CSE3142 Integrated Circuit Design

CMOS inverter and multiplexer — Analog simulation

Practical Experiment 1 **Duration**: two weeks

1.1 About this tutorial

In this tutorial we will study two Mentor Graphics tools: the **Design Architect** for schematic entry, and **Eldo** for analog circuit simulation.
We will be using the ASIC Design Kit (ADK) which supplements Mentor Graphics tools with libraries required for integrated circuit design. The Mentor Graphics tools will now be invoked with the **adk** prefix and possibly with the **ic** postfix.

In this tutorial we will investigate behaviour of the **CMOS inverter** and the **"weak" multiplexer** which are the basic building blocks of the CMOS **gate logic** and the **switch logic**, respectively.

1.1.1 Related documents

We will be directly or indirectly referring to the following documents:

- *Designing ASICs with the ADK Design Kit and Mentor Graphics Tools* from /sw/mentor/ADK/2.1/doc/pdf/adk.pdf.
- *Design Architect-IC User's Manual* and *Design Architect-IC Reference Manual* from /sw/mentor/IC_Flow/2004.1/shared/pdfdocs/daic_user.pdf, /sw/mentor/IC_Flow/2004.1/shared/pdfdocs/daic_ref.pdf
- *AMPLE User's Manual* and *AMPLE Reference Manuals* from /sw/mentor/IC_Flow/2004.1/shared/pdfdocs/icample_user.pdf, /sw/mentor/IC_Flow/2004.1/shared/pdfdocs/icample_ref.pdf
- *Eldo User's Manual* from /sw/mentor/IC_Flow/2004.1/shared/pdfdocs/eldo_ur.pdf
- *CSE3142 Lecture Notes* from `http://www.csse.monash.edu.au/~app/CSE3142`

We would like also to acknowledge *Using the ASIC Design Kit for Schematic Driven Layout* by David M. Zar from Washington University in St. Louis available at:
`http://ge.ee.wustl.edu/dzar/tutorials/adk_sdl/sdl_toc.html`
Our practicals have been influenced by the material from the above tutorial.

1.2 Initial Setup

Check that the environment variable `$HOME` points to your home directory, then:

- In your `.cshrc` file replace all Mentor Graphics setup by:
  `source $HOME/mgc/mgcrc`

- Create directories `$HOME/mgc` and `$HOME/mgc/ICdes`
- Create a file `$HOME/mgc/mgcrc` with the following contents:

```
setenv ICDES  $HOME/mgc/ICdes
setenv MGC_WD .
```

  You will be using the UNIX environment variable `$ICDES` rather frequently. Another often used environment variable is `$ADK` specified in the above scripts.

  You can examine all the scripts if you are interested in other details of the setup.

- Check that your `.login` file does NOT set `$path` or `$PATH` .
  If it does, move it to the `.cshrc` file.
- Create directory `$ICDES/prac1`
- Open a new terminal window in order to source `.cshrc` and change the directory to `$ICDES/prac1` . This will be your working directory.

1.3 Part1: CMOS Inverter

- From a directory `$ICDES/prac1` invoke `adk_daic &`
- From the pull-down menu select

  MGC > **Location Map** > **Show Location Map**

  and verify that your Location Map is correct and that `$MGC_WD` (working directory) points to your current directory.

- From the pull-down menu select

  MGC > **Location Map** > **Set Working Directory:**

  and verify that `$ICDES/prac1` is really your working directory.

- Open a new schematic sheet by selecting from the **session_palette Schematic** entry.

  In the Open Schematic dialog box specify the new component, namely, **Uinv** where **U** = your initials, e.g., appinv.

  The new schematic sheet Uinv sheet1, will be opened.

- Maximise the window by typing in **max win**. This is one of many ways of invoking AMPLE commands.

Before we start creating a schematic a few comments about Ample scripts.

