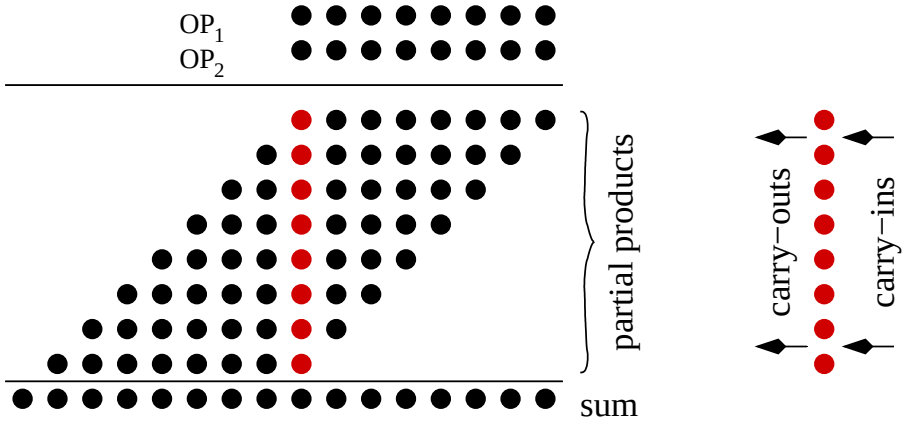
Power Comparison of Low Bitwidth Multipliers

Ralf Hildebrandt, Fraunhofer IPMS, Germany

- Hardware multipliers accelerate many computations.
- Low bithwidth microcontrollers are often used today in embedded low power systems. These microcontrollers can be improved with multipliers.
- The gain of speed can be used for higher throughput or longer power-down periods.
- Used example: $16 \cdot 16 \Rightarrow 32$ bit Booth-encoded multiplier for signed and unsigned multiplication mapped to a *standard cell library.*

Parallel Multiplier Structures



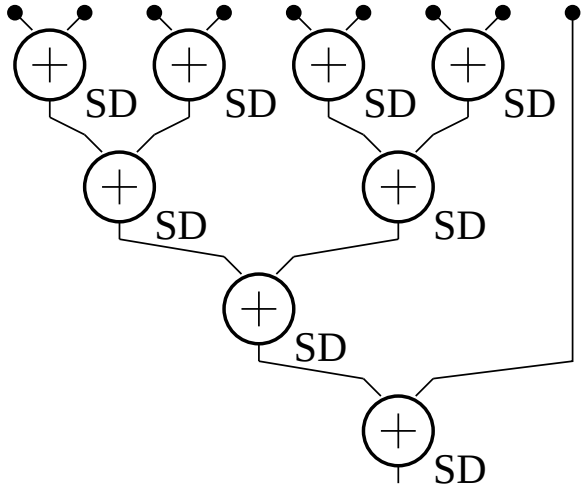
design concept (no sign extension)

discussion is focused on one partitial product reduction column

PPG = Partial Product Generation (includes Booth-Encoding, if used)
PPR = Partial Product Reduction
CPA = Carry-Propagate-Adder

As in every combinational logic: Energy dissipation depends on the existence of glitches and the length of ripple chains.

PPR: Signed Digit Arithmetics



tree structure of the SD-PPR (simplified)

delay estimation:

$$\frac{t}{t_{XOR}} = \lceil 3 \cdot \log_2 \lfloor k \rfloor \rceil$$

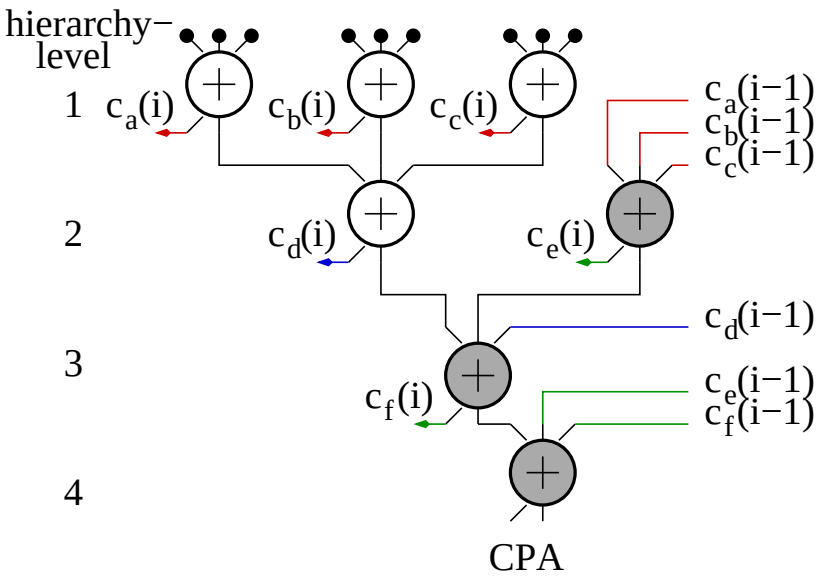
area estimation:

