EE224 Handout
Fast Adders

Madhav P. Desai

February 7, 2018

1 The problem

We are given two binary numbers

$$a_{n-1}a_{n-2}\ldots a_0$$
$$b_{n-1}b_{n-2}\ldots b_0$$

and we wish to compute their sum (modulo 2^n)

$$s_{n-1}s_{n-2}\ldots s_0$$

2 A simple, but slow implementation

It is easy to see that the addition algorithm can be translated to the following Boolean formulas.

$$\begin{aligned}
s_0 &= (a_0 \oplus b_0) \\
c_1 &= (a_0.b_0) \\
s_1 &= (a_1 \oplus b_1 \oplus c_1) \\
c_2 &= ((a_1 \oplus b_1).c_1 \ + \ a_1.b_1) \\
s_2 &= (a_2 \oplus b_2 \oplus c_2) \\
c_3 &= ((a_2 \oplus b_2).c_2 \ + \ a_2.b_2) \\
\ldots & \\
s_{n-1} &= (a_{n-1} \oplus b_{n-1} \oplus c_{n-1})
\end{aligned}$$

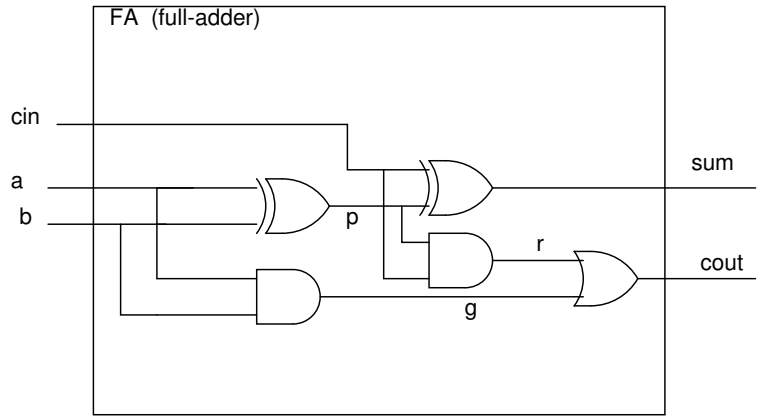

Figure 1: Full Adder

In these formulas, c_i can be thought of as the carry generated *out* of the i^{th} bit position.

We design a full-adder (FA) which has inputs a, b, c_{in} and outputs sum, c_{out} and implements the formulas

$$\begin{aligned} sum &= a \oplus b \oplus c_{in} \\ c_{out} &= (a \oplus b).c_{in} + a.b \end{aligned}$$

which can be implemented by the circuit shown in Figure 1. Note the following: The full-adder can be implemented with five logic gates, and if we assume that the delay of each gate is 1 unit, then the maximum delay through the full-adder is 3 units (the path a, p, r, c_{out}) and the minimum delay through the full-adder is 1 unit (the path c_{in}, sum).

Using the full-adder, we can construct a simple ripple-carry adder as shown in Figure 2. Note the following: An $n-$bit adder constructed in this fashion needs $5n$ logic gates, and the maximum delay through the adder is determined by the path a_0, c_1, c_2, ..., s_{n-1}, which is $n+2$ units. The cost of the adder measured in number of gates needed is $O(n)$ and the delay of the adder is also $O(n)$.

3 Faster addition

We would like to reduce the time required by an $n-$bit adder to $O(\log n)$ if possible (Why $\log n$? Why not smaller?). The chief difficulty is the propa-

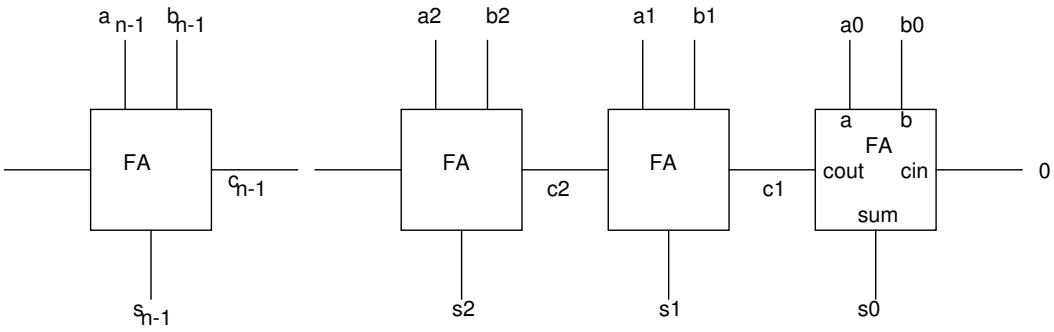

Figure 2: Ripple-carry Adder

gation of the carry from the lowest bit position to the highest bit position. Lets see if we can do this faster.

Introduce the following intermediate signals

$$\begin{aligned} p_i &= a_i \oplus b_i \\ g_i &= a_i.b_i \end{aligned}$$

It is easy to convince ourselves that

$$c_{i+1} = p_i.c_i + g_i$$

which is why the p_i is called the *propagate* signal at position i and g_i is called the *generate* signal at position i.

In terms of p_i, g_i we can then write

$$\begin{aligned} c_{i+1} &= p_i.c_i + g_i \\ &= p_i.(p_{i-1}.c_{i-1} + g_{i-1}) + g_i \\ &= (p_i.p_{i-1}).c_{i-1} + (p_i.g_{i-1} + g_i) \end{aligned}$$

Continuing to substitute for c_{i-1} until we reach c_0, we find that

$$\begin{aligned} c_{i+1} = {} & (p_i.p_{i-1}.p_{i-2}...p_0).c_0 + \\ & (g_i + p_i.g_{i-1} + p_i.p_{i-1}.g_{i-2} + \ldots p_i.p_{i-1}..p_1.g_0) \end{aligned}$$

which can be rewritten as

$$c_{i+1} = bp_i.c_0 + bg_i \tag{1}$$

where

$$\begin{aligned} bp_i &= (p_i.p_{i-1}.p_{i-2}...p_0) \\ bg_i &= (g_i + p_i.g_{i-1} + p_i.p_{i-1}.g_{i-2} + \ldots p_i.p_{i-1}..p_1.g_0) \end{aligned}$$

Thus, it follows that if we can calculate bp_i and bg_i quickly, then we can do fast addition. Using two-input gates (each with delay 1, it is easy to see that:

- bp_i can be calculated in $O(\log i)$ units of time, using $O(i)$ gates.
- bg_i can be calculated in $O(\log i)$ time, but using $O(i^2)$ gates.

Thus, we can construct a fast adder by calculating the bp_i and bg_i signals in $O(\log i)$ time. The computation of the carries then requires one additional unit of delay and the final sum requires an additional unit of delay. Thus, addition is performed in $O(\log n)$ time. But there is a catch: we will need $O(n^3)$ gates which is not affordable.

4 A practical fast adder

A practical carry-lookahead adder (CLA) is constructed using the observations made in the previous section, but using far fewer gates (See Figure 3).

The different blocks in the CLA can be described as follows:

- The PG-block calculates p_i, g_i from a_i, b_i for $i = 0, 1, 2, \ldots, n-1$. This requires an XOR gate and an AND gate and takes 1 unit of delay.
- The BP,BG block combines the p_i and g_i in blocks of size k. The signals bp_k, bg_k are calculated as

  $$\begin{aligned} bp_i &= (p_{k-1}.p_{k-2}.p_{k-3}...p_0) \\ bg_i &= (g_{k-1} + p_{k-1}.g_{k-2} + p_{k-1}.p_{k-2}.g_{k-3} + \ldots p_{k-1}.p_{k-2}..p_1.g_0) \end{aligned}$$

  Using log-depth circuits, this can be done in $O(\log k)$ time, with $O(k^2)$ gates.
- The Block-carry stage calculates the carries at bits k, $2k$ etc. using the $bpk, bp2k, \ldots$ and $bgk, bg2k, \ldots$ signals calculated in the previous stage. This can be done using Equation 1, in $O(\log(n/k))$ time, using $O((n/k)^2)$ gates.

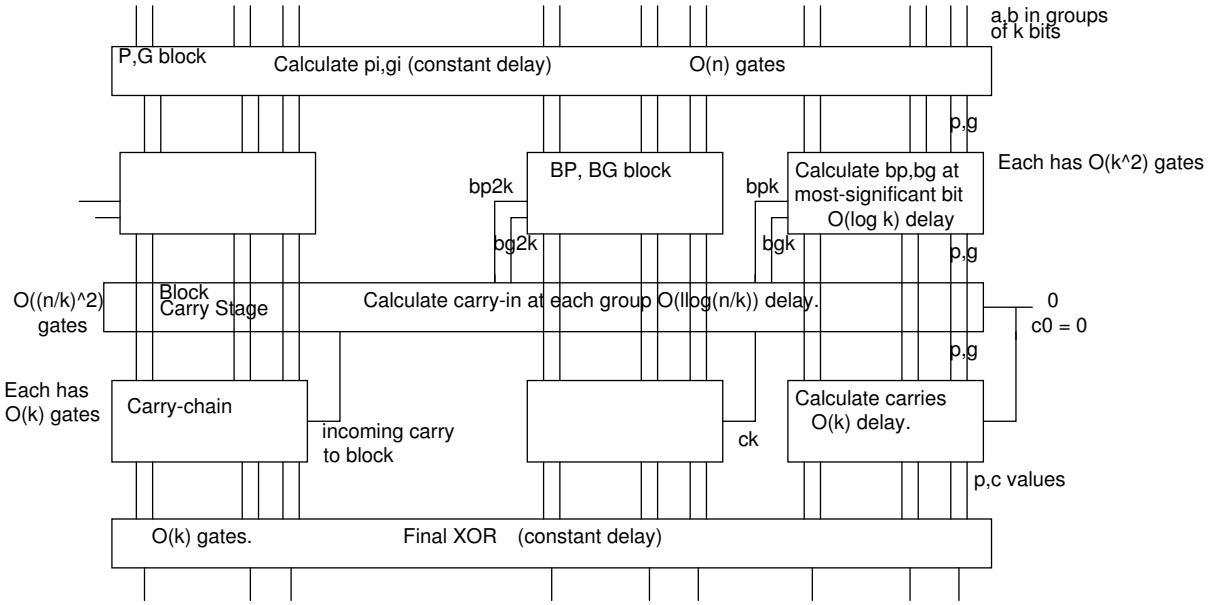

Figure 3: Carry-lookahead Adder

- The carry-chains calculate the carries within each group of k bits using the block carry in calculated in the previous stage. This can be done in a rippled manner and takes $O(k)$ time, with $O(k)$ gates.
- Finally, in the XOR stage, the p_i is XOR-ed with the carry computed in the carry chain, in order to finish the task. This takes n gates and is done in 1 unit of time.

Putting it all together, if $k = \sqrt{n}$, we see that we can compute an addition in close to $O(\log n)$ time (for small values of n), with $O(n)$ gates!

5 Homework

Assume that you have at your disposal two-input XOR, AND, OR gates and NOT gates. Let $n = 16$ and $k = 4$. Complete the design of each block in the CLA of Figure 3. How many gates does the final circuit require? If the delay of each gate is 1 unit, what is the maximum delay through the CLA?

6 Further Reading

There are many other ways of building fast adders: carry-select addition, carry-skip addition, Ling-adders, and hybrid strategies. Those who are interested can purchase the book [1] which is an excellent resource for arithmetic circuits.

References

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