Programmable bias generator user guide

Luis Alejandro Camuñas Mesa, Tobi Delbruck

With the help of this guide, we expect that you can build your own bias generator just by properly combining the cells that we have already designed. The design kit resources are available at the following URL:

http://www.ini.unizh.ch/~tobi/biasgen

This site also has links to a conference paper that describes the architecture and performance of these circuits and includes information useful for system integration (microcontroller and host software). These resources are supplied under a very non-restrictive license.

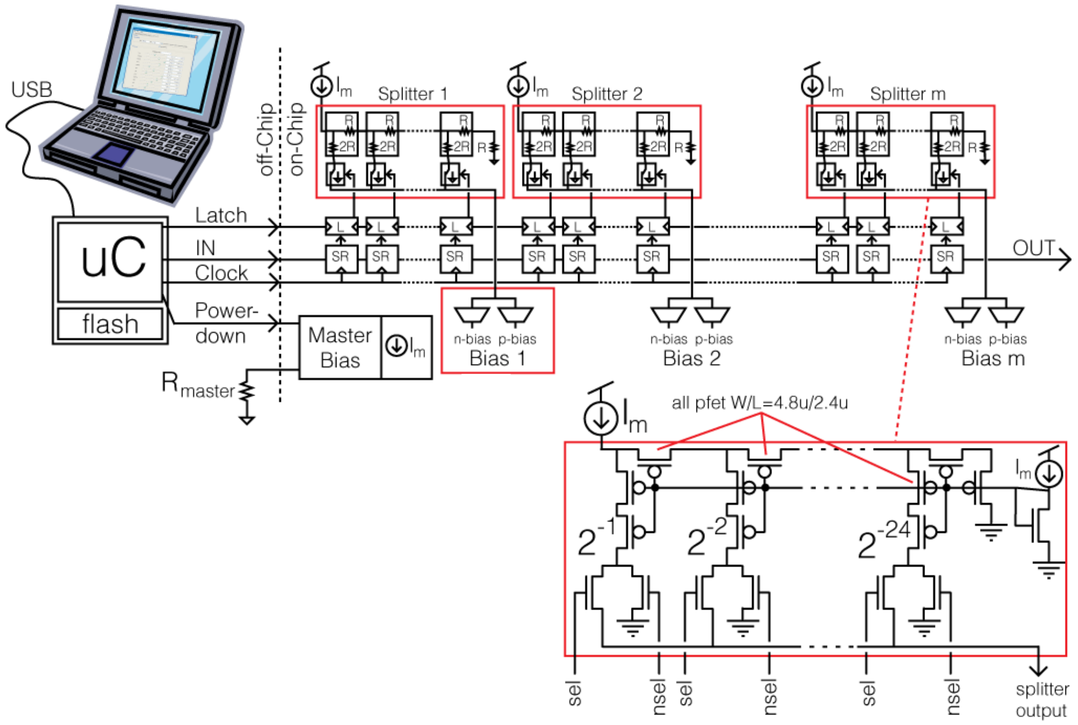
Functionality

The basic functionality of the programmable bias generator is that it allows generation of programmable current ranging over many orders of magnitude. The kit uses these programmable currents to generate bias voltages sourcing the bias current into diode-connected transistors and then buffering the resulting gate voltage. All the bias currents are derived by splitting a single master current. The biases are programmed over a 3-wire serial digital interface to the chip. Typically we have supplied these digital values using an off-chip microcontroller.

Requirements

This kit is designed specifically for the Austria Microsystems (AMS) 0.35um 4M-2P process. Using this process means you will need to do a minimum of layout modifications.

In the following figure you can see an overview of the basic bias generator structure. The Master Bias block generates the input current for the splitters, and the single-phase registers (SR) select one of the outputs from the splitter. This output current will be used to generate the n-bias and p-bias voltage signals that are used in the chip core.



For the cells called ‘Bias m’, we have designed three different configurations, so that you can use for each bias the one that better fits what you need:

- A dual n-type and p-type current buffer consisting of two active mirrors, which serve to isolate the splitter from the rest of the chip and to greatly lower the bias voltage impedance.
- A similar current buffer, but including cascoded mirrors. These supply bias voltages that are two diode drops from the supply rail and are useful for biasing cascodes.
- An alternative pair of sub-picoampere current mirrors which will be used when we want to generate small bias currents.

Now, we will talk about each one of the cells inside the bias generator, describing first the basic behaviour of them, and after that explaining in detail how you can use them to build your own bias generator. Then, we will give some general considerations to help you building the whole circuit.

Master Bias

This cell is a bootstrapped current reference which generates the master current, including also start-up and power control circuits. This master current ‘Im’ will be the input of the current splitters, by driving the global node ‘pMirrorGate’ to the gate of the input transistors of these splitters. There is also another global node called ‘BiasGenNBias’, which is used for biasing the splitters, and it is just a copy of the master current. Let’s take a look now at the input and output signals that you have to use.

