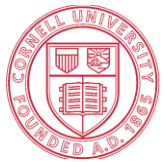
IMpress: Large Integer Multiplication Expression Rewriting for FPGA HLS

Ecenur Üstün[1], Ismail San[2,1], Jiaqi Yin[3], Cunxi Yu[3], Zhiru Zhang[1]

[1] Cornell University

[2] Eskisehir Technical University

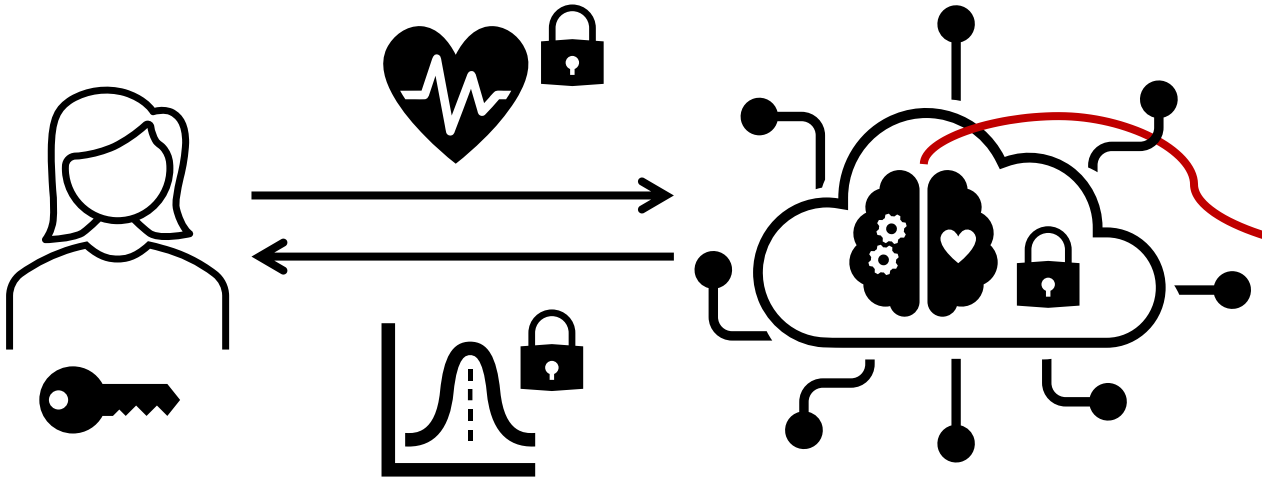
[3] University of Utah



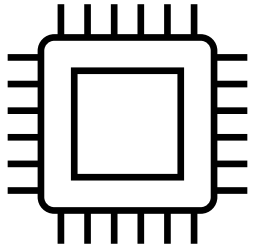
Cornell University

CSL

Large Integer Multiplication in Cryptography



Integer arithmetic of size 1K - 1M



lack of libraries
and compiler support

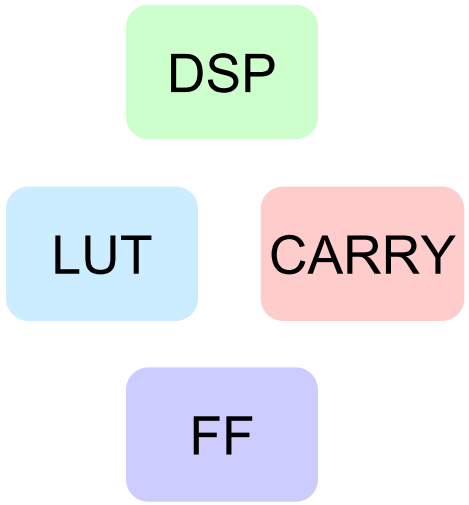
Kaplan-Meier survival analysis and
genome-wide association study [1]

Privacy-preserving machine learning [2][3]

[1] D. Froelicher *et al.* Truly Privacy-Preserving Federated Analytics for Precision Medicine with Multiparty Homomorphic Encryption. Nature Communications, 2021.
[2] N. Dowlin *et al.* CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. ICML, 2016.
[3] Microsoft SEAL github.com/Microsoft/SEAL

FPGA-Based Integer Multiplication

Heterogeneity



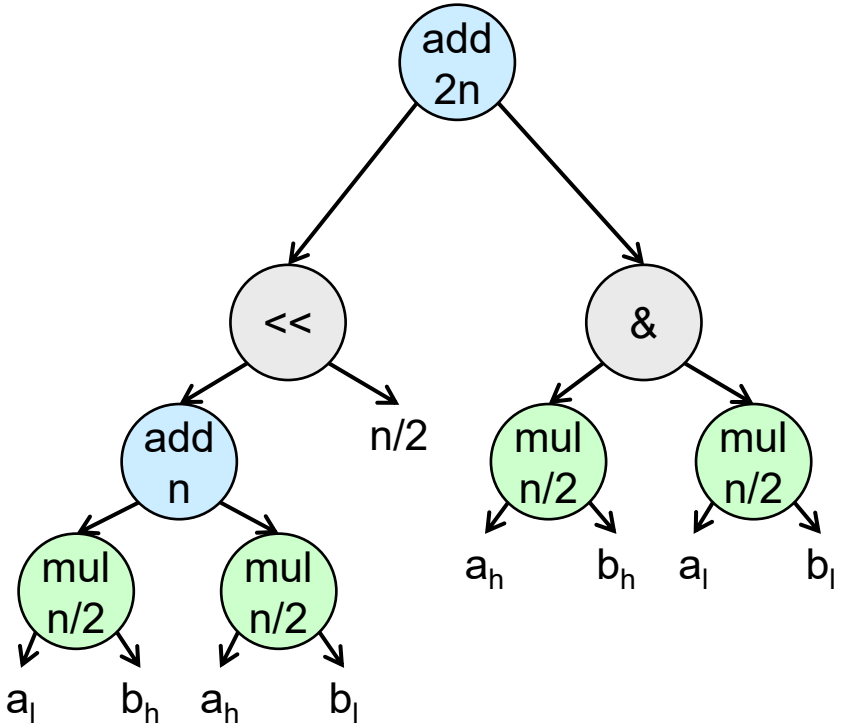
Default HLS Results

BW	DSP (DSP%)	LUT (LUT%)	CARRY (CARRY%)	FF (FF%)	Latency (cycles)	Frequency (MHz)
256	225 (1.83%)	225 (0.01%)	6 (<0.01%)	692 (0.02%)	4	18
512	900 (7.32%)	434 (0.03%)	6 (<0.01%)	1,329 (0.04%)	4	5
1,024	NA	NA	NA	NA	NA	NA
2,048	NA	NA	NA	NA	NA	NA

