

ECE 514  
Introduction to VLSI Systems Laboratory  
**Project Handbook**  
Electrical and Computer Engineering Department  
University of Louisville

© 2006  
Alexander Broadhead, MSEE' 93  
Piotr A. Habas  
Mehmet K. Muezzinoglu  
Grzegorz M. Boratyn  
Prof. Jacek M. Zurada

Addendum © 1997  
Jason B. Cole

## INTRODUCTION

This manual presents mandatory projects in ECE 514, Introduction to VLSI Systems Laboratory. It is not intended as a stand alone tutorial on VLSI.

The principal tool for modern VLSI design is a computer. Design, layout, simulation at all levels, and to a large degree commercial fabrication process are all computer-based. Projects described in this manual are intended to be performed on the terminals in the S.T. Fife VLSI Laboratory, located in room 219 of W.S. Speed Hall. The design and simulation tools are computer applications: MAGIC, SPICE, and IRSIM all running under UNIX system.

Students that are not familiar with the UNIX operating system are required to familiarize themselves with it during first two weeks of classes before the projects are started. Manuals and tutorials for the lab design and simulation tools as well as for using the UNIX system can be found in the VLSI Laboratory and on the laboratory web site at <http://ci.uofl.edu/zurada/courses/ece515>. During one of the first lectures the students will be given accounts on the VLSI server. If you do not have the account yet, contact the TA, otherwise you will not be able to fulfill requirements of the projects.

All projects are to done in groups (generally of two). All files created in each project have to be located in directory named *project\_X*, where X is the project number. The TA will cross-check those files in case of undesirable duplications or plagiarisms in any of the present or past reports. Points will be taken if the TA will have trouble locating any of the required files. Remember about the appropriate file extensions.

## PROJECT 1: LAYOUT DESIGN, PARAMETER EXTRACTION AND SPICE SIMULATION

### Description:

In this project you will design a two-input CMOS gate using MAGIC, extract the design into SPICE and simulate it with SPICE. All files created in this project should be save in folder *project\_1*.

### Procedures:

1. Design a two-input CMOS (A) NAND or (B) NOR gate using MAGIC. Use identically sized ( $L = W$ ) n- and p-type transistors. All voltage supply, ground, inputs, and outputs must be labeled. Use special labels for voltage supply ("*Vdd!*") and ground ("*Gnd!*"). Circuit such as gates are usually elementary building blocks of bigger designs. Therefore, your design should be as compact as possible. Painted layers should not be too thick and there should not be too much of empty space between parts of the design. It is a good idea to use grid (*:grid* command in MAGIC) as a reference for size of a circuit. The gate will receive inputs from and provide outputs for other parts of the circuit, thus inputs, outputs, voltage and ground lines of your circuit should be prepared for easy connections. Save your design as (A) *nand.mag* or (B) *nor.mag*.
2. Extract circuit information from MAGIC using the *:ext* command. This will create a new file (named (A) *nand.ext* or (B) *nor.ext*) containing a description of your design in the ext format.
3. Convert the circuit description from the ext into SPICE format. Run the *ext2spice3* script with your extracted file as one of the arguments. The syntax of *ext2spice3* is as follows:

*ext2spice3 [-R][-C] file.ext*

where *-R* and *-C* options make the script ignore all resistances and capacities, respectively. Ignore only resistances. A new file (named (A) *nand.spice* or (B) *nor.spice*) will be created. It is not a complete input file for SPICE, yet. Additional circuit parameters and the desired type of analysis need to be defined. *Note:* In some cases the extracted circuit will have interchanged drains and sources of few transistors. You can disregard this change as it will not affect further analysis.

4. Edit the file received in the previous step to create a full input file for SPICE. For transistors, use threshold values  $V_{tn} = -V_{tp} = 1V$ . All other parameter values should be default.

Obtain:

- a) the Voltage Transfer Characteristic (VTC) of the gate with both inputs shorted together (save as *vtc.ps*),
  - b) the response to the transient signals shown in Fig. 1 (save as *trans.ps*). Use  $C_L = 0.2pF$  and set all rise and fall times to 1ns.
5. Run SPICE (*spice3 file.spice*) with your edited spice file and obtain the necessary plots.

