Carry Save Adder

The carry save adder seems to be the most useful adder for our application. It is simply a parallel ensemble of k full-adders without any horizontal connection. Its main function is to add three k-bit integers A, B, and C to produce two integers C' and S such that

$$C' + S = A + B + C \ .$$

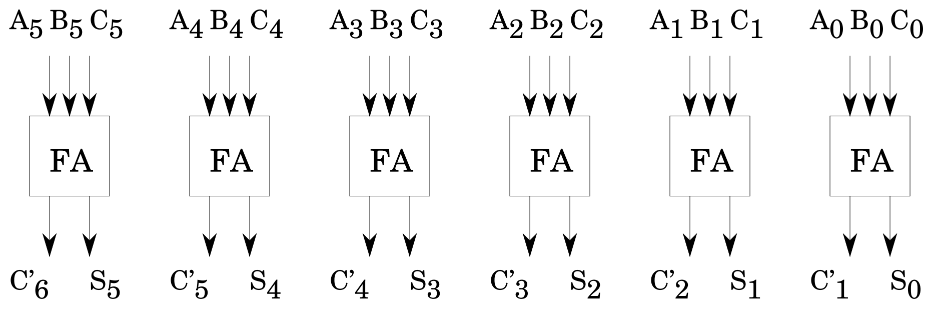
As an example, let $A = 40$, $B = 25$, and $C = 20$, we compute S and C' as shown below:

$$\begin{array}{rclcccccc}
A = 40 & = & & 1 & 0 & 1 & 0 & 0 & 0 \\
B = 25 & = & & 0 & 1 & 1 & 0 & 0 & 1 \\
C = 20 & = & & 0 & 1 & 0 & 1 & 0 & 0 \\
\hline
S = 37 & = & & 1 & 0 & 0 & 1 & 0 & 1 \\
C' = 48 & = & 0 & 1 & 1 & 0 & 0 & 0 & \\
\hline
\end{array}$$

The ith bit of the sum S_i and the $(i+1)$st bit of the carry C'_{i+1} is calculated using the equations

$$\begin{aligned}
S_i &= A_i \oplus B_i \oplus C_i \ . \\
C'_{i+1} &= A_iB_i + A_iC_i + B_iC_i \ ,
\end{aligned}$$

in other words, a carry save adder cell is just a full-adder cell. A carry save adder, sometimes named a one-level CSA, is illustrated below for $k = 6$.



Since the input vectors A, B, and C are applied in parallel, the total delay of a carry save adder is equal to the total delay of a single FA cell. Thus, the addition of three integers to compute two integers requires a single FA delay. Furthermore, the CSA requires only k times the areas of FA cell, and scales up very easily by adding more parallel cells. The subtraction operation can also be performed by using 2's complement encoding. There are basically two disadvantages of the carry save adders:

- It does not really solve our problem of adding two integers and producing a single output. Instead, it adds three integers and produces two such that sum of these two is equal to the sum of three inputs. This method may not be suitable for application which only needs the regular addition.
- The sign detection is hard: When a number is represented as a carry-save pair (C, S) such that its actual value is $C + S$, we may not know the exact sign of total sum $C + S$. Unless the addition is performed in full length, the correct sign may never be determined.

We will explore this sign detection problem in an upcoming section in more detail. For now, it suffices to briefly mention the sign detection problem, and introduce a method of sign detection. This method is based on adding a few of the most significant bits of C and S in order to calculate (estimate) the sign. As an example, let $A = -18$, $B = 19$, $C = 6$. After the carry save addition process, we produce $S = -5$ and $C' = 12$, as shown below. Since the total sum $C' + S = 12 - 5 = 7$, its correct sign is 0. However, when we add the first most significant bits, we estimate the sign incorrectly.

A	=	-18	=		1	0	1	1	1	0	
B	=	19	=		0	1	0	0	1	1	
C	=	6	=		0	0	0	1	1	0	
S	=	-5	=		1	1	1	0	1	1	
C'	=	12	=	0	0	0	1	1	0		
					1						(1 MSB)
					1	1					(2 MSB)
					0	0	0				(3 MSB)
					0	0	0	1			(4 MSB)
					0	0	0	1	1		(5 MSB)
					0	0	0	1	1	1	(6 MSB)

The correct sign is computed only after adding the first three most significant bits. In the worst case, up to a full length addition may be required to calculate the correct sign.

