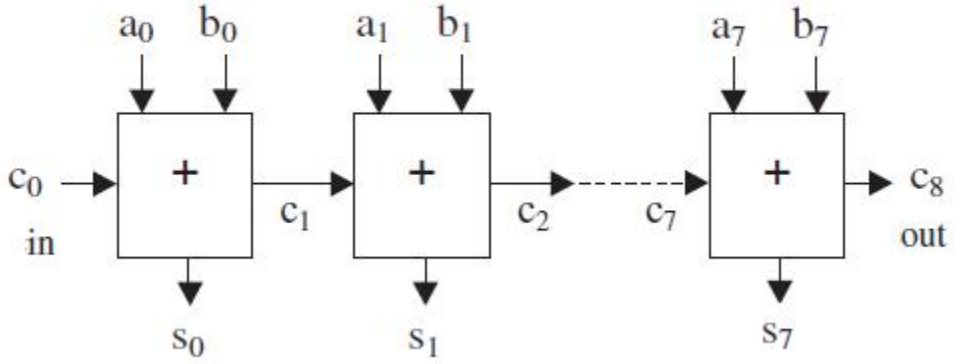
Carry Ripple Adder

Figure shows an 8-bit unsigned carry ripple adder. The top-level diagram shows the inputs and outputs of the circuit: a and b are the input vectors to be added, cin is the carry-in bit, s is the sum vector, and cout is the carry-out bit. The one-levelbelow-top diagram shows how the carry bits propagate (ripple).

Each section of the latter diagram is a full-adder unit .Thus its outputs can be computed by means of:

sj ¼ aj XOR bj XOR cj
cjþ1 ¼ (aj AND bj) OR (aj AND cj) OR (bj AND cj)

Two solutions are presented, being one generic (that is, for any number of bits, and the other specific for 8-bit numbers. Moreover, we illustrate the use of vectors and FOR/LOOP in the first solution, and of integers and IF in the second. Simulation results from either solution are shown in figure.




```
LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
ENTITY adder IS
6 GENERIC (length : INTEGER := 8);
7 PORT ( a, b: IN STD_LOGIC_VECTOR (length-1 DOWNTO 0);
8 cin: IN STD_LOGIC;
9 s: OUT STD_LOGIC_VECTOR (length-1 DOWNTO 0);
10 cout: OUT STD_LOGIC);
11 END adder;
12 ------------------------------------------------
13 ARCHITECTURE adder OF adder IS
14 BEGIN
15 PROCESS (a, b, cin)
16 VARIABLE carry : STD_LOGIC_VECTOR (length DOWNTO 0);
```

```
17 BEGIN
18 carry(0) := cin;
19 FOR i IN 0 TO length-1 LOOP
20 s(i) <= a(i) XOR b(i) XOR carry(i);
21 carry(i+1) := (a(i) AND b(i)) OR (a(i) AND
22 carry(i)) OR (b(i) AND carry(i));
23 END LOOP;
24 cout <= carry(length);
25 END PROCESS;
26 END adder;
```