SPFD-Based Resynthesis for Dual-Output LUT Networks

Andrea Costamagna
EPFL
Lausanne, Switzerland
andrea.costamagna.ac@protonmail.com

Chang Meng
TU/e
Eindhoven, Netherlands
c.meng@tue.nl

Giovanni De Micheli
EPFL
Lausanne, Switzerland
giovanni.demicheli@epfl.ch

***Abstract*—Lookup-table (LUT) networks underpin modern FPGAs and ASIC flows, yet most optimization tools are restricted to single-output logic and fail to exploit the multiple-output capabilities widely supported by existing architectures. Prior work exploits multiple-output gates only during technology mapping, which makes the quality of results biased by the subject-graph structure; consequently, the mapped netlist can be suboptimal in the multiple-output design space. To address this limitation, this paper proposes a resynthesis engine for dual-output LUT networks that optimizes directly mapped netlists. Our approach combines (i) merging of single-output nodes via global matching and (ii) augmented Boolean resubstitution that upgrades nodes to dual-output to eliminate logic cones. On EPFL benchmarks, our method simultaneously reduces LUT count and critical-path depth versus the state-of-the-art, achieving a $\mathbf{2.7}\%$ geomean area improvement and $\mathbf{3.6}\%$ geomean depth reduction. Applied to the best-known EPFL mapped results, refined over a decade of community optimization, our resynthesis yields an $\mathbf{11}\%$ geomean area reduction with better or equal delay.**

***Index Terms*—FPGA, Dual-Output LUT, Resynthesis.**

I. Introduction

DEMANDS for higher computing performance continue to grow. A practical way to meet these requirements is to increase logic density, packing more functionality per unit area to reduce area and power. Modern FPGA architectures [1]–[3] support dual-output *lookup tables (LUTs)*, yet mainstream synthesis methods [4]–[8] remain largely focused on single-output logic and underutilize this capability.

This paper studies the optimization of combinational circuits represented as *dual-output LUT networks*, where each node is a single-output k-input LUT (k-*LUT*) or a dual-output $(k-1)$-input LUT. This model reflects modern FPGA architectures [1], [2], in which a k-LUT can be fractured into two $(k-1)$-LUTs with shared inputs (see Fig. 1). Previous research in logic synthesis focused on extending LUTs mapping algorithms to include dual-output LUTs [9], [10].

The problem with technology mapping is that its quality of results is sensitive to the structural choices made in the subject graph, and even optimal mappings can lead to suboptimal netlists in the multiple-output design space [11]. Conversely, resynthesis directly rewrites the mapped netlist, reducing this structural bias. Furthermore, existing post-mapping LUT optimizers operate exclusively on single-output nodes and cannot upgrade them to dual-output, leaving an entire dimension of the optimization space unaddressed.

To address this research gap, this paper presents a resynthesis engine that optimizes LUT networks through two strategies:

- **Merging:** We formulate global selection of single-output node pairs to be packed into a dual-output node as a weighted matching problem on a compatibility graph (capturing feasibility constraints such as shared-input and level constraints). An iterative matching procedure, combined with a level-based analyzer, applies merges while preventing the introduction of combinational cycles.
- **Resubstitution:** We extend classical Boolean resubstitution with rules tailored to dual-output LUTs. Beyond substituting suboptimal logic cones using existing signals or newly created intermediate signals, we synthesize new signals by upgrading existing nodes to dual-output whenever this enables eliminating logic cones.

Applying our engine to the EPFL benchmarks, our method simultaneously reduces LUT count and critical-path depth versus the state-of-the-art, achieving a $\mathbf{2.7}\%$ geomean area improvement and $\mathbf{3.6}\%$ geomean depth reduction. Applied to the best-known EPFL mapped results, refined over a decade of community optimization, our resynthesis yields an $\mathbf{11}\%$ geomean area reduction with no delay degradation. These results demonstrate that accounting for dual-output nodes during resynthesis unlocks gains that neither single-output rewriting nor dual-output technology mapping alone can achieve.

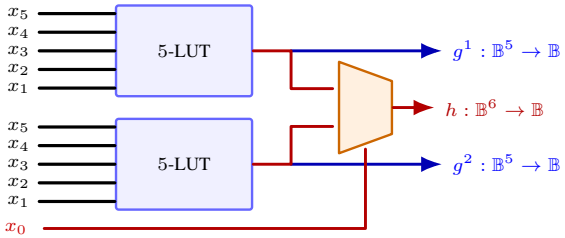

Fig. 1. $\mathbf{6^5_5}$**-LUT**: logic block in modern FPGA architectures [1], [2] that can either be used as a 6-*LUT* driven by $x_0 \dots x_5$, or fractured into two 5-*LUTs*.

