Multipliers

Multiplication is less common than addition, but is still essential for microprocessors, digital signal processors, and graphics engines. The most basic form of multiplication consists of forming the product of two unsigned (positive) binary numbers. For example, the multiplication of two positive 6-bitbinary integers, 2510 and 3910, proceeds as shown in Fig.4.2.1 (a).M × N-bit multiplication P = Y × X is performed by forming N partial products of M bits each, and then summing the appropriately shifted partial products to produce an M+ N-bit result P. Binary multiplication is equivalent to a logical AND operation. Therefore, generating partial products consists of the logical ANDing of the appropriate bits of the multiplier and multiplicand. Each column of partial products is added and the carry values are passed to the next column.

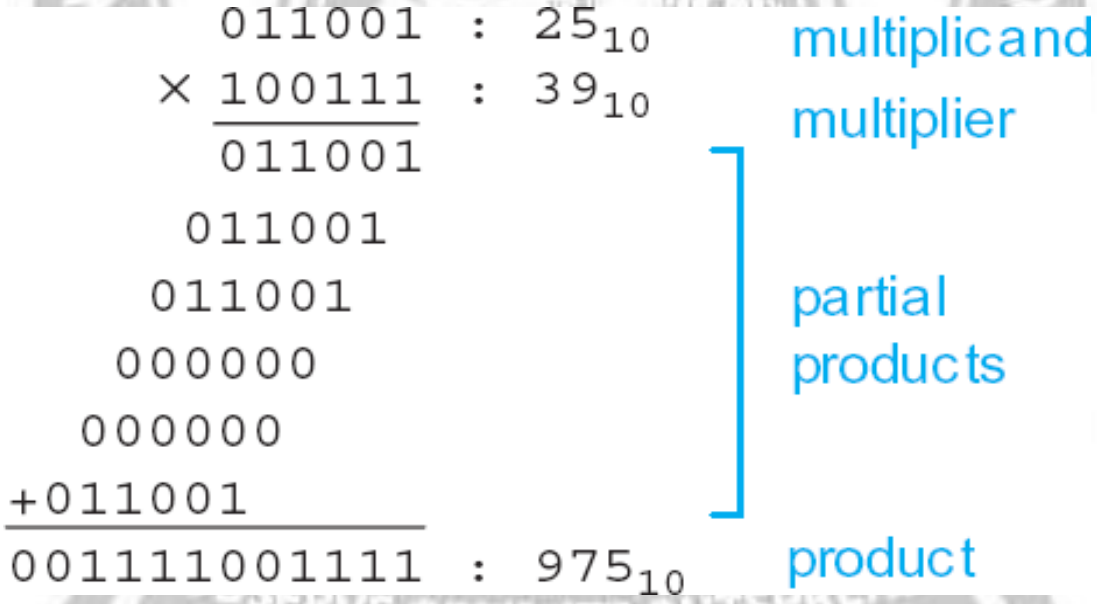

Fig.4.2.1 (a) Multiplication example

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

The multiplicand is denoted as Y = {yM — 1, yM — 2... y1, y0} and the multiplier is denoted as X = {xN — 1, xN — 2… x1, x0}. The product is given in equation (1). Fig.4.4 (b) illustrates the generation, shifting, and summing of partial products in a 6 × 6-bit multiplier. This set of operations can be mapped directly into hardware and the resulting structure is called an array multiplier.

$$P = \left(\sum_{j=0}^{M-1} y_j 2^j\right)\left(\sum_{i=0}^{N-1} x_i 2^i\right) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} x_i y_j 2^{i+j} \quad \text{—(1)}$$

