

A bias generator design kit

Tobi Delbruck (tobi@ini.phys.ethz.ch, <http://www.ini.unizh.ch/~tobi>)

Bias generator design kit home page at <http://www.ini.unizh.ch/~tobi/biasgen>

Download PDF version of this document at <http://www.ini.unizh.ch/~tobi/biasgen/biasgenDesignKit.pdf>.

To see what I mean by “bias current generator,” take a look at the paper about these circuits accepted to ISCAS 2004 in <http://www.ini.unizh.ch/~tobi/biasgen/delbruckBiasgenISCAS2004.pdf> or the slides about these circuits in <http://www.ini.unizh.ch/~tobi/biasgen/biasgenSlides.pdf>.

These circuits are for chips with known requirements for a number of fixed reference currents that don't depend on process technology or supply voltage. The bias currents could range over many orders of magnitude—from tens of microamps to a few picoamps. This design kit uses the Tanner design tools (www.tanner.com/eda) to compile the layout for a set of fixed bias currents that you specify in the schematic.

Motivation

The idea of this design kit is to make it much easier to design and use one of these bias generators. Lack of use of these generators has held back development of certain styles of analog chips (especially neuromorphic chips) because chip developers do not want to spend time and effort to understand and design what they consider to be standard circuits. As a result, designers put off this important part of the chip design. This procrastination results in chips that must be carefully “tuned” for proper functioning using off-chip potentiometers that set gate voltages that set bias currents. This is obviously not acceptable for production and not even very satisfactory for sample users, because each chip requires tuning and the values of the bias voltages supplied by the potentiometers strongly depend on the threshold voltage (and temperature), which varies from chip to chip and run to run. And designers too often fall back on the old line “it's not quite tuned up right” to excuse why the chip is not functioning correctly. Our experience with naïve chip users is that the fewer parameters they can adjust, the better. Using this kit should make it much easier and quicker to add these reference current generators to your chip and will make it much easier to support users of sample chips. In addition, you could feel a nice satisfaction that you really understand how your chip works and that it probably be manufactured in quantity.

The basic architecture of this bias generator is shown in the sketch below: This bias generator uses the bootstrapped current reference attributed to Widlar in its bipolar form (see the [paper](#) or the [slides](#), or for example see (Razavi, 00)) to generate a fixed reference current that is determined by a single resistor. This *masterbias* current is then split by a Bult&Geelen (Bult & Geelen, 92) R-2R splitter by octaves to make a series of smaller currents that are used to generate the desired currents. Each bias current is then generated by copying one of the splitter currents to a diode-connected transistor of the correct W/L geometry. The voltages on the gates of these diode-connected transistors are used to bias your circuits.

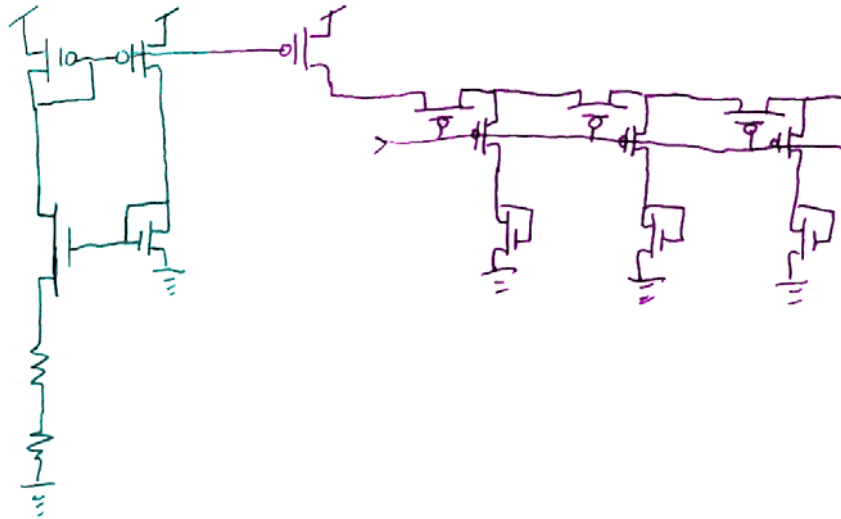


Figure 1 Architecture of bias generator

All the generated biases scale proportional to the master current, so you cannot change them individually except by overriding them by external connections. However, you can very easily scale the overall current by varying the off-chip resistance.

An experimental or development design should bring all biases that are generated out to pads for possible overriding, and should generate an individual bias for each type of circuit. The biases should not be shared, because it is generally not known ahead of time which circuits need different currents that those you designed—you want to be able to override them individually.

What the compiler does

You specify the currents by setting parameters of special cells in the schematic editor S-Edit—one cell for each bias current.

The compiler assembles all the biases, computes the range and determines the masterbias current and the required number of current splitter cells.

The desired currents are rounded to the nearest splitter current. The generated report file **biasgen-log.txt** and the ports placed on the individual bias cells show the ratio between actual (programmed) and desired currents.

The compiler uses a set of library layout cells to build the complete layout of a set of bias currents (and cascode biases) that are specified by parameters cells that are added to a schematic of the circuit.

The compiler parses the commands written out when you export the schematic to a SPICE netlist. The compiler handles all routing, and ports in the layout show where to wire up the resulting layout to your other circuits. A bus is generated that carries all the biases, for easy connection to your circuits. All the buses are labeled.

For LVS and simulation, you must modify the schematic for the correct number of splitter cells and their connections to the individual biases.

A set of supplied SPICE test benches makes it easy to simulate the bias generator with your favorite transistor models.

Design style and rules

The design kit was designed for maximum compatibility with a wide range of processes available from MOSIS (<http://www.mosis.org>) using scalable \square -based design rules. Design rules are MOSIS SCMOS_SUBM with additional restrictions:

1. Only 2-metal and single poly are used in the layout.
2. Only MOS capacitors are used (no poly2).
3. All low current nodes are shielded with metal2, and N-well guard rings that absorb photo-generated minority carriers surround N-fets.
4. All contacts use following contact cells: **polyContact**, **activeContact**, **via1**, **subsContact**, **wellContact**. This style makes changing the contact style easier for a different process.
5. The MOSIS SCMOS_SUBM design rules that are used are compatible with all MOSIS processes 0.35u and up, and can be fabricated directly in 0.25u, for example, using a slightly relaxed lambda of 0.15u instead of 0.12u.

Tool Requirements

Tanner Tools L-Edit version 10+. The l-comp compiler and UPI (user programmable interface) calls were developed this version of L-Edit. Version 9.x may work, but 8.x definitely will not.

Chip requirements

These bias generators generate fixed currents that are scaled from a single master current. Circuits that you can use these currents for are, for example

- Amplifiers
- Pulse-generating circuits that generate a fixed pulse width

- Current DACs
- Integrators and differentiators (or other filters) with fixed time constants
- Fixed current sources
- Cascode biases for circuit legs with known bias currents

Building your own generator

Getting the resources

Download the ZIP archive <http://www.ini.unizh.ch/~tobi/biasgen/biasgenDesignKit.zip> and unzip it to a new folder. You will see two folders, **biasgen** and **setup_SCMOS_SUBM_035u**. The setup directory contains the Tanner setup files for the Mosis SCMOS_SUBM rules as developed by us here at INI. These files are used for extraction and LVS, and there is a sample SPICE model file T0AD.md from a TSMC 0.35u process. You may need to do some work here if you use a different Tanner setup.

RUNNING THE DEMO

1. Open `biasgen-scmos-subm.tdb` in L-Edit
2. Use Tools/Macro.../Load... to load the macro `biasgen.c`, or open it in a text window in L-Edit and use F6 to load the macro.
3. Go back to the layout window and run the `biasgen` macro from the Tools menu, or by the shortcut F5. Running the macro will build the cell `biasgen`—a demo compiled generator. The result looks will look somewhat like this:

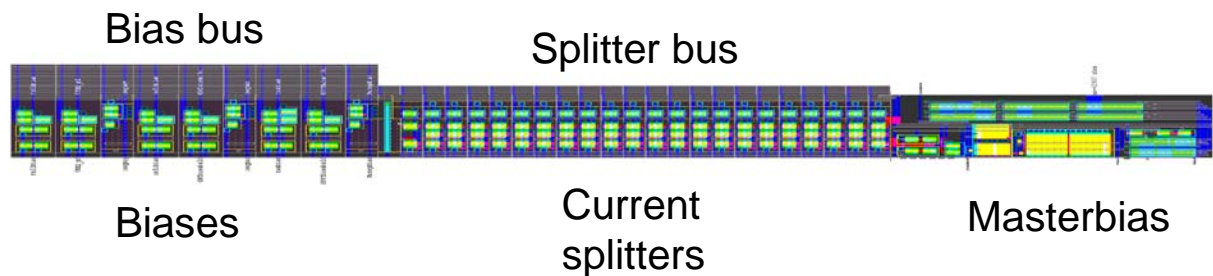


Figure 2 Layout of demo compiled bias generator

4. It corresponds to the schematic of the generator in the module `biasgen` in the schematic S-Edit `biasgen-scmos-subm.sdb`, which looks something this:

Current splitters

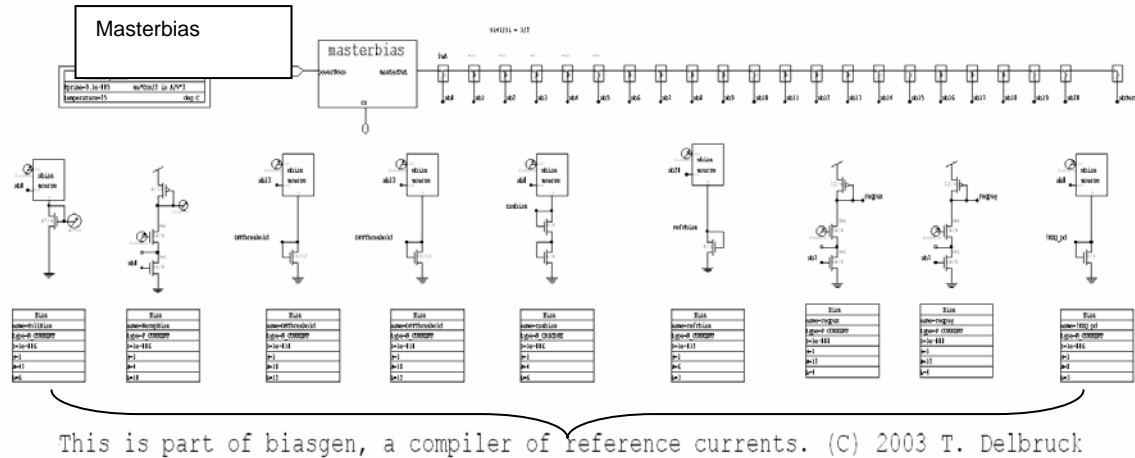


Figure 3 The schematic of the demo bias generator

Step-by-step guide to build your bias generator

1. To design and build your own generator, open the S-Edit file **biasgen-scmos-subm.sdb**. Open the module **biasgen**.
2. Add or change existing biases by duplicating the examples of the individual biases and modifying the parameters in the **.biasParameter** cells to specify the bias current, the sex of transistor, it's W/L, etc.

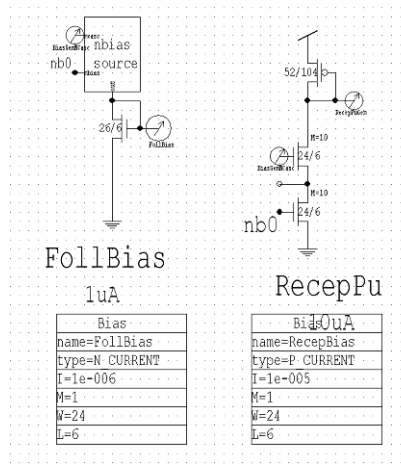


Figure 4 Example parameter cells for biases

3. The type of current can be one of the following: **N_CURRENT|P_CURRENT|N_CASCODE|P_CASCODE**; the different choices will result in different prototype cells for the individual biases. The **CURRENT** biases will just mirror the appropriate splitter output into a diode-connected transistor of the correct W/L, while the **CASCODE** biases will mirror the current into 2 series-diode-connected transistors.

4. Setting the current I will wire up the bias cell to the appropriate splitter output. The W/L and M settings will be placed as a port on the cell to remind you how to modify the $W/L(s)$ of the diode-connected transistor(s).
5. All the biases combined will be used by the **biasgen** macro to compute the range of currents, the required number of splitter cells, and the master bias current.
6. Edit the parameters in the **.biasProcess** cell for the process you will fabricate in. The $K'=\mu \cdot C_{ox}/2$ will affect the calculation of above-threshold masterbias resistance value, while the temperature will affect the Subthreshold masterbias resistance.

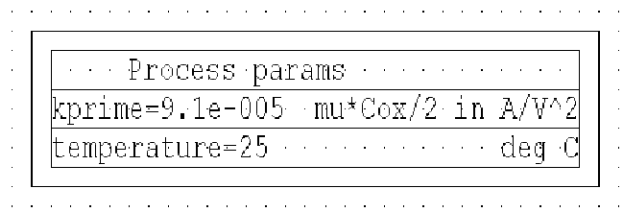


Figure 5 The process parameter cell

7. Export the SPICE netlist from S-Edit and save it as **biasgen.sp**.
8. Run the **biasgen** macro from L-Edit to build the bias generator. If you run it several times, you will be asked whether you want to overwrite your individual bias cells. These cells are copies of prototypes that you will modify so that the transistors have the correct W/L .
9. Modify the individual bias cells in L-Edit to make the transistors the correct W/L as shown by the ports in the individual bias cells. (The cells are named the same as the bias name.)
10. You need to fix the schematic to accord with the compiled layout. Look at the log output **biasgen-log.txt** and put the proper number of splitter cells in the schematic and the correct connections from splitter outputs to individual biases.
11. Run a design rule check on the layout (it should be OK, if not there is a bug!).
12. LVS the generated layout versus the schematic, making sure to check the W/L 's of the transistors. You can use the supplied **lvs.vdb** as a setup.
13. Wire the completed bias generator to your circuits and pads. Remember that all biases should come to unbuffered pads for possible overriding and filtering.
14. Wire up the bias generator according to the table in the next section.

Inputs and outputs

The following table summarizes the connections to the compiled bias generator.

| Port | Description |
|------|-------------|
| | |

| | |
|-------------------|---|
| Gnd | Ground. The lowest power supply voltage |
| Vdd | The highest supply voltage |
| rx | The external resistor is connected from here to ground. Parasitic capacitance here will destabilize the masterbias—see nbias. The resistor need not necessarily be external. An on-chip poly or well resistor could be used, but you need to watch out when using a well resistor that you don't make too much parasitic capacitance and that light-induced photocurrent is not significant. |
| powerDown | Soft power-down enable. Setting this high will turn off the bias generator. Setting it low (quickly enough) will turn it back on. Note that turning the bias generator on again requires some minimum down-going slew rate, which you can determine by simulation. |
| nbias | The n-type masterbias gate voltage. This may need to be externally bypassed with a few tens of pF to ground to stabilize the generator. |
| Individual biases | These should be brought out to unbuffered pads for possible bypass or overriding. N-type biases should be bypassed to ground and p-type biases should be bypassed to Vdd. Note that leakage current in the pad due to ESD protection, especially if light shines on the chip, can directly affect the bias current, especially if it is very small. |

Compiler (I-comp) bugs and other gotchas

Sometimes you cannot run the biasgen macro twice in a row without reloading it. The runtime stdio library has a bug. If you get the error “kprime is not set...” reload the biasgen.c macro from the Tools/Macro... menu.

If the alignment of the masterbias cell to the splitter cells is not correct, make sure that the Icon/Outline special layer (defined in the Setup/Special Layers... dialog) is set to be the “Cell Outline Layer”.

Simulations

1. Simulations can be done in T-Spice using the test benches **0masterbias.sp** and **0biasgen.sp** as starting examples. Include the transistor models for your fabrication process. The transistor dimensions are in lambda units, so don't forget to set the **.options scale=XX** correctly for your process. For example, for a 1.6u process, use **scale=0.8u**.
2. You can use **0splitter.sp** and **0testFets.sp** to check in more detail how the splitter will split the currents according to the SPICE transistor models.

3. The compiler writes the file **biasgen-print.sp** to generate test transistors and **.print** statements that measure the currents generated by the masterbias and the individual biases. You should check these values in the **Obiasgen.out** file or by probing the AC small signal parameters in the schematic and adjust the value of the external resistor. Real transistor characteristics that are actually captured in SPICE models may cause deviations in the predicted currents, so it is wise to check if the simulated values are close to what you desire.

Ensuring stability

Make sure you bring out the **nbias/BiasGenNBias** node to a pad, in case you have too much parasitic capacitance on the **rx** node. If you don't bring out **nbias** and your generator oscillates, then there is nothing you can do! (This has happened to me and to some other very experienced designers.)

Accuracy

This particular layout and design is presently in fabrication and results will not be available until Q4/2003, but earlier designs very similar to this one reliably create a set of bias currents spanning 6 decades of current (from 10uA to 10pA) and functioning in a 5V 1.6u process from 3.3-8V with a power supply sensitivity of less than 5%/Volt.

SPICE simulations of the design kit in a 0.35u process show a similar span of bias currents.

Use of bias currents that are rounded to the nearest octave splitter value means that actual bias currents can vary in principle by up to 25% from the desired values.

Non-ideal effects such as non-zero drain conductance will probably reduce this accuracy considerably. Based on past experience in a 1.6u process, actual currents vary by about a factor of 2-3 over a 6-decade range from the ideal splitter values but only a factor of 50% from the value predicted by BSIM3v3 SPICE simulations. (That means that the SPICE models are actually successful in capturing some of the non-idealities and more accurately predicting the actual behavior.)

Modifying for a different fabrication process

1. To use this design kit for a different process, you will need to adapt the cell layout to the new design rules.
2. You will need to change all the schematic W/L to (probably) non-scalable sizes.
3. The program will probably not need modification because there are no absolute sizes except in the **#define PORT_SIZE** and the occasional use of **LC_Lambda** to derive a dimension in internal units.

Comparison with other bias generators

This reference current generator generates known and fixed bias currents, which is good if you require known transconductances, time constants or pulse widths. By comparison, a specific

current reference generates a current that sits optimally just at the middle of moderate inversion, where maximum performance is obtained for minimum power.

A known and precise voltage reference is better created by a band gap reference. This design kit can generate stable reference voltages, but the threshold voltage and other parameters of the transistors will determine their value. These are OK for cascode biases, but not as precise, technology-independent voltage references.

References

1. Bult, G. and Geelen, G. (1992) An inherently linear and compact MOST-only current division technique, IEEE Journal of Solid-State Circuits, vol. 27, pp. 1730-1735,
2. Razavi, B. (2000) Design of Analog CMOS Integrated Circuits, <http://www.mhhe.com>

Licensing and use terms

By using this design kit, you agree to the following terms:

1. This kit is distributed under the terms of the GNU Public License (<http://www.gnu.org/licenses/licenses.html> - TOCGPL).
2. You are responsible for any use of these resources that infringe on existing patents.
3. We have built these circuits on several chips and they function correctly, but chip design is a tricky business, so there is absolutely no warranty as to functionality of these circuits.
4. Users of this resource on chips that result in publications must cite this work as the following: Tobias Delbruck, "A bias generator design kit", <http://www.ini.unizh.ch/~tobi/biasgen>, 2003.

Credits

Work on this style of bias generator was done during the period 1999 to the present by Tobi Delbruck and Andre van Schaik. Andre invented the soft power-down scheme presently used in the masterbias. Oliver Landolt taught us about the splitter circuits in Telluride. This bias generator design kit was conceived by Delbruck and begun at the 2003 Telluride Neuromorphic Engineering Workshop. Project members included Andre Van Schaik, Bernabe Linares, Guy Racmuth, and Ryan Tier. Delbruck completed the bulk of the work in Pasadena at Caltech and in Zurich at the Institute of Neuroinformatics just after the workshop in July and August 2003.

This is part of **biasgen**, a compiler of reference currents.
Copyright (C) 2003 T. Delbruck

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by

the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA