Introduction

Overview

A Digital-to-Analog Converter (DAC) is an essential component in modern electronics, responsible for converting digital signals into analog ones. This allows devices such as audio systems, communication systems, and measurement instruments to interpret digital data in a form that can interact with the physical world. Among the various DAC architectures, the **R-2R Ladder DAC** stands out for its simplicity and scalability, making it a popular choice for applications requiring moderate resolution and performance.

This tutorial guides you through designing an 8-bit R-2R DAC schematic using **Cadence Virtuoso**. The design will leverage the **gpdk 45nm** technology node and focus on schematic capture, providing students with a hands-on introduction to Cadence's powerful design environment.

Objective

By the end of this tutorial, you will:

- Understand the fundamentals of the R-2R DAC architecture.
- Gain hands-on experience with Cadence Virtuoso for schematic design.
    - Learn how to create libraries, cells and schematics
    - Learn basic navigation of the Virtuoso Environment
- Simulate and verify the DAC's functionality using testbenches and performance metrics.

Pre-requisites

Before starting this tutorial, ensure you have:

1. **Access to Cadence Virtuoso**: Installed with the **gpdk 45nm PDK** library.
2. Circuit theory, particularly resistive networks and CMOS basics.

Section 1: Setting Up the Environment

In this section, you will set up the necessary environment in Cadence Virtuoso to begin designing the 8-bit R-2R DAC. Follow these steps to ensure a smooth workflow.

Step 1: Launch Cadence Virtuoso

1. **Log in to your Palmetto Desktop** where Cadence Virtuoso is installed.
2. Open a terminal and navigate to the directory where your Cadence setup is located.
3. Load the IC and SPECTRE Module by typing:

```
module load cadence/IC/618
module load cadence/SPECTRE/211
```

4. Launch Virtuoso by typing:

```
virtuoso &
```

   This command starts the Cadence Virtuoso Design Environment.

Step 2: Setting Up Library Paths

1. From the **Virtuoso CIW**, go to the top menu and select:

```
Tools -> Library Path Editor
```

2. Ensure that your path editor has the following libraries with their respective paths as shown below:

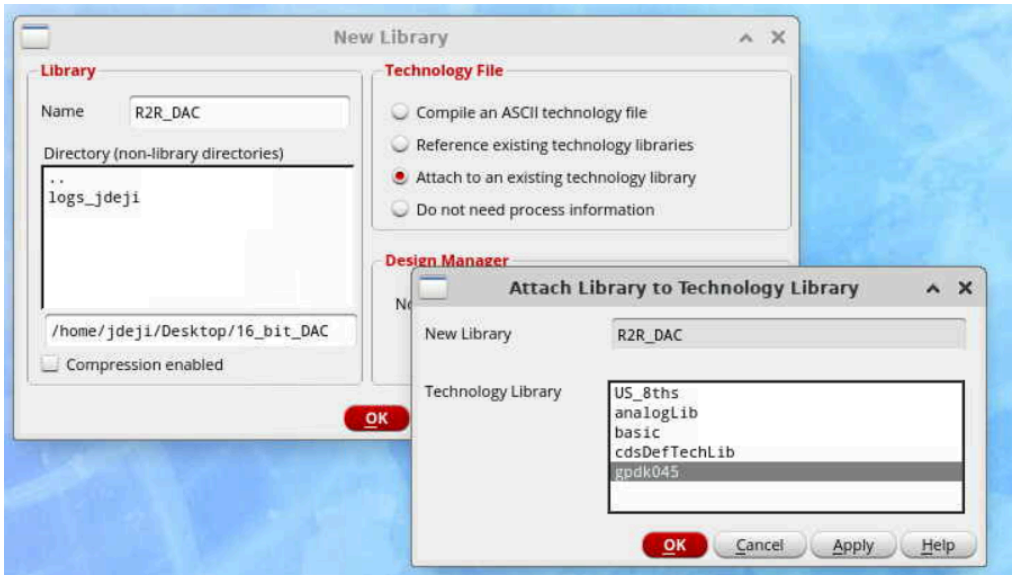
3. Click `File -> Save` to save any changes.

Step 3: Create a New Library

4. From the **Virtuoso Workspace**, go to the top menu and select:

   `File -> New -> Library`

5. In the dialog box:
   - Enter a **name** for your library, e.g., `R2R_DAC`
   - Choose the option to **attach a technology file**.
   - Select the **gpdk 45nm** technology file from the dropdown.
6. Click **OK** to create the library.



Step 3: Create a New Cell View

1. Navigate to the Library Manager `Tools -> Library Manager`
2. In the Library Manager:
   - Select your newly created library, `R2R_DAC`
   - Choose:

     `File -> New -> Cell View`

3. In the dialog box:
   - Enter a **cell name**, e.g., `R2R_DAC_Schematic`.
   - Select **Virtuoso Schematic** as the view type.
4. Click **OK** to open the Schematic Editor.

Step 4: Familiarize Yourself with the Schematic Editor

1. **Toolbars**:
    - Locate the tools for placing components, wires, and labels. Familarize yourself with some of the features from the toolbars.

2. **Grid Settings**:
    - Adjust the grid for easier placement of components:

      ```
      Options -> Display
      ```

      Set a reasonable grid spacing, such as 0.05.

Step 5: Load the gpdk 45nm Library

1. In the schematic editor, ensure the **gpdk 45nm PDK** is available in your Library Manager.
2. Explore the library to locate:
    - Resistors (R and 2R components).
    - Transistors for switches (e.g., NMOS and PMOS).

- Voltage sources for digital control signals.

Step 6: Save Your Work

1. After setting up the cell view, save the empty schematic as a starting point:

   ```
   File -> Save
   ```

2. Ensure the cell is correctly linked to your library.

Next Steps

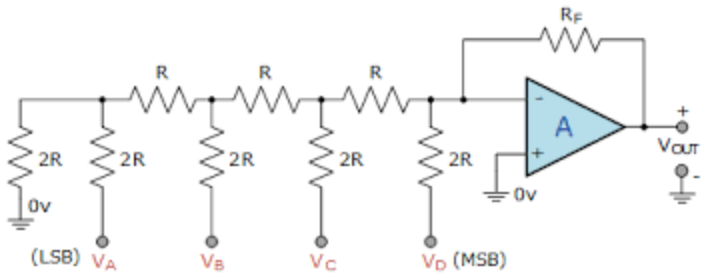
With the environment set up, you're ready to begin building the R-2R ladder network. In the next section, we will dive into **understanding the R-2R structure** and start creating the DAC schematic.

Section 2: Understanding the R-2R Ladder Network

What is an R-2R Ladder?

The **R-2R Ladder Network** is a simple, yet effective way to implement a DAC. It consists of a ladder-like structure of resistors arranged in a specific pattern:

1. **R and 2R Resistors**:
   - The "R" represents a base resistor value.
   - The "2R" is twice the value of "R."
2. **Binary Weighting**:
   - The network leverages this resistor ratio to produce voltage levels proportional to a binary digital input.
3. **Scalability**:
   - Adding more bits to the DAC only involves adding more R-2R stages, making it easy to extend resolution.