$$\frac{A}{A_{fulladder}} = 2(k-1)$$

(k is the number of addends)

The result can be converted back from SD-form to 2th complement using a subtractor (similar to a conventional CPA).

PPR: Carry-Save-Tree



variant of a Wallace-Tree

delay estimation:

$$\frac{t}{t_{XOR}} = \lceil 2 \left(\log_{\frac{3}{2}} \lfloor k \rfloor - \log_{\frac{3}{2}} 2 \right) \rceil$$

area estimation:

$$\frac{A}{A_{fulladder}} = k - 2$$

(k is the number of addends)

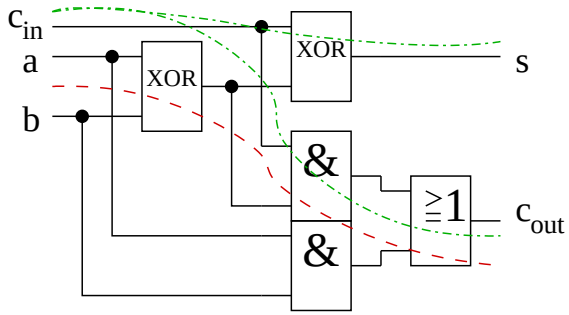
$\log_{3/2}$ (Carry-Save-Tree) grows faster than $\log_2$ (Signed-Digit-Tree), but the constant offset $\log_{3/2} 2$ and the factor 2 give the CST a speed advantage over the SDT up to 63 bits word width.

A CST is harder to route than a SDT because of the carry paths. This reduces the speed advantage.

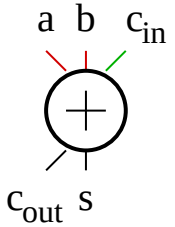
PPR: Carry-Save-Array

A CSA is highly regular (simple routing), but has a linear relationship between calculation time and number of addends: $t/t_{XOR} = 2\,(\lfloor k \rfloor - 2)$

But fulladders in standard cell libraries have two delay paths with different length.



