Silicon Neurons Lab

October 30, 2015

Automated data acquisition

The primary purpose of this first lab is to characterize the properties of a silicon neuron, using the lab equipment and the MatLAB software which controls it.

The specific objectives of this lab are as follows:

1. To become acquainted with the lab instruments and data-acquisition techniques.
2. To become acquainted with the MATLAB program for data acquisition and manipulation.
3. To measure the characteristics of the low-power I&F neuron.

1.1 Computer account

Everyone in the class will use the same account, login `avlsi`, and password `7devil7`. This account exists only on the lab machines and these machines cannot be accessed from outside INI. If you want to send your data outside INI, you need to do it before you leave the lab. Once you log in, create a folder (subdirectory) for your data, named "bc-mygroup", where 'bc' stands for "block-course" and *mygroup* is a name your choose for your work group. Make sure you store all your data in this folder, and copy it to your personal files when you are done, for your records.

1.2 MatLab

Open a terminal on your workstation (e.g., by pressing `CTRL-ALT-T`), and type the command "matlab &" to get started. It is important that you launch matlab from the main "avlsi" directory (i.e., before changing directory). From within matlab change directory to your *bc-mygroup* folder.

For those who do not know `Matlab` yet there is online help available. Type "help help" and "help lookfor" to get useful starting information. The command helpdesk may bring up a browser showing HTML help if you are lucky. There are several custom commands that you will need to use in this lab. Type "help avlsi" to get a list.

As a first step, learn how to plot a Sine wave, from 0 to 2π with 20 points, using circles as markers (try "help plot"). Then learn how to subsequently add a plot of a Cosine with the same range to the Sine plot, but with a different point marker and color.

Now, make a plot of the following equation

$$I_{out} = I_0 e^{\frac{\kappa}{U_T} V_g}$$

using the following parameters: $V_g = 0 : 0.05 : 1$; $I_0 = 1e-13$; $\kappa = 0.6$; $U_T = 0.025$.

Whenever you generate plots it is important to properly label the axis and add figure legends, when necessary. Put the right labels on the x-axis and y-axis, including the dimensions in brackets (*e.g.* (A) for values expressed in Amperes).

Generate two figures (type "help figure"). In one figure produce a plot using the command "plot". In the other use the command "semilogy" ("help semilogy"). Then produce a third figure in which you put both types charts on a single plot (type "help subplot"). Remember to place the labels and legends where necessary.

Important: save all three figures, and generate PDF files for the final report. To produce proper PDF files, you will have to first save the plots as EPS (Encapsulated Postscript) files, and convert them to PDF afterward. To save your plots as EPS files use the matlab command "print -depsc <filename.eps>". Once you have your EPS files, you can use the linux command "epstopdf <filename.eps>" from a terminal to create the PDF file. You will need these files for your report, so make sure you store them in a safe place, that you can access from other PCs.

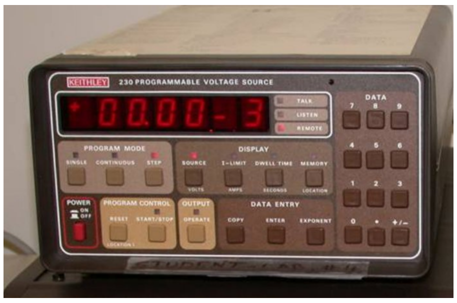

Figure 1.1: Keithley 230 Programmable Voltage Source.

1.3 The PotBox

You will also be using a “potbox” (a box with lots of potentiometers) for connecting to the chip and supplying it with analog bias voltages.

The potboxes are custom devices that provide access to the class chip pins and easily allow you to supply analog voltages (via the potentiometers) and to connect from the various BNC coax and triax connections to the chip. The potbox has a power switch and an power input BNC, on the right side. It also has an output BNC for the multimeter that allows you to measure which bias voltages you are applying. The rotating switch routes particular pots to the multimeter. There are two additional positions for measuring the power-supply voltage and arbitrary signals from the multimeter inputs on the potbox. Be very gentle with the potboxes; they can easily break if you are not careful with them.

1.4 The Keithley 230 Programmable Voltage Source

In its simplest manual mode, the 230 (Fig. 1.1) is a variable, floating voltage, source with output voltages from 0.1mV to 101V. Both positive and negative output voltages are available. The difference is floating relative to the chassis ground. The output voltage is set by the following procedure:

(i) Ensure that the OUTPUT OPERATE light is off by pressing the OPERATE button. This disables the output.

(ii) Ensure that the DISPLAY SOURCE light is ON by pressing the VOLTS button beneath it.

(iii) Type in the voltage using the key-pad in the DATA section (some part of the display will flash to indicate that you are entering a value) and press the ENTER key in the DATA ENTRY section.

(iv) Press the OUTPUT OPERATE button to apply the voltage to the test device.

Be *extremely* careful in typing in voltages, especially in setting the sign of all numbers and the correct exponent (with the EXPONENT key): transistors can tell the difference between 100mV and 100V! This is why you should *always* disable the output before you type in a new value. In order to try to protect the load device, the 230 has an internal current limit. The default of 2mA is adequate for testing integrated circuits. If the output is current-limited, the SOURCE light will flash.

1.4.1 Documentation

The K230 should have a brief guide attached to its top surface. For both instruments there is a Quick Reference Guide booklet. In addition, complete operation manuals for all lab equipment are available on a shelf in the lab.

1.5 General Facts about CMOS Chips

The following sections tell you about protecting the chip from static electricity, powering it up, and finding the correct pins.

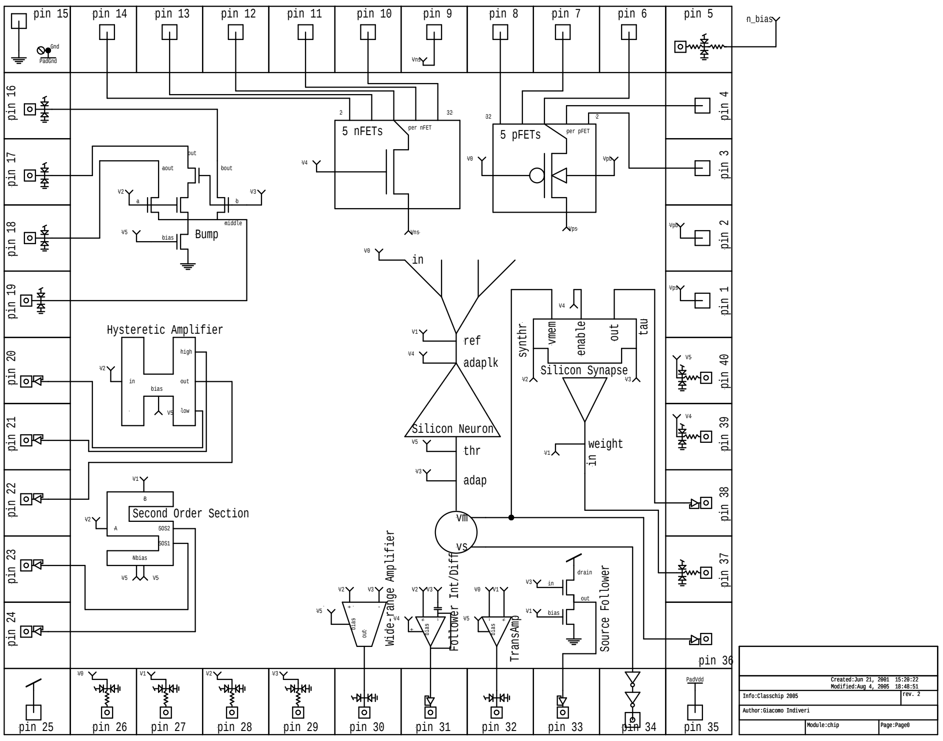

Figure 1.2: Class chip architecture and pinout.

