SECTION OF ELECTRICAL ENGINEERING

EPFL STI – SEL
ELG
Station nº 11
CH-1015 Lausanne

Téléphone : +4121 693 1346
Fax :
E-mail : alexandre.levisse@epfl.ch
Site web : https://sti.epfl.ch/fr/sel/

EE429 2025/2026: Full custom labs 1 : Schematic

SEL September 2025

PRACTICAL LABORATORY SESSION No. 1
Schematic Edition

1. OBJECTIVES

The goal of this labs is to learn how full custom design is being done in the industry. This first practical session is to get familiar with the working environment and software that you will use throughout the next few weeks. After setting up the Design Framework, you will perform the schematic design and learn the basics of the Virtuoso software from Cadence.

Note that this document contains important information that will save you a lot of trouble during the remaining exercise sessions. It is in your best interest to read it carefully, from start to finish, even though the tasks we ask you to perform may sometimes seem very simple.

2. INTRODUCTION TO THE DESIGN ENVIRONMENT

The software used for these practical exercises is referred to as **Cadence Virtuoso Design Environment**. It consists of a number of tools integrated into a common environment:

- The **Command Interpreter Window (CIW)** is the main window that provides an access to the variety of tools through menu commands, or through direct entering of commands in the scripting language called **SKILL**. All the important information, warnings and error messages are reported within this window.
- The **Library Manager** is the tool that manages your design data such as circuit schematics, layouts, simulation test benches etc.
- **Virtuoso** is the platform for creating and simulating your designs. It consists of the **Schematic Editor**, the **Layout Editor**, the **Analog Design Environment (ADE)** which is the graphical front-end for the circuit simulator, and many other tools.
- **Calibre** is the suite for full-custom layout verification and parasitics extraction. It allows the comparison of the layout with the schematic using the **LVS** (**Layout vs. Schematic**) tool, the verification of your layout versus the foundry's design rules (**DRC - Design Rules Check**), and the extraction of a detailed schematic containing parasitics (from the layout) for accurate post-layout simulation (**PEX - Parasitics EXtraction** tool).

3. CONFIGURING AND RUNNING THE SOFTWARE

3.1. CONNECTING TO THE SERVERS

In this labs, design and simulation tasks are being performed from specific server configured for circuit design tools. Integrated circuit design tools (commonly called EDA for Electronic Design Automation) require specific configuration which the EPFL EDA team had setup for you.

- **Generally, when working in a company, a dedicated team will do this step for you**
- **Still. You are expected to understand and know how to use and debug a linux environment. Basic commands can be found online[1].**

The picture below describes the environment you will use in the labs using EDA tools. From your computer (or the thin client), connect to VDI.
From VDI connect to the jed cluster. Run the EDA environment from there.

Why this setup ?

1- EDA tools and IC technologies restrictions
   a. Specific operating systems, libraries, network configuration, access limitation are required by the design tools.
2- Reliability :
   a. The link between the VDI VM and the EDA environment is stable
   b. You can turn off and on your computer or lose wifi without losing your work
3- Resources management
   a. Non-used VDI VMs are killed after 1.5h if not used, i.e., releasing non-used resources and software licenses.

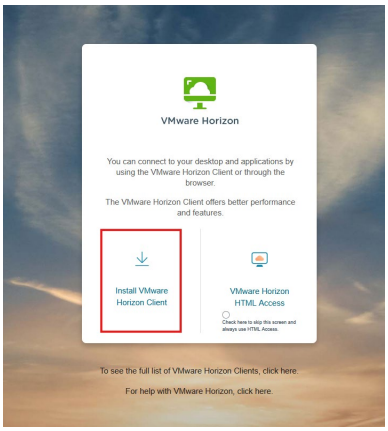
Gaspar credentials are mandatory to access the EPFL IT infrastructure. Without these, there is no much the teachers can do. So, make sure you have them with you. Be careful that the keyboard may not be the one you are used to. The computer room use a swiss keyboard qwertz layout[2].

[1] https://www.guru99.com/linux-commands-cheat-sheet.html
[2] https://kbdlayout.info/KBDSG/

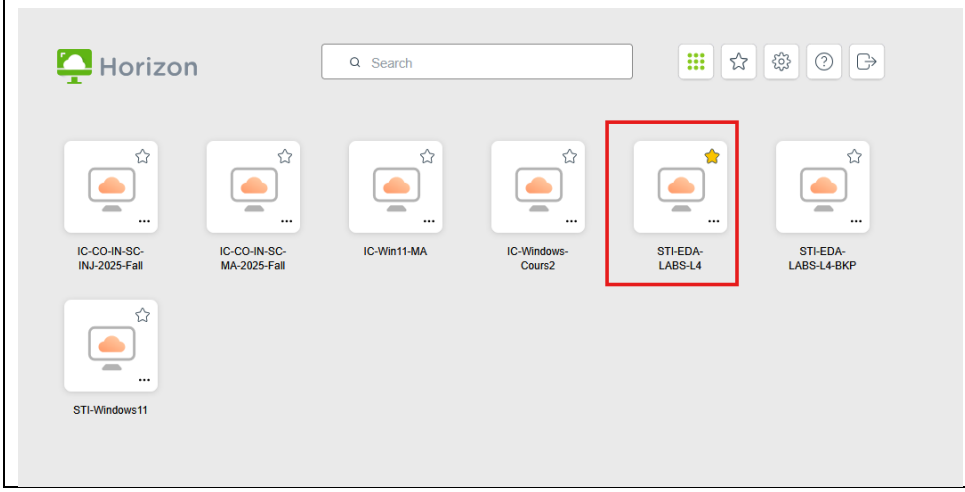
3.1.1. CONNECT TO THE VDI WORKSTATION

From a thin client in a computer room, connect to the VDI machine STI-EDA-LABS-L4.
From your own computer or a from windows computer in a computer room, EPFL VDI recommends the utilization of the Omnissa horizon client. The utilization of the web client shall be reserved for exceptional cases as it is less stable a more resources consuming.

1- Download the client :



2- From the client, select the VM STI-EDA-LABS-L4:



3.1.2. CONNECT TO THE LABS EDA SERVERS

From the spring semester 2024-2025, the EPFL EDA infrastructure migrated from selsrvs to the EPFL SCITAS HPC cluster[3]. This migration comes with the end of support of the historic selsrv1 and 2 servers, and provides strong improvements in terms of computing capabilities.
For this lab, we use the SCITAS jed cluster.

/!\ Disclaimer /!\

The access to the JED cluster for classes on EDA is solely reserved for educational activities and shall not be used for research. If you start a semester or master project within a lab and it requires EDA tools, please synchronize with your advisors and the EPFL EDA team (alexandre.levisse@epfl.ch) to get access to a dedicated environment.

To connect to the Jed cluster, from the linux VM, open a new terminal and type the following command :

```
> ssh -X username@jed.hpc.epfl.ch
```

Where username is your gaspar username.

To setup the EPFL EDA environment on SCITAS, run the following command :

```
>/work/fvlsi/run_edadk
```

This command shall be run for every new terminal. It does the following :

- Reserve and connect to a node in the jed cluster
- Configure the EPFL EDA environment
- Your reservation has the following parameters :
    - 4cores and 16GB of RAM
    - A 12h per session uptime – this means that after 12h your connection to the node cluster will automatically close. You would need to run the *>/work/fvlsi/run_edadk* command again.

Alternative solution if you do not want to type the complete path, you can create once a symbolic link in your home directory with the following command. Then you can directly start the environment from your home directory.

From the jed cluster :

> cd → *places yourself in your home directory : /home/username/*

>ln -s /work/fvlsi/run_edadk → *creates a symbolic link in your home directory*

Once you have a symbolic link, you can simply run the following command

[3] https://www.epfl.ch/research/facilities/scitas/

> cd → *places yourself in your home directory : /home/username/*

>./run_edadk

3.2. SETTING UP YOUR WORKING ENVIRONMENT

Start by creating a folder called FVLSI_LABS in your home directory. And go in it.

```
> cd
> mkdir FVLSI_LABS
> cd FVLSI_LABS
```

You can check your position in the hierarchy with *pwd* command. Run it and check where you are.

Each project is held in a separate directory which contains different configuration files and design data. Since many tools use configuration files that are stored in the *current* directory, the project directory actually defines a *working environment*. The file *"edabd2021_fullcust.tar.gz"*, contains the full and already configured working directory. Here we will only extract it and ***you will use the same directory throughout the laboratory sessions.*** **Please copy the directory as shown below. Please note that you can also use selsrv2 server.**

✓ Extract the directory for your project.

```
[1]username@jed.hpc.epfl.ch>
tar -xvf /education/classes/2025-2026/EE429/EE429_FULLCUSTOM.tar.gz
[2] username@jed.hpc.epfl.ch> cd EE429_FULLCUSTOM ↵
```

3.3. RUNNING VIRTUOSO DESIGN ENVIRONMENT

✓ Enter in the **EE429_FULLCUSTOM** folder

`username>` **`cd EE429_FULLCUSTOM ↵`** *(already done in the previous step)*

ℹ *It is **extremely** important to always start Virtuoso **from the Virtuoso project directory (here called EE429_FULLCUSTOM)**. The project directory contains many configuration files; therefore, the tool will not work as expected when run from a different place. Even worse, it may override other configuration files, especially when run from your home directory – this is a common mistake. This also applies for other tools that you will be using later, each one will have the corresponding working directory from where it needs to be launched.*

✓ Run the software by typing:

`>` **`virtuoso & ↵`**

✓ If a window appears asking for the available licences, always simply click **Yes** and proceed.

At this point, the software should start and the **CIW** window should appear (Figure 1).

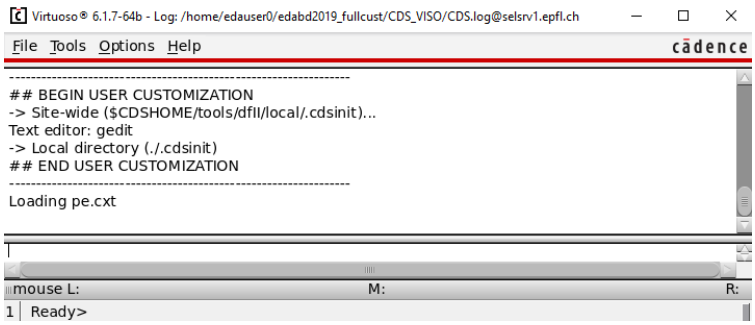


Figure 1 - The CIW (Command Interpreter Window)

✓ A text window may pop up displaying information about the design kit and the current version of Virtuoso. You can simply turn this window off. To prevent the window from opening every time you start the software, you can select ***File→ Close and Do Not Show Again*** from the window menu.

✓ To exit the design environment you can either:

- choose ***File→ Exit…*** from the CIW menu,
- close the CIW window, or
- typing `exit` in the command prompt at the bottom of the CIW window.

3.4. THE LIBRARY MANAGER AND DESIGN HIERARCHY

The Library Manager is a graphical interface used to organize the design data in the Virtuoso Design Environment. You can open it by choosing ***Tools→ Library Manager…*** from the CIW menu. Tick the "show Categories" box. The following window should appear:

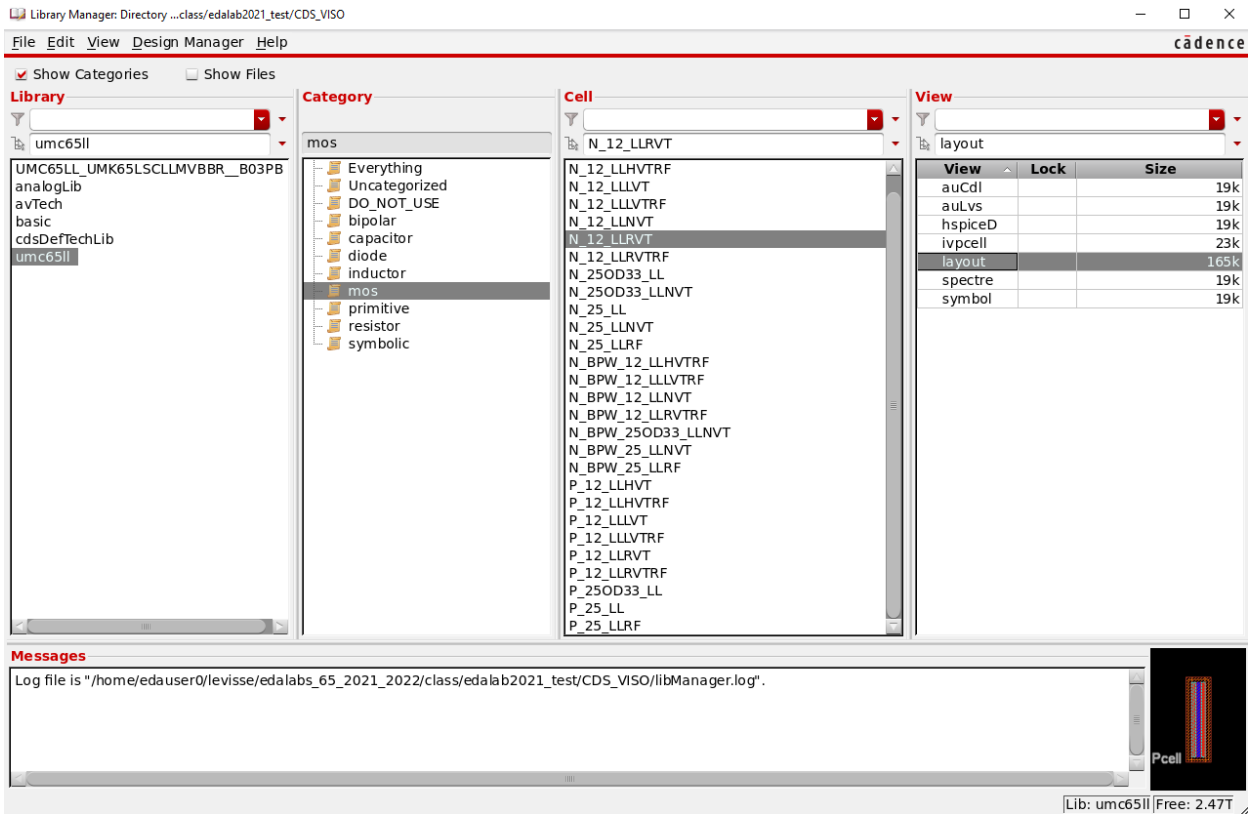


Figure 2 - The Library Manager Window

In the Virtuoso Design Environment, all the design data is stored in a collection of **libraries**. Libraries can be divided in different categories (subgroups), grouping the sets of related **cells**, each cell being an individual unit. Cells can have multiple **views** that can be different ways of representing the circuit or the simulation setup, waveform data, models, etc. For example, the same circuit can be represented as a symbol, a circuit schematic, or a mask layout, the testbench data can be saved in text files, etc. Figure 3 shows the organisation of cells and views in a library as a tree. The term **cellview** refers to a particular view of a particular cell, i.e. actual design data.

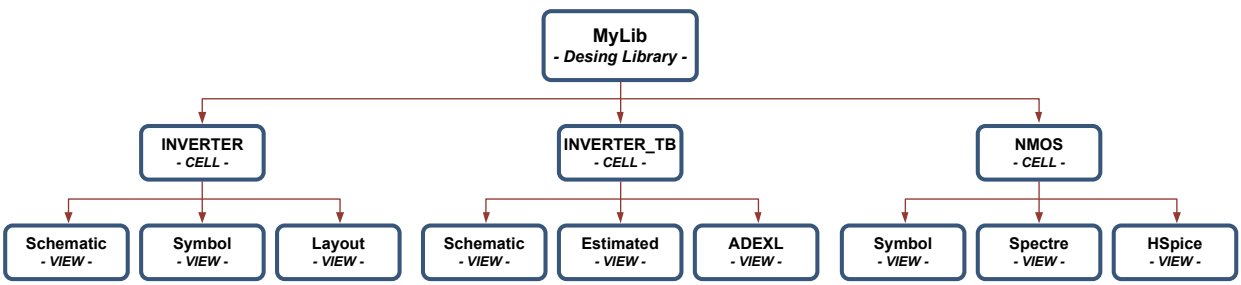


Figure 3 - Sample Library Structure

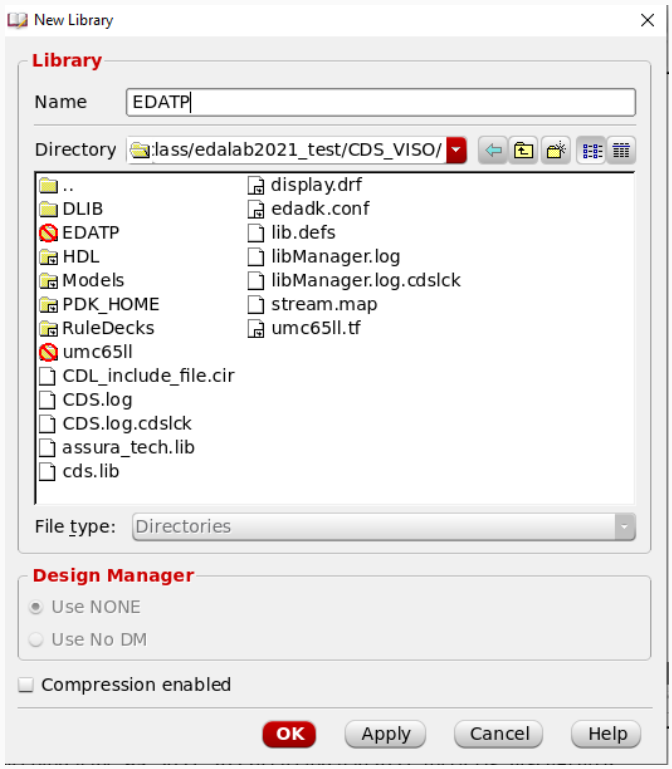
By default, a number of libraries will be available to you: some are the tool's default, and some are provided by the foundry design kit. The most important libraries in our case are briefly described in the following list:

- **basic** contains mostly graphical elements for circuit schematics.
- **analogLib** contains many elements useful for simulation such as voltage and current sources, ideal resistors, capacitors and inductors, switches, etc… These cells are mostly used to create simulation testbenches.
- **umc65ll (Primitives Library)** contains all the primitive devices (MOSFETS, resistors, capacitors, inductors …) from the UMC foundry design kit. You will use these devices in this (full-custom) part, and third (Analog Design) part of EDA-TP to create your own designs.
- **UMC65LL_UMK65LSCLLMVBBR__B03PB (Standard Cell Library)** contains the layout views from the UMC foundry design kit. You do not need to use these cells, but you can take them as examples to see how standard cell are usually designed.

