Automating High-Efficiency Compressor Trees in Vivado™

Thomas B. Preußer, Shikha Soni, Fan Zhang, Michaela Blott

01-Jul-2026

AMD

together we advance_

Itinerary

- Background: FINN & Compressors
- Embracing Versal
- Mainstreaming in Vivado
- Summary

[Journal Talk]
Konstantin Hoßfeld, Hans Jakob Damsgaard, Jar Nurmi, Michaela Blott, Thomas B. Preußer: *High-Efficiency Compressor Trees for Latest AMD FPGAs*. ACM TRETS, June 2024.
https://doi.org/10.1145/3645097

Background and Motivation

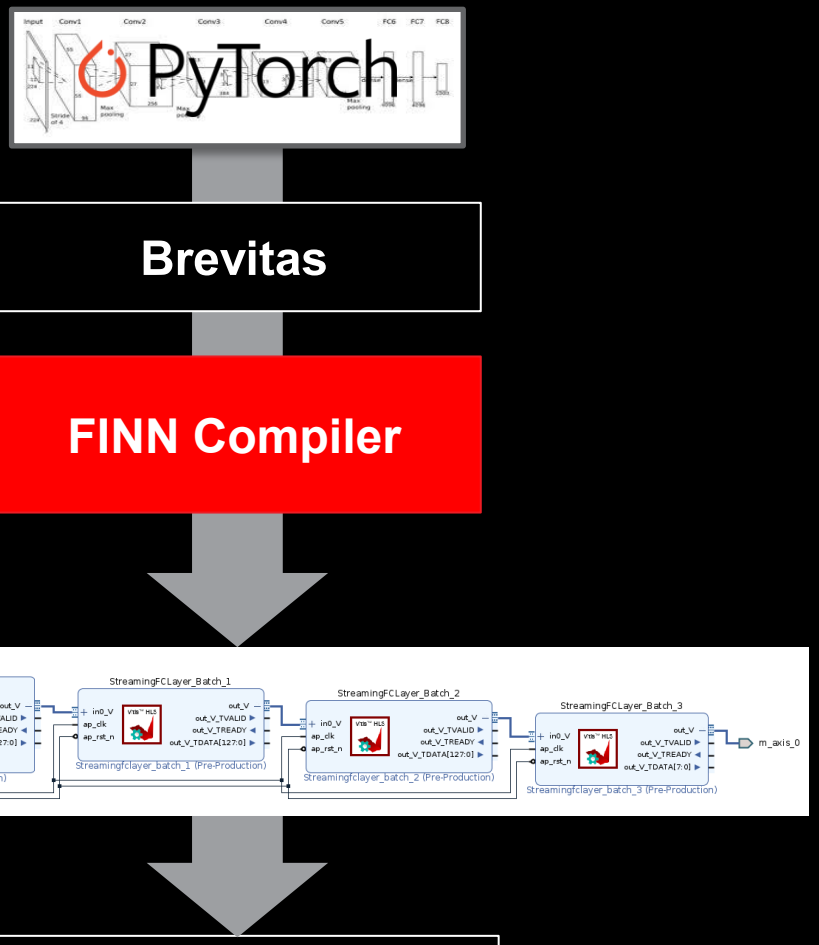
Background: FINN

- Building deterministic ***Fabric ML*** dataflow solutions.
- Designs are ***highly customizable*** for the desired performace-resource trade-off by tuning the implemented compute parallelism.

- Models are co-designed, optimized and quantized using Brevitas, which supports techniques like QAT, PTQ, sparsification, ...
- While the range of relevant precisions has extended to about 8 bits, **linear operations** (matmul, perceptron, convolution) over narrow integer inputs **remain a key workload**.

fp4	**E2M1 → int5**
fp6	**E2M3 → int7**

- The additive reduction of subtotals computed in parallel is a critical part of these operations.
- Compressor trees for multi-operand addition are a key technique in constructing optimized solutions.



