LUXOR: An FPGA Logic Cell Architecture for Efficient Compressor Tree Implementations

SeyedRamin Rasoulinezhad [1], Siddhartha [1], Hao Zhou [2],
Lingli Wang [2], David Boland [1], and Philip H.W. Leong [1]

[1] School of Electrical and Information Engineering, The University of Sydney,
[2] State Key Lab of ASIC and System, Fudan University

THE UNIVERSITY OF SYDNEY


Luxor Temple is an Ancient Egyptian temple complex located on the east bank of the Nile River

THE UNIVERSITY OF SYDNEY

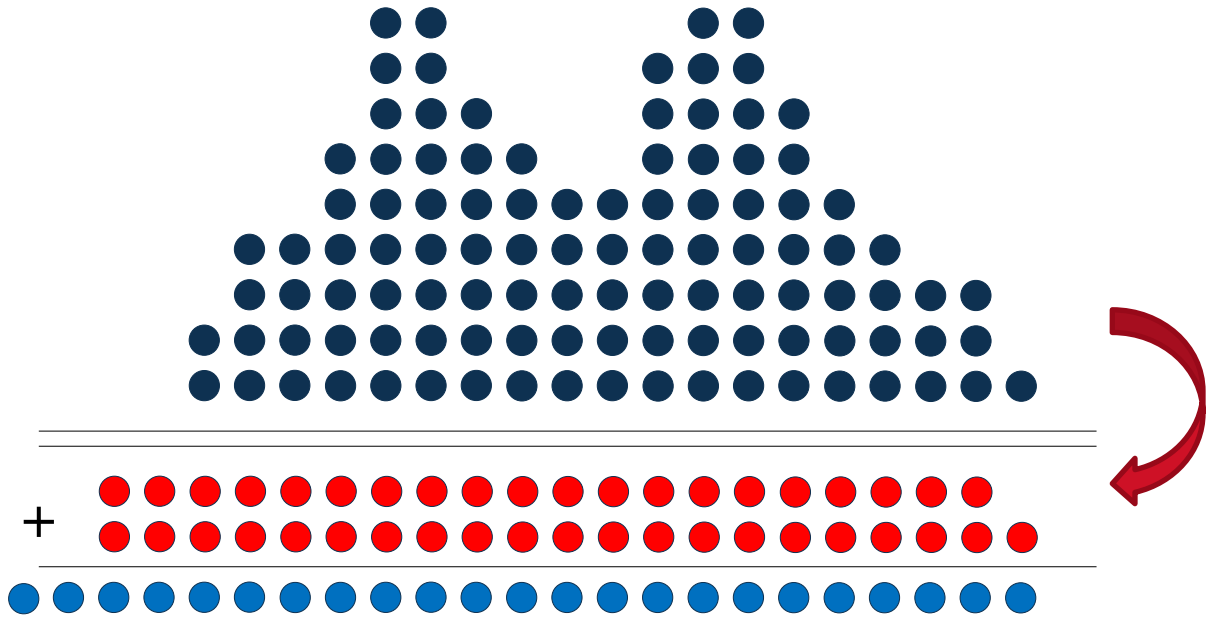
Overview

- Introduction
- LUXOR Architectural Modifications
- Results
- Conclusion

Background

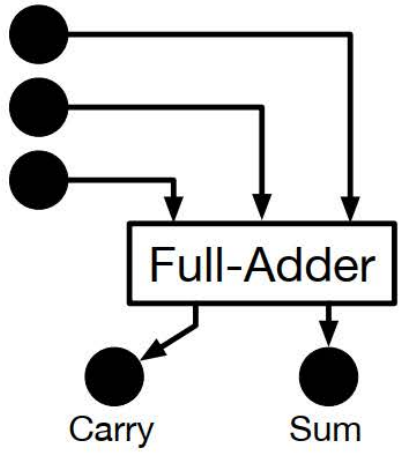
› Parallel computer arithmetic and **Compressor Trees**

- A circuit that takes in a set of binary values (or dots) that represent multiple operands, and outputs the result as a sum and carry.
- Final result involves an addition of binary numbers.

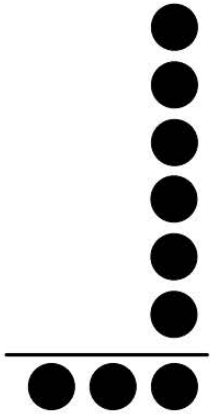


Generalized Parallel Counters

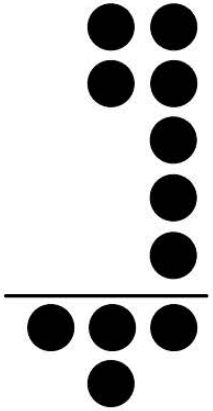
- Notation: (p_{n-1}, ...,p_1,p_0 : q_{m-1}, ...,q_1,q_0) where:
  - p_i is the number of input bits in the i^{th} column
  - q_j is the number of output bits in the j^{th} column.



(3:11)



(6:111)

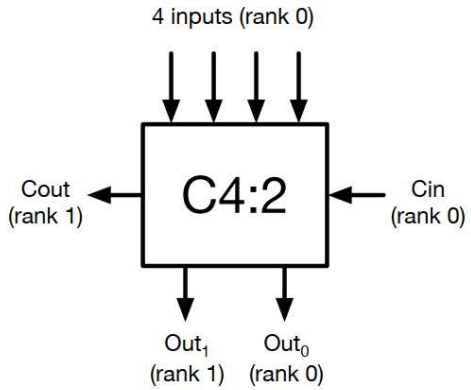


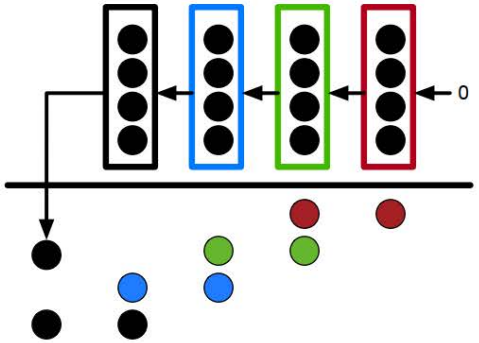
(25:121)

Primitives 2

› Compressors

- Connected in a carry-save manner (Cout is not connected to Cin).
- (4:2) compressor is the only Xilinx FPGA-friendly Kumm et al. [11]