1.3.1 Ample Scripts

If you look at the shell window from which you have invoked the Design Architect, you can note that every operation in the Design Architect window is equivalent to an AMPLE script command. You should see at least the following commands:

```
 $$open_sheet("$HOME/mgc/ICdes/prac1/Uinv", "schematic",
                        "sheet1", @editable, void, "", @noupdate);
//     max win
  $maximize_window();
```

You can collect commands of interest in a script file, say, **pr1.do**, and then execute the script typing in:
`$dofile("pr1.do")` or `dof pr1.do`
in order to re-do all your work if something goes wrong. You can also execute an **individual command** by pasting it in the little window which appears when you type anything, say, SPACE. Description of the AMPLE commands for a specific tool is also available through the **Help** menu.

Events that occur during a session, including commented messages from the system, are recorded in a **transcript file** as AMPLE functions. You can save all or part of the text in the transcript window to a file by using the pull-down menu
MGC > **Transcript** > **Show Transcript**
and Notepad editing procedures. Because the logical transcript records events as AMPLE functions, it creates a smaller and more easily edited file than a physical transcript does.

1.3.2 Help on Strokes and Function Keys

At this stage it is advisable for you to refresh you skill regarding Strokes and Function keys. Use an appropriate help.

1.3.3 CMOS inverter — schematic entry

Now you will create a schematic of an inverter in one of the following graphical forms presented in Figure 1. The form with a horizontal transistors is the preferred one because its topology more closely resembles the topology of the resulting integrated circuit.
The main steps in creating a schematic are as follows:

- From the **schematic_edit** palette select **Library**.
- From the **ic_ibrary** palette select and place appropriately on the schematic sheet one instance of the followin components:

  pmos, **nmos** (MOS FETransistors), **cap** (capacitor), **GND**, **VDD** (power terminals), **In** and **Out** (I/O ports).
- Type in **view all** in order to see all details.
- To get back to the **schematic_edit** palette, select the **Schematic** entry.

  From the **Edit** section of the **schematic_edit** palette you can execute required Move, Copy, Delete, etc. operations.