How It Works

1. **Input Bits**:
    - The DAC converts a digital binary word into an analog voltage.
    - Each bit controls a switch, connecting the corresponding resistor leg to either a reference voltage Vref (logic HIGH) or logic LOW.
2. **Voltage Division**:
    - The resistor network creates a weighted sum of the binary inputs.
    - This sum determines the output voltage.
3. **Output**:
    - The analog output Vout is as shown below where B{i} is the digital input at that bit position(0 or 1). In this case our logic low is 0.4V and logic high is 0.9V.

$$V_{out} = 0.4 + (0.9 - 0.4)(\frac{B7}{2} + \frac{B6}{4} + \frac{B5}{8} + \frac{B4}{16} + \frac{B3}{32} + \frac{B2}{64} + \frac{B1}{128} + \frac{B0}{256})$$

Schematic Representation

In an 8-bit R-2R DAC:

- The first stage (MSB) connects to the highest voltage through a 2R resistor.
- Subsequent stages use alternating R and 2R resistors, creating the characteristic ladder structure.
- Each bit's switch determines whether its leg contributes to the voltage sum or is grounded.

Next Steps

In the next section, you will:

1. **Start Building the Ladder**:
    - Place the resistors and configure their values in Cadence Virtuoso.
2. **Add Binary Switches**:
    - Use CMOS transistors to simulate the bit-controlled switches.

Section 3: Building the DAC Schematic

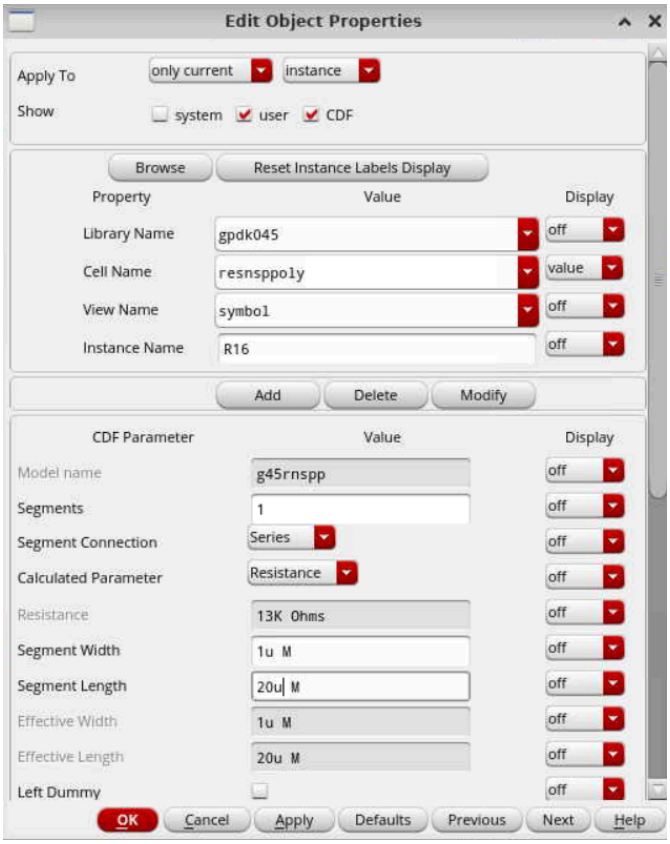
In this section, you will construct the 8-bit DAC schematic using Cadence Virtuoso. This involves placing the resistors, switches, input control signals opamp to create a functional schematic.

Step 1: Place the Resistors

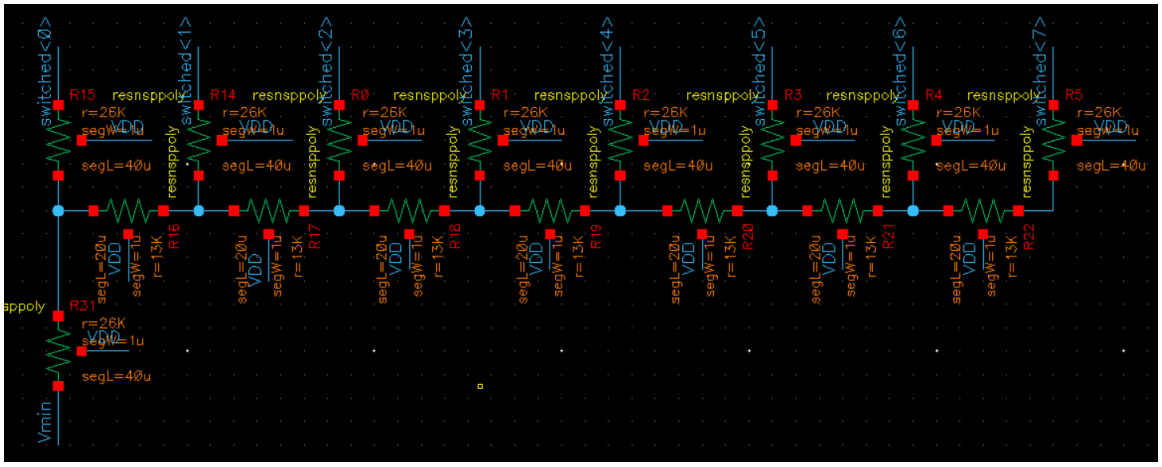
1. Open the **Schematic Editor** for your cell R2R_DAC_Schematic.
    - If not already open, navigate to your library, select the cell, and open the schematic view.
2. **Access the Resistors**:
    - From the top menu, select (or press “i” on your keyboard):

Create -> Instance

    - In the **Component Browser**, choose:
        - Library: gpdk 45nm
        - Cell: resnsnpoly



3. **Place R and 2R Resistors**:
    - For "R," set the resistor value to a base value (e.g., 10kΩ).
    - For "2R," set the resistor value to twice the base value (e.g., 20kΩ).
        - Use the **Property Editor** to change the resistance value after placing the instance.
4. **Arrange Resistors**:
    - Arrange the resistors in the ladder structure, alternating between R and 2R.
    - Label the bulk terminal of each resistor VDD by drawing a wire(using keyword “w”) from each terminal and labelling each wire by typing “l” as shown below.
    - Also Label each terminal of the resistor as shown below
    - Save schematic for now and ignore errors.

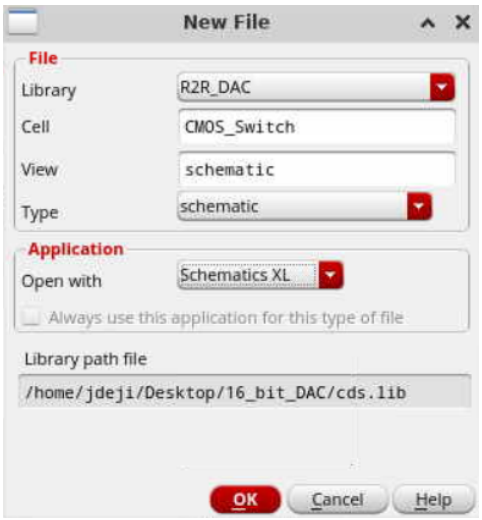


Step 2: Add Switches for Input Control

1. From your library manager
2. In the Library Manager:
   - Select your newly created library, `R2R_DAC`
   - Choose:

     `File -> New -> Cell View`

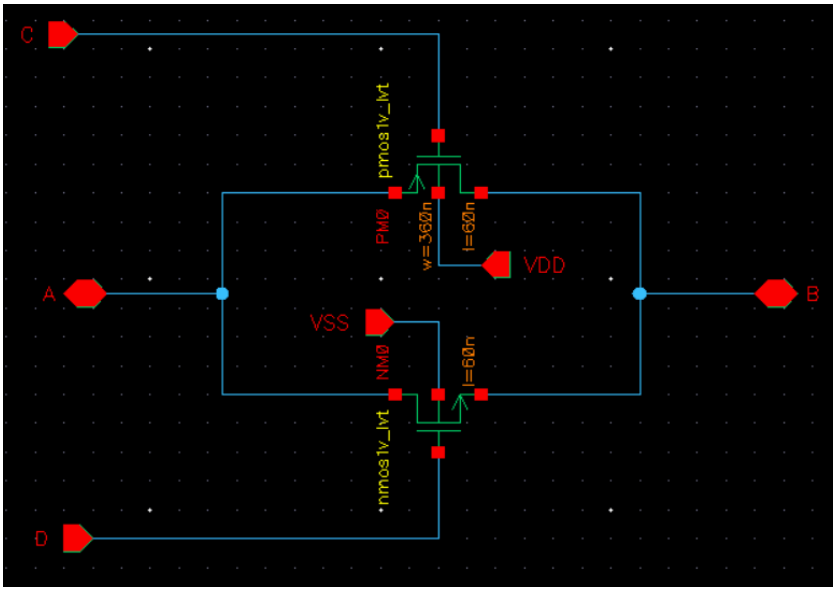
3. In the dialog box:
   - Enter a **cell name**, e.g., `CMOS_switch`
   - Select **schematic** as the view type.



4. Click **OK** to open the Schematic Editor.
   - From the **Component Browser**, select:

- Library: gpdk 45nm
- Cell: pmos1v_lvt (PMOS transistor for switches).

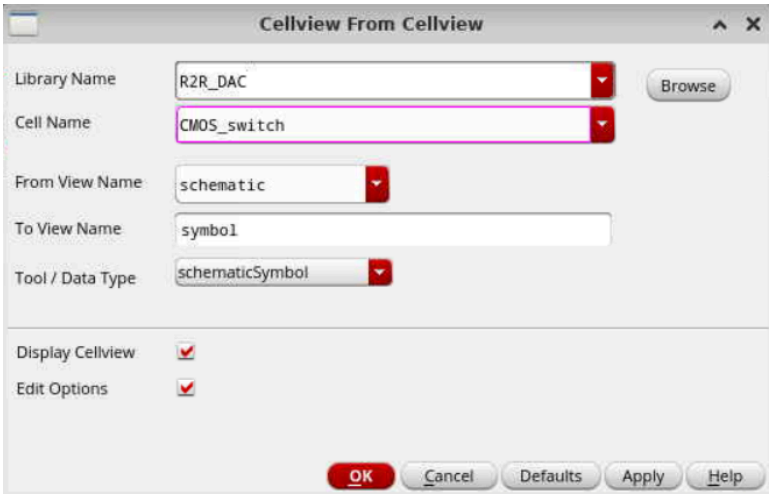
5. **Place NMOS Transistors**:
   - Repeat step 4 for the nmos1v_lvt with 60n length and 120n finger Width
   - Arrange the transistors so that their drains connect to the ladder and their sources connect to ground as shown below.



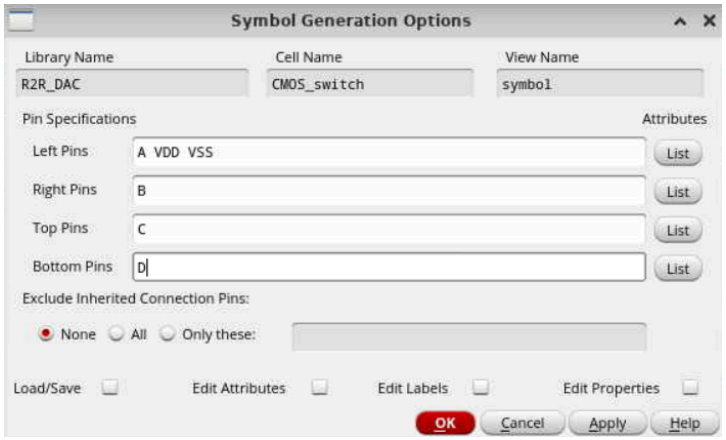
Note: To place pins simply use the create pin icon in the toolbar. Also note that pins A and B are **“inputOutput”** Direction and **VDD, VSS** are power pins while others are signal pins.

6. **Creating Symbol for CMOS switch**
    - Save your progress
    - In the Schematic editor for the switch:
        - Choose:

```
Create -> Cell View -> From Cell View
```



- Cick **OK**
- Enter the pin placements as shown below



- Cick **OK**, save and close the symbol Editor

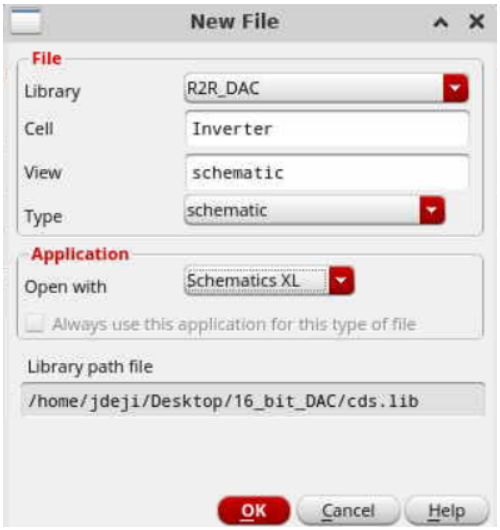
Step 3: Creating Inverter

1. From your library manager
2. In the Library Manager:
   - Select your newly created library, `R2R_DAC`
   - Choose:

     ```
     File -> New -> Cell View
     ```

3. In the dialog box:

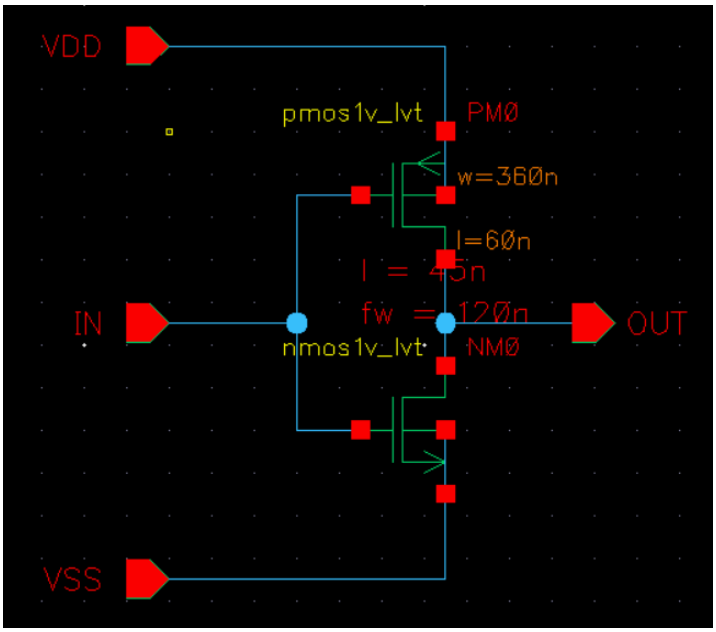
- Enter a **cell name**, e.g., `Inverter`
- Select **schematic** as the view type.



