Internship Report

On

Modelling and Simulation of Analog Systems and Digital Circuits Using EDA Tools

National Institute of Technology, Jamshedpur

For the Fulfilment of Internship for a duration of one and half month from 19th May to 5th July 2025

Submitted by:

Kandregula Siddartha

Roll no. = 418034

Indira Gandhi Institute of Technology, Sarang

Acknowledgement

I would like to express my sincere gratitude to the **Department of Electronics and Communication Engineering, NIT Jamshedpur**, for providing me with the opportunity to undertake this enriching internship.

I am especially thankful to Dr. Akhilesh Kumar Sir for his invaluable guidance, support, and insightful feedback throughout the internship, which significantly deepened my understanding of circuit design and simulation. His mentorship greatly contributed to the development of my technical skills and confidence. I would also like to sincerely thank my senior, Mr. Abhisekh Kumar, Ph.D. student, for his continuous support and helpful suggestions that enriched my learning experience.

I am also grateful to my home institute, **Indira Gandhi Institute of Technology, Sarang**, for encouraging and supporting me in pursuing this internship opportunity. Lastly, I would like to thank my family and friends for their continuous encouragement and moral support.

This experience has significantly deepened my practical knowledge of analog and digital circuit design and has been a crucial step in my journey as an aspiring electronics and communication engineer.

.

THANK YOU.

Date-

CANDIDATE'S DECLARATION

I hereby declare that

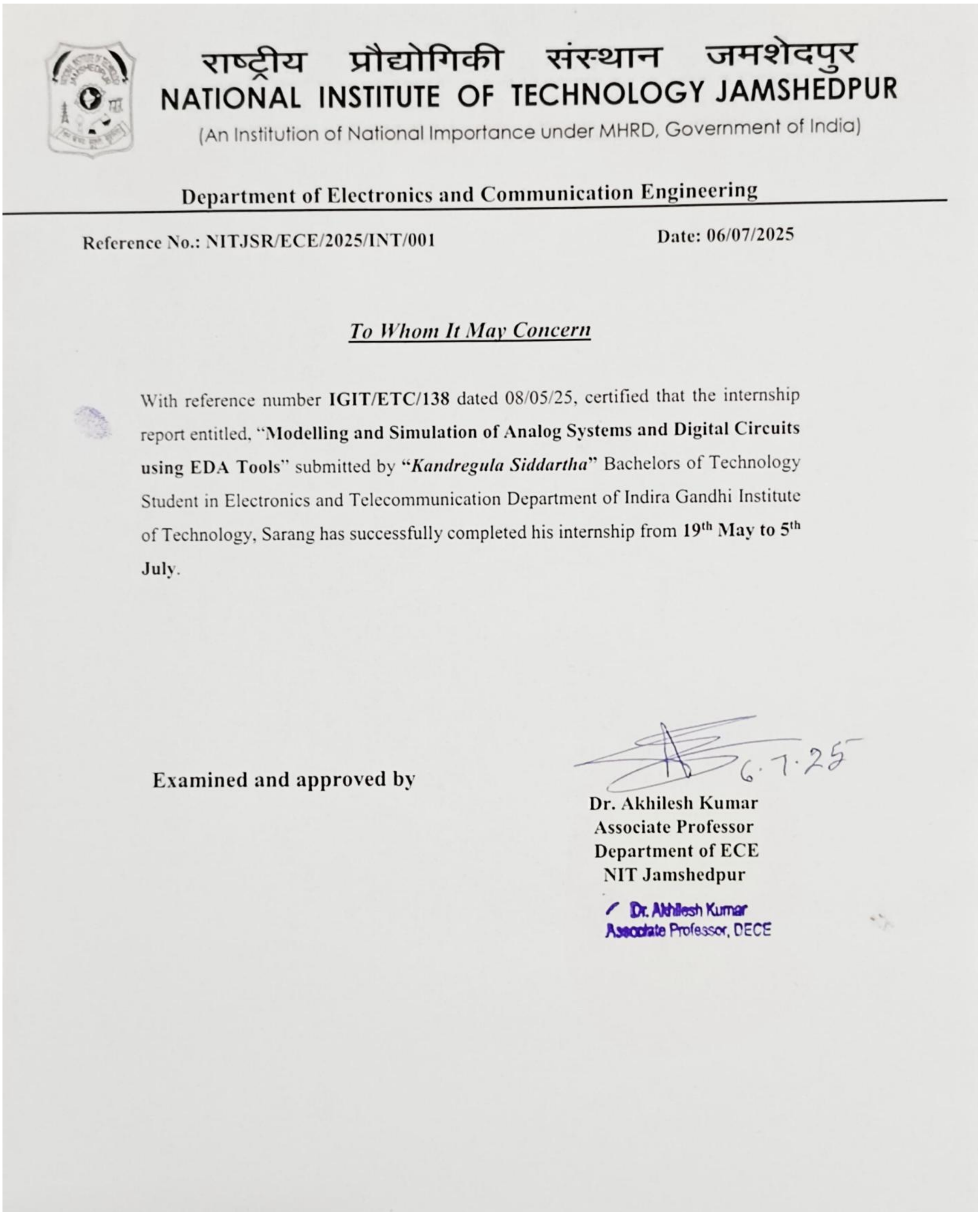
a. The work contained in this report is original and has been done by me under the guidance of my supervisor ***Dr Akhilesh Kumar,*** Associate professor, Department of Electronics and communication Engineering, NIT Jamshedpur.

b. I have followed all the guidelines provided by the Institute in preparing the report.

c. I have conformed to the norms and guideline given in the Ethical Code of Conduct of this Institute.

Signature of the Student

Kandregula Siddartha


राष्ट्रीय प्रौद्योगिकी संस्थान जमशेदपुर

NATIONAL INSTITUTE OF TECHNOLOGY JAMSHEDPUR

(An Institution of National Importance under MHRD, Government of India)

Department of Electronics and Communication Engineering

Reference No.: NITJSR/ECE/2025/INT/001 **Date: 06/07/2025**

To Whom It May Concern

With reference number **IGIT/ETC/138** dated 08/05/25, certified that the internship report entitled, "**Modelling and Simulation of Analog Systems and Digital Circuits using EDA Tools**" submitted by "***Kandregula Siddartha***" Bachelors of Technology Student in Electronics and Telecommunication Department of Indira Gandhi Institute of Technology, Sarang has successfully completed his internship from **19th May to 5th July**.

Examined and approved by

[illegible] 6.7.25

Dr. Akhilesh Kumar
Associate Professor
Department of ECE
NIT Jamshedpur

Dr. Akhilesh Kumar
Associate Professor, DECE

Contents

1. Introduction

In today’s era of rapid technological evolution, practical exposure to industry-grade electronic design tools is crucial for bridging the gap between theoretical knowledge and real-world application. As part of my academic and professional development, I undertook a technical internship/project that provided a valuable opportunity to work extensively with PSPICE, Vivado, and Cadence Virtuoso—each playing a distinct yet complementary role in the field of electronics and system design.

PSPICE enabled me to explore and simulate a wide range of analog and digital circuits. I designed, tested, and analyzed circuit behavior under various conditions, allowing for a deeper understanding of fundamental concepts such as transient response, frequency response, logic gate operation, and counter design. This environment helped solidify my grasp on core electronics topics by providing immediate visual and data-driven feedback from the simulated circuits.

Moving toward digital system design, Vivado Design Suite allowed me to develop and implement HDL-based designs on FPGA platforms. I worked with Verilog to describe digital logic circuits, performed simulation and synthesis, and verified designs using testbenches. This workflow introduced me to concepts such as hardware constraints, timing analysis, and modular design practices that are vital in modern embedded systems and VLSI development.

To complete the analog-mixed signal design experience, I used Cadence Virtuoso, a powerful tool for designing and simulating transistor-level analog circuits. Here, I engaged in schematic capture, layout creation, and simulation tasks, focusing on accurate modeling of real-world components. I also gained exposure to techniques like DC analysis, AC analysis, transient simulations, and layout vs. schematic (LVS) checks, which are essential in custom IC design.

This report documents the various tasks and projects I completed using these platforms, the challenges encountered, and the skills acquired throughout the experience. It reflects not only my technical progress but also my growing interest and competence in electronic design automation (EDA), which forms the backbone of innovation in semiconductor industries today.

2. Pspice

PSPICE (Personal Simulation Program with Integrated Circuit Emphasis) is a powerful simulation tool used for analyzing both analog and digital circuits. During my internship/project, I used PSPICE to design, simulate, and test a variety of electronic circuits, ranging from basic components to more complex systems. The tool enabled me to understand real-time circuit behavior by providing graphical outputs for voltage, current, and power waveforms. I worked on projects including diode and transistor characteristics, rectifier circuits, operational amplifier configurations, logic gate implementations, and sequential circuits like counters and flip-flops. PSPICE’s ability to perform transient, DC, and AC analysis allowed me to visualize how circuits respond to different inputs and frequencies. This hands-on experience greatly enhanced my understanding of circuit theory and reinforced key concepts such as signal propagation, timing, and feedback mechanisms in electronics. Below given are the simulations I performed using pspice.

2.1 Op-Amp

An **Operational Amplifier (Op-Amp)** is a versatile and widely used electronic device in analog circuit design. It functions as a high-gain voltage amplifier with two input terminals **inverting (−)** and

non-inverting (+)—and a single output terminal. The output voltage of an op-amp is proportional to the difference between the voltages applied to the two input terminals. Op-amps are powered by a dual supply voltage, typically denoted as +V and −V, which allows the output to swing both positively and negatively. In ideal conditions, op-amps are characterized by **infinite open-loop gain**, **infinite input impedance**, **zero output impedance**, **infinite bandwidth**, and **zero input offset voltage**, although practical op-amps have finite and well-defined parameters. These characteristics enable op-amps to perform a wide variety of linear and nonlinear functions such as amplification, integration, differentiation, filtering, and signal conditioning. The flexibility of op-amps lies in the use of external feedback components (resistors, capacitors) that define their behaviour in closed-loop configurations. Common circuits include **inverting amplifiers**, **non-inverting amplifiers**, **voltage followers**, **summing amplifiers**, **differential amplifiers**, and **active filters**. Due to their high precision, low cost, and ease of implementation, op-amps are foundational components in analog electronics, instrumentation systems, and signal processing applications.

2.1.1 Inverting Amplifier

In this design, an **inverting amplifier circuit** was implemented using an Op-Amp subcircuit in **PSPICE**. The inverting configuration is a widely used operational amplifier topology in which the input signal is applied through a resistor to the inverting terminal of the op-amp, while the non-inverting terminal is grounded. A feedback resistor connects the output to the inverting input, creating a closed-loop system that provides controlled gain. The theoretical gain of the inverting amplifier is given by the formula:

$$\text{Gain} = -\, Rf / Rin$$

where Rf is the feedback resistor and Rin is the input resistor. In PSPICE, the standard OPAMP model was used to simulate the circuit. By setting appropriate resistor values gain was achieved, which was verified by running a transient analysis and observing the phase-inverted amplified output waveform. The PSPICE netlist allowed for flexible customization of the op-amp parameters and circuit behaviour, thereby providing accurate simulation results that aligned well with theoretical expectations. Below is the given pspice code used for simulation:-

```
Inverting Amplifier

R1 1 2 1K

R2 2 6 10K

VIN 1 0 SIN (0 1M 1K)

VP 7 0 DC 12V

VN 4 0 DC -12V

X 0 2 7 4 6 UA741

.LIB NOM.LIB

.TRAN 0 5MS

.PROBE

.END
```

The simulation of the inverting op-amp circuit was carried out using PSPICE, and the output waveform as shown in ***Figure 1*** was observed using the transient analysis tool. A sinusoidal input signal of 1 V peak amplitude at a frequency of 1 kHz was applied through the input resistor to the inverting terminal of the op-amp. The simulation output clearly demonstrated the expected behavior of the inverting amplifier: the output signal was a sinusoidal waveform of the same frequency but with an amplitude multiplied by the gain factor and a phase shift of 180°, indicating signal inversion.

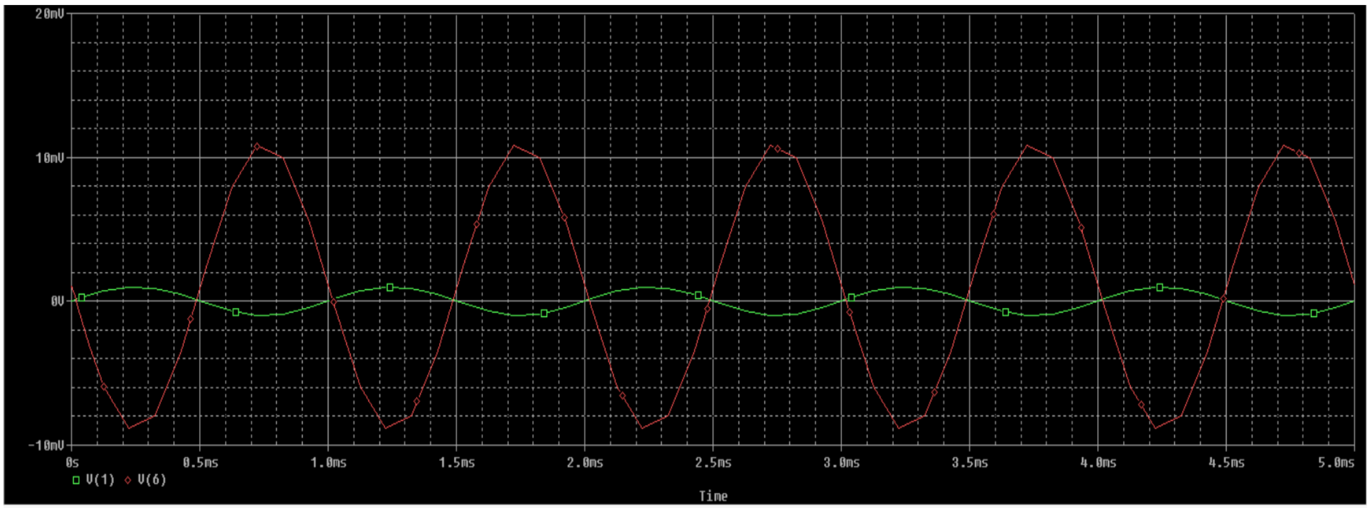


Figure 1 *Transient response of inverting amplifier*

2.1.2 Non Inverting Amplifier

The **non-inverting amplifier** is another fundamental configuration using an operational amplifier, where the input signal is applied directly to the **non-inverting terminal (+)** of the op-amp. The inverting terminal (−) is connected to a voltage divider formed by two resistors: one between the output and the inverting input (feedback resistor Rf) and another between the inverting input and ground . This configuration provides **positive voltage gain** without inverting the signal, making it ideal for applications where signal polarity must be preserved. The theoretical gain of a non-inverting amplifier is given by the formula:

$$\text{Gain}=1+Rf/Rin$$

This means the gain is always greater than or equal to 1. In PSPICE, this circuit was implemented using a built-in or custom-defined op-amp subcircuit (such as uA741). Appropriate resistor values were chosen to set the desired gain. The circuit was simulated in PSPICE using transient analysis and DC sweep techniques to study time-domain behaviour and gain characteristics. Below is the given pspice code used for simulation:-

```
Non-Inverting Amplifier

R1 0 2 1K

R2 2 6 10K

*VIN 3 0 DC 1V

*VIN 3 0 AC 1V

VIN 3 0 SIN (0 1M 1K)

VP 7 0 DC 12V

VN 4 0 DC -12V
```

```
X 3 2 7 4 6 UA741

.LIB NOM.LIB

*.DC VIN -12 12 1V

*.AC DEC 100 1 100K

.TRAN 0 5MS

.PROBE

.END
```

The transient analysis is showed in ***Figure 2*** the output signal was a faithful reproduction of the input with a tenfold increase in amplitude and no phase inversion. A 1 V peak sine wave at 1 kHz applied at the input resulted in a 10 V peak output, validating the expected gain. The simulation also showed excellent linearity, confirming proper biasing and ideal op-amp behavior in the closed-loop configuration.

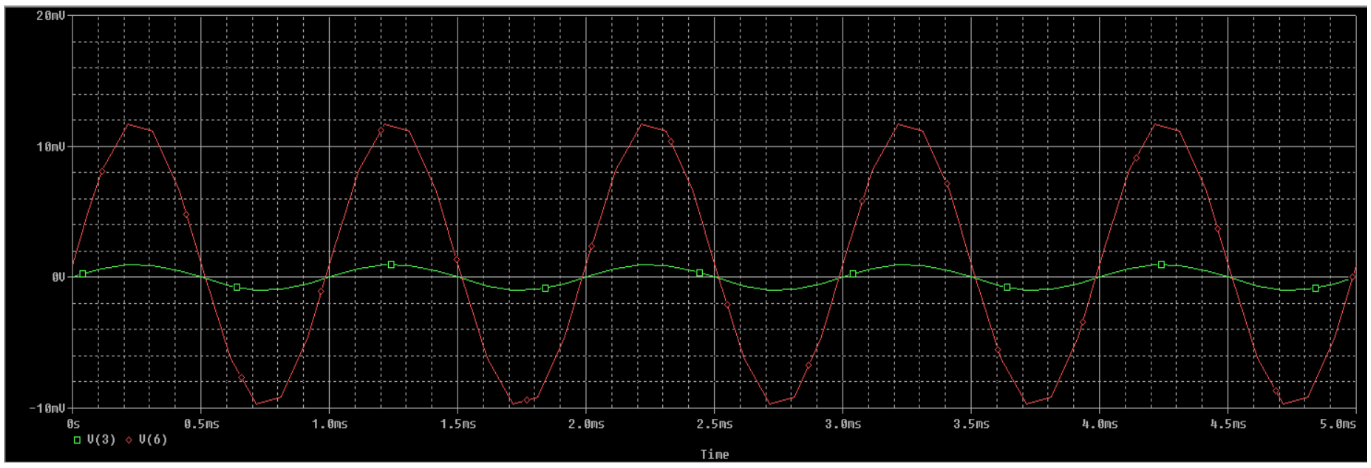


Figure 2 *Transient Response of Non Inverting Amplifier*

2.1.3 Integrator

An **Op-Amp Integrator** is an important analog circuit configuration that performs the mathematical operation of integration, converting an input voltage signal into an output voltage that represents the time integral of the input. In its most basic form, the integrator circuit uses a resistor (R) connected between the input signal and the **inverting terminal** of the op-amp, and a capacitor (C) connected between the **inverting terminal** and the **output**, forming the feedback path. The **non-inverting terminal** is connected to ground. The integrator exploits the fact that the current through a capacitor is proportional to the rate of change of voltage across it, and the op-amp maintains a virtual ground at the inverting input. The output voltage Vout(t) is given by the expression:

$$Vout(t)=-1/RC\int Vin(t)\, dt$$

In the PSPICE design Appropriate resistor and capacitor values were chosen to set the desired Output. The circuit was simulated in PSPICE using transient analysis to study time-domain behaviour. Below is the given pspice code used for simulation:-

```
Integrator

R1 1 2 2.5K

C1 2 6 10U
```

```
VIN 1 0 PULSE(2 -2 0 1NS 1NS 0.05S 0.1S)

VP 7 0 DC 12V

VN 4 0 DC -12V

X 0 2 7 4 6 UA741

.LIB NOM.LIB

.TRAN 1MS 1S

.PROBE

.END
```

The simulation shown in ***Figure 3*** revealed a triangular waveform at the output when a square wave was used as input, consistent with ideal integrator behavior. The output showed smooth linear ramps, demonstrating accurate integration. The results confirmed the circuit's time-domain performance and showed no distortion, indicating minimal phase error and reliable component modeling.

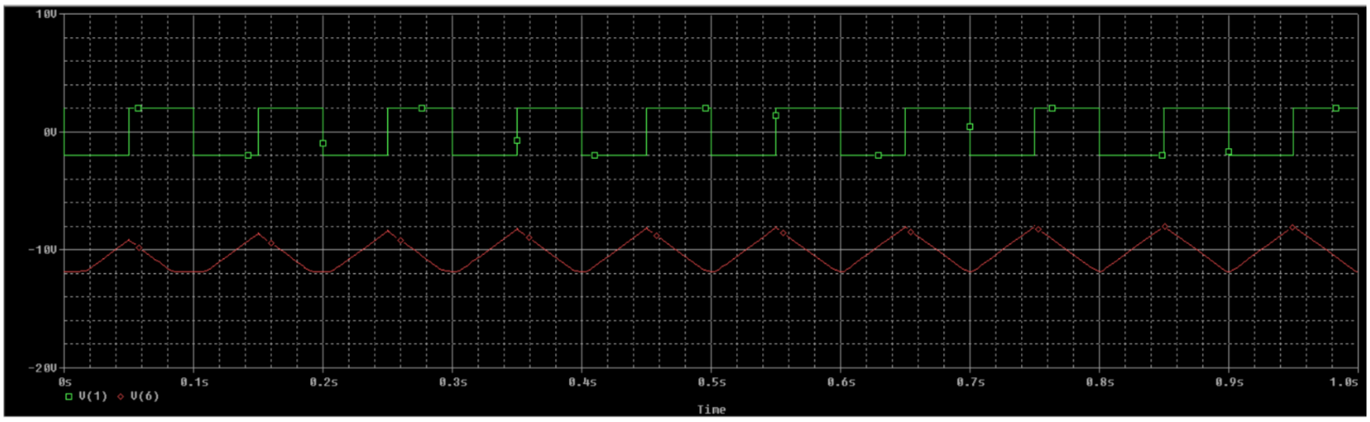


Figure 3 *Transient response of Integrator*

2.1.4 Differentiator

An **Op-Amp Differentiator** is an analog circuit configuration that performs the mathematical differentiation of the input voltage with respect to time. In this circuit, the **input signal** is applied through a **capacitor** to the **inverting terminal** of the op-amp, while a **resistor** is placed in the feedback path from the output to the inverting input. The **non-inverting terminal** is grounded. This configuration causes the output voltage to be proportional to the **time derivative** of the input signal. The theoretical output voltage Vout(t) of the differentiator is given by:

$$\text{Vout}(t) = -RC\cdot d\text{Vin}(t)/dtV$$

where R is the feedback resistor and C is the input capacitor. This makes the differentiator particularly useful in applications like edge detection, high-pass filtering, and waveform shaping. However, ideal differentiators are prone to noise amplification due to their high gain at high frequencies. To address this, **practical differentiators** include a small resistor in series with the capacitor or a capacitor in parallel with the feedback resistor to limit the bandwidth and improve stability.