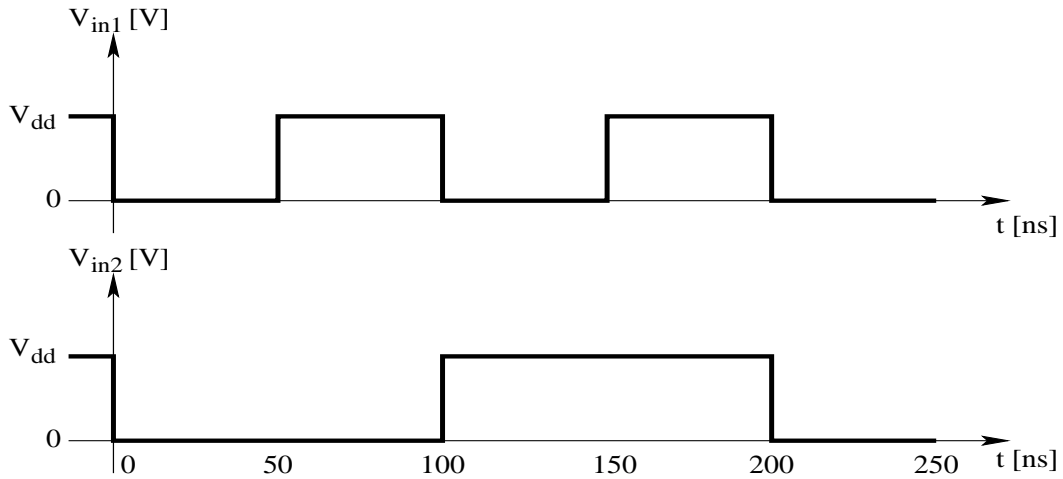


Figure 1: Input pulses for simulation of the project 1 design

**Report Requirements:**

- A graphic representation of your layout ((A) *nand.ps* or (B) *nor.ps*). To print a design, create a postscript file with the *:plot* command:

*:plot postscript file.ps*

and print the postscript file. Unfortunately this command does not always work perfectly. If so, use the *xv* application.

- The SPICE input file created in step 4 ((A) *nand.spice* or (B) *nor.spice*).
- The plots created in SPICE requested in step 5 (*vtc.ps* and *trans.ps*).
- All files created in this project should be stored in the *project\_1* directory.

## PROJECT 2: CMOS FULL ADDER DESIGN AND SIMULATION

### Description:

In this project you will design a layout for a CMOS static full adder with the MAGIC editor and simulate the designed circuit with SPICE.

### Procedures:

1. Design a static Full Adder (FA) in CMOS technology using  
(A) two-input NOR gates and the design presented in Fig. 2

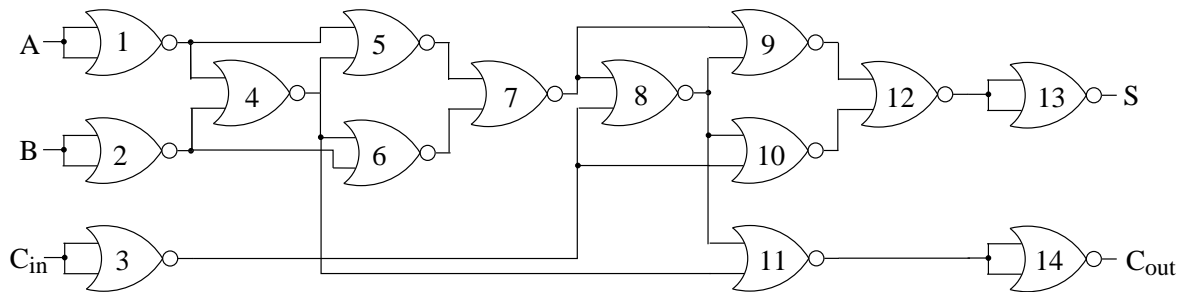


Figure 2: Static full adder using two-input NOR gates

- (B) the design presented in Fig. 3

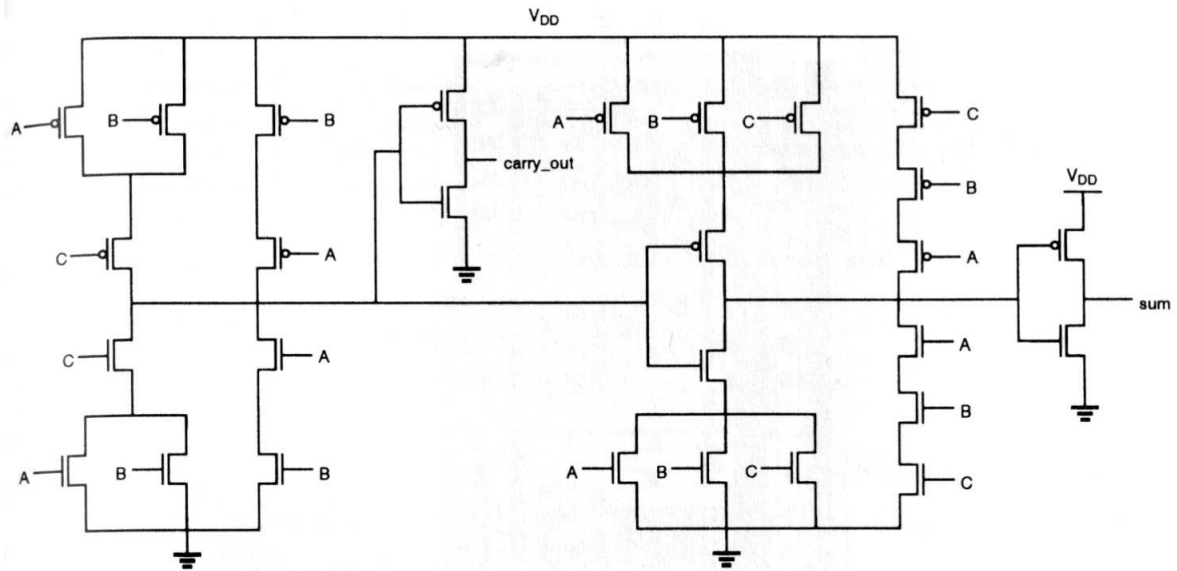


Figure 3: Static full adder

2. In MAGIC, use sub-cells to avoid repetitive work and reduce the design time (see MAGIC manual and help for the `:getcell` command). Label inputs, outputs, voltage source and ground. Use special labels for voltage source ("`Vdd!`") and ground ("`Gnd!`").