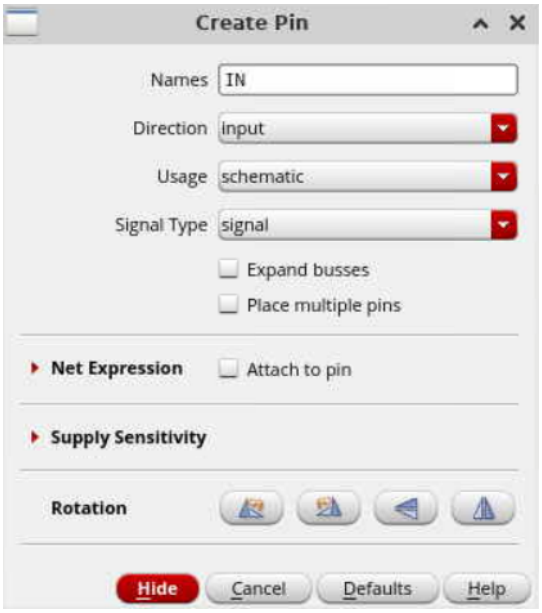
4. Click **OK** to open the Schematic Editor.
   - From the **Component Browser**, select:
     - Library: gpdk 45nm
     - Cell: pmos1v_lvt (PMOS transistor for switches).

5. **Place NMOS Transistors**:
   - Repeat step 4 for the nmos1v_lvt with 45n length and 120n finger Width

- Arrange the transistors so that their drains connect to the ladder and their sources connect to ground as shown below.



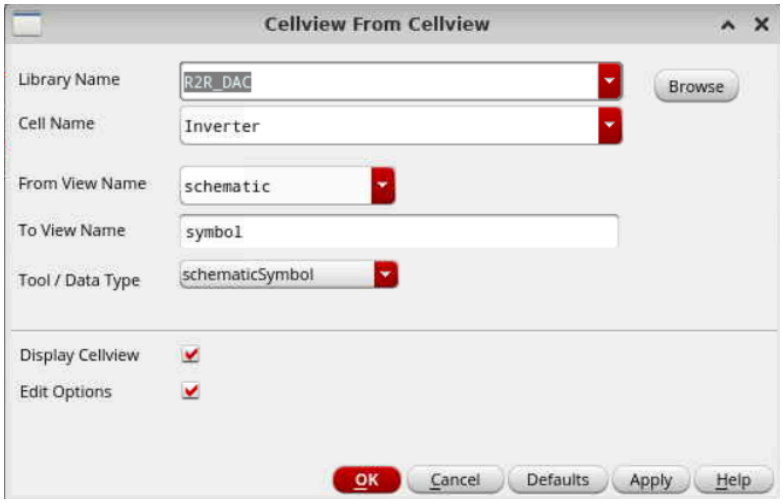
Note: To place pins simply use the create pin icon in the toolbar. Also note that pins IN, VDD and VSS are **"input"** Direction. **VDD, VSS** are power pins while others are signal pins.



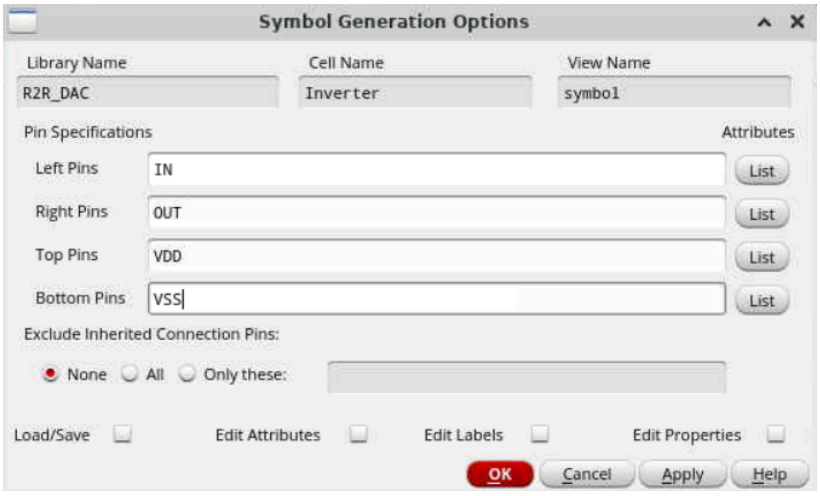
6. **Creating Symbol for Inverter**
   - Save your progress

- In the Schematic editor for the switch:
  - Choose:

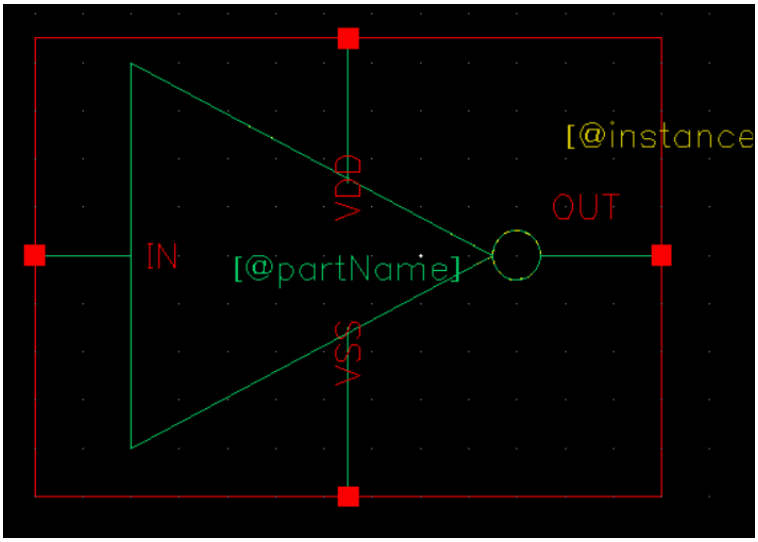
```
Create -> Cell View -> From Cell View
```



- Cick **OK**
- Enter the pin placements as shown below

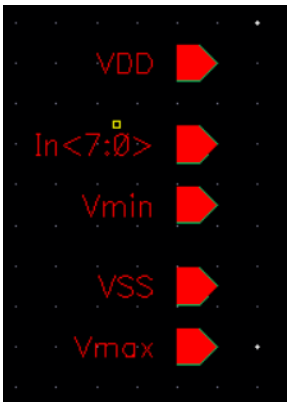


- Cick **OK**
- In the Symbol Editor you can shape the Inverter to look more like one as shown below. Do this by simply drawing the outline(within the red boundaries) using the drawing tools in the tool bar.



Step 4: Importing Switch and Inverter Symbols

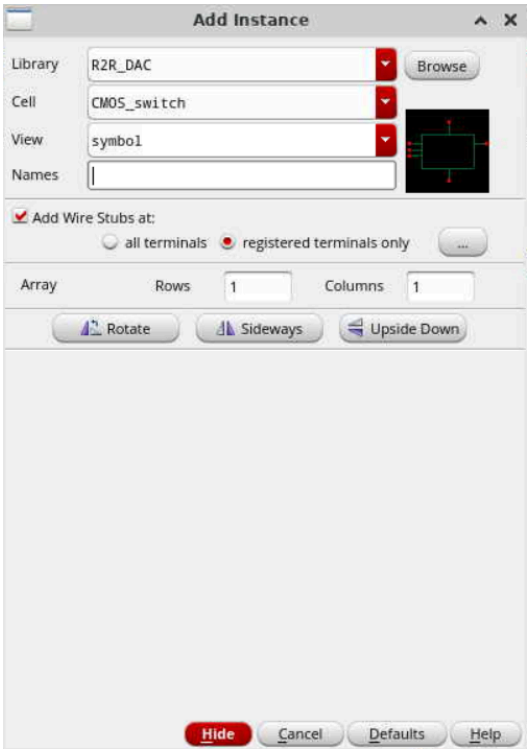
1. **Input Pins**:
    - If not already open, open the R-2R DAC_Schematic.
    - Add the following input pins using the create pin icon in the tool bar



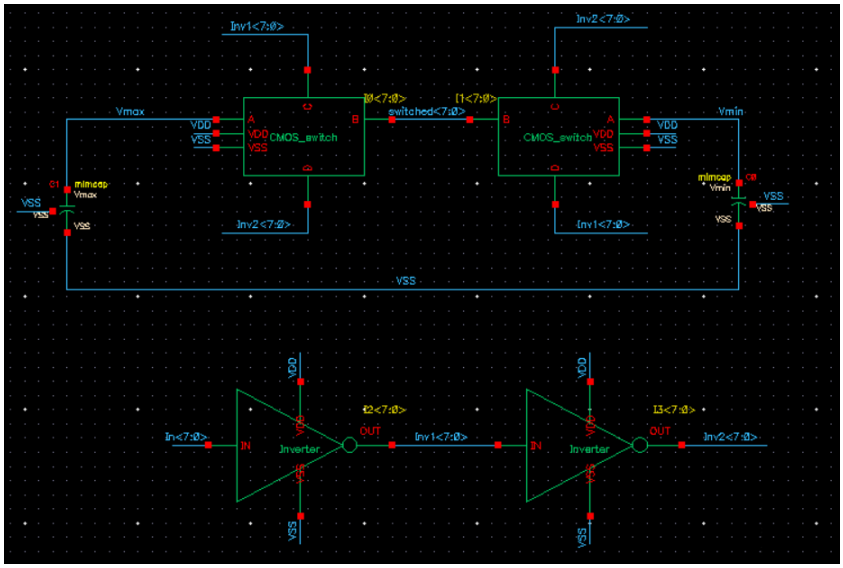
    - You do not have to connect them using a wire to anything as we will use connect the nets by label later on.

2. Place an output pin at the output of the Resistor Ladder named “out”

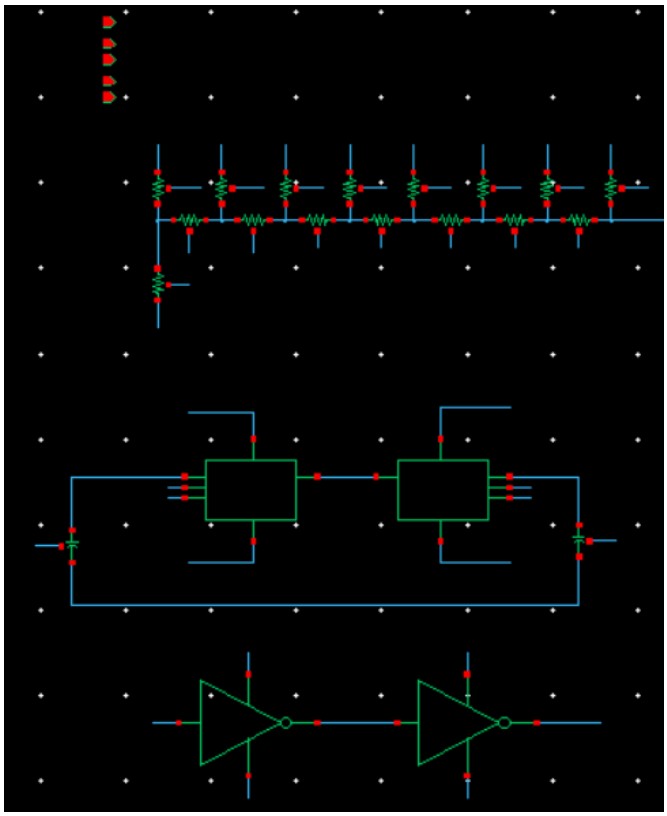
3. **Output Label**:
    - Import the CMOS switch and the Inverter using the shortcut key “i” as shown below



    - Assemble the schematic shown below with the imported CMOS_switch and Inverters. **Ensure that the Wires are labelled correctly as shown.**



- Your zoomed-out DAC should look something like the image below

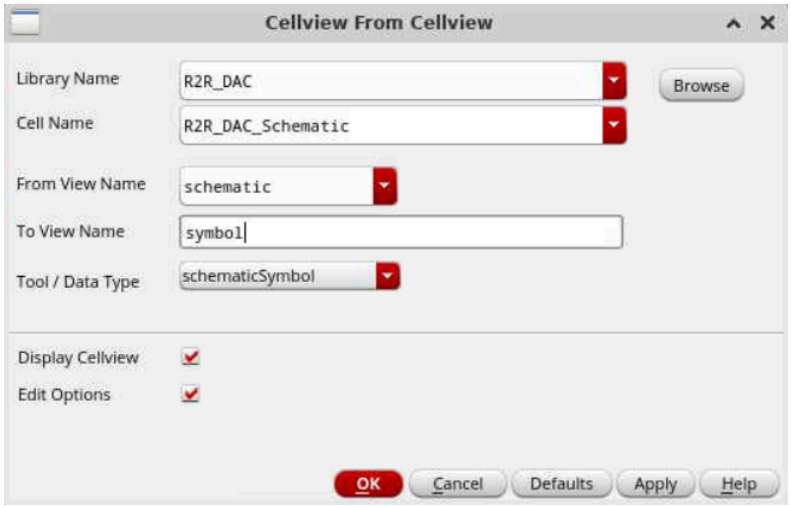


Step 5: Create DAC symbol

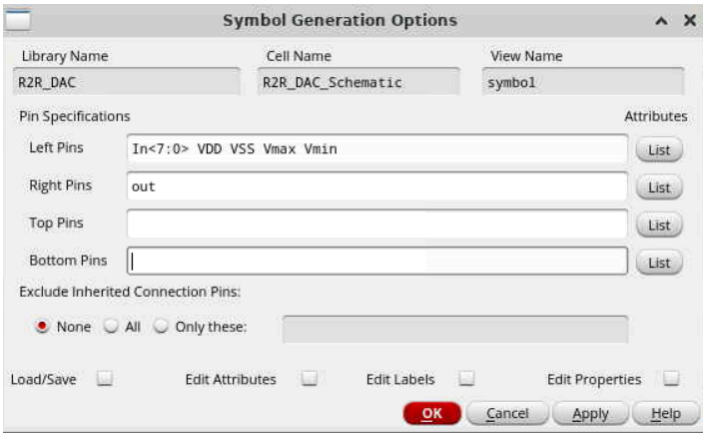
- Save your progress

- In the Schematic editor for the switch:
  - Choose:

```
Create -> Cell View -> From Cell View
```



- Cick **OK**
- Enter the pin placements as shown below



- Cick **OK**

Step 6: Verify Connections

1. **Check for Errors**:
   - Use the **Check and Save** function in the Schematic Editor:

     Check -> Current Cellview

2. **Finalize the Schematic**:

- Ensure the ladder structure, switches, and connections are correctly implemented.

Next Steps

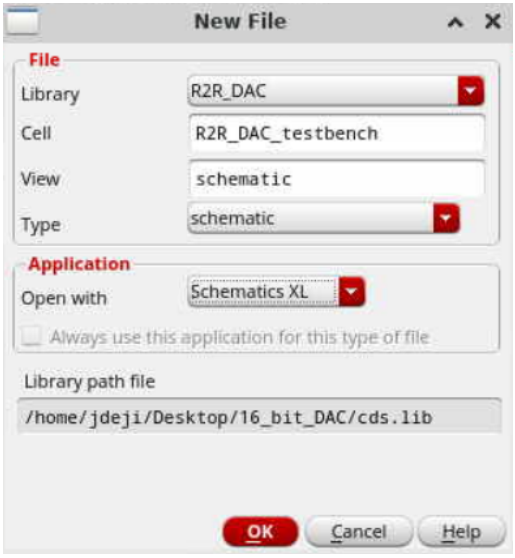
With the R-2R ladder schematic complete, you're ready to simulate its behavior. In **Section 4**, we'll set up a testbench to evaluate the DAC's performance and functionality.

Section 4: Simulating the R-2R DAC

In this section, we will set up a testbench to simulate the functionality of your 8-bit R-2R DAC. This involves creating a new testbench schematic, applying digital inputs, and analyzing the DAC's output waveform.

Step 1: Create a New Testbench

1. In the **Library Manager**, create a new cell view:
   - Library: R2R_DAC
   - Cell Name: R2R_DAC_Testbench
   - View Type: Schematic



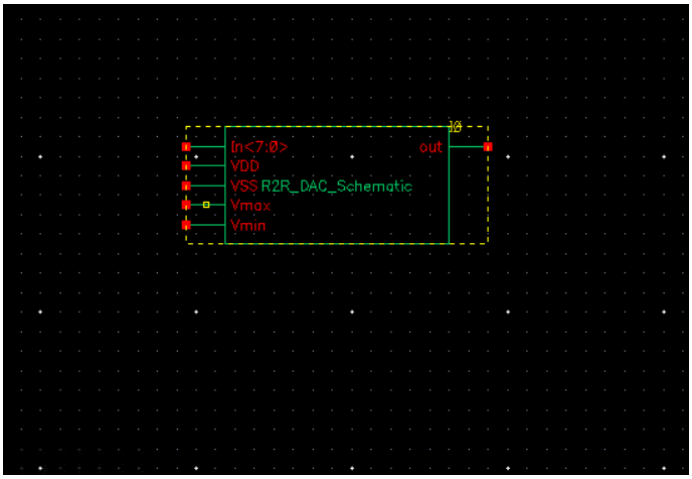
2. Open the schematic view of your testbench.

Step 2: Instantiate the R-2R DAC

1. Add your DAC schematic to the testbench:
    - From the menu, select:

Create -> Instance

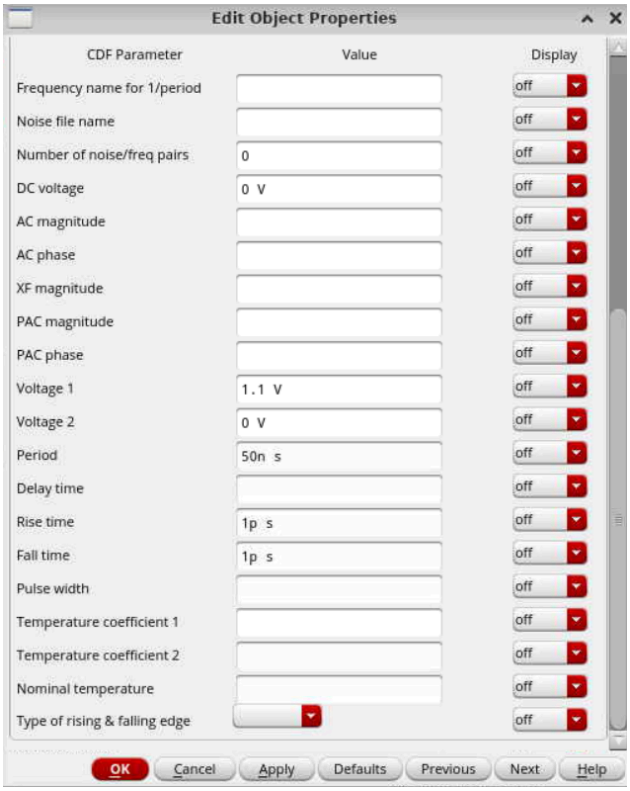
    - In the **Component Browser**, choose:
        - Library: R2R_DAC_Project
        - Cell: R2R_DAC_Schematic
        - View: schematic
    - Place the instance in the testbench.



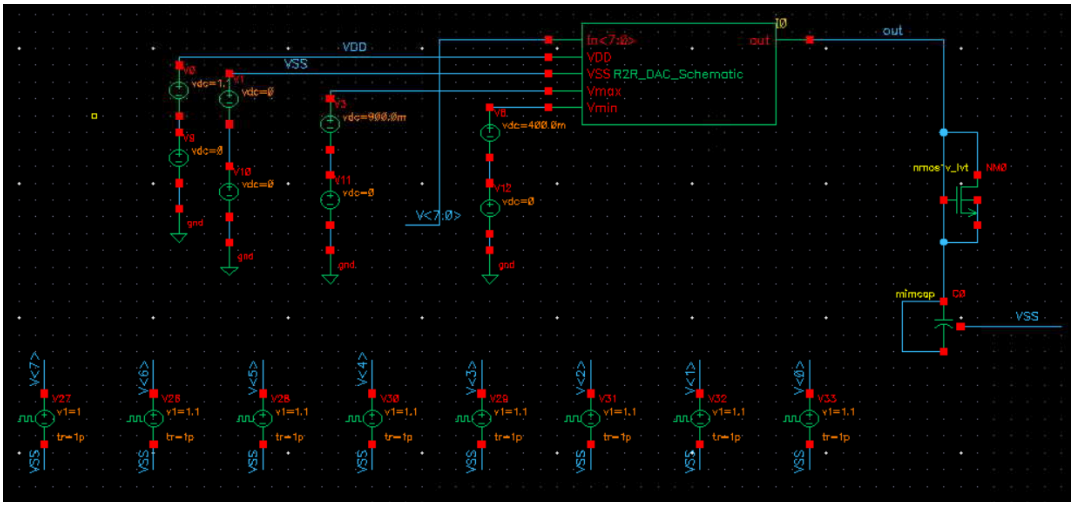
Step 3: Add Input Sources

1. **Digital Input Signals**:
    - Use a **pulse voltage source** for each input bit:
        - From the **Component Browser**, select:
            - Library: analogLib
            - Cell: vpulse
    - Configure each source to generate a square wave with a duty cycle of 50%.
        - Adjust the frequency of each pulse source to create a binary counting sequence.

- Example settings for B0 (least significant bit):
    - Voltage 1: 0V
    - Voltage 2: Vdd (e.g., 3.3V)
    - Period: 1µs
    - Delay: 0s
    - Duty Cycle: 50%
- For higher bits (B1 to B7), double the period for each subsequent source.