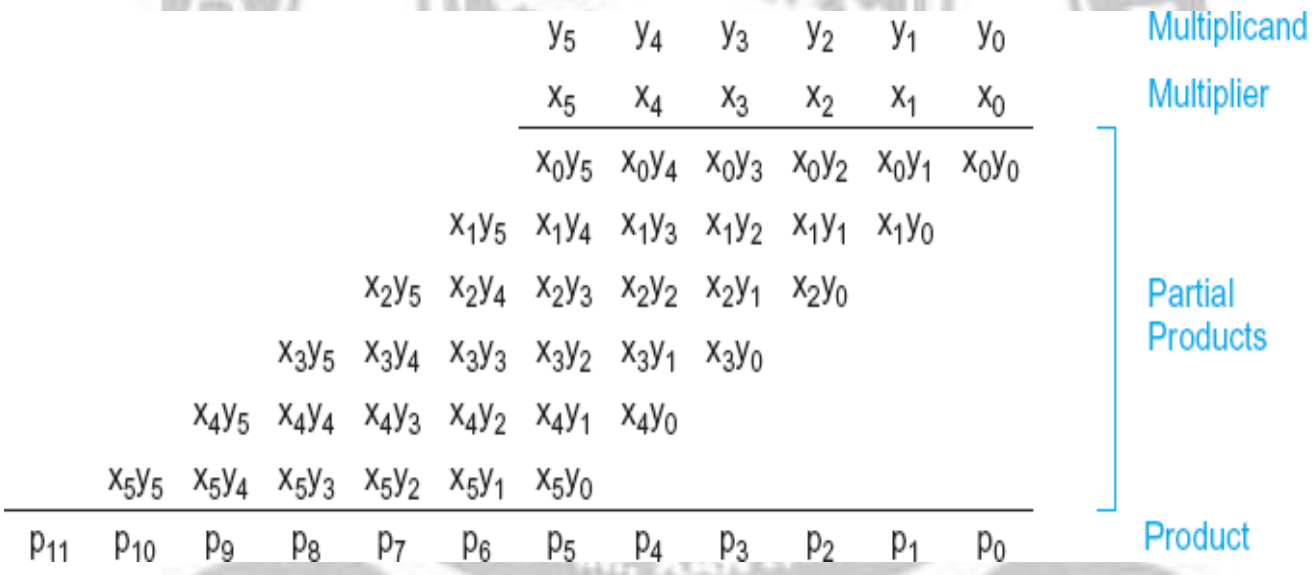

Fig.4.2.2 (b) Partial products the array Multiplier

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

The composition of an array multiplier is shown in Fig.4.4 (c). There is a one-to- one correspondence between this hardware structure and the manual multiplication in Fig.4.4 (a). The generation of partial product requires a multiplication by 1 or 0 (i.e.) AND operation. Generating the N partial products requires N M-bit AND gates. The

shifting of the partial products is performed by simple routing and does not require any active logic. The overall structure can be compacted into a rectangle, resulting in a very efficient layout.