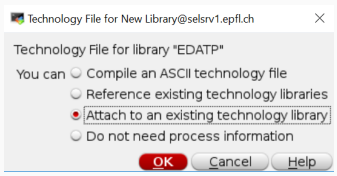
In the next steps, we will create a new library for your designs, and experiment some features of the library manager.

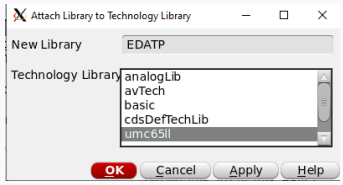
Creating a New Library

✓ Using the library manager, create a new library by choosing ***File→New→Library…*** from the menu. You are prompted to enter a name for your library: enter **EDATP**. Below is a space to specify the directory where the data will be physically stored on the disk. You don't have to change anything there.

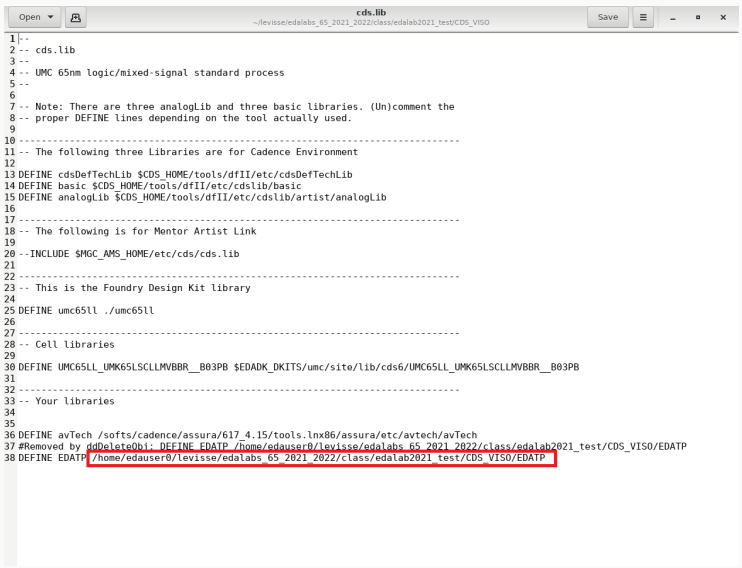


✓ You will be prompted to attach a technology file to the new library (this window sometimes appears below other windows - find it). The technology file contains technology-specific information, mostly related to layout, and is provided by the foundry design kit. Select ***Attach to an existing technology library*** and then choose the technology library **umc65ll**.





Handling Libraries



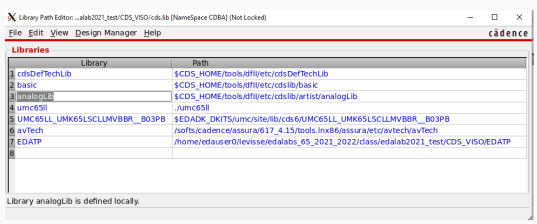
✓ Once you create your library, it is automatically defined in the file called ***cds.lib*** located in your Virtuoso project directory. This file contains paths towards all the libraries that will be available once you start ***Virtuoso Design Environment*** from this specific directory**.** You can check the content of this file by using any text editor (in this e.g. ***gedit***). Simply type:

```
> gedit cds.lib & ↵
```

✓ Note that you can also check the content of files with tools like ***cat*** (for small files), ***less*** and ***more*** (you can check how to use them online – to exit a ***less*** or ***more*** instance, type the ***q*** button on your keyboard) → `cat cds.lib`

✓ You will be able to see that your library has been added to the list of the existing default libraries. Please **do not** try to modify this file manually. After you observe that your new library is defined under the section *"Your libraries"*, close it.

✓ Try to find your ***EDATP*** directory that corresponds to the specific path. This is where your library data will be physically placed. Note here that the ***logical name*** of your library, defined in the ***cds.lib*** file after the ***define*** statement, **MUST** be the same as the destination directory of the library (marked by the red square).



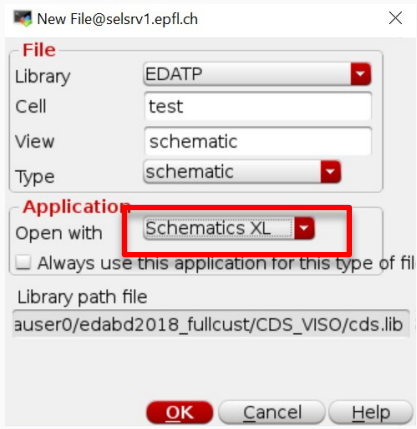
ℹ ***Very Important!*** *The directories listed in the cds.lib file including user defined libraries, contain data and info about all the cells and their cell views that are (or will be) defined within the specific libraries.* ***You should never*** *modify the content of a directory hosting a design library by yourself. Moreover, you should never define*

such a directory as a target of other tools such as simulators or DRC/LVS/PEX tools. Otherwise, unexpected errors and/or problems may occur.

- ✓ Instead of modifying ***cds.lib*** file manually, there is a "*safer*" way to perform this action if needed. In the ***Library Manager*** window, select ***Edit→ Library Path***. ***Library Path Editor*** window will pop-up. This editor provides an overview of the libraries defined in the ***cds.lib*** file and makes it possible to add, modify or remove some of the existing definitions.
- ✓ Keep in mind however, that ***cds.lib*** file and path editor, define only the paths to libraries. Therefore, removing the library path does not remove the physical library. If needed, library can be removed by using library manager. However, unless the library is not useful or unless specifically required, ***do not try to change the physical destination of the library***.
- ✓ In order to save your changes to ***cds.lib*** file, you need to select ***File→ Save*** in the library path editor menu. However, in this case we will not change our ***cds.lib*** file, so **do not** perform this action.

3.5. MANAGING CELLVIEWS

- ✓ Now create a cellview in your new library. Select **EDATP** in the *Library* field, and then choose ***File→ New→ CellView*** from the menu. A small dialog will appear (sometimes below other windows - look for it). Enter ***test*** as the cell name, and ***schematic*** as the view name and as the view type. Choose ***Schematics XL*** as the desired application (see figure).



  - *Notice the **Application** field. When changing the view **type**, the application name changes. This is because different tools are associated with different view types: schematic editor, symbol editor, layout editor, etc… Each view type has a standard name (i.e.* schematic *for a circuit schematic), but they can be changed. However, it is advised to keep the default names to avoid problems.*
  - ***Important!** When defining a name of the cell or library, typically only letters, numbers and underscore should be used. Avoid using special characters, such as: **- , . * / or \.***
  - ***Generally, it is a good practice to never start any cell name with a number.***

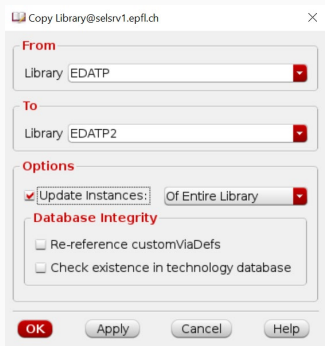
    ***Good examples**: "test_01","TestAmpifier4", etc.*

    ***Bad examples**: "test-01", "Op.Amp.2", etc.*
- ✓ A **cell** named ***test*** is created, with one view named **schematic**, and ***Virtuoso Schematic Editor XL*** appears to edit your new cellview.

✓ Choose ***File→ Check and Save*** (or ***Shift-X***) from the schematic editor menu to have the cellview data written to the disk. Then close the schematic editor.

Deleting, Copying and Renaming Cellviews

✓ In the **Library Manager**, select the **EDATP** library, right-click on it and in the context menu, choose **Copy.** The dialog window will appear. Change copy ***"To"*** field to **EDATP2,** check ***"Update Instances"*** box, and click OK. You will be prompted with another window asking you to specify the physical location of the library. Directory **EE429_FULLCUSTOM** should be chosen by default, in which case do not change anything, and click **OK** (if **EE429_FULLCUSTOM** is not a default destination, please set it manually). A new library **EDATP2** will be created.