The differentiator circuit in ***Figure 4(a)*** was designed in PSPICE using a TL084 op-amp. A 4.7 nF capacitor (C1) is placed in series with the input signal and connected to the inverting terminal through a 3.4 kΩ resistor

(R2) to improve noise stability. The feedback path consists of a 34 kΩ resistor (R1) and a 0.47 nF capacitor (C2) in parallel, which limits high-frequency gain and enhances stability. The non-inverting input is grounded, and the op-amp is powered by ±15 V supplies. This configuration ensures the output is proportional to the time derivative of the input voltage, as defined by the differentiator function.

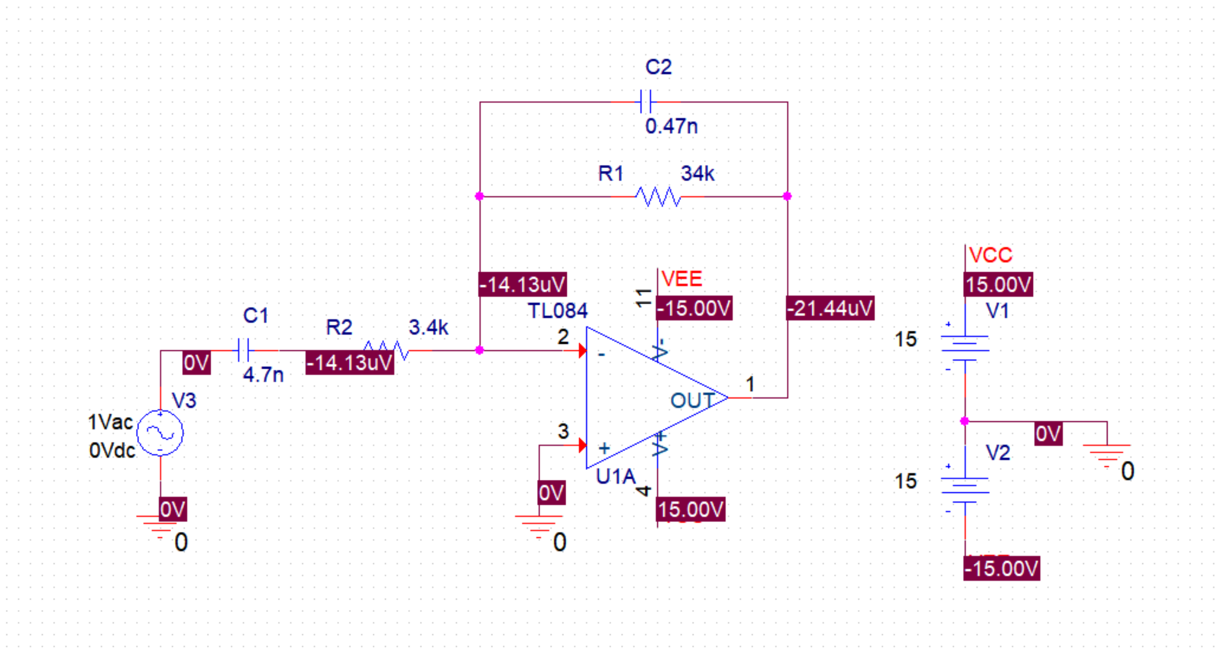


Figure 4(a) *Schematic of Differentiator using Op-Amp*

Transient simulation shown in ***Figure 4(b)*** with a sinusoidal input showed a phase-shifted, cosine-like output, confirming the differentiating behaviour. The output waveform was smooth and noise-free, indicating effective high-frequency control from the feedback network. Voltage values at key nodes supported expected theoretical operation, with clean differentiation observed. The circuit accurately demonstrated the core functionality of a practical op-amp differentiator in PSPICE.

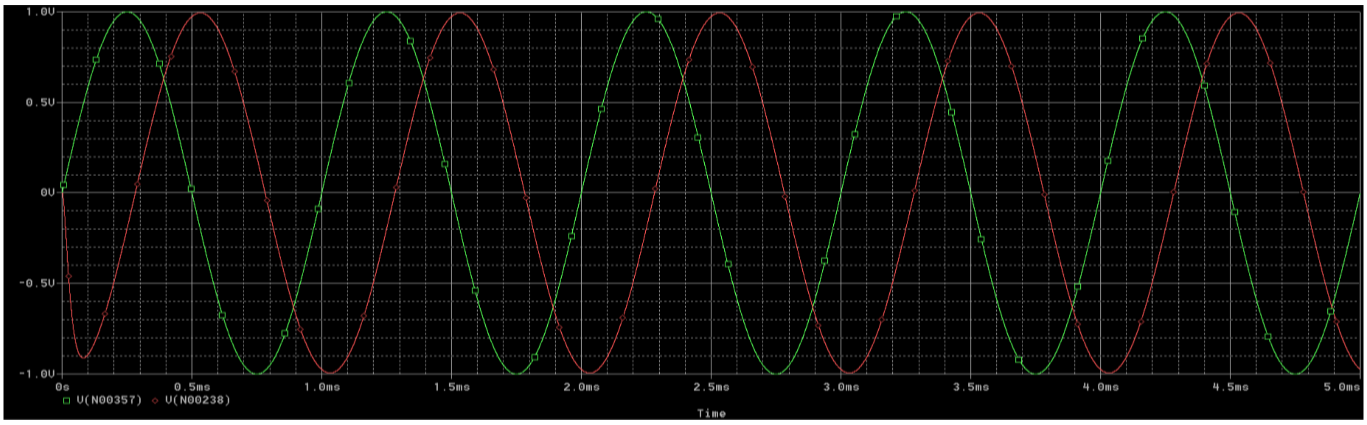


Figure 4(b) *Transient Response of Differentiator*

2.1.5 Schmitt Trigger

A **Schmitt Trigger** is a comparator circuit with **hysteresis**, meaning it uses two different threshold voltages to switch its output: one for rising input and another for falling input. This prevents false triggering due to noise or slow signal transitions. When the input exceeds the **upper threshold**, the output switches high; when it drops below the **lower threshold**, the output switches low. The gap between these two thresholds improves **noise**

immunity and provides **stable digital output** from noisy or analog signals. Schmitt Triggers are commonly used in **signal conditioning**, **debouncing switches**, and **wave shaping** circuits.

The schematic shown in ***Figure 5(a)*** shows a **non-inverting Schmitt Trigger** using a **TL084 op-amp**. A sinusoidal input signal (8 V, 1 kHz) is applied to the **inverting input**, while the **non-inverting input** receives feedback from a voltage divider formed by **R1 = 15 kΩ** and **R2 = 10 kΩ**. This creates **positive feedback**, introducing **hysteresis**—two distinct threshold voltages for switching the output high or low. The op-amp is powered by ±15 V, allowing the output to swing rail-to-rail. The resistor values define the upper and lower threshold voltages, making the circuit stable and resistant to noise

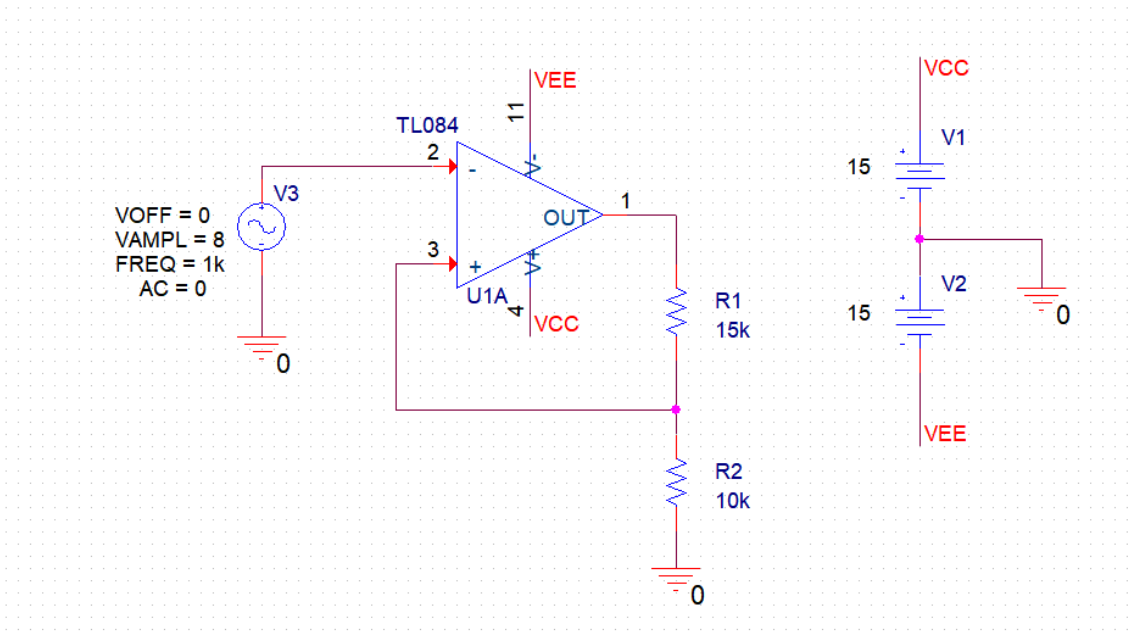


Figure 5(a) *Schematic of Schmitt Trigger using Op-Amp*

Simulation results in ***Figure 5(b)*** show a **clean square wave output** that toggles only when the input crosses defined thresholds, not at every zero crossing. The output transitions to +15 V and −15 V based on the hysteresis window set by the resistor network. This confirms proper Schmitt Trigger behavior, providing a reliable way to convert analog signals into noise-immune digital levels.

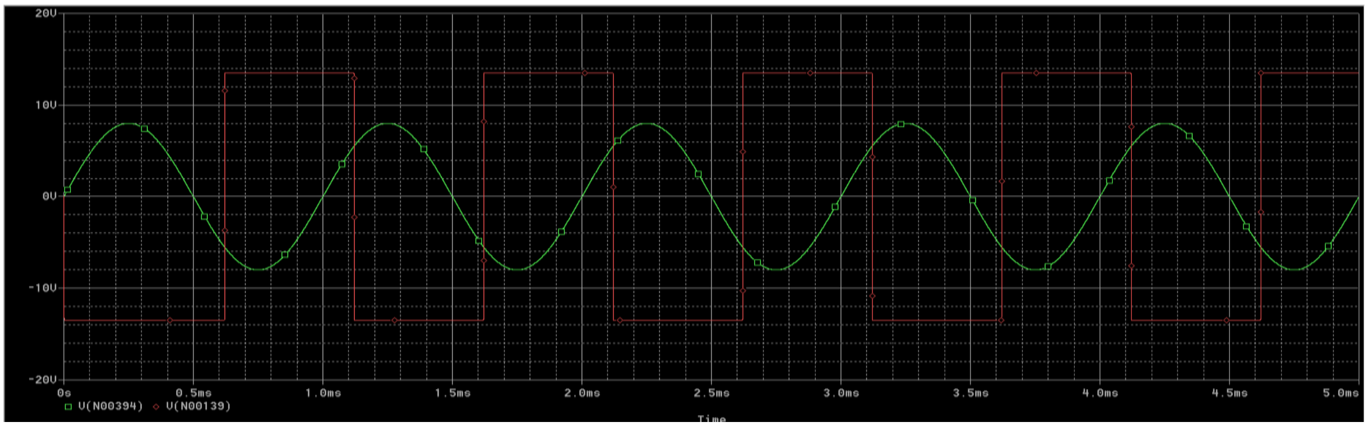


Figure 5(b) *Transient Response of Schmitt Trigger*

2.1.6 Full wave Rectifier

A **Full-Wave Rectifier (FWR)** is a circuit that converts both the positive and negative halves of an AC input signal into a pulsating DC output. Unlike half-wave rectifiers that only utilize one half of the waveform, full-wave rectifiers make use of both cycles, resulting in higher efficiency and smoother output. Traditional FWRs are built using a bridge of diodes, but op-amp-based FWR circuits are capable of **precision rectification**, allowing rectification of even very low amplitude signals without the 0.7 V diode voltage drop. Such circuits are widely used in signal processing, AC voltmeters, demodulation, and audio electronics where accurate rectification is critical. The given schematic shown in ***Figure 6(a)*** implements a **precision full-wave rectifier** using **two stages of op-amps (TL084)** and **diodes (D1, D2: 1N4148)**. The input is a sinusoidal signal (**2 V peak**, **1 kHz**) applied through resistor **R1 (1 kΩ)** to the **inverting input of op-amp U1A**. The resistors **R2 and R3 (1 kΩ each)** configure U1A as an **inverting amplifier**, while diodes D1 and D2 direct current flow depending on the polarity of the input. During the **positive half-cycle**, D1 conducts and routes the signal through **R4 and R5 (1 kΩ)** into op-amp **U1B**, which is configured as a **summing amplifier**. During the **negative half-cycle**, D2 conducts and a similar path is followed, ensuring that both halves of the input waveform are processed and flipped to positive polarity.

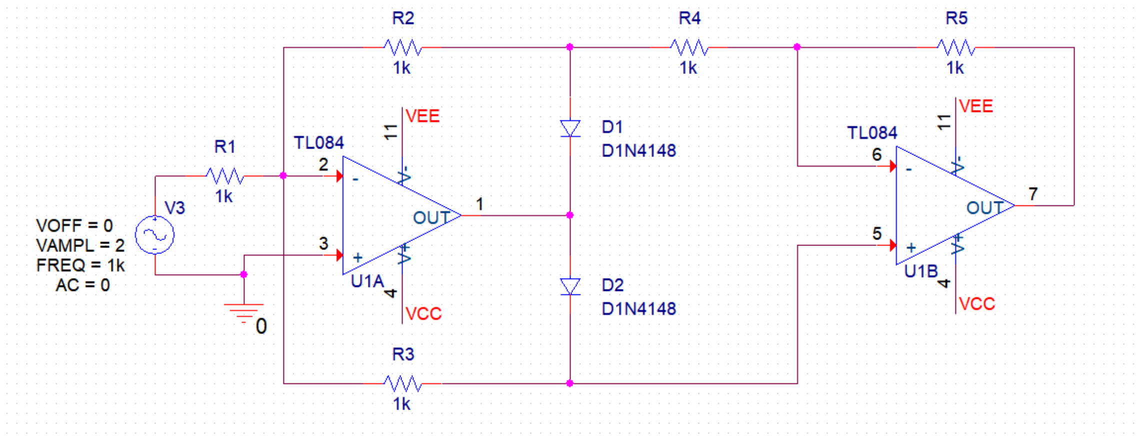


Figure 6(a) *Schematic of Full Wave Rectifier using Op-Amp*

The transient simulation shown in ***Figure 6(b)*** of this circuit shows a **full-wave rectified output** at the final op-amp output (U1B, pin 7). Regardless of the polarity of the input sinusoidal signal, the output remains **entirely positive**, indicating successful rectification of both half-cycles. The waveform shows clean and symmetrical rectified peaks corresponding to the input frequency (1 kHz), with minimal distortion, thanks to the high-speed switching of diodes and precision of the op-amp configuration. This design overcomes the limitations of diode-only rectifiers and demonstrates high accuracy, making it suitable for low-voltage or precision AC measurement applications.

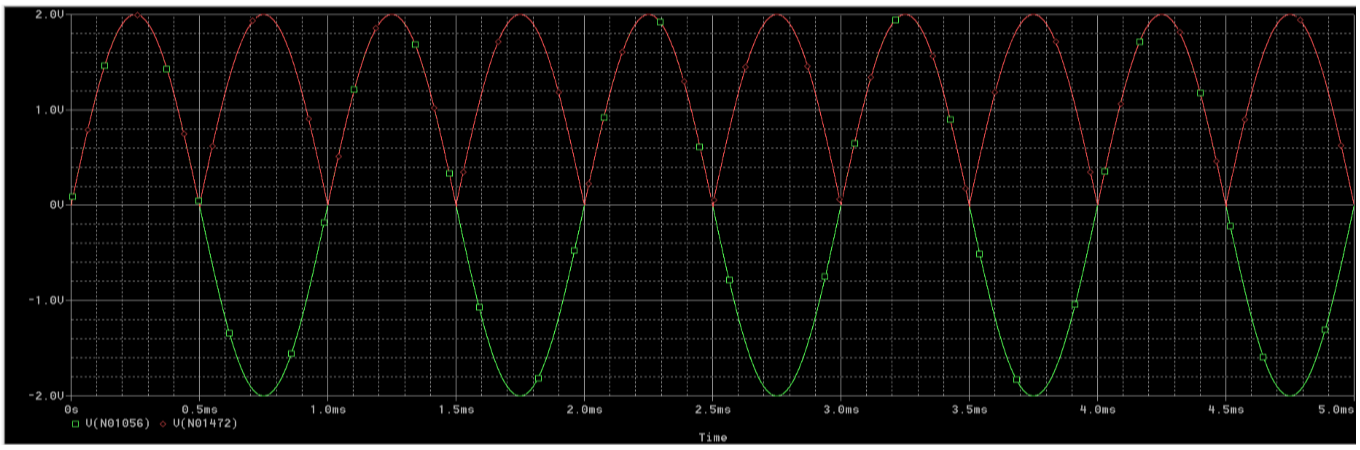


Figure 6(b) *Transient Response of Full Wave Rectifier*

2.2 Flipflop

A **flip-flop** is a fundamental digital memory circuit used to **store one bit of binary data (0 or 1)**. It has two stable states and can change its output based on control inputs, making it a **bistable multivibrator**. Flip-flops are the building blocks of **sequential logic circuits** and are widely used in registers, counters, memory units, and digital systems requiring storage and timing operations.There are several types of flip-flops, each with different triggering and control mechanisms:

- **SR (Set-Reset) Flip-Flop**: The basic type, using two inputs to set or reset the output.
- **D (Data or Delay) Flip-Flop**: Stores the value of the input when a clock pulse arrives; widely used in shift registers and data storage.
- **JK Flip-Flop**: A universal flip-flop that solves the invalid state problem of SR and allows toggling.
- **T (Toggle) Flip-Flop**: Changes state (toggles) with every clock pulse if the input is high; commonly used in counters.

Flip-flops are typically triggered by a **clock signal**, and can be **edge-triggered** (changing output on rising or falling clock edges) or **level-triggered** (responding to high or low levels). They are essential for designing circuits that require **timing, synchronization**, or **state memory** in digital electronics.

2.2.1 Synchronous 4 bit counter

A **synchronous 4-bit counter** is a digital sequential circuit that counts in binary from 0 to 15 (i.e., 24−12^4 - 124−1) in a synchronized manner using a common **clock signal**. In this design, all the flip-flops receive the clock **simultaneously**, ensuring that all bits in the counter change state **in sync** with the clock pulses. This synchronous nature eliminates the cumulative delay seen in asynchronous (ripple) counters, making synchronous counters faster and more reliable for high-speed operations.

Each flip-flop in the 4-bit counter represents one bit of the binary count (Q3 Q2 Q1 Q0), where Q0 is the least significant bit (LSB) and Q3 is the most significant bit (MSB). The output progresses in binary sequence (0000, 0001, 0010, ..., 1111) with each clock pulse.

Typically, **JK or T flip-flops** are used to build the counter, with their inputs configured to toggle at the appropriate conditions:

- Q0 toggles with every clock pulse.
- Q1 toggles when Q0 = 1.
- Q2 toggles when Q0 and Q1 = 1.
- Q3 toggles when Q0, Q1, and Q2 = 1.

This logical arrangement ensures that each bit toggles at **half the frequency** of the previous one, maintaining the correct binary count sequence.The **synchronous 4-bit counter** schematic in PSPICE shown in ***Figure 7(a)*** is built using **four JK flip-flops**, such as the standard **7476** ICs. In a synchronous counter, all flip-flops share the **same clock input**, ensuring that all state transitions occur simultaneously on each clock edge. These control signals can be generated using **AND gates** or by combining logic using PSPICE's built-in logic primitives or digital components. The flip-flop outputs (Q0 to Q3) represent the 4-bit binary count. A **clock pulse generator** is used to feed the clock input at a suitable frequency and logic probes or digital oscilloscopes in PSPICE are connected to observe the outputs.