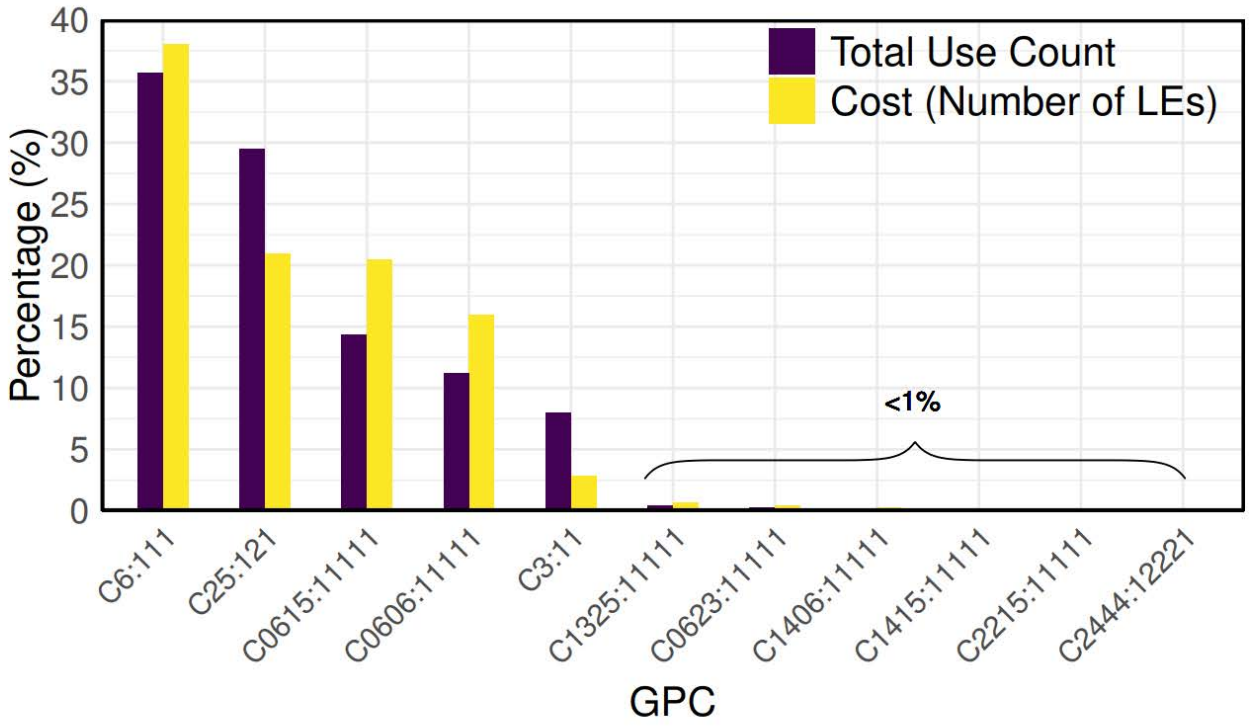




- For brevity, we describe adders/compressors/GPCs with a simplified notation (which doesn't distinguish whether it has carry in and out):
  - GPC (6:1,1,1) as **C6:111**,
  - Compressor (4:2) as **C4:2**

Motivation

- FPGA lookup table (LUT) based architectures are not particularly efficient for implementation of compressor trees.
- **We observe that a small number of GPCs proposed by Preußer [22] are sufficient: C6:111, C25:121 -> improving these will improve performance**



[22] Thomas B. Preußer. Generic and universal parallel matrix summation with a flexible compression goal for Xilinx FPGAs. *FPL* 2017

THE UNIVERSITY OF SYDNEY

Aim

- Enhance FPGA logic elements for better performance and utilization
    1. Vendor-agnostic: LUXOR
    2. Vendor-specific: I-LUXOR+ and X-LUXOR+

THE UNIVERSITY OF SYDNEY

Overview

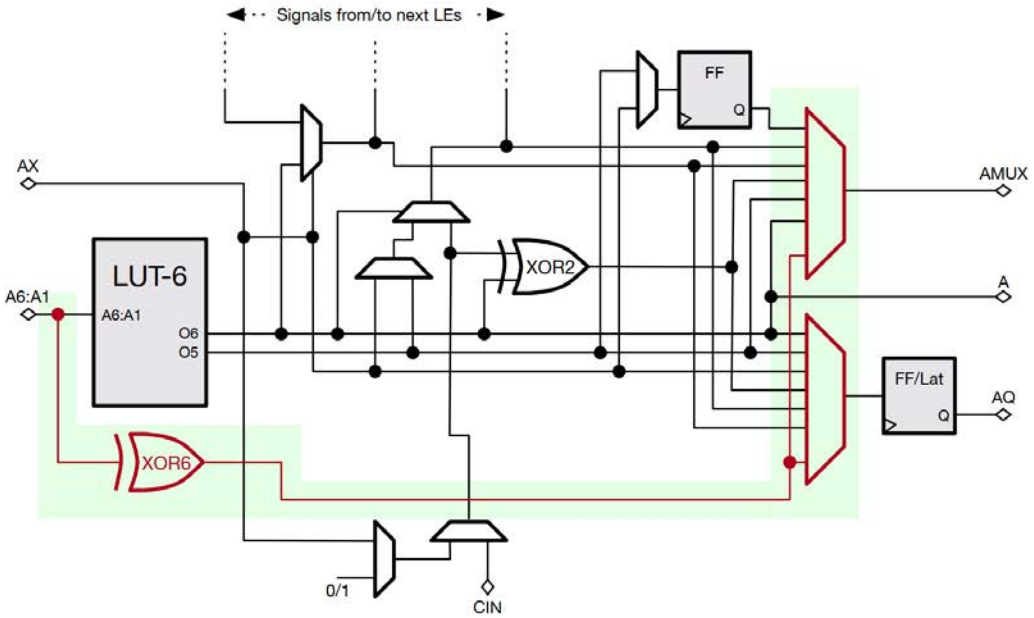
- Introduction
- LUXOR Architectural Modifications
- Results
- Conclusion

LUXOR

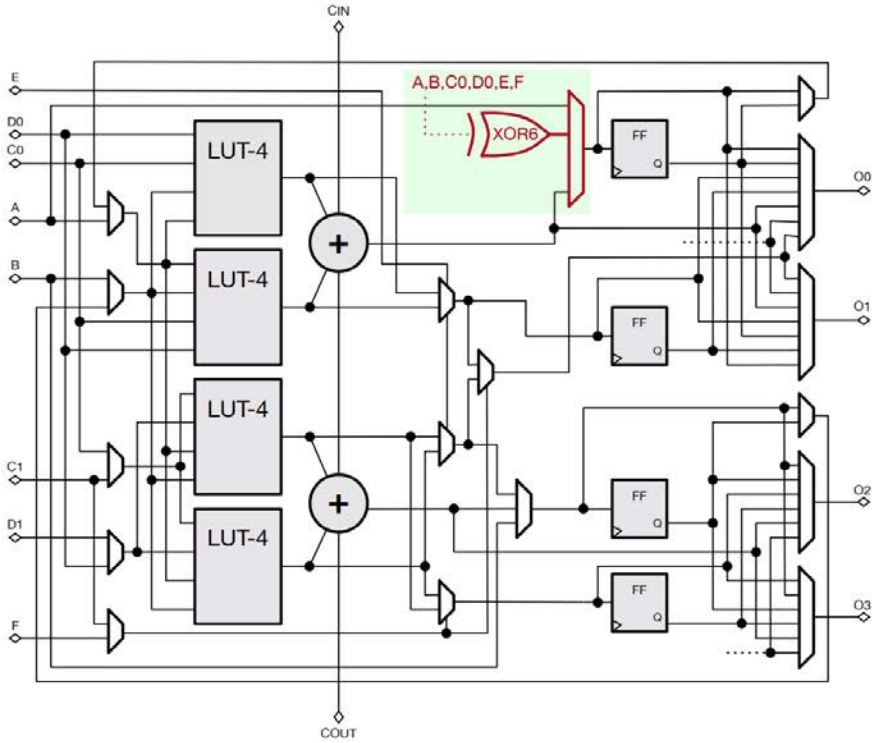
› Adding 6-input XOR gate in parallel to each 6-input LUT (shared inputs)

- Natively support C6:111 in 2x LEs (2x quarter slice or 2x ALM)
  - Cost of C6:111: 3LEs ➔ 2 LEs

