EE224 Handout
Fast Multipliers

Madhav P. Desai

February 9, 2018

1 The problem

We are given two binary numbers

$$
\begin{aligned}
A &= (a_{n-1}a_{n-2}\ldots a_0) \\
&= \sum_{k=0}^{n-1} a_k.2^k \\
B &= (b_{n-1}b_{n-2}\ldots b_0) \\
&= \sum_{k=0}^{n-1} b_k.2^k
\end{aligned}
$$

and we wish to compute their product

$$
\begin{aligned}
P &= p_{2n}p_{2n-1}\ldots p_{n-1}p_{n-2}\ldots p_0 \\
&= \sum_{k=0}^{2n} p_k.2^k
\end{aligned}
$$

It is easy to see that

$$P = \sum_{k=0}^{2n-2}(\sum_{j=0}^{k} a_j.b_{k-j}).2^k$$

. Thus the direct way to do multiplication is to compute all the pair-wise terms $a_i.b_j$ and combine them to form the final product. Define the set of pair-wise products

$$\mathcal{S}_k = \{a_i.b_j |\ i+j=k\}$$

the time required to compute is proportional to 2n

three bit number can be added using the method called reduction method

we simply add the three numbers without thinking abt the carry this generates n bit no and takes constant time we then add the calculated no to the no generated by carry

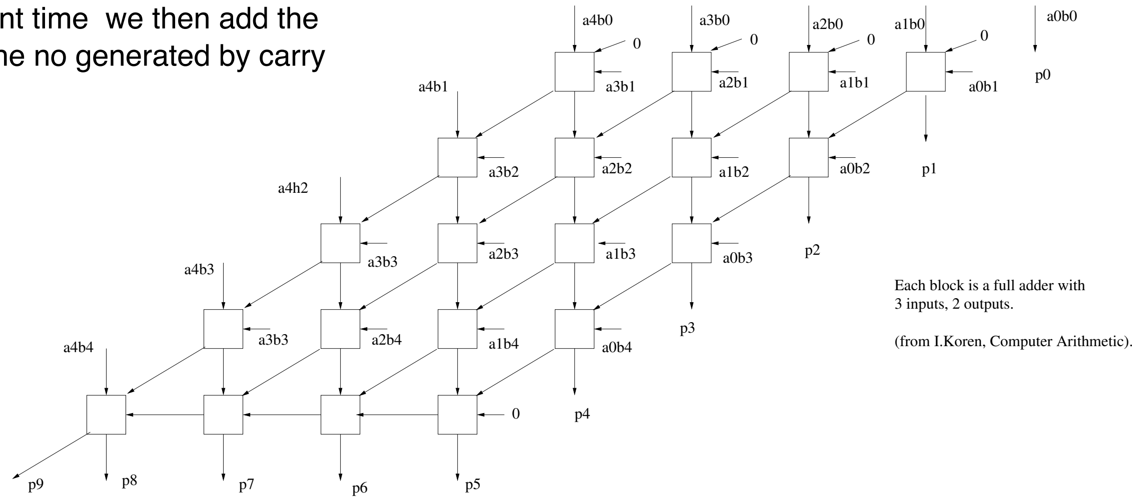

Figure 1: Array Multiplier

Note that

$$
\begin{aligned}
p_0 &= a_0.b_0 \\
p_1 &= (a_1.b_0 \oplus a_0.b_1) \\
p_2 &= (a_2.b_0 \oplus a_1.b_1 \oplus a_0.b_2) \oplus (a_1.b_0 + a_0.b_1) \\
\ldots
\end{aligned}
$$

Thus, p_k is obtained by taking the sum of bits in $\mathcal{S}_k$ together with the carries generated during the computation of p_{k-1}.

2 Array multiplier: regular structure for generating the product bits

We use a full-adder which adds three bits and produces a two-bit result. The generation of product bits can be realized by the regular structure (illustrated for $n = 5$) shown in Figure 1 [1].

This scheme is simple, but as n increases, observe that the delay of this circuit increases as $O(n)$. The number of gates required is $O(n^2)$. We need a faster solution.

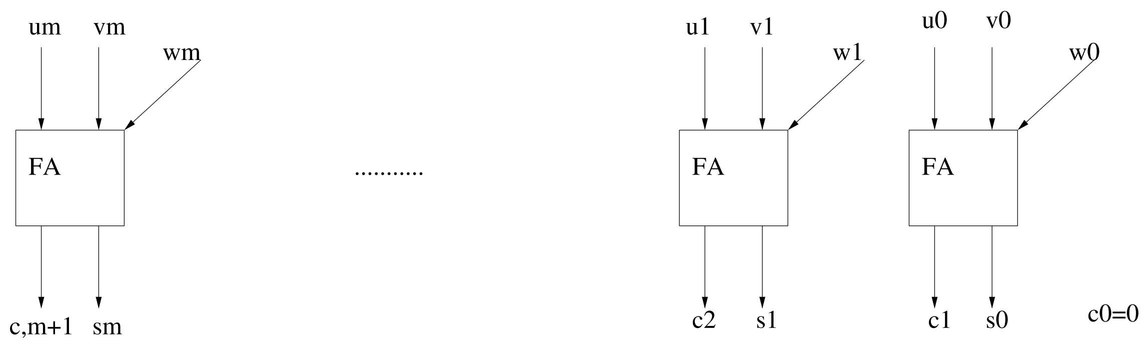

Figure 2: Carry-save adder using full-adders

3 Using carry-save addition

Suppose we wish add three $n-$bit numbers. If we do this naively, then we will need two $n-$bit adders to get the job done. But there is a much simpler way to achieve this. A carry-save adder has three inputs $U = (u_{n-1}u_{n-2}\ldots u_0)$, $V = (v_{n-1}v_{n-2}\ldots v_0)$ and $W = (w_{n-1}w_{n-2}\ldots w_0)$ and has outputs $S = (s_{n-1}s_{n-2}\ldots s_0)$ and $C = (c_{n-2}c_{n-1}\ldots c_0)$ such that

$$\begin{aligned} s_k &= u_k \oplus v_k \oplus w_k \\ c_k &= (u_{k-1} \oplus v_{k-1}).w_{k-1} + u_{k-1}.v_{k-1}, \quad (c_0 = 0) \end{aligned}$$

That is, at each k, u_k and v_k are bits to be added and w_{k-1} is treated as an incoming carry. The output c_k is to be added at position k while computing the final sum. We have

$$U + V + W = S + C$$

The carry-save adder can be constructed using full-adders (Figure 2). Observe that a carry-save adder has a constant delay and needs $O(n)$ gates.

Add S and C using a normal adder, and the result obtained will simply be the integer sum (modulo 2^n) of U, V, W. Thus, only a single conventional adder is needed.

4 Fast multiplier using carry-save addition

The basic idea is as follows: we list the bits to be added at each position and use full-adders to combine the bits. Note that the use of a full adder at position k produces a result at position k and a result at position $k+1$. Repeated use of full adders eventually gives us two bits to be added at each position. When this situation is reached, we use a conventional adder to complete the sum.

This concept is illustrated (for $n = 5$) in Figure 3 and an implementation using full-adders is illustrated in Figure 4. Note the drastic reduction in the maximum delay relative to the regular scheme used in the array multiplier.

5 Problem set

1. You are asked to design a combinational circuit which adds 5 $n-$bit numbers and produces their sum modulo 2. Using carry-save adders and a single $n-$bit conventional adder, design the fastest circuit that you can (assume that the full adder has a constant delay of 1 unit, and the conventional adder has a delay of $\log n$ units).

2. Design a circuit which has seven input bits and produces 3 output bits, with the three output bits giving a count of the number of input bits that are 1.

References

[1] I. Koren, *Computer Arithmetic Algorithms*, second edition, Universities Press, Hyderabad, 2002.

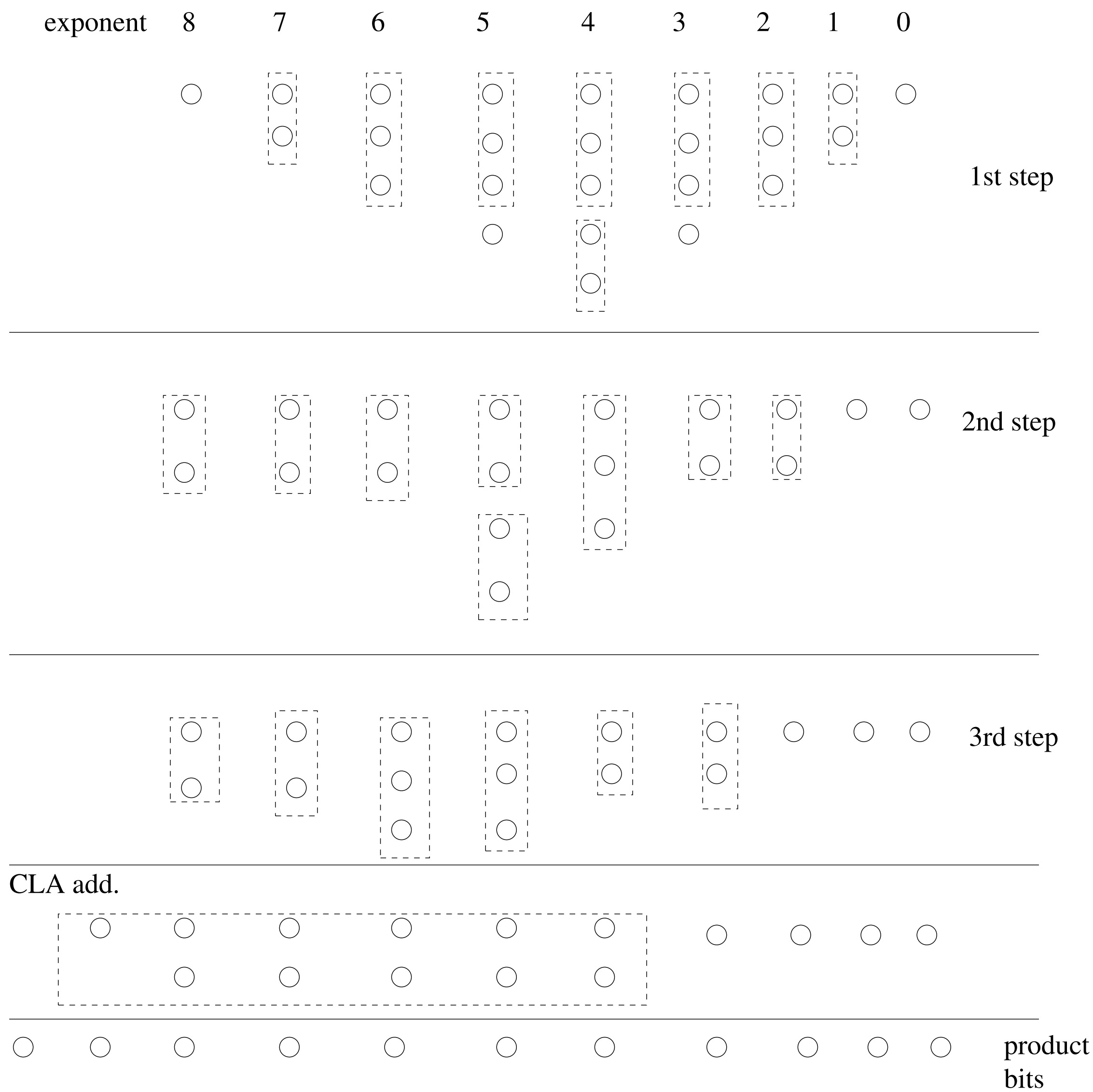

Figure 3: Reduction of bit addition using carry-save addition

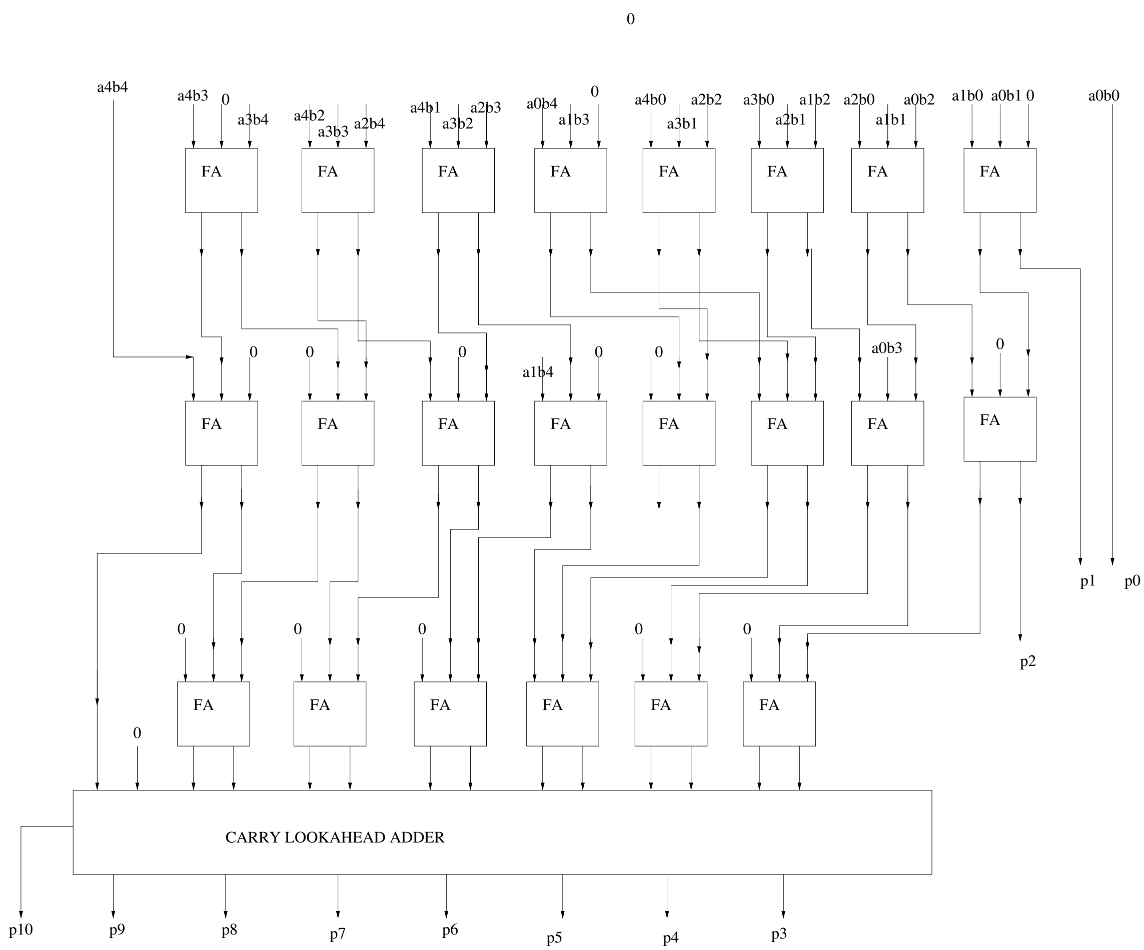

Figure 4: Implementation of bit addition using full-adders and a single conventional adder