The remainder of this paper is organized as follows. Section II introduces the background. Section III describes the proposed resynthesis engine. Section IV presents experimental results. Section V concludes the paper.

II. Background

A. Boolean Functions and Circuit Terminologies

Let $\mathbb{B} = \{0, 1\}$. An incompletely specified Boolean function $f : \mathbb{B}^n \to \{0, 1, *\}$ maps n binary inputs to either 0, 1, or $*$ (*don't care*). Let f_M denote the value of f at minterm $M \in \mathbb{B}^n$. The input space can be partitioned into the *on-set*, *off-set*, and *don't-care set*, corresponding to minterms for which $f_M = 1$, $f_M = 0$, and $f_M = *$, respectively.

We represent an incompletely specified Boolean function using two completely specified Boolean functions:

1) *On-set function* $\tau : \mathbb{B}^n \to \mathbb{B}$, $\tau_M = 1 \Leftrightarrow f_M = 1$;
2) *Care-set function* $\mu : \mathbb{B}^n \to \mathbb{B}$, $\mu_M = 0 \Leftrightarrow f_M = *$.

Combinational circuits are modeled as *directed acyclic graphs (DAGs)* whose nodes are classified as *primary inputs (PIs)*, *primary outputs (POs)*, and *internal nodes*. Let $\mathbb{B}^n$ denote the Boolean space induced by the n PIs. Each node u implements a global Boolean function $x_u : \mathbb{B}^n \to \mathbb{B}$. We use u and x_u interchangeably when the context is clear.

B. LUT Networks and Configurable Logic Blocks

LUT networks are combinational circuits whose internal nodes are LUTs. They serve as an abstraction of FPGA logic and as an intermediate representation for ASIC optimization.

This work targets FPGAs, whose basic logic unit is the *Configurable Logic Block* (CLB). Hutton et al. [12] introduced CLBs supporting dual-output functions, a feature widely adopted in modern FPGAs [1], [2]. As shown in Fig. 1, such a CLB can implement either a single-output 6-LUT realizing $h : \mathbb{B}^6 \to \mathbb{B}$ or two 5-LUTs realizing $g = (g^1, g^2) : \mathbb{B}^5 \to \mathbb{B}^2$.

A *k-LUT network* is a network in which every node is a single-output k-input LUT. A *dual-output k-LUT network* is a network in which each node can be either a single-output k-input LUT or a dual-output $(k-1)$-input LUT (k^{k-1}_{k-1}-LUT).

The fracturable nature of CLBs justifies assigning the same area cost to a dual-output LUT and a single LUT. For timing, we use a unit-delay model, as commonly adopted in FPGA synthesis research [8], [10]. Although fracturable LUTs exhibit pin-dependent delays, their impact is secondary to unmodeled routing delays; therefore, finer-grained LUT delay models would provide limited accuracy without corresponding interconnect modeling. Our framework can nevertheless be extended to incorporate such information.

C. Information Measures in Boolean Spaces

The information associated with a Boolean function consists of the pairs of input minterms that it distinguishes, i.e., that are mapped to different logic values. This information is compactly represented by the *set of pairs of functions to be distinguished* (SPFD). Following [13], we use the notation $\Upsilon_{x_u} = \{(\tau, \mu)\}$ to encode the SPFD associated with global function x_u, with on-set and care-set representation (τ, μ).

Let Υ_{x_u} and Υ_{x_i} denote the SPFDs of x_u and x_i, respectively, and let x'_i be the complement of x_i. The *SPFD difference* removes from Υ_{x_u} the information shared with Υ_{x_i}:

$$\Upsilon_{x_u} \setminus \Upsilon_{x_i} = \{(\tau, \mu x_i), (\tau, \mu x'_i)\}.$$

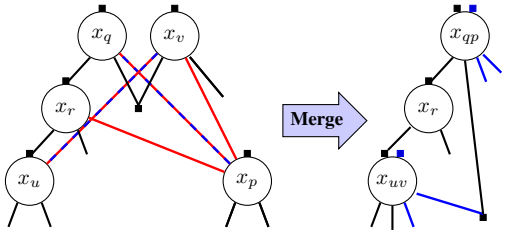

Fig. 2. **Merging**: The matching heuristic selects from the candidate merges (—) on a 4^3_3-*LUT* network, a subset of compatible ones (—).

The operation extends recursively:

$$\Upsilon_{x_u} \setminus \bigcup_{d \in \{i,j\}} \Upsilon_{x_d} = \{(\tau, \mu x_i x_j), (\tau, \mu x'_i x_j), (\tau, \mu x_i x'_j), (\tau, \mu x'_i x'_j)\}.$$

A set $\mathcal{C} = (u, \mathcal{L})$ is a *dependency cut* if the information of u is fully represented by the signals in $\mathcal{L}$

$$\Upsilon_{x_u} \setminus \bigcup_{d \in \mathcal{L}} \Upsilon_{x_d} = \varnothing.$$

In other terms, $\mathcal{L}$ preserves all distinctions made by x_u.

Window-based resubstitution simulates a local sub-network (*window*) to compute node signatures, which approximate the global functionality of the nodes in the window. Dependency analysis is then used to replace a target node with a lower-area implementation. Costamagna *et al.* [8] proposed an SPFD-based resubstitution algorithm for LUT networks. This work extends resynthesis to dual-output LUT networks.

III. Resynthesis for Dual-Output LUT Networks

This section presents our framework for resynthesizing dual-output LUT networks. Section III-A gives an overview of the method. Section III-B discusses our generalization of SPFDs to multiple-output functions. Section III-C introduces the *tiers evaluator*, our engine for efficient timing updates. The two core optimization strategies are then detailed: merging (Section III-D) and augmented resubstitution (Section III-E).

A. Overview

Given a mapped k-LUT network, our engine assigns per-output arrival and required times to each signal and organizes nodes into integer *levels*, maintained by an incremental *levels evaluator*. Two complementary passes are then executed:

1) **Merging** (Fig. 2): single-output nodes are fused into dual-output nodes in two phases: first *adjacent* pairs (connected by a fanin edge), then *disjoint* pairs (no direct connection), verified acyclic via a reachability check.
2) **Augmented Resubstitution** (Fig. 3): nodes are visited in reverse topological order. Around each node a local sub-network (*window*) is extracted, and four don't-care-aware strategies are tried in sequence, stopping at the first success.

After every transformation, the levels evaluator is updated incrementally to maintain consistent timing information.

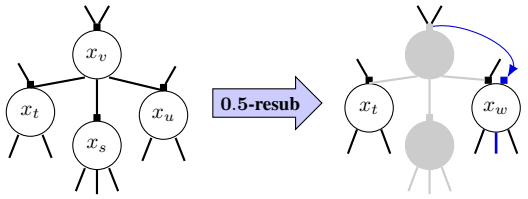

Fig. 3. **0.5-resub**: The single-output nodes x_v and x_s (○) are removed (●) by fracturing node x_u as node x_w (○), so that $x_w^1 = x_u$, $x_w^2 = x_v$.

B. Multiple-Output Boolean Information

Let x_u be a node in a k_{k-1}^{k-1}-LUT network with m outputs; we write x_u^p for its p-th output signal. Once the window is collected, each signal is labelled with functional information for resynthesis. Leaf signals are assigned to projections in $\mathbb{B}^{|L|}$, all window signals are simulated, and their truth tables are encoded as signatures, defining the Boolean space in which the SPFD procedures search for alternative implementations.

In a multiple output node x_u, each output signal x_u^p carries a simulation signature $x_u^p : \mathbb{B}^n \to \mathbb{B}$. A cut $\mathcal{C} = (u, \mathcal{L})$ with leaf set $\mathcal{L} = \{x_j\}_{j=1}^{L}$ is a *dependency cut* of u if and only if:

$$\Upsilon_{x_u^p} - \bigcup_{x_j \in \mathcal{L}} \Upsilon_{x_j} = \varnothing, \qquad \text{for } p = 1, \ldots, m.$$

Since each LUT output is an observable signal, preserving the SPFD of every output preserves the multiple-output node functionality. This extends the SPFD-based dependency analysis of [13] to multiple-output nodes and enables our engine.

C. Incremental Timing Update

A full timing recomputation after every network edit would be prohibitively expensive, yet incomplete propagation risks introducing combinational cycles or timing violations. We address this with an incremental evaluator that confines work to the affected cone via two directed sweeps.

Forward pass. Starting from the modified nodes, a BFS propagates arrival-time changes to their fanouts in level-sorted waves, recomputing levels and arrival times. A node is forwarded to the next wave only if at least one of its arrivals changed, pruning unaffected paths immediately.

Backward pass. The modified nodes, their direct fanins, and all arrival-changed nodes are marked and define an initial tier window $[low, high]$. The sweep processes all marked nodes tier by tier from $high$ down to low, recomputing required times. Whenever a required time changes, the node's fanins are marked; if any lie below low, the window extends downward and the sweep continues into the newly added tiers.

This reduces the $O(|V|^2)$ timing update overhead to $O(|V|)$.

D. Node Merging

A merge is profitable whenever two nodes u and v satisfy $|\text{fanins}(u) \cup \text{fanins}(v)| \leq k{-}1$ (with k=6, at most 5 inputs). Two configurations are considered.

Adjacent. If v is a direct fanin of u, the pair is accepted only if the timing constraints are not violated and no other fanin of u can reach v, preventing cycles.

Disjoint. If u and v are topologically independent, candidate search is restricted to levels within a timing window; nodes in the TFI or TFO of u are marked and skipped, and the remaining candidates must satisfy the timing constraints and pass a reachability check confirming topological independence.

Surviving pairs are registered as compatible in a compatibility graph, on which we greedily solve the binary matching problem: nodes are visited in ascending order of match-degree, and for each node its candidates are tried in descending order of fanin overlap (preferring tighter merges). For each accepted pair, the truth tables over the merged support are computed by symbolic composition of the individual LUT functions; a dual-output node is then created and both originals are substituted.

E. Augmented Resubstitution

Resubstitution replaces a target node with an existing or newly synthesized signal whenever the area gain outweighs the cost. Dual-output nodes introduce a new degree of freedom: a single-output node can be augmented into a dual-output node at zero area cost, since no extra LUT is added. We call such transformations *fractional-cost*, as they consume a node's capacity for future augmentation.

For each node u, a window is constructed and simulated, and four strategies are attempted in order of increasing cost.

- **0-resub.** Each output of u is matched against window divisors. A divisor is accepted if it reproduces the target function exactly. Fully matched nodes are deleted and their wires re-routed; partially matched nodes are decomposed into independent single-output cells.
- **0.5-resub.** For single-output u, a compatible node v in the timing window with $|\text{fanins}(v)| < k{-}1$ is augmented by greedily adding divisors until u's function is covered in the SPFD sense. A dual-output node w with $x_w^0 \equiv v$ and $x_w^1 \equiv u$ replaces both at zero net area cost.
- **1-resub.** When the MFFC of u contains at least two nodes, a replacement is synthesized by Boolean interpolation over a dependency cut. Both output functions of dual-output nodes are interpolated simultaneously.
- **1.5-resub.** An SPFD-guided greedy loop selects divisors one at a time, maximizing information coverage; divisors that benefit from augmentation are promoted to dual-output cells on the fly. Once coverage is complete, a replacement node is interpolated from its fanin set.

IV. Experimental Results

In this section, we evaluate our dual-output LUT resynthesis engine. All experiments were conducted on a Linux machine equipped with an `i7-1365U` CPU.

A. Setup and Experimental Assumptions

We evaluate our engine on the EPFL combinational benchmark suite [14], targeting *dual-output* LUT primitives: a single logic block with k inputs that can be used as a k-LUT or as two $(k-1)$-LUTs with shared support, packed into a single block at no additional cell cost. This reflects the physical capability of modern FPGAs, such as Xilinx UltraScale [2]. Node count

TABLE I
DUAL-OUTPUT OPTIMIZATION FROM A STANDARD MAPPING FLOW. GMEAN VALUES <1 FAVOR OUR METHOD.

	$\|LUT\|$				D		t (s)		
Bench	ABC	SoTA	Mrg	Rsb	SoTA	Rsb	SoTA	Mrg	Rsb
adder	257	179	191	191	51	51	0.29	0.02	0.15
bar	512	448	448	448	4	4	0.39	0.06	0.59
div	9931	7279	7403	7377	866	856	56.31	1.07	63.57
log2	8077	6476	6372	6350	76	71	16.26	4.02	44.86
max	793	620	610	610	39	39	0.54	0.14	1.66
multiplier	5922	4324	4379	4369	54	53	9.98	4.79	39.91
sin	1477	1203	1182	1177	40	36	1.33	0.20	2.69
sqrt	4645	3908	3665	3634	1118	1017	9.71	0.53	14.77
square	3949	2692	2959	2950	50	50	6.34	3.78	17.48
arbiter	2722	2468	2470	2470	18	18	2.98	0.59	4.47
cavlc	118	105	104	104	4	4	0.10	0.01	8.48
ctrl	29	18	17	17	2	2	0.06	0.00	0.05
dec	287	144	152	152	2	2	0.09	0.06	0.15
i2c	316	242	247	245	4	4	0.21	0.03	0.52
int2float	47	41	42	42	3	3	0.07	0.00	0.98
mem_ctrl	11747	9475	9067	9009	30	25	59.88	23.80	105.46
priority	178	159	161	149	26	26	0.14	0.01	0.24
router	50	72	46	46	6	6	0.07	0.00	0.15
voter	1736	1380	1345	1342	16	13	0.72	0.40	3.25
Sum	52787	41233	40860	**40682**	2409	**2280**	165.5	**39.5**	309.4
Gmean	—	1.000	**0.979**	**0.973**	1.000	**0.964**	1.000	**0.123**	2.522

TABLE II
AREA OPTIMIZATION ON BEST-KNOWN-SIZE EPFL BENCHMARKS. INPUT NETWORKS ARE THE BEST PRIOR AREA RESULTS (LOADED FROM BLIF).

	Area			**Levels**		t (s)	
Benchmark	BEST	Mrg	Rsb	BEST	Rsb	Mrg	Rsb
adder	129	127	127	126	126	0.00	0.04
arbiter	261	211	211	93	93	0.01	0.91
bar	512	448	448	4	4	0.04	0.44
cavlc	49	49	49	7	7	0.00	2.34
ctrl	26	16	16	2	2	0.00	0.01
dec	264	260	260	2	2	0.04	0.57
div	3213	3104	3091	1100	1100	0.24	9.70
hyp	36505	34546	34543	4571	4571	306.02	1051.56
i2c	190	166	166	7	7	0.02	0.26
int2float	24	21	20	6	6	0.00	0.33
log2	6012	5584	5583	257	257	3.86	37.33
max	511	449	449	134	134	0.05	2.16
mem_ctrl	2925	1845	1744	15	14	1.00	8.00
multiplier	4442	4139	4114	209	209	3.33	32.29
priority	100	95	94	31	31	0.00	0.56
router	20	20	19	9	9	0.00	0.09
sin	1020	974	973	107	107	0.16	3.27
sqrt	2966	2958	2957	1185	1185	0.19	10.69
square	2937	2780	2777	199	199	2.71	12.43
voter	1165	1151	1151	27	27	0.28	2.69
Sum	63271	58943	58792	8091	8090	317.94	1175.66
Gmean	1.000	0.899	0.891	1.000	0.997	—	—

therefore directly measures FPGA resource utilization. In both experiments, **Mrg** denotes the merge-only pass and **Rsb** merging followed by resubstitution.

B. Comparison with Dual-Output Mapping

This experiment shows that resynthesis improves both area and delay over the state-of-the-art [9]. Each circuit is first optimized at the AIG-level (`resyn; resyn2`) and mapped to 6-input LUTs using ABC's `if` mapper, then passed to our resynthesis engine. This matches the preprocessing flow of the *state-of-the-art* (**SoTA**) [10] and the ABC mapper augmented by its authors, ensuring a fair comparison.

As shown in Table I, **Rsb** reduces LUT count by 23.0% over the ABC baseline, compared with 21.9% for **SoTA**, yielding a **2.7%** geomean area improvement over **SoTA**. It also reduces depth by **3.6%** geomean, achieving a Pareto improvement. These gains result from the greater flexibility of resynthesis: unlike dual-output mapping, which optimizes the covering of a fixed subject graph, resynthesis can restructure logic cones under timing constraints and explore a broader design space.

Mrg captures most of the area gains of **SoTA** while reducing runtime by $8\times$. The complete flow achieves additional area improvements at higher runtime, which remains within the same order of magnitude as **SoTA**. Unlike **SoTA**, our approach performs timing updates during optimization; this added overhead is the main runtime cost, but is essential for achieving simultaneous area and delay improvements.

C. Improving on a Decade of Community Effort

A key advantage of resynthesis over mapping is that it operates directly on a mapped network, without first converting it to a subject graph. This means that when the input is already highly optimized, the choices encoded in the mapped netlist are preserved and exploited rather than discarded. To test this, we use the EPFL best-known single-output 6-LUT networks as the optimization baseline, loaded directly without remapping. These represent the result of over a decade of optimization research within the community and constitute the strongest possible starting point for evaluating residual area savings.

Table II reports results of applying our flow. Despite starting from an unconstrained, area-optimized single-output state-of-the-art, our method achieves a geomean area reduction of $\mathbf{10.9}\%$ (**Rsb**), with improvements on every benchmark where dual-output packing is structurally applicable. The largest gains appear on circuits with high logical density and reconvergence: *mem_ctrl* (40.4%), *ctrl* (38.5%), *arbiter* (19.2%), *int2float* (16.7%), and *i2c* (12.6%). These are savings that no single-output optimization pass can recover, regardless of runtime, because they require a fundamentally different cell primitive. Depth is preserved or reduced on all benchmarks.

V. CONCLUSIONS

To the best of our knowledge, this paper presents the first dual-output LUT resynthesis engine, combining node merging and Boolean resubstitution to transform single-output nodes into dual-output implementations. Compared with state-of-the-art dual-output mapping [10], it achieves Pareto improvements of $\mathbf{2.7}\%$ geomean area reduction and $\mathbf{3.6}\%$ geomean depth reduction. Applied to single-output networks refined by the community over a decade [14], it achieves $\mathbf{10.9}\%$ geomean area reduction, with per-benchmark gains of up to $\mathbf{40}\%$ enabled by dual-output optimization. These results establish resynthesis as a natural complement to dual-output technology mapping, motivating extensions to other cost functions and cell-based synthesis.

REFERENCES

[1] Intel, "Intel® Arria® 10 device overview." Intel, Inc., 2025.

[2] Xilinx, "UltraScale architecture libraries guide." AMD, Inc., 2025.

[3] J. Chromczak, M. Wheeler, C. Chiasson, D. How, M. Langhammer, T. Vanderhoek, G. Zgheib, and I. Ganusov, "Architectural enhancements in intel® Agilex™ FPGAs," in *International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2020, pp. 140–149.

[4] R. K. Brayton, "The decomposition and factorization of boolean expressions," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 1982, pp. 49–54.

[5] H. Sawada *et al.*, "Logic synthesis for look-up table based FPGAs using functional decomposition and Boolean resubstitution," *IEICE Transactions on Information and Systems*, vol. 80, no. 10, pp. 1017–1023, 1997.

[6] A. Mishchenko, R. Brayton, S. Jang, and V. Kravets, "Delay optimization using SOP balancing," in *International Conference on Computer-Aided Design (ICCAD)*, 2011, pp. 375–382.

[7] S.-Y. Lee and G. D. Micheli, "Heuristic logic resynthesis algorithms at the core of peephole optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 42, no. 11, pp. 3958–3971, 2023.

[8] A. Costamagna, A. T. Calvino, A. Mishchenko, and G. De Micheli, "Area-oriented resubstitution for networks of look-up tables," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2025.

[9] F. Wang, L. Zhu, J. Zhang, L. Li, Y. Zhang, and G. Luo, "Dual-output LUT merging during FPGA technology mapping," in *International Conference on Computer-Aided Design (ICCAD)*, 2020, pp. 1–9.

[10] S. Lu, L. Shang, Q. Qu, S. Jung, Q. Liang, and C. Pan, "An efficient multi-output lut mapping technique for field-programmable gate arrays," *Electronics*, vol. 14, no. 9, p. 1782, 2025.

[11] S. Chatterjee, A. Mishchenko, R. K. Brayton, X. Wang, and T. Kam, "Reducing structural bias in technology mapping," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 12, pp. 2894–2903, 2006.

[12] M. Hutton, J. Schleicher, D. Lewis, B. Pedersen, R. Yuan, S. Kaptanoglu, G. Baeckler, B. Ratchev, K. Padalia, M. Bourgeault *et al.*, "Improving fpga performance and area using an adaptive logic module," in *International Conference on Field Programmable Logic and Applications*. Springer, 2004, pp. 135–144.

[13] A. Costamagna, C. Meng, and G. De Micheli, "Spfd-based delay resynthesis," in *2025 21st International Conference on Synthesis, Modeling, Analysis and Simulation Methods, and Applications to Circuits Design (SMACD)*. IEEE, 2025, pp. 1–4.

[14] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The epfl combinational benchmark suite," *Hypotenuse*, vol. 256, no. 128, p. 214335, 2015.