3. Extract the circuit design as in Project 1.
4. Convert the extracted circuit information into SPICE format (as in Project 1).
5. Add all necessary lines to the SPICE input file. Use model descriptions available at [http://ci.louisville.edu/zurada/ece515/spice\\_models.spice](http://ci.louisville.edu/zurada/ece515/spice_models.spice)

```
.model nfet NMOS LEVEL=2.0 LD=0.28U TOX=500.0E-10 +NSUB=1.0E16 VTO=0.827125
KP=3.286649E-5 GAMMA=1.3596 +PHI=0.6 U0=200.0 UEXP=1.001E-3 UCRIT=999000.0
+DELTA=1.2405 VMAX=100000.0 XJ=0.4U LAMBDA=1.604983E-2 +NFS=1.234795E12
NEFF=1.001E-2 NSS=0.0 TPG=1.0 +RSH=25.0 CGSO=5.2E-10 CGDO=5.2E-10 CJ=3.2E-4
MJ=0.5 CJSW=9.0E-10 MJSW=0.33
```

```
.model pfet PMOS LEVEL=2.0 LD=0.28U TOX=500.0E-10 +NSUB=1.121088E14
VTO=-0.894654 KP=1.526452E-5 GAMMA=0.879003 +PHI=0.6 U0=100.0 UEXP=0.153441
UCRIT=16376.5 +DELTA=1.93831 VMAX=100000.0 XJ=0.4U LAMBDA=4.708659E-2
+NFS=8.788617E11 NEFF=1.001E-2 NSS=0.0 TPG=-1.0 +RSH=95.0 CGSO=4.0E-10
CGDO=4.0E-10 CJ=2.0E-4 MJ=0.5 CJSW=4.5E-10 MJSW=0.33
```

6. Obtain the response of the circuit ("sum" and "carry out") to the transient signals shown in Fig. 4, set all rise and fall times to 1ns.

