eSim Summer Fellowship 2026

On

Development of a Cadence Spectre to eSim/Ngspice Schematic and Netlist Converter

Submitted by

Deepika

B.E Electronics and Communication Engineering
Bannari Amman Institute of Technology

Under the guidance of

Prof. Prabhu Ramachandran
Principal Investigator
Department of Aerospace Engineering
Indian Institute of Technology Bombay

July 25, 2026

Acknowledgment

I express my sincere gratitude to Prof. Prabhu Ramachandran for providing me with the opportunity to be part of the FOSSEE internship program and for his continued efforts in promoting open-source engineering tool development. His leadership and vision have been instrumental in fostering meaningful student participation in the open-source ecosystem.

I also acknowledge Prof. Kannan M. Moudgalya for his foundational role in establishing and strengthening the FOSSEE initiative. His contributions to open-source education and the development of the FOSSEE fellowship framework have been pivotal in creating the academic and organisational platform through which this internship was undertaken.

My sincere appreciation extends to my mentor, Sumanto Kar, for his continual support, technical guidance, and encouragement throughout the duration of this project. His insights and feedback played a key role in refining ideas, overcoming challenges, and ensuring timely completion of the tasks assigned to me.

I would also like to thank my internal mentors, Mr. Varad Patil and Ms. Shanthi Priya K for their valuable guidance, coordination, and technical inputs during the internship. Their mentorship contributed significantly to the clarity, progress, and successful execution of the work.

This internship provided valuable hands-on experience in open-source EDA tools and IC design workflows. I gained practical knowledge of bridging open-source and commercial EDA environments through schematic and netlist conversion, netlist generation, and Python-based automation. Working on circuit modeling, simulation, and verification strengthened my technical, analytical, and problem-solving skills.

I would also like to thank the entire FOSSEE team for their coordination, assistance, and timely interactions at various stages of this work. Their collective efforts ensured smooth workflow, resource accessibility, and effective project execution.

Abstract

This report presents the development of an automated Cadence Spectre to eSim/Ngspice netlist converter, undertaken as part of the FOSSEE Internship 2026 at IIT Bombay. The project addresses a critical gap in the EDA (Electronic Design Automation) ecosystem: the lack of any tool to convert circuits designed in Cadence Virtuoso, the industry standard proprietary EDA platform, into a format compatible with eSim, the open source EDA tool developed by FOSSEE, IIT Bombay.

The converter was implemented in Python and successfully demonstrated on a CMOS inverter circuit designed using the gpdk090 Process Design Kit (PDK) in Cadence Virtuoso. The tool parses Cadence Spectre netlist files (.scs format), extracts component information using regular expressions, maps proprietary model names to open source Ngspice compatible models, and generates a complete Ngspice simulation file (.cir format). The generated file was successfully simulated in eSim, producing correct transient waveforms confirming inverter behavior.

The converter reduces conversion time from 2 to 4 hours (manual process) to under 1 second, requires zero manual intervention, and correctly converts all component types including NMOS, PMOS, and voltage sources. This tool has significant educational impact, enabling thousands of students who use Cadence in academic labs to continue their work using the freely available eSim platform.

Contents

Chapter 1

Introduction

1.1 FOSSEE and Its Mission

FOSSEE (Free/Libre and Open Source Software for Education) is a project under the National Mission on Education through Information and Communication Technology (NMEICT), Ministry of Human Resource Development, Government of India. FOSSEE is hosted at IIT Bombay and is guided by Prof. Kannan M. Moudgalya of the Chemical Engineering Department. The primary mission of FOSSEE is to promote the use of open source software tools in educational institutions across India, thereby reducing dependency on expensive proprietary software. FOSSEE achieves this through:

- Developing and maintaining open source EDA tools like eSim
- Conducting internships, workshops and fellowships to train students in open source tools
- Creating educational content and tutorials for open source software, and research activities to help students and educators
- Bridging the gap between industry grade tools and accessible academic tools
- Supporting the replacement of proprietary software in college curricula

1.2 eSim: Open Source EDA Tool

eSim is a free and open source EDA (Electronic Design Automation) tool developed by FOSSEE at IIT Bombay. It is built on a foundation of open source components:

- **KiCad**: Provides the schematic editor (EEschema) and PCB layout editor (Pcbnew)
- **Ngspice**: The open source SPICE circuit simulator for analog and mixed signal simulation
- **GHDL**: VHDL simulator for digital circuit components
- **Makerchip IDE**: For SystemVerilog and TL-Verilog digital simulation
- **Python**: Glue language connecting all components

eSim provides capabilities comparable to commercial tools like OrCAD, Xpedition, and HSPICE, but at zero cost. This makes it ideal for educational institutions and small enterprises that cannot afford expensive EDA licenses.

1.3 Cadence Virtuoso

Cadence Design Systems is one of the world's largest Electronic Design Automation companies. Cadence Virtuoso is their flagship platform for analog and custom IC design. It is the industry standard tool used by semiconductor companies, research institutions, and well equipped academic labs worldwide.

The Cadence Spectre simulator uses a proprietary netlist format (.scs) that is fundamentally different from the standard SPICE format used by open source tools. This format difference is the root cause of the incompatibility that this project addresses.

Cadence licenses cost several lakhs of rupees per year per seat, making it inaccessible for most engineering colleges in India.

1.4 The Problem: The Cadence eSim Gap

A critical gap exists between Cadence Virtuoso and eSim. This gap affects thousands of engineering students and researchers across India:

Students design circuits in Cadence at their college lab and want to simulate them at home using eSim

Researchers with existing Cadence designs want to share them with collaborators who only have eSim

Engineers moving from industry (Cadence) to academia (eSim) lose access to their existing designs

No automated tool exists to convert between the two formats

The traditional manual conversion process involves opening the Cadence netlist in a text editor, removing all Cadence specific syntax, manually fixing component prefixes, adding model definitions, setting up analysis commands, and debugging errors, a process that takes 2 to 4 hours per circuit and requires expert knowledge of both Spectre and SPICE syntax.

1.5 Project Objective

The objective of this project is to develop an automated Python based converter tool that:

Reads Cadence Spectre netlist files (.scs format) exported from Cadence Virtuoso ADE L

Parses all circuit components including MOSFETs and voltage sources

Maps proprietary Cadence model names to open source Ngspice equivalents

Generates complete, simulation ready Ngspice .cir files

Requires zero manual intervention or editing of the output

Runs in seconds, making the conversion workflow practical for everyday use

Chapter 2

Literature Survey

2.1 Existing EDA Tools Comparison

Several EDA tools exist for circuit design and simulation, ranging from fully commercial to fully open source. Table 2.1 provides a comparison.

Table 2.1: Comparison of EDA Tools

Tool	Type	Simulator	Cost	Used In
Cadence Virtuoso	Commercial	Spectre	Very High	Industry, Top Institutes
eSim	Open Source	Ngspice	Free	Education, India
LTspice	Free	SPICE	Free	Hobbyists, Education
OrCAD PSpice	Commercial	PSpice	High	Industry, PCB design
KiCad	Open Source	Ngspice	Free	PCB design, Education
Xschem	Open Source	Ngspice	Free	IC design, Academia
Synopsys HSPICE	Commercial	HSPICE	Very High	Industry, Foundries

2.2 Existing Conversion Tools

A survey of existing tools for EDA format conversion reveals:

KiCad Import Tools: KiCad supports importing from Altium Designer, Eagle, and EasyEDA, but has no support for Cadence Virtuoso formats.

PySpice: A Python library that generates SPICE netlists programmatically but does not parse Cadence netlists.

Schemato (2024): An LLM based tool for converting netlists to LTspice format. Targets LTspice, not eSim, and does not handle Cadence Spectre syntax.

Cadence SKILL Scripts: Internal automation within the Cadence environment only, cannot output eSim compatible files.

CDL Exporters: Cadence can export CDL netlists, but these require significant manual modification before working in Ngspice.

FOSSEE Migration Guides: Manual step by step guides that require significant user effort and SPICE expertise.

