ECE-319

CMOS VLSI DESIGN LABORATORY

PROJECT TITLE:- 8-BIT LINEAR FEEDBACK SHIFT REGISTER (LFSR) USING CADENCE VIRTUOSO

LOVELY PROFESSIONAL UNIVERSITY

Transforming Education Transforming India

NAME	Reg No.	Roll No
Suryawanshi Anosh Anand	12306237	12
Vikirthan T	12307334	14
Patil Sairaj Suresh	12303809	10

1) Introduction:

Linear Feedback Shift Registers (LFSRs) play an important role in modern digital, communication, and VLSI systems because of their ability to generate long pseudo random binary sequences using minimal hardware resources. An LFSR consists of a series connection of flip flops that shift their contents on every clock edge, while the next input bit is generated by applying a feedback function, typically an XOR or XNOR, on selected tap outputs of the register. Although the architectural structure is extremely simple, the behavior that emerges from the linear feedback mechanism is powerful, mathematically elegant, and highly predictable. These properties make LFSRs ideal for applications where random looking sequences are needed without relying on heavy computational algorithms or large memory-based sequence generators. Their deterministic nature also makes them easy to analyze and verify, which is an important advantage in hardware design.

LFSRs have been widely adopted in a broad range of engineering applications due to their speed, low hardware cost, and energy efficiency. In communication systems, they form the foundation of spreading codes used in CDMA, direct sequence spread spectrum techniques, and scrambling operations. Their ability to generate long, well distributed sequences allows communication channels to minimize interference and improve data integrity. In digital design and VLSI testing, LFSRs are a key component in Built In-Self-Test architectures, where they generate pseudorandom test patterns that help detect stuck at, bridging, and delay faults with high coverage. They are also used in cyclic redundancy check (CRC) generation, checksum computation, scrambling and descrambling circuits, lightweight cryptographic protocols, and hardware security primitives. Because LFSRs require far fewer gates than alternative sequence generators, they are able to operate at high frequencies while consuming significantly lower power, making them suitable for both high performance and low power embedded systems.

In this project, an 8-bit LFSR is designed, implemented, and analyzed using the Cadence Virtuoso VLSI design platform. The circuit is constructed at the transistor level using CMOS implementations of fundamental logic blocks such as XOR gates, inverters, buffers, and edge triggered D flip flops. These building blocks are interconnected according to the chosen primitive polynomial to form a maximal length LFSR capable of cycling through 255 unique non zero states. After building the schematic, transient simulations are carried out to observe the internal node transitions, validate the shifting behavior, and ensure that the feedback network produces the expected pseudo random sequence. The waveform results clearly show the staircase like propagation of bits from one flip flop to the next, with the feedback value consistently inserted into the first stage on each rising clock edge. The absence of glitches, metastability, or incorrect transitions confirms the functional correctness of the design.

Beyond functional verification, the project also focuses on understanding the internal timing behavior and CMOS characteristics of the LFSR. Important concepts such as propagation delay, rise and fall time, clock to Q delay, power consumption, and the power–delay product are analyzed to evaluate performance. Transistor sizing, capacitive loading on tap nodes, setup and hold time constraints, and routing considerations are examined to better understand how theoretical digital logic maps onto physical hardware.

2) Literature Review/Background Theory:

A Linear Feedback Shift Register (LFSR) is a simple yet surprisingly powerful digital circuit used to generate sequences of bits that look random, even though they follow a fully predictable pattern. At its core, an LFSR is just a row of D flip-flops connected one after another, forming a shift register. With every clock pulse, the contents of the register move one step forward like a conveyor belt shifting bits from one flip-flop to the next.

What makes an LFSR special is the feedback path. Instead of feeding the input with a fixed value, the circuit takes outputs from selected flip-flops, called taps, and combines them using XOR or XNOR logic. The result of this operation is then fed back into the first flip-flop. This creates a circular dependency where the new bit entering the register depends on the current state of specific positions in the register. Mathematically, this process is modeled using polynomials defined over the Galois Field GF(2), where addition is equivalent to XOR. The specific arrangement of taps corresponds to the LFSR's characteristic polynomial, which determines how long and how complex the output sequence will be.

LFSRs generally come in two popular architectures Fibonacci and Galois. The Fibonacci form, which is the one used in this project, gathers all the tap outputs, performs the XOR operation, and feeds the result only into the input of the first flip-flop. This makes it straightforward to understand and visually easy to follow. The Galois form, on the other hand, distributes the feedback to multiple flip-flops, which often reduces hardware cost and propagation delay, but makes the structure slightly harder to interpret. Regardless of the implementation, both forms rely on the same mathematical foundation. If the chosen characteristic polynomial is primitive, the LFSR will generate what is known as a "maximal length sequence," cycling through all possible non-zero states exactly $2^n - 1$unique combinations for an n-bit register before repeating.