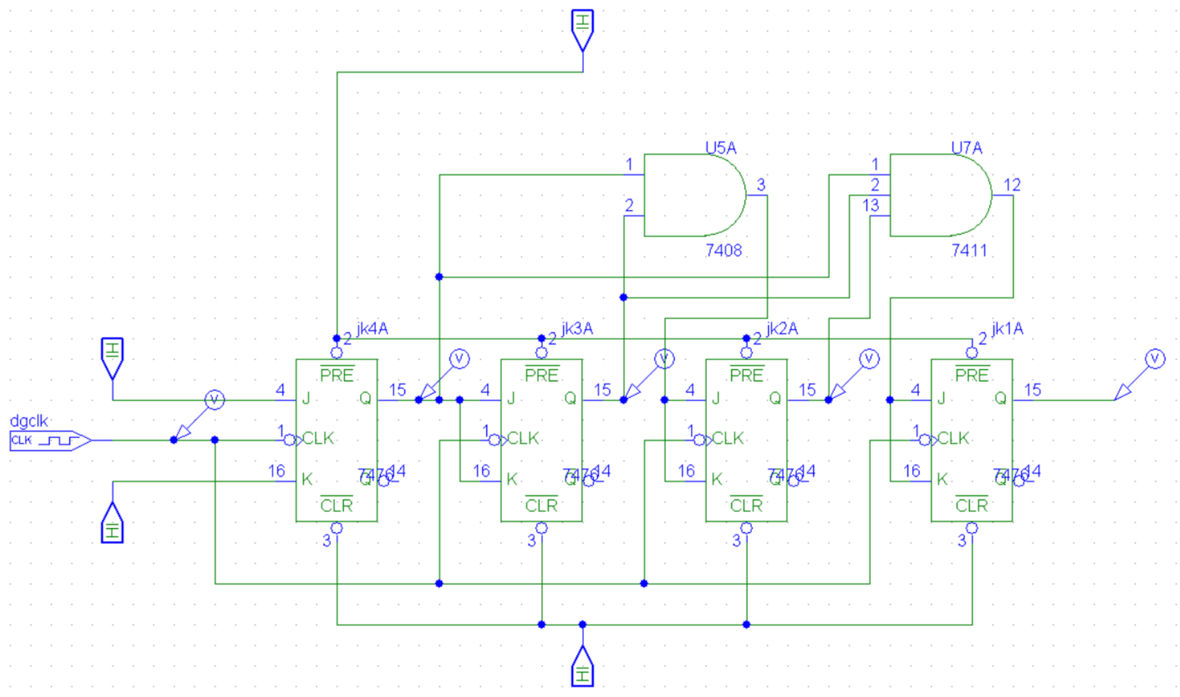


Figure 7(a) *Schematic of Synchronous 4 Bit Counter*

Upon running a **transient simulation** in PSPICE, the **output waveforms (Q0 to Q3)** show a clear binary counting sequence from 0000 to 1111 (0 to 15) with each clock pulse. The key characteristics observed include:

- **Q0** toggles on every clock pulse (highest frequency).
- **Q1** toggles on every 2nd clock pulse.
- **Q2** toggles on every 4th pulse.
- **Q3** toggles on every 8th pulse (lowest frequency).

This results in a **binary up-counter**, with each output bit changing state synchronously. The output waveform resembles a group of square waves, each with half the frequency of the previous one, confirming correct operation.

The simulation shown in ***Figure 7(b)*** validates the behaviour of the synchronous counter: minimal propagation delay, synchronized state transitions, and stable counting. These advantages make it suitable for high-speed counting applications.

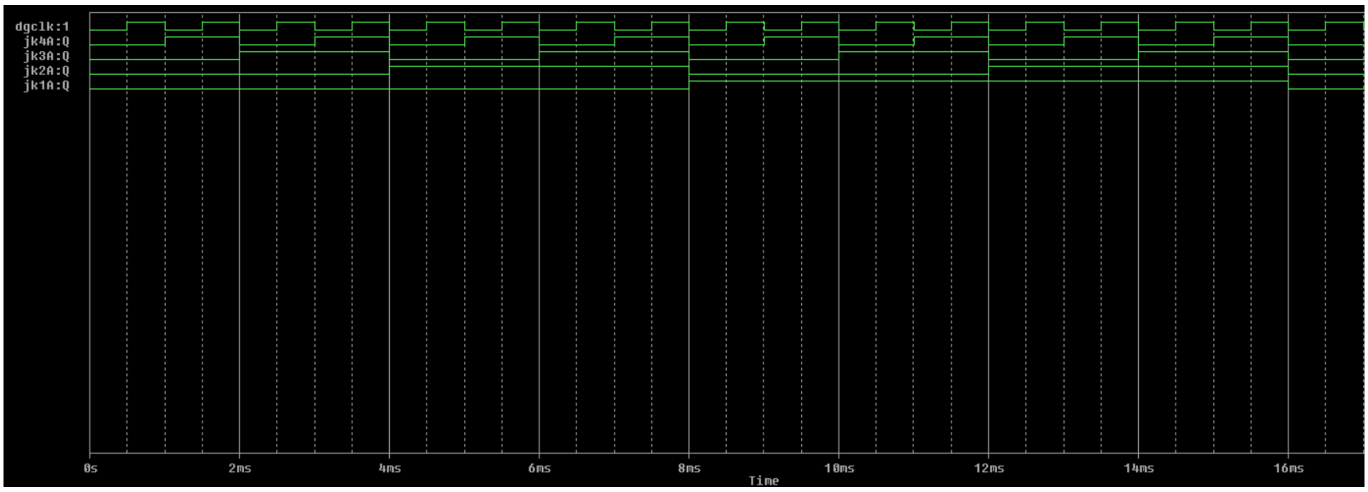


Figure 7(b) *Transient Response of Synchronous 4 Bit Counter*

2.2.2 Asynchronous 4 bit counter

An **asynchronous 4-bit counter**, also known as a **ripple counter**, is a sequential digital circuit that counts in binary from 0 to 15 using four flip-flops. The term "asynchronous" means that **not all flip-flops receive the same clock signal**—only the first flip-flop is directly clocked, while the clock for each subsequent flip-flop is derived from the **output of the previous one**. As a result, changes in output ripple through the flip-flops, introducing **propagation delays** that make asynchronous counters slower compared to synchronous ones.

Each flip-flop toggles when its clock input receives a falling (or rising) edge, depending on the configuration:

- The first flip-flop (Q0) toggles with every clock pulse.
- The second (Q1) toggles when Q0 goes from high to low.
- The third (Q2) toggles when Q1 transitions, and so on.

This creates a **binary count sequence** that progresses from 0000 to 1111, i.e., 0 to 15 in decimal. Asynchronous counters are simple to design and require fewer connections, but due to ripple delays, they are not suitable for high-speed applications.

In PSPICE, an asynchronous 4-bit counter can be designed using **four T flip-flops** or **JK flip-flops** as shown in ***Figure 8(a)*** configured in toggle mode (J = K = 1). The key characteristic is how the clock is distributed:

- The first flip-flop receives the **external clock signal** .
- The second flip-flop's clock is connected to **Q0**, the output of the first flip-flop.
- The third receives **Q1** as its clock, and the fourth receives **Q2**.

This clock cascading results in the ripple effect. PSPICE digital components (such as 7473 or 7476 IC models) can be used, or logic gates can be used to construct T or JK flip-flops. Output probes are connected to each flip-flop output (Q0–Q3) to monitor the counter state.

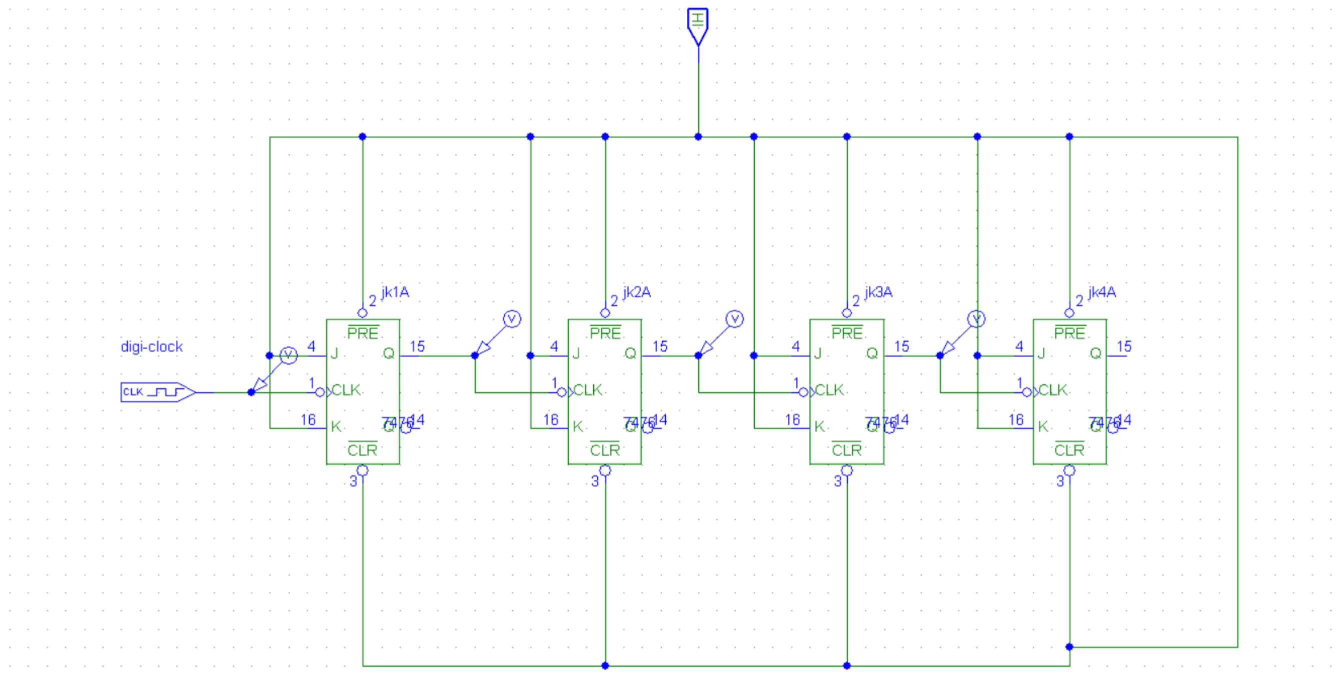


Figure 8(a) *Schematic of Asynchronous 4 Bit Counter*

When a transient analysis is run in PSPICE, the output waveforms show:

- Q0 toggling at the clock frequency.
- Q1 toggling at half of Q0's frequency.
- Q2 toggling at half of Q1's frequency.
- Q3 toggling at half of Q2's frequency.

The counter progresses through 16 binary states (0000 to 1111). However, due to **propagation delay**, the transitions at Q1, Q2, and Q3 are slightly delayed compared to Q0, creating a **ripple effect**. This delay accumulates with each stage and is evident in the timing diagram. Despite this, the asynchronous counter works correctly and demonstrates the fundamental principles of binary counting.

The simulation shown in ***Figure 8(b)*** confirms the simplicity and functionality of the asynchronous counter, though it also highlights its limitations for high-speed or time-critical applications.

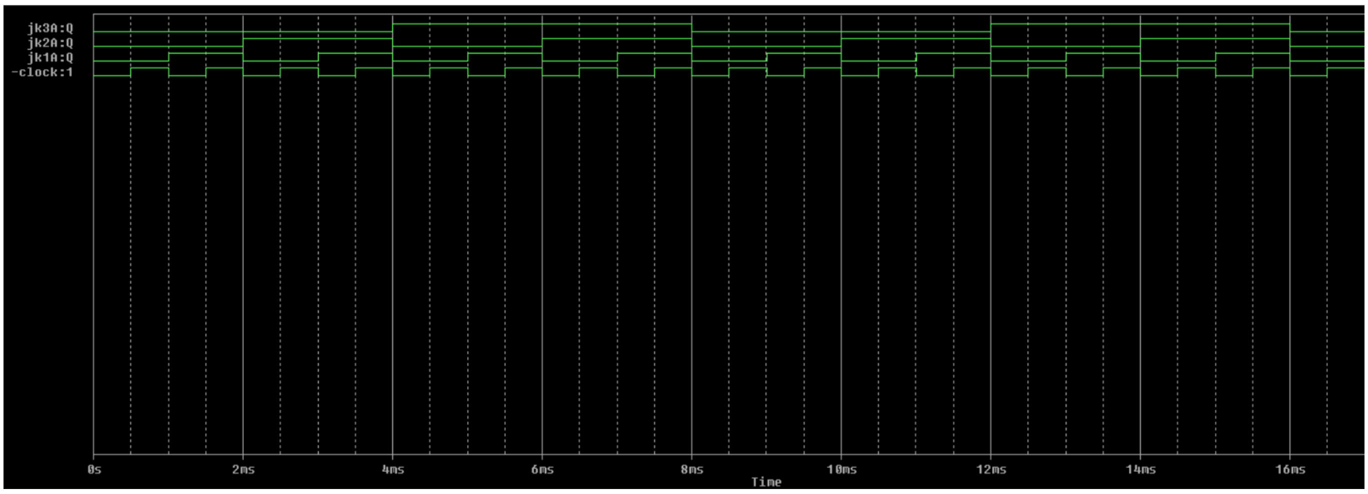


Figure 8(b) *Transient Response of Asynchronous 4 Bit Counter*

2.3 DC-DC coverters

A **DC-DC converter** is an electronic circuit that converts one level of **direct current (DC) voltage** into another. It plays a crucial role in **power management** for portable electronics, electric vehicles, renewable energy systems, and embedded devices. Based on the desired output, DC-DC converters are classified into different types:

- **Buck Converter** (step-down),
- **Boost Converter** (step-up),
- **Buck-Boost Converter** (step-up/down), and

These converters operate using **switching elements (transistors or MOSFETs)**, **energy storage components (inductors and capacitors)**, and **feedback control circuits**. They work on the principle of storing energy temporarily and releasing it at controlled intervals using high-frequency switching, which allows for high efficiency (often >90%).

DC-DC converters are widely used to power components that require a voltage level different from the supply, and are essential in battery-powered systems to ensure **voltage regulation**, **energy efficiency**, and **stable performance** under varying load conditions.

2.3.1 Boost Converter

A **Boost Converter** is a type of DC-DC converter that steps up (increases) the input voltage to a higher output voltage. It operates on the principle of energy storage in an inductor and its controlled release through a switching device, typically a transistor or MOSFET, along with a diode and capacitor. When the switch is **on**, current flows through the inductor, storing energy in its magnetic field. When the switch is **off**, the inductor releases energy to the load through the diode, resulting in a voltage boost.

Boost converters are widely used in **battery-powered devices**, **solar charge controllers**, and **portable electronics**, where input voltage is lower than the required operating voltage. Key parameters include the **duty cycle**, **switching frequency**, **inductor value**, **load resistance**, and **output capacitor**, which all affect the voltage gain and efficiency.

The Boost Converter was designed using **PSPICE code (netlist format)** rather than a graphical schematic. The code is given below:-

```
* SWITCH DRIVER

VCTRL 10 0 PULSE (0V 5V 0 0.01US 0.01US 5US 20US)

R10 10 0 1MEG

VIN 1 0 DC 20

SW1 1 2 10 0 SW

D1 0 2 DSCH

L1 2 3 50UH

C1 3 0 25UF

RL 3 0 5

.MODEL SW VSWITCH (VON=5V VOFF=0V RON=0.01 ROFF=1MEG)

.MODEL DSCH D(IS=0.0002 RS 0.05 CJO=5e-10 )

.TRAN 1US 800US

.PLOT TRAN V(2) V (3)

.PROBE

.END
```

In PSPICE code, the switching signal was defined using a VPULSE source with specific TON, TOFF, and PERIOD values to control the duty cycle and a **.TRAN** command was used to perform transient analysis. The transient simulation in PSPICE as shown in ***Figure 9*** showed the key behaviour of a boost converter. As the simulation progressed:

- The **inductor current** increased linearly during the ON time of the switch.
- During the OFF time, the inductor discharged through the diode, transferring energy to the output capacitor and load.
- The **output voltage (Vout)** was observed to rise above the input voltage (e.g., from 5 V input to ~9 V–12 V), depending on the duty cycle.

The simulation plot showed a steady rise in output voltage, eventually stabilizing due to the smoothing effect of the capacitor. The switching waveform also confirmed correct operation of the MOSFET and inductor behaviour. The converter successfully demonstrated **voltage boosting**, validating both the theoretical operation and PSPICE code design.

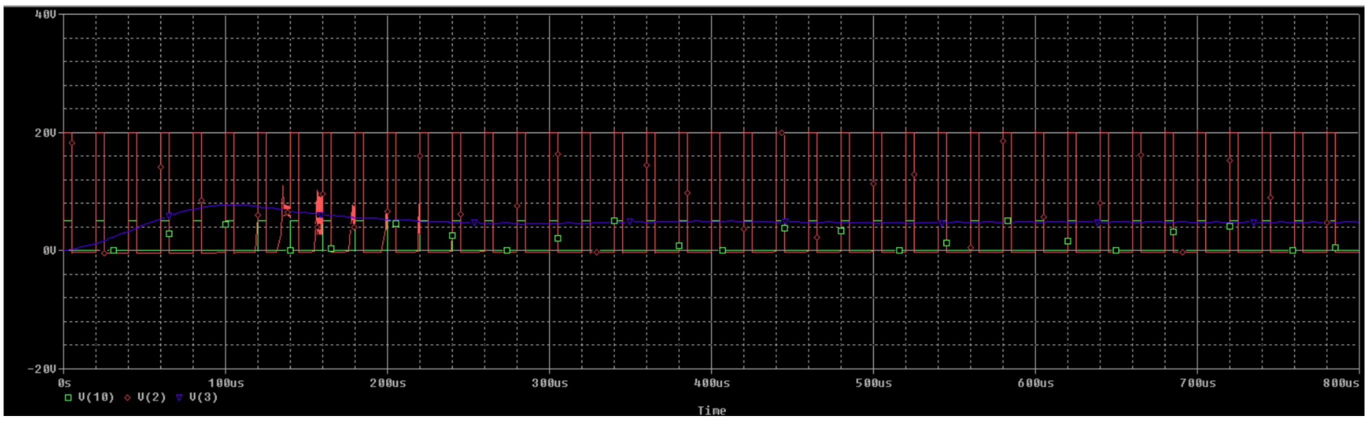


Figure 9 *Transient Response of Boost Converter*

3. Vivado Design Suite

Vivado Design Suite, developed by Xilinx (now part of AMD), is an advanced FPGA design and verification environment used for the development of digital systems on Xilinx FPGAs and SoCs. It provides an integrated workflow that includes RTL design (using VHDL, Verilog, or SystemVerilog), synthesis, implementation, simulation, and bitstream generation. Vivado also supports IP integration, high-level synthesis (HLS), and system-level design through its IP integrator and block design tools. Its intuitive graphical interface and powerful debugging features make it a preferred choice for complex digital design projects. I worked extensively with Vivado Design Suite to design, simulate, and implement LIF neuron model for future implementations in researches . This included writing and analysing HDL code, testing functionality through simulation, and verifying timing and resource utilization. The experience significantly enhanced my understanding of FPGA-based system design and digital logic implementation in a professional EDA environment.

3.1 LIF Neuron Model

3.1.1 Introduction

The human brain is a highly efficient computational system, capable of processing massive amounts of information with remarkable energy efficiency. Inspired by biological neurons, Spiking Neural Networks (SNNs) represent the third generation of neural networks that incorporate the concept of time into their processing model. Among the various neuron models proposed, the Leaky Integrate-and-Fire (LIF) model is one of the most widely used due to its simplicity and biological relevance.

This project focuses on the FPGA-based implementation of the LIF neuron model using Verilog HDL, aiming to create an efficient, real-time neuromorphic processing unit suitable for future developments in brain-inspired computing.

3.1.2 Motivation

Conventional neural networks require large computational resources and often consume significant power, making them less ideal for real-time and embedded applications. In contrast, neuromorphic systems, based on spiking neuron models, offer low-power and event-driven computation, closely mimicking biological processes. Implementing the LIF neuron on FPGA provides a flexible and efficient platform for prototyping neuromorphic systems. It enables real-time processing, reconfigurability, and scalability, making it ideal for applications in robotics, autonomous systems, IoT devices, and brain-computer interfaces.The motivation behind this project is to bridge the gap between theoretical neuroscience models and practical hardware implementations for energy-efficient intelligent systems.

The Leaky Integrate-and-Fire (LIF) model is one of the simplest yet biologically realistic neuron models. It captures the essential features of a spiking neuron without excessive computational complexity.