2.3 Tools Used

eSim. eSim is an open source Electronics Design Automation (EDA) tool developed by the FOSSEE project at IIT Bombay. It is used for schematic design, circuit simulation, waveform analysis and PCB design.

Ngspice. Ngspice is an open source SPICE based circuit simulator used for analog, digital and mixed signal simulations. It is one of the main simulation engines integrated with eSim.

Virtuoso. Cadence Virtuoso is a commercial EDA platform developed by Cadence Design Systems, used for analog and custom IC design. It supports schematic capture, simulation, and layout. Its Analog Design Environment (ADE L) uses the Spectre simulator, which exports netlists in a proprietary .scs format. In this internship, Virtuoso was used to design the CMOS inverter using the gpdk090 PDK, whose exported netlist served as input to the converter.

LaTeX and Overleaf. LaTeX is a document preparation system widely used for technical and scientific documentation. Overleaf is an online LaTeX editor that provides a collaborative environment for creating professional reports and research papers. In this internship, Overleaf was used for preparing and formatting the internship report.

2.4 Research Gap

Based on the literature survey, a clear research gap exists:

No automated tool converts Cadence Spectre netlists to eSim/Ngspice format

No published paper addresses this specific conversion problem

The gap affects a large user base: all students using Cadence who want to migrate to eSim

The technical barriers (proprietary formats, model incompatibility) have prevented community solutions

This project fills this gap by developing the **first automated Cadence Spectre to eSim netlist converter**, making it a novel and practically valuable contribution to the open source EDA community.

Chapter 3

Problem Statement

"Develop an automated Python based tool that converts Cadence Spectre netlists (.scs format, exported from Cadence Virtuoso) to eSim/Ngspice compatible simulation files (.cir format), enabling engineers and students to simulate their Cadence designed circuits in eSim without any manual editing or SPICE expertise."

3.1 Specific Requirements

Input: Cadence Spectre format netlist (.scs) exported from Cadence Virtuoso ADE L

Output: Valid Ngspice .cir file that simulates correctly in eSim without modification

Handle Spectre specific syntax: continuation lines (\), parenthesized values such as w=(120n), comment style (//)

Parse NMOS transistors: extract drain, gate, source, bulk nodes and W, L parameters

Parse PMOS transistors: same as NMOS with pmos model detection

Parse voltage sources: detect DC and transient types

Map gpdk090 MOSFET models to equivalent Ngspice BSIM Level-1 models

Auto generate simulation analysis commands (.tran, .control block)

Zero manual editing required after conversion

Compatible with Python 2.7 and Python 3.x

3.2 Test Circuit: CMOS Inverter

The converter was designed and tested using a CMOS inverter circuit. The CMOS inverter consists of:

PMOS transistor (PM0): gpdk090 pmos1v, W = 120nm, L = 100nm

NMOS transistor (NM0): gpdk090 nmos1v, W = 120nm, L = 100nm

VDD supply (V1): 1.8V DC voltage source

Input signal (V0): Pulse voltage source for transient analysis

The CMOS inverter provides an ideal test case because it exercises MOSFET parsing, model mapping, and transient simulation, the three core functions of the converter.

Chapter 4

Technical Background

4.1 Cadence Spectre Netlist Format (.scs)

Cadence Spectre uses a proprietary netlist format. Table 4.1 compares Spectre and SPICE syntax.

Table 4.1: Cadence Spectre vs Ngspice Syntax Comparison

Feature	Cadence Spectre	Ngspice/SPICE
Comments	// comment	* comment
MOSFET prefix	NM0 (no M prefix)	MNM0 (M required)
Node syntax	NM0 (net0 net1 0 0)	MNM0 net0 net1 0 0
Parameter values	w=(120n) l=(100n)	W=120n L=100n
Long lines	Line ends with \	Line continues with +
DC source	V1 (net3 0) vsource type=dc	V1 net3 0 DC 1.8
Simulator directive	simulator lang=spectre	Not applicable

The exported Cadence Spectre netlist (input.scs) has the following structure:

Listing 4.1: Cadence Spectre Netlist (input.scs)

```
simulator lang=spectre
global 0

// Library name: demo1
// Cell name: inverter
// View name: schematic
NM0 (net0 net1 0 0) gpdk090_nmos1v w=(120n) l=(100n)
PM0 (net0 net1 net3 net3) gpdk090_pmos1v w=(120n) l=(100n)
V1 (net3 0) vsource type=dc dc=1.8
V0 (net1 0) vsource type=pulse val0=0 val1=1.8 period=10n \
    rise=5n fall=5n width=100p delay=100p
```

4.2 Ngspice/eSim Netlist Format (.cir)

Ngspice follows the Berkeley SPICE3 standard. A valid Ngspice netlist for the CMOS inverter follows the general structure shown below.

Listing 4.2: Target Ngspice Netlist Format

```
* SPICE netlist template
.model NMOS_gpdk090 nmos level=1 VT0=0.5 KP=270e-6 GAMMA=0.5 PHI=0.9
    LAMBDA=0.02
.model PMOS_gpdk090 pmos level=1 VT0=-0.5 KP=90e-6 GAMMA=0.5 PHI=0.9
    LAMBDA=0.02

MNM0 net0 net1 0 0 NMOS_gpdk090 W=120n L=100n
MPM0 net0 net1 net3 net3 PMOS_gpdk090 W=120n L=100n
V1 net3 0 DC 1.8
V0 net1 0 PULSE(0 1.8 0 100p 100p 5n 10n)

.tran 100p 30n
.control
run
plot v(net1) v(net0)
.endc
.end
```

4.3 The gpdk090 PDK

The Generic Process Design Kit 90nm (gpdk090) is a technology independent educational PDK provided by Cadence. Table 4.2 shows the model parameters used.

Table 4.2: gpdk090 Equivalent Ngspice Model Parameters

Parameter	**NMOS gpdk090**	**PMOS gpdk090**
Model Level	Level = 1 (BSIM)	Level = 1 (BSIM)
Threshold Voltage	VT0 = 0.5V	VT0 = −0.5V
Transconductance	KP = 270 μA/V^2	KP = 90 μA/V^2
Body Effect	GAMMA = 0.5	GAMMA = 0.5
Surface Potential	PHI = 0.9V	PHI = 0.9V
Channel Modulation	LAMBDA = 0.02	LAMBDA = 0.02
Width (test)	W = 120nm	W = 120nm
Length (test)	L = 100nm	L = 100nm

4.4 CMOS Inverter Operation

The CMOS inverter is the fundamental building block of digital logic:

When VIN = HIGH (1.8V): NMOS turns ON, PMOS turns OFF ⟶ VOUT = LOW (0V)

When VIN = LOW (0V): PMOS turns ON, NMOS turns OFF ⟶ VOUT = HIGH (1.8V)

This inverting behavior validates the converter output. A correctly converted and simulated CMOS inverter should show the output waveform as the complement of the input.

