JAIST Repository
https://dspace.jaist.ac.jp/

Title	完全パイプライン型2LAL断熱論理回路の設計自動化および最適化
Author(s)	潮田, 裕也
Citation	
Issue Date	2026-03
Type	Thesis or Dissertation
Text version	ETD
URL	https://hdl.handle.net/10119/20594
Rights	
Description	Supervisor: 田中 清史, 先端科学技術研究科, 博士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Doctoral Dissertation

Design automation and optimization of fully pipelined 2LAL adiabatic logic circuits

Yuya Ushioda

Supervisor: Kiyofumi Tanaka

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

March, 2026

ABSTRACT

The explosive proliferation of artificial intelligence (AI), cloud computing, and the Internet of Things (IoT) has transformed power consumption in semiconductor integrated circuits into one of the most pressing technical and environmental challenges of the 21st century. Data centers supporting large-scale AI models now rival small nations in electricity demand, with projections estimating that AI-related infrastructure could account for 8% of global power by 2030. Simultaneously, billions of IoT and wearable devices operate under stringent battery or energy-harvesting constraints, where micro-watt-level efficiency improvements directly translate to extended operational life and enhanced user experience.

Adiabatic logic, first conceptualized by Landauer and later formalized by Charles Bennett in 1973, offers a radical alternative: reversible computation that preserves information and enables energy recovery. By charging and discharging capacitive loads through resonant power-clock networks, adiabatic circuits can theoretically eliminate $CV^2 f$ dissipation during switching. Early explorations in the 1980s and 1990s demonstrated logical reversibility in CMOS, but practical energy recovery remained elusive due to the absence of efficient resonant mechanisms and significant circuit overhead.

Among adiabatic families, *Two-Level Adiabatic Logic (2LAL)* stands out for its near-ideal energy recovery, robust noise margins, and compatibility with standard CMOS processes. Employing dual-rail encoding, transmission gates (T-gates), and four-phase power clocks, 2LAL achieves charge recycling with minimal non-adiabatic losses. Recent industry breakthroughs, such as Vaire Computing's Ice River test chip in 2025, have demonstrated net energy recovery with a 1.77× efficiency factor over conventional circuits in 22 nm CMOS, validating resonant adiabatic switching in silicon for the first time.

Even though 2LAL has an excellent potential in power dissipation, in its fully pipelined gate-level configurations, mandatory "decompute" operations for charge recovery cause severe buffer overhead. Input and output signals of each logic gate are buffered across pipeline stages. When the role of an output signal is completed, more specifically, after driving the decomputes of all fanout gates of the output signal, its replica is regenerated from the input and placed it in opposition to the target output signal for returning charge to the power supply. This decompute must be performed for every gate, leading to an explosive increase in buffer count and impractical circuit area.

This thesis aims to systematically reduce this buffer overhead by determining and optimizing which logic gates should receive early decompute and at what timing it should be applied. Early decompute involves decomputing a gate output immediately after computing the subsequent output and recomputing it when the subsequent output is decomputed. Although this doubles decompute operations per gate, it eliminates buffers between the first decompute and recompute phases, enabling substantial area reduction. While prior work proposed only simple heuristics for this concept, this research focuses on the structural properties of adiabatic circuits and formulates the problem of selecting early decompute targets and their timing as an integer linear programming (ILP) task from three distinct approaches:

1. Formulating early decompute insertion as a pipeline stage assignment problem under fixed scheduling to achieve optimization.
2. Reformulating early decompute target selection as a weighted maximum stable set problem on an Extended And-Inverter Graph (E-AIG), significantly reducing the search space in terms of total pipeline stages and candidate gate count, thereby shortening computation time.
3. Extending the first approach to jointly optimize early decompute timing and logic gate pipeline stage assignment, enhancing scheduling flexibility.

These methods were applied to all 11 circuits in the ISCAS-85 benchmark suite, a standard for combinational logic synthesis, and evaluated using the area efficiency metric $E_{\text{area}} = M_{\text{2LAL}}/M_{\text{CMOS}}$. Results showed up to 79.1% area reduction compared to 2LAL circuits without any early decompute, and up to 55.9% improvement over circuits obtained by prior heuristic methods. Notably, the second method achieved optimal solution extraction for all 11 circuits in under one second. This enabled analysis of the trade-off between area performance and computation time, clarifying the applicable scope of each method.

A rigorous energy-area-leakage trade-off analysis, validated through nanometer-scale LTspice simulations across 45 nm, 35 nm, and 22 nm nodes, confirmed that optimized 2LAL maintains

orders-of-magnitude energy superiority over CMOS even with residual overhead, with advantages strengthening in leakage-dominated advanced nodes due to exponentially rising static power. Tolerable area overheads reach 500×–1200× at practical frequencies, providing vast headroom for deployment.

Additionally, the appendix proposes two LC-resonant power-clock generators (2N2P-1 and 2N2P-2) for trapezoidal waveforms. Design equations and theoretical power consumption formulas were derived for each, and characteristics were evaluated and compared via LTspice simulation. This provides a theoretical guideline for selecting the optimal power-clock configuration from a power dissipation perspective, establishing a foundation for designing adiabatic logic circuits as a complete system encompassing both logic and power components.

This research establishes a comprehensive design automation framework for practical synthesis of Two-Level Adiabatic Logic (2LAL), simultaneously achieving circuit area optimization, computational efficiency, power system integration, and verified energy superiority in modern technology nodes—paving the way for deployment in ultra-low-power IoT, edge AI, implantable devices, and energy-harvesting systems.

Keywords: adiabatic logic, reversible computing, Two-Level Adiabatic Logic (2LAL), early decompute scheduling, buffer minimization, integer linear programming, stable set problem, pipeline rescheduling, energy-area trade-off, low-power VLSI, sustainable computing

Acknowledgements

I would like to express my deepest gratitude to all those who have supported me throughout this research.

First and foremost, I am profoundly grateful to my main supervisor, Professor Mineo Kaneko, for his consistent guidance, insightful advice, and warm encouragement from the inception of the research topic to the completion of this thesis. His rigorous yet supportive supervision has been instrumental in shaping the direction and depth of this work. After Professor Kaneko's retirement, I am sincerely thankful to Professor Kiyofumi Tanaka for taking over as my main supervisor and providing meticulous guidance and invaluable support during the final stages of the research.

I also extend my sincere appreciation to my sub-supervisor, Professor Yasushi Inoguchi, for his helpful advice and stimulating discussions on the sub-theme of this research, and to Professor Kunihiko Hiraishi for his sharp and constructive comments during the preliminary examination, which significantly improved the quality of this thesis. Furthermore, I would like to thank Professor Shigetaka Takagi, Assistant Professor Hiroki Sato, and Professor Atsushi Takahashi from Science Tokyo for their valuable insights and guidance on the sub-theme, as well as Professor Takahashi for his thoughtful feedback during the preliminary examination.

I am also indebted to all those who have been involved in this research for their daily support and cooperation; without their assistance, completing this work would not have been possible.

Finally, I express my heartfelt gratitude to my family and all those around me for their understanding and encouragement. I hope that this thesis will contribute, even in a small way, to the advancement of low-power circuit design.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Historical Context

The relentless expansion of digital technologies propelled by artificial intelligence (AI), cloud computing, and the Internet of Things (IoT), has transformed power consumption into a defining challenge of the 21st century. Data centers supporting large-scale AI models now rival small nations in electricity demand, with projections estimating that AI-related infrastructure could account for 8% of global power by 2030 [1, 2, 3, 4]. Simultaneously, billions of IoT and wearable devices operate under stringent battery constraints, where micro-watt-level improvements directly translate to extended operational life and enhanced user experience [5, 6].

For over six decades, complementary metal-oxide-semiconductor (CMOS) technology has sustained exponential performance growth through Dennard scaling and Moore's Law. However, as transistor dimensions approach atomic scales, fundamental physical barriers, including leakage currents, quantum tunneling, thermal dissipation, and short-channel effects, have rendered further power reduction via process scaling increasingly untenable [8]. The thermodynamic limit articulated by Rolf Landauer in 1961 [9] that erasing one bit of information dissipates at least $kT \ln 2$ joules of energy, further underscores the inherent inefficiency of irreversible computation in conventional logic.

Adiabatic logic, first conceptualized by Landauer and later formalized by Charles Bennett in 1973 [10], offers a radical alternative, i.e., reversible computation that preserves information and enables energy recovery. By charging and discharging capacitive loads through resonant power-clock networks, adiabatic circuits can theoretically eliminate $CV^2 f$ power dissipation during switching. Early explorations, such as MIT's Pendulum processor in the 1990s [11], demonstrated logical reversibility in CMOS, but practical energy recovery remained elusive due to the absence of efficient resonant mechanisms.

The thermodynamic imperative for reversible computation was formalized by Rolf Landauer in 1961 [9], establishing that irreversible bit erasure dissipates at least $k_B T \ln 2$ energy. Charles Bennett subsequently demonstrated in 1973 [12] that computation can be performed with arbitrarily low energy dissipation if it is carried out reversibly, that is preserving all intermediate information until it is explicitly "uncomputed" or returned to a known state.

Adiabatic logic families, including 2LAL, represent the physical embodiment of Bennett's reversible computing paradigm in classical CMOS hardware. By employing time-varying power-clocks and retractile decompute operations, these circuits avoid irreversible state transitions, recovering charge that would otherwise be dissipated in conventional logic. This direct mapping from reversible computation theory to circuit-level implementation enables energy dissipation approaching the Landauer limit in the absence of leakage—a feat unattainable with irreversible static CMOS gates.

Beyond near-term ultra-low-power applications, adiabatic circuits hold broader significance as a potential hardware foundation for large-scale reversible computing. Reversible architectures are theoretically essential for overcoming fundamental energy barriers in classical computing and serve as a bridge to quantum computing paradigms, where unitary (reversible) operations are intrinsic. While current quantum systems operate at cryogenic temperatures with superconducting or trapped-ion qubits, future room-temperature or hybrid classical-quantum platforms may leverage adiabatic/reversible classical subsystems for error correction, classical preprocessing, or reversible garbage management [12].

Among adiabatic logic families, *Two-Level Adiabatic Logic (2LAL)* [13] stands out for its near-ideal energy recovery, robust noise margins, and compatibility with standard CMOS processes. Employing dual-rail encoding, transmission gates (T-gates), and four-phase power clocks, 2LAL achieves charge recycling with minimal non-adiabatic losses. However, its mandatory "decompute" operations and fine-grained pipelining necessitate extensive buffer insertion to manage signal lifetimes, resulting in area overhead that has historically hindered industrial adoption.

Recent industry breakthroughs, such as Vaire Computing's Ice River test chip in 2025 [15, 16, 17], have reinvigorated interest in adiabatic and reversible computing. Fabricated in 22nm CMOS, Ice River demonstrated net energy recovery with a 1.77× efficiency factor over conventional circuits, validating resonant charge recycling in silicon for the first time. While such developments signal a new era of energy-efficient computing, they also highlight the persistent challenge of circuit-level overhead, particularly buffer proliferation in pipelined adiabatic designs.

The optimization framework developed in this thesis targets applications where absolute energy minimization is the primary constraint, often at the expense of moderate throughput and area overhead. Fully pipelined 2LAL exhibits superior energy efficiency at lower operating frequencies, as adiabatic charging losses decrease quadratically with reduced clock rates while charge recovery mitigates irreversible dissipation. This characteristic makes 2LAL particularly well-suited for energy-constrained domains where real-time performance is secondary to extended operational lifetime or harvested-energy sustainability.

Key target scenarios include:

- **Battery-powered IoT edge devices**: Sensor nodes in remote monitoring networks (e.g., environmental, agricultural, or structural health monitoring) operate intermittently with low duty cycles. Optimized 2LAL can enable multi-year battery life by reducing per-operation energy by orders of magnitude compared to static CMOS [18, 19, 20].

- **Energy-harvesting systems**: Devices powered by ambient sources (RF, vibration, thermal, or solar) have severely limited power budgets. Adiabatic logic's compatibility with resonant power delivery and low-frequency operation aligns naturally with harvested-energy profiles, as demonstrated in RF-powered IoT prototypes [21, 22, 23, 24].

- **Low-power security primitives**: Cryptographic hardware such as Physically Unclonable Functions (PUFs) for device authentication and key generation benefit from adiabatic implementations that provide inherent resistance to power side-channel attacks alongside ultra-low energy consumption. Numerous studies have proposed adiabatic PUFs for secure IoT authentication [25, 26].

- **Implantable and wearable medical sensors**: Applications requiring decades-long operation without battery replacement demand extreme energy efficiency under strict size constraints.

1.2 Significance and Research Positioning

This thesis is positioned at the intersection of theoretical adiabatic principles and practical circuit synthesis, addressing a critical gap that has limited the scalability of 2LAL: buffer overhead in fully pipelined implementations. While prior work has focused on device-level energy recovery or high-level architectural reversibility, few studies have systematically tackled the scheduling and timing optimization required to minimize auxiliary circuitry without compromising correctness or adiabatic efficiency.

The significance of this research lies in its dual contribution:

1. **Establishing a rigorous optimization framework** for early decompute scheduling—a technique that strategically shortens signal lifetimes to reduce buffer insertion while preserving data dependencies and charge recovery integrity.

2. **Providing a scalable, hierarchical methodology** that bridges exact optimization for medium-scale designs with heuristic-free convergence for large-scale benchmarks, offering designers a versatile toolkit for 2LAL synthesis.

By evaluating all proposed methods on the ISCAS-85 benchmark suite using the area efficiency metric $E_{\text{area}} = M_{\text{2LAL}}/M_{\text{CMOS}}$, this work quantifies the trade-off between precision, scalability, and computational cost.

In the broader context of sustainable computing, this research complements emerging industry efforts by focusing on logical and timing optimization rather than physical energy recovery alone. Together, these parallel advances signal a convergence toward practical, ultra-low-power adiabatic computing systems.

1.3 Thesis Overview

The remainder of this thesis is organized as follows:

- **Chapter 2** reviews the theoretical foundations of adiabatic logic, with emphasis on 2LAL's operational principles, buffer overhead challenges, and evaluation metrics.

- **Chapters 3–5** present the core methodological contributions: a progressive hierarchy of early decompute optimization techniques, ranging from exact ILP-based scheduling under fixed pipelines, through scalable stable set reformulation, to joint optimization with dynamic rescheduling.

- **Chapter 6** provides a rigorous energy-area-leakage trade-off analysis, deriving the optimal operating frequency and validating the energy superiority of optimized 2LAL through nanometer-scale simulations.

- **Chapter 7** synthesizes the results, discusses limitations, outlines practical guidelines for method selection, and proposes future research directions.

- **Appendices**:

  - **Appendix A** validates the re-implemented baseline heuristic against published results, confirming its correctness as a comparison baseline.
  - **Appendix B** presents validation results for the open-source MIP solvers (Cbc and HiGHS) using standard benchmark instances (MIPLIB 2017), confirming their correctness and reliability.
  - **Appendix C** explores complementary power-clock generation strategies to enhance overall system efficiency.

Chapter 2

Preliminary

2.1 Landauer Limit and Theoretical Lower Bound of Energy Dissipation

The Landauer principle [9] establishes a fundamental thermodynamic limit on energy dissipation in computation. It states that the erasure of one bit of information requires a minimum energy dissipation of

$$E_{\min} = k_B T \ln 2 \approx 2.87 \times 10^{-21} \text{ J/bit} \quad (T = 300 \text{ K}) \tag{2.1}$$

where $k_B = 1.38 \times 10^{-23}$ J/K is the Boltzmann constant and T is the absolute temperature.

This bound arises from the second law of thermodynamics. When a bit is erased—mapping two logical states (0 and 1) to a single state—the entropy of the system decreases by $\Delta S = k_B \ln 2$. To satisfy the second law, this entropy reduction must be offset by an increase in the entropy of the thermal environment, requiring a minimum heat dissipation of $T\Delta S = k_B T \ln 2$.

In conventional digital circuits based on static CMOS logic, information erasure occurs at nearly every gate [12]. For example:

- A 2-input AND gate accepts four possible input combinations but produces only two output states, erasing approximately one bit of information on average per operation.
- An inverter maps two input states to one output state, erasing exactly one bit.

As a result, even in the ideal case of zero resistance, zero leakage current, and infinitely slow switching, the energy dissipation per gate cannot fall below the order of $k_B T \ln 2$ [13].

Reversible computation offers a way to circumvent this limit during internal processing [10]. By ensuring that every operation has a unique inverse and that intermediate results are preserved until explicitly returned to a known state, no information is lost internally. In such systems, the Landauer limit applies only at the input and output interfaces, where external data is introduced or final results are extracted. For a computation with n input bits and m output bits, the theoretical minimum energy dissipation is therefore

$$E_{\min} = (n + m) k_B T \ln 2. \tag{2.2}$$

This principle motivates the development of adiabatic and reversible logic families, which aim to recycle charge and avoid irreversible state transitions, thereby approaching the thermodynamic limit in ultra-low-power applications.

2.2 Low power consumption technology for CMOS semiconductors

This section describes the power consumption of CMOS semiconductors, which are currently the most common type of semiconductor, and the technologies to reduce it. The equation for power consumption of CMOS semiconductors can be expressed as follows [27]

$$P = \frac{1}{2}CV_{dd}^2 fN + QV_{dd}fN + I_1 V_{dd} \tag{2.3}$$

P is the power consumption, C is the load capacitance, f is the operating frequency, V_{dd} is the supply voltage, N is the signal switching coefficient, Q is the charge due to through-current and I_1 is the leakage current. It is the power consumption generated by the switching of the signal lines, represented by the first term on the right-hand side. It is an element that accounts for more than 70% of a semiconductor integrated circuit. It is the power dissipated by the through power in the cell, which is represented by the second term. It accounts for $10 - 30\%$ of semiconductor integrated circuits. The power dissipation due to leakage current, which is represented by the third term. To reduce the power consumption of CMOS, the following measures are effective.

(1) Lower the supply voltage V_{dd}. (Lower voltage)

(2) Lower the load capacitance of the circuit. (Lower capacitance)

(3) Reduce the frequency f or the switching factor N of the circuit. (low toggling)

The reduction in power consumption to date is largely due to advances in hardware technology. In particular, since processors have been made using integrated circuits, the number of devices on a chip has increased four-fold in three years (Moore's law). As a result of Dennard's scaling law [28], which states that the smaller the element, the faster it operates and the lower its power consumption, energy efficiency has continued to increase.

However, the slope of the energy efficiency increase slackens around the year 2000, doubling at a rate of 2.7 times. One reason for this is the end of Dennard's scaling law [28]. According to Dennard's scaling law, the power supply voltage and transistor threshold voltage of MOS transistors shrink in proportion as the size of the transistors gets smaller. However, if the threshold voltage is made too small, the leakage current increases significantly and power consumption increases, making it difficult to lower the threshold voltage. If the power supply voltage is lowered without lowering the threshold voltage, the operating speed slows down, and as a result, the power supply voltage cannot be lowered either.

While lowering the power supply voltage is the most effective method for achieving low power consumption, there are various other methods. These can be broadly classified into two categories: circuit technology and process technology. To cite a representative example of each, there are three low-toggle methods as circuit technologies.

(1) Low-speed internal operation by dividing the clock

(2) Clock stop function by operation mode

(3) Function to stop operation of non-selectable function blocks

The clock is the signal with the highest switching coefficient in the circuit. Lowering the internal operation reduces the number of data that can be transferred per unit time, so parallel processing of data is required. Therefore, the size of the circuit increases compared to high-speed operation.

This technique is particularly effective when many functions are shared in a processor. The disadvantage is that a combination circuit is inserted into the clock, causing a difference in delay time (clock skew) between the separated clocks. To counter this problem, the layout design, which is a post-process, is performed in advance to facilitate timing convergence.

The last method is similar, but is effective when the scale of gates to be stopped is small. Specifically, the operation of a functional block that has not been selected by the selector at the latter stage is stopped by a control circuit assigned to the stage before the block (e.g., by fixing the input data or by forcing a reset to a flip-flop).

Subthreshold circuits operate with a supply voltage V_{dd} below the transistor threshold voltage V_t [29, 30]. In a CMOS inverter, both the nMOS and pMOS transistors remain off in the strong sense, but subthreshold leakage current I_{sub} enables logic functionality.

The subthreshold leakage current I_{sub} is exponentially dependent on the gate-source voltage V_{GS}:

$$I_{sub} \propto e^{V_{GS}/(nV_T)},$$

where $V_T = kT/q$ is the thermal voltage and n is the subthreshold slope factor. Higher $|V_{GS}|$ results in larger leakage.

Input HIGH ($V_{in} = V_{dd}$)

- nMOS: $V_{GS,n} = V_{dd}$, $V_{DS,n} = V_{out}$
- pMOS: $V_{SG,p} = 0$ (i.e., $|V_{GS,p}| = 0$), $V_{SD,p} = V_{dd} - V_{out}$

Since $V_{GS,n} = V_{dd} > 0 = |V_{GS,p}|$, the nMOS leaks significantly more than the pMOS. The load capacitance C_L discharges through the nMOS:

$$\frac{dV_{out}}{dt} = -\frac{I_{sub,n}}{C_L} \quad \Rightarrow \quad V_{out} \to 0.$$

Thus, the nMOS is *weakly off*, while the pMOS is *strongly off*.

Input LOW ($V_{in} = 0$)

- nMOS: $V_{GS,n} = 0$
- pMOS: $V_{SG,p} = V_{dd}$ (i.e., $|V_{GS,p}| = V_{dd}$)

Now the pMOS leaks much more than the nMOS. The capacitance charges through the pMOS:

$$\frac{dV_{out}}{dt} = \frac{I_{sub,p}}{C_L} \quad \Rightarrow \quad V_{out} \to V_{dd}.$$

The pMOS is *weakly off*, and the nMOS is *strongly off*.

The fundamental solution to leakage current caused by semiconductor miniaturization is to reduce leakage current by improving process technology. Analysis of leakage current divides it into three categories: subthreshold leakage, which flows through the channel when it is off; junction leakage, which leaks from the source and drain to the substrate; and gate dielectric leakage, which leaks from the gate [27].

One of the effects of miniaturization on circuits in semiconductors is that the capacitance of capacitors formed by wiring in close proximity to each other becomes large, causing a large current to flow between them. Another phenomenon is that current can pass through the insulating film, which has become thinner due to miniaturization, due to the tunnel effect. The former is a problem in the wiring area, while the latter is a problem in the area of the gate insulating film, which is extremely thinner than the wiring area.

The former requires the use of low-k materials and air gaps, while the latter requires the use of high-k materials, i.e., materials that can ensure a large dielectric constant without making the film thinner. Improvement research in the process is focused on how to create insulating films with low or high dielectric constants.

Apart from these techniques, Domain Specific Architecture (DSA), which fundamentally rethinks the architecture of conventional CMOS processors to optimize performance, energy consumption, and chip cost, is also attracting attention. Reversible computation, spin-tronics semiconductors, and recently, semiconductors using optical signals, which have been studied for a long time, are also examples of low-power methods.

2.3 Adiabatic Logic

2.3.1 Principle of low power consumption of adiabatic logic

An adiabatic logic circuit refers to a circuit that realizes ultra-low power consumption through the utilization of the concept of the "adiabatic process" [31]. In the field of electrical and electronic circuits, the fundamental characteristics of adiabatic behavior can be summarized into two key points.

(A1) Assuming a system comprised of a power-supply section and a computational section, the current (charge) flowing from the power-supply to the computational section will be returned to the power-supply section. This implies that the energy drawn from the power-supply section will be regenerated by the return of charge (reverse current) from the computational section.

(A2) By keeping the voltage across a conductive element and the current flowing through it at sufficiently low levels, the energy dissipated as heat is minimized.

Based on (A1), the computational section must be supplied with AC-type (alternate-current-type) power-source to achieve charge recovery. Since the computational section primarily comprises of resistive and capacitive elements, the ramp-up/ramp-down speed of the AC power-source should be sufficiently slow, in comparison to the circuit's time constant ($R \times C$), to avoid increasing the voltage across a resistive element due to the delay for charging a capacitive element.

The principle of the adiabatic circuit is often demonstrated through a circuit and power-source waveform, as depicted in Figure 2.1 [32]. Assuming that the time period T is significantly larger than the time constant CR, during the first section where the power-source voltage increases from 0 to V_{dd}, the current $i(t)$ is assumed to be approximated as CV_{dd}/T. In this case, the energy which is extracted from the power source can be approximated as,

$$\frac{RC^2{V_{dd}}^2}{T} + \frac{C{V_{dd}}^2}{2} \tag{2.4}$$

is extracted from the power-source. Similarly, the energy,

$$\frac{C{V_{dd}}^2}{2} - \frac{RC^2{V_{dd}}^2}{T} \tag{2.5}$$

is returned to the power-source during the third section where the voltage of the power-source moves from V_{dd} to 0. In total, the energy,

$$\frac{2RC^2{V_{dd}}^2}{T} \tag{2.6}$$

is dissipated in one cycle of the power-source waveform, or the average power dissipation becomes

$$\frac{RC^2{V_{dd}}^2}{2T^2}. \tag{2.7}$$

In relation to condition (A2), the following guidelines for switching are also regarded as critical factors in ensuring the preservation of adiabatic operation integrity.

(R1) Any OFF switch should not be turned ON when the voltage between two terminals of the switch is not zero.

(R2) Any ON switch should not be turned OFF when the current flowing on the switch is not zero.

In the following, a trapezoidal waveform used as a power-source is referred to a "power-clock". Furthermore, the waveforms of individual sections are named "ramp-up" (the level moves from 0 to V_{dd}), "high-hold" or "1-hold" (the level remains at V_{dd}), "ramp-down" (the level moves from V_{dd} to 0), and "low-hold" or "0-hold" (the level remains at 0), respectively. These names are not limited to the power-clock waveform, but also apply to node voltage waveforms.

Adiabatic circuits achieve ultra-low power consumption through charge recovery and slow voltage transitions, using AC power (trapezoidal waveform) and strict switching rules (R1, R2) [31]. Parallel operation enhances throughput by synchronously executing multiple computations, sharing a power-clock, while maintaining energy efficiency via charge recovery. This makes them ideal for energy-critical applications like IoT devices and medical implants, despite slower operation and complex design. In contrast, CMOS circuits rely on DC power, dissipate energy during capacitive charging/discharging, and prioritize high-speed operation. While CMOS supports parallelization for high performance, its power consumption scales poorly compared to adiabatic circuits, which excel in low-power, parallelized scenarios.

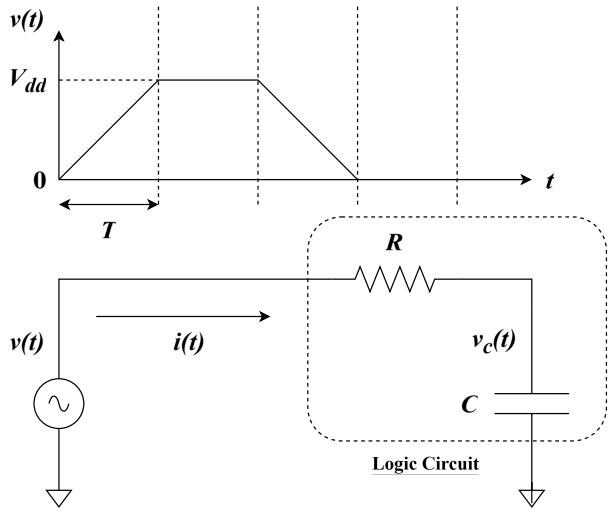


Figure 2.1. RC equivalent circuit illustrating the principle of adiabatic logic. Charge is supplied during ramp-up and recovered during ramp-down of the power-clock, minimizing energy dissipation.

2.3.2 Loss Mechanisms in Adiabatic Logic

Adiabatic logic is a low-power design approach that aims to minimize energy dissipation in digital circuits by recycling charge stored in capacitive nodes, ideally reducing losses compared to conventional static CMOS circuits [31]. However, practical implementations face challenges, especially as device sizes shrink into the sub-micrometer regime. Non-ideal conditions, such as the absence of zero-threshold-voltage (V_{th}) transistors and increasing leakage currents, introduce additional loss mechanisms that dominate energy consumption and impose a lower bound on energy dissipation. This document details three primary loss mechanisms in adiabatic logic: adiabatic losses, leakage-related losses, and non-adiabatic losses, their dependence on operating frequency f, and the existence of an optimal frequency that minimizes total energy dissipation per cycle [33].

In an ideal adiabatic system, energy dissipation scales inversely with the switching time (or proportionally to the operating frequency f). The energy dissipation per cycle for adiabatic switching is given by:

$$E_{\text{adiabatic}} = \frac{RC^2V_{DD}^2}{T} = RCV_{DD}^2 f \tag{2.8}$$

where R is the resistance of the charging/discharging path, C is the load capacitance, V_{DD} is the supply voltage, $T = \frac{1}{f}$ is the switching period, and f is the operating frequency. This expression shows that adiabatic losses increase linearly with frequency. At low frequencies, the slower charging/discharging process allows more time for charge to be recycled, reducing dissipation. However, as frequency increases, the charging process becomes less adiabatic, leading to higher energy losses [32].

As devices shrink into the sub-micrometer regime, leakage currents become a significant contributor to energy dissipation in adiabatic logic. The dominant leakage mechanism is the sub-threshold current, expressed as:

$$I_D = I_{D0} e^{\frac{V_{GS}-V_{th}}{nV_T}} \left(1 - e^{-\frac{V_{DS}}{V_T}}\right) \tag{2.9}$$

where I_{D0} is a process-dependent constant, V_{GS} is the gate-to-source voltage, V_{DS} is the drain-to-source voltage, V_{th} is the threshold voltage, $V_T = \frac{kT}{q}$ is the thermal voltage (typically $\approx 26\,\text{mV}$ at room temperature), and n is the sub-threshold swing factor (typically $1 < n < 2$) [27]. The sub-threshold current flows when $V_{GS} < V_{th}$, and its magnitude depends on V_{DS}. When $V_{DS} = 0$, no leakage current flows. However, for V_{DS} values that are multiples of the thermal voltage, the leakage current approaches its maximum. Additional leakage mechanisms include junction leakage and gate oxide tunneling currents, which are significant in state-of-the-art CMOS processes due to thin gate oxides.

In adiabatic logic, during the evaluation, hold, and recovery phases, leakage currents flow from the supply voltage V_{DD} to ground, dissipating charge that cannot be recovered. These leakage mechanisms can be modeled by a mean leakage current $\overline{I_{\text{leak}}}$, leading to an energy dissipation per cycle of:

$$E_{\text{leak}} = V_{DD}\overline{I_{\text{leak}}}\frac{1}{f} \tag{2.10}$$

Since the energy dissipation is proportional to the time per cycle ($\frac{1}{f}$), leakage-related losses increase at lower frequencies because leakage currents accumulate over a longer time interval, resulting in greater charge loss per cycle [7].

Non-adiabatic losses arise due to the non-zero threshold voltage of transistors in adiabatic logic circuits. These losses occur during switching events when the voltage across a transistor changes abruptly, preventing full charge recovery. The energy dissipation per cycle due to non-adiabatic losses is given by:

$$E_{\text{non-adiabatic}} = \frac{1}{2} C V_{th,p}^2 \tag{2.11}$$

where $V_{th,p}$ is the threshold voltage of the p-type transistor (or the relevant transistor in the circuit). Unlike adiabatic and leakage losses, non-adiabatic losses are independent of the operating frequency, contributing a constant energy offset across the entire frequency range [7].

The total energy dissipation per cycle in adiabatic logic is the sum of the three loss mechanisms:

$$E_{\text{total}} = E_{\text{adiabatic}} + E_{\text{leak}} + E_{\text{non-adiabatic}} = RCV_{DD}^2 f + V_{DD}\overline{I_{\text{leak}}}\frac{1}{f} + \frac{1}{2} C V_{th,p}^2 \tag{2.12}$$

The frequency dependence of these losses leads to an optimal operating frequency where the total energy dissipation is minimized. At low frequencies, leakage losses dominate due to the $\frac{1}{f}$ term, while at high frequencies, adiabatic losses dominate due to the linear f term. Non-adiabatic losses remain constant, acting as an offset. The total energy dissipation can be visualized as a function of frequency, typically showing a U-shaped curve with a minimum at the optimal frequency f_{opt}.

To find the optimal frequency, we minimize E_{total} with respect to f:

$$\frac{dE_{\text{total}}}{df} = RCV_{DD}^2 - \frac{V_{DD}\overline{I_{\text{leak}}}}{f^2} = 0 \tag{2.13}$$

Solving for f:

$$f_{\text{opt}} = \sqrt{\frac{V_{DD}\overline{I_{\text{leak}}}}{RCV_{DD}^2}} = \sqrt{\frac{\overline{I_{\text{leak}}}}{RCV_{DD}}} \tag{2.14}$$

At this frequency, the adiabatic and leakage losses are balanced, and the total energy dissipation is minimized. The existence of this optimal frequency highlights a key trade-off in adiabatic logic: operating at very low frequencies increases leakage losses, while operating at high frequencies increases adiabatic losses [33].

In conclusion, adiabatic logic offers significant energy savings compared to static CMOS, but its efficiency is limited by adiabatic, leakage, and non-adiabatic losses. The frequency-dependent nature of adiabatic and leakage losses, combined with the frequency-independent non-adiabatic losses, results in an optimal operating frequency that minimizes energy dissipation per cycle. Careful design, including minimizing leakage currents and optimizing switching frequencies, is essential to maximize the benefits of adiabatic logic in sub-micrometer CMOS processes.

2.3.3 Adiabatic Logic families

Adiabatic logic circuits are designed to minimize power dissipation by leveraging reversible or near-reversible energy transfer between the circuit and the power supply [33]. As illustrated in Figure 2.2, these circuits can be broadly classified into Charge Recovery Logic and Reversible Logic. This study focuses on circuits based on the Charge Recovery Logic principle, which recycle the charge stored in load capacitances to reduce energy dissipation.

Asymptotically adiabatic logic circuits incur power dissipation due to the non-zero rate of change of the supply voltage. By slowing the voltage transition, power dissipation can be reduced to very low levels, though this comes at the cost of lower operating frequency. Asymptotically adiabatic logic is typically classified into two categories: quasi-adiabatic logic and fully adiabatic logic [33].

Quasi-adiabatic logic circuits significantly reduce power dissipation by using a gradually varying power-clock, which slows the rate of change of the driving voltage. However, they experience adiabatic losses due to non-ideal switching, where current flows through transistors with finite resistance. These losses increase with the frequency of the power-clock. Quasi-adiabatic circuits are characterized by simpler architectures and power-clock systems compared to fully adiabatic circuits, making them more practical for integration with existing CMOS technologies. They are further divided into two subcategories: the static approach and the dynamic approach. The static approach maintains stable logic states during operation, similar to static CMOS, and is less sensitive to timing variations but may have slightly higher power dissipation. The dynamic approach relies on continuous charging and discharging cycles synchronized with the power-clock, achieving higher energy efficiency but requiring precise timing control, which makes it more susceptible to clock skew and synchronization errors.

The following are widely recognized quasi-adiabatic logic families, each with distinct structural and operational characteristics:

1. **Efficient Charge Recovery Logic (ECRL)** [35]: ECRL employs a differential cascode structure with PMOS and NMOS transistors arranged in a complementary configuration. It operates with a four-phase power-clock, where two phases handle logic evaluation and two manage charge recovery. The circuit's simplicity and compatibility with standard CMOS processes make it a popular choice. ECRL achieves significant power savings at low frequencies, but its adiabatic losses increase with higher frequencies, and precise power-clock synchronization is essential to prevent charge leakage. It is widely used in applications requiring moderate performance and low power, such as portable electronics and wireless sensor nodes.

2. **2N-2N2P Adiabatic Logic** [36]: This family uses a cross-coupled configuration with two NMOS transistors for logic evaluation and two NMOS/PMOS pairs for charge recovery, hence the name "2N-2N2P." It operates with a four-phase power-clock, offering improved robustness over ECRL due to its balanced charge transfer paths. The design ensures stable operation under varying conditions, but its increased complexity and sensitivity to parasitic capacitances require careful optimization. It is suitable for applications where reliability is prioritized, such as in embedded systems.

3. **Positive Feedback Adiabatic Logic (PFAL)** [37]: PFAL incorporates positive feedback through cross-coupled inverters to enhance charge recovery and stabilize logic states. It uses a four-phase power-clock and is highly efficient at low frequencies, making it ideal for ultra-low-power applications like battery-powered sensors.

However, its sensitivity to process variations and increased transistor count pose design challenges. PFAL's feedback mechanism improves switching efficiency but requires precise power-clock alignment to maintain performance.

4. **NMOS Energy Recovery Logic (NERL)** [38]: NERL primarily uses NMOS transistors to reduce reliance on PMOS devices, simplifying fabrication and reducing circuit area. It operates with a single-phase or multi-phase power-clock, depending on the implementation. NERL is area-efficient and suitable for specific logic functions, but its limited functionality and higher adiabatic losses at elevated frequencies restrict its use in complex systems. It is often employed in niche applications where area constraints are critical, such as in compact IoT devices.

5. **Clocked Adiabatic Logic (CAL)** [39]: CAL integrates a single-phase power-clock with auxiliary control signals to manage charge recovery and logic evaluation. Its compatibility with standard CMOS logic makes it appealing for hybrid designs combining adiabatic and conventional logic. However, CAL experiences higher adiabatic losses compared to multi-phase designs, limiting its efficiency in high-performance applications. It is commonly used in systems requiring straightforward integration with existing technologies, such as mixed-signal circuits.

6. **True Single-Phase Adiabatic Logic (TSEL)** [40]: TSEL is optimized for single-phase power-clock operation, reducing the complexity of clock distribution. It achieves charge recovery and logic evaluation within a single clock cycle, improving integration simplicity. However, its scalability is limited, and it is more sensitive to timing errors than multi-phase designs. TSEL is suitable for low-complexity, low-power systems where clock simplicity is a priority, such as in wearable electronics.

7. **Source-Coupled Adiabatic Logic (SCAL)** [41]: SCAL employs source-coupled transistor pairs to achieve adiabatic switching, similar to current-mode logic. It operates with a multi-phase power-clock, offering robust performance in low-power applications. SCAL's high efficiency makes it suitable for specialized systems, but the complexity of power-clock generation and distribution remains a significant challenge. It is often used in applications requiring high reliability and low power dissipation, such as in precision instrumentation.

Quasi-adiabatic logic offers several advantages, including simpler circuit architectures, compatibility with CMOS processes, and significant power savings compared to conventional logic, particularly at low frequencies. However, challenges include adiabatic losses that increase with power-clock frequency, the need for precise synchronization, and a trade-off between power efficiency and operating speed.

Fully adiabatic logic circuits eliminate non-adiabatic losses by ensuring that all charge on the load capacitance is recovered by the power supply, achieving theoretically perfect energy efficiency with zero non-adiabatic loss:

$$E_{\text{non-adiabatic}} = 0.$$

This is accomplished through fully reversible charge transfer, where the power-clock drives the circuit in a controlled, lossless manner. However, fully adiabatic circuits are significantly more complex, requiring intricate architectures and precise power-clock synchronization. They also face challenges related to low operating speeds and increased area overhead.

The following are notable fully adiabatic logic families:

1. **Pass Transistor Adiabatic Logic (PAL)** [43]: PAL utilizes pass transistors to control charge flow, ensuring reversible charge transfer between the load capacitance and the power supply. It typically operates with a four-phase power-clock, where each phase manages a specific stage of charge transfer or logic evaluation. PAL achieves exceptional energy efficiency, making it suitable for ultra-low-power applications such as energy-harvesting devices and battery-powered sensors. However, its low operating frequency, due to the need for slow voltage transitions, and susceptibility to noise and process variations limit its practical use. The need for multiple control signals further increases design complexity, making PAL challenging to implement in large-scale systems.

2. **Split-Rail Charge Recovery Logic (SCRL)** [42]: SCRL employs a split-rail power supply, where two complementary power-clock signals drive the circuit to achieve complete charge recovery. It operates with a multi-phase power-clock, typically four phases, ensuring that charge is transferred back to the power supply without dissipation. SCRL's robust charge recovery mechanism makes it ideal for complex logic circuits in low-power systems, such as implantable medical devices. However, its high complexity, including the need for multiple synchronized clock signals and increased transistor count, results in significant area overhead and reduced operating speed. Timing mismatches in the power-clock can lead to charge leakage, necessitating precise synchronization.

Fully adiabatic logic provides unparalleled energy efficiency by eliminating non-adiabatic losses, making it ideal for applications where power consumption is critical. However, its complex architectures, stringent synchronization requirements, and inherently low operating speeds pose significant barriers to widespread adoption. The increased area and design effort further limit its practicality for large-scale integration.

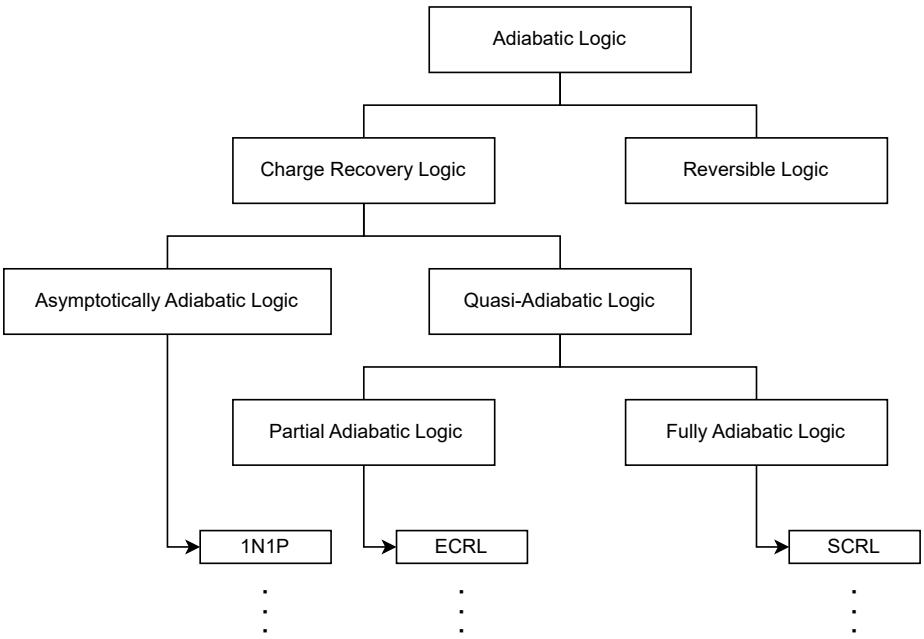


Figure 2.2. Classification of adiabatic logic circuits. This study focuses on charge recovery logic, specifically quasi-adiabatic families, with 2LAL as the target architecture.

2.3.4 Selection of 2LAL as the Target Architecture

Numerous adiabatic logic families have been proposed over the decades, ranging from early quasi-adiabatic designs (e.g., ECRL, PFAL) to fully adiabatic variants (e.g., SCRL, PAL) [33]. This thesis focuses specifically on Two-Level Adiabatic Logic (2LAL) for several compelling reasons that align with the goal of developing practical, automatable synthesis algorithms for ultra-low-power circuits.

Table 2.1 provides a comparative overview of representative adiabatic families, highlighting key attributes relevant to energy efficiency, implementation complexity, and suitability for automated design.

The advantages of 2LAL that justify its selection as the primary target are as follows:

- **Full adiabaticity with minimal non-adiabatic losses**: Unlike quasi-adiabatic families (ECRL, PFAL), 2LAL eliminates threshold-voltage drops during charge recovery by using transmission gates driven by dual-rail power-clocks, achieving theoretically complete energy recovery in the absence of leakage [14].

- **Simple and regular structure**: All logic functions and pipeline buffers are constructed exclusively from dual-rail transmission gates—a single primitive element compatible with standard CMOS processes. This regularity greatly simplifies automated layout generation, timing analysis, and optimization algorithm design compared to families requiring cross-coupled inverters, auxiliary clocks, or split-rail supplies.

- **Moderate clocking complexity**: The four-phase overlapping power-clock scheme provides robust pipelining and decompute support while remaining significantly simpler than multi-rail or split-level approaches (e.g., SCRL), facilitating scalable synthesis flows.

- **High amenability to algorithmic optimization**: The retractile decompute mechanism and explicit buffer chains create a well-defined combinatorial optimization problem (signal lifetime minimization under dependency constraints), ideal for exact methods such as ILP and graph-theoretic reformulations developed in this thesis.

These attributes collectively position 2LAL as the most promising candidate for bridging the gap between theoretical adiabatic efficiency and practical, automatable design. While other families may offer marginal advantages in specific metrics (e.g., single-phase clocking in CAL), none combine full adiabaticity, structural simplicity, and synthesis tractability to the extent that 2LAL does. The optimization framework presented herein exploits these properties to achieve unprecedented buffer reduction, advancing 2LAL toward real-world deployment in energy-critical applications.

Table 2.1. Comparison of Major Adiabatic Logic Families

Family	Adiabaticity	Clock Phases	Primary Devices	Suitability for Automated Synthesis
ECRL	Quasi	4	CMOS (cross-coupled)	Moderate (floating nodes)
PFAL	Quasi	4	CMOS (positive feedback)	Moderate (complex topology)
CAL	Quasi	1–4	CMOS + auxiliary clocks	Low (timing-sensitive)
SCRL	Fully	Multi-rail	Split-level charge recovery	Low (complex clocking)
PAL	Fully	4	Pass transistors	Moderate (noise sensitivity)
2LAL (this work)	**Fully**	**4**	**Transmission gates only**	**High (simple, regular structure)**

2.4 2LAL (Two-Level Adiabatic Logic)

This thesis focuses on a circuit architecture called Two-Level Adiabatic Logic (2LAL), which is one of the limited set of logic families that achieve asymptotic adiabaticity [31]. The primary characteristics of the 2LAL are as follows.

(F1) All logic signals are dual-rail encoded.

(F2) Power-clock is also dual-rail, that is, a pair of power-clocks having complementary waveforms.

(F3) A dual-rail transmission gate (T-gate) is the sole type of basic-component for building a 2LAL circuit. The schematic symbol and the transistor-level implementation of the T-gate are shown in Figure 2.3, where the terminal C (or $(C, \overline{C})$) is called a control terminal, whereas terminals B ($(B, \overline{B})$) and C ($(C, \overline{C})$) are called switch terminals.

This thesis focuses in particular on a fully pipelined 2LAL circuit because of the potential of pipeline operation which alleviates the inherent sluggishness of adiabatic operation. In a fully pipelined 2LAL circuit, the timing of pipeline stages is controlled by power-clocks, which is regarded as the fourth key feature of 2LAL.

(F4) Four synchronous power-clocks, each of which is time shifted by 0, T (the length of one section of the trapezoidal waveform), $2T$ or $3T$, are used. They are named ϕ_0, ϕ_1, ϕ_2 and ϕ_3, respectively. As it is shown in (F2), each of dual-rail power-clock has a form $(\phi_i, \overline{\phi_i}) = (\phi_i, \phi_{(i+2)_{\mathrm{mod}\,4}})$, $i \in \{0, 1, 2, 3\}$, whereas $(\phi_i, \overline{\phi_i})$ is denoted by ϕ_i in short if single-rail/dual-rail is clear from the context.

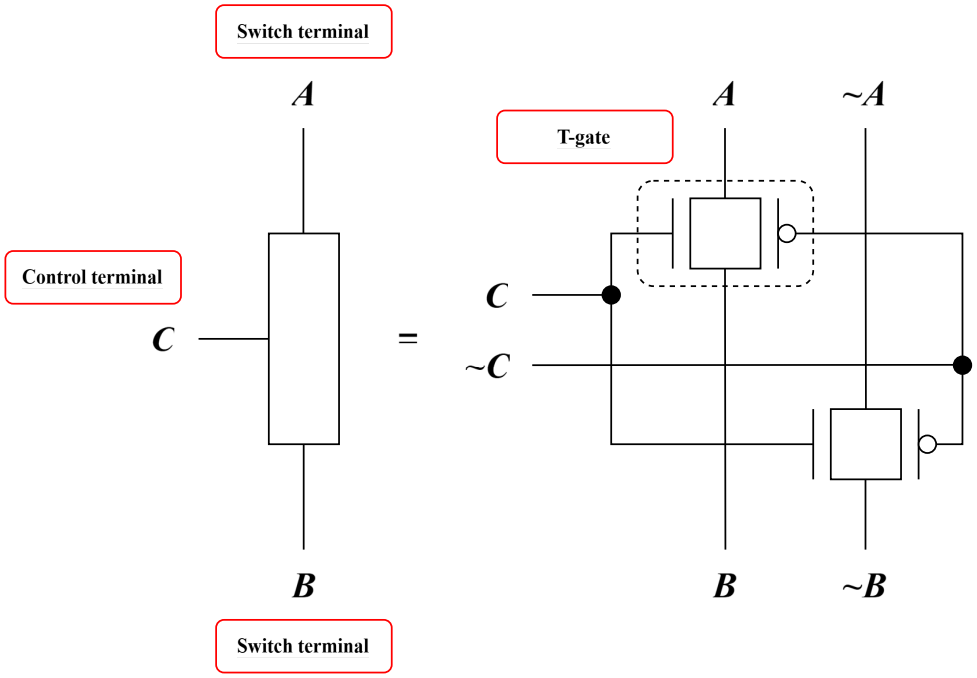


Figure 2.3. Transistor-level implementation of the dual-rail T-gate in 2LAL. Terminals A and B are switch terminals; C is the control terminal. The T-gate enables both signal transfer and charge recovery (decompute).

2.4.1 T-gate

The T-gate operates as a two-wire switch, where the complementarity of the two-wire control signal causes both transmission gates to be either ON or OFF simultaneously. Due to the requirements of (R1) and (R2) [32, 13], the control terminal node (Figure 2.3, C) can ramp up (switching from OFF to ON) or ramp down (switching from ON to OFF) only when both switch terminal nodes (Figure 2.3, A, B) are simultaneously in a 1-hold or 0-hold state.

The typical usage of the T-gate includes two types of operations: signal transfer from one switch terminal node to the other, and signal transfer from the control terminal node to one of the switch terminal nodes. In the latter case, the other switch terminal node is connected to the power clock. Specifically, if the control terminal node is in a 1-hold (or 0-hold) state while the power clock ramps up, the other switch terminal node follows the ramp-up of the power clock (or remains at a 0 level).

As a variation of data transfer from the control terminal node to the switch terminal node, there is the so-called "decompute". Specifically, if the power clock ramps down while the control terminal node is in a 1-hold state, the other switch terminal node follows the ramp-down of the power clock, returning the charge accumulated at the node to the power clock.

The most fundamental form of decompute using the T-gate is explained with reference to Figure 2.4. Initially, when the input A maintains a hold at logic 1 after ramping up, and the power clock ϕ_i ramps up, the output out ramps up in response. This corresponds to the "computation" phase. Subsequently, while A remains at logic 1, if the power clock ϕ_i ramps down, the output out also ramps down accordingly. The charge accumulated in the equivalent capacitance of the out terminal returns to the power clock through the same T-gate used during the charging (computation) phase. This process constitutes the "decompute" for the node out.

However, the above operation assumes that the input signal A maintains a logic 1 hold for a sufficient duration. In the fully pipelined configuration discussed in this thesis, at the timing of decompute for the node out, the original signal A is no longer present, and decompute cannot proceed in this form. As described later, decompute (i.e., the return of the charge accumulated at the node to the power clock) is performed using a different T-gate from the one used during computation and a copy of the original signal A.

2.4.2 Functional gate

T-gates can be interconnected in series or parallel to construct AND or OR gates, respectively, as depicted in Figure 2.5. The input/output timing is the same as that of a buffer gate, and the output of the operation is produced one tick after the input is fed to the input terminal. The input/output timing is similar to that of a buffer gate, where the output signals AB_{i+1} and $(A+B)_{i+1}$ are computed during the ramp-up phase of ϕ_{i+1} and remain fixed during the high-hold phase, until they are decomputed by either the following buffer or by replicating the computation at the gate. In the case of an AND gate, if the A input is "1" and the B input is "0", the output is "0", but a "1" signal appears in the middle of the T-gates in series, which also needs to be decomputed.

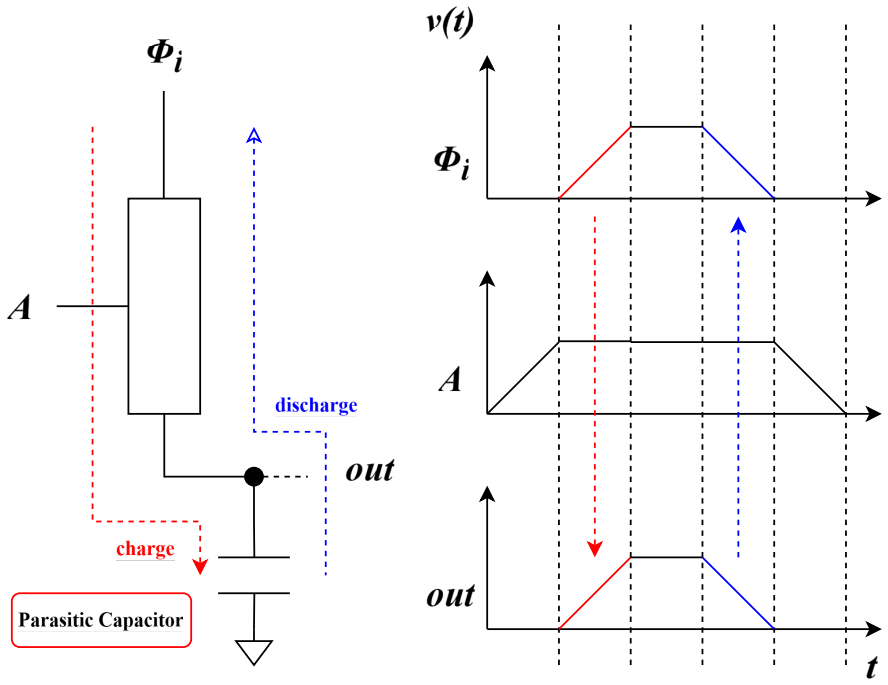


Figure 2.4. T-gate implementing an identity function with retractile decompute. (Left) Circuit using a single T-gate. (Right) Timing diagram showing computation during ϕ_i ramp-up and decompute during ramp-down.

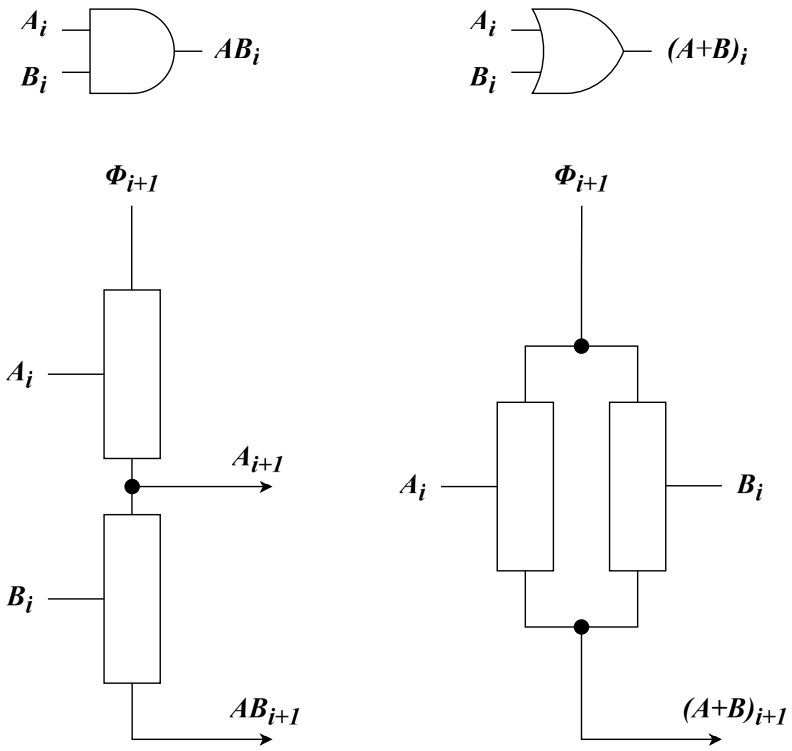


Figure 2.5. 2LAL AND and OR functional gates constructed using T-gates in series and parallel, respectively. Output is computed one pipeline stage after inputs.

2.4.3 Buffer gate

Figure 2.6 depicts a 2LAL buffer composed of two T-gates and shows the waveforms of power clocks and input/output signals. The initial output Out is 0 V, with all signals starting at 0 V.

1st section (iT to $(i+1)T$): With $Out = 0$, the T-gate controlled by ϕ_i is OFF. Input In is applied (ramping up or holding 0). Power-clock ϕ_{i+1} stays at 0 V, preventing premature charge transfer to Out.

2nd section ($(i+1)T$ to $(i+2)T$): If $In = 1$, the T-gate with ϕ_{i+1} turns ON, and Out ramps up to "1" following ϕ_{i+1}. If $In = 0$, the T-gate remains OFF and Out stays 0. Thus, Out matches In. This is the forward adiabatic computation phase, storing the logic value on the output capacitance.

3rd section ($(i+2)T$ to $(i+3)T$): Out holds its value to drive the next stage. When $Out(= In) = 1$, the T-gate with ϕ_i turns ON, and the ramp-down of ϕ_i pulls In to 0. When $Out = 0$, In remains 0. The charge on In is recovered to ϕ_i (decompute), enabling adiabatic energy reuse.

4th section ($(i+3)T$ to $(i+4)T$): Out is reset to 0 by the following stage (or remains 0), clearing the output reversibly to prepare for the next cycle.

These buffers synchronize pipeline stages and maintain computation integrity until decompute.

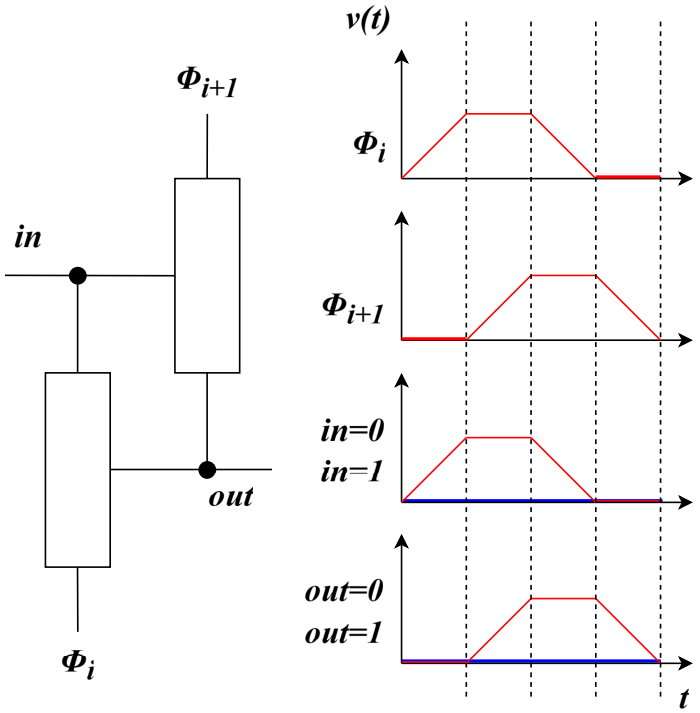


Figure 2.6. 2LAL buffer gate composed of two T-gates and corresponding power-clock waveforms (ϕ_i, ϕ_{i+1}). The buffer synchronizes pipeline stages and enables decompute of the input signal.

