with single or multiple stage hierarchical techniques . In order to enhance its speed performance, the carry-select adder increases its area requirements. In carry-select adders both sum and carry bits are calculated for the two alternatives: input carry "0" and "1". Once the carry-in is delivered, the correct computation is chosen (using a MUX) to produce the desired output. Therefore instead of waiting for the carry-in to calculate the sum, the

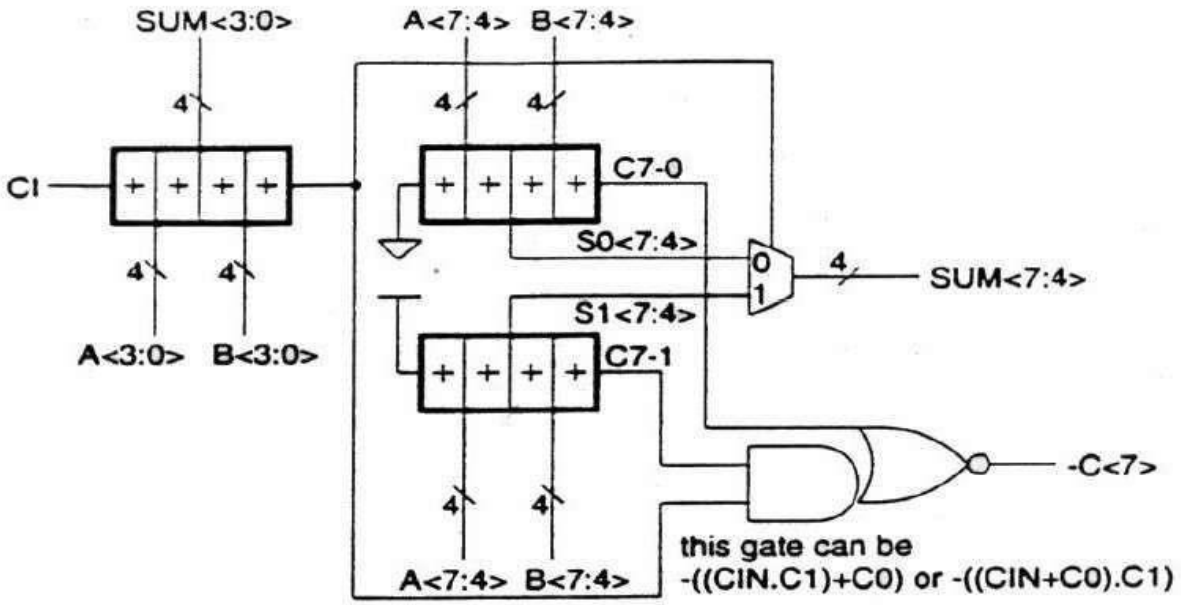

Fig 4.1.2: Concept of carry select adder

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

sum is correctly output as soon as the carry-in gets there. The time taken to compute the sum is then avoided which results in a good improvement in speed.

Carry-select adders can be divided into equal or unequal sections. For each

section, the calculation of two sums is accomplished using two 4-bit ripple-carry adders. One of these adders is fed with a 0 as carry-in whereas the other is fed a 1. Then using a multiplexer, depending on the real carryout of the previous section, the correct sum is chosen. Similarly, the carryout of the section is computed twice and chosen depending of the carryout of the previous section. The concept can be expanded to any length for example a 16-bits carry-select adder can be composed of four sections. Each of these sections is composed of two 4-bits ripple-carry adders. This is referred as linear expansion.

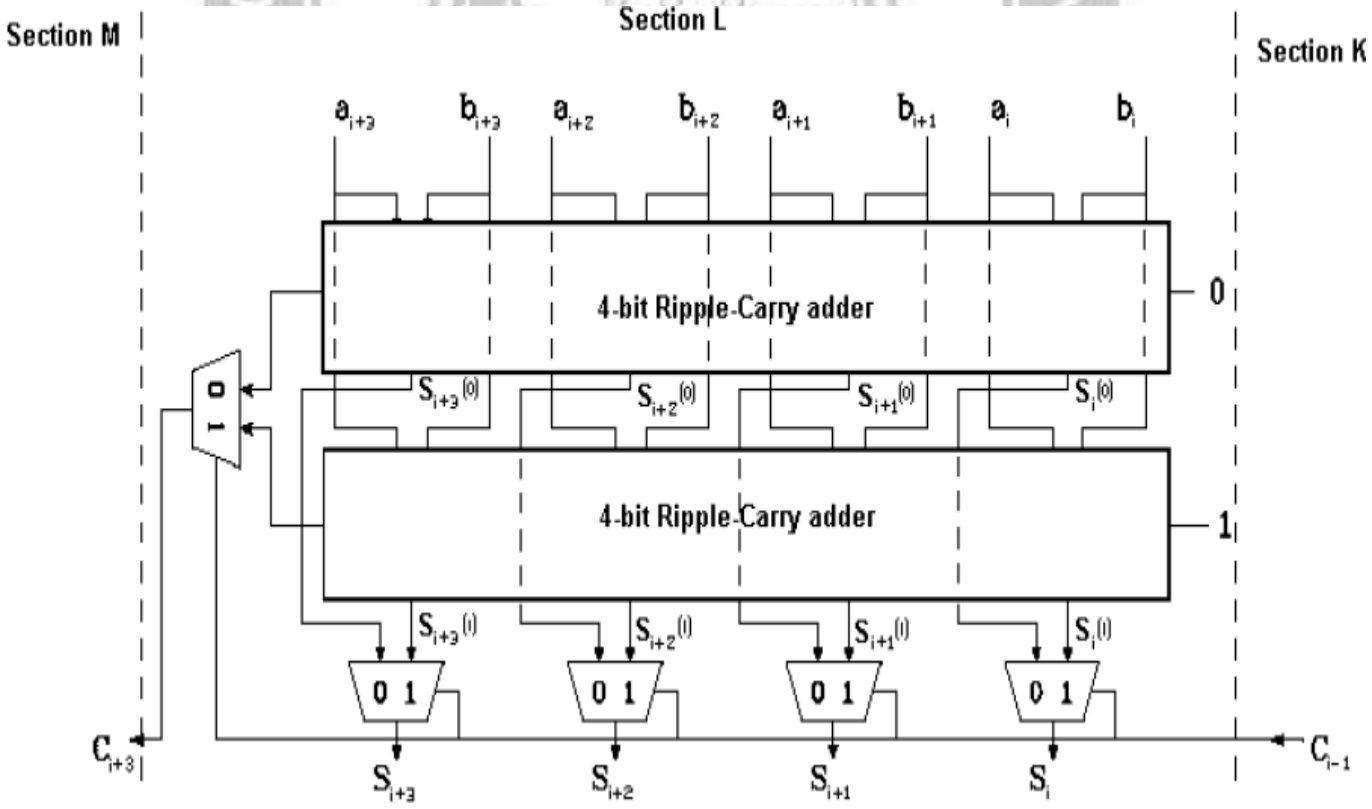

Fig 4.1.3: one section of a large carry select adder

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

Carry save adders:

In most computations, we need to add several operands together, carry save adders are ideal for this type of addition. A carry save adder consists of a ladder of stand alone full adders, and carries out a number of partial additions. The principal idea is that the carry has a higher power of 2 and thus is routed to the next column. Doing additions with Carry save adder saves time and logic.