Xilinx Quarter Slice

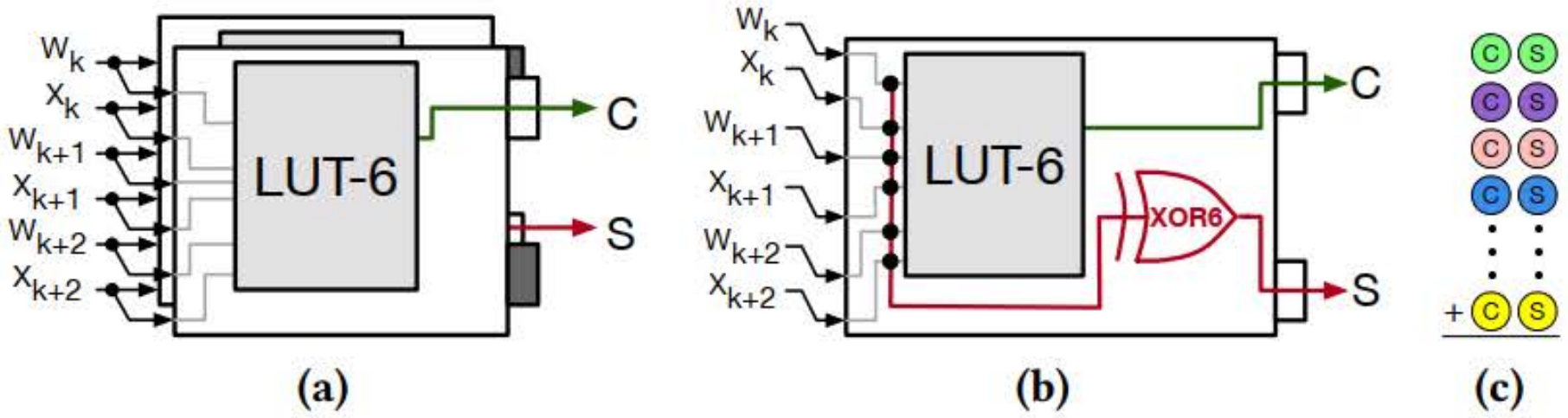


Intel ALM



Benefit of LUXOR - BNNs

Supports three-pair input XnorPopcount (binary MAC) in 1 LE



$$\text{Sum} = (w_0 \overline{\oplus} x_0) \oplus (w_1 \overline{\oplus} x_1) \oplus (w_2 \overline{\oplus} x_2)$$

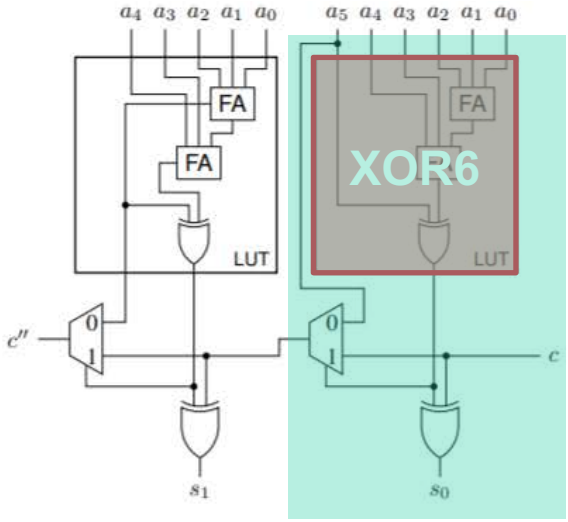
$$\text{Carry, C} = (w_0 \overline{\oplus} x_0) \cdot (w_1 \overline{\oplus} x_1) + (w_2 \overline{\oplus} x_2)[(w_0 \overline{\oplus} x_0) \oplus (w_1 \overline{\oplus} x_1)]$$

$$\text{Sum} = (\overline{w_0} \oplus x_0) \oplus (\overline{w_1} \oplus x_1) \oplus (\overline{w_2} \oplus x_2)$$

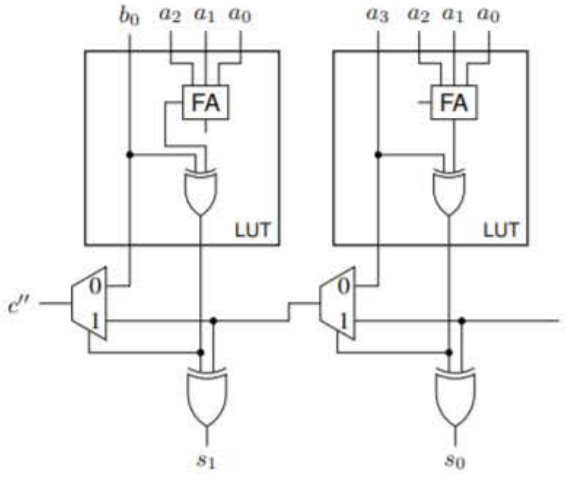
X-LUXOR+ Atoms

› Preußer [22]: slice-based GPCs based on three atoms -06-, -14-, and -22-

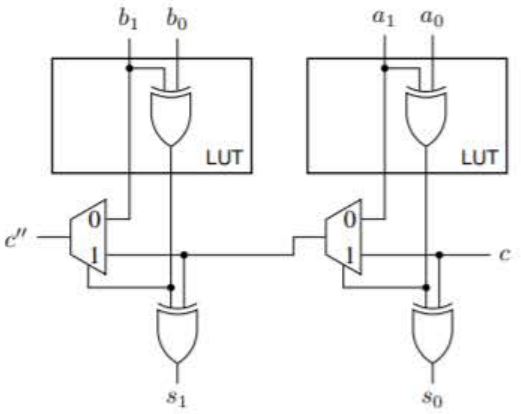
- Each atom maps to two LEs
- Atoms can be combined to make arbitrary slice based GPCs



Atom -06-



Atom -14-

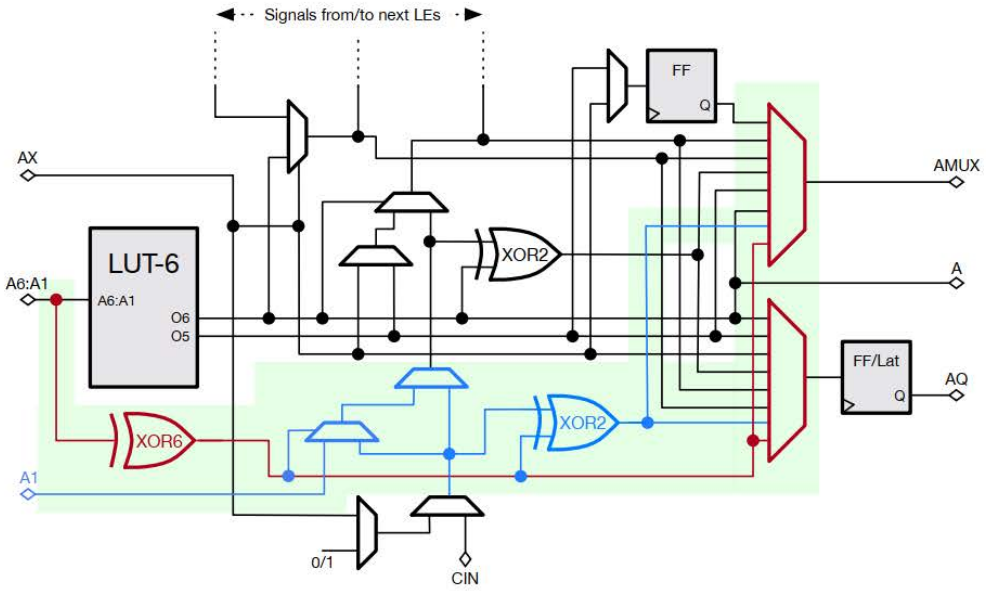


Atom -22-

[22] Thomas B. Preußer. Generic and universal parallel matrix summation with a flexible compression goal for Xilinx FPGAs. *FPL* 2017

X-LUXOR+

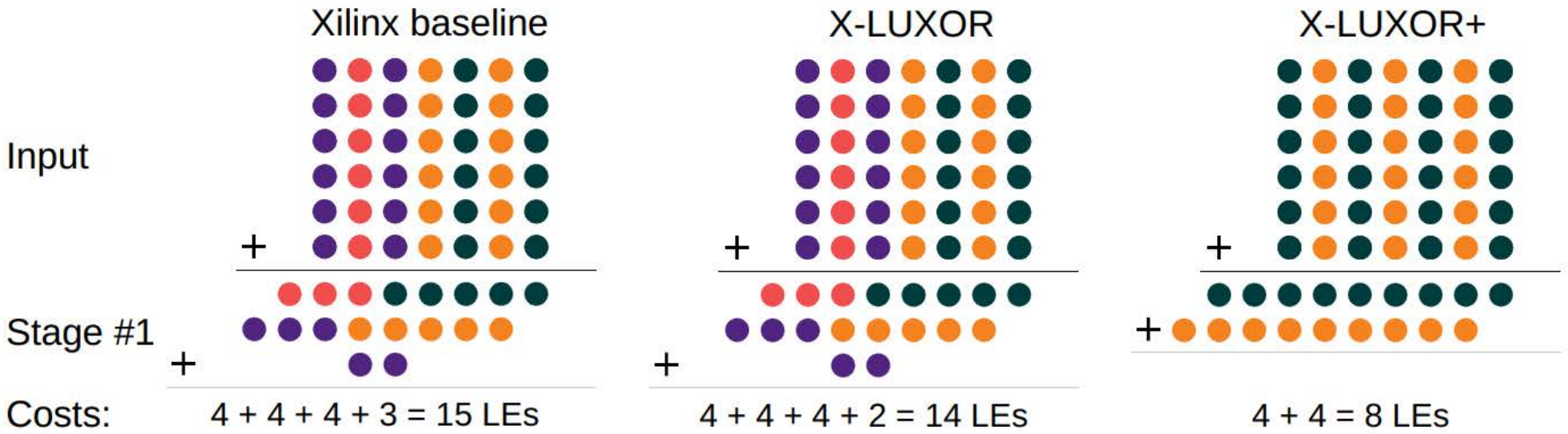
› New Atom -06- maps to a quarter Slice



	GPCs	*p*	*q*	LUTs	*E*	*S*
Baseline [11, 22]	C0606:11111	12	5	4	1.75	2.40
	C1415:11111	11	5	4	1.50	2.20
	C2215:11111	10	5	4	1.25	2.00
	C0615:11111	12	5	4	1.75	2.40
	C1423:11111	10	5	4	1.25	2.00
	C2223:11111	9	5	4	1.00	1.80
	C0623:11111	11	5	4	1.50	2.20
	C1406:11111	11	5	4	1.50	2.20
	C2206:11111	10	5	4	1.25	2.00
	C1325:11111	11	5	4	1.50	2.20
X-LUXOR+	C06060606:111111111	24	9	4	**3.75**	**2.67**
	C140606:1111111	17	7	4	2.50	2.43
	C220606:1111111	16	7	4	2.25	2.29
	C060606:1111111	18	7	4	2.75	2.57
	C060615:1111111	18	7	4	2.75	2.57
	C060623:1111111	17	7	4	2.50	2.43
	C061406:1111111	17	7	4	2.50	2.43
	C062206:1111111	16	7	4	2.25	2.29

Strength S = # inputs / # outputs
Efficiency E = (# inputs - #outputs) / #LEs

Xilinx vs. X-LUXOR vs. X-LUXOR+



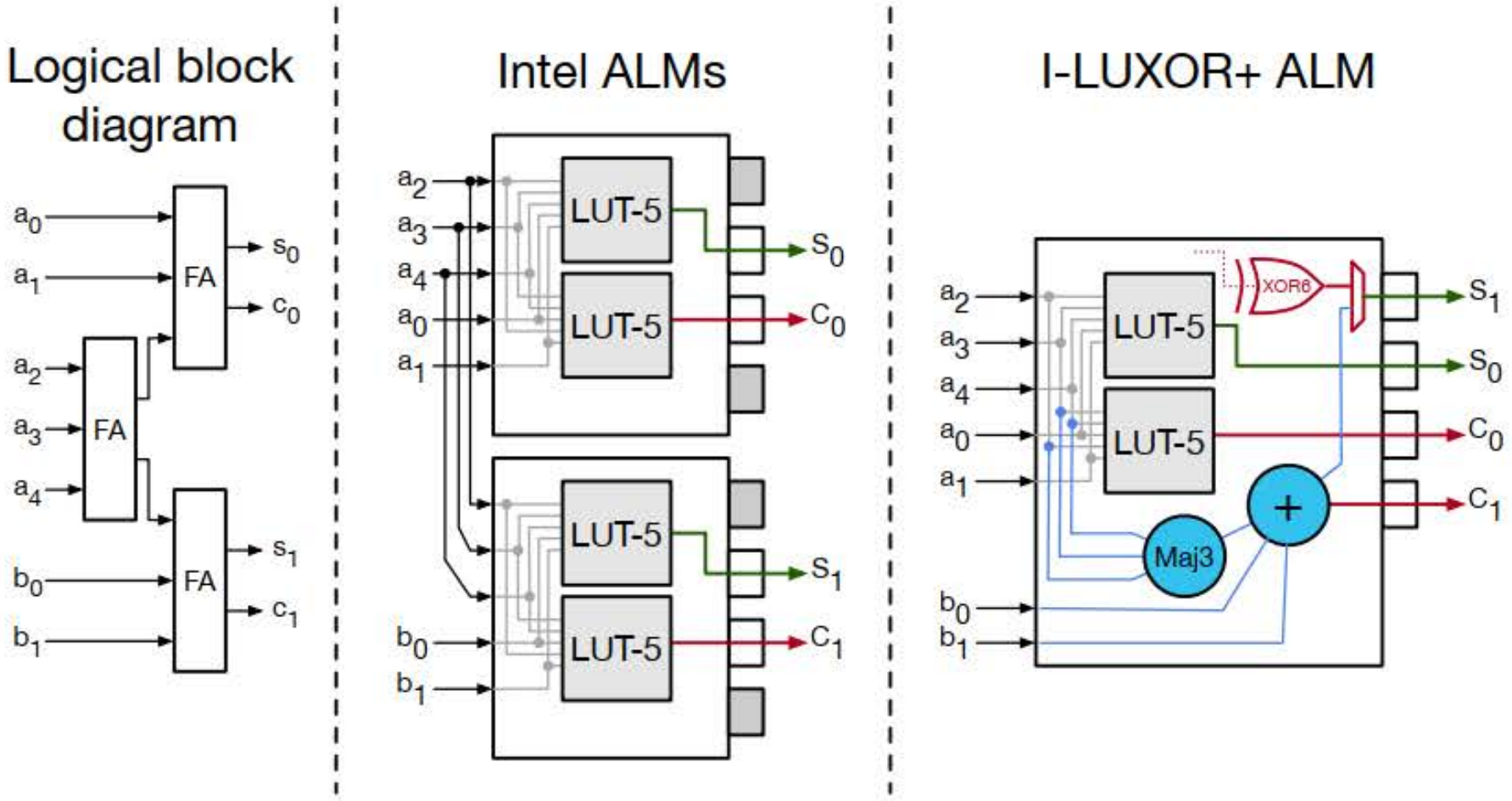
3x -0606- + 1x C6:111

3x -0606- + 1x C6:111

2x -06060606-

I-LUXOR+

Native support for C25:121 (one LE rather than 2 LEs)



THE UNIVERSITY OF SYDNEY

Overview

- Introduction
- LUXOR Architectural Modifications
- Results
- Conclusion

Overheads

- SMIC 65-nm technology standard cell by Synopsis Design Compiler 2013.12.
- Post synthesis results
  - Delay is critical path of a CLB/ALM

		Xilinx	X-LUXOR	X-LUXOR+
Area	um^2	6045	6002	6361
	ratio	1.00	0.99	1.06
Delay	*ns*	0.84	0.89	0.92
	ratio	1.00	1.06	1.09

		Intel	I-LUXOR	I-LUXOR+
Area	um^2	1680	1687	1767
	ratio	1	1.00	1.05
Delay	*ns*	1.42	1.44	1.46
	ratio	1	1.01	1.03

ILP-based Synthesizer

› Test cases mapped to LUXOR via CPLEX ILP which optimally solves compressor tree problem using a library of GPC and atom primitives

Test cases	[1]H1/H2/H3[22]		Our ILP Solver					
			Baseline		X-LUXOR		X-LUXOR+	
	[2]LE	[2]Stage	LE	Stage	LE	Stage	LE	Stage
S128	101/102/101	4/3/4	100	3	79	3	78	3
S256	209/209/209	4/4/4	195	4	159	4	154	4
S512	418/422/418	5/5/5	380	5	319	5	312	4
D128	178/205/178	5/4/5	168	4	156	4	150	4
D256	360/417/360	6/5/6	328	5	315	5	298	4
D512	721/839/721	7/6/7	709	5	631	5	586	5

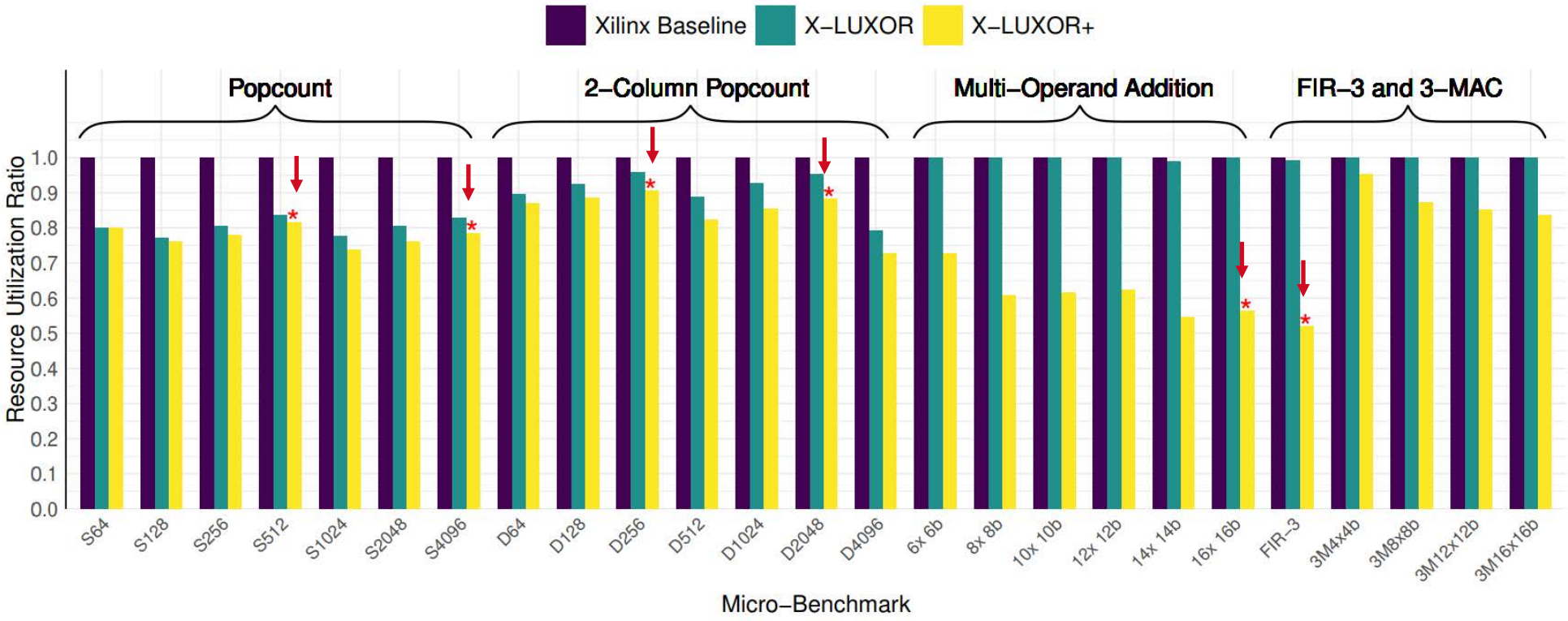
[1]Heuristics used in [22]: Efficiency/Strength/Product, reported in that order.

[2]LE = logic elements (LUTs), Stage = # of compressor tree stages

X-LUXOR(+) Performance on Micro Benchmarks

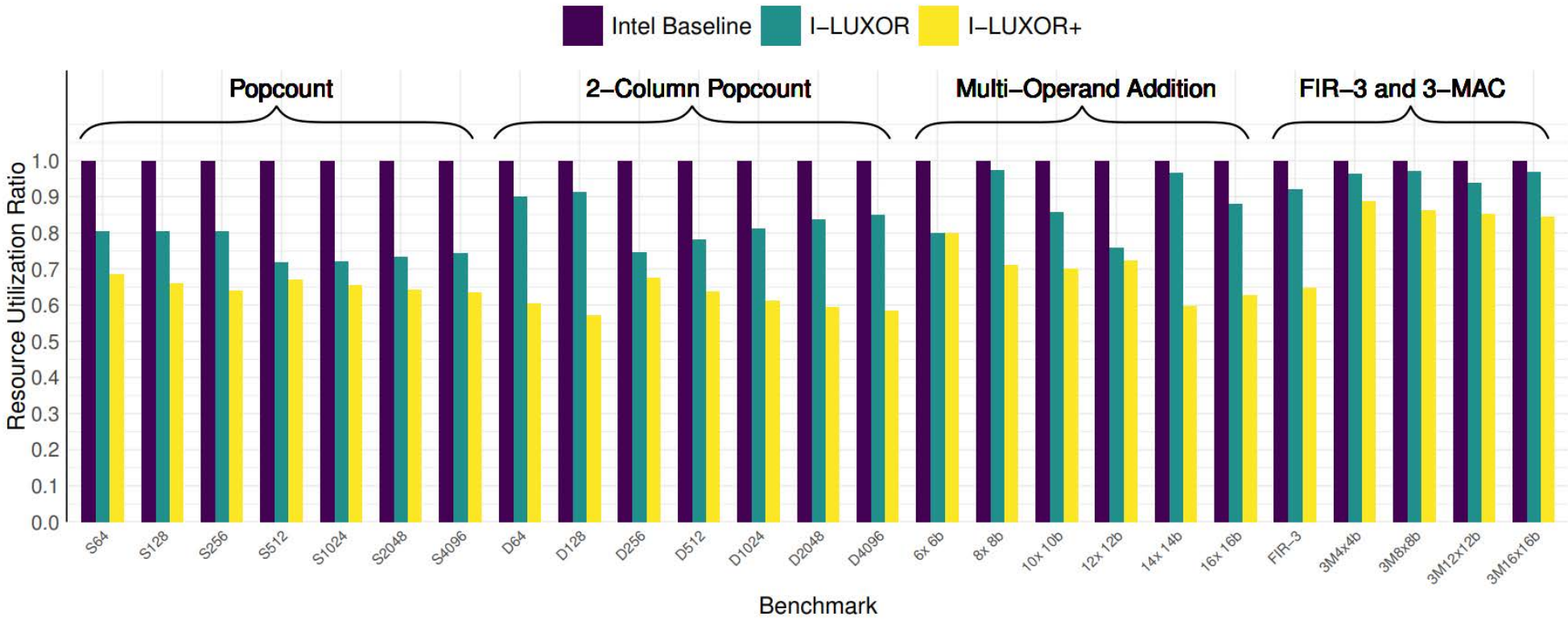
› Over 50+ microbenchmarks e.g. popcount, multi-operand addition, FIR, MAC etc

