Cairo University
Electronics and Communications Engineering

Computer Arithmetic, Lecture 8: The big summation

Hossam A. H. Fahmy

Reduction techniques

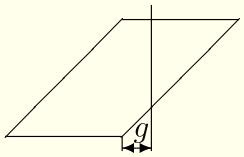
After the generation of the PPs, we must sum them by first reducing them to two PPs then use a final CPA. The reduction can use:

- Arrays (2D structures)
  - Single arrays
  - Double arrays
  - Higher order arrays
- Trees (3D structures flattened in ICs)
  - Wallace trees
  - Binary trees
  - ZM trees
  - OS trees

A multiplication produces a result with $2n$ bits, but we usually store only n bits. *Which? Why?*

Do we really need all the PPs?

- For the IEEE standard, we need the complete PPs but the low order bits are only used to get the sticky bit. $\Rightarrow$ *It is not necessary to sum them, just get the sticky and any carry into the higher part.*
- In non-IEEE compliant circuits, we can truncate the PPs if we leave some guard zone in case there is a carry into the higher part.



Is $(X \times Y)^t = (Y \times X)^t$?

Yes for direct multiplication without Booth recoding.

No for Booth recoding.

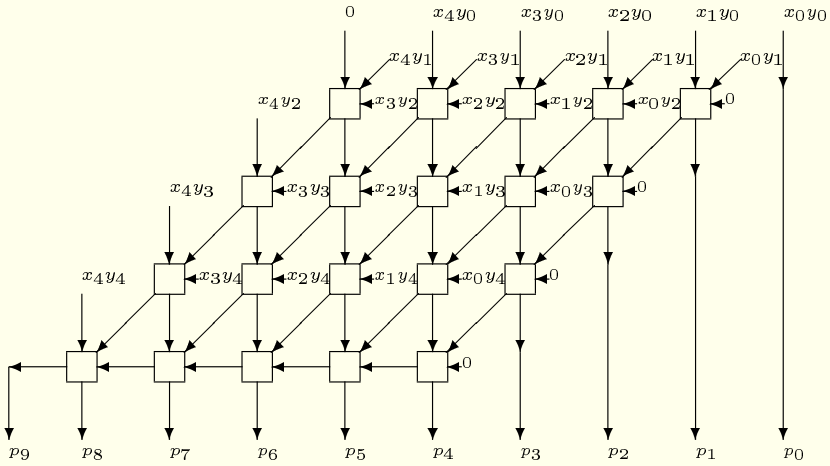
In recoding we rely on the fact that $2X - X$ should equal X but

- if $X = 000101$ then $2X = 001010$.
- We truncate to get $X^t = 00010$ and $(2X)^t = 00101$,
- then $C(X^t) = 11110$ and $(2X)^t + C(X^t) = 00011 \neq X$.

Multi-operand addition

- A full adder sums a, b, and c. It *counts* the number of ones in the three inputs and codes that number in binary in two bits c_{out} and s. It is a $(3,2)$ *counter*.
- In two operand addition, we propagate c_{out} to the next higher position in a row of full adders.
- In multi-operand addition, we *defer* that propagation. We use three different input vectors to the row of full adders and get two output vectors. $\Rightarrow$ *Deferred Carry Adder, or Carry Save Adder*
- A row of full adders receives those two vectors grouped with a fourth input vector to sum them and produce yet another two vectors and so on.

A simple array for unsigned numbers



- Each cell is a CSA and requires 3 wires.
- For n operands, we need $(n-2)$ CSA levels.

(A Half Adder replaces the FA cells with a 0 carry.)

Merits of different topologies

A multiplier topology refers to the way we interconnect the bit positions in the PPs reduction.

Two important measures of topologies are considered:

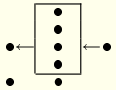
1. the minimum number of wires needed within a single bit position, and
2. the number of counter delays to reduce the PPs to two bit vectors.

Why think of wires?

- The number of wires needed per bit position often determines the minimum bit pitch. If this is not compatible with the rest of the datapath width, the design of the multiplier must be reviewed.
- Some circuit techniques represent each signal by two wires (for example, dual rail domino). These are faster but we can only use them if the total number of wires can be accomodated.
- Arrays minimize the wires at the expense of gate delays while trees do the opposite.

The $[4:2]$ compressor

A $[4:2]$ compressor takes four input bits and one carry-in to generate two output bits and a carry-out *independent of the carry-in*.



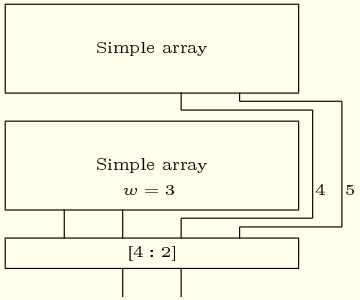
It is equivalent to two $(3,2)$ counters.

1. *How is c_{out} independent of c_{in}?*
2. *Is the delay equal to twice that of $(3,2)$ counters?*

Double arrays

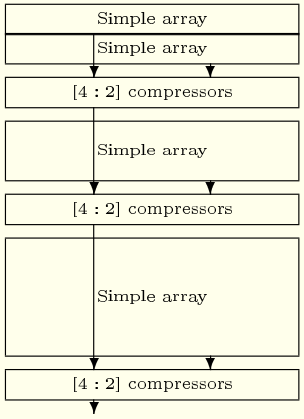
Since a single array has $w = 3$ and $h = n - 2$, a double array uses two sub arrays of $h = \frac{n}{2} - 2$ followed by a $[4:2]$.

The double array has $w = 5$.



Higher order arrays

Use more than just two sub arrays and arrange them so that the shortest sub array has the longest compressor chain giving $w = 5$.



Comparison between arrays

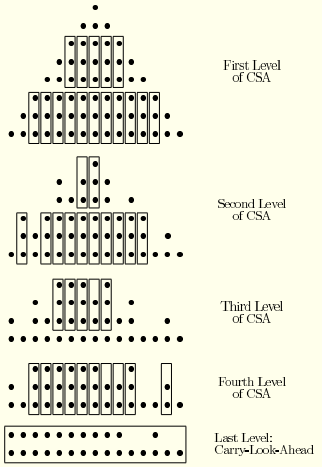
Array type	w (lines per bit)	number of $(3,2)$ counters
Simple	3	$n-2$
Double	5	$(\frac{n}{2}-2)+2 = \frac{n}{2}$
Higher order	5	$\mathcal{O}(2\sqrt{n})$

Flourishing trees

- Trees optimize the depth (the delay) on the expense of w.
- Trees are either regular with a specific w or irregular where w is determined by the design layout.

Wallace tree

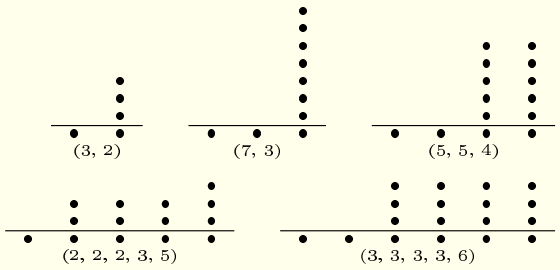
- Wallace (1964) proposed an irregular tree using $(3,2)$ counters to reduce n operands to two bit vectors.
- In each step we get a reduction of $\frac{3}{2}$ in the number of remaining vectors. Hence the number of levels $\approx \lceil \log_{\frac{3}{2}} \frac{n}{2} \rceil$.
- The number of wires w depends on the actual layout.



An example for an 8×8 multiplication.

Other counters and compressors

In $(c_{r-1}, \cdots, c_0, d)$ counters each c_i is the height of the i^{th} column and d is the number of output bits.



The $(5,5,4)$ and $(7,3)$ counters are the usual alternatives to $(3,2)$ counters.

The different counters are realized by ROMs, custom logic, PLAs, or (usually) with $(3,2)$ counters reconfigured.

Binary trees

Weinberger (1981) proposed the binary trees. Those

- use $[4:2]$ compressors,
- have $\lceil \log_2 \frac{n}{2} \rceil$ levels of$[4:2]$ compressors and hence $\approx 2(\lceil \log_2 n \rceil - 1)$ CSA levels, and
- require $w = 2\lceil \log_2 n \rceil$.

