LOGIC IMPLEMENTATION USING PROGRAMMABLE DEVICES

The speed of the multiplication can be increased by using a special encoding called booth encoding of the multiplier word that reduces the number of required addition stages. Instead of traditional binary encoding the multiplier word is recoded into radix-4 scheme.

$$Y = \sum_{j=0}^{(N-1)/2} Y_j 4^j \text{ with } (Y_j \in \{-2, -1, 0, 1, 2\}$$

The radix-4 multiplier produces N/2 partial products. Each partial product is 0, Y, 2Y, or 3Y, depending on a pair of bits of X. Computing 2Y is a simple shift, but 3Y is a hard multiple requiring a slow carry propagate addition of Y + 2Y before partial product generation begins. The advantage of the recoding is that the number of partial products and hence the number of additions is halved, which results in a speed-up as well as area reduction. The only expense is somewhat more involved multiplier cell. While multiplication with {0,1} is equivalent to an AND operation, multiplying with {-2,-1,0,1,2} requires a combination of inversion and shift logic.

Here 3Y = 4Y – Y and 2Y = 4Y – 2Y. However, 4Y in a radix-4 multiplier array is equivalent to Yin the next row of the array that carries four times the weight. Hence, partial products are chosen by considering a pair of bits along with the most significant bit from the previous pair. If the most significant bit from the previous pair is true, Y must be added to the current partial product. If the most significant bit of the current pair is true, the current partial product is selected to be negative and the next partial product is incremented. Table 1 shows how the partial products are selected based on bits of the multiplier. Negative partial products are generated by taking the two’s complement of the multiplicand (possibly left-shifted by one column for –2Y). An unsigned radix-4 Booth encoded multiplier requires partial products rather than N. Each partial product is M+ 1 bits to accommodate the 2Y and –2Y multiples. Even though X and Y are unsigned, the partial products can be negative and must be sign extended properly.

Inputs			Partial Product	Booth Selects		
x_{2i+1}	x_{2i}	x_{2i-1}	PP_i	$SINGLE_i$	$DOUBLE_i$	NEG_i
0	0	0	0	0	0	0
0	0	1	Y	1	0	0
0	1	0	Y	1	0	0
0	1	1	$2Y$	0	1	0
1	0	0	$-2Y$	0	1	1
1	0	1	$-Y$	1	0	1
1	1	0	$-Y$	1	0	1
1	1	1	$-0\ (=0)$	0	0	1

Table 4.4: Radix-4 modified Booth encoding values

In a radix-4 Booth-encoded multiplier design, each group of 3 bits (a pair, along with the most significant bit of the previous pair) is encoded into several select lines(SINGLEi, DOUBLEi, and NEGi, given in the rightmost columns of Table 1) and driven across the partial product row as shown in Fig.4.4.1.

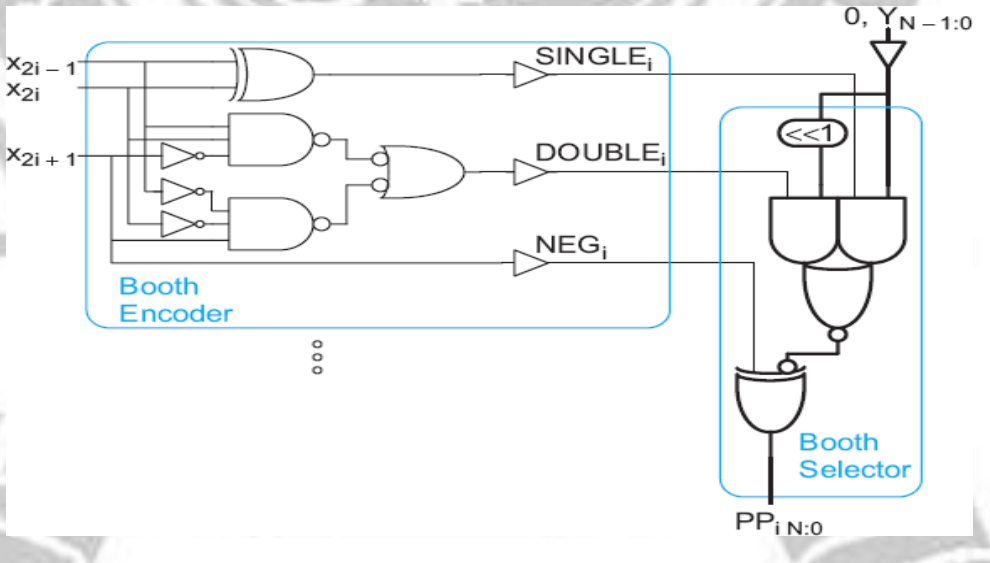

Fig.4.4.1: Radix-4 Booth encoder and selector

[Source: Wayne Wolf, —Modern VLSI Design: System On Chip]

The multiplier Y is distributed to all the rows. The select lines control Booth selectors that choose the appropriate multiple of Y for each partial product. The Booth selectors substitute for the AND gates of a simple array multiplier to determine the i^{th} partial product. Fig.4.4 (f) shows a conventional Booth encoder and selector design. Y is zero-extended to M + 1 bit. Depending on SINGLEi and DOUBLEi, the gate selects either 0, Y, or 2Y. Negative partial products should be two's-complemented (i.e., invert and add 1). If NEGi is asserted, the partial product is inverted. The extra 1 can be added in the least significant column of the next row to avoid needing a CPA.

Wallace-Tree Multiplier

The partial-sum adders can also be re-arranged in a tree-like fashion. In Fig.4.4(g) vertical slice is extracted from a generic carry-save multiplier and hence the data ripples from top to bottom similar to what happens in ripple-carry adder. The number of stages equals the number of bits in the multiplier word minus 2. Now the linear chain is translated into a tree structure as shown in Fig.4.4 (h). This topology which has an $O(\log_2 N)$ multiplication time, is called the Wallace multiplier. It is faster than the carry-save structure but has the disadvantage of being irregular. This complicates the task of coming up with a dense and efficient layout. Wallace multipliers are used only in designs where performance is critical and design time is only a secondary consideration.

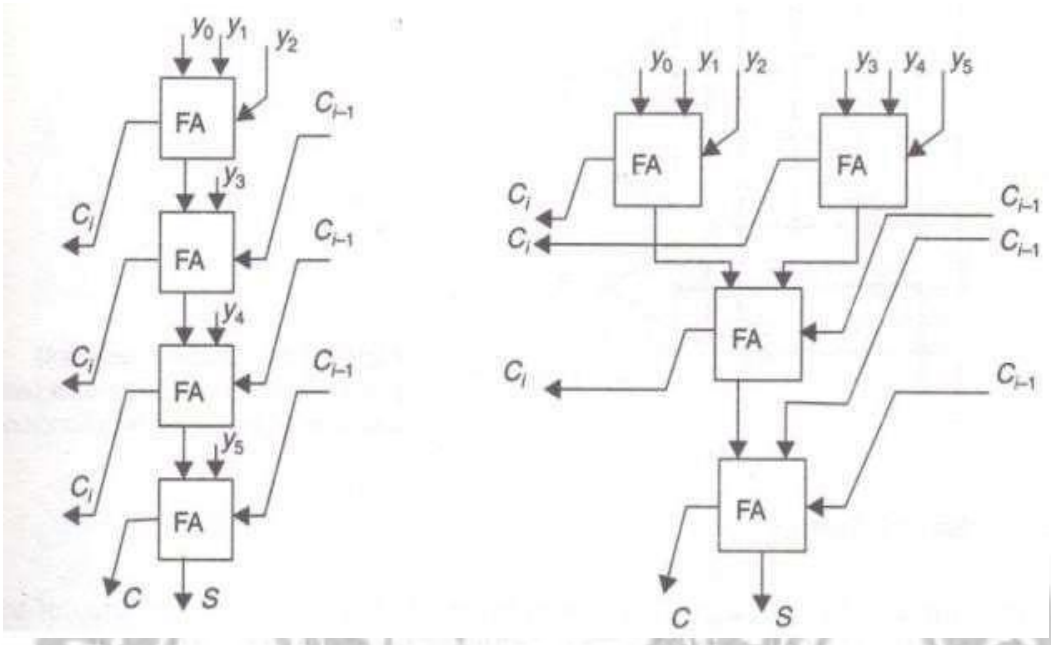

Fig.4.4.2: **IMPLEMENTATION USING PROGRAMMABLE DEVICES**

[Source: Wayne Wolf, —Modern VLSI Design: System On Chi]