- Red dots: one less logic level



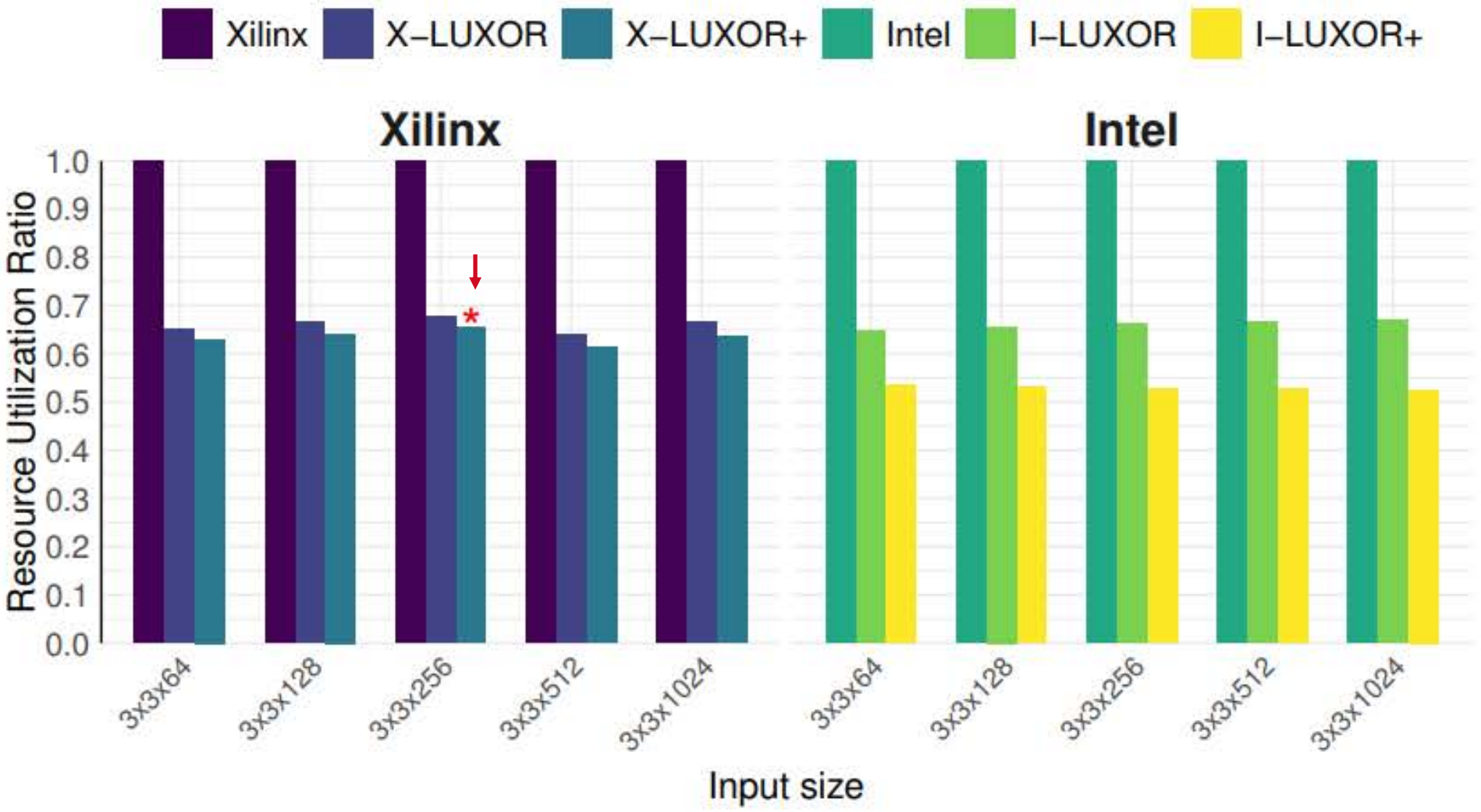
I-LUXOR(+) Performance on Micro Benchmarks

› No logic level reduction (doesn't implement atoms)



THE UNIVERSITY OF SYDNEY

LUXOR(+) Performance for BNNs



THE UNIVERSITY OF SYDNEY

Overview

- Introduction
- LUXOR Architectural Modifications
- Results
- Conclusion

THE UNIVERSITY OF SYDNEY

Conclusion

› Described

- Two tiers of modifications to FPGA logic cell architectures
    1. Vendor-agnostic: LUXOR
    2. Vendor-specific: I-LUXOR+ and X-LUXOR+
- A novel ILP-based compressor tree synthesizer


GitHub **github.com/raminrasoulinezhad/LUXOR_FPGA20**

› Results:

- LUXOR: with <0.5% area overhead offers up to 36% LUT reduction (average 12-19%)
- LUXOR+: 5-6% silicon area overhead, on average 26-34% LUT reduction
- LUXOR+: up to 48% more area efficient (for XnorPopcount operation)

Spare Slides

Enhancement for I-LUXOR(+) GPCs

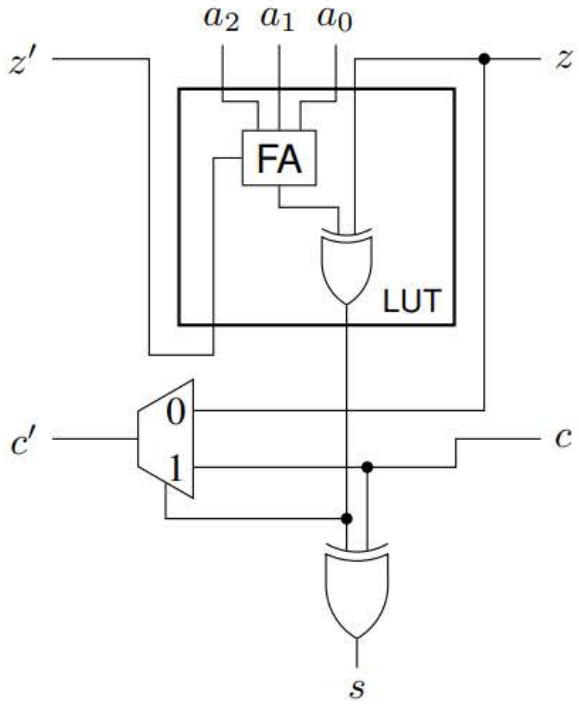
Table 3: Comparison of different GPCs proposed in [22] and new GPCs supported by I-LUXOR and I-LUXOR+

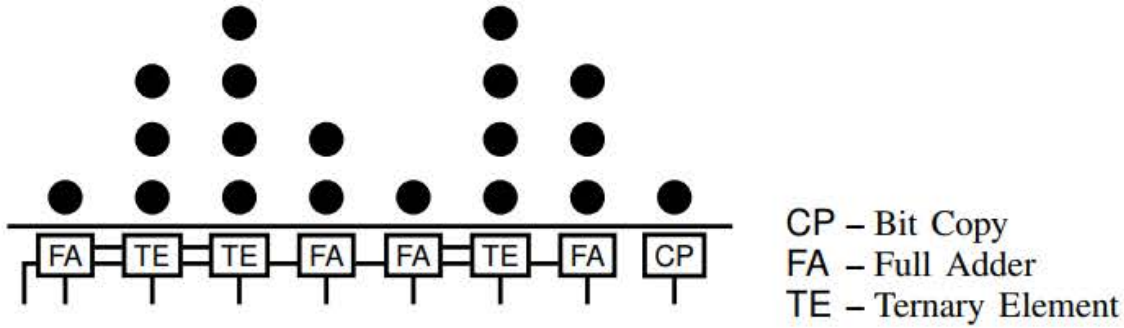
GPCs		*S*	*A*	Delay	LUTs	APD
[20]	C6:111	2	0.13	0.38	3	7.9
	C15:111	2	0	0.38	3	7.9
	C23:111	1.67	0	0.38	2	5.3
[22]	C25:121	1.75	0	0.38	2	11.8
Ours	C6:111	2	0.13	0.39*	2	10.95*
	C25:121	1.75	0	0.39*	1	21.9*

*Area/delay overheads for I-LUXOR+ are included (Section 5).

› Adder

- Ripple-carry / Carry-save
- Intel FPGAs: 3-input adders
- Xilinx FPGAs: Flexible Ternary adders (proposed by Preußer [22])





[22] Thomas B. Preußer. Generic and universal parallel matrix summation with a flexible compression goal for Xilinx FPGAs. *FPL* 2017

Flowchart of ILP-based Compressor tree synthesis

› Objective (minimize area)

$$\min \sum_{s=0}^{St-1} \sum_{c=0}^{C-1} \sum_{t=0}^{T-1} V_t R_{s,t,c}$$

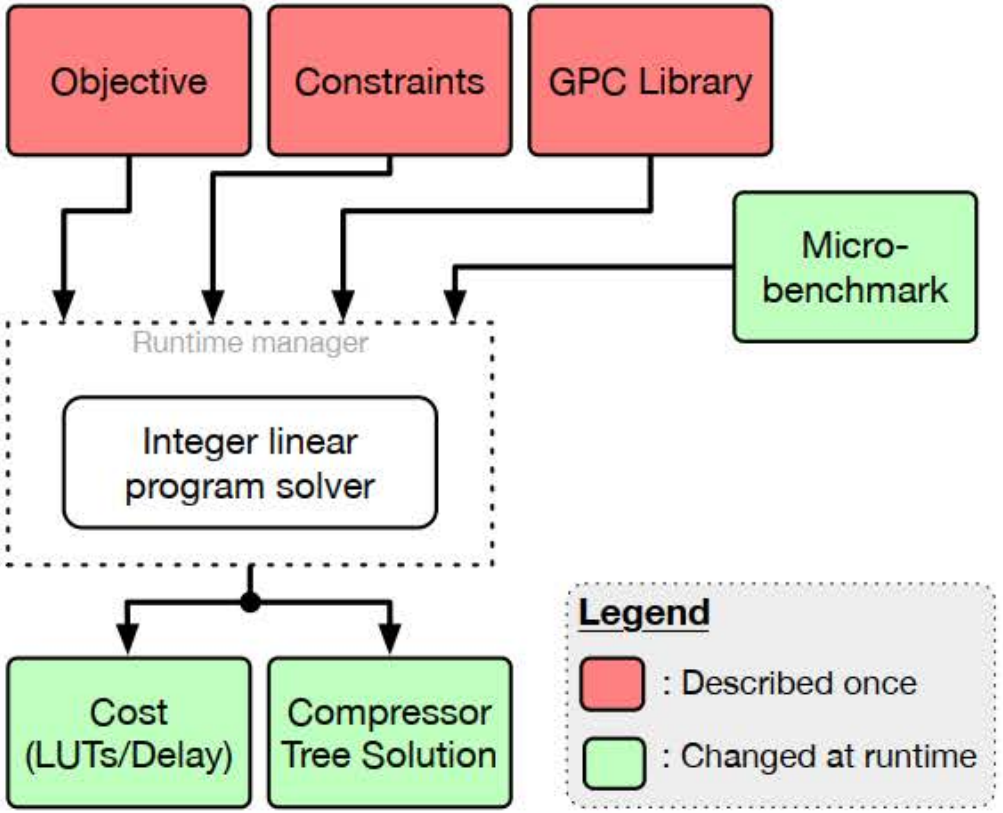
› Constraints (match I/O req)

$$N_{0,c} = X_c \qquad \text{for c = 0,1,2,...,C-1}$$

$$\sum_{t=0}^{T-1} \sum_{c'=0}^{O_t-1} M_{t,c'} * R_{s-1,t,c-c'} \geq N_{s-1,c}$$

$$\sum_{t=0}^{T-1} \sum_{c'=0}^{I_t-1} K_{t,c'} * R_{s-1,t,c-c'} = N_{s,c}$$

for c = 0,1,2,...,C-1
for s = 1,2,3,...,St-1



› GPC libraries (specs)

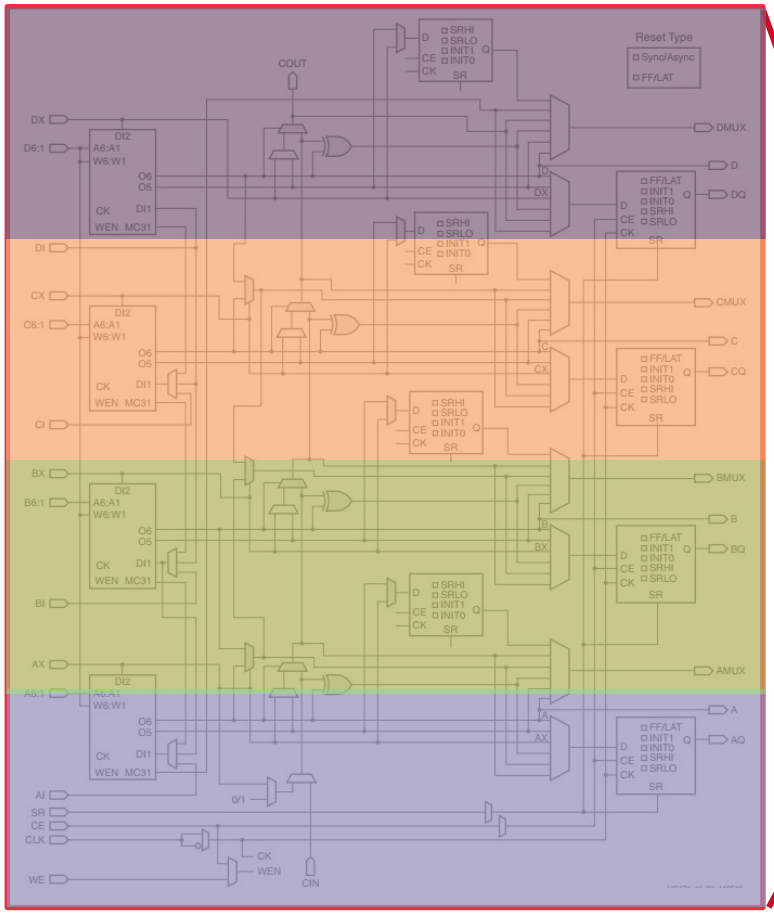
› Iteratively increase number of stages until we find a solution

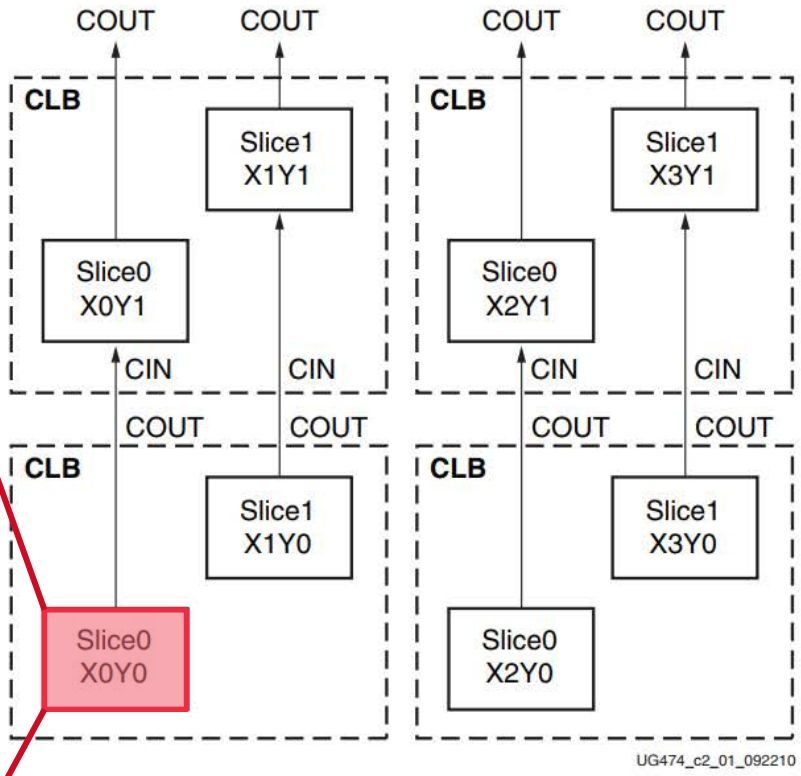
› Microbenchmark (ours + others collected from a number of different sources)