Bit Matrix Compression Problem

Multi-Operand Addition

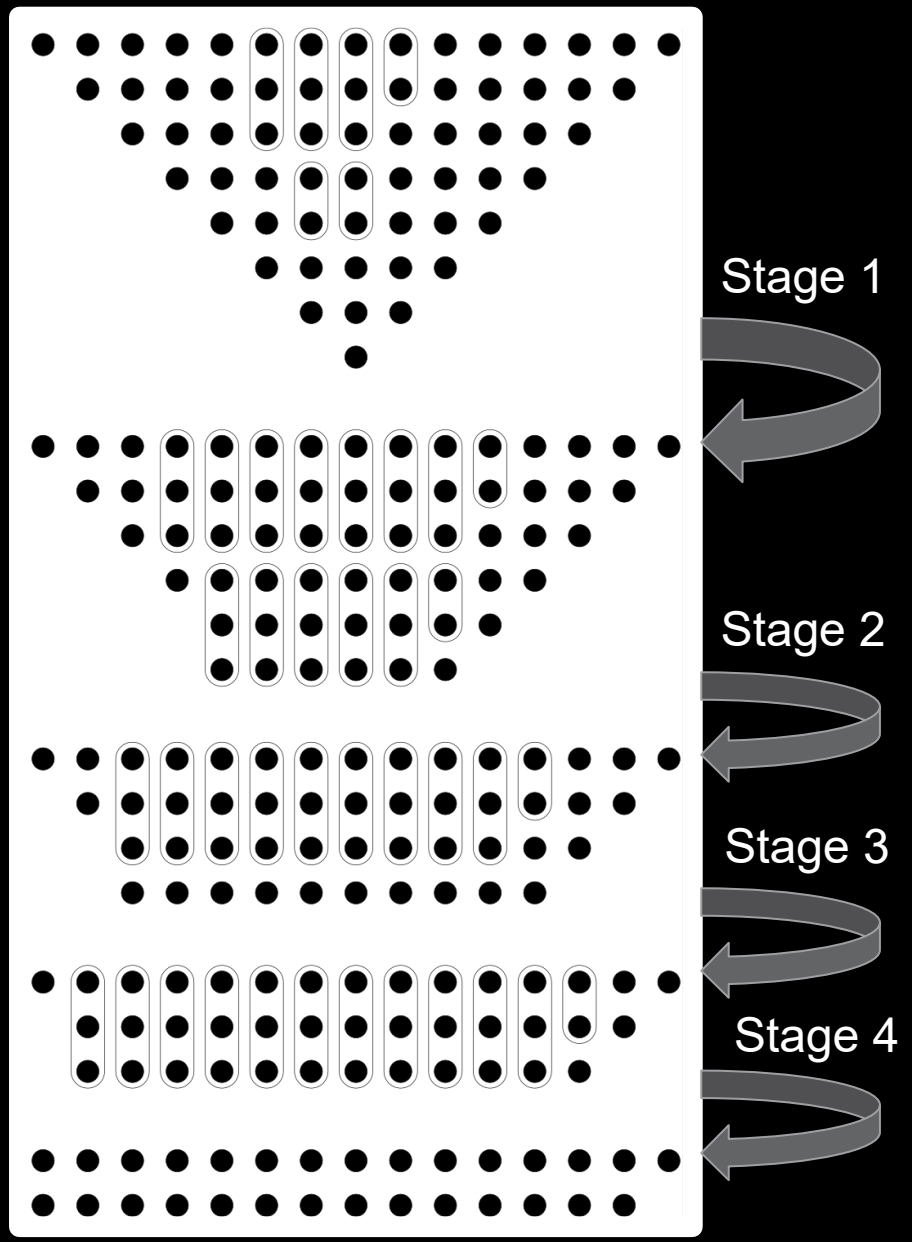
- Sum up an arbitrary bit matrix.
- Conventional ASIC Solution: Dadda reduction.

Issues with FPGA Adaptation

- FAs underutilize LUT capabilities.
- Heavy reliance on general-purpose routing while ignoring fast designated carry chains.

Objective

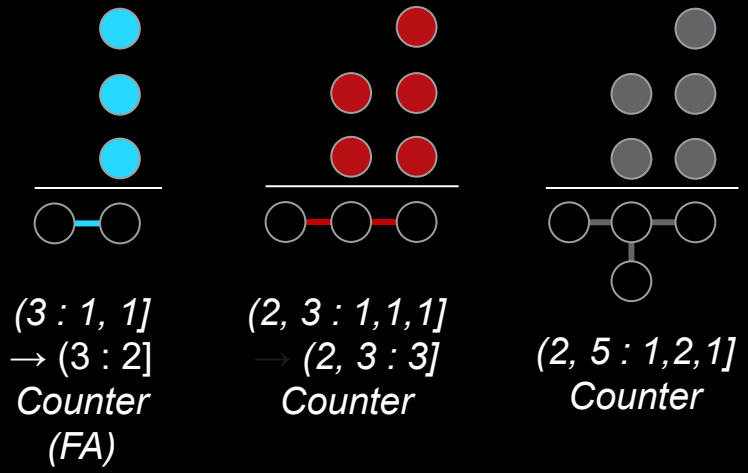
- Increase LUT utilization.
- Leverage carry chains for improved mappings while bounding general-purpose routing depth.



Compressor Construction

Design Generalized Parallel Counters (GPCs)

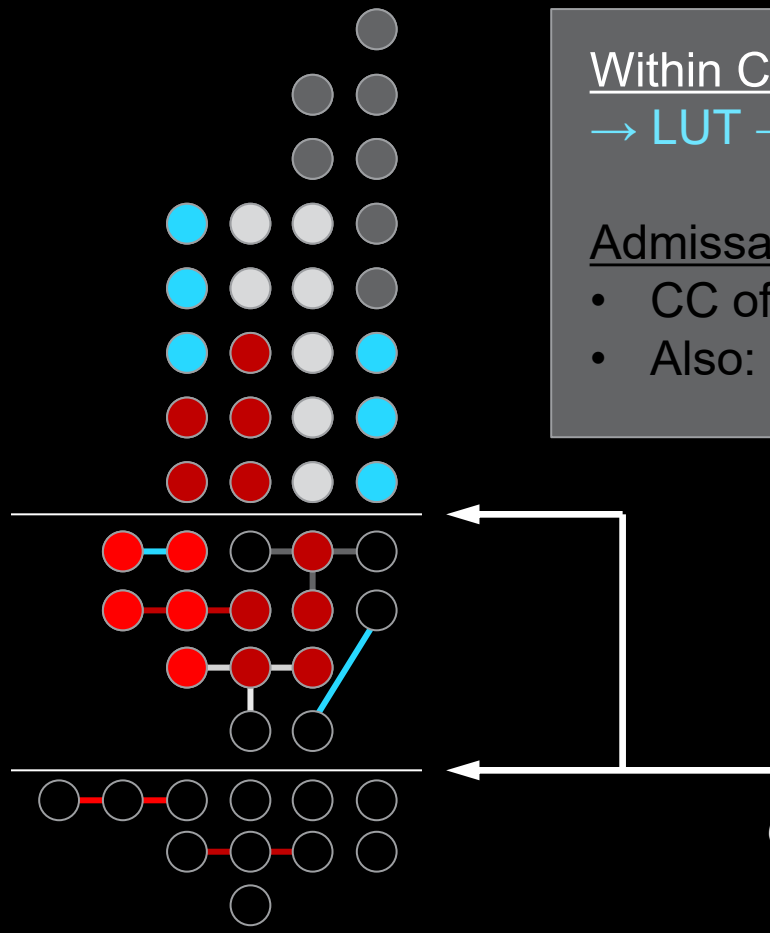
Use Them Constructing Compressor Trees



(3 : 1, 1]
→ (3 : 2]
Counter
(FA)

(2, 3 : 1,1,1]
→ *(2, 3 : 3]*
Counter

(2, 5 : 1,2,1]
Counter



<u>Within Counter Stage:</u>
→ LUT → CC* →

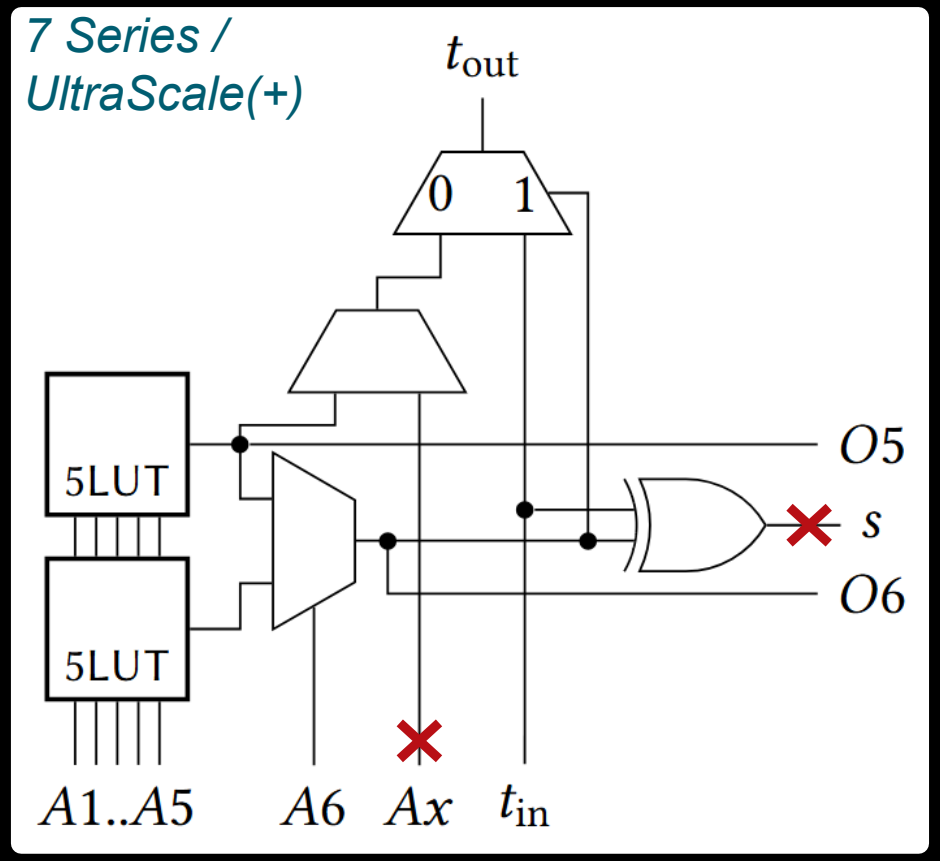
<u>Admissable Chaining:</u>
- CC of full slices before Versal.
- Also: LUT cascade on Versal.

Natural Pipelining Opportunities

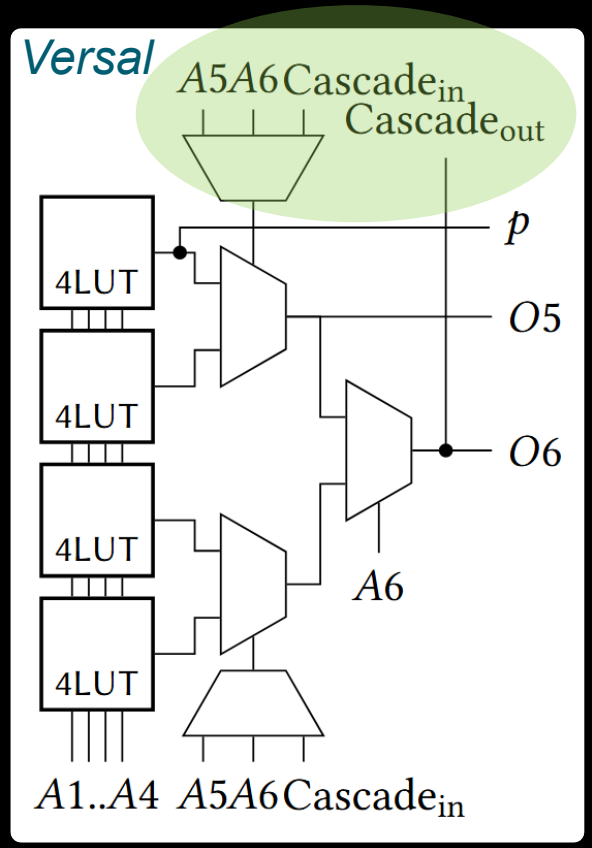
Embracing Versal™

Map Compressor Trees for Versal

Structural changes to Versal CLB architecture demand a specialized GPC library.



- Lost LUT bypass feed into carry chain.
- Lost designated `XORCY` gate for sum bit completion.
- Reorganized propagate compute.
- Gained more generic cascading capability (without `LOOKAHEAD` acceleration).



Konstantin Hoßfeld, Hans Jakob Damsgaard, Jar Nurmi, Michaela Blott, Thomas B. Preußer: *High-Efficiency Compressor Trees for Latest AMD FPGAs*. ACM TRETS, June 2024. https://doi.org/10.1145/3645097

Counter Selection Metrics

- p – number of input bits
- q – number of output bits

$(p > q)$

Efficiency:

$$E = \frac{p - q}{\#\text{LUTs}}$$

Signal reduction in relation to hardware investment.