✓ You can experiment with different commands: creating copying, renaming and deleting libraries, cells and/or views.

✓ Now right-click on the EDATP2 library and choose **Delete**. A dialog will appear specifying the libraries to delete (EDATP2 should be on the Delete list). Click **OK**, and in the following window click **Yes**. The library will be deleted.

4. THE GRAPHICS EDITOR

The different tools for graphical editing (schematic editor, symbol editor, layout editor) use a similar interface named the **graphics editor**. We will now experiment with the features of graphical editing by creating a simple circuit schematic.

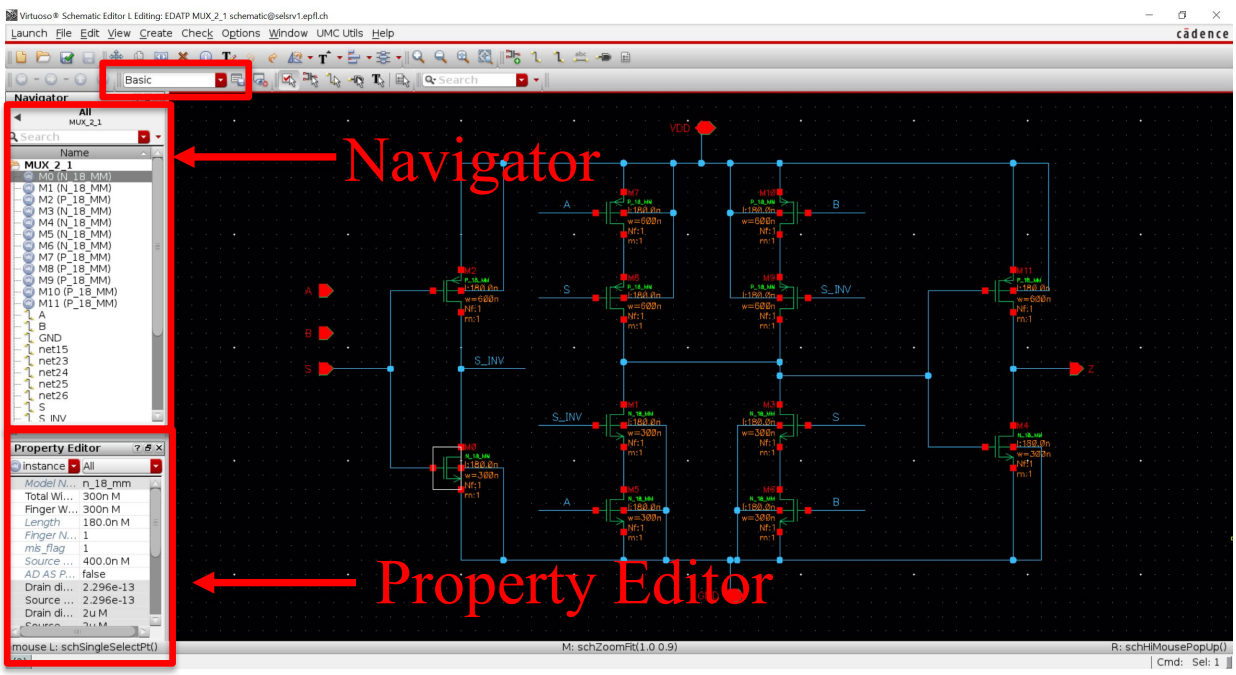
> ✓ Create a new schematic cellview named **MUX_2_1** in your EDATP library. The schematic editor will appear.

If the workspace is set as ***basic***, you will be able to see ***Navigator*** and ***Property Editor*** design assistant sub-windows. Navigator allows you to quickly browse through all the instantiated devices in your design. Property Editor shows all the parameters of the currently selected device, allowing you to make some quick changes.

In the next steps, you will learn how to use the ***Schematic Editor XL*** by creating a simple CMOS Multiplexer schematic shown in the figure below.

> ℹ You can note that if you right click on a schematic cell view and select "open with", several options are available for the application. Specifically Schematic L or XL. Here, we make you practice with XL, though note that you could use L. For the schematic editor, the difference is small. Though XL generally gives access to more features accessible from the workspace type. Keep it to basic for these labs.

Figure 4 - The Schematic Editor Window



4.1. CREATING INSTANCES

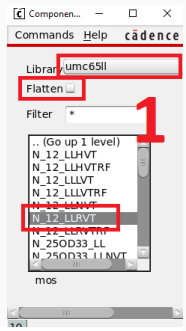
The term **instance** denotes the occurrence of a cell inside another cell. A cell can be **instantiated** multiple times in another cell. Instances define a hierarchical relationship between cells, where the cell containing the instance(s) is higher in the hierarchy than the instantiated cell.

- ✓ Choose ***Create→ Instance*** from the menu, or use the keyboard shortcut for this command by simply pressing ***i*** on the keyboard.

  *Most commands have a keyboard shortcut, or **bindkey**. They are shown on the right of the command name on the menus. Learning how to use and to remember the bindkeys may seem complicated in the beginning, but it can save you a lot of time in later stages of the project.*

- ✓ Depending on the version of Cadence one of the two windows (on the right, marked as ***1*** and ***2***) will pop-up.

- ✓ In case it is the window marked as ***1*** (on the right), select the **umc65ll** library. Then you need to select the **N_12_LLRVT** cell for instantiation. For that, either select the **Mos** category and select **N_12_LLRVT** cell; or check the **Flatten** option so that all the cells from different categories appear under the same list, then select the **N_12_LLRVT** cell. Just after selecting the **N_12_LLRVT** cell, an option form (window marked as ***2,*** on the right) appears which prompts you for the library name, cell name and a desired view of the cellview you wish to instantiate. Since you have already selected the desired library and cell only make sure that the view is written as **symbol**.



- ✓ In case it is the window marked as ***2*** (on the right), choose **umc65ll** library and type **N_12_LLRVT** and **symbol** for the symbol view of the NMOS transistor. By clicking on the **Browse** button, you can see the window marked as 1 on the right hence you can select the desired library and cell by applying the steps which are explained in the previous item.

- ✓ Return to the schematic editor window **without closing the option form** and place 6 NMOS transistors in your schematic as in the Figure 4 or Figure 5. To turn them sideways, you can use ***Sideways*** button on the ***Add Instance*** window (see the figure - right). When you are done, either press **Escape** or click **Cancel** on the option form to stop placing transistors.

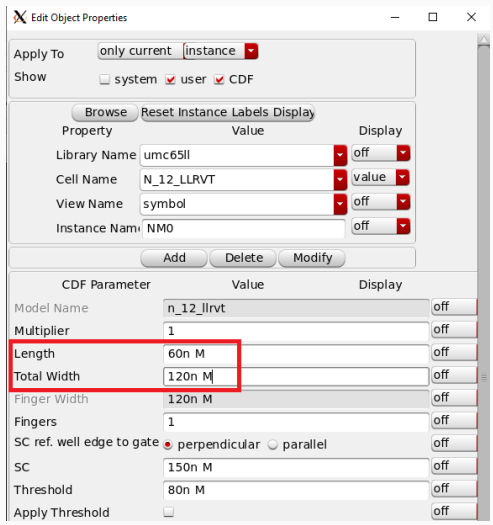
  Notice how you can add more instance as long as you do not cancel the command. Many commands work in this manner: activating the commands bring you into a new "mode" that lasts until you press Escape. Some commands, however, only work once.

  *The "Add Wire Stubs at" allows you to automatically create wires on the terminals of your instances. If you select "registered terminals only" and click on "…" you are directly define label names for the corresponding wires. Alternatively, you can also press **spacebar** from the schematic view to create wires on a selected instance.*

- ✓ Repeat the same procedure, and place 6 PMOS transistors. Select the **umc65ll** library, choose **Mos** category and select **P_12_LLRVT.** Alternatively, simply type **P_12_LLRVT** in the ***Add Instance*** window instead of **N_12_LLRVT.**

  ***It is very important to save** your design as often as possible. Otherwise, if something goes wrong, you may lose all of your unsaved work. You can save your schematic by selecting **File→ Save (Schematic)** from the main menu. Or click on this icon:* 

4.2. EDIT OBJECT PROPERTIES



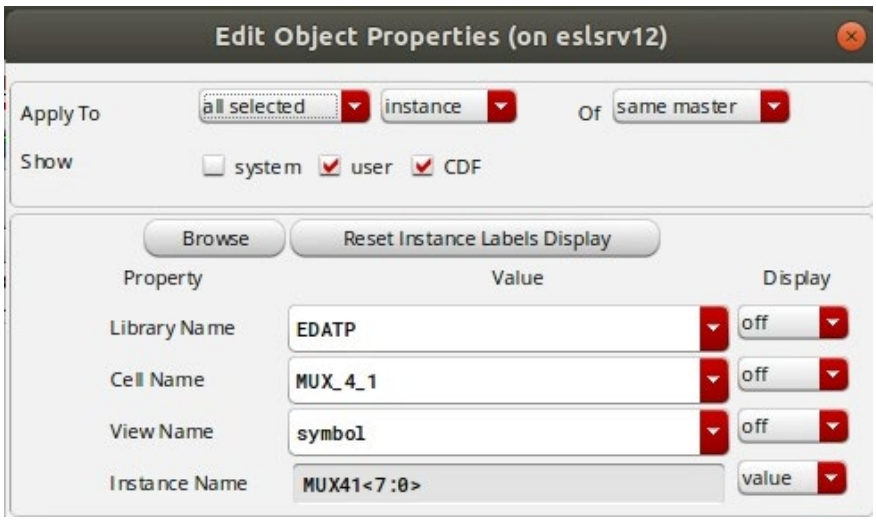
- ✓ First, we will edit the properties of the NMOS transistors. We will do this in a ***one-by-one*** manner.
- ✓ Left click on an NMOS transistor to select it. Choose ***Edit→ Properties→ Objects*** from the editor main menu, or simply click on a bindkey: ***q***. Object properties window will appear, displaying all the properties and the parameters of the selected cellview (**N_12_LLRVT**).

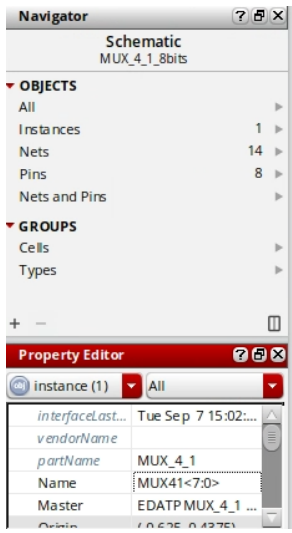
  ℹ *Object property window can be used to quickly change the cells or cellviews. You can try rewriting **N_12_LLRVT** into **P_12_LLRVT** and clicking OK. The cell in the schematic will change to PMOS. **Switch it back to NMOS** before you proceed!*

- ✓ You'll notice two lines called **Length** and **Total Width** in the **Object Properties** window. Length corresponds to the transistor length. In the 65nm technology, the minimum Length is 60nm. Always keep it 60nm in this phase of the FVLSI_LABS . **Total Width** is equal to **Fingers** * **Finger width** parameters.
- ✓ Click on the ***Total Width*** box and change the width of the NMOS transistor to *120nm*. Click OK and repeat the procedure for all the six NMOS transistors in the design.

  ℹ *Take a look at the updated cell list in the Navigator sub-window. All the NMOS and the PMOS transistors that you added are listed there. Find which transistors are PMOS and which are NMOS.*

ℹ In case the tool does not let you modify the instance name (this may happen in some conditions, cf screenshot). First, check that the file is not read-only. If yes, make it editable. **File>Make Editable**. Otherwise, use the left panel as shown in the right-hand side screenshot.

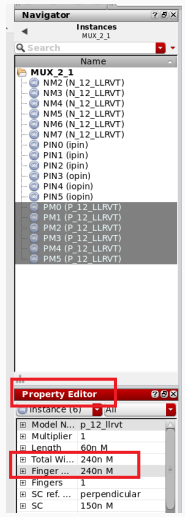




Using Navigator and Property Editor

- ✓ We can also perform the same action on multiple instances ***all-in-one-step.***
- ✓ Hold a **CTRL** key and click on (select) all the PMOS transistors in your ***Navigator*** window. Notice that they will be selected in the schematic as well.
- ✓ Observe that the ***Property Editor*** will now show all the mutual parameters for the selected cells, so there is no need to enter the object properties window.
- ✓ Make sure that the falling menu in the top right of the Property Editor is set to ***All***. Left click on the ***Total Width*** number field and change the width of all the PMOS transistors to *240nm*. Note that the width of all the PMOS transistors will be set to *240nm.*

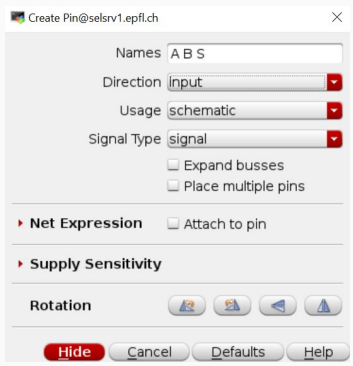
  You can also select multiple cells by holding SHIFT key and clicking on the desired devices directly on the schematic. The Navigator window and the Property Editor will be updated accordingly. Also notice that in this case you can unselect a selected device by holding CTRL and clicking on the device. Moreover, CTRL+A works.



4.3. Creating Pins

Pins define the connections (interface) between a cell and its environment. Pins are defined by the name and the direction (input, output or input-output). The purpose of the direction is to check for wrong connections (e.g. two outputs shorted together, or floating inputs). Typically, input-output pins are used for power supplies and bidirectional interfaces.

- ✓ Choose ***Create→ Pin*** from the menu *(bindkey: **p**)*. The option form appears, prompting you to enter the name of the pin as well as different options (see figure). Enter ***A, B*** and ***S*** as the pin names, and choose ***input*** as the pin direction. Click ***Hide*** and place the pins on the schematic one by one. The pins will be placed in the same order as you specified them.
- ✓ Once this is done, return to the option form *(bindkey: **p**)* and create the supply pins ***VDD*** and ***GND*** with the direction set as ***inputOutput***. Place ***VDD*** and ***GND*** on your schematic.
- ✓ Repeat the procedure to create the output pin ***Z***. Place the output pin on the schematic.



4.4. Creating Wires and Labelling Nets

Wires define the connections between the different instances in a cell. Wires can connect to the pins, instances, or to other wires. All the connected wires are electrically at the same potential and together define a **net**. Nets can be labeled to make the schematic and simulation results more readable – if they are not labeled, the name will be assigned automatically.

- ✓ Choose ***Create→Wire (narrow)*** from the menu *(bindkey: **w**)*, and add wires to your schematic to connect the different elements as shown on the Figure 5 (next page).
  - *Use the **Zoom (View→Zoom In…** or **]**) and **Fit (Window→Zoom to Fit…** or **f**) commands to adjust the zoom. (Please note that zoom in and zoom out can also be done by turning the wheel of the mouse.) You can even use these commands while in the middle of creating a wire without interrupting.*
- ✓ Choose ***Create→ Wire Name*** from then menu *(bindkey: **l**)*. In the option form, type ***A, B, S*** and ***S_INV,*** and name the corresponding wires as shown in the Figure 4 (before) or in Figure 5 (next page). Finally, check if your schematic corresponds completely to the Figure 5 (be careful - check the transistor bulk connections).
  - ***Important!*** *Try to add wire names (use labels) for all the important wires (nets) in your design. After simulation, automatic names can be hard to distinguish and trace, especially in large schematics.*

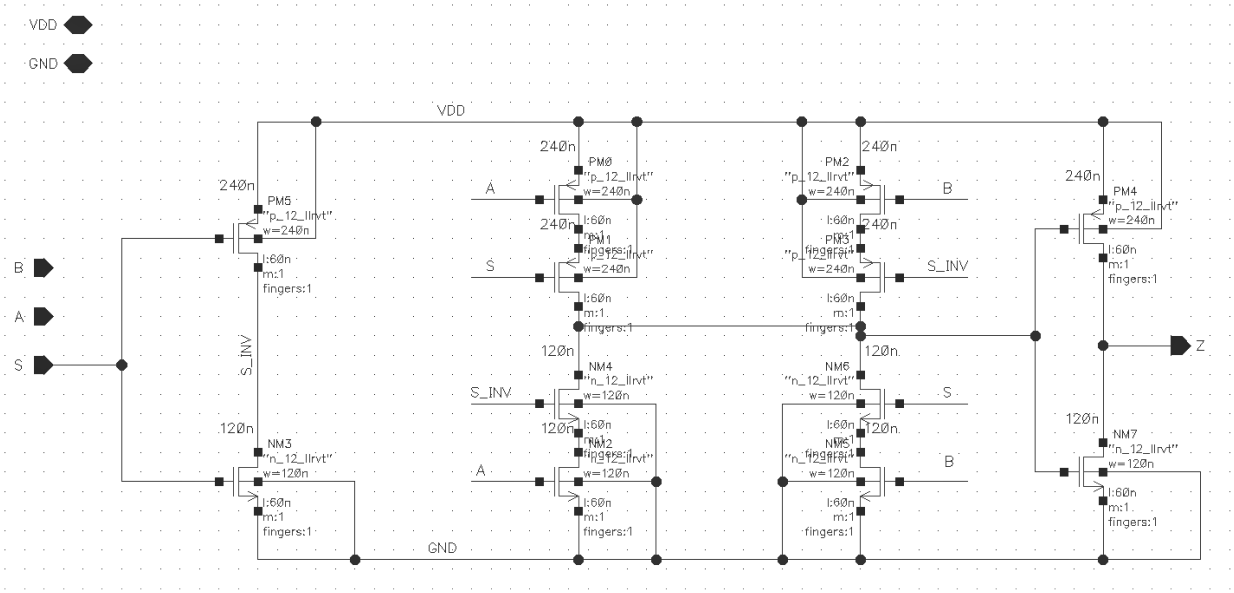


Figure 5 - The Final MUX 2-to-1 Schematic *(S=1 => Z=B; S=0 => Z=A)*

QUESTION : what is the different between a Pin and a Label ? What does a Label do ? What does a Pin do ?

- *To help you with the understanding, you can enable/disable a function called "Net Highlighting" in the "view" dropdown menu of the schematic editor.*

QUESTION : Look at the two inverters in the figure 5 schematic. Based on your knowledge of the sizing guidelines, assuming these inverters are balanced, what can you tell about the carrier mobility ?

4.5. CHECKING AND SAVING THE SCHEMATIC

- ✓ Choose ***File→ Check and Save*** (or ***Shift-X***) to save your design. The schematic will be checked for errors. If there are errors or warnings, a dialog box will inform you about them. The **CIW** will display a detailed message for each problem found.
- ✓ Go to the **CIW** window and check the messages. If no problems were found, you should find a message similar to the following:

```
Extracting "MUX_2_1 schematic"
Schematic check completed with no errors.
"EDATP MUX_2_1 schematic" saved.
```

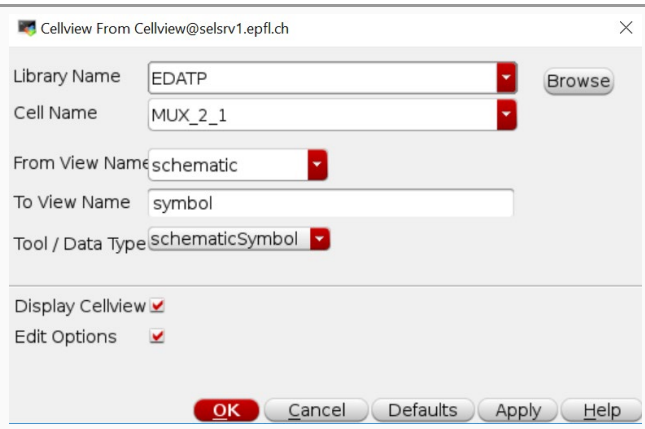
When something goes wrong, always check the CIW for error or warning messages.

4.6. GENERATING THE SYMBOL

A symbol is a graphical abstraction of a cell that provides only the necessary information for using the cell at a higher hierarchical level – that is, it provides information on how to connect the cell from the outside. Symbols also provide a visual clue about the function of the underlying circuit. As such, symbols are only made of pins (connections) and graphical shapes.

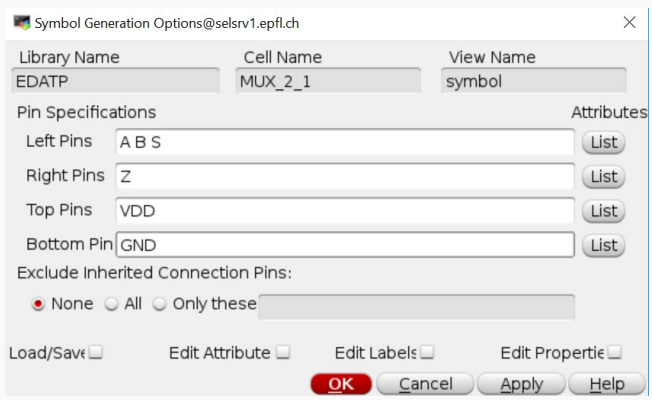
Symbols can be created manually by choosing the **Composer-Symbol** tool when creating a new cellview, then drawing the shapes and pins. However, it is also possible and much more convenient to have them generated automatically. A square box with pins is generated that can then be modified if needed.

- ✓ From the Schematic Editor menu, choose ***Create→ Cellview→ From Cellview***.
- ✓ In the first form coming up, all options should be set correctly. Press **Ok**.
- ✓ In the second form, you get a chance to specify the location of the pins on your symbols. It is common to have input pins on the left, output pins on the right, and power/ground pins on the top and bottom of the symbol. When you are done, press **Ok.**



✓ Shape your symbol however you like, then **Check and Save** it.

*You don't need to change the labels [@instanceName] and [@partName] in the generated symbol. When you instantiate the cell, these labels will display the instance name and the cell name respectively. By default, the instances you place in a schematic will be named I0, I1, I2, etc. To change the name of an instance, select the instance and press **q** to open its Object Properties dialog.*



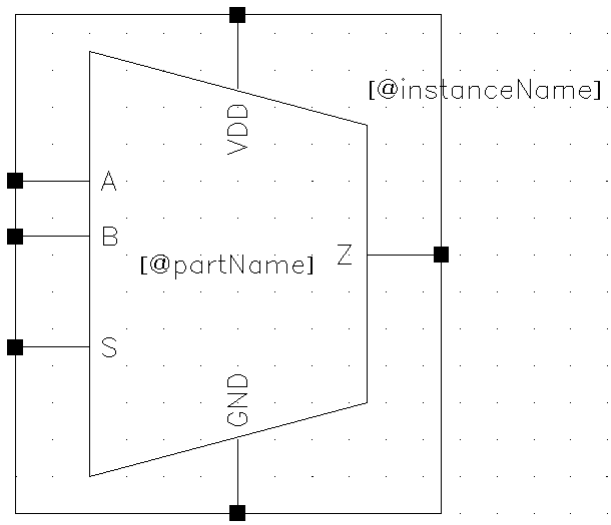
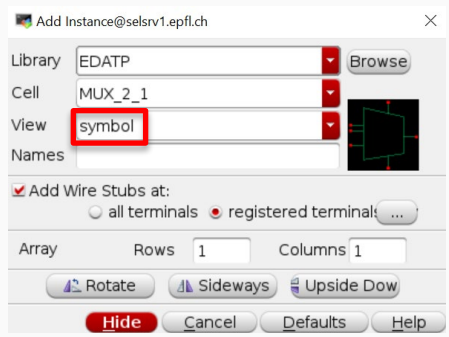


Figure 6 - An Example of the Modified MUX Symbol

4.7. CREATING A HIERARCHICAL CIRCUIT

In the following steps, you will create the circuit schematic for a 4-to-1 multiplexer by using your 2-to-1 MUX cells.

- ✓ Create a new schematic in your **EDATP** library. Name it **MUX_4_1**.
- ✓ Draw the schematic as shown in Figure 7. Use the created **MUX_2_1** symbols and instantiate them in the new cell.
- ✓ While instantiating (add instance window - see on the right), you need to provide every instance with a meaningful name, otherwise it will be given a default name (such as: ***I0, I1, I2...***).



- ✓ Name your instances exactly as in Figure 7: ***MUX1, MUX2*** and ***MUX3***.

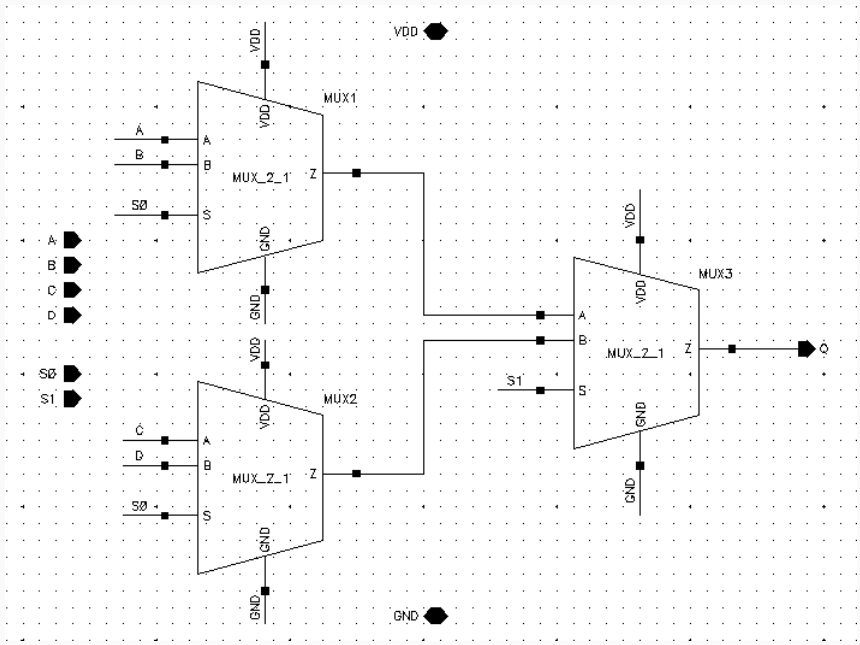


Figure 7 - The 4-to-1 multiplexer circuit schematic

- ✓ When you are done, **Check and Save** your schematic. Correct any errors or warnings, until no more are reported.
- ✓ Create a symbol for the 4-to-1 multiplexer.