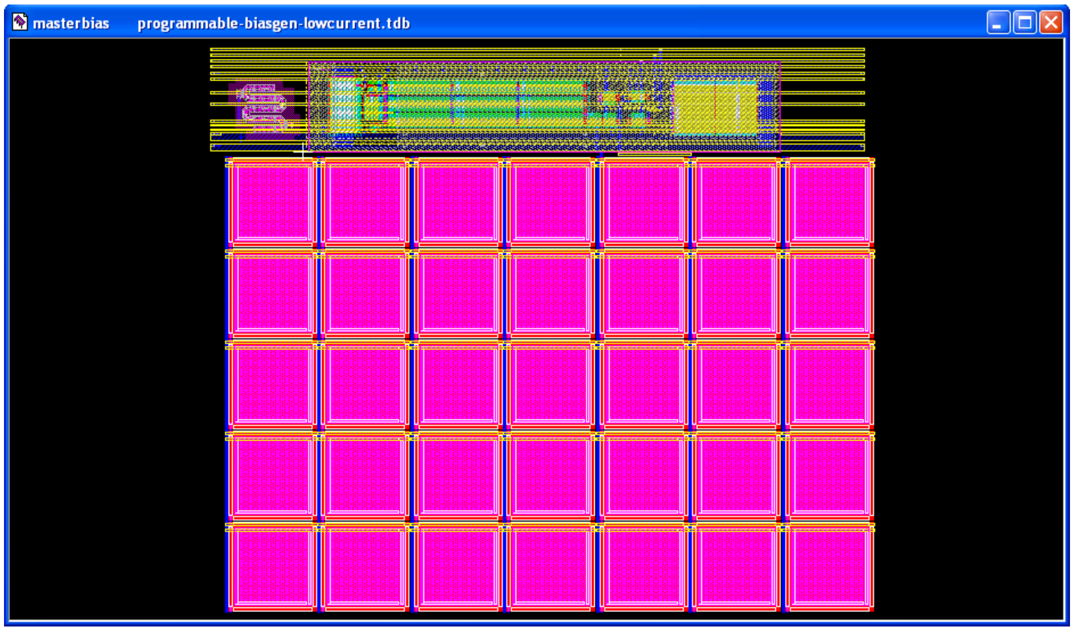
Input signals

- powerDown: this signal can be an output of the microcontroller, and is used to turn off all currents. It is also used by the 'biasProgrammable' cells, but we will talk about that later.
- rx: this signal will be also driven to a pad, so that we can use an off-chip resistor instead of the integrated one.

Output signals

- pMirrorGate: you have to bring this signal to the gate of the input transistors in the splitters, so that the master current will be the input of these cells.
- BiasGenNBias: this signal is just a copy of the master current, and you have to connect it to the corresponding transistors in the splitters. Both of these output signals are considered global nodes, so you don't have to do anything with them in the schematic view, but you must make sure that they are properly connected in the layout view.

In the following figure you can see the layout of the cell called 'masterbias'. Below the main circuit, we have included a group of capacitors to achieve a stable master current.



Bias Buffer Splitter

This cell is a programmable current splitter with 8 bits, whose input is the master current, and it's used to generate the bias currents for the current buffers inside the 'biasProgrammable' cells. This cell has been added to the circuit to reduce power consumption by the current buffers, which used to be biased by the master current before.

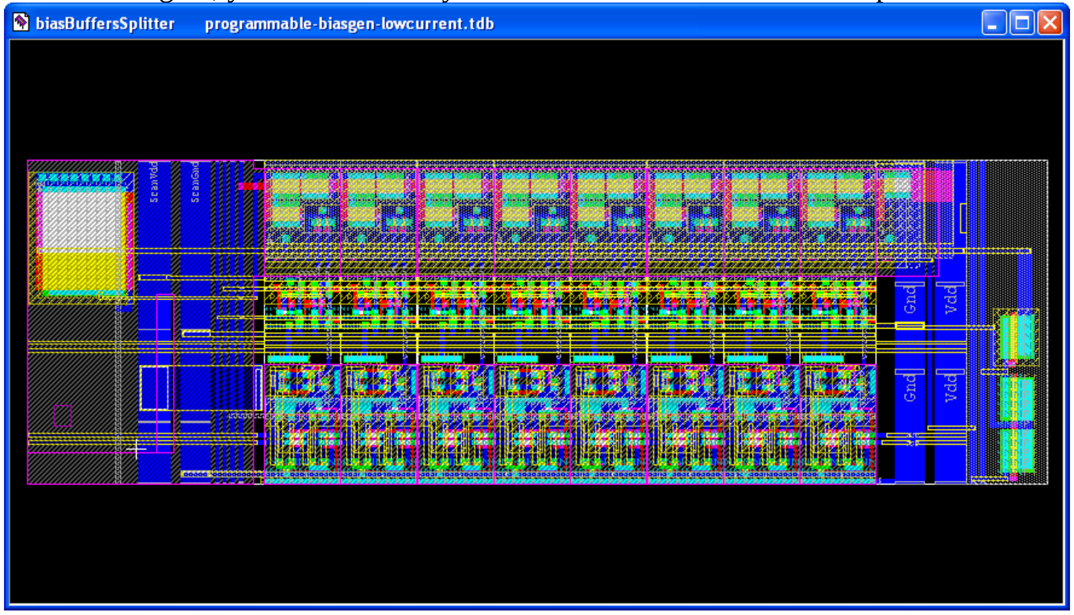
Input signals

- pMirrorGate: this is the global node which is the output of the 'masterbias', and you have to bring it to the input transistor of this cell.
- in and nin (inverted in): these are the digital inputs of the single-phase register, and they should be connected to the microcontroller. In the schematic view, these inputs are called 'd' and '!d'.
- clock: this is also a signal generated by the microcontroller, and you have to make sure that it's connected to all the registers. In the schematic view, this signal is called 'phi'.
- latch and latch!: they are also outputs from the microcontroller, and they are used to program the register properly. You must make sure that they are connected to all the registers. In the schematic view, these inputs are called 'latchenb' and '!latchenb'.

Output signals

- BiasBufferNBias and BiasBufferPBias: they both are global nodes which are used to bias the current buffers and sub-picoampere mirrors in all the 'biasProgrammable' cells.
- out and nout (inverted out): these are the digital outputs of the single-phase register, and they should be connected to the inputs of the next register (out connected to in, and nout connected to nin). In the schematic view, they are called 'q' and '!q'.

In the next figure, you can see the layout of the cell called 'biasBuffersSplitter'.



Bias Programmable

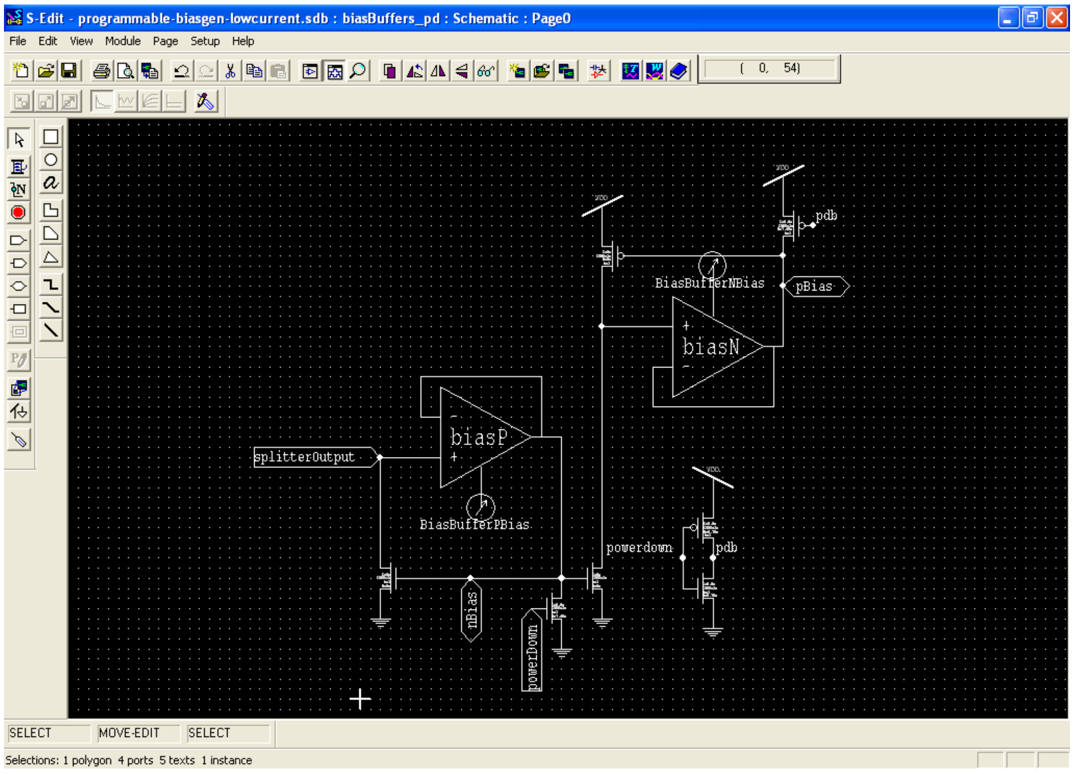
This cell must be considered in a different way from the previous ones. When we were talking about the 'masterbias' and the 'biasBuffersSplitter', it was important to notice that only a single instance of each would be included in the hole circuit. However, you have to insert as many 'biasProgrammable' cells as you need.

First, you have to make a list of all the bias currents that your circuit needs, and then you have to decide what kind of bias current each one should be. You have three different cells to implement the last stage of the bias generator:

- The standard bias, which is a dual n-type and p-type current buffer consisting of two active mirrors, that serve to isolate the splitter from the rest of the chip and to greatly lower the bias voltage impedance. You can use this one if you don't have any other constrictions.
- The cascoded bias, which is the same, but with the only difference of using cascoded buffers.
- The low-current bias, which is the one you will have to use for those bias currents below pico-amperes.

For each one of these three possibilities, you just have to insert the corresponding 'biasProgrammable' circuit, and that's all. They all have the same splitter, with 24 different outputs and a 24-bit single-phase register to select the correct output. The only difference between the three of them is just the buffering of the output current coming from the splitter.

Inside the buffer circuit, they all include a power down mechanism to make sure that the bias generator is not consuming power while it's not really biasing the circuit. The buffer circuit can be seen in next figure.


S-Edit - programmable-biasgen-lowcurrent.sdb : biasBuffers_pd : Schematic : Page0
File Edit View Module Page Setup Help
(0, 54)
pdb
BiasBufferNBias
pBias
biasN
biasP
splitterOutput
BiasBufferPBias
powerdown
pdb
nBias
powerDown
SELECT
MOVE-EDIT
SELECT
Selections: 1 polygon 4 ports 5 texts 1 instance

So, let's now describe the inputs and outputs of this cell.

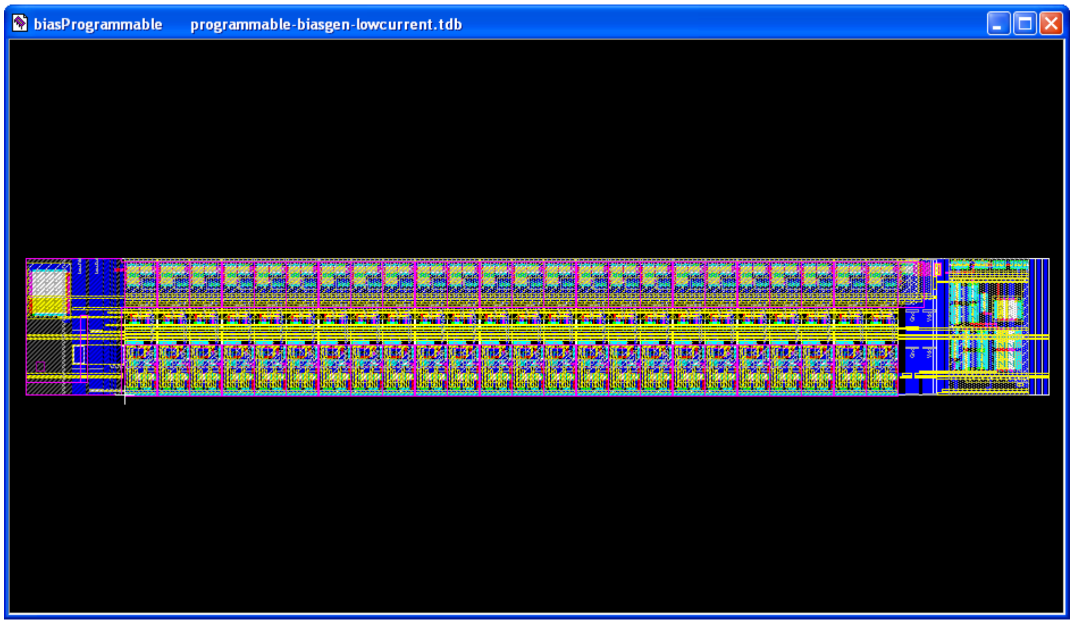
Input signals

- pMirrorGate: this is again the global node that you have to connect to the input transistor of the splitter to obtain a copy of the master current.
- in and nin (inverted in): these are the digital inputs of the single-phase register, and they should be connected to the outputs of the previous splitter. In the schematic view, these inputs are called ‘d’ and ‘!d’.
- clock: this is a signal generated by the microcontroller, and you have to make sure that it’s connected to all the registers. In the schematic view, this signal is called ‘phi’.
- latch and latch!: they are also outputs from the microcontroller, and they are used to program the register properly. You must make sure that they are connected to all the registers. In the schematic view, these inputs are called ‘latchenb’ and ‘!latchenb’.
- powerDown: this is the same signal that we use in the ‘masterbias’ cell, and will be generated by the microcontroller to make sure that the bias generator is not consuming power while it’s not generating output currents. In this case, this signal turns off the transistors that should generate the bias currents.

Output signals

- pbias and nbias: these are the two possible outputs of each bias cell. When you decide the kind of cell that you need for your bias current, you also must know if it’s n- or p- bias. Anyway, in each cell you have both outputs ready to use, you only have to connect the right one to your circuit while leaving the other one floating.
- out and nout (inverted out): these are the digital outputs of the single-phase register, and they should be connected to the inputs of the next register (out connected to in, and nout connected to nin). In the schematic view, they are called ‘q’ and ‘!q’.

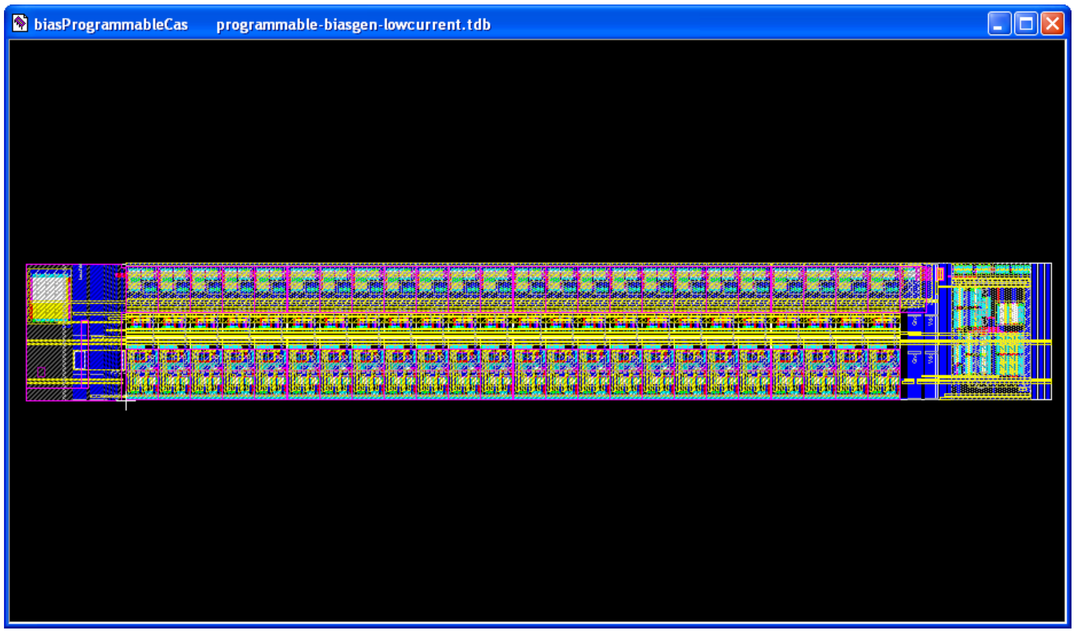
The layout of the cell called ‘biasBufferProgrammable’ is shown in the next figure.



Bias Programmable Cas

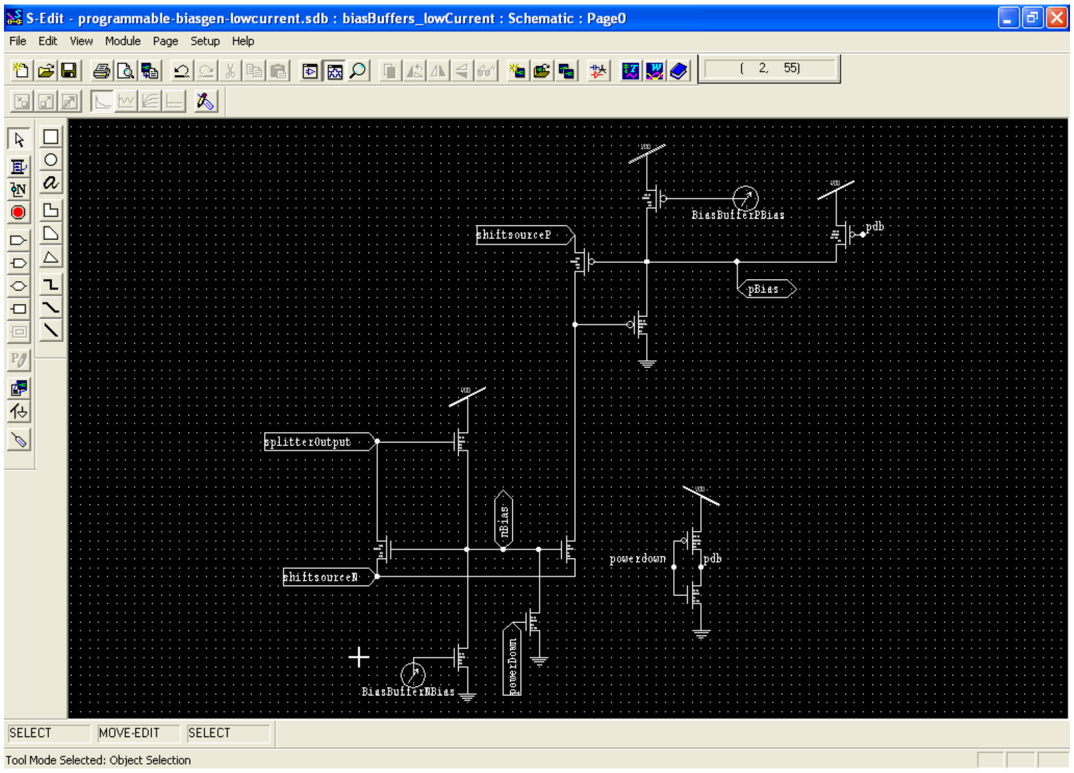
This cell is just a small variation from the previous one. The only difference is that the buffer circuit contains cascoded current mirrors to generate the outputs ‘nbias’ and ‘pbias’.

It’s not necessary to mention the input and output signals in this case, because they are exactly the same ones that we described in the previous cell, so you have to do the same things that before. Let’s just take a look at the layout of the cell called ‘biasProgrammableCas’ in the following figure.



Bias Programmable Low-Current

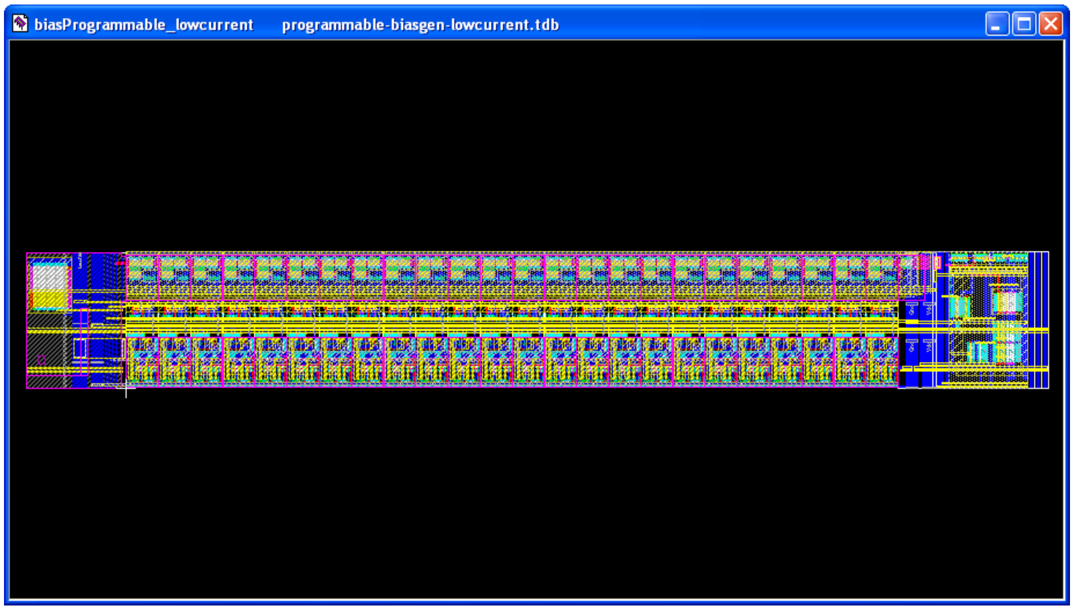
This is also a variation from the standard ‘biasProgrammable’ that we have already described. In this case, the splitter is exactly the same, and the only difference is in the buffer circuit. To produce low-current outputs, we don’t use the same current buffers as before, but we just use sub-picoampere mirrors. We can see the schematic in the next figure.


S-Edit - programmable-biasgen-lowcurrent.sdb : biasBuffers_lowCurrent : Schematic : Page0
File Edit View Module Page Setup Help
(2, 55)
shiftsourceP
BiasBufferPBias
pdb
pBias
splitterOutput
nBias
powerdown
pdb
shiftsourceN
powerDown
BiasBufferNBias
SELECT
MOVE-EDIT
SELECT
Tool Mode Selected: Object Selection

Concerning the inputs and outputs, most of them are exactly the same ones as those in the previous circuits. The only difference is that we have two new input signals in this case:

- shiftsourceP and shiftsourceN: as you can see in the previous figure, the sub-picoampere current mirrors that we have implemented here don't have the sources of the transistors connected to ground or vdd, but to a different voltage, which will be several hundreds of milivolts above ground (for n-mirrors) or below vdd (for p-mirrors). So, these are two extra input signals that are generated by the cells that we will describe afterwards. It's important to notice that if you use one of these low-current biases, you will have to use these 'shiftsourceP' or 'shiftsourceN' also inside the core, so that the current mirror can work properly.

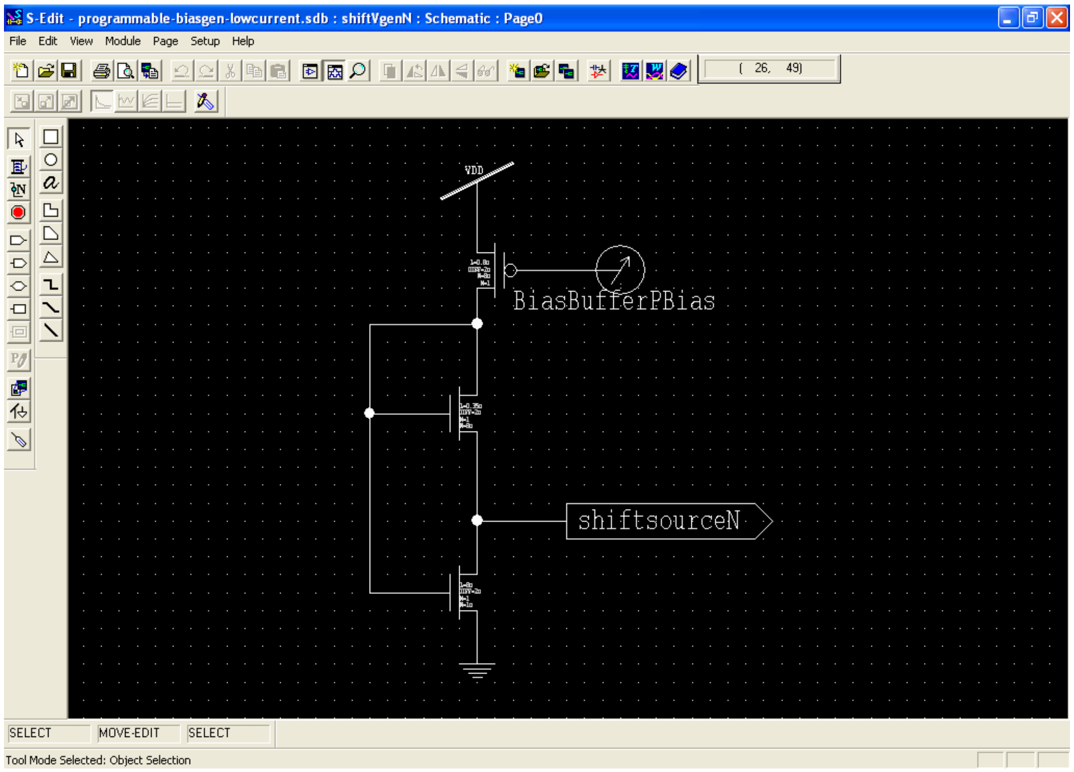
So, let's now take a look at the layout of the cell called 'biasProgrammable_lowcurrent' in the following figure.


biasProgrammable_lowcurrent programmable-biasgen-lowcurrent.tdb

And after that, we have to talk about the cells that generate these ‘shiftsourceP’ and ‘shiftsourceN’ voltages.

ShiftVgenN

This cell is necessary to generate the bias voltage for the source terminals of the n-type sub-picoampere mirrors. You have to bias this cell with the global node ‘BiasBufferPBias’, and then connect the output to all the mirrors that need it. The schematic of this cell can be seen in the following figure.


S-Edit - programmable-biasgen-lowcurrent.sdb : shiftVgenN : Schematic : Page0
File Edit View Module Page Setup Help
(26, 49)
VDD
BiasBufferPBias
shiftsourceN
SELECT MOVE-EDIT SELECT
Tool Mode Selected: Object Selection

As you can see, the inputs and outputs are as follows.

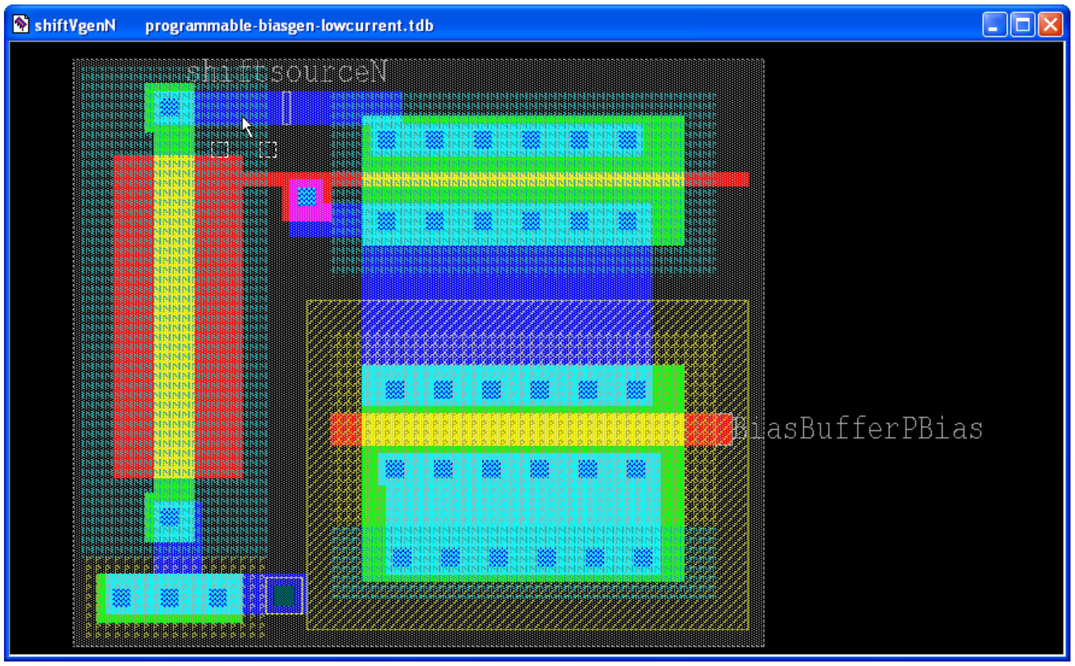
Input signals

- BiasBufferPBias: this global node is generated by the cell ‘biasBuffersSplitter’, and is just the output current obtained after dividing the master current accordingly with the programmed data.

Output signals

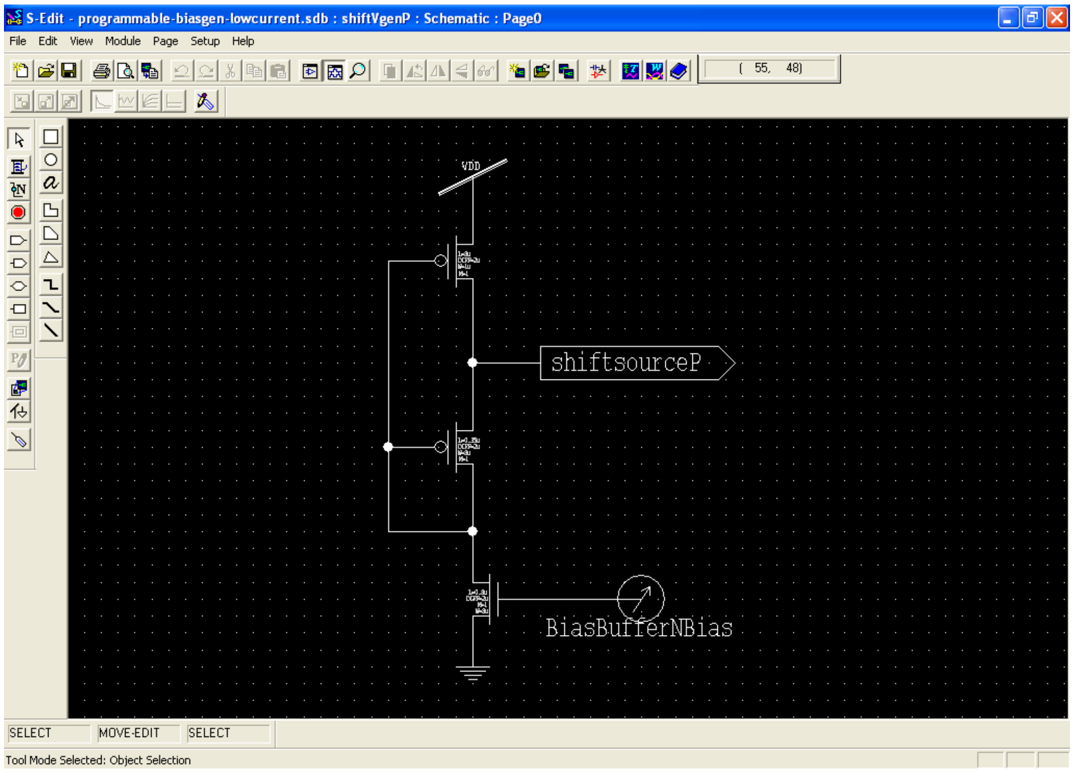
- shiftsourceN: this is the signal that you have to make sure that is connected to all the sources of the n-type sub-picoampere current mirrors that you use.

And the layout of this cell called ‘shiftVgenN’ is shown in next figure.



ShiftVgenP

This cell is necessary to generate the bias voltage for the source terminals of the p-type sub-picoampere mirrors. You have to bias this cell with the global node ‘BiasBufferNBias’, and then connect the output to all the mirrors that need it. The schematic of this cell can be seen in the following figure.


S-Edit - programmable-biasgen-lowcurrent.sdb : shiftVgenP : Schematic : Page0
File Edit View Module Page Setup Help
(55, 48)
VDD
shiftsourceP
BiasBufferNBias
SELECT
MOVE-EDIT
SELECT
Tool Mode Selected: Object Selection

As you can see, the inputs and outputs are as follows.

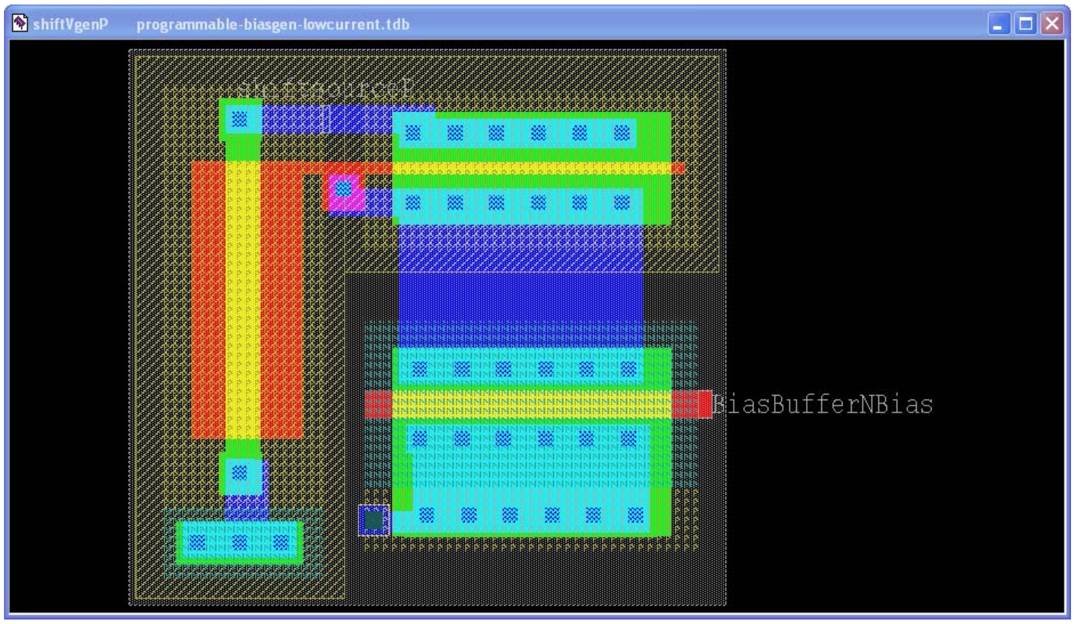
Input signals

- BiasBufferNBias: this global node is generated by the cell ‘biasBuffersSplitter’, and is just the output current obtained after dividing the master current accordingly with the programmed data.

Output signals

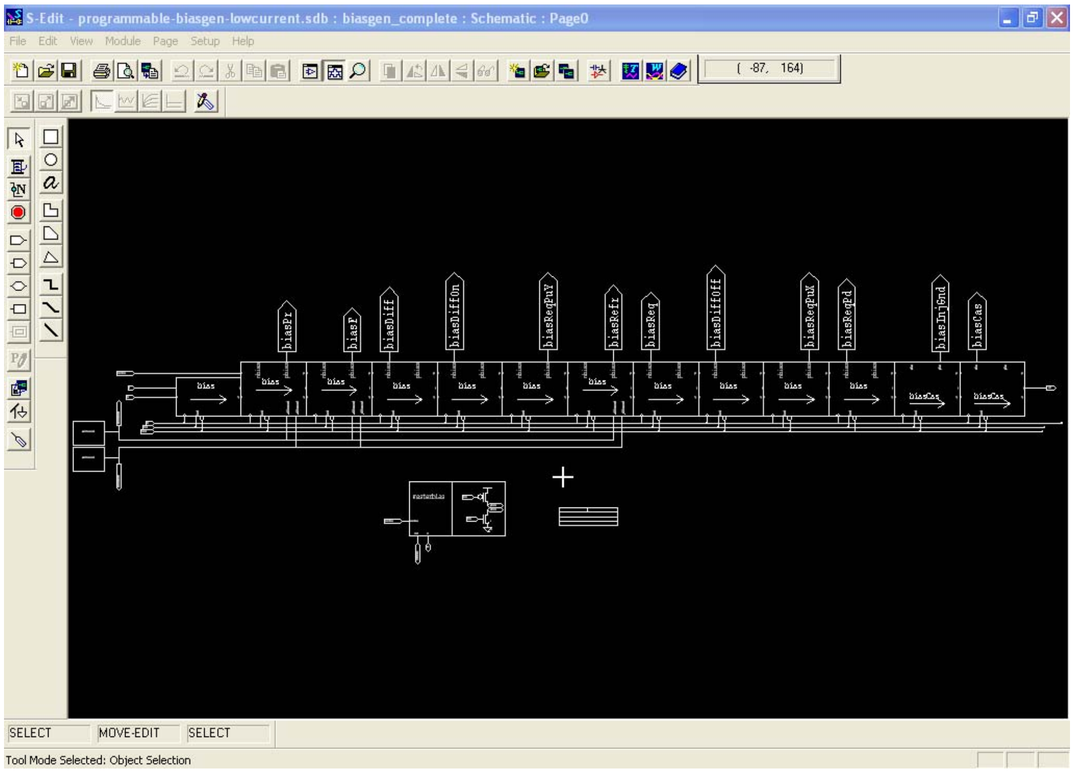
- shiftsourceP: this is the signal that you have to make sure that is connected to all the sources of the p-type sub-picoampere current mirrors that you use.

And the layout of this cell called ‘shiftVgenP’ is shown in next figure.



Summary

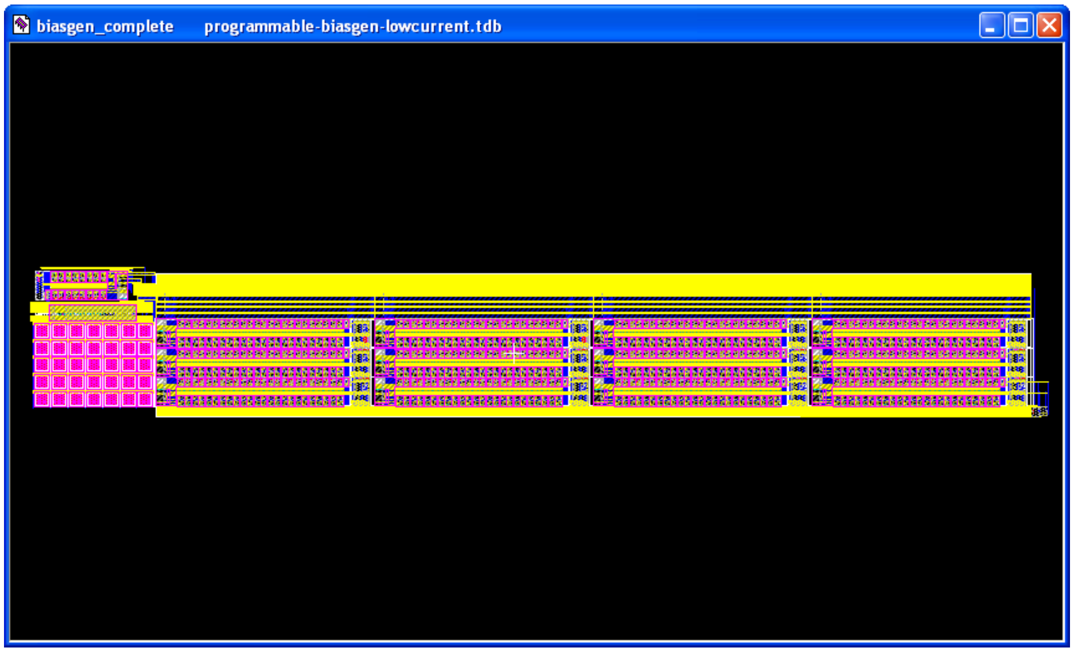
1) The first thing you must do is write a list of all the current biases that you need for your circuit
2) Over that list, indicate if it’s n- or p-bias, and also what kind of output stage you will need (standard, cascoded or low-current)
3) Then, you can start to build the schematic of your bias generator. You have to instance the cell called ‘masterbias’, the cell called ‘biasBuffersSplitter’, and as many cells called ‘biasProgrammable’, ‘biasProgrammableCas’ and ‘biasProgrammable_lowcurrent’ as you need. Also, if you have one single ‘biasProgrammable_lowcurrent’, you also have to instance the cells called ‘shiftVgenN’ and ‘shiftVgenP’.
4) Once you have all the cells, you have to connect them properly. You just have to put all the ‘biasProgrammables’ cells together, in the order that you decide, making sure that the digital outputs and inputs of the registers are connected. Then, you have to put the cell ‘biasBuffersSplitter’ in front of all of them, and connect all the inputs to the registers together. Then, connect the outputs of the blocks ‘shiftVgen’ to the right ports. You don’t have to connect the ‘masterbias’ to anyone, because it only uses global nodes as outputs. Then, you will have something similar to next figure.


S-Edit - programmable-biasgen-lowcurrent.sdb : biasgen_complete : Schematic : Page0
File Edit View Module Page Setup Help
(-87, 164)
biasPr
biasF
biasDiff
biasDiffOn
biasRegPuX
biasRefr
biasReq
biasDiffOff
biasRegPuX
biasRegPd
biasInjGnd
biasCas
bias
biasCas
masterbias
SELECT
MOVE-EDIT
SELECT
Tool Mode Selected: Object Selection

5) Now, you can start with the layout of the circuit. First, you have to instance all the cells, just as you did in the schematic.
6) Then, you must make sure that the 'biasProgrammables' cells are connected together in an array configuration (in our example, three rows, four columns). Then, you have to add one cell called 'biasShiftRegisterConnectRight' on the right of every row that you have, and another cell called 'biasShiftConnectLeft' on the left of every row, to make sure that they are all connected.
7) Now, connect the cell 'biasBuffersSplitter' making sure that the digital outputs will be connected to the inputs of the next stage. Also, check that all the common input signals to the registers are connected to all of them.
8) Connect the outputs of the cell 'masterbias' and the outputs of the cell 'biasBuffersSplitter' to the rest of the blocks. You should use buses for these, the cells are designed so that it won't be difficult to route them.
9) After that, you just have to connect the outputs of the cells 'shiftVgenN' and 'shiftVgenP' to the cells that need it.
10) Once you have reached here, you must connect all the bias outputs to make it accessible to the rest of the circuit, and check that you connect properly all the supply nodes, which are four: digital vdd, digital ground, analog vdd and analog ground.

In general, I would recommend you to check periodically for Design Rules Check errors, and also try to check Layout Versus Schematics to make sure that you have everything connected.

If you could follow all the steps, now you have your bias generator ready to be used in your circuit. In the next figure, you can see the layout