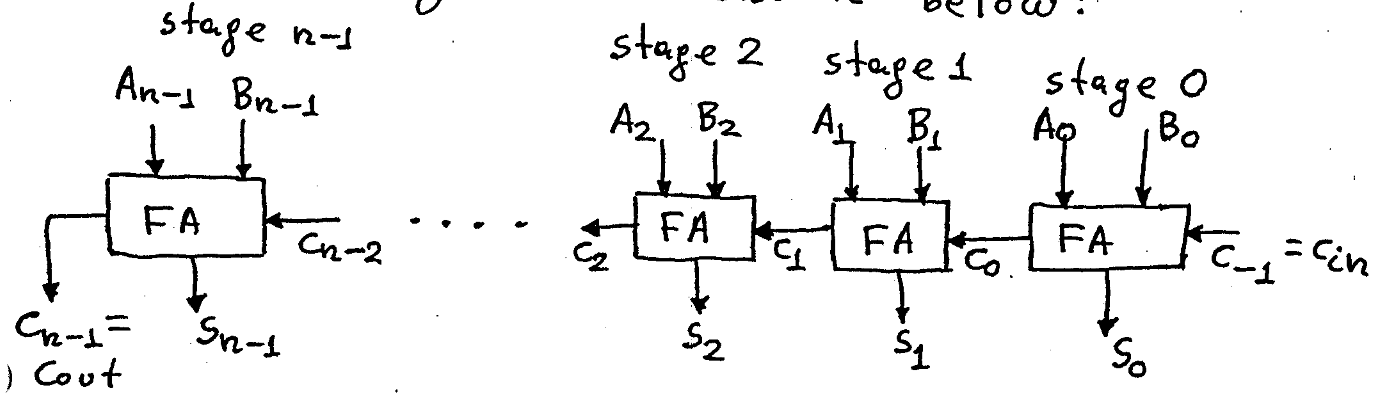
Class Notes for EE 7715 Part I

Instructor: Alex Skavantzos

The Carry Lookahead Adder

Consider two n-bit binary numbers
$A = (A_{n-1} A_{n-2} \cdots A_1 A_0)_2$
$B = (B_{n-1} B_{n-2} \cdots B_1 B_0)_2$

One simple design of a binary adder is the ripple carry adder shown below:

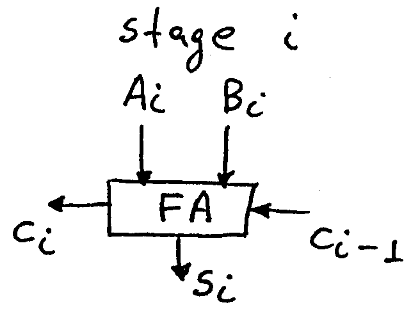


The ripple carry adder shown above is a slow design. Its propagation delay is $n \times D_{FA}$ where D_{FA} is the propagation delay through a Full Adder (FA).

A faster design of a binary adder is the carry lookahead adder presented next. The following notations will be used in this handout:

- $\cdot$ denotes AND operation
- $+$ denotes OR operation
- $\oplus$ denotes Exclusive-OR operation

Consider the stage i of the addition of two n-bit numbers ② a



The truth table and logic equations of the Full Adder (FA) are shown below:

A_i	B_i	c_{i-1}	c_i	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$c_i = A_i \cdot B_i + c_{i-1} \cdot (A_i \oplus B_i)$$

$$s_i = A_i \oplus B_i \oplus c_{i-1}$$

The above truth table can be rewritten as shown below

A_i	B_i	c_i
0	0	0
0	1	c_{i-1}
1	0	c_{i-1}
1	1	1

Define the functions G_i and P_i as follows:

$$\boxed{\begin{aligned} G_i &= A_i \cdot B_i \\ P_i &= A_i \oplus B_i \end{aligned}} \quad (1)$$

The function G_i is called "carry generate" function and reflects the condition where a carry out is generated at the ith stage. The function P_i is called "carry propagate" function. This function P_i is true when the ith stage will propagate the incoming carry c_{i-1} to the next higher stage.

Using the expressions of G_i, P_i of equation (1), the logic equations for c_i and s_i of the previous page become

$$\boxed{c_i = G_i + c_{i-1} \cdot P_i} \quad (2)$$

$$\boxed{s_i = P_i \oplus c_{i-1}} \quad (3)$$

Repeatedly applying the recursive equat.(2) one gets the following set of carry equations in terms of the G_i's, the P_i's and the initial carry input c_{-1}:

$$\boxed{\begin{aligned}
c_0 &= G_0 + c_{-1} \cdot P_0 \\
c_1 &= G_1 + c_0 \cdot P_1 = G_1 + G_0 \cdot P_1 + c_{-1} \cdot P_0 \cdot P_1 \\
c_2 &= G_2 + c_1 \cdot P_2 = G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + c_{-1} \cdot P_0 \cdot P_1 \cdot P_2 \\
c_3 &= G_3 + c_2 \cdot P_3 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 \\
&\qquad + c_{-1} \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3 \\
&\vdots \qquad\qquad\qquad\qquad \vdots \\
c_{n-1} &= G_{n-1} + G_{n-2} \cdot P_{n-1} + G_{n-3} \cdot P_{n-2} \cdot P_{n-1} + \cdots + G_0 \cdot P_1 \cdot P_2 \cdots P_{n-1} \\
&\qquad + c_{-1} \cdot P_0 \cdot P_1 \cdot P_2 \cdot \cdots \cdot P_{n-1}
\end{aligned}} \qquad (4)$$

④ a

The previous derivations form the basis for the design of the carry lookahead (CLA) adder. The block diagram below shows a carry lookahead (CLA) adder.

