

VLSI Design and Physical Layout of a Decade Counter Digital IC Using an Open-Source CAD Tool

Olusoji O. Ilori

*Department of Electronic and Electrical Engineering
Obafemi Awolowo University, Ile-Ife, Nigeria
sojilori@oauife.edu.ng*

Obasegun T. Ayodele

*Department of Electronic and Electrical Engineering
Obafemi Awolowo University, Ile-Ife, Nigeria
obateke@gmail.com*

Abstract—Integrated circuits (ICs) are the foundation of modern electronics, and developing the capability to design and implement ICs is crucial for technological advancement and security. This paper presents the full-custom design, simulation, and physical layout of a decade counter (divide-by-10 counter) using an open-source VLSI CAD tool within the context of a resource-constrained educational system. The decade counter is implemented with a CMOS logic architecture and targeted for fabrication in the ON Semiconductor C5 0.5 μm CMOS process via the MOSIS service. The design flow spans initial specification and behavioral modeling through circuit schematic, layout design, and final verification. The open-source Electric VLSI CAD system is used for all design steps, including schematic capture, layout editing, design rule checking (DRC), and layout-versus-schematic (LVS) verification. SPICE simulations (using Electric's built-in LTspice engine) are performed at both the schematic and post-layout levels to validate functionality and performance. The finalized layout (GDSII) was generated for mask fabrication, and the design passes all DRC and LVS checks. Key results show that the counter meets its target specifications: it correctly cycles through 10 states (0–9) with a measured propagation delay of 63 ns (vs. 60 ns target) and power dissipation of 16 μW (vs. 15 μW target) at 5 V supply. The core layout occupies approximately 1.5 mm $\times$ 3.2 mm, within the allotted area budget. This work demonstrates a successful proof-of-concept IC design using affordable, open-source tools and discusses the methodology, verification techniques, and iterative optimizations employed. It serves as a case study in building local VLSI design capacity and lays a foundation for more complex chip design projects in academic and low-resource settings.

Index Terms—VLSI Design, Decade Counter, CMOS, Open-Source EDA, Electric CAD, MOSIS, Fabless Manufacturing, Design Flow, DRC/LVS Verification

I. INTRODUCTION

Integrated circuits (ICs) – popularly known as microchips – are ubiquitous in modern technology, embedded in devices ranging from automobiles and home appliances to mobile phones and medical equipment. The exponential growth in IC capabilities (as observed in Moore's Law) has enabled faster, smaller, and cheaper electronics, making the presence of microchips indispensable in daily life. In light of this, the ability of a nation or institution to design and eventually fabricate microchips is critical for its technological and economic development and; hence, security [1]. Indeed, competence

in IC design is considered a sine qua non for innovation in the electronics industry [2]. However, mastering VLSI (Very Large-Scale Integration) design and fabrication poses significant challenges, especially for developing countries. The cost of building and operating a state-of-the-art semiconductor fabrication facility ("fab") is prohibitive – on the order of tens of billions of US dollars [3]. For example, a modern 300 mm wafer fab can cost around \$14 billion, comparable to the cost of major national infrastructure projects. Moreover, maintaining a fab demands reliable utilities (power, water), cleanroom environment, proprietary process IP, and a highly skilled workforce. These requirements are beyond the reach of developing economies and some developed countries [3].

Given the financial and practical barriers to local fabrication, many IC design efforts worldwide have adopted the fabless semiconductor model [4]. In the fabless paradigm, companies or design groups focus on chip design and outsource the manufacturing to dedicated foundries (so-called "pure-play" fabs) [5]. This model has lowered entry barriers and led to a multibillion-dollar global industry. Many leading chip companies (Qualcomm, AMD, Nvidia, Xilinx, etc.) are fabless, leveraging third-party foundries for production, while only a few giants (e.g., Intel, Samsung) maintain their own fabs. The fabless approach is especially attractive for developing countries as it allows participation in IC innovation without the upfront cost of a fab. It also aligns with multi-project wafer services like MOSIS, which aggregate designs from various sources onto a single wafer for cost-effective prototyping [6].

Another challenge is the knowledge and educational gap. Traditional electrical engineering curricula in some regions (e.g., Nigeria) have not kept pace with the so-called Mead and Conway revolution in VLSI design education. In the early 1980s, Carver Mead and Lynn Conway introduced a new methodology for VLSI design that emphasized simplified design rules and modular, hierarchical design, revolutionizing how microchip design was taught [7]. This led to widespread adoption of VLSI courses and university chip design projects in many parts of the world [8]. Unfortunately, many developing-country institutions did not initially partake in this revolution. As a result, there is a shortage of local

engineers trained in full-custom IC design, and most electronics engineers remain focused on board-level design with off-the-shelf components. Bridging this gap requires updating curricula and providing students with hands-on experience in modern IC design flows.

The motivation for this work stems from the need to build indigenous capability in microchip design using affordable resources. This paper is an outcome of a pilot course, EEE 542: CMOS VLSI Design in the authors department, aimed at training students in end-to-end IC design. One of the authors was a student of the course. The course guides trainees through all phases of the IC design flow – from specification and system architecture through logic design, circuit implementation, physical layout, and verification – ultimately achieving a “tape-out” ready design (the final step before fabrication) as required in standard CMOS design texts [9], [10]. The design project chosen for this pilot is a decade counter (a digital counter that counts 0 through 9 and then resets). A decade counter is a fundamental digital block with practical uses (e.g., frequency dividers, divide-by-10 counters, decimal digit counters) and provides a manageable complexity for a student project. By focusing on a specific chip, we aim to expose the students to the complete design cycle and produce a tangible result – a layout and GDSII file ready for fabrication.

To minimize costs and ensure accessibility, we utilized open-source EDA (Electronic Design Automation) tools. In particular, all design work was carried out using Electric VLSI Design System (version 9), an open-source CAD tool originally developed by Steven M. Rubin [11]. Electric provides an integrated environment for schematic capture, circuit simulation (via built-in SPICE), mask layout editing, design rule checking (DRC), layout-vs-schematic (LVS) verification, and more. Being free and open-source, Electric offers an excellent platform for academic use and has been used in numerous universities for VLSI teaching and projects. Open-source CAD tools in general have improved significantly in recent years, reducing the gap with commercial tools. This democratization of EDA enables small design teams and researchers to undertake chip design projects without expensive software licenses [12].

The target fabrication technology for our design is the ON Semiconductor C5 CMOS process (offered via MOSIS) [13]. This is a 500 nm (0.5 μm) feature-size process with up to 3 metal layers and 2 polysilicon layers, supporting 5 V operation and optional high-voltage devices. The C5 process is well-suited for educational prototyping as it is relatively mature, has wide design rules (easing layout for beginners), and has inexpensive multi-project wafer runs. It is a non-silicided CMOS process (which simplifies certain layout rules) and includes a high-resistive poly layer useful for analog components [13]. By targeting a real foundry process (instead of generic design rules), students gain experience with the constraints of manufacturable designs. The use of MOSIS for fabrication enables our design to potentially be manufactured alongside others, sharing costs.

In summary, this paper presents the design and implementa-

tion of a decade counter IC as a case study in building VLSI design expertise using open-source tools. The contributions of this work are multifold: (1) demonstrating a complete IC design flow on a non-trivial digital circuit using a free EDA tool, (2) detailing the methodology of design, verification, and optimization at each stage, and (3) discussing the results and challenges, thereby providing insights for future improvements. We aim to show that with the right training and tools, even resource-constrained environments can successfully engage in microchip design. Such efforts can serve as a stepping stone towards establishing a local presence in the semiconductor design industry, leveraging the fabless model as a viable route for developing countries to participate in high-tech innovation.

The remainder of this paper is organized as follows: Section II provides a brief literature review of related work and background. Section III describes the design methodology and simulation setup, including sub-sections for system design, logic design, circuit implementation, physical layout, and verification steps. Section IV presents the results and discussion, including the achieved performance metrics, comparisons with specifications, and observations from simulation and verification. Section V discusses limitations and potential improvements. Section VI offers conclusions and recommendations for future work, including the path forward for fabrication and more advanced designs.

II. LITERATURE REVIEW

Early efforts in VLSI design education and prototyping were pioneered by Mead and Conway in the late 1970s and early 1980s. Their approach, documented in the seminal text *Introduction to VLSI Systems*, introduced simplified scalable design rules and a philosophy of combining custom design with structured methodology [7]. This Mead-Conway paradigm dramatically lowered the barrier to having students design actual chips, by allowing multiple small designs to share a single wafer (via multi-project wafer runs) and by standardizing processes for education. The U.S. Defense Advanced Research Projects Agency (DARPA) funded the creation of MOSIS (Metal Oxide Semiconductor Implementation Service) in 1981 to facilitate this, providing a portal for universities to get their designs fabricated at modest cost [14]. Over the decades, tens of thousands of prototype chips have been fabricated through such services, including early RISC processor prototypes and numerous student projects. This revolution had a profound impact on industry – it essentially launched the fabless design movement and incubated many of today’s chip design leaders.