Another important aspect of LFSR operation is the seed, or initial value. Because XOR-based feedback cannot transform an all-zero input into anything else, the register would be stuck forever in the zero state if it ever entered it. To prevent this, the LFSR is always initialized with a non-zero seed. Once started, the LFSR cycles through a sequence that seems random but is completely deterministic. This balance of unpredictability and reproducibility is why LFSRs are widely used in applications such as pseudorandom number generation, scrambling, built-in self-test (BIST), and communication systems.

Behind the scenes, the XOR gate plays a critical role it is the element that governs how bits interact and evolve from one clock cycle to the next. Since all flip-flops update simultaneously with the global clock, timing parameters such as propagation delay, setup and hold times, and clock-to-Q delay become essential considerations, especially when implementing an LFSR at the transistor level in a CMOS environment. Understanding how these timing factors influence the circuit helps bridge the gap between the clean mathematical model of an LFSR and the practical realities of building one in Cadence precisely the objective of this project.

3) Architecture And Working Principle:

The architecture of the LFSR designed in this project follows the traditional Fibonacci-style shift register, where the feedback is computed from selected output bits and applied directly to the input of the first flip-flop. The main structure is built from eight D flip-flops arranged in a series chain, with each flip-flop holding one bit of the register. On every rising clock edge, the entire register shifts to the right. This means each flip-flop passes its stored value to the next stage, while the newly generated feedback bit is inserted into the first flip-flop. This simple and uniform shifting action is what allows the LFSR to progress smoothly through its sequence of states.The feedback mechanism forms the functional core of the LFSR. It works by selecting specific tap positions from the register outputs and combining them using an XOR gate. In this design, the chosen tap configuration corresponds to a characteristic polynomial that produces a maximal length sequence. As a result, the LFSR is able to cycle through all possible non-zero 8-bit combinations before returning to its starting point. The XOR gate performs a modulo 2-addition of the tapped bits, and the output of this operation becomes the next input to the register. Since the XOR operation is linear, the overall register behavior remains linear, which is the reason the circuit is called a Linear Feedback Shift Register.

During actual operation, the LFSR begins from a non-zero initial seed. With each clock pulse, the feedback bit is inserted into the first flip flop and all other bits shift one position to the right. For instance, if the register currently holds the sequence Q8, Q7, …, Q1 then the next state becomes feedback, Q8, Q7, …, Q2. This shifting pattern combined with the feedback logic ensures that the LFSR follows a sequence that appears random but is completely predictable. Each state transitions to exactly one unique next state, and no state repeats until the full period has completed. This makes the LFSR operate like a finite state machine with a single long cycle.

One of the main advantages of this architecture is its efficiency in terms of hardware. Only a single XOR gate and a series of flip flops are needed to implement an 8-bit pseudo random generator. The uniform and repetitive structure also helps maintain clean timing behavior, since all flip flops are clocked simultaneously and driven by similar logic paths.

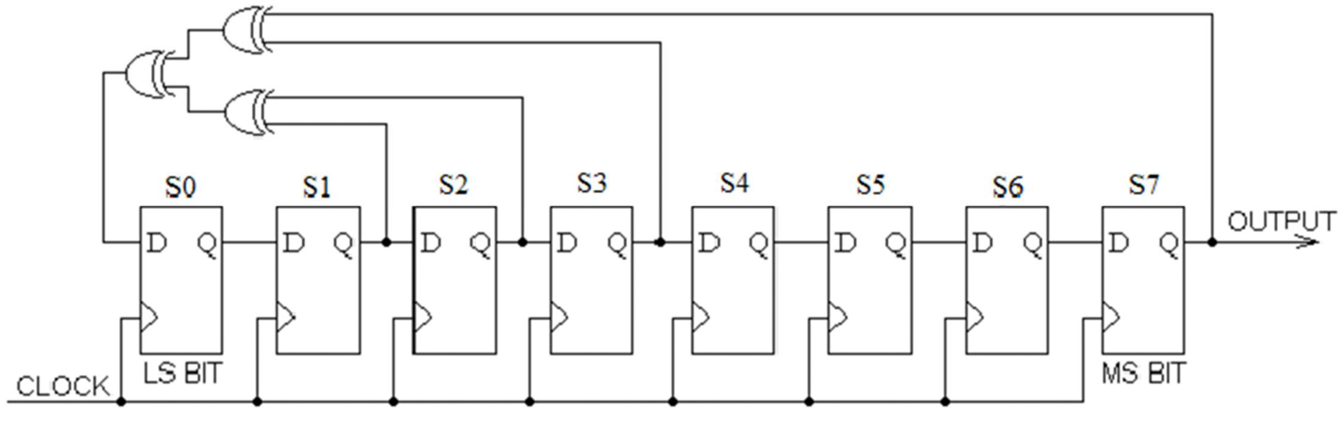


Figure 3.1 Circuit diagram for LFSR

4) System Level Design:

The system level design of the LFSR focuses on arranging the core building blocks of the circuit in a clean and organized way so that the register can perform its sequential state transitions accurately and without interruption. Instead of thinking about individual transistors, the system level perspective treats the LFSR as a complete digital subsystem made up of interconnected modules. The design begins with an 8-bit chain of D flip flops, where each flip flop stores one bit of the current pseudo random value. All of these flip flops are driven by the same global clock signal. This shared clock ensures that every stage updates at exactly the same moment, so the shift operation happens smoothly and consistently on each rising clock edge. Because all bits move together, the system avoids problems like race conditions or timing mismatches that might occur if different parts of the circuit updated at slightly different times.

At the system level, the XOR gate acts as the engine that produces the new feedback bit. Specific outputs of the register, known as taps, are selected and passed into the XOR logic. These tap positions were chosen to match a primitive characteristic polynomial, which guarantees that the LFSR will produce a maximal length sequence. By doing this, the circuit is able to cycle through all non-zero 8-bit patterns before repeating. The output of the XOR gate is routed directly into the input of the first flip flop, while the remaining flip flops simply shift their stored values to the next stage. This arrangement allows the entire register to behave as a single coordinated machine that evolves through its states in a repeatable and mathematically defined manner. Several additional considerations are important at the system level.

The design must maintain clean signal transitions so that each flip flop receives a stable and reliable input. Care must be taken to avoid metastability, a condition where a flip flop becomes uncertain about whether to store a zero or a one. The register must also be initialized with a non-zero seed, since an all zero starting point would trap the LFSR permanently in the zero state. These details ensure that the system operates correctly from the moment it powers up and throughout every clock cycle that follows. Overall, the system level architecture emphasizes simplicity, modularity, and predictability. The design uses only one XOR gate alongside a row of identical flip flops, yet it is capable of generating a very long and useful pseudo random sequence.

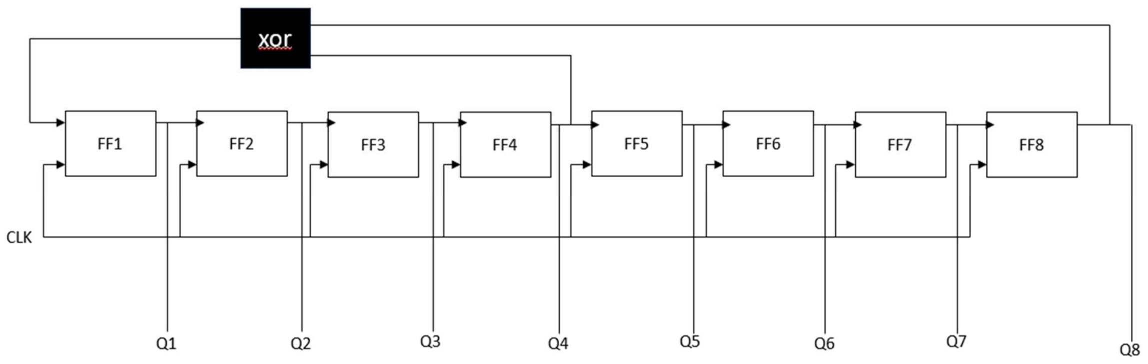


Figure 4.1 System design for LFSR

5) Transistor Level Design:

The transistor level design looks closely at how the entire LFSR is physically built inside Cadence Virtuoso using CMOS logic. At this level, every logic function in the circuit is formed by arranging PMOS and NMOS transistors in a way that produces the correct output for each possible input. Even though the LFSR appears simple at the schematic level, its transistor level realization reveals the detailed electrical behavior that makes the system work reliably.

The most fundamental CMOS gate used throughout the design is the inverter, which is built from one PMOS and one NMOS transistor connected in a complementary way. The PMOS pulls the output high when the input is low, and the NMOS pulls the output low when the input is high. These inverters appear everywhere in the design because they form the basis of the more complex gates. They are also used inside D flip flops to store and maintain stable logic values.

The XOR gate plays an especially important role because it generates the feedback bit that drives the entire LFSR sequence. At the transistor level, the XOR function is created through a network of transistors that only switches its output when the two inputs differ. This behavior is critical because the feedback bit must represent a modulo two addition of the selected taps. If both tap outputs are the same, the XOR produces a zero. If they differ, the XOR produces a one. Since the XOR gate directly influences how the next state is formed, its timing, propagation delay, and sizing are carefully controlled.

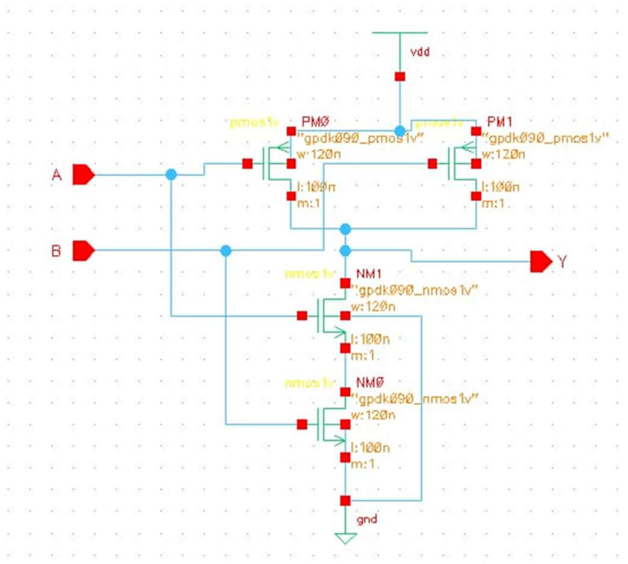
The AND gate is sometimes used inside supporting circuits, especially when creating pulse shaping networks or enabling specific signals. In CMOS, an AND gate is implemented using a series combination of NMOS transistors in the pull down network and a parallel combination of PMOS transistors in the pull up network. This arrangement ensures that the output becomes high only when both inputs are high. Although the classical LFSR does not always require an AND gate, it is often added when designers need to implement clock gating, edge detection, or enable logic.

NOT gate is simply another name for the inverter. It reverses the input value and is used heavily inside the D flip flop structure as well as inside the pulse generation circuits. In the LFSR, NOT gates may also appear when building positive or negative edge detectors, which produce narrow clock dependent pulses used for triggering simulation events or for controlling certain internal nodes.

The NAND gate also appears frequently because it is one of the most efficient CMOS gates. A NAND gate is the complement of an AND gate. It produces a low output only when both inputs are high. In CMOS this is convenient to build because the pull down network uses NMOS transistors in series while the pull up network uses PMOS transistors in parallel. Digital designers often prefer NAND gates because they provide strong noise margins, fast switching, and lower area compared to combinations of AND and NOT gates. Inside flip flops, NAND gates are commonly used to build latches since they respond cleanly to enable signals and create stable feedback loops.

As the design moves deeper into the transistor level domain, parasitic capacitances, wiring delays, and loading effects become important. The feedback signal generated by the XOR gate must settle before the clock edge arrives so that the first flip flop receives the correct next value. Each transistor contributes small capacitive loads that slow down the rising and falling transitions. Flip flops must be carefully sized to maintain the right balance between speed and power consumption, especially because eight identical cells are required in the LFSR.

Building the entire circuit using CMOS transistors results in several advantages. The static power consumption is extremely low, the logic levels have strong noise immunity, and the design remains reliable even when variations occur in manufacturing or operating conditions. Most importantly, seeing the LFSR at the transistor level helps bridge the gap between abstract logical behavior and the way these functions are physically implemented in silicon, allowing designers to appreciate how simple transistors combine to create sophisticated digital systems.



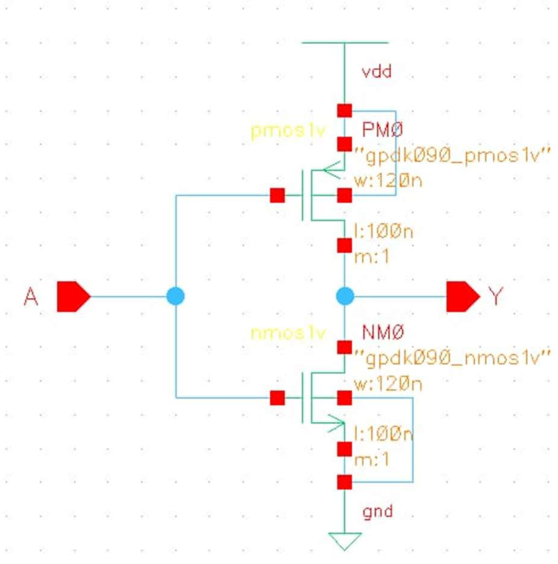
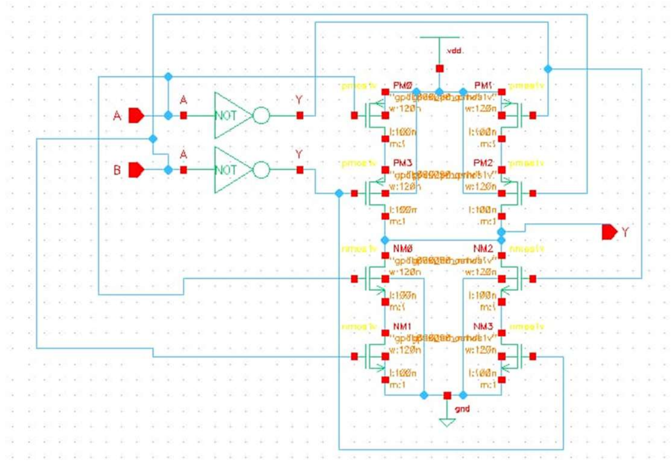


Figure 5.1 NAND GATE

Figure 5.2 NOT GATE



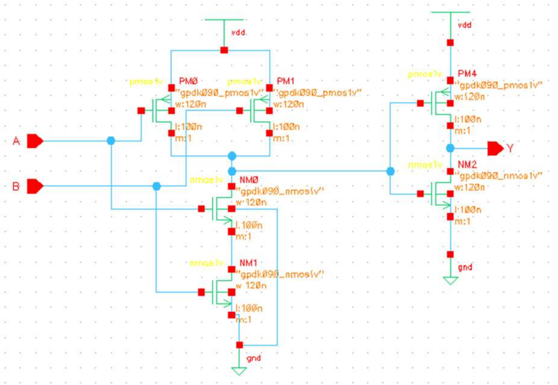


Figure 5.3 XOR GATE

Figure 5.4 AND GATE

6) Complete LFSR Schematics Explaination:

The complete schematic of the 8-bit LFSR represents the integration of all the circuit blocks D flip flops, XOR based feedback logic, clock distribution, and power rails into a unified system. In Cadence Virtuoso, the schematic is arranged in a structured left to right flow, making it easy to visualize the shifting behavior of the register. The eight D flip flops are placed in a horizontal chain, each connected such that the Q output of one stage feeds the D input of the next. This sequential arrangement ensures that during every clock pulse, the stored data propagates through the register in a consistent and controlled manner. Each flip flop receives the same clock signal, guaranteeing synchronous updates across all stages.

At the input side of the chain, the feedback logic is implemented using an XOR gate. The outputs of selected flip flops, called taps, are routed to the inputs of the XOR gate. These taps correspond to a primitive characteristic polynomial that determines the sequence length. For example, in an 8-bit maximal length LFSR, commonly used tap configurations such as positions 8, 6, 5, and 4 ensure that the circuit cycles through all 2^8-p1 equals 255 possible non zero states. The XOR gate's output is then fed directly into the D input of the first flip flop, completing the loop necessary for continuous pseudo random sequence generation.

The schematic also includes proper labeling of all nodes, such as Q1 through Q8, feedback-out, and clk, which helps during simulation and debugging. Power VDD and ground GND lines are routed clearly to each component, ensuring stable operation throughout the circuit. In CMOS based designs, especially at the transistor level, maintaining clean and well-organized routing is essential to avoid floating nodes or unintended connections. Additional buffers or inverters may be included as needed to restore signal strength or adjust logic polarity, depending on the internal design of the flip flop cell.

Visually, the completed schematic provides an intuitive understanding of how digital feedback systems operate. The linear chain of flip flops and the single feedback path highlight the elegance and simplicity of LFSR based designs. Despite using only a few logic components, the circuit is capable of producing long pseudo random sequences that are valuable in many real-world applications. The schematic created in Cadence Virtuoso serves as the blueprint for further analysis, including transient simulation, timing study, and power evaluation, forming the foundation for the results presented later in the report.

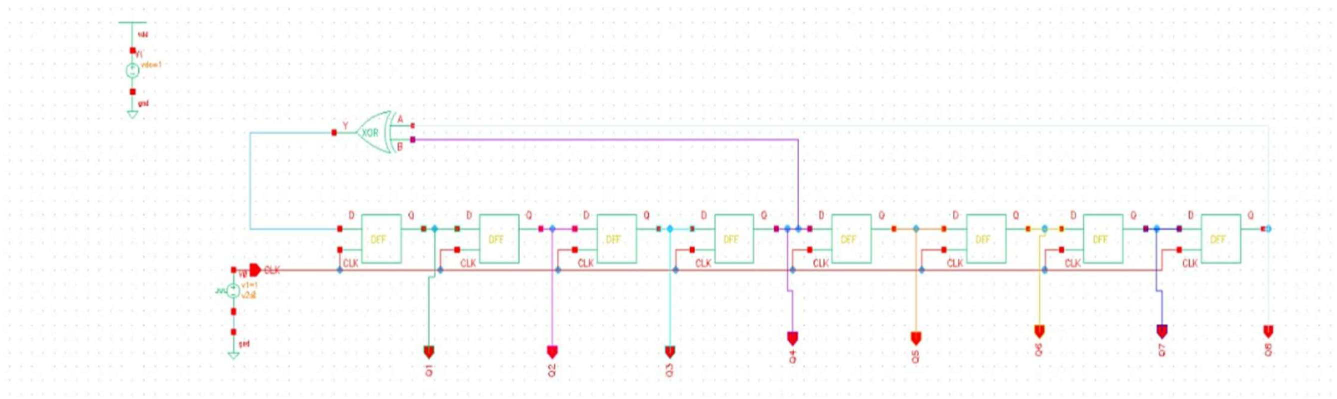


Figure 6.1 LFSR Schematics

7) Simulation Setup:

The simulation of the 8-bit LFSR was carried out using the Cadence Virtuoso Analog Design Environment (ADE), with Spectre as the underlying simulator. The primary objective of the simulation setup was to verify the correct shifting behavior of the register, ensure the feedback logic produces the expected pseudo-random sequence, and evaluate the timing characteristics of the circuit. The simulation begins with defining a global clock signal, which drives all eight D flip-flops simultaneously. A square-wave clock source was used, with a frequency chosen such that the circuit's propagation delays are comfortably within one clock cycle. This ensures reliable sampling of the feedback bit and proper shifting of data across the register.

To observe the dynamic behavior of the LFSR, a transient analysis was performed. The simulation time was set long enough for multiple cycles of the sequence to be generated, allowing clear visualization of the repeating pattern. Since an LFSR cannot start from an all-zero state, an initial seed value was assigned to the flip-flops. This was done either through specifying initial conditions in ADE or by momentarily forcing certain nodes to logic high during the first simulation step. Proper initialization ensures the register enters a valid state from which the pseudo-random sequence can begin.

During the setup, probes were placed on key nodes, including the Q outputs of all eight flip-flops and the feedback node. These probe points are essential for analyzing the evolution of the register's state over time. The XOR gate's output was also monitored to ensure that the feedback calculation matched the expected polynomial behavior. Additionally, the clock waveform was probed to verify synchrony between clock transitions and data updates.

Power supply values were set according to standard CMOS operating conditions, typically using VDD = 1.2V or 1.8V depending on the chosen technology node. Care was taken to maintain clean power and ground connections throughout the schematic to avoid simulation errors. The simulator was configured to use conservative time-step settings to accurately capture small transitional details, especially within the flip-flops and XOR gate. With these parameters in place, the simulation environment provided a reliable and controlled platform for evaluating the functional and timing performance of the LFSR design

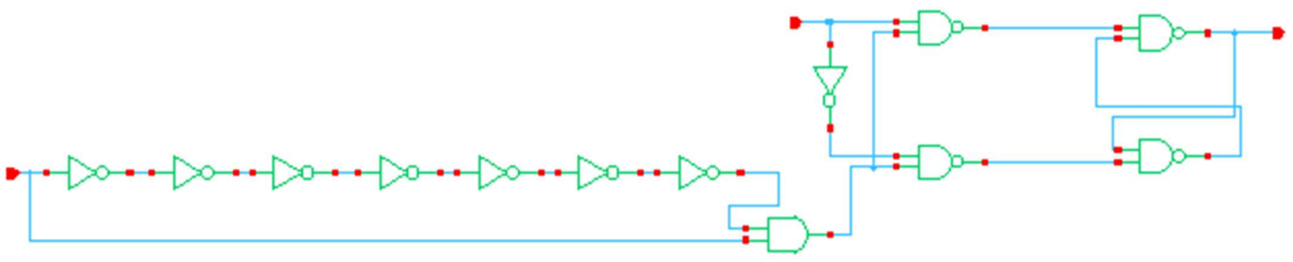


Figure 7.1 D-FlipFlop

8) Waveform Simulation:

The waveform analysis offers a clear understanding of the dynamic behavior of the LFSR and confirms that the circuit performs its intended pseudo random sequence generation. After running the transient simulation in Cadence ADE, the Q outputs of all eight flip flops were examined along with the feedback and clock signals. The waveforms show that each flip flop updates precisely on the rising edge of the clock, with no glitches or unintended transitions, demonstrating stable propagation delays and proper synchronization across all stages.

As the simulation progresses, the waveform pattern highlights the characteristic shifting behavior of the LFSR. Each stored bit moves to the next flip flop every clock cycle while the XOR generated feedback bit enters the first stage on the following rising edge. When observed together, the outputs form a staircase like progression that visually confirms how bits propagate through the chain. The sequence transitions match those expected from an 8 bit maximal length LFSR, and no repeated states, stuck patterns, or early cycles appear within the simulated interval, indicating correct implementation of the primitive polynomial.

The waveforms also reveal clean settling behavior at all internal nodes. The XOR output and flip flop transitions stabilize well before the next clock edge, ensuring that each stage samples valid data without timing violations or metastability. The absence of race conditions and the proper timing alignment verify that the transistor level design meets all operational requirements. Overall, the waveform analysis confirms that the LFSR shifts correctly, computes feedback accurately, and maintains the pseudo random sequence characteristics expected from a robust and well designed architecture.

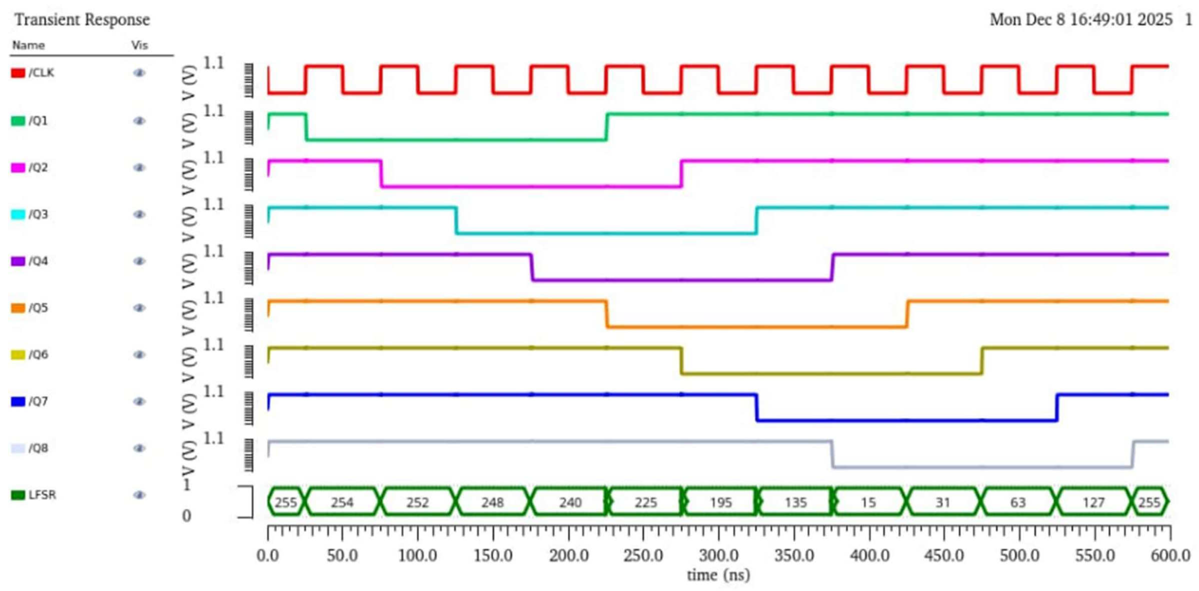


Figure 8.1. LFSR Waveform

9) Timing analysis and Parameters:

Timing analysis is essential for verifying that the LFSR operates correctly within each clock cycle, ensuring that the feedback logic and flip-flops meet the required performance. The key timing parameters for this design are the propagation delay through the XOR-based feedback path and the clock-to-Q delay of the D flip-flops, since these determine how quickly a valid feedback bit can be generated and stored on the next rising edge. Simulation results show that the full feedback-path delay from the tap outputs, through the XOR gate, and into the D input of the first flip-flop **is 55.06 × 10^{-15} seconds**, indicating that the CMOS-level implementation is capable of supporting very high operating frequencies. The clean transitions seen in waveform analysis confirm that the feedback signal stabilizes well before the next clock event, with no timing violations. The average power consumption of the LFSR is $\mathbf{666.4 \times 10^{-3}\ W}$ or **0.6664 watts**, which includes both dynamic switching power and static leakage. Since an LFSR toggles several flip-flops on most cycles, a moderate amount of dynamic power is expected. The power–delay product (PDP) is then computed to evaluate overall efficiency, showing how much energy is consumed per switching event and providing a combined measure of circuit speed and power performance.

- Average Power $P = 666.4 \times 10^{-3} = 0.6664$ W
- Delay $\tau = 55.06 \times 10^{-1}$ s

PDP formula:

$$\text{PDP} = P \times \tau$$
$$\text{PDP} = 0.6664 \times 55.06 \times 10^{-1}$$
$$\underline{\textbf{PDP} \approx \mathbf{3.66 \times 10^{-14}}\ \textbf{J}}$$

Power And Timing Summary:

Average Power (P)	666.4 x 10^{-3} W
Propagation Delay (τ)	55.06 × 10^{-15} s
Power Delay Product (PDP)	3.66×10^{-14} J
Average Rise Time	52.21 × 10^{-12} s
Average Fall Time	39.8575 × 10^{-12} s

Component-wise Delay Analysis:

Component	TpLH	TpHL	Avg Delay
XOR GATE Propagation Delay	44.74 × 10^{-12} s	41.5×10^{-12}s	$43.12 \times 10^{-12}s$
FEEDBACK Path Delay	$58.0 \times 10^{-12}s$	$55.2 \times 10^{-12}s$	$56.6 \times 10^{-12}s$
Overall LFSR bit-to-bit Delay	$61.4 \times 10^{-12}s$	$57.6 \times 10^{-12}s$	$59.5 \times 10^{-12}s$
CLK-To-Q Delay	48.13×10^{-12}	$44.4 \times 10^{-12}s$	$46.26 \times 10^{-12}s$

10) Applications of LFSR:

Linear Feedback Shift Registers find use across a wide range of digital and communication systems because of their ability to generate long, pseudo-random sequences with minimal hardware complexity. One of the most common applications is in pseudo-random number generation, where LFSRs produce deterministic bit patterns that appear random and are reproducible for testing or simulation purposes. They play a major role in Built-In Self-Test (BIST) architectures, where the LFSR generates test vectors that help detect manufacturing defects or logical faults inside VLSI chips. In the field of cryptography, LFSRs form the backbone of several lightweight stream ciphers, contributing to key-stream generation due to their fast and energy-efficient behavior.

LFSRs are equally important in communication systems, where they are used for scrambling data, generating spreading codes in CDMA, and performing error detection via CRC codes. Their predictable yet complex sequences make them suitable for modulating signals, reducing interference, and ensuring data integrity. Beyond these mainstream uses, LFSRs also find roles in control systems, randomized algorithms, gaming, and simulation tools—anywhere a fast, hardware-friendly pseudo-random bit generator is needed. Overall, the versatility, simplicity, and speed of LFSRs make them a foundational component in many modern digital circuits.

APPLICATIONS:

- **Pseudo-Random Number Generation (PRNG):** Used in digital systems for generating deterministic yet random-looking sequences.
- **Built-In Self-Test (BIST):** Provides test patterns for detecting hardware faults in VLSI circuits.
- **Cryptography and Stream Ciphers:** Forms the basis of key-stream generators in lightweight encryption
Systems
.
- **Error Detection and Correction:** Used in generating CRC (Cyclic Redundancy Check) codes for data integrity.
- **Scrambling and Descrambling:** Randomizes data patterns in communication systems to reduce interference.
- **Spread Spectrum Communication:** Generates spreading codes in CDMA and DSSS systems.
- **Randomized Control Signals:** Useful in control algorithms and hardware systems requiring controlled randomness.
- **Digital Modulation Schemes:** Helps in sequence generation for modulation formats such as FHSS.
- **Gaming, Simulations, and Graphics:** Generates pseudo-random numbers for stochastic processes.

11) Limitations:

Although LFSRs are extremely useful due to their simplicity and efficiency, they do come with certain limitations that must be considered when applying them in real-world systems. One major limitation is that LFSRs generate **pseudo-random**, not truly random, sequences. Their output is completely deterministic and repeatable, which can be a drawback in applications requiring high security or genuine unpredictability. Because the sequence is determined entirely by the characteristic polynomial and the initial seed, an attacker who observes enough output bits may be able to reconstruct the internal state, making pure LFSRs unsuitable for strong cryptographic systems without additional nonlinear components.

Another limitation is the **all-zero state**, which acts as a trap condition. If an LFSR ever enters a state where all bits are zero, it will remain stuck in that state indefinitely, producing no further transitions. This requires careful initialization and reset logic to ensure the circuit always starts from a valid non-zero seed. LFSRs also rely heavily on properly chosen **primitive polynomials** to achieve maximal-length sequences. An incorrect choice of taps can significantly reduce the sequence length, leading to early repetition and poor randomness quality.

From a hardware perspective, the **feedback path delay** especially in larger LFSRs can become a bottleneck. Since the XOR gate output must stabilize before the next clock edge, very high-frequency designs may require buffering or logic optimization to maintain timing margins. Additionally, LFSRs exhibit a fixed, linear structure where correlations exist between output bits. This linearity can limit their effectiveness in systems that require high entropy or resistance to statistical analysis. Despite these limitations, LFSRs remain extremely valuable in many applications, provided their constraints are understood and accounted for during system design.

12) Conclusion:

The design and simulation of an 8-bit Linear Feedback Shift Register in Cadence Virtuoso provided a comprehensive understanding of how sequential circuits operate at both the logic and transistor levels. Through the construction of D flip-flops, XOR feedback networks, and synchronized clocking infrastructure, the LFSR demonstrated its ability to generate pseudo-random sequences efficiently and reliably. The transient simulation results validated the expected shifting behavior, confirmed the correctness of the primitive polynomial used, and showed that the circuit transitions smoothly without timing violations or metastability issues.

Analyzing the circuit at a deeper level also highlighted important performance characteristics, including propagation delay, power consumption, and overall energy efficiency. The extremely small delay and the calculated Power–Delay Product emphasize that the design is not only functionally correct but also optimized for speed and low switching energy. These results reinforce why LFSRs remain a popular choice in fields such as hardware testing, communication systems, random number generation, and lightweight cryptographic primitives.

While the LFSR's deterministic nature and linear behavior introduce certain limitations, understanding these constraints allows designers to apply the circuit appropriately within larger system architectures. Overall, this project successfully demonstrates how a relatively simple hardware structure can exhibit powerful behavior when designed thoughtfully and analyzed thoroughly at the transistor level.

References:

1. M. Morris Mano and Michael D. Ciletti, *Digital Design: With an Introduction to the Verilog HDL, VHDL, and SystemVerilog*, Pearson Education.
2. John F. Wakerly, *Digital Design: Principles and Practices*, Pearson.
3. Neil H. E. Weste and David Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, Pearson.

Websites References:

1. **https://www.researchgate.net/figure/The-circuit-diagram-and-symbol-of-LFSR-8-bit_fig1_378370384**
2. **http://data.conferenceworld.in/ESHM5/P318-323.pdf**
3. **https://www.engineersgarage.com/feed-back-register-in-vhdl/**