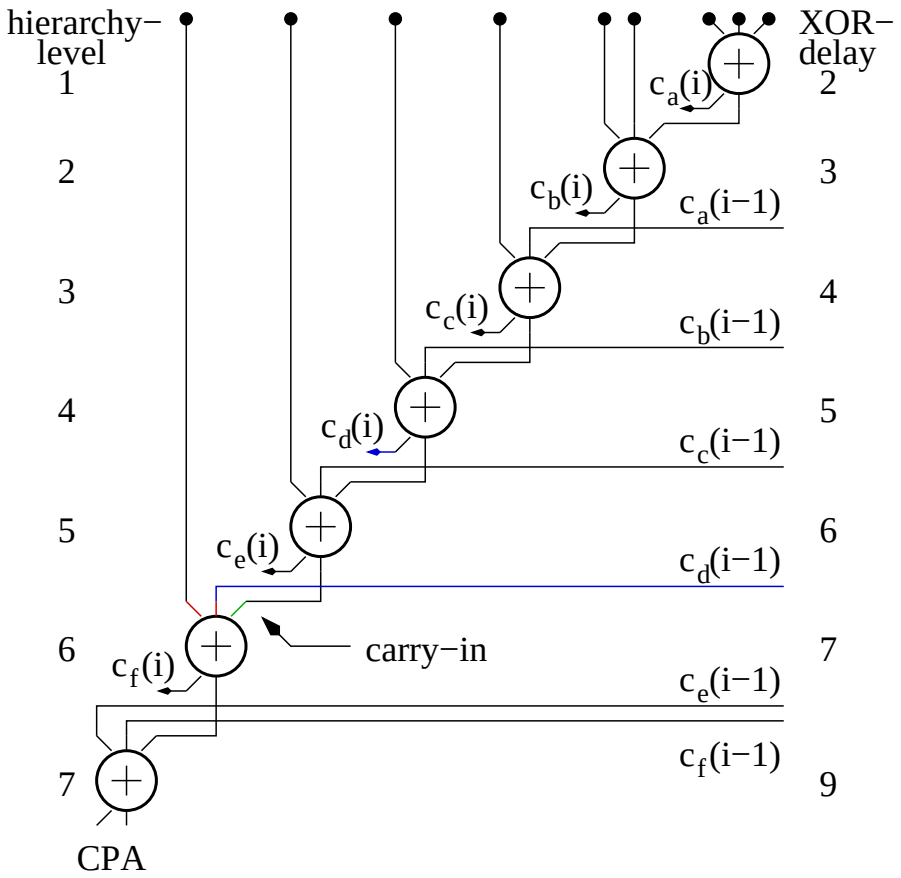
delay paths of fulladders



fulladder

Connecting a late-arriving signal to the carry-in will be advantageous.

PPR: Leapfrog-Carry-Save-Array



new modified Leapfrog-CSA

delay estimation:

$$\frac{t}{t_{XOR}} = \lfloor k \rfloor$$

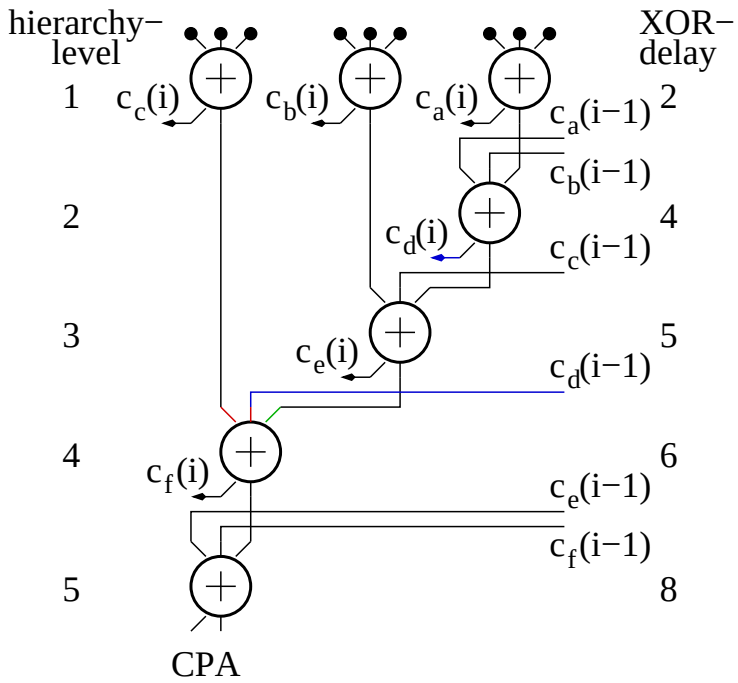
area estimation:

$$\frac{A}{A_{fulladder}} = k - 2$$

(k is the number of addends)

modified version: carrys jump over one hierarchy level - not the sum outputs
Complete PPR has better balanced signal arrival times in the higher order PPR columns.

PPR: Carry-Save-Tree-Leapfrog-Array



CSTLFA

delay estimation:

$$\frac{t}{t_{XOR}} = 2 + 2 \cdot \lfloor k/3 \rfloor + (k \,\mathrm{MOD}\, 3)$$

$$(k \geq 5)$$

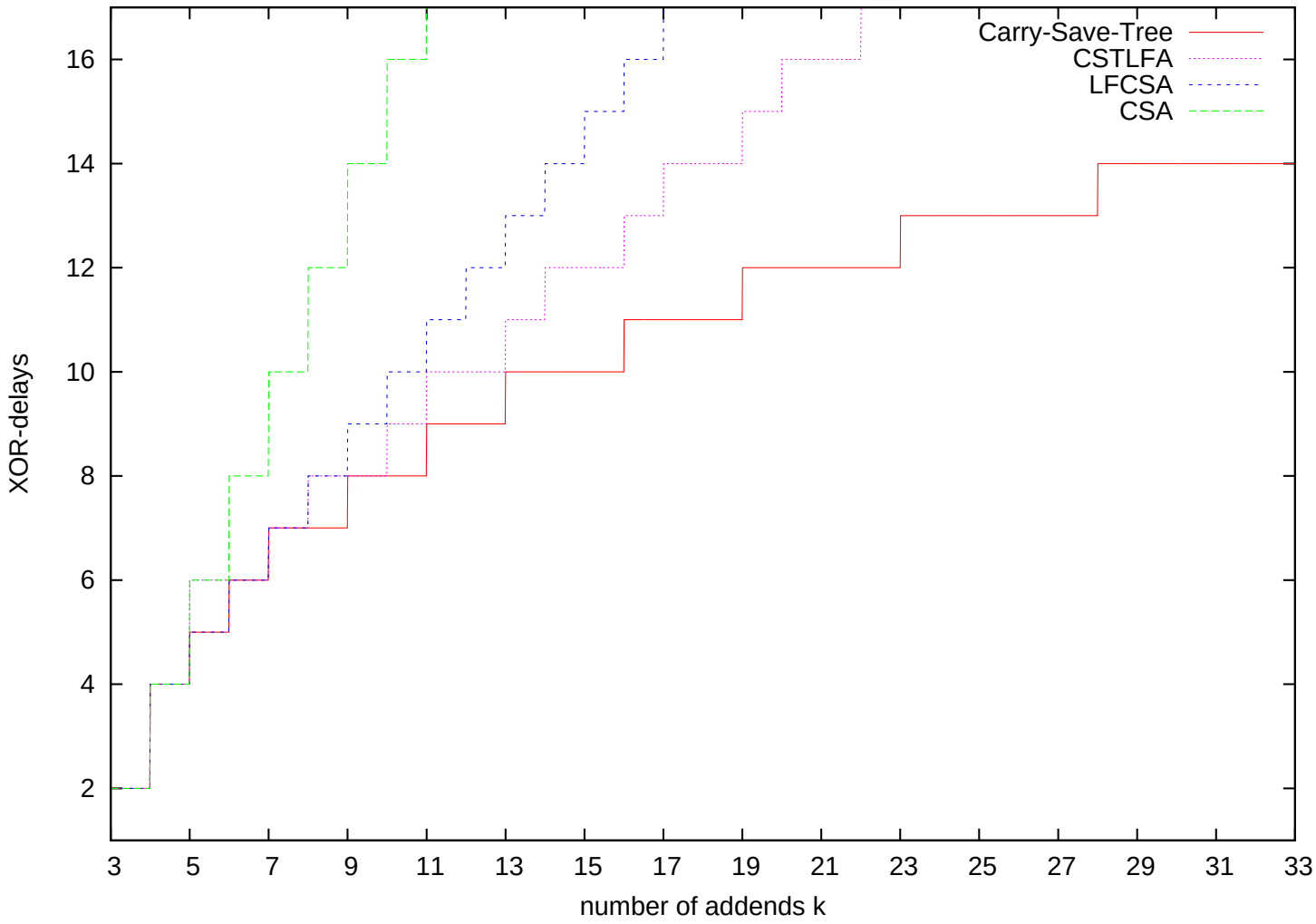
area estimation:

$$\frac{A}{A_{fulladder}} = k - 2$$

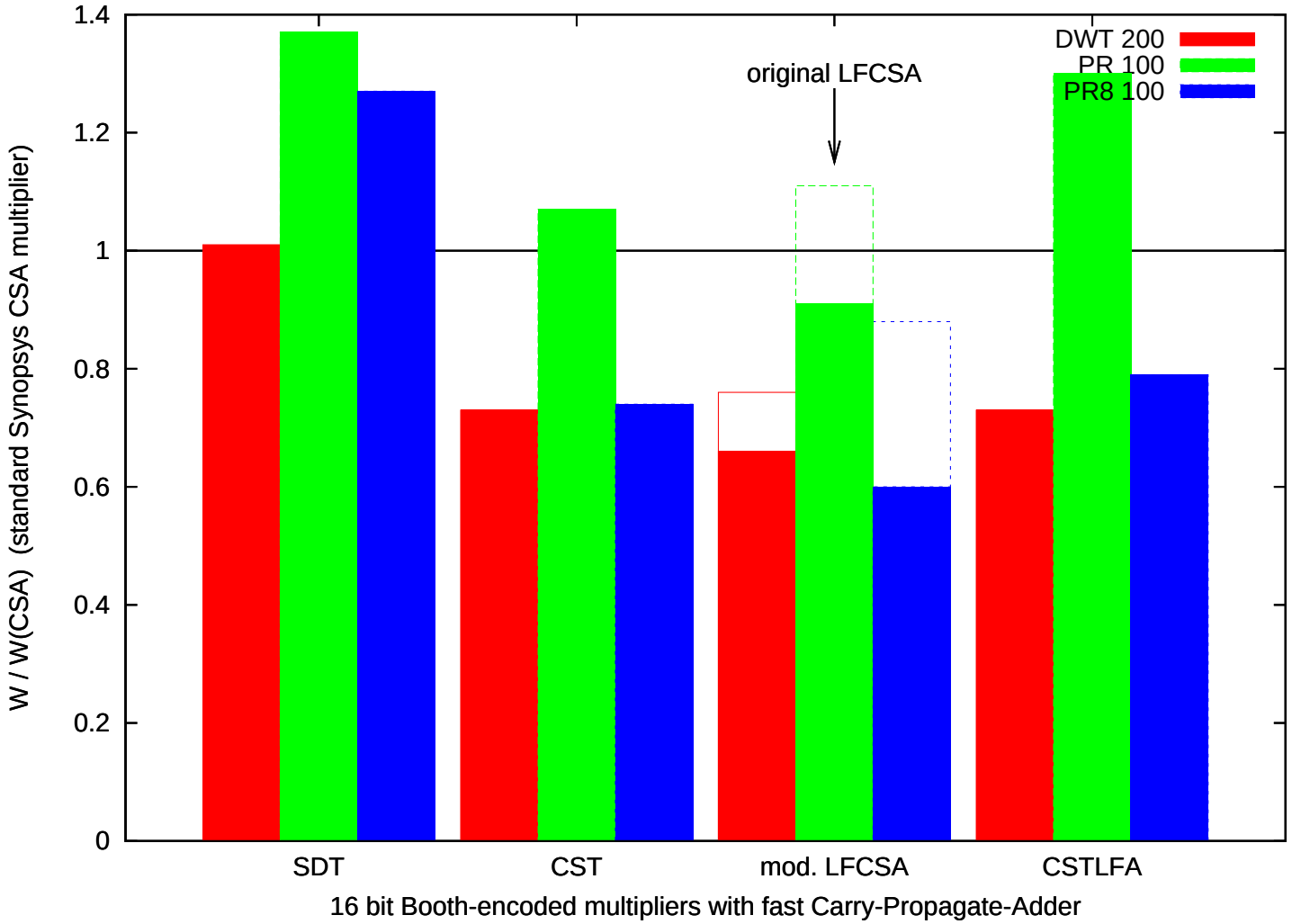
(k is the number of addends)

The CSTLFA is a compromise between tree and array. The tree structure may be used for more than 1 hierarchy level.

PPR delay: Carry-Save arrays versus tree



PPR: dissipated energy normalized with CSA - benchmarks

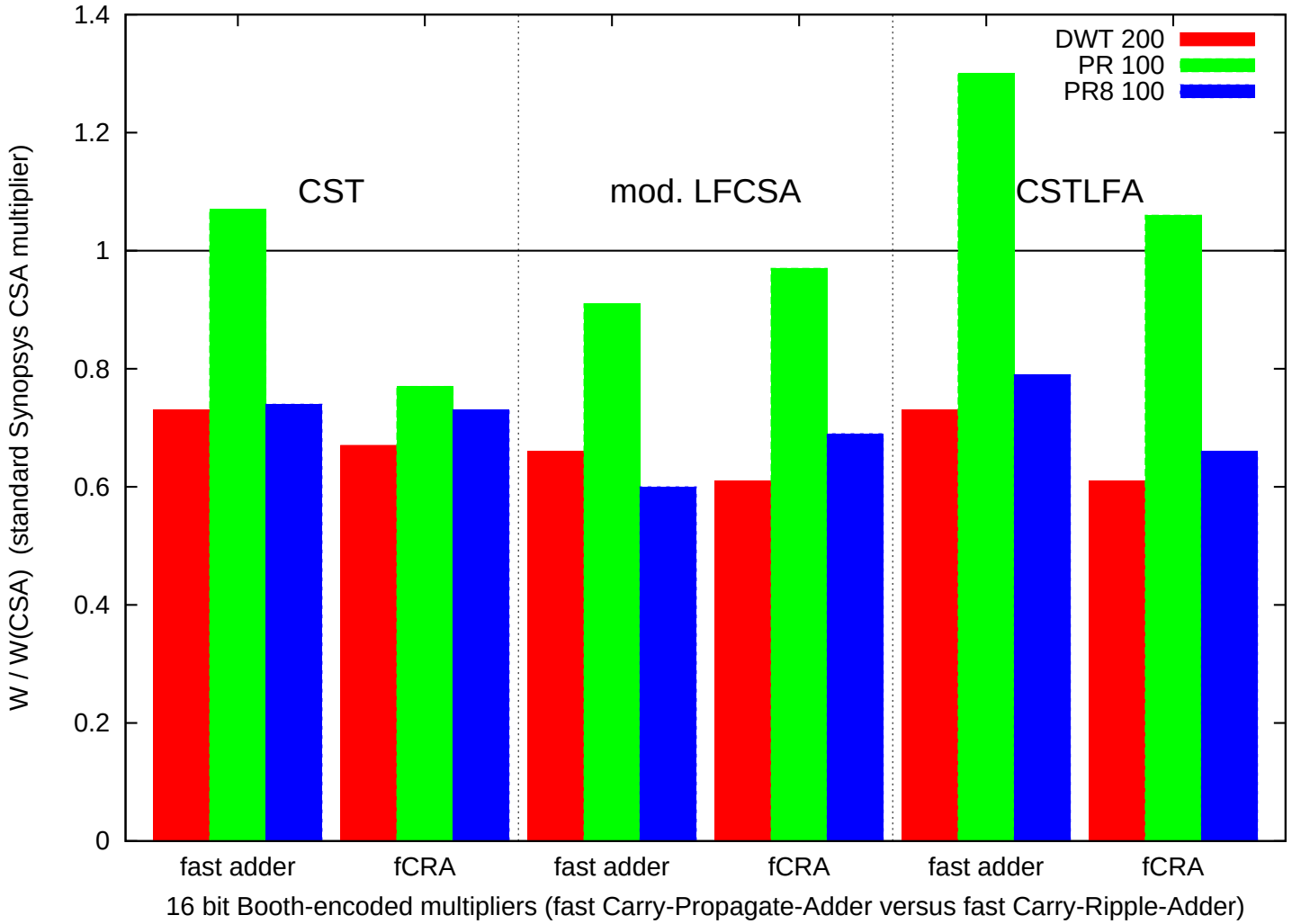


Carry-Propagate-Adder

- Possible options:
  - high speed: automatically optimized by synthesis (in a HDL: a+b) or manually optimized (e.g. with algorithm GEF[a])
  - minimal area: Carry-Ripple-Adder (CRA) and fast Carry-Ripple-Adder (fCRA; approx. 75% delay of the CRA in a $0.5\mu m$ standad cell library)
- The speed of the CPA rules the speed of the multiplier.
- Fast CPAs are as twice as big as a CRA but this difference results in less than 13% difference if one compares the complete areas of the multipliers.
- The CPA (especially in the most significant bits) dissipates a lot of energy because of unavoidable signal ripple.

[a] Generalized Earliest First

CPA: dissipated energy normalized with CSA - benchmarks



Conclusion

- PPG: Booth-Encoding reduces of the number of partial products but may produce glitches while computing pseudo-random data.
- PPR:
  - The new modified Leapfrog-CSA has well balanced signal paths and therefore decreases ribble.
  - A combined tree and array structure (CSTFLA) may be a compromise for wider word widths.
- CPA:
  - Signal ribble is often unavoidable so an area minimal CPA like the fCRA may be superior to fast adders in terms of low power dissipation, if the reduced speed is acceptable.
  - The fCRA decreases the speed gap between classic CRA and fast adders.