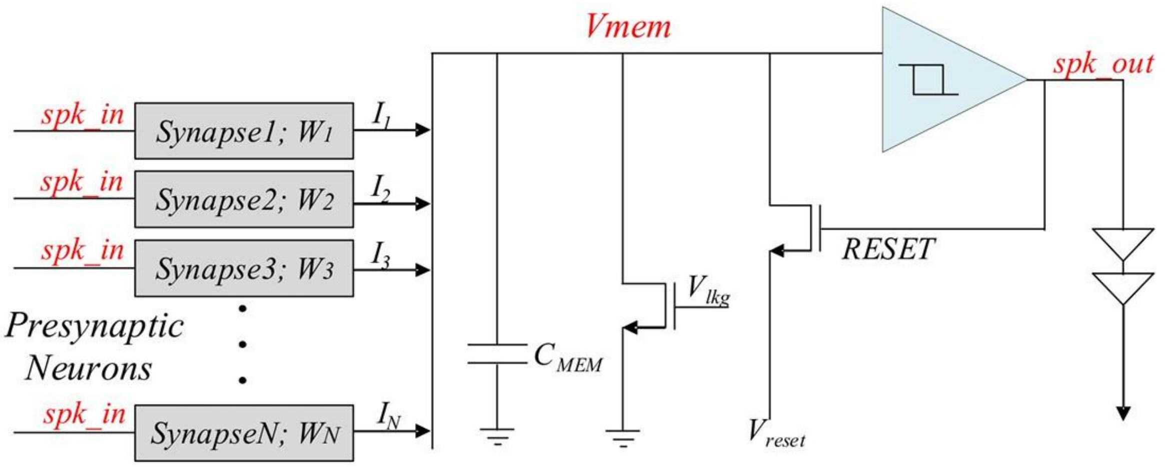


Figure 10 *LIF Neuron Model*

Figure 10 models the LIF, where the neuron is represented by simple CMOS devices. The evolution of the neuronal membrane potential Vmem can be modeled by a Resistor–Capacitor circuit, which is composed of a membrane capacitor Cmem and a membrane resistance Rmem (leakage path).

A spiking neuron receives input spikes from the several pre-synaptic neurons interconnected via synapses. Each synapse acts a current source and injects a current equivalent to its strength called weights (W). The state (potential) of Cmem is updated by injecting current through multiple current sources. Upon reaching a predefined threshold value, the comparator decides to evoke an output spike and resets the Vmem.

$$I(t) = I_R(t) + I_C(t),$$

$$I(t) = \frac{V_{mem} - V_{reset}}{R_{mem}} + C_{mem}\frac{\partial V_{mem}}{\partial t},$$

$$C_{mem}\frac{\partial V_{mem}}{\partial t} = I(t) - \frac{V_{mem} - V_{reset}}{R_{mem}}.$$

When Vmem ≥Vth, then Vmem = Vreset. In Equation (3) above, Vmem is the membrane potential, Vreset is the resting potential, and Vth is the predefined threshold potential. When there is no input current, then the capacitive charge will decay by leaking through the resistance until it reaches the resting potential.

3.1.3 Digital Design Considerations

While implementing the LIF neuron digitally on an FPGA, several considerations must be made:

- Fixed-point arithmetic: To reduce resource usage compared to floating- point.
- Memory usage: Membrane potentials and thresholds are stored in registers.

- Timing: Clock-driven updates ensure synchronized operation.
- Spike generation: Simple comparators are used to check if Vmem≥Vth.
- Reset mechanism: After spiking, the membrane potential is reset without additional clock cycles to ensure real-time operation.

The modularity of FPGA platforms enables scaling up the single neuron to build large-scale Spiking Neural Networks with parallel neuron processing.

3.1.4 FPGA for Neuromorphic Computing

Field-Programmable Gate Arrays (FPGAs) offer highly parallel and reconfigurable hardware, making them ideal for neuromorphic system design. Key advantages include:

- Parallelism: Multiple neurons and synapses can be processed simultaneously.
- Flexibility: Hardware can be reprogrammed to accommodate various neuron models.
- Low latency: Real-time spike processing is achievable.
- Power efficiency: Event-driven architectures minimize dynamic power consumption.

Therefore, FPGA-based digital implementations of LIF neurons form the stepping stone toward building energy-efficient, scalable neuromorphic systems for real- world applications

3.1.5 Tools and platform used

- FPGA Board: Boolean Board with Spartan-7 XC7S50CSGA324 FPGA
- Software: Vivado Design Suite 2022.1
- Programming Language: Verilog HDL
- Simulation Tool: Vivado XSim (Behavioral and Post-synthesis simulation)
- Clock Frequency: 100 MHz system clock
- Display Interface: 7-Segment Display (for real-time membrane potential and spike count visualization)

3.1.6 Implementation Strategy

The following steps were undertaken to implement and validate the LIF neuron model:

- Modelling:
    - The LIF neuron was modelled in Verilog HDL based on the discrete- time equation derived using Euler's method.
    - Parameters such as membrane resistance (Rmem), time constant, reset potential, threshold potential were parameterized for flexibility.
- Discretization:
    - Continuous variables were discretized into fixed-point representation to suit FPGA hardware constraints.
    - Membrane potential updates and spike generation were clock-driven at each positive edge of the clock.
- Spike Generation and Reset:
    - When the membrane potential exceeded the threshold, a spike signal was asserted.
    - Immediately after spiking, the membrane potential was reset to the predefined reset value.
- Resource Optimization:
    - Arithmetic operations were optimized using fixed-point multipliers and adders.
    - Bit-widths were carefully chosen to balance between precision and resource usage.
- Simulation:
    - Initial simulations were conducted using test benches to verify functional correctness.
    - Test cases included constant current injection, pulse inputs, and random noise inputs.

- Synthesis and Implementation:
  - After successful simulation, the design was synthesized using Vivado.
  - The synthesized design was implemented on the Spartan-7 FPGA, and real-time outputs were visualized using onboard LEDs and 7-segment displays.
- Verification:
  - Behavioural simulation results were compared with theoretical LIF neuron behaviour.
  - Timing analysis and static timing reports were reviewed to ensure design stability.

3.1.7 Input Stimuli Used

Pulse Train Input: Periodic pulses to examine time-based behaviour and spike frequency adaptation.

3.1.8 Parameters Set

Parameter	**Value**	**Description**
Vinitial	0 units	Resting membrane potential
Vthreshold	50 units	Spike threshold
Vreset	0 units	Reset value after spike
Clock Frequency	100 MHz	System clock frequency
Fixed-point Format	Q8.8	16-bit representation

3.1.9 Button Debounce Handling

In FPGA-based systems, mechanical buttons are prone to bouncing—a phenomenon where a single press generates multiple unwanted transitions due to physical contact oscillations. To ensure clean and single-action triggering, a debounce circuit was implemented. Debounce Circuit Details:

- The button input was sampled and passed through a synchronous debounce module.
- A counter-based debounce algorithm was used:
- If the button state remained stable for a predefined number of clock cycles (debounce interval), it was considered a valid press.
- Otherwise, spurious transitions were filtered out.
- Debounce Interval: Configured based on the clock frequency (typically 10– 20 ms equivalent at 100 MHz).

This was particularly important when manually injecting current pulses into the neuron model using onboard push buttons or for resetting the spike counter manually.

3.1.10 Spiky Edge Detection

In the digital domain, when the LIF neuron fires, the spike output becomes high. However, for counting the number of spikes accurately and triggering external events, it is necessary to detect the rising edge of the spike signal, not the level itself.

- Edge Detector Module Details:
  - A simple rising-edge detector was designed.
  - It worked by storing the previous state of the spike signal and comparing it with the current state.
  - If the previous state was '0' and the current state is '1', a spike event was detected.
- Purpose:
  - To increment the spike counter only once per spike, avoiding multiple counts during a prolonged high state.

- To trigger output displays like updating the 7-segment display showing the total number of spikes.
- To ensure synchronous operation without missing any spike even at high firing rates.

3.1.11 Key Observations

- The membrane potential increased linearly with constant input and decayed exponentially during idle periods.
- Spikes were consistently generated once the threshold was crossed, followed by an immediate reset.
- In the presence of noise, the neuron showed minor fluctuations but maintained correct spike generation, demonstrating stability.
- Timing closure was achieved with no setup or hold time violations at 100 MHz.
- Resource usage was minimal, making the design scalable for multiple neurons on the same FPGA.

3.1.12 Code and Waveform Results

Verilog code for waveform simulation:

```
module lif_neuron ( input clk,

input reset, input spike_in,

output reg spike_out,

output [7:0] potential_dbg, // expose potential

output [7:0] threshold_dbg, // expose parameter as signal output [7:0] increment_dbg,

output [7:0] leak_dbg

);

 reg [7:0] potential;

parameter THRESHOLD = 100; parameter LEAK = 1;

parameter INCREMENT = 20;

assign potential_dbg = potential; assign threshold_dbg = THRESHOLD;

assign increment_dbg = INCREMENT; assign leak_dbg = LEAK;

always @(posedge clk or posedge reset) begin if (reset) begin

potential <= 0;

spike_out <= 0; end else begin

if (spike_in)

potential <= potential + INCREMENT - LEAK;

if (potential >= THRESHOLD) begin spike_out <= 1;

potential <= 0; end else begin

spike_out <= 0; end

end end

endmodule
```

Behavioral simulation:

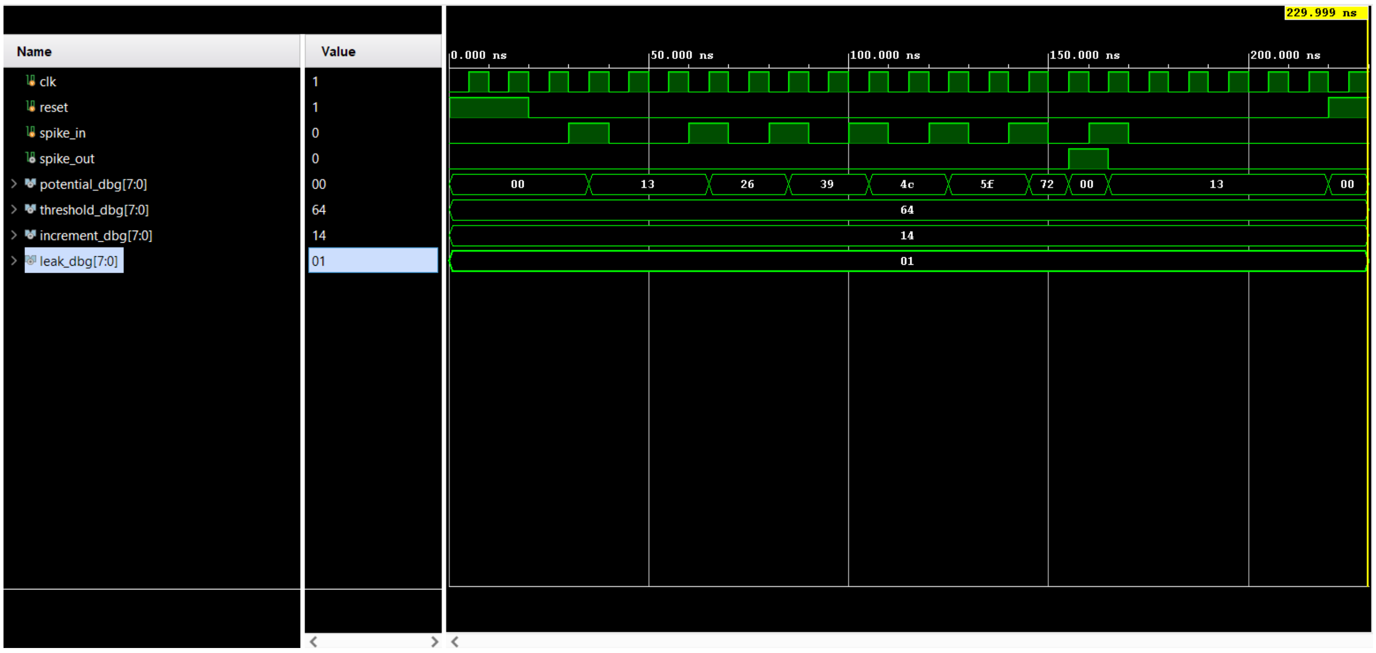


Figure 11 *Simulation of LIF Neuron Model*

Final code for FPGA implementation:

```
module lif_neuron_top (

input clk,            // Clock input (100 MHz) input reset_btn,               // Button J2

input spike_btn,  // Button J5 output spike_led,                   // LED G1 output clean_spike_led, // LED E2 output
spike_edge_led, // LED F1 output [7:0] potential_leds // potential

);

wire clean_reset, clean_spike; wire spike_edge;

wire spike_out; wire [7:0] potential;

// Debounce for both reset and spike debounce db_reset (

.clk(clk),

.noisy_in(reset_btn),

.clean_out(clean_reset)

);

debounce db_spike (

.clk(clk),

.noisy_in(spike_btn),

.clean_out(clean_spike)

);

// Edge detector for spike reg prev_spike;

always @(posedge clk) prev_spike <= clean_spike;

assign spike_edge = clean_spike & ~prev_spike;

// LIF Neuron lif_neuron neuron (
.clk(clk),
.reset(clean_reset),
.spike_in(spike_edge),
.spike_out(spike_out),
.potential(potential)
);
// Hold spike LED for visibility (~100ms) reg [23:0] spike_timer;
always @(posedge clk) begin if (spike_out)
spike_timer <= 24'd10_000_000; // ~100 ms at 100 MHz else if (spike_timer > 0)
spike_timer <= spike_timer - 1;
end
assign spike_led = (spike_timer > 0);
// Debug LEDs
assign clean_spike_led = clean_spike; assign spike_edge_led = spike_edge; assign potential_leds = potential;
endmodule
module lif_neuron ( input clk,
input reset, input spike_in,
output reg spike_out, output reg [7:0] potential
);
parameter THRESHOLD = 100; parameter INCREMENT = 20; parameter LEAK = 1;
always @(posedge clk or posedge reset) begin if (reset) begin
potential <= 0;
spike_out <= 0; end else begin
if (spike_in)
potential <= potential + INCREMENT - LEAK;
if (potential >= THRESHOLD) begin spike_out <= 1;
potential <= 0; end else begin
spike_out <= 0; end
end end
endmodule module debounce (
input clk, input noisy_in,
output reg clean_out
);
reg [19:0] counter = 0; reg stable_state = 0;
parameter THRESHOLD = 200_000; // ~2ms
always @(posedge clk) begin
```

```
if (noisy_in != stable_state) begin counter <= counter + 1;

if (counter >= THRESHOLD) begin stable_state <= noisy_in;

counter <= 0; end

end else begin counter <= 0;

end

clean_out <= stable_state; end

endmodule
```

Constraint file:

```
## Clock

set_property -dict {PACKAGE_PIN F14 IOSTANDARD LVCMOS33} [get_ports {clk}]

## Buttons

set_property -dict {PACKAGE_PIN J2 IOSTANDARD LVCMOS33} [get_ports {reset_btn}] set_property -dict
{PACKAGE_PIN J5 IOSTANDARD LVCMOS33} [get_ports {spike_btn}]

## Spike Edge LEDs

set_property -dict {PACKAGE_PIN G1 IOSTANDARD LVCMOS33} [get_ports {spike_led}] set_property -dict
{PACKAGE_PIN E2 IOSTANDARD LVCMOS33} [get_ports

{clean_spike_led}]

set_property -dict {PACKAGE_PIN F1 IOSTANDARD LVCMOS33} [get_ports

{spike_edge_led}]

## Potential output on 8 LEDs

set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports

{potential_leds[0]}]

set_property -dict {PACKAGE_PIN E5 IOSTANDARD LVCMOS33} [get_ports

{potential_leds[1]}]

set_property -dict {PACKAGE_PIN E6 IOSTANDARD LVCMOS33} [get_ports

{potential_leds[2]}]

set_property -dict {PACKAGE_PIN C3 IOSTANDARD LVCMOS33} [get_ports

{potential_leds[3]}]

set_property -dict {PACKAGE_PIN B2 IOSTANDARD LVCMOS33} [get_ports

{potential_leds[4]}]

set_property -dict {PACKAGE_PIN A2 IOSTANDARD LVCMOS33} [get_ports

{potential_leds[5]}]

set_property -dict {PACKAGE_PIN B3 IOSTANDARD LVCMOS33} [get_ports

{potential_leds[6]}]

set_property -dict {PACKAGE_PIN A3 IOSTANDARD LVCMOS33} [get_ports

{potential_leds[7]}]
```

Implementation on FPGA:

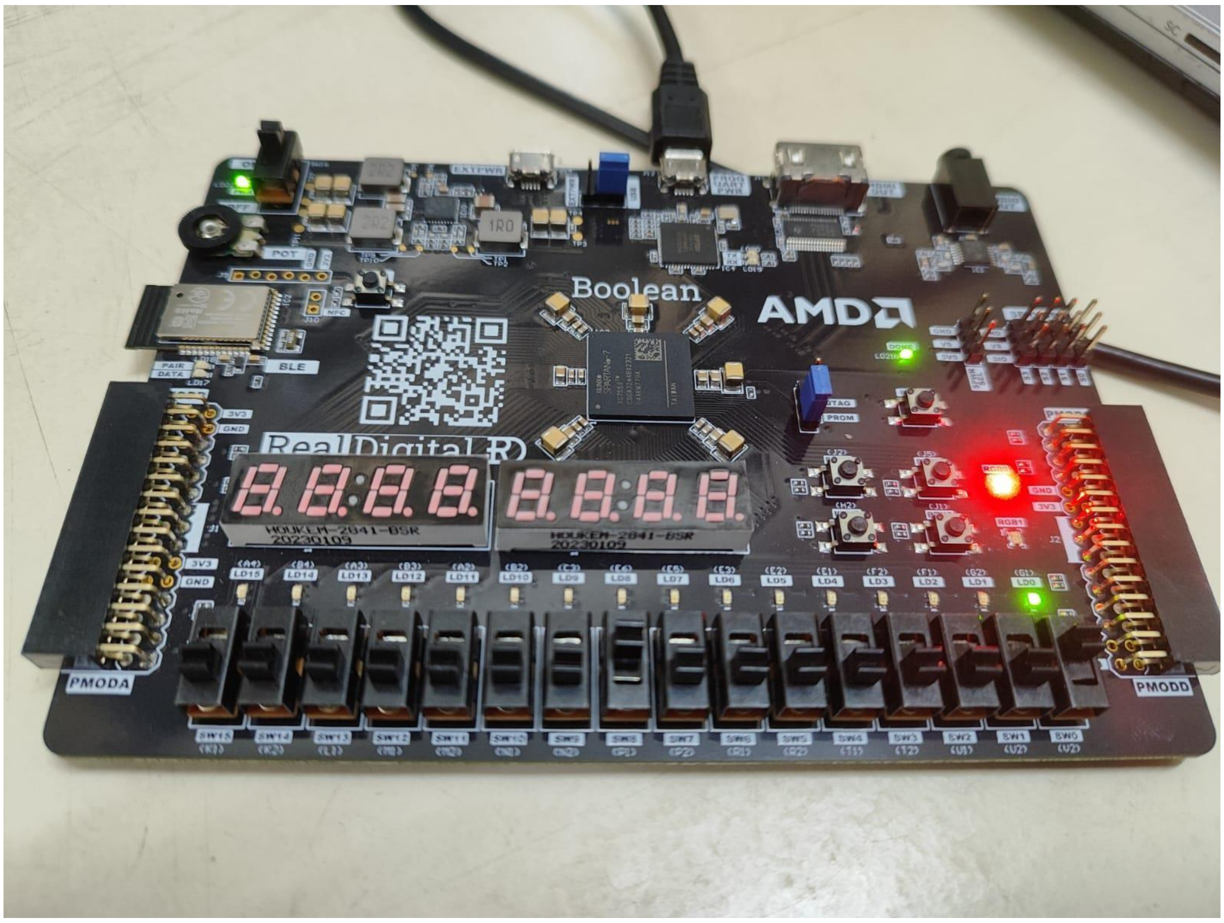


Figure 12 *FPGA Implementation*

3.1.13 Conclusion

In this project, a biologically inspired Leaky Integrate-and-Fire (LIF) neuron model was successfully implemented on an FPGA platform. The project demonstrated how neuromorphic behaviour, typically studied in neuroscience and brain-inspired computing, can be effectively modelled using digital logic design principles. Key modules such as button debounce circuits, current pulse generators, spike edge detectors, and spike counters were integrated seamlessly to simulate the neuron dynamics. The real-time updates of membrane potential and spike counts were displayed through 7-segment displays, ensuring a complete end-to-end working system.

The use of hardware description principles allowed for:

- Low-latency, parallel, and real-time operation.
- Precise control over the spike timings and neuron state.
- Modular design, enabling scalability toward multi-neuron networks.

Thus, the project not only achieved a working model of a single neuron but also laid down a strong foundation for building more complex neuromorphic systems in the future using FPGA technology.

