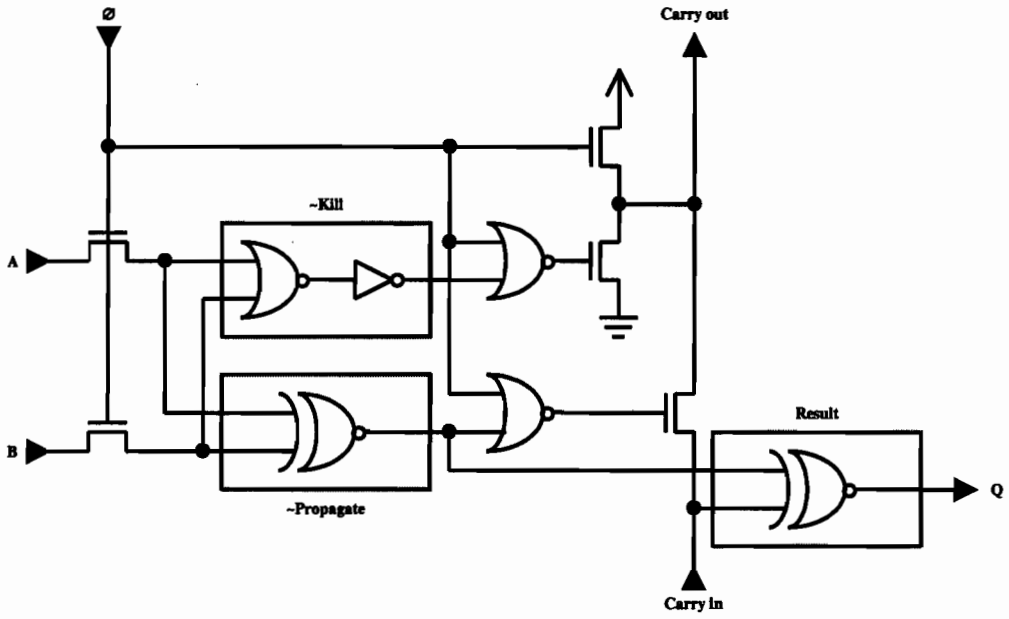
Answer

$$\text{Sum} = A \oplus B \oplus C_{in}$$

and $$C_{out} = A.B + A.C_{in} + B.C_{in}$$

Chain carry-out to carry-in for ripple carry.

Manchester carry is precharged, also latching the A and B inputs, when Ø=1 and evaluated when Ø=0:



Carry look-ahead refines the carry:

$$\begin{aligned} C_{out} &= A.B + A.C_{in} + B.C_{in} \\ &= A.B + (A + B).C_{in} \\ &= G + P.C_{in} \end{aligned}$$

where G and P are generate and propagate signals defined as:

$$G = A.B$$

and $$P = A + B$$

In general, for stage i of an n-bit adder:

$$C_{i+1} = G_i + P_i.C_i$$

where

$$G_i = A_i.B_i$$

and $$P_i = A_i + B_i$$

Several stages can then be grouped together:

$$\begin{aligned} C_{i+1} &= G_i + P_i.G_{i-1} + P_i.P_{i-1}.G_{i-2} + P_i.P_{i-1}.P_{i-2}.G_{i-3} + P_i.P_{i-1}.P_{i-2}.P_{i-3}.C_{i-3} \\ &= G_{i-3,i} + P_{i-3,i}.C_{i-3} \end{aligned}$$

where

$$G_{i-3,i} = G_i + P_i.G_{i-1} + P_i.P_{i-1}.G_{i-2} + P_i.P_{i-1}.P_{i-2}.G_{i-3}$$

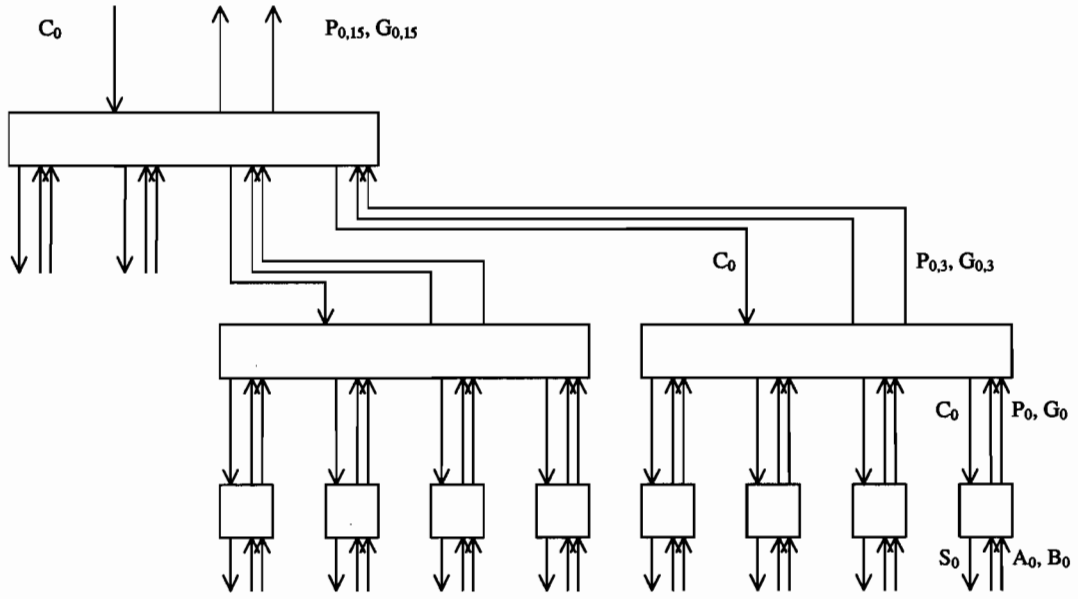
and $$P_{i-3,i} = P_i.P_{i-1}.P_{i-2}.P_{i-3}$$

Moreover:

$$G_{i-15,i} = G_{i-3,i} + P_{i-3,i}.G_{i-7,i-4} + P_{i-3,i}.P_{i-7,i-4}.G_{i-11,i-8} + P_{i-3,i}.P_{i-7,i-4}.P_{i-11,i-8}.G_{i-15,i-12}$$

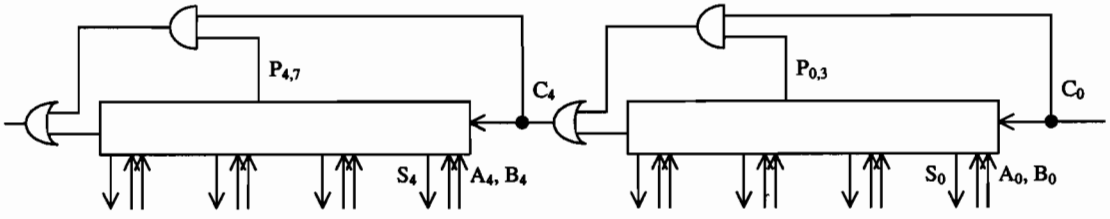
and $$P_{i-15,i} = P_{i-3,i}.P_{i-7,i-4}.P_{i-11,i-8}.P_{i-15,i-12}$$

These equations are implemented as a tree structure of identical circuits:



Carry select divides the n-bit word up into blocks of, say, 4 bits. Two adders are used for each block, one assuming a carry-in of 0 and the other a carry-in of 1. The actual carry-in is then used to select between the outputs of the two adders, including selecting the carry-out which is rippled on to the selector for the next block.

Carry skip exploits the observation that the propagate signals are much easier to compute than generate, allowing fast carry propagation alongside blocks of bits producing the generated carry signals more slowly. The system is particularly well suited to dynamic logic.



The system can be further optimised by having blocks of different sizes – short at the beginning and the end of the word, and longer in the middle. This reduces the worst case carry propagation delays.

Performance can be summarised as follows:

	Time	**Space**
Ripple	$O(n)$	$O(n)$
Carry look-ahead	$O(\log n)$	$O(n \log n)$
Carry select	$O(\sqrt{n})$	$O(n)$
Carry skip	$O(\sqrt{n})$	$O(n)$

Question draws on adders section of system design.