Strength:

$$S = p/q$$

Stage count optimization.

(3 : 2] Counter $(n = 1)$

(1, 4, 1, 5 : 5] Counter $(n = 4)$

(7 : 3, 1] Counter $(n = 3)$

Atoms	$(\ldots, 2, 3)$ E	S	$(\ldots, 1, 5)$ E	S	$(\ldots, 0, 7)^{\S}$ E	S
$(2, 2, \ldots)$	1	$1.8^{\dagger}$	1.25	2	1.5	2.2
$(1, 4, \ldots)$	1.25	2	1.5	$2.2^{\ddagger}$	1.75	$2.4^{\ddagger}$
$(0, 6, \ldots)$	1.5	2.2	1.75	2.4	2	$2.6^{\ddagger}$

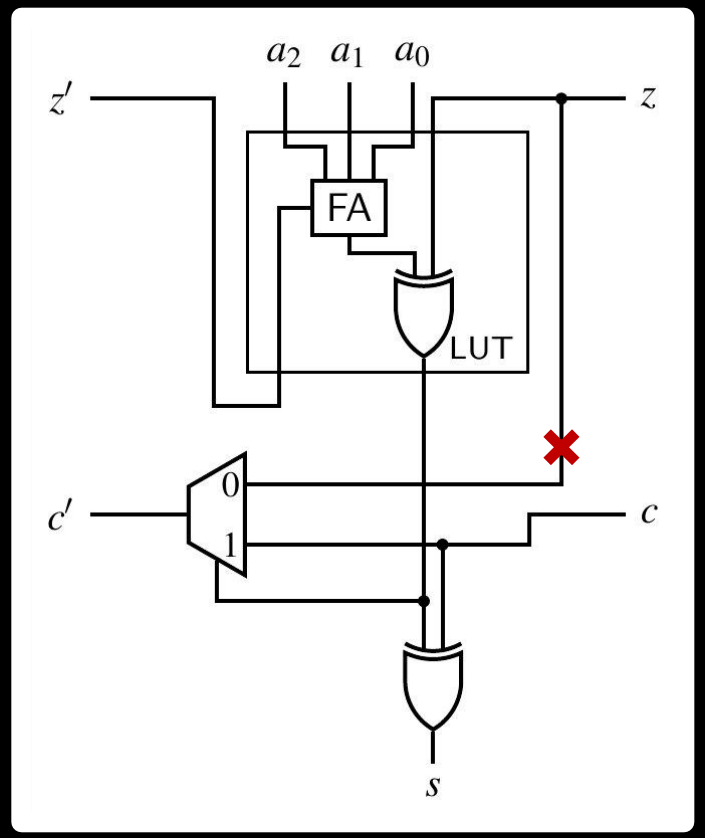
†Standard 4-bit ripple-carry adder (RCA).
‡Counters originally proposed by Kumm and Zipf [13, 14]
§Counters further improved by Yuan et al. [32]

Counter	E	S
(2) Atom Chain (Ripple-Carry)	1	$2 - \frac{1}{n+1}$
(1, 4) Atom Chain	1.5	$2.5 - \frac{3}{2n+2}$
Ripple-Sum	1	$2 - \frac{1}{n+1}$
Dual-Rail Ripple-Sum	1.5	$2.5 - \frac{3}{2n+2}$
(2, 2, 2) Atom Chain (Ripple-Carry)	1.5	$2 - \frac{1}{n+1}$

Overview: Counter Metrics

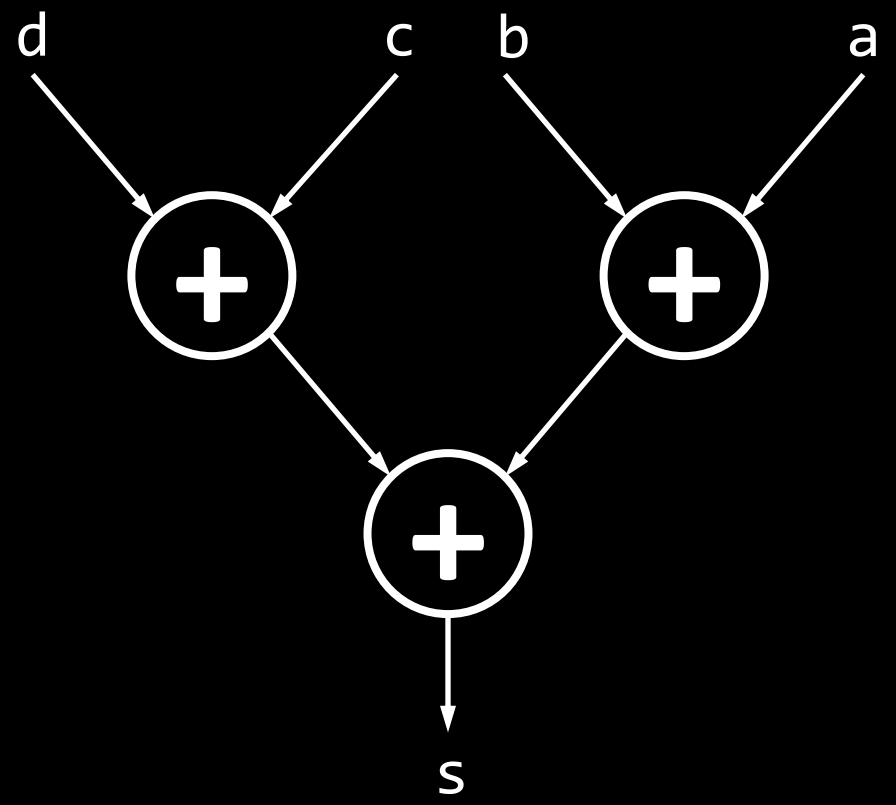
	Type	Efficiency	Strength n: Cascade Length	Cascadable	Lookahead
Old counters	(2) Atom	1	$2 - 1/(n+1)$	Yes	Yes
	(4,1) Atom	1.5	$2.5 - 1/(n+1)$	Yes	Yes
	~~*(0,6) Atom*~~	~~*2*~~	~~$3 - 2/(2n+1)$~~	~~---~~	
	(6:3]	1	2	No	
	(2,5:1,2,1]	*1.5*	1.75	No	
New counters	Ripple Sum	1	$2 - 1/(n+1)$	Yes	No
	Double Ripple Sum	**1.5**	$2.5 - 1/(n+1)$	Yes	No
	(10:4,2]	1.3	1.66	No	
	(2,2,2) Atom	**1.5**	$2 - 1/(3n+1)$	Yes	No
	5:2 Adder	**1.5**	$2.5 - 1/(n+1)$	Yes	Yes
Final Adder	~~*Ternary Adder*~~	~~***2***~~	~~$3 - 2/(n+1)$~~	~~-----~~	
	Quaternary Adder	**1.5**	$4 - 6/(n+2)$	Yes	Yes