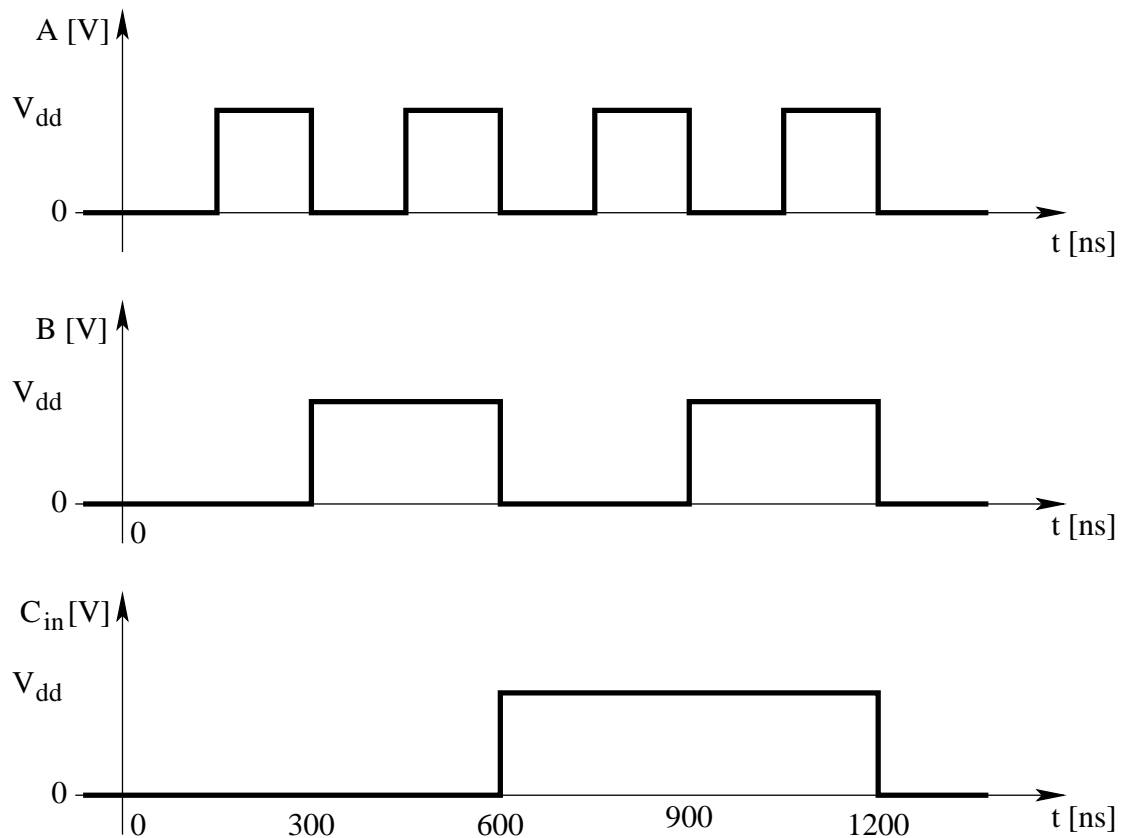


Figure 4: Input signals for the Full Adder (FA) transient simulation

**Report Requirements:**

- The MAGIC graphic representation of your layout (*adder.mag* captured and saved as *adder.ps*).
- The complete SPICE input file (*adder.spice*).
- The plots created in SPICE requested in step 6.
- All files created in this project should be stored in the directory named *project\_2*.

## PROJECT 3: A 4-BIT CMOS PARALLEL ADDER DESIGN AND SIMULATION

**Description:**

In this project you will design, simulate and test a 4-bit CMOS parallel adder. IRSIM package will be used for simulation, because SPICE cannot efficiently handle more complicated logic circuits.

**Procedures:**

1. Design a 4-bit CMOS Parallel Adder circuit as it is shown in Fig. 5.

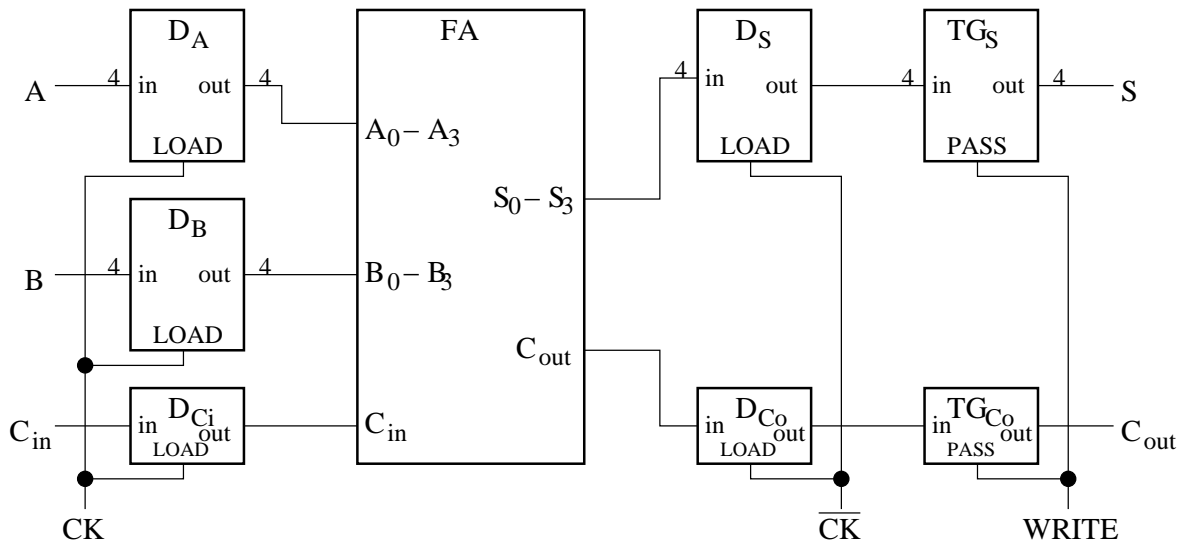


Figure 5: A 4-bit Parallel Adder block diagram

The design consists of three 4-bit registers ( $D_A$ ,  $D_B$ ,  $D_S$ ), two 1-bit registers ( $D_{C_i}$ ,  $D_{C_o}$ ), and four cascaded Full Adders (FA). The Full Adder design may be taken from Project 2. The registers are D-latches (Fig. 6) used to store input and output information.

All nine input bits should be read when the  $CK$  control signal goes high. Then data is processed asynchronously in the Full Adders and stored in the output registers when the  $\overline{CK}$  control signal goes high. The output is buffered with the array of transmission gates ( $TG_S$ ,  $TG_{C_o}$ ) so that the output is floating unless the  $WRITE$  signal is high.

2. Extract the circuit design (*adder.mag*) as (*adder.ext*).
3. Convert the extracted design into the IRSIM input file format using the *ext2sim* script. The syntax of *ext2sim* is as:

$$ext2sim [-R][ -C] adder.ext$$

where  $-R$  and  $-C$  options have the same meaning as for *ext2spice3*. Three files of the same name as the extracted file with extensions *.al*, *.nodes*, and *.sim* will be created.

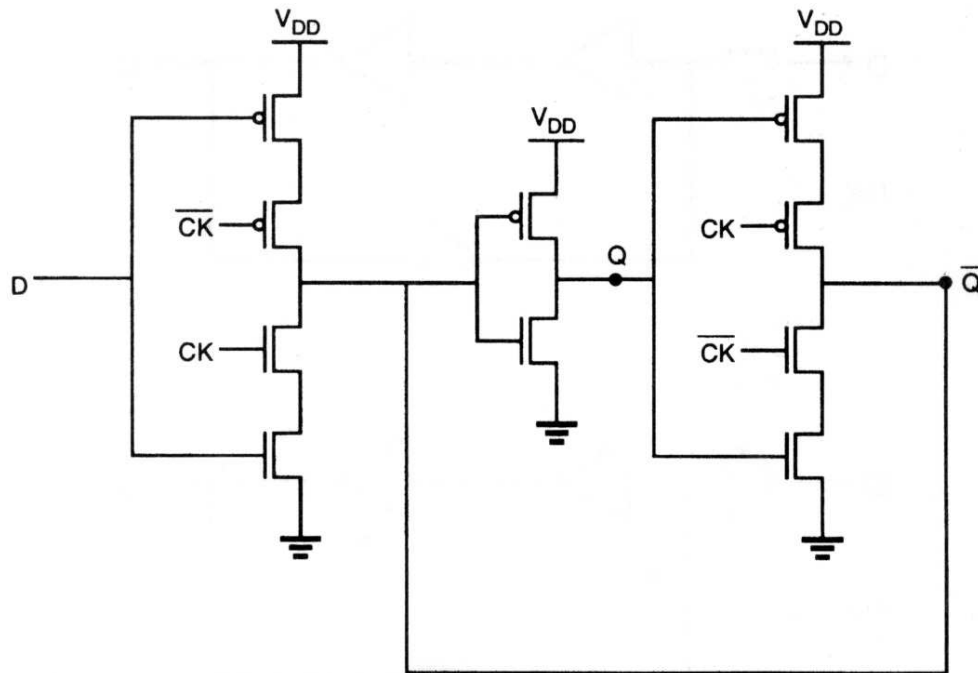


Figure 6: CMOS implementation of the D-latch

4. Evaluate your circuit performance with the IRSIM package.

*irsim -s scmos100.prm adder.sim*

It is not necessary to test the circuit exhaustively, but at least 5 different combinations of inputs should be tried. The testing combinations should be chosen with care and show the functionality of the circuit. You will be also graded on this basis.

An IRSIM manual can be either found in the lab or downloaded from [http://bear.ces.cwru.edu/eecs\\_cad/tut\\_irsim\\_gate.html](http://bear.ces.cwru.edu/eecs_cad/tut_irsim_gate.html).

#### Report Requirements:

- A graphic representation of your layout (*adder.ps*). See Project 1 report requirements for printing instructions.
- Printouts of the IRSIM analyzer window, showing the tests of the circuit. See IRSIM manual for printing instructions.
- All files created in this project should be stored in the directory named *project\_3*.

## PROJECT 4: THE STANDARD FRAME AND I/O PADS

### Description:

The purpose of this project is to acquaint yourself with a standard frame for MOSIS project submissions and with the steps necessary to acquire the frame and properly connect a sub-circuit within it to receive input and send output.

### Procedures:

1. Download the frame and its elements from:

*ftp://ftp.isi.edu/pub/sondeen/magic/SCNA.80.PADS.2000/mag*

You need all the *mag* files and the file: *SCNA.80.tech27*, which is a magic technology file that stores information about layers and design rules. White dots in your design and errors given by the *:drc* command are violations of the design rules specified in the technology file. You can see MOSIS SCMOS Design Rules at:

*http://www.mosis.org/Technical/Designrules/scmos/scmos-main.html*

2. Open file *40p2200.mag* in Magic with the downloaded technology file, type:

```
magic -T SCNA.80.tech27 40p2200
```

3. Use the *:getcell* command to load your 4-bit parallel adder layout design from Project 3.
4. Connect the Parallel Adder to the frame (inputs, outputs, Vdd, and ground). Inputs and outputs should be connected to the I/O pads. The I/O pads can be set to work as input or output pads. Connecting the enable line (labeled *EN*) to Vdd causes the pad to work as output, connecting it to Gnd creates an input pad. Use the lines labeled *IN* and *OUT* for inputs and outputs, respectively. **Note:** There are Vdd and Gnd rails in each pad. Those are supplied from the corner pads. Your Vdd and Gnd should be connected there. Multiple connections to Vdd and Gnd are often a good idea. The Vdd and Gnd pads in the middle of top and bottom row of the frame are floating. They can be used if a circuit uses independent Vdd or ground. You do not need to use them.
5. Simulate the layout with the IRSIM package:
  - Label input and output pads of the frame.
  - Extract the layout design as in Project 1.
  - Use *ext2sim* to convert the extracted layout design into the IRSIM format.
  - Simulate the layout with IRSIM.
6. Set the *lambda* value and extract the layout design into the *cif* format (accepted by MOSIS), type:

```
:cif ostyle lambda=0.80()  
:cif
```

This will create a file with *.cif* extension that can be submitted for MOSIS fabrication.

### Report Requirements:

- A graphic representation of your layout. See Project 1 report requirements for printing instructions.
- Two graphic representations of part of your layout, one containing correctly connected input pad, the other - correctly connected output pad. **Note:** This can be done with the `:plot` command, with only part of the layout selected. Add any missing labels by hand.
- A graphic representation of your layout obtained from the `cif` file:
  - Convert the `cif` file into postscript format with the `cif2ps` command, type in UNIX shell window:

```
cif2ps file.cif > file.ps
```

A postscript file (with `ps` extension) will be created.
  - Print out the obtained postscript file with the `lp` command.  
**Note:** This will be a gray level image, not color.
- Printout of IRSIM analyzer window showing one test simulation of the circuit.
- Answer the following questions:
  1. Why are multiple connections to Vdd and ground a good idea?
  2. How can the unused i/o pads be utilized? Where would you connect them and for what reason?
- All files created in this project have to be stored in the directory named `project.4`.

## PROJECT 5: THE FINAL PROJECT

The final project is an independent design, simulation and testing experience. Each group will submit a design proposal outlining the circuit they are attempting to design. The TA will review the proposals and (possibly) suggest modifications. The groups then design the circuit, test it using SPICE and IRSIM, and prepare it for submission to MOSIS. The TA will be submitting all the projects, after all of them are handed in. MOSIS tests the design and has to approve it for fabrication. In case of MOSIS disapproval, TA will return the design to the group with modification suggestions. **Important:** The final project is not finished until it is approved by MOSIS.

The 40p2200 standard frame (the same one that was used in Project 4) has 34 I/O pads (which can be used for both, input and output, with proper multiplexing), and 3 power and ground pads (one of each is separate floating pad, so that multiple power levels can be utilized). You need to limit your design to the number of inputs and outputs and to the size of the standard frame.

Past projects were typically of the following types:

- **ALU:** An Arithmetic Logic Unit.
- **Memory:** Not acceptable project any more, as the design is usually very simple.
- **Co-Processing:** Any of several mathematical functions not standardly implemented in an ALU could be selected. Multiplexers, special purpose multipliers (complement of 2, pipelined, cascaded, floating point, etc.), specialized adders (floating point, look-ahead, etc.), divider have all been done. Other specialized functions (powers, trigonometric, exponential, etc.) are also possible.
- **Signal Processing:** One group has designed a 4x4x4 (4-bit by 4-bit by 4 point) DFT chip. Other functions (matrix algebra, other transforms, filters, etc.) are also possible.
- **Miscellany:** All others. If you have seen something done digitally (timers, counters, other measurement devices, controllers, etc.), it is likely that you can at least approximate it here. Past examples include an odometer for mounting on bicycle and a change counter for a vending machine. These projects require careful thought and fairly large amounts of preliminary design for the initial project proposal. You must convince the TA that your project is within reason.

This is by no means a comprehensive listing. Groups are welcome to submit any idea as a possible project. **Important:** Bear in mind that the burden of selling your project proposal as worthwhile and realizable falls with you. The TA cannot take responsibility for becoming familiar with the intimate detail of every project, nor can he bear responsibility for miscommunication of objectives. **You must make sure that everyone concerned is aware of what you are trying to do and that it is acceptable.**

### Procedures:

1. Meet with the TA to present your group's preliminary proposal for the Final Project. The TA will review the proposal and either accept it or make suggestions for a more acceptable project. Preliminary scale decisions should be made.
2. Submit a brief description of the project. The following parts should be included:
  - **Objective:** What will the circuit do when completed? What is its function?
  - **Design Approach:** List the main building blocks and describe how they will cooperate. If you are going to use some special purpose circuitry (for example to optimize space or time) describe its main idea.

- Scale: Number of inputs and outputs.
  - Testing: How do you plan to verify the circuit functionality?
3. Submit a block diagram of your circuit. Show main building blocks of the circuit, indicate all input, output, and control signals.
  4. Read the *Project Handbook Addendum*.
  5. Design and test your circuit. **Note:** Do not forget to properly stitch your circuit to avoid latch-up. Any modifications of the original design objectives should be discussed and accepted by the TA.
  6. Place the circuit in the frame, connect it as described in Project 4, and test it. Simply attach tabs to the pads of the frame and label the tabs for Irsim. Remember to remove the tabs when finished.
  7. Extract the *cif* file of your Final Project layout design (including the frame) as described in Project 4. Note carefully errors and warnings generated. Errors must be corrected. Warnings may be unavoidable.

#### Report Requirements:

- A title page stating names of group members, name of the project, and date of submission.
- Description of the project, like in point 2 of the Procedures.
- Block diagram of the circuit, like in point 3 of the Procedures.
- File description containing username of the system account, path to the directory in which the project files are stored, and a short description of each magic file used in this project (example: nor.mag - two input nor gate).
- A graphic representation of your layout. See the report requirements of Project 1 for printing instructions.
- Results of simulations. This should include enough simulation results to convince the reader that the project can reasonably be expected to perform as designed. Each simulation result should be presented separately, preceded by a statement of what is being tested, what is the meaning of the inputs, and what outputs are expected, concluded by a short comment about whether the test succeeded or failed. This section should be concluded with a summery stating which circuit functionalities were tested, which succeeded and which failed, and a comment whether some functionalities should be excluded because of failed tests. **Important:** Testing failures will not affect your grade if this part is done right (unless, of course, your circuit has 0% succesful tests).
- All files used in this project should be stored in directory named finalproject. **Important:** It is very likely that the TA will have to look at your layout design after receiving some errors from MOSIS. Thus, if you played with the file access control, the whole content of the finalproject directory must be readable for everyone after your project submission (do not forget about allowing others execute your finalproject directory). Many things can be fixed quickly by the TA (if he has access) so this is also in your interest.

The finished 40-pin chip packages should be available approximately eight weeks after submission. you should keep in contact with the TA or Prof. Zurada so that any problems discovered in the MOSIS design rule checking process can be corrected, and so that you can be contacted when the chips arrive. Each group is expected to perform logic (and speed, if possible) testing on their circuit and to fill out the MOSIS Project Report form.

## PROJECT HANDBOOK ADDENDUM

**The Substrate Connection**

A substrate connection (or “downbond”) will be required in your design and requested of MOSIS when your final project is submitted for fabrication. This connection is simply a wire between bonding finger (in the package cavity) to the bottom of the package cavity itself, which is gold-plated. Hence, this connects the package cavity to the chip’s substrate, both physically and electrically (since the substrate is the bottom of the chip and is attached to the package cavity via conductive glue). This substrate connection also connects the bonding finger to Pad #1 in the TinyChip Standard Frame. *Therefore, leave the digital I/O pad corresponding to external package Pin #1 unconnected in your final design.* Refer to the standard frame bonding diagram, one of which is included in the appendix, to see which pad corresponds to the external package pin #1. It is typically the fifth pad down from the top right of the 40pc22x22 (unrotated) TinyChip Standard Frame. For N-well CMOS designs, you will connect the external package Pin #1 (the substrate connection) to GND when you are testing your fabricated device. For P-well CMOS designs, you would connect the substrate to VDD instead.

**Adding Stitches to Your Final Design**

There are two types of stitches that should be added (to avoid latch-up) to the lowest hierarchical subcells in your final project design before it is submitted for MOSIS fabrication. While stitches can be placed anywhere in your final design layout, you are encouraged to incorporate them into your design lowest hierarchical subcells. Note that the only metalization layer included in each stitch is Metal 1. Thus to connect a stitch to Metal 2 VDD or GND rail, first connect the stitch to a Metal 1 - Metal 2 VIA, then connect to the appropriate rail. *Also, note that transmission gate transistors do not need to be stitched.*

**N-WELL stitches for PMOS transistors:**

- Place one stitch per N-WELL, or one stitch per two or three PMOS transistors.
- Stitches should be placed as close as possible to the PMOS transistor(s) you are trying to stitch.
- Connect the stitch to a nearby VDD rail.
- The stitch layer in Magic is “nnc” (“nsubstratencontact”, “nncontact”, “nsc”, “nwc”, or “nwcontact”).
- Each “nnc” stitch is actually a *contact* between Metal 1 and the layer “nnd” (“nsubstratendiff”).

**P-WELL (p-type substrate) stitches for NMOS transistors:**

- Place one stitch per two or three NMOS transistors.
- Stitches should be placed as close as possible to the NMOS transistor(s) you are trying to stitch.
- Connect the stitch to a nearby GND rail.
- The stitch layer in Magic is “ppc” (“psubstratepcontact”, “ppcontact”, “psc”, “pwc”, or “pwcontact”).
- Each “ppc” stitch is actually a *contact* between Metal 1 and the layer “ppd” (“psubstratepdiff”).
- Each “ppc” stitch must be placed contiguous to, or at least 4 lambda away from an, “ndiffusion” area.

## Viewing N-wells

Normally, Magic does not highlight or otherwise show the designer the exact location, placement, and geometry of N-wells; but if you would like to view them, this may be accomplished via the following Magic command:

```
:cif see CWN
```

## Your Final Project's "Lambda"

As of late 1996, there were at least four different MOSIS fabrication CMOS-technology lambdas (i.e. minimum active transistor gate widths) available for final project submissions:  $2.0\mu$ ,  $1.2\mu$  /  $1.0\mu$ ,  $0.8\mu$ , and  $0.5\mu$ . Note that the lambda value for a given CMOS technology is one-half of the minimum transistor gate width, in microns. This laboratory handbook was written assuming that a MOSIS 2.0 micron, N-well CMOS fabrication run would be used for your final project submission; and hence the lambda for your final project should be "1.0." Note that your particular MOSIS fabrication run *may be for a different CMOS technology and a different lambda*. If your fabrication process is not CMOS N-well,  $2.0\mu$  (lambda = 1.0), then be sure to use appropriate MOSIS TinyChip Standard Frame corresponding to your final project's lambda and CMOS technology. Also, everywhere a cif extraction command is given in this handbook as "lambda = x", replace "x" with the lambda for your particular MOSIS fabrication run. Ask your TA if the lambda value for your final project will be anything other than "1.0."

## The Use of Metal 3, Poly 2, and Linear Capacitors

Most likely, your device will be fabricated using a CMOS-technology, N-well, 2 Poly MOSIS process. Metal 3 (and linear capacitors) *may or may not* be available in the technology for your fabrication run, even though it is possible to use Metal 3 in your Magic designs. Since routing may be implemented in Metal 1, Metal 2, and Polysilicon, Metal 3 should not be needed, in general. Regardless, if you need to use Metal 3 in your final project design, ask your TA if it is available for your MOSIS fabrication run.

## Use a Hierarchical Design Approach

You are strongly encouraged to use a hierarchical design approach for your final project. That is, develop your design from the smallest logical cells and instantiate them (via the Magic command ":getcell") into higher-level, larger cells. If you simply "dump" all cells (thereby "flattening" your design to one level of hierarchy) into your final project's Standard Frame layout, minor changes – such as adding stitches – would prove to be much more time consuming and difficult than they would be if you had used a hierarchical design approach.

## Using IRSIM to Simulate Logic Devices with Feedback

In some cases, logic devices with feedback paths (such as flip-flops) must have their output states initialized (to avoid perpetually undetermined states) before simulation can be successfully accomplished in IRSIM. When testing a device, such as a flip-flop, first force the outputs (typically "Q" and " $\bar{Q}$ ") to an arbitrary initial state, execute at least one time step, then use the "x" command to disconnect the output nodes from their forced values to allow them to assert or unassert in subsequent time steps to their appropriate logic states. The following is an example of IRSIM commands used to test a "D" type, edge-triggered flip-flop with outputs, "Q" and "QP", input "D", and clock "CLK." Example:

```
stepsize 50
h Vdd
l GND!
clock CLK 1 0
l D
h Q
```

```
l QP
ana D CLK Q P
p
x Q
x QP
p
```

## Using IRSIM to Simulate Your Final Project Design

Since the MOSIS TinyChip Standard Frames are tested, evaluated, and guaranteed by MOSIS to be functional, it is not necessary to test the functionality of your final project's design *and* the digital I/O pads in IRSIM. Once you have completed your final design within the Standard Frame, simply remove the entire frame (i.e. all the pads) and re-save the file under a new name, for testing purposes only. Then extract the file for IRSIM using the Magic command “:ext.” Then, at the UNIX prompt, run the “ext2sim” utility:

```
ex2sim [-R] -T scmos input_filename
```

Then invoke IRSIM:

```
irsim -s scmos100.prm input_filename.sim
```

## Configuring Digital I/O Pads on the TinyChip Standard Frame

Documentation on how the TinyChip (CMOS N-well, 20 $\mu$  technology) Standard Frame digital, buffered I/O pads are designed and on how to configure them is contained in the “scn20\_pads.doc” text file in the Standard Frame directory.

- To configure a pad as a buffered input pad, connect “enable” to GND.
- To configure a pad as a buffered output pad, connect “enable” to VDD.

Although you are encouraged to use buffered inputs on input pads, an unbuffered input connection is also available at each pad.

## Unused Pads in Your Final Design

You are strongly encouraged to leave a few pads available in your final design for testing and debugging purposes only. That is, configure and connect these I/O pads to internal routings that have strategic significance in your design, or ones that might otherwise assist you in debugging problems when you are in the final testing phase of your device.

## The “Floating” VDD and GND Pads

In the 40pc22x22 Standard Frame, the top center and bottom center pads are isolated from the VDD and GND rails in the pad frame and may be used for separate VDD and GND connections within your project design. You are not required to use these pads unless you need them. However, even if the power and ground nodes for your final project are not connected to the Standard Frame's VDD and GND rails, you must still connect Standard Frame's VDD and GND pads (at the corners of the Standard Frame) correctly when you test your fabricated device, since the frame's power and ground rails are used to power and enable the digital I/O pads. Additionally, if you do not need either or both of the “floating” pads, and could use one or two additional digital I/O pads instead, you could delete the floating VDD and GND pads and replace them with digital I/O pads (named “io.mag”).

### **MOSIS Information on the Internet**

The MOSIS (Metal Oxide Semiconductor Implementation Service) is a low-cost prototyping and small-volume ASIC fabrication service for custom and semi-custom ASIC VLSI circuit development. Digital CMOS, mixed-signal digital and analog CMOS, GaAs, and multi-chip (MCM) fabrication technologies are available. MOSIS accumulates designs from different sources onto one mask set, allowing designers to obtain small quantities and share the cost of fabrication. As of late 1996, MOSIS information may be found on the World Wide Web at the URL: "<http://www.mosis.org>." An extensive online user manual is available, describing the design submission process, layout conventions, process technologies, and available packaging and bonding. An anonymous MOSIS ftp site is also available for downloading technology file, Standard Frames, and documentation files at the following address: "<ftp://ftp.mosis.org/pub/mosis>."

### **Miscellaneous Information**

A MOSIS Standard Frame with analog I/O pads is also available. MOSIS Technology files (although limited in scope and complexity) are also available for designers who wish to incorporate MEMS (Micro Electro-Mechanical System) devices into their final project designs.