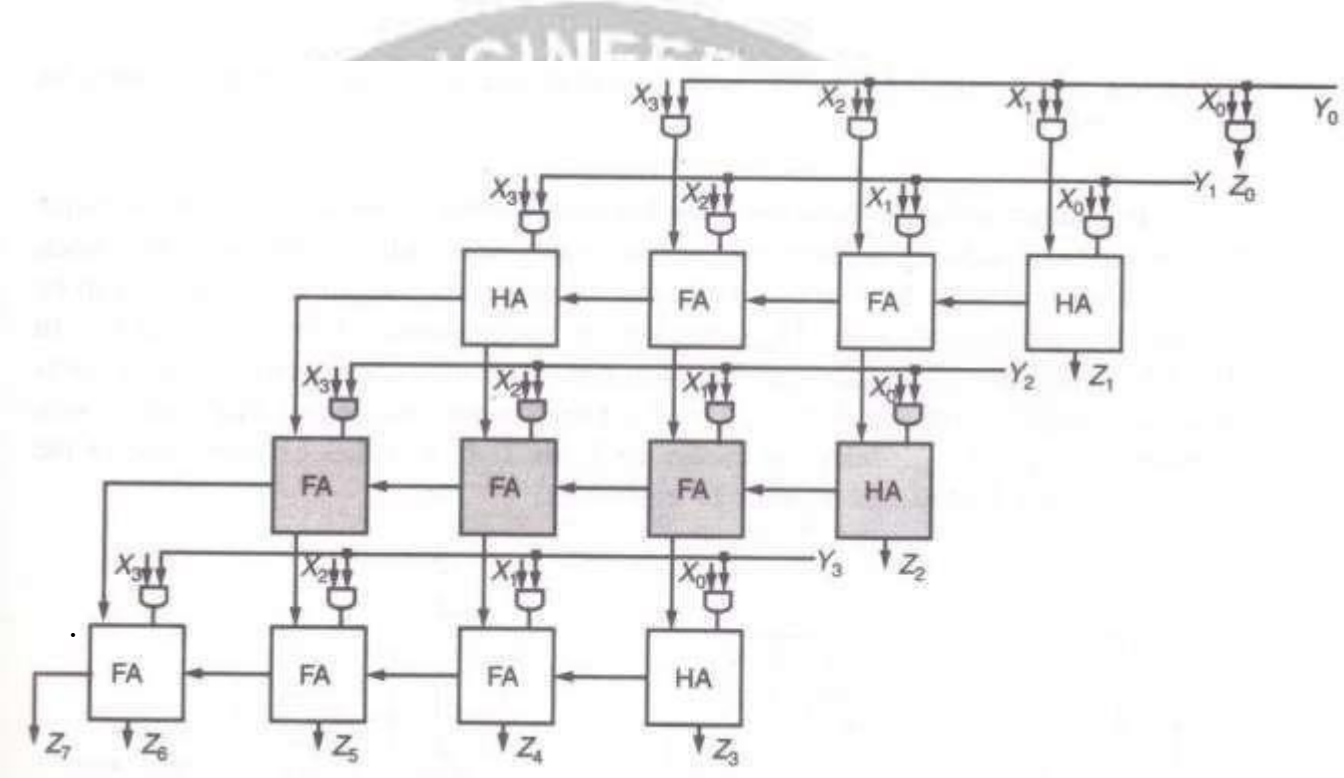

Fig.4.2.1 (c) 4x4 Multiplier for unsigned numbers

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

Due to array organization, determining the propagation delay is difficult. Partial sum adders are implemented as ripple-carry adders. Performance optimization requires critical timing path to be identified. Two such paths are highlighted in Fig.4.4 (d). The propagation delay is given as,

$$t_{mult} = [(M-1) + (N-2)]t_{carry} + (N-1)t_{sum} + t_{and} \quad \text{(2)}$$

where t_{carry} is the propagation delay between input and output carry, t_{and} is

the delay between the input carry and sum bit of the full adder and is the delay of the AND gate. t_{sum}