Motivation

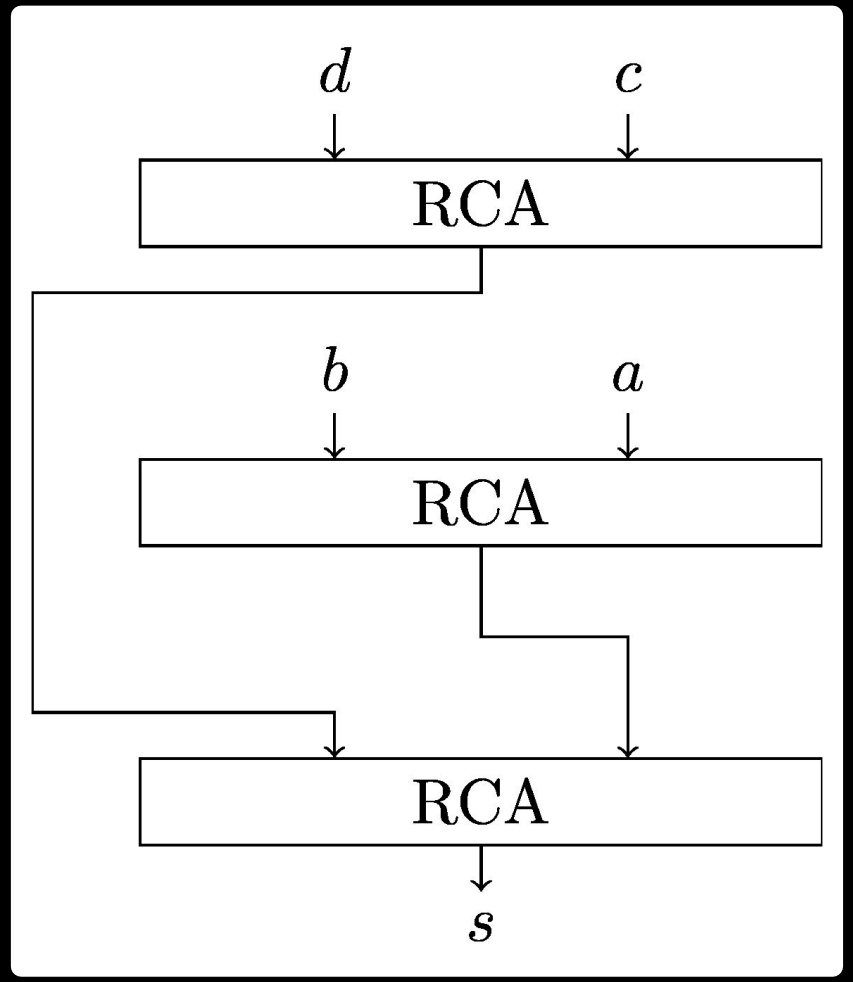


- ***Compact*** ternary addition in same footprint as binary RCA is no longer possible in Versal CLB.
- Improve beyond a balanced binary adder tree for a quaternary adder.