4.8. MOVING, STRETCHING AND DELETING OBJECTS

- ✓ Open the **MUX_4_1** schematic. Choose ***Edit→ Move*** from the menu or press ***Shift+m***. Left click on a **MUX_2_1** symbol, and drag it to a different location. Once you decide on a new location and left click again to execute the move.

  - *Multiple objects can be selected at the same time. After selecting **Edit→ Move**, left-click and drag the mouse to select multiple objects. Left-click again to define a reference point, and you will be able to move a group of objects.*

- ✓ Choose ***Edit→ Undo*** or press **u** to cancel the move. (If you want to do the same move again, use ***Edit→ Redo*** or ***Shift+u***, then to cancel again, you can use ***Edit→ Undo*** or press **u.**). Then press ***Ctrl+d*** to clear the selection, or you can simply click on an empty place in the schematic.

  - *Please note that **Edit→ Undo (**bindkey: **u)** and **Edit→ Redo (**bindkey: **Shift+u)** does not only work with move command. They can also be used to undo and redo other commands and can be quite useful for accidental use of commands.*

  - *Commands that work on objects, such as move, stretch or delete, need a **selection** to work on. If an object or a set of objects is selected before applying the command, it will operate on this existing selection. If not, you are prompted to select the objects before you perform the move. **Once you finish the command, press ESC to exit the specified command mode!***

    *Notice that when there is an existing selection, you can move the object only once before the command exits, while you can move multiple objects when there is no prior selection. Thus, in one case you can apply multiple actions, sequentially, to a set of selected objects, and in the other case you can apply the same action to a number of sequentially selected objects. Remember **Ctrl+d** to clear the selection.*

- ✓ Repeat the same manipulations with the **Stretch** command instead (***Edit→ Stretch*** or ***m***).

  - *Observe that with the **stretch** command, wires connected to the instance are rerouted to keep the connections, while with the **move** command, the selected objects were moved regardless of the connections. **Stretch** also allows to reshape existing wires.*

- ✓ When moving or stretching an object, an option form will appear (if not, you can use **F3**.). In the option form showing up, there are buttons for **rotating** and **mirroring** the instance. Try these.

- ✓ *Many commands have an option form which does not always show up automatically. Use the **F3** key to show or hide this option form.*

Deleting Objects

- *The **Delete** command (**Edit→ Delete** or **Del**) works in the same way as **Move** or **Stretch** with respect to the selection. **Objects can be accidentally deleted if they are selected prior to pressing Del.***

- *Pressing **Del** before any selection, starts a **delete mode**. Any object selected while in delete mode will be deleted. **Be careful!***

- *Note also that depending on the mode (Move, Stretch, Delete...), the mouse cursor changes the shape... Remember, you can always exit the specified mode by pressing **Esc**.*

4.9. MOVING UP AND DOWN THE HIERARCHY

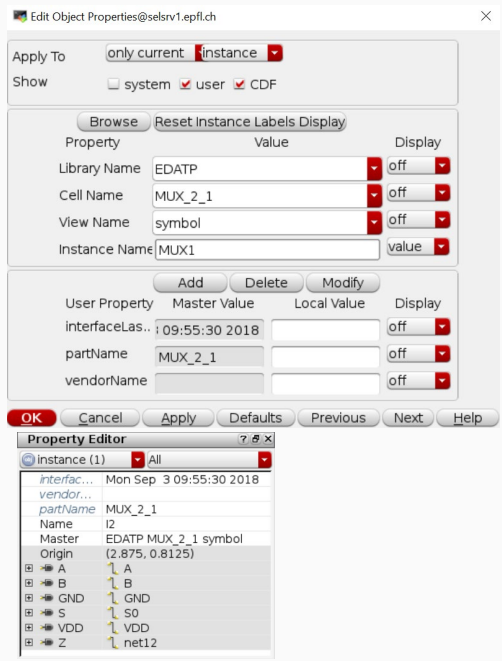
Now that you have designed a hierarchical schematic (**MUX_4_1**), you can try the commands for moving up and down the hierarchy.

- ✓ Open the **MUX_4_1** schematic.
- ✓ Select one of the **MUX_2_1** instances.
- ✓ Choose ***Edit→ Hierarchy→ Descend Edit*** or press ***Shift+e***. You will be prompted to select a view: choose the **schematic** view and press OK.
- ✓ The current schematic will be changed to **MUX_2_1**. To return up the hierarchy to **MUX_4_1**, choose ***Edit→ Hierarchy→ Return*** or press ***Ctrl+e.***

4.10. INSTANCE PROPERTIES

For the MUX instances you created, you can use same property editing methods as it was done for NMOS and PMOS transistors earlier.

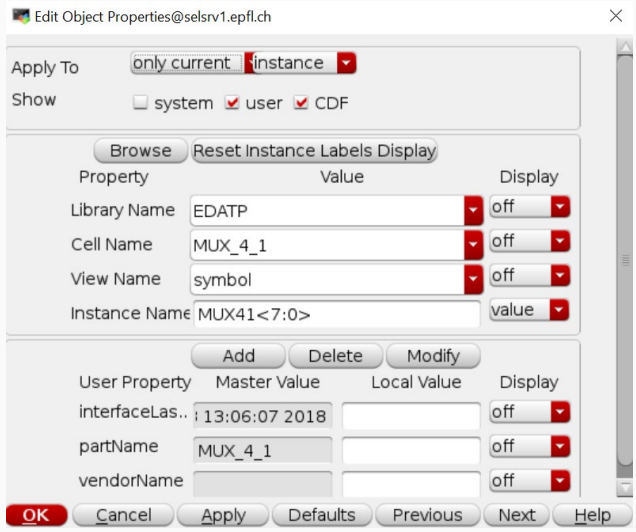
- ✓ Select one of the **MUX_2_1** instances.
- ✓ Enter the object properties window (***Edit→ Properties→ Objects,*** or press: ***q***). Here you can modify the name of the instance (see figure on the right).
- ✓ Make sure once again your MUX names correspond to Figure 7.
- ✓ Check the ***Property Editor*** window. There you can also see the object properties for the selected instance, and eventually change the instance name.



4.11. USING MULTIPLIED INSTANCES AND CREATING BUSES

Cells can be instantiated multiple times using a single symbol, which is very useful for large designs (imagine having to place 256 multiplexers in your schematic). We will learn how to do this by designing an 8-bit 4-to-1 multiplexer, from our single-bit version.

- ✓ Create a new schematic cellview in EDATP library called **MUX_4_1_8bit**.
- ✓ Choose ***Create→ Instance*** from the menu *(bindkey: **i**)*, and add an **MUX_4_1** instance. Specify the instance name with a desired number of instances. For example: **MUX41<7:0>.**
- ✓ This creates 8 multiplexer instances numbered from 7 down to 0.
- ✓ Alternatively, you can also add a single instance, select it, and then enter the object properties window (***Edit→ Properties→ Objects,*** or press: ***q***). Here you can also modify the name and the number of instances (see figure right).
- ✓ Another alternative is to use ***Property Editor*** sub-window. When you select an instance you can modify the name and the number of instances within the sub-window directly.

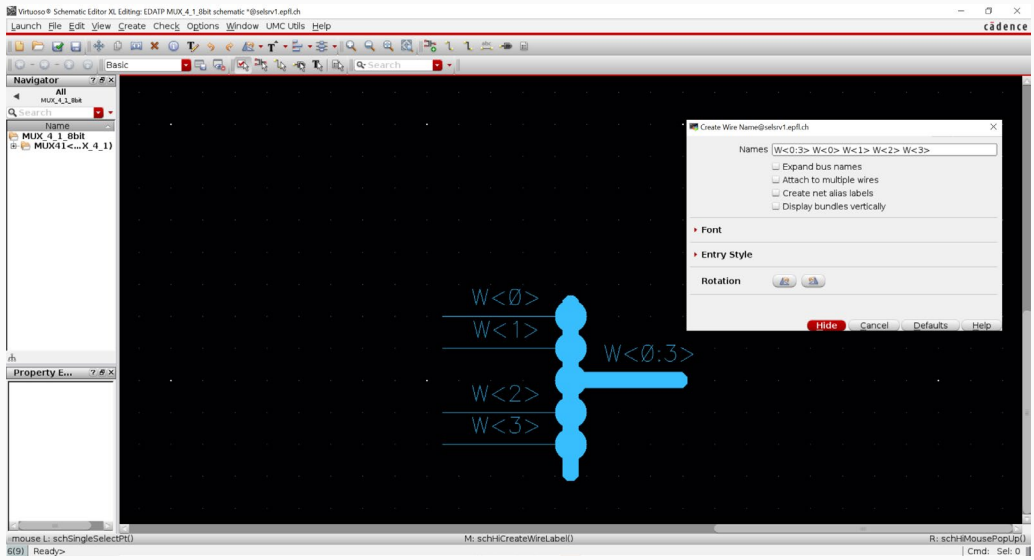


Creating Buses and Wire Bundles

Wires can be grouped in **buses**, where each wire is following the common naming convention. E.g. if we have a 4-bit bus **W<0:3>** (or **W<3:0>**), then the wires that are forming the bus must be labeled as **W<0>**, **W<1>**, **W<2>** and **W<3>**. The '<>' brackets are marking the bus expansion.

- ✓ Choose ***Create→ Wire (wide)*** from the menu *(bindkey: **Shift+w**)*, and add a bus that connects all the wires that should be a part of this bus.

- Choose ***Create→ Wire Name*** *(bindkey: **l**)* from then menu. In the option form, type **W<0:3>** (you can also use **W<3:0>**) for the bus, and **W<0>**, **W<1>**, **W<2>** and **W<3>** for the wires that are forming that bus.
- Alternatively, to create the wires, type only **W<0:3>** in the option form and select **Expand bus names.** This way you can place the labels in one go.

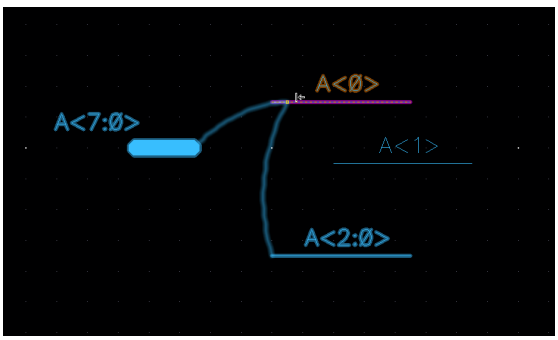


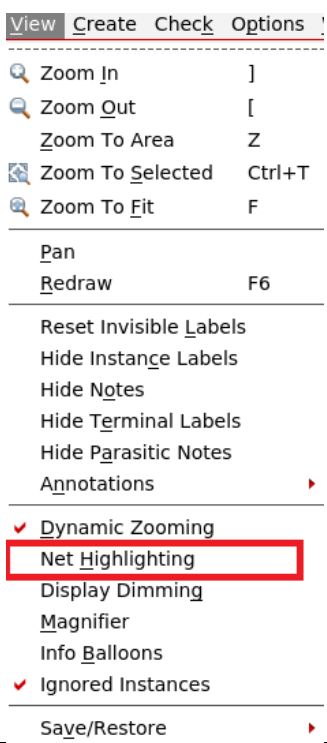
Buses do not specifically need to be built with “wide” wires. You could set them up similarly with small (w key) wires. Always keep in mind that all the labels and pins are connected by name. Though, you could not physically connect “small” wires.

To check the connectivity of wires, you can use the “net Highlighting” tool from the View menu. You can enable and disable it by ticking and unticking it. it is a really useful tool to check if some wires are properly connected.

- Note that you must check and save your design, in order for it to work.

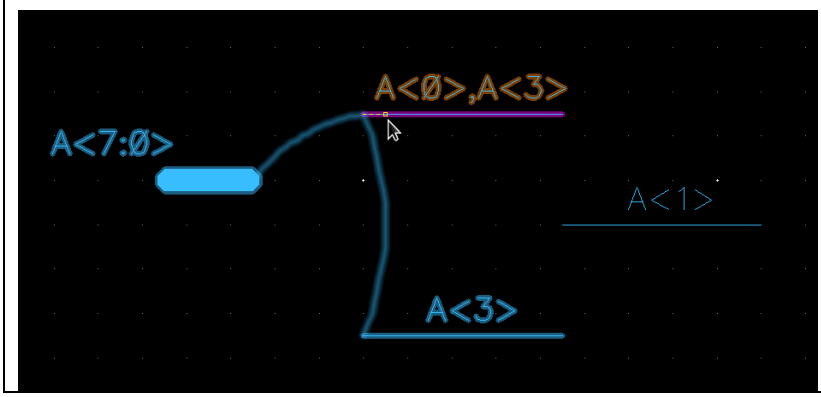
In the following screenshot note how both large and small wire behave the same way. A<0> is indeed part of A<7:0> and A<2:0>.





Bundling wires is a good way to create buses of wires which do not have the same names.

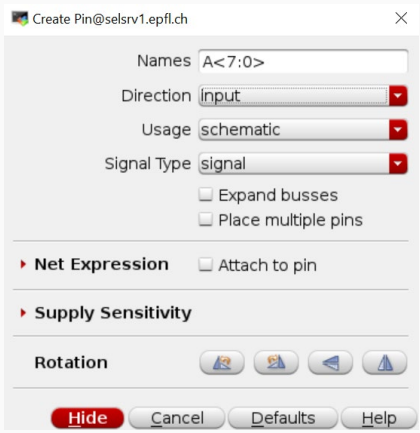
In a wire label, you could for e.g. use the following syntax: "A<0>, A<3>". This will create a 2bit bus that has A<0> as MSB and A<3> as LSB. Note how in the following screenshot this bus is actually connected to A<7:0> through A<0>. Obviously it is connected to A<3> as the bundle contains A<3>. But it does not connect to A<1>. These ways of creating buses can be extremely useful when manipulating complex wiring.



Multiple Pins

Pins can be added multiple times using the same principle as for the instances and wires. Brackets '<>', along with appropriate numbering, can be used to define a specific number of pins of the same type.

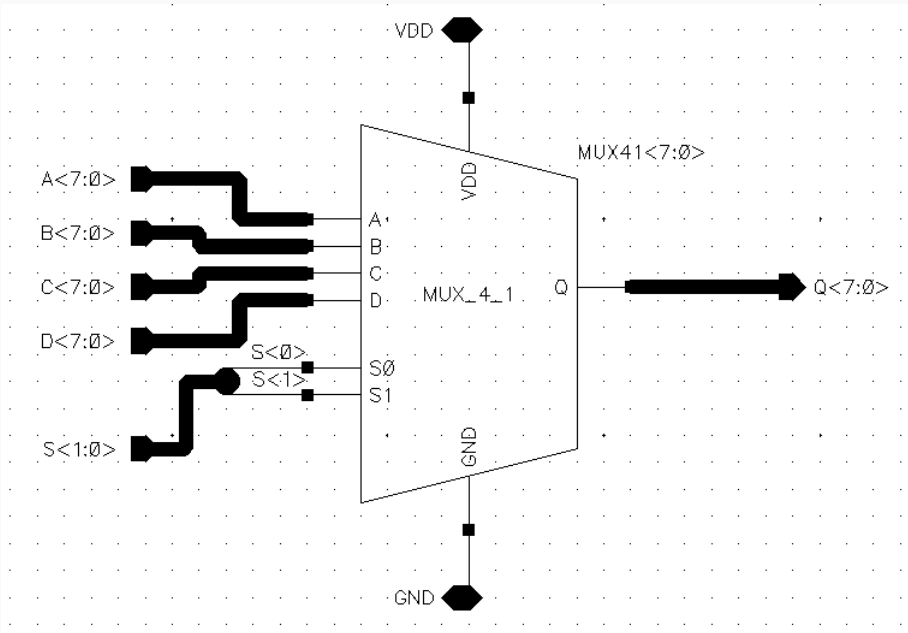
- ✓ Choose ***Create→ Pin*** from the menu *(bindkey:* ***p****)*, and add a pin name with a desired number of pins. Example: **A<7:0>**.
- ✓ Repeat the same procedure for all the required pins in the schematic.



QUESTION : The order in the "<>" is important. What happens if you connect a <7:0> bus to a <0:7> bus ?

4.12. FINAL SCHEMATIC

- ✓ Finally, you should be able to create the final 8-bit 4-to-1 MUX schematic as depicted in the following figure.
- ✓ Create the symbol for your 8-bit MUX.



- ***Use buses when you need to connect several wires together.*** *Refer to the section 4.11.*
- *Note that in* ***Cadence Virtuoso*** *we can use both ascending order e.g.* ***<0:7>*** *and descending order notation* ***<7:0>*** *(the latter is more common for digital circuits)****.*** *The schematic editor recognizes both notations as correct and both can be used. However, you should be careful and stick to one single notation for the full design (be consistent). Also, some other tools may have one of the orders set as default (such as LVS tool in some cases). If so, we also need to be careful about what notation to use.*
- ***Label your wires!*** *The wires that are connected to pins (as in the case of our MUX) will automatically be named according to the pin. Note also that if your label is not placed directly on a wire, you will be prompted to click on a wire to which the label should be attached.*

If you are done before the end of the session, start with LAB02 !