Carry Delayed Adder

The carry delayed adder is a two-level carry save adder. As we will see in Section 7.3, a certain property of the carry delayed adder can be used to reduce the multiplication complexity. The carry delayed adder produced a pair of integers (D, T), called a carry delayed number, using the following set of equations:

$$\begin{aligned} S_i &= A_i \oplus B_i \oplus C_i \ , \\ C_{i+1} &= A_iB_i + A_iC_i + B_iC_i \ , \\ T_i &= S_i \oplus C_i \ , \\ D_{i+1} &= S_iC_i \ , \end{aligned}$$

where $D_0 = 0$. Notice that C_{i+1} and S_i are the outputs of a full-adder cell with inputs A_i, B_i, and C_i, while the values D_{i+1} and T_i are the outputs of an half-adder cell.

An important property of the carry delayed adder is that $D_{i+1}T_i = 0$ for all $i = 0, 1, \ldots, k-1$. This is easily verified as

$$D_{i+1}T_i \ = \ S_iC_i(S_i \oplus C_i) \ = \ S_iC_i(\bar{S}_iC_i + S_i\bar{C}_i) \ = \ 0 \ .$$

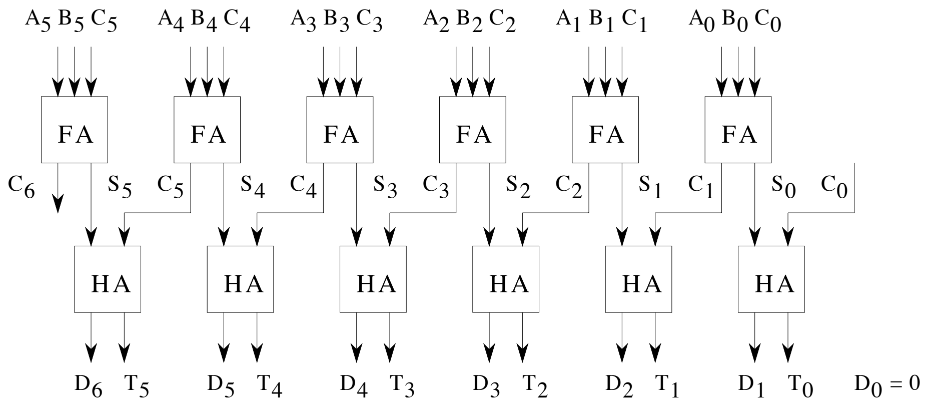
As an example, let $A = 40$, $B = 25$, and $C = 20$. In the first level, we compute the carry save pair (C, S) using the carry save equations. In the second level, we compute the carry delayed pair (D, T) using the definitions $D_{i+1} = S_iC_i$ and $T_i = S_i \oplus C_i$ as

$A = 40$	$=$		1	0	1	0	0	0
$B = 25$	$=$		0	1	1	0	0	1
$C = 20$	$=$		0	1	0	1	0	0
$S = 37$	$=$		1	0	0	1	0	1
$C = 48$	$=$	0	1	1	0	0	0	0
$T = 21$	$=$		0	1	0	1	0	1
$D = 64$	$=$	1	0	0	0	0	0	0

Thus, the carry delayed pair $(64, 21)$ represents the total of $A + B + C = 85$. The property of the carry delayed pair that $T_iD_{i+1} = 0$ for all $i = 0, 1, \ldots, k-1$ also holds.

$T = 21$	$=$		0	1	0	1	0	1
$D = 64$	$=$	1	0	0	0	0	0	0
T_iD_{i+1}	$=$		0	0	0	0	0	0

We will explore this property in Section 7.3 to design an efficient modular multiplier which was introduced by Brickell [1]. The following figure illustrates the carry delayed adder for $k = 6$.



References

[1] E. F. Brickell. A fast modular multiplication algorithm with application to two key cryptography. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *Advances in Cryptology, Proceedings of Crypto 82*, pages 51–60. Plenum Press, 1982.