Binary Adder Tree for Quaternary Addition



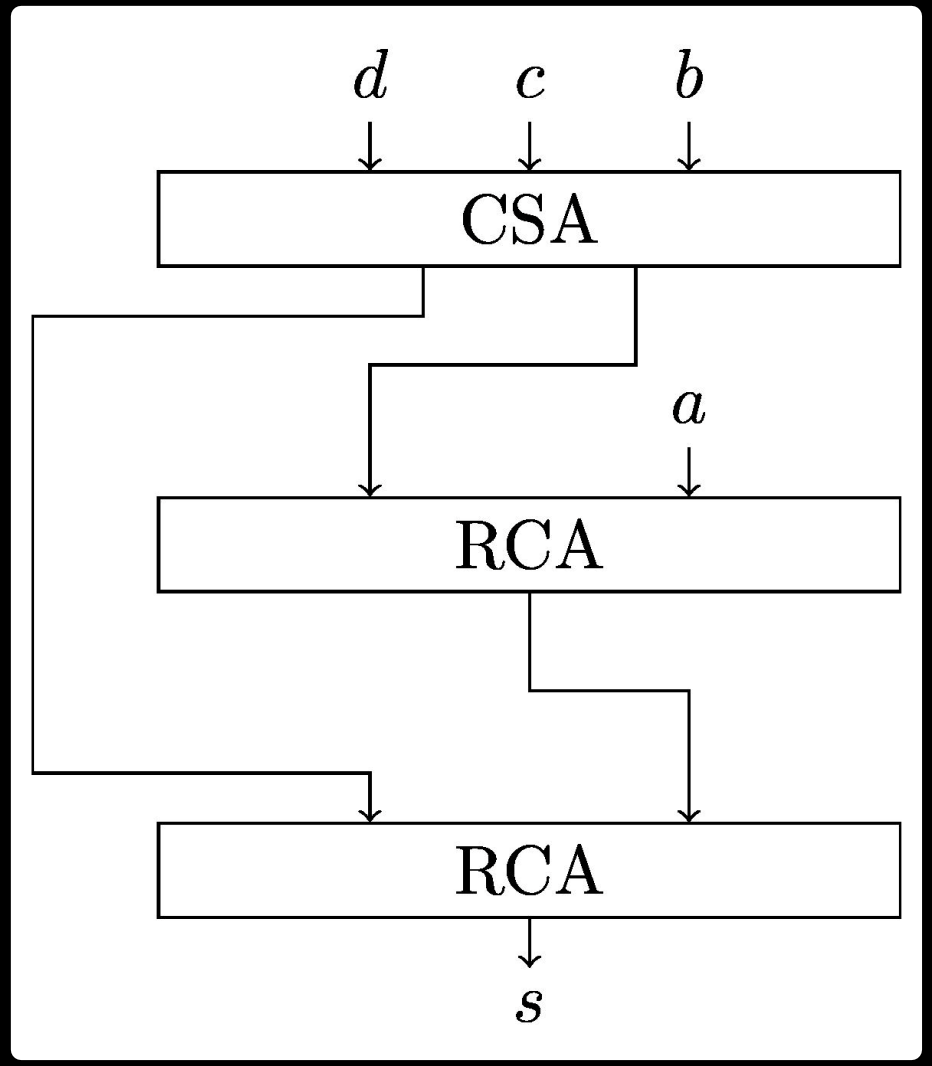
Binary Adder Tree Characteristics



$$C = 3 \cdot n$$
$$D = 2 \cdot T + n \cdot \tau$$

- Complexity linear in operand width.
- Delay according to logarithmic tree height with a small linear contribution through carry chain.

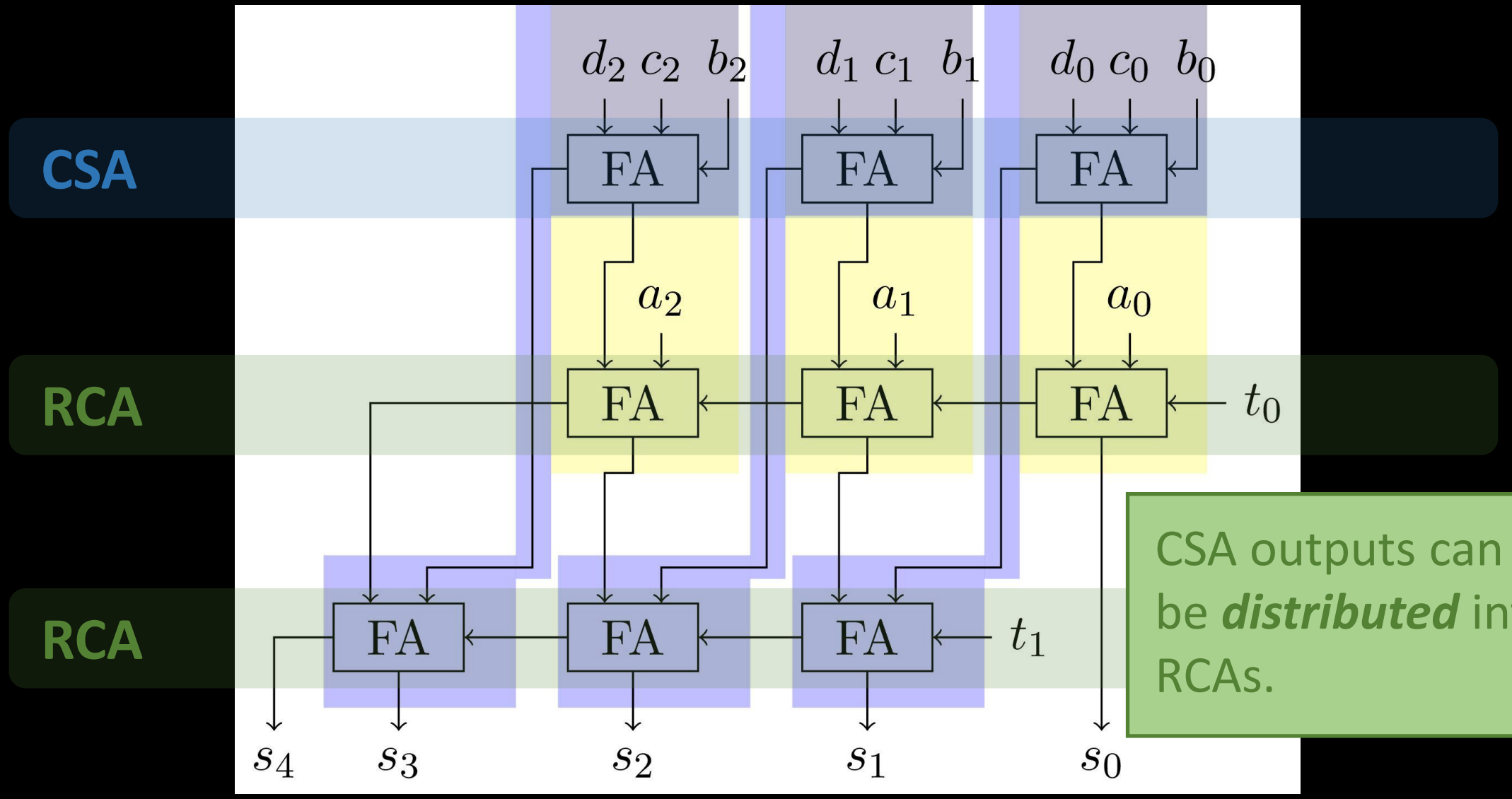
Alternative with Initial Carry-Save Reduction



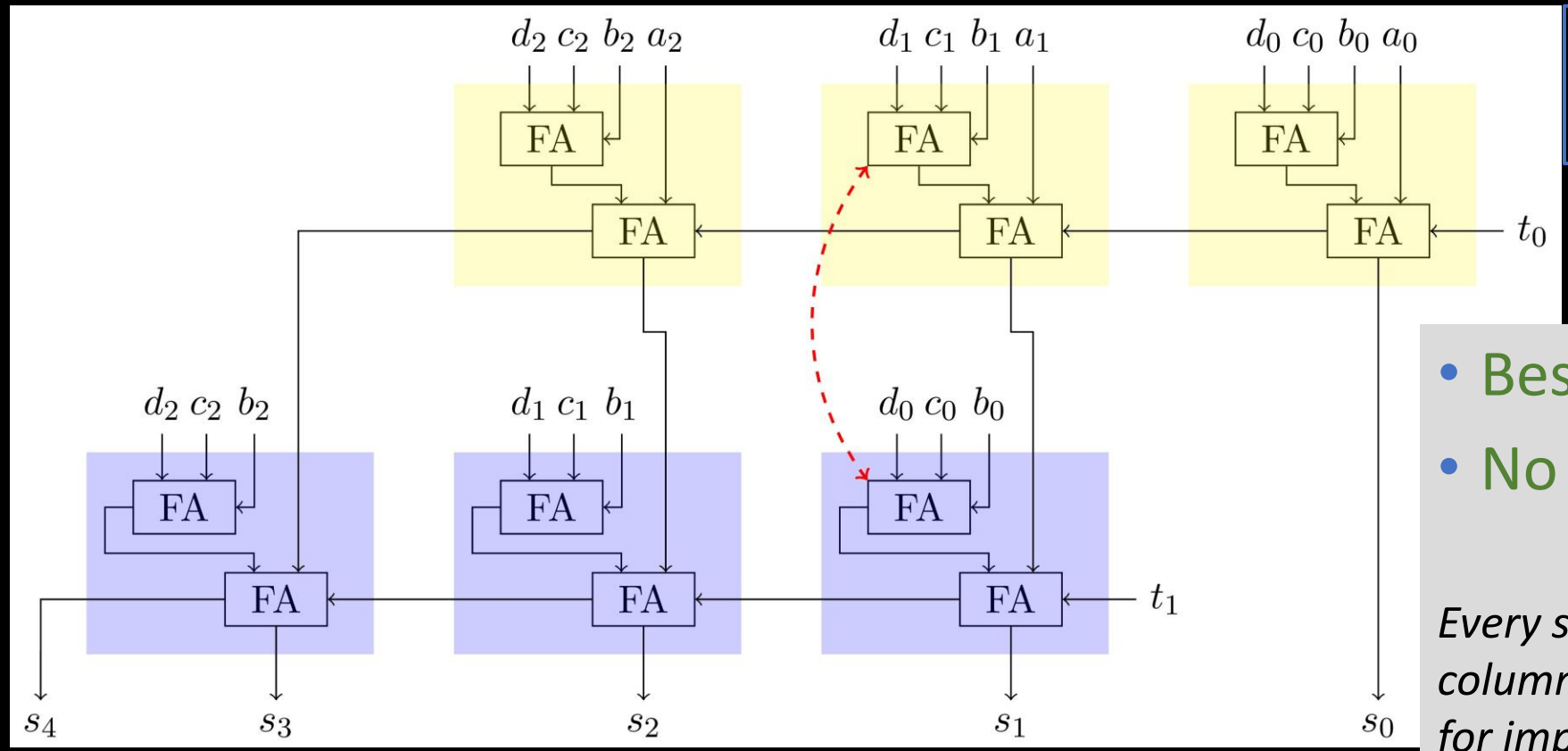
$$C = 3 \cdot n$$
$$D = 3 \cdot T + n \cdot \tau$$

- Complexity unchanged.
- Delay suffers additional LUT level.

Initial Carry-Save Reduction Spelled Out



LUT Mapping of Quaternary Adder



$$C = 2 \cdot n$$
$$D = 2 \cdot T + n \cdot \tau$$

- Best complexity.
- No delay penalty.

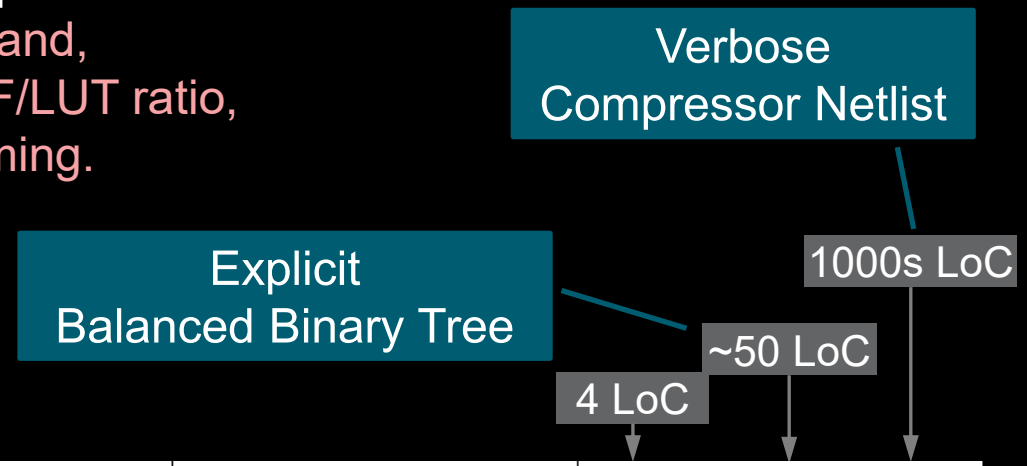
Every second CSA output column can be switched around for improved slice fanin locality.

Mainstreaming Compressors in Vivado

Multi-Operand Addition in Vivado 2025.1

```
always_comb begin
    sum0 = 0;
    foreach(arg[i])  sum0 += arg[i];
end
```

Behavioral description would be *trivial* but
- High LUT demand,
- Unbalanced FF/LUT ratio,
- Challenging timing.



Improvement of metrics
- Demonstrates significant headroom
- After investing significant effort.

PExSIMD_WxA	12x60_n3x6			14x72_w3x5			16x32_n3x4			20x56_w4x4			23x47_w3x3		
	Loop	Tree	Comp	Loop	Tree	Comp	Loop	Tree	Comp	Loop	Tree	Comp	Loop	Tree	Comp
LUT	14840	11413	8521	19397	15910	11509	7404	7216	5476	20291	17205	12701	15763	14805	11002
FF	6836	17278	16260	9505	31686	22877	5951	10607	10747	11907	27658	25593	11146	25653	21820
DSP	180	180	180	288	288	288	128	128	128	280	280	280	235	235	235
WNS @250 MHz	0.000	1.362	0.998	0.025	0.510	0.359	0.747	0.887	0.971	0.057	1.449	0.741	0.410	1.536	0.338
FF/LUT	0.46	1.51	1.91	0.49	1.99	1.99	0.80	1.47	1.96	0.59	1.61	2.02	0.71	1.73	1.98
LUT (vs. Loop)	1.00	0.77	0.57	1.00	0.82	0.59	1.00	0.97	0.74	1.00	0.85	0.63	1.00	0.94	0.70
FF (vs. Loop)	1.00	2.53	2.38	1.00	3.33	2.41	1.00	1.78	1.81	1.00	2.32	2.15	1.00	2.30	1.96

Multi-Operand Addition since Vivado 2025.2

PExSIMD_WxA	12x60_n3x6			14x72_w3x5			16x32_n3x4			20x56_w4x4			23x47_w3x3		
	Loop	Tree	Comp	Loop	Tree	Comp	Loop	Tree	Comp	Loop	Tree	Comp	Loop	Tree	Comp
LUT	14840	11413	8521	19397	15910	11509	7404	7216	5476	20291	17205	12701	15763	14805	11002
FF	6836	17278	16260	9505	31686	22877	5951	10607	10747	11907	27658	25593	11146	25653	21820
DSP	180	180	180	288	288	288	128	128	128	280	280	280	235	235	235
WNS @250 MHz	0.000	1.362	0.998	0.025	0.510	0.359	0.747	0.887	0.971	0.057	1.449	0.741	0.410	1.536	0.338
FF/LUT	0.46	1.51	1.91	0.49	1.99	1.99	0.80	1.47	1.96	0.59	1.61	2.02	0.71	1.73	1.98
LUT (vs. Loop)	1.00	0.77	0.57	1.00	0.82	0.59	1.00	0.97	0.74	1.00	0.85	0.63	1.00	0.94	0.70
FF (vs. Loop)	1.00	2.53	2.38	1.00	3.33	2.41	1.00	1.78	1.81	1.00	2.32	2.15	1.00	2.30	1.96

Use synthesis directive LogicCompaction

```
always_comb begin
        sum0 = 0;
        foreach(arg[i])  sum0 += arg[i];
end
(* retiming_compressor = "TRUE" *)
logic [W-1:0]  Sum[DEPTH] = '{ default: 'x };
always_ff @(posege clk)  Sum <= { sum0, Sum[0:DEPTH-2] };
```

- Sophisticated design mapping ***for free***.
- Encouraging ***high-level entry*** giving Vivado better grasp on design semantics and optimizations.

```
set opt_par synth.elaboration.rodinMoreOptions
set_param $opt_par \
"[get_param $opt_par];rt::set_parameter compressorTreePipe 1"
```

Optional ***auto-pipelining*** using retiming registers for performance.

Summary

- Compressor tree generation is a valuable technique for efficient FPGA fabric arithmetic.
- Result quality hinges on an available GPC suite tailored for the targeted fabric archticture.
- Direct tool integration makes the benefits attainable painlessly and it is happening in Vivado.

Thank you!

Disclaimer

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION

Copyrights & Relevant References

Cited References

[13] Martin Kumm, Peter Zipf: Efficient high speed compression trees on Xilinx FPGAs. Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV 2014), Cuvillier Verlag.

[14] Martin Kumm, Peter Zipf: Pipelined compressor tree optimization using integer linear programming. International Conference on Field Programmable Logic and Applications (FPL 2014), IEEE.

[32] Yuelai Yuan, Le Tu, Kan Huang, Xiaoqiang Zhang, Tiejun Zhang, Dihu Chen, Zixin Wang: Area optimized synthesis of compressor trees on Xilinx FPGAs using generalized parallel counters. IEEE Access 7 (2019).

Based on

Konstantin Hoßfeld, Hans Jakob Damsgaard, Jar Nurmi, Michaela Blott, Thomas B. Preußer: *High-Efficiency Compressor Trees for Latest AMD FPGAs*. ACM TRETS, June 2024.
https://doi.org/10.1145/3645097


AMD