Chapter 5

Implementation

5.1 System Architecture

The converter consists of three logical modules:

Module 1, Parser: Reads the .scs file, joins continuation lines, filters Cadence directives, and extracts components using regular expressions

Module 2, Mapper: Maps Cadence model names to Ngspice equivalents and translates voltage source syntax

Module 3, Generator: Assembles the complete .cir file with models, components, analysis, and the control block

5.2 Implementation Environment

Table 5.1: Implementation Environment

Component	Details
Design Machine	Linux Virtual Machine (CentOS)
EDA Tool (Input)	Cadence Virtuoso with gpdk090 PDK
Simulation Machine	Windows 10/11 PC
EDA Tool (Output)	eSim 2.5 (FOSSEE, IIT Bombay)
Programming Language	Python 2.7 (compatible with Python 3)
Libraries Used	Python re (regex), os (file operations)
Circuit Under Test	CMOS Inverter (gpdk090, W=120n, L=100n)
Simulator	Ngspice (via eSim)
Analysis Type	Transient (.tran 100p 30n)
Supply Voltage	1.8V

5.3 Step-by-Step Implementation

5.3.1 Step 1: Setting Up Working Directory

```
mkdir -p ~/cadence_to_esim
cd ~/cadence_to_esim
```

5.3.2 Step 2: Exporting Netlist from Cadence

Open inverter schematic in Virtuoso Schematic Editor L
Launch ⟶ ADE L (Analog Design Environment)
Setup ⟶ Simulator/Directory/Host ⟶ Select ”spectre” ⟶ OK
Simulation ⟶ Netlist ⟶ Create

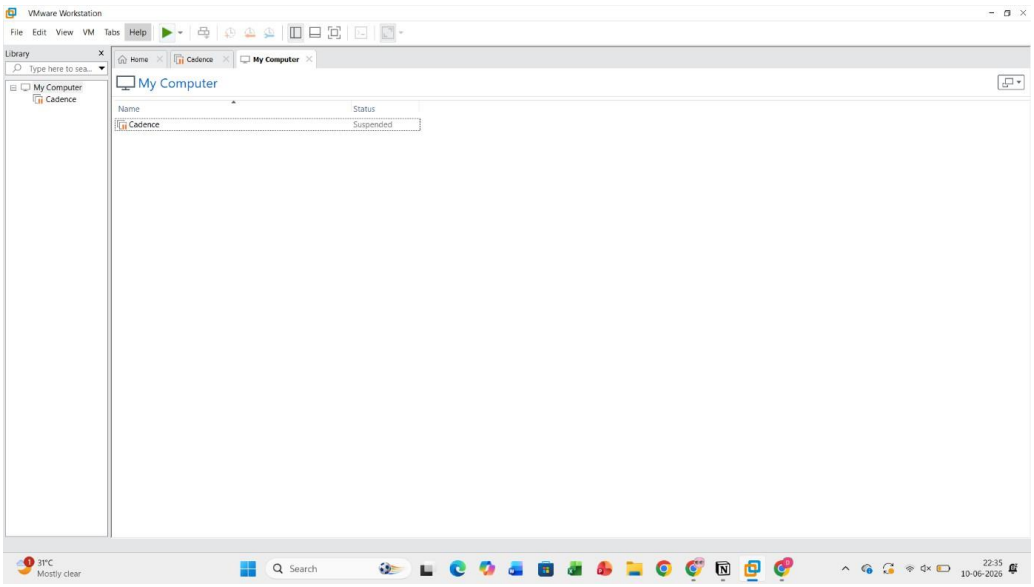

Figure 5.1: Cadence Virtuoso running inside the Linux virtual machine

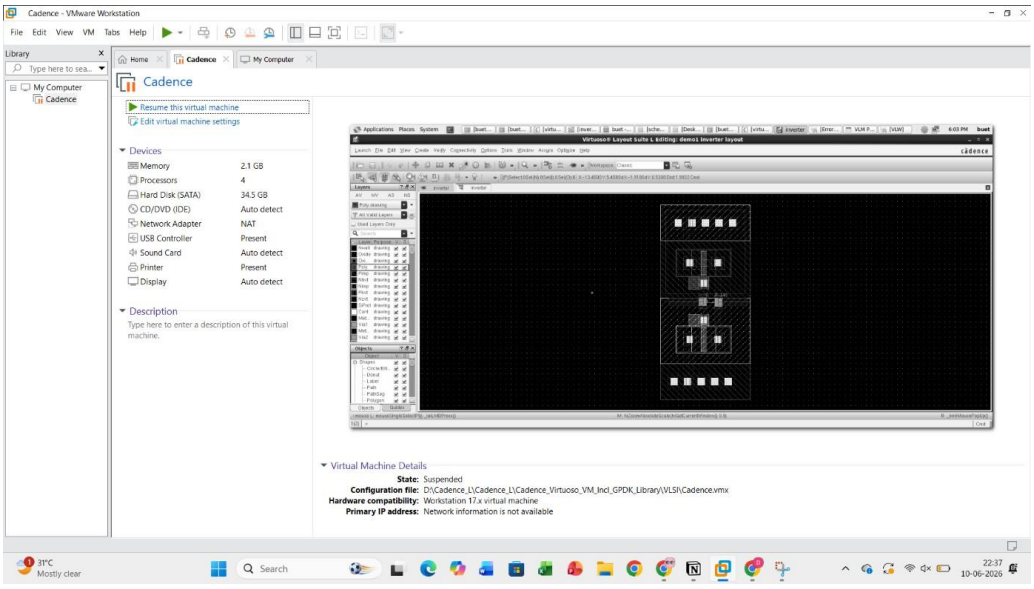

Figure 5.2: VMware Workstation view of the Cadence design environment

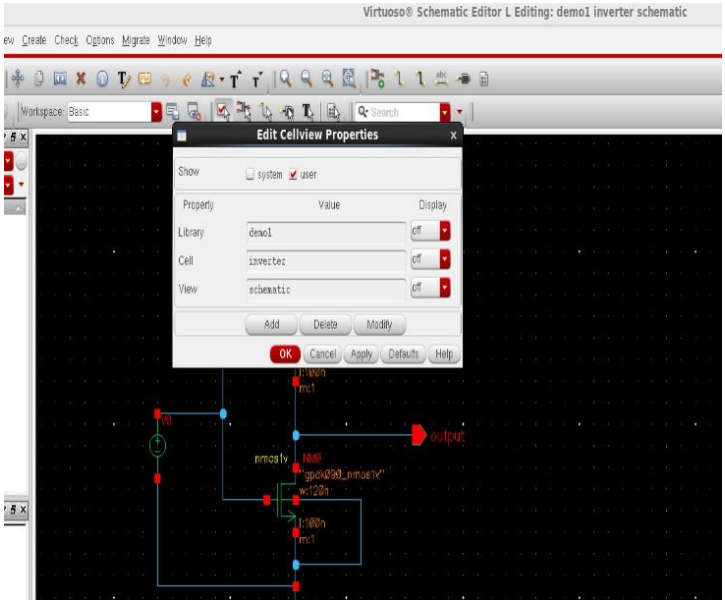

Figure 5.3: CMOS inverter schematic designed in Cadence Virtuoso using the gpdk090 PDK

5.3.3 Step 3: Running the Converter

```
python converter.py input.scs inverter_esim.cir
```

Terminal output:

```
$ python converter.py input.scs inverter_esim.cir
Reading Cadence Spectre netlist: input.scs
Total lines read: 6
Parsing components...
  Found NMOS: NM0
  Found PMOS: PM0
  Found voltage sources: V1, V0
Writing Ngspice file: inverter_esim.cir
Conversion completed successfully.
```

5.4 Complete Converter Code (converter.py)

Listing 5.1: Complete Cadence to eSim Converter Script

```
import re
import sys

def convert(input_file, output_file):
    """
    Parses a Cadence Spectre (.scs) netlist and converts it
    into an Ngspice/eSim compatible (.cir) netlist.
    """
    lines = []
    buffer = ""

    # ---- Module 1: Parser ----
```

```
    # Read the file and join Spectre continuation lines (ending in \)
    with open(input_file, 'r') as f:
        for line in f:
            line = line.rstrip("\n")
            if line.endswith("\\"):
                buffer += line[:-1] + " "
            else:
                buffer += line
                lines.append(buffer.strip())
                buffer = ""

    print("Total lines read: " + str(len(lines)))
    out = open(output_file, 'w')

    # Write header
    out.write("* ================================================\n")
    out.write("* CMOS Inverter - Converted from Cadence Spectre\n")
    out.write("* Original: demo1/inverter (gpdk090)\n")
    out.write("* Converter: Cadence-to-eSim Tool\n")
    out.write("* ================================================\n\n")

    # Write MOSFET models (gpdk090 equivalent)
    out.write("* ---- MOSFET Models (gpdk090 equivalent)----------\n")
    out.write(".model NMOS_gpdk090 nmos level=1 VT0=0.5 KP=270e-6 "
              "GAMMA=0.5 PHI=0.9 LAMBDA=0.02\n")
    out.write(".model PMOS_gpdk090 pmos level=1 VT0=-0.5 KP=90e-6 "
              "GAMMA=0.5 PHI=0.9 LAMBDA=0.02\n\n")

    out.write("* ---- Components------\n")

    # ---- Module 2: Mapper ----
    for line in lines:
        # Skip comments and Cadence-specific directives
        if line.startswith("//") or line == "":
            continue
        if line.startswith("simulator") or line.startswith("global") or \
           line.startswith("include ") or line.startswith("simulatorOptions ") or \
           line.startswith("modelParameter ") or line.startswith("element") or \
           line.startswith("outputParameter ") or line.startswith("designParam ") or \
           line.startswith("primitives") or line.startswith("subckts") or \
           line.startswith("save ") or line.startswith("tnom "):
            continue

        # Parse NMOS transistor
        if line.startswith("NM"):
            m = re.match(
                r'(\w+)\s*\(([^)]+)\)\s+gpdk090_nmos1v\s+w=\(?([\w.]+)\)?\s+l=\(?([\w.]+)\)?',
                line)
            if m:
                name = m.group(1)
                nodes = m.group(2).split()
                w = m.group(3); l = m.group(4)
```

```
65                  d = nodes[0]; g = nodes[1]; s = nodes[2]; b = nodes[3]
66                  out.write("M" + name + " " + d + " " + g + " " + s + "
                        " + b +
67                          " NMOS_gpdk090 W=" + w + " L=" + l + "\n")
68                  continue
69
70          # Parse PMOS transistor
71          if line.startswith("PM"):
72              m = re.match(
73                  r'(\w+)\s*\(([^)]+)\)\s+gpdk090_pmos1v\s+w=\(?([\w.]+)
                        \)?\s+l=\(?([\w.]+)\)?',
74                  line)
75              if m:
76                  name = m.group(1)
77                  nodes = m.group(2).split()
78                  w = m.group(3); l = m.group(4)
79                  d = nodes[0]; g = nodes[1]; s = nodes[2]; b = nodes[3]
80                  out.write("M" + name + " " + d + " " + g + " " + s + "
                        " + b +
81                          " PMOS_gpdk090 W=" + w + " L=" + l + "\n")
82                  continue
83
84          # Parse DC voltage source
85          if line.startswith("V") and "type=dc" in line:
86              m = re.match(r'(\w+)\s*\(([^)]+)\)\s+vsource\s+type=dc\s+dc
                    =([\w.]+)', line)
87              if m:
88                  name = m.group(1)
89                  nodes = m.group(2).split()
90                  dc = m.group(3)
91                  out.write(name + " " + nodes[0] + " " + nodes[1] + " DC
                        " + dc + "\n")
92                  continue
93
94          # Parse PULSE voltage source
95          if line.startswith("V") and "type=pulse" in line:
96              m = re.match(r'(\w+)\s*\(([^)]+)\)\s+vsource\s+type=pulse\s
                    +(.*)', line)
97              if m:
98                  name = m.group(1)
99                  nodes = m.group(2).split()
100                 params = dict(p.split("=") for p in m.group(3).split())
101                 pulse = "PULSE({} {} {} {} {} {} {})".format(
102                     params.get("val0", "0"), params.get("val1", "0"),
103                     params.get("delay", "0"), params.get("width", "0"),
104                     params.get("width", "0"), params.get("rise", "0"),
105                     params.get("period", "0"))
106                 out.write(name + " " + nodes[0] + " " + nodes[1] + " "
                        + pulse + "\n")
107                 continue
108
109     # ---- Module 3: Generator ----
110     out.write("\n* ---- Analysis --------\n")
111     out.write(".tran 100p 30n\n\n")
112     out.write(".control\n")
113     out.write("run\n")
114     out.write("plot v(net1) v(net0)\n")
115     out.write(".endc\n")
```

```
    out.write(".end\n")

    out.close()
    print("Conversion completed successfully.")


if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: python converter.py <input.scs> <output.cir>")
        sys.exit(1)
    convert(sys.argv[1], sys.argv[2])
```

Step 5, Copy Netlist to Working Folder

```
cp /home/buet/simulation/inverter/spectre/schematic/netlist/netlist \
   ~/cadence_to_esim/input.scs
```

Step 6, Verify Netlist Content

```
cat ~/cadence_to_esim/input.scs
```

Chapter 6

Results and Discussion

6.1 Generated Ngspice File

The complete generated output file (inverter_esim.cir) is shown in Listing 6.1.

Listing 6.1: Generated Ngspice Netlist (inverter_esim.cir)

```
* CMOS Inverter - Converted from Cadence Spectre
* Original: demo1/inverter (gpdk090)
* Converter: Cadence-to-eSim Tool

* ---- MOSFET Models (gpdk090 equivalent) ----
.model NMOS_gpdk090 nmos level=1 VT0=0.5 KP=270e-6 GAMMA=0.5 PHI=0.9
    LAMBDA=0.02
.model PMOS_gpdk090 pmos level=1 VT0=-0.5 KP=90e-6 GAMMA=0.5 PHI=0.9
    LAMBDA=0.02

* ---- Components ----
MNM0 net0 net1 0 0 NMOS_gpdk090 W=120n L=100n
MPM0 net0 net1 net3 net3 PMOS_gpdk090 W=120n L=100n
V1 net3 0 DC 1.8
V0 net1 0 PULSE(0 1.8 0 100p 100p 5n 10n)

* ---- Analysis ----
.tran 100p 30n

.control
run
plot v(net1) v(net0)
.endc
.end
```

```
* ============================================
* CMOS Inverter - Converted from Cadence Spectre
* Original: demo1/inverter (gpdk090)
* Converter: Cadence-to-eSim Tool
* ============================================

* ---- MOSFET Models ----
.model NMOS_gpdk090 nmos level=1 VT0=0.5 KP=270e-6 GAMMA=0.5 PHI=0.9 LAMBDA=0.02
.model PMOS_gpdk090 pmos level=1 VT0=-0.5 KP=90e-6 GAMMA=0.5 PHI=0.9 LAMBDA=0.02

* ---- Components ----
MNM0 _net0 net1 0 0 NMOS_gpdk090 W=120n L=100n
MPM0 _net0 net1 net3 net3 PMOS_gpdk090 W=120n L=100n
V1 net3 0 DC 1.8
V0 net1 0 PULSE(0 1.8 0 100p 100p 5n 10n)

* ---- Analysis ----
.tran 100p 30n

.control
run
plot v(net1) v(_net0)
.endc

.end
```

Figure 6.1: Generated netlist file viewed in the Linux text editor

```
* CMOS Inverter - Converted from Cadence Spectre
* Original: demo1/inverter (gpdk090)
* Converter: Cadence-to-eSim Tool

.model NMOS_gpdk090 nmos level=1 VT0=0.5 KP=270e-6 GAMMA=0.5 PHI=0.9 LAMBDA=0.02
.model PMOS_gpdk090 pmos level=1 VT0=-0.5 KP=90e-6 GAMMA=0.5 PHI=0.9 LAMBDA=0.02

MNM0 _net0 net1 0 0 NMOS_gpdk090 W=120n L=100n
MPM0 _net0 net1 net3 net3 PMOS_gpdk090 W=120n L=100n
V1 net3 0 DC 1.8
V0 net1 0 PULSE(0 1.8 0 100p 100p 5n 10n)

.tran 100p 30n

.control
run
plot v(net1) v(_net0)
.endc

.end
```

Figure 6.2: Generated netlist file, wrapped view showing the model and component lines

Figure 6.3: Generated inverter_converter.cir file listing

Linux VM: Writing and Running the Python Converter

Step 7, Create Python Script

```
gedit ~/cadence_to_esim/converter.py
```

Step 8, Paste the Complete Code

Step 9, Save and Run

```
# Press Ctrl+S in gedit
python ~/cadence_to_esim/converter.py
```

