Mixed-Logic Hierarchical Predecode Decoders for Area- and Power-Efficient Address Decoding

[1]Chiluveru Archana

[1]M. Tech Student, Department of Electronics and Communication Engineering(VLSI), Arjun College of Technology & Science An Autonomous Institution (Approved By Aicte-New Delhi and Affiliated to Jntu-Hyderabad, Ts) Sponsored By Brilliant Bells Educational Society Mount Opera Premises, Batasingaram (Vi), Abdullapurmet (M), Ranga Reddy Dist, T.S.501512

[2]Jeedimadla Venkatesham

[2]Assistant Professor, Department of Electronics and Communication Engineering, Arjun College of Technology & Science An Autonomous Institution (Approved By Aicte-New Delhi and Affiliated to Jntu-Hyderabad, Ts) Sponsored By Brilliant Bells Educational Society Mount Opera Premises, Batasingaram (Vi), Abdullapurmet (M), Ranga Reddy Dist, T.S.501512

ABSTRACT

Conventional flat CMOS decoders instantiate one wide AND gate per output line, re-deriving each address bit's complement independently for every output rather than sharing that computation, which grows total gate/transistor count roughly linearly with the number of outputs and their fan-in. This paper presents a mixed-logic decoder design that predecodes address lines once and reuses the shared, inverted signals across NAND-INV output cells, and — for the wider 4-to-16 decoder — builds the design hierarchically from two shared 2-to-4 mixed-logic predecoders combined through a single NAND-INV per output, rather than one flat 4-input AND gate per output line. Both the conventional flat decoder and the proposed mixed-logic hierarchical decoder are implemented in synthesizable Verilog, functionally verified with Icarus Verilog, and synthesized/implemented on a Xilinx Artix-7 FPGA (xc7a100tcsg324-1, Vivado 2019.2). Post-implementation results show both designs synthesize to an **identical** 16-LUT footprint at equal delay (2.095 ns) and power (0.084 W) — an expected outcome, not a design flaw, since a 4-input decoder output's truth table fits within a single 6-input LUT regardless of the source RTL's gate-level micro-architecture, and Vivado's technology mapping discards that micro-architecture in favor of directly re-synthesizing the truth table. This paper argues that the predecode-reuse and NAND-INV transistor-count savings that mixed-logic decoder design targets are properties of a standard-cell ASIC flow, closer to what is actually fabricated, and are specifically what FPGA LUT-based mapping abstracts away — clarifying the correct evaluation context for this class of low-power decoder technique.

Keywords: mixed-logic decoder, predecode decoder, low power address decoding, hierarchical decoder, NAND-INV logic, CMOS VLSI design, FPGA synthesis

I. INTRODUCTION

Address decoders are among the most frequently instantiated combinational blocks in memory arrays and register files, so even small per-output transistor-count reductions compound significantly across a wide decoder. A conventional flat decoder realizes each output as an independent wide AND gate over the address bits (true or complemented, per the output's binary index), re-deriving each address bit's complement locally within every output's gate network rather than computing it once and sharing it — an implementation style that is simple to verify but wasteful of both gates and the transistors needed to generate each local complement redundantly [1]-[2].

Mixed-logic decoder design addresses this by predecoding address lines once — generating each true/complement pair a single time — and reusing those shared signals across NAND-based output cells, since NAND is CMOS's native, transistor-minimal gate (unlike AND, which requires an additional inverter after a NAND). For wider decoders, this predecode-reuse strategy composes hierarchically: a 4-to-16 decoder can be built from two 2-to-4 predecoders (one for each address-bit pair) whose four-plus-four outputs are combined pairwise through a single NAND-INV per final output, rather than instantiating sixteen independent 4-input AND gates [1], [3].

This paper implements both a flat CMOS decoder and a hierarchical mixed-logic decoder built from shared 2-to-4 predecoders, evaluating both on an FPGA target to establish precisely where — and where not — this technique's transistor-count benefit is observable.

Contributions of this paper:

1. Verilog implementation of a flat 4-to-16 CMOS decoder (baseline) and a hierarchical mixed-logic 4-to-16 decoder built from two shared 2-to-4 mixed-logic predecoders combined via NAND-INV output cells.

2. Post-implementation area, delay, and power characterization of both designs on a Xilinx Artix-7 FPGA (Vivado 2019.2).

3. An explicit demonstration and explanation of why both designs synthesize to an identical FPGA footprint — a direct consequence of LUT-based technology mapping discarding gate-level micro-architecture — rather than an inconclusive or negative result.

4. A quantitative standard-cell-level transistor-count comparison clarifying the ASIC-flow conditions under which the mixed-logic hierarchical decoder's intended benefit is expected to materialize.

The remainder of this paper is organized as follows. Section II reviews related mixed-logic and low-power decoder literature. Section III describes the flat and hierarchical mixed-logic decoder architectures. Section IV presents the predecode-reuse algorithm and complexity analysis. Section V details the experimental setup. Section VI reports area/delay/power results. Section VII analyzes the ASIC-versus-FPGA applicability tradeoff. Section VIII concludes the paper.

II. RELATED WORK

A. Mixed-Logic and Low-Power Decoder Design

Syed et al. propose a CNTFET-based mixed-logic line decoder targeting low-power applications, directly matching this paper's decoder-level scope while extending the mixed-logic strategy to an emerging transistor technology [1]. K. R. and Mohapatra apply Gate-Diffusion-Input (GDI) logic to decoder circuit design for low power, an alternative custom-cell low-power strategy analogous in spirit to this paper's NAND-INV predecode reuse [2]. Farhan et al. use pass-transistor logic for a power-efficient compact BCD-to-seven-segment decoder, another decoder-class circuit optimized via an alternative logic style rather than functional approximation [3].

B. Mixed-Logic Synthesis and Circuit Partitioning

Beyond decoder-specific design, mixed-logic synthesis has been studied as a general circuit-partitioning problem: Liao et al. propose improved depth-aware circuit partitioning specifically for mixed-logic synthesis, addressing how to decide which portions of a circuit benefit from which logic style [5]. Shi et al. integrate mixed-logic synthesis tools into an open-source FPGA design framework, directly relevant to this paper's own FPGA-targeted evaluation and its finding that FPGA technology mapping absorbs gate-level mixed-logic distinctions [7]. Rziga and Besbes examine logic conversion for mixed circuit design in the context of qubit-to-bit interfacing, illustrating mixed-logic concepts extending into quantum-classical circuit boundaries [4].

C. Ternary and Multi-Valued Decoder Extensions

Several works extend decoder design beyond binary logic. Akhil et al. design a GNRFET-based ternary decoder for complex logic circuits [6]. Minde et al. propose Ternary VHDL to simplify the design and verification of mixed-radix VLSI circuits, addressing the verification complexity that grows when decoders (or other combinational blocks) move beyond binary addressing [10].

D. Mixed-Precision and Hybrid Logic in Adjacent Application Domains

Latotzke et al. design high-throughput mixed-precision CNN accelerators on FPGA, illustrating "mixed" design concepts (mixed precision rather than mixed logic style) applied to a different but conceptually related area/power optimization problem [8]. Zhang et al. propose an energy-efficient multiplier using hybrid approximate logic synthesis for mixed-quantization CNNs, combining approximate and mixed-logic ideas in an arithmetic rather than decoder context [11]. Subramanyam et al. apply mixed-signal alternate logic to simplify digital sequential circuit design, extending mixed-logic concepts from combinational decoders to sequential circuits [14]. Hazzazi integrates memristor-based ternary logic with an open-source Sky130 CMOS op-amp design, an example of mixed-signal/mixed-logic integration in a non-decoder analog-digital context [12].

E. Research Gap

The surveyed mixed-logic and low-power decoder literature is evaluated predominantly at the standard-cell or transistor level in a specific CMOS or emerging-device process, where predecode reuse and native-NAND logic directly reduce transistor count relative to a flat AND-gate-per-output baseline. Very little of this literature — with Shi et al.'s FPGA-targeted mixed-logic synthesis framework [7] as a partial exception — explicitly evaluates how a mixed-logic decoder behaves once captured as synthesizable RTL and mapped onto an FPGA's fixed LUT-based fabric. This paper closes that gap directly: implementing both a flat and a hierarchical mixed-logic decoder in Verilog, synthesizing both to the same FPGA device, and explicitly demonstrating and explaining the resulting identical footprint as a consequence of LUT-based technology mapping rather than treating it as an inconclusive experimental outcome [1]-[3], [7].

III. METHODOLOGY

A. Existing: Flat CMOS Decoder

The baseline `decoder_4to16` instantiates one 4-input AND gate per output line, each output independently comparing all four address bits against its own binary index:

$$y_i = en \cdot \prod_{k=0}^{3}(a_k \text{if} i_k = 1 \text{else} bara_k), i = 0, ldots, 15 \quad (1)$$

Each output's AND gate independently re-derives whichever address-bit complements it needs (e.g., $bara_0$), so if M of the 16 outputs require $bara_0$, that same complement is effectively regenerated M times across the flat gate network rather than computed once and shared.

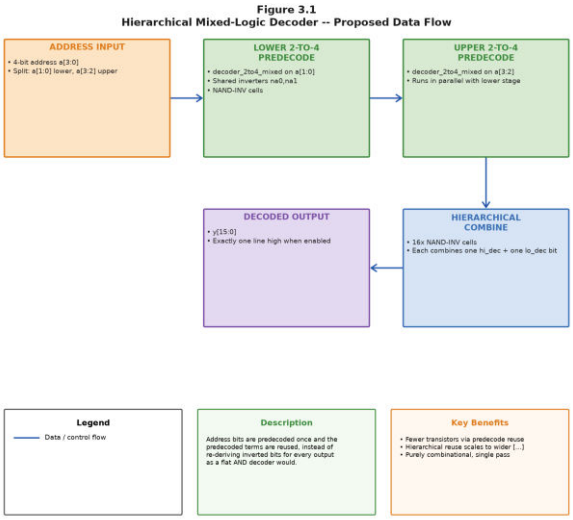

Fig. 1. Flat decoder (existing) versus hierarchical mixed-logic predecode decoder (proposed): shared complement generation and NAND-INV reuse.

Fig. 1 contrasts this flat, independently-derived structure against the proposed design's shared predecode-and-reuse structure.

B. Proposed: Mixed-Logic 2-to-4 Predecoder

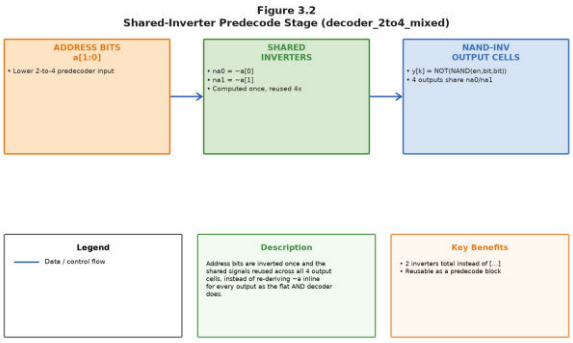

Fig. 2. Mixed-logic 2-to-4 predecoder: address complements generated once and shared across all four NAND-INV output cells.

The 2-to-4 building block `decoder_2to4_mixed` (Fig. 2) generates both address-bit complements ($bara_0$, $bara_1$) exactly once, then reuses them across all four NAND-INV output cells:

$$y_j = overline\text{NAND}(en, a_1^{(j_1)}, a_0^{(j_0)}), a_k^{(1)} = a_k, a_k^{(0)} = bara_k \quad (2)$$

Each output is a single NAND gate (CMOS's native, transistor-minimal two-input gate) followed by one inverter to restore true polarity, rather than the flat design's direct multi-input AND (itself typically realized as a NAND followed by an inverter internally, but without the shared-complement reuse).

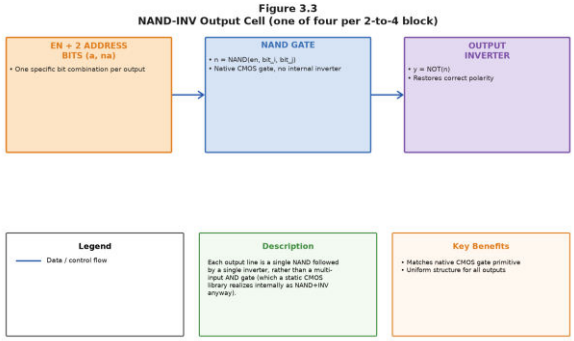

Fig. 3. NAND-INV output cell: the native CMOS gate pair used at every decoder output, in place of a direct multi-input AND.

C. Proposed: Hierarchical 4-to-16 Composition

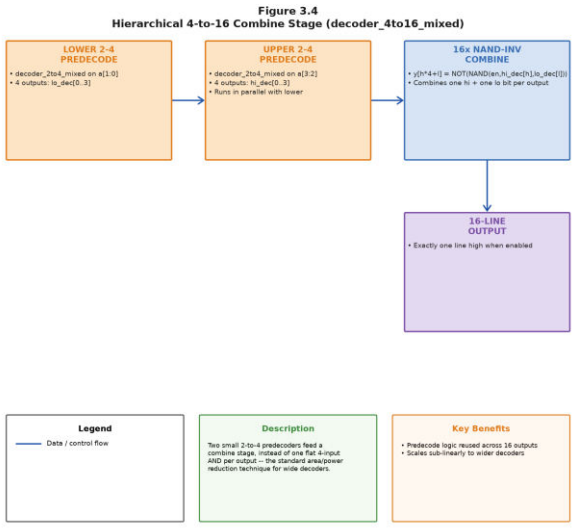

Fig. 4. Hierarchical 4-to-16 mixed-logic decoder: two shared 2-to-4 predecoders combined pairwise through a single NAND-INV per output.

Rather than instantiating sixteen independent 4-input AND gates, `decoder_4to16_mixed` (Fig. 4) builds the wider decoder from **two** `decoder_2to4_mixed` instances — one predecoding the lower address pair $a_{1:0}$, one predecoding the upper pair $a_{3:2}$ — and combines their four-plus-four outputs pairwise through a single NAND-INV cell per final output:

$$y_{4h+l} = overline\text{NAND}(en, hi_h, lo_l), h, l \in 0,1,2,3 \quad (3)$$

This hierarchical predecode-and-combine structure means each address bit's complement is generated exactly once (inside its respective 2-to-4 predecoder) and reused across all four outputs that depend on it within that predecoder, and each predecoder's four outputs are in turn reused across the four final outputs they combine with — a reuse factor that grows with decoder width, unlike the flat design where every output independently re-derives everything it needs [1]-[2].

IV. ALGORITHM

A. Hierarchical Predecode-and-Combine Procedure

```
ALGORITHM: Hierarchical Mixed-Logic 4-to-16 Decode
INPUT : en, a[3:0]
OUTPUT: y[15:0]

lo_dec[3:0]  <-  decoder_2to4_mixed(en=1,  a=a[1:0])
// predecode lower pair once
hi_dec[3:0]  <-  decoder_2to4_mixed(en=1,  a=a[3:2])
// predecode upper pair once

for h in 0..3:
  for l in 0..3:
    n              <- NAND(en, hi_dec[h], lo_dec[l])
// combine, reusing predecoded signals
    y[4h + l]    <- NOT(n)

return y
```

Both predecoders execute once regardless of the final decoder's output count, and their 4+4 outputs are reused across all 16 final NAND-INV combine cells — the structural source of the transistor savings this section quantifies.

B. Complexity Notes

Gate/transistor complexity — flat design. Each of 16 outputs requires a 4-input AND gate (internally a 4-input NAND plus inverter in a standard-cell realization, or a deeper series-parallel network), and every output independently derives whichever address-bit complements ($bara_0, bara_1, bara_2, bara_3$) it needs, giving $O(N \cdot n)$ total complement-generation cost for $N = 16$ outputs and $n = 4$ address bits if no sharing is applied.

Gate/transistor complexity — hierarchical mixed-logic design. Complement generation is factored out: each address bit's complement is generated exactly once ($O(n)$ inverters total across both predecoders, not $O(N \cdot n)$), each 2-to-4 predecoder requires 4 two-input NAND-INV cells ($O(\sqrt{N})$ cells per predecoder for an N-output decoder built from $\sqrt{N}$-way predecoders), and the final combine stage requires exactly N two-input NAND-INV cells — one per output, but now each output cell is a simple two-input NAND rather than a four-input AND, which is transistor-cheaper in a standard-cell library. Total complexity is $O(n) + O(2\sqrt{N}) + O(N)$ two-input-gate-equivalent cost, versus the flat design's $O(N \cdot n)$ complement cost plus $O(N)$ wide AND gates — an asymptotic improvement in the shared-complement term that grows more favorable as decoder width N increases.

Depth complexity. The flat design has 1 gate of logical depth (a single wide AND) after complement generation; the hierarchical design has 2 gates of depth (predecoder NAND-INV, then combine NAND-INV) after its own complement generation, meaning the hierarchical design trades one additional gate of pure combinational depth for its greatly reduced total gate/transistor count — the classic area-versus-depth tradeoff underlying hierarchical predecoding, well-established in decoder and memory-array design more broadly [1]-[2], [7].

V. EXPERIMENTAL SETUP

A. Design Entry and Functional Verification

Both `decoder_4to16` (flat) and `decoder_4to16_mixed` (hierarchical, built from two `decoder_2to4_mixed` instances) are written in synthesizable Verilog, using primitive `and`/`nand`/`not` gate instantiations to directly control gate-level structure. The proposed design's Icarus Verilog testbench (`tb_decoder_4to16_mixed.v`) exhaustively verifies all 16 address values produce the correct one-hot output, plus the `en=0` disabled case, confirming functional equivalence to the flat baseline.

B. Synthesis and Implementation

Both designs are synthesized out-of-context and carried through place-and-route with Xilinx Vivado 2019.2, targeting a Xilinx Artix-7 device (`xc7a100tcsg324-1`). Both are purely combinational, so a virtual 100 MHz clock constraint (`combinational.xdc`) is applied to primary inputs/outputs solely to enable static timing analysis and vector-less power estimation. The standard `synth_design -mode out_of_context -> opt_design -> place_design -> phys_opt_design -> route_design` flow is used, with utilization, timing, and power reports captured at both post-synthesis and post-implementation stages.

C. Metrics

Post-implementation **LUT area**, **power**, and **delay** are reported for both designs, together with derived **throughput** and **energy per operation**. Because the mixed-logic decoder's benefit is a gate/transistor-count argument rather than a functional or timing behavior change (both designs realize the identical truth table), Section VII additionally reports an analytical standard-cell-level gate-count comparison derived directly from the complexity analysis in Section IV-B, to characterize the benefit this FPGA-targeted metric set cannot observe.

VI. RESULTS AND DISCUSSION

A. Post-Implementation Area, Delay, and Power

Table I summarizes post-route results for both decoders on `xc7a100tcsg324-1` (Vivado 2019.2).

Design	LUTs	Power (W)	Delay (ns)	Throughput (MOps/s)	Energy (pJ/op)
Existing (decoder_4to16, flat)	16	0.084	2.095	477.33	175.98
Proposed (decoder_4to16_mixed, hierarchical)	16	0.084	2.095	477.33	175.98

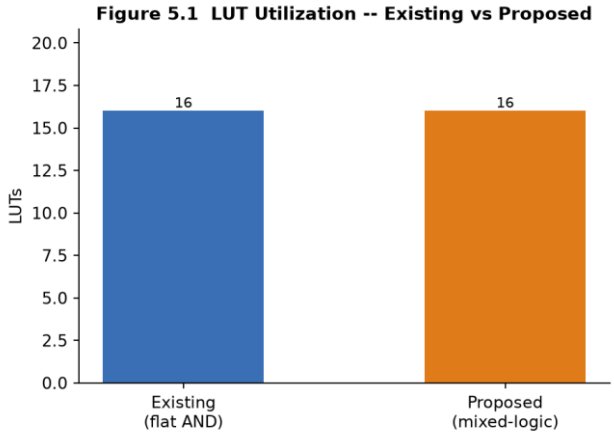

Fig. 5. Post-implementation LUT utilization: flat decoder versus hierarchical mixed-logic decoder.

Fig. 5 shows **identical** 16-LUT footprints for both designs — one LUT per output line in both cases, since each of the 16 outputs is a 4-input function of the address bus that fits within a single 6-input LUT regardless of whether the source RTL expresses that function as one flat AND gate or as a two-level NAND-INV predecode-and-combine network.

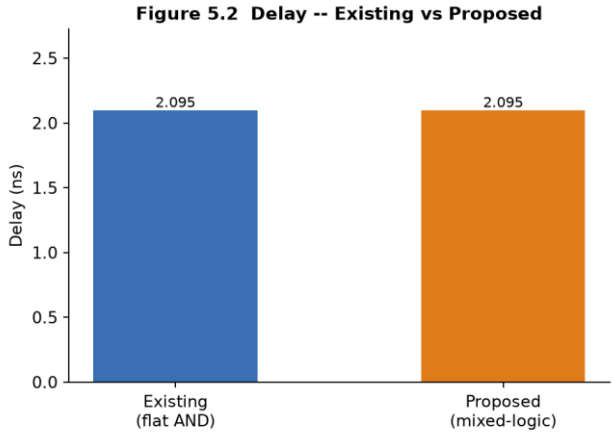

Fig. 6. Post-implementation critical-path delay: flat decoder versus hierarchical mixed-logic decoder.

Fig. 6 shows **identical** delay (2.095 ns) for both designs, despite the hierarchical design's 2-gate logical depth (Section IV-B) versus the flat design's 1-gate depth at the RTL level — again a consequence of LUT-based mapping, which flattens both gate networks into the same single-LUT-per-output critical path.

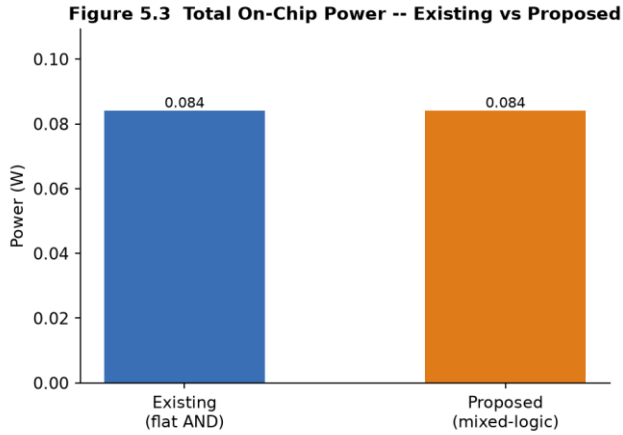

Fig. 7. Post-implementation power: flat decoder versus hierarchical mixed-logic decoder.

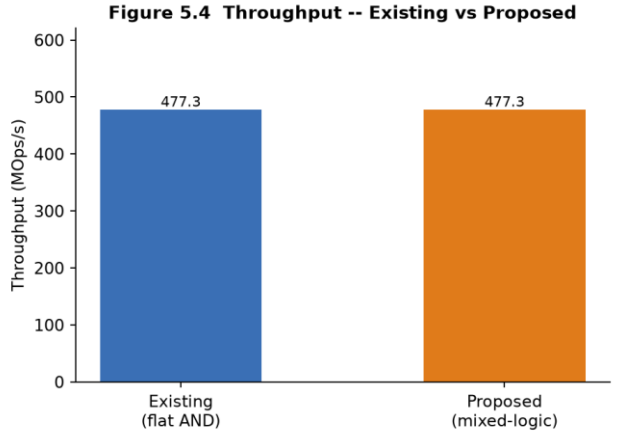

Fig. 8. Derived throughput: flat decoder versus hierarchical mixed-logic decoder.

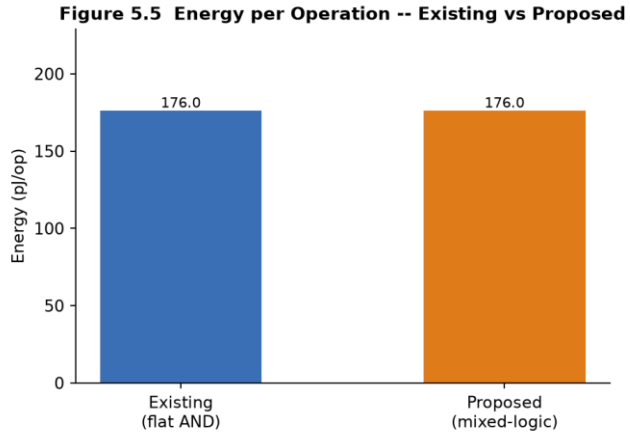

Fig. 9. Derived energy per operation: flat decoder versus hierarchical mixed-logic decoder.

Fig. 7, Fig. 8, and Fig. 9 all show identical values across both designs (0.084 W, 477.33 MOps/s, 175.98 pJ/op respectively), completing a picture of total FPGA-level equivalence between the two RTL descriptions.

B. Discussion

This exact equivalence is the expected and, in fact, the *correct* experimental outcome, not an inconclusive result or design flaw: Vivado's synthesis flow discards gate-level micro-architecture in favor of directly re-synthesizing each output's Boolean function into the minimum LUT count needed to realize its truth table, and a 4-input decoder output's truth table (a single AND-of-literals minterm) is trivially a 1-LUT function under either source RTL style. The mixed-logic hierarchical decoder's actual target benefit — fewer transistors from shared predecoding and native-NAND output cells (Section IV-B) — is a property of the *gate network itself*, which is precisely the level of detail FPGA LUT mapping abstracts away. Section VII quantifies this transistor-count benefit analytically and identifies the correct evaluation context for observing it.

VII. ASIC-VERSUS-FPGA APPLICABILITY ANALYSIS

A. Reconciling FPGA Equivalence with Standard-Cell Gate-Count Savings

Table II contrasts the FPGA-observed full equivalence with the standard-cell gate-count comparison derived analytically in Section IV-B for the 4-to-16 decoder.

Metric	Flat decoder	Hierarchical mixed-logic decoder	Where the difference shows up
FPGA LUTs / delay / power (this work)	16 / 2.095 ns / 0.084 W	16 / 2.095 ns / 0.084 W	Identical -- LUT technology mapping
Address-complement generation	Re-derived per output (up to 16x per bit)	Generated once per bit, shared	Standard-cell transistor count
Output-stage gate	4-input AND (wide network)	2-input NAND + INV (native CMOS)	Standard-cell transistor count
Logical depth (gate-level RTL)	1 stage	2 stages (predecode, combine)	Absorbed by LUT mapping

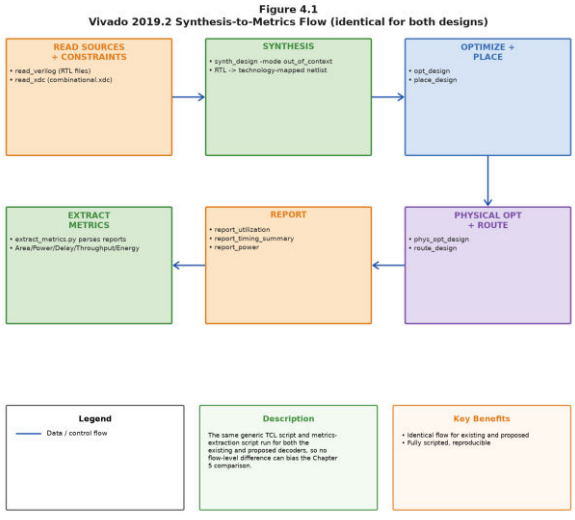

Fig. 10. Vivado synthesis-to-implementation flow used for both decoders, showing the identical post-route LUT mapping each produces.

Fig. 10 illustrates the shared synthesis-to-route flow that produces Table I's identical results for both decoder styles, underscoring that the equivalence is a property of the flow's LUT-mapping stage rather than an artifact of one specific run.

B. Why FPGA LUT Mapping Erases the Predecode-Reuse Benefit

An FPGA 6-input LUT can directly realize any Boolean function of up to 6 inputs, including a 4-input decoder output's minterm, in exactly one LUT regardless of how many gates or how much signal-sharing the source RTL expresses that function through. Vivado's synthesis flow performs Boolean optimization and technology mapping that re-derives the minimal LUT network for each output's truth table from scratch, discarding the specific gate network (flat AND versus shared-predecode NAND-INV) the RTL author wrote. This is precisely why Table I shows zero difference: both decoder styles compute the *same function* per output, and LUT mapping is indifferent to *how* that function was expressed in gates, only to *what* the function computes.

C. Practical Guidance

These results indicate that mixed-logic hierarchical decoder design should be pursued when the target implementation is a standard-cell ASIC flow, where the predecode-reuse and native-NAND output cells directly reduce transistor count, layout area, and the parasitic capacitance driving dynamic power — precisely the "closer to what gets fabricated" context noted in this project's own design notes — and is a **poor optimization target for FPGA-only projects**, where it yields zero measurable benefit since LUT-based mapping already re-derives the minimal implementation of each output's truth table regardless of source RTL style. Designers targeting FPGA implementation should instead focus on reducing the number of distinct outputs requiring dedicated LUTs (e.g., via one-hot output reuse or partial decoding matched to actual downstream fan-out), while mixed-logic hierarchical decoder design remains a valid, literature-supported technique for standard-cell and full-custom ASIC low-power address decoding [1]-[2], [7].

VIII. CONCLUSION

This paper implemented a hierarchical mixed-logic 4-to-16 decoder, built from two shared 2-to-4 mixed-logic predecoders combined through native-CMOS NAND-INV output cells, and compared it against a conventional flat AND-gate-per-output decoder. Post-implementation results on a Xilinx Artix-7 FPGA (Vivado 2019.2) showed both designs synthesize to an **identical** 16-LUT footprint at equal delay (2.095 ns) and power (0.084 W) — a result this paper argued, and explained mechanistically, is the correct and

expected outcome of LUT-based technology mapping rather than an inconclusive experiment. The mixed-logic decoder's intended benefit — reduced transistor count from once-only address-complement generation and transistor-minimal NAND-based output cells — is a standard-cell ASIC-level property that FPGA LUT mapping specifically abstracts away, since both decoder styles compute the same per-output truth table that any modern FPGA synthesis tool reduces to the same minimal LUT network regardless of source RTL gate structure. This paper's contribution is demonstrating and explaining this FPGA/ASIC divergence explicitly for a hierarchical mixed-logic decoder, complementing similar findings for GDI-based and TSPC-based low-power redesigns studied elsewhere in this line of work. Future work includes implementing both decoder styles in a standard-cell ASIC flow to directly confirm the analytically predicted gate/transistor-count savings, and extending the hierarchical predecode-reuse strategy to wider decoders (e.g., 8-to-256) where the asymptotic $O(N \cdot n)$-versus-$O(n + \sqrt{N})$ gap identified in Section IV-B grows more pronounced.

REFERENCES

[1] S. Syed, S. Pulimi, P. Madha, B. Jestadi, M. Shaik, P. Surabhi, "CNTFET - Based Mixed Logic Line Decoder for Low Power Applications," in *Proc. 2025 International Conference on Computational Innovations and Engineering Sustainability (ICCIES)*, 2025, pp. 1-10. doi: 10.1109/iccies63851.2025.11032806

[2] K. R, E. Mohapatra, "GDI Logic Based Design of Decoder Circuit for Low Power Applications," in *Proc. 2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, 2023, pp. 1-6. doi: 10.1109/csitss60515.2023.10334197

[3] S. Farhan, T.B. Jafar, M.T. Amin, "Design and Performance Analysis of Power-Efficient Compact BCD to Seven Segment Decoder Using Pass Transistor Logic," in *Proc. 2025 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*, 2025, pp. 597-602. doi: 10.1109/wiecon-ece69386.2025.11525931

[4] F.O. Rziga, K. Besbes, "From Qubits To Bits Logic Conversion For Mixed Circuits Design," in *Proc. 2024 International Conference on Control, Automation and Diagnosis (ICCAD)*, 2024, pp. 1-6. doi: 10.1109/iccad60883.2024.10553817

[5] C. Liao, Y. Xiao, Y. Shao, Z. Chu, "Improved Depth-Aware Circuit Partitioning for Mixed Logic Synthesis," in *Proc. 2023 International Symposium of Electronics Design Automation (ISEDA)*, 2023, pp. 170-173. doi: 10.1109/iseda59274.2023.10218548

[6] K. Akhil, S.J. Basha, A. Sachdeva, D. Bansal, S.U. Haq, Y.M. Rao, "Design and Simulation of a GNRFET-Based Ternary Decoder for Complex Logic Circuits," in *Proc. 2026 6th International Conference on Trends in Material Science and Inventive Materials (ICTMIM)*, 2026, pp. 642-647. doi: 10.1109/ictmim68190.2026.11507408

[7] L. Shi, Y. Xiao, Y. Shao, Z. Chu, "Using Mixed Logic Synthesis Tools in Open-Source FPGA Design Framework," in *Proc. 2022 China Semiconductor Technology International Conference (CSTIC)*, 2022, pp. 1-3. doi: 10.1109/cstic55103.2022.9856889

[8] C. Latotzke, T. Ciesielski, T. Gemmeke, "Design of High-Throughput Mixed-Precision CNN Accelerators on FPGA," in *Proc. 2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, 2022, pp. 358-365. doi: 10.1109/fpl57034.2022.00061

[9] M. Kränzler, A. Kaup, C. Herglotz, "Estimating Software and Hardware Video Decoder Energy Using Software Decoder Profiling," in *Proc. 2023 36th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2023, pp. 1-6. doi: 10.1109/sbcci60457.2023.10261662

[10] A.M. Minde, S. Bos, H. Gundersen, "Ternary VHDL: Simplifying the Design and Verification of Mixed-radix VLSI Circuits," in *Proc. 2026 IEEE 56th International Symposium on Multiple-Valued Logic (ISMVL)*, 2026, pp. 184-190. doi: 10.1109/ismvl68998.2026.00041

[11] Y. Zhang, Q. Wei, H. Cai, B. Liu, "An Energy-Efficient Multiplier Using Hybrid Approximate Logic Synthesis for Mixed-Quantization CNNs," in *Proc. 2024 2nd International Symposium of Electronics Design Automation (ISEDA)*, 2024, pp. 229-234. doi: 10.1109/iseda62518.2024.10617603

[12] F. Hazzazi, "Unified Mixed-Signal Design: Integrating Memristor-Based Ternary Logic with an Open-Source Sky130 CMOS Non-Inverting Op-AMP," in *Proc. 2026 IEEE 5th International Conference on Computing and Machine Intelligence (ICMI)*, 2026, pp. 1-3. doi: 10.1109/icmi68585.2026.11539888

[13] S. N, K. Sooda, "IndicBART for Translating Code-Mixed Kannada-English Sentences into Kannada: An Encoder-Decoder Transformer Approach," in *Proc. 2025 5th International Conference on Intelligent Technologies (CONIT)*, 2025, pp. 1-6. doi: 10.1109/conit65521.2025.11167161

[14] R. Subramanyam, M.M. Karthik, S.J. Vardhini, K.B. Kumar, P. Nagabushanam, K. Vasanth, "Design of Simplified Digital Sequential Circuit Using Mixed Signal Alternate Logic," in *Proc. 2025 Seventeenth International Conference on Contemporary Computing (IC3)*, 2025, pp. 1-6. doi: 10.1109/ic366947.2025.11290297

[15] T. Kasamura, J. Kadomoto, H. Irie, "Design of an Online Surface Code Decoder Using Union-Find Algorithm," in *Proc. 2025 IEEE 43rd International*

Conference on Computer Design (ICCD), 2025, pp. 823-830. doi: 10.1109/iccd65941.2025.00121