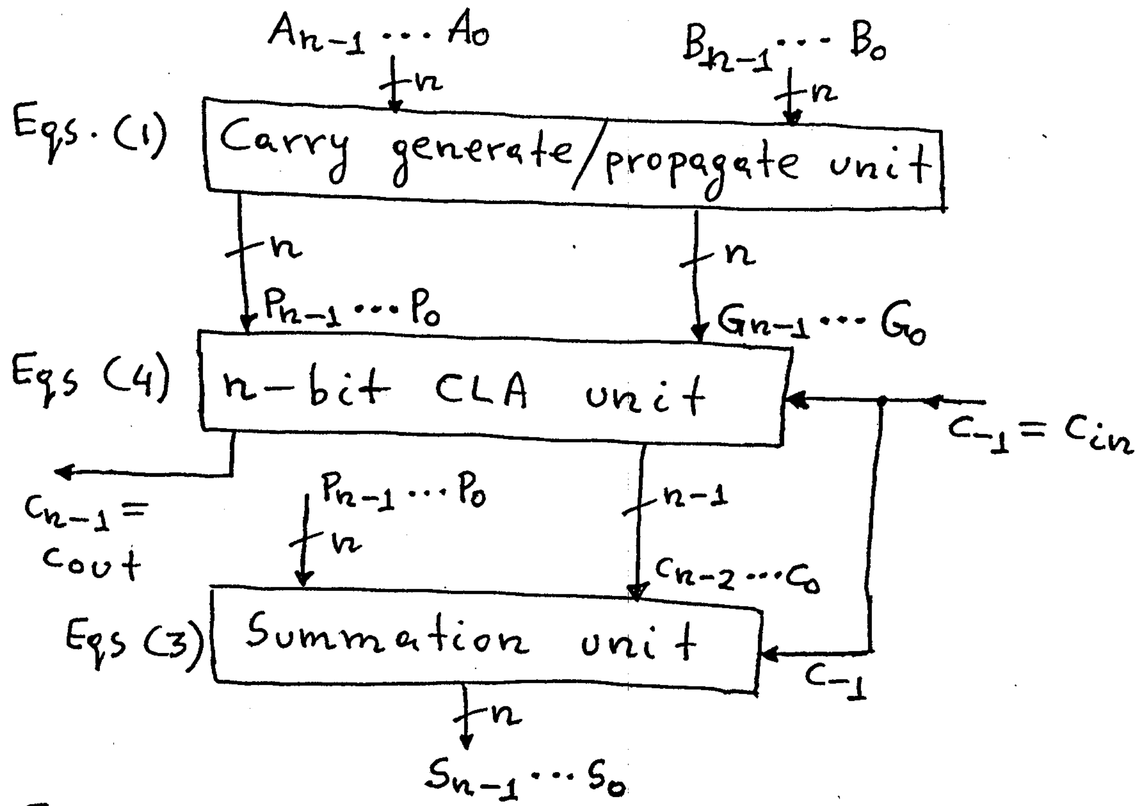


Fig 1: The block diagram of a CLA adder.

If

D_1 = propagation delay through the carry generate/propagate unit;

D_2 = propagation delay through the CLA unit;

D_3 = propagation delay through the summation unit

then the propagation delay through the above CLA adder is $D_1 + D_2 + D_3$.

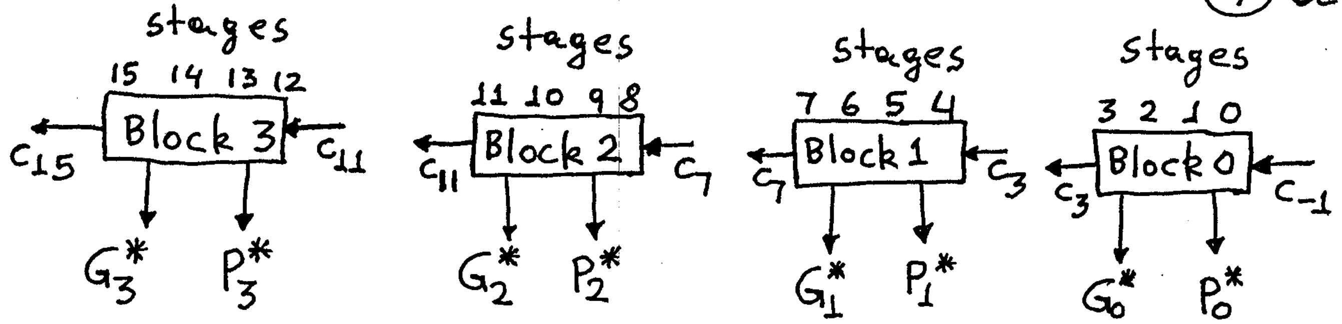
There is a major problem when the n-bit CLA unit of figure 1 relies on implementing the set of equations (4). The problem is that as n becomes large, the number of inputs to the high-order gates in the CLA logic also becomes large. From equations (4), it can be seen that C_{n-1} (the carry out) relies on AND/OR logic, with the largest AND and OR gates requiring $n+1$ inputs. For large values of n, current technologies might not supply logic gates with such a large number of inputs.

The above problem can be solved by partitioning the adder into blocks. The situation is clarified by the presentation offered below.

- <u>Block generate/block propagate</u>

Consider, for example, a 16-bit addition decomposed into four 4-bit blocks as shown below.

$\longrightarrow$ go to next page



The above figure does not reflect any implementation but just shows the decomposition of the 16-bit addition into four blocks.

The new functions $G_0^*, G_1^*, G_2^*, G_3^*$ are the "block generate" functions of blocks 0, 1, 2, 3 respectively. The functions $P_0^*, P_1^*, P_2^*, P_3^*$ are the "block propagate" functions of blocks 0, 1, 2, 3.

- The function G_i^* is true when the carry out of the ith block is generated within the ith block itself.
- The function P_i^* is true when the carry in to the ith block is propagated through the entire ith block.

Expressions for $G_0^*, P_0^*, G_1^*, P_1^*$ follow:

$$G_0^* = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3$$

$$P_0^* = P_0 \cdot P_1 \cdot P_2 \cdot P_3$$

$$G_1^* = G_7 + G_6 \cdot P_7 + G_5 \cdot P_6 \cdot P_7 + G_4 \cdot P_5 \cdot P_6 \cdot P_7$$

$$P_1^* = P_4 \cdot P_5 \cdot P_6 \cdot P_7$$

Similar expressions hold true for G_2^*, P_2^*, G_3^*, P_3^*.

Important Observation: As seen from the above expressions, the block generates G_i^*'s and block propagates P_i^*'s are functions of only the bit-level generate and propagate functions but do not depend on any carry-in.

Expressions for the carry outputs c_3, c_7, c_{11}, c_{15} follow. The carry outputs are expressed as functions of the block generate and propagate functions as well as c_{-1} (the carry-in).

$$c_3 = G_0^* + c_{-1} \cdot P_0^*$$

$$c_7 = G_1^* + G_0^* \cdot P_1^* + c_{-1} \cdot P_0^* \cdot P_1^*$$

$$c_{11} = G_2^* + G_1^* \cdot P_2^* + G_0^* \cdot P_1^* \cdot P_2^* + c_{-1} \cdot P_0^* \cdot P_1^* \cdot P_2^*$$

$$c_{15} = G_3^* + G_2^* \cdot P_3^* + G_1^* \cdot P_2^* \cdot P_3^* + G_0^* \cdot P_1^* \cdot P_2^* \cdot P_3^* + c_{-1} \cdot P_0^* \cdot P_1^* \cdot P_2^* \cdot P_3^*$$

The previous discussions form the basis for the two-level carry lookahead (CLA) adder. A 32-bit two-level CLA adder is shown by figure 2 on the next page.