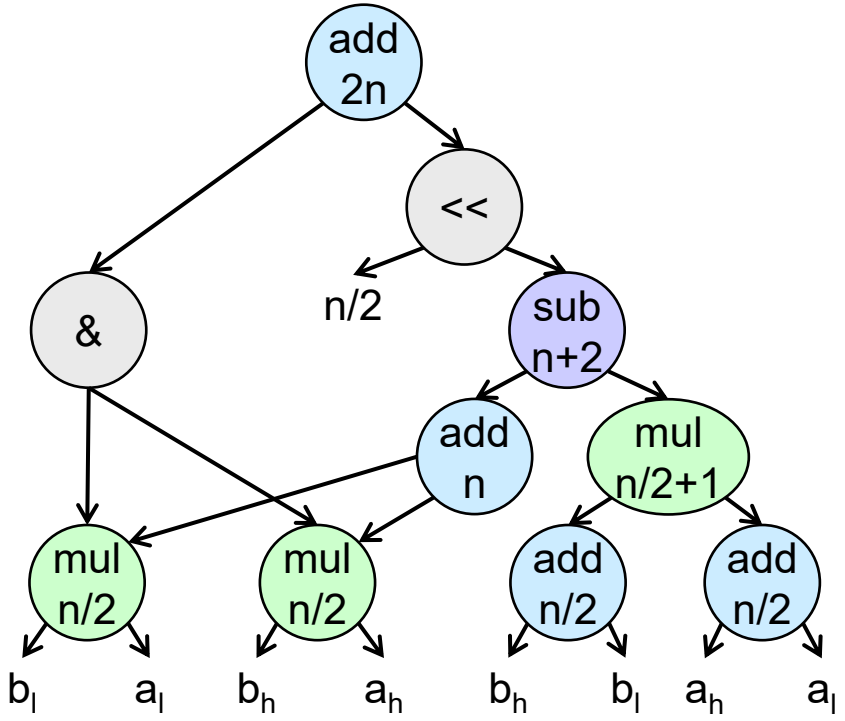
Target: AMD Xilinx Virtex Ultrascale+, 300 MHz

Existing Approaches to Optimize Multiplication

Based on decomposing

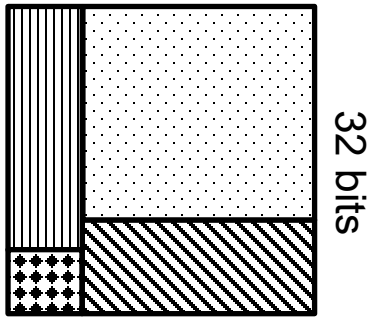


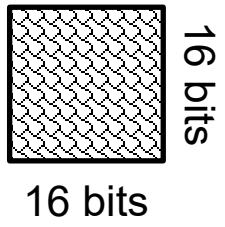
schoolbook(mul_n)



karatsuba(mul_n)

n	Implementation	DSP	LUT
256	HLS Def	225	225
	Schl	200	3,483
	Karat	164	4,339
512	HLS Def	900	434
	Schl	900	3,368
	Karat	675	5,182
1,024	HLS Def	NA	NA
	Schl	3,600	7,759
	Karat	2,700	12,012



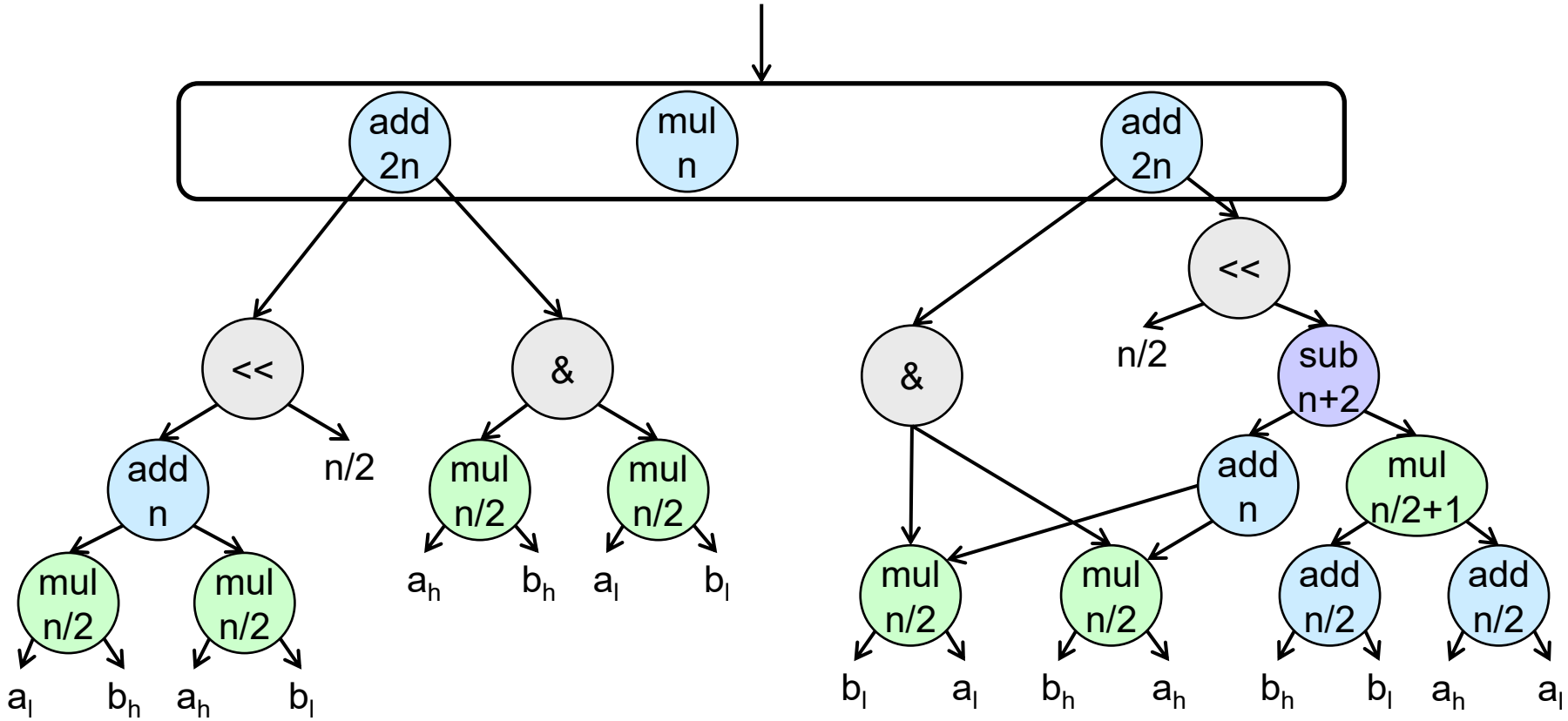


$$a \cdot b = a_h b_h 2^n + (a_h b_l + a_l b_h) 2^{n/2} + a_l b_l$$

$$a_h b_l + a_l b_h = (a_h + a_l)(b_h + b_l) - a_h b_h - a_l b_l$$

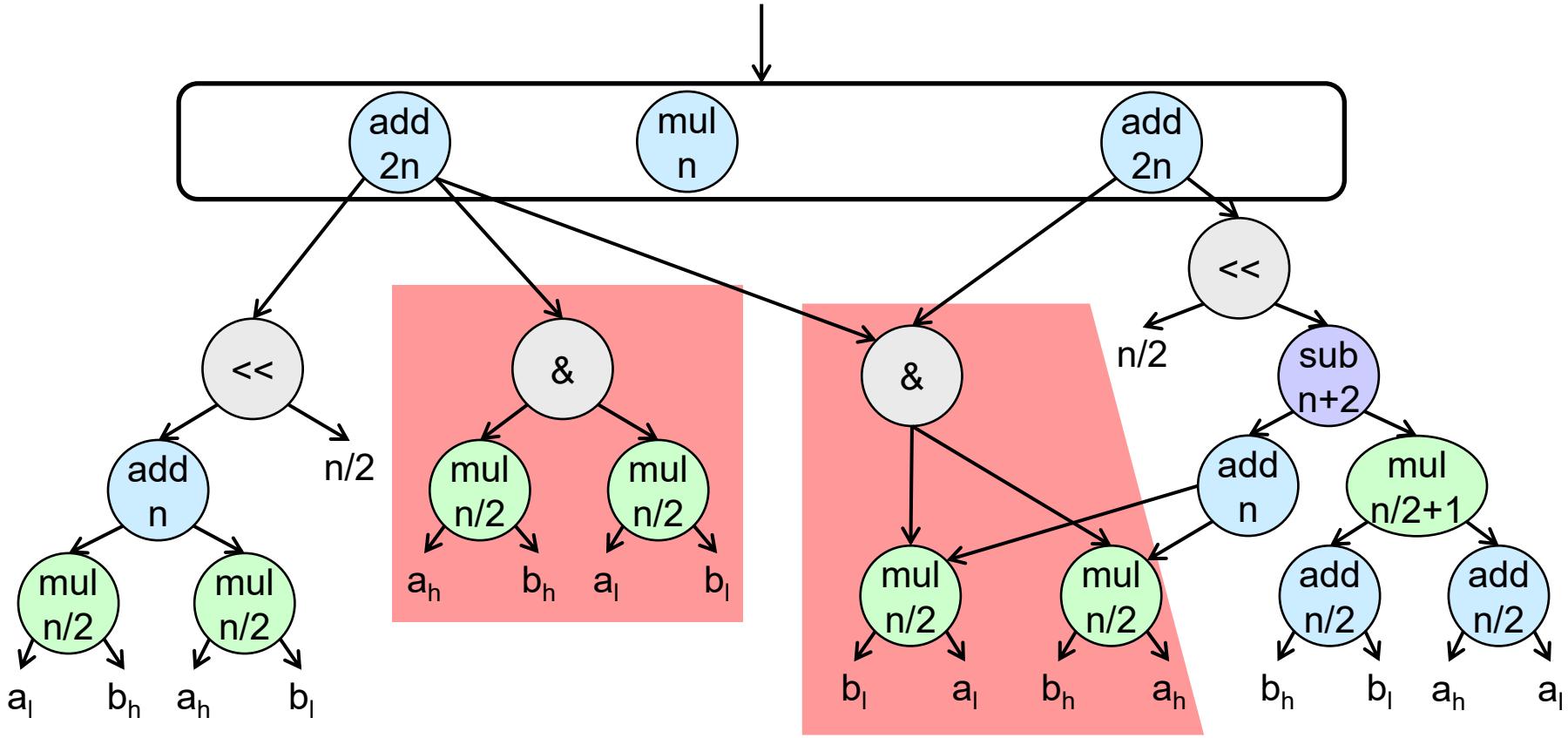
Equivalence Graph

Combine different ways of implementing an expression in one graph



Equivalence Graph (E-Graph)

Common sub-expressions

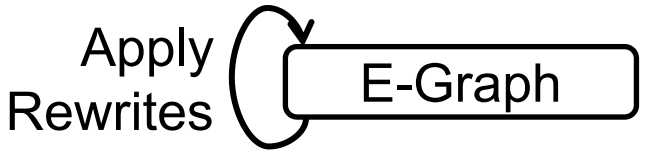


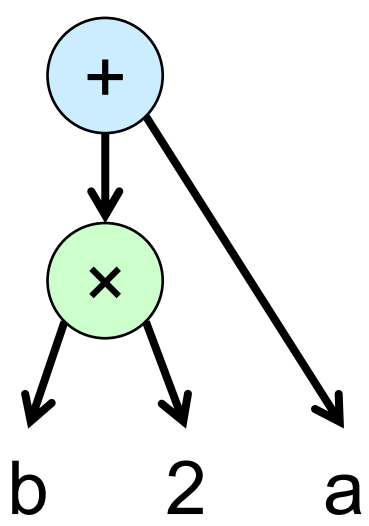
C. G. Nelson. Techniques for Program Verification. Stanford University, 1980.
R. Nieuwenhuis *et al.* Proof-Producing Congruence Closure. RTA, 2005.

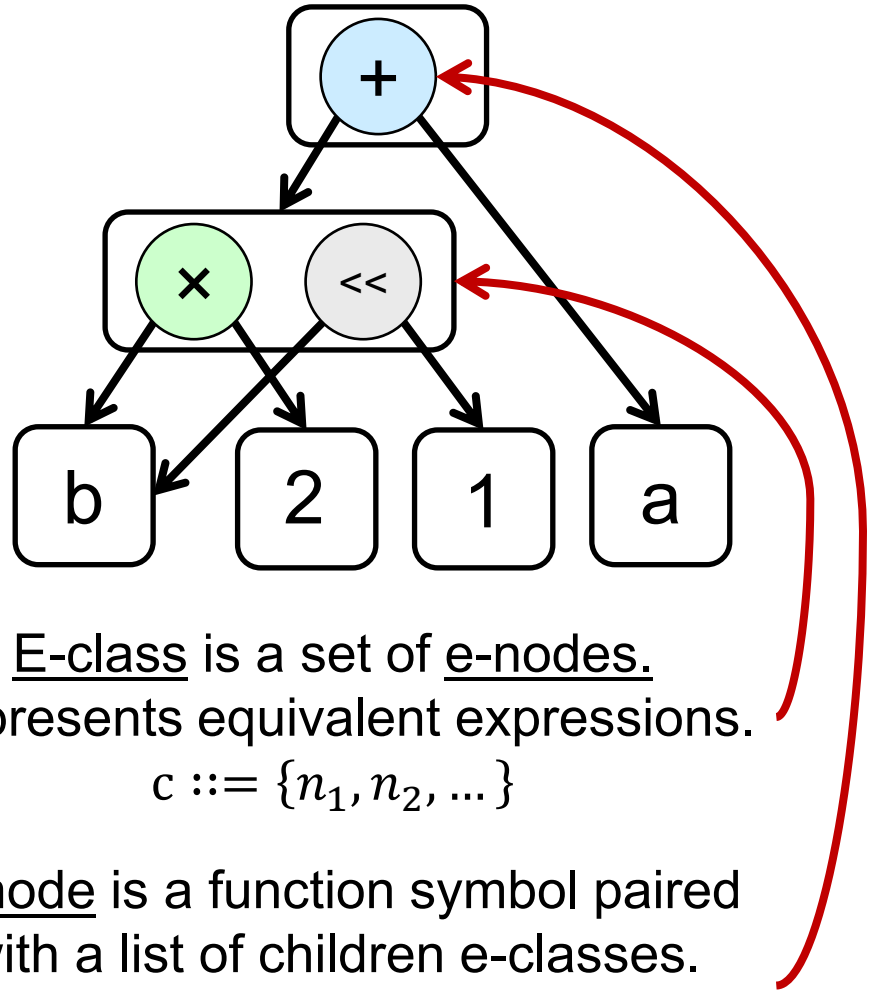
E-Graphs and Equality Saturation

Given

- An input program: a + b × 2
- A rewrite rule: t × 2 → t ≪ 1







Integer multiplication with equality saturation:

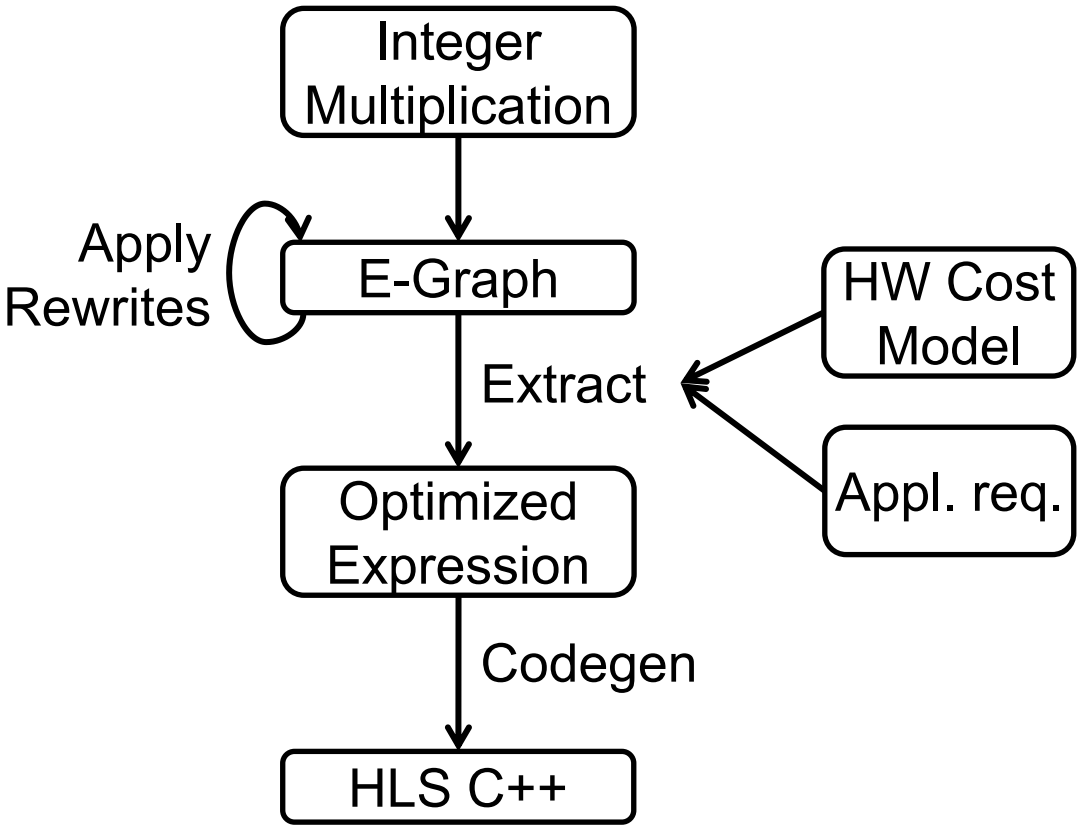
BW	E-Nodes	E-Classes	Expressions
64	252	194	1.08e6
128	1,171	878	1.37e24
256	5,546	4,078	3.55e96
512	26,769	19,426	1.58e386
1,024	130,936	94,218	6.31e1544
2,048	645,935	462,342	1.59e6179

E-class is a set of e-nodes.
Represents equivalent expressions.
$c ::= \{n_1, n_2, \ldots\}$

E-node is a function symbol paired with a list of children e-classes.
Represents expression(s).
$n ::= f(c_1, c_2, \ldots)$

IMpress: Large Integer Multiplication Expression Rewriting

- Optimizes large integer multiplication
  - Rewrite at arithmetic level & DSP block level
  - Rich design space through equality saturation
  - Scalable due to efficient data structure
- Extracts optimal expression(s) for multiple FPGA resource objectives
- Translates final expression to HLS C++
- Fits more instances of cryptographic applications on FPGA



Equality Saturation with IMpress

$$R = \{\langle l_i, r_i \rangle | i \in \{0,1,\dots,k-1\}\}$$

$$\langle mul_n, schoolbook(mul_n)\rangle \in R, n \in \{32, 64, \dots, 2048, \dots\}$$

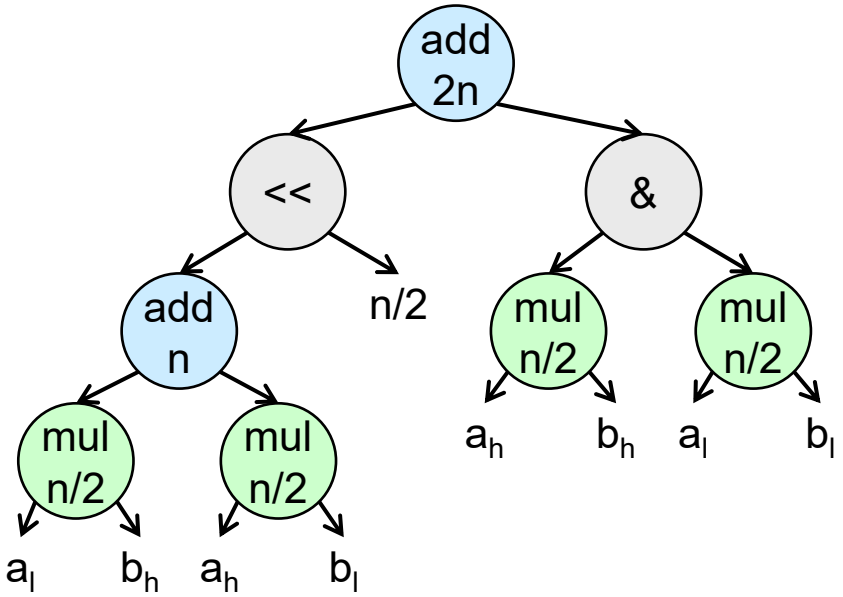
$$\langle mul_n, karatsuba(mul_n)\rangle \in R, n \in \{32, 64, \dots, 2048, \dots\}$$

$$\langle mul_{32}, tiling_m(mul_{32})\rangle \in R, m \in \{0, 1, 2, 3, 4\}$$

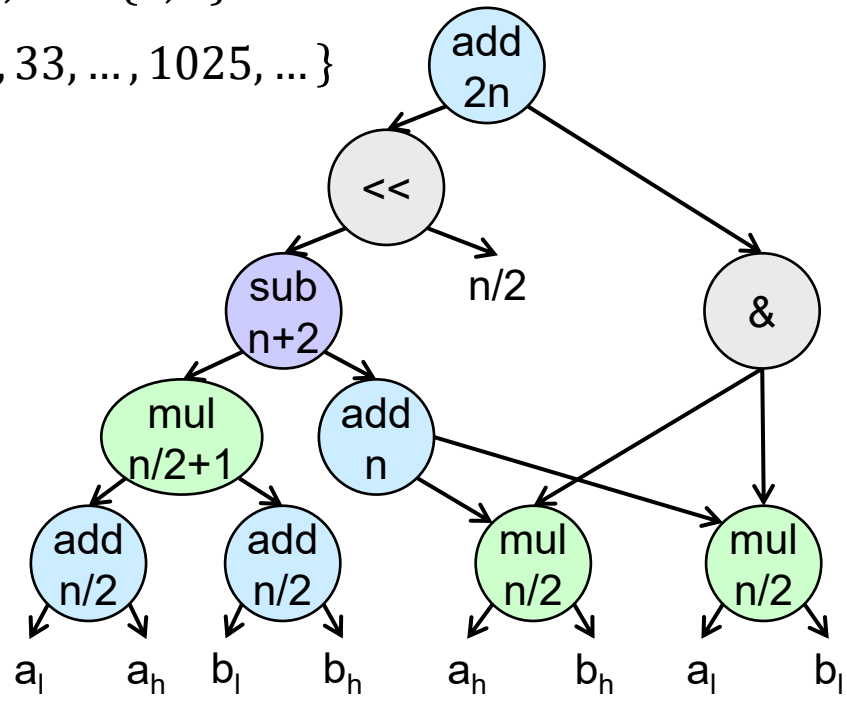
$$\langle mul_{16}, tiling_m(mul_{16})\rangle \in R, m \in \{0, 1\}$$

$$\langle mul_n, mul_{n-1}\rangle \in R, n \in \{17, 33, \dots, 1025, \dots\}$$

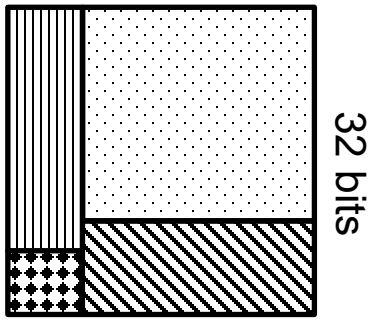
Input Bitwidth (n)	# Rewrite Rules (k)
32	7
64	10
128	13
256	16
512	19
1024	22
2048	25



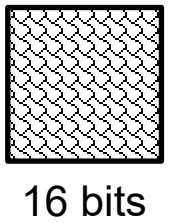
schoolbook(mul_n)



karatsuba(mul_n)



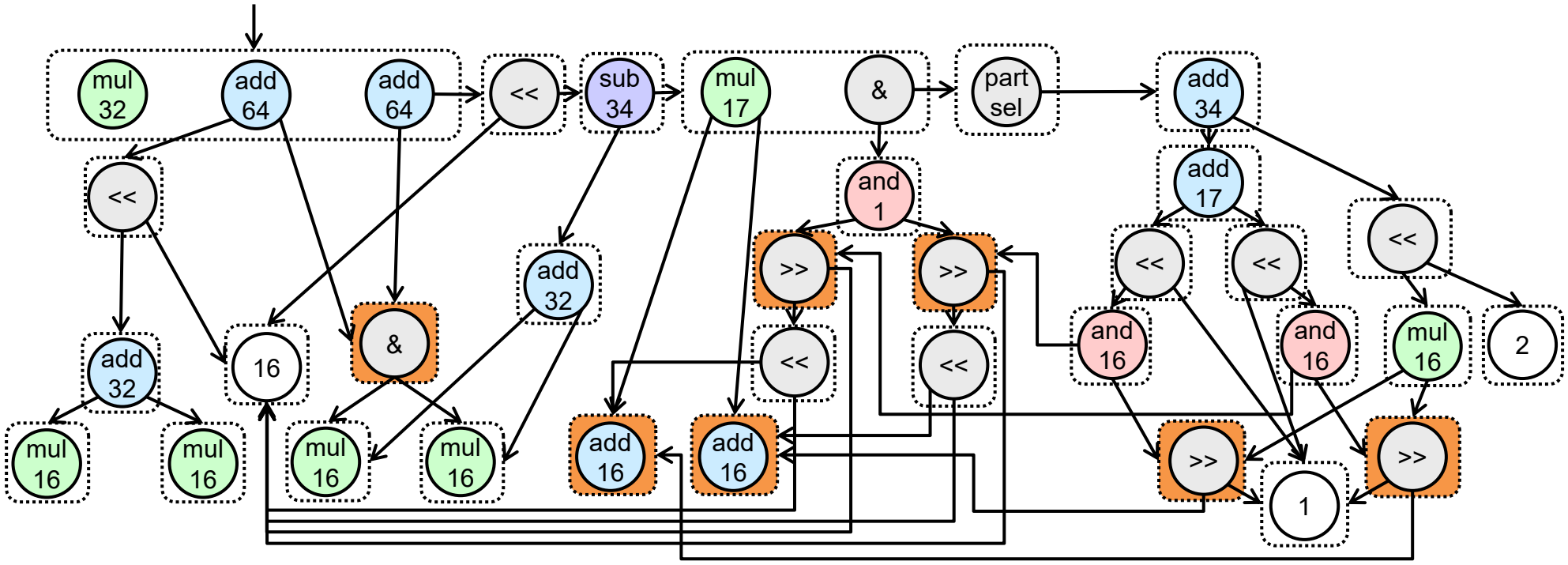
tiling(mul_{32})



tiling(mul_{16})

Equality Saturation with IMpress

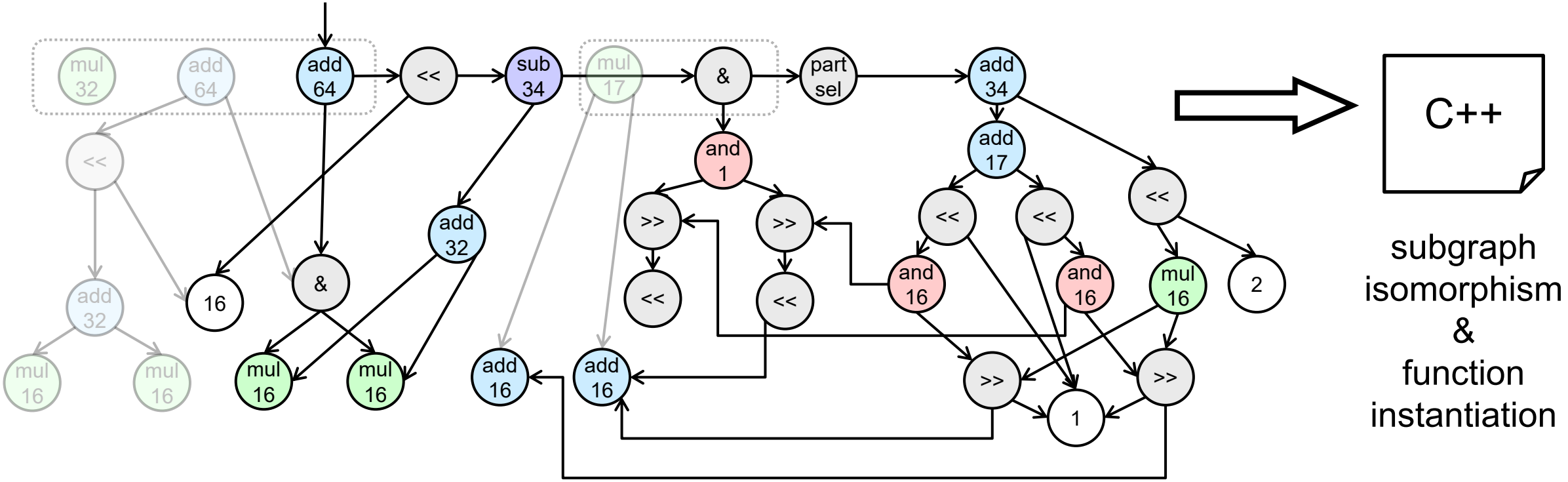
- $\langle mul_{32}, schoolbook(mul_{32}) \rangle$
- $\langle mul_{32}, karatsuba(mul_{32}) \rangle$
- $\langle mul_{17}, mul_{16} \rangle$
- Inter- and intra-rule subexpression sharing



M. Willsey *et al.* Egg: Fast and Extensible Equality Saturation. POPL, 2021. Tool available at github.com/egraphs-good/egg.

Extraction and Code Generation

- DSP minimization – egg (heuristic) & IMpress (exact)
- DSP-constrained LUT minimization – IMpress (exact)
- DSP and LUT co-minimization – IMpress (exact)



Evaluation

Scalability

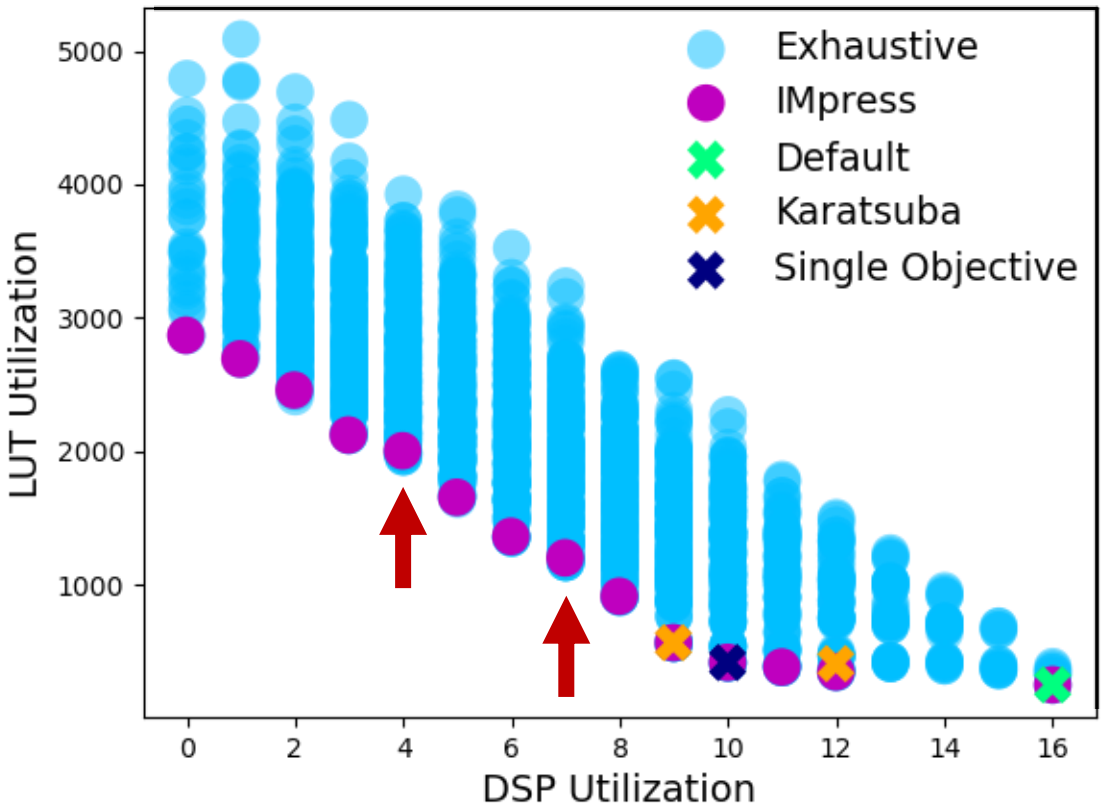
BW	E-Nodes	E-Classes	Expressions	Rewrite Rules	Saturation (s)	Extraction (s)	
						egg	IMpress
64	252	194	1.08e6	10	0.01	0.01	0.01
128	1,171	878	1.37e24	13	0.06	0.02	0.03
256	5,546	4,078	3.55e96	16	0.29	0.11	0.09
512	26,769	19,426	1.58e386	19	1.51	0.63	0.50
1,024	130,936	94,218	6.31e1544	22	8.60	2.72	4.07
2,048	645,935	462,342	1.59e6179	25	49.19	18.26	45.33

Extraction quality

Post-Place Cost	64	128	256	512	1,024	2,048
egg	1,823	6,698	25,359	85,117	278,265	1,101,986
IMpress	1,823	6,466	22,399	78,434	264,610	883,931

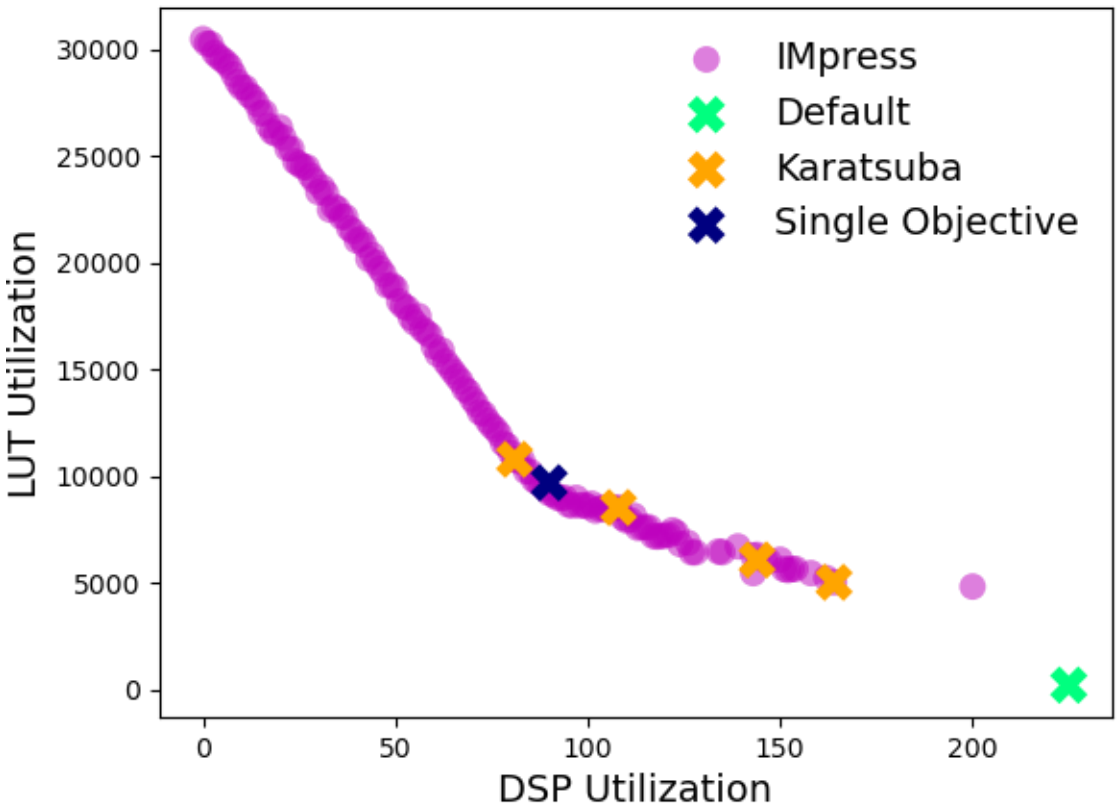
Evaluation

Pareto-optimality analysis



64-bit integer multiplication

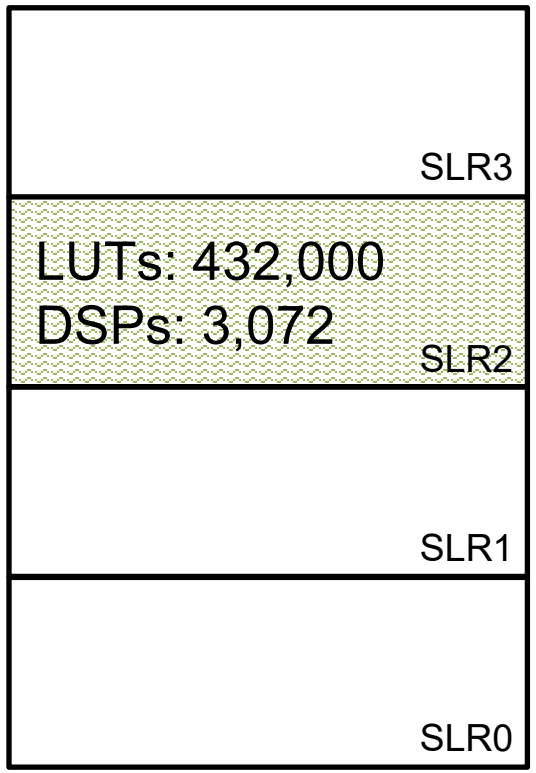
Flexibility in controlling DSP-LUT



256-bit integer multiplication

Evaluation

Placement-constrained million-bit multiplication for FHE



xcu250-figd2104-2L-e
(AMD Xilinx U250)

BW	DSP (DSP%)	LUT (LUT%)	Latency (cycles)	Frequency (MHz)
HLS Default	NA	NA	NA	NA
K(32, 64)	3,950 (32.15%)	339,999 (19.68%)	13,123,773	212
K(32, 32)	3,227 (26.26%)	408,301 (23.63%)	12,927,158	239
K(16, 32)	2,984 (24.28%)	439,775 (25.45%)	13,222,078	239
K(16, 16)	2,431 (19.78%)	511,916 (29.62%)	13,811,922	229
IMpress	3,071 (24.99%)	381,697 (22.09%)	12,746,927	225

Evaluation

RSA

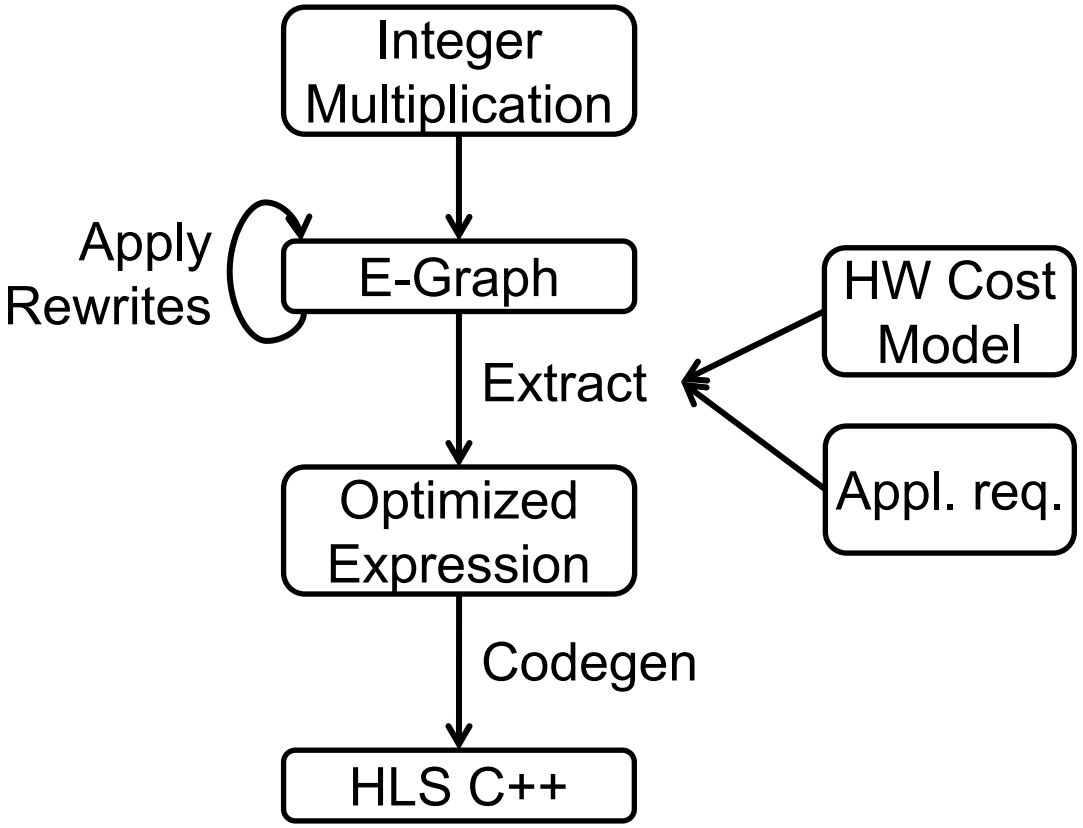
BW	DSP (DSP%)	LUT (LUT%)	Latency (cycles)	Frequency (MHz)	# Instances
Vitis-1	0 (0%)	17,055 (0.99%)	39,565	238	101
Vitis-2	4,530 (36.87%)	41,882 (2.42%)	3,638	18	2
2-level K	3,072 (25.00%)	156,339 (9.05%)	5,192	223	4
3-level K	2,352 (19.14%)	223,964 (12.96%)	4,772	241	5
4-level K	1,782 (14.50%)	291,507 (16.87%)	5,150	236	5
IMpress	2,030 (16.52%)	247,706 (14.33%)	5,024	231	6

AMD Xilinx Vitis Security Library, available at github.com/Xilinx/Vitis_Libraries/tree/master/security.

IMpress: Large Integer Multiplication Expression Rewriting for FPGA HLS

Ecenur Üstün, Ismail San, Jiaqi Yin, Cunxi Yu, Zhiru Zhang

- Optimize large integer multiplication at the HLS level
- Equality saturation based rewriting
  - Rich design space
- Extraction with multiple objectives
  - Flexibility for application requirements
  - Increase number of design instances



Thank you!