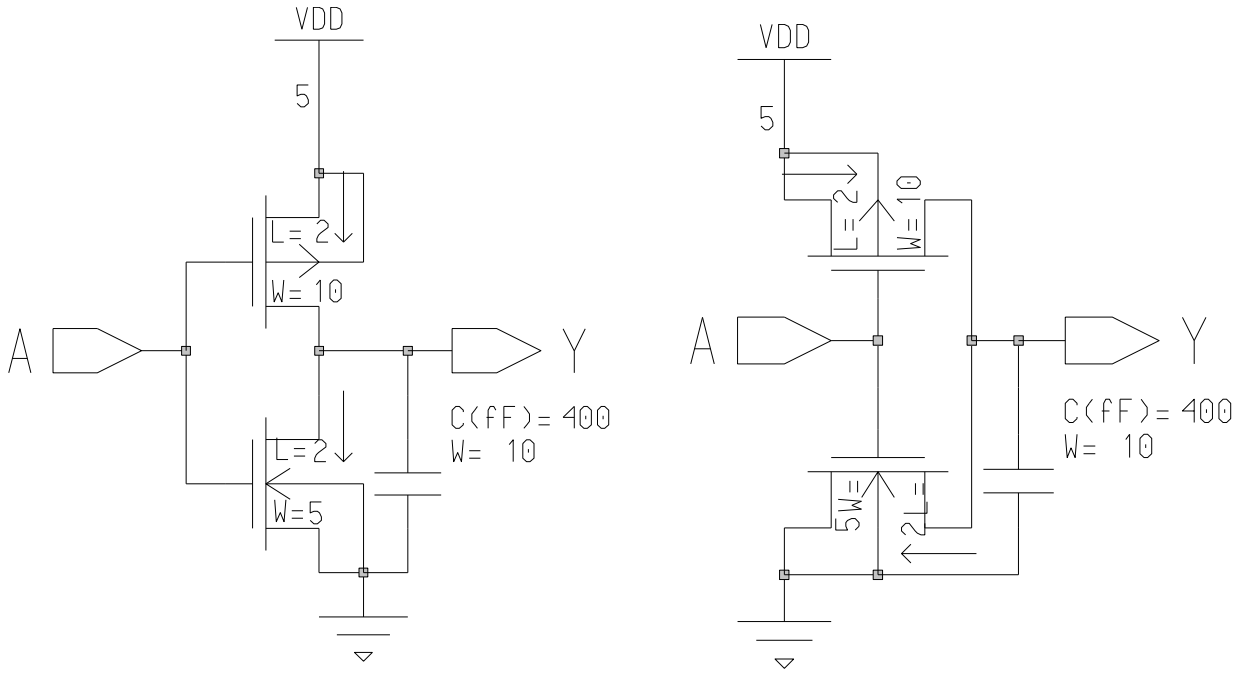


Figure 1: An inverter with a capacitive load in two graphical forms.

- In order to wire up the inverter, select **Wire** and create all connection required to build the inverter.
- Rename the **NET** names on the ports to **A** and **Y** as shown in the schematic. You can do it in a number of ways. On is to click on **Text** to go to the **schematic_text** palette and select Change Value. Select the name and change its value.

  Alternatively, you can use the Shift-F7 key.
- Next, change the value of the **capacitance** to 0.4pF (400fF). You can use previously described methods, or, after selecting the capacitor, invoke the pop-up menu (RHB) from which you select

  Instance > **Properties** > **Modify ...**
- Similarly, change the **width** on the pMOS transistor to 10λ to make it wider. Leave other dimensions unchanged.
- Now you can **Check & Save** the schematic sheet with defaults. If any errors are reported, fix them. Try to understand all warnings.

I used the AMPLE script similar to the following to create the inverter schematic. If you plan to use my script fix the bugs existing in it.

```
   //   dof p1S.do
{ local x, y, y1, y2 ;
  x = 2 ; y = 2 ; y1 = 1 ; y2 = 3.25 ;
 $add_instance("$ADK/lib/sdl/nmos4","",[x,y], @perproperty, \
                          void,["width", "5"],@noflip,-90);
```

```
 $add_instance("$ADK/lib/sdl/pmos4","",[x,y+0.25],   \
             @perproperty,void,["width", "10"],@noflip,90);

 $set_net_width(@p1);
 $add_net([[x,y], [x,y+0.25]]);
 x = x+0.5 ;
 $add_net([[x,y1+0.25], [x,y2-0.25]]);
  x = 1.5 ;
 $add_instance("$ADK/lib/sdl/gnd","",[x,y1],@perproperty,void,[],@noflip,0);
 $add_instance("$ADK/lib/sdl/vdd","",[x,y2],@perproperty,void,[],@noflip,0);
 $add_net([[x,y2], [x,y2-0.25]]);
 $add_net([[x,y2], [x+0.5,y2], [x+0.5,y2-0.25]]);
 $add_net([[x,y1], [x,y1+0.25]]);
 $add_net([[x,y1], [x+0.5,y1], [x+0.5,y1+0.25]]);
  x = 0.75;  y =  2.25 ;
 $add_instance("$MGC_GENLIB/portin","", [x+1,y],@perproperty, \
                                          void,[],@noflip,0);
 $add_net([[x+1,y], [x+1.25,y]]);
 $unselect_all();
 $select_area([[x,y],[x,y]],@comment,void,@frame,@instance,  \
         @net,@pin,@property,@segment,@symbolpin,@text,@vertex);
 $change_text_value("A");
  x = 4 ;
 $add_instance("$MGC_GENLIB/portout","",[x-1,y],  \
                       @perproperty,void,[],@noflip,0);
 $add_net([[x-1,y], [x-1.5,y]]);
 $unselect_all();
 $select_area([[x,y],[x,y]],@comment,void,@frame,@instance,  \
         @net,@pin,@property,@segment,@symbolpin,@text,@vertex);
 $change_text_value("Y");
 x = 2.75; y = 1 ;
 $add_instance("$ADK/lib/sdl/cap","",[x,y],   \
                           @perproperty,void,[],@noflip,0);
 $unselect_all();
 $select_area([[x+1,y+.85],[x+1,y+.85]],@comment,void,@frame,  \
   @instance,@net,@pin,@property,@segment,@symbolpin,@text,@vertex);
 $change_text_value("400");
 $add_net([[x,y], [x-0.75,y]]);
 $add_net([[x,y+1], [x,y+1.25]]);
 $view_all(); $unselect_all();
}
```

- Finally, when the schematic is **checked and saved** you can **make a symbol** for your inverter. From the pull-down menu select

  Miscellaneous > **Generate Symbol**

  menu item to automatically generate a symbol. Do not change any of the options in the dialog box and click on the OK button to generate your symbol.

- Your symbol will be a simple rectangle with two pins. You could modify the body of the

symbol to look like a typical inverter. We will only place the name of the inverter on the box for future identification. To do this, you can use the

Edit > **Add Graphics** > **Text**

pull-down menu, or click on **Text** to go to the **symbol_text** palette and **Add** Comment Text. Use the name **Uinv**. Place this text in the symbol. Modify the size and font if needed.

- Now you can **Check & Save** the symbol. You can ignore the warnings about the properties not being on the interface. If you have any other errors, however, you need to fix them before moving on.
- Close the symbol window. You can use famous Mentor Graphics strokes!

1.3.4 Printing a schematic

Finally you might like to print the schematic for your report. To produce a post script file suitable for encapsulation in your report, use the pull-down menu :

File > **Print ...**

Select the **Export to File** button and set up the path name for your schematic. I suggest to change the path name to: `$ICDES/prac1/UinvSch.eps`.
Untick the **Color Output** box to avoid a yellow and light blue colour scheme.

Fixing a postscript file

Usually a postscript file generated by Mentor Graphics tools is too big and badly placed with respect to its bounding box, hence, difficult to encapsulate in the text. In order to fix it edit a '*.eps' file with your favorite text editor.

- Search for 'scale' so you can find the following sequence of postscript statements:

```
0.240000 0.240000 scale
sm 1 slw
0.000000 300.000000 translate
```

- Modify them to read, say:

```
10 -90 translate
0.16 0.16 scale
sm 1 slw
```

- In addition fix the bounding box accordingly to read, say:

```
%%BoundingBox: 0 0 360 430
```

At this stage you can safely quit and re-enter **adk_daic** (from the directory `$ICDES/prac1`) if required.

1.4 Simulation – Selecting an IC technology

Up to now we have not specified any technology in which our integrated circuit will be manufactured. Simulation results strongly depend on the selected technology.

From the **schematic_edit** palette select
Simulation > **TSMC0.35**
This will specify the TSMC0.35 technology and create the following **viewpoints** which will be used in the analog and digital simulation and in the circuit layout creation and verification.

tsmc035	for use with the digital QuickSim II Simulator
tsmc035a	for use with the analog simulators
layout	for use with IC Station when doing standard cell place and route
lvs	for use with IC Station when performing LVS of your design
sdl	for use with IC Station when doing SDL place and route

Examine the structure of the **Uinv** directory to see that all the above viewpoints have been created. Do not mess up in the **Uinv** directory. Go back one level up to `$ICDES/prac1`.

1.5 Analog Simulation with Eldo

We are ready to test our schematic. To do this we will perform analog simulation using Eldo.

Eldo is a SPICE-based analog simulator that can be invoked from the *Design Architect – IC*. After the *Eldo User's Manual* p.1-3 we reproduce in Figure 2 the following flowchart showing the basic

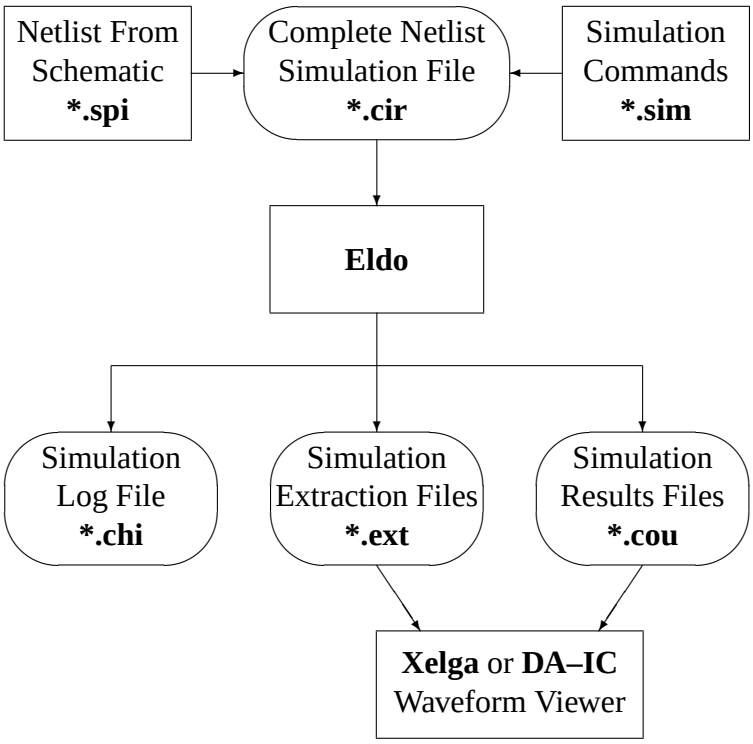


Figure 2: Eldo simulation flowchart

input and output files. The main files are:

***.cir** The main Eldo control file, containing circuit netlist, stimulus and simulation control commands. This file is SPICE compatible, the Eldo control language being a superset of the Berkeley SPICE syntax. This file includes:

***.spi** The circuit netlist SPICE file generated by the Eldo netlister,

***.sim** The simulation commands and parameters prepared by the user.

***.chi** The SPICE compatible output log file containing ASCII data, including results and error messages.

***.cou** A binary file containing Eldo analog simulation results data. A special interface is provided giving you access to this data from your own post-processor software if required. ANACAD post-processors also read and write to this file.

- **Creating the netlist file**.

  From the **schematic_sim_palette** invoke Netlist > Write.

  This will create a number of files with the circuit description in the directory `.../Uinv/tsmc035a`. You can inspect the netlist file, `Uinv_tsmc035a.spi` using the Netlist > Edit command.

- **Setting up the default viewer.**

  From the **schematic_sim_palette** select:

  Setup > Session > Setup Simulator/Viewer...

  and in the dialog window select **EZwave** as the viewer.

- **Preparing the simulation**

  Click on Analyses... from the **schematic_sim_palette** , this brings up a dialog window. Tick **Transient**, then click the Setup... button. In the dialog window tick the **Initial Conds** box to make Eldo calculate initial conditions before simulation. Ok both dialogs to continue.

- **Selecting nets to view**

  Select both **A** and **Y** nets and invoke Wave Outputs > Save Selected... and **OK** the dialog boxes.

- **Forcing input and power**

  Now we apply stimulus to the input net and provide power. Select **A** and click on Forces/ICs > Add Forces... from the palette. Force **A** to be a pulse from zero to five volts, with a duty cycle of about half, and a period of 200ns.

  Using the same buttons from the palette force **Vdd** to +5V DC, using the DC tab from the **Add Force** dialog box.

  The two forces should show up in red near the appropriate nets.

- **Including library file**.

  Up until now we have placed components and wired them together with nets. For simulation Eldo requires models of the pmos4 and nmos4 transistor.

  For this we include the tsmc035 spice library by invoking in the **schematic_sim_palette** Lib/Temp/Inc > Libraries... command.

  Enter **$ADK/technology/accusim/tsmc035.mod** as the library path (The navigator can be used to simplified the task) and **NOM** as the name of the library variant.

- **Invoking Simulation**.

  From the **schematic_sim_palette** invoke Run ELDO.

  This will pop-up two ASCII windows, the first is the results of running the netlister EldoNet, and the second is the informative output of Eldo itself as it runs the simulation. On completion press enter in each window to close them.

  You can examine the resulting log file using the ASCII Files > View Log command.

- **Invoking a viewer**

  From the **schematic_sim_palette** invoke View Waves > New Window. By default EZwave plots **A** and **Y** on top of each other, you can grab the text "V(Y)" from the **Transient Restults** window and place it below the plot of **A**. This should produce the plots as in Figure 3

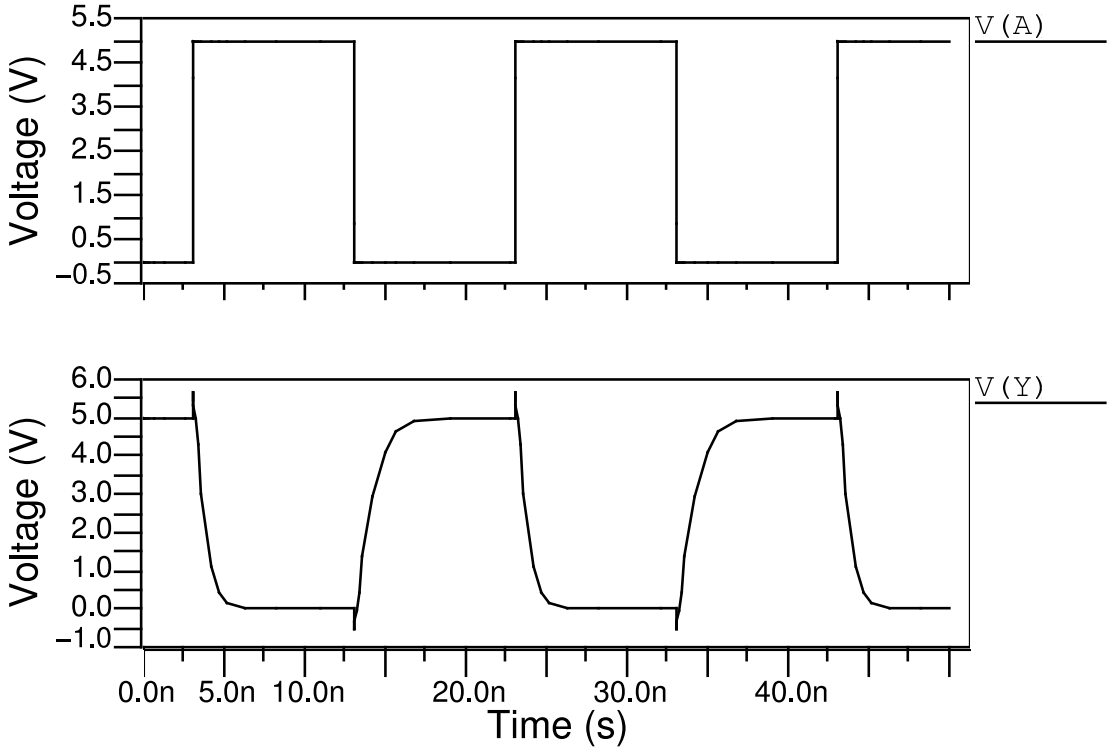


Figure 3: Analog simulation waveforms for an inverter with the capacitance load.

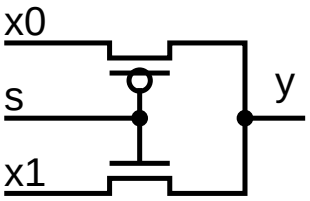
- You can now exit the waveform viewer, end the simulation and quit the Design Architect.

1.6 Part 2: XOR circuit based on a "weak" multiplexer

In Part 2 you will repeat steps from Part 1 for an XOR circuit based on a "weak" multiplexer. We will study essential DC and transient properties of such a circuit.

1.6.1 A "weak" multiplexer

A "weak" multiplexer is a basic building block of the "**switch logic**". The concept of the switch logic is that logic circuits are implemented as combination of switches, rather than logic gates.
A 2-to-1 "weak" multiplexer has the following structure:



$$y = \begin{cases} x_0 & \text{if} \quad s = 0 \\ x_1 & \text{if} \quad s = 1 \end{cases}$$

When the signal s is **low**, only the **pMOS** pass transistor is "on" and $y = x_0$, and, conversely, when the signal s is **high**, only the **nMOS** pass transistor is "on" and $y = x_1$.
Note that the "weak" multiplexer is similar to an inverter in that that it consists of an nMOS–pMOS pair, with gates driven by the same signals. However the inverter's VDD and GND connections have been replaced with inputs x_0, x_1, which can be driven by logic signals.
The problem with the above circuit, which explains its name, is that when an input signal, x_0 or x_1, is being passed to the output, the circuit **degrades**:

- the **low level** of the x_0 signal passing through the **pMOS** transistor,
- the **high level** of the x_1 signal passing through the **nMOS** transistor,

You will study this effect in your simulation.

1.6.2 A "weak" XOR circuit

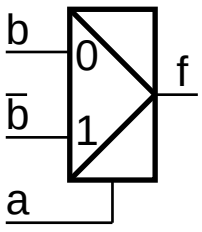
The Exclusive-OR (XOR) function:

$$f = a \oplus b$$

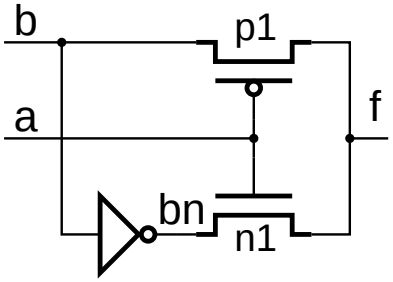
can be implemented using a **2-to-1 multiplexer** in the following way:

a	b	f
0	0	0
0	1	1
1	0	1
1	1	0

a	f
0	b
1	$\bar{b}$



A **"weak" XOR** can be built using a multiplexing pair of complementary MOS transistors and an inverter ($bn = \bar{b}$) as in Figure 4. In the table we indicate when a specific transistor is "on" and what is the level of the output signal, f.



a	b	"on"	f	level
0	0	p1	b	weak zero
0	1	p1	b	strong one
1	0	n1	bn	weak one
1	1	n1	bn	strong zero

Figure 4: A "weak" XOR circuit

1.6.3 Schematic entry

Use the Design Architect to create a transistor-level schematic of the "weak" XOR presented in Figure 4. Name the component **Uxor** . Arrange transistors horizontally, two pMOS transistors in one line and two nMOS transistors below so that relevant gates face each other.
All transistors should have size of: $L = 2\lambda, W = 5\lambda$.

In addition create a schematic of a component **UxorC** which has a capacitive load added to the output f. A capacitor can be built using a gate-to-substrate capacitance as in Figure 5. Use an nMOS

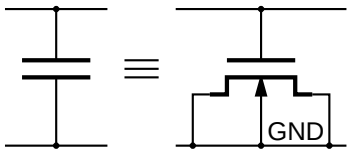


Figure 5: A gate-to-substrate capacitance

with dimensions: $L = 2\lambda, W = 20\lambda$.

1.6.4 Analog simulation

Simulate the "weak" XOR using the Eldo simulator for two cases **without**, and **with the capacitive load**. Prepare suitable forces and arrange all signals to obtain waveforms similar to those in Figure 6. From your simulation waveforms estimate:

- The average propagation delay[1], rise and fall times (10% to 90%) of the signals bn and f (zoom in around transitions).
- The voltage level of the **weak zero** and **weak one**.

[1] Chapter 3 lecture notes

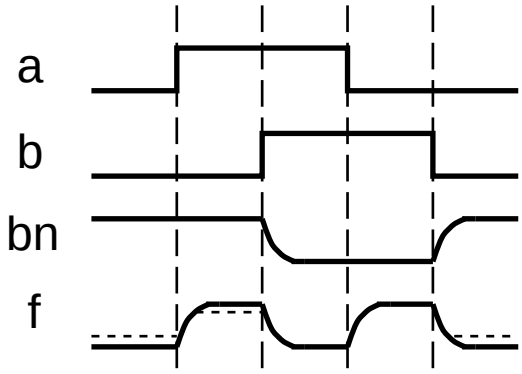


Figure 6: Expected simulation waveform for the “weak” XOR with a capacitive load.

1.7 Demonstration and Report

- Demonstrate your working circuits to your tutor.
- Write a brief report in which you include a short description of your activities and the results obtained. Include schematics and simulation results.

Independent Voltage Source

Independent Source Element

```
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TIME_DEPENDENT_FUNCTION[1]] [RPORT=val [NONOISE]]
+ [TC1=val] [TC2=val] [IPORT=val] [CPORT=val] [LPORT=val]
+ [NOISETEMP=val]
```

Noise Source

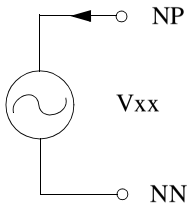
```
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [RPORT=val [NONOISE]] [TC1=val] [TC2=val] [IPORT=val]
+ [CPORT=val] [LPORT=val] [NOISETEMP=val]
+ NOISE [THN=VAL] [FLN=VAL] [ALPHA=VAL] [FC=VAL] [N=VAL]
+ [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
```

Tabular Noise Source

```
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [RPORT=val [NONOISE]] [TC1=val] [TC2=val] [IPORT=val]
+ [CPORT=val] [LPORT=val] [NOISETEMP=val] NOISE TABLE
+ [[INTERP=val] [DEC|OCT|LIN|LOG]] (f1 val1) (f2 val2) ...
```

Multi-Tone Source

```
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [RPORT=val [NONOISE]] [TC1=val] [TC2=val] [IPORT=val]
+ [CPORT=val] [LPORT=val] [NOISETEMP=val]
+ FOUR fund1 [fund2 [fund3]] MA|RI|DB|PMA|PDB|PDBM
+ (int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2
+ {(int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2}
```



1. Refer to the **EXP**, **PATTERN**, **PULSE**, **PWL**, **SFFM** and **SIN** source functions.

Pulse Function

PULSE (V0 V1 [TD [TR [TF [PW [PER]]]]])

Generates a periodic pulse as described below. To be used in combination with independent voltage (**V**xx) or current (**I**xx) sources.

Parameters

V0	Initial value of DC voltage or current.
V1	Pulse magnitude in volts or amperes.

Optional Parameters

TD	Delay time in seconds. Default value is zero.
TR	Rise time in seconds. Default value is TPRINT.
TF	Fall time in seconds. Default value is TPRINT.
PW	Pulse width in seconds. Default value is TSTOP.
PER	Pulse period in seconds. Default value is TSTOP.

Note

If the parameters TR and TF are both set to 0, they are assigned a value of **TPRINT**, the minimum internal time step of the simulator, defined in the **.TRAN** command.

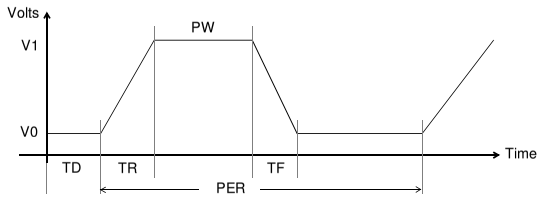


Figure 5-5. Pulse Function