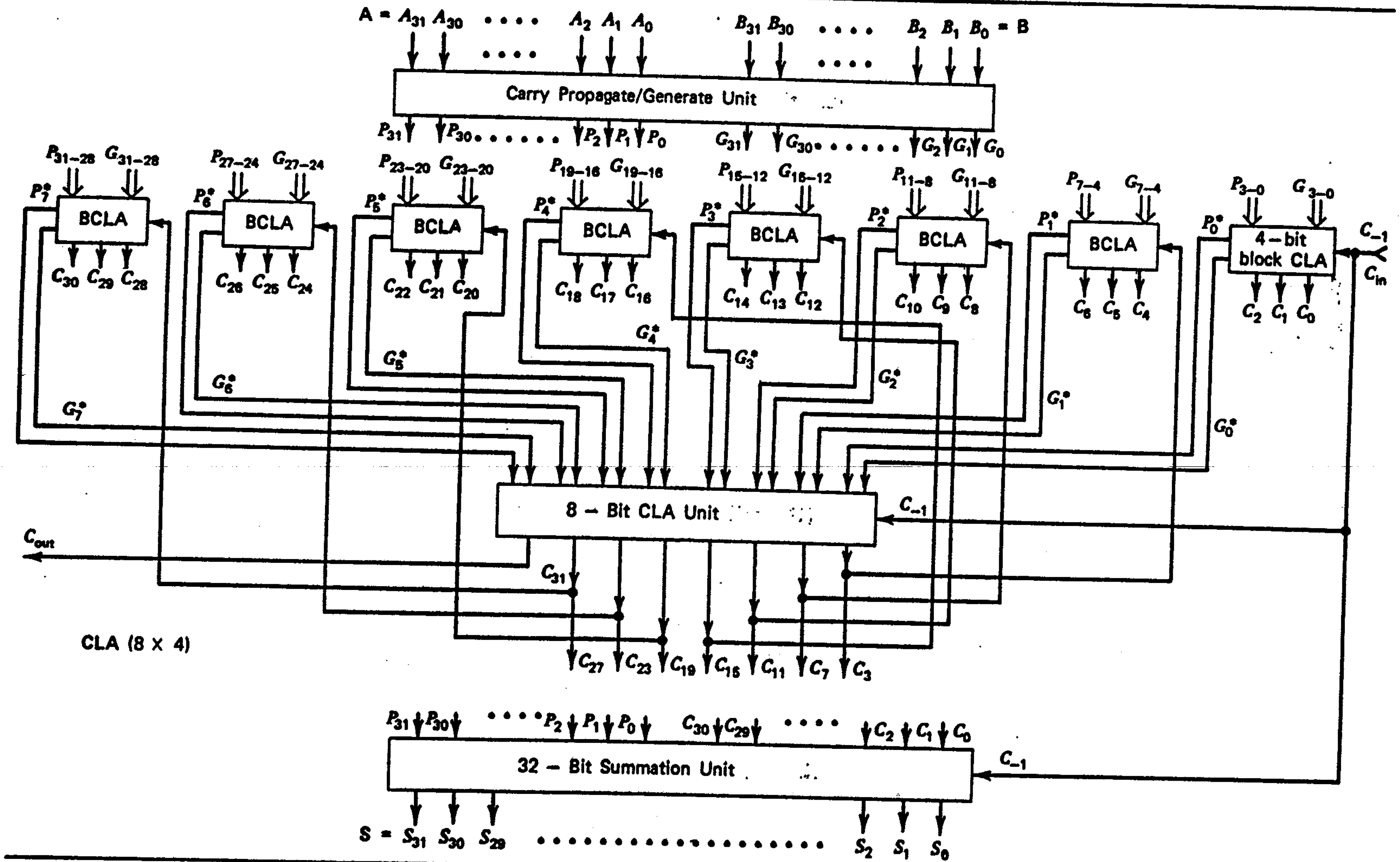

A = A_{31} A_{30} .... A_2 A_1 A_0
B_{31} B_{30} .... B_2 B_1 B_0 = B
Carry Propagate/Generate Unit
P_{31} P_{30} ...... P_2 P_1 P_0
G_{31} G_{30} ...... G_2 G_1 G_0
P_{31-28} G_{31-28} P_7^* BCLA C_{30} C_{29} C_{28} G_7^*
P_{27-24} G_{27-24} P_6^* BCLA C_{26} C_{25} C_{24} G_6^*
P_{23-20} G_{23-20} P_5^* BCLA C_{22} C_{21} C_{20} G_5^*
P_{19-16} G_{19-16} P_4^* BCLA C_{18} C_{17} C_{16} G_4^*
P_{15-12} G_{15-12} P_3^* BCLA C_{14} C_{13} C_{12} G_3^*
P_{11-8} G_{11-8} P_2^* BCLA C_{10} C_9 C_8 G_2^*
P_{7-4} G_{7-4} P_1^* BCLA C_6 C_5 C_4 G_1^*
P_{3-0} G_{3-0} P_0^* 4-bit block CLA C_2 C_1 C_0 G_0^*
C_{-1} C_{in}
8 - Bit CLA Unit
C_{-1}
C_{out}
C_{31}
CLA (8 × 4)
C_{27} C_{23} C_{19} C_{15} C_{11} C_7 C_3
P_{31} P_{30} .... P_2 P_1 P_0 C_{30} C_{29} .... C_2 C_1 C_0
32 - Bit Summation Unit
C_{-1}
S = S_{31} S_{30} S_{29} .... S_2 S_1 S_0

Figure 2 Two-level carry lookahead adder with 32-bit word length arranged in an 8-by-4 configuration.

9 a

Explanation of figure 2

Let D_1, D_2, D_3, D_4 be:

D_1 = worst case propagation delay through the carry generate/propagate unit.

D_2 = worst case propagation delay through a 4-bit BCLA unit.

D_3 = worst case propagation delay through the 8-bit CLA unit.

D_4 = worst case propagation delay through the summation unit.

- The inputs $A = A_{31} \cdots A_0$, $B = B_{31} \cdots B_0$ and $c_{-1} = c_{in}$ are available at time 0 (zero).
- The bit-level generates and propagates $G_0, \cdots, G_{31}, P_0, \cdots, P_{31}$ are available at time D_1.
- The block generates and block propagates $G_0^*, \cdots, G_7^*, P_0^*, \cdots, P_7^*$ are available at time $D_1 + D_2$. Also, the correct c_0, c_1, c_2 are available at time $D_1 + D_2$.
- The carry outputs $c_3, c_7, c_{11}, c_{15}, c_{19}, c_{23}, c_{27}, c_{31} = c_{out}$ are available at time $D_1 + D_2 + D_3$.

- The correct $c_4, c_5, c_6, c_8, c_9, c_{10}, c_{12}, c_{13}, c_{14}, c_{16}, c_{17}, c_{18}, c_{20}, c_{21}, c_{22}, c_{24}, c_{25}, c_{26}, c_{28}, c_{29}, c_{30}$ are available at time $D_1 + D_2 + D_3 + D_2$.

- The correct summation bits $s_0, s_1, \cdots, s_{31}$ are available at time $D_1 + D_2 + D_3 + D_2 + D_4$.

Thus, the worst case propagation delay through the entire 32-bit CLA adder of figure 2 is $D_1 + 2D_2 + D_3 + D_4$.

<u>Question 1:</u> Write the equations by which the carry generate/propagate unit computes G_{10} and P_7.

<u>Answer:</u> $G_{10} = A_{10} \cdot B_{10}$; $P_7 = A_7 \oplus B_7$.

<u>Question 2:</u> Write the equations by which the appropriate BCLA unit computes P_3^* and G_3^*.

<u>Answer:</u> $P_3^* = P_{12} \cdot P_{13} \cdot P_{14} \cdot P_{15}$

$$G_3^* = G_{15} + G_{14} \cdot P_{15} + G_{13} \cdot P_{14} \cdot P_{15} + G_{12} \cdot P_{13} \cdot P_{14} \cdot P_{15}$$

Question 3: Write the equation by which the 8-bit CLA unit computes c_{15}

Answer:

$$c_{15} = G_3^* + G_2^* \cdot P_3^* + G_1^* \cdot P_2^* \cdot P_3^* + G_0^* \cdot P_1^* \cdot P_2^* \cdot P_3^* + c_{-1} \cdot P_0^* \cdot P_1^* \cdot P_2^* \cdot P_3^*$$

Question 4: Write the equation by which the appropriate BCLA unit computes c_{22}.

Answer:

$$c_{22} = G_{22} + G_{21} \cdot P_{22} + G_{20} \cdot P_{21} \cdot P_{22} + c_{19} \cdot P_{20} \cdot P_{21} \cdot P_{22}$$

Question 5: Write the equation by which the summation unit computes s_{12}.

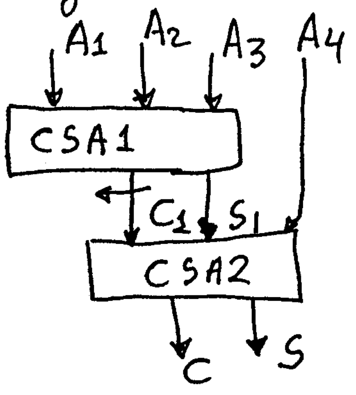
Answer: $s_{12} = P_{12} \oplus c_{11}$

EE 7715

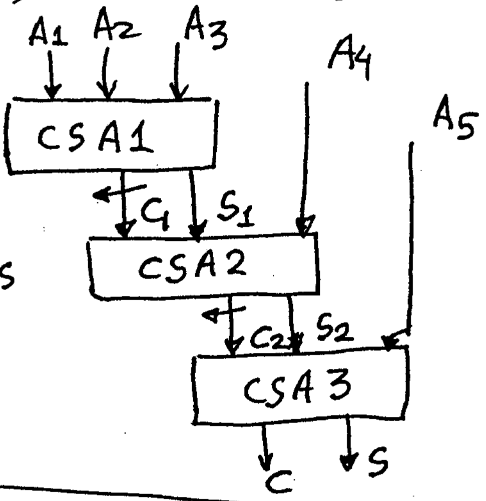
Minimum delay CSA trees

The figures below show several minimum delay CSA trees for adding k numbers and reducing them down to two; (S vector, C vector)

