About Wallace and Dadda trees

M. Martina

Wallace and Dadda trees are famous structures for adding multiple operands. In both cases the idea is to exploit the so-called partial addition by columns. In the case of multipliers the terms to be added together come from the partial product generation. The first step to design Wallace or Dadda trees is to align data so that partial terms with the same weight are in the same column.

Then, partial addition by columns resorts to counters, namely elements which are able to count the number of ones in each column. Simple circuits for counters are Half-Adders (HAs) and Full-Adders (FAs). An HA (FA) compresses two (three) bits with weigth k to one bit of weight k and one bit of weight $k+1$. As it can be oserved FAs allows for a compression ratio of at most $3/2$. As a consequence, starting from a structure with n operads several levels of HA/FA are needed to compress up to two operands. Finally, when the tree has reduced the operands to two operands, they are added together with a fast two-operand adder.

Let us concentrate on the reduction tree. The number of required levels L and the maximum number of elements at each level can be obtained as follows. The maximum number of elements at the last level (before the fast adder) is two ($l_0 = 2$). The maximum number of elements at level one is

$$l_1 = \left\lfloor \frac{3}{2} l_0 \right\rfloor = 3. \tag{1}$$

Thus, in general at level j the maximum number of elements is

$$l_j = \left\lfloor \frac{3}{2} l_{j-1} \right\rfloor. \tag{2}$$

The number of required levels can be obtained by iterating (2) and the stopping condition is $j \geq n$.

As an example consider an 8×8 bit multiplier. Partial product alignment is shown with dot notation in the left upper part of Fig. 1. The maximum number of operands to be added together in a column is $n = 8$. Through (2) we obtain the sequence:

$$l_0 = 2 \quad l_1 = 3 \quad l_2 = 4 \quad l_3 = 6 \quad l_4 = 9. \tag{3}$$

As a consequence, we need five levels to compress eight operands to two operands by the means of HAs and FAs. The maximum number of operands at each level is given in (3).

Thus, given this structure Wallace and Dadda give the rules to allocate HAs and FAs at each level. Wallace allocation is As-Soon-As-Possible (ASAP), meaning that at each level we divide dots in three-row-wide stripes and we fill each stripe with the maximum possible amount of FAs and HAs. Stripes made of only one or two operands are forwarded to the next level. The complete tree for $n = 8$ is shown in the left part of Fig. 1.

Dadda allocation is As-Late-As-Possible (ALAP), meaning that at each level j we allocate the minimum number of FAs and HAs such that the maximum number of operands at the next level $(j-1)$ is the one is (3). Thus, as show in the top right part of Fig. 1, at level $j = 4$ we have eight operands and at level $j = 3$ we must have no more than six operands ($l_3 = 6$). As a consequence, the first six columns do not require compression. The seventh column has seven elements thus with one HA we compress it to six elements. The eighth column has eight elements, but the HA in column seven added one element. Thus, we have to compress nine elements to six. One FA compresses three elements to one (reduction by 2) and one HA compresses two elements to one (reduction by 1). Similarly, the ninth column has seven elements but the eighth column added two elements (one from the FA and one from the HA). As a consequence, the ninth column must compress nine elements to six. This can be done with one FA and one HA. The same idea is extended to the other columns and to to the other levels up to $l_0 = 2$, as shown in the left part of Fig. 1.

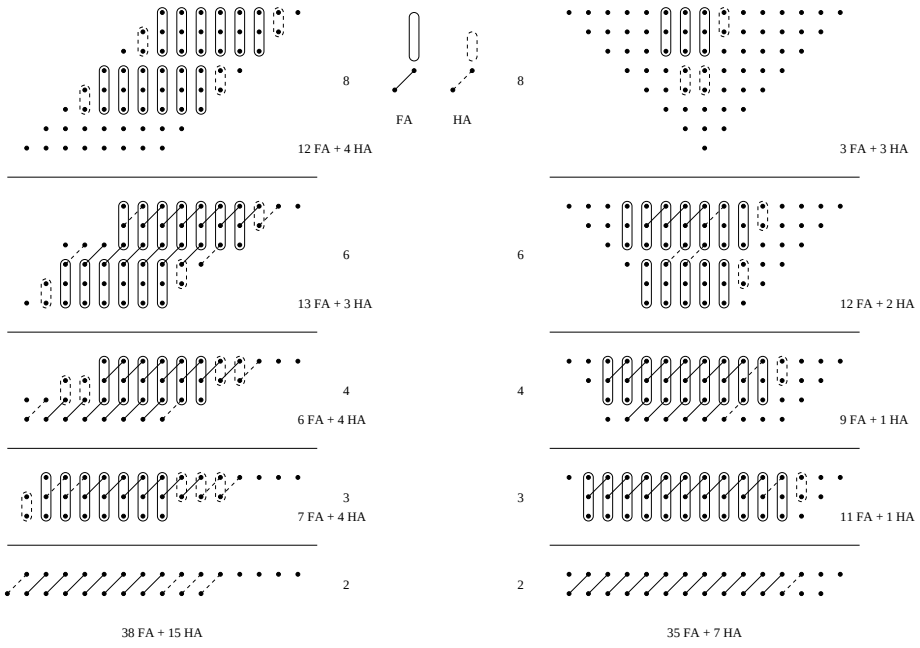

Figure 1: Wallace and Dadda trees for an 8×8 bit multiplier.