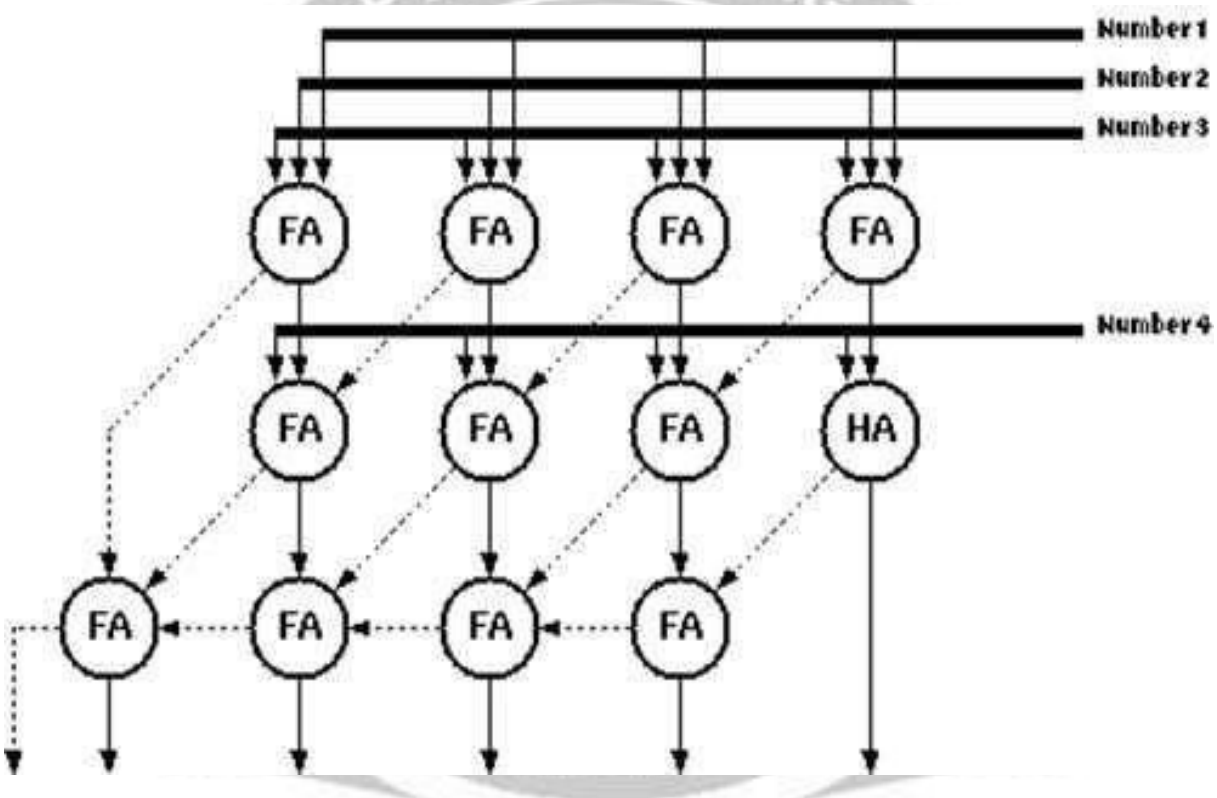

Fig 4.1.4: Carry save adder for 4 bit number

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

In this method, for the first 3 numbers a row of full adders are used. Then a row of full adders is added for each additional number. The final results, in the form of two numbers SUM and CARRY, are then summed up with a carry propagate adder or any other adder

Ripple Carry Adder

An N-bit adder can be constructed by casing N full adders as shown in Fig.4.1

(a) for N=4. This is called a carry-ripple adder (or ripple-carry adder). The carry-out of bit i, C_i is the carry-in to bit i + 1. This carry is said to have twice the weight of the sum Si. The delay of the adder is set by the time for the carries to ripple through the N stages, so the $t_{c \rightarrow c_{out}}$ delay should be minimized.

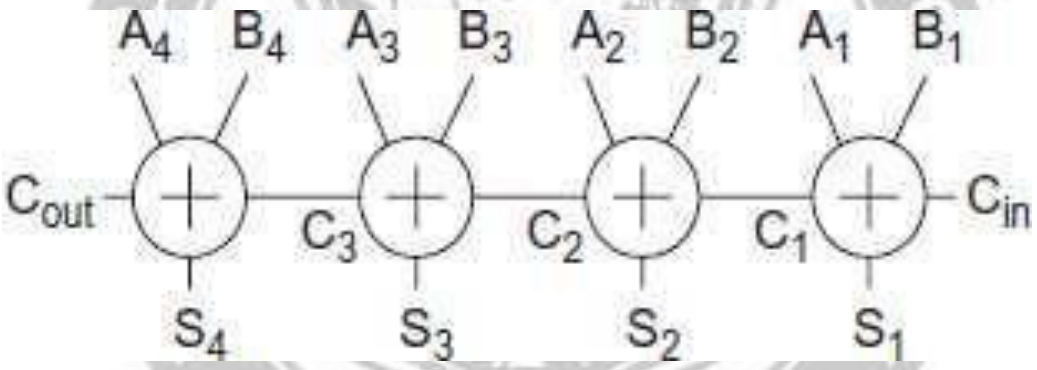

Fig.4.1.4 (a) 4-bit carry-ripple adder

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

In carry-ripple adders, the critical path goes from C to Cout through many full adders, so the extra delay computing S is unimportant. This delay can be reduced by omitting the inverters on the outputs. Fig.4.1 (b) shows the adder with transistor sizes optimized to favor the critical path using a number of techniques:

- Feed the carry-in signal (C) to the inner inputs so the internal capacitance is already discharged.
- Make all transistors in the sum logic whose gate signals are connected to the

carry-in and carry logic minimum size (1 unit, e.g., 4λ). This minimizes the branching effort on the critical path. Keep routing on this signal as short as possible to reduce interconnect capacitance.

- Determine widths of series transistors by logical effort and simulation. Build an asymmetric gate that reduces the logical effort from C to C_{out} at the expense of effort to S.
- Use relatively large transistors on the critical path so that stray wiring capacitance is a small fraction of the overall capacitance.
- Remove the output inverters and alternate positive and negative logic to reduce delay and transistor count to 24.

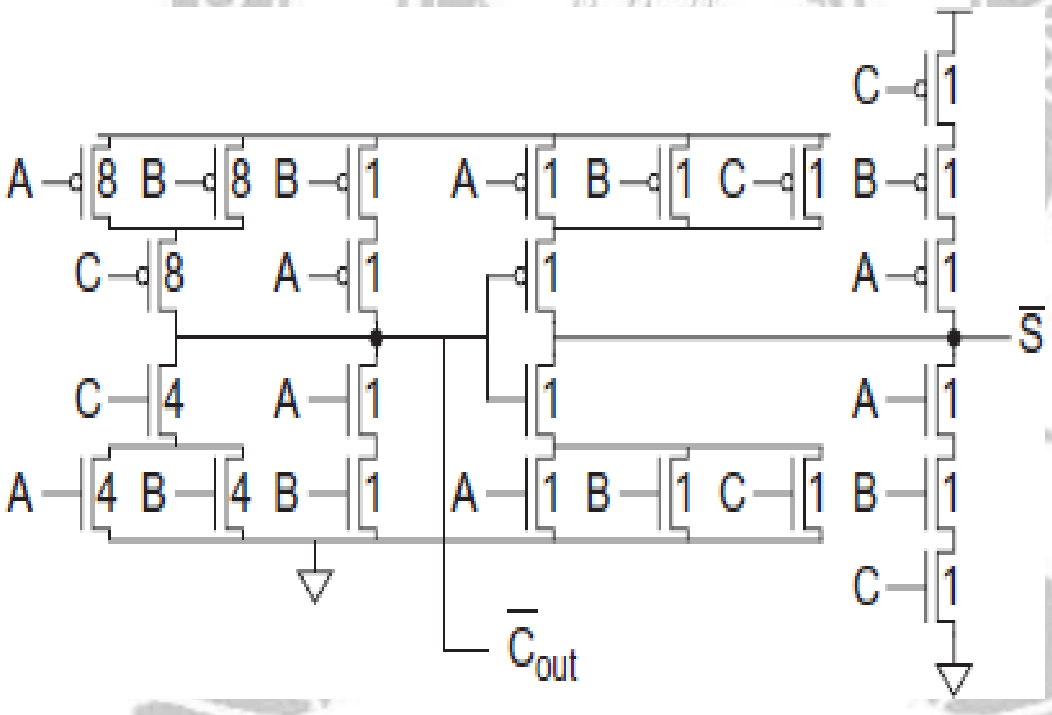

Fig.4.1.4(b) Full adder for carry-ripple operation

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

This delay can be reduced by omitting the inverters on the outputs, as was done in Fig.4.1 (b). Because addition is a self-dual function (i.e., the function of complementary inputs is the complement of the function), an inverting full adder receiving complementary inputs produces true outputs. Fig.4.1 (c) shows a carry ripple adder built from inverting full adders. Every other stage operates on complementary data. The delay inverting the adder inputs or sum outputs is off the critical ripple-carry path.

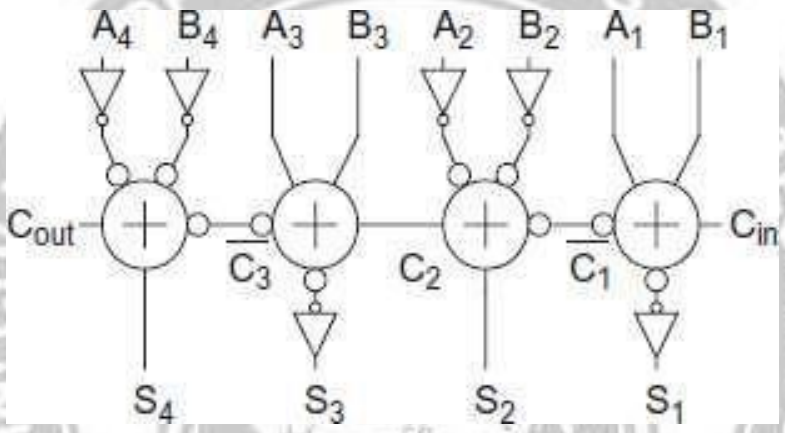

Fig.4.1.4 (c) 4-bit carry-ripple adder (inverting full adders)

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

- **RIPPLE CARRY ADDITION USING PG**

The critical path of the carry-ripple adder passes from carry-in to carry-out along the carry chain majority gates. As the P and G signals will have already stabilized by the time the carry arrives, we can use them to simplify the majority

$$C_i = A_i B_i + (A_i + B_i) C_{i-1}$$
$$= A_i B_i + (A_i \oplus B_i) C_{i-1}$$
$$= G_i + P_i C_{i-1}$$

function into an AND-OR gate:

Because $C_i = G_{i:0}$, carry-ripple addition can now be viewed as the extreme case

of group

$$G_{i:0} = G_i + P_i \cdot G_{i-1:0}$$

PG logic in which a 1-bit group is combined with an i-bit group to form an (i+1) bit group

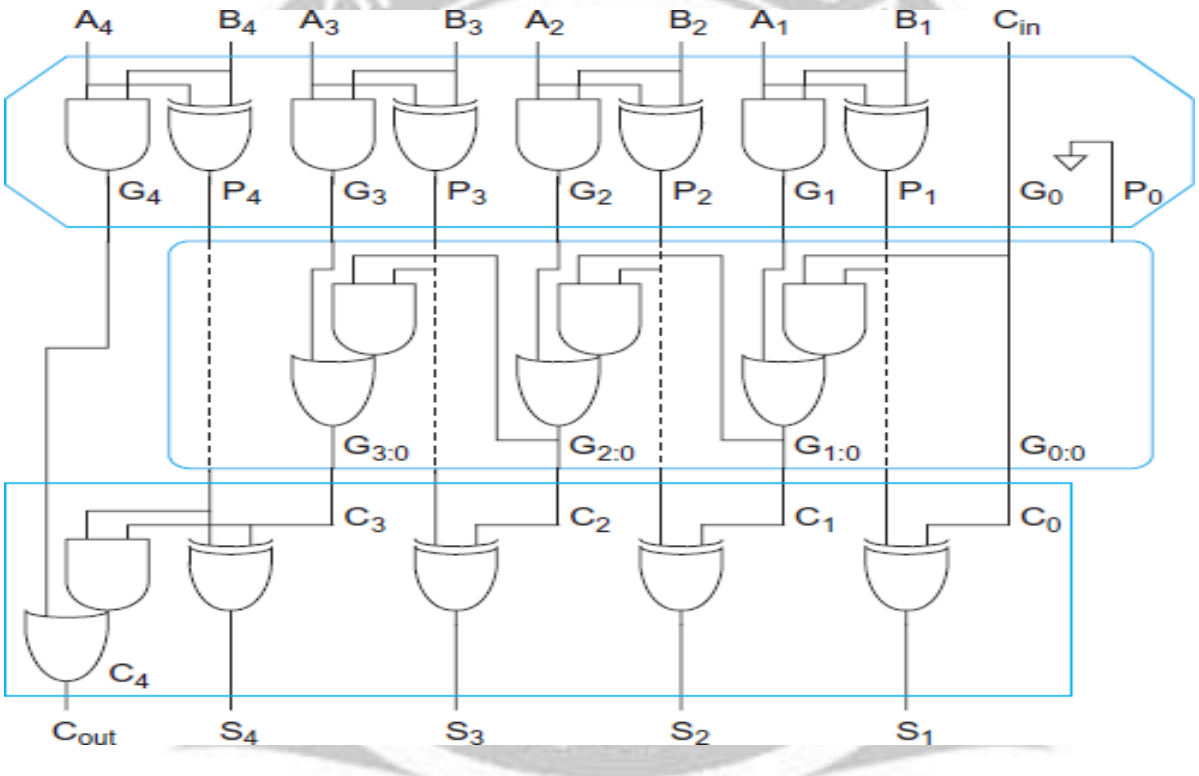

Fig.4.1.4 (d) 4-bit ripple carry adder

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

Fig.4.1 (d) shows a 4-bit carry-ripple adder. The critical carry path now proceeds through a chain of AND-OR gates rather than a chain of majority gates.