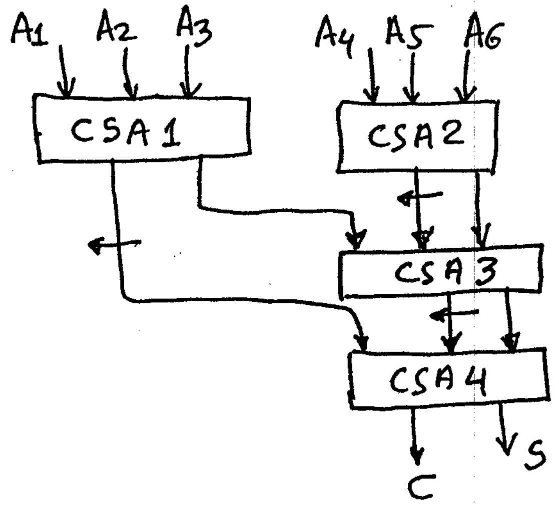
k = 4
2 CSAs
2 levels



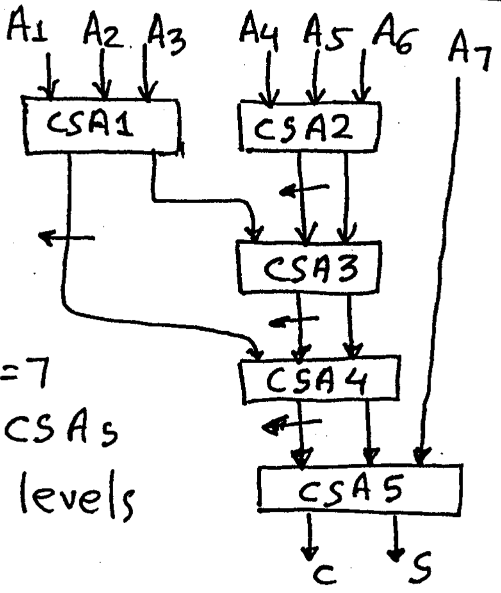
k=5
3 CSAs
3 levels



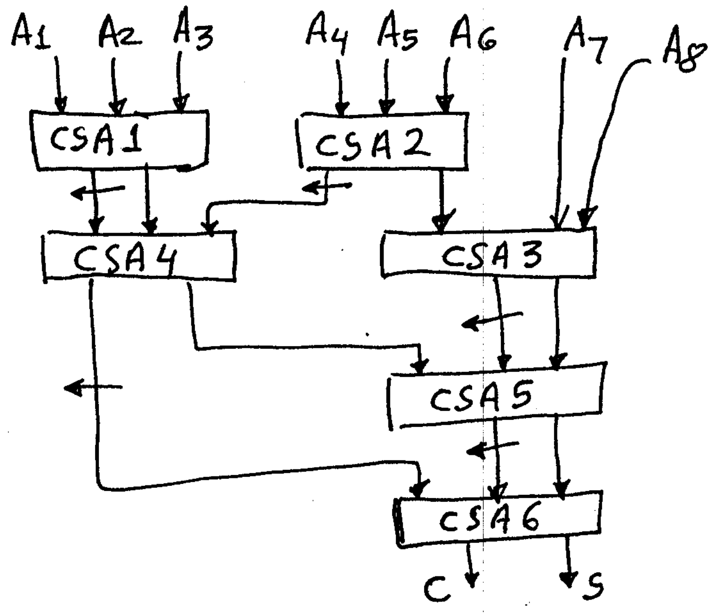
k=6
4 CSAs
3 levels



k=7
5 CSAs
4 levels



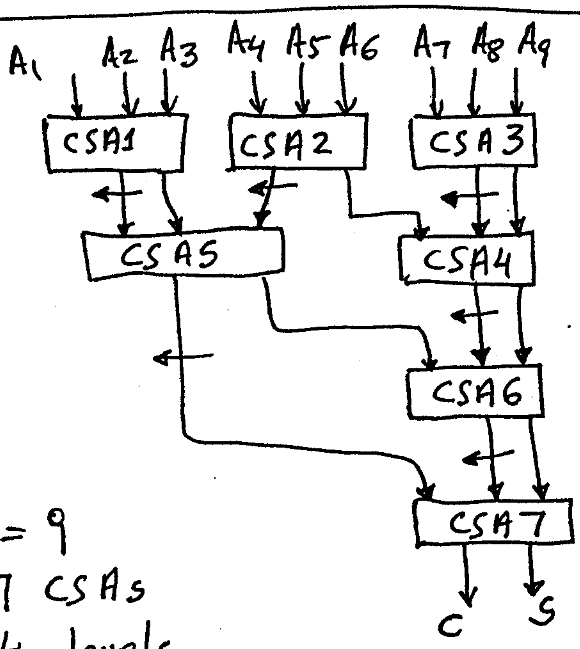
② b



k=8

6 CSAs

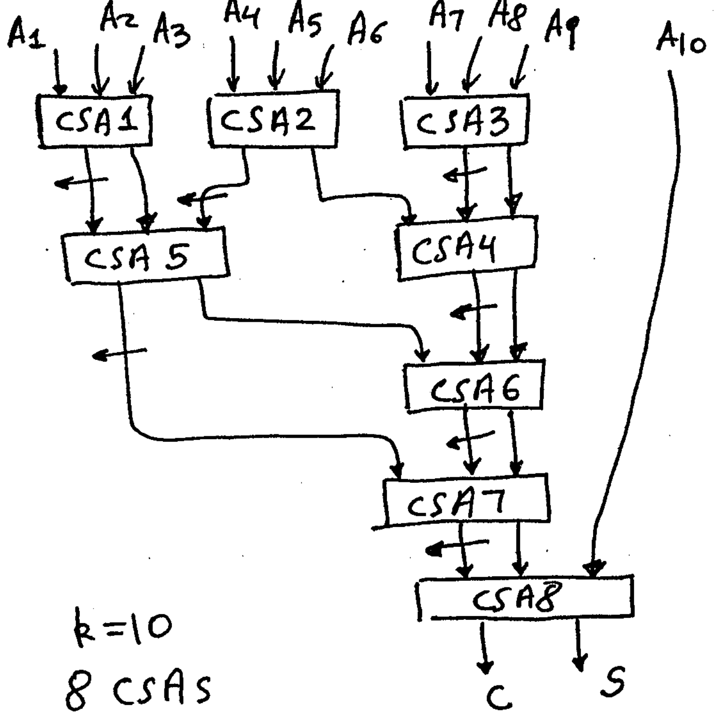
4 levels



k = 9

7 CSAs

4 levels



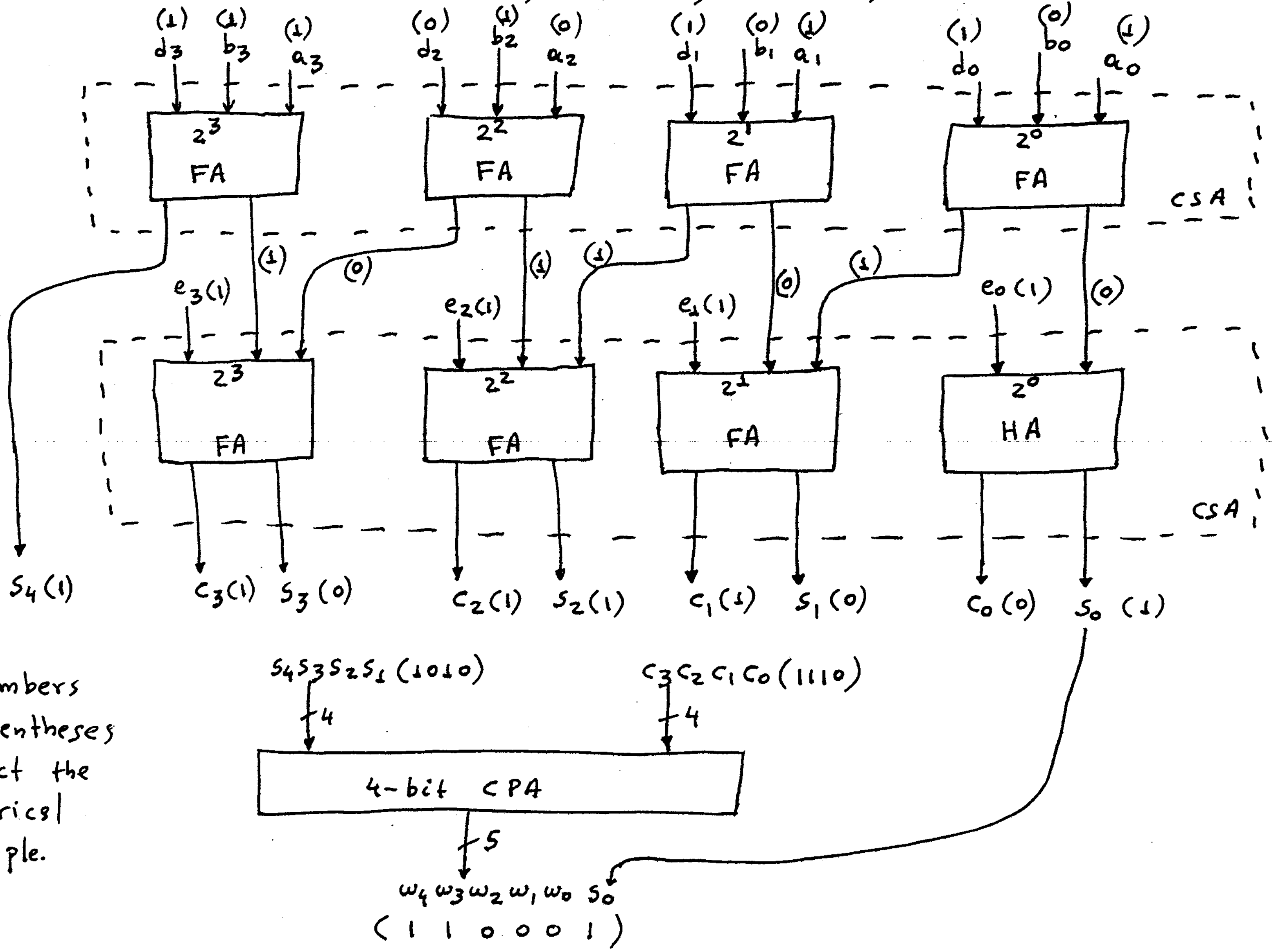