3.1.14 Future Scope

While this project was centred around the design of a single LIF neuron model, it opens up several promising future directions:

- Multi-Neuron Networks: Extending the design to implement an array of interconnected neurons to form a complete Spiking Neural Network (SNN) architecture on FPGA.
- Learning Mechanisms: Incorporating Spike-Timing Dependent Plasticity (STDP) or other learning rules to enable on-chip learning and adaptation capabilities.
- Optimization for Low Power: Further optimizations can be made at the RTL level to minimize dynamic power consumption, which is crucial for real- world neuromorphic applications.
- Hybrid Digital-Analog Designs: Future implementations can explore mixed- signal approaches combining the digital control of FPGAs with analogue neuron circuits for enhanced biological realism.
- Interface with External Sensors: The neuron models can be connected to real-world input signals (audio, vision, etc.) for building event-driven sensory systems.
- Use of Specialized FPGAs: Deploying the neuron models on neuromorphic chips like Intel Loihi or developing on low-power FPGAs for edge AI applications.
- Parallel Implementation: Exploring full parallel SNN accelerators by utilizing the massive parallelism offered by FPGA fabrics to simulate thousands of neurons in real time.

By addressing these future directions, FPGA-based neuromorphic computing can evolve into a powerful, energy-efficient alternative to traditional deep learning models, especially in the domain of edge AI, robotics, and brain-machine interfaces.

4 Cadence Virtuoso

Cadence Virtuoso is a powerful and industry-standard Electronic Design Automation (EDA) platform used primarily for the design of custom analog, mixed-signal, and RF integrated circuits. It offers a complete suite of tools covering schematic design, layout editing, simulation, and system-level verification. Key components include the Virtuoso Schematic Editor for circuit creation, the Layout Suite for physical design with real-time checks, and the Analog Design Environment (ADE) for simulation and analysis. The platform supports advanced technology nodes and enables seamless front-to-back design flows with features like parasitic-aware design and AI-driven productivity tools. Widely used in semiconductor, automotive, aerospace, and communication industries, Virtuoso plays a crucial role in enabling complex IC designs with high precision, efficiency, and reliability. I utilized Cadence Virtuoso to design and simulate various analog and digital circuits, gaining hands-on experience in circuit modeling, layout creation, and performance verification. This exposure helped me understand industry-standard workflows and develop practical skills in IC design. Below are the Circuit Designs I did in cadence virtuoso.

4.1 Two Stage MOS Amplifier Circuit

The two-stage MOS amplifier is a widely used configuration in analog circuit design, known for offering high gain and improved output swing. It consists primarily of a differential amplifier stage followed by a gain stage, which together enhance voltage amplification while maintaining stability. The first stage typically provides high input impedance and differential gain, whereas the second stage increases the overall gain and drives the output load.

Using Cadence Virtuoso, the two-stage amplifier was designed at the transistor level employing CMOS technology as shown in ***Figure 13(a)*** . The schematic includes carefully biased NMOS and PMOS transistors, with compensation capacitors included to improve phase margin and stability. Bias currents were set through current sources, and resistive elements were used to set appropriate gain and operating points. After schematic capture, the circuit was simulated using the Analog Design Environment (ADE), with AC analysis performed to evaluate frequency response.

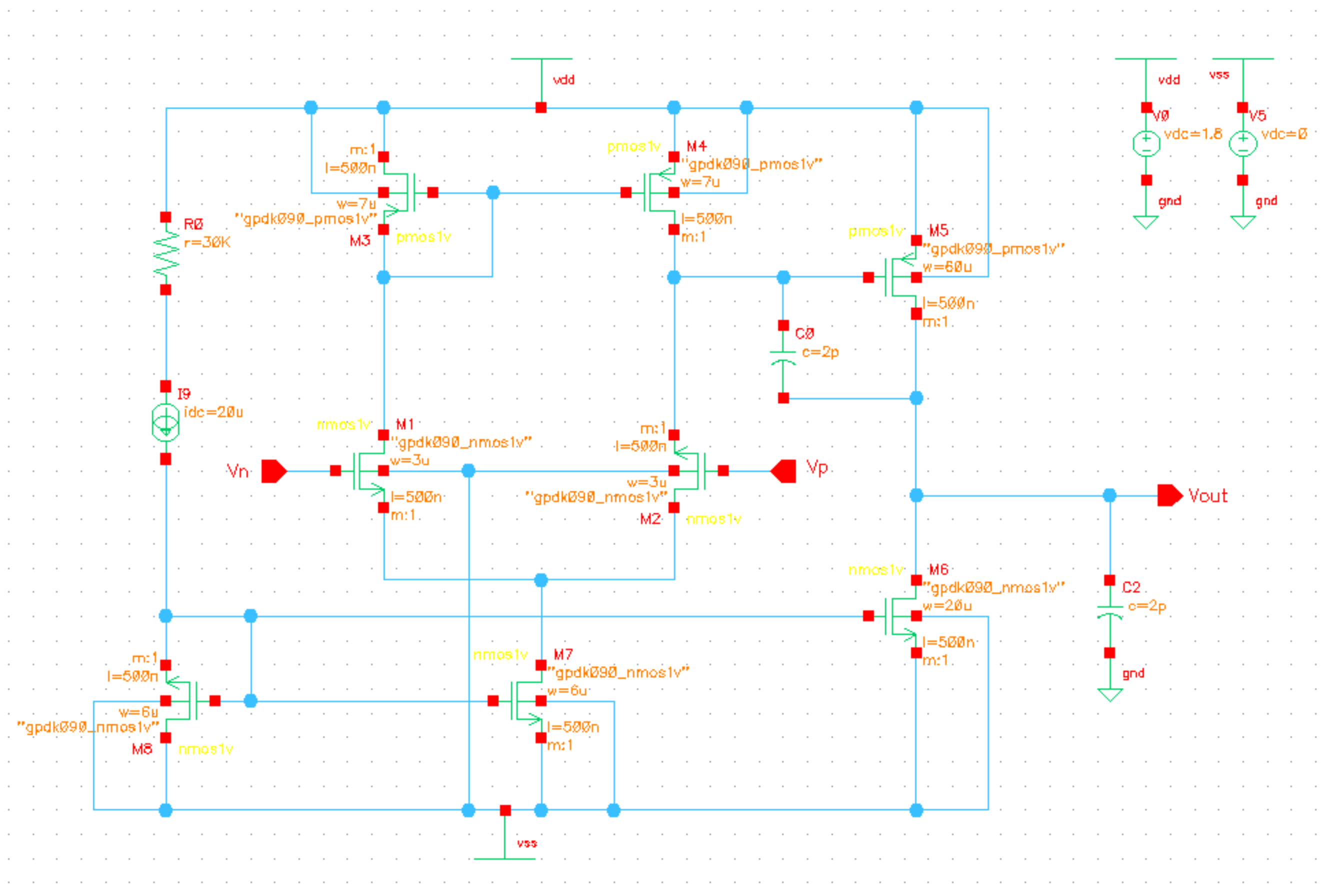


Figure 13(a) *Schematic of Two Stage MOS Amplifier Circuit*

The simulation results shown in ***<u>Figure 13(b)</u>*** demonstrate the amplifier's gain and phase behavior over a range of frequencies. The Bode plot clearly shows the gain-bandwidth characteristics, with the amplifier achieving a mid-band gain near 40 dB and a bandwidth extending beyond 1 MHz. The phase margin and gain roll-off indicate good stability, suitable for high-performance analog applications. This exercise not only reinforced the theoretical concepts of multistage amplification but also enhanced practical skills in transistor-level design and verification using Cadence Virtuoso.

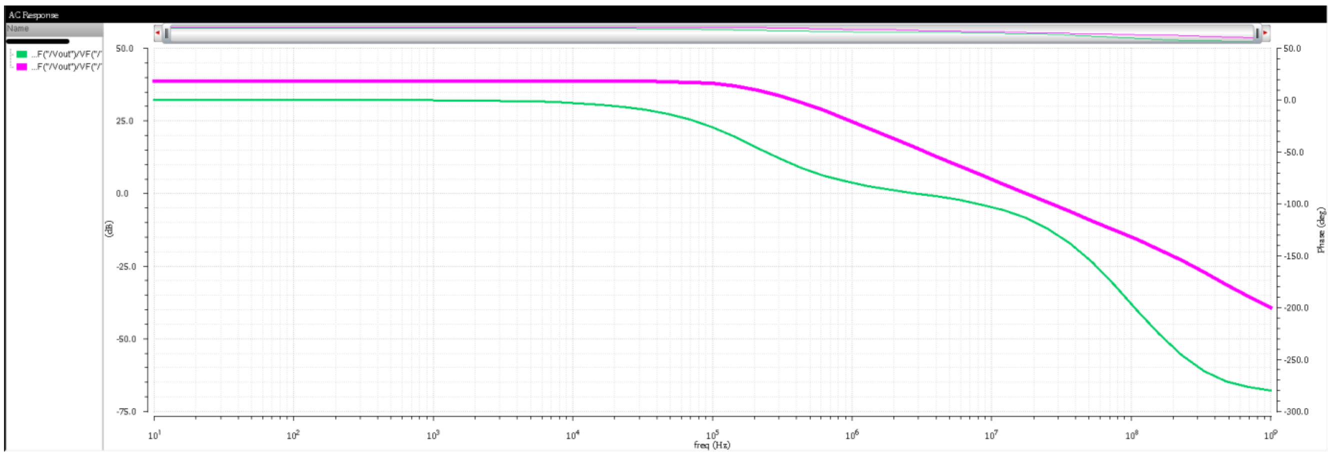


Figure 13(b)

4.2 Schmitt Trigger

A Schmitt Trigger is a type of comparator circuit with hysteresis, used to convert a noisy or slowly varying analog input signal into a clean digital output. It is particularly effective in eliminating noise and providing sharp transitions in digital systems. The circuit works by defining two distinct threshold voltages—one for transitioning from low to high and another for transitioning from high to low—ensuring stable switching behaviour.

In Cadence Virtuoso, the Schmitt Trigger was designed using CMOS technology as shown in ***Figure 14(a)*** with cross-coupled PMOS and NMOS transistors. The schematic comprises an inverter-like structure with positive feedback introduced via additional transistors to create the required hysteresis behavior. The sizing of transistors and biasing voltages were carefully selected to achieve proper threshold separation and reliable switching.

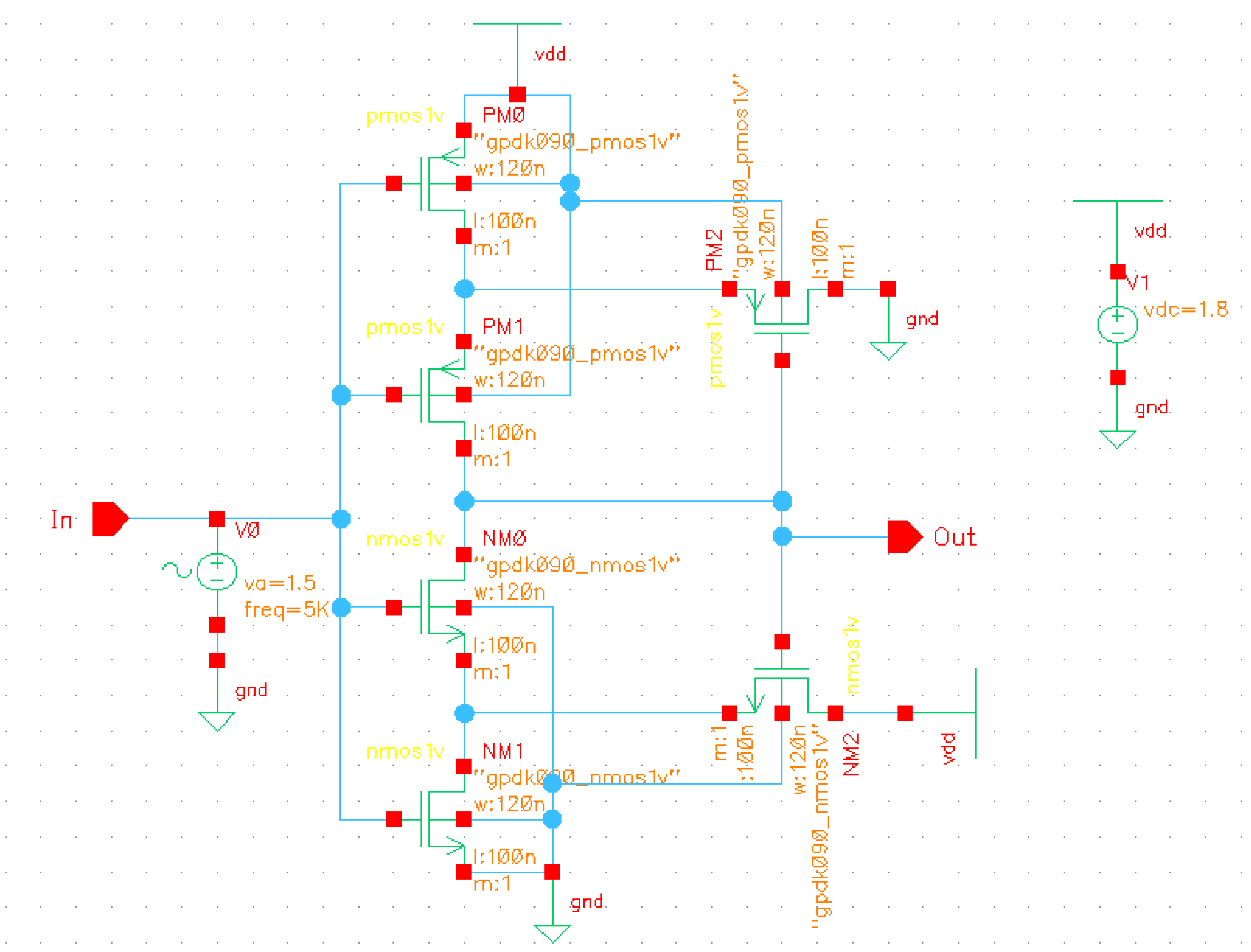


Figure 14(a) Schematic of Schmitt trigger

The transient simulation output shown in ***Figure 14(b)*** validates the operation of the Schmitt Trigger. A sinusoidal input signal with 5 kHz frequency and 1.5 V amplitude was applied, and the output waveform clearly demonstrates square-wave behavior. As shown in the plot, the circuit switches cleanly at two distinct input voltages, eliminating any mid-level ambiguity or noise sensitivity. This successful implementation and simulation confirmed the circuit's functionality and its importance in digital signal conditioning applications.

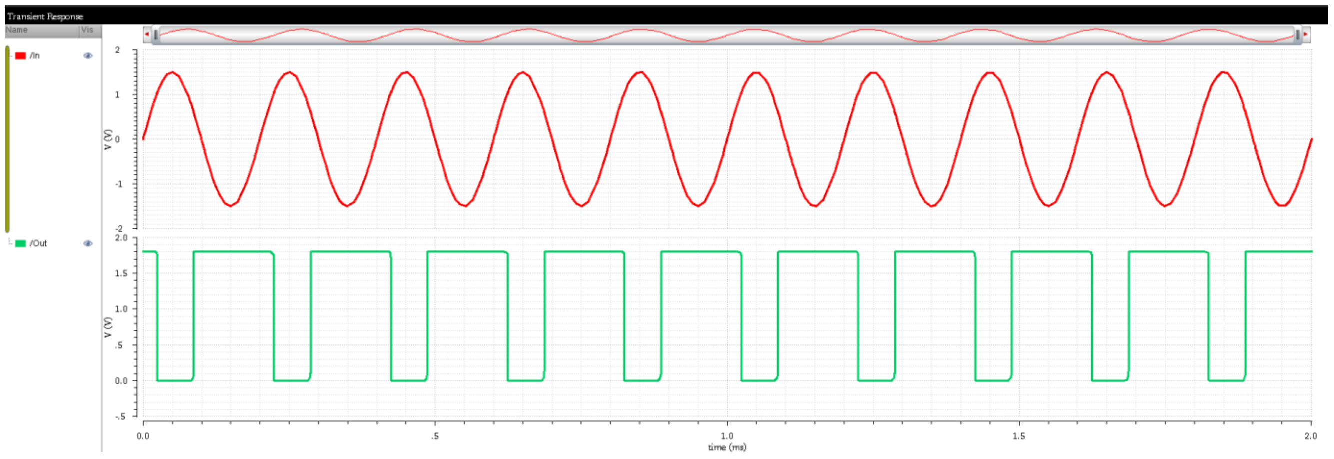


Figure 14(b) Transient analysis of Schmitt trigger

4.3 Memory Chip

In modern computer systems, various types of memory are employed to store and manage data. These memories can be broadly categorized into the following category

- **Volatile Memory :** Volatile memory is temporary storage that loses its contents when the power is turned off. The primary volatile memory types include RAM. RAM is a fast, volatile memory used for temporary data storage. It allows quick read-and-write access, making it essential for active applications and the operating system. There are mainly two kinds of RAM:
    - **Dynamic Random Access Memory (DRAM):** DRAM is a type of RAM that stores each bit of data in a separate capacitor within an integrated circuit. It requires periodic refresh cycles to maintain data integrity.
    - **Static Random Access Memory (SRAM):** SRAM is another form of volatile memory that does not require refreshing. It is often used for cache memory in CPUs due to its faster access times compared to DRAM.

- **Non-volatile Memory :** Non-volatile memory retains its data even when power is turned off. Common types of non-volatile memory include:
    - **Read-Only Memory (ROM):** ROM is used to store permanent data, typically in firmware and software. It is non-volatile and is not easily modified.
    - **Flash Memory:** Flash memory is a type of non-volatile storage that can be electrically erased and reprogrammed. It is commonly used in USB drives, SSDs, and memory cards.

4.3.1 Static Random Access Memory (SRAM)

SRAM is a type of volatile memory that uses bistable latching circuitry to store each bit. Unlike DRAM, SRAM does not require refreshing, making it faster and more energy-efficient. SRAM cells offer an advantage over other types of volatile memory, such as Dynamic Random Access Memory (DRAM), which is slower and requires periodic refresh cycles. This makes SRAM cells a fundamental building block of modern computing systems and a key technology for achieving high-speed data processing.

Applications of SRAM

- SRAM is widely used in various applications due to its speed, low power consumption, and suitability for cache memory. Some key applications include:

- **Cache Memory:** SRAM is commonly used as cache memory in CPUs to provide fast access to frequently used data.
- **Register Files:** SRAM is used in register files within microprocessors for storing temporary data during computation.
- **Networking Devices:** SRAM is employed in networking devices for buffering and storing routing tables.
- **Embedded Systems:** SRAM finds applications in embedded systems, providing quick access to critical data.

4.3.2 Objectives

The primary objective of this task was to gain a comprehensive understanding of Static Random Access Memory (SRAM) by designing a complete 6T SRAM cell, including its peripheral circuitry, using Cadence tools on the 180nm technology node (GPDK180). This project aims to explore various aspects of SRAM design with a focus on optimizing read and write stability, transistor sizing for enhanced data stability, and the design of peripheral circuits to achieve faster read and write operations

The project aims not only to achieve the defined objectives but also to foster a collaborative environment for knowledge sharing and continuous improvements. Feel free to explore the repository, contribute, raise issues, and engage in discussions as we work together to achieve the project goals.

4.3.3 SRAM Memory Architecture

The SRAM memory architecture employs a systematic arrangement of cells, word lines, and bit lines, all orchestrated through precise address decoding. The design facilitates efficient read and write operations, crucial for the reliable functioning of SRAM in various computing applications.

a. SRAM Cell

The SRAM memory architecture is structured around the fundamental unit known as the SRAM cell. This cell, typically a 6T configuration, holds a bit of data, representing either a 1 or 0. The architecture is designed for efficient data storage and retrieval, with specific components contributing to its functionality.

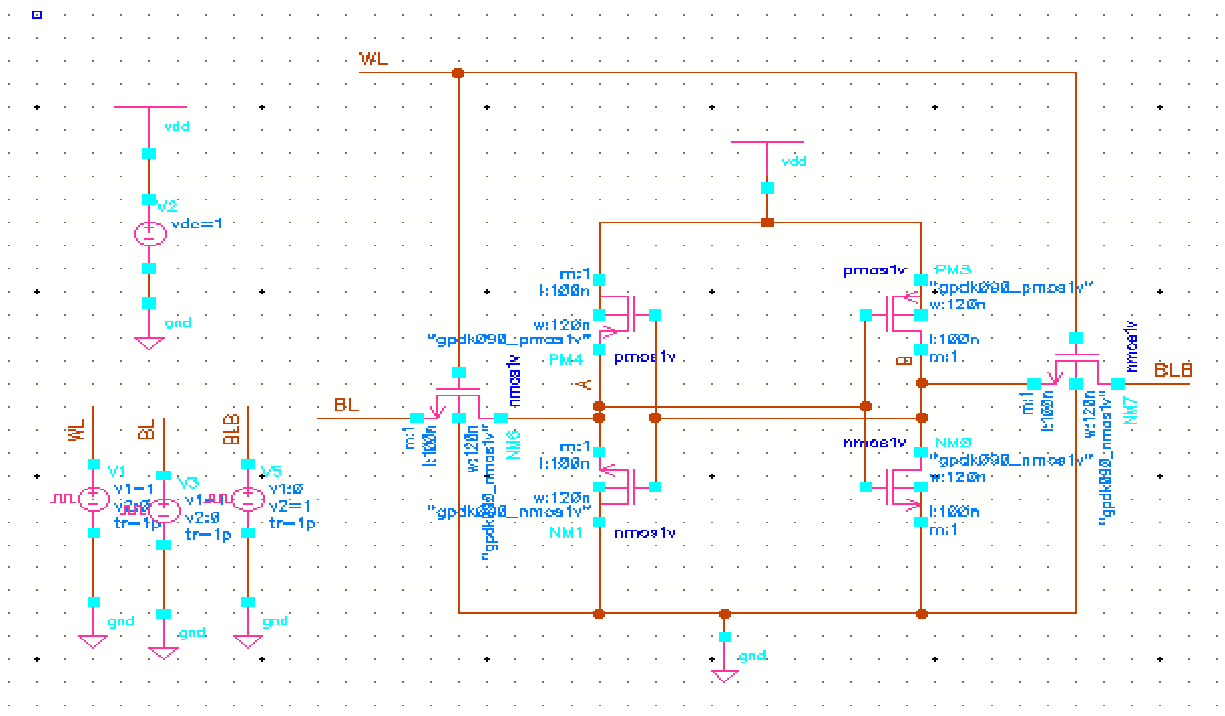


Figure 15 Schematic of SRAM cell

b. Pre-charge circuit

Pre-circuit is used to pre-charge the bit lines to Vdd before the Read operation. It consists of:

- Equalizer transistor which reduces the required pre-charge time by ensuring that both bit lines are at nearly equal voltages even if they are not pre-charged to Vdd.
- Bias transistors that pull each of the bit lines to Vdd.

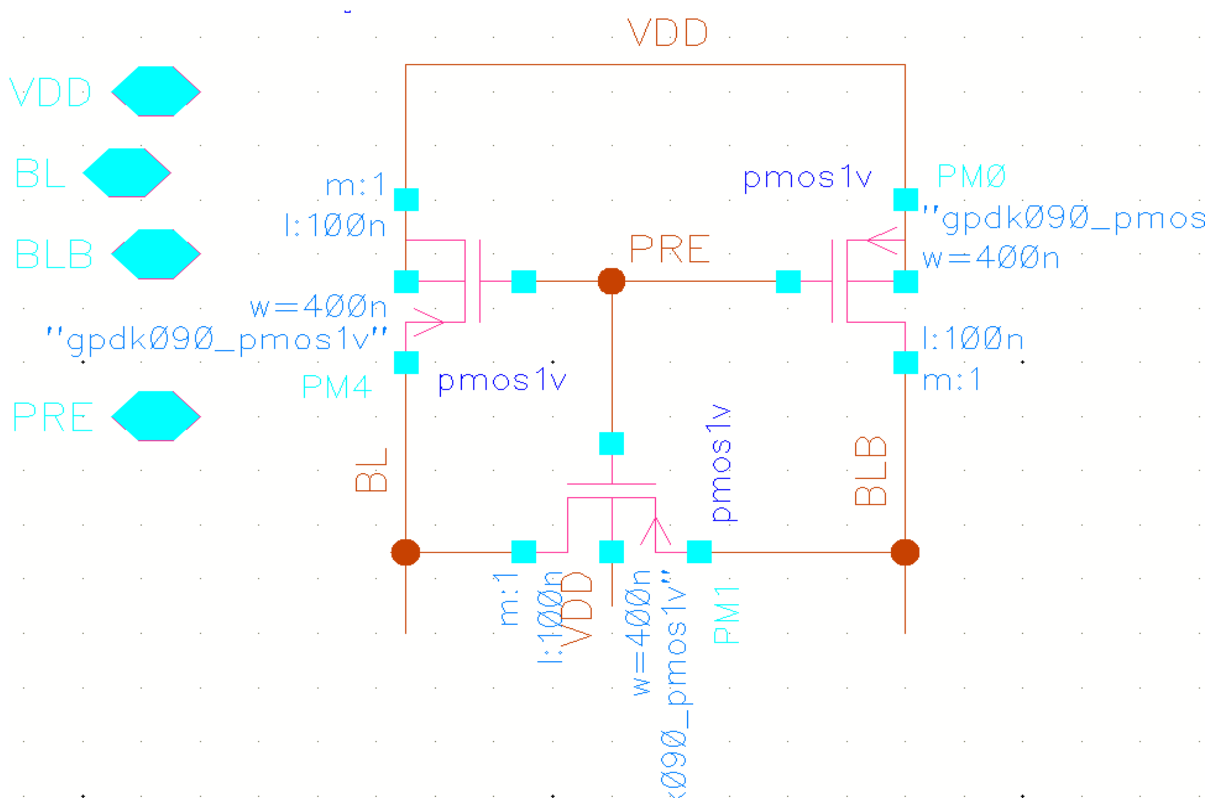


Figure 16 Schematic of Pre –Charge Circuit

c. Decoder circuit

A decoder circuit is a fundamental component in memory cell design, used to select specific rows or columns in memory arrays such as RAM or ROM. In essence, a decoder translates binary address

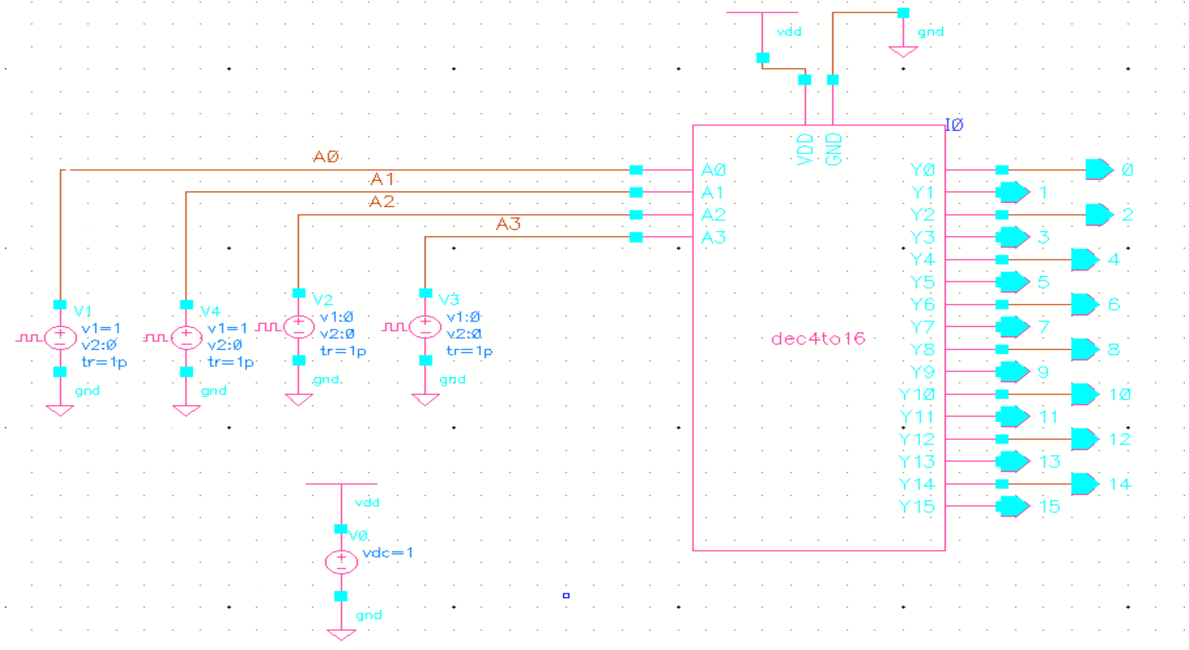


Figure 17(a) Schematic of Decode circuit inputs into a single active output line, enabling access to a specific memory location. For example, a 2-to-4 decoder activates one of four outputs based on a 2-bit address input.In memory cells, row decoders are used to select the word line, while column decoders help in identifying the bit line for read/write operations. These decoders are typically implemented using combinations of **AND gates** or NAND/NOR gate logic, and must be fast, area-efficient, and consume minimal power to ensure reliable performance in large-scale memory arrays. Proper decoder design ensures that only one memory cell is accessed at a time, preventing data corruption and enabling accurate data storage and retrieval. The circuit shown in ***Figure 17(a)*** is a 4 to 16 decoder which is designed in cadence software. In which 4 inputs A0, A1, A2, A3, And the outputs are I0,I1, I2, I3, I4, I5, I6, I6, I7, I8, I9, I10, I11, I12, I13, I14, I15.

Output From Decoder

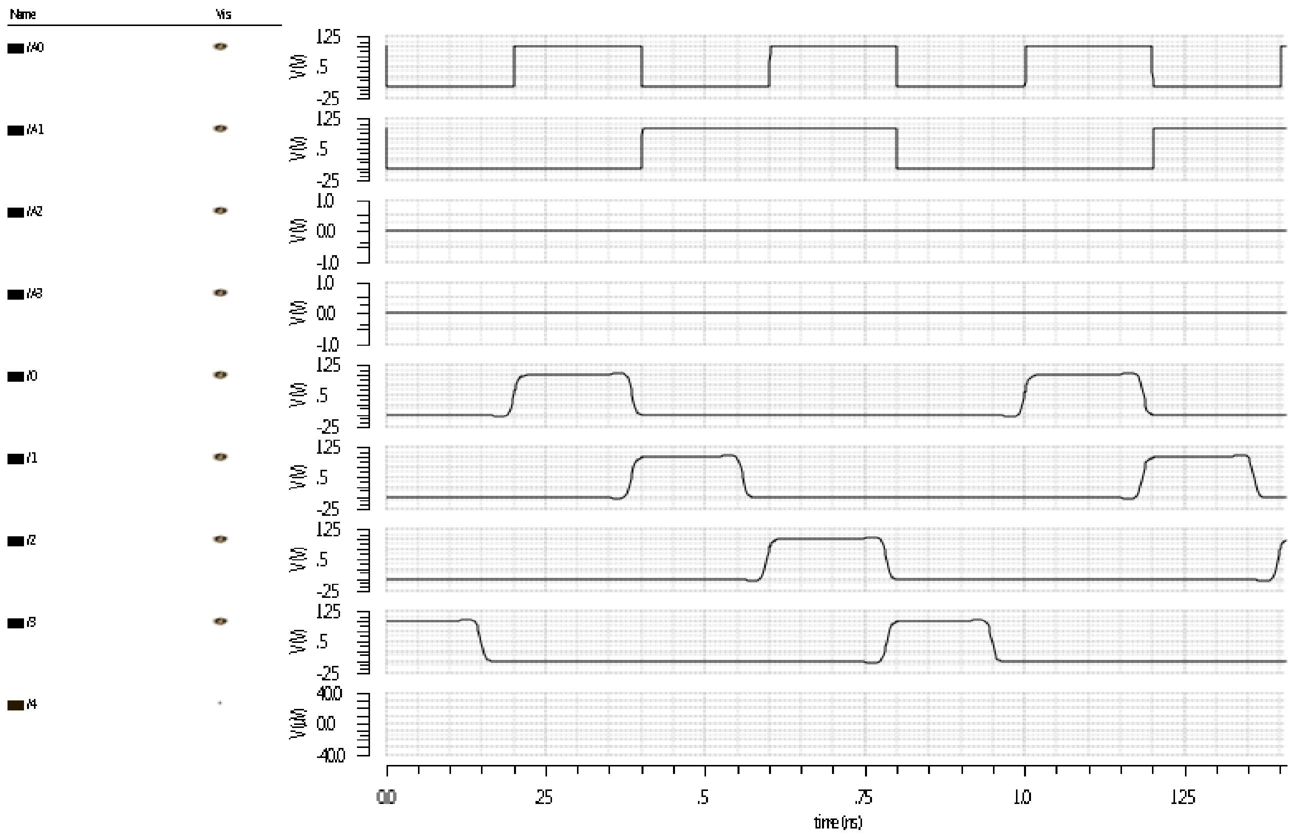


Figure 17(b) Transient analysis of decoder

The output follows the input but after some interval of time. Because of the delay in the design i.e 270 ps. So we are getting the desired output after some intervals. We need to minimize these interval inorder to get some specification.

d. Sense Amplifier circuit

Sense amplifier are the vital component in the memory design. The job of sense amplifier is to sense the bit line and bit line bar for proper monitoring action. It improves the read and write speed of the memory cell. Its another job is to reduce the power needed for the operation. The sense amplifiers primary job is to amplification of the voltage difference is being produced on the bit line and bit line bar at the time of operation. As it has the important job in the memory so it has different circuits for the operation.

As we know that in SRAM operation we don't need refresh of the memory for the further process, so the sense amplifiers non destructive at the time of operation. As the column multiplexers are connected in the memory cell at that time multiplexer should choose one sense amplifier for the single input. So that we can get proper use of the sense amplifier in the designing circuit.

These are the various parameters of an sensing amplifier-

- Gain A $=V_{out}/V_{in}$.
- Sensitivity S $=v_{in}$ min – least noticeable signal.
- Rise time t_{rise}, fall time t_{fall} – 10% to 90%

.

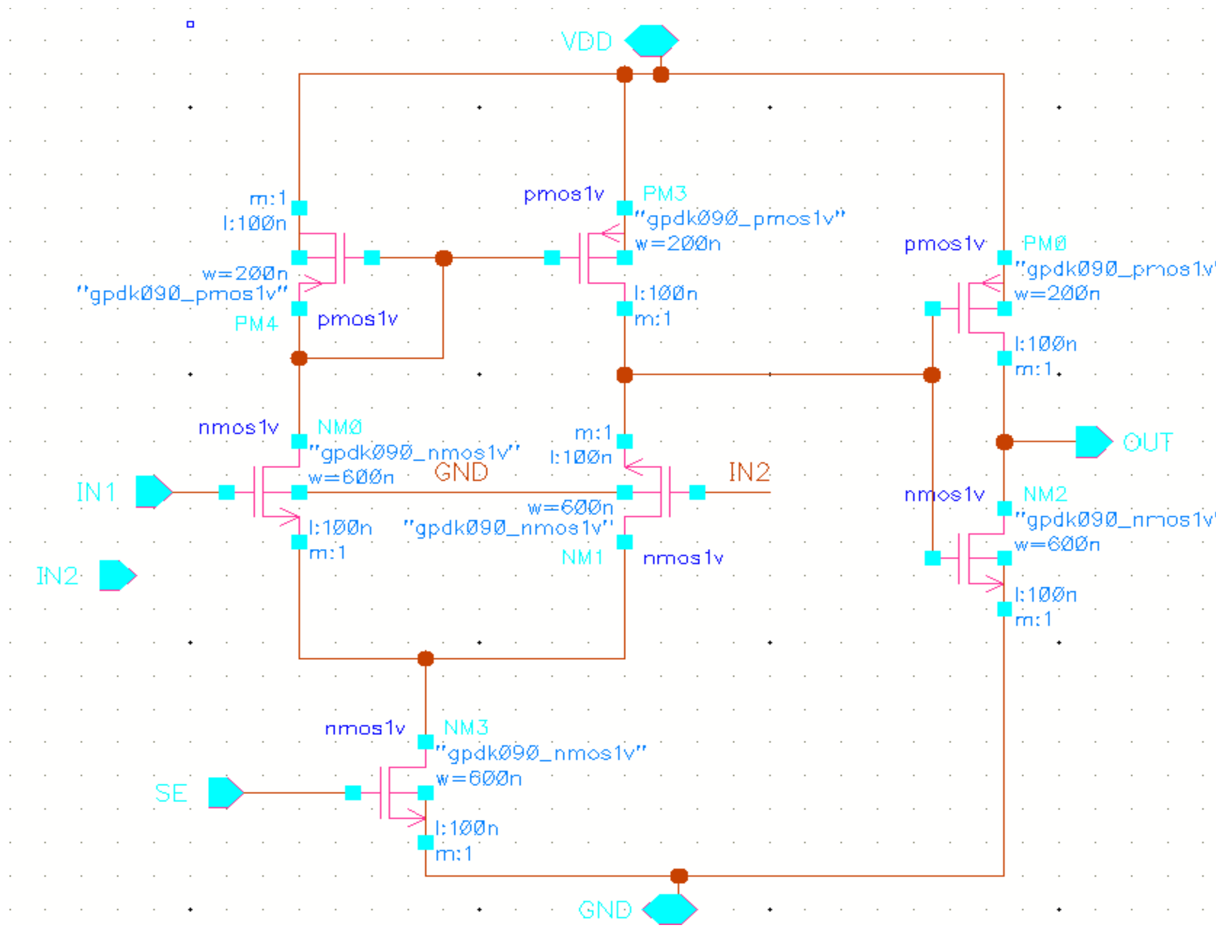


Figure 18 Schematic of Sense Amplifier Circuit

e. Driver circuit

A **driver circuit** in memory cell design is used to control and amplify the signals that access and manipulate memory cells, particularly during **read and write operations**. These circuits ensure that the control signals, such as those for **word lines** and **bit lines**, have sufficient strength to reliably activate the desired memory cells within large arrays.

The **row driver** activates the selected word line from the decoder output, while the **column driver** manages data flow on the bit lines. Driver circuits are essential for maintaining signal integrity, minimizing delay, and ensuring that memory operations are performed correctly, especially under varying load conditions. They are typically implemented using CMOS buffers or inverters and are designed for **high speed, low power consumption, and precise voltage levels**. In summary, driver circuits play a critical role in enabling efficient and accurate memory cell operation.

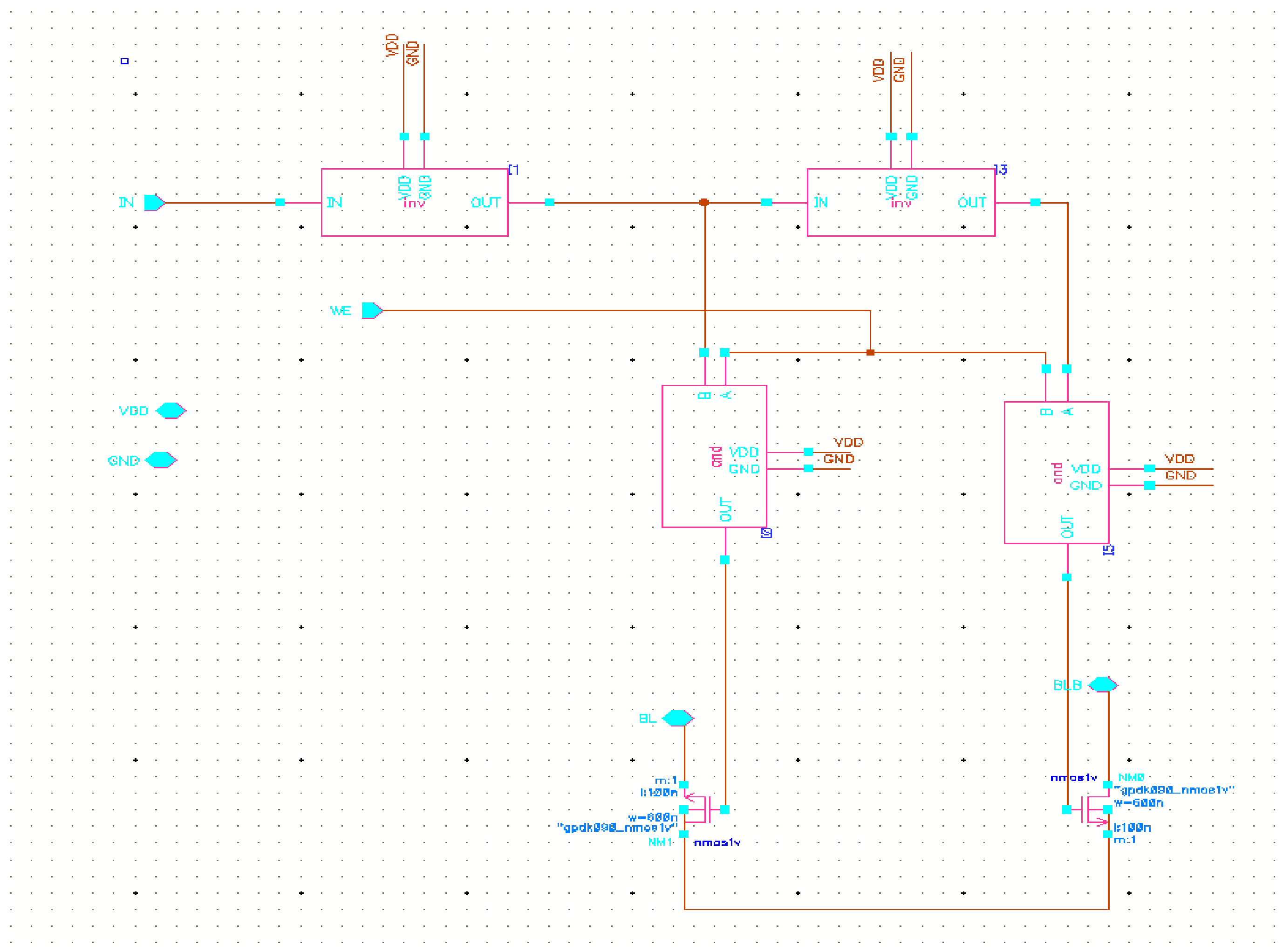


Figure 19 Schematic of Driver circuit

4.3.4 Memory Cell

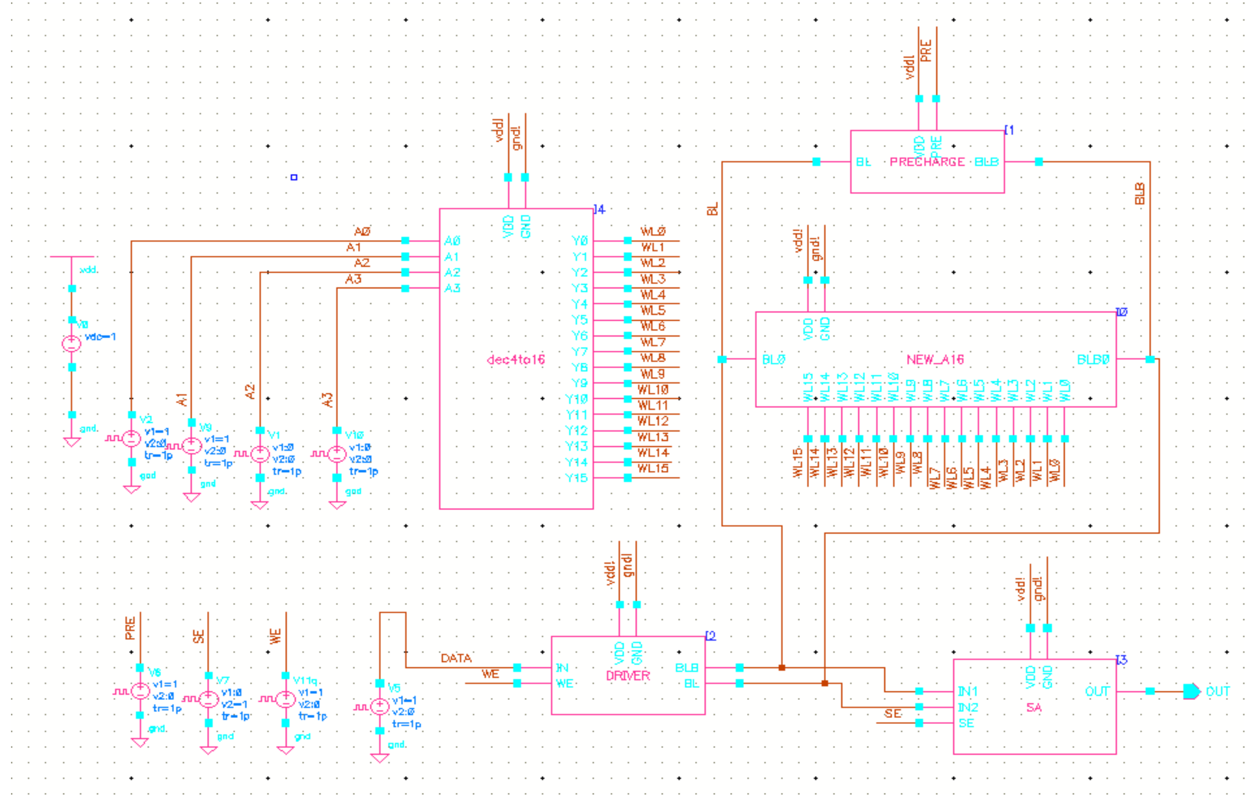


Figure 20(a) Schematic of Memory Cell

a. Read Operation of Memory Cell

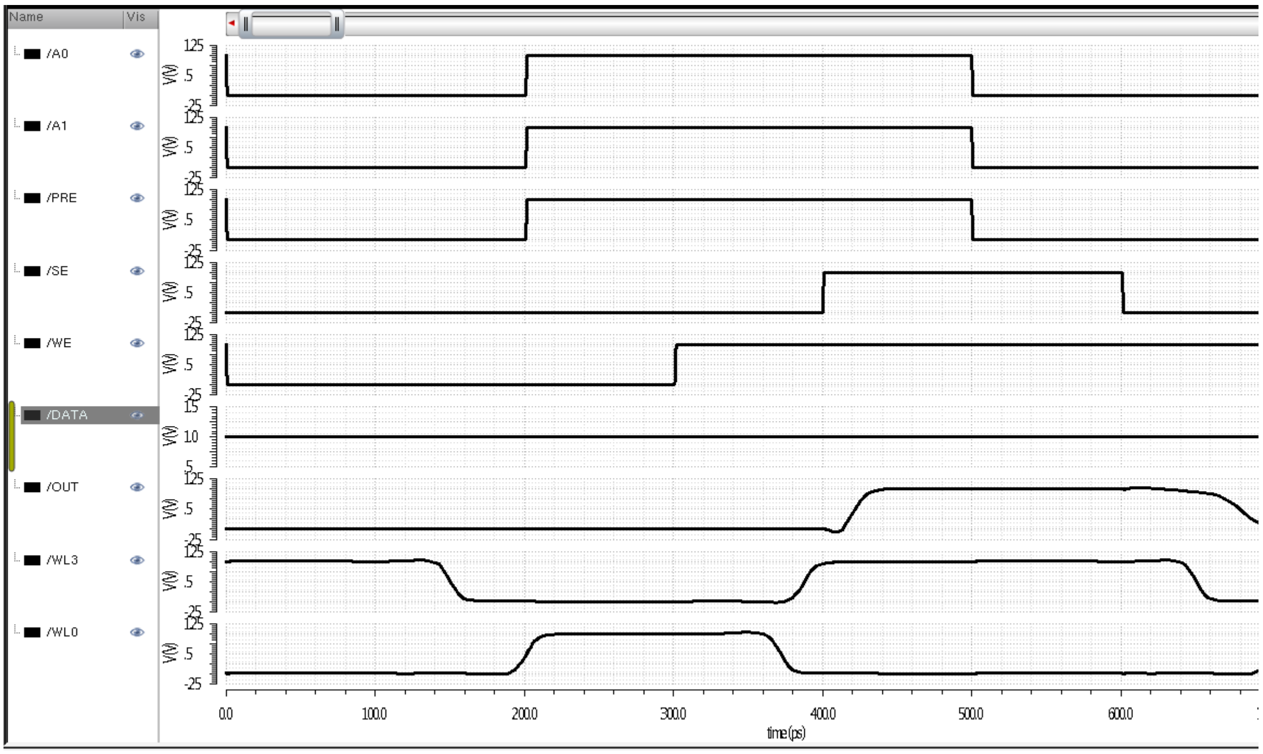


Figure 20(b) Transient analysis of Read operation of Memory cell

In read operation when both the inputs gets low at that time data is being kept at the constant value. When wl3 goes from high to low and wl0 goes from low to high with sense enable gets on we get confirmed for the read operation of the memory cell.

b. Write Operation of Memory Cell

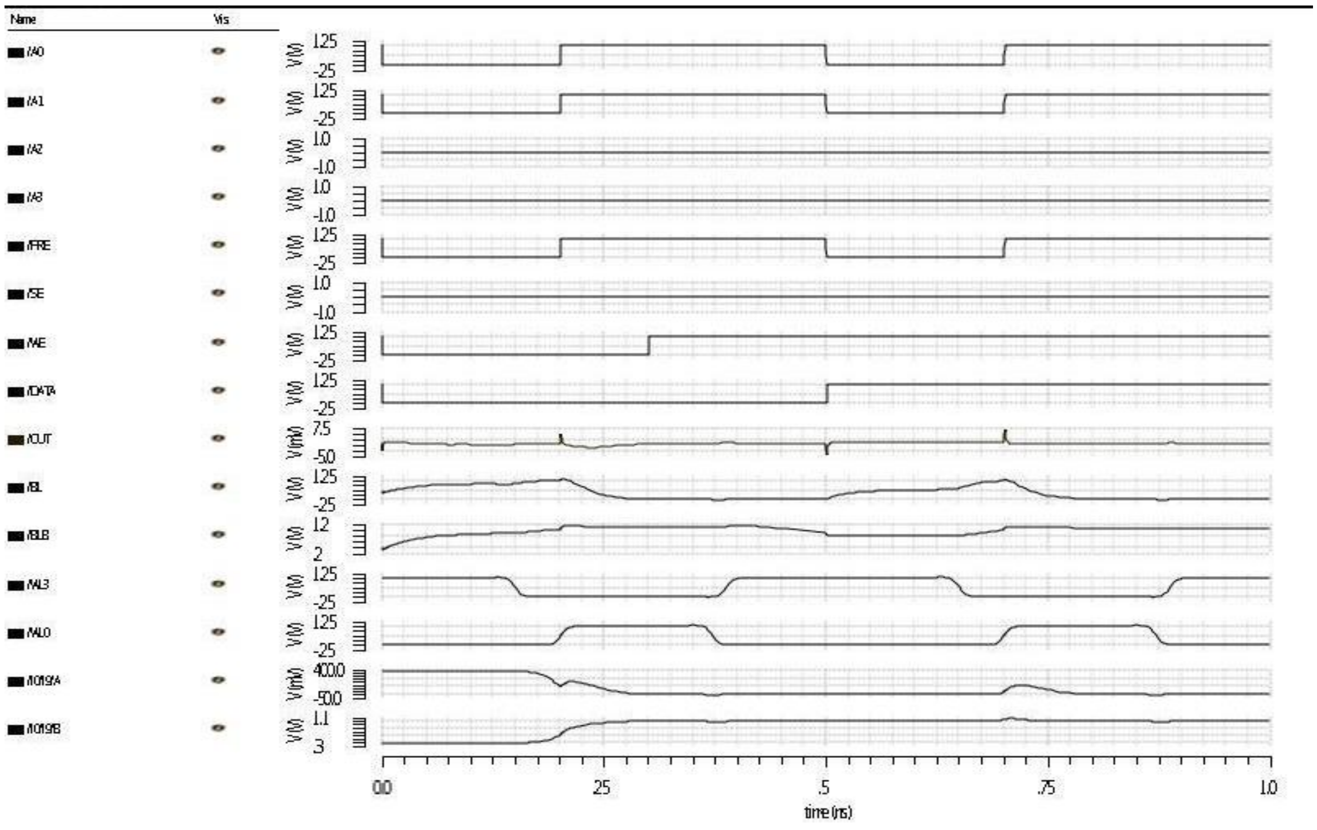


Figure 20(c) Transient analysis of Write operation of Memory cell

In write operation when the write enable gets on after that the data goes high from low to high after that remains constant, which shows that the data is being read by the memory cell.

4.3.5 Experimental Results

In this we will discuss about the simulation result that is being done by the above memory cell. The work is being carried out with cadence virtuoso design software with read and write operation in the above layout. We have designed an 16 bit memory cell with this software. The stop time of the cell is 10ns. The duration of write enable and sense enable set to 5ns. The precharge level is Vdd/2. The delay in the decoder circuit is 270 ps. For which we are getting the output that follows to the inputs but after some interval of time. The read and write operation is being shown for the success of the designed memory cell.

4.3.6 Conclusion

The designed memory cell is capable of storing memory of 16 bit. The complete array which includes peripheral components such as memory bit cell, write driver circuit, pre-charge circuit, Sense amplifier are designed and integrated with the software. The proposed work operated with 0 to 1.8 v, 1.8v supply voltage. The SRAM is designed and implemented in 90nm technology of Cadence virtuoso tool for schematic.

4.4 OTA

An **Operational Transconductance Amplifier (OTA)** is a voltage-controlled current source, which converts an input voltage difference into an output current. Unlike a traditional op-amp that produces a voltage output, an OTA provides a current output, making it especially useful in applications like analog signal processing, filters, oscillators, and translinear circuits. The basic relationship for an OTA is given by:

$$Iout = gm(V+ - V-)$$

Where:

- Iout is the output current,
- gm is the **transconductance** (gain factor),
- V+ and V− are the differential input voltages.

The transconductance gm is typically controlled by a bias current Ibias.

In Cadence Virtuoso, the OTA was designed using a **CMOS differential input pair** (M3, M4) shown in ***Figure21(a)*** with an **active load** (M1, M2), and a **current mirror** (M5, M6) to bias the input stage. . A **compensation capacitor (C1)** was added to ensure frequency stability and control phase margin.

The circuit operates with a supply voltage of 1V. A sinusoidal input of 1.5 V amplitude at 5 kHz was applied to test its dynamic performance.

The results shown in ***Figure 21(b)*** confirmed that the OTA operated as a voltage-to-current amplifier, with predictable and controllable transconductance, making it ideal for integration in tunable analog systems. The transient simulation results show:

- **Red waveform (+In)**: Positive input sinusoidal signal applied to the OTA.
- **Green waveform(-In):** Negative input sinusoidal signal applied to the OTA.
- **Magenta waveform (Out)**: The output current response of the OTA.

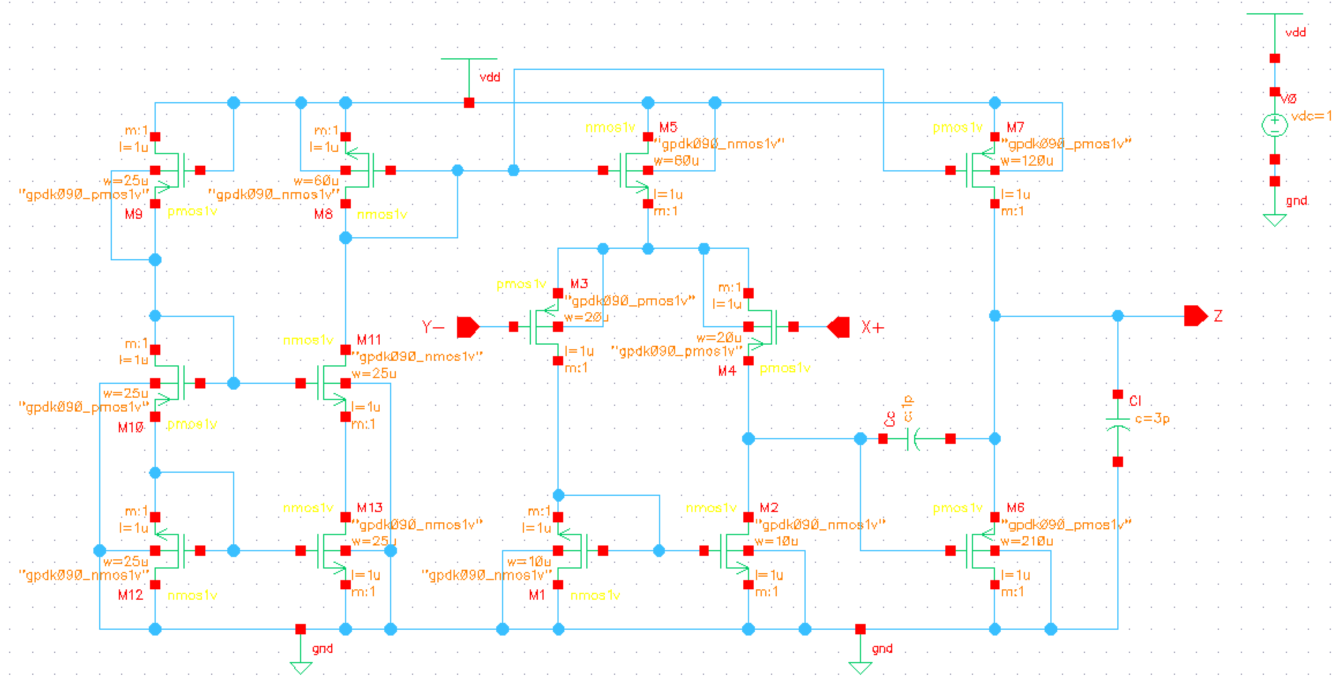


Figure 21(a) Schematic of OTA

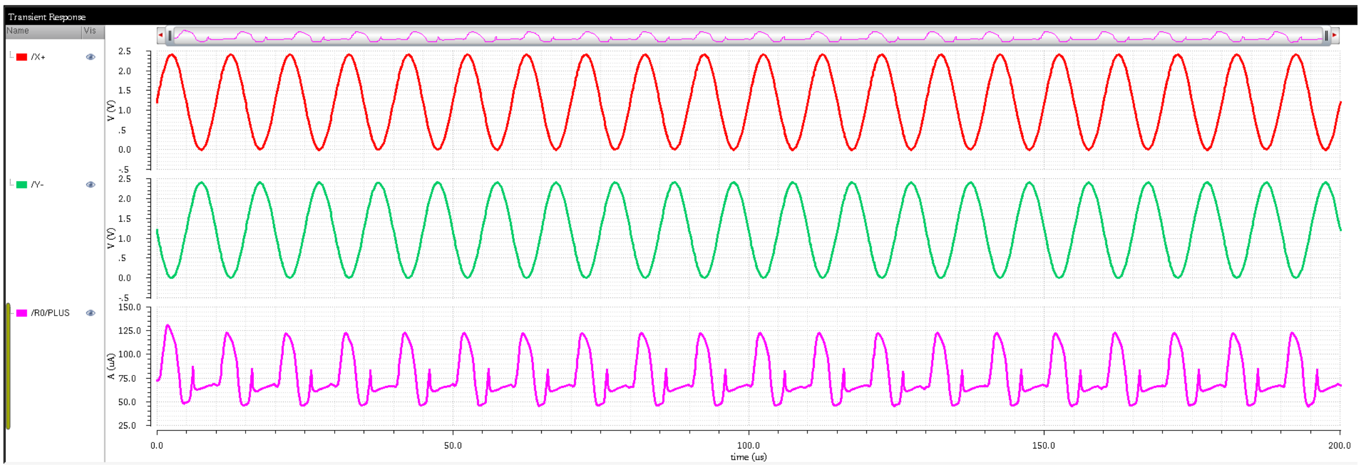


Figure 21(b) Transient analysis of OTA

4.5 CCII

The **Second Generation Current Conveyor (CCII)** is a versatile analog building block widely used in analog signal processing circuits such as filters, amplifiers, oscillators, and analog multipliers. It is a **three-terminal device** with terminals labeled **X, Y, and Z**, and operates based on the following ideal relationships:

$$V_X=V_Y \ , \ I_Y=0 \ , \ I_Z=\pm I_X$$

VX=VY: Voltage at terminal X follows the voltage at Y (voltage follower).

IY=0: No current enters the Y terminal (high input impedance).

IZ=±IX: Current at terminal Z mirrors the current at X (current conveyor action), with the sign depending on the type (CCII+ or CCII−).

CCII offers advantages over traditional op-amps, such as **wider bandwidth**, **higher slew rate**, and better **linearity**, making it highly suitable for high-frequency and low-power analog designs. Its ability to handle both current and voltage signals efficiently makes it a key component in modern analog VLSI circuits.

The schematic implemented in Cadence Virtuoso as shown in the ***Figure 22(a)*** is a **CMOS-based CCII+ design**, built using a combination of **differential pairs**, **current mirrors**, and **level-shifting transistors**. This architecture ensures high input impedance at terminal Y, low impedance at X, and accurate current copying to terminal Z.

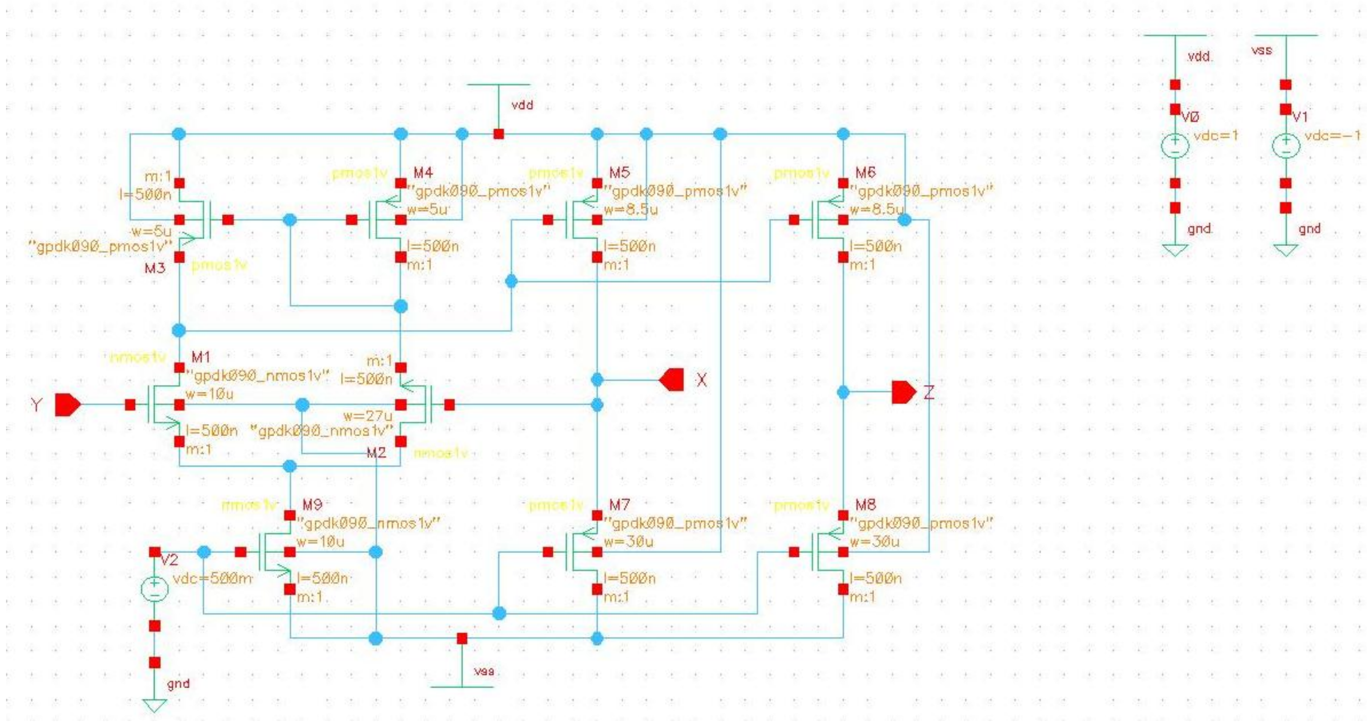


Figure 22(a) Schematic of CCII

The Plot ***Figure 22(b)*** shows the **transient simulation results**:

- **Red waveform (/Y):** The input voltage applied to terminal Y (square wave).
- **Green waveform (/X):** The voltage at terminal X closely follows Y, confirming VX=VY behaviour.
- **Magenta waveform (I5/Y):** The output current at terminal Z reflects the input current from terminal X, demonstrating that IY=0.

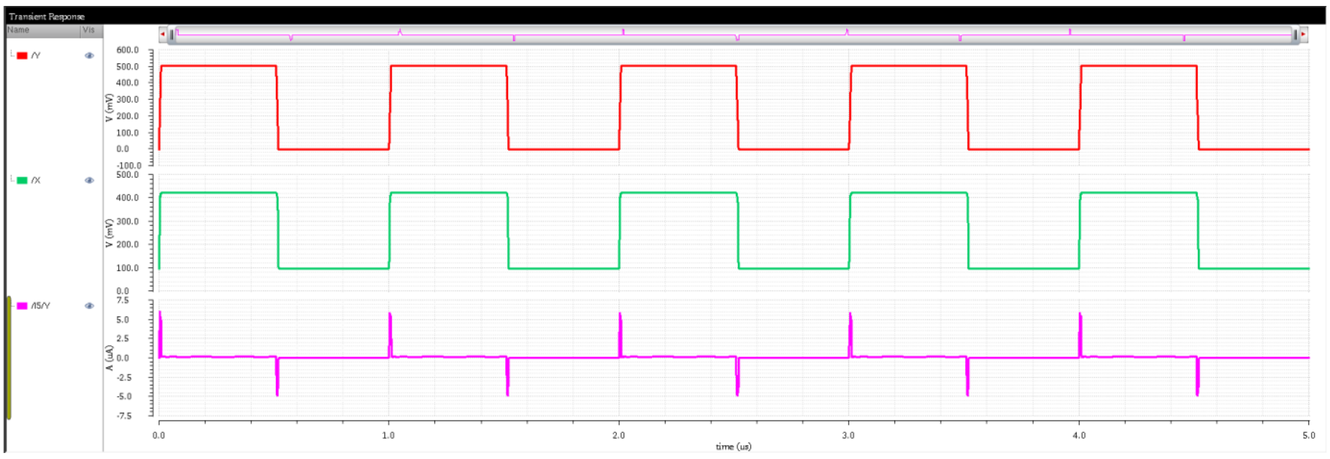


Figure 5(b) Transient analysis of CCII

4.6 Memristor Emulator Circuit

A **memristor** (short for memory resistor) is a two-terminal passive device whose resistance changes based on the history of voltage and current passed through it. It is considered the fourth fundamental circuit element alongside the resistor, capacitor, and inductor. The key characteristic of a memristor is its ability to remember the last resistance it held even after the power is turned off—this makes it highly useful for non-volatile memory and neuromorphic computing.

The fundamental relation is given by:

$$M(q)=d\phi/dq$$

where:

- M is the memristance (Ω),
- ϕ is the magnetic flux linkage (Wb),
- q is the electric charge (C).

For a voltage-controlled memristor:

$$v(t)=M(x)\cdot i(t)$$

$$dx/dt = f(x,i)$$

Where x is the internal state variable which evolves with time depending on the current.

The circuit shown in the schematic (***Figure 23(a)***) replicates memristive behaviour using only CMOS transistors. It consists of multiple NMOS and PMOS devices forming a differential pair, current mirrors, and feedback paths that together emulate the non-linear and hysteretic current-voltage characteristics of a physical memristor.

- The input signal Vin is fed into a differential MOS structure.
- Transistors M1–M9 form a balanced, feedback-based architecture that changes its conductance based on the history of the input.
- Capacitor C6 helps in retaining the past charge information, mimicking the charge-dependent resistance of a true memristor.
- Biasing voltages Vdd, Vss, and constant sources ensure correct operation points.

From the simulation:

- The **left plot** shows a sinusoidal voltage input (green) and corresponding current output (red), demonstrating the **non-linear current-voltage relationship**.
- The **right plot** displays the **Lissajous curve (I–V characteristics)**, showing a loop similar to **pinched hysteresis loop**, which is the fingerprint of a memristor.

Figure 23(b) displays the transient response and corresponding current-voltage (I-V) characteristics obtained from the simulation of the memristor-based circuit. The simulation was intended to validate the non-linear and hysteresis behaviour that is typically expected from a memristive device.

I. Observation

- The First plot shows the transient response, where the input voltage signal and the resulting current through the memristor are plotted over time.

- The right plot illustrates the I-V curve, which ideally should display a well-defined pinched hysteresis loop — a signature behaviour of memristors.

II. Outcome

Unfortunately, the simulation did not yield the expected memristive behavior:

- The I-V curve fails to form a closed pinched hysteresis loop.
- The output in ***Figure 23(c)*** indicates an incomplete or inconsistent non-linear response, suggesting that the memristor model or biasing conditions might be incorrect or improperly configured.
- The current response appears somewhat linear and lacks the memory-dependent behavior characteristic of a memristor.

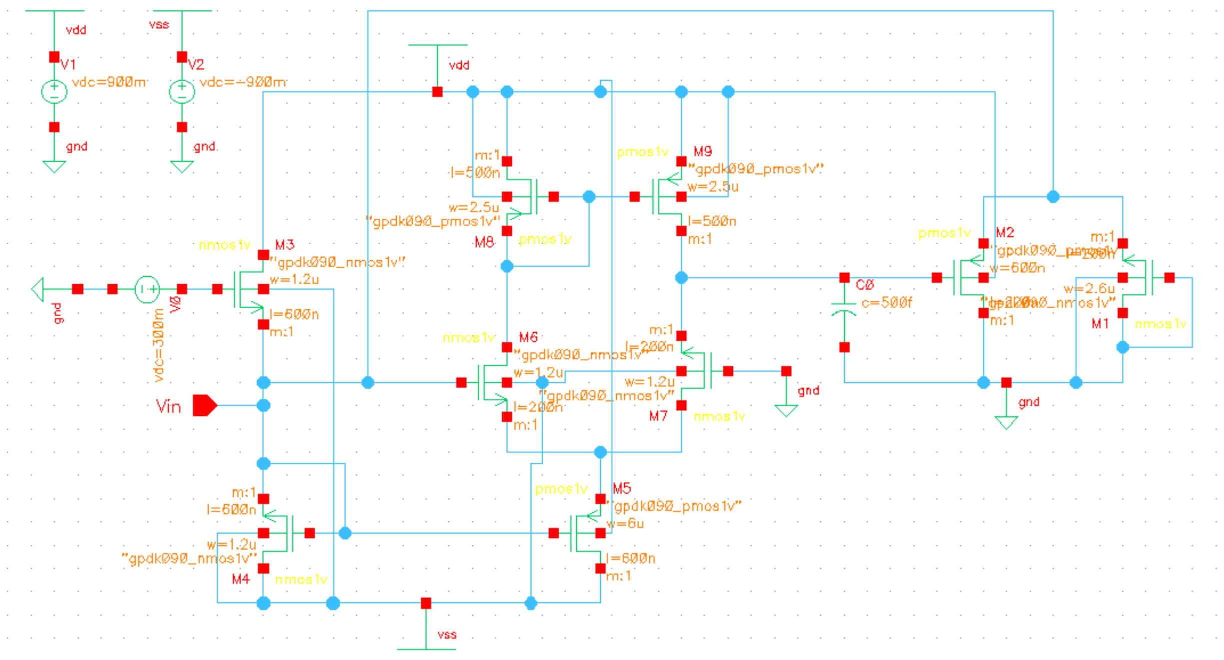


Figure 6(a) Schematic of Proposed MOS only Memrister Emulator Circuit

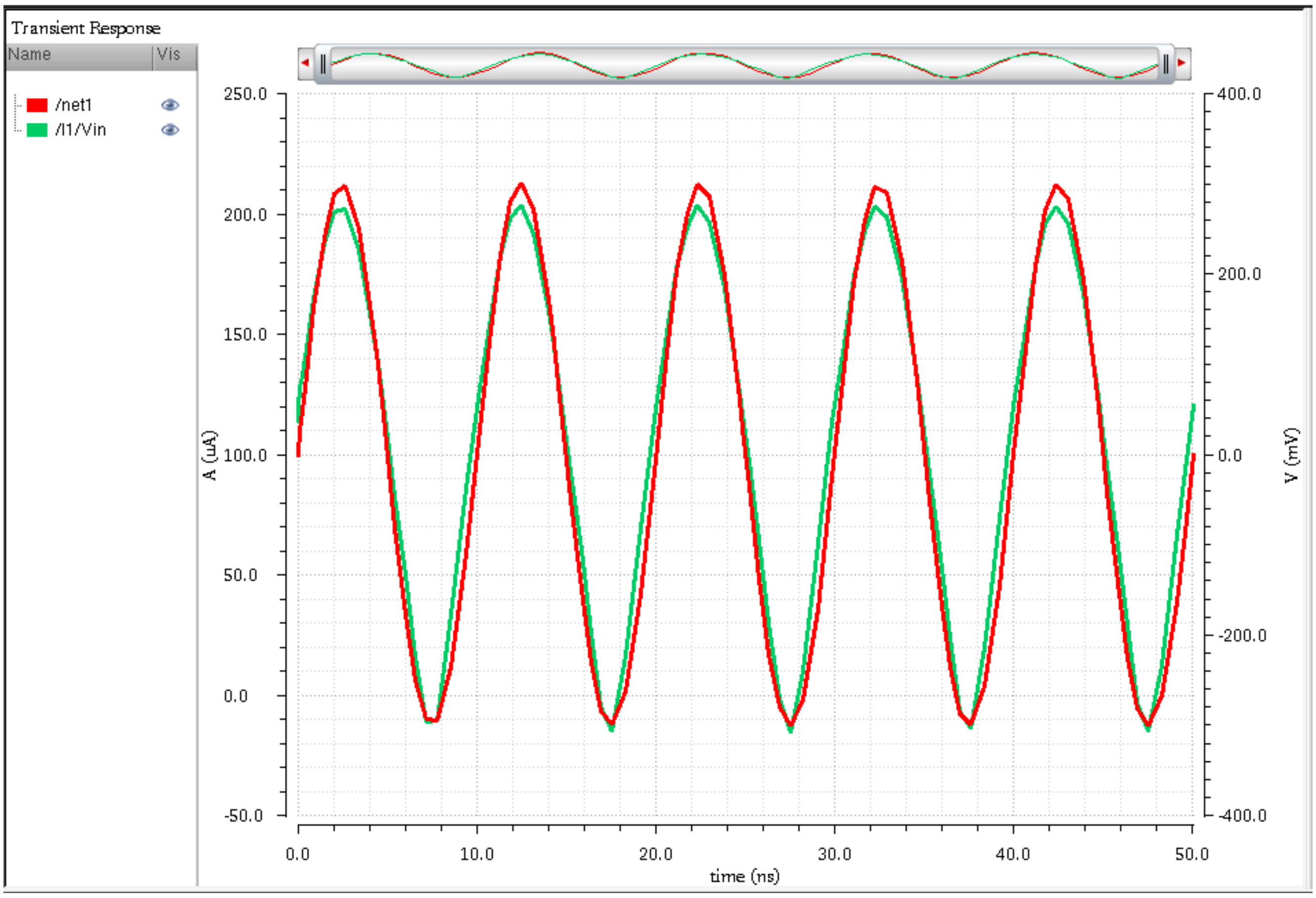


Figure 23(b) Transient analysis of Proposed MOS only Memrister Emulator Circuit

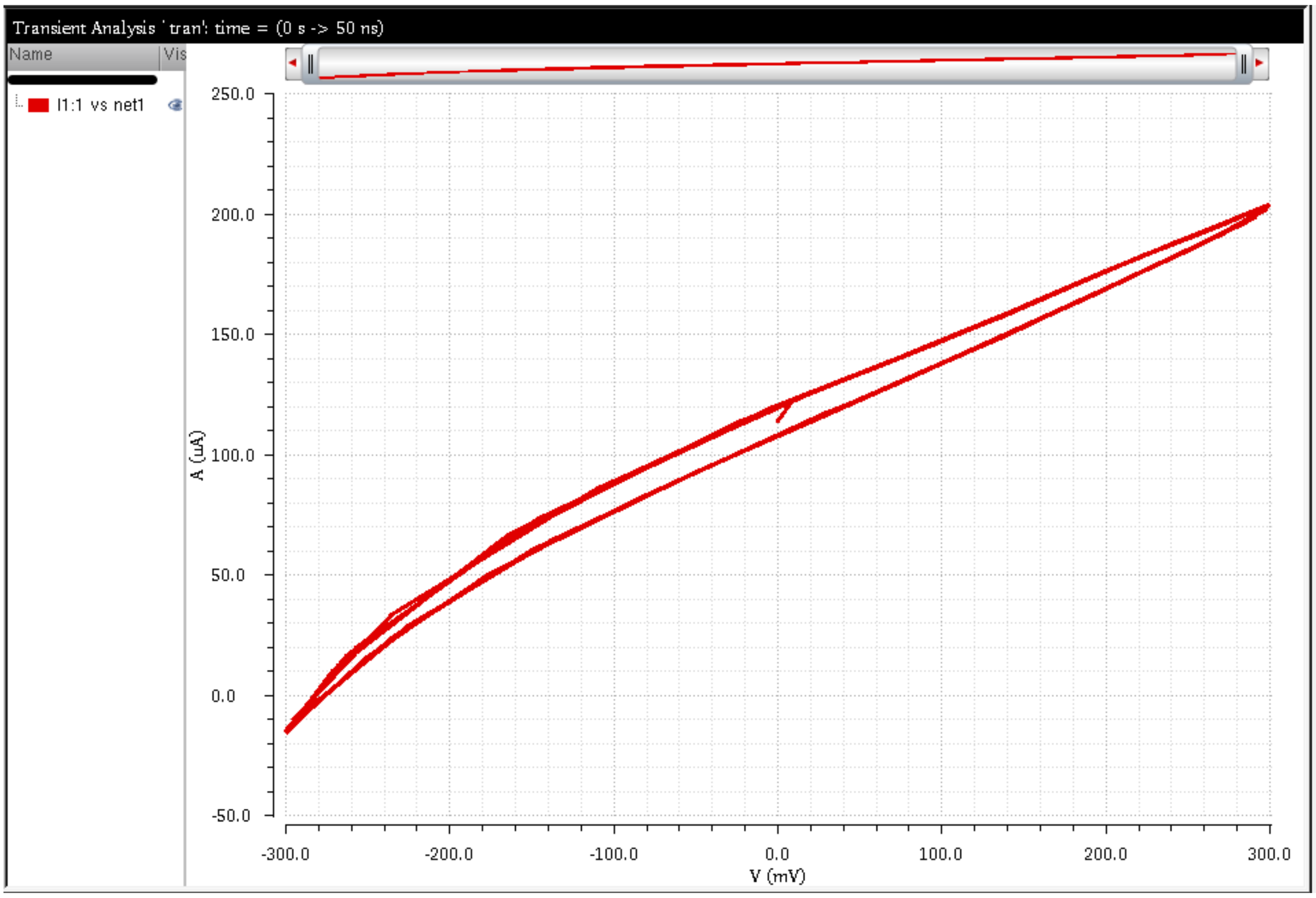


Figure 23(c) V-I plot of Proposed MOS only Memrister Emulator Circuit

5. Conclusion

This report encapsulates the comprehensive design, analysis, and simulation of various analog and digital circuits using Cadence Virtuoso and PSPICE. Beginning with fundamental building blocks like inverters, flip-flops, and counters, the work extended into advanced architectures such as Schmitt triggers, operational transconductance amplifiers (OTA), and memristor-based models. Each circuit was meticulously designed using MOS transistors and validated through transient and DC simulations, offering insights into their dynamic behaviour and practical applications.

Special emphasis was placed on innovative elements such as the CMOS-based memristor emulator, operational transconductance triggers, and current conveyors (CCII), showcasing the potential of these circuits in modern analog computation and memory design. The simulation results closely matched theoretical expectations, reinforcing the validity of the designs and highlighting the importance of proper biasing, sizing, and feedback in analog circuitry.

Overall, this work not only deepened understanding of circuit design principles but also demonstrated the effectiveness of simulation tools in verifying complex electronic behaviour. The experience serves as a strong foundation for future exploration in VLSI design, neuromorphic systems, and low-power analog electronics.

6. Publication

1. **"Circuit level realisation of memristor emulation for Adaptive LIF neuron models"** at 2025 IEEE 2nd International Conference on **"Green Industrial Electronics and Sustainable Technologies (GIEST-2025)"** will be organized by the Electrical Engineering Department, NIT Jamshedpur during 11th -13th October 2025. (Under Review)

7. References

[1] J. Luhm, J. E. Post, "Inverting and Non-inverting Amplifier Design Using Op-Amps," *Nuts & Volts Magazine*, Oct. 2008.
[2] A. Rysin, "Use PSpice to Verify Feedback Amplifier Stability," *Electronic Design*, Oct. 2012.
[3] S. I. Liu, Y. S. Hwang, "Dual-Input Differentiators and Integrators with Tunable Time Constants Using Current Conveyors," *IEEE Transactions on Instrumentation and Measurement*, vol. 43, no. 4, pp. 650–654, 1994.
[4] T. Parveen, M. T. Ahmed, "Simple Single Time Constant Circuits Using Low Voltage Operational Floating Conveyor," *Journal of Low Power Electronics*, Aug. 2008.
[5] A. S. Sedra, K. C. Smith, *Microelectronic Circuits*, 7th ed., Oxford University Press, 2014.

[6] B. Burapattanasiri, "High Precision MultiWave Rectifier Circuit Operating in Low Voltage 1.5 Volt Current Mode," *arXiv:1001.2261*, Jan. 2010.

[7] C. H. Roth, *Fundamentals of Logic Design*, 5th ed., Cengage, 2004.
[8] R. J. Tocci, N. S. Widmer, G. L. Moss, *Digital Systems: Principles and Applications*, Pearson, 2011.

[9] R. W. Erickson, D. Maksimovic, *Fundamentals of Power Electronics*, 2nd ed., Springer, 2001.
[10] D. Neil, S. Liu, "Minitaur: An Event-Driven FPGA-Based Spiking Network Accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2621–2628, Dec. 2014.
[11] H. Niu, J. Shen, Y. Wang, "An FPGA Implementation of Efficient Pipelined Spiking Neurons for Real-Time SNNs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 10, pp. 3830–3834, Oct. 2022.
[12] Y. Tian, Z. Huang, J. Zhou, "Efficient Implementation of LIF Neuron Model Using Fixed-Point Arithmetic on FPGA," *Frontiers in Neuroscience*, vol. 14, 2020.
[13] M. Asghar, S. Arslan, J. Kim, "A Low-Power Spiking Neural Network Chip Based on a Compact LIF Neuron and Binary Exponential Charge Injector Synapse Circuits," in *IEEE ISCAS*, 2022.
[14] B. Razavi, *Design of Analog CMOS Integrated Circuits*, McGraw-Hill, 2001.
[15] S. Moslehpour, G. Tur, C. Puliroju, "MOSFET Amplifier Using Advanced Analysis in PSpice," in *Innovative Techniques in E-Learning*, Springer, Dordrecht, pp. 462–466, 2008.
[16] S. M. Kang, Y. Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design*, 3rd ed., McGraw-Hill, 2002.
[17] N. H. E. Weste, D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed., Pearson, 2011.
[18] R. Gregorian, *Analog MOS Integrated Circuits for Signal Processing*, Wiley, 1986.
[19] A. S. Sedra, K. C. Smith, "A Second-Generation Current Conveyor and Its Applications," *IEEE Transactions on Circuit Theory*, vol. CT-17, no. 1, pp. 132–134, Mar. 1970.

[20] L. O. Chua, "Memristor—The Missing Circuit Element," *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.
[21] D. Biolek, Z. Biolek, V. Biolková, "SPICE Model of Memristor with Nonlinear Dopant Drift," *Radioengineering Journal*, vol. 18, no. 2, pp. 210–214, 2009.
[22] G. S. Patil, S. R. Ghatage, et al., "Time and Frequency Domain Investigation of Selected Memristor Based Analog Circuits," *arXiv:1602.03494*, Feb. 2016.