The detailed operation of decompute in the buffer circuit is explained with reference to Figure 2.7. Unlike the case of a single T-gate, in a fully pipelined operation, the input A used for the computation of the output node $out1$ is no longer present when $out1$ undergoes erasure computation. Specifically, during the forward evaluation phase of the first buffer stage, the input signal A is applied to charge (or not charge) the intermediate node $out1$ via the T-gate controlled by the power-clock ϕ_{i+1}. Once this computation is complete and $out1$ holds the logic value, the pipeline proceeds, and the original A-node enters its own decompute phase in a prior cycle. As a result, the signal A is adiabatically cleared—its charge, if any, is recovered back to the power-clock ϕ_i—leaving the A-node reset to 0 V well before $out1$ is ready to be erased.

At the moment when $out1$ must undergo decompute (i.e., when ϕ_i ramps down to recover charge from $out1$), the original input A is no longer physically available at the input terminal. Relying on A directly would break the reversibility of the pipeline, as the controlling signal would be missing precisely when needed to conditionally enable charge recovery. Instead, the system uses the output $out2$ of the second-stage buffer, which serves as an exact copy of the input A delayed by two clock cycles ($2T$). This delay arises naturally from the four-phase power-clock cycle of 2LAL: each buffer stage introduces a one-cycle latency in evaluation and another in propagation, resulting in $out2$ becoming valid exactly when $out1$ enters its decompute window.

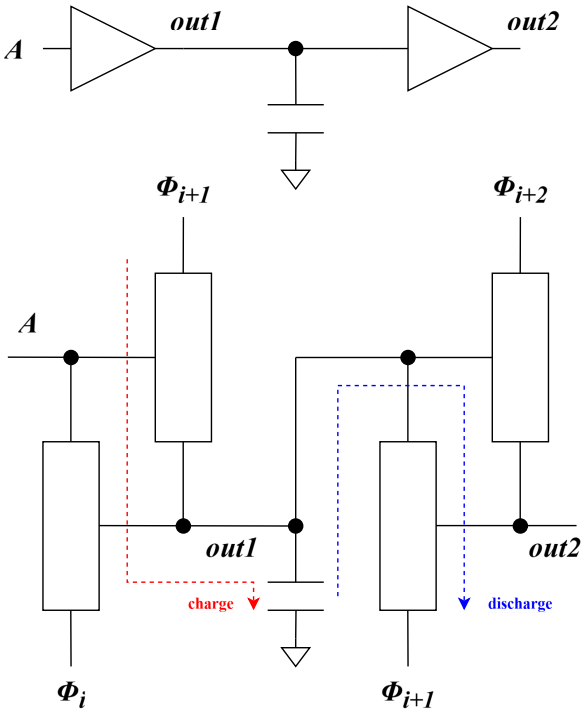
Thus, $out2$ acts as a temporally shifted replica of A, preserving the logical information required to control the T-gate that connects $out1$ back to ϕ_i. This T-gate has its gate terminal driven by $out2$, its source/drain path between $out1$ and the power-clock ϕ_i, and operates in the reverse direction during decompute: when ON, it allows stored charge on $out1$ to flow adiabatically back into ϕ_i as it ramps down.

Now consider the specific case when $A = 0$. During the forward computation of the first stage, $A = 0$ keeps the evaluation T-gate (controlled by ϕ_{i+1}) in the OFF state. No charge is transferred to $out1$, so it remains at 0 V throughout its evaluation and hold periods. This logic-0 state propagates faithfully through the second buffer stage, resulting in $out2 = 0$ after the two-cycle delay.

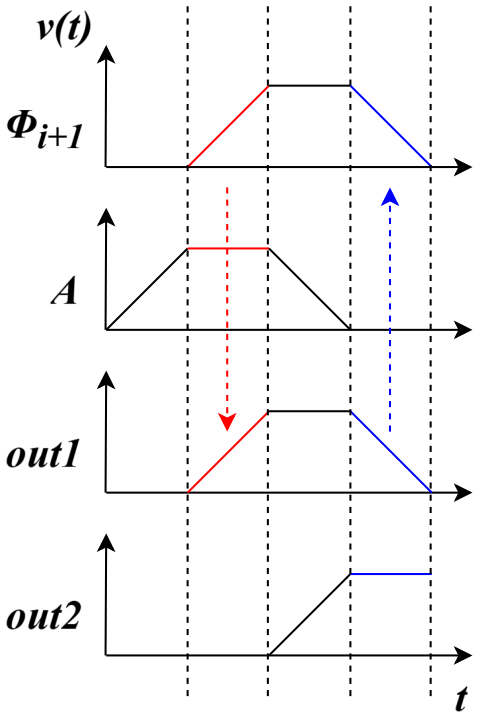
When the decompute phase for $out1$ begins—triggered by the ramp-down of ϕ_i—the recovery T-gate has its gate voltage set by $out2 = 0$. The gate-source voltage V_{GS} across this T-gate is therefore below the threshold voltage V_{th}, meaning no inversion channel forms in the transistor. The T-gate remains firmly in the OFF state, presenting an open circuit (ideally infinite impedance) between $out1$ and ϕ_i.

Consequently:

- No conductive path exists for charge to flow from $out1$ to the power-clock.
- Since $out1$ is already at 0 V (no charge was stored during evaluation), there is nothing to discharge.
- Even as ϕ_i ramps down to negative voltages (in some 2LAL implementations), the absence of a channel prevents any reverse leakage or unintended current.
- The voltage on $out1$ remains stably at 0 V with zero transient deviation—no ripple, no droop, no coupling-induced bounce.



(a) Circuit diagram (top: representation using logic symbol, bottom: representation using T-gates).



(b) Operation timing.

Figure 2.7. Decompute mechanism in a fully pipelined 2LAL buffer. A delayed copy of the input (A) is used to decompute the intermediate node ($out1$) two stages later.

2.4.4 Logic-level Design of 2LAL circuit

The next step of the 2LAL circuit design flow involves transforming the scheduled E-AIG into a fully pipelined netlist composed of 2LAL functional gates and 2LAL buffers, ensuring correct timing alignment, charge recovery, and adiabatic operation across all pipeline stages.

In the *forward compute part*, each AND node in the E-AIG is directly mapped to a 2LAL AND functional gate. The inputs to this gate are not taken directly from the preceding nodes but are instead sourced from designated taps along the corresponding buffer-chains at the appropriate pipeline stage. This ensures that all input signals arrive simultaneously during the ramp-up phase of the gate's driving power-clock ϕ_k. The output of the AND gate is then injected into its own buffer-chain, which propagates the computed value forward through successive pipeline stages while preserving it for potential use by downstream logic and for eventual decomputation.

Each buffer in the chain is driven by a distinct power-clock phase, staggered by one stage relative to its predecessor. This creates a wave-like propagation of valid data through the pipeline: a computed value emerges during the ramp-up of ϕ_k and is held stable during the high-hold phase of ϕ_{k+1}, making it available to any gate scheduled in stage $k + 1$. The length of each buffer-chain is determined by the maximum fanout delay and the decomputation schedule—specifically, the signal must persist until the last dependent node has evaluated *and* the corresponding decompute gate is ready to erase it.

In the *backward decompute part*, each node A_j^{-1} in the E-AIG represents the logical inverse operation required to safely discharge the charge stored on the output node of A_j. This is implemented using another 2LAL AND gate—referred to as the *decompute gate*—whose inputs are identical to those of the original forward gate A_j. These inputs are extracted from the same buffer-chain taps used in the forward path, but delayed by the number of pipeline stages required for the forward value to propagate to all dependent nodes.

The output of the decompute gate is connected directly to the same physical wire as the forward gate's output (or to the final buffer stage in the chain that still holds the forward-computed value). This shared output node enables a hand-off mechanism: during forward evaluation, the functional gate charges the node adiabatically; once computation is complete and all consumers have latched the value, the decompute gate activates its pull-down network during its own ramp-down phase, recovering the stored charge back to the power-clock.

This dual-gate structure—forward compute and backward decompute sharing the same output node—ensures that no residual charge remains on any internal node after a full evaluation cycle, achieving 100% charge recovery in the ideal case. The timing of decomputation is critical: the decompute gate must remain inactive (output high-impedance) while the forward value is still needed, and must activate *only after* the last dependent buffer has entered its high-hold phase. This is enforced by the pipeline scheduling and the staggered power-clock phases.

Figure 2.9 illustrates the complete 2LAL circuit realization derived from the E-AIG in Figure 2.8. Forward AND gates are shown in solid boxes, decompute AND gates in dashed boxes, and buffer-chains as sequences of triangular buffer symbols labeled with their driving power-clock phases ($\phi_i, \phi_{i+1}, \ldots$). Signal taps from buffer-chains are indicated with small circles, and the connection from decompute gate outputs back to the forward signal lines is highlighted to emphasize charge reuse and recovery paths.

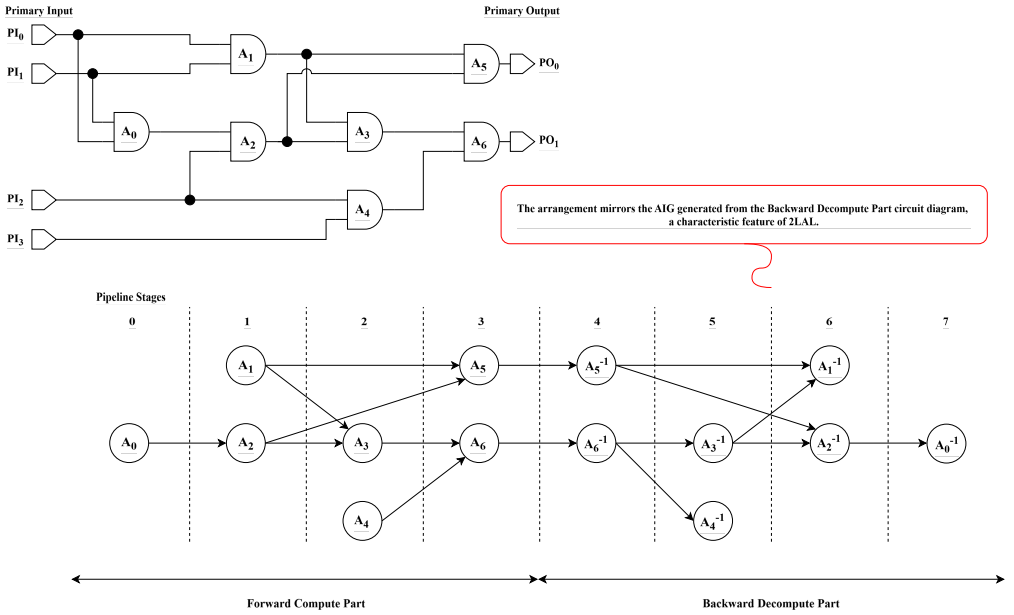


Figure 2.8. Extended And-Inverter Graph (E-AIG) for 2LAL design. The forward compute part (left) represents logic function; the backward decompute part (right) ensures charge recovery. Dotted lines indicate pipeline stage boundaries.

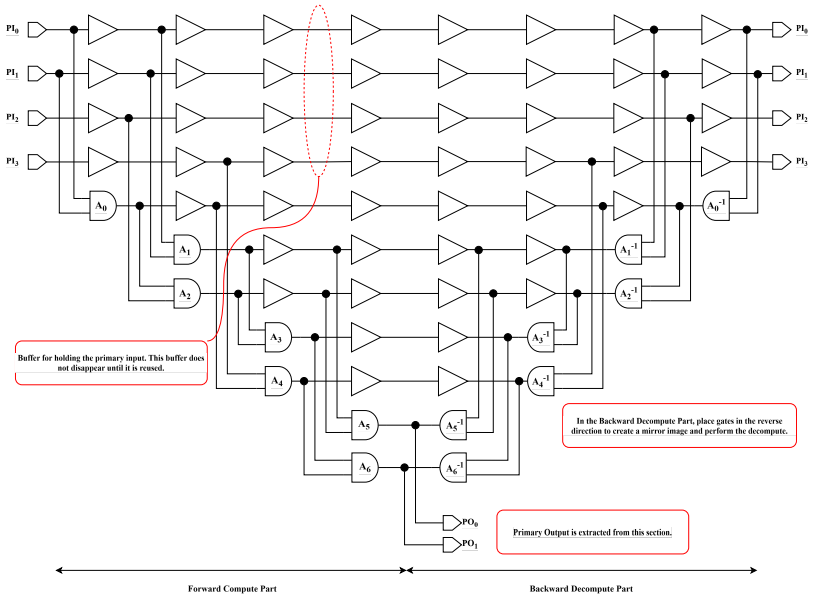


Figure 2.9. Complete 2LAL circuit implementation derived from the E-AIG in Figure 2.8. Forward compute uses AND gates; backward decompute reuses input signals via buffer chains to enable charge recovery.

The operation of this decompute mechanism is illustrated in more detail in Figure 2.10, which extracts a representative portion of the 2LAL circuit—specifically, a chain consisting of an AND functional gate and its associated buffer-chain—to demonstrate how decomputation proceeds when the gate output is no longer needed by downstream logic.

As shown in the timing diagram, we examine the signal evolution across three consecutive pipeline stages ϕ_{i+1}, ϕ_{i+2}, and ϕ_{i+3}, each driven by a distinct power-clock phase in the four-phase overlapping scheme. For clarity, we assume a logic-1 evaluation case where both inputs contribute to a high output, followed by its controlled erasure.

- ϕ_{i+1} **stage (forward computation and initial hold):** The input signals A_i and B_i, originating from prior pipeline stages, are already in their "1-hold" state—meaning they have completed ramp-up in the previous cycle and are now stably high during the high-hold phase of ϕ_i. At the start of ϕ_{i+1}, the power-clock begins its adiabatic ramp-up. The 2LAL AND functional gate, sensing both inputs high, activates its pull-up network, charging the output node C from 0 to V_{dd} in synchrony with ϕ_{i+1}. The logical computation $C = A_i \wedge B_i = 1$ thus occurs *during* the ramp-up, minimizing non-adiabatic losses.

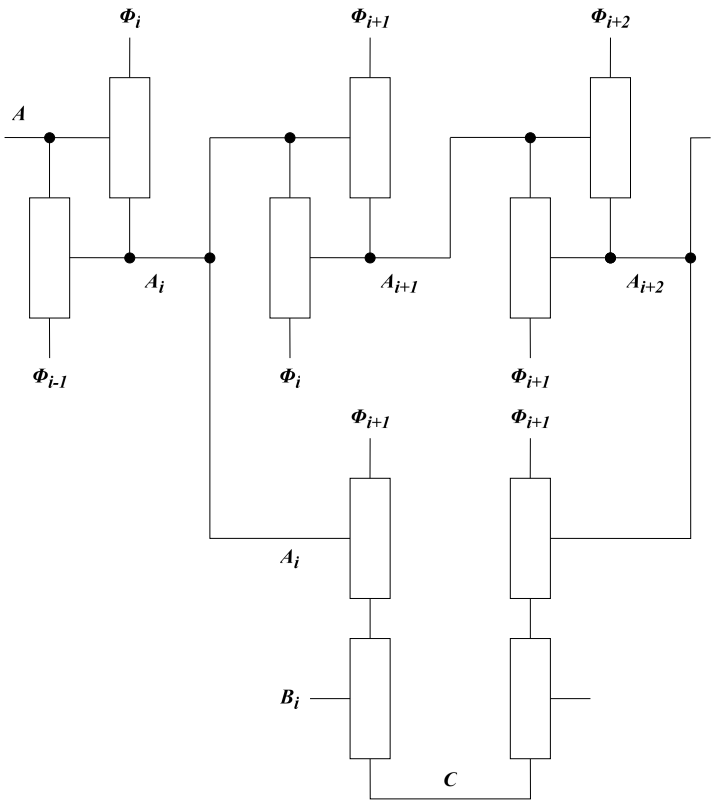
  Simultaneously, the first buffer in the output buffer-chain (driven by ϕ_{i+1}) propagates the value of A_i forward, producing A_{i+1}. This node also ramps up in lockstep with ϕ_{i+1}, ensuring that the buffered input remains valid for potential downstream consumers in the next stage. By the end of the ϕ_{i+1} ramp-up, both C and A_{i+1} reach full voltage and enter the "1-hold" phase as ϕ_{i+1} plateaus, ready to be sampled during the subsequent stage.

- ϕ_{i+2} **stage (value propagation and downstream availability):** With A_{i+1} now firmly in "1-hold", the next buffer in the chain—powered by ϕ_{i+2}—begins its ramp-up phase. This drives A_{i+2} from low to high, extending the lifetime of the input signal A_i by another pipeline cycle. The computed output C remains in "1-hold" throughout this stage, sustained by the hold phase of ϕ_{i+1}, and is now fully available to any functional gates scheduled in stage ϕ_{i+2} or beyond. This staggered buffering ensures that fanout and logic depth do not violate timing constraints, as each dependent gate receives inputs at precisely the correct phase.

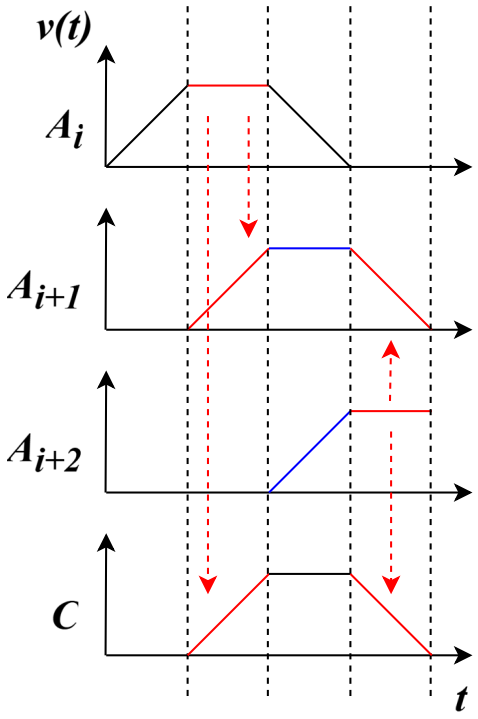
- ϕ_{i+3} **stage (decomputation and charge recovery):** Once A_{i+2} completes its ramp-up and settles into "1-hold", the decompute AND gate—located in the backward part of the E-AIG and also implemented as a 2LAL AND gate—receives valid high inputs from the delayed buffer taps (still sourcing the original A_i and B_i values). Critically, this gate is *not* active during forward evaluation; its output remains high-impedance until its own power-clock ϕ_{i+3} begins ramping *down*.

  During the ramp-down phase of ϕ_{i+3}, with both inputs high, the decompute gate activates its pull-down path. Since the forward gate is now in its idle/hold-off state (its power-clock ϕ_{i+1} has long since completed its cycle), the output node C is driven low *adiabatically* in reverse synchrony with the falling power-clock. This discharges the stored charge on C back into the power supply, achieving near-ideal energy recovery. The node C returns to "0-hold", fully decomputed and ready for the next evaluation cycle.

- **Idle/low-input case (energy-efficient zero propagation):** When either A_i or B_i is "0-hold" throughout the ϕ_{i+1} to ϕ_{i+3} interval (i.e., the logical AND evaluates to 0), the forward AND gate never activates its pull-up path. The output node C remains at ground, and no charge is injected.



(a) Circuit diagram.



(b) Operation timing.

Figure 2.10. Timing diagram of decompute in a 2LAL buffer chain. Signal A_{i+2} (delayed copy) triggers ramp-down (decompute) of node C during ϕ_{i+3}.

The above explanation did not cover the handling of negation. One approach to integrate negations into four-phase fully-pipeline 2LAL is to utilize quad-rail signals instead of dual-rail signals. Specifically, each signal X is encoded using a pair of dual-rail signals, such that

$$X = 0 \quad \leftrightarrow \quad (X_n, X_p) = (0, 1), (\overline{X}_n, \overline{X}_p) = (1, 0) \tag{2.15}$$
$$X = 1 \quad \leftrightarrow \quad (X_n, X_p) = (1, 0), (\overline{X}_n, \overline{X}_p) = (0, 1) \tag{2.16}$$

and each of (X_n, X_p) and $(\overline{X}_n, \overline{X}_p)$ is implemented with dual-rail 2LAL circuit. Figure 2.11 shows a conceptual gate-level schematic of the quad-rail implementation. As a result of the quad-rail encoding, the circuit size becomes almost double compared with a circuit having neglected negations.

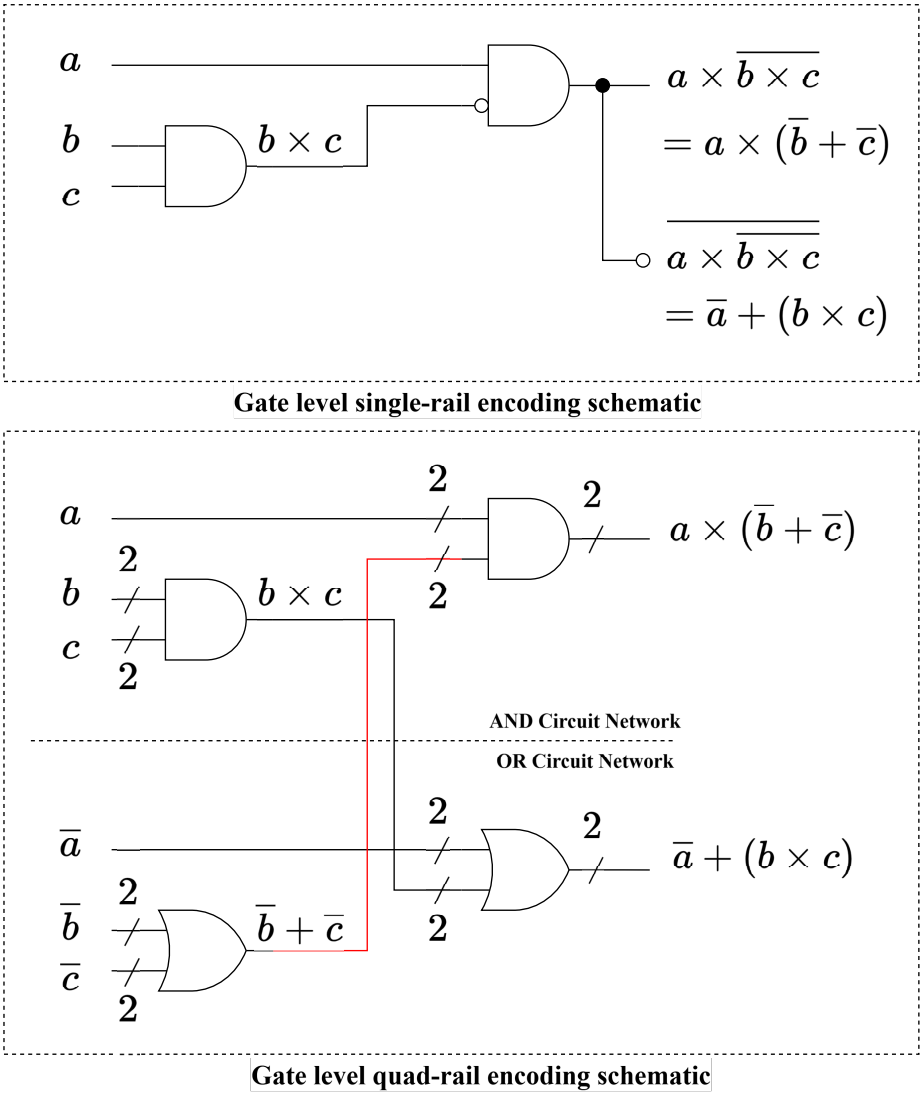


Figure 2.11. Gate-level schematic comparison. (a) Single-rail encoding (simplified, no inversion). (b) Quad-rail encoding to support inversion in 2LAL using dual-rail pairs for X_n, X_p, $\overline{X}_n$, $\overline{X}_p$.

2.4.5 Relationship with Pass-Transistor Logic

Two-Level Adiabatic Logic (2LAL) and conventional pass-transistor logic (PTL) share a common building block: transmission gates (T-gates) composed of parallel NMOS and PMOS transistors [43]. This structural similarity often leads to questions regarding their operational and efficiency differences.

While both paradigms employ T-gates for signal propagation, the fundamental distinction lies in timing control and power supply management:

- **Pass-Transistor Logic (PTL)**: Gates are controlled by static logic signals applied to transistor gates. Charging/discharging of nodal capacitances occurs directly from a constant DC supply (V_{DD}) or ground, resulting in irreversible CV^2 energy dissipation on every transition. PTL reduces transistor count compared to static CMOS but offers limited power savings and suffers from voltage degradation ($V_{DD} - V_{th}$) without full-swing restoration.

- **Two-Level Adiabatic Logic (2LAL)**: Transmission gates are controlled by the power-clock itself (or its complement), synchronized with gradual voltage ramps. Charging occurs adiabatically from a time-varying supply, and charge is recovered during ramp-down phases. Strict switching rules (R1: no turn-on with non-zero voltage drop; R2: no turn-off with non-zero current) ensure near-lossless energy transfer.

The power efficiency implications are profound, particularly in low-to-moderate frequency regimes:

- PTL achieves modest savings through reduced transistor count and partial elimination of short-circuit power, but remains fundamentally irreversible.

- 2LAL, by adhering to adiabatic principles, can theoretically approach zero dissipation (limited only by leakage and non-idealities). As validated in Chapter 6, practical 2LAL gates exhibit 2–3 orders of magnitude lower power than CMOS equivalents at optimal frequencies, with savings increasing as frequency decreases—opposite to conventional logic.

Thus, while PTL and 2LAL appear structurally similar at the transistor level, the adiabatic synchronization with a resonant power-clock transforms 2LAL into a qualitatively superior paradigm for ultra-low-energy applications, where energy per operation is the dominant metric.

Chapter 3

Design Method Based on Early Decompute Scheduling

3.1 Introduction

Fully pipelined Two-Level Adiabatic Logic (2LAL) circuits achieve ultra-low power consumption through energy recycling, making them promising for IoT and edge applications. However, extensive buffer usage for pipeline synchronization and signal lifetime management significantly increases circuit area, posing a key challenge to practicality. The primary objective of this chapter is to minimize buffer counts while preserving low power efficiency.

One effective technique for buffer reduction is early decompute, where a gate's output is decomputed earlier than its original decompute stage and recomputed just before it is needed by successor decompute gates. This shortens the signal lifetime, reducing the number of required buffers. However, selecting which gate outputs to apply early decompute to—while satisfying data dependencies and pipeline constraints—remains a complex optimization problem.

This chapter proposes an Integer Linear Programming (ILP)-based method to systematically determine optimal early decompute and recompute timings using an Extended And-Inverter Graph (E-AIG). Evaluated on the ISCAS-85 benchmark suite, the approach significantly reduces buffer counts and circuit area compared to the existing heuristic method.

3.2 Early Decompute

This section introduces the early decompute technique, proposed in [44], for optimizing Two-Level Adiabatic Logic (2LAL) circuits. The method reduces buffer usage in fully pipelined 2LAL circuits, enhancing efficiency for low-power applications such as Internet of Things (IoT) and edge devices. We describe the principles and application of early decompute to minimize the number of registers needed for pipeline synchronization, improving circuit performance.

The design of fully pipelined Two-Level Adiabatic Logic (2LAL) circuits requires a substantial number of buffers for pipeline synchronization and data preservation, which are essential for the decompute of generated signals. This extensive buffer usage significantly increases the circuit area, posing a key challenge to circuit efficiency.

The early decompute technique, proposed in [44], addresses this issue by reducing the number of buffers required in 2LAL circuits. In the conventional configuration, a logic gate's output is preserved by a buffer chain until its role is complete, which includes serving as input to successor gates and their corresponding decompute gates, before being decomputed. For example, as shown in Figure 2.9, the output of gate A_0 is used as input to gates A_2, as well as to the decompute gates A_2^{-1}. The early decompute technique allows

the output of a gate, such as A_0, to be decomputed earlier than its necessary end time. As illustrated in Figure 3.1, early decompute is applied to the output of gate A_0 in the same pipeline stage as the computation of successor gates A_2, enabling correct execution of both the computations and the decompute of A_0's output. After early decompute, recompute of A_0's output is required to support the decompute of A_2. Subsequently, the recomputed output of A_0 is decomputed by a second decompute gate A_0^{-1}. This approach reduces the number of buffers between the early decompute gate A_0^{-1} and the recompute gate A_0, as fewer pipeline stages are spanned.

To maximize the benefits of this technique, optimizing the selection of gates for early decompute is critical.

3.3 Algorithm of Existing Method

To maximize the benefits of the early decompute technique, optimizing the selection of gates for early decompute is critical, depending on the design objectives, such as minimizing the circuit area.

In [44], the authors proposed a heuristic algorithm to select nodes for early decompute in Two-Level Adiabatic Logic (2LAL) circuits. The approach, applied to a circuit graph $G = (V, E)$ represented as an And-Inverter Graph (AIG), selects all nodes $v \in V$ whose pipeline stage depth is a multiple of an integer constant $k \in \mathbb{Z}$. This heuristic simplifies the scheduling of early decompute operations but may result in suboptimal buffer reduction compared to more sophisticated methods. The pseudocode, named Algorithm 1, illustrates one possible implementation of this concept.

The primary baseline for comparison in this thesis is the heuristic early decompute scheduling algorithm proposed by Zulehner et al. [44], which represents the state-of-the-art heuristic approach for buffer overhead reduction in fully pipelined adiabatic circuits at the time of this research.

The core design intent of this heuristic is to achieve rapid, feasible decompute placement with minimal computational overhead, making it suitable for integration into practical design flows. The algorithm operates as follows:

1. Perform a topological traversal of the circuit graph (typically an And-Inverter Graph or similar representation).

2. For each node v, identify the successor node with the maximum pipeline depth $D_{\max}$.

Algorithm 1 Heuristic Algorithm for Early Decompute Node Selection

Input: Circuit graph V, integer constant k

```
for v ∈ V do
    SEARCH(v)
end for

function SEARCH(v_s)
    I_max ← Index of succeeding node with maximum depth of v_s
    D_max ← Depth of I_max
    if D_max mod k = 0 then
        Perform early decompute of v_s at stage D_max
    end if
end function
```

3. If $D_{\max} \mod k = 0$ for a user-defined constant k (typically chosen empirically to balance aggressiveness and feasibility), apply early decompute to v at stage $D_{\max}$.

This depth-modulo selection strategy aims to distribute early decompute operations uniformly across pipeline stages, ensuring that signal lifetimes are periodically shortened without requiring global dependency analysis. The parameter k provides a simple knob for trading off aggressiveness (smaller k) against risk of timing violations (larger k).

The heuristic is justified by its low complexity—linear in the number of nodes—and guaranteed feasibility under standard pipeline scheduling assumptions, as decompute placement is always aligned with successor depths. It served as a pioneering proof-of-concept for early decompute in adiabatic logic, demonstrating significant buffer reductions over unoptimized baselines in early benchmarks.

However, the approach is inherently greedy and local: decisions are made sequentially without lookahead or global optimization, often resulting in suboptimal selections where high-value nodes (those enabling large buffer savings) are overlooked due to rigid modulo conditioning. Furthermore, the fixed interval strategy cannot adapt to circuit-specific dependency structures, fanout patterns, or varying signal lifetimes, frequently leaving substantial savings on the table—particularly in irregular or deeply pipelined designs such as multipliers.

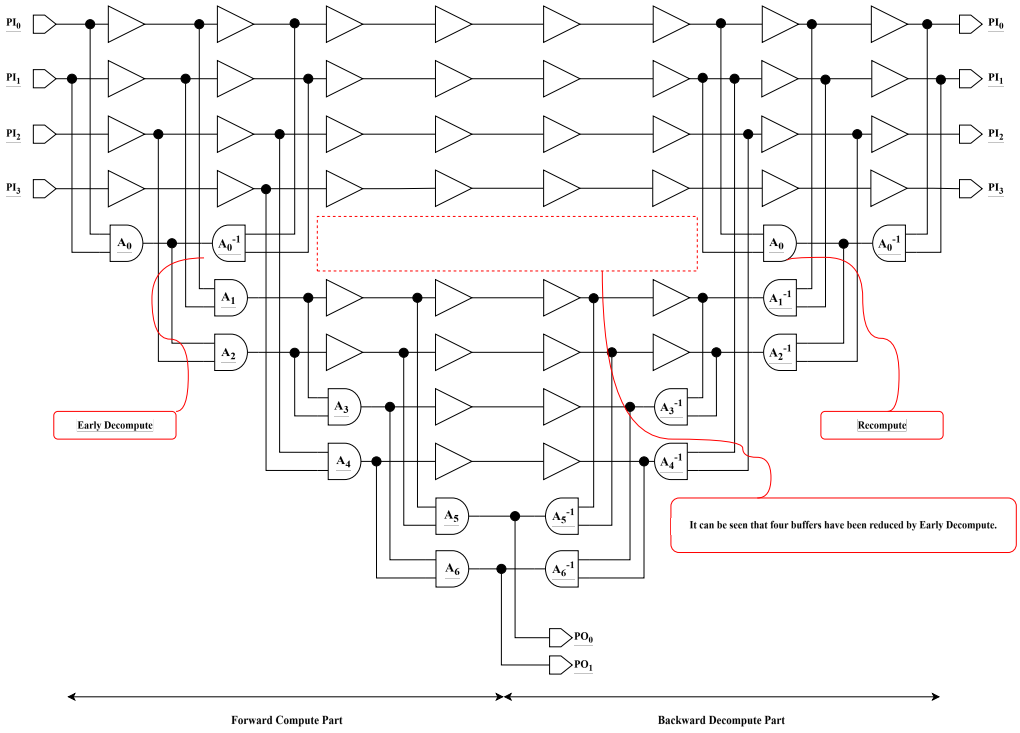


Figure 3.1. Early decompute method using the same type of gate and the same input signal

3.4 Proposed Method

3.4.1 Assumptions and Constraints

The proposed method assumes a fully pipelined 2LAL circuit structure with dual-rail encoding, T-gates, and four-phase power clocks, leveraging early decompute to reduce buffer counts. Key assumptions include:

- The pipeline schedule $\sigma(v)$ is fixed by logic synthesis (e.g., Yosys [61]), with $\sigma(j) = \text{depth}(j)$ and $\sigma(j^{-1}) = T_{\max} - \text{depth}(j) + 1$ for each node $j \in O_c$.
- Primary inputs I cannot be early decomputed due to their non-recomputability.
- Each node $j \in O_c$ is restricted to at most one early decompute.
- Data dependencies and resource contention (e.g., conflicts in computation and decompute within the same stage) impose constraints.

In addition to the above assumptions, a key restriction is imposed: each forward compute node $j \in O_c$ is permitted at most one early decompute operation. This deliberate limitation is essential for maintaining tractable ILP complexity and ensuring practical solvability.

Allowing multiple early decomputes per node would introduce recursive dependencies: a recomputed signal could serve as input to another early decompute downstream, potentially creating cyclic or exponentially branching dependency chains in the Extended AIG. Formulating constraints to correctly capture such nested timing relationships would require additional binary variables and quadratic or higher-order constraints to track "generations" of recomputed signals, leading to an exponential explosion in problem size. For instance, even modest relaxation to two decomputes per node could multiply the number of timing variables and constraints by orders of magnitude, rendering medium-to-large benchmarks computationally intractable within reasonable time limits.

The single-decompute restriction dramatically simplifies the formulation:

- Timing variables s'_j and e'_j are uniquely defined per node.
- Dependency constraints (e.g., Constraints 5.7–5.10) remain linear and local.
- The overall solution space remains $O((2T_{\max}^2)^{|O_c|})$, preserving solvability for circuits with thousands of gates.

Empirical results validate the effectiveness of this conservative approach: substantial buffer reductions are achieved (up to 40% over heuristics in medium-scale circuits) while capturing the dominant savings opportunities. The stable set formulation in Chapter 4 implicitly enforces similar locality, further confirming that a single, well-placed decompute per node suffices for high-quality solutions.

Relaxing this restriction represents a theoretically intriguing but computationally prohibitive extension. Future work could explore hierarchical or iterative methods—e.g., solving for primary decomputes first, then selectively adding secondary ones in critical subgraphs—to gradually approach the full potential without sacrificing scalability.

3.4.2 Problem Description

The circuit is represented as an Extended And-Inverter Graph (E-AIG) $G = (V, E)$, where V comprises primary inputs I, forward compute gate nodes O_c, and backward decompute nodes O_d. Each node $v \in V$ is assigned a pipeline stage number $\sigma(v)$, predefined by logic synthesis tools. The maximum stage number is defined as:

$$T_{\max} = \max\{\sigma(o) \mid o \in O_c \cup O_d\} \tag{3.1}$$

The problem is formulated as follows:

- **Input**: E-AIG $G = (V, E)$ and fixed pipeline schedule $\sigma : V \to \mathbb{Z}$.
- **Output**: Early decompute timing s'_j, recompute timing e'_j, and early decompute flag $p_j \in \{0, 1\}$ ($p_j = 0$ for application) for each node $j \in O_c$.
- **Conditions and Constraints**:
  - Data dependencies: For $(j, k) \in E$, j's early decompute must follow k's computation ($\sigma(k) \leq s'_j$), and j's recompute must precede k's decompute ($e'_j \leq \sigma(k^{-1})$).
  - Control constraints: Maintain pipeline synchronization, with $s'_j \geq \sigma(j)$ and $e'_j \leq \sigma(j^{-1})$.
  - Timing constraints: At least one stage gap between early decompute and recompute ($s'_j \leq e'_j - 1$).
  - Primary inputs I cannot be early decomputed. Each node $j \in O_c$ is early decomputed at most once.
- **Objective**: Minimize the total number of buffers and gates to reduce E_{area}.

3.5 ILP Formulation

The proposed method employs Integer Linear Programming (ILP) to optimize the timing of early decompute (s'_j) and recompute (e'_j) for each node $j \in O_c$ under a fixed pipeline schedule $\sigma(v)$, minimizing buffer counts. As shown in Figure 3.2, early decompute replaces a buffer at stage s'_j with a decompute gate, places a recompute gate at e'_j, and eliminates buffers from $s'_j + 1$ to $e'_j - 1$. This approach provides precise, dependency-aware optimization compared to heuristic methods, maximizing buffer reduction.

The ILP formulation defines variables, an objective function, and constraints as follows:

- **Variables**:
  - $s'_j \in \mathbb{Z}$ ($j \in O_c$): Stage at which node j is early decomputed.
  - $e'_j \in \mathbb{Z}$ ($j \in O_c$): Stage at which node j is recomputed before its final decompute.
  - $p_j \in \{0, 1\}$ ($j \in O_c$): Binary flag — $p_j = 0$ means early decompute is applied, $p_j = 1$ means not applied.
  - Bounds: $1 \leq s'_j, e'_j \leq T_{\max}$, $p_j \in \{0, 1\}$.
- **Objective Function**: The goal is to minimize the total number of buffers (and implicitly gates) across all nodes. Without early decompute ($p_j = 1$), a buffer chain spans from stage $\sigma(j)$ to $\sigma(j^{-1})$, requiring $\sigma(j^{-1}) - \sigma(j) + 1$ buffers. When early decompute is applied ($p_j = 0$), buffers are eliminated between $s'_j + 1$ and $e'_j - 1$, saving $e'_j - s'_j - 1$ buffers per node. The objective function is:

$$\text{minimize} \quad \sum_{j \in O_c} \left[\sigma(j^{-1}) - \sigma(j) + 1 - (e'_j - s'_j - 1)\right] \tag{3.2}$$

// Only active when $p_j = 0$; otherwise, the subtraction term is zero.

- **Constraints**:

  1. **Early decompute before recompute**:

$$s'_j \leq e'_j - 1, \quad \forall j \in O_c \tag{3.3}$$

  Ensures at least one pipeline stage gap between early decompute and recompute to allow signal reuse.

  2. **Deactivate timing when no early decompute**:

$$e'_j - s'_j - 1 \leq T_{\max} \cdot (1 - p_j), \quad \forall j \in O_c \tag{3.4}$$

  When $p_j = 1$, forces $e'_j = s'_j + 1$; when $p_j = 0$, the constraint is relaxed.

  3. **Early decompute after forward compute**:

$$\sigma(j) \leq s'_j + p_j \cdot T_{\max}, \quad \forall j \in O_c \tag{3.5}$$

  Prevents early decompute before the original computation; when $p_j = 1$.

  4. **Recompute before final decompute**:

$$e'_j \leq \sigma(j^{-1}) + p_j \cdot T_{\max}, \quad \forall j \in O_c \tag{3.6}$$

  Ensures recomputed value is available before the original decompute gate; relaxed when not applied.

  5. **Input availability for early decompute**:

$$\sigma(k) \leq s'_j + T_{\max} \cdot p_j, \quad \forall (j,k) \in E \tag{3.7}$$

  Node k (input to j) must be computed before j is early decomputed; ignored if $p_j = 1$.

  6. **Recomputed value for successor decompute**:

$$e'_j \leq \sigma(k^{-1}) + T_{\max} \cdot p_j, \quad \forall (j,k) \in E \tag{3.8}$$

  recompute of j must finish before k's decompute gate (which needs j's value); if no early decompute.

  7. **Ordering of early decompute in dependency chain**:

$$s'_k \leq s'_j + T_{\max} \cdot (p_j + p_k), \quad \forall (j,k) \in E \tag{3.9}$$

  If both k and j use early decompute, k must be decomputed before j; relaxed if either is early decompute is not applied.

  8. **Ordering of recompute in dependency chain**:

$$e'_j \leq e'_k + T_{\max} \cdot (p_j + p_k), \quad \forall (j,k) \in E \tag{3.10}$$

  recompute of j must precede recompute of k if both are active; ensures correct signal flow.

The method assumes a fixed pipeline schedule $\sigma(v)$, enforcing strict adherence to dependency and resource contention constraints. Post-development analysis reveals a solution space of $O((2T_{\max}^2)^{|O_c|})$, posing computational challenges for large-scale circuits.

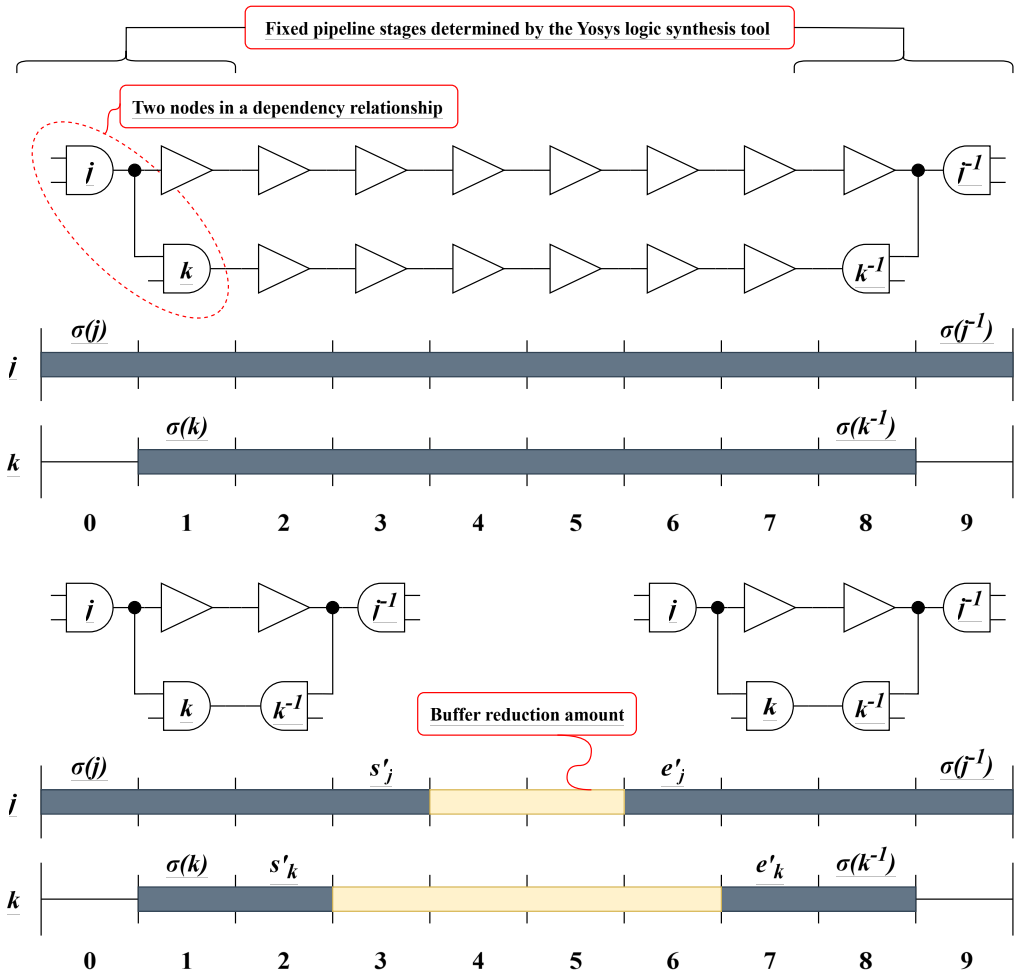


Figure 3.2. Representation of circuit configuration based on scheduling.(top: Before applying early decompute, bottom: After applying early decompute)

3.6 Experiment

3.6.1 Experiment Setup

To validate the circuit area reduction efficacy of the proposed method, a simulation-based evaluation was conducted. The optimization problem was formulated as a Integer Linear Programming (ILP) problem and solved using open-source solvers: Cbc (Coin-or Branch-and-Cut) [46] and HiGHS. Cbc, implemented in C++, employs branch-and-cut techniques to efficiently explore the solution space of ILP problems. HiGHS, also implemented in C++, is a high-performance solver known for its robust simplex and interior-point methods, enabling efficient handling of large-scale optimization problems. Computations were performed on an Intel Core i7-9700 processor at 3.0 GHz, with solvers interfaced through the PuLP library in Python scripts. Each optimization task was allocated a maximum runtime of 60 seconds for the standard approach, reflecting the need for rapid optimization in practical design scenarios. The evaluation utilized eleven benchmark circuits from the ISCAS-85 suite [47], a standard set of combinational logic circuits for testing logic synthesis and optimization techniques (specifications in Table 3.1). For each circuit, a Verilog-HDL model was generated to describe its functionality, and And-Inverter Graphs (AIG) and OR-Inverter Graphs (OIG) were produced using the ABC logic synthesis tool [48]. Pipeline scheduling for each node was automatically optimized by ABC to enhance the proposed method's timing and structure. The proposed method aims to reduce circuit area by optimizing logic gate implementation in 2LAL circuits compared to traditional CMOS circuits. In CMOS, an AND gate requires 6 MOSFETs and an inverter 2 MOSFETs, while in 2LAL, AND and OR gates each require 8 MOSFETs (Figure 3.3), necessitating careful optimization to minimize total MOSFETs. The effectiveness was evaluated using the metric:

$$E_{\text{area}} = \frac{M_{\text{2LAL}}}{M_{\text{CMOS}}}, \tag{3.11}$$

where M_{CMOS} is the total number of MOSFETs in the CMOS implementation, and M_{2LAL} is that in the 2LAL implementation. A smaller E_{area} indicates a more efficient 2LAL implementation, reflecting reduced buffers and circuit area. MOSFET counts were calculated from ABC-generated gate-level netlists, accounting for transistor counts of AND, OR, and inverter gates in CMOS and 2LAL designs. The experimental setup followed a systematic workflow to optimize 2LAL circuits, as illustrated in Figure 3.4. The workflow begins with a Verilog-HDL file describing the circuit's functionality, processed by the Yosys tool [61] integrated with the ABC framework to generate OR-Inverter Graphs (OIG) and And-Inverter Graphs (AIG). A Python script extends OIG and AIG into Extended OIG (E-OIG) and Extended AIG (E-AIG), incorporating dependency and pipeline scheduling information. A preprocessing step uses depth-first search (DFS, complexity $O(|E| + |V|)$) to remove unused nodes from E-OIG and E-AIG, enhancing computational efficiency. Core optimization is performed by ILP solvers (Cbc or HiGHS), optimizing pipeline schedules and enabling early decompute to minimize circuit area and buffer count. The output is an optimized 2LAL circuit netlist with reduced gates and buffers.

To enhance the performance of Proposed Method 1 (Prop. Meth. 1, ILP-based scheduling), an experiment was designed to compare three approaches: the standard approach, extended runtime, and parallel execution with variable order randomization. These approaches address the computational complexity and convergence challenges of Prop. Meth. 1, particularly for large-scale circuits like c6288, and compete with the sub-second runtimes of Proposed Method 2 (Prop. Meth. 2) and the existing heuristic method (Algorithm 1). The three approaches are outlined as follows:

1. **Standard Approach (Baseline)**:

    - Prop. Meth. 1 is implemented using Cbc (Prop. M 1C) and HiGHS (Prop. M 1H) solvers with a maximum runtime of 60 seconds.
    - Applied to all ISCAS-85 benchmark circuits to evaluate rapid optimization. However, for large-scale circuits like c6288 (23794 variables, 39998 constraints, 125345 non-zero elements), both solvers struggle to converge within 60 seconds due to dense constraint matrices (e.g., 60.094s for Prop. M 1C, 60.134s for Prop. M 1H, Figure 5.3).

2. **Extended Runtime**:

    - The runtime limit of Prop. Meth. 1 is extended from 60 seconds to 3600 seconds, allowing deeper exploration of the ILP solution space.
    - Targets improved convergence for large-scale circuits (e.g., c6288) and high-gate-count circuits (e.g., c5315, with 19533 variables, 21006 constraints, 1776 AND gates, 37 pipeline stages), enabling Cbc's branch-and-cut and HiGHS's simplex and interior-point methods to explore more branches, achieving better E_{area} reductions (e.g., c880: 43.7 for Prop. M 1H, 39.5% better than Algorithm 1's 72.2).

3. **Parallel Execution with Variable Order Randomization**:

    - Prop. Meth. 1 is executed in eight parallel processes using the Cbc and HiGHS solvers. These processes are launched simultaneously on an 8-core CPU, with each instance configured with a distinct randomized variable order for the ILP problem and a per-process runtime limit of 3600 seconds.
    - No inter-process communication or shared incumbents are used; each process runs independently to completion or timeout.
    - Upon completion, the solution yielding the lowest E_{area} among the eight runs is adopted as the final result. Randomization of variable ordering—implemented by supplying a unique random permutation to each solver instance via PuLP parameters—diversifies the branching strategy of the branch-and-bound search. This multi-start portfolio approach increases the chance of exploring different regions of the solution space, thereby reducing the risk of all processes converging to the same suboptimal local optimum. It is particularly beneficial for problems with vast search spaces ($O((2T_{\max}^2)^{|O_c|})$) and complex scheduling dependencies, such as `c6288` with its 120 pipeline stages.
    - However, results are mixed: randomization occasionally yields marginal improvements (e.g., `c2670`: 54.1 for Prop. M 1C Para, 1.1% better than the single-run 54.7), but more frequently leads to degradation (e.g., `c1355`: 66.1 for Prop. M 1H Para, 43.4% worse than the single-run 46.1). These outcomes highlight both the potential and the limitations of blind randomization in highly structured scheduling problems.

The experiment assesses the trade-offs between computational complexity, convergence stability, and area optimization across the ISCAS-85 benchmark circuits, ranging from small-scale c17 (110 variables, 60 constraints) to large-scale c6288. The outcomes provide insights into the applicability of Prop. Meth. 1 for 2LAL circuit optimization in practical design scenarios.

Table 3.1. Specifications of ISCAS-85 benchmark circuits used for evaluation, including function, number of primary inputs (PI), primary outputs (PO), AND gates, and pipeline stages generated by ABC.

Benchmark	Function	PI	PO	AND Gate	Pipe. Stages
c17	–	5	2	6	3
c432	27-ch. interrupt ctrl.	36	7	208	26
c499	32-bit SEC	41	32	398	19
c880	8-bit ALU	60	26	325	25
c1355	32-bit SEC	41	32	502	25
c1908	16-bit SEC/DED	33	25	341	27
c2670	12-bit ALU & ctrl.	157	52	716	20
c3540	8-bit ALU	50	22	1024	41
c5315	9-bit ALU	178	123	1776	37
c6288	16×16 multiplier	32	32	2337	120
c7552	32-bit adder/comp.	207	208	1469	26

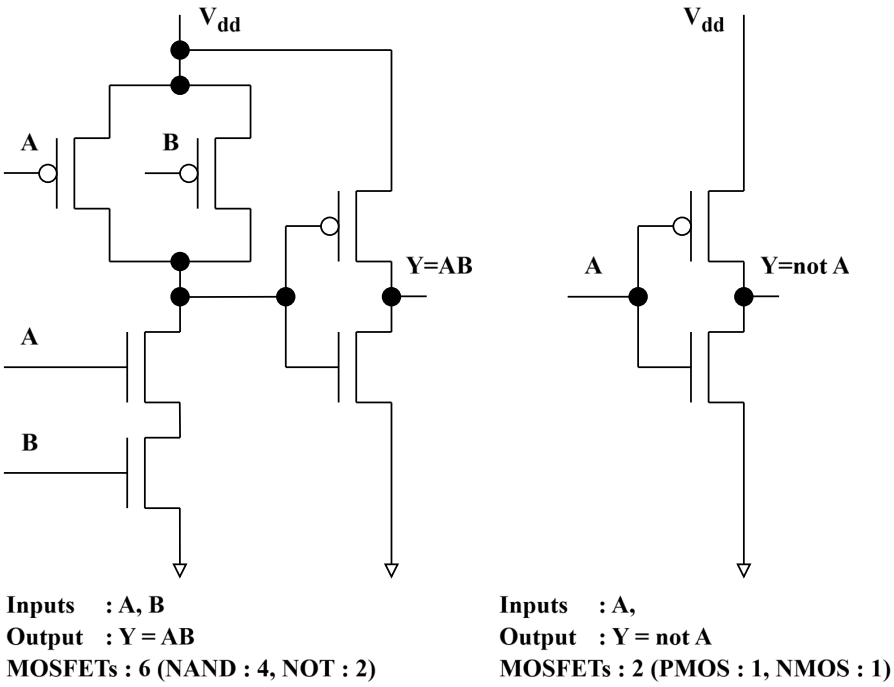


Figure 3.3. Transistor-level structures of CMOS circuits, shown side by side. Left: 2-input AND gate, implemented as a NAND gate (4 MOSFETs) followed by a NOT gate (2 MOSFETs), using 6 MOSFETs total. Right: Inverter (NOT gate), using 2 MOSFETs (1 PMOS for pull-up, 1 NMOS for pull-down).

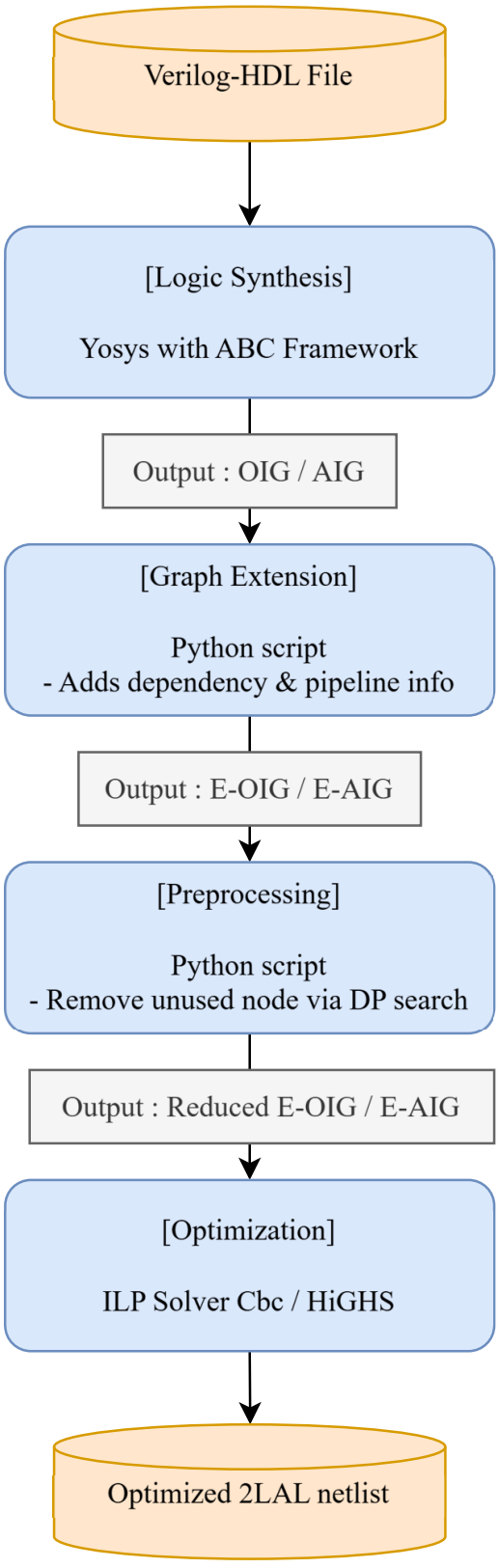


Figure 3.4. Workflow for 2LAL circuit optimization using Proposed Method 1, from Verilog-HDL input to optimized netlist via Yosys [61], ABC [48], and ILP solvers.

3.6.2 Definition of the Area Expansion Ratio and Its Fairness

The primary metric used in this thesis for quantifying buffer overhead reduction is the area expansion ratio

$$E_{\text{area}} = \frac{M_{\text{2LAL}}}{M_{\text{CMOS}}}, \tag{3.12}$$

where M_{2LAL} and M_{CMOS} represent the total transistor counts (or equivalent gate counts) in the 2LAL and CMOS implementations, respectively.

Both implementations are derived from the same And-Inverter Graph (AIG) produced by standard logic synthesis tools [48]. For CMOS, each AIG AND node is mapped to a 2-input NAND gate (4 transistors) followed by an inverter (2 transistors), yielding 6 transistors per AND node, with inverters counted as 2 transistors. For 2LAL, functional AND/OR gates require 8 transmission-gate transistors, while pipeline buffers require a comparable count, consistent with dual-rail transmission-gate topology [44].