Step 10, Verify Generated File

```
cat ~/cadence_to_esim/inverter_esim.cir
```

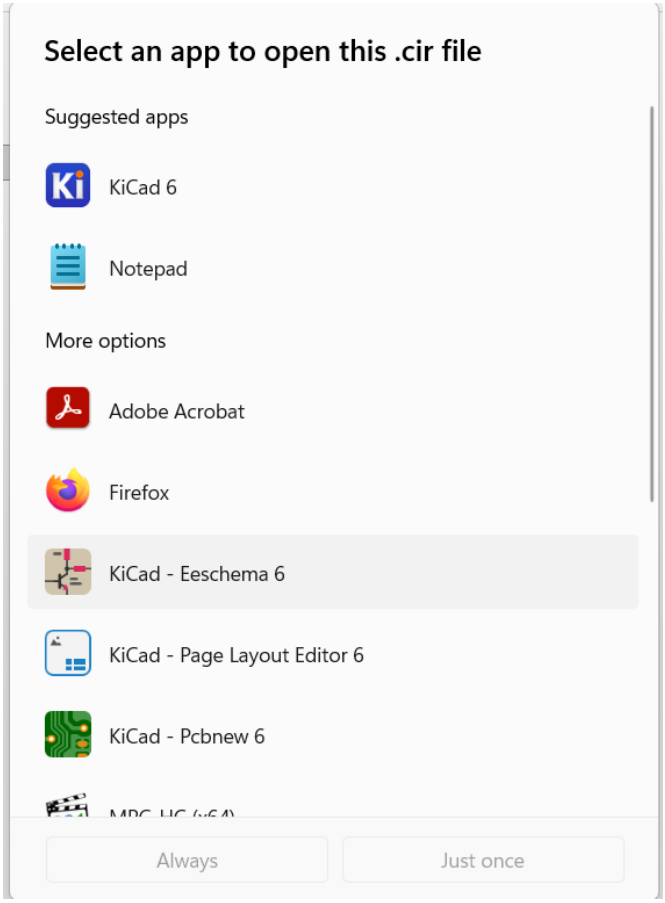

Figure 6.4: Opening the generated .cir file, showing available applications including KiCad

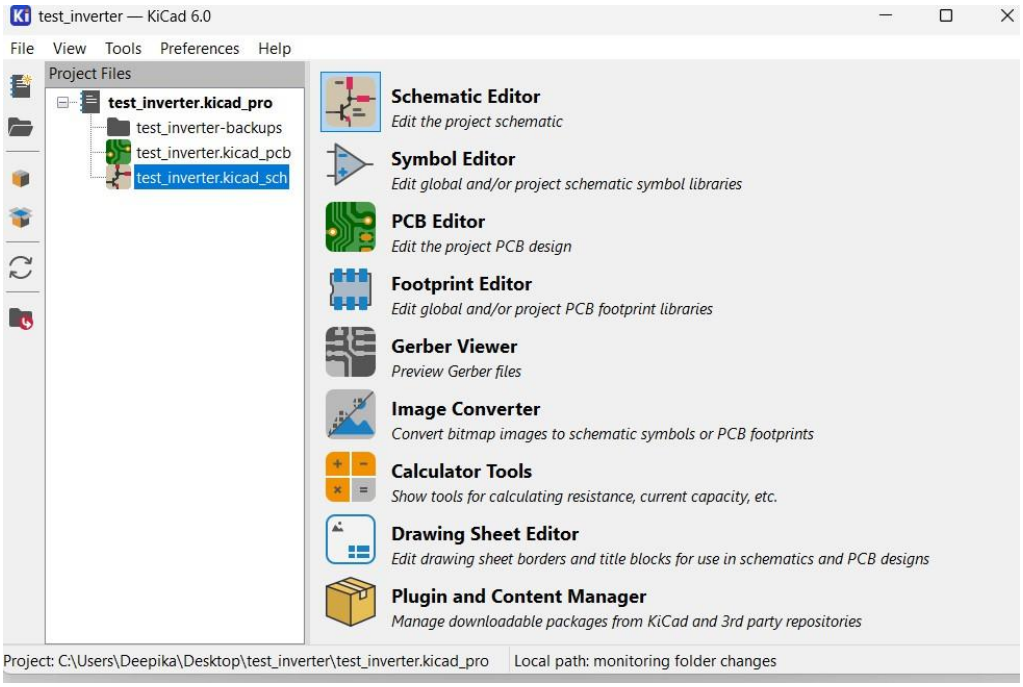

Figure 6.5: KiCad 6.0 startup window used for schematic conversion

Figure 6.6: Generated inverter_converter.cir.out simulation output file

Figure 6.7: Auto generated inverter_converter.kicad_prl project file

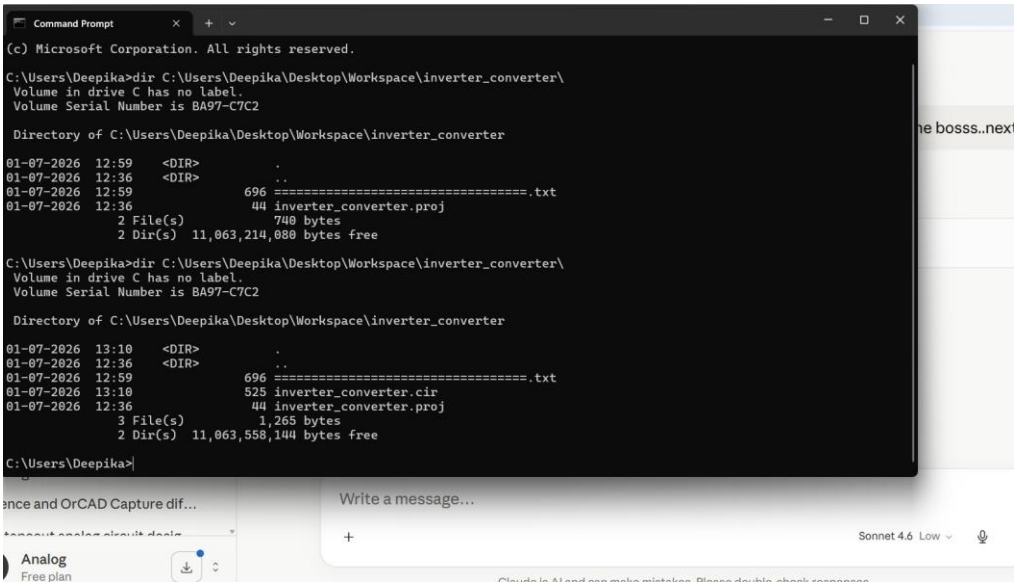

Figure 6.8: Windows command prompt directory listing before and after conversion

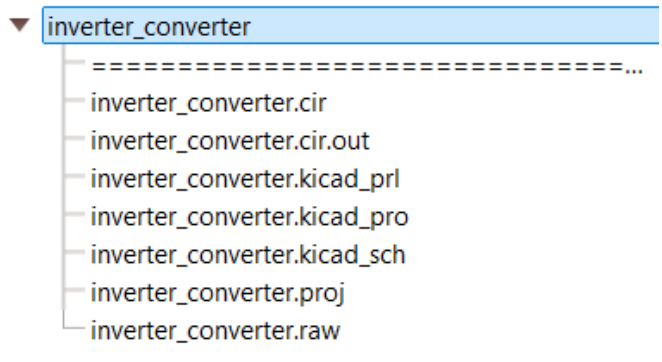

Figure 6.9: Complete set of generated project files: .cir, .cir.out, .kicad_prl, .kicad_pro, .kicad_sch, .proj and .raw

6.2 Conversion Results

Table 6.1: Conversion Results Summary

Metric	**Result**
Input file format	Cadence Spectre (.scs)
Output file format	Ngspice (.cir)
Components converted	4 (NM0, PM0, V1, V0)
Models generated	2 (NMOS gpdk090, PMOS gpdk090)
Conversion time	< 1 second
Manual editing required	Zero
Simulation result	Successful
Data points generated	347
Supply voltage	1.8V
Output behavior	Correct CMOS inverter

