BIT-PARALLEL ARITHMETIC

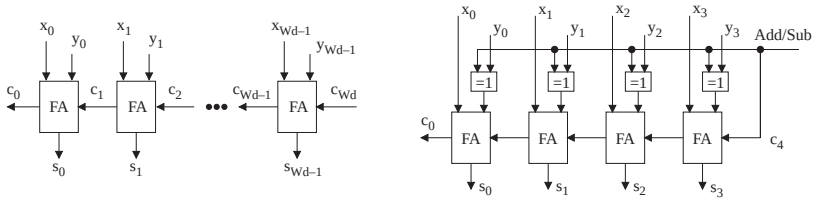
Addition and Subtraction

Two binary numbers,

$$\boldsymbol{x} = (x_0 . x_1 x_2 \ldots x_{Wd-1})_2 \text{ and } \boldsymbol{y} = (y_0 . y_1 y_2 \ldots y_{Wd-1})_2,$$

can be added using bit-parallel operation in $O(\log(W_d))$ steps

- **Ripple-Carry Adder/Subtractor**



Adder $x + y$

Subtractor $x - y$

Addition time is determined by the carry path in the full-adders.

Notice that even though the adder is said to be bit-parallel, computation of the output bits is done sequentially.

In fact, at each time instant only one of the full-adders performs useful work. The others have already completed their computations or may not yet have received a correct carry input, hence they may perform erroneous computations.

Because of these wasted switch operations, power consumption is higher than necessary.

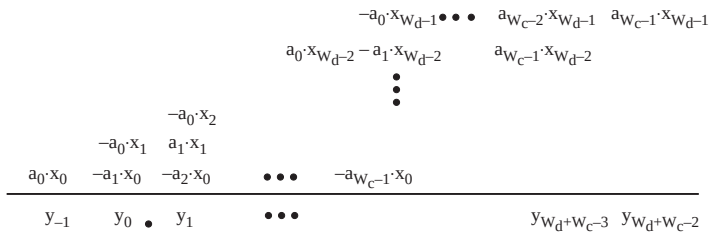
DSP Integrated Circuits
Lars Wanhammar
Department of Electrical Engineering
Linköping University
larsw@isy.liu.se
http://www.es.isy.liu.se/

- **Carry–Look–Ahead Adder**
- **Brent–Kung's Adder**
- **Carry–Save Adder**
- **Carry–Select Adder**
- **Carry–Skip Adder**
- **Conditional–Sum Adder**

DSP Integrated Circuits
Lars Wanhammar
Department of Electrical Engineering
Linköping University
larsw@isy.liu.se
http://www.es.isy.liu.se/

Bit-Parallel Multiplication

In order to multiply two two's-complement numbers, $\boldsymbol{a}$ and $\boldsymbol{x}$, we form the partial bit-products $a_i \cdot x_k$, as shown below.



High-speed, bit-parallel multipliers can be divided in three classes

- **Add–and–shift multipliers**

The partial bit-products are generated sequentially and accumulate them successively as they are generated. This type are therefore the slowest multiplier, but the required chip area is low.

DSP Integrated Circuits
Lars Wanhammar
Department of Electrical Engineering
Linköping University
larsw@isy.liu.se
http://www.es.isy.liu.se/

Parallel multipliers

All bit-products are generated in parallel and uses a multi-operand adder (i.e., an adder tree) for their accumulation.

A parallel multiplier structure can be partitioned into three parts: partial product generation, carry-free addition, and carry-propagation addition.

These three parts can be implemented using different schemes.

For example, the partial product generation can be implemented by AND gates or by using the Booth's algorithm.

The carry-free addition part is often implemented by using a Wallace or redundant binary addition tree.

Array multipliers

An array of almost identical cells for generation of the bit-products and accumulation.

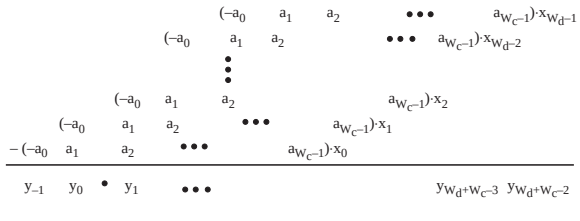
This type of multipliers requires the least amount of area, but it is also the slower

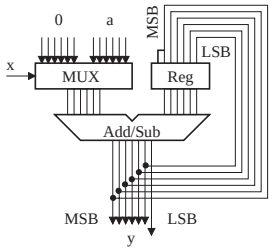
Add–And–Shift Multiplication

Consider the multiplication of two two's-complement numbers, $\boldsymbol{y} = \boldsymbol{a} \cdot \boldsymbol{x}$.

$$\boldsymbol{y} = \boldsymbol{a}\left(-x_0 + \sum_{i=1}^{W_d-1} x_i 2^{-i}\right) = -\boldsymbol{a}x_0 + \sum_{i=1}^{W_d-1} \boldsymbol{a}x_i 2^{-i}$$

The multiplication can be performed by generating the partial products, for example row wise.





Horners' method

Modified Booth's Algorithm

By considering three bits at a time from the input operand, $\boldsymbol{x}$, is used to recode the other input operand, $\boldsymbol{y}$, into partial products ($-2\boldsymbol{y}$, $-\boldsymbol{y}$, $\boldsymbol{0}$, $\boldsymbol{y}$, $2\boldsymbol{y}$) that are added (subtracted) to form the result.

Hence, the number of recoded partial products is reduced.

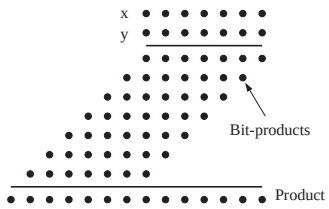
Booth's algorithm reduces the number of additions/subtractions cycles to $W_d/2 = 8$, which is only half that required in the ordinary shift–and–add algorithm.

Notice that the reduction of the number of cycles directly corresponds to a reduction in the power consumption.

Booth's algorithm is also suitable for bit-serial or digit-serial implementation.

Tree–Based Multipliers

Short-hand version of the bit-products



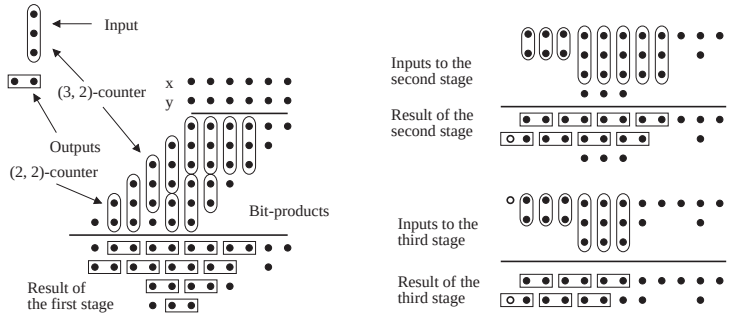
In a *tree-based multiplier* the bit-products are often added using full-adders.

A full-adder can be considered as a counter, denoted (3, 2)-counter, that adds three inputs and forms a two bit result.

The output equals the number of 1*s* in the inputs – counter.

A half-adder is denoted (2, 2)-counter.

In 1964 Wallace showed that a tree structure (*Wallace tree*) of such counters is an efficient method, $O(\log(W_d))$, to add the bit-products.



In the fourth stage, a RCA or a CLA can be used to obtain the product.

Several alternative addition schemes are possible.

For example, Dadda has derived a scheme where all bit-products with the same weight are collected and added using a Wallace tree with a minimum number of counters and minimal critical paths.

However, the multiplier is irregular and difficult to implement efficiently since a large wiring area is required.

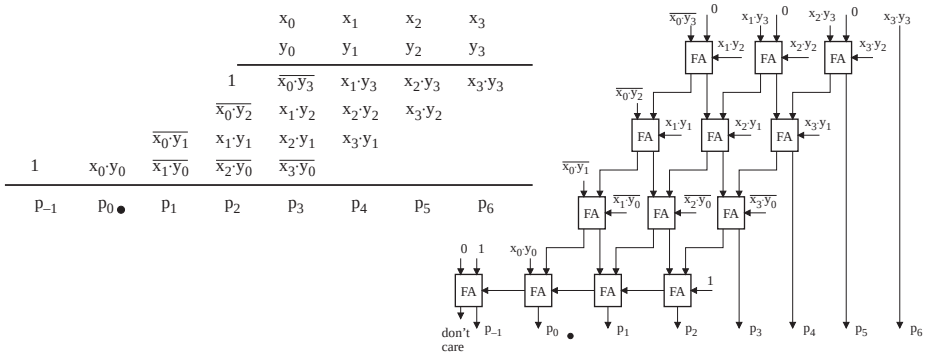
Wallace tree multipliers should therefore only be used for large word length and where the performance is critical.

Another more regular structure is the Overturn-Stair Multiplier

Array Multipliers

Many array multiplier schemes with varying degrees of pipelining have been proposed.

A typical array multiplier—the *Baugh–Wooley's multiplier* with a multiplication time proportional to $2W_d$.



Notice that some slight variations in the way the partial products are formed occur in the literature.

All bit-products are generated in parallel and collected through an array of full-adders and a final RCA.