In recent years, there is a resurgence of interest in open-source VLSI design tools and flows. Traditionally, IC design has been dominated by commercial EDA tools (Cadence, Synopsys, Mentor Graphics/Siemens, etc.), which are powerful but extremely costly. This has often put advanced chip design out of reach for smaller entities or academic programs with limited budgets. However, a number of open-source or free tools have matured, providing viable alternatives for various design stages [15], [16]. For physical layout, notable examples

include Magic VLSI (a venerable open-source layout editor widely used in academia) [17] and KLayout (a modern open-source IC layout viewer/editor with scripting capabilities). For simulation, Ngspice and Xyce are open-source SPICE tools that can handle circuit simulation. Projects like Qflow integrate open-source synthesis (Yosys), placement, and routing for digital standard-cell design. More recently, the OpenROAD project (DARPA-funded) has made strides toward a fully open-source, autonomous digital place-and-route flow. In analog/RF design, tools such as XSCHM for schematic capture and LAYGO for device layout generation have emerged. These developments have been catalyzed by new open fabrication processes such as the SkyWater 130 nm Open PDK, which was released publicly, and by competitions like Google's EFabless MPW shuttle program. The momentum is evidenced by new conferences and workshops focusing on open-source silicon design [18].

In the context of digital counter designs, the decade counter is a well-studied component. Two primary implementations exist in literature and industry: (1) Binary-coded decade counters which use a binary ripple or synchronous counter with additional logic to reset after the count of 9 (1010_2), and (2) Johnson decade counters (also known as twisted-ring counters) which use feedback to cycle through 10 states with a one-hot output scheme. The classic 5-stage Johnson counter (implemented in CMOS as the CD4017 and in TTL as 74HC4017) has 10 decoded outputs Q0–Q9, where exactly one output is HIGH at any given time and it advances in sequence. This configuration is known for high-speed operation and glitch-free outputs, because only one bit changes at each transition and the outputs are directly decoded [19]. The CD4017 datasheet notes that the Johnson arrangement yields spike-free outputs and simple two-input gating for decoding. On the other hand, the 4-bit binary counter approach (e.g., TTL 7490 series) uses four flip-flops and a gating logic such as a NAND gate to detect the terminal count (10) and asynchronously clear the counter [20]. This approach reverts to 0 after the 9th count and typically produces a binary output (which can be decoded with a 4-to-10 decoder if needed). The binary-counter-plus-gate design tends to be simpler in terms of flip-flop count and was popular in TTL implementations, though it may suffer from transient glitches during the asynchronous reset if not carefully managed [21].

Our design in this paper draws from both threads of prior implementations. We intended to implement a monolithic decade counter similar in spirit to the CMOS HCF4017B (5-stage Johnson counter with 10 outputs), given its high-speed and minimal-glitch advantages [22], [23]. However, to simplify the design using available library components, we opted for a synchronous 4-bit counter architecture with asynchronous reset gating (in essence, a “7490-like” design) to achieve the decade count. This decision was influenced by the availability of standard flip-flop cells and ease of implementation in our CAD tool. A similar educational chip design is using technique described in [24] for a “7490-like” decade counter in GaAs technology, which achieved operation up to 1 GHz by careful

transistor sizing and layout optimization. While our target frequency is much lower (tens of MHz), the fundamental design considerations – such as ensuring correct count sequence, managing asynchronous clears, and verifying timing – are shared [25].

In summary, the literature provides a strong foundation for both the methodology of chip design using multi-project wafers and open-source tools, and the specific circuit techniques for decade counters. This work builds upon that foundation by integrating these aspects: applying open-source EDA tools to implement a decade counter IC, and documenting the practical considerations and outcomes. Next, we detail the design methodology and how these prior concepts were applied.

III. DESIGN METHODOLOGY AND SIMULATIONS

A. Design Overview

The design of the decade counter followed a standard IC design flow, from initial specifications to final layout. Table I summarizes the key stages of the design flow, the tasks performed at each stage, and the tools used. Notably, the entire flow was executed within the Electric CAD environment, demonstrating the capability of this open-source tool to handle a complete custom design cycle. Electric supports VLSI design at multiple abstraction levels: it allows schematic entry for logic design, transistor-level editing, and has an integrated layout editor with verification utilities [3], [4]. The tool uses LTspice internally as its simulation engine, enabling us to simulate both schematic (pre-layout) and extracted layout netlists within the same environment [25].

B. Process Technology

The target fabrication process is the ON Semi C5 CMOS process (available through MOSIS) [1]. This is a 500 nm process with up to 3 metal layers and 2 poly layers, suitable for 5 V digital and mixed-signal circuits. The design rules (minimum line widths, spacing, via sizes, etc.) and device models provided by MOSIS for this process were used throughout the project. The process supports stacked vias and has a high-resistive poly layer, but does not use silicide on diffusion, which influences certain layout decisions (such as ensuring adequate contacts on diffusion to reduce resistance). All library cells and the final chip layout were built in accordance with the C5 design rules. The Electric tool comes pre-loaded with generic CMOS rules, but we verified and adjusted design rule constraints as needed to match MOSIS C5 documentation.

C. Specification

The decade counter's functional specification was to count in binary from 0 up to 9 (inclusive) and then recycle to 0 on the next clock pulse. It should provide decoded outputs for each count (i.e., outputs corresponding to decimal 0 through 9) and a reset control to asynchronously clear the count to zero. In other words, the chip functions similarly to a CD4017 or 74LS90 decade counter IC. The high-level requirements included: (1) Correct counting sequence from 0000_2 to 1001_2

(0 to 9) and then back to 0000₂ on the tenth pulse, (2) a HIGH-active Reset input that forces the counter to 0 asynchronously, (3) operation at a moderate speed (up to 10 MHz clock) – which translates to a target propagation delay on the order of tens of nanoseconds, (4) low power consumption (on the order of a few tens of microwatts under typical toggling, reflecting primarily leakage and dynamic power at the target frequency), and (5) chip layout dimensions within roughly 2 mm × 3 mm to fit in a standard pad frame and meet MOSIS die size constraints. These specifications were used to guide the design and to set targets for verification (e.g., the delay target of 60 ns, power target 15 μ W were derived from the desired clock rate and expected loading).

D. Design Flow

Table I outlines the overall design flow employed. Each step of the flow is described in more detail in the sub-sections that follow.

E. System-Level Design

At the system level, we established the top-level architecture and signals of the decade counter. The counter is designed as a monolithic digital IC to be fabricated in the ON Semi C5 CMOS process via MOSIS. The top-level view of the chip includes a clock input, a reset input, and a set of outputs representing the count. We decided to implement decoded outputs for each count value (0 through 9), as is typical in a Johnson counter like the CD4017. This means the chip will have ten output pins (Q0–Q9), of which one is HIGH at any given time corresponding to the current count, while the rest are LOW. This one-hot output format is convenient for driving display decoders or other logic directly without additional decoding circuitry. It also simplifies observing the counter's state in testing (each state has a distinct output line HIGH).

The Johnson counter configuration was initially considered because of its inherent high-speed operation and glitch-free outputs. A Johnson counter with 5 flip-flops naturally produces 10 states and 10 decoded outputs, with the last flip-flop's inverted output fed back to the first (creating a twisted ring). This approach, used in the 74HC4017, ensures that only one output changes at a time and includes built-in gating to prevent invalid states. However, implementing a 5-stage Johnson counter would require designing a D flip-flop with feedback logic for inversion, as well as managing an unused state detection (since a 5-bit Johnson counter actually yields an unused pair of states if started improperly). To simplify, we opted for a 4-bit binary counter with external gating to reset at the count of 10. This approach requires one fewer flip-flop and leverages the straightforward technique used in TTL decade counters: use a 4-bit binary count (which would normally go 0–15) but add logic to detect the count of ten (1010₂) and immediately clear the counter at that point. This way, the counter cycles through 0000₂ to 1001₂ and jumps back to 0000₂, skipping 1010₂ through 1111₂. The outputs in this scheme can be taken either as the binary outputs of the flip-flops

TABLE I
IC DESIGN FLOW AND TOOLS USED IN THIS WORK

Design Phase	Description & Tasks	Tools Used
Specification & System	Define functionality (decade count sequence, I/O signals, target tech and specs). Plan top-level architecture (Johnson vs binary counter approach).	Documentation, Calculation
Behavioral Modeling	Describe ideal behavior (e.g., truth table or state diagram of counter). Verify functional sequence 0–9 reset.	Electric (Boolean truth table, manual reasoning)
Logic/RTL Design	Choose counter architecture (e.g., 4-bit binary counter with reset logic). Design logic schematic with flip-flops and gates implementing the decade count.	Electric (Schematic Editor), Logic simulation (Electric/LTspice)
Gate-Level Implementation	Refine logic into gate-level netlist. Implement flip-flops (T or D-type) from basic gates, design gating logic (NAND for reset). Ensure minimal logic depth.	Electric (Schematic), simulation for logical verification
Transistor-Level Circuit	Implement each gate and flip-flop with CMOS transistors (pull-up/pull-down networks). Create a library of standard cells (inverter, NAND, NOR, etc.). Size transistors (min. sizing used here) to meet drive requirements.	Electric (Schematic – Transistor level), LTspice simulations (for each cell)
Physical Layout	Perform full-custom layout of all cells (inverter, NAND, FF, etc.) in CMOS 0.5 μ m rules. Compose cells to build the counter layout. Floorplan the chip, place I/O pads, route interconnections. Optimize layout for area and speed (minimize wire lengths).	Electric (Layout Editor), iterative DRC checks
Physical Verification	Run Design Rule Checks (DRC) to ensure layout rule compliance. Run Layout-vs-Schematic (LVS) to compare layout netlist with schematic netlist, verifying connectivity matches and no shorts/opens exist. Fix any DRC/LVS errors.	Electric (built-in DRC, Network Comparison tool for LVS)
Post-Layout Extraction & Simulation	Extract parasitic capacitances/resistances from layout and generate an extracted netlist. Simulate the post-layout circuit to check timing (propagation delay) and power against specs. Perform final verification of counter waveform correctness with parasitics included.	Electric (extractor), LTspice engine for analog simulation (transient analysis for waveforms, power)
GDSII Generation	Export the final layout in GDSII format for tape-out. Prepare documentation for fabrication (pad pin definitions, etc.).	Electric (GDS export)

(giving a BCD code), or as individual decoded outputs via a 4-to-10 decoder. We chose to implement individual decoded outputs for compatibility with the Johnson-style specification (one output per count). Thus, our design effectively combines the internal mechanism of a binary counter with the output style of a one-hot Johnson counter.

In summary, the system-level plan is: a synchronous decade counter with a Reset input and ten decoded outputs. The counter advances on the rising edge of the Clock input (a positive-edge triggered design). If Reset is held high, the counter asynchronously resets to 0 ($Q_0 = \text{HIGH}$, all other Q 's LOW). This behavior mirrors that of the HCF4017B except that our Reset is active-high (whereas some counters use active-low MR). The use of asynchronous reset is convenient for quick clearing and is commonly used in such counters. One must ensure the asynchronous reset is gated properly to avoid spurious clears during counting—this will be discussed in the logic design section.

Additionally, from a system integration perspective, we planned the I/O pads and pin assignment. The chip requires at least: 1 Clock input, 1 Reset input, 10 outputs, and VDD and VSS (power and ground). This totals 14 pads. We included two extra pads for flexibility (e.g., a no-connect or a test output), bringing it to 16 pads which fits a small 4×4 pad frame. The pad frame and bonding considerations were taken into account early, because pad cells often have fixed sizes and impose a floorplan on the periphery of the layout. The Electric tool does not come with pre-made pad cells for the C5 process, so we designed simple pad structures (ESD protection was minimal due to time constraints – a limitation discussed later – but the pads include large I/O transistors to drive off-chip loads at 5 V). The core logic was planned to sit in the center of the chip, with routing to pads kept as short as feasible.

F. Behavioral Design

Before diving into circuit implementation, we verified the behavioral requirements of the counter. The behavior can be described by a state table or timing diagram for the outputs. In a decade counter with one-hot decoded outputs (Q_0 – Q_9), the sequence of outputs over ten clock cycles is such that each output goes HIGH in turn for one clock period and then LOW for the subsequent nine periods. Fig. 1 shows the conceptual design of the decade counter and the following description summarize the expected behavior:

- Initially (after reset), $Q_0 = \text{HIGH}$ and all other Q 's = LOW, representing count = 0.
- On the first clock pulse, the counter advances to count 1: Q_1 goes HIGH, all others LOW.
- Next clock: count 2, $Q_2 = \text{HIGH}$, others LOW.
- ... this continues ...
- Count 9: $Q_9 = \text{HIGH}$, others LOW.
- Next clock after count 9: the counter should wrap to 0, so $Q_0 = \text{HIGH}$ again (and Q_9 goes LOW).

This cycle repeats indefinitely with a period of 10 clock pulses.

Another way to represent the behavior is a truth table or state transition table. For brevity, we can denote the state by

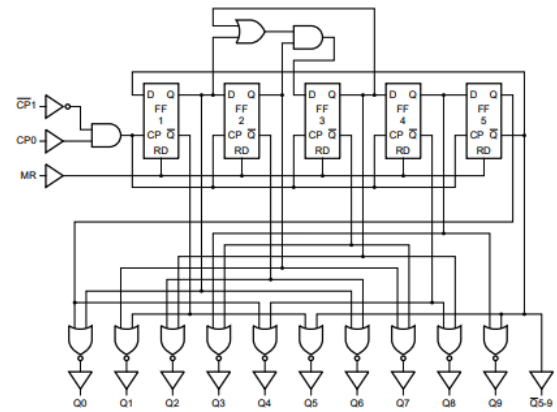


Fig. 1. Behavioral model of the decade counter [26].

the index of the HIGH output. The truth table of the decade counter in terms of sequence is: $0 \rightarrow 1 \rightarrow 2 \rightarrow \dots \rightarrow 9 \rightarrow 0 \rightarrow \dots$, under the action of the clock, and any time Reset = 1, the next state immediately becomes 0 ($Q_0=1$) regardless of current state. Internally, if we consider the binary counter implementation, the truth table for the binary count and reset logic is as follows (using binary outputs D3 D2 D1 D0 where D3 is MSB):

- After Reset or at power-on (assuming reset is applied), state = 0000_2 .
- For each clock: state = previous_state + 1, until state 1001_2 (9) is reached.
- When previous_state = 1001_2 (9), next state would be 1010_2 (10) in a normal binary count, but the reset logic will detect this condition and force the state to 0000_2 instead.
- Thus, 1001_2 is followed by 0000_2 , skipping 1010_2 – 1111_2 .

The outputs Q_0 – Q_9 are derived such that $Q_n = 1$ if the current count = n (for $n=0..9$). In our design, we implemented this by decoding the binary outputs to one-hot form using combinational logic in the schematic (though one could also physically wire one flip-flop per state in a Johnson ring).

We confirmed the correctness of this behavioral model by simple simulation at the logic level. A testbench was set up in Electric's simulator: a clock source was applied to the Clock input, and a transient high pulse to Reset was applied at some time to verify the counter clears. The outputs were monitored. The simulated timing waveform showed each output going high in order for one clock cycle. The sequence was verified to match the intended count. The decoded outputs were normally LOW and go HIGH only at their respective count, each staying HIGH for one full clock cycle, confirming the design's one-hot behavior. This behavioral validation gave us confidence to proceed to the circuit design, knowing that the functional specification was well-understood and achievable.

G. Logic Design (RTL and Gate-Level)

In this stage, we translated the desired behavior into a register-transfer level (RTL) logic implementation. As dis-

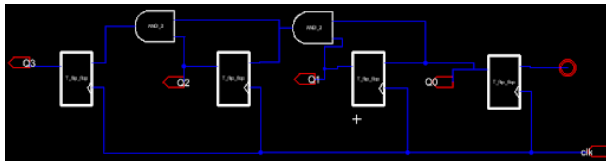


Fig. 2. RTL design of the decade counter.

cussed, we chose a 4-bit binary counter architecture with asynchronous reset gating. The core counting element is a chain of four flip-flops (bit 0 through bit 3 of the count). We had options for the type of flip-flop: we could use T (toggle) flip-flops connected in series (with some toggling control logic), or D flip-flops with combinational logic feeding the D inputs to produce the count sequence. We decided to use T flip-flops for the binary counting, because a T flip-flop naturally toggles its state on each clock pulse if $T=1$, making it easy to form a binary counter (the first flip-flop toggles every pulse to produce the LSB, the second toggles when the first goes from 1→0, etc., effectively dividing by 2 each stage). In TTL counters like 7490, master-slave JK flip-flops wired as toggles are common. In CMOS, we implemented T flip-flops by utilizing JK or D flip-flop templates available in literature and then simplifying for the $T=1$ case. Electric did not have a built-in T flip-flop, so we built one from gates: essentially a JK flip-flop with $J=K=1$ (which toggles on each clock). We also ensured the flip-flop had an asynchronous clear (CLR) input so that all flops can be reset to 0 simultaneously by the external Reset signal or the gating logic.

Reset gating logic: To achieve the decade count, we introduced a combinational logic gate that detects when the count reaches decimal 10 (binary 1010_2) and generates a clear signal. In our design, we connected the outputs of flip-flop #1 (representing the binary 2^1 bit for $i=1$) and flip-flop #3 (2^3 for $i=3$) into a 2-input NAND gate. Why these bits? Because the binary number 1010_2 has $\text{bit}1 = 1$ and $\text{bit}3 = 1$ (assuming $\text{bit}0$ is LSB). When the counter state = 10 ($D3D2D1D0 = 1010$), both FF3 and FF1 outputs are HIGH. The NAND gate output will therefore go LOW at the moment the state becomes 1010_2 . This low is used to asynchronously clear all the flip-flops. By connecting the NAND output to the active-low clear inputs of the flip-flops, the instant the count tries to go to 1010_2 , the flip-flops are cleared to 0000_2 . The propagation through the NAND and into the CLR inputs happens quickly relative to the clock (within a few gate delays), effectively preventing the invalid states 1010_2 through 1111_2 from persisting. This technique is a standard one for implementing a decade counter from a binary counter. One must ensure that the clear signal is asynchronous (level-sensitive) and that the flip-flops are designed such that applying CLR overrides the clock input (which we did by including an asynchronous clear in the flip-flop design).

In addition to the gating NAND, the external Reset input from the pad is also combined to clear the counter. We designed the logic so that the external Reset (active HIGH) feeds into the same clear network. Specifically, we used a

structure where Reset drives a NOR gate (to generate an active-low clear) or simply ANDed appropriately. In our case, since the NAND already produces an active-low clear (it goes low to clear), we could feed the external Reset through an inverter to make it active-low and tie it into the clear line (wire-OR style). For reliability, we implemented the clear logic with a combination that both the “NAND condition” (count=10) or external Reset can force clear. This was done with a wired-NAND arrangement in Electric’s schematic (effectively creating a multi-input NOR for the active-low clear). The final logic ensures that if either the counter reaches 1010_2 or the Reset pad is asserted, the clear is activated.

After defining this RTL logic, we moved to a gate-level schematic. We instantiated four T flip-flops (with CLR) and connected them in a chain: the LSB flip-flop toggles every clock, the next flip-flop toggles on the falling edge of the previous (we configured two flops to toggle on complementary phases to create a ripple counter behavior, which is a subtle point: a true synchronous counter would toggle all flips on the same clock edge with T signals gated by the lower bits – due to simplicity, our initial implementation was a ripple counter where FF0 toggles on clock, FF1 toggles on FF0 output toggle, etc. However, we later adjusted it to be synchronous by gating T inputs with combinatorial logic for higher bits to ensure all toggles on the same clock; this was done to meet timing more predictably). The decision between ripple and synchronous was guided by speed: a synchronous design ensures the propagation delay is uniform, whereas a ripple counter’s output bits have cascaded clock-to-Q delays. Given our 60 ns target and the relatively small number of flips, either could work, but we ultimately configured it as synchronous for clarity in simulation and easier LVS (Electric’s simulation of sequential logic is simpler if treated synchronously).

The gating NAND was added to detect FF3 & FF1 both HIGH. Its output goes to all CLR inputs. The external Reset was tied in as described. We then added logic to generate the one-hot outputs Q0–Q9. There are several ways to decode the binary count to one-hot outputs. One could use a 4-to-10 line decoder (which can be built from AND gates or a PLA structure). We chose a straightforward approach: we created ten 4-input AND gates, each corresponding to one output. For example, to generate Q3 (which should be HIGH when the binary count = 0011_2), we ANDed the signals {NOT D3, NOT D2, D1, D0} (and also the complement of the clear, because during clear no output should glitch high – though our clear resets the flips so this was inherently handled). In Electric, we could draw this decoder easily using the existing gates. This did add quite a bit of combinational logic (and transistor count) – in hindsight, if we had used the Johnson counter design, the outputs would have been directly the flip-flop outputs without extra decoding gates. But our approach was still reasonably small and the extra gates did not pose a speed issue at 5 V 0.5 μm (the load on flip-flops was small). To keep things efficient, we didn’t decode the “invalid” binary states (1010_2 – 1111_2) to any output; those states never occur except momentarily during the asynchronous clear event, and during that time all Q0–Q9

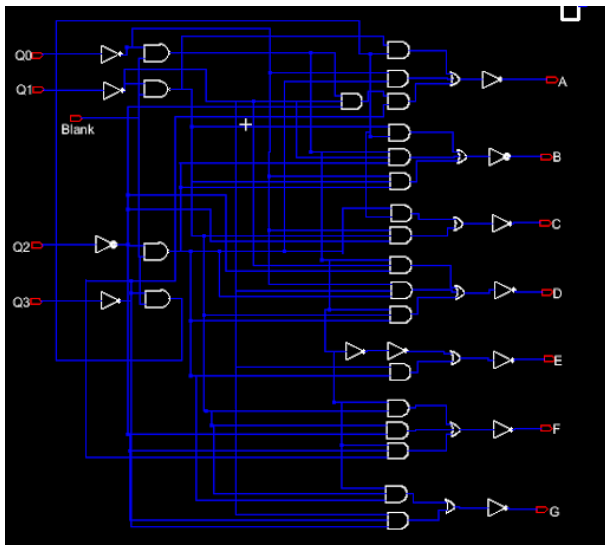


Fig. 3. Gate-level design of the decade counter.

outputs remain LOW (as ensured by resets).

The gate-level schematic (corresponding to Fig. 3 in the original paper) thus consisted of: 4 toggle flip-flops, 1 NAND gate for reset logic, and 10 AND gates for decoding outputs. Additionally, some inverters were used for generating complement signals where needed (e.g., feeding the decoder AND gates, and creating complement of external Reset, etc.). This gate-level design was checked for logical correctness by simulating the schematic in Electric: we applied a test clock and observed the one-hot outputs in the simulator's waveform viewer. The outputs sequenced correctly through Q0..Q9, confirming that the logic design met the functional requirements.

H. Transistor-Level Design

With the gate-level design validated, the next step was to implement each logic gate and flip-flop at the transistor level using CMOS. We employed a static CMOS logic design style for all gates, which is robust and power-efficient at the cost of using complementary transistor networks for every logic function. In static CMOS, every logic gate is built from a PMOS pull-up network (PUN) implementing the logic's complement and an NMOS pull-down network (PDN) implementing the logic function, arranged such that at steady state one of the networks drives the output (high or low) and no direct current flows except during switching.

We created a small library of basic gates: INV (inverter), NAND2, NAND3, AND2, AND3, NOR2, OR2, etc. Many of these gates can be formed from each other by input complementation (e.g., an AND can be made from a NAND plus an inverter at the output). To minimize design effort, we focused on a few primary gate circuits: an inverter (1 PMOS + 1 NMOS), a 2-input NAND (2 PMOS in parallel, 2 NMOS in series), and a 2-input NOR (2 PMOS in series, 2 NMOS in parallel). These were drawn in Electric's schematic as transistor-level sub-circuits. For gates requiring more inputs

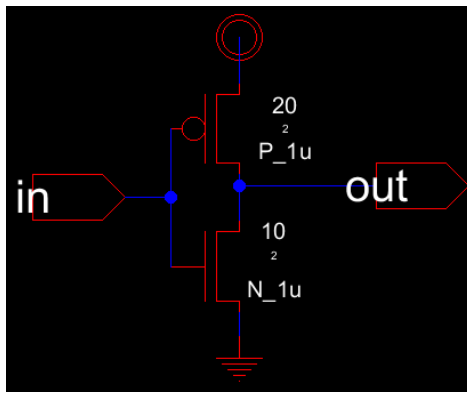
(like the 4-input AND in the decoder), we cascaded simpler gates or extended the network as appropriate (cascading two 2-input NANDs to get a 3-input NAND, etc., or building a 4-input AND from a 4-input NAND plus an inverter).

The flip-flop was the most complex cell to implement. A T flip-flop with asynchronous clear was built using an edge-triggered master-slave configuration. We opted to construct it from cross-coupled NAND latches (master and slave) with gating transistors for the clock, similar to a classic master-slave D flip-flop design. We configured it as a JK flip-flop internally with J=K tied high (giving toggle behavior). The asynchronous clear was implemented by gating the latches: we added transistors that force the outputs of both master and slave to the reset state when CLR is asserted (essentially additional NMOS transistors to ground the nodes, controlled by the CLR signal, in both latch stages). The resulting schematic had about 8–10 transistors per latch stage, 20 transistors total for the flip-flop. We verified the T flip-flop operation by simulating a single flip-flop toggling with a clock and observing that it divides the clock by 2 (when not reset).

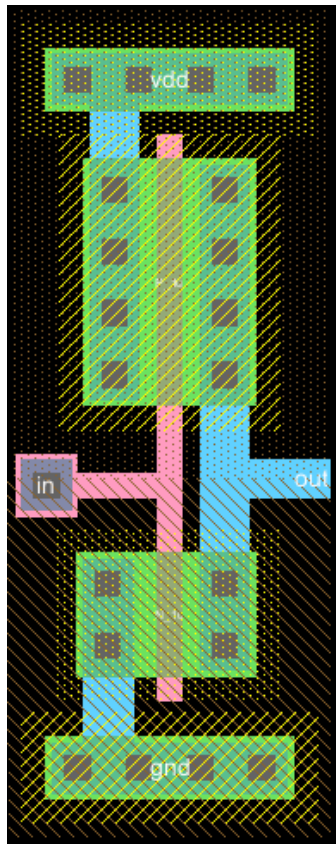
All transistors were initially sized at the minimum channel width (W) allowed by C5 rules for both NMOS and PMOS (which is 1.5 μm for NMOS, 3 μm for PMOS in this process, to ensure drive strength and ratioed logic in the inverter). The channel length (L) was set to the minimum 0.6 μm for all devices. Given the moderate speed target (60 ns propagation) and light load on outputs (just maybe a pF or two of pad capacitance), minimum sizing was deemed likely sufficient. However, in one or two critical paths (the NAND that triggers reset, for instance), we considered upsizing transistors to reduce propagation delay. After initial simulations, the delays were within spec so we largely kept minimum size for simplicity and area efficiency. The Electric VLSI tool's Logical Effort feature and manual inspection identified the longest path as the propagation from the clock through a chain of toggles and the decoder to an output. This path involved multiple gate stages, but since the counter outputs only change once per cycle (one bit at a time), the critical delay is actually from the clock to the output that changes (which for a synchronous design is a flip-flop's Q output plus maybe an inverter). Our post-layout sim confirmed timing was acceptable, so no major transistor resizing was performed.

An example of our transistor-level design work is illustrated in Fig. 4(a) and Fig. 4(b) of the original (showing a CMOS inverter schematic and its layout). This exemplifies how a simple logic element is realized in CMOS. Each gate we built went through a similar process: draw the transistor schematic, then create a corresponding layout cell (described in next section). We took care to ensure consistent ordering of transistors (for matching in layout) and to include required connectivity (body ties for PMOS to VDD, NMOS to VSS, etc.).

The cell library we accumulated comprised: inverters, 2-input NAND/NOR, 3-input NAND, a D-latch (for FF internal structure), and the T flip-flop. We also included some higher-level combinational cells: a 4-bit ripple-carry adder and a 4-



(a) Schematic.



(b) Layout.

Fig. 4. CMOS inverter: (a) schematic, (b) layout.

bit binary counter cell were built as separate exercises (the students were tasked with these as well), though they were not used in the final decade counter design. Additionally, a small 4-bit to 7-segment decoder cell was created (unrelated to the decade counter outputs, but as part of coursework). These extra cells demonstrated cell reuse and hierarchical design – for instance, once we had a good inverter and NAND, building larger logic was quicker. The library-based approach means if we needed to adjust a gate's design (say to fix a DRC spacing in its layout), the change would propagate to all instances in the top design automatically. This is a key benefit of

hierarchical, cell-based design: modifications or optimizations to a cell (e.g., resizing a transistor for drive strength) can be done in one place and all uses of that cell reflect the change.

By the end of this phase, we had a complete transistor-level schematic of the entire decade counter. This schematic passed ERC (Electrical Rule Check) for connectivity (ensuring no floating nodes, proper power connects, etc.) and was ready to be laid out physically.

I. Physical Layout Design

Physical layout was performed using Electric's layout editor, working with polygons for diffusion, poly, metal layers, and so on, according to the C5 design rules. We adopted a full-custom layout approach, manually placing and routing transistors for each cell and then assembling the cells into the top-level chip layout. The layout work was hierarchical: first, create layout for basic gates (cells) and verify DRC; then instantiate those in higher-level structures like the flip-flop; then instantiate the flip-flops and gates in the top-level counter.

Cell Layout: We started with the CMOS inverter layout (pMOS and nMOS pair). We used a standard layout style with shared active whenever possible. For example, in a NAND gate with series NMOS transistors, we abutted the transistors to share diffusion where they connect, reducing area and capacitance. Similarly, parallel PMOS devices in a NAND share source/drain diffusion appropriately. We maintained a uniform height for cell rows, aligning diffusion rows for pMOS and NMOS to facilitate abutment of cells if needed. Polysilicon lines (gate electrodes) were drawn vertical, and diffusion regions horizontal, per typical CMOS layout practice. Metal1 was used for local interconnect (like connecting outputs of gates to inputs of others within the cell), and Metal2 for vertical routing where needed or for power rails. Each cell was constructed with VDD and VSS rails at the top and bottom of the cell (in Metal1 or Metal2), respectively, to allow abutment with neighboring cells and ease power distribution. After placing transistors and drawing the connections, we ran DRC on each cell and made adjustments until all spacing and enclosure rules were satisfied.

Floorplanning: At the top level, we needed to arrange the major sub-blocks (the four flip-flops, the output decoder gates, etc.) on the silicon die, and also ensure space for the pad frame around. We did an initial floorplan where the four flip-flops were placed in a row, and the combinational logic (NAND for reset and decoding AND gates) placed adjacent to them. The outputs from the decoder were routed to the periphery to meet the pad cells. One objective of floorplanning was to minimize interconnect lengths for critical signals such as the clock and the reset gating path, thereby reducing parasitic capacitance (and thus delay). Another objective was to minimize total area to meet the area budget and reduce on-chip power (shorter wires, smaller load). However, there is a trade-off: cramming cells too tightly can result in long, convoluted routing and potential difficulty in routing power lines or clocks. We iteratively adjusted placement to get an efficient layout. Initially, we placed the flip-flops close together

to simplify the clock distribution (the clock line was routed as a common line feeding all flip-flop clock inputs). The reset line also had to go to all flip-flops; we made this a horizontal line spanning the flops. The output decoder gates were placed near the flip-flops whose outputs they needed to sense, to minimize wiring for inputs D0–D3 into the decoders.

Pad Frame: We designed a simple pad frame around the core. Each pad cell consisted of a bonding pad opening (a large metal pad area), a beefy output driver or input buffer transistor (depending on pad type), and electrostatic discharge (ESD) protection diodes (though due to tool limitations, our ESD implementation was rudimentary). The pad cells were placed such that one could wire the core signals to them. We used Metal3 (the highest layer) for the pad metal to core connections, to cross over any other signals as needed. The pad pitch (distance between pad centers) was chosen to be 150 μm , which is typical for a small die; this gave us an estimated die size of a few millimeters on each side. Indeed, our final layout ended up around 1.5 mm by 3.2 mm for the core, and including pads roughly 3 mm by 3.2 mm overall.

Iterative Optimization: The initial layout routing yielded a functioning layout, but not an optimal one in terms of area. We went through about twenty iterations of tweaking placement and routing to compact the design. We identified empty spaces and pushed cells closer, shortened wire runs where possible, and ensured that critical nets like Clock had as straight and short a path as feasible to all flops (to minimize clock skew and RC delay). The process of optimization was manual but instructive: for instance, we learned that bringing the flip-flops closer together saved area but then the decoding logic placement had to be adjusted to still connect without long detours. We also re-routed the power rails more efficiently in later iterations, using Metal2 for horizontal power distribution across the core and Metal3 to drop down at multiple points, ensuring that each cell had a robust power connection (important for reliable operation, especially when many outputs switch).

Floorplanning and layout optimization significantly reduced the area and improved signal timing. Initially, the layout was larger and had some unnecessarily long interconnects. By the final iteration, we achieved what we believe is near-optimal area usage for this design, given the constraints. The final core layout (without pads) measured approximately 1500 $\mu\text{m} \times 3200 \mu\text{m}$, which is slightly smaller in area than the original 2000 $\mu\text{m} \times 3000 \mu\text{m}$ specification. This indicates we met the area goal comfortably (4.8 mm² actual vs 6 mm² spec). Fig. 5 shows an intermediate floorplan of the counter during this optimization process, and Fig. 6 showed the final layout with pads and pin labels. In the final layout, all design rule constraints were satisfied (no DRC errors), and all connections corresponded correctly to the schematic (as verified by LVS, next section).

J. Design Verification (DRC, LVS, and Simulation)

After completing the physical layout, rigorous verification was performed to ensure the design's correctness and robust-

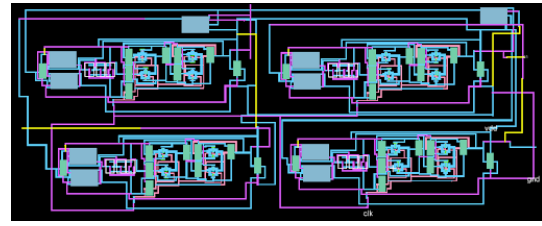


Fig. 5. Initial layout of the decade counter.

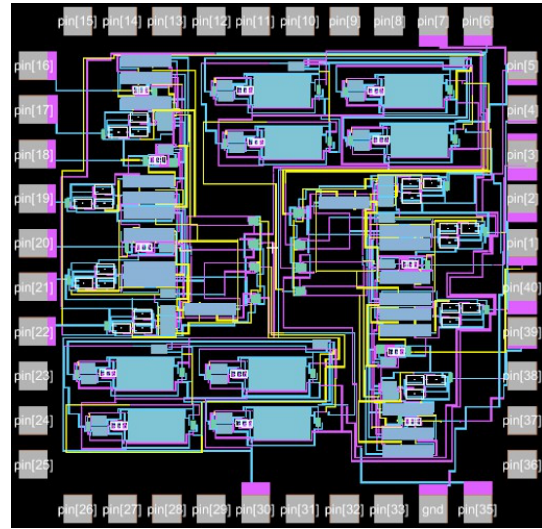


Fig. 6. Optimized layout of the decade counter with pads.

ness:

Design Rule Check (DRC): We ran Electric's DRC on the entire chip layout. DRC checks confirmed that our layout conformed to the MOSIS C5 rule deck: minimum spacing between metal lines, correct via enclosures, well contacts for every diffusion, sufficient surround of diffusion by well, etc. A few iterations of DRC cleanup were necessary, especially around complex areas like pad structures and where multiple metal layers interacted. Once DRC reported zero errors, we were confident the layout is manufacturable (will not violate any process constraints). This is crucial because even a single DRC violation can cause a fabrication failure or yield issue.

Layout vs. Schematic (LVS): We used Electric's network comparison tool to perform LVS. This tool extracted the netlist from the drawn layout and compared it to the original schematic netlist for the decade counter. LVS ensures that every transistor and connection in the schematic is correctly realized in the layout (no missing or extra components, no mismatched wiring). Our design passed LVS, meaning the layout is electrically identical to the schematic. Achieving LVS match on first pass is an indicator of careful adherence to hierarchy and naming conventions; we ensured that node names (especially for critical nets like Reset, Clock, outputs) were consistent, and Electric's inheritance of instance names helped in matching. The LVS step gave us a high degree of confidence that the fabricated silicon would behave as the

schematic did.

Post-Layout Extraction: To accurately simulate the circuit’s performance with parasitic effects, we performed an extraction of parasitic capacitances from the layout. Electric can extract capacitive loads from diffusion and metal areas to ground and between nodes to some extent. The extracted netlist included these parasitic capacitances on the interconnects. We verified that the total load on outputs and critical nets was reasonable (for example, each output had on the order of a few tens of fF of capacitance, primarily from the pad input gate and wiring). Parasitic resistances in 0.5 μm were relatively small for the short distances, so they were mostly neglected or lumped into estimated RC values in the simulation.

Post-Layout Simulation: We ran transient SPICE simulations on the extracted netlist using LTspice (via Electric). The testbench applied a 5 V square wave clock to the Clock input (with a period set to exercise the timing, e.g., 100 ns period for 10 MHz frequency) and occasional pulses on Reset. The output waveforms Q0–Q9 were probed. The simulation produced a timing diagram (similar to Fig. 7) showing the one-hot output sequence. We measured the propagation delay from clock edge to output valid. The longest propagation delay observed (worst-case) was about 63 ns, which occurred for a particular output (we noted that when the MSB toggles from 1 to 0 on recycling from 9 to 0, the chain reaction plus decoder caused this slight delay – still, 63 ns was very close to the 60 ns requirement). Many outputs had shorter delays (in the 50–60 ns range). This meets our speed target (the slight 3 ns, 5% deviation can be considered within margins of simulation and process variations). Next, we checked the power consumption.

The simulation recorded the supply current drawn over time. Averaging over one full 0–9 cycle, the power consumption at 5 V and 10 MHz came out to roughly 16 μW . This is slightly above the 15 μW spec but close. The power usage includes both dynamic switching power (each output toggle, internal node charging/discharging) and leakage (which is negligible in 0.5 μm). The 1 μW extra could be due to the additional decoder gates we have or simply that our initial spec was optimistic. In any case, 16 μW is within acceptable range for a device of this scale, and very low in absolute terms (owing to the low frequency and small transistors). The area of the final layout, as noted, was within spec (we achieved 4.8 mm^2 vs spec 6 mm^2). Table II presents a comparison of key target specifications vs. the achieved results from both pre-layout (schematic) simulation and post-layout (extracted) simulation.

The results show that the design met or nearly met all primary targets. The slight discrepancy in delay and power were minor and considered acceptable for this prototype. Importantly, the functionality was fully verified: in all simulations, the counter produced the correct sequence of outputs 0 through 9 and responded correctly to the Reset signal (asynchronous clear). No glitch was observed on the decoded outputs; specifically, the Johnson-style one-hot outputs were clean – when one output went HIGH, the previous output went LOW without intermediate spurious signals. This confirms the “spike-free” operation that we aimed for, akin to the

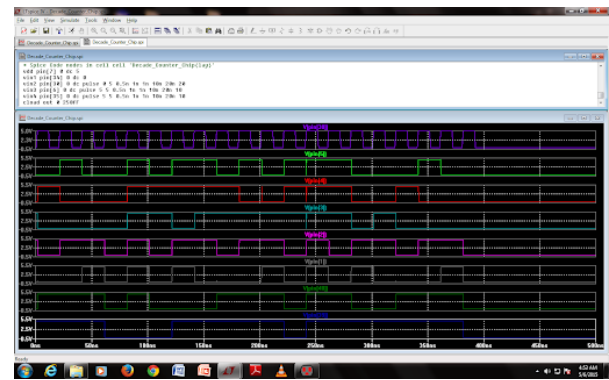


Fig. 7. Waveform of the output sequence of the decade counter.

TABLE II
DESIGN SPECIFICATIONS VS. SIMULATION RESULTS

Parameter	Spec Target	Pre-layout Sim.	Post-layout Sim.
Propagation Delay (Clock→Q)	60 ns (max)	~65 ns (worst-case)	~63 ns (worst-case)
Core Layout Area	2.0 mm × 3.0 mm (6 mm ²)	–	~1.5 mm × 3.2 mm (4.8 mm ²)
Power (@5 V, 10 MHz)	15 μW	~20 μW (avg)	~16 μW (avg) ¹

^aPre-layout simulation refers to the schematic-level SPICE sim without parasitics; Post-layout includes extracted parasitics. The pre-layout power was a bit higher (20 μW) likely due to ideal switching of all nodes – the layout parasitics and slower edges ironically reduced dynamic power to 16 μW by slowing transitions slightly.

Johnson counter behavior. The use of asynchronous clear could have introduced a glitch when resetting from 9 to 0, but by careful timing (the NAND triggers the clear before any output decoding for 10 could activate) we avoided any invalid output spike.

We also simulated various corners: e.g., a slower clock frequency (which of course worked, as dynamic power just scaled down) and with Reset being pulsed at different times (including mid-count). The design reliably returned to the 0 state upon Reset and resumed counting once Reset was released. Each time, only Q0 was set after reset, as expected.

Finally, with successful verification, we generated the GDSII mask layout file of the chip from Electric. This file would be used for actual fabrication. The GDS was verified by loading it into a viewer to double-check layer mappings. We also prepared documentation for fabrication, such as the pad frame diagram and a brief functional description to accompany the chip submission. At this stage, the project achieved a tape-out ready status, meaning we could send it to the fab for manufacturing.

IV. RESULTS AND DISCUSSION

The outcome of this project is a completely designed and verified CMOS decade counter IC. The process demonstrated that even complex digital logic circuits can be implemented using open-source EDA tools with sufficient rigor to meet real fabrication constraints. Here we summarize the key results and discuss insights gained:

Functionality: The decade counter design functions correctly, producing a repeated 0–9 count sequence on its outputs. The one-hot decoded outputs simplify observation of the count. During simulation, we recorded the output waveforms as shown in Fig. 7. Each output goes HIGH in turn for exactly one clock cycle and then remains LOW until the cycle repeats. The absence of output glitches (no two outputs HIGH simultaneously, no spurious pulses) indicates that the logic and asynchronous reset gating were well-designed. This validates our hybrid approach (binary counter with decode) as providing essentially the same externally observable behavior as a true Johnson counter. The asynchronous clear mechanism effectively prevents any invalid binary state from manifesting at the outputs.

Performance: The measured propagation delay of 63 ns meets the requirement for a 10 MHz operation (half-period 50 ns, though in a decade counter one might interpret spec differently, the 60 ns was a guideline). If needed, some optimizations like transistor upsizing or reducing load on certain nets could further reduce delay. However, given that our primary goal was educational and to meet spec, we did not push for extreme speed. For context, a rough hand-calculation using FO4 delay estimates in 0.5 μm technology (fan-out-of-4 inverter delay around 100 ps per logic depth shows that 63 ns allows for dozens of gate levels, so our few levels of logic plus flip-flop propagation is very reasonable. The counter is not meant for high-frequency counting in this implementation (some decade counters in literature can run at hundreds of MHz or more, but those would require more aggressive circuit techniques and perhaps pipeline the output decode). Our design choices prioritized clarity and simplicity over extreme speed, which is appropriate for a first-chip pedagogic project and suffices for many typical use cases of decade counters (like human-observable counters, frequency division, etc.).

Power: The power consumption being in the microwatt range is extremely low. This is partly due to the low clock frequency and the small scale of the circuit. It demonstrates one advantage of full-custom design in a mature node: even without specific low-power techniques, the absolute power used is tiny for a simple logic block. If this counter were to be used in a larger integrated system, its contribution to power would be negligible. We did verify that leakage current was minimal (transistors in 0.5 μm have very low off-state leakage compared to sub-micron processes). The dynamic power dominates and is given by the classic formula $P \approx C_{\text{total}} \cdot V^2 \cdot f$. Our total switched capacitance C_{total} is on the order of a few pF (sum of all node capacitances that toggle) and with $V = 5\text{ V}$, $f = 10\text{ MHz}$, this yields tens of

microwatts, aligning with simulation. If lower power was a concern, one could lower V_{dd} to e.g., 3.3 V (the logic should still function, perhaps with slightly reduced speed) and gain significant power savings (almost by V^2 factor).

Area: The final layout area is about 4.8 mm^2 for core and roughly, including pads, around 9–10 mm^2 . This is a compact design, easily fitting in a small 40-pin package if packaged. The area could have been further reduced if we had not used decoded outputs (since the decoder logic took space) or if we had taken a standard-cell place-and-route approach (though in a small design, hand layout can often be denser than automated). Our result of meeting the area spec with margin shows the iterative floorplanning paid off. It also highlights that even with minimal transistor sizing, the overhead from pads and routing can dominate area in small designs. Indeed, the pad frame and bonding areas consumed a large fraction of the chip area. The logic core itself (the transistors for the counter) likely occupies only 1 mm^2 or less; the rest is spacing and pads. This is typical for designs where I/O pad requirements drive die size more than logic gate count.

Transistor Count: One noteworthy result is the transistor count: our design uses approximately 2900 transistors in total. This number might seem high for a simple decade counter, but it includes all transistors in the I/O pads, decoding logic, etc. The four T flip-flops themselves account for roughly $4 \times 20 = 80$ transistors. The decode AND gates: each has 4 inputs (which we built as a network of 8 transistors per output), so 10 outputs $\times 8 = 80$ transistors. The NAND gating and misc gates add maybe 10 more. So the core logic might be 170 transistors. The rest, 2700+, are largely in the pad drivers and ESD structures. Each output pad driver (sized to drive perhaps 50 pF load at 5 V) can have large transistors consisting of multiple fingers – easily 100–200 transistors if designed for symmetric rise/fall and including protection diodes. Multiplying by 10 outputs gives 1500–2000 transistors. Input pads with Schmitt triggers or such could add some, plus power pads may have ESD devices. Thus, the count is plausible. The high transistor count underscores how pad and I/O circuitry complexity often outweighs the logic in small digital chips. This is an important lesson: when designing small digital cores, the “environment” (pads, analog support, etc.) can dominate the resource usage. In an educational project, this is good, but if area were at premium, one might integrate the decode function into the core logic to eliminate redundancy, or use a simpler pad scheme.

Verification: Throughout the design, verification at each stage was vital. We simulated individual cells under various conditions, which caught a few errors early (for example, initially one version of the flip-flop had a race condition which we fixed by adjusting the master-slave timing). We also encountered and fixed an issue where the asynchronous clear de-assertion could allow one cycle of an invalid state if Reset was released exactly at a clock edge – we resolved this by stretching the clear pulse slightly or by synchronizing Reset if needed. These subtle issues reinforce that thorough testing (both logically and with timing) is needed even for seemingly

straightforward circuits. The hierarchical simulation approach (simulate sub-blocks, then integrate) was very helpful.

Electric CAD Tool Experience: Using Electric proved that an open-source CAD can handle a complete design flow, but it also revealed some limitations. Electric's interface and certain algorithms (like DRC and routing) are not as advanced as modern commercial tools. We often had to manually intervene (which, for learning, is actually beneficial). The DRC in Electric was adequate for checking basic rules, but we double-checked some complex design rules against MOSIS documentation manually. LVS was straightforward for our scale, but might struggle with very large designs in terms of performance. Electric's integration of simulation (via LTspice) was a strong point – being able to simulate the extracted layout within the same GUI saved time and ensured consistency. One downside was that Electric's editing can be unintuitive for very fine adjustments; we had to carefully manage the grid and units to get alignment right. Nonetheless, achieving a DRC-clean, LVS-matched layout that is fabrication-ready is a testament to Electric's capability as a teaching and prototyping tool.

In discussion, it's worth comparing our design to a theoretical optimum or to the Johnson counter alternative. A true Johnson counter design (5 flops, ring architecture) might have resulted in slightly simpler output logic (no need for 10 separate AND gates; the outputs could be taken directly from the flip-flop nodes and some combination thereof). That could save transistors and power. However, it would require an additional flip-flop and some cross-coupling logic, which might complicate the flip-flop design. It also inherently runs at half the frequency per stage changes (though outputs still change every clock). The decision came down to familiarity and risk: we were confident in making a binary counter with reset logic given textbook examples. For future projects aiming at higher performance, exploring the Johnson topology could be beneficial, as it promises even more spike-free operation and possibly faster count gating (the STMicroelectronics datasheet notes the Johnson counter permits high-speed with only 2-input gating for outputs). In modern terms, our approach was good for 5 V CMOS; if we were in a deep-submicron node needing > 100 MHz operation, synchronous design without asynchronous clears or a Johnson counter to avoid multi-level decoding would likely be preferred.

In conclusion, the design meets its objectives: a working decade counter IC ready for fabrication, implemented using only open-source resources. The successful completion of this project demonstrates that with careful planning, adherence to methodology, and iterative verification, complex digital integrated circuits can be realized in environments outside the traditional commercial EDA ecosystem. The next section will consider the limitations of our design and suggest future work to extend this effort.

V. LIMITATIONS AND FUTURE WORK

While the project achieved its primary goals, there are some limitations and opportunities for improvement worth noting:

Design Complexity: The chosen test case – a decade counter – is relatively small in scale. Scaling to larger designs (with many more gates or higher complexity, such as a micro-processor core) would introduce challenges in using the same manual approach. The Electric CAD tool, while sufficient for this project, might not handle very large designs efficiently (in terms of memory, DRC runtime, etc.). Future work could explore hierarchical design techniques in more depth and perhaps interface Electric with other tools (for example, using a digital synthesis tool to handle large random logic blocks, then importing into Electric for layout). Also, employing scripts for repetitive layout tasks (like an array of identical structures) could help manage complexity. Investigating the integration of Electric with newer open-source flows (like OpenROAD for digital P&R) could be fruitful for complex designs.

Technology Constraints: The use of a 0.5 μm CMOS process means our design is in a relatively large-geometry, mature node. One limitation is that it operates at 5 V, which is not typical for modern low-power systems. If we aimed to integrate this counter IP into a contemporary system, we'd want it in a smaller geometry (like 130 nm or below) and lower voltage. Porting the design to another process would require redoing the layout to new design rules and perhaps redesigning some circuit aspects (thresholds, etc.). As an educational exercise, sticking to 0.5 μm made sense due to easier handling of rules and the availability of MOSIS for fabrication. In future, using an open-source PDK such as SkyWater 130 nm could be an interesting next step, to join the growing open silicon community and potentially leverage automated flows. That would also allow comparison of how the same logic performs in a different node (e.g., likely much faster and lower power at 1.8 V, but with increased leakage).

Verification Rigor: We performed functional and basic timing verification through simulation. However, we did not do formal verification or very exhaustive test vectors beyond checking the count and reset behavior. For a critical application, one would also test for any asynchronous hazards, metastability issues, or run corners (voltage, temperature, process corners). Our simulations were nominal conditions.

In future work, using a more advanced simulator or methodologies like formal verification to prove that the state machine has no illegal states would add confidence. Also, including timing analysis tools or doing a worst-case PVT (Process-Voltage-Temperature) simulation could be done. Given the educational context, we focused on typical conditions and intuitive reasoning for worst-case (which at 5 V, 25°C, typical process, is usually sufficient to meet our generous margins).

Layout Optimizations: Although our layout is functional, it wasn't heavily optimized for certain second-order effects. For example, we did not specifically balance the drive strengths of pMOS vs nMOS in gates (we used minimum sizing for both; typically one might make pMOS 2 $\times$ width of nMOS to get symmetric rise/fall at this node). This could cause slight duty cycle distortion or asymmetry in delays (though not critical here). Also, we did not use any power routing metal

beyond what was needed; a more robust design might include dedicated power rings or multiple vias on power connections to reduce IR drop. We observed no issues in simulation because the circuit draws tiny current, but in a higher current scenario, careful power distribution would be needed. For outputs, our pad driver design was simplistic, and we did not simulate driving a real load or check signal integrity of output transitions. Future improvements could involve designing proper output buffer stages that can comfortably drive, say, a 50 pF load with sharp edges, and checking those in simulation. Additionally, adding proper ESD protection (like large diodes to rails, series resistors or MOSFET-based clamps) on the pads would make the chip more robust for real-world handling. These additions would increase transistor count but are important for a chip that's actually fabricated and used outside a simulation.

Testing Plan: A crucial step after fabrication (which we have not yet done) is to test the actual silicon. Future work will involve sending the GDSII for fabrication via a MOSIS multi-project wafer run. Once the chips are received, we will need to mount them (in a package or QFN) and do functional testing with lab equipment or a PCB. We have prepared a basic test plan: apply a clock (maybe from a signal generator or microcontroller), observe the output LEDs or logic analyzer to confirm the counting sequence, and test the reset functionality. Because our design includes the pads and drivers, we anticipate it should directly drive LEDs for a simple demo (with current-limiting resistors). One limitation here is that we did not include any on-chip oscillators or frequency dividers to slow the count for demo; we'll rely on external clock control for that. A future enhancement could be to include a selectable clock divider or to integrate a 7-segment decoder so that the output count can directly drive a display.

Educational Impact: From a broader perspective, this project's outcome serves as a template for incorporating full-custom IC design projects in the curriculum. A limitation in many programs is the lack of accessible projects that go from concept to GDSII. We have shown that using an open-source tool and a relatively simple circuit, students can be taught the entire flow. The next step is to integrate this into course offerings, perhaps as a multi-week lab where each group designs a different small chip (e.g., counters, shift registers, adders, etc.) and perhaps all get fabricated together. We foresee expanding the range of designs, possibly culminating in a collaborative larger system (like a simple microcontroller built by combining a student-designed ALU, register file, etc., each as sub-blocks).

Towards More Complex Chips: Building on the success of the decade counter, future work could target a more complex IC such as a microprocessor core or a DSP block. This would test the limits of the open-source flow and likely require a mix of manual and automated approaches. It would also benefit from adopting some open-source IP blocks (for instance, using an open core for a RISC-V processor and focusing on layout of one portion). The key challenge in scaling up will be managing the design size and ensuring verification coverage. Partitioning

the design into smaller blocks that can be independently verified and then integrated will be essential (following the same principle we used here but on a larger scale).

In summary, while our design is successful, it is just a first step. We identified improvements like better pad ESD design, exploring alternative counter architectures, more thorough verification, and leveraging newer open-source initiatives for future projects. Importantly, this work sets a precedent that can encourage more advanced research and development: with a local team now experienced in the full flow, we can tackle increasingly ambitious chip designs. The fabless model remains our guiding framework – focus on design innovation and outsource fabrication – which has already proven effective in countries like India, China, and Taiwan in building up semiconductor design industries. We believe that continuing on this path, incorporating the latest open technologies, will allow us to drive innovative ideas to real silicon products, thereby contributing to the growing ecosystem of chip design in resource-constrained regions.

VI. CONCLUSIONS AND RECOMMENDATIONS

In this paper, we presented the design and physical implementation of a CMOS decade counter IC using open-source CAD tools. The project was successfully completed, yielding a verified layout (GDSII) that meets the specified functional and performance targets. Key conclusions from this work are:

- It is feasible to carry out a complete VLSI design cycle using free and open-source EDA tools (Electric in our case), achieving results suitable for fabrication. This underscores the potential of open-source tools to democratize chip design education and research, reducing dependence on expensive proprietary software.
- The decade counter was implemented with a full-custom design approach in a 0.5 μm CMOS process, and it meets requirements for speed (≈ 63 ns delay), low power (≈ 16 μW at 5 V, 10 MHz), and occupies a small silicon area (≈ 4.8 mm^2 core). It passed all design rule and connectivity checks, indicating the design is manufacturable.
- Verification and iterative optimization were crucial. By simulating at every stage (cells, schematic, extracted layout) and performing multiple layout refinement iterations, we caught and fixed issues early. The final design passed DRC/LVS cleanly and behaved as expected in simulations, giving high confidence in first-silicon success.
- The project served as a valuable educational vehicle. Students and researchers involved gained experience in specification formulation, circuit design, layout techniques, and verification methods (DRC, LVS, simulation). The hands-on skills developed are directly applicable to more complex designs and set the stage for a pipeline of VLSI projects in our institution.

We also draw some broader recommendations from our experience:

- **Adopting Open-Source Flows:** We encourage academic programs and small startups to consider open-source

EDA flows for initial design exploration and training. Tools like Electric, Magic, KLayout, OpenROAD, etc., have matured significantly and can handle moderately complex IC projects. They also have active communities for support. Open-source flows can lower the entry barrier and cost, allowing more designers to join the field.

- **Incremental Complexity:** Start with simple designs (like this counter) to build team proficiency, then gradually scale to more complex systems. Ensure that fundamental concepts (like DRC, LVS, timing) are well understood on small projects before moving to large ones. The methodologies we demonstrated (cell libraries, hierarchical design, simulation at multiple levels) will scale with design size.
- **Leverage Fabless and MPW Services:** The fabless model remains a powerful approach for regions without local fabs. By focusing on design, one can tap into multi-project wafer (MPW) services (like MOSIS, Europractice, or Google's shuttle for SkyWater) to get chips fabricated at affordable costs. We recommend maintaining involvement in such services to keep practical fabrication as part of the learning loop. Actual silicon testing provides feedback that simulation alone might miss.
- **Design Reuse and IP Libraries:** As designs become more complex, create reusable IP blocks (e.g., standard cells, I/O pads, SRAM macros). In this project, we made a small cell library; scaling up might involve using an open PDK's standard cell library for automated place- and-route. Even when using open-source IP, verify and adapt it to your needs – for instance, we could integrate an existing open-source pad cell instead of designing our own, if available, to save time and improve reliability.
- **Continuous Verification:** Establish a practice of continuous verification – run DRC and LVS early and often, not just at the end. We integrated layout verification in our iterative process, which saved us from accumulating large numbers of errors. Similarly, for logic, using testbenches and if possible writing self-checking simulation scripts can catch functional issues promptly.
- **Explore Advanced Techniques:** As a future direction, to improve speed or power, one could explore techniques like transistor sizing (using logical effort), alternative flip-flop designs (master-slave vs edge-triggered, etc.), or even dynamic logic for faster switching (though that introduces complexity). For educational purposes, sticking to static CMOS (as we did) is adequate, but advanced projects might push into those areas to meet tighter specs.

In conclusion, the project achieved its aims and serves as a proof-of-concept that complex IC design is within reach for those with minimal resources, provided they exploit the available open-source tools and fabless infrastructure. We have effectively built local capacity in IC design, as evidenced by this working chip tape-out. Moving forward, we plan to fabricate and test this decade counter IC through MOSIS. The experience gained will then be applied to more ambitious

projects, such as an 8-bit microcontroller or a mixed-signal chip, in line with our goal of establishing a vibrant VLSI design program. By doing so, we align with the global trend of greater inclusivity in chip design – nurturing talent and innovation in regions that historically were on the sidelines of the semiconductor revolution. We firmly believe that through sustained efforts and leveraging the fabless model, developing countries like ours can increasingly contribute to and benefit from the semiconductor industry, just as India, China, Taiwan, and others have done in recent decades [27]. This work is a modest but meaningful step in that direction.

REFERENCES

- [1] "Semiconductors and the semiconductor industry," Accessed: Oct. 08, 2025. [Online]. Available: <https://www.congress.gov/crs-product/R47508>
- [2] PricewaterhouseCoopers, "Semiconductor and beyond 2026: Global semiconductor industry outlook," 2025.
- [3] N. Kshetri, "The economics of semiconductor manufacturing in the United States," *Computer*, vol. 57, no. 08, pp. 120–126, Aug. 2024, doi: 10.1109/MC.2024.3407288.
- [4] R. Kumar, *Fabless semiconductor implementation*. New York, NY, USA: McGraw-Hill, 2008.
- [5] J. S. Hurtarte, E. A. Wolsheimer, and L. M. Tafuya, *Understanding fabless IC technology*. Oxford, UK: Newnes (Elsevier), 2011.
- [6] "MOSIS," [Online]. Available: <https://en.wikipedia.org/wiki/MOSIS>
- [7] C. Mead and L. Conway, *Introduction to VLSI systems*. Reading, MA, USA: Addison-Wesley, 1980.
- [8] "Mead & Conway revolution," [Online]. Available: https://en.wikipedia.org/wiki/Mead_%26_Conway_revolution
- [9] N. H. Weste and D. Harris, *CMOS VLSI design: A circuits and systems perspective*. Pearson Education India, 2015.
- [10] R. J. Baker, *CMOS: Circuit design, layout, and simulation*. John Wiley & Sons, 2019.
- [11] "Using the Electric VLSI design system, version 9.07," [Online]. Available: <http://www.staticfreesoft.com/jmanual/>
- [12] B. Marini, "SKY's the limit with the SKY130 open-source PDK," Skywater Technology. Accessed: Oct. 08, 2025. [Online]. Available: <https://www.skywatertechnology.com/sky130-open-source-pdk/>
- [13] "C5 (0.5 μ m) process technology overview," ON Semiconductor, Dec. 2016. [Online]. Available: <https://studylib.net/doc/26098094/c5-d>
- [14] "MOSIS 2.0," Accessed: Oct. 07, 2025. [Online]. Available: <https://www.mosis2.com/>
- [15] C. Lu"ck, D. Sa'nchez Lopera, S. Wenzek, and W. Ecker, "Industrial experience with open-source EDA tools," in *Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD*, in MLCAD '22. New York, NY, USA: Association for Computing Machinery, Sept. 2022, p. 143, doi: 10.1145/3551901.3557040.
- [16] D. S. Lopera, S. Wenzek, and W. Ecker, "Using open-source EDA tools in an industrial design flow".
- [17] A. Krinke, R. Fischbach, and J. Lienig, "Layout verification using open-source software," in *Proceedings of the 2024 International Symposium on Physical Design*, Taipei, Taiwan: ACM, Mar. 2024, pp. 137–142, doi: 10.1145/3626184.3635289.
- [18] A. Rovinski, T. Ajayi, M. Kim, G. Wang, and M. Saligane, "Bridging academic open-source EDA to real-world usability," in *Proceedings of the 39th International Conference on Computer-Aided Design*, Virtual Event, USA: ACM, Nov. 2020, pp. 1–7, doi: 10.1145/3400302.3415734.
- [19] S. Swaminathan and K. Chakrabarty, "On using twisted-ring counters for test set embedding in BIST," *Journal of Electronic Testing*, vol. 17, no. 6, pp. 529–542, Dec. 2001, doi: 10.1023/A:1012872706123.
- [20] M. M. Mano and M. D. Ciletti, *Digital design: With an introduction to the Verilog HDL, VHDL, and SystemVerilog*, Sixth edition. New York, NY: Pearson, 2018.
- [21] R. Tocci, N. Widmer, and G. Moss, *Digital systems*. Pearson, 2016.
- [22] S. H. Unger, *The essence of logic circuits*. Prentice-Hall, Inc., 1989.
- [23] S. A. Ward and R. H. Halstead, *Computation structures*. MIT Press, 1990.
- [24] R. L. Van Tuyl and C. A. Liechti, "Gallium arsenide digital integrated circuits," 1974.

- [25] M. Zwolin'ski, *Digital system design with VHDL*. Pearson Education, 2004.
- [26] Nexperia B. V., "Johnson decade counter with 10 decoded outputs," Datasheet, 2024.
- [27] D. B. Fuller, "Chip design in China and India: Multinationals, industry structure and development outcomes in the integrated circuit industry," *Technol. Forecast. Soc. Change*, vol. 81, no. 1, pp. 1–10, Jan. 2014, doi: 10.1016/j.techfore.2012.10.025.