- Popcount / Double Popcount / Multi-Add / Multi-MAC / FIR filters / XnorPopcount

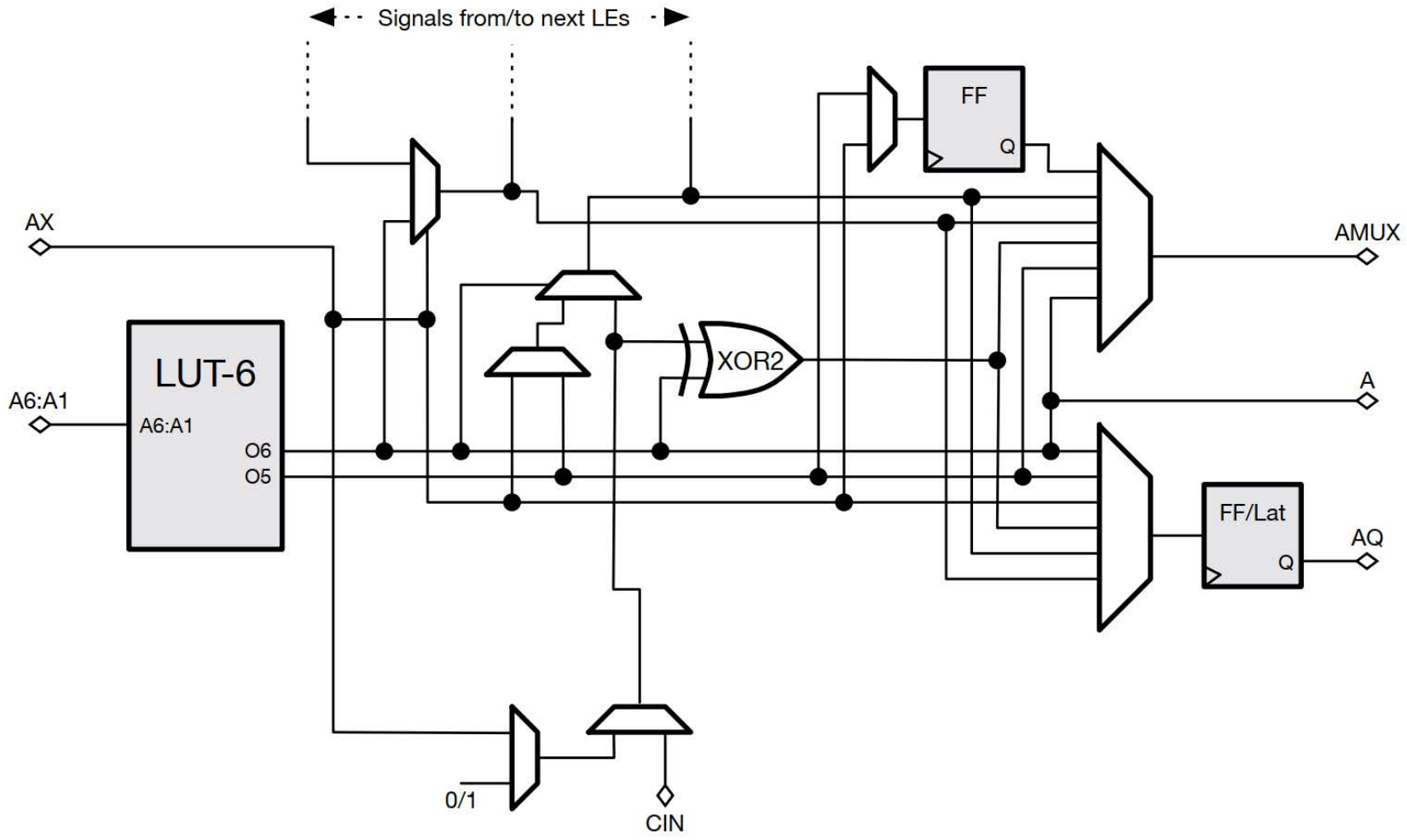
FPGA Logic Elements - Xilinx

The Xilinx Configurable Logic Block (CLB)



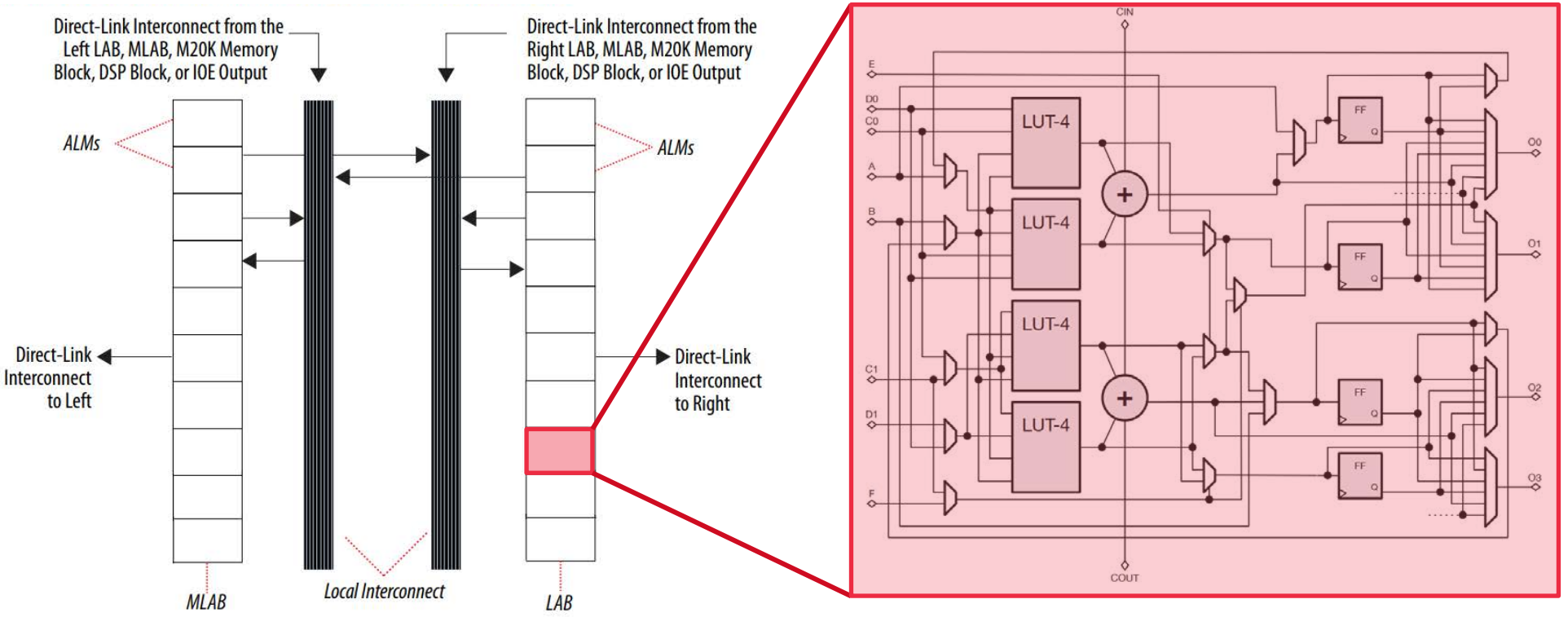


Xilinx – Quarter Slice



FPGA Logic Elements - Intel

Intel Adaptive Logic Modules



[8] Intel. UG-S10LAB Intel®Stratix®10 Logic Array Blocks and Adaptive Logic Modules User Guide, 9 2018