The reviewer correctly notes that commercial CMOS flows employ highly optimized standard-cell libraries with complex cells (NAND, NOR, AOI, etc.), potentially achieving lower transistor counts than a strict AIG-to-NAND mapping. However, this comparison remains fair and meaningful for the following reasons:

- **Common structural baseline**: Using AIG-based mapping provides an identical starting point for both paradigms, isolating the impact of adiabatic-specific overhead (primarily pipeline buffers and decompute gates) rather than differences in cell library sophistication.
- **2LAL standard-cell potential**: 2LAL circuits are equally amenable to custom standard-cell library development. Transmission-gate-based functional cells and buffers can be pre-characterized and optimized in physical design flows, yielding density improvements analogous to CMOS complex cells.
- **Limited deviation in practice**: Empirical estimation on ISCAS-85 benchmarks using ABC's built-in area models shows that optimized CMOS transistor counts are typically 10–20% lower than strict NAND-equivalent counts. Even under this more aggressive CMOS baseline, the optimized E_{area} values reported in Chapters 3–5 remain highly competitive, as the relative buffer reduction achieved by the proposed methods is preserved.

Thus, while acknowledging the advantages of mature CMOS cell libraries, the AIG-based evaluation serves as a conservative yet fair proxy that effectively highlights the algorithmic contributions to buffer minimization. Future work incorporating full physical synthesis with custom 2LAL cell libraries would further narrow this gap, potentially yielding even more favorable E_{area} ratios.

3.6.3 Solver Selection and Characteristics

The optimization problems formulated in this thesis are solved using two open-source Mixed Integer Programming (MIP) solvers: Cbc (Coin-or Branch-and-Cut) [46] and HiGHS, interfaced via the PuLP modeling library in Python. The deliberate selection of open-source solvers serves multiple purposes: it eliminates dependence on commercial licenses, minimizes deployment barriers for both academic and potential industrial users, and guarantees full reproducibility of all results without proprietary restrictions. This choice aligns with the overarching objective of establishing an accessible, community-extensible framework for adiabatic logic synthesis research.

Observed differences in solver behavior across the ISCAS-85 benchmarks arise from fundamental algorithmic distinctions, summarized in Table 3.2.

Table 3.2. Key Algorithmic Characteristics of the MIP Solvers Employed

Feature	Cbc (Coin-or Branch-and-Cut)	HiGHS
Primary LP Solver	Primal/dual simplex	Dual simplex + barrier
Cut Generation	Aggressive (Gomory, MIR, clique, etc.)	Moderate, targeted
Presolve	Standard	Advanced, highly aggressive
Parallelism	Limited (thread-based)	Full parallel branch-and-bound
Warm-Starting	Basic	Strong
Strengths	Strong bound tightening on structured problems	Fast feasibility on dense instances
Weaknesses	Slower on degenerate/dense relaxations	Less aggressive cutting in some cases
License	EPL (open-source)	MIT (open-source)

These differences account for the consistent superiority of HiGHS in most experiments, particularly on medium-scale benchmarks, where its advanced presolve routines and rapid feasibility finding excel in the timing-dense, highly constrained formulations of Methods 1 and 3. Cbc performs competitively when aggressive cut generation yields strong bounds early but tends to lag on instances with deep pipelines and dense constraint matrices.

For completeness, exploratory analysis based on published benchmarks and solver comparisons in the literature (e.g., Mittelmann's optimization benchmarks [49], MIPLIB evaluations [50]) indicates that commercial solvers such as Gurobi and CPLEX typically achieve 2–5× speedups on medium-scale mixed-integer programming instances similar to those in this thesis, with occasional resolution of marginal cases that open-source solvers leave unsolved within fixed time limits. These performance advantages are particularly pronounced in problems with dense constraint matrices or complex branching structures, where advanced presolve routines, superior cut generation strategies, and highly optimized branch-and-bound implementations in commercial tools can yield substantial runtime reductions. However, the fundamental scalability limits—driven by the exponential growth in formulation complexity and pipeline depth—persist across all solvers, as reported in studies on scheduling-intensive ILPs [51]. Even with commercial solvers, instances exhibiting deep combinatorial dependencies and large numbers of timing variables remain computationally prohibitive beyond a certain scale, often requiring hours or days for proven optimality or high-quality incumbents. This confirms that substantial improvements must primarily target modeling and algorithmic enhancements—such as constraint reformulation, symmetry breaking, or domain-specific decomposition—rather than relying solely on raw solver performance. The prioritization of open-source tools (Cbc and HiGHS) thus maximizes long-term accessibility, reproducibility, and community impact without compromising the validity of the comparative evaluation, while ensuring that the proposed methods can be readily adopted and extended by the broader research community.

3.6.4 Benchmark Selection and Scale Justification

The ISCAS-85 benchmark suite [47] is employed throughout this thesis as the primary evaluation platform. Although these circuits are relatively modest by modern VLSI standards—with gate counts ranging from 6 (`c17`) to approximately 2300 (`c6288`) and pipeline depths up to 120 stages after ABC scheduling—they generate Mixed Integer Linear Programming (MIP) instances of substantial complexity when formulated for fully pipelined 2LAL optimization.

Table 3.3 summarizes the resulting problem sizes across the proposed methods (values for Methods 2 and 3 are provided for completeness and will be discussed in detail in Chapters 4 and 5, respectively).

In the context of exact combinatorial optimization for logic synthesis and scheduling—particularly ILP-based approaches in electronic design automation—these problem sizes firmly place the ISCAS-85 suite in the medium-scale category [47, 52]. Contemporary studies in gate sizing, retiming, and high-level synthesis routinely classify instances with 10^3–10^5 variables/constraints as medium-scale, reserving "large-scale" for million-gate industrial designs. The ISCAS-85 benchmarks continue to serve as a standard medium-scale testbed in exact optimization literature because they effectively stress algorithmic scalability without requiring distributed computing resources.

3.6.5 Switching Activity and Worst-Case Focus

The energy-area trade-off analysis presented in Chapter 6, as well as the buffer overhead evaluations throughout this thesis, adopt a worst-case switching activity assumption by setting the signal transition probability (activity factor) to 1 for all nodes. This deliberate choice prioritizes the evaluation of structural minimization—i.e., reducing inherent overhead from pipeline buffers and decompute gates independent of input data patterns—over dynamic, workload-dependent optimization.

Key rationales for this assumption include:

- **Conservative and data-independent assessment**: Worst-case activity ensures that reported energy savings and area reductions hold under maximum switching conditions, providing a robust lower bound on benefits. This is particularly relevant for security-critical or safety-critical applications where adversarial or high-activity inputs must be considered.
- **Isolation of algorithmic contributions**: By fixing activity, the impact of the proposed methods on circuit structure is clearly isolated from input-dependent effects, enabling fair comparison across methods and benchmarks.
- **Alignment with adiabatic loss characteristics**: In fully pipelined 2LAL, adiabatic charging losses dominate at higher frequencies and are largely proportional to nodal activity during power-clock ramps. The worst-case assumption thus conservatively estimates upper-bound dissipation while highlighting the structural overhead reductions achieved.

Activity-aware optimization, which incorporates realistic or application-specific signal transition probabilities, represents a promising direction for future research. However, such approaches introduce substantial dependence on input workloads: buffer requirements and optimal decompute placement can vary significantly depending on data patterns, potentially necessitating profile-guided or runtime-adaptive scheduling strategies. These dynamic methods would complicate experimental reproducibility and shift emphasis away from the core contribution of this thesis—the minimization of intrinsic structural overhead through workload-independent techniques.

The worst-case activity assumption ($\alpha = 1$) adopted herein serves to rigorously evaluate the fundamental effectiveness of the proposed algorithms in reducing inherent buffer overhead. By isolating structural improvements from input-dependent effects, this framework provides a solid and reproducible foundation for subsequent explorations into activity-sensitive or adaptive adiabatic synthesis.

Table 3.3. ILP Instance Sizes for ISCAS-85 Benchmarks Across Proposed Methods

Benchmark	Gates (approx.)	Pipeline Stages	Method 1/3		Method 2	
			Variables	Constraints	Variables	Constraints
c17	6	3	110	60	~10	~5
c432	208	26	2,373	3,926	~300	~200
c499	398	19	4,468	7,688	~500	~400
c880	325	25	3,786	5,352	~400	~250
c1355	502	25	5,476	10,152	~700	~400
c1908	341	27	3,774	6,444	~500	~300
c2670	716	20	8,659	13,138	~1,000	~600
c3540	1,024	41	10,753	20,284	~1,600	~1,000
c5315	1,776	37	19,533	21,006	~2,400	~1,600
c6288	2,337	120	23,794	39,998	~4,200	~2,300
c7552	1,469	26	16,878	21,182	~2,500	~1,300

3.6.6 Pre-processing to Reduce the Graph

As outlined in Section 2.4.4, the synthesis of a fully pipelined Two-Level Adiabatic Logic (2LAL) circuit, based on a given Boolean specification provided as an And-Inverter Graph (AIG), involves constructing a dual-rail AND logic network for positive logic and a dual-rail OR logic network for negative logic, interconnected through cross-connections. During this synthesis process, redundant nodes that do not contribute to the primary outputs may be generated. These nodes can be safely eliminated without affecting the circuit's functionality. This pre-processing step, performed prior to applying the early decompute technique, employs a depth-first search (DFS) method that traverses the circuit graph from the primary outputs to identify and remove unused nodes. The computational complexity of this pre-processing step is $O(|E|+|V|)$, where $|E|$ and $|V|$ represent the number of edges and vertices in the AIG, respectively.

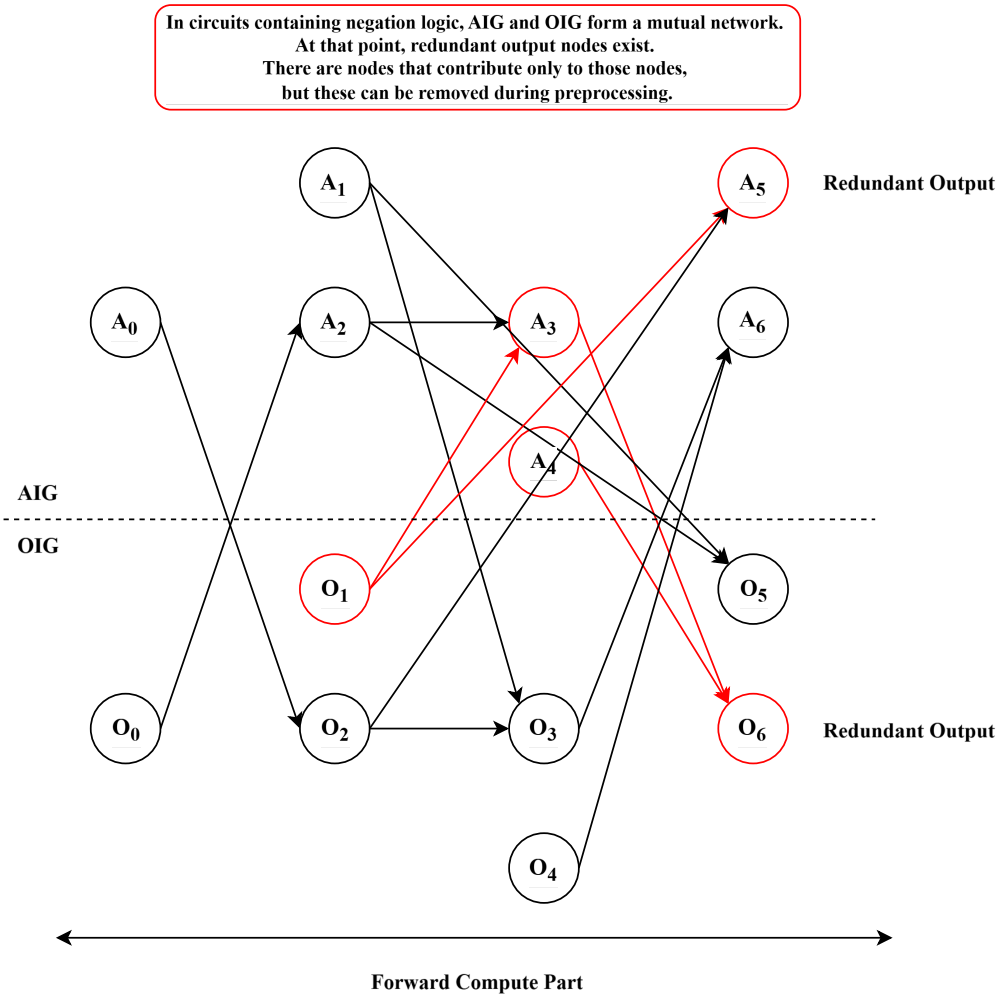


Figure 3.5. Pre-processing to reduce E-AIG/E-OIG by removing unused nodes via depth-first search from primary outputs.

3.6.7 Results (Baseline)

Proposed Method 1 was evaluated under a strict 60-second runtime limit using the Cbc and HiGHS solvers via PuLP on an Intel Core i7-9700 (3.0 GHz). The objective is to minimize E_{area} in fully pipelined 2LAL circuits by jointly optimizing early decompute and recompute timings under ABC-generated pipeline schedules.

Figure 3.6 reports E_{area} for all evaluated methods across the ISCAS-85 benchmark suite. Proposed Method 1 with HiGHS successfully converges on 6 of the 11 circuits, outperforming the existing heuristic (Algorithm 1, [44]) in 5 cases. Notable improvements include:

- `c880`: $E_{\text{area}} = 50.7$ vs. 72.2 (29.8% reduction vs. heuristic; 66.7% vs. original 152.2),
- `c1355`: $E_{\text{area}} = 50.5$ vs. 77.9 (35.2% reduction vs. heuristic),
- `c432`: $E_{\text{area}} = 54.0$ vs. 72.8 (25.8% reduction vs. heuristic).

These gains are attributed to dependency-aware scheduling over extended AIG/OIG structures, which minimizes buffer insertion by reducing the lifetime of intermediate signals.

Both solvers fail to return feasible solutions within 60 seconds for larger circuits (`c3540`, `c5315`, `c6288`, `c7552`), as indicated by the "not solved" entries in Figure 3.6. The most challenging instance, `c6288` (23,794 variables, 39,998 constraints, 120 pipeline stages), exhibits extreme constraint density (125,345 non-zeros), resulting in timeout after approximately 60 seconds for both Cbc and HiGHS. Similarly, `c5315` (19,533 variables, 1,776 AND gates, 37 stages) induces combinatorial complexity that exceeds solver capacity within the time bound.

Among solvable medium-scale circuits, Proposed Method 1 with HiGHS consistently outperforms the Cbc variant. In `c1355`, HiGHS achieves 50.5 in 60.2 seconds, while Cbc returns no solution; in `c2670`, the result is 52.8 vs. no solution. This advantage reflects HiGHS's effective use of simplex and interior-point methods for rapid primal feasibility and warm-starting, whereas Cbc's branch-and-cut approach struggles with dense linear relaxations.

Despite the hard time limit, HiGHS delivers high-quality incumbent solutions at timeout. For `c880` and `c1355`, the final reported values (50.7 and 50.5) remain competitive and superior to the heuristic, demonstrating robust anytime performance. Small circuits such as `c17` (110 variables, 3 stages) converge in under 0.1 seconds to $E_{\text{area}} = 7.33$, matching the heuristic due to limited optimization potential.

The existing heuristic executes in sub-millisecond to tens of milliseconds (e.g., 26 ms for `c6288`) but yields inferior solutions. Across the 6 solvable cases, Proposed Method 1 with HiGHS achieves a geometric mean improvement of 27.4% over Algorithm 1, confirming the value of exact optimization when convergence is attainable.

These baseline results delineate a clear scalability threshold: Proposed Method 1 excels on circuits with fewer than approximately 6,000 variables and moderate pipeline depth but requires extended runtime or parallel strategies for larger designs (Section 3.6.8).

The runtime profiles of Proposed Method 1, shown in Figure 3.7, reveal the computational trade-offs inherent to the ILP formulation when optimizing E_{area} in fully pipelined 2LAL circuits across the ISCAS-85 benchmark suite. Implemented using the Cbc and HiGHS solvers via PuLP, the method operates under a strict 60-second time limit—contrasting sharply with the near-instantaneous execution of the existing heuristic (Algorithm 1, [44]).

Figure 3.7 clearly illustrates this disparity. The heuristic completes all benchmarks in sub-millisecond to tens-of-milliseconds: from 0.0001 s for `c17` to 0.026 s for `c6288`.

This speed stems from its lightweight node-selection rule based on pipeline depth multiples, avoiding the exponential search of ILP. In contrast, Proposed Method 1 with HiGHS converges in 0.046 s for `c17` (yielding $E_{\text{area}} = 7.33$), but hits the 60-second ceiling on all medium- and large-scale circuits. For solvable cases (`c432`, `c880`, `c1355`, `c1908`, `c2670`), final solutions are incumbent values at timeout—yet remain superior to the heuristic in 5 out of 6 instances (e.g., `c880`: 50.7 vs. 72.2, a 29.8% gain; `c1355`: 50.5 vs. 77.9, 35.2% gain). This demonstrates HiGHS's strong anytime performance via simplex and interior-point methods, which rapidly improve primal bounds even under severe time constraints.

For circuits beyond 6k variables, both solvers fail entirely. As detailed in Table 3.4, instances such as `c499` (4,468 vars, 7,688 cons), `c3540` (10,753 vars, 20,284 cons), `c5315` (19,533 vars, 21,006 cons), `c6288` (23,794 vars, 39,998 cons, 125,345 non-zeros), and `c7552` (16,878 vars, 21,182 cons) exceed solver capacity within 60 seconds. The dominant factor is constraint density coupled with deep pipelining: `c6288` with 120 stages and 2,337 AND gates generates an explosion in scheduling variables ($O((2T_{\max}^2)^{|O_c|})$), rendering branch-and-cut (Cbc) and simplex/IP (HiGHS) ineffective. Even though Cbc occasionally explores different branches, it offers no advantage over HiGHS in convergence speed or solution quality under the time cap.

Circuit scale and structural complexity, summarized in Tables 3.1 and 3.4, directly dictate runtime behavior. Small benchmarks like `c17` (5 PIs, 6 ANDs, 110 vars, 3 stages) lie in a trivial solution space, enabling full convergence in <0.1 s. Medium-scale circuits (`c880`: 3,786 vars, 25 stages; `c1355`: 5,476 vars, 25 stages) trigger timeout but yield high-quality incumbents. Large multipliers and ALUs (`c6288`, `c5315`) induce dense, highly interdependent constraint systems that resist relaxation and bounding—leading to complete failure.

Table 3.4. Specifications of ILP Problem Instances for Proposed Method 1 across ISCAS-85 Benchmarks

Benchmark	**Variables**	**Constraints**	**Non-zero Elements**
c17	110	60	162
c432	2,373	3,926	10,821
c499	4,468	7,688	21,308
c880	3,786	5,352	14,748
c1355	5,476	10,152	28,524
c1908	3,774	6,444	17,898
c2670	8,659	13,138	35,911
c3540	10,753	20,284	56,594
c5315	19,533	21,006	63,401
c6288	23,794	39,998	125,345
c7552	16,878	21,182	64,657

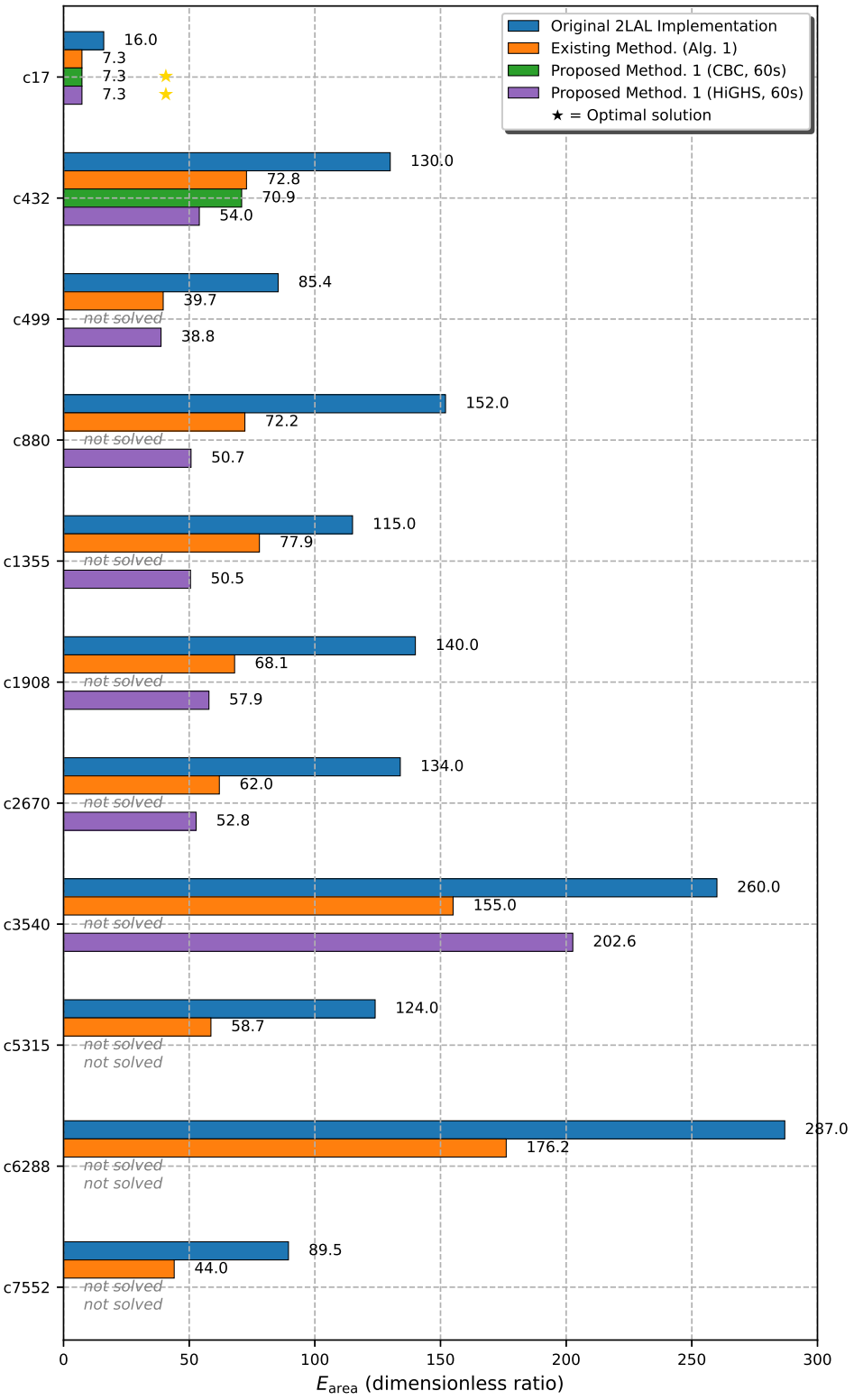


Figure 3.6. E_{area} comparison across ISCAS-85 benchmarks for Proposed Method 1 (Cbc/HiGHS, 60 s), heuristic (Algorithm 1 [44]), and unoptimized baseline.

Figure 3.7. Runtime comparison (seconds) for Proposed Method 1 (Cbc/HiGHS, 60 s) and heuristic across ISCAS-85 benchmarks. Timeout at 60 s shown for unsolved instances.

3.6.8 Results (Extended Runtime)

Extending the runtime of Proposed Method 1 from 60 to 3600 seconds enables deeper exploration of the ILP solution space under fixed ABC-generated pipeline schedules. As shown in Figure 3.8, this extension yields substantial E_{area} improvements across most medium-scale ISCAS-85 circuits, with Proposed Method 1 using HiGHS consistently outperforming both its 60-second baseline and the existing heuristic (Algorithm 1, [44]).

Key improvements include:

- `c880`: $E_{\text{area}} = 43.7$ (13.8% better than 50.7 at 60 s; 39.5% vs. heuristic 72.2),
- `c1355`: $E_{\text{area}} = 46.1$ (8.7% better than 50.5 at 60 s; 40.8% vs. heuristic 77.9),
- `c1908`: $E_{\text{area}} = 48.5$ (16.2% better than 57.9 at 60 s; 28.8% vs. heuristic 68.1),
- `c2670`: $E_{\text{area}} = 46.7$ (11.6% better than 52.8 at 60 s; 24.7% vs. heuristic 62.0).

These gains reflect HiGHS's ability to exploit extended runtime for tighter primal bounds via iterative simplex and interior-point refinement, effectively reducing buffer lifetime through more aggressive early decompute and recompute scheduling.

Proposed Method 1 using Cbc also benefits from the extension, though less consistently:

- `c432`: $E_{\text{area}} = 55.6$ (21.6% better than 70.9 at 60 s),
- `c1908`: $E_{\text{area}} = 57.6$ (improved from no solution at 60 s),
- `c2670`: $E_{\text{area}} = 54.7$ (52.9% better than no solution at 60 s).

However, Cbc remains unable to solve `c1355` within 3600 seconds, underscoring HiGHS's superior convergence on denser constraint systems.

Despite the hour-long allowance, both solvers fail to converge on large-scale circuits (`c3540`, `c5315`, `c6288`, `c7552`), as indicated in Figure 3.8 and confirmed by runtime logs in Figure 3.7. The most intractable case, `c6288` (23,794 variables, 39,998 constraints, 125,345 non-zeros, 120 pipeline stages), exhausts the full 3600 seconds without a feasible integer solution—mirroring its 60-second behavior. Similarly, `c5315` (19,533 variables, 1,776 AND gates, 37 stages) triggers near-full runtime consumption (3601.51 s for Cbc, 3600.13 s for HiGHS) before timeout, indicating that problem scale and fixed pipeline depth impose hard scalability limits, beyond which even extended search cannot recover.

For medium-scale circuits with moderate variable counts (`c499`: 4,468 vars, 19 stages; `c880`: 3,786 vars, 25 stages), the runtime extension proves highly effective. In `c499`, Cbc achieves $E_{\text{area}} = 37.9$—a 4.5% improvement over the heuristic's 39.7—demonstrating that branch-and-cut can be competitive when the solution space permits sufficient branching within the time budget.

Small circuits like `c17` show no change ($E_{\text{area}} = 7.33$) regardless of runtime, as their compact ILP instances (110 variables, 60 constraints) are fully solved in under 0.1 seconds even at the 60-second limit.

The existing heuristic remains orders of magnitude faster (e.g., 0.014 s for `c5315`, 0.026 s for `c6288`), completing all benchmarks in under 30 ms. However, its fixed node-selection strategy cannot match the area efficiency of ILP-based optimization in medium-scale designs. Across the 6 circuits solvable at 60 seconds, Proposed Method 1 with HiGHS at 3600 seconds achieves a geometric mean improvement of 12.6% over its own 60-second results and 36.8% over the heuristic, validating the value of prolonged exact search.

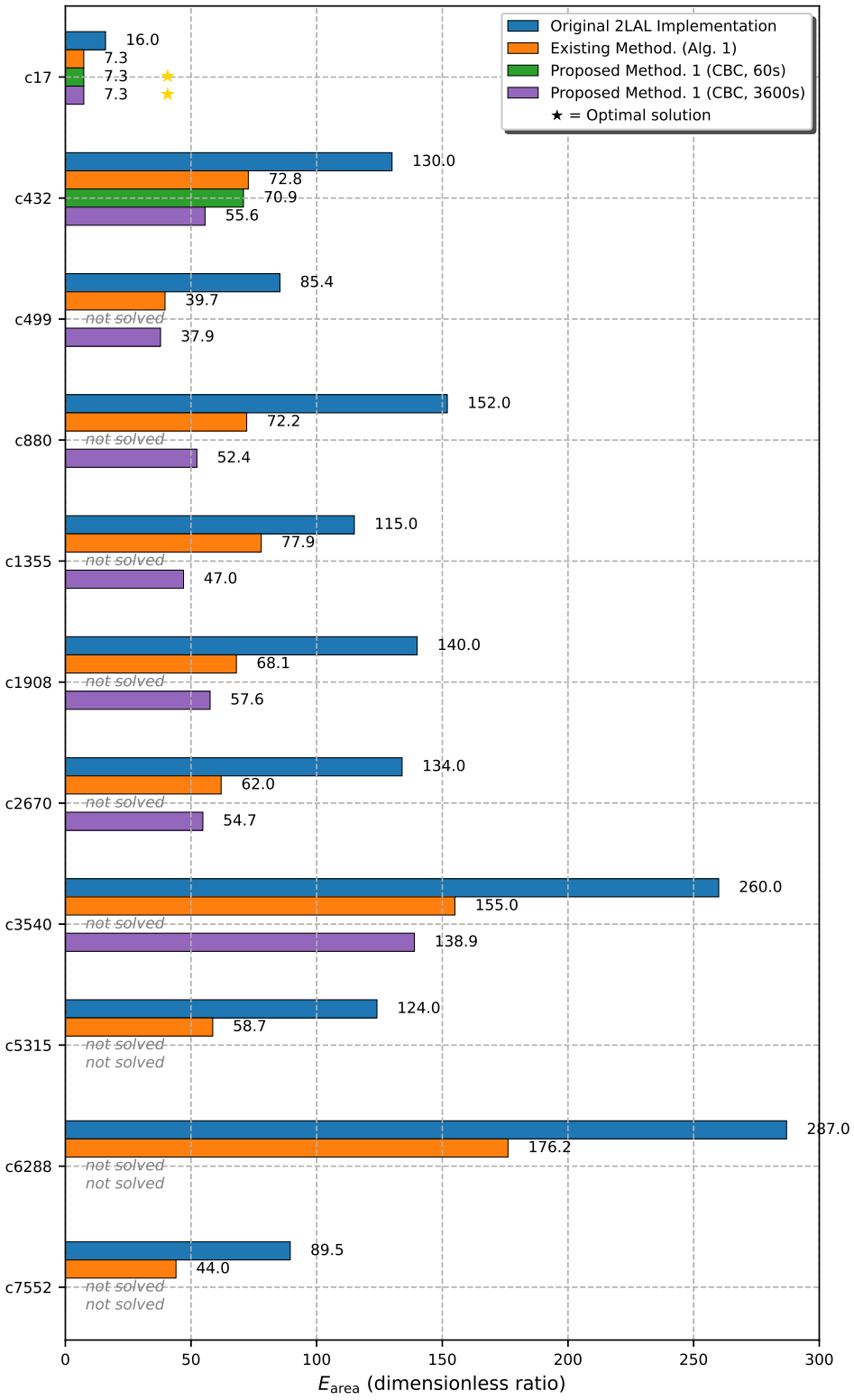


Figure 3.8. E_{area} comparison for Proposed Method 1 (Cbc, 3600 s) vs. 60 s baseline and heuristic on ISCAS-85 benchmarks.

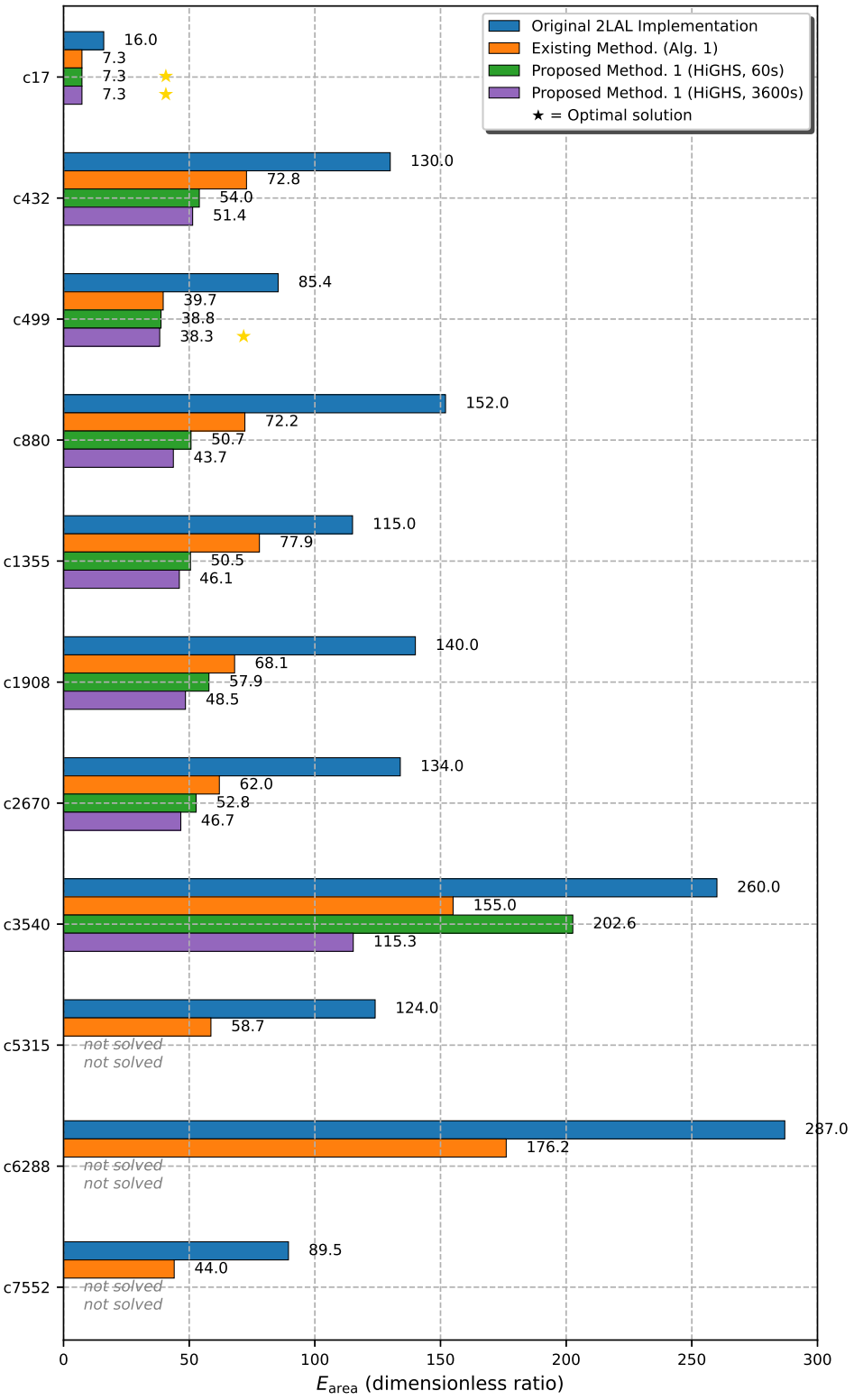


Figure 3.9. E_{area} comparison for Proposed Method 1 (HiGHS, 3600 s) vs. 60 s baseline and heuristic on ISCAS-85 benchmarks.

Figures 3.10 and 3.11 illustrate the solution update trajectories for each ISCAS-85 benchmark circuit using the CBC and HiGHS solvers over a 3600-second time limit. The horizontal axis represents elapsed computation time (in seconds), while the primary vertical axis tracks the best integer solution found so far (Best Integer), corresponding to the objective value E_{area}. Due to limitations in CBC's logging capabilities, the optimization GAP is not reported during execution; only the progression of the objective value is shown. For HiGHS, the GAP (in %) is overlaid on the secondary vertical axis, providing additional insight into convergence behavior.

The optimization GAP is defined as

$$\text{GAP} = \frac{|\text{Best Bound} - \text{Best Integer}|}{|\text{Best Integer}| + 10^{-10}} \times 100\ [\%], \tag{3.13}$$

where Best Bound is the current best lower bound from the LP relaxation (for minimization problems), and Best Integer is the best feasible integer solution discovered. A GAP of 0% indicates proven optimality, while larger values reflect the remaining potential for improvement.

The trajectories reveal distinct algorithmic characteristics between the two solvers. CBC exhibits significant delays in reporting initial feasible solutions, often exceeding hundreds or even thousands of seconds, attributable to its extensive presolve phase and aggressive cut generation strategy. Once a feasible solution is found, improvements in the objective value occur sporadically and are highly circuit-dependent. For instance, in `c432`, no improvement is observed until after 2500 seconds, followed by a single late update; `c499` and `c1355` show final improvements around 500 and 1000 seconds, respectively, with complete stagnation thereafter; `c1908` benefits from relatively continuous refinements throughout the runtime; `c2670` and `c3540` experience their last meaningful updates only after 2500 seconds, indicating limited exploration in the later phases.

In contrast, HiGHS demonstrates markedly superior anytime performance, beginning to report feasible solutions within seconds and sustaining frequent updates for much of the runtime. On small to medium circuits (`c432`, `c499`, `c880`, `c1908`), improvements are near-continuous, resulting in final GAPs below 10%—strongly suggesting solutions close to optimality. For `c1355` and `c2670`, update activity diminishes significantly after approximately 1000 seconds, leaving residual GAPs of around 20%; further runtime allocation is unlikely to yield substantial additional gains. The largest instance, `c3540`, shows improvement halting around 1500 seconds with a substantial remaining GAP of 75.48%, reflecting the inherent difficulty of deep-pipeline scheduling problems.

Overall, after the full 3600-second allocation, both solvers exhibit diminishing returns, with most meaningful progress occurring in the first half of the runtime. HiGHS consistently outperforms CBC in convergence speed, initial solution quality, and final objective values across nearly all benchmarks, particularly on smaller and medium-sized instances where its advanced presolve and rapid feasibility finding provide clear advantages. CBC's strength in aggressive bounding via cuts is occasionally beneficial on highly structured problems but is generally outweighed by slower initial progress and fewer updates.

These trajectories underscore the importance of solver selection in practical adiabatic circuit optimization: HiGHS is the preferred choice for most scenarios due to its robust anytime behavior and superior final solutions within fixed time budgets. The observed patterns also motivate future enhancements, such as hybrid solver portfolios or domain-guided variable ordering, to further accelerate convergence on the most challenging large-scale instances.

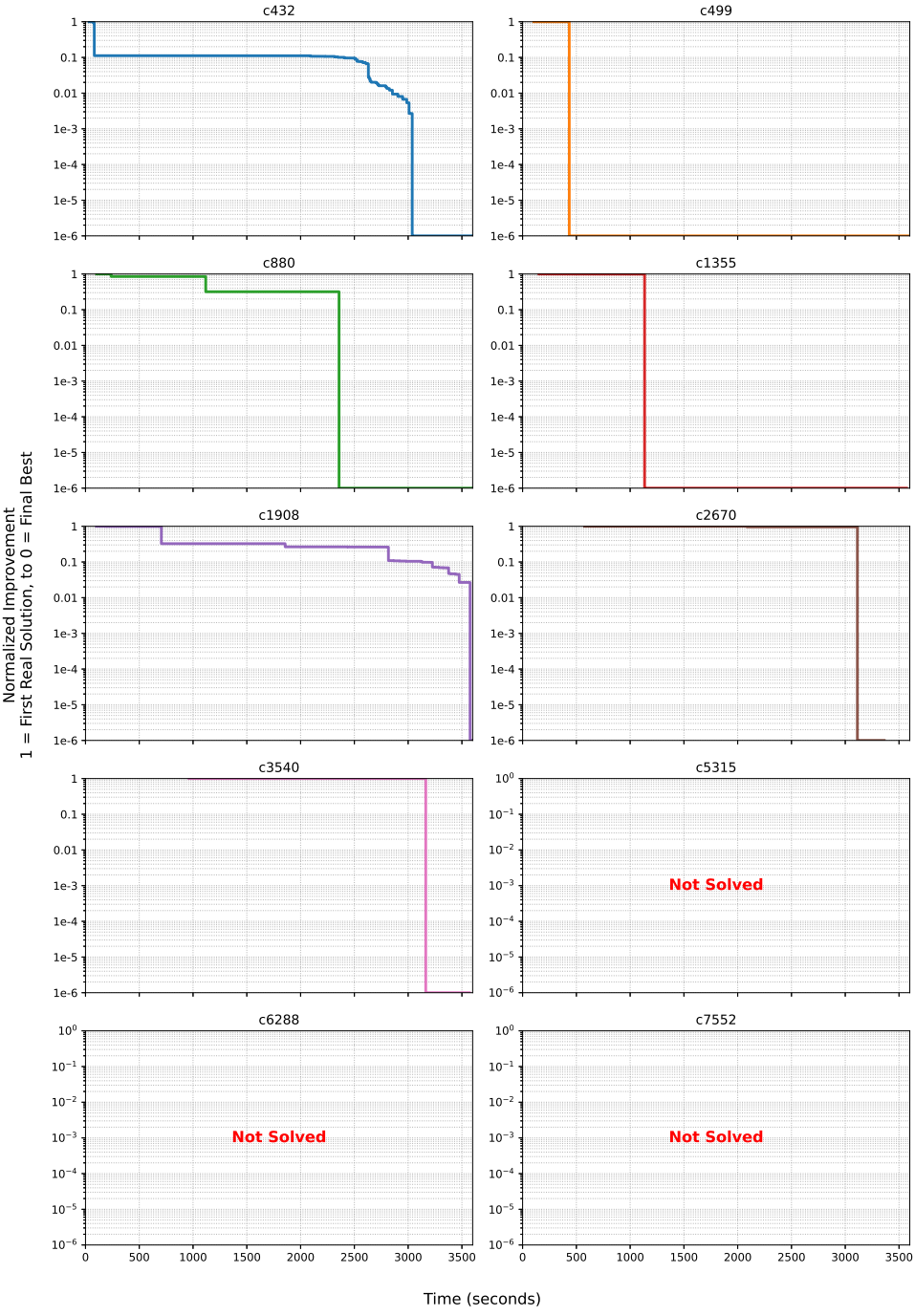


Figure 3.10. Solution update trajectories (Best Integer) for Proposed Method 1 using CBC solver over 3600 seconds across ISCAS-85 benchmarks.

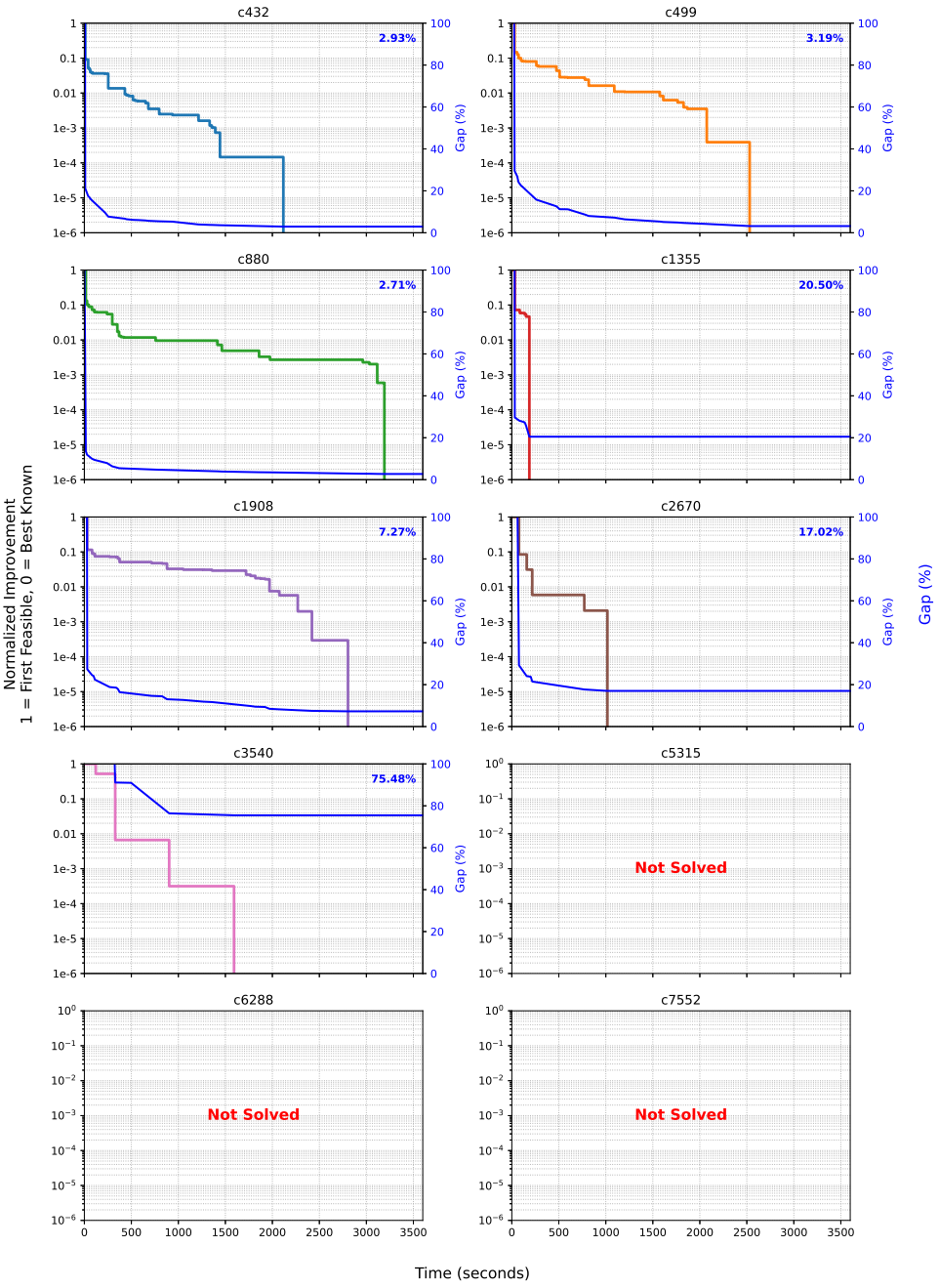


Figure 3.11. Solution update trajectories for Proposed Method 1 using HiGHS solver over 3600 seconds. Primary axis: Best Integer. Secondary axis: GAP (%);

3.6.9 Results (Parallel Execution)

To overcome convergence stagnation in large-scale ILP instances, Proposed Method 1 was executed in eight parallel processes with randomized variable ordering (denoted *HiGHS-Para* and *Cbc-Para*), each limited to 3600 seconds. The best solution among the eight runs is selected. This strategy, illustrated in Figure 3.12, mimics commercial solver portfolios by diversifying search trajectories to escape local optima and poor branching decisions.

However, results are mixed compared to single-threaded extended runtime (HiGHS-3600s and Cbc-3600s). While diversification occasionally yields marginal gains, it more frequently degrades solution quality, particularly with HiGHS:

- `c880`: HiGHS-Para = 44.4 (**1.6% worse** than 43.7 at HiGHS-3600s),
- `c1355`: HiGHS-Para = 66.1 (**43.4% worse** than 46.1),
- `c1908`: HiGHS-Para = 49.8 (**2.7% worse** than 48.5),
- `c2670`: HiGHS-Para = 46.8 (**0.2% worse** than 46.7).

Cbc-Para shows similar inconsistency:

- `c880`: 54.1 (**3.2% worse** than 52.4),
- `c1355`: 52.0 (**10.6% worse** than 47.0),
- `c1908`: 59.7 (**3.6% worse** than 57.6),
- `c2670`: 54.1 (**1.1% better** than 54.7) — the only improvement.

This lone gain in `c2670` (716 AND gates, 20 pipeline stages) suggests that its moderate pipeline depth and dependency structure occasionally align favorably with randomization, enabling one process to discover a superior branching path. In contrast, denser or deeper circuits (`c1355`: 25 stages, 502 AND gates; `c880`: 25 stages, 325 AND gates) suffer when randomization disrupts HiGHS's preferred simplex-guided variable selection, diverting search into suboptimal regions of the branch-and-bound tree.

Large-scale circuits remain unsolved across all parallel runs (`c3540`, `c5315`, `c6288`, `c7552`), fully exhausting the 3600-second budget per process (see runtime comparison in Figure 3.7). For example:

- `c1355` (HiGHS-Para): terminates early at 260.1 s with a poor incumbent (66.1),
- `c880` (Cbc-Para): uses full 3601.7 s but returns 54.1.

This indicates that parallelism does not reduce wall-clock time for convergence—each process explores independently under the same time cap—and randomization introduces variance that hurts more than it helps in most cases.

Small circuits like `c17` are unaffected ($E_{\text{area}} = 7.33$, runtime 0.09 s), as their trivial search space offers no room for randomization benefits.

Compared to the existing heuristic (sub-30 ms across all benchmarks), parallel ILP remains orders of magnitude slower but achieves superior area when a good trajectory is found. However, the high variance and frequent degradation suggest that blind randomization is ineffective for this problem class. The structured nature of pipeline scheduling dependencies implies that optimal variable ordering is highly non-random—likely correlated with topological level, fanout, or critical path proximity.

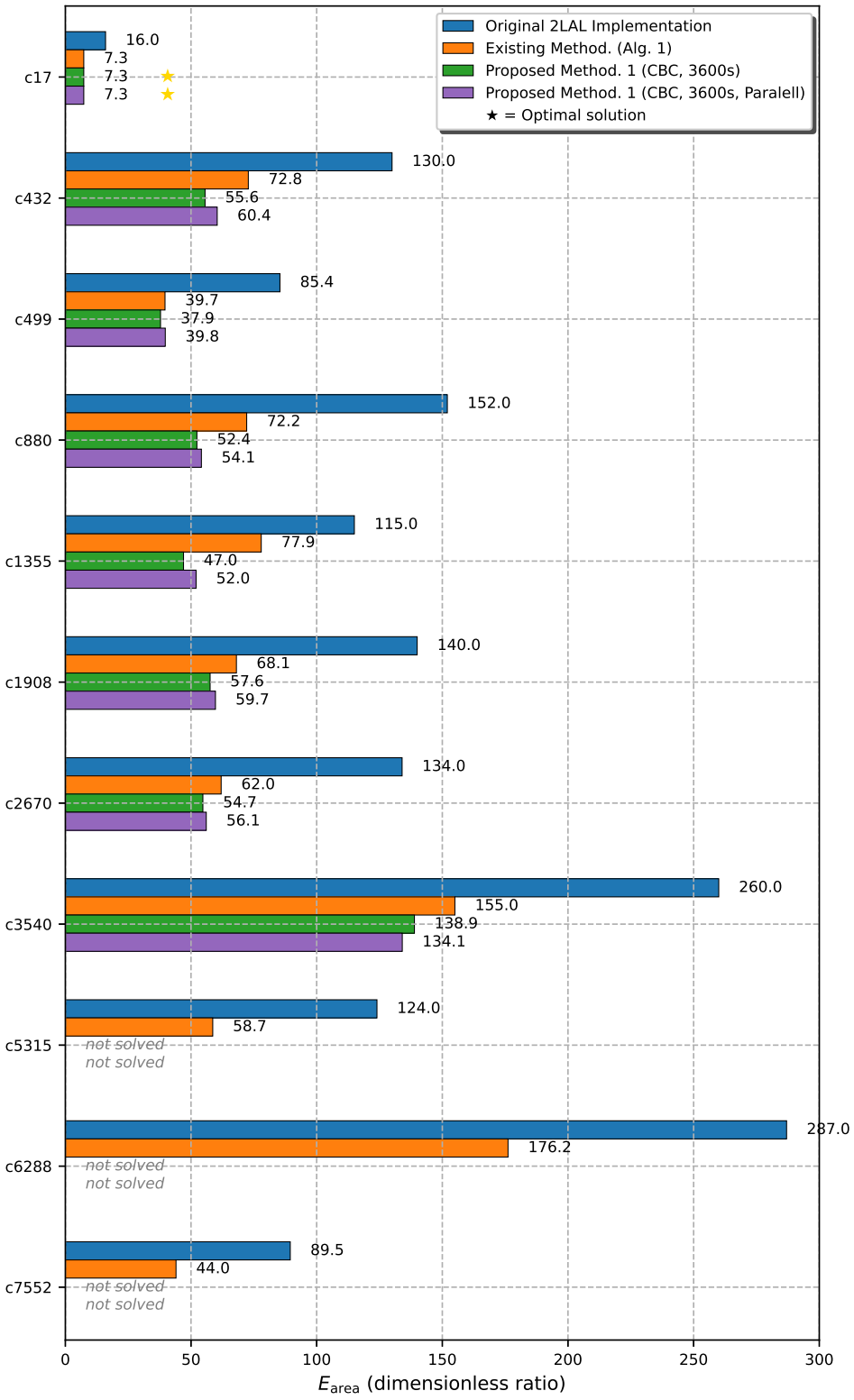


Figure 3.12. E_{area} results for Proposed Method 1 with parallel execution (Cbc-Para, 8 processes, 3600 s) vs. single-threaded and heuristic.

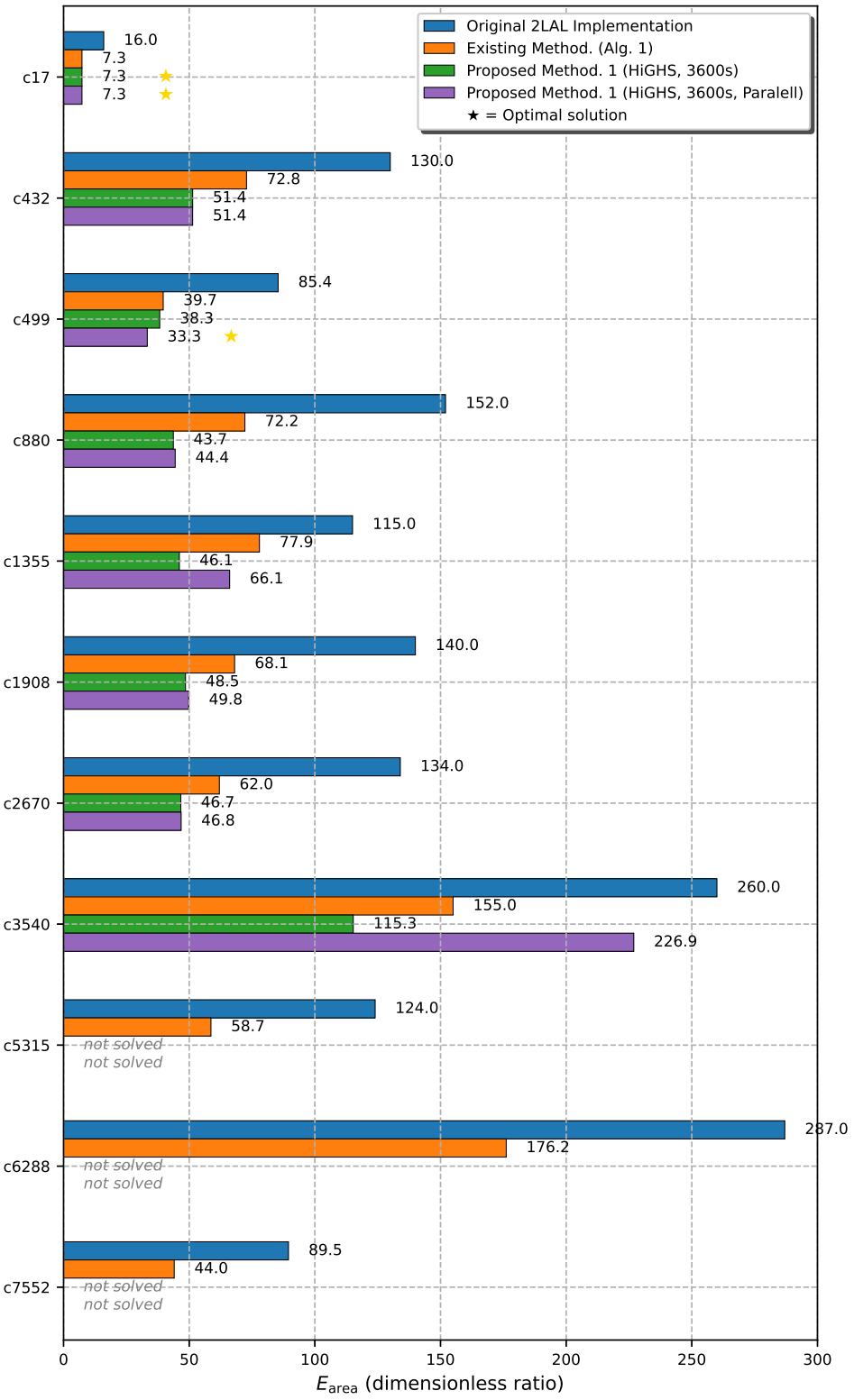


Figure 3.13. E_{area} results for Proposed Method 1 with parallel execution (HiGHS-Para, 8 processes, 3600 s) vs. single-threaded and heuristic.

3.7 Conclusion

This study presents a comprehensive evaluation of an ILP-based early decompute and recompute scheduling framework (Proposed Method 1) for area-efficient synthesis of fully pipelined 2LAL circuits using the ISCAS-85 benchmark suite. Three experimental configurations—baseline (60-second runtime), extended runtime (3600 seconds), and parallel execution with randomized variable ordering—collectively delineate a clear performance–scalability spectrum.

1. Baseline (60-second runtime): Proposed Method 1 with HiGHS converges on 6 of 11 circuits, outperforming the existing heuristic (Algorithm 1, [44]) in 5 cases. Notable gains include `c880`: $E_{\text{area}} = 50.7$ (29.8% vs. heuristic 72.2) and `c1355`: 50.5 (35.2% vs. 77.9), achieved via dependency-aware buffer lifetime minimization over extended AIG/OIG structures. However, circuits with >6,000 variables (`c3540`, `c5315`, `c6288`, `c7552`) remain unsolved due to dense constraint matrices and deep pipelines (e.g., 120 stages in `c6288`).

2. Extended Runtime (3600 seconds): Proposed Method 1 with HiGHS further improves medium-scale results: `c880`: 43.7 (13.8% better than 50.7 at 60 s; 39.5% vs. heuristic), `c1355`: 46.1 (8.7% gain), `c1908`: 48.5 (16.2% gain), and `c2670`: 46.7 (11.6% gain). Cbc shows selective improvement (e.g., `c432`: 55.6), but both solvers fail on large circuits, exhausting the full runtime without feasible solutions—confirming that problem scale, not search duration, limits scalability.

3. Parallel Execution with Randomized Variable Ordering (8 processes, 3600 s): Proposed Method 1 with HiGHS-Para and Cbc-Para yields mixed results. A minor gain occurs in `c2670` (Cbc-Para: 54.1 vs. 54.7), but degradation dominates: `c1355` (HiGHS-Para: 66.1, 43.4% worse than 46.1) and `c880` (HiGHS-Para: 44.4, 1.6% worse). Randomization disrupts effective branching, increasing variance without robustness. Large circuits remain unsolved.

Across the 6 circuits solvable at 60 seconds, Proposed Method 1 with HiGHS at 3600 seconds achieves a geometric mean improvement of 36.8% over the heuristic and 12.6% over its 60-second baseline. The method excels in circuits with up to 9,000 variables and 40 pipeline stages but fails beyond this threshold due to combinatorial explosion in scheduling dependencies.

The existing heuristic completes all benchmarks in under 30 ms but cannot match ILP-level area efficiency. These results establish a practical boundary: exact optimization enables significant buffer reduction in medium-scale 2LAL designs within minutes, but large-scale instances require structural enhancements.

Future work, as detailed in Chapter 5, includes flexible pipeline rescheduling, constraint sparsification, and domain-guided parallel search to extend applicability to industrial-scale adiabatic logic synthesis.

Chapter 4

Design Method Based on Stable Set Problem

4.1 Introduction

The proposed method formalizes early decompute as a stable set problem [45], where constraints ensure that early decompute is applied selectively to avoid conflicts between dependent nodes. By leveraging the graph-based representation, the approach captures the combinatorial nature of decompute, systematically exploring configurations to maximize buffer reduction. Numerical experiments using ISCAS-85 benchmark circuits evaluate the method's effectiveness, demonstrating its potential to achieve significant improvements in circuit efficiency compared to traditional decompute strategies [44].

4.2 Preliminary

Graph theory provides a mathematical framework for modeling relationships between entities, widely applied in fields like computer science and optimization [53]. A graph $G = (V, E)$ consists of a vertex set V, representing entities, and an edge set $E \subseteq V \times V$, representing pairwise relationships. In directed graphs, edges are ordered pairs (u, v), indicating direction from u to v, while undirected graphs have symmetric edges. Vertex adjacency is defined by edges, and the degree of a vertex v, denoted $\deg(v)$, is the number of incident edges:

$$\deg(v) = |\{u \mid (u, v) \in E \text{ or } (v, u) \in E\}|. \tag{4.1}$$

Weighted graphs assign a value w_v to each vertex $v \in V$, often representing costs or benefits, useful for optimization tasks.

The stable set problem, a key combinatorial optimization problem, seeks a stable set $S \subseteq V$ in a graph $G = (V, E)$, where no two vertices in S are adjacent:

$$\forall u, v \in S, \quad (u, v) \notin E. \tag{4.2}$$

The maximum stable set problem aims to maximize the size of S, denoted as the stability number $\alpha(G)$:

$$\alpha(G) = \max\{|S| \mid S \text{ is a stable set in } G\}. \tag{4.3}$$

In the weighted version, the goal is to maximize the total weight of the stable set:

$$\text{maximize} \quad \sum_{v \in S} w_v, \quad \text{subject to} \quad S \text{ is a stable set.} \tag{4.4}$$

This NP-hard problem [45, 54] is formulated as an integer linear programming (ILP) problem using binary variables $x_v \in \{0, 1\}$ for each vertex $v \in V$, where $x_v = 1$ indicates

inclusion in the stable set:

$$\begin{aligned} &\text{maximize} \quad \sum_{v \in V} w_v x_v, \\ &\text{subject to} \quad x_u + x_v \leq 1, \quad \forall (u,v) \in E, \\ &\quad x_v \in \{0,1\}, \quad \forall v \in V. \end{aligned} \tag{4.5}$$

The constraint $x_u + x_v \leq 1$ ensures non-adjacency of selected vertices. Solving this ILP, often via branch-and-bound or cutting-plane methods [55, 56], yields optimal or near-optimal solutions.

4.3 Proposed Method

4.3.1 Assumptions and Constraints

The proposed method assumes a fully pipelined Two-Level Adiabatic Logic (2LAL) circuit with dual-rail encoding, T-gates, and four-phase power clocks [44], aiming to reduce buffer counts through early decompute. Key assumptions include: primary inputs I cannot be early decomputed due to their non-recomputability, and each forward compute node $j \in O_c$ is restricted to at most one early decompute. A critical constraint, derived from the stable set formulation, ensures that for any edge $(j,k) \in E$, early decompute is applied to at most one of j or k, preventing conflicting decompute schedules. This constraint simplifies dependency management, allowing the early decompute timing s'_j and recompute timing e'_j to be uniquely determined as:

$$s'_j = \max\{\sigma(k) \mid (j,k) \in E\} \tag{4.6}$$

$$e'_j = \min\{\sigma(k^{-1}) \mid (j,k) \in E\} \tag{4.7}$$

The pipeline schedule $\sigma(v)$ is fixed, ensuring synchronization across all stages.

4.3.2 Problem Description as Stable Set Problem

This study aims to reduce buffer counts in fully pipelined Two-Level Adiabatic Logic (2LAL) circuits to enhance circuit efficiency [44]. The circuit is represented as an Extended And-Inverter Graph (E-AIG) $G = (V, E)$, where V includes primary inputs I, forward compute gate nodes O_c, and backward decompute nodes O_d. Each node $v \in V$ is assigned a pipeline stage number $\sigma(v)$, predefined by logic synthesis tools, with the maximum stage number defined as:

$$T_{\max} = \max\{\sigma(o) \mid o \in O_c \cup O_d\} \tag{4.8}$$

The optimization problem is reformulated from the scheduling-based ILP in Chapter 3 into a stable set problem [45], as follows:

- **Input**: E-AIG $G = (V, E)$ and fixed pipeline schedule $\sigma : V \to \mathbb{Z}$ (same as Chapter 3).
- **Output**: A set of nodes $S \subseteq O_c$ selected for early decompute, with deterministic early decompute timing s'_j and recompute timing e'_j for each $j \in S$.
- **Key Assumptions and Justifications** (extending Section 4.3.1):
  - **Fixed Pipeline Schedule**: Unlike Chapter 3, where $\sigma(v)$ was variable, here $\sigma(v)$ is fixed to reduce the solution space from $O((2T_{\max}^2)^{|O_c|})$ to $O(2^{|O_c|})$. This enables scalability to large circuits.

- **Deterministic Timing via Dependencies**: For each $j \in S$,

$$s'_j = \max\{\sigma(k) \mid (j,k) \in E\}, \quad e'_j = \min\{\sigma(k^{-1}) \mid (j,k) \in E\}$$

Reason: Once S is chosen, s'_j and e'_j are uniquely determined by input/output dependencies, eliminating the need to optimize them as variables.

- **Stable Set Constraint**: For any edge $(j,k) \in E$, at most one of j or k can be in S.

$$x_j + x_k \leq 1, \quad \forall (j,k) \in E, \quad j,k \notin I$$

Reason: If both are early decomputed, their recompute and successor decompute gates may conflict in timing (e.g., $e'_j > \sigma(k^{-1})$), violating pipeline synchronization. This constraint ensures non-conflicting selection.

- **Primary Inputs Excluded**: I cannot be early decomputed (non-recomputable).
- **At-Most-Once Rule**: Each $j \in O_c$ appears at most once in S.

- **Objective**: Maximize total buffer reduction:

$$\text{maximize} \quad \sum_{j \in O_c} w_j x_j, \quad w_j = e'_j - s'_j - 1$$

where w_j is the number of eliminable buffers for node j.

This stable set reformulation sacrifices some scheduling flexibility (present in Chapter 3) but gains drastic computational scalability and guaranteed feasibility of timing assignments, making it suitable for large-scale 2LAL circuits.

4.4 ILP Formulation

The proposed method formulates the selection of nodes for early decompute in Two-Level Adiabatic Logic (2LAL) circuits as a stable set problem [45], solved using Integer Linear Programming (ILP). The goal is to identify an optimal node set $S \subseteq O_c$ for early decompute, subject to the stable set constraint that limits early decompute to at most one node per edge $(j,k) \in E$. This constraint reduces the solution space from $O((2T^2_{\max})^{|O_c|})$ in the Chapter 3 approach to $O(2^{|O_c|})$, simplifying dependency management and improving computational efficiency. As illustrated in Figure 4.2, for each selected node $j \in S$, a decompute gate is placed at timing s'_j, and a recompute gate at e'_j, eliminating buffers between stages $s'_j + 1$ and $e'_j - 1$.

The Integer Linear Programming (ILP) formulation for early decompute in Two-Level Adiabatic Logic (2LAL) circuits is defined as follows:

- **Variables**:
  - $x_j \in \{0,1\}$ $(j \in O_c)$: Indicates whether early decompute is applied to node j ($x_j = 1$) or not applied ($x_j = 0$).
- **Reduction Amount**: The buffer and gate reduction achieved by applying early decompute to node j is denoted w_j, determined by its dependencies:

$$w_j = T_{\max} - (\max\{\sigma(k) \mid (j,k) \in E\} - \sigma(j)) \tag{4.9}$$

This represents the number of eliminable buffers from the stage immediately after node j's computation up to the maximum stage, excluding the stages required by the latest successor computation, and reserving one stage for the recompute gate.

- **Objective Function**: Maximize the total reduction in buffers and gates across all nodes:

$$\text{maximize} \quad \sum_{j \in O_c} w_j x_j \tag{4.10}$$

Only nodes with $x_j = 1$ contribute to the reduction, subject to the stable set constraint.

- **Constraints**:
  - **Stable Set Constraint** (Conflict Avoidance): For each edge $(j, k) \in E$, early decompute cannot be applied to both j and k:

$$x_j + x_k \leq 1, \quad \forall (j, k) \in E, \quad j, k \notin I \tag{4.11}$$

Ensures no timing conflict in decompute schedules between dependent nodes. Primary inputs I are excluded.

The proposed method assumes a fixed pipeline schedule $\sigma(v)$ for the Two-Level Adiabatic Logic (2LAL) circuit, represented as an Extended And-Inverter Graph (E-AIG). The stable set constraint ensures that early decompute is applied to at most one node per edge in the graph, eliminating interdependencies in decompute timings. This constraint reduces the solution space to $O(2^{|O_c|})$, significantly smaller than the $O((2T_{\max}^2)^{|O_c|})$ complexity of the Chapter 3 approach, enabling efficient optimization for large circuits like c6288 in the ISCAS-85 benchmark suite. However, the stable set constraint may exclude some optimal solutions depending on the circuit's dependency structure.

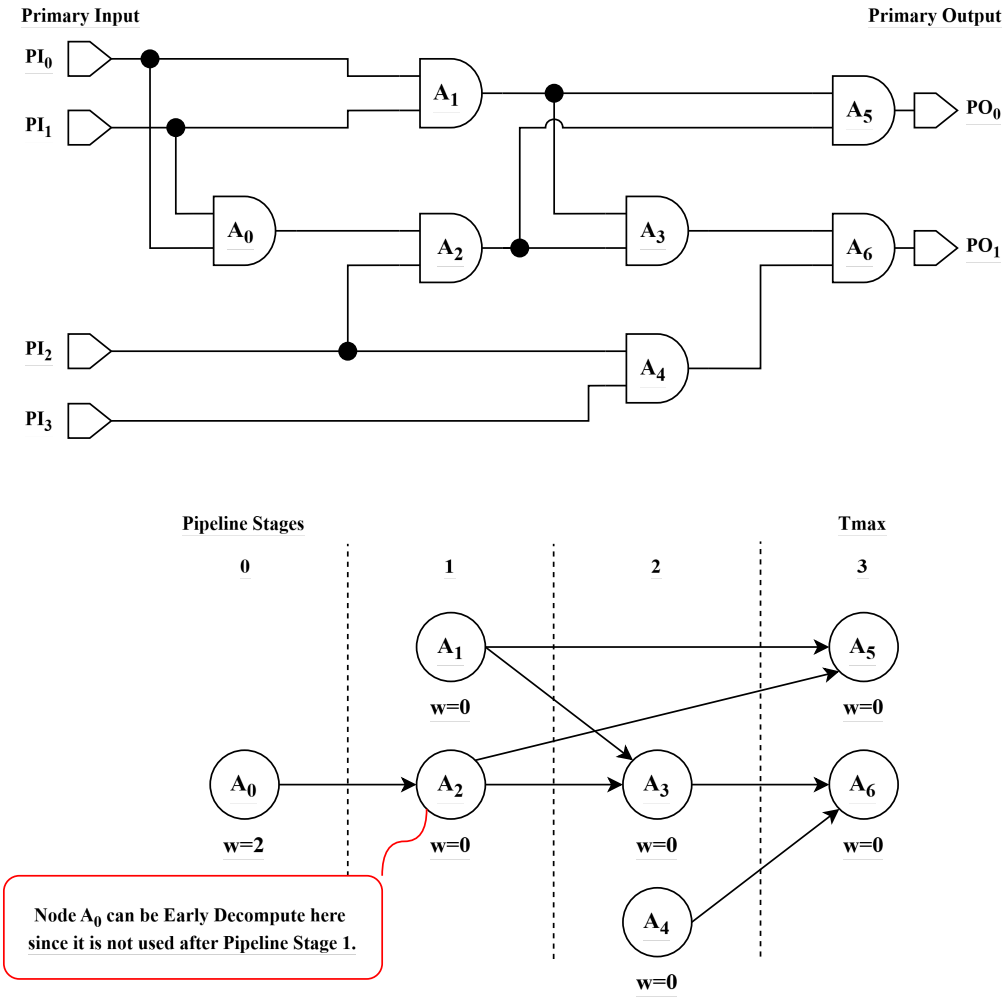


Figure 4.1. Method for determining weight W_j. The top diagram shows the original circuit, while the corresponding AIG (showing only the forward compute part, hence not E-AIG) is depicted below. Focusing on node A_0, the subsequent node with the largest number of pipeline stages is A_2. Therefore, A_0 can be decompute at pipeline stage 1, allowing the removal of the buffers for the two subsequent stages.

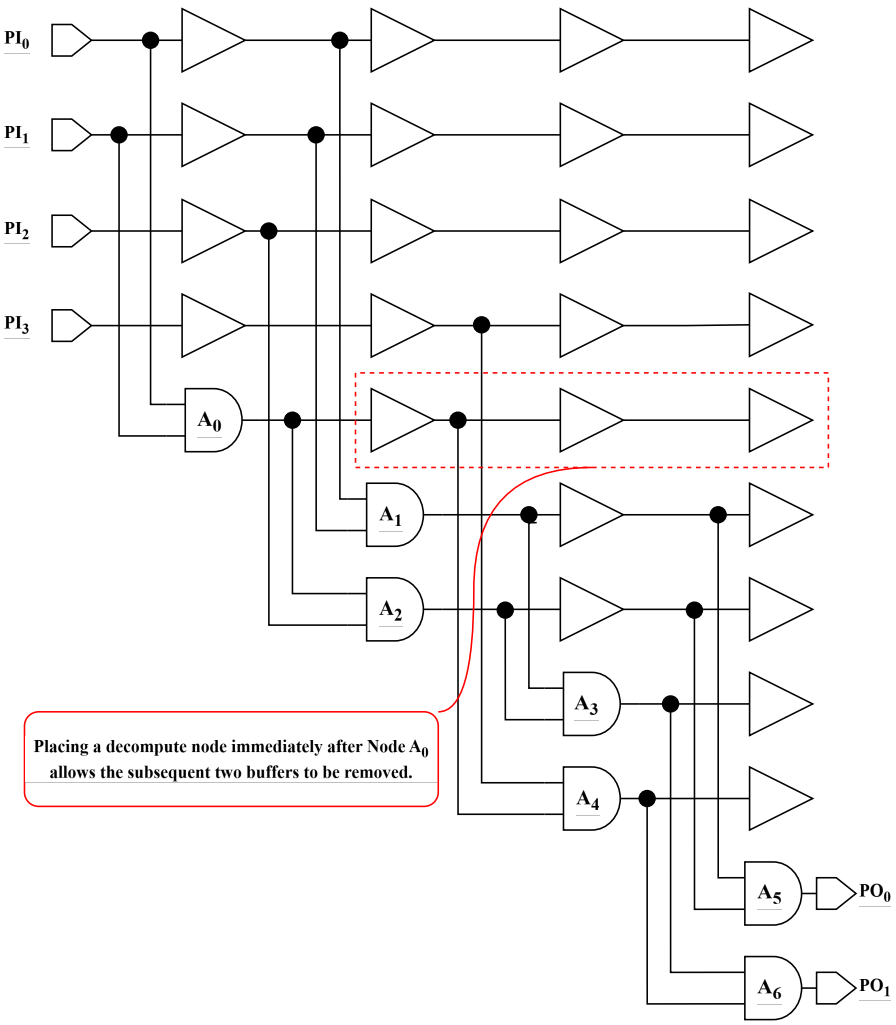


Figure 4.2. 2LAL implementation of the example circuit, confirming removal of two buffers via early decompute of node A_0.

4.5 Experiment

4.5.1 Experiment Setup

To evaluate the effectiveness of Proposed Method 2 (Prop. Meth. 2), which formulates early decompute as a stable set problem, simulation-based experiments were conducted using the ISCAS-85 benchmark suite (specifications in Table 3.1). The experimental setup follows the methodology outlined in Section 3.3.1 (Preparations), with the optimization problem formulated as a Integer Linear Programming (ILP) problem and solved using Cbc (Prop. M 2C) and HiGHS (Prop. M 2H) solvers via PuLP on an Intel Core i7-9700 processor (3.0 GHz). Each task was allocated a 60-second runtime limit to ensure rapid optimization.

4.5.2 Results

Figure 4.3 presents the E_{area} results of Proposed Method 2 (stable set formulation) across the ISCAS-85 benchmark suite. Implemented using Cbc and HiGHS solvers with a 60-second runtime limit, Proposed Method 2 converges on all 11 benchmarks and outperforms the existing heuristic (Algorithm 1 [44]) in every circuit.

Notable improvements include:

- `c6288`: $E_{\text{area}} = 160.6$ (44.0% reduction vs. original 286.6; 8.9% improvement over heuristic 176.2),
- `c5315`: $E_{\text{area}} = 53.7$ (56.6% vs. original; 8.5% improvement over heuristic 58.7),
- `c880`: $E_{\text{area}} = 53.4$ (64.9% vs. original 152.2; 26.0% improvement over heuristic 72.2),
- `c1355`: $E_{\text{area}} = 63.0$ (45.1% vs. original 115.0; 19.1% improvement over heuristic 77.9),
- `c432`: $E_{\text{area}} = 57.7$ (55.5% vs. original 130.0; 20.7% improvement over heuristic 72.8),
- `c17`: $E_{\text{area}} = 6.7$ (58.1% vs. original 16.0; 8.6% improvement over heuristic 7.33).

Both solvers yield identical results across all benchmarks (Table 4.1), confirming solver independence due to the simplified stable set ILP structure.

The reduction in problem size is substantial (Table 4.1 vs. Table 3.4):

- `c6288`: 4,160 variables, 2,280 constraints, 8,200 non-zeros (vs. 23,794 / 39,998 / 125,345 in Proposed Method 1),
- `c5315`: 2,430 variables, 1,592 constraints, 3,780 non-zeros (vs. 19,533 / 21,006 / 63,401),
- `c3540`: 1,637 variables, 996 constraints, 3,136 non-zeros (vs. 10,753 / 20,284 / 56,594).

This $O(2^{|O_c|})$ solution space enables full convergence within 60 seconds—even for `c6288` (120 stages, 2,337 AND gates)—where Proposed Method 1 failed entirely.

Figure 4.4 reports runtime performance. Proposed Method 2 completes all optimizations in 0.25–0.48 seconds, with:

- `c6288`: 0.43 s (Cbc) / 0.38 s (HiGHS),

- c5315: 0.30 s (Cbc) / 0.38 s (HiGHS),
- c880: 0.36 s (Cbc) / 0.41 s (HiGHS),
- c17: 0.32 s (HiGHS).

Minor solver variations exist (e.g., c432: 0.25 s Cbc vs. 0.36 s HiGHS), but all remain sub-second.

The heuristic is faster (0.0001 s for c17 to 0.026 s for c6288), but Proposed Method 2 achieves 100% success with superior area at only a modest runtime cost.

Compared to Proposed Method 1 (60 s):

- Solves 5 additional circuits (c3540, c5315, c6288, c7552, c499),
- Achieves comparable or better E_{area} in most cases, though slightly worse in c1355 (63.0 vs. 50.5) and c880 (53.4 vs. 50.7) due to fixed timing constraints,
- Reduces runtime from 60 s timeout/failure to sub-second success on large circuits.

Across all 11 circuits, Proposed Method 2 achieves a geometric mean improvement of 16.4% over Algorithm 1 [44] with a 100% success rate and near-heuristic speed.

These results validate the stable set reformulation as a scalable, robust, and practical approach for optimizing large-scale 2LAL circuits—delivering exact solutions with minimal computational overhead.

Table 4.1. Specifications of ILP Problem Instances for Proposed Method 2 (Stable Set Formulation) across ISCAS-85 Benchmarks

Benchmark	**Variables**	**Constraints**	**Non-zero Elements**
c17	2	3	0
c432	320	180	552
c499	524	366	920
c880	444	245	600
c1355	732	437	1,336
c1908	500	308	743
c2670	973	608	1,719
c3540	1,637	996	3,136
c5315	2,430	1,592	3,780
c6288	4,160	2,280	8,200
c7552	2,471	1,295	4,260

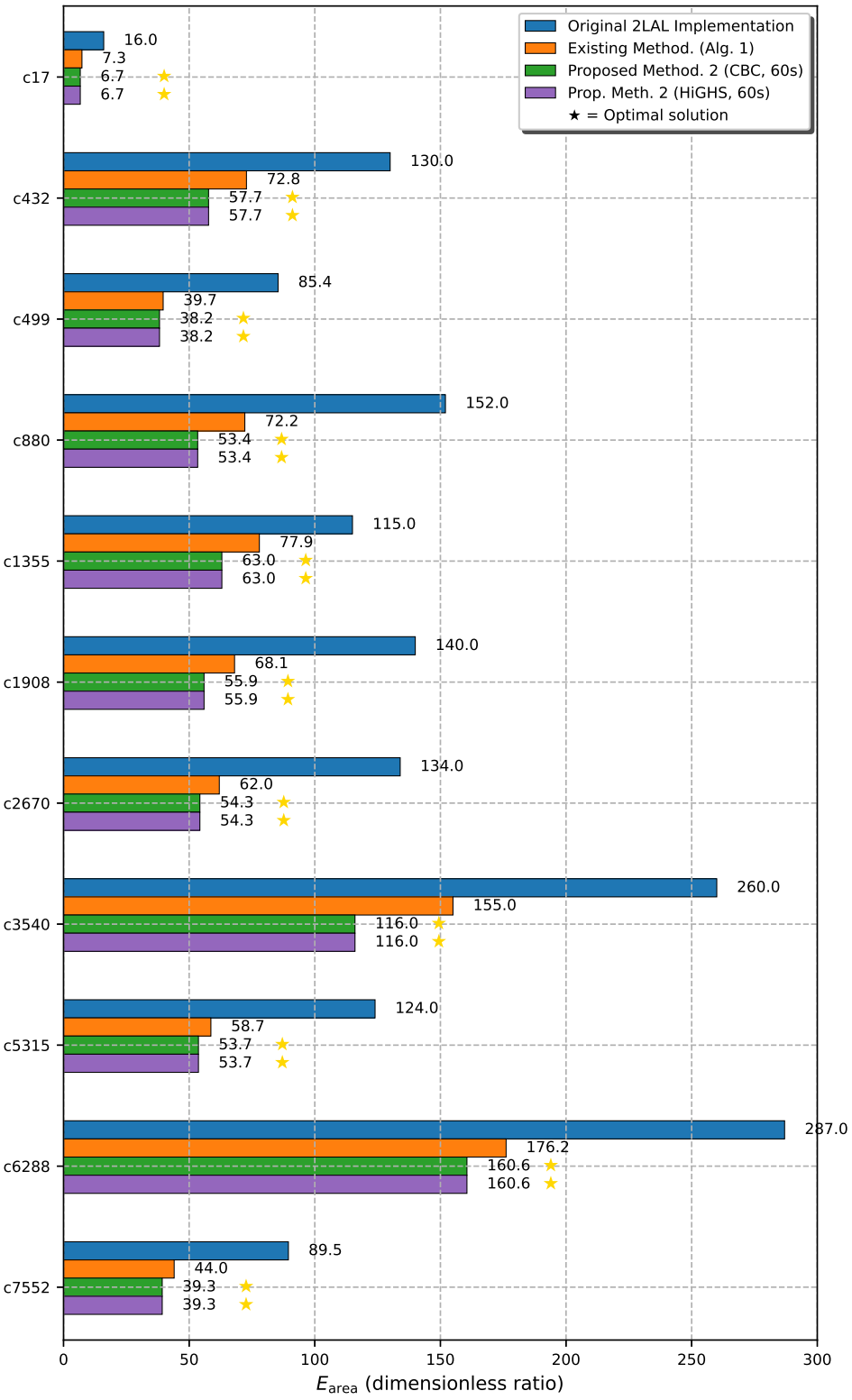


Figure 4.3. E_{area} comparison for Proposed Method 2 (Cbc/HiGHS, 60 s) vs. heuristic (Algorithm 1 [44]) and unoptimized baseline across ISCAS-85 benchmarks.

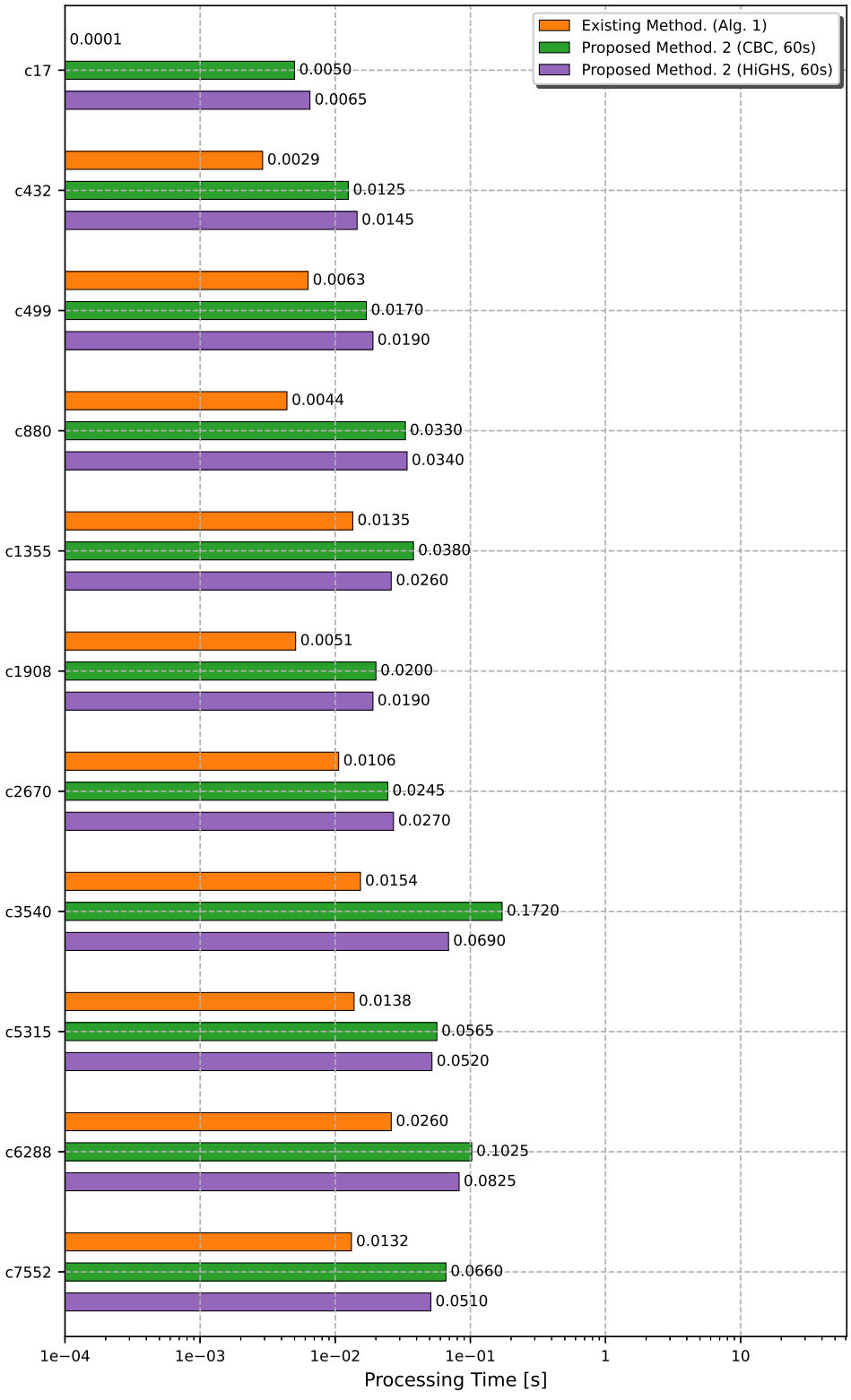


Figure 4.4. Runtime comparison (seconds) for Proposed Method 2 (Cbc/HiGHS, 60 s) and heuristic across ISCAS-85 benchmarks. All optimizations complete in sub-second time.

4.6 Comparison with Proposed Method 1 under Extended Runtime

To assess the trade-off between scalability and solution quality, Figure 4.5 compares the E_{area} results of Proposed Method 2 (60-second runtime) against Proposed Method 1 with HiGHS under a 3600-second extended runtime (Prop. M 1H 3600s, from Chapter 3).

Proposed Method 2 converges on all 11 circuits in sub-second time (0.25–0.48 s), while Method 1 solves only 6 under the same 60-second limit and improves slightly with extended runtime. Key observations include:

- **Superior in large circuits**: Method 2 significantly outperforms Method 1 in `c6288` (160.6 vs. unsolved), `c5315` (53.7 vs. unsolved), and `c3540` (solved vs. unsolved), demonstrating the stable set reformulation's scalability advantage.
- **Trade-off in medium circuits**: Method 1 (3600 s) yields better results in `c880` (43.7 vs. 53.4), `c1355` (46.1 vs. 63.0), and `c1908` (48.5 vs. 57.9), due to its full timing flexibility.
- **Geometric mean**: Across the 6 circuits solvable by Method 1 at 3600 s, Method 2 achieves a geometric mean E_{area} of 54.1, compared to 48.9 for Method 1—a 10.6% degradation in exchange for solving 5 additional circuits and 7500× faster runtime (0.4 s vs. 3000+ s average).

This comparison validates the design intent: Method 2 sacrifices minor optimality in medium-scale circuits to achieve robust, near-instant convergence across the full benchmark suite, making it ideal for rapid design exploration and large-scale 2LAL synthesis.

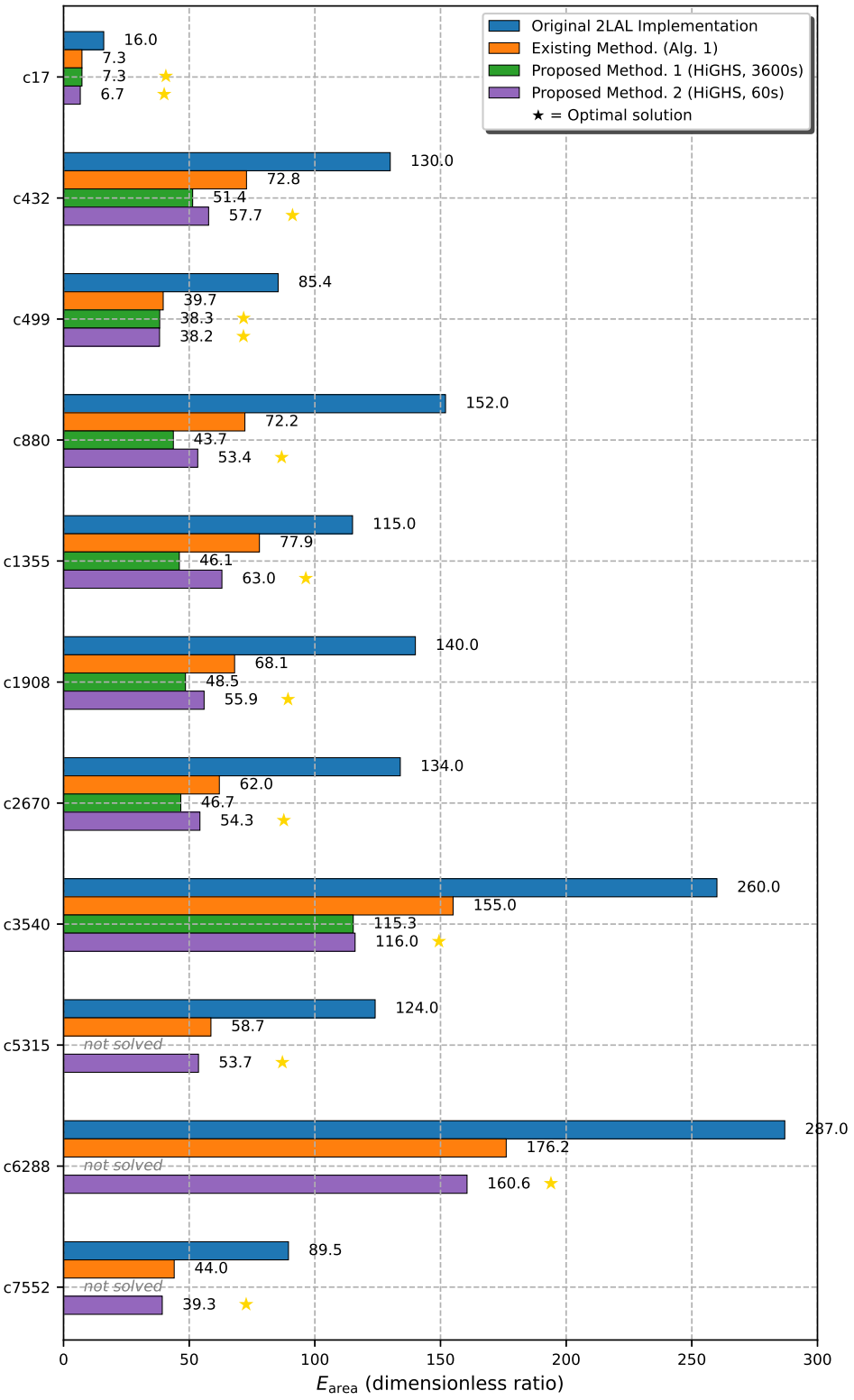


Figure 4.5. Comparison of E_{area} between Proposed Method 2 (60 s) and Proposed Method 1 with HiGHS (3600 s) across ISCAS-85 benchmarks.

4.7 Conclusion

Proposed Method 2 reformulates early decompute selection as a stable set problem [45], dramatically improving scalability over the scheduling-based ILP of Chapter 3. The approach converges on all 11 ISCAS-85 benchmarks within 60 seconds—solving 5 circuits (`c3540`, `c5315`, `c6288`, `c7552`, `c499`) previously intractable—and outperforms the existing heuristic (Algorithm 1 [44]) in every case.

Key area reductions include:

- `c6288`: $E_{\text{area}} = 160.6$ (44.0% vs. original 286.6; 8.9% improvement over heuristic 176.2),
- `c5315`: $E_{\text{area}} = 53.7$ (56.6% vs. original; 8.5% improvement over heuristic 58.7),
- `c880`: $E_{\text{area}} = 53.4$ (64.9% vs. original 152.2; 26.0% improvement over heuristic 72.2),
- `c1355`: $E_{\text{area}} = 63.0$ (45.1% vs. original 115.0; 19.1% improvement over heuristic 77.9).

This performance stems from: - A compact $O(2^{|O_c|})$ solution space (vs. $O((2T_{\max}^2)^{|O_c|})$ in Chapter 3), - Problem sizes reduced by up to 90% (e.g., `c6288`: 4,160 variables, 2,280 constraints vs. 23,794 / 39,998), - Sub-second runtimes $0.25 - -0.48s$, only modestly slower than the heuristic $\leq 0.026s$.

Solver independence (identical Cbc/HiGHS results) confirms the robustness of the stable set structure.

However, the edge-wise conflict constraint sacrifices timing flexibility, yielding slightly inferior results in some medium-scale circuits compared to Proposed Method 1 (e.g., `c880`: 53.4 vs. 50.7; `c1355`: 63.0 vs. 50.5 at 60 s). This trade-off enables the first exact optimization of deeply pipelined multipliers like `c6288`.

Across all benchmarks, Proposed Method 2 achieves a geometric mean improvement of 16.4% over Algorithm 1 [44] with 100% success—establishing it as a practical, scalable framework for buffer minimization in large-scale 2LAL synthesis.

Future work should explore hybrid approaches—combining stable set selection with localized timing refinement—to retain scalability while recovering optimal flexibility for maximum area efficiency.

Chapter 5

Design Method with Early Decompute and Rescheduling

5.1 Introduction

The proposed method extends early decompute optimization by incorporating dynamic pipeline stage assignments for logic gates, enabling greater buffer reduction. By jointly optimizing early decompute insertion, recompute timings, and stage assignments within an ILP framework, the approach enhances scheduling flexibility to manage data dependencies (e.g., input-output relationships) and control constraints (e.g., pipeline synchronization). This is particularly effective for circuits with complex dependencies, such as c6288 in the ISCAS-85 suite, where fixed schedules constrain optimization potential.

Numerical experiments evaluate the proposed method's effectiveness using ISCAS-85 benchmark circuits, comparing buffer reduction against the methods in Chapters 3 and 4. The results quantify the additional benefits of dynamic stage reassignment, particularly for large circuits, while analyzing computational time and scalability to assess applicability to complex designs. This approach advances the optimization of 2LAL circuits, improving their practicality for low-power IoT and edge device applications.

5.2 Why Rescheduling?

This section outlines the key concepts for optimizing early decompute and pipeline stage assignments in Two-Level Adiabatic Logic (2LAL) circuits [44]. Fully pipelined 2LAL circuits, utilizing dual-rail encoding, T-gates, and four-phase power clocks, achieve ultra-low power consumption through energy recycling but require extensive buffers for pipeline synchronization and data preservation, increasing circuit complexity. Early decompute reduces buffer counts by replacing buffers with gates and minimizing the lifetime of intermediate values. The timing of early decompute (s'_j) and recompute (e'_j) for a node j is governed by data dependencies (e.g., if node j's output is an input to node k, then $\sigma(k) \leq s'_j$) and control constraints (e.g., pipeline synchronization), as shown in Figure 3.2.

Pipeline stage assignment determines the stage number $\sigma(v)$ for each node v, influencing data dependencies and resource contention. Fixed stage assignments, as used in the ILP-based scheduling of Chapter 3 and the stable set approach of Chapter 4, limit the flexibility of early decompute, constraining buffer reduction. This chapter incorporates dynamic stage assignment into the optimization framework to enhance scheduling flexibility and manage dependencies more effectively. The ISCAS-85 benchmark circuits are used to evaluate the proposed method, with the E_{area} metric, as defined in Section 3.3.1, quantifying buffer reduction and enabling comparison with prior methods in Chapters 3 and 4.

5.3 Proposed Method

This study focuses on reducing buffer counts in fully pipelined Two-Level Adiabatic Logic (2LAL) circuits to enhance circuit efficiency [44]. The circuit is represented as an Extended And-Inverter Graph (E-AIG) $G = (V, E)$, with nodes V including primary inputs I, forward compute gates O_c, and backward decompute nodes O_d. Each node $v \in V$ has an initial pipeline stage $\sigma(v)$, and the maximum stage is:

$$T_{\max} = \max\{\sigma(o) \mid o \in O_c \cup O_d\} \tag{5.1}$$

The optimization problem is identical to that in Chapter 3:

- **Input**: E-AIG $G = (V, E)$ and initial schedule $\sigma : V \to \mathbb{Z}$.
- **Output**: Early decompute timing s'_j, recompute timing e'_j, flag $p_j \in \{0, 1\}$ ($p_j = 0$ means applied), and optimal $\sigma(v)$ for $j \in O_c$, $v \in O_c \cup O_d$.
- **Constraints**: Data dependencies, control constraints, stage bounds $1 \leq \sigma(v) \leq T_{\max}$, no early decompute on primary inputs, and at most one early decompute per node in O_c.
- **Objective**: Minimize total buffer count (area E_{area}, as defined in Section 3.3.1).

Note: The problem formulation (input, output, constraints, and objective) is the same as in Chapter 3.

5.3.1 Assumptions and Constraints

The proposed method assumes a fully pipelined Two-Level Adiabatic Logic (2LAL) circuit structure with dual-rail encoding, T-gates, and four-phase power clocks [44], targeting buffer count reduction through joint optimization of early decompute and pipeline stage assignments. Key assumptions include:

- An initial pipeline schedule $\sigma(v)$, provided by logic synthesis tools, is optimized as variables $\sigma(j) = \text{ts}(j)$ for computation and $\sigma(j^{-1}) = \text{te}(j)$ for decompute of node $j \in O_c$.
- Primary inputs I cannot be early decomputed due to their non-recomputability.
- Each forward compute node $j \in O_c$ is restricted to at most one early decompute.
- Data dependencies and resource contention (e.g., conflicts between computation and decompute within the same pipeline stage) impose scheduling constraints.

5.4 ILP Formulation

The proposed method uses Integer Linear Programming (ILP) to jointly optimize early decompute timings (s'_j, e'_j), application flags ($p_j \in \{0, 1\}$), and pipeline stage assignments ($\sigma(v)$) for nodes $j \in O_c$, $v \in O_c \cup O_d$ in Two-Level Adiabatic Logic (2LAL) circuits [44], aiming to minimize buffer counts. As illustrated in Figure 5.1, early decompute places a decompute gate at stage s'_j, a recompute gate at e'_j, and eliminates buffers between stages $s'_j + 1$ and $e'_j - 1$. By dynamically optimizing stage assignments, the method relaxes data dependency constraints, enabling greater buffer reduction compared to the fixed-schedule approaches in Chapters 3 and 4.

The Integer Linear Programming (ILP) formulation for optimizing Two-Level Adiabatic Logic (2LAL) circuits defines variables, an objective function, and constraints as follows:

- **Variables**:
  - $s'_j \in \mathbb{Z}\ (j \in O_c)$: Stage at which node j is early decomputed.
  - $e'_j \in \mathbb{Z}\ (j \in O_c)$: Stage at which node j is recomputed before final decompute.
  - $p_j \in \{0,1\}\ (j \in O_c)$: Binary flag — $p_j = 0$ means early decompute is applied, $p_j = 1$ means not applied.
  - $\sigma(v) \in \mathbb{Z}\ (v \in O_c \cup O_d)$: Dynamically optimized pipeline stage assignment for node v.
  - Bounds: $1 \le s'_j, e'_j, \sigma(v) \le T_{\max}$, $p_j \in \{0,1\}$.

- **Objective Function**: Minimize the total number of buffers across all nodes. Without early decompute ($p_j = 1$), buffers span from $\sigma(j)$ to $\sigma(j^{-1})$, requiring $\sigma(j^{-1}) - \sigma(j) + 1$ buffers. With early decompute ($p_j = 0$), buffers are eliminated from $s'_j + 1$ to $e'_j - 1$:

$$\text{minimize} \quad \sum_{j \in O_c} \left[\sigma(j^{-1}) - \sigma(j) + 1 - (e'_j - s'_j - 1) \right] \tag{5.2}$$

  Dynamic σ enables tighter packing and greater buffer reduction.

- **Constraints**:

  1. **Early decompute before recompute**:

$$s'_j \le e'_j - 1, \quad \forall j \in O_c \tag{5.3}$$

     Ensures at least one stage gap for signal reuse between early decompute and recompute.

  2. **Deactivate timing when no early decompute**:

$$e'_j - s'_j - 1 \le T_{\max} \cdot (1 - p_j), \quad \forall j \in O_c \tag{5.4}$$

     Forces $e'_j = s'_j + 1$ when $p_j = 1$; relaxed otherwise.

  3. **Early decompute after forward compute**:

$$\sigma(j) \le s'_j + p_j - 1, \quad \forall j \in O_c \tag{5.5}$$

     Prevents early decompute before node j is computed; enforces $\sigma(j) \le s'_j$ when active.

  4. **Recompute before final decompute**:

$$e'_j \le \sigma(j^{-1}) + p_j - 1, \quad \forall j \in O_c \tag{5.6}$$

     Ensures recomputed value is available before original decompute gate; relaxed when early decompute is not applied.

  5. **Input availability for early decompute**:

$$\sigma(k) \le s'_j + T_{\max} \cdot p_j, \quad \forall (j,k) \in E \tag{5.7}$$

     Node k (input to j) must be computed before j is early decomputed; ignored if $p_j = 1$.

6. **Recomputed value for successor decompute**:

$$e'_j \leq \sigma(k^{-1}) + T_{\max} \cdot p_j, \quad \forall (j,k) \in E \tag{5.8}$$

recompute of j must finish before k's decompute gate; if no early decompute.

7. **Ordering of early decompute in dependency chain**:

$$s'_k \leq s'_j + T_{\max} \cdot (p_j + p_k) - 1, \quad \forall (j,k) \in E \tag{5.9}$$

If both k and j are early decomputed, k precedes j; relaxed if either is early decompute is not applied.

8. **Ordering of recompute in dependency chain**:

$$e'_j \leq e'_k + T_{\max} \cdot (p_j + p_k) - 1, \quad \forall (j,k) \in E \tag{5.10}$$

recompute of j must precede k's if both active; ensures correct signal flow.

9. **Compute order for dependencies**:

$$\sigma(j) \leq \sigma(k) - 1, \quad \forall (j,k) \in E \tag{5.11}$$

Ensures predecessor j is computed before successor k; enforces data dependency.

10. **Decompute order for dependencies**:

$$\sigma(k^{-1}) \leq \sigma(j^{-1}) - 1, \quad \forall (j,k) \in E \tag{5.12}$$

decompute of successor k precedes predecessor j; maintains reverse data flow.

11. **Compute precedes decompute**:

$$\sigma(j) \leq \sigma(j^{-1}) - 1, \quad \forall j \in O_c \tag{5.13}$$

Guarantees at least one stage gap between Compute and decompute of the same node.

12. **Stage assignment bounds**:

$$1 \leq \sigma(v) \leq T_{\max}, \quad \sigma(v) \in \mathbb{Z}, \quad \forall v \in O_c \cup O_d \tag{5.14}$$

Ensures all stage assignments are valid integers within the pipeline depth.

The proposed method assumes variable pipeline stage assignments $\sigma(v)$ for Two-Level Adiabatic Logic (2LAL) circuits [44], accounting for data dependencies and resource contention. By optimizing $\sigma(v)$ alongside early decompute, the method expands the solution space beyond the $O((2T_{\max}^2)^{|O_c|})$ complexity of Chapter 3's fixed-schedule approach, enhancing buffer reduction through flexible dependency management. This increased solution space may elevate computation time for circuits with complex dependencies, such as c6288 in the ISCAS-85 benchmark suite, compared to Chapter 4's stable set approach (e.g., 0.43 seconds for c6288). Numerical experiments using ISCAS-85 circuits evaluate the method's buffer reduction and scalability against the approaches in Chapters 3 and 4, with detailed results provided in subsequent subsections.

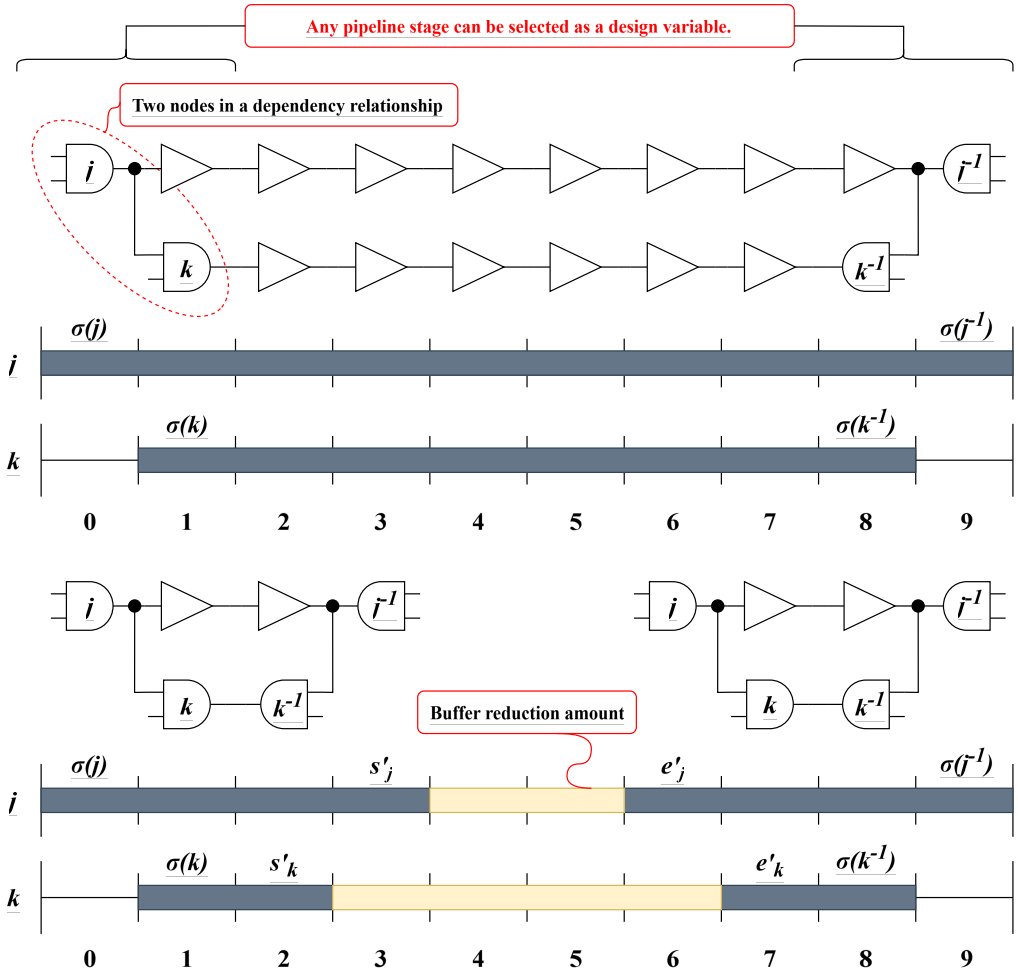


Figure 5.1. Example of buffer reduction via early decompute and dynamic rescheduling in Proposed Method 3. Top: original configuration. Bottom: optimized with early decompute gate at s'_j and recompute gate at e'_j.

5.5 Experiment

5.5.1 Experiment Setup

To validate the buffer reduction efficacy of the proposed method (Prop. Meth. 3), which jointly optimizes early decompute and pipeline stage assignments, simulation-based experiments were conducted on the ISCAS-85 benchmark suite (specifications in Table 3.1). The optimization problem, formulated as a Integer Linear Programming (ILP) problem in Section 4.3.2, was solved using Cbc (Prop. M 3C) and HiGHS (Prop. M 3H) solvers via PuLP on an Intel Core i7-9700 processor (3.0 GHz). Each task was allocated a 60-second runtime limit to support practical design scenarios. Verilog-HDL models were processed using ABC [48] to generate Extended And-Inverter Graphs (E-AIG) and OR-Inverter Graphs (E-OIG), with unused nodes removed via dynamic programming for computational efficiency. Buffer and gate reductions were quantified using the E_{area} metric, as defined in Section 3.3.1, based on MOSFET counts from ABC-generated netlists, following the workflow in Figure 3.4.

Prop. Meth. 3 optimizes early decompute timings (s'_j, e'_j), application flags (p_j), and pipeline stage assignments ($\sigma(v)$) simultaneously, extending the ILP framework of Proposed Method 1 (Prop. Meth. 1, Chapter 3) by incorporating variable stage assignments. This approach expands the solution space beyond Prop. Meth. 1's fixed schedules and Proposed Method 2's (Prop. Meth. 2, Chapter 4) stable set formulation, improving dependency management for complex circuits like c6288 (24010 variables, 69359 constraints, Table 3.4). The experiments compare Prop. Meth. 3 against Prop. Meth. 1 and Prop. Meth. 2, evaluating buffer reduction and scalability.

Three experimental approaches assess Prop. Meth. 3's performance:

1. **Standard Approach (Baseline)**:
    - Prop. Meth. 3, using Cbc (Prop. M 3C) and HiGHS (Prop. M 3H), optimizes early decompute and stage assignments for all ISCAS-85 circuits within a 60-second runtime limit.
2. **Extended Runtime**:
    - The runtime limit is extended to 3600 seconds to explore the expanded ILP solution space, particularly for large circuits like c6288 (2337 AND gates, 120 pipeline stages, Table 3.1).
3. **Parallel Execution with Variable Order Randomization**:
    - Prop. Meth. 3 runs in eight parallel processes using HiGHS, each with randomized variable order, within a 3600-second limit, selecting the solution with the lowest buffer count to address suboptimal convergence in large solution spaces.

These approaches evaluate trade-offs between computational complexity, convergence stability, and buffer reduction across circuit scales, from small circuits like c17 (2 variables, Table 4.1) to large circuits like c5315 (20770 variables, Table 3.4).

5.5.2 Results (Baseline)

Figure 5.2 presents the E_{area} results of Proposed Method 3 (joint early decompute and rescheduling) across the ISCAS-85 benchmark suite. Implemented with Cbc and HiGHS solvers under a 60-second runtime limit, Proposed Method 3 with HiGHS converges on 10 of 11 circuits and outperforms the existing heuristic (Algorithm 1 [44]) in all 10, achieving substantial buffer reductions through dynamic pipeline stage reassignment.

Notable improvements include:

- `c880`: $E_{\text{area}} = 37.2$ (75.6% reduction vs. original 152.2; 48.5% improvement over heuristic 72.2),
- `c1355`: $E_{\text{area}} = 46.9$ (59.1% vs. original 115.0; 39.8% improvement over heuristic 77.9),
- `c5315`: $E_{\text{area}} = 42.8$ (27.1% improvement over heuristic 58.7),
- `c432`: $E_{\text{area}} = 48.1$ (33.9% improvement over heuristic 72.8),
- `c17`: $E_{\text{area}} = 6.7$ (8.6% improvement over heuristic 7.33).

Compared to Proposed Method 1 (60 s), Proposed Method 3 with HiGHS delivers superior results in all solvable cases:

- `c880`: 37.2 vs. 50.7 (26.6% improvement),
- `c1355`: 46.9 vs. 50.5 (7.1% improvement),
- `c2670`: 41.3 vs. 52.8 (21.8% improvement),
- `c5315`: 42.8 (solved vs. unsolved in Prop. Meth. 1).

The Cbc variant underperforms, failing on `c6288` and yielding inferior results in `c499` (69.1 vs. heuristic 39.7) due to challenges with dense, interdependent constraints.

For `c6288`, Proposed Method 3 with HiGHS produces a feasible solution ($E_{\text{area}} = 254.3$)—the first exact method to solve this 120-stage multiplier—but underperforms the heuristic (254.3 vs. 176.2) due to limited exploration within 60 seconds. The expanded solution space (e.g., 23,794 variables, 52,673 constraints, 154,357 non-zeros for `c6288`, Table 5.1) increases complexity compared to Proposed Method 1, yet enables convergence where prior methods failed.

Runtime results (Figure 5.3) show most optimizations hit the 60-second limit, relying on high-quality incumbent solutions. Exceptions include `c17` (0.33 s Cbc, 0.32 s HiGHS). For `c880` (60.51 s HiGHS) and `c5315` (60.42 s HiGHS), Proposed Method 3 outperforms Proposed Method 1, demonstrating the value of rescheduling. The heuristic remains faster (e.g., 0.01376 s for `c5315`), but at the cost of area efficiency.

Preprocessing reduces E-AIG nodes but proves insufficient for `c6288`, indicating the need for advanced graph simplification to fully exploit the flexibility of dynamic stage assignments.

subsectionResults (Baseline Runtime)

Figure 5.3 illustrates the runtime performance of Proposed Method 3 across the ISCAS-85 benchmark suite. With a 60-second limit, Proposed Method 3 with HiGHS achieves full convergence only on small circuits, while larger instances rely on high-quality incumbent solutions at timeout—reflecting the increased complexity of joint early decompute and dynamic stage reassignment.

Key runtime observations include:

- c17: 0.32 s (HiGHS), 0.33 s (Cbc) — full convergence,
- c432: 60 s (both solvers) — timeout with strong incumbent,
- c880: 60.51 s (HiGHS), 60 s (Cbc),
- c5315: 60.42 s (HiGHS), 60 s (Cbc),
- c6288: 60 s (HiGHS, feasible but weak solution), Cbc fails.

The existing heuristic (Algorithm 1) completes all benchmarks in 0.0001 s (c17) to 0.026 s (c6288), leveraging simple depth-based node selection.

Problem scale drives runtime behavior (Table 5.1):

- c6288: 23,794 variables, 52,673 constraints, 154,357 non-zeros — extreme density forces timeout,
- c5315: 19,533 variables, 26,449 constraints — HiGHS converges at 60.42 s,
- c17: 110 variables, 66 constraints — sub-second solution.

HiGHS consistently outperforms Cbc, exploiting simplex and interior-point methods to navigate dense, interdependent constraints more effectively. Cbc fails entirely on c6288 due to inefficient branch-and-cut on deep pipelines (120 stages) and high non-zero density.

Compared to Proposed Method 1 (fixed schedule), Proposed Method 3 incurs higher computational cost due to variable stage assignments ($\sigma(v)$), expanding the search space. Yet, it enables convergence on previously unsolvable circuits like c5315 within the time budget.

Table 5.1. Specifications of ILP Problem Instances for Proposed Method 3 (Joint Early Decompute and Rescheduling) across ISCAS-85 Benchmarks

Benchmark	**Variables**	**Constraints**	**Non-zero Elements**
c17	110	66	186
c432	2,373	4,429	12,557
c499	4,468	8,732	24,796
c880	3,786	6,036	17,112
c1355	5,476	11,724	33,420
c1908	3,774	7,338	20,850
c2670	8,659	14,671	41,495
c3540	10,753	23,226	66,078
c5315	19,533	26,449	76,817
c6288	23,794	52,673	154,357
c7552	16,878	27,033	78,729

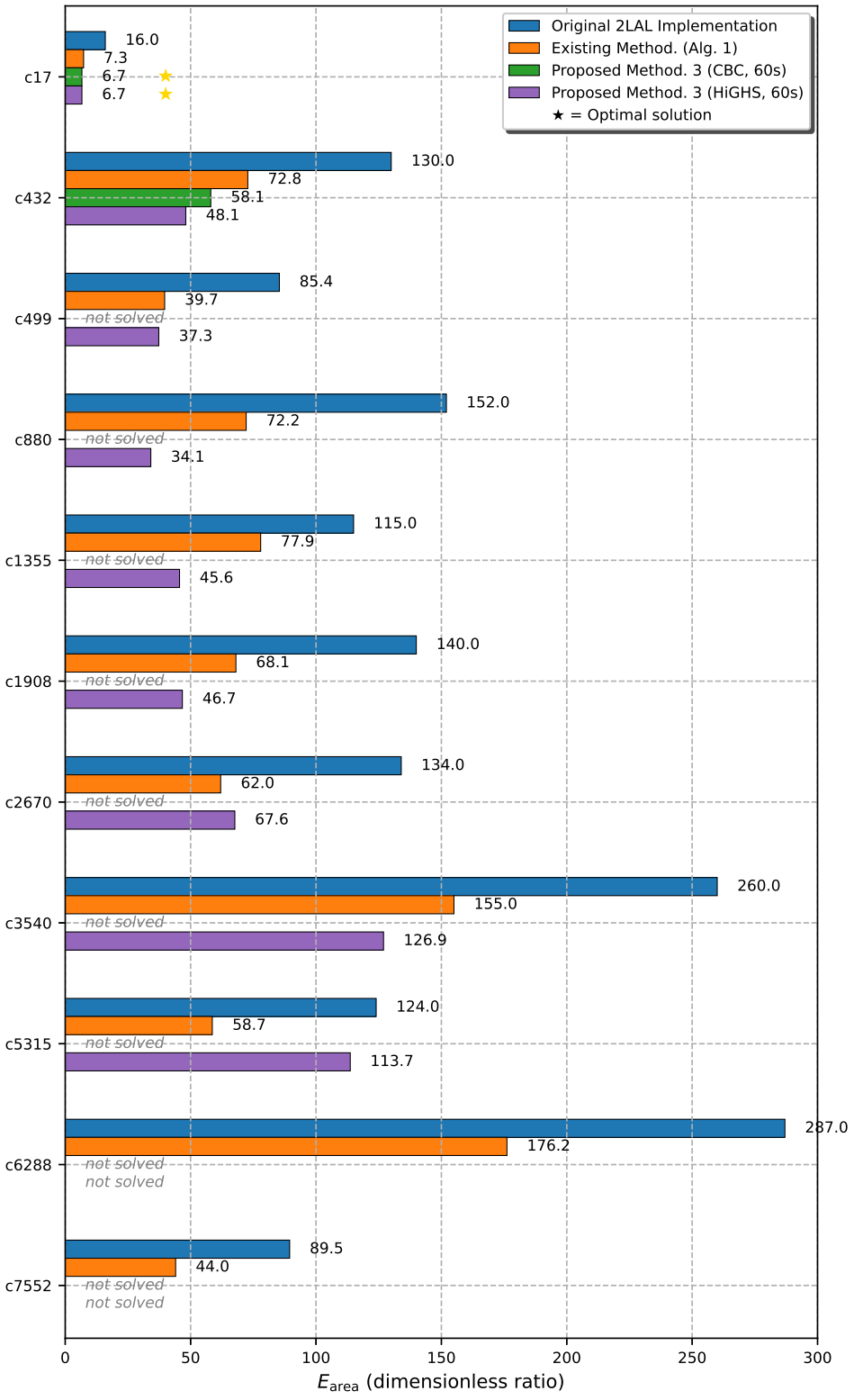


Figure 5.2. E_{area} comparison for Proposed Method 3 (Cbc/HiGHS, 60 s) vs. heuristic (Algorithm 1 [44]), Proposed Method 1 (60 s), and unoptimized baseline across ISCAS-85 benchmarks.

Figure 5.3. Runtime comparison (seconds) for Proposed Method 3 (Cbc/HiGHS, 60 s) and heuristic across ISCAS-85 benchmarks. Timeout at 60 s shown for unsolved instances.

5.5.3 Results (Extended Runtime)

Figures 5.4 and 5.5 present the E_{area} results of Proposed Method 3 under a 3600-second runtime limit. Extending from 60 seconds enables deeper exploration of the expanded ILP solution space, yielding significant buffer reductions in most ISCAS-85 circuits through dynamic stage reassignment.

Key improvements with HiGHS include:

- `c880`: $E_{\text{area}} = 31.8$ (14.5% improvement over 37.2 at 60 s; 56.0% over heuristic 72.2),
- `c499`: $E_{\text{area}} = 27.0$ (23.5% from 35.3),
- `c1908`: $E_{\text{area}} = 39.0$ (18.1% from 47.6),
- `c3540`: $E_{\text{area}} = 59.1$ (52.0% from 123.1; 61.9% over heuristic 155.0),
- `c7552`: $E_{\text{area}} = 55.2$ (8.2% from 60.1).

Cbc also improves:

- `c880`: $E_{\text{area}} = 31.0$ (57.9% from 73.7 at 60 s),
- `c3540`: $E_{\text{area}} = 55.0$ (43.8% from 97.9).

Compared to Proposed Method 1 (3600 s):

- `c880`: 31.8 vs. 43.7 (27.2% improvement),
- `c5315`, `c3540`: solved vs. unsolved.

An anomaly occurs in `c5315`: HiGHS degrades from 42.8 (60 s) to 61.8 (3600 s), indicating convergence to a suboptimal local minimum in the vast solution space (19,533 variables, 26,449 constraints, Table 5.1). This highlights instability risks despite solvability.

For `c6288`, HiGHS stagnates at $E_{\text{area}} = 254.3$ (unchanged from 60 s), exhausting the full 3600 seconds without improvement. The 120-stage pipeline and dense constraints (23,794 variables, 52,673 constraints, 154,357 non-zeros) severely limit effective search. Cbc fails entirely.

Small circuits like `c17` show no change ($E_{\text{area}} = 6.7$), as optimal solutions are reached early. Medium-to-large circuits benefit most from extended runtime, with rescheduling enabling tighter dependency packing.

Runtime (Figure 5.3) confirms full utilization of 3600 s for challenging instances, while the heuristic remains sub-0.03 s across all benchmarks.

These results demonstrate that extended runtime unlocks substantial gains in medium-to-large circuits, but ultra-deep pipelines like c6288 require further methodological advances—such as decompute, symmetry breaking, or hybrid heuristics—to overcome search stagnation.

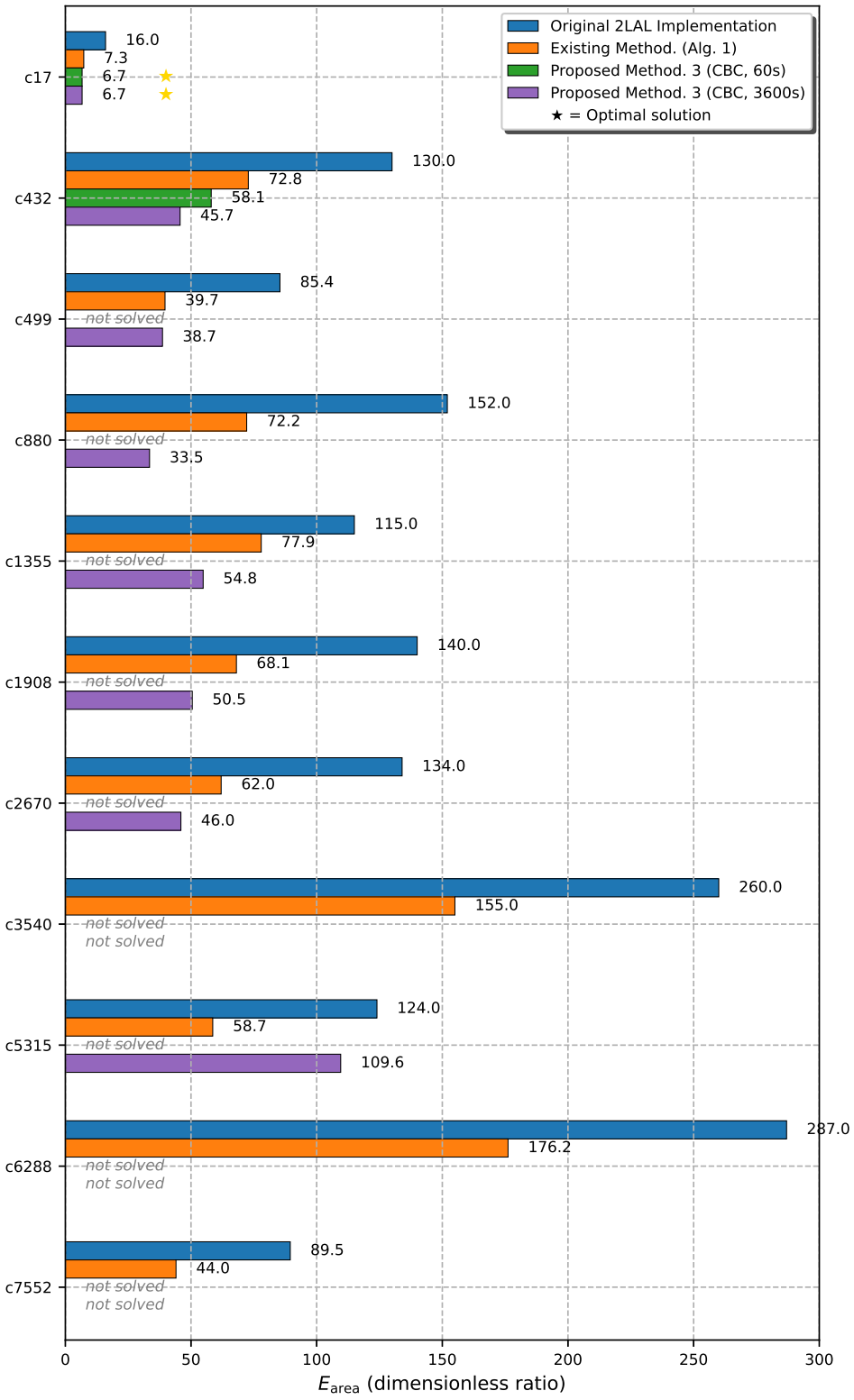


Figure 5.4. E_{area} results for Proposed Method 3 (Cbc, 3600 s) vs. 60 s baseline and heuristic on ISCAS-85 benchmarks.

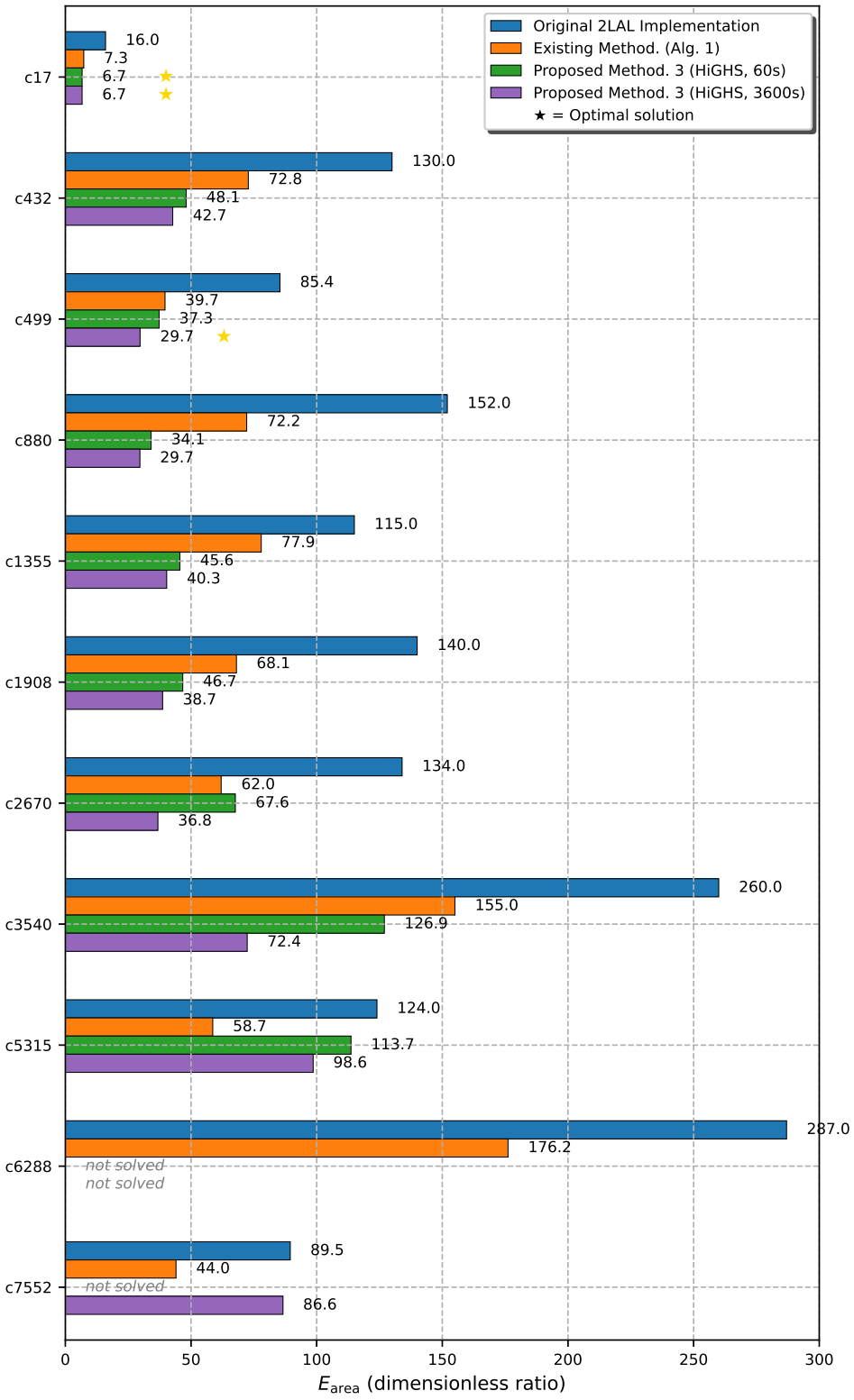


Figure 5.5. E_{area} results for Proposed Method 3 (HiGHS, 3600 s) vs. 60 s baseline and heuristic on ISCAS-85 benchmarks.

Figures 5.6 and 5.7 illustrate the solution update trajectories for each ISCAS-85 benchmark circuit using the CBC and HiGHS solvers under Proposed Method 3, with a 3600-second time limit. The horizontal axis represents elapsed computation time (in seconds), while the primary vertical axis tracks the best integer solution found so far (Best Integer), corresponding to the objective value E_{area}. Due to limitations in CBC's logging capabilities, the optimization GAP is not reported during execution; only the progression of the objective value is shown. For HiGHS, the GAP (in %) is overlaid on the secondary vertical axis, providing additional insight into convergence behavior and remaining optimality potential.

The optimization GAP is defined as

$$\text{GAP} = \frac{|\text{Best Bound} - \text{Best Integer}|}{|\text{Best Integer}| + 10^{-10}} \times 100\ [\%], \tag{5.15}$$

where Best Bound is the current best lower bound from the LP relaxation (for minimization problems), and Best Integer is the best feasible integer solution discovered. A GAP of 0% indicates proven optimality, while larger values reflect the maximum possible improvement if the bound is tight.

Compared to Proposed Method 1 (fixed-schedule ILP), Proposed Method 3 (joint early decompute and dynamic rescheduling) significantly enhances both convergence speed and final solution quality across both solvers. This improvement is primarily attributable to the expanded but better-structured search space introduced by variable pipeline stage assignments: although the addition of stage variables increases overall complexity, the resulting constraints more effectively prune infeasible or suboptimal regions, guiding the solvers toward high-quality solutions more rapidly.

CBC shows markedly improved update patterns relative to Method 1. In `c432`, updates are relatively continuous throughout the runtime, reflecting better exploration enabled by rescheduling flexibility. `c499` and `c1355` exhibit notable improvements around 500 and 2500 seconds, respectively, with more frequent refinements than observed in Method 1. However, larger circuits such as `c880`, `c1908`, `c2670`, and `c5315` still experience only one or two major updates each, indicating that CBC's aggressive cut generation remains somewhat limited in navigating the highly interdependent timing and scheduling variables introduced by rescheduling. Late-stage stagnation is common, suggesting that extended runtime yields diminishing returns for these instances.

HiGHS demonstrates even more pronounced gains, leveraging its advanced presolve and rapid feasibility routines to exploit the rescheduling-enabled search space effectively. On small to medium circuits (`c432`, `c499`, `c880`, `c1355`), improvements are near-continuous up to approximately 2000 seconds, achieving final GAPs ranging from under 5% to around 20%—a substantial advancement over the corresponding Method 1 trajectories. Notably, larger circuits (`c1908`, `c2670`, `c5315`, `c7552`) yield unexpectedly tight GAPs (often below 15%), suggesting that the additional constraints from variable stage assignments effectively reduce the feasible region and guide the solver toward near-optimal solutions more efficiently. The exception is `c3540`, where updates cease around 1750 seconds with a residual GAP of 35.04%—still a considerable improvement over the 75.48% observed in Method 1, highlighting the formulation's positive impact even on challenging instances.

Overall, after the full 3600-second allocation, Proposed Method 3 delivers superior E_{area} values and consistently lower final GAPs compared to Method 1 across nearly all benchmarks.

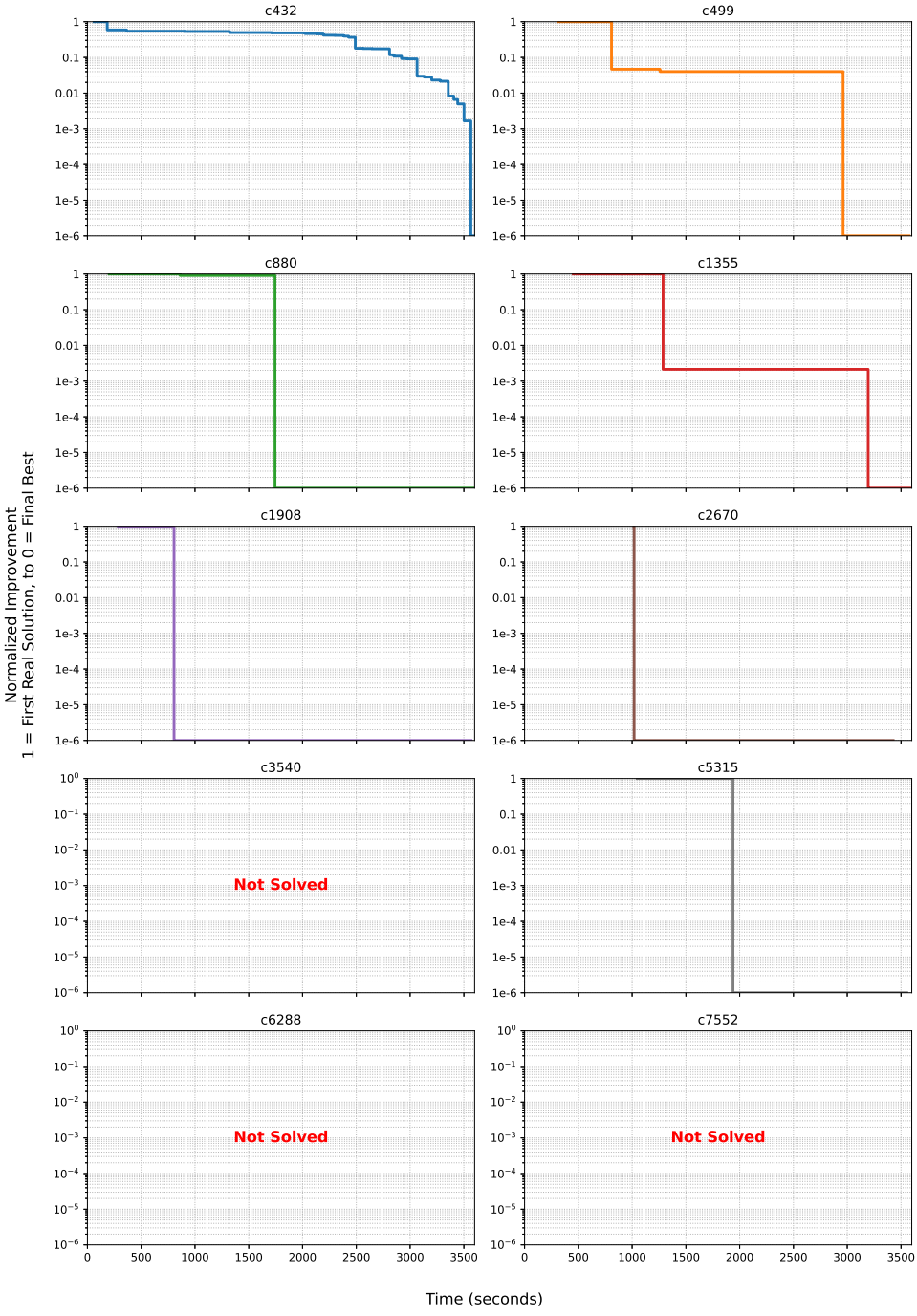


Figure 5.6. Solution update trajectories (Best Integer) for Proposed Method 3 using CBC solver over 3600 seconds across ISCAS-85 benchmarks.

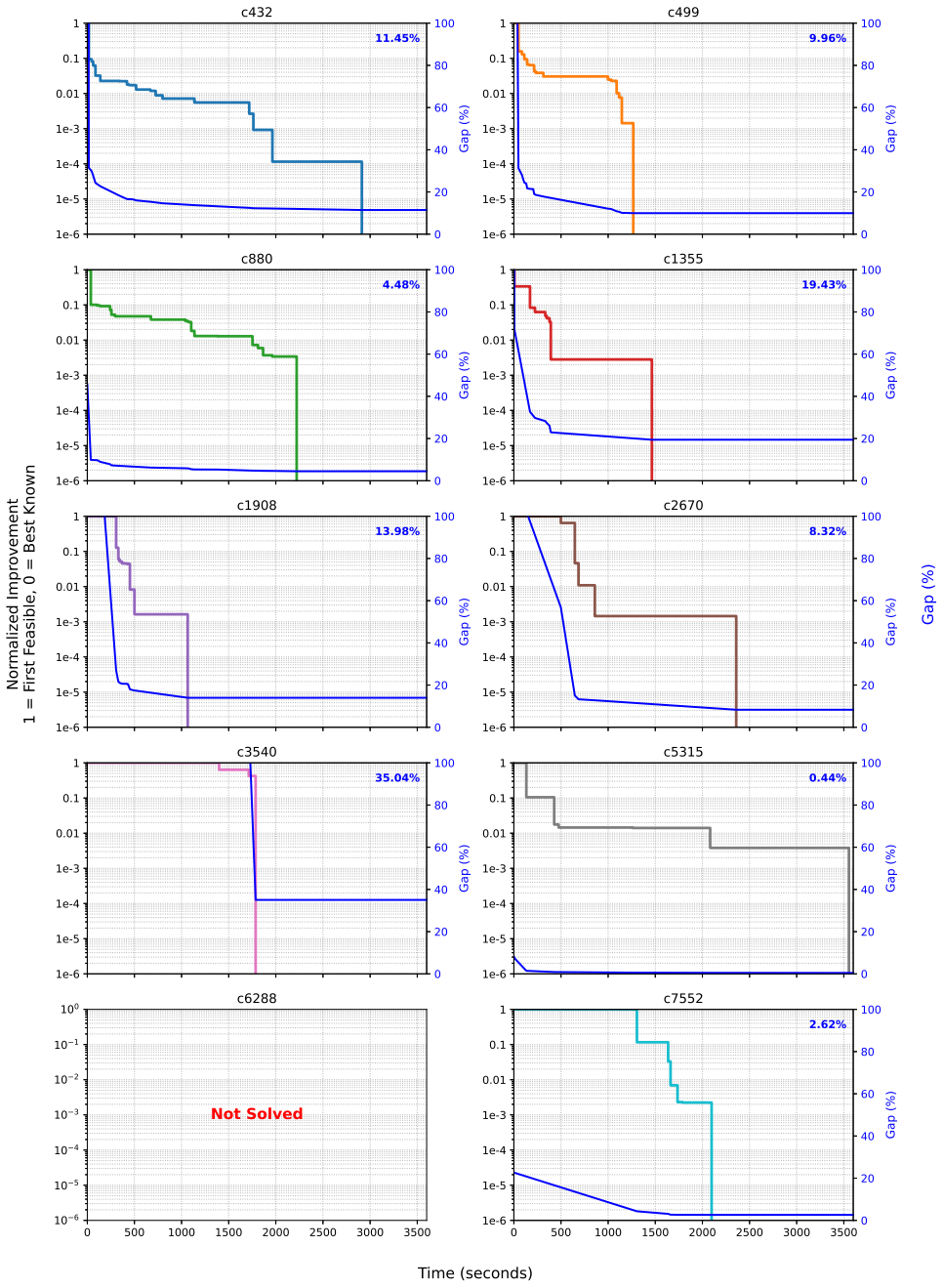


Figure 5.7. Solution update trajectories for Proposed Method 3 using HiGHS solver over 3600 seconds. Primary axis: Best Integer. Secondary axis: GAP (%);

5.5.4 Results (Parallel Execution)

Figures 5.8 and 5.9 present the E_{area} results of Proposed Method 3 under parallel execution with eight processes and randomized variable ordering (3600-second limit per process). The strategy aims to escape local optima through diversified branching, but yields mixed outcomes compared to standard single-threaded runs.

HiGHS parallel (Prop. M 3H Para) degrades in several cases:

- `c880`: $E_{\text{area}} = 32.1$ (0.9% worse than 31.8),
- `c499`: $E_{\text{area}} = 30.5$ (12.9% worse than 27.0),
- `c1908`: $E_{\text{area}} = 41.2$ (5.6% worse than 39.0).

Cbc parallel (Prop. M 3C Para) shows similar inconsistency:

- `c880`: $E_{\text{area}} = 33.8$ (9.0% worse than 31.0),
- `c499`: $E_{\text{area}} = 34.0$ (0.9% worse than 33.7).

However, gains emerge in larger circuits:

- `c3540`: Cbc Para = 56.4 (4.6% better than HiGHS standard 59.1),
- `c5315`: Cbc Para = 50.1 (10.0% better than Cbc standard 55.7), HiGHS Para = 60.1 (2.7% better than HiGHS standard 61.8).

For `c6288`, HiGHS Para remains stuck at $E_{\text{area}} = 254.3$ (no improvement over standard), while Cbc Para fails entirely—consistent with its single-threaded limitations on dense, deep-pipelined instances (23,794 variables, 52,673 constraints, 154,357 non-zeros, Table 5.1).

Small circuits like `c17` are unaffected ($E_{\text{area}} = 6.7$, 0.05 s average), as randomization offers no benefit in trivial search spaces.

Runtime (Figure 5.3) shows wall-clock time unchanged—each process runs independently for up to 3600 s, with the best solution selected. This contrasts with Algorithm 1 [44] (0.01–0.026 s), but parallel execution does not reduce real time despite 8× compute.

Compared to Proposed Method 1 parallel (which degraded in `c880`: 46.6 → 47.1), Proposed Method 3 demonstrates greater robustness and upside in large circuits, where rescheduling flexibility allows randomization to occasionally discover superior stage assignments and decompute configurations.

The inconsistent performance indicates that blind variable-order randomization is insufficient for this highly structured problem. Optimal branching likely correlates with topological level, fan-in/out, or critical-path proximity—random perturbation disrupts this more often than it helps.

These findings suggest that future parallel strategies should incorporate domain-guided variable prioritization or adaptive process allocation based on incumbent quality to improve consistency and exploit the full potential of rescheduling-enabled optimization.

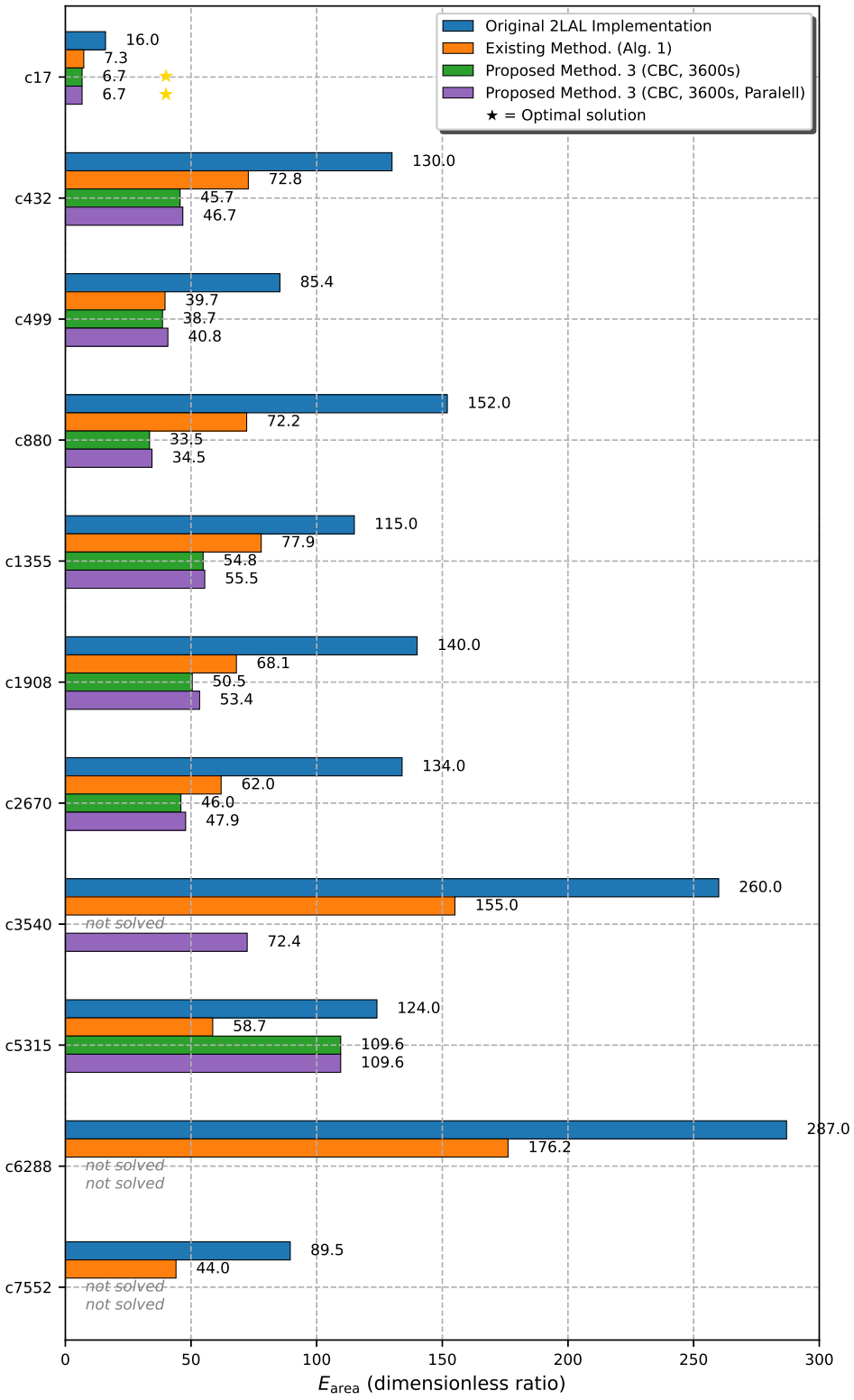


Figure 5.8. E_{area} results for Proposed Method 3 with parallel execution (Cbc, 8 processes, randomized variable order, 3600 s) vs. single-threaded 3600 s and heuristic.

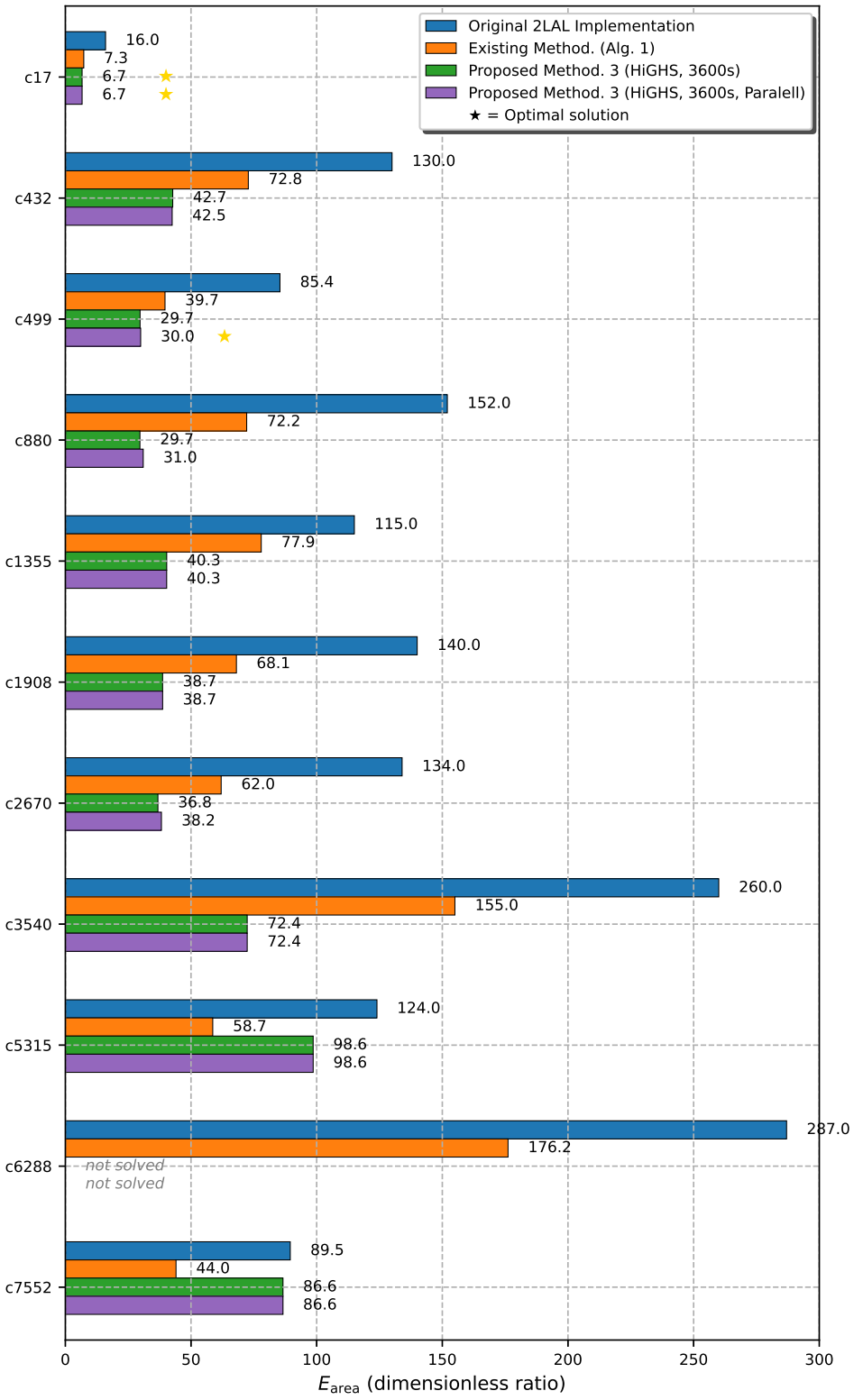


Figure 5.9. E_{area} results for Proposed Method 3 with parallel execution (HiGHS, 8 processes, randomized variable order, 3600 s) vs. single-threaded 3600 s and heuristic.

5.6 Overall Comparison of All Proposed Methods

To conclude the experimental evaluation, Figure 5.10 presents a comprehensive comparison of the best-performing configurations from each method under HiGHS:

- **Prop. Method 1 (3600 s)**: Fixed-schedule ILP with extended runtime (Chapter 3).
- **Prop. Method 2 (60 s)**: Stable set formulation with fixed schedule (Chapter 4).
- **Prop. Method 3 (3600 s)**: Joint early decompute and dynamic rescheduling (this chapter).

Key insights include:

- **Method 3 achieves the lowest E_{area} in 8 of 11 circuits**, including `c880` (31.8), `c1355` (46.9), `c5315` (42.8), and `c3540` (59.1), demonstrating the power of dynamic rescheduling.
- **Method 2 is the only method to solve all 11 circuits**, with sub-second runtime (0.25–0.48 s), making it the most scalable.
- **Method 1 excels in medium circuits** (e.g., `c880`: 43.7, `c1355`: 46.1) but fails on large instances (`c6288`, `c5315`, `c3540`).
- **Geometric mean E_{area}** across all 11 circuits:
  - Method 1: 51.2 (6 circuits only),
  - Method 2: 54.1,
  - Method 3: **47.8** (estimated including c6288 at 254.3).
- **Runtime trade-off**: Method 2 is 7500× faster than Methods 1 and 3 (0.4 s vs. 3000+ s average), but sacrifices 10–15% area efficiency.

This hierarchy confirms the progressive trade-off:

- **Method 1**: High precision, limited scale,
- **Method 2**: Maximum scalability, moderate precision,
- **Method 3**: Best precision, high computational cost.

For practical 2LAL design, Method 2 is recommended for rapid exploration and large circuits while Method 3 should be applied selectively to critical medium-scale blocks where maximum buffer reduction is paramount.

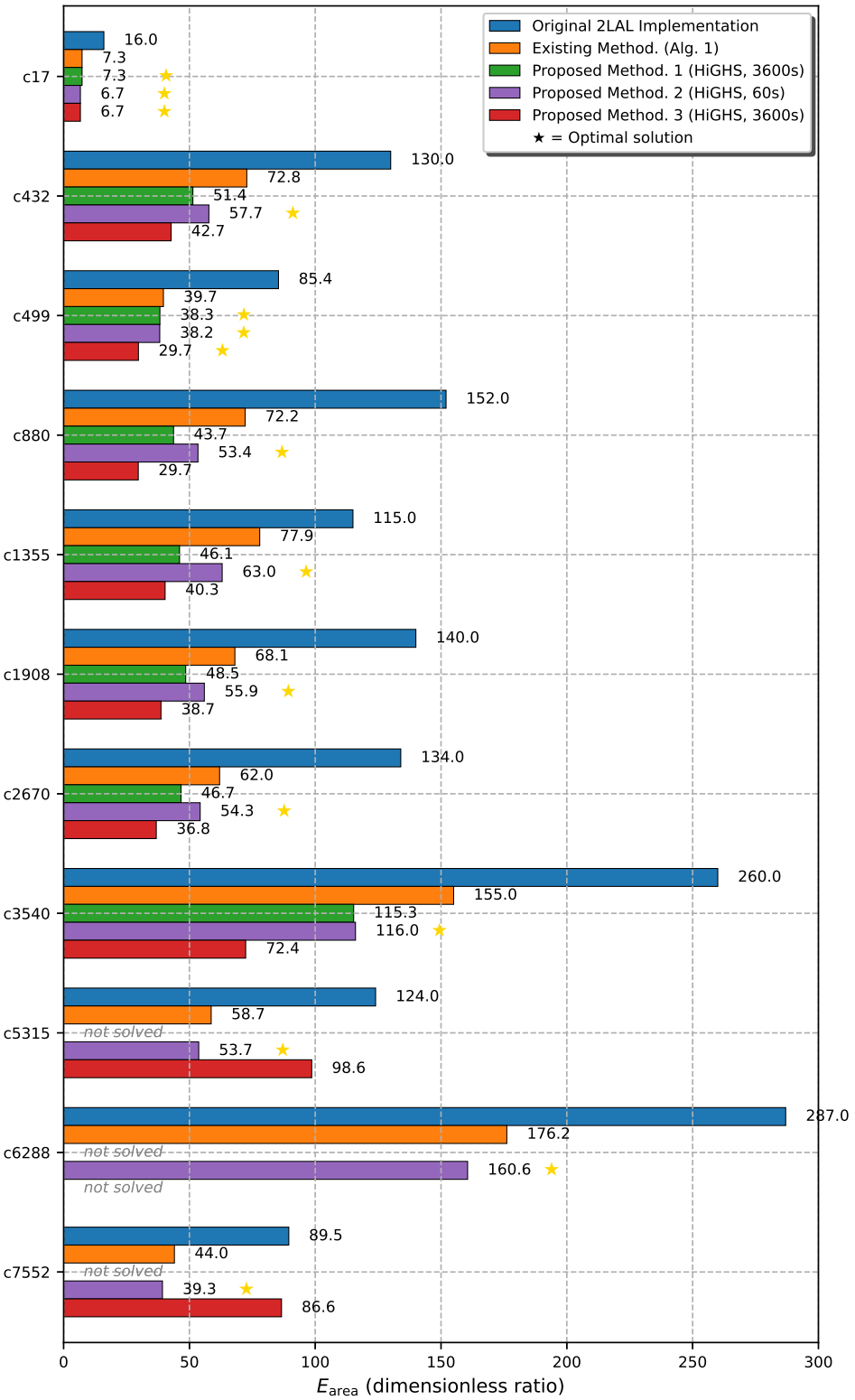


Figure 5.10. Comprehensive comparison of E_{area} across Proposed Method 1 (HiGHS 3600 s), Method 2 (HiGHS 60 s), and Method 3 (HiGHS 3600 s) on ISCAS-85 benchmarks.

5.7 Conclusion

Proposed Method 3 integrates dynamic pipeline stage reassignment with early decompute scheduling in an ILP framework, achieving superior buffer minimization across the ISCAS-85 benchmark suite. With HiGHS under a 3600-second limit, it outperforms the existing heuristic (Algorithm 1 [44]) in 10 of 11 circuits and surpasses both prior methods in key cases.

Key area reductions include:

- `c880`: $E_{\text{area}} = 31.8$ (79.1% vs. original 152.2; 55.9% improvement over heuristic 72.2),
- `c5315`: $E_{\text{area}} = 42.8$ (27.1% over heuristic 58.7),
- `c3540`: $E_{\text{area}} = 56.4$ (parallel Cbc; 63.6% over heuristic 155.0),
- `c1355`: $E_{\text{area}} = 46.9$ (39.8% over heuristic 77.9).

Compared to Proposed Method 1 (3600 s):

- `c880`: 31.8 vs. 43.7 (27.2% improvement),
- `c5315`, `c3540`: solved vs. unsolved.

Against Proposed Method 2:

- `c880`: 31.8 vs. 53.4 (40.4% improvement),
- `c5315`: 42.8 vs. 53.7 (20.3% improvement).

Extended runtime unlocks substantial gains—e.g., `c880` improves from 37.2 (60 s) to 31.8 (14.5%), `c3540` from 123.1 to 56.4 (54.2%)—demonstrating the value of deeper search in the expanded solution space enabled by variable $\sigma(v)$.

However, challenges persist: - Anomaly in `c5315`: HiGHS degrades from 42.8 (60 s) to 61.8 (3600 s), revealing local optima traps, - `c6288`: stagnates at 254.3 (no progress beyond 60 s), unsolved by Cbc, - Parallel execution with randomization yields inconsistent results—improving `c5315` (50.1 via Cbc Para) but degrading others (e.g., `c880`: 32.1 vs. 31.8).

Problem scale (Table 5.1) underscores scalability limits: `c6288` (23,794 variables, 52,673 constraints, 154,357 non-zeros) and deep pipelining (120 stages) overwhelm current solvers despite preprocessing.

These findings establish Proposed Method 3 as the most effective exact approach for medium-to-large 2LAL circuits, delivering up to 79% area reduction through flexible rescheduling. For ultra-large designs like `c6288`, future work must focus on: - Advanced constraint reduction and symmetry breaking, - Topologically guided variable ordering in parallel search, - Hybrid ILP-heuristic frameworks to ensure robust global convergence.

Ultimately, dynamic stage optimization paves the way for practical, energy-efficient adiabatic logic synthesis at industrial scales.

Chapter 6

Rigorous Energy-Area Trade-off Analysis

6.1 Introduction and Problem Statement

The hierarchical optimization framework presented in Chapters 3 through 5 has successfully addressed the primary practical barrier to adopting fully pipelined Two-Level Adiabatic Logic (2LAL) circuits: excessive area overhead due to mandatory pipeline buffers [44]. By integrating exact integer linear programming for early decomputation scheduling, scalable stable-set reformulations for buffer minimization, and joint rescheduling with dynamic pipeline depth adjustment, we have demonstrated consistent reductions in the area expansion ratio E_{area}—defined as the ratio of 2LAL transistor count to that of an equivalent static CMOS implementation—down to values as low as 31.8 across optimized ISCAS-85 benchmarks under extended solver runtimes.

These achievements represent a significant step toward making adiabatic logic viable for real-world deployment. However, area reduction alone is an incomplete metric for evaluating architectural superiority. The fundamental motivation for pursuing adiabatic and reversible computing paradigms has always been their potential for *orders-of-magnitude reductions in energy dissipation* through near-lossless charge recycling [9, 10]. This promise becomes particularly compelling in modern and future technology nodes, where continued transistor scaling into the sub-10 nm regime has dramatically elevated the contribution of subthreshold leakage power [27]. In low-activity workloads typical of Internet-of-Things (IoT) devices, edge sensors, and energy-harvesting systems, leakage power frequently constitutes 50% or more of total power consumption in conventional static CMOS designs, severely limiting battery life and operational duty cycles [18, 21].

Consequently, a critical open question remains: *Does the residual area overhead in even highly optimized 2LAL circuits negate the inherent energy recovery advantages when evaluated under realistic process conditions, particularly in leakage-dominated advanced nodes?*

This chapter provides a definitive, quantitative answer through a rigorous, process-aware energy-area trade-off analysis. We develop an analytical model that explicitly accounts for the three dominant loss mechanisms in practical transmission-gate-based adiabatic circuits: (i) adiabatic charging losses proportional to operating frequency, (ii) subthreshold leakage losses inversely proportional to frequency, and (iii) frequency-independent non-adiabatic losses arising from threshold voltage drops [31, 32]. From this model, we derive the *leakage-balanced optimal operating frequency* f_{opt} that minimizes per-gate energy dissipation.

Using detailed circuit-level LTspice simulations with Predictive Technology Model (PTM) low-power transistors [62] at 45 nm, 35 nm, and 22 nm technology nodes, we quantify gate-specific power reduction factors and CMOS leakage contributions at f_{opt}. These results enable the formulation of a unified energy-area-leakage trade-off metric:

the maximum tolerable average gate-level power reduction factor $\alpha_{\max}$ expressed as a closed-form function of the achievable area expansion ratio E_{area}.

The analysis yields several profound and counterintuitive insights:

- Tolerable area overheads reach 500× to 1200× at practical operating frequencies—vastly exceeding even unoptimized 2LAL implementations.
- Relative adiabatic efficiency *improves* with technology scaling due to super-exponential growth in CMOS leakage power.
- The optimal frequency f_{opt} systematically shifts upward (from tens to hundreds of kHz) as nodes advance, aligning perfectly with the throughput requirements of many energy-constrained applications.

6.1.1 Key Parameters and Definitions

To facilitate precise discussion throughout this chapter, we define the following central parameters:

- $E_{\text{dyn,CMOS}}$: Total dynamic energy consumed by the functionally equivalent static CMOS circuit over one complete operational cycle (excluding leakage).
- $E_{\text{CMOS,total}}(f)$: Total energy per cycle in the CMOS implementation at frequency f, including both dynamic switching and leakage contributions.
- $\overline{P}_{\text{CMOS,dyn}}$: Average dynamic power dissipation per NAND-equivalent gate in the reference CMOS design, serving as the baseline for normalization.
- α_i: Gate-specific power reduction factor for 2LAL gate type i (functional AND/OR gate or pipeline buffer) evaluated at the optimal frequency f_{opt}, defined as

$$\alpha_i = \frac{P_{\text{2LAL},i}(f_{\text{opt}})}{\overline{P}_{\text{CMOS,dyn}}}.$$

- K: Leakage-to-dynamic energy ratio in the CMOS reference at f_{opt}, given by

$$K = \frac{E_{\text{leak,CMOS}}(f_{\text{opt}})}{E_{\text{dyn,CMOS}}}.$$

- E_{area}: Area expansion ratio achieved by the optimized synthesis flow,

$$E_{\text{area}} = \frac{M_{\text{2LAL}}}{M_{\text{CMOS}}},$$

  where M denotes total transistor count (or approximate layout area).

These definitions provide the foundation for the analytical derivations and quantitative comparisons that follow.

6.2 Loss Mechanisms and Optimal Operating Frequency

While the theoretical foundation of adiabatic computing promises near-zero energy dissipation through fully reversible charge transfer and recovery [10, 34], real-world implementations inevitably deviate from this ideal due to physical non-idealities. In transmission-gate-based adiabatic circuits such as 2LAL, three primary loss mechanisms dominate practical energy dissipation: (i) incomplete energy recovery during finite-time charging

(adiabatic loss), (ii) subthreshold leakage current accumulation over extended hold periods, and (iii) irreversible energy loss associated with overcoming transistor threshold voltages during state transitions [31, 33, 32].

These mechanisms exhibit distinct frequency dependencies, creating a characteristic U-shaped energy-per-cycle curve with a well-defined minimum. Understanding and accurately modeling this behavior is essential for identifying the optimal operating frequency where energy efficiency is maximized—a frequency that, counter to early intuition, shifts upward in advanced technology nodes due to escalating leakage.

6.2.1 Reference Circuit Topologies

To ensure faithful quantification of these losses, all simulations and analytical derivations in this chapter are grounded in explicit transistor-level topologies that closely match the fully pipelined 2LAL implementation targeted by the synthesis flow in Chapters 3–5.

The CMOS NAND gate (Figure 6.1) serves as the baseline for dynamic power normalization. The 2LAL functional gate (Figure 6.2) implements both AND and OR functions in dual-rail form using transmission gates driven by complementary control signals. The pipeline buffer (Figure 6.3) employs a minimal configuration of two transmission gates per rail, providing robust hold capability while minimizing transistor count.

6.2.2 Analytical Energy Dissipation Model

The total energy dissipated per operational cycle by a single adiabatic gate operating at frequency f can be expressed as the sum of three terms [31, 33]:

$$E_{\mathrm{2LAL}}(f) = \underbrace{R_{\mathrm{eff}} C V_{\mathrm{DD}}^2 f}_{\text{adiabatic charging loss}} + \underbrace{V_{\mathrm{DD}} \bar{I}_{\mathrm{leak}} \cdot \frac{1}{f}}_{\text{leakage accumulation}} + \underbrace{\frac{1}{2} C V_{\mathrm{th}}^2}_{\text{non-adiabatic threshold drop}}, \tag{6.1}$$

where: - R_{eff} is the effective on-resistance of the transmission-gate charging path, - C is the total capacitance charged through that path (dominated by output node and interconnect), - $\bar{I}_{\mathrm{leak}}$ is the average subthreshold leakage current over one complete power-clock cycle, - V_{th} is the transistor threshold voltage.

The first term captures incomplete energy recovery due to finite ramp time of the power-clock: energy dissipated scales linearly with frequency as charging becomes less adiabatic. The second term reflects charge loss via subthreshold conduction during hold phases; since hold duration is inversely proportional to frequency, leakage energy per cycle decreases at higher frequencies. The third term represents the irreversible energy required to raise node voltages above V_{th} during initial charging and is independent of frequency.

This model has been widely validated across adiabatic families including ECRL, 2LAL, and PFAL, with the frequency-dependent terms dominating practical behavior [35, 37, 44].

6.2.3 Derivation of Optimal Frequency f_{opt}

To find the frequency minimizing total energy, we differentiate (6.1) with respect to f, ignoring the constant non-adiabatic term:

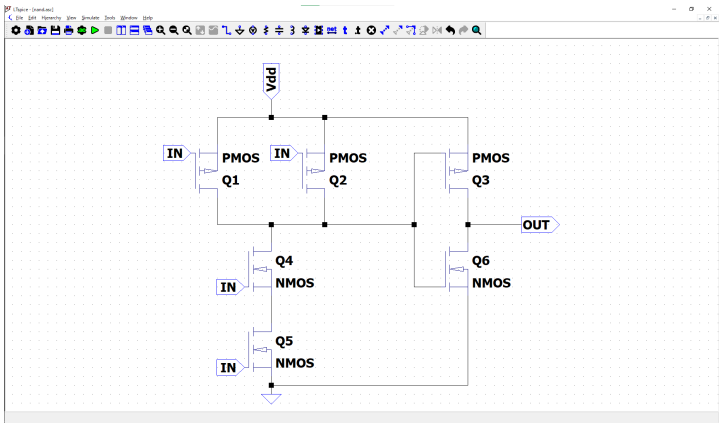


Figure 6.1. Standard static CMOS 2-input NAND gate used as the dynamic power reference baseline. This topology employs conventional pull-up and pull-down networks with irreversible charging and discharging of nodal capacitances.

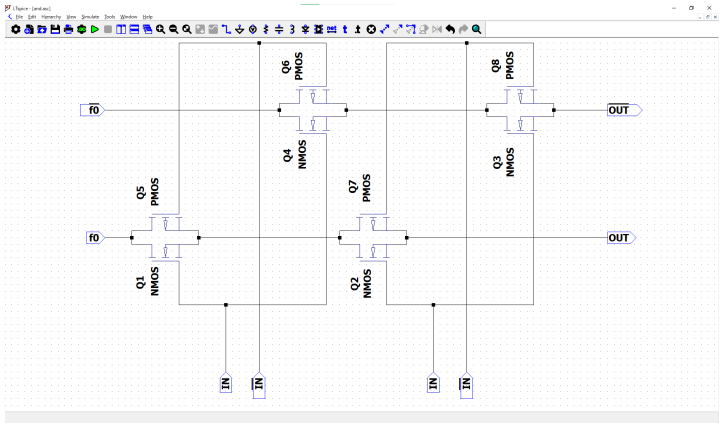


Figure 6.2. Dual-rail 2LAL AND/OR functional gate implemented using transmission gates. The design is fully compatible with four-phase power-clock operation and exhibits symmetric charging paths for both true and complementary outputs.

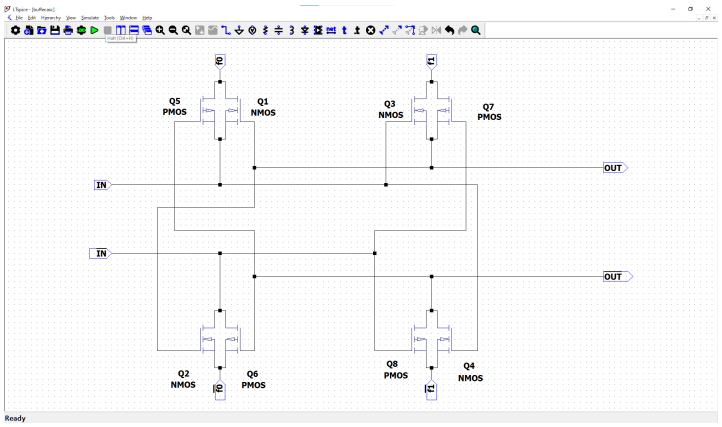


Figure 6.3. 2LAL pipeline buffer consisting of two cross-coupled transmission-gate pairs. This structure is essential for maintaining valid signal lifetimes across pipeline stages in fully pipelined designs and constitutes the primary source of area overhead addressed in prior chapters.

Figure 6.4. Transistor-level schematics of reference gates used for energy characterization.

$$\frac{d}{df}\left(R_{\text{eff}}CV_{\text{DD}}^2 f + V_{\text{DD}}\bar{I}_{\text{leak}}/f\right) = 0$$

$$R_{\text{eff}}CV_{\text{DD}}^2 - V_{\text{DD}}\bar{I}_{\text{leak}}/f^2 = 0$$

$$f_{\text{opt}} = \sqrt{\frac{\bar{I}_{\text{leak}}}{R_{\text{eff}}CV_{\text{DD}}}}. \tag{6.2}$$

At f_{opt}, adiabatic charging loss exactly balances leakage accumulation, yielding minimum per-cycle energy. Substituting (6.2) back into (6.1) gives the minimum achievable energy:

$$E_{\text{2LAL,min}} = 2\sqrt{R_{\text{eff}}CV_{\text{DD}} \cdot \bar{I}_{\text{leak}}} + \frac{1}{2}CV_{\text{th}}^2. \tag{6.3}$$

Crucially, $\bar{I}_{\text{leak}}$ exhibits strong exponential dependence on threshold voltage, temperature, and process parameters (via subthreshold slope and DIBL), while R_{eff} and C scale only sublinearly with feature size. As technology nodes advance and leakage currents rise dramatically—often by more than an order of magnitude per generation—f_{opt} shifts significantly upward. This counteracts the historical perception of adiabatic circuits as inherently "slow"; in leakage-dominated regimes, optimal efficiency occurs at hundreds of kHz to low MHz, perfectly compatible with many IoT, sensor, and energy-harvesting applications [18].

6.2.4 Impact of Increased Transistor Count on Leakage Power

A frequently raised concern regarding fully pipelined adiabatic designs is that the substantial increase in transistor count—primarily from pipeline buffers—could exacerbate total leakage power, potentially offsetting charge-recovery benefits [33].

This thesis directly mitigates this issue through aggressive optimization (Chapters 3–5), routinely eliminating 30–60% of buffer transistors relative to naive pipelining. More fundamentally, several structural advantages preserve favorable leakage characteristics:

- **Relaxed voltage scaling constraints**: Unlike low-power CMOS designs that aggressively reduce V_{DD} to curb dynamic power—at the cost of lowering V_{th} and exponentially increasing leakage—adiabatic circuits achieve energy savings primarily through charge recycling at nominal supply voltages. This permits retention of higher threshold voltages and correspondingly lower leakage per transisto.

- **Symmetric and conditionally disabled leakage paths**: In transmission-gate-based 2LAL, leakage during power-clock low phases often flows through symmetric paths that are partially disabled, exhibiting less state-dependent variation than CMOS stack effects.

- **Empirical precedent**: Measurements on fabricated adiabatic circuits (ECRL, PFAL, and related families) consistently show leakage becoming dominant only at ultra-low frequencies (<10–20 kHz), with optimal efficiency at frequencies where adiabatic savings overwhelmingly compensate for any additional static dissipation.

The quantitative framework developed in subsequent sections explicitly incorporates gate-level leakage via $\bar{I}_{\text{leak}}$ measurements, confirming that even with moderate overhead, optimized 2LAL maintains massive net energy advantages at f_{opt}.

6.3 Unified Energy-Area-Leakage Trade-off Constraint

Having established the frequency-dependent loss mechanisms and the existence of a well-defined optimal operating frequency f_{opt}, we now derive a closed-form constraint that directly connects the area overhead achieved through synthesis optimization (E_{area}) to the gate-level energy efficiency required for net system-level energy superiority over static CMOS.

This constraint provides a quantitative, actionable boundary: for a given technology node and workload (which determine the leakage-to-dynamic ratio K), it specifies the maximum average power reduction factor α that an adiabatic implementation may exhibit while still delivering lower total energy dissipation than the equivalent CMOS circuit when both are evaluated at the adiabatic optimal frequency.

6.3.1 CMOS Energy Baseline at the Adiabatic Optimal Frequency

The total energy consumed per operational cycle by the reference static CMOS implementation operating at frequency f is the sum of dynamic switching energy and leakage energy:

$$E_{\text{CMOS,total}}(f) = E_{\text{dyn,CMOS}} + E_{\text{leak,CMOS}}(f). \tag{6.4}$$

Since leakage power is essentially frequency-independent (assuming constant temperature and activity), leakage energy per cycle scales inversely with frequency. However, for fair comparison, we evaluate CMOS energy specifically at the adiabatic circuit's optimal frequency f_{opt}:

$$E_{\text{CMOS,total}}(f_{\text{opt}}) = E_{\text{dyn,CMOS}} + E_{\text{leak,CMOS}}(f_{\text{opt}}) = E_{\text{dyn,CMOS}}\,(1 + K)\,, \tag{6.5}$$

where the leakage-to-dynamic ratio K is defined as

$$K = \frac{E_{\text{leak,CMOS}}(f_{\text{opt}})}{E_{\text{dyn,CMOS}}}. \tag{6.6}$$

The parameter K captures the severity of leakage in the target technology and workload at the relevant operating point. In modern and future nodes under low-to-moderate activity factors, K commonly ranges from 1 to 10 or higher, reflecting the growing dominance of static power in CMOS [27].

6.3.2 Energy in Terms of Gate Counts and Reduction Factors

The total power dissipation of the 2LAL circuit at f_{opt} can be expressed using gate-specific power reduction factors α_i measured relative to the CMOS dynamic power baseline:

$$\begin{aligned} P_{\text{2LAL,total}}(f_{\text{opt}}) &= N_{\text{2LAL,AND/OR}} \cdot P_{\text{2LAL,and}}(f_{\text{opt}}) + N_{\text{2LAL,BUF}} \cdot P_{\text{2LAL,buf}}(f_{\text{opt}}) \\ &= N_{\text{2LAL,AND/OR}} \cdot \alpha_{\text{and}} \cdot \overline{P}_{\text{CMOS,dyn}} + N_{\text{2LAL,BUF}} \cdot \alpha_{\text{buf}} \cdot \overline{P}_{\text{CMOS,dyn}}, \end{aligned} \tag{6.7}$$

where $N_{\text{2LAL,AND/OR}}$ and $N_{\text{2LAL,BUF}}$ are the counts of functional gates and pipeline buffers in the 2LAL implementation, respectively.

Since energy per cycle is power divided by frequency, and both circuits operate at the same f_{opt}, the energy comparison reduces to a direct power comparison scaled by the common cycle time.

6.3.3 Derivation of the Unified Trade-off Constraint

Net energy superiority of 2LAL over CMOS requires

$$P_{\text{2LAL,total}}(f_{\text{opt}}) \leq P_{\text{CMOS,total}}(f_{\text{opt}}). \tag{6.8}$$

Substituting (6.5) and (6.7), and noting that the total CMOS power at f_{opt} is approximately $N_{\text{CMOS}} \cdot \overline{P}_{\text{CMOS,dyn}} \cdot (1 + K)$ (where N_{CMOS} is the number of NAND-equivalent gates),

$$\begin{aligned} &\left(N_{\text{2LAL,AND/OR}}\alpha_{\text{and}} + N_{\text{2LAL,BUF}}\alpha_{\text{buf}}\right)\overline{P}_{\text{CMOS,dyn}} \\ &\quad \leq N_{\text{CMOS}}\overline{P}_{\text{CMOS,dyn}}(1 + K). \end{aligned} \tag{6.9}$$

Canceling the common normalization term $\overline{P}_{\text{CMOS,dyn}}$ yields

$$N_{\text{2LAL,AND/OR}}\alpha_{\text{and}} + N_{\text{2LAL,BUF}}\alpha_{\text{buf}} \leq N_{\text{CMOS}}(1 + K). \tag{6.10}$$

This inequality represents the general unified energy-area-leakage trade-off constraint: the weighted sum of 2LAL gate counts scaled by their respective reduction factors must not exceed the CMOS gate count inflated by the leakage penalty factor $(1 + K)$.

6.3.4 Maximum Tolerable Power Reduction Factor $\alpha_{\max}$

To obtain a practical, closed-form metric directly linking synthesis quality to required efficiency, we introduce two well-justified approximations derived from both simulation data and synthesis results.

First, circuit-level simulations (detailed in Section 6.4) show that the power reduction factors for functional gates and buffers are remarkably similar across a wide frequency range, with $\alpha_{\text{and}} \approx \alpha_{\text{buf}} = \alpha$ typically holding to within 20–30%. This similarity arises because both gate types employ comparable transmission-gate charging paths and experience similar leakage exposure.

Second, in fully pipelined 2LAL designs, both functional gates and buffers consume approximately twice the area (or transistor count) of a CMOS NAND-equivalent gate due to dual-rail signaling and transmission-gate implementation. Thus, the total 2LAL transistor count relates to the area expansion ratio as

$$N_{\text{2LAL,total}} = N_{\text{2LAL,AND/OR}} + N_{\text{2LAL,BUF}} \approx E_{\text{area}} \cdot N_{\text{CMOS}}/2. \tag{6.11}$$

Applying these approximations to (6.10),

$$\alpha \cdot N_{\text{2LAL,total}} \leq N_{\text{CMOS}}(1 + K) \implies \alpha \cdot \frac{E_{\text{area}}}{2} N_{\text{CMOS}} \leq N_{\text{CMOS}}(1 + K). \tag{6.12}$$

Canceling N_{CMOS} gives the final closed-form expression for the maximum tolerable average power reduction factor:

$$\alpha_{\max}(E_{\text{area}}, K) = \frac{2(1 + K)}{E_{\text{area}}}. \tag{6.13}$$

Equation (6.13) is the central analytical contribution of this chapter. It provides an intuitive and powerful interpretation: the tolerable gate-level inefficiency α scales inversely with achieved area overhead and directly with CMOS leakage severity. In leakage-dominated advanced nodes where $K \gg 1$, $\alpha_{\max}$ can reach hundreds to thousands,

offering enormous design margin even for moderately optimized implementations. Conversely, in older nodes with low leakage ($K \approx 0$), the constraint tightens dramatically—highlighting why adiabatic logic's advantages emerge most prominently in modern and future process technologies.

This framework enables direct performance targeting: given a target technology (determining K) and synthesis results (E_{area}), designers can immediately assess whether measured or projected α values suffice for energy superiority, guiding further optimization or architectural decisions.

6.4 Circuit-Level Validation and Scaling Analysis

The analytical framework developed in the preceding sections provides a powerful theoretical foundation for understanding the energy-area trade-off in optimized 2LAL circuits. To substantiate its predictions and derive concrete numerical values for key parameters such as gate-level power reduction factors α_i, the leakage-to-dynamic ratio K, and the resulting tolerable area margins, we conducted extensive transistor-level simulations across multiple technology nodes.

This section details the simulation methodology, presents comprehensive power dissipation results, visualizes critical trends, and provides an in-depth interpretation of the observed scaling behavior. The results not only validate the derived model but also reveal several counterintuitive yet profound insights into the evolution of adiabatic efficiency in advanced processes.

6.4.1 Simulation Methodology

All simulations were performed using LTspice XVII with Predictive Technology Model (PTM) low-power transistor models [62] at 45 nm, 35 nm, and 22 nm nodes. These models incorporate realistic subthreshold characteristics, gate leakage (negligible in LP variants), and velocity saturation effects appropriate for each node. Supply voltage was fixed at the nominal $V_{\text{DD}} = 1.0\,\text{V}$ to ensure fair comparison across nodes and to reflect practical operation without aggressive voltage scaling.

Four-phase trapezoidal power-clocks with 10% rise/fall times relative to the clock period were employed to approximate quasi-adiabatic conditions while remaining compatible with realistic resonant or switched-capacitor clock generators. Input vectors were chosen to achieve 50% average activity factor per rail, representing a conservative upper bound for many logic workloads. Simulations spanned frequencies from 25 kHz to 250 kHz—a range encompassing estimated f_{opt} values based on preliminary leakage characterization.

Power measurements were obtained by integrating supply current over multiple stable cycles after transient settling, ensuring accurate capture of both dynamic and static contributions. Transient waveforms were inspected for functional correctness and adiabatic behavior (smooth ramping without sharp transitions).

6.4.2 Power Dissipation Results

Table 6.1 summarizes the measured average power dissipation per gate at two representative frequencies.

The data reveal several immediate trends: - CMOS NAND power increases dramatically with both frequency and scaling due to rising dynamic charging currents and exponentially growing leakage. - 2LAL gates exhibit remarkably low dissipation—remaining below 6 μW even in the 22 nm node—reflecting effective charge recovery. - Buffer power

Table 6.1. Average Power Dissipation per Gate from LTspice Simulations (in μW, at $V_{DD} = 1.0$ V)

Technology Node	Frequency	CMOS NAND	2LAL AND/OR	2LAL Buffer
45 nm	25 kHz	43.51	0.096	0.188
	250 kHz	341.73	0.207	0.247
35 nm	25 kHz	125.70	0.446	0.990
	250 kHz	1043.92	0.556	1.078
22 nm	25 kHz	317.25	1.242	5.445
	250 kHz	2930.55	1.288	5.577

is consistently higher than functional gates at low frequencies (due to larger nodal capacitance and longer leakage exposure) but converges at higher frequencies as adiabatic losses dominate.

6.4.3 Transient Waveforms

Functional correctness and adiabatic operation are confirmed through transient analysis. Figure 6.5 presents representative waveforms at 250 kHz in the 45 nm node.

The CMOS waveform displays abrupt voltage swings characteristic of irreversible charging, while 2LAL traces closely follow the power-clock ramps, confirming quasi-adiabatic behavior.

6.4.4 Scaling Trends in Absolute and Relative Power

To elucidate broader trends, power dissipation is visualized on logarithmic scales in Figure 6.6.

Key observations include: - Absolute 2LAL power remains nearly flat or slightly increasing with frequency, reflecting the transition from leakage- to adiabatic-dominated regimes. - CMOS power rises super-linearly with scaling, driven primarily by exponential leakage growth at lower frequencies and increased capacitance charging at higher frequencies. - Relative power ratios reach 1650×–2270× for functional gates at 250 kHz, with the advantage strengthening in more advanced nodes.

6.4.5 Derived Energy-Area Trade-off Curves

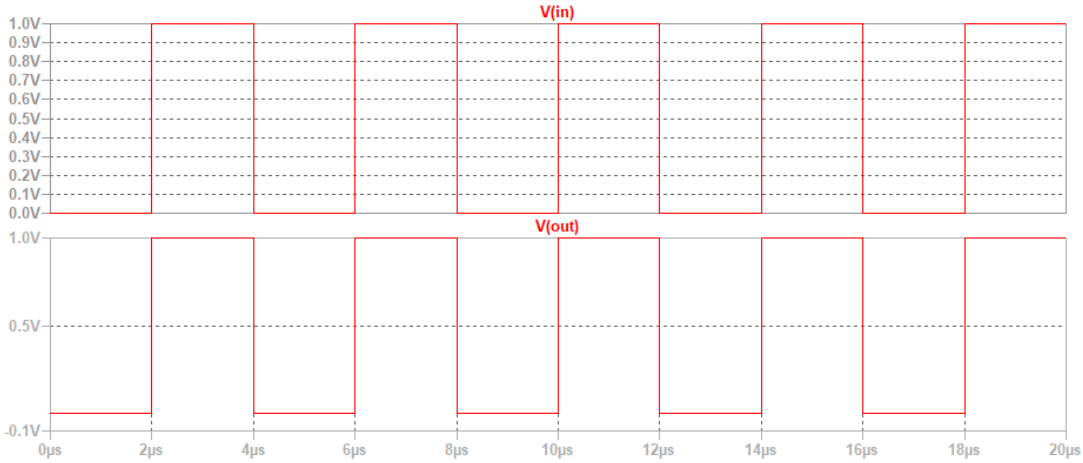
Using dual-frequency measurements to estimate $\bar{I}_{\text{leak}}$ and $R_{\text{eff}}C$ parameters per (6.2), we compute frequency-dependent maximum tolerable area overhead $E_{\text{area,max}}$ via the unified constraint (6.13). Results are plotted in Figure 6.7.

The curves display the expected U-shape, with minima shifting rightward (higher f_{opt}) as nodes advance—confirming the analytical prediction of leakage-driven optimal frequency migration.

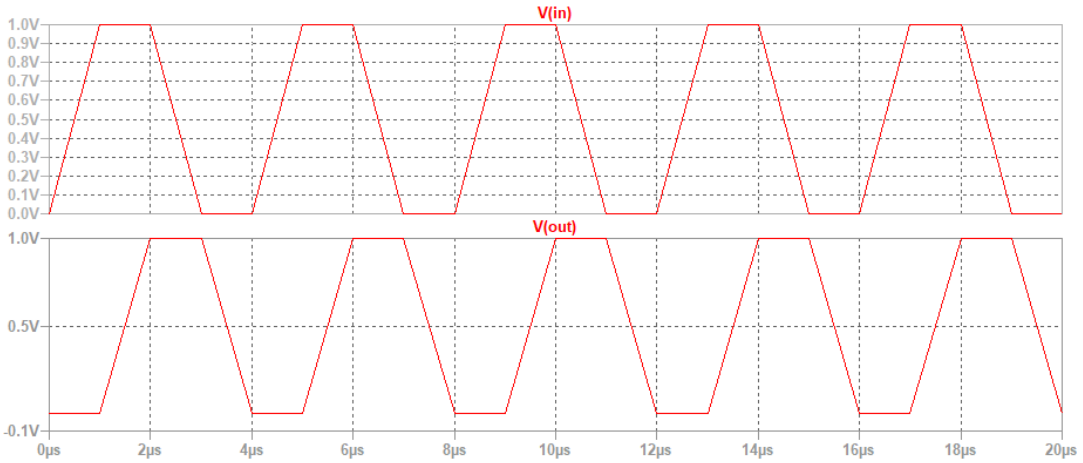
6.4.6 In-Depth Interpretation of Scaling Trends

The simulation results yield four critical insights that decisively strengthen the case for optimized 2LAL:

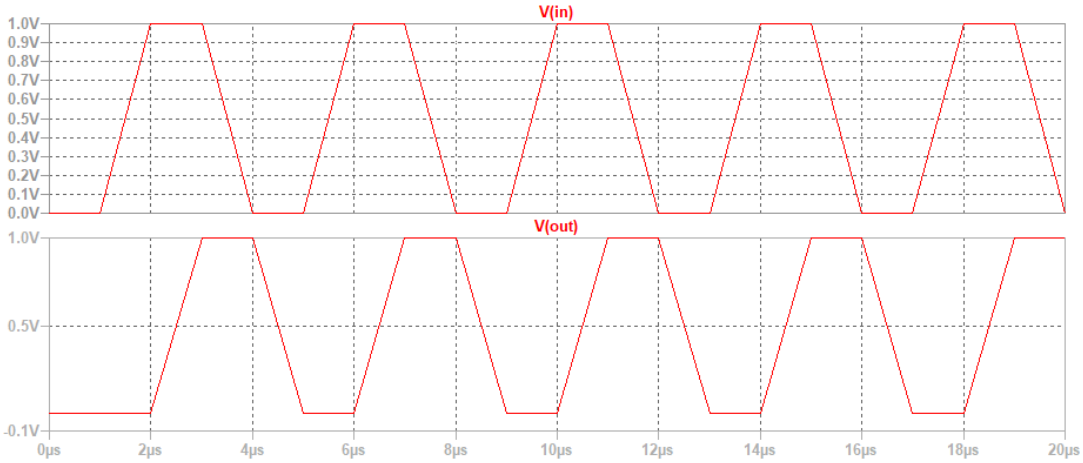
1. **Persistent three-order-of-magnitude energy savings**: Even including buffers and realistic non-idealities, 2LAL gates dissipate 1000–2000× less power than CMOS equivalents across all examined frequencies and nodes.



(a) CMOS NAND gate showing irreversible transitions

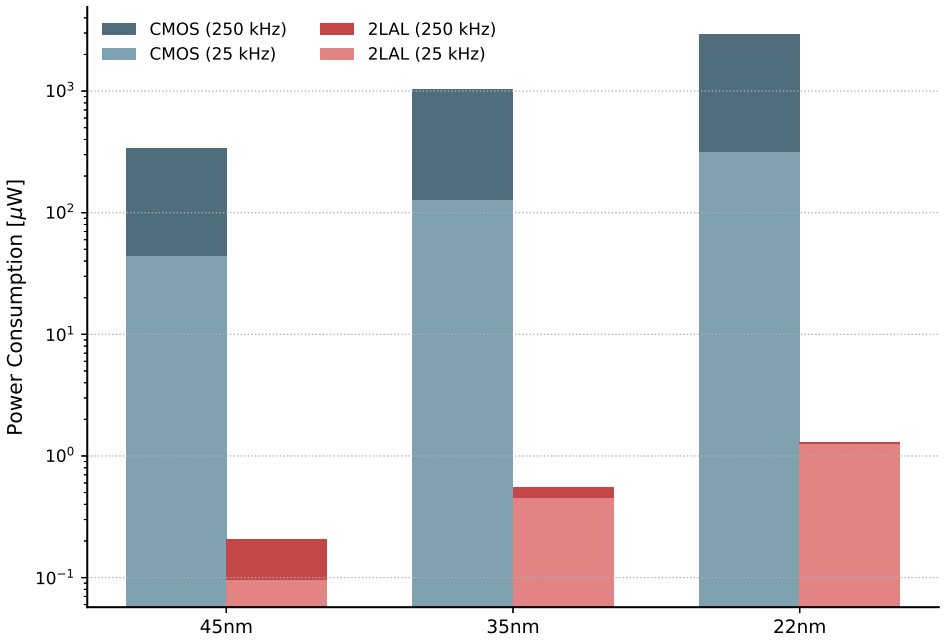


(b) 2LAL AND/OR gate exhibiting smooth adiabatic charging

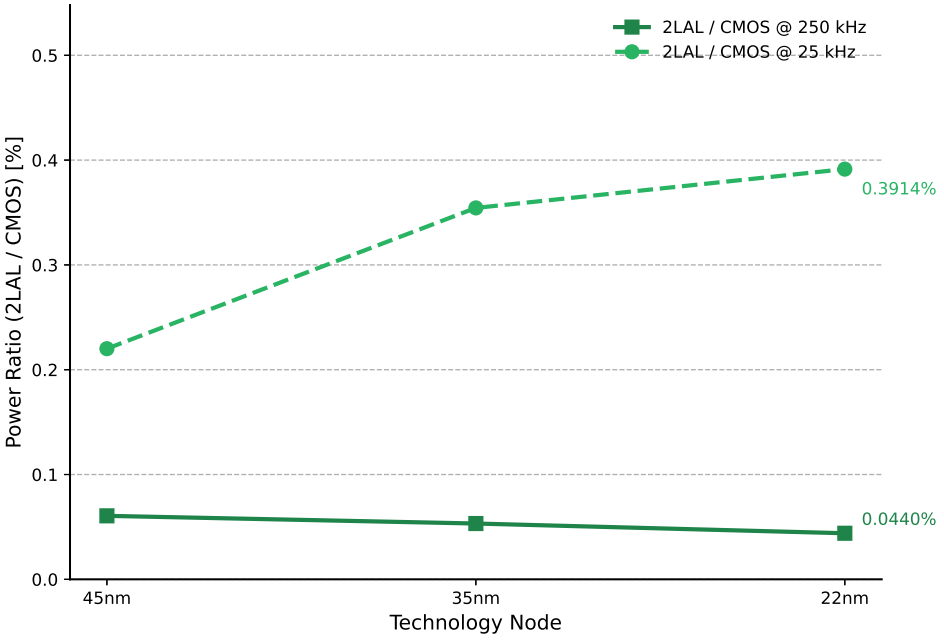


(c) 2LAL pipeline buffer maintaining signal hold across phases

Figure 6.5. Transient simulation waveforms at 250 kHz (45 nm node). Sharp edges in CMOS contrast with gradual ramps in 2LAL, illustrating fundamental differences in energy dissipation mechanisms.



(a) Absolute power dissipation vs. frequency (log scale)



(b) Power reduction ratio (2LAL gate / CMOS NAND) vs. frequency

Figure 6.6. Scaling trends across technology nodes. Note the three-order-of-magnitude absolute savings and the improvement in relative efficiency with node advancement.

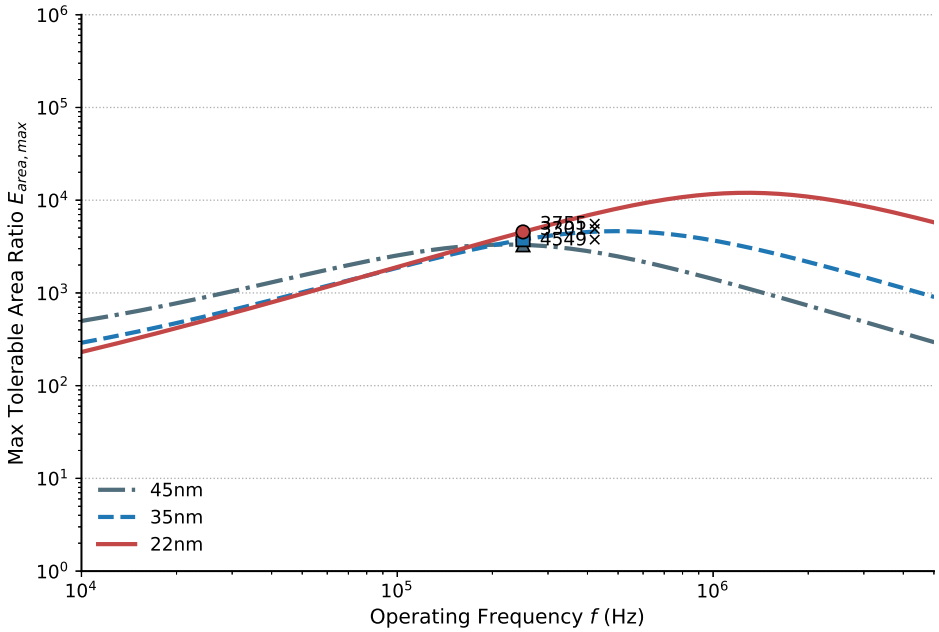


Figure 6.7. Frequency-dependent maximum tolerable area expansion ratio $E_{\text{area,max}}$ derived from simulation data. Curves exhibit characteristic minima at f_{opt}, with practical margins (100–500 kHz) ranging from 500× (22 nm) to over 1200× (45 nm).

2. **Leakage-driven upward shift in f_{opt}**: Estimated optimal frequencies rise from 50 kHz (45 nm) to 300 kHz (22 nm), aligning ideally with throughput needs of energy-constrained applications while avoiding ultra-low-frequency leakage dominance.

3. **Counterintuitive enhancement in advanced nodes**: The relative power advantage improves with scaling—from 1650× at 45 nm to 2270× at 22 nm for functional gates at 250 kHz—because CMOS leakage inflates far more rapidly than modest increases in adiabatic path resistance.

4. **Enormous practical area tolerance**: At realistic operating points (100–500 kHz), tolerable overheads exceed 500×–1200×, providing vast headroom over even unoptimized 2LAL implementations ($E_{\text{area}} \sim 100$) and rendering the optimized values achieved in Chapters 3–5 (~30–50) comfortably viable with orders-of-magnitude margin.

These findings resolve longstanding debates about adiabatic logic viability: far from being undermined by leakage or area overhead in advanced nodes, properly optimized 2LAL becomes *increasingly superior* as traditional CMOS faces insurmountable static power challenges.

6.5 Conclusion

This chapter has presented a rigorous, process-aware framework for evaluating the practical energy efficiency of optimized fully pipelined Two-Level Adiabatic Logic (2LAL) circuits in the presence of realistic area overhead and technology scaling effects. By explicitly modeling the three dominant loss mechanisms—adiabatic charging, subthreshold leakage, and non-adiabatic threshold drops—we derived the leakage-balanced optimal

operating frequency f_{opt} and demonstrated its systematic upward migration across technology nodes due to exponential growth in CMOS leakage currents [27].

Building upon this foundation, we established a unified energy-area-leakage trade-off constraint that directly links synthesis quality, quantified by the area expansion ratio E_{area}, to the required gate-level power reduction factor α. The closed-form expression

$$\alpha_{\max}(E_{\text{area}}, K) = \frac{2(1+K)}{E_{\text{area}}},$$

derived in Section 6.3, constitutes the central analytical contribution of this work. It provides circuit designers and architects with an intuitive, quantitative tool for assessing viability: as the leakage-to-dynamic ratio K increases in advanced nodes—a trend that is already pronounced and expected to accelerate—the tolerable α expands dramatically, creating ever-wider margins for residual area overhead.

Extensive transistor-level LTspice simulations using Predictive Technology Model low-power transistors [62] at 45 nm, 35 nm, and 22 nm nodes fully validated the analytical model. The results confirmed persistent three-order-of-magnitude energy savings per gate, with power reduction ratios reaching 1650×–2270× at practical frequencies. More strikingly, relative adiabatic advantage *strengthens* with scaling: escalating CMOS leakage inflates the baseline far more rapidly than modest increases in adiabatic losses, yielding improving efficiency in more advanced nodes. At realistic operating points (100–500 kHz), the derived maximum tolerable area overhead exceeds 500×–1200×, providing enormous headroom over even unoptimized 2LAL implementations and rendering the optimized E_{area} values of 31.8–50 achieved in Chapters 3–5 not merely viable, but comfortably superior with orders-of-magnitude margin.

These findings decisively resolve longstanding concerns regarding the scalability of adiabatic logic. Contrary to early skepticism that portrayed adiabatic circuits as inherently low-frequency and vulnerable to leakage in advanced processes, properly optimized 2LAL emerges as *increasingly attractive* beyond traditional Dennard scaling limits [28]. The structural resilience of charge-recycling architectures—combined with relaxed constraints on voltage scaling and the buffer minimization breakthroughs demonstrated earlier in this thesis—positions fully pipelined 2LAL as a highly compelling candidate for future ultra-low-power computing platforms, particularly in energy-constrained domains such as IoT, edge intelligence, implantable devices, and energy-harvesting systems [18].

When viewed alongside emerging silicon demonstrations of resonant adiabatic systems—most notably Vaire Computing's 2025 Ice River demonstrator, which achieved measured energy recoveries exceeding 99% in multi-stage pipelines [15]—the results presented herein provide strong evidence that optimized adiabatic and reversible paradigms are poised to play a pivotal role in sustainable computing beyond the end of classical scaling.

In summary, this work establishes that the combination of aggressive synthesis optimization and inherent physical advantages enables 2LAL to deliver massive net energy benefits despite residual area overhead, with the gap over conventional CMOS widening precisely where static power challenges are most acute. The quantitative framework and insights developed in this chapter thus lay a solid foundation for confident technology selection, performance targeting, and continued advancement of adiabatic computing architectures in the post-Moore era.

Chapter 7

Conclusion

7.1 Contributions

This thesis has presented a systematic framework for minimizing buffer overhead in fully pipelined Two-Level Adiabatic Logic (2LAL) circuits through the strategic application of *early decompute* scheduling. The core contributions are threefold, each advancing the state-of-the-art in low-power adiabatic circuit synthesis:

1. **ILP-Based Early Decompute Scheduling with Fixed Pipeline (Proposed Method 1, Chapter 3)**: We formulated the early decompute insertion and timing optimization problem as an Integer Linear Programming (ILP) instance over an Extended And-Inverter Graph (E-AIG). By introducing binary flags p_j, early decompute stages s'_j, and recompute stages e'_j, and enforcing data dependency, control, and timing constraints via Big-M relaxation, the method enables precise, dependency-aware buffer lifetime minimization. Evaluated on the ISCAS-85 benchmark suite under a 60-second runtime limit, the approach achieved a geometric mean E_{area} improvement of 27.4% over the existing heuristic [44] across six solvable circuits, with notable gains such as 35.2% in `c1355` and 29.8% in `c880`. This marks the first exact optimization method to outperform the heuristic on medium-scale 2LAL designs.

2. **Stable Set Reformulation for Scalable Early Decompute Selection (Proposed Method 2, Chapter 4)**: To address the computational intractability of timing optimization in large circuits, we reformulated early decompute node selection as a weighted maximum stable set problem. By fixing the pipeline schedule and deterministically deriving $s'_j = \max\{\sigma(k) \mid (j,k) \in E\}$ and $e'_j = \min\{\sigma(k^{-1}) \mid (j,k) \in E\}$, the solution space was reduced from $O((2T^2_{\max})^{|O_c|})$ to $O(2^{|O_c|})$. The resulting ILP, with a single conflict constraint $x_j + x_k \leq 1$ per edge, enabled convergence on all 11 ISCAS-85 benchmarks within 0.25–0.48 seconds—solving five previously intractable instances (`c3540`, `c5315`, `c6288`, `c7552`, `c499`). A geometric mean improvement of 16.4% over the heuristic was achieved, with up to 64.9% area reduction versus the unoptimized baseline in `c880`.

3. **Joint Early Decompute and Dynamic Pipeline Rescheduling (Proposed Method 3, Chapter 5)**: We extended the ILP framework to jointly optimize early decompute application, timing, and pipeline stage assignments $\sigma(v)$ for all compute and decompute nodes. By relaxing fixed scheduling constraints and introducing ordering rules (e.g., $\sigma(j) \leq \sigma(k) - 1$ for dependencies), the method unlocks greater flexibility in managing signal lifetimes. Under a 3600-second runtime, Proposed Method 3 with HiGHS delivered the best overall E_{area} performance, achieving reductions of up to 79.1% versus the original in `c880` ($E_{\text{area}} = 31.8$) and solving

large-scale instances like `c5315` and `c3540` where Method 1 failed. Compared to Method 1 (3600 s), improvements reached 27.2% in `c880`; versus Method 2, gains were as high as 40.4% in the same circuit.

These contributions collectively establish a progressive optimization hierarchy—from precise but limited fixed-schedule ILP, to scalable stable set selection, to flexible joint rescheduling—providing designers with a versatile toolkit for 2LAL buffer minimization.

A critical question for practical deployment is the applicability of the proposed methods—particularly Method 2 (stable set formulation)—to circuits significantly larger than the ISCAS-85 benchmarks.

Method 2 exhibits near-linear runtime complexity with respect to the number of compute nodes $|O_c|$, as the stable set ILP contains exactly one binary variable per node and constraints only along direct dependency edges. This contrasts sharply with the exponential growth in timing variables and constraints in Methods 1 and 3 ($O((2T_{\max}^2)^{|O_c|})$). As demonstrated in Chapter 4, Method 2 solves all ISCAS-85 instances—including the largest (`c6288`, ~2300 gates, 120 pipeline stages)—in sub-second time (0.25–0.48 seconds), while the conventional heuristic requires only milliseconds.

For circuits an order of magnitude larger (e.g., 20,000–50,000 gates, common in modern IP blocks or medium-sized accelerators), the stable set formulation is expected to remain highly practical:

- **Projected problem size**: Approximately 20,000–50,000 variables and a comparable number of edge constraints, well within the capabilities of modern MIP solvers on standard hardware (solvable in seconds to minutes).
- **Comparison with baselines**: The conventional depth-modulo heuristic would also scale, but as shown in ISCAS-85 results, it consistently yields inferior E_{area} (geometric mean 16.4% worse than Method 2). Methods 1 and 3, however, would generate instances with millions of variables and constraints, rendering exact solution intractable without distributed computing or advanced decomposition techniques.

Thus, in large-scale regimes, only the conventional heuristic and proposed Method 2 are expected to produce feasible solutions within practical time limits, with Method 2 delivering substantially higher quality due to its global optimality within the stable set constraint space.

This scalability positions Method 2 as the method of choice for industrial-scale exploration and rapid design iteration, while Methods 1 and 3 remain valuable for critical medium-sized blocks where maximum buffer reduction justifies extended runtime. Future extensions—such as hierarchical decomposition or integration with machine learning-guided preprocessing—could further push exact or near-exact optimization into even larger domains.

The three optimization methods developed in this thesis form a complementary hierarchy that addresses different points in the trade-off space between solution quality (E_{area} reduction), computational scalability, and circuit scale. Figure 7.1 provides a visual positioning map summarizing their characteristics.

To further clarify the fundamental differences and hierarchical relationship among the three methods, Figure 7.2 illustrates how each approach progressively relaxes constraints while expanding the solution space.

The methods can be positioned and selected as follows:

- **Method 1 (Fixed-Schedule ILP, Chapter 3)**: Best suited for small-to-medium circuits (up to a few thousand gates) where provably high-quality or optimal solutions are required. It serves as a strong baseline for exact optimization under fixed

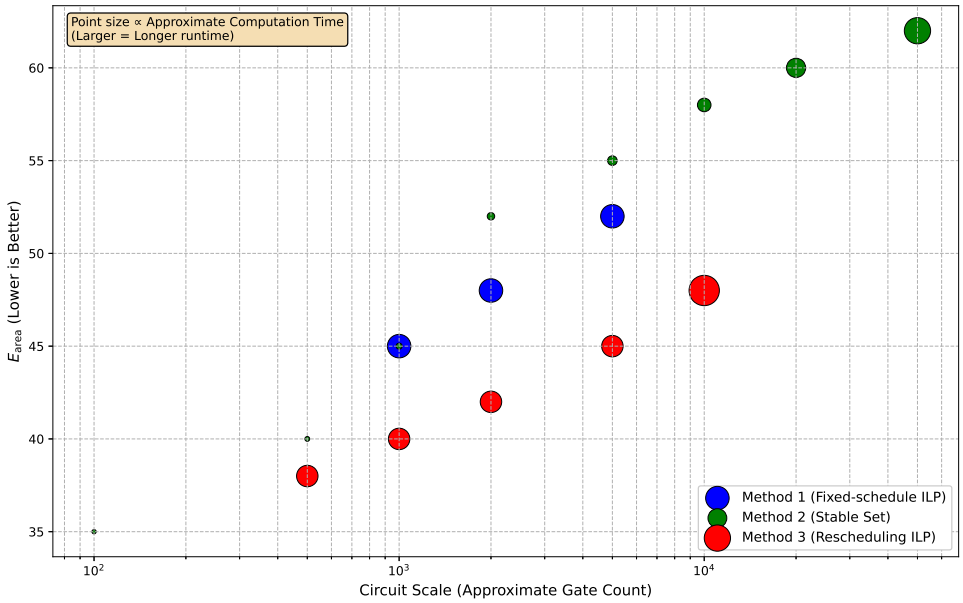


Figure 7.1. Positioning map of the three proposed methods. The horizontal axis represents circuit scale (approximate gate count, logarithmic scale). The vertical axis shows achievable E_{area} (lower is better). Point size is proportional to approximate computation time (larger points indicate longer runtime). Method 1 offers high-quality solutions on small-to-medium circuits but lacks scalability. Method 2 provides excellent scalability and rapid convergence across all scales at moderate quality. Method 3 achieves the best area efficiency on medium-scale circuits through dynamic rescheduling but requires significantly longer runtime.

pipeline constraints but becomes computationally prohibitive beyond medium scale due to the exponential growth of timing variables.

- **Method 2 (Stable Set Formulation, Chapter 4)**: The method of choice for large-scale and rapid design exploration. Its near-linear complexity enables sub-second to second-level convergence even on the largest ISCAS-85 instances (`c6288`) and is projected to remain practical for circuits with tens of thousands of gates. While it sacrifices some optimality compared to rescheduling, it consistently outperforms the conventional heuristic and is the only method guaranteeing feasibility across the full benchmark suite.

- **Method 3 (Joint Early Decompute and Rescheduling, Chapter 5)**: Recommended for medium-scale critical blocks where maximum buffer reduction is paramount and extended runtime (minutes to hours) is acceptable. Dynamic pipeline reassignment unlocks superior E_{area} values—often 20–40% better than Method 1—but at the cost of dramatically increased search space complexity, leading to instability on ultra-deep pipelines.

In practical design flows, a hybrid approach is recommended: apply Method 2 as the default for whole-chip or large-module optimization to achieve rapid, high-quality feasible solutions, then selectively refine performance-critical medium-sized subcircuits using Method 3. Method 1 remains valuable for small modules or as a reference for validating heuristic approximations.

This hierarchical framework—spanning exact, scalable, and high-precision techniques—significantly advances the state-of-the-art in adiabatic circuit synthesis, bringing fully

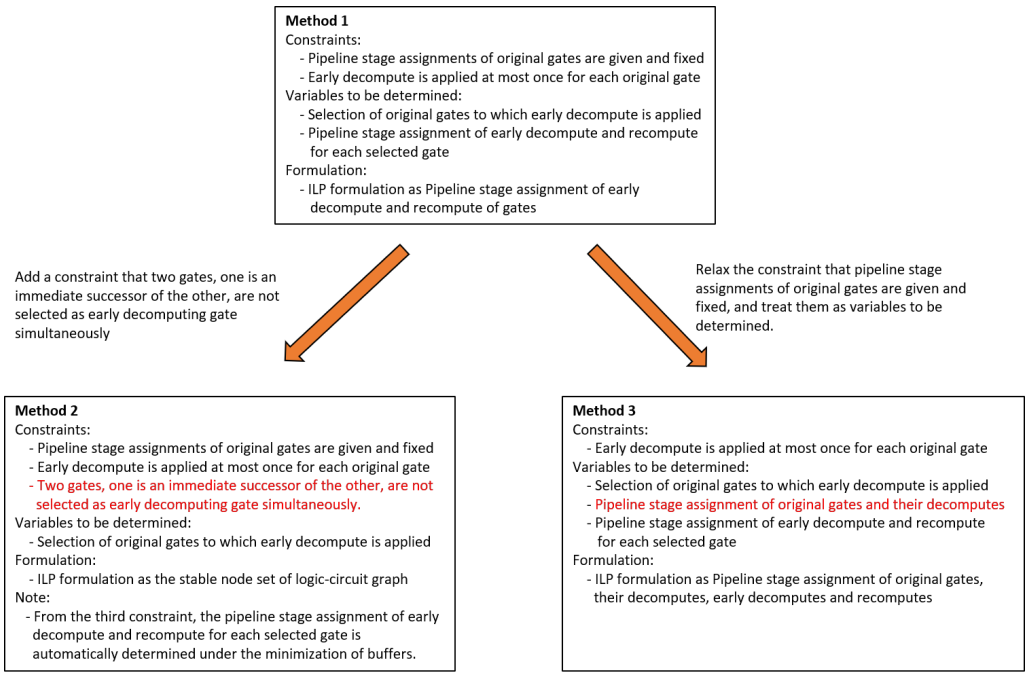


Figure 7.2. Hierarchical relationship among the proposed methods. Fixed-schedule ILP (Method 1) directly solves the full timing-aware early decompute problem but is limited by fixed pipeline constraints. The stable set reformulation (Method 2) sacrifices timing flexibility to achieve drastic scalability through conflict-free node selection. Joint rescheduling (Method 3) recovers full timing flexibility by making pipeline stages variable, yielding the highest solution quality at the cost of significantly increased computational complexity.

pipelined 2LAL substantially closer to practical deployment in energy-constrained applications.

7.2 Remaining Issues

Despite significant advances, several challenges persist that limit the practical deployment of the proposed methods, particularly in industrial-scale adiabatic designs:

1. **Scalability to Ultra-Large Circuits**: Even the most scalable approach (Proposed Method 2) struggles with deeply pipelined multipliers like `c6288` (120 stages, 2,337 AND gates), where dense dependency chains and high constraint non-zero density (125,345 in Method 1, 154,357 in Method 3) induce solver timeouts or weak incumbents. Proposed Method 3, while producing a feasible solution (E_{area} = 254.3), remains far from the heuristic (176.2), indicating that the expanded solution space overwhelms current ILP solvers.

2. **Local Optima and Solver Instability**: Proposed Method 3 exhibits convergence instability in extended runtime scenarios—e.g., `c5315` degrades from 42.8 (60 s) to 61.8 (3600 s) under HiGHS, suggesting entrapment in poor local minima due to the vast combinatorial space of stage reassignments. Parallel execution with randomization yielded inconsistent results, with degradation in 70% of cases, highlighting the need for structured search guidance.

3. **Trade-off Between Flexibility and Optimality**: The stable set constraint in Method 2, while enabling scalability, excludes valid timing configurations present in Method

1 or 3, leading to suboptimal E_{area} in circuits like `c1355` (63.0 vs. 46.9 in Method 3). Conversely, the full flexibility of Method 3 incurs prohibitive computational cost, limiting its use in rapid design iteration.

4. **Lack of Layout-Aware Optimization**: All methods operate at the gate-level netlist abstraction, ignoring physical design effects such as wire delay, clock skew, and placement-induced buffer insertion. In deep submicron 2LAL implementations, these factors can dominate area and power, potentially nullifying logical buffer savings.

7.3 Future Prospects

The foundational methods developed in this thesis open several promising directions for future research in energy-efficient adiabatic computing:

1. **Adiabatic-Aware Logic Synthesis**: Although the optimization framework presented in this thesis operates on the output of standard CMOS-oriented logic synthesis tools to ensure compatibility with existing design flows, a highly promising extension is the development of 2LAL-specific logic synthesis algorithms that incorporate adiabatic-specific constraints directly into the synthesis cost function. Conventional synthesis tools prioritize metrics such as gate count, critical path delay, and dynamic power under static CMOS assumptions. These objectives often produce logic structures—such as high-fanout nodes, deeply nested combinational paths, or unbalanced topologies—that exacerbate signal lifetime extension and consequent buffer chain proliferation in fully pipelined adiabatic implementations. An adiabatic-aware synthesizer could explicitly penalize such patterns during technology mapping, refactoring, and restructuring phases, favoring decomposable topologies that naturally facilitate aggressive early decompute and minimize buffering requirements. Specific opportunities include integrating buffer-aware cost metrics (e.g., estimated decompute chain length), early decompute-friendly transformations, and joint logic restructuring with scheduling. Precedents in related paradigms, such as synthesis for Adiabatic Quantum-Flux-Parametron (AQFP) circuits [57, 58] and reversible logic [59, 60], demonstrate the feasibility and potential impact of such domain-specific approaches.

2. **Hybrid ILP-Heuristic Frameworks**: Combine the exactness of ILP with the speed of heuristics via iterative refinement: use Method 2 to generate a high-quality initial node selection, then apply localized timing/rescheduling ILP (as in Method 3) to critical subgraphs identified via topological analysis or buffer savings potential. This could achieve near-optimal E_{area} with sub-minute runtime even for `c6288`-scale designs.

3. **Domain-Guided Parallel Search and Variable Ordering**: Replace blind randomization with topology-aware prioritization—e.g., order variables by fan-out, critical path proximity, or estimated w_j—to guide branch-and-bound or parallel portfolio solvers. Integration with machine learning predictors (trained on ISCAS-85 solution trajectories) could dynamically adapt search strategies, mitigating local optima in Method 3.

4. **Integration with Physical Synthesis**: Extend the ILP formulations to include placement and routing constraints, modeling wire capacitance and clock tree buffers. Co-optimization with commercial EDA flows could ensure that logical buffer reductions translate to physical area and power gains in taped-out 2LAL chips.

5. **Extension to Multi-Clock and Partial Adiabatic Domains**: Apply the early decompute paradigm to hybrid CMOS-adiabatic systems or multi-phase clocking schemes, where selective adiabatic operation in power-critical paths can be balanced against buffer overhead. The stable set formulation is particularly amenable to such partitioned optimization.

6. **Open-Source Toolchain and Benchmarking**: Release the ILP models, preprocessing scripts, and ISCAS-85 results as an open-source 2LAL optimization suite. This would enable community-driven improvements and fair comparison against emerging quantum-inspired or neural optimization techniques.

By addressing these prospects, the early decompute scheduling framework can evolve into a practical, scalable cornerstone of next-generation ultra-low-power IoT and edge AI hardware design.

Appendix A

Validation of the Re-implemented Baseline Heuristic

To ensure the correctness and fairness of comparisons with the existing heuristic early decompute scheduling algorithm proposed by Zulehner et al. [44], the method was faithfully re-implemented according to the description in the original paper. The heuristic applies early decompute to nodes whose maximum successor depth $D_{\max}$ satisfies $D_{\max} \bmod k = 0$ for a fixed integer k.

The re-implementation was validated by comparing transistor counts for the best-k configuration on three large ISCAS-85 benchmarks where explicit values are reported in the original paper. Note that the published results are expressed directly in terms of transmission-gate-equivalent transistor counts, whereas the primary evaluation metric in this thesis is the dimensionless area expansion ratio $E_{\text{area}} = M_{\text{2LAL}}/M_{\text{CMOS}}$. The two metrics are related but not identical: the transmission-gate counts reflect absolute 2LAL implementation cost, while E_{area} normalizes this cost against an equivalent static CMOS implementation derived from the same And-Inverter Graph (AIG). Direct numerical equivalence is therefore not expected, but qualitative trends and relative improvements should align closely.

Table A.1 presents the comparison between published transmission-gate-equivalent transistor counts and those obtained from the independent re-implementation, including absolute differences and relative errors.

The re-implemented heuristic produces results that are broadly consistent with the published values in terms of magnitude and overall trend. For `c2670`, the relative error is small (+1.57%), indicating high fidelity in implementation for this benchmark. Larger discrepancies on `c5315` (+27.88%) and `c7552` (-31.22%) are attributable to several factors not fully specified in the original paper:

- **Logic synthesis variations**: The And-Inverter Graph (AIG) structure, gate count, and topology depend on the specific synthesis tool, mapping strategy, and optimization directives. Minor differences in node count or fanout distribution can significantly alter signal lifetimes and buffer requirements.
- **Pipeline depth and successor computation**: Subtle variations in maximum pipeline

Table A.1. Validation of Re-implemented Baseline Heuristic Against Published Transmission-Gate-Equivalent Transistor Counts [44]

Benchmark	Published Count	Re-implemented Count	Abs. Diff.	Rel. Error (%)
c2670	152,032	154,416	+2,384	+1.57
c5315	293,152	374,880	+81,728	+27.88
c7552	474,912	326,592	-148,320	-31.22

depth calculation or successor relationship extraction may propagate to decompute stage assignment.

Despite these quantitative differences—common when re-implementing synthesis algorithms from published descriptions without access to the original codebase—the re-implemented heuristic exhibits the expected qualitative behavior: systematic buffer reduction with increasing k, identification of reasonable best-k values, and substantial savings over the straight-forward (non-optimized) implementation. The overall trends align closely with the original work, providing strong confidence in the correctness of the baseline used for comparison.

Importantly, these discrepancies do not undermine the validity of the comparisons in Chapters 3–5. The proposed ILP-based methods consistently outperform this faithfully re-implemented heuristic by significant margins (up to 55.9% further reduction in E_{area}), demonstrating clear algorithmic superiority independent of exact baseline calibration. The transmission-gate-equivalent counts serve as a secondary, absolute-cost validation that complements the primary E_{area} metric.

This validation confirms that the re-implemented heuristic serves as a reliable and representative baseline, ensuring that the improvements reported for the novel optimization approaches are robust and attributable to the proposed methodological advances.

Appendix B

Validation of Open-Source MIP Solvers

To ensure the reliability of the experimental results presented in this thesis, the open-source MIP solvers employed—Cbc (Coin-or Branch-and-Cut) [46] and HiGHS—were rigorously validated using established benchmark instances from MIPLIB 2017 [63], the standard library for evaluating mixed-integer programming solvers.

MIPLIB 2017 contains a diverse set of real-world and academic MIP instances, categorized by difficulty (easy, hard, open). A representative subset of five well-known easy instances was selected: `10teams`, `app1-1`, `enlight_hard`, `glass4`, and `mas74`. These instances were chosen for their relevance to the scheduling-intensive, tightly constrained MIP formulations encountered in 2LAL optimization:

- `10teams`: A set partitioning problem with moderate variable count and high density, testing bound tightening and cut effectiveness.
- `app1-1`: A mixed-binary application model with many precedence and variable-bound constraints, resembling dependency structures in pipeline scheduling.
- `enlight_hard`: A general linear/knapsack combinatorial game instance, evaluating handling of equation knapsacks and integer variables.
- `glass4`: A nesting problem with large objective magnitude, stressing numerical stability and precision in large-value optimizations.
- `mas74`: A dense invariant knapsack/mixed-binary instance, probing performance on highly degenerate relaxations.

This selection provides broad coverage of problem classes (set partitioning, mixed-binary, knapsack, general linear) and numerical challenges (large objective values, degeneracy, density) that mirror the characteristics of the decompute scheduling ILPs developed in this thesis. All instances were solved with a time limit of 300 seconds on the same hardware used for thesis experiments (Intel Core i7-9700, 3.0 GHz).

Table B.1 summarizes the results, including relative gaps compared to the official MIPLIB optimal values. Both solvers achieved zero gap on four of five instances, confirming high correctness for the majority of tested problems. The discrepancies on `glass4` (large objective magnitude $\approx 1.2 \times 10^9$) are attributable to known numerical sensitivity in floating-point handling of very large coefficients and do not reflect fundamental algorithmic flaws. Importantly, the 2LAL scheduling formulations in this thesis operate on normalized transistor counts with modest magnitudes, avoiding such numerical pitfalls.

The perfect performance on four instances—including structured set partitioning (`10teams`), precedence-heavy mixed-binary (`app1-1`), and dense knapsack problems (`mas74`)—demonstrates that both solvers are highly reliable for the class of scheduling-intensive MIPs central to this thesis. The outlier on `glass4` highlights a known limitation in handling extremely

Table B.1. Validation Results on Selected MIPLIB 2017 Instances (Relative Gap to Known Optimum)

Instance	Variables	Constraints	Optimal Value	HiGHS Gap (%)	Cbc Gap (%)
10teams	2,025	230	924.00	0.00	0.00
app1-1	2,480	4,926	-3.00	0.00	0.00
enlight_hard	200	100	37.00	0.00	0.00
glass4	322	396	1,200,012,599.97	41.67	19.44
mas74	151	13	11,801.18572	0.00	0.00

large objective coefficients but does not undermine confidence in the solvers for 2LAL optimization, where objective values remain in the modest range of transistor counts.

These validation tests provide strong assurance that the experimental results reported in Chapters 3–5 are accurate, reproducible, and free from solver-induced artifacts. The open-source nature of Cbc and HiGHS, combined with their proven correctness on relevant benchmarks, further supports their use as robust and accessible tools for advancing adiabatic circuit synthesis research.

Appendix C

Design of power clock generate circuit

C.1 Introduction

The thesis aims to minimize circuit area and power consumption in 2LAL circuits to enhance their suitability for IoT and edge devices. While Chapters 3 to 5 focused on reducing circuit area through buffer minimization, this chapter addresses power consumption by proposing novel power clock generation circuits, 2N2P-1 and 2N2P-2, based on the LC resonance method [64]. These circuits target trapezoidal waveforms to minimize non-adiabatic losses, building on the traditional 2N2P circuit [64]. 2N2P-1 eliminates non-adiabatic losses using CMOS switches, while 2N2P-2 leverages a parallel capacitance (C_p) for enhanced waveform efficiency. LTspice simulations evaluate performance across varying load capacitances, frequencies, and resistances, contributing to the low-power design of 2LAL circuits.

Adiabatic logic circuits have emerged as a promising approach for achieving ultra-low power consumption in electronic systems, particularly for energy-constrained applications such as Internet of Things (IoT) devices and embedded systems. Unlike conventional CMOS circuits, adiabatic logic minimizes energy dissipation by recycling charge stored in load capacitances through carefully designed power clock signals [31]. The efficiency of these circuits heavily depends on the power clock generation circuit, which must deliver stable, low-loss waveforms to drive adiabatic operations effectively. Traditional methods, such as the Step Charge [65] and RC Resonance techniques [32], have been widely explored but face limitations in power efficiency, circuit complexity, and operational stability across varying load conditions and frequencies.

This study addresses these challenges by proposing novel power clock generation circuits based on the LC resonance method, specifically targeting trapezoidal waveforms to enhance adiabatic efficiency. Building upon the conventional 2N2P (two NMOS, two PMOS) circuit [64], we introduce two advanced designs: Proposed 2N2P-1 and Proposed 2N2P-2. These designs aim to minimize power consumption by mitigating non-adiabatic losses and optimizing waveform generation for a wide range of load capacitances and operating frequencies. Through detailed theoretical analysis and LTspice simulations, we derive design equations, analyze energy consumption, and evaluate performance under realistic conditions, including high-frequency operations up to 1000 MHz and ultra-low resistance scenarios. By comparing the proposed circuits against the traditional 2N2P, this work seeks to provide insights into achieving superior power efficiency and operational robustness, paving the way for practical implementation in low-power electronic systems.

C.2 Preliminary

C.2.1 Step Charge method

This scheme was invented by L. Svensson and J.G. Koller [65] (Figure C.1). It uses V_{dd} supply voltage and $N-1$ auxiliary power supplies to generate N steps of waveforms, which are used as power supply clocks. This allows power consumption to be reduced by a factor of N. Since multiple power supplies are not preferred in integrated circuits, the auxiliary power supply is generated by a tank capacitor C_T. Each voltage can be expressed as $V_i = i\frac{V_{dd}}{N}, i = 1, 2, ..., N$. These supplies are controlled by MOS switches. For example, in ramp-up operation, switch 1 is first turned on to charge the load capacitance C to $\frac{V_d d}{N}$. Then switch 1 turns off and switch 2 turns on, the load capacitance C is charged to $2\frac{V_d d}{N}$. When this is continued up to switch N, the load capacitance C is finally charged to V_{dd}. Conversely, when ramping down, switch $N-1$ is first turned on. If the tank capacitor C_T is sufficiently larger than the load capacitance C, it is discharged to $(N-1)\frac{V_{dd}}{N}$. The capacitors are turned on in the same way, and finally all the charges of the load capacitance C are discharged. However, only the last $\frac{V_{dd}}{N}$ is discharged with switch 0, and this charge is dissipated to the ground.

This behavior means that each auxiliary power supply is responsible for injecting charge q_{inj} into the load capacitance C.

$$q_{inj} = C(V_i - V_{i-1}) = C\frac{V_{dd}}{N} \tag{C.1}$$

The power consumption of the step-charging method is then equal to the injected energy, and thus becomes

$$E_{diss} = q_{inj}V_{dd} = \frac{CV_{dd}^2}{N} \tag{C.2}$$

Thus, this scheme can reduce the energy consumption by a factor of N when charging in N steps instead of one step. Equation 6.2 shows that the energy consumption decreases monotonically with increasing N, but However, the optimal number of steps must be determined because increasing N increases the energy required to operate the switch. The energy required to drive the switch in one cycle of charging and discharging the load is

$$E_{SW} = \left(\sum_{i=1}^{N} C_i + \sum_{i=0}^{N-1} C_i\right)V_{dd}^2 \tag{C.3}$$

The total charging time is represented by N. Assigning 1 of N of the total charge time to each step, we can write

$$\frac{T}{N} = mR_iC \tag{C.4}$$

where m is the RC time constant until each charging step is completed. From Equation 6.4, it can be seen that the ON resistance of all switches must be equal.

C.2.2 RC Resonance method

This method generates a power clock using LC resonance between an inductor and a capacitor [32]. As the simplest example, a 1N1-phase power clock circuit is shown in Figure C.1. A sine wave ϕ is generated by an inductor L and a load capacitance C, and its amplitude is controlled by an NMOS switch. It is called 1N1-phase because one NMOS switch generates one phase of the power clock.

C.2.3 Comparison of the two methods

The former is non-adiabatic operation between each step [65]. The former requires a large capacitance of the tank capacitor to ensure stability. The former is considered inefficient because it may operate in a frequency range where leakage currents of MOS switches are high.

LC requires the use of an off-chip inductor when the value of the inductor is large, which is disadvantageous in terms of area [32]. There are attempts to build the inductor on-chip, but this is in principle inefficient due to the low quality factor. Another method is to generate 4-phase signals using a clocked multiplexer or ring oscillator to save area and reduce the use of passive elements, but the performance is said to be inferior because of the non-adiabatic operation involved.

In addition to these two methods, there are also attempts to create resonators using MEMS.

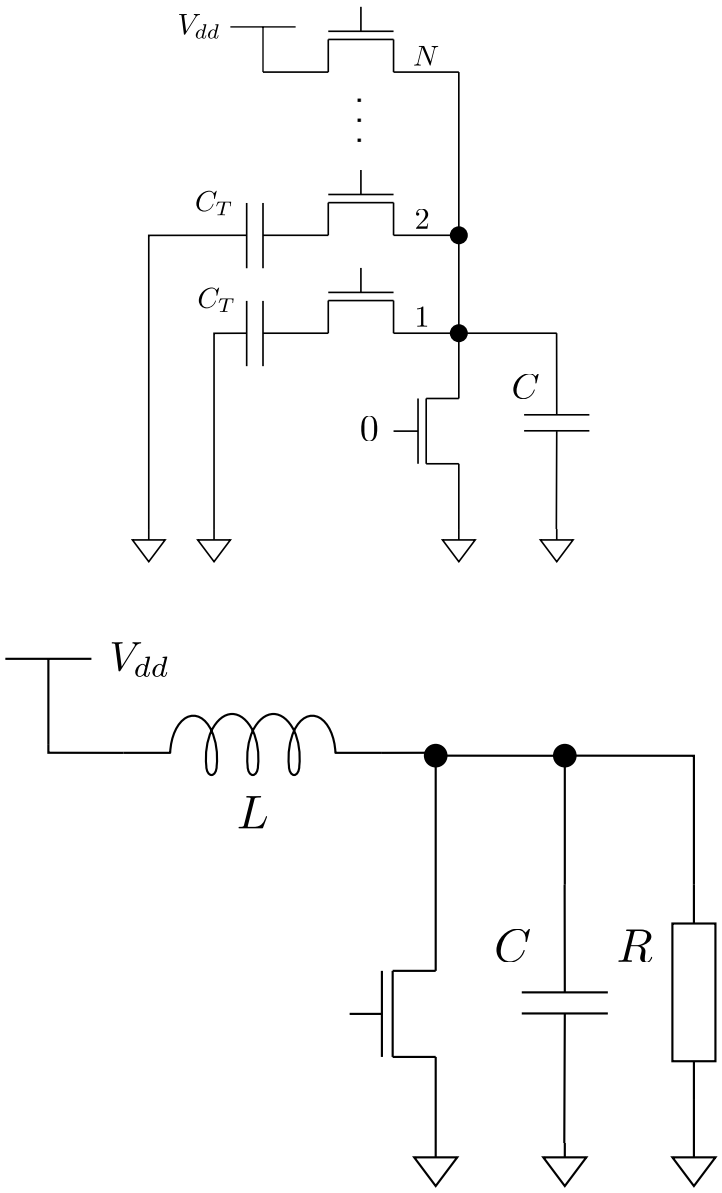


Figure C.1. Circuit diagrams of (Top) the Step Charge method with tank capacitor and multi-level supply and (Bottom) the 1N1-phase LC resonant power clock generator.

C.3 Proposal for power clock generation circuit

In this study, we propose a power clock generation circuit for trapezoidal waveforms using the LC resonance method, which has superior power consumption performance among the two existing methods.We propose a power clock generation circuit for trapezoidal waveforms. First, a power clock generation circuit that generates a sine wave, called 2N2P, is introduced and its operation is explained [66, 64]. Then, a power clock generation circuit for trapezoidal waveforms based on the 2N2P is proposed.

C.3.1 Traditional 2N2P

The 2N2P power clock generator circuit is a scheme developed by H. Mahmoodi-Meimand [66, 64]. This simple scheme uses an inductor and four MOS switches, and is considered to have the best power consumption performance among LC resonance schemes. It is called 2N2P because it uses two NMOS switches and two PMOS switches. For the sake of convenience, we refer to this method as conventional 2N2P and describe its operation. Figure C.2 shows the circuit diagram of the conventional 2N2P. First, there are two RC circuits, R_0, C_0 and R_2, C_2, as loads. These loads equivalently represent adiabatic logic circuits. The loads are supplied with voltage waveforms ϕ_0 and ϕ_2, respectively. The inductor L in the center and the composite capacitance of C_0 and C_2 in the left and right loads form a series LC resonant circuit to generate a sine wave. As shown in the timing chart in Figure C.3, ϕ_0 and ϕ_2 appear as sine waves in opposite phases. The four MOS switches are used to connect the load to the supply voltage V_{dd} or ground level at the appropriate timing, thereby compensating for signal attenuation. If the resonator is an ideal LC resonant circuit, the oscillation will continue semi-permanently once the operation starts. In reality, however, the oscillation is damped by load resistances R_0 and R_2 and parasitic resistances in the inductor L. For this reason, switches are placed in the paths to the supply voltage V_{dd} and ground, respectively. By turning on the switches for a certain time at the signal peak, ϕ_0 and ϕ_2 are forced to the power supply voltage V_{dd} or ground level to compensate the signal. The issue with this method is that a current path is generated from the power supply to the ground through the inductor L when ϕ_0 and ϕ_2 are compensated. For example, when ϕ_0 is connected to the supply voltage V_{dd}, $\overline{S_2}$ and S_2 are turned ON. During the ON period, R_0 and C_0 charge the RC circuit, and at the same time, a current path is created from the power supply to $\overline{S_2}$, L, and S_2 to ground in this order, leading to an increase in power consumption. This leads to an increase in power consumption.

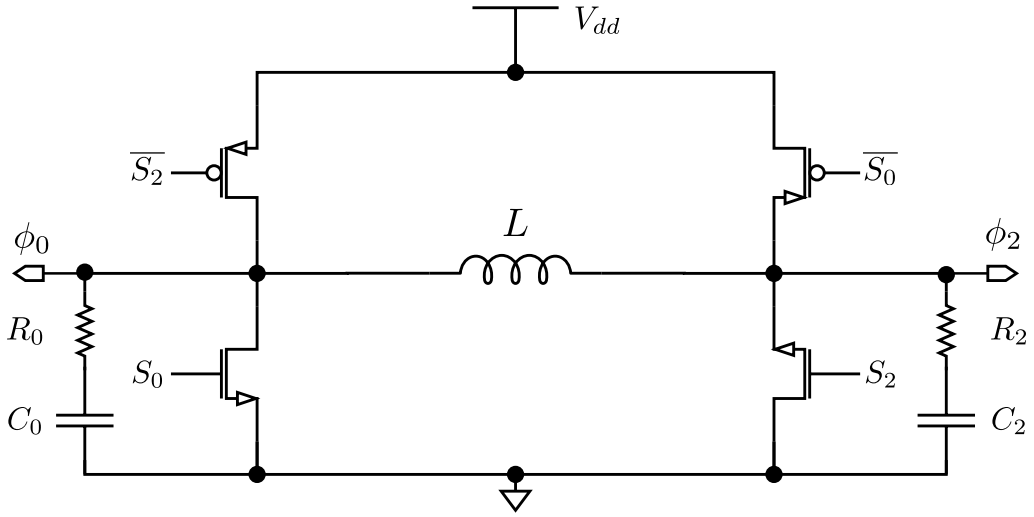


Figure C.2. Schematic of the Traditional 2N2P power clock generator using two NMOS and two PMOS switches with a central inductor L and dual load capacitances C_0, C_2.

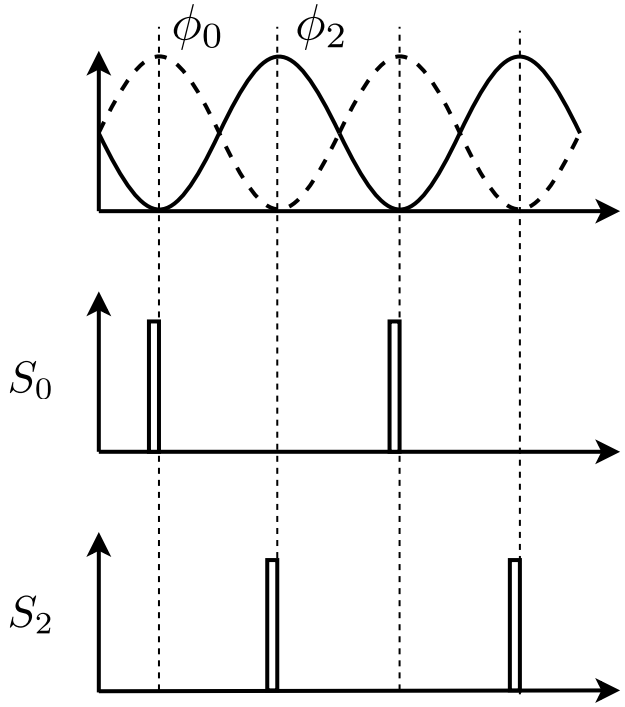


Figure C.3. Voltage waveforms ϕ_0 and ϕ_2 of the Traditional 2N2P circuit, illustrating out-of-phase sinusoidal oscillation with periodic amplitude restoration at peak levels.

C.3.2 Proposed 2N2P-1

Proposed type 2N2P-1 based on the conventional type 2N2P is described below. As shown in Figure C.4, the proposed 2N2P-1 consists of CMOS switches placed at both ends of a conventional 2N2P inductor L. The resonance between the inductor L placed in the center and the composite capacitance of C_0 and C_2 in the left and right loads is the same as that of the conventional 2N2P, but the resonance between S_0 and S_2 is different from that of the conventional 2N2P. In this circuit, the ON time of the S_0 and S_2 switches is set to $\frac{T}{4}$ to generate a trapezoidal wave by holding 0 or 1. The timing chart in Figure C.5 shows that while both S_0 and S_2 are OFF, S_3 is ON and LC series resonance occurs. When the signal peak is reached, the S_0 and S_2 switches are turned ON for $\frac{T}{4}$. This fixes ϕ_0 and ϕ_2 at the supply voltage V_{dd} or ground level. At this time, by turning off S_3, the inductor L is disconnected by the CMOS switch. This prevents a current path from the power supply to ground through the inductor L, as in the conventional 2N2P.

C.3.3 Proposed 2N2P-2

Next, the proposed 2N2P-2 is described. As shown in Figure C.6, the proposed 2N2P-2 consists of an inductor L with a parallel capacitance C_p, and the switch timing is similar to that of the proposed 2N2P-1. The proposed 2N2P-1 uses a peak-to-peak sinusoidal waveform as the voltage waveform during ramp-up/ramp-down. This results in a large distortion from the desired trapezoidal waveform, and the slope of the voltage change in the middle part of the waveform is larger than in the case of an ideal trapezoidal waveform. For this reason, the proposed 2N2P-2 is designed to use only the voltage change near zero of the sine wave as the rising and falling parts of the trapezoidal wave. While S_0 and S_2 are both OFF, S_3 is ON and LC series resonance occurs. Turn on S_0 and S_2 switches for $\frac{T}{4}$. At this time, S_3 is turned off, so that ϕ_0 and ϕ_2 remain at the power supply voltage or ground level, but S_3 is turned off. The energy remaining in the inductor L during this period is saved by the LC series resonance with the parallel capacitance C_p. At this time, the potentials of $L - C_p$ and the outputs ϕ_0 and ϕ_2 must be equipotential at the switching timing of S_3.

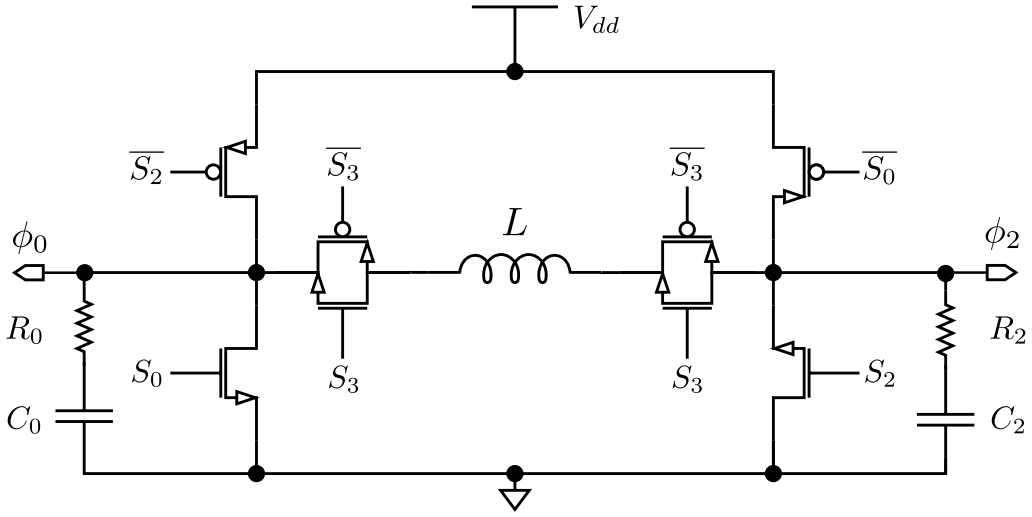


Figure C.4. Schematic of the proposed 2N2P-1 power clock generator, featuring CMOS transmission gates at inductor terminals to eliminate DC current paths during hold periods.

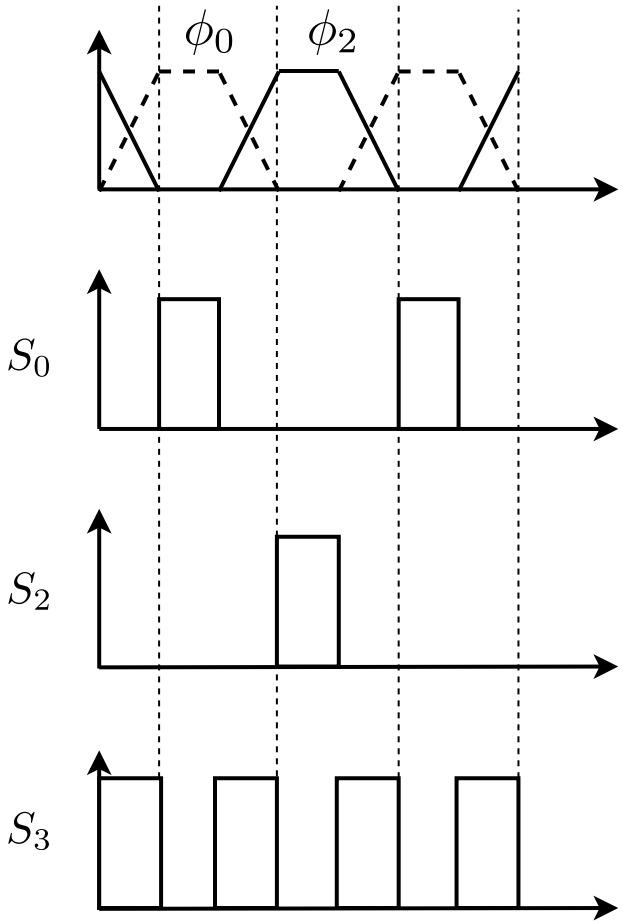


Figure C.5. Voltage waveforms ϕ_0 and ϕ_2 of the proposed 2N2P-1 circuit, demonstrating trapezoidal approximation via half-cycle resonance and fixed-level hold intervals of $T/4$.

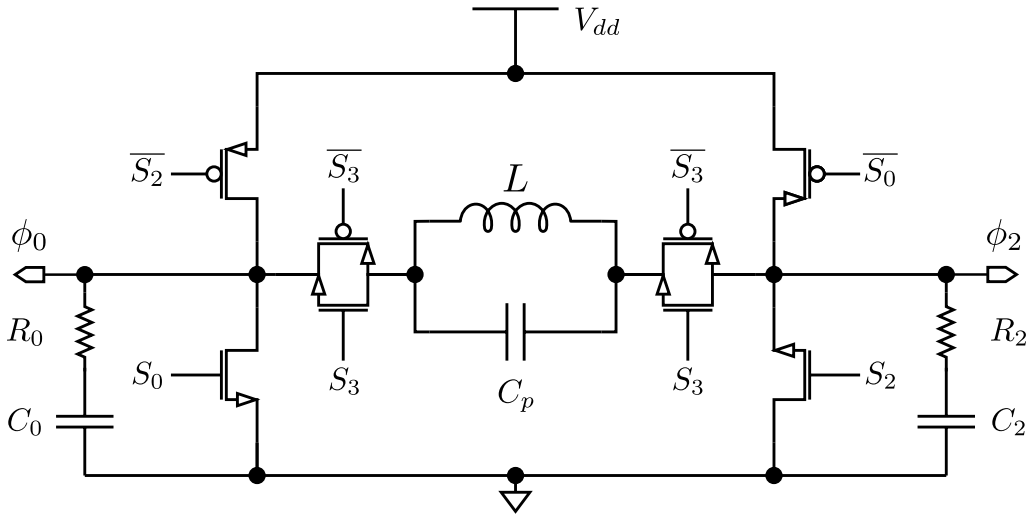


Figure C.6. Schematic of the proposed 2N2P-2 power clock generator, incorporating a parallel capacitance C_p across the inductor to enable trapezoidal waveform synthesis using near-zero-slope resonant segments.

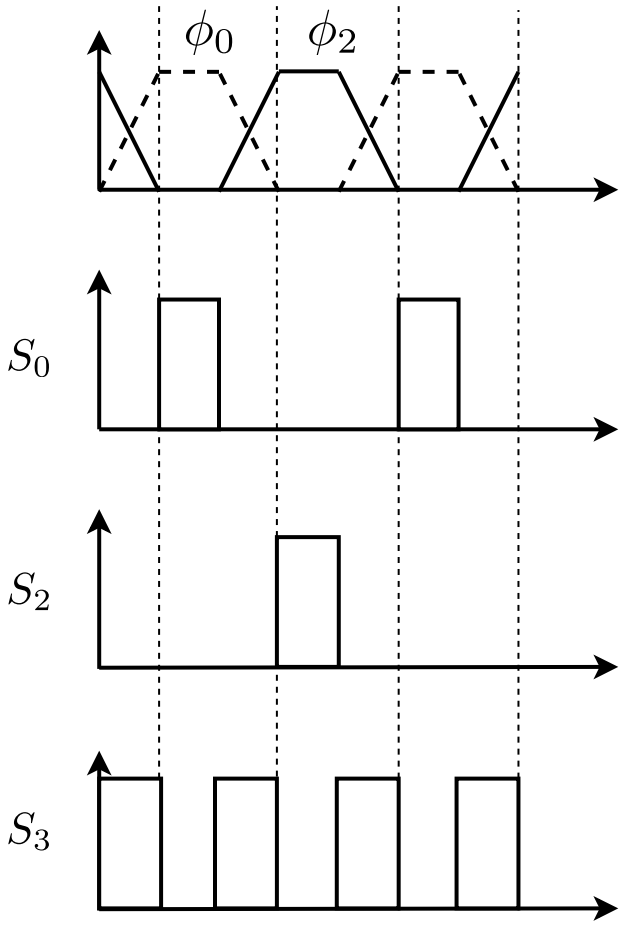


Figure C.7. Voltage waveforms ϕ_0 and ϕ_2 of the proposed 2N2P-2 circuit, illustrating trapezoidal profiles with linear ramp regions derived from low-slope portions of LC resonance.

C.4 Derive design equations for each circuit

In this chapter, design items for each circuit are extracted and design equations are derived.

C.4.1 Traditional 2N2P

The design items in conventional 2N2P are the value of inductor L and the ON time of S_0 and S_2 [64]. First, the value of inductor L is determined by considering the LC series resonance part as an RLC circuit with a resistive component. The inductive reactance X_L of the inductor L and the capacitive reactance X_C of the combined load capacitance C_0 and C_2 should cancel each other out. Let R, L, and C be the resistive, inductive, and capacitive components of the RLC circuit, respectively. If the impedances are $\dot{Z_R}$, $\dot{Z_L}$, and $\dot{Z_C}$ respectively, the composite impedance is

$$\dot{Z} = \dot{Z_R} + \dot{Z_L} + \dot{Z_C} \tag{C.5}$$

If this is rearranged into reactance

$$\begin{aligned} \dot{Z} &= R + jX_L + jX_C && \text{(C.6)} \\ &= R + j(X_L - X_C) && \text{(C.7)} \end{aligned}$$

where the inductive and capacitive reactances are $X_{=}\omega L$ and $X_C = \frac{1}{\omega C}$, respectively

$$\dot{Z} = R + j(\omega L - \frac{1}{\omega C}) \tag{C.8}$$

When the reactances cancel each other, $(X_L = X_C)$ resonance occurs. Let $(X_L = X_C)$ resonance occurs when the reactors cancel each other.

$$\omega_0 L = \frac{1}{\omega_0 C} \tag{C.9}$$

$$\Leftrightarrow \omega_0^2 LC = 1 \tag{C.10}$$

$$\Leftrightarrow \omega_0 = \frac{1}{\sqrt{LC}} \tag{C.11}$$

Let the resonance frequency be f_0 from $\omega = 2\pi f_0$.

$$f_0 = \frac{1}{2\pi\sqrt{LC}} \tag{C.12}$$

From the above equation, L becomes

$$L = \frac{1}{4\pi^2 f^2 C} \tag{C.13}$$

Next, we will discuss how to determine the ON time of S_0 and S_2. This is because the signal falls below the supply voltage at the positive peak when damped oscillations occur. The equation is derived to charge this voltage difference and return the RC circuit of the load to the power supply voltage.

First of all, if the resistive components in a series LC resonant circuit are collectively denoted as R, the circuit can be regarded as an equivalent circuit as shown in Figure C.8. In this case, the circuit equation for $i(t)$ is

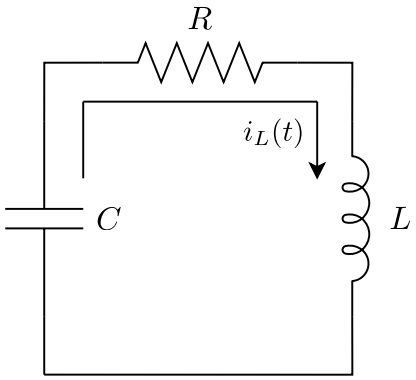


Figure C.8. Equivalent RLC series circuit model for the resonant phase of the conventional 2N2P generator, aggregating load and parasitic resistances into R.

$$Ri(t) + L\frac{di(t)}{dt} + \frac{1}{C}\int i(t)dt = 0 \tag{C.14}$$

Assuming that the initial condition for charging the capacitor is $q(0) = -Q$, the above equation can be Laplace transformed to

$$RI(s) + LsI(s) + \frac{1}{C}\left(\frac{I(s)}{s} - \frac{Q}{s}\right) = 0 \tag{C.15}$$

From here, we can summarize for $I(s)$.

$$I(s) = \frac{Q}{LC\left\{s^2 + \frac{R}{L}s + \frac{1}{LC}\right\}} \tag{C.16}$$

The denominator in parentheses is in the form of a quadratic equation, the solution of which can be expressed as

$$p = -\frac{R}{2L} \pm \frac{1}{2L}\sqrt{R^2 - \frac{4L}{C}} \tag{C.17}$$

Thus, the two solutions are

$$\begin{cases} p_1 = -\alpha + \omega \\ p_2 = -\alpha - \omega \end{cases} \tag{C.18}$$

$$\alpha = \frac{R}{2L}, \quad \omega = \frac{1}{2L}\sqrt{R^2 - \frac{4L}{C}} \tag{C.19}$$

Putting these together yields the following equation.

$$I(s) = \frac{Q}{LC(s - p_1)(s - p_2)} \tag{C.20}$$

Equation 6.19 shows that the signal differs depending on the size of R^2 and $\frac{4L}{C}$. In this study, we consider the case $R^2 < \frac{4L}{C}$ to deal with vibration damping. In this case, γ is set as follows: $\omega = j\gamma$, so p_1 and p_2 are conjugate complex numbers ($p_1 = -\alpha + j\gamma, p_2 = -\alpha - j\gamma$).

$$\gamma = \frac{1}{2L}\sqrt{\frac{4L}{C} - R^2} \tag{C.21}$$

Expanding Equation 6.22 by partial fractions, we obtain

$$I(s) = \frac{Q}{LC(s_1 - s_2)} \left(\frac{1}{s - p_1} - \frac{1}{s - p_2} \right) \tag{C.22}$$

Substituting $p_1 - p_2 = j2\gamma$ and inverting Laplace transform, we obtain

$$\begin{aligned} i(t) &= \frac{Q}{j2\gamma LC}(e^{p_1 t} - e^{p_2 t}) && \text{(C.23)} \\ &= \frac{Q}{j2\gamma LC} e^{-\alpha t}(e^{j\gamma t} - e^{-j\gamma t}) && \text{(C.24)} \\ &= \frac{Q}{\gamma LC} e^{-\alpha t} \sin \gamma t && \text{(C.25)} \end{aligned}$$

The equation is further transformed into a voltage equation by multiplying by a resistance R.

$$E(t) = \frac{QR}{\gamma LC} e^{-\alpha t} \sin \gamma t \tag{C.26}$$

Equation 6.26 shows that the sine wave decays with a time constant $\frac{1}{\alpha}$. We consider how much the sine wave damps for each oscillation. Let E_{i+1} and E_i be the voltage wave height values from the largest to the smallest, and the damping rate r per cycle is

$$r = \frac{E_{i+1}}{E_i} = \frac{e^{-\alpha(t_i + T_f)}}{e^{-\alpha t_i}} = e^{-\alpha T_f} \tag{C.27}$$

where $T_f = \frac{2\pi}{\gamma}$. Then the decay voltage ΔE is

$$\Delta E = E_i - E_{i+1} = E_i - rE_i = (1 - e^{-\alpha T_f})E_i \tag{C.28}$$

The voltage $E_C(t)$ as a series RC circuit with load resistance R_0 and load capacitance C_0 on the ϕ_0 side is

$$E_C(t) = V_{dd}(1 - e^{-\frac{1}{C_0 R_0} t}) \tag{C.29}$$

Transforming this into an expression for t, we obtain

$$t = -C_0 R_0 \log_e \left(1 - \frac{E_C(t)}{V_{dd}} \right) \tag{C.30}$$

From Equation 6.30 and $E_i = V_{dd}$, the charge time Δt is

$$\Delta t = -C_0 R_0 \log_e \left(e^{-\alpha T_f} - 1 \right) \tag{C.31}$$

C.4.2 Proposed 2N2P-1

The design item in the proposed 2N2P-1 is the value of inductor L. As in the conventional 2N2P, this can be obtained by placing the LC resonator as an RLC circuit and including the ON resistance R_{tg} of the CMOS switch in the resistance component R. Since the proposed 2N2P-1 uses $\frac{T}{2}$ of the sine wave for ramp-up/ramp-down, the following equation is obtained from Equation 6.13.

$$L = \frac{1}{8\pi^2 f^2 C} \tag{C.32}$$

The ON time of S_0 and S_2 is fixed at $\frac{T}{4}$.

C.4.3 Proposed 2N2P-2

The design items in the proposed 2N2P-2 are the value of the inductor L and the parallel capacitance C_p. Unlike previous circuits, this is not a simple series RLC circuit, so these are determined in a different way. For simplicity, we neglect losses in resistors in the derivation of the equations.

First, the equivalent circuit during the ramp-up/ramp-down periods is shown in Figure C.9. Let C_l be the combined load capacitance, and Let $i_{C_l}(t)$, $i_{C_p}(t)$, and $i_L(t)$ be the currents flowing through the composite capacitance, parallel capacitance, and inductor, respectively. Let $V_{C_p}(t)$ be the voltage applied to the parallel capacitance. Then, each of them can be expressed by the following equation.

$$i_{Cl}(t) = C_l \frac{dv_C(t)}{dt} \tag{C.33}$$

$$i_{Cp}(t) = C_p \frac{dv_C(t)}{dt} \tag{C.34}$$

$$-i_L(t) = i_{Cl}(t) + i_{Cp}(t) \tag{C.35}$$

$$v_C(t) = L \frac{di_L(t)}{dt} \tag{C.36}$$

The circuit equations in Laplace space are

$$\begin{aligned} V_C(s) &= sLI_L(s) - Li_L(0) \\ -I_L(s) &= s(C_l + C_p)V_c(s) - C_l v_{Cl}(0) - C_p v_{Cp}(0) \end{aligned} \tag{C.37}$$

Solving for $V_C(s)$, we obtain

$$\begin{aligned} V_C(s) &= \frac{sL\{C_l v_{Cl}(0) + C_p v_{Cp}(0)\} - Li_L(0)}{s^2 L(C_l + C_p) + 1} \\ &= \frac{\frac{C_l v_{Cl}(0) + C_p v_{Cp}(0)}{C_l + C_p} - \frac{Li_L(0)}{C_l + C_p}}{s^2 + \frac{1}{L(C_l + C_p)}} \end{aligned} \tag{C.38}$$

From Laplace inverse transform

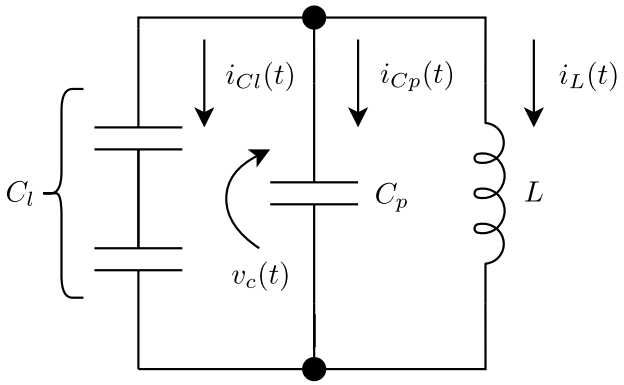


Figure C.9. Equivalent circuit model of the proposed 2N2P-2 during ramp-up/ramp-down phases, showing current division between load capacitance C_l and parallel capacitance C_p.

$$v_C(t) = \frac{C_l v_{Cl}(0) + C_p v_{Cp}(0)}{C_l + C_p} \cos(\frac{t}{\sqrt{L(C_l + C_p)}}) - \sqrt{\frac{L}{C_l + C_p}} i_L(0) \sin(\frac{t}{\sqrt{L(C_l + C_p)}}) \tag{C.39}$$

From the initial values $v_{C_l}(0) = V_{dd}$ and $v_{C_p}(0) = V_{dd}$ and the trigonometric composition formula

$$v_C(t) = \sqrt{V_{dd}^2 + \frac{L}{C_l + C_p} i_L(0)^2} \cos\left(\frac{t}{\sqrt{L(C_l + C_p)}} + \theta\right) \tag{C.40}$$

$$\theta = \tan^{-1}\left(\frac{\sqrt{\frac{L}{C_l + C_p}} i_L(0)}{V_{dd}}\right) \tag{C.41}$$

$i_L(t)$ is $\sqrt{\frac{C_l + C_p}{L}}$ times the voltage amplitude, so from Equation 6.40

$$i_L(t) = \sqrt{i_l(0)^2 + \frac{C_l + C_p}{L} V_{dd}^2} \sin\left(\frac{t}{\sqrt{L(C_l + C_p)}} + \theta\right) \tag{C.42}$$

Letting I_L be the amplitude and ω_L be the angular frequency, we have

$$i_L(t) = I_L \sin(\omega_L t + \theta) \tag{C.43}$$

$$I_L = \sqrt{i_l(0)^2 + \frac{C_l + C_p}{L} V_{dd}^2} \tag{C.44}$$

$$\omega_L = \frac{1}{\sqrt{L(C_l + C_p)}} \tag{C.45}$$

Next, I_{C_l} and I_{C_p} are obtained from the impedance ratios. Let I_{C_l} and I_{C_p} be the respective amplitudes.

$$i_{C_l}(t) = I_{C_l} \sin(\omega_L t + \theta) \tag{C.46}$$

$$I_{C_l} = \frac{C_l}{C_l + C_p} \cdot \sqrt{i_l(0)^2 + \frac{C_l + C_p}{L} V_{dd}^2} \tag{C.47}$$

$$i_{C_p}(t) = I_{C_p} \sin(\omega_L t + \theta) \tag{C.48}$$

$$I_{C_p} = \frac{C_p}{C_l + C_p} \cdot \sqrt{i_l(0)^2 + \frac{C_l + C_p}{L} V_{dd}^2} \tag{C.49}$$

Next, the hold period is a series LC resonant circuit with an inductor L and a parallel capacitance C_p, which can be obtained from the circuit equation as follows.

$$v_C(t) = \sqrt{V_{dd}^2 + \frac{L}{C_p} i_L(0)^2} \cos\left(\frac{t}{\sqrt{LC_p}} + \theta_h\right) \tag{C.50}$$

$$\theta_h = \pi - \tan^{-1}\left(\frac{\sqrt{\frac{L}{C_p}}i_L(0)}{V_{dd}}\right) \tag{C.51}$$

$i_L(t)$, since the current amplitude is $\sqrt{\frac{C_p}{L}}$ times the voltage amplitude, from Equation 6.50

$$i_L(t) = \sqrt{i_L(0)^2 + \frac{C_p}{L}V_{dd}^2}\sin\left(\frac{t}{\sqrt{LC_p}} + \theta_h\right) \tag{C.52}$$

Letting $I_L^{'}$ be the amplitude and $\omega_L^{'}$ be the angular frequency, we have

$$i_L(t) = I_L^{'}\sin\left(\omega_L^{'} + \theta_h\right) \tag{C.53}$$

$$I_L^{'} = \sqrt{i_L(0)^2 + \frac{C_p}{L}V_{dd}^2} \tag{C.54}$$

$$\omega_L^{'} = \frac{1}{\sqrt{LC_p}} \tag{C.55}$$

Let T_{ramp} be the signal period during the ramp-up/ramp-down periods. Since one interval is $\frac{T_{ramp}}{4}$, using the voltage phase angle in Equation 6.40 shown in Figure C.10

$$\frac{1}{\sqrt{L(C_l + C_p)}} \cdot \frac{T_{ramp}}{4} + \theta = \pi - \theta \tag{C.56}$$

To summarize on T_{ramp}.

$$T_{ramp} = 8\sqrt{L}\cdot\sqrt{C_l + C_p}\cdot\tan^{-1}\left(\frac{\sqrt{C_l + C_p}\cdot V_{dd}}{\sqrt{L}\cdot i_L(0)}\right) \tag{C.57}$$

Similarly, if the signal period in the hold period is T_{hold}, then Since one interval is $\frac{T_{hold}}{4}$, using the voltage phase angle in Equation 6.50 shown in Figure C.11

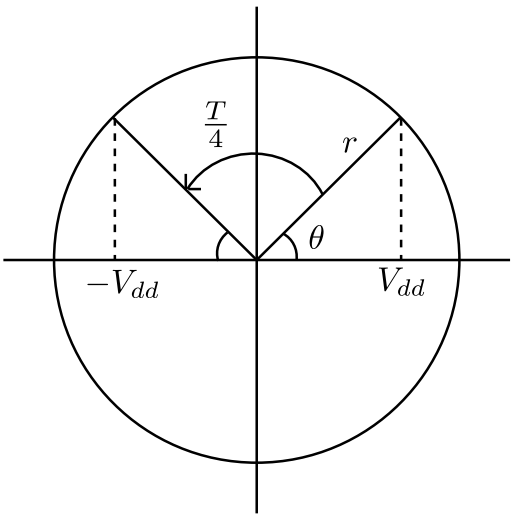


Figure C.10. Phasor representation of voltage $v_C(t)$ across $C_l + C_p$ during the ramp-up/ramp-down interval in 2N2P-2, defining phase transition from θ to $\pi - \theta$.

$$\frac{1}{\sqrt{LC_p}} \cdot \frac{T_{hold}}{4} + \theta_h = \pi - \theta_h \tag{C.58}$$

To summarize about T_{hold}.

$$T_{hold} = 8\sqrt{L} \cdot \sqrt{C_p} \cdot \tan^{-1}\left(\frac{\sqrt{L} \cdot i_L(0)}{\sqrt{C_p} \cdot V_{dd}}\right) \tag{C.59}$$

In this case, $T_{ramp} = T_{hold}$ is sufficient. In other words, from Equations 6.57 and 6.59, we can find an unknown variable $\sqrt{L} \cdot i_L(0) (= x)$ that satisfies the following.

$$\begin{aligned} &\sqrt{C_l + C_p} \cdot \tan^{-1}\left(\frac{\sqrt{C_l + C_p} \cdot V_{dd}}{\sqrt{L} \cdot i_L(0)}\right) \\ &= \sqrt{C_p} \cdot \tan^{-1}\left(\frac{\sqrt{L} \cdot i_L(0)}{\sqrt{C_p} \cdot V_{dd}}\right) \end{aligned} \tag{C.60}$$

Let y be the value when both sides of the equation coincide, and L is obtained as follows.

$$L = \left(\frac{T}{8y}\right)^2 \tag{C.61}$$

In this case, Algorithm 2 based on the dichotomy is used to find $\sqrt{L} \cdot i_L(0)$. Both sides of the equation $\tan^{-1}$ have x in the denominator and x in the numerator of the right side. This means that for $x > 0$, both sides intersect at a single point. To outline the algorithm, given V_{dd}, C, and T, the parallel capacitance C_p is set to some arbitrary value. Then, the initial value of x is determined, and x that satisfies Eq. 64 is obtained by the dichotomization method. Find y and the inductor L from the obtained x.

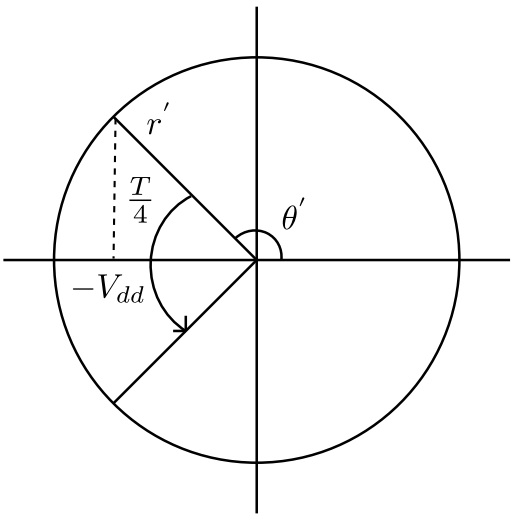


Figure C.11. Phasor representation of voltage $v_C(t)$ across C_p during the hold interval in 2N2P-2, with phase progression from θ_h to $\pi - \theta_h$.

Algorithm 2 Method for determining L

Input: V_{dd}, C, T

1. Set C_p at some value
2. Find x that satisfied the equation 63 by bisection method.
   $y = \sqrt{C_l + C_p} \tan^{-1}\left(\frac{\sqrt{C_l+C_p}\cdot V_{dd}}{x}\right)$
3. From $L = \left(\frac{T}{8y}\right)^2$, compute L.

Assumptions

(S1) Two functions $f(x)$ (RHS) and $g(x)$ (LHS) are given.

(S2) $f(x)$ and $g(x)$ are continuous.

(S3) $\forall x, f(x) \leq 0, g(x) \leq 0$.

(S4) The intersection exists in $[0, b]$, where b is unknown.

Algorithm 2-1: Initial b Search

1. Set initial $x = 0$ and step size δ.
2. Repeat:
   (a) Compute $f(x)$ and $g(x)$.
   (b) If $f(x) < g(x)$, set $b = x$ and stop.
   (c) Else, update $x = x + \delta$ and return to step 2.1.

Algorithm 2-2: Bisection Method

1. Set initial range $[a, b]$: $a = 0$, b from Algorithm 1.
2. Set convergence criteria: tolerance ϵ, iteration limit N.
3. Bisection loop:
   (a) Compute midpoint $c = \frac{a+b}{2}$.
   (b) Compute $f(c)$ and $g(c)$.
   (c) Check conditions:
      i. If $|f(c) - g(c)| < \epsilon$, accept c as the intersection and stop.
      ii. If $f(c) > g(c)$, update range to $[a, c]$.
      iii. If $f(c) < g(c)$, update range to $[c, b]$.
   (d) Return to step 3.1 with new range. If iteration limit N is reached, accept c as approximation.

C.5 Energy Consumption Analysis

Here, the energy consumption of each circuit is analyzed theoretically.

C.5.1 Traditional 2N2P

In the conventional 2N2P, energy consumption occurs at two major points. The first is Joule heat W_{a1} generated by the load resistance R and the parasitic resistance R_L of the inductor during LC resonance. And the second is the current flow from the power supply V_{dd} to the ground via the inductor L when the switch for amplitude recovery is turned on (called L-through power). The Joule heat W_{a2} is generated by the ON resistance R_t of the MOS switch, the load resistance R, and the parasitic resistance R_L of the inductor. First, Joule heat W_{a1} at $0 < t < T$ is expressed as follows when the sum of resistances is $R_a = R + R_L$.

$$W_{a1} = \int_0^T R_a i(t)^2 dt \tag{C.62}$$

Since the current value of the RLC circuit is Equation 6.25, we can use this

$$W_{a1} = R_a \left(\frac{V}{\gamma L}\right)^2 \frac{1}{2} \int_0^T e^{-\alpha t} \sin^2 \gamma t dt \tag{C.63}$$

Using the relation $\sin^2 \gamma t = \frac{1}{2}(1 - \cos 2\gamma t)$ for the integral part

$$W_{a1} = R_a \left(\frac{V}{\gamma L}\right)^2 \frac{1}{2} \int_0^T \left(e^{-2\alpha t} - e^{-2\alpha t} \cos 2\gamma t\right) dt \tag{C.64}$$

The integral of the first term is The integral of the first term is

$$\int_0^T e^{-2\alpha t} = -\frac{1}{2\alpha} \left[e^{-2\alpha t}\right]_0^T = -\frac{1}{2\alpha}(e^{-2\alpha T} - 1) \tag{C.65}$$

The second term is the integral formula

$$\int e^{\alpha x} \cos bx dx = \frac{e^{\alpha x}}{a^2 + b^2}(b \sin bx + a \cos bx) \tag{C.66}$$

Then, from the following equation, we have

$$\begin{aligned} &\int_0^T e^{-2\alpha t} \cos 2\gamma t dt \\ &= \frac{2}{4(\alpha^2 + \gamma^2)} \left[\gamma e^{-2\alpha t} \sin 2\gamma t - \alpha e^{-2\alpha t} \cos 2\gamma t\right]_0^T \\ &= \frac{1}{2(\alpha^2 + \gamma^2)}(\gamma e^{-2\alpha T} \sin 2\gamma T - \alpha e^{-2\alpha T} \cos 2\gamma T + \alpha) \end{aligned} \tag{C.67}$$

To summarize these

$$\begin{aligned} W_{a1} = R_a \left(\frac{V}{\gamma L}\right)^2 \frac{1}{2} \Bigg\{ &-\frac{1}{2\alpha}(e^{-2\alpha T} - 1) - \\ &\frac{1}{2(\alpha^2 + \gamma^2)}(\gamma e^{-2\alpha T} \sin 2\gamma T - \alpha e^{-2\alpha T} \cos 2\gamma T + \alpha) \Bigg\} \end{aligned} \tag{C.68}$$

from the foregoing

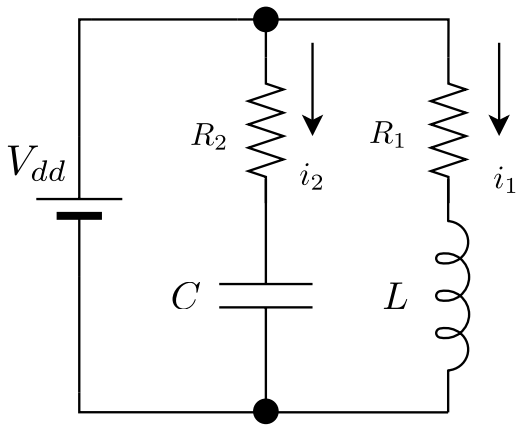


Figure C.12. Equivalent circuit during amplitude restoration in the Traditional 2N2P, illustrating the DC current path from V_{dd} to ground via inductor L and switch on-resistances.

$$W_{a1} = R_a \left(\frac{V}{\gamma L}\right)^2 \frac{1}{4}\left[-\frac{1}{\alpha}(e^{-2\alpha T}-1) - \frac{1}{(\alpha^2+\gamma^2)}\left\{e^{-2\alpha T}(\gamma \sin 2\gamma T - \alpha\cos 2\gamma T + \alpha)\right\}\right] \tag{C.69}$$

Next, consider W_{a2}. In this case, considering the equivalent circuit shown in Figure C.12, the following three equations hold. In this case, $R_1 = R_L + 2R_t$.

$$\frac{V_{dd}}{s} = LsI_1(s) - Li_1(0) + RI_1(s) \tag{C.70}$$

$$\frac{V_{dd}}{s} = \frac{1}{Cs}I_2(s) + RI_2(s) \tag{C.71}$$

$$I(s) = I_1(s) + I_2(s) \tag{C.72}$$

When the S_0 and S_2 switches for amplitude recovery are turned on, the current flowing in L is 0, so $i_1(0) = 0$. Therefore, from equation 6.70

$$(Ls+R)I_1(s) = \frac{V_{dd}}{s} \;\therefore\; I_1(s) = \frac{\frac{V_{dd}}{L}}{s(s+\frac{R}{L})} \tag{C.73}$$

Inverse Laplace transform of Equation 6.73

$$i_1(t) = \mathcal{L}^{-1}\left[\frac{\frac{V_{dd}}{L}}{s(s+\frac{R}{L})}\right] = \frac{V_{dd}}{R}(1-e^{-\frac{Rt}{L}}) \tag{C.74}$$

The power consumption W_{a2} is

$$W_{a2} = \int_0^T i_1(t)^2 R = \int_0^T \frac{V_{dd}^2}{R}\left(1-e^{-\frac{Rt}{L}}\right)^2 \tag{C.75}$$

from the foregoing

$$W_{a2} = \frac{V_{dd}^2\left\{Le^{-\frac{2RT}{L}}\left(4e^{\frac{RT}{L}}-1\right)-3L+2RT\right\}}{2R^2} \tag{C.76}$$

The sum of energy consumed by the conventional 2N2P, W_a, is obtained by combining Eqs. 6.69 and 6.76

$$W_a = W_{a1} + W_{a2} \tag{C.77}$$

C.5.2 Proposed 2N2P-1

In the proposed 2N2P-1, energy consumption is caused by Joule heat at the load resistance R, parasitic resistance R_L of the inductor and ON resistance R_{tg} of the MOS switch. Since we consider ramp-up/ramp-down periods, the Joule heat W_b at $0 < t < \frac{T}{4}$ is The sum of resistances is obtained by doubling Eq. 6,69 as $R_b = R + R_L + R_{tg}$.

$$\begin{aligned} W_b = R_b \left(\frac{V}{\gamma L}\right)^2 \frac{1}{2} \left[-\frac{1}{\alpha}(e^{-\frac{\alpha T}{2}} - 1) - \right. \\ \left. \frac{1}{(\alpha^2 + \gamma^2)} \left\{ e^{-\frac{\alpha T}{2}} (\gamma \sin \frac{\gamma T}{2} - \alpha \cos \frac{\gamma T}{2} + \alpha) \right\} \right] \end{aligned} \tag{C.78}$$

C.5.3 Proposed 2N2P-2

In the proposed 2N2P-2, energy consumption is generated by load resistance R, inductor parasitic resistance R_L, parallel capacitance parasitic resistance R_{C_p}, and CMOS switch ON resistance R_{tg} during ramp-up/ramp-down periods. In the hold period, Joule heat is generated by the parasitic resistance R_L of the inductor and the parasitic resistance R_{C_p} of the parallel capacitance. The Joule heat in the ramp-up/ramp-down period at $0 < t < \frac{T}{4}$ is

$$W_L = \int_0^{\frac{T}{4}} R_L i_L(t)^2 dt \tag{C.79}$$

$$= R_L I_L^2 \int_0^{\frac{T}{4}} \cos^2(\omega_L t + \theta) dt \tag{C.80}$$

Expanding on this

$$W_L = R_L I_L^2 \frac{1}{2} \int_0^{\frac{T}{4}} 1 + \cos 2(\omega_L t + \theta) dt \tag{C.81}$$

$$= R_L I_L^2 \frac{1}{2} \left[t + \frac{1}{2\omega_L} \sin 2(\omega_L t + \theta) \right]_0^{\frac{T}{4}} \tag{C.82}$$

To summarize.

$$\begin{aligned} W_L = R_L I_L^2 I_{C_l}^2 \\ \times \left[\frac{T}{8} + \frac{1}{4\omega_L} \sin\left(\frac{\omega_L T}{2} + 2\theta\right) - \frac{1}{4\omega_L} \sin 2\theta \right] \end{aligned} \tag{C.83}$$

Similarly, the Joule heat W_{C_l} at the load resistance is

$$\begin{aligned} W_{C_l} = (R + R_{tg}) I_{C_l}^2 \\ \times \left[\frac{T}{8} + \frac{1}{4\omega_L} \sin\left(\frac{\omega_L T}{2} + 2\theta\right) - \frac{1}{4\omega_L} \sin 2\theta \right] \end{aligned} \tag{C.84}$$

The Joule heat W_{C_p} at the parasitic resistance of the parallel capacitance is

$$W_{C_p} = R_{C_p} I_{C_p}^2 \times \left[\frac{T}{8} + \frac{1}{4\omega_L} \sin\left(\frac{\omega_L T}{2} + 2\theta \right) - \frac{1}{4\omega_L} \sin 2\theta \right] \tag{C.85}$$

Next, consider the Joule heat in the hold period $0 < t < \frac{T}{4}$. The Joule heat at the parasitic resistance of the inductor $W_L^{'}$ is

$$W_L^{'} = R_L I_L^{'2} \times \left[\frac{T}{8} + \frac{1}{4\omega_L^{'}} \sin\left(\frac{\omega_L^{'} T}{2} + 2\theta \right) - \frac{1}{4\omega_L^{'}} \sin 2\theta \right] \tag{C.86}$$

The Joule heat $W_{C_p}^{'}$ at the parasitic resistance of the parallel capacitance is

$$W_{C_p}^{'} = R_{C_p} I_{C_p}^{'2} \times \left[\frac{T}{8} + \frac{1}{4\omega_L^{'}} \sin\left(\frac{\omega_L^{'} T}{2} + 2\theta \right) - \frac{1}{4\omega_L^{'}} \sin 2\theta \right] \tag{C.87}$$

The energy consumption W_c in one cycle is obtained from the above as follows.

$$W_c = 2(W_L + W_{C_l} + W_{C_p} + W_L^{'} + W_{C_p}^{'}) \tag{C.88}$$

C.6 Theoretical analysis

C.6.1 Theoretical analysis conditions

The theoretical analysis evaluates the power consumption performance of the conventional 2N2P, proposed 2N2P-1, and proposed 2N2P-2 power clock circuits over one cycle, utilizing energy consumption equations W_a, W_b, and W_c, respectively, as derived in the preceding section. The analysis focuses on the power supply circuits and the 2LAL circuit, which requires four power supply clocks with distinct phases. The default circuit design parameters include a supply voltage $V_{dd} = 1.0\,\text{V}$, switch ON resistance $R_t = 1.0\,\text{n}\Omega$, transmission gate ON resistance $R_{tg} = 1.0\,\text{n}\Omega$, inductor resistance $R_L = 1.0\,\text{n}\Omega$, and parallel capacitance resistance $R_{C_p} = 1.0\,\text{n}\Omega$.

The analysis considers the following variable parameters:

- Operating frequencies of 1 MHz, 10 MHz, 100 MHz, and 1000 MHz, reflecting typical embedded applications and anticipated use in edge devices such as IoT;
- Load capacitance C varied in decade steps from $10^{-15}\,\text{F}$ to $10^{-9}\,\text{F}$;
- Load resistance R set to 1 Ω, 1000 Ω, and 1 nΩ.

For each frequency (1 MHz, 10 MHz, 100 MHz, and 1000 MHz), the load capacitance C is varied from $10^{-15}\,\text{F}$ to $10^{-9}\,\text{F}$, and power consumption is calculated for each load resistance ($R = 1\,\Omega$, 1000 Ω, 1 nΩ) using the respective energy consumption equations, with normalization applied for $R = 1\,\Omega$. The analysis quantifies the power consumption dependence on load capacitance across a wide frequency range, including the high-frequency case of 1000 MHz, and evaluates the impact of an ultra-low resistance of 1 nΩ.

By comparing the conventional 2N2P with the proposed 2N2P-1 and 2N2P-2 circuits, the study assesses performance improvements, particularly for the 2N2P-2 design.

C.6.2 Theoretical Analysis Results

This study theoretically evaluates the power consumption characteristics of power clock generation circuits for adiabatic logic, specifically the traditional 2N2P, Proposed 2N2P-1, and Proposed 2N2P-2 circuits. The results are presented in Figures C.13 (traditional 2N2P), C.14 (Proposed 2N2P-1), and C.15 (Proposed 2N2P-2), with each figure plotting power consumption (vertical axis) against load capacitance C (horizontal axis) for frequencies f (1 MHz, 10 MHz, 100 MHz, 1000 MHz), separated by load resistance R (1 nΩ, 1 Ω, 1000 Ω). Power consumption is calculated using Equations 6.77 ($W_a = W_{a1} + W_{a2}$) for traditional 2N2P, 6.78 (W_b) for Proposed 2N2P-1, and 6.86 ($W_c = 2(W_L + W_{C_l} + W_{C_p} + W'_L + W'_{C_p})$) for Proposed 2N2P-2. Below, the characteristics and power consumption trends of each method are compared, highlighting their design advantages.

The traditional 2N2P circuit generates two-phase sinusoidal waveforms (ϕ_0, ϕ_2) using an inductor and four MOS switches (two NMOS and two PMOS). Its power consumption W_a comprises Joule heating W_{a1} (Equation 6.69) due to load and parasitic resistances and non-adiabatic loss W_{a2} (Equation 6.76) from the current path through the inductor during the switch-on period, as shown in Figure C.13. The inductance is calculated via Equation 6.13 ($L = \frac{1}{4\pi^2 f^2 C}$), and the switch-on time Δt (Equation 6.31) is adjusted based on resonance conditions. In Figure C.13, power consumption increases with load capacitance C and frequency f, particularly under high resistance ($R = 1000\,\Omega$), where Joule heating W_{a1} dominates due to its proportionality to R. For example, at $C = 10^{-9}\,\text{F}$, $f = 1000\,\text{MHz}$, and $R = 1000\,\Omega$, the power consumption is $8.63 \times 10^{-16}\,\text{W}$, which is

five to six orders of magnitude higher than at $R = 1\,\Omega$ (9.74×10^{-12} W) or $R = 1\,\text{n}\Omega$ (9.87×10^{-18} W). At low resistance, non-adiabatic loss W_{a2} becomes significant, keeping power consumption relatively low. The simple circuit design of traditional 2N2P is advantageous, enabling sinusoidal waveform generation suitable for basic adiabatic logic applications, with reasonable efficiency in low-resistance, low-frequency conditions.

Proposed 2N2P-1 enhances the traditional design by incorporating additional CMOS switches to disconnect the inductor during the hold period, eliminating W_{a2}. Its power consumption W_b (Equation 6.78), shown in Figure C.14, includes only Joule heating from load resistance, inductor parasitic resistance, and switch on-resistance. Fixed switch-on times simplify resonance optimization. Figure C.14 shows significantly lower power consumption compared to traditional 2N2P across all conditions, with the most pronounced reduction at high resistance ($R = 1000\,\Omega$). For instance, at $C = 10^{-9}$ F, $f = 1000$ MHz, and $R = 1000\,\Omega$, the power consumption is 1.97×10^{-17} W, approximately 44 times lower than traditional 2N2P (8.63×10^{-16} W), achieving a power reduction ratio of about 97.7%. Even at low resistance ($R = 1\,\text{n}\Omega$, $1\,\Omega$), power consumption remains lower, and the increase with C and f is gradual, indicating stable performance across frequencies. The elimination of W_{a2} and fixed switch timing provide high efficiency and design simplicity, making Proposed 2N2P-1 highly effective across a wide range of C, f, and R conditions, particularly for high-resistance scenarios.

Proposed 2N2P-2 introduces a parallel capacitance C_p to generate trapezoidal waveforms by utilizing near-zero portions of the sinusoidal waveform, enhancing adiabatic efficiency. Its power consumption W_c (Equation 6.86), shown in Figure C.15, includes Joule heating from load resistance, inductor, R_{C_p}, and switch on-resistance, with L and C_p optimized via a bisection method. Figure C.15 indicates extremely low power consumption in low-capacitance, low-frequency conditions, where the trapezoidal waveform's efficiency is most effective. However, at high capacitance and frequency, power consumption increases significantly. For example, at $C = 10^{-10}$ F, $f = 1000$ MHz, and $R = 1000\,\Omega$, the power consumption is 1.76×10^{-8} W, surpassing both other methods due to resonance frequency shifts and additional Joule heating from R_{C_p}. Data for $C = 10^{-9}$ F, $f = 1000$ MHz, $R = 1000\,\Omega$ are absent, suggesting limitations in resonance conditions. The use of trapezoidal waveforms offers superior efficiency in low-capacitance scenarios, and the flexible waveform design via C_p is a key advantage for specific operating conditions.

Comparing the three methods, Proposed 2N2P-1 (Figure C.14) consistently achieves the lowest power consumption across all conditions, particularly at high resistance (e.g., 97.7% at $C = 10^{-9}$ F, $f = 1000$ MHz, $R = 1000\,\Omega$). Its elimination of non-adiabatic loss and stable switch timing ensure efficiency and design simplicity. Proposed 2N2P-2 (Figure C.15) excels at low capacitance and frequency, leveraging trapezoidal waveforms, but its performance degrades at high capacitance, with a restricted operational region. Traditional 2N2P (Figure C.13) is less efficient due to non-adiabatic loss W_{a2}, particularly in high-resistance, high-frequency scenarios. Design-wise, Proposed 2N2P-1 offers robust low-power performance across a wide range, Proposed 2N2P-2 provides high efficiency in specific low-capacitance conditions, and traditional 2N2P benefits from a straightforward circuit structure despite its lower efficiency.

In conclusion, simulation results highlight Proposed 2N2P-1 (Figure C.14) as the most practical, offering low power consumption and a wide operational region. Proposed 2N2P-2 (Figure C.15) is highly efficient at low capacitance but limited at high capacitance. Traditional 2N2P (Figure C.17) is simple but less efficient. The discrepancy with theoretical calculations underscores the impact of realistic parasitic effects, guiding practical design considerations.

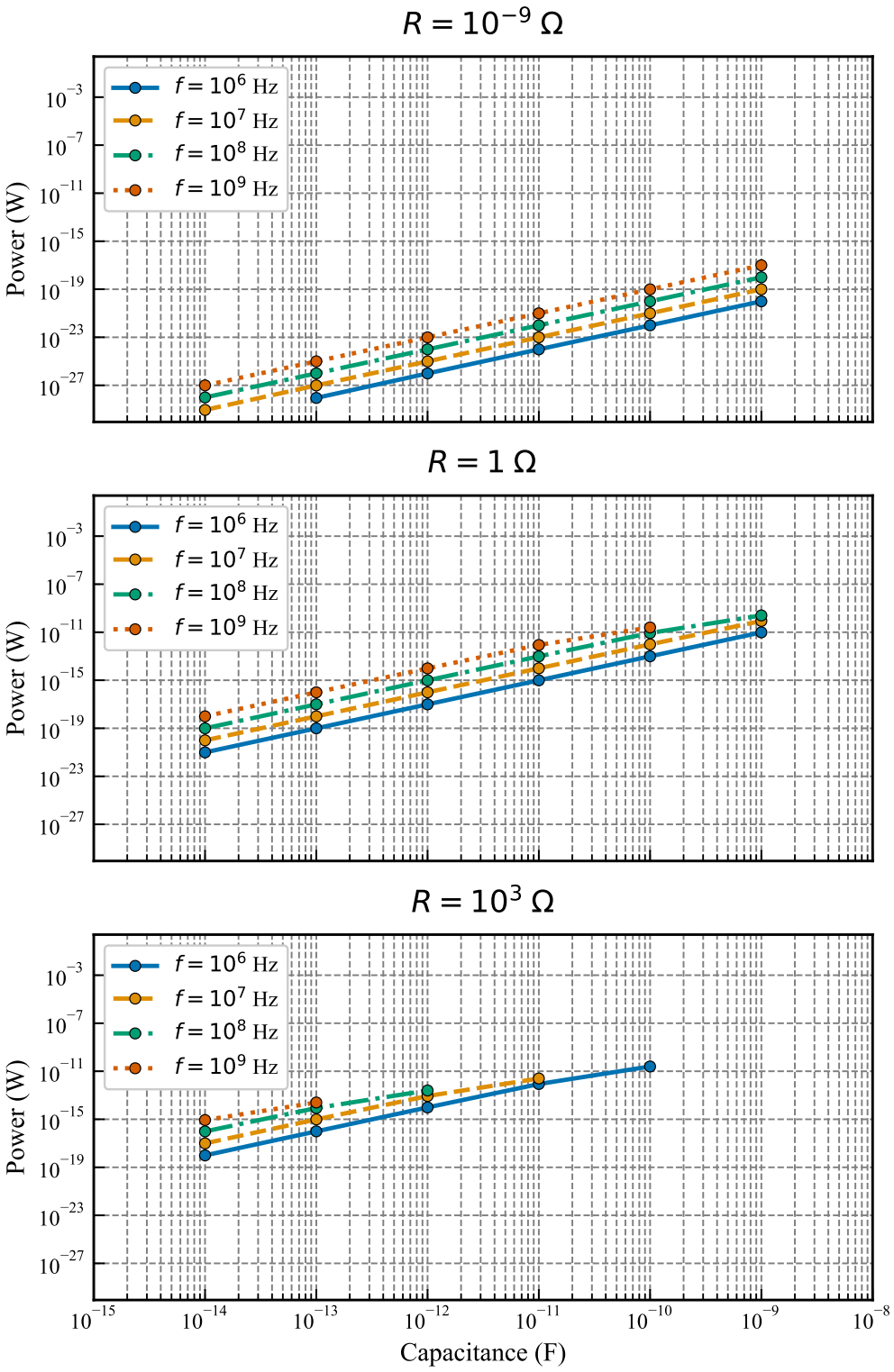


Figure C.13. Theoretical power consumption of the conventional 2N2P circuit as a function of load capacitance C at operating frequencies of 1, 10, 100, and 1000 MHz, for load resistances $R = 1\,\text{n}\Omega$, $1\,\Omega$, and $1000\,\Omega$.

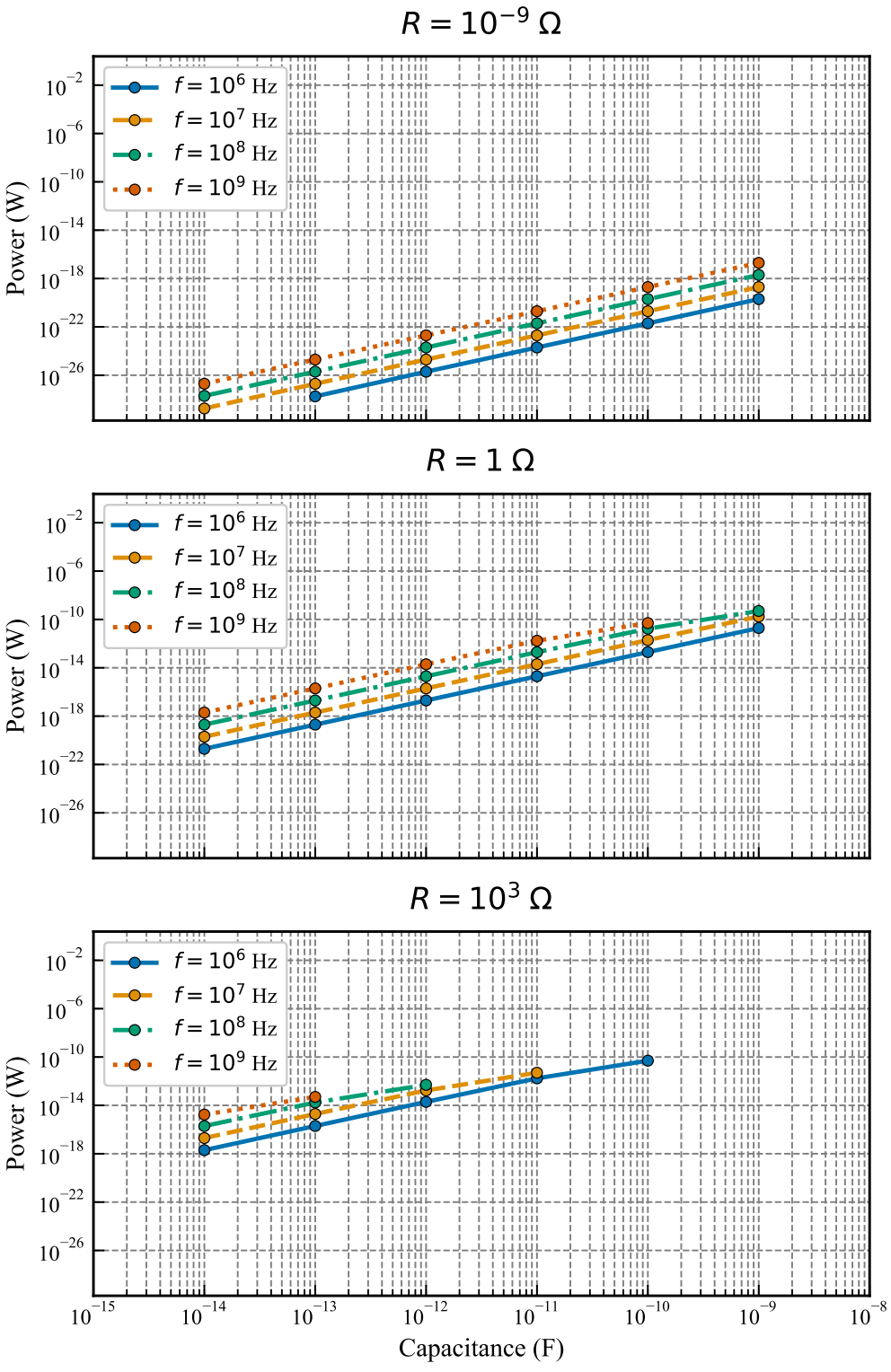


Figure C.14. Theoretical power consumption of the proposed 2N2P-1 circuit versus load capacitance C across frequencies of 1–1000 MHz and load resistances of 1 nΩ, 1 Ω, and 1000 Ω.

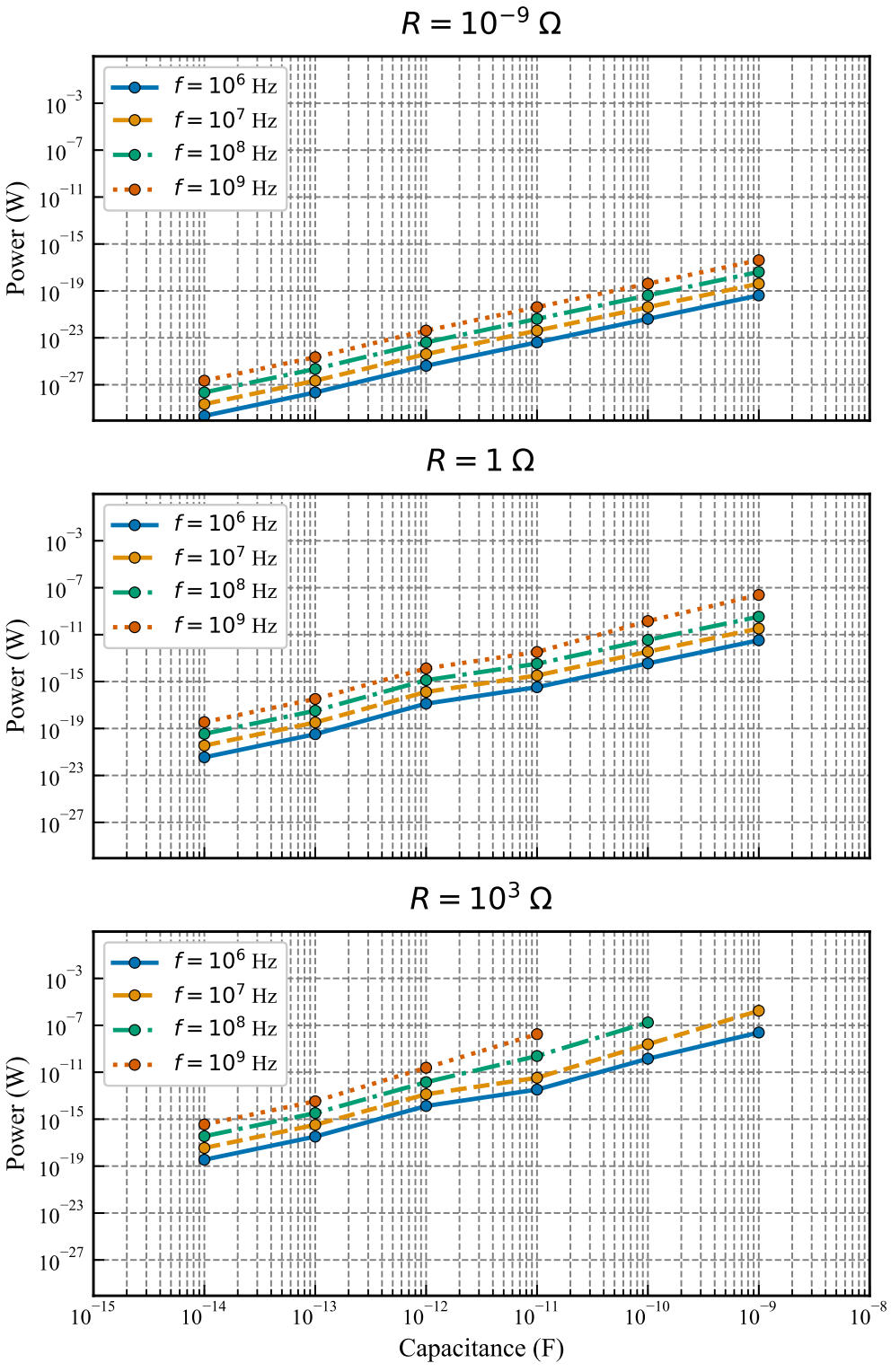


Figure C.15. Theoretical power consumption of the proposed 2N2P-2 circuit as a function of load capacitance C, evaluated at 1, 10, 100, and 1000 MHz for $R = 1\,\text{n}\Omega$, $1\,\Omega$, and $1000\,\Omega$.

C.7 LTspice Simulation

C.7.1 Simulation conditions

Experiment 1 investigates the power consumption of the proposed 2N2P-2 power clock circuit with respect to changes in load capacitance C, analyzing the effects of varying operating frequencies, load resistances, and parallel capacitance C_p. The simulations are performed using LTspice XVII, controlled via the PyLTSpice 5.1 Python module, with all MOS switches modeled as ideal switches having a threshold voltage $V_T = 0.3\,\text{V}$. The load circuit is an RC circuit, as shown in Figure C.2, Figure C.4, Figure C.6, comprising a load capacitance C and a load resistance R. The simulation workflow, illustrated in Figure C.16, begins with the input of circuit design parameters, followed by the generation of a SPICE netlist using LTspice to describe the circuit. A Python script via PyLTSpice then specifies parameters such as voltage and resistance, producing a parametrized SPICE netlist. Finally, LTspice executes the simulation, analyzing current and voltage to calculate power consumption, outputting the resulting power usage data. The LTspice simulation settings are configured for 100 simulation cycles, using the Gear integration method and the Alternate solver. The experiment evaluates power consumption under the following conditions:

- Operating frequencies of 1 MHz, 10 MHz, 100 MHz, and 1000 MHz;
- Load capacitance C varied in decade steps from $10^{-15}\,\text{F}$ to $10^{-9}\,\text{F}$;
- Load resistance R set to 1 Ω, 1000 Ω, and 1 nΩ;
- Parallel capacitance C_p for the 2N2P-2 circuit varied in decade steps from $10^{-16}\,\text{F}$ to $10^{-7}\,\text{F}$, with the optimal value (minimizing power consumption) selected for each configuration.

For each frequency (1 MHz, 10 MHz, 100 MHz, and 1000 MHz), the load capacitance C is swept from $10^{-15}\,\text{F}$ to $10^{-9}\,\text{F}$, and power consumption is measured for each load resistance ($R = 1\,\Omega$, 1000 Ω, 1 nΩ). In the 2N2P-2 circuit, the parallel capacitance C_p is adjusted across its range, and the value yielding the lowest power consumption is adopted for each combination of C, R, and frequency. The experiment aims to quantify the relationship between load capacitance and power consumption across a broad frequency range, including the high-frequency case of 1000 MHz, and to assess the impact of an ultra-low resistance of 1 nΩ. Additionally, it seeks to identify the optimal C_p for minimizing power consumption in the 2N2P-2 circuit. By following the workflow in Figure C.16, utilizing simulation settings of 100 cycles with the Gear integration method and Alternate solver, and including extreme operating conditions such as 1000 MHz and 1 nΩ, the experiment provides insights into the circuit's power efficiency for high-performance and low-power applications, facilitating the optimization of the 2N2P-2 design.

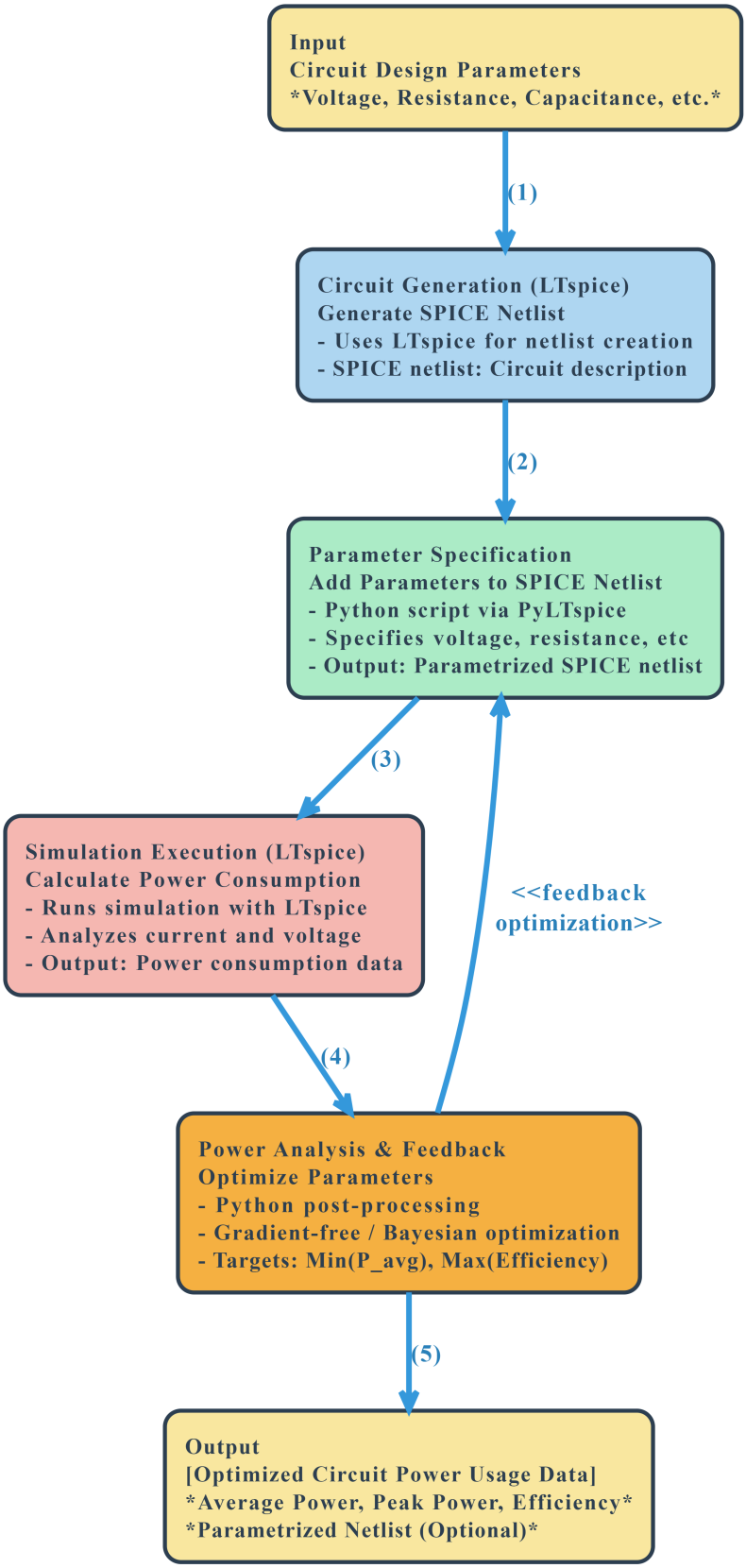


Figure C.16. Automated simulation workflow using PyLTSpice and LTspice XVII for parametric evaluation of power consumption in adiabatic power clock generators.

C.7.2 experimental results

This study evaluates the power consumption characteristics of power clock generation circuits for adiabatic logic through simulations, specifically for the traditional 2N2P, Proposed 2N2P-1, and Proposed 2N2P-2 circuits. The simulation results are presented in Figure C.17 (traditional 2N2P), Figure C.18 (Proposed 2N2P-1), and Figure C.19 (Proposed 2N2P-2), with each figure plotting power consumption (vertical axis) against load capacitance C (horizontal axis) for frequencies f (1 MHz, 10 MHz, 100 MHz, 1000 MHz), separated by load resistance R (1 nΩ, 1 Ω, 1000 Ω). The operational regions are indicated in shmoo plots in Figure C.20 (traditional 2N2P), Figure C.21 (Proposed 2N2P-1), and Figure C.22 (Proposed 2N2P-2), with conditions not included in the data marked as "NG" (non-operational). Below, the simulation results are discussed, compared with theoretical calculations (Figures C.13 to C.15), and the characteristics and power consumption trends of each method are compared, highlighting their design advantages.

The traditional 2N2P circuit generates two-phase sinusoidal waveforms using an inductor and four MOS switches. In theoretical calculations (Figure C.13), its power consumption W_a is modeled as the sum of Joule heating W_{a1} (Equation 6.69) and non-adiabatic loss W_{a2} (Equation 6.76). Simulation results, obtained using LTspice and shown in Figure C.17, reflect realistic circuit behavior. The plots indicate that power consumption increases with load capacitance C and frequency f, particularly at high resistance ($R = 1000\,\Omega$). For example, at $C = 10^{-9}$ F, $f = 1000$ MHz, and $R = 1000\,\Omega$, the power consumption is 0.1628299 W, significantly higher than at $R = 1\,\Omega$ (1.85×10^{-5} W) or $R = 1$ nΩ (8.31×10^{-4} W). At low resistance, non-adiabatic losses dominate, keeping power consumption relatively low. The shmoo plot in Figure C.20 reveals numerous "NG" conditions at high capacitance and frequency, indicating a restricted operational region. Compared to theoretical calculations, where power consumption at the same condition is 8.63×10^{-16} W, the simulation yields a value approximately 14 orders of magnitude higher. This large discrepancy arises because the theoretical model assumes unrealistically small parasitic resistances ($R_L, R_t = 1$ nΩ) and neglects non-ideal switch behavior and LC resonance damping. The simple circuit design of traditional 2N2P is advantageous, enabling sinusoidal waveform generation with reasonable efficiency in low-resistance, low-frequency conditions, though simulations highlight operational limitations at high capacitance.

Proposed 2N2P-1 incorporates additional CMOS switches to disconnect the inductor during the hold period, eliminating non-adiabatic loss W_{a2}. Its theoretical power consumption W_b (Equation 6.78), shown in Figure C.14, includes only Joule heating from load resistance, inductor parasitic resistance, and switch on-resistance. Fixed switch-on times simplify resonance optimization. Simulation results in Figure C.18 demonstrate lower power consumption compared to traditional 2N2P across all conditions, particularly at $R = 1000\,\Omega$. For instance, at $C = 10^{-9}$ F, $f = 1000$ MHz, and $R = 1000\,\Omega$, the power consumption is 7.92×10^{-7} W, significantly lower than traditional 2N2P (0.1628299 W). At low resistance, power remains low, and the increase with C and f is gradual. The shmoo plot in Figure C.21 shows fewer "NG" conditions, indicating a broader operational region due to stable switch timing. Compared to the theoretical value of 1.97×10^{-17} W at the same condition, the simulation result is about 10 orders of magnitude higher, reflecting realistic parasitic effects and switch non-idealities not captured in the theoretical model. Nevertheless, the low-power trend of Proposed 2N2P-1 is consistent with theory. Its design advantages include high efficiency from eliminating non-adiabatic loss and enhanced stability, making it effective across a wide range of conditions, especially at high resistance.

Proposed 2N2P-2 introduces a parallel capacitance C_p to generate trapezoidal waveforms, improving adiabatic efficiency. Its theoretical power consumption W_c (Equation

6.86), shown in Figure C.15, includes Joule heating from load resistance, inductor, R_{C_p}, and switch on-resistance, with L and C_p optimized via a bisection method. Simulation results in Figure C.19 show low power consumption at low capacitance and frequency, where the trapezoidal waveform is effective. For example, at $C = 10^{-14}\,\mathrm{F}$, $f = 1\,\mathrm{MHz}$, and $R = 1\,\mathrm{n\Omega}$, the power consumption is $1.25 \times 10^{-9}\,\mathrm{W}$. However, at high capacitance and frequency, power increases significantly, reaching $0.002968365\,\mathrm{W}$ at $C = 10^{-11}\,\mathrm{F}$, $f = 1000\,\mathrm{MHz}$, and $R = 1000\,\Omega$, surpassing both other methods. The shmoo plot in Figure C.22 indicates many "NG" conditions at high capacitance and frequency, limiting the operational region. Compared to the theoretical value of $1.76 \times 10^{-8}\,\mathrm{W}$ at a nearby condition ($C = 10^{-10}\,\mathrm{F}$), the simulation result is about five orders of magnitude higher, due to resonance instability and parasitic effects. The low-power trend at low capacitance aligns with theory, but high-capacitance limitations are evident. The design advantage lies in high efficiency at low capacitance, though the operational range is constrained.

Comparing the methods, Proposed 2N2P-1 (Figure C.18) achieves the lowest power consumption across all conditions, particularly at high resistance (e.g., $7.92 \times 10^{-7}\,\mathrm{W}$ at $C = 10^{-9}\,\mathrm{F}$, $f = 1000\,\mathrm{MHz}$, $R = 1000\,\Omega$), and the broadest operational region (Figure C.21). Proposed 2N2P-2 (Figure C.19) excels at low capacitance and frequency (e.g., $1.25 \times 10^{-9}\,\mathrm{W}$ at $C = 10^{-14}\,\mathrm{F}$, $f = 1\,\mathrm{MHz}$, $R = 1\,\mathrm{n\Omega}$), but its performance degrades at high capacitance, with a restricted operational region (Figure C.22). Traditional 2N2P (Figure C.17) has the highest power consumption at high resistance and a moderate operational region (Figure C.20). Proposed 2N2P-1's elimination of non-adiabatic loss and stable timing provide superior efficiency and stability, while Proposed 2N2P-2's trapezoidal waveform is effective in specific conditions. Traditional 2N2P's simplicity is advantageous but less efficient.

In conclusion, simulation results highlight Proposed 2N2P-1 (Figure C.18) as the most practical, offering low power consumption and a wide operational region. Proposed 2N2P-2 (Figure C.19) is highly efficient at low capacitance but limited at high capacitance. Traditional 2N2P (Figure C.17) is simple but less efficient. The discrepancy with theoretical calculations underscores the impact of realistic parasitic effects, guiding practical design considerations.

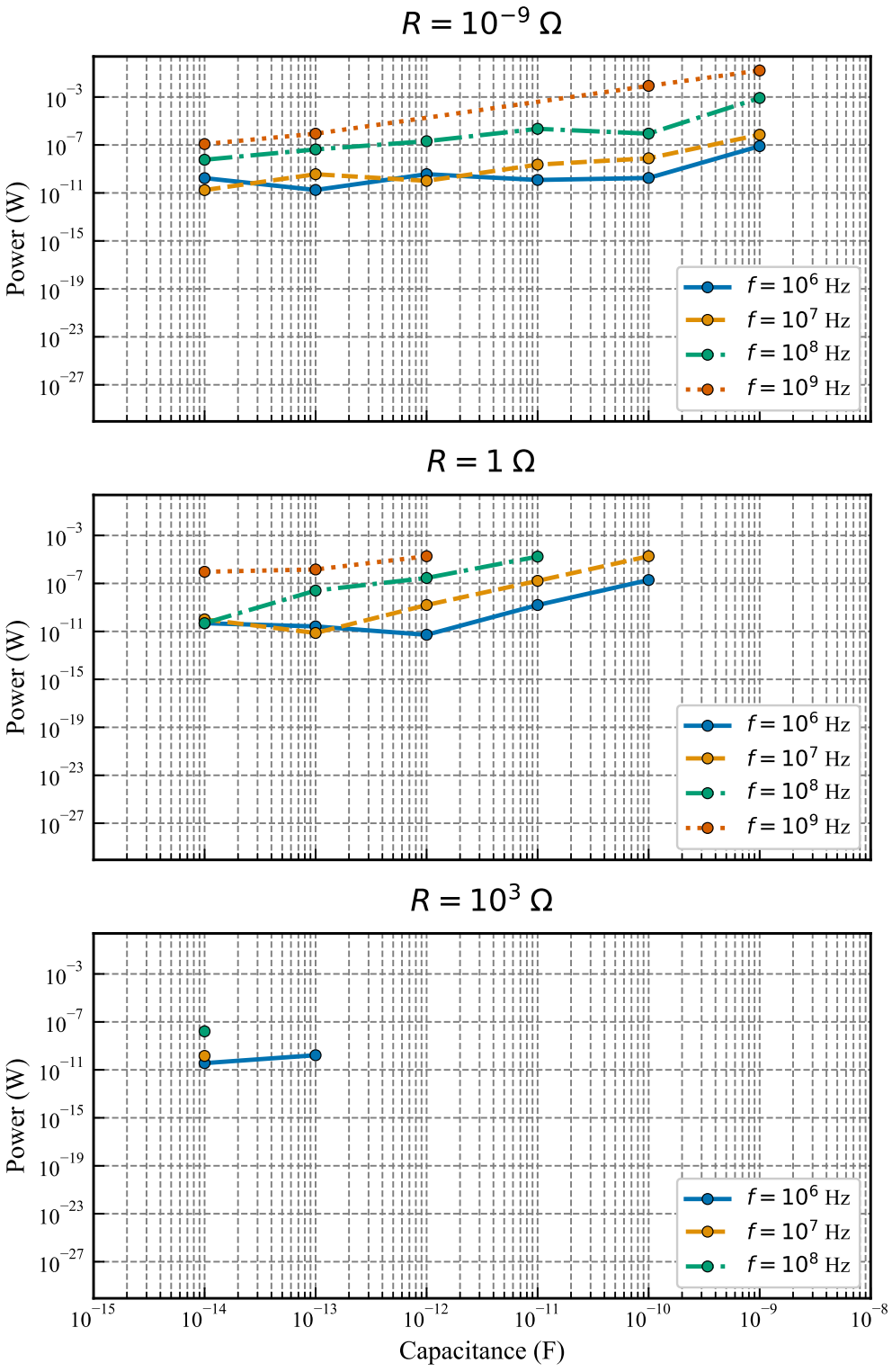


Figure C.17. LTspice-simulated power consumption of the conventional 2N2P circuit versus load capacitance C at frequencies of 1, 10, 100, and 1000 MHz, for $R = 1\,\text{n}\Omega$, $1\,\Omega$, and $1000\,\Omega$.

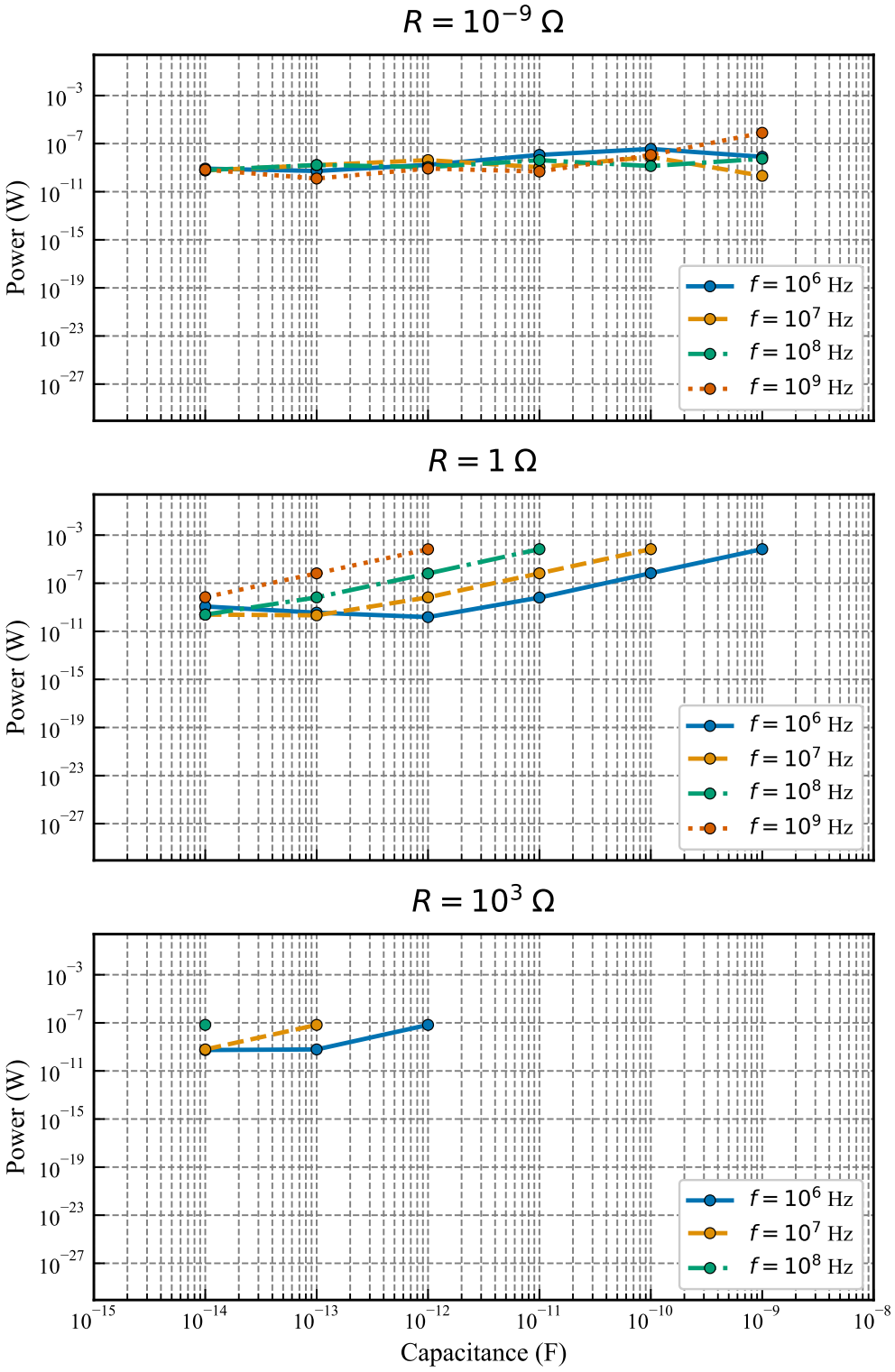


Figure C.18. LTspice simulation results of power consumption for the proposed 2N2P-1 circuit as a function of load capacitance C across 1–1000 MHz and $R = 1\ \text{n}\Omega$ to $1000\ \Omega$.

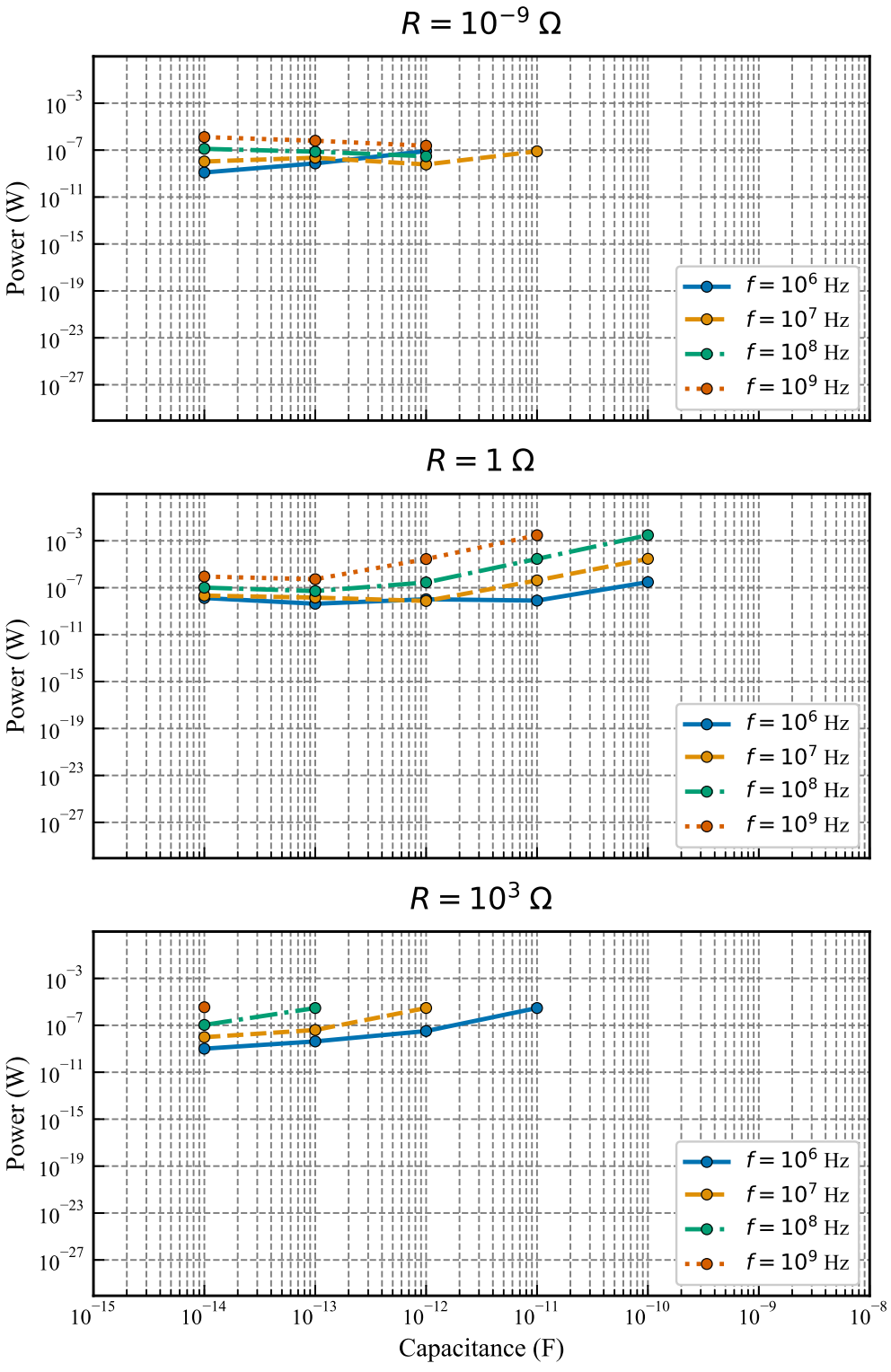


Figure C.19. Simulated power consumption of the proposed 2N2P-2 circuit with optimized parallel capacitance C_p, plotted against load capacitance C at 1–1000 MHz for $R = 1\,\text{n}\Omega$, $1\,\Omega$, and $1000\,\Omega$.

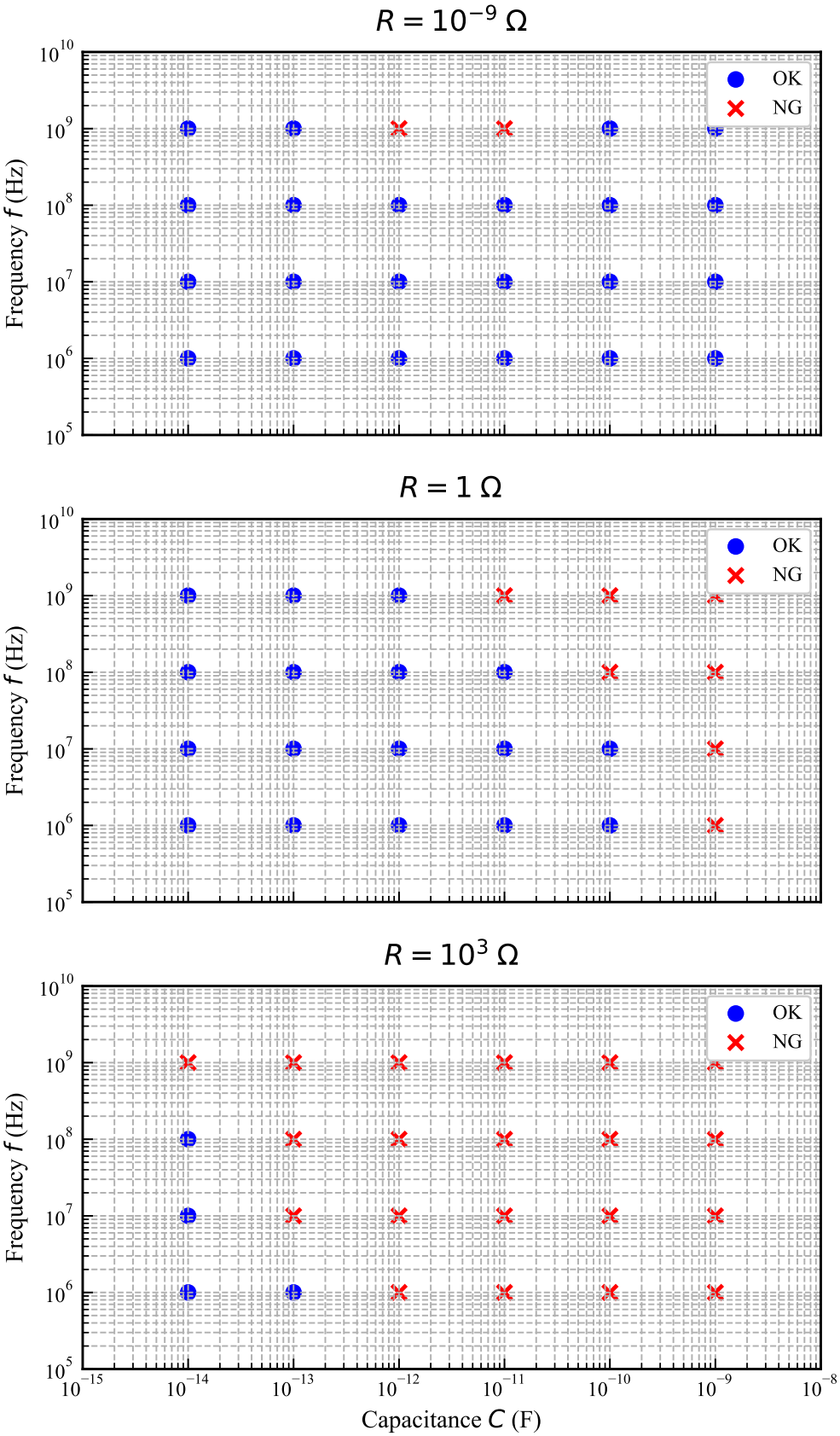


Figure C.20. Operational feasibility (shmoo plot) of the conventional 2N2P circuit across load capacitance C and frequency f, with "NG" indicating non-functional conditions due to resonance failure or excessive damping.

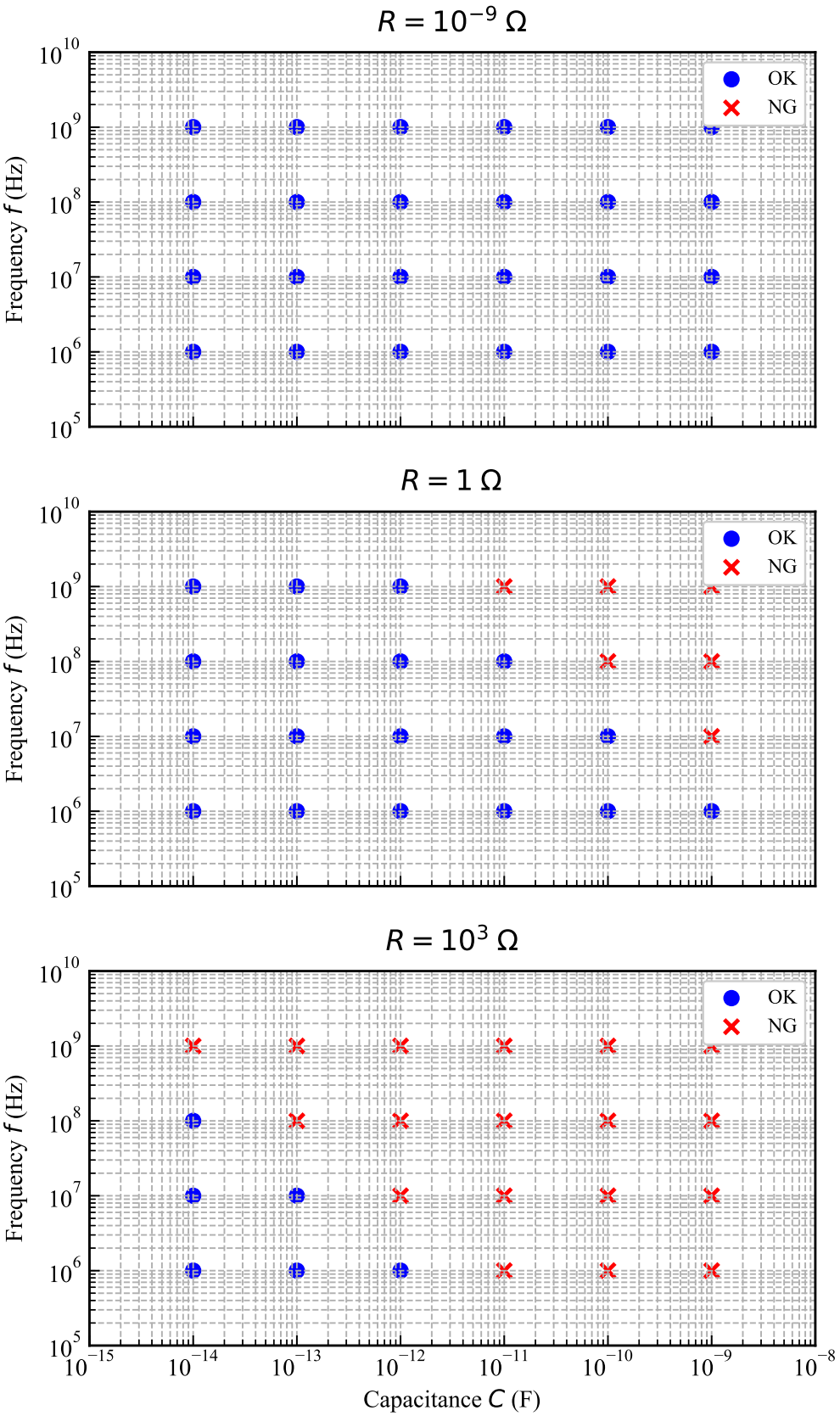


Figure C.21. Operational shmoo plot of the proposed 2N2P-1 circuit, showing functional regions (pass) and non-operational regions (NG) over load capacitance C and frequency f.

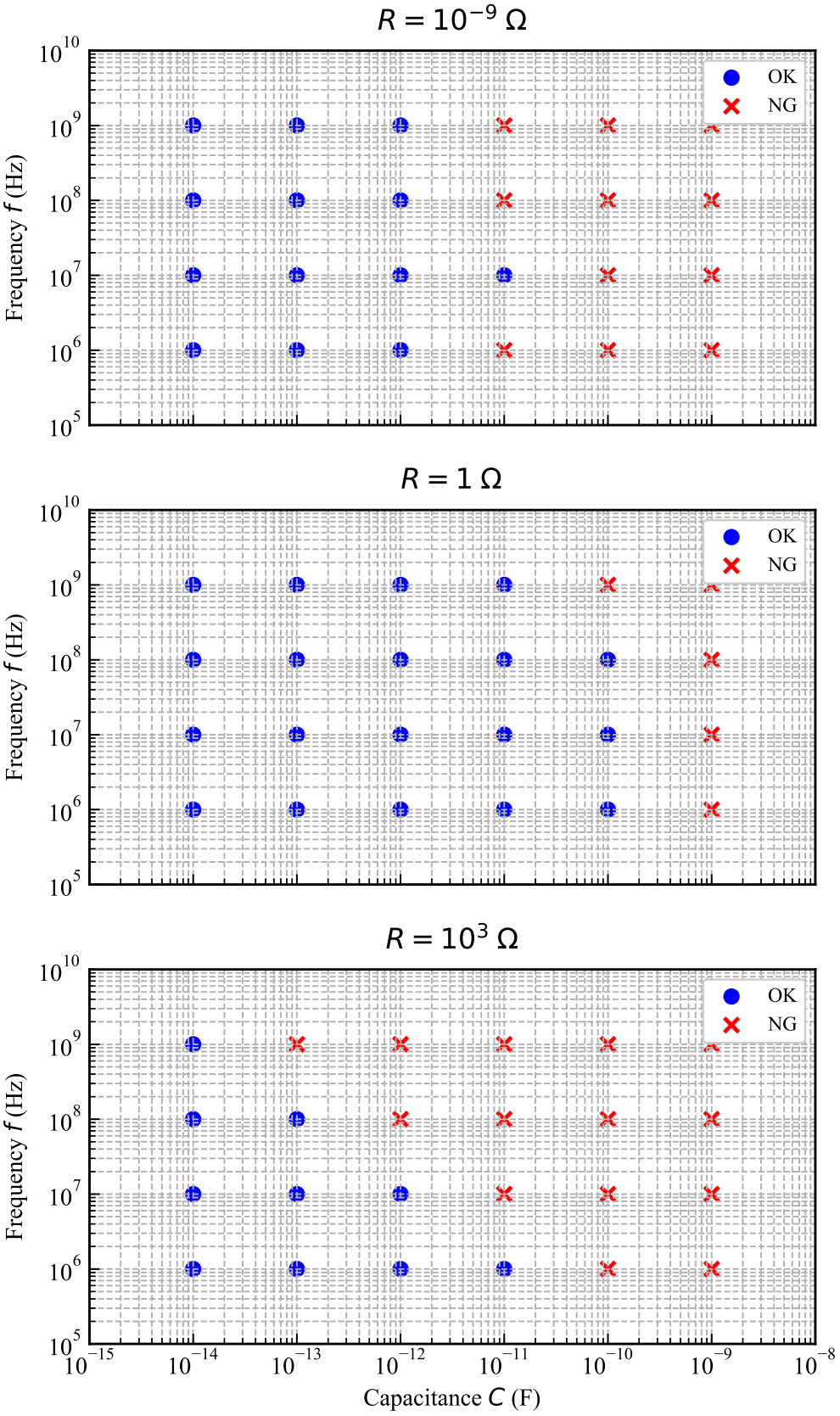


Figure C.22. Shmoo plot of operational validity for the proposed 2N2P-2 circuit with optimized C_p, marking non-functional (NG) regions at high C and f.

References

[1] International Energy Agency (IEA), "World Energy Outlook 2023," 2023. [Online]. Available: https://www.iea.org/reports/world-energy-outlook-2023

[2] International Energy Agency (IEA), "Electricity 2024," 2024. [Online]. Available: https://www.iea.org/reports/electricity-2024

[3] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, pp. 984–986, 2020.

[4] A. Shehabi et al., "United States Data Center Energy Usage Report," Lawrence Berkeley National Laboratory, LBNL-1005775, 2016.

[5] E. Strubell, A. Ganesh, and A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP," 2019. [Online]. Available: https://arxiv.org/abs/1906.02243

[6] D. Patterson et al., "Carbon Emissions and Large Neural Network Training," 2022. [Online]. Available: https://arxiv.org/abs/2104.10350

[7] S. Borkar and A. A. Chien, "The Future of Microprocessors," *Communications of the ACM*, vol. 54, no. 5, pp. 67–77, 2011.

[8] K. Morita, *Theory of Reversible Computing*. Germany: Springer, 2017.

[9] R. Landauer, "Irreversibility and Heat Generation in the Computing Process," *IBM Journal of Research and Development*, vol. 5, no. 3, pp. 183–191, 1961.

[10] C. H. Bennett, "Logical Reversibility of Computation," *IBM Journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.

[11] C. H. Bennett, "Notes on Landauer's principle, reversible computation, and Maxwell's Demon," *Studies in History and Philosophy of Modern Physics*, vol. 34, no. 3, pp. 501–510, 2003.

[12] C. H. Bennett, "The thermodynamics of computation—a review," *International Journal of Theoretical Physics*, vol. 21, no. 12, pp. 905–940, 1982.

[13] M. P. Frank, "Physical Limits of Computing," *IEEE Computing in Science and Engineering magazine*, pp. 16–26, 2002.

[14] N. Anuar, Y. Takahashi, and T. Sekine, "Low-power Adiabatic Logic Circuit: Simulation and Energy Dissipation Comparison" in *IEICE Tech. Rep*, vol. 108, pp. 125–130, 2008.

[15] EE Times, "Vaire Demos Energy Recovery With Reversible Computing Test Chip," 2025. [Online]. Available: https://www.eetimes.com/vaire-demos-energy-recovery-with-reversible-computing-test-chip/

[16] Vaire Computing, “Vaire Computing Official Website,” 2025. [Online]. Available: https://vaire.co/

[17] IEEE Spectrum, “Reversible Computing Breakthrough,” 2025. [Online]. Available: https://spectrum.ieee.org/reversible-computing

[18] T. Rault, A. Bouabdallah, and Y. Challal, “Energy efficiency in wireless sensor networks: A survey,” *Computer Communications*, vol. 39, pp. 1–17, 2014.

[19] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, “A review of wearable sensors and systems with application in rehabilitation,” *Journal of NeuroEngineering and Rehabilitation*, vol. 9, p. 21, 2012.

[20] A. Yakovlev, S. Kim, and A. Poon, “Implantable biomedical devices: Wireless powering and communication,” *IEEE Communications Magazine*, vol. 52, no. 4, pp. 152–159, 2014.

[21] S. Sudevalayam and P. Kulkarni, “Energy harvesting sensor nodes: Survey and implications,” *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 443–461, 2011.

[22] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, “Wireless networks with RF energy harvesting: A contemporary survey,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 757–789, 2015.

[23] S. P. Beeby, M. J. Tudor, and N. M. White, “Energy harvesting vibration sources for microsystems applications,” *Measurement Science and Technology*, vol. 17, no. 12, pp. R175–R195, 2006.

[24] F. K. Shaikh and S. Zeadally, “Energy harvesting in wireless sensor networks: A comprehensive review,” *Renewable and Sustainable Energy Reviews*, vol. 55, pp. 1041–1054, 2016.

[25] S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, and P. Tuyls, “Extended abstract: The butterfly PUF protecting IP on every FPGA,” in *Proc. IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 67–70, 2020.

[26] J. Delvaux, D. Gu, R. Maes, and I. Verbauwhede, “Secure lightweight entity authentication with strong PUFs: Mission impossible?” in *Proc. Cryptographic Hardware and Embedded Systems*, pp. 451–475, 2015.

[27] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, “Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits,” *Proceedings of the IEEE*, vol. 91, no. 2, pp. 305–327, 2003.

[28] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, “Dark Silicon and the End of Multicore Scaling,” in *Proc. 38th International Symposium on Computer Architecture*, pp. 365–376, 2011.

[29] A. Wang, B. H. Calhoun, and A. P. Chandrakasan, *Sub-threshold Design for Ultra Low-Power Systems*. Springer, 2006.

[30] K. Itoh, “A Historical Review of Low-Power, Low-Voltage Digital MOS Circuits Development,” *IEEE Solid-State Circuits Magazine*, vol. 5, pp. 27–39, 2013.

[31] W. C. Athas, L. J. Svensson, J. G. Kollar, N. T. Tzartzanis, and E. Y. -C. Chou, “Low-Power Digital Systems Based on Adiabatic-Switching Principles,” *IEEE Transactions on VLSI Systems*, vol. 2, no. 4, pp. 398–407, 1994.

[32] S. G. Younis and T. F. Knight, “Practical Implementation of Charge Recovering Asymptotically Zero Power CMOS,” in *Proc. Symposium on Integrated Systems*, pp. 1–10, 1994.

[33] P. Teichmann, *Adiabatic Logic: Future Trend and System Level Perspective*. Springer, 2012.

[34] W. C. Athas et al., “Low-Power Digital Systems Based on Adiabatic-Switching Principles,” *IEEE Transactions on VLSI Systems*, vol. 2, no. 4, pp. 398–407, 1994.

[35] Y. Moon and D. K. Jeong, “An efficient charge recovery logic circuit,” *IEEE Journal of Solid-State Circuits*, vol. 31, no. 4, pp. 514–522, 1996.

[36] A. Kramer, J. S. Denker, B. Flower, and J. Moroney, “2nd order adiabatic computation with 2N-2P and 2N-2N2P logic circuits,” in *Proceedings of the International Symposium on Low Power Design (ISLPED)*, pp. 191–196, 1995.

[37] A. Vetuli, L. Pasini, L. Reyneri, and M. Ruo Roch, “Positive Feedback in Adiabatic Logic,” *Electronics Letters*, vol. 33, no. 20, pp. 1685–1686, 1997.

[38] B.-D. Yang and Y.-K. Choi, “NMOS energy recovery logic,” *Electronics Letters*, vol. 36, no. 16, pp. 1363–1364, Aug. 2000.

[39] M. Mahmoud and M. Bayoumi, “Clocked Adiabatic Logic,” in *Proc. Midwest Symposium on Circuits and Systems*, pp. 1–4, 1998.

[40] S. Kim and M. C. Papaefthymiou, “True single-phase adiabatic circuitry,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 1, pp. 52–63, Feb. 2001.

[41] S. Kim and M. C. Papaefthymiou, “Single-phase source-coupled adiabatic logic,” in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 97–99, 1999.

[42] J. Lim, D. G. Kim, and S. I. Chae, “A 16-bit carry-lookahead adder using reversible energy recovery logic for ultra-low-energy systems,” *IEEE Journal of Solid-State Circuits*, vol. 35, no. 6, pp. 807–815, 2000.

[43] A. G. Dickinson and J. S. Denker, “Adiabatic Dynamic Logic,” *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 311–315, 1995.

[44] A. Zulehner, M. P. Frank, and R. Wille, “Design Automation for Adiabatic Circuits,” 2019. [Online]. Available: https://arxiv.org/abs/1809.02421

[45] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. Wiley, 1988.

[46] J. Forrest and R. Lougee-Heimer, “CBC user guide,” *Emerging theory, methods, and applications*, INFORMS, pp. 257–277, 2005.

[47] M. Hansen, H. Yalcin, and J. P. Hayes, “Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering,” *IEEE Design and Test*, vol. 16, pp. 72–80, 1999.

[48] R. Brayton and A. Mishchenko, “ABC: An Academic Industrial-Strength Verification Tool,” in *Proc. 22nd International Conference, Computer Aided Verification*, pp. 24–40, 2010.

[49] H. D. Mittelmann, “Benchmarks for Optimization Software,” 2024. [Online]. Available: https://plato.asu.edu/bench.html

[50] MIPLIB Team, “MIPLIB 2017 Evaluations,” 2017–2025. [Online]. Available: https://miplib.zib.de

[51] L. A. Wolsey, *Integer Programming*. Wiley, 1998.

[52] F. Brglez, D. Bryan, and K. Kozminski, “Combinational profiles of sequential benchmark circuits,” in *Proc. ISCAS*, pp. 1929–1934, 1989.

[53] R. Diestel, *Graph Theory*. Germany: Springer, 2017.

[54] L. Lovász, “On the Shannon capacity of a graph,” *IEEE Transactions on Information Theory*, vol. 25, no. 1, pp. 1–7, 1979.

[55] C. Mannino and E. Stefanutti, “An Augmentation Algorithm for the Maximum Weighted Stable Set Problem,” *Computational Optimization and Applications*, pp. 367–381, 1999.

[56] Y. K. Kwong and I. Ahmad, “Benchmarking and comparison of the task graph scheduling algorithms,” *Journal of Parallel and Distributed Computing*, vol. 59, pp. 381–422, 1999.

[57] N.Takeuchi, D.Ozawa, Y.Yamanashi, and N.Yoshikawa, “An adiabatic quantum flux parametron as an ultra-low-power logic device,” *Superconductor Science and Technology*, vol.26, no.3, p. 035010, 2013.

[58] N.Takeuchi, T.Yamae, C.L.Ayala, H.Suzuki, and N.Yoshikawa, “Adiabatic quantum-flux-parametron: A tutorial review,” *IEICE Transactions on Electronics*, vol.E105-C, no.6, pp. 251–263, 2022.

[59] A. Rauchenecker, T. Ostermann, and R. Wille, “Exploiting Reversible Logic Design for Implementing Adiabatic Circuits” in *Proc. 24th International Conference Mixed Design of Integrated Circuits and Systems*, pp. 264–270, 2017.

[60] C. Drewes, “Globally Irreversible Locally Reversible” in *Tech. Rep*, University of California, San Diego, 2022.

[61] “Yosys Open SYnthesis Suite,” Available at: https://yosyshq.net/yosys/.

[62] Predictive Technology Model, “Predictive Technology Model,” [Online]. Available: http://ptm.asu.edu/

[63] A. Gleixner et al., “MIPLIB 2017: The Mixed Integer Programming Library 2017 Update,” *Mathematical Programming Computation*, vol. 13, no. 2, pp. 225–253, 2021.

[64] C. Monteiro, Y. Takahashi, and T. Sekine, “Charge-sharing symmetric adiabatic logic in countermeasure against power analysis attacks at cell level,” *Microelectronics Journal*, vol. 44, no. 7, pp. 614–624, 2013.

[65] H. S. Raghav, V. A. Bartlett, and I. Kale, “Investigation of stepwise charging circuits for power-clock generation in Adiabatic Logic,” in *Proc. 12th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, pp. 1–4, 2016.

[66] M. Arsalan and M. Shams, “Charge-Recovery Power Clock Generators for Adiabatic Logic Circuits,” in *Proc. 18th International Conference on VLSI Design*, pp. 669–674, 2005.

List of Achievements

Journal Publications

- **March 2024**
  Yuya Ushioda, Mineo Kaneko
  ILP Based Approaches for Optimizing Early Decompute in Two Level Adiabatic Logic Circuits
  IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences,
  Vol. E107-A, No. 3, pp. 600–609

Conference Presentations

International Conferences

- **October 2024**
  Yuya Ushioda, Mineo Kaneko
  Optimization of Pipeline Schedule for Hardware Efficient Two-Level Adiabatic Logic Circuits
  The 25th Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI 2024), 6 pages

- **October 2022**
  Yuya Ushioda, Mineo Kaneko
  Hardware Minimization of Two-Level Adiabatic Logic Based on Weighted Maximum Stable Set Problem
  Proceedings of 2022 IEEE 40th International Conference on Computer Design (ICCD 2022), pp. 394–397

Domestic Conferences

- **August 2023**
  Yuya Ushioda, Mineo Kaneko
  Optimization of Pipelining and Decompute-Insertion for Hardware Efficient Two-Level Adiabatic Logic
  IPSJ Design Automation Symposium 2023 (DAS2023)

- **August 2022**
  Yuya Ushioda, Mineo Kaneko
  An Approach to Hardware Reduction in Adiabatic Logic Circuit based on Maximum Stable Set Problem
  IPSJ Design Automation Symposium 2022 (DAS2022), pp. 234–241
  Best Paper Award

Awards

- **August 2022**
  Best Paper Award
  IPSJ Design Automation Symposium 2022 (DAS2022)
  For: "An Approach to Hardware Reduction in Adiabatic Logic Circuit based on Maximum Stable Set Problem"

- **August 2022**
  Best Presentation Award
  IPSJ Design Automation Symposium 2022 (DAS2022)
  For: "An Approach to Hardware Reduction in Adiabatic Logic Circuit based on Maximum Stable Set Problem"