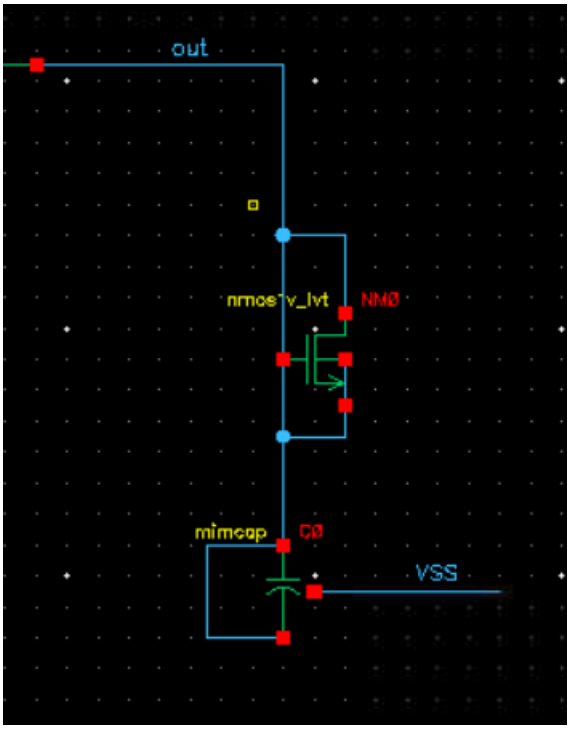


2. **Connect Inputs**:
    - Connect each vpulse source to the corresponding input (B0 to B7) of the DAC.



Step 4: Add Load and Measurement Points

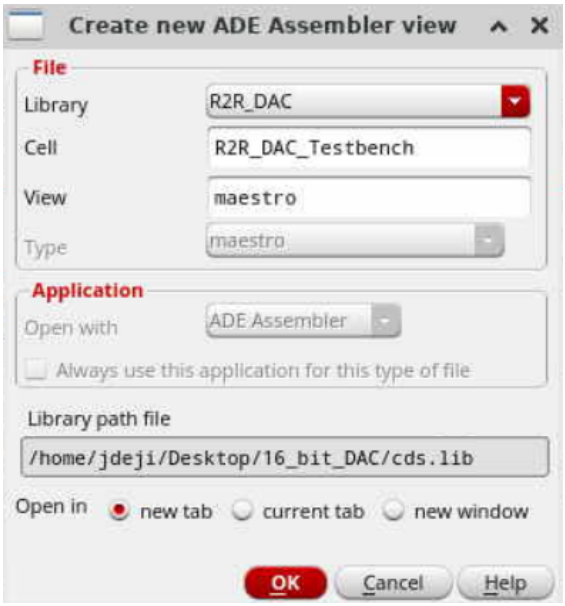
1. **Load**:



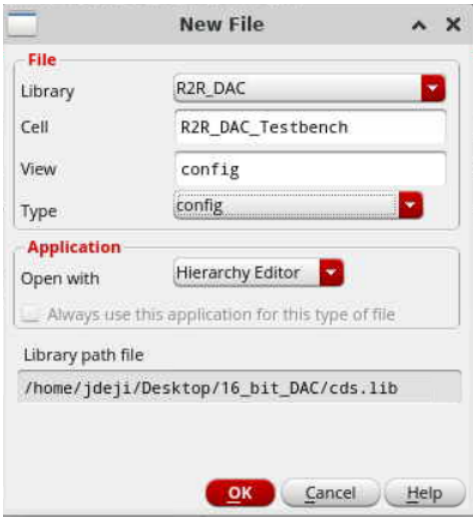
Step 5: Set Up the Simulation

1. **Launch ADE (Analog Design Environment)**:
    - From the Schematic Editor, go to:

Launch -> ADE Assembler -> Create New View



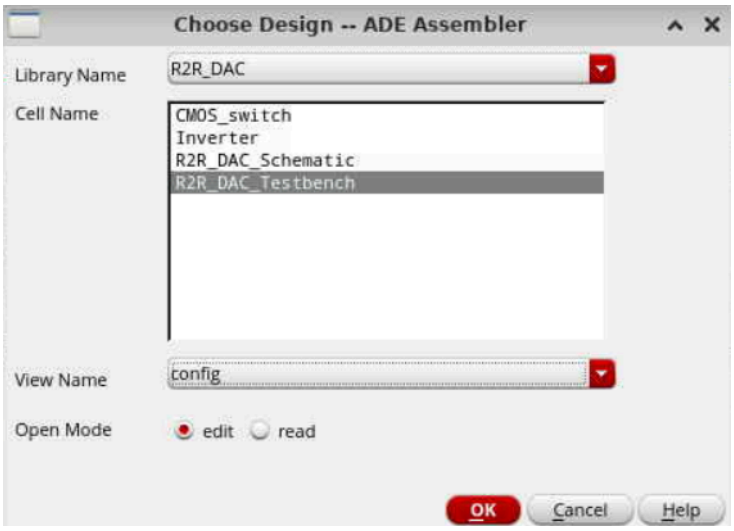
2. **Creating Testbench Config view**
    - Navigate to the Library Manager and Create a New Cell View:
        - Library: R2R_DAC
        - Cell Name: R2R_DAC_Testbench
        - View Type: Schematic



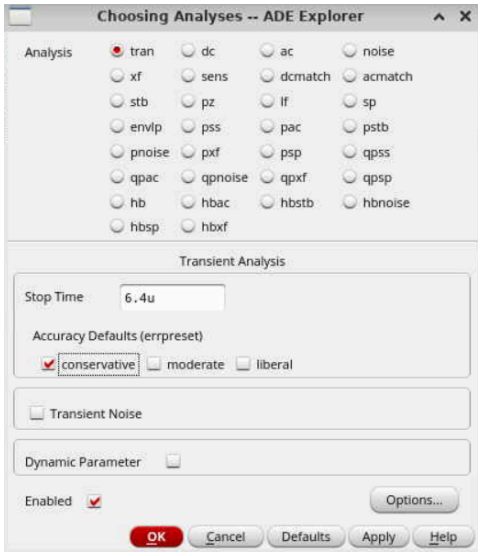
    - In the configuration window, set view as “schematic” and click “Use Template”
    - In the template window, select “spectre” as Name

- Click **OK** in both subsequent windows
- Save the config window and close it

3. **Create test and choose Simulation Settings**:
   - In the ADE expolorer create a new test using Create -> test

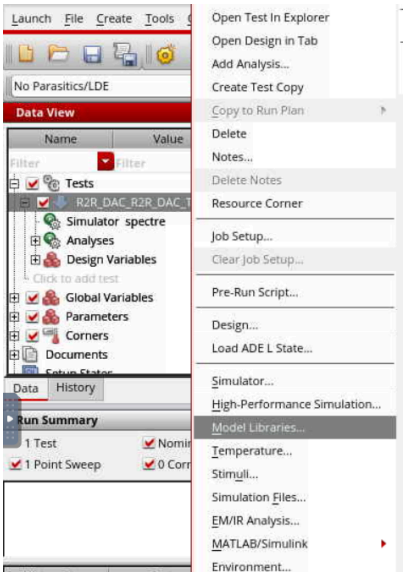


   - **Simulator**: Set to Spectre.
   - **Analysis Type**: Create an Analysis using Analyses -> Choose
     - Stop Time: Sufficient for a few full cycles (e.g., 6.4µs).