k=10

8 CSAs

5 levels

1 U

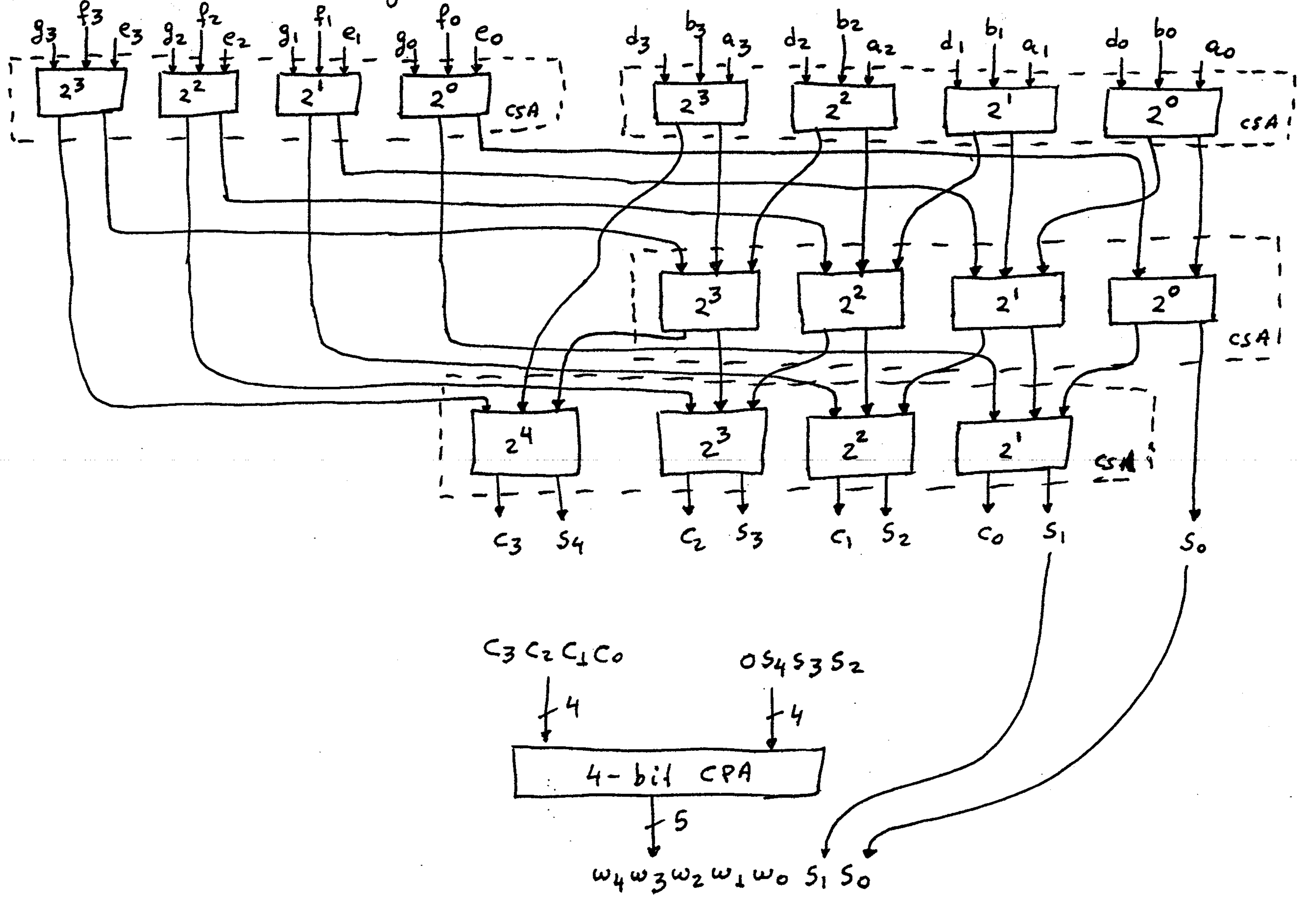
Using a CSA tree and a CPA add four 4-bit numbers $A=(a_3a_2a_1a_0)_2$; $B=(b_3b_2b_1b_0)_2$; $D=(d_3d_2d_1d_0)_2$; $E=(e_3e_2e_1e_0)_2$. On your design show the numerical example with A = 1011; B = 1100; D = 1011; E = 1111. (Show detailed design).



* * Numbers in parentheses reflect the numerical example.

② c

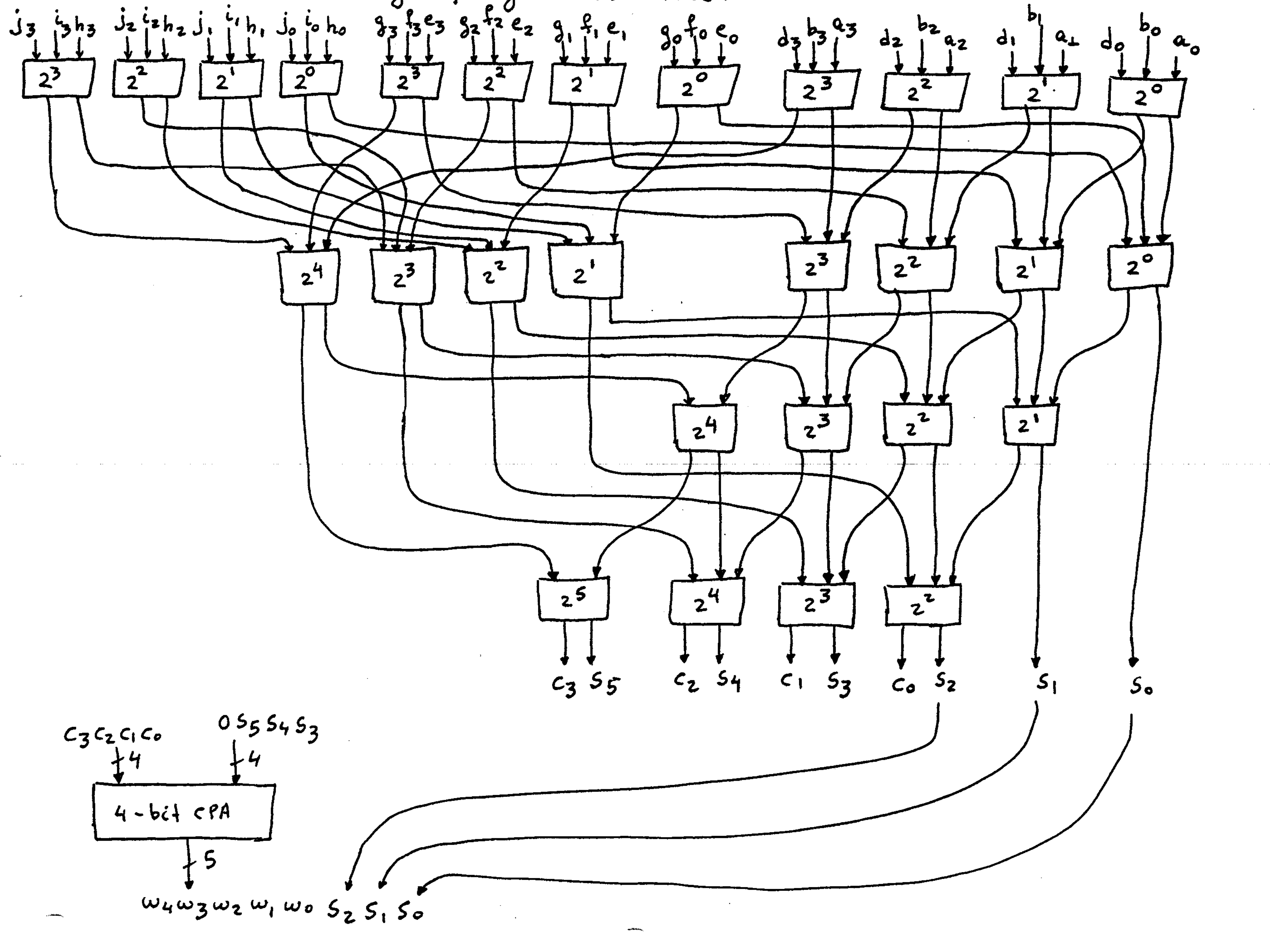
Using a CSA tree and a CPA add six 4-bit numbers $A=(a_3a_2a_1a_0)_2$; $B=(b_3b_2b_1b_0)_2$; $D=(d_3d_2d_1d_0)_2$; $E=(e_3e_2e_1e_0)_2$; $F=(f_3f_2f_1f_0)_2$; $G=(g_3g_2g_1g_0)_2$.
Show detailed design for your CSA tree.



g_3 f_3 e_3 g_2 f_2 e_2 g_1 f_1 e_1 g_0 f_0 e_0
d_3 b_3 a_3 d_2 b_2 a_2 d_1 b_1 a_1 d_0 b_0 a_0
2^3 2^2 2^1 2^0
CSA
CSA
2^3 2^2 2^1 2^0
CSA
2^4 2^3 2^2 2^1
CSA
c_3 s_4 c_2 s_3 c_1 s_2 c_0 s_1 s_0
$c_3c_2c_1c_0$
$0s_4s_3s_2$
4
4
4-bit CPA
5
$w_4w_3w_2w_1w_0\ s_1s_0$

3 c

Using a CSA tree and a CPA add nine 4-bit numbers A; B; D; E; F; G; H; I; J. Show detailed design for your CSA tree.



$j_3\ i_3\ h_3$
$j_2\ i_2\ h_2$
$j_1\ i_1\ h_1$
$j_0\ i_0\ h_0$
$g_3\ f_3\ e_3$
$g_2\ f_2\ e_2$
$g_1\ f_1\ e_1$
$g_0\ f_0\ e_0$
$d_3\ b_3\ a_3$
$d_2\ b_2\ a_2$
$d_1\ b_1\ a_1$
$d_0\ b_0\ a_0$
2^3
2^2
2^1
2^0
2^4
2^5
$c_3\ s_5$
$c_2\ s_4$
$c_1\ s_3$
$c_0\ s_2$
s_1
s_0
$c_3 c_2 c_1 c_0$
$0 s_5 s_4 s_3$
4
4-bit CPA
5
$w_4 w_3 w_2 w_1 w_0\ s_2\ s_1\ s_0$