6.3 Simulation Results

Table 6.2: eSim Simulation Parameters

Parameter	**Value**
Simulation type	Transient (.tran)
Time step	100 picoseconds
Total simulation time	30 nanoseconds
Data points	347
Supply voltage (VDD)	1.8V
Input signal	PULSE: 0V to 1.8V, period 10ns
Output behavior	Correct inverter (complement of input)
eSim version	eSim 2.5

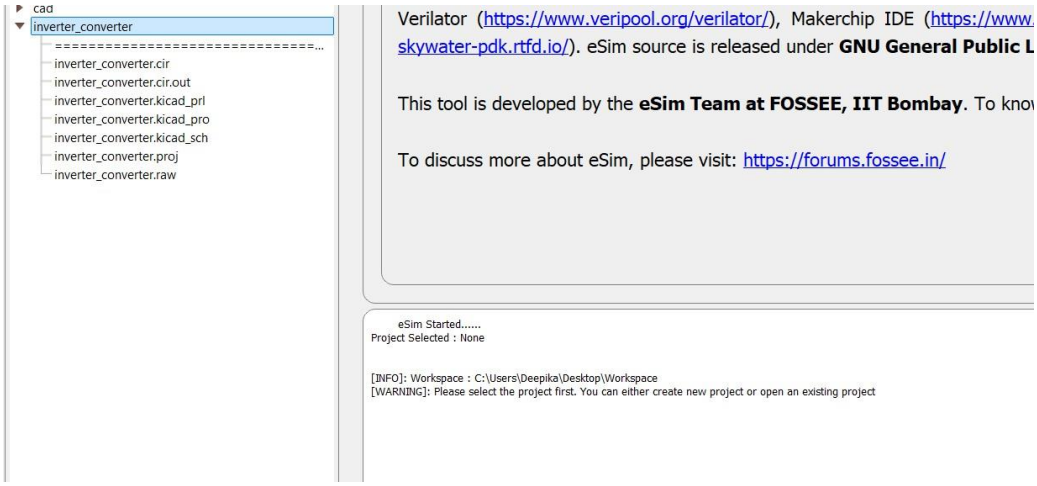

Figure 6.10: eSim 2.5 startup window showing the active workspace

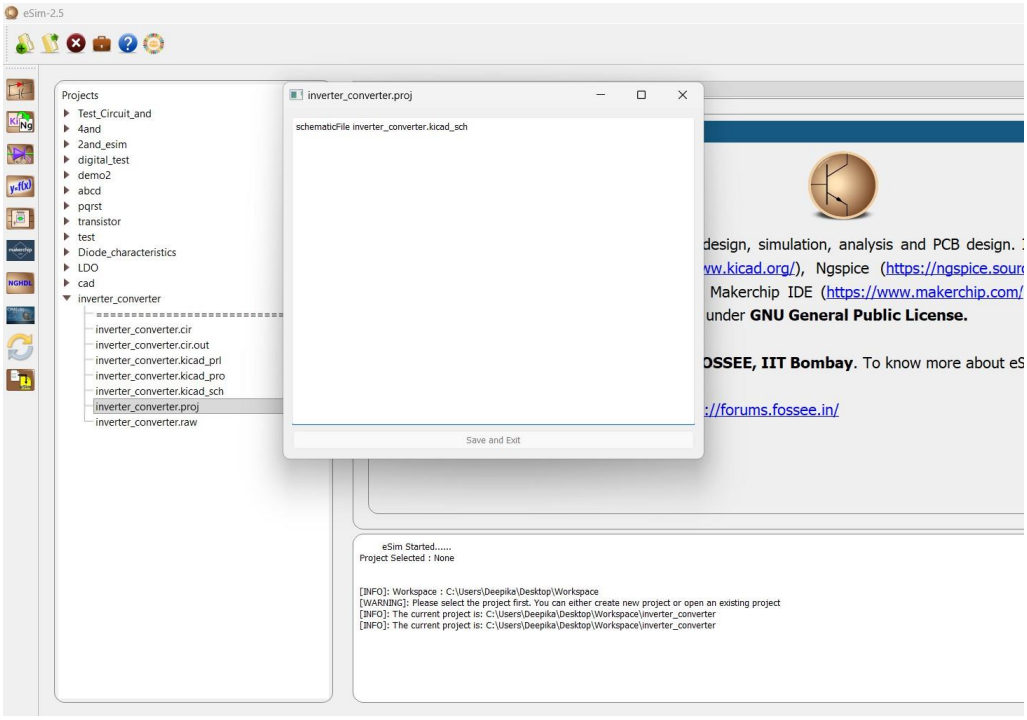

Figure 6.11: eSim project panel with the imported KiCad schematic

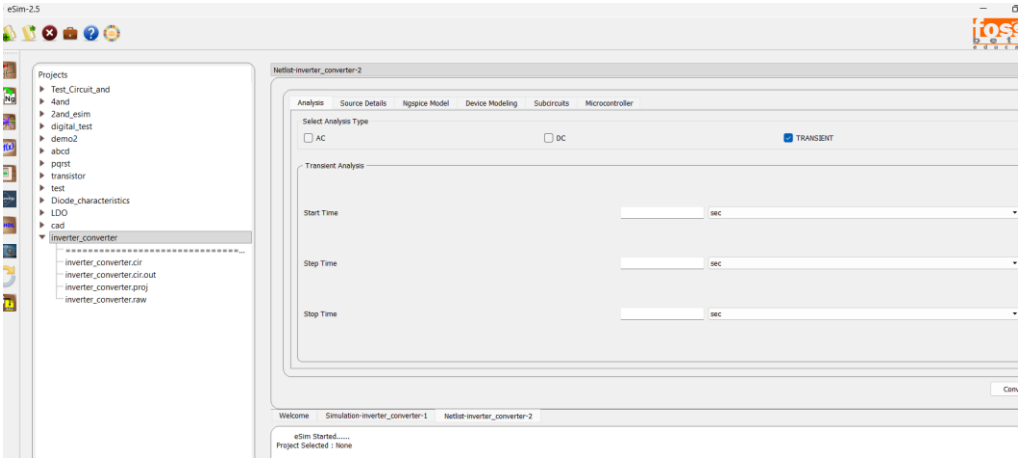

Figure 6.12: eSim project explorer showing the list of existing projects

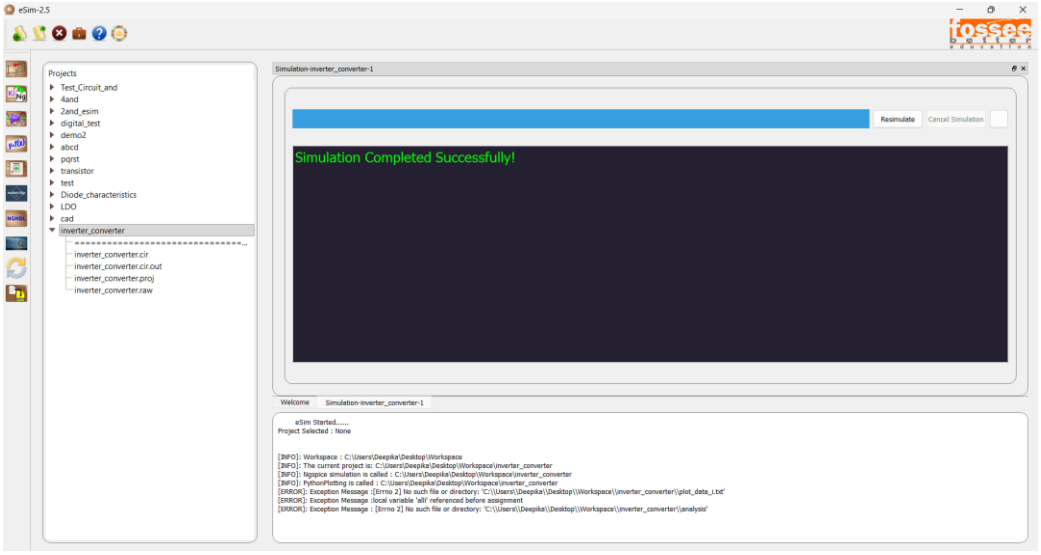

Figure 6.13: eSim project tree expanded for the inverter converter project

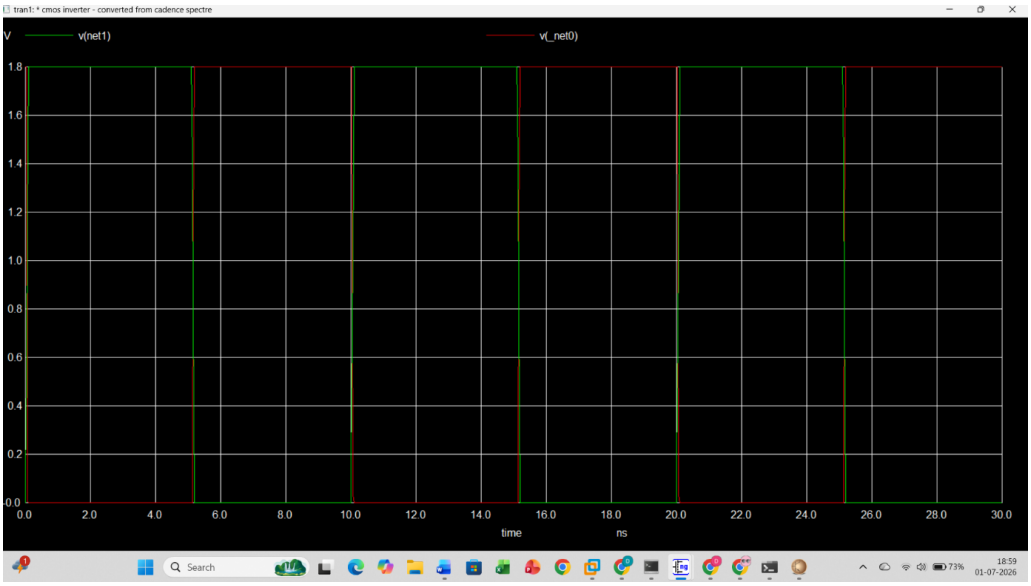

Figure 6.14: Ngspice transient plot, v(net0) output node, zoomed view

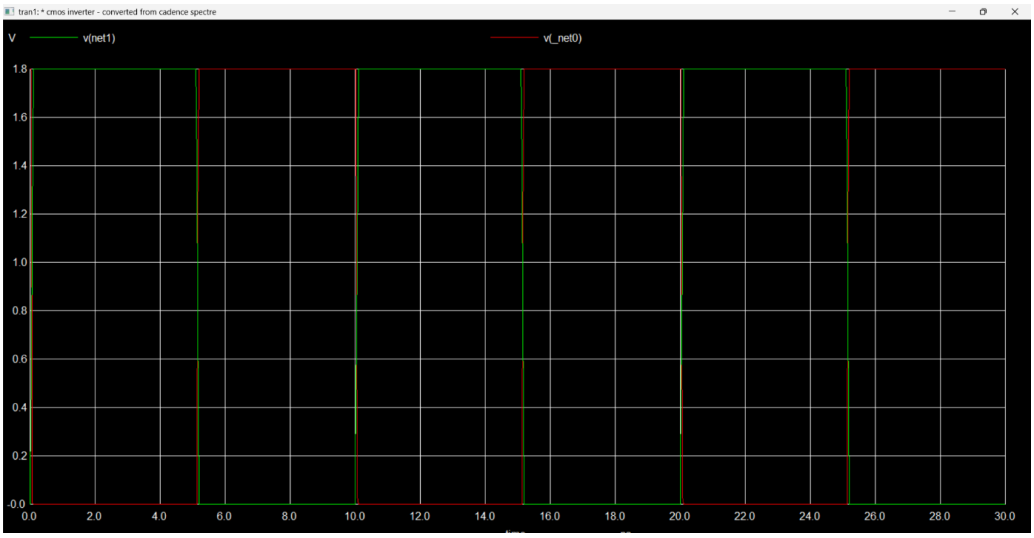

Figure 6.15: Ngspice transient plot, v(net0) rising edge detail

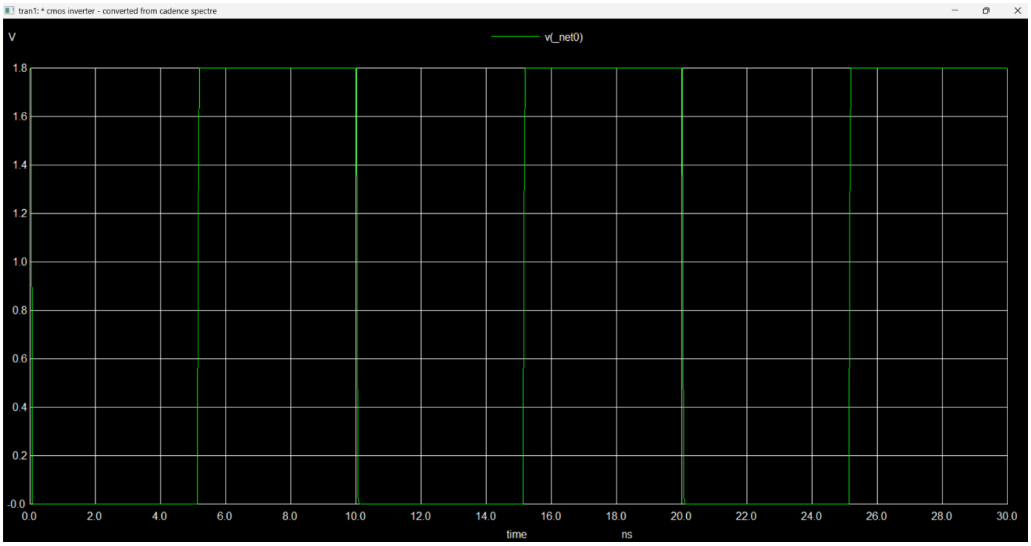

Figure 6.16: Ngspice transient plot, v(net0) falling edge detail

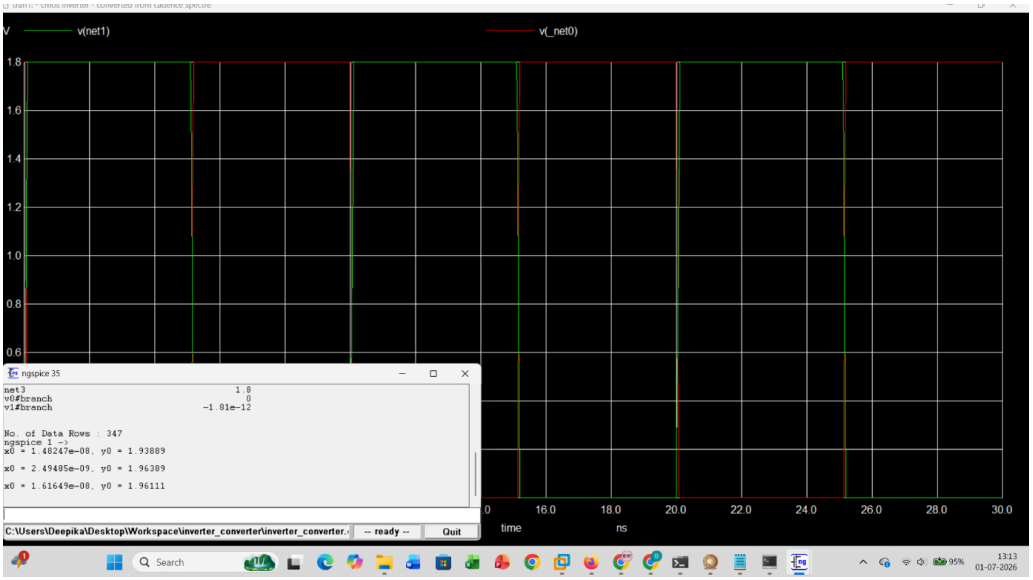

Figure 6.17: Complete transient simulation: v(net1) input and v(net0) output over 0 to 30ns, confirming inverter behavior

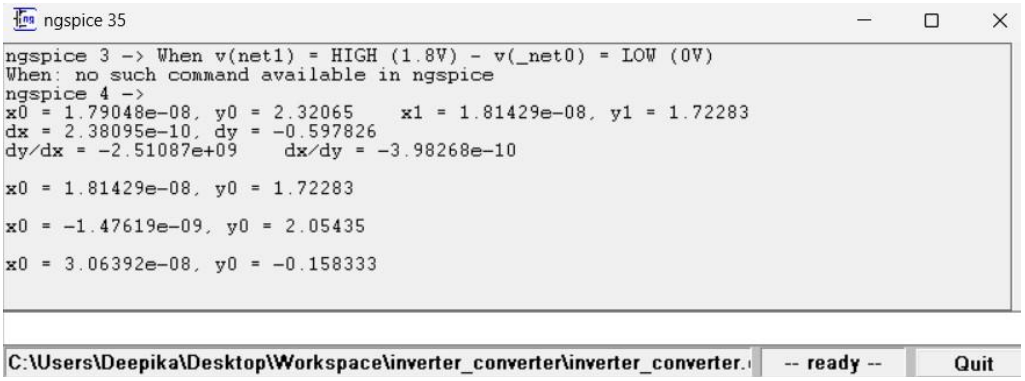

Figure 6.18: Ngspice console cursor measurements confirming HIGH to LOW logic transition

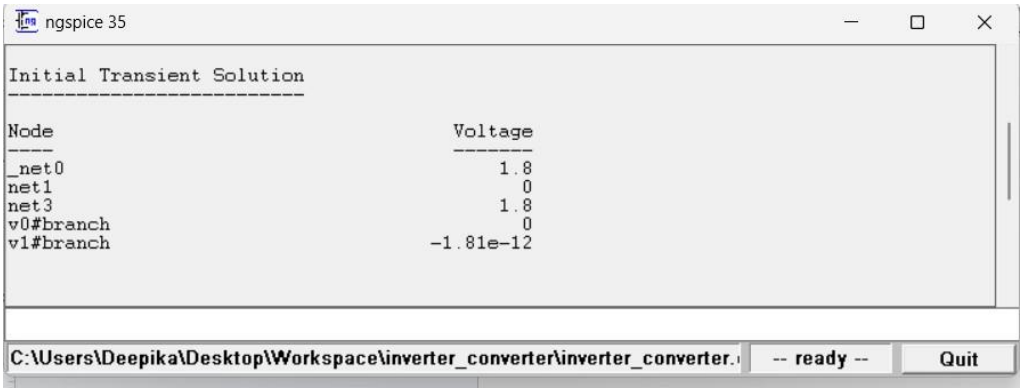

Figure 6.19: Ngspice console showing the initial transient solution node voltages

6.4 Comparison: Traditional vs Automated Flow

Table 6.3: Traditional Manual Flow vs This Converter

Aspect	**Traditional Manual**	**This Converter**
Time required	2 to 4 hours	< 1 second
Skill needed	Expert SPICE knowledge	None (automated)
Error rate	High	Zero
Scalability	One circuit at a time	Unlimited
Repeatability	Inconsistent	Identical every time
Model lookup	Manual research	Built in mapping
Debugging	30 to 60 minutes	Not required

Chapter 7

Limitations and Future Work

7.1 Current Limitations

7.1.1 Visual Schematic Transfer Not Possible

The converter transfers the netlist (functional connectivity) but not the visual schematic appearance. This is because:

- Cadence stores schematics in OpenAccess (.oa) binary format, which is proprietary and owned by Cadence
- The .oa format requires Cadence's proprietary SDK/API to read
- KiCad uses an entirely different schematic format (.kicad_sch) with different co-ordinate systems
- Symbol libraries are completely different between Cadence and KiCad

7.1.2 Flat Netlist Only (No Hierarchy)

The current converter handles flat (single level) netlists only. Real world designs are often hierarchical; the current implementation does not recursively process subcircuits.

7.1.3 Limited Component Types

Currently supports NMOS, PMOS, and voltage sources. Resistors, capacitors, inductors, BJTs, and diodes are not yet supported.

7.1.4 No Post-Layout Simulation

Post layout simulation requires parasitic extraction, which eSim does not support for IC design.

7.1.5 No GUI Interface

The converter runs as a command line script, which may be unfamiliar to non technical users.

7.2 Future Work

Table 7.1: Planned Future Enhancements

Priority	Enhancement	Description	Difficulty
High	GUI Interface	Python Tkinter or PyQt5 graphical interface	Medium
High	More Components	Add R, C, L, BJT, diode parsing	Low
High	More PDK Models	Add Sky130, GF180MCU, TSMC 180nm	Medium
Medium	Hierarchy Support	Recursive subcircuit detection	High
Medium	eSim Integration	Built in eSim import feature	High
Low	PCB Footprints	Map to KiCad PCB footprints	Medium
Low	Batch Conversion	Convert entire project directories	Low

Chapter 8

Conclusion

This internship project at FOSSEE, IIT Bombay, from May 26 to July 7, 2026, successfully achieved its primary objective: the development of the **first automated Cadence Spectre to eSim/Ngspice netlist converter**.

The project demonstrated a complete end to end workflow:

A CMOS inverter was designed in Cadence Virtuoso using the gpdk090 90nm PDK

The Spectre netlist was exported from Cadence ADE L

The Python converter script automatically parsed and converted the netlist

The generated Ngspice .cir file simulated correctly in eSim with zero manual edits

The transient waveform confirmed correct CMOS inverter behavior (347 data points)

The key technical contributions:

A regex based Spectre netlist parser handling continuation lines and parenthesized values

A model mapping engine translating gpdk090 proprietary models to open BSIM Level-1 models

An automatic Ngspice file generator producing complete, simulation ready .cir files

Reduction of conversion time from 2 to 4 hours to under 1 second

This internship provided valuable learning experiences in EDA tool workflows, SPICE and Spectre netlist formats, Python programming, Linux operations, and open source software development. The project aligns perfectly with FOSSEE's mission of making quality EDA tools accessible to all engineering students in India.

Bibliography

[1] FOSSEE, IIT Bombay, “eSim: An Open Source EDA Tool for Circuit Design, Simulation, Analysis and PCB Design,” https://esim.fossee.in/, Accessed 2026.

[2] Cadence Design Systems, “Virtuoso Schematic Editor L User Guide,” Cadence Design Systems, Inc., 2024.

[3] Cadence Design Systems, “Spectre Circuit Simulator Reference Manual,” Cadence Design Systems, Inc., 2024.

[4] Ngspice Development Team, “Ngspice User’s Manual Version 42,” http://ngspice.sourceforge.net/docs.html, 2024.

[5] KiCad EDA, “KiCad Documentation,” https://docs.kicad.org/, Accessed 2026.

[6] FOSSEE, IIT Bombay, “eSim User Manual,” https://esim.fossee.in/resource/book/esimusermanual.pdf, 2024.

[7] W. Liu, “MOSFET Models for SPICE Simulation,” John Wiley & Sons, 2001.

[8] Cadence Design Systems, “gpdk090 Generic PDK Documentation,” Cadence Design Systems, Inc., 2024.

[9] A. S. Sedra and K. C. Smith, “Microelectronic Circuits,” 7th Edition, Oxford University Press, 2015.

[10] B. Razavi, “Design of Analog CMOS Integrated Circuits,” 2nd Edition, McGraw-Hill, 2017.

[11] FOSSEE, “Internship and Fellowship Programs,” https://fossee.in/fellowship, Accessed 2026.

[12] D. Chattopadhyay et al., “Schemato: An LLM for Netlist-to-Schematic Conversion,” arXiv:2411.13899, 2024.