1.5.1 Static Protection Measures

All MOSFET chips are *extremely* prone to damage by static electricity. The current through the transistors is controlled by an insulated gate. Even a few tens of volts can blow up the gate. A short walk across the room can build up kilovolts of static potential. There are electrostatic discharge (ESD) protection structures on the chip inputs that are designed to leak off the static charge before it can damage the chip, but often this will not be enough.

There are two simple precautions that can definitely keep the chip safe.

1. **When the chip is not powered up in a socket, keep it in its anti-static plastic case or stuck into a piece of black conductive foam.** This will short all the pins together.

2. **Always ground yourself to chassis (potbox) ground before picking up or touching a chip.** This will discharge the static charge.

1.5.2 Powering up The Chip

Before inserting a chip into the socket, hook up all the power and ground connections to the correct pins. You will need to hook up Vdd and *Gnd* even when testing isolated transistors, because the static protection structures at the pads won't work otherwise, and the transistor bulks need to be biased correctly.

Then, hook up whatever bias voltages you might need. Next, turn on the power supply, turn up the voltage to whatever you want (+5 volts for the n–well chips you will be testing), and then turn down the current limit until the voltage just begins to drop. Then turn up the current limit a little bit. This procedure will keep excessive current from flowing into your chip in the event there is a wiring error.

Next, turn the power on the potbox off, disable any other voltage sources you have hooked up, and then (while grounding yourself to the potbox chassis), put the chip in the ZIF (zero-insertion force) socket (make sure you put it in the right way, with pin 1, below the cutout on the package, plugged into to correspond with pin 1 on the potbox!). Don't forget to close the clamp on the ZIF socket. You can now power the potbox back on and enable the other voltage sources – in that order. If the voltage from the power supply suddenly drops, it probably means you have something hooked up wrong and the power supply is current limiting. The chips we use should only draw a few milliamps.

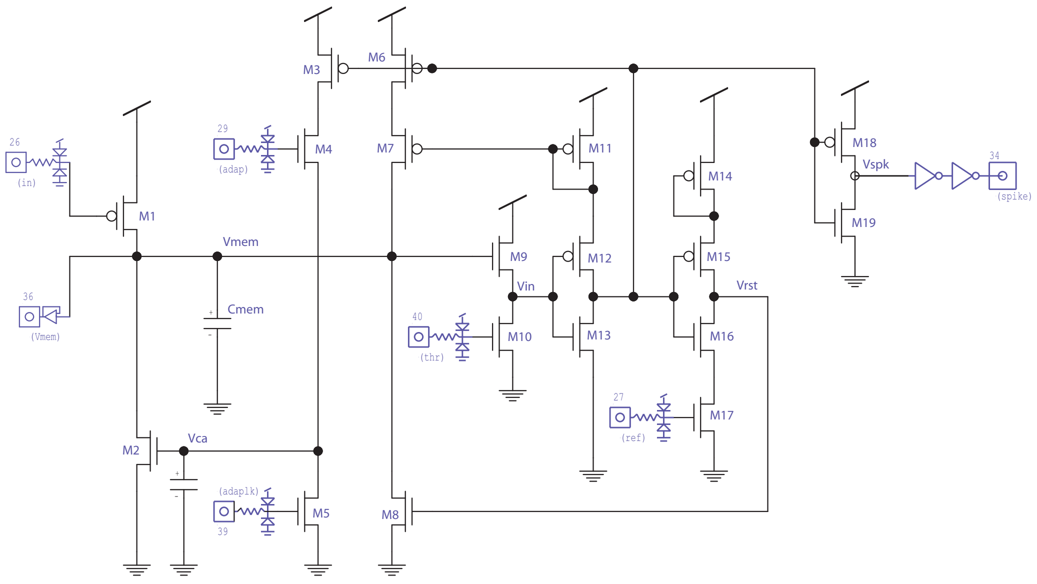

Figure 1.3: Schematic diagram of the low-power I&F neuron.

1.5.3 The Class chip and Chip Pinout Numbering

You will be using a test chip that was designed by the class in 2001 and later modified to improve it. The chip is called *Classchip2005Rev1*, and it can be identified by its fab ID *T29V-AK* (first fabrication lot, mostly broken) or *T71H-AF* (second fabrication lot). You can find a complete manual, schematics, and layout of the classchip on the class web page at http://avlsi.ini.uzh.ch/classwiki/doku.php?id=classchips.

Chip pins are numbered as shown in Fig. 1.2. Note that pin 5 is tied to Padbias. Padbias is the bias voltage for the wide range follower pads that buffer out analog signals from the chip. Connect this pin to a pot on the potbox and set it at 0.7V. Connect pins 25 and 35 to Vdd, and connect pin 15 to Gnd. Vdd is +5V for these chips. Its a good idea to also bias appropriately the unused N-FETs and P-FETs on the chip (*i.e.* connect pins 1, 2, 3, 4, 6, 7, 8 to Vdd and pins 9, 10, 11, 12, 13, 14 to Gnd).

The chips are numbered. Make sure you write down the number of the chip you use, so we can keep track of which chips might be bad. If you think you have a bad chip,**don't just put it back and grab another one**, and don't write "bad chip" on it either (in many cases the chip is fine and the chip tester is having problems). Point out the problem to your lab TA and let him/her figure out what to do about it.

1.6 The low-power I&F neuron

In this experiment you will need to find the bias parameters of the circuit such that it will model a real neuron's behavior as realistically as possible. Refer to the Fig. 1.3 for the circuit schematics. Connect all the neuron's bias voltages to pots with the following bias values:$V_{thr} = 0.7V$, $V_{ref} = 0.6V$, $V_{adap} = 0V$, $V_{adaplk} = 1V$.

1.6.1 Single spike plots

Inject a constant current I_{in} by connecting V_{in} (pin 26) to the Keithley 230 and use the oscilloscope to view time evolution of V_{mem} (pin 36). Inject a subthreshold current ($V_{in} \approx 4.3V$). Change V_{in} and V_{ref} until you obtain a biologically plausible trace (*e.g.* mean spiking frequency of $\approx 50Hz$, with refractory period of a few milliseconds).

Capture and plot three traces of V_{mem} for three different values of V_{ref}. Plot everything in the same figure and find values of V_{ref} such that the differences are visible on the plot. Save the figures and export the plots for your writeup.

To capture and plot scope traces in matlab use the commands "get_scope". Type: `>help get_scope` for instructions. Keep a log of all the biases and parameters you used, and all the observations you made for the final report. Similarly, save the plots to PDF files and store them in a safe place for inclusion in the report.

1.6.2 FI curves

Compute the frequency of the spikes generated by the circuit, as a function of input current and bias voltages.

Set V_{ref} to a very low subthreshold value (*e.g.* around 0.2V) and find the limits of the FI curve by changing V_{in} manually: at which value the neuron begins to spike, and after what value the frequency of the spikes stops increasing (saturates). Then measure (manually) the frequency of the spikes for at least 10 (but more are better) values of V_{in} in the range you just found, observing either the analog output (pin 36) or the digital one (pin 34). You can use the scope's measuring function or its cursors to measure the frequency. Save both input voltage and frequency values in two vectors (within Matlab), and repeat the same procedure for 2 different values of V_{ref}. Set the different V_{rf} values in a small neighborhood of the first V_{ref} setting, such that the ranges of valid V_{in} voltages are approximately the same.

After you collected all the data, plot on the same figure the three curves. Plot the curves on a semilog scale, and add appropriate legends with commands like:
`semilogy(Vin1,Fout1,Vin2,Fout2,Vin3,Fout3).`
`legend('V_{ref}=0.18V','V_{ref}=0.2V','V_{ref}=0.22V')`. **Note**: the values specified here are not necessarily the best values!
Save the plots to PDF files for inclusion in your report.