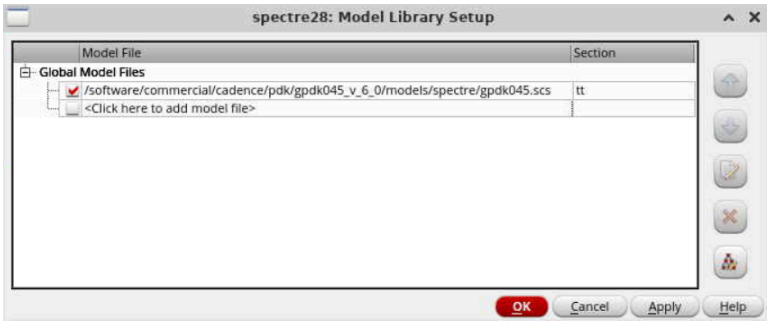


4. **Specify Outputs and Model Library file**:
    - Ensure you are in ADE Assembler by clicking the blue arrow next to the test name till if points down
    - Add the output voltage to the output variables for plotting: Click on the probe icon to add an output

    - Right Click the test from the assembler Data View Tab and choose Model Libraries…

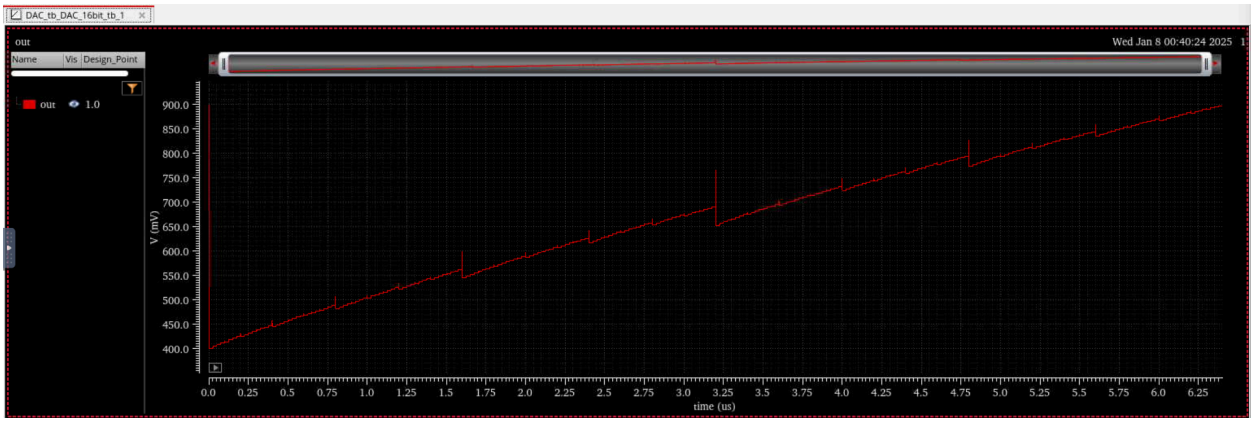


- Add the Model file as shown below and click OK



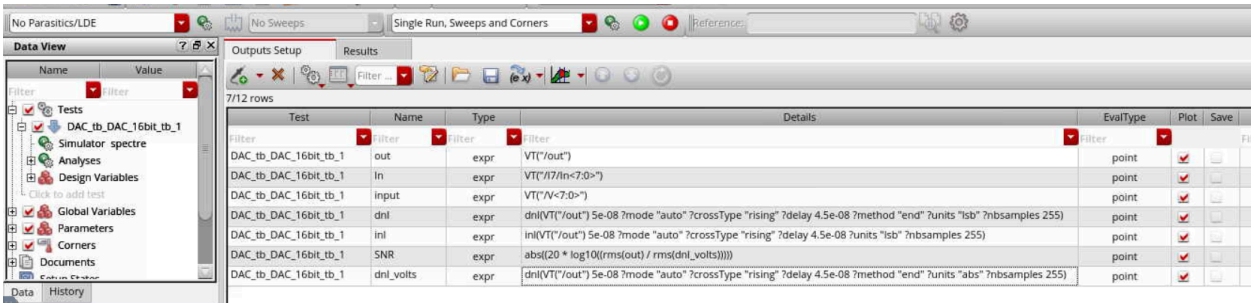
Step 6: Run the Simulation

1. **Run the Simulation**:
   - Click the **Run** button in ADE.(The green play button)
2. **View Results**:
   - Plot the output voltage (Vout) over time.
   - Analyze the staircase waveform corresponding to the digital inputs.



Step 7: Analyze the Results

1. **Verify the Staircase Output**:
    - Check that the output waveform correctly steps through levels corresponding to the binary-weighted inputs.
2. **Measure Linearity**:
    - Add the following measurements to your test to measure linearity



Next Steps

With the simulation complete, you've verified the functionality of your 8-bit R-2R DAC. In the next section, we'll explore how to optimize the design for performance, such as improving accuracy or minimizing power consumption.

Section 5: Conclusion and Further Learning Opportunities

In this final section, we'll briefly summarize what we've achieved in this project and discuss how students can expand their understanding of DACs, Cadence Virtuoso, and related concepts.

Step 1: Summarize the Tutorial

1. **Overview of Accomplishments**:
    - Designed an 8-bit R-2R DAC at the schematic level in Cadence Virtuoso.
    - Verified the DAC's functionality using transient simulations.
    - Optimized the design for accuracy, performance, and power efficiency.
2. **Key Takeaways**:
    - **R-2R Ladder Network**: Understand the significance of resistor matching for accuracy.
    - **Switching Mechanisms**: Learn the role of NMOS switches or transmission gates in digital-to-analog conversion.
    - **Simulation Tools**: Utilize transient analysis to validate circuit performance.

Step 2: Encourage Further Learning

1. **Explore Advanced Topics**:
    - **Layout Design**: Move beyond the schematic to layout-level design, including DRC and LVS checks.
    - **Parasitic Extraction**: Simulate the effects of parasitics on DAC performance.
    - **Alternative DAC Architectures**: Study weighted current sources, segmented DACs, or oversampling techniques.
2. **Cadence Virtuoso Features**:
    - Practice other analyses, such as AC analysis for frequency response.
    - Explore Monte Carlo simulations to study the effects of resistor mismatches and process variations.

Step 3: Suggested Projects

1. **Implement a 10-bit DAC**:
    - Design a higher-resolution DAC and observe how performance metrics scale.
2. **Build a Current-Steering DAC**:

- Experiment with a current-mode architecture and compare it with the R-2R topology.

3. **Combine the DAC with an ADC**:
    - Create a mixed-signal system by integrating an analog-to-digital converter (ADC) with your DAC.

Step 4: Additional Resources

1. **Documentation and Tutorials**:
    - Cadence Virtuoso Documentation: Explore in-depth guides on the tools.
    - Online Tutorials: Look for advanced tutorials on DAC design and mixed-signal circuits.
2. **Relevant Courses**:
    - Analog and Digital Circuit Design.
    - Mixed-Signal System Design.
3. **Textbooks and Papers**:
    - *Analog Integrated Circuit Design* by Razavi.
    - Research papers on DAC innovations and challenges.

Step 5: Wrap-Up

1. **Next Steps in Cadence Virtuoso**:
    - Continue practicing to deepen your expertise.
    - Consider how this project applies to real-world analog IC design challenges.

End of Tutorial