ZM trees

Zuras and McAllister (1986) proposed the ZM trees. Those

- use $(3,2)$ counters,
- are regular and have balanced delays, and
- are recursively defined by a tree body and a chain.

Type 1 reduce to the same thing as higher order arrays with $w = 5$ and $\mathcal{O}(2\sqrt{n})$ CSA levels.

Type 2 and higher order ZM trees are less regular. An order-p ZM tree requires $w = 2p + 3$ and has $\mathcal{O}(n^{\frac{1}{p+1}})$ CSA levels.

Overturned staircase (OS) trees

- Similar to ZM but achieve the delay of Wallace trees for many values of n.
- Designed recursively and are regular.
- Higher order trees are also defined
- For order-p we get $w = 3p + 3$ and $\mathcal{O}(n^{\frac{1}{p+1}})$ CSA levels.

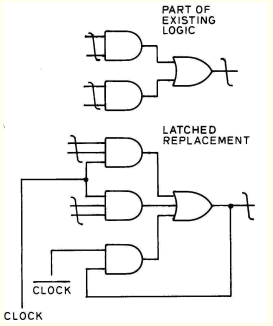
Summary of toplogies

- There is a big variety of topologies with various trade-offs.
- The reduction is only one part of the structure. It must be compatible with the generation and the final CPA.
- The CPA is still taking a big portion of the time ($\approx 30\%$)

Iteration

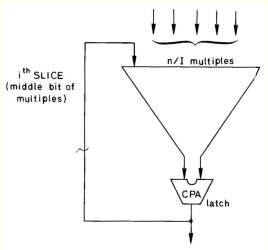
There is no need to fully build the array or the tree. A smaller structure may be used and the data iterated.

We can add latches to the existing logic with little overhead.



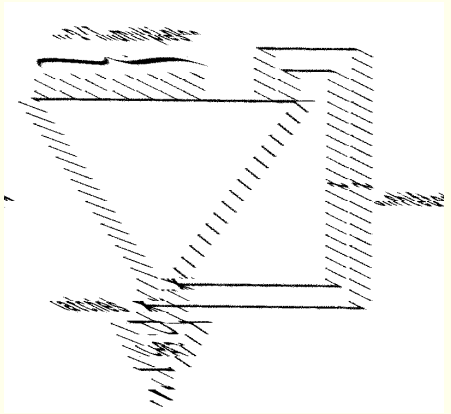
Iterate on the CPA

- Brings the CPA output back into the tree.
- The tree has $\lceil \frac{n}{I} + 1 \rceil$ inputs.
- The CPA width is $(n + \lceil \frac{n}{I} + 1 \rceil)$ and its output is shifted by $k\frac{n}{I}$ when Booth-k is used.



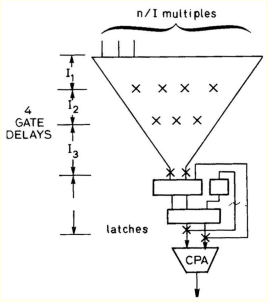
Iterate on the tree

- Brings the tree output (shifted) back into the tree.
- The tree has $\lceil \frac{n}{I} + 2 \rceil$ inputs.
- The CPA delay is not part of the iteration.



Iterate on a "compressor"

- The iteration is on the lowest level of the tree.
- For a $[4:2]$ compressor, the iteration is about four gate delays. Both the tree and the PP generator must support that rate.
- The tree carries the PPs for several iterations.



Comparing the iterative approach with a full tree

- The iteration reduces the cost of PP generation and reduction.
- If the iteration overhead is low, the result is comparable to a fully built tree.

Two 64 bit multipliers

Type	PPs generated	Depth in $(5,5,4)$	CPA size
Tree	15	3	128 bit
Iterative on a $(5,5,4)$	3	5 (plus clock overhead)	74 bit

Conclusions

- Mutlipliers consume a large area with potential wiring problems.
- Clever designs can reduce the hardware *and* improve the speed.