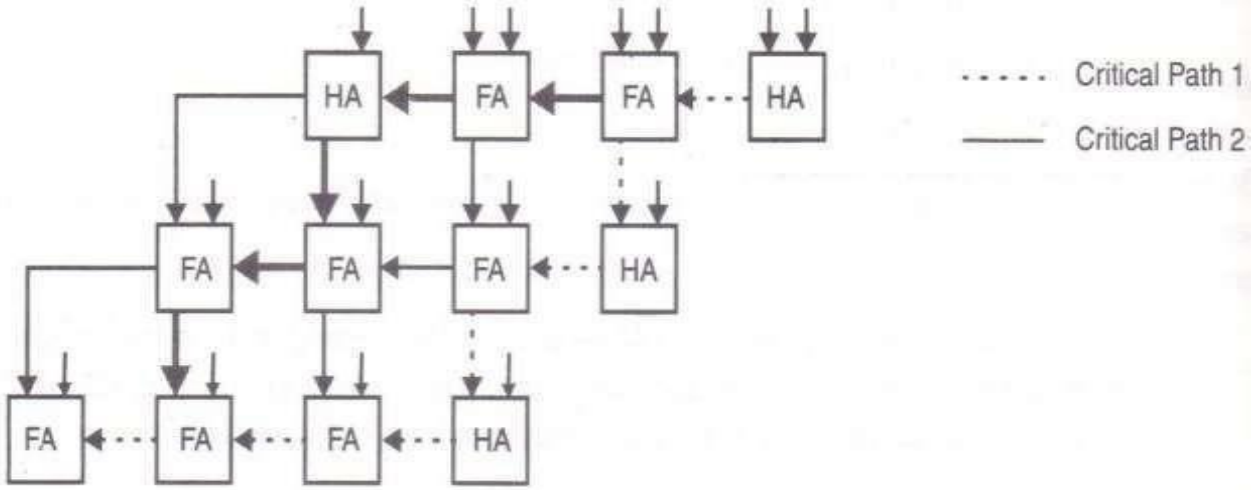

Fig.4.2.1 (d) Ripple carry based 4x4 multiplier with two critical paths highlighted

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]

Since all critical paths have the same length, speeding up one of them does not make much difference. All the critical paths have to be speeded up at the same time. From equation (2), it is deduced that the minimization of requires the minimization of both t_{carry} and t_{sum}.

Due to large number of identical critical paths, increasing the performance of the structure shown in Fig.4.4 (d) is achieved with careful transistor sizing. A more efficient multiplier structure is obtained by noticing that the multiplication result does not change when the output carry bits are passed diagonally downwards instead of to the right. An extra adder called as a vector-merging adder, is added to generate the final result. Such multiplier is called as **carry-save multiplier,** because the carry bits are not immediately

added but are rather saved for the next adder stage. This structure has a slightly increased area cost but it has the advantage that its worst-case-critical path is uniquely defined as shown in fig.4.4 (e) and expressed in equation(3).

$$t_{mult} = (N - 1)t_{carry} + t_{and} + t_{merge} \quad \text{_____(3)}$$

assuming that $t_{and} < t_{carry}$

A simple way to reduce the propagation delay of this structure is to minimize t_{merge}. This is achieved by using a fast adder implementation such as a carry-select or a lookahead structure for the merging folder.

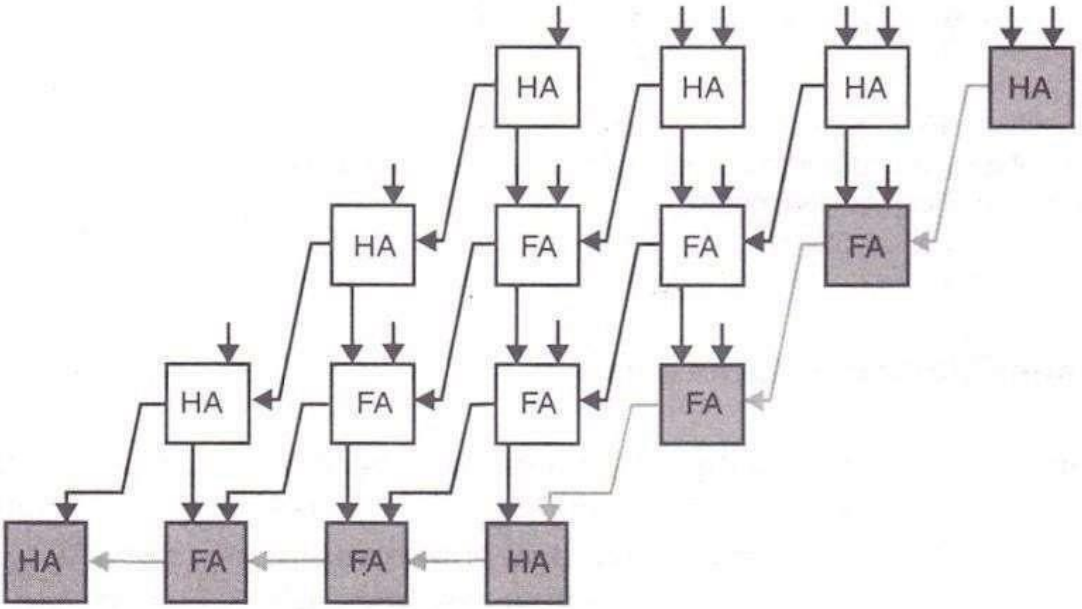

Fig.4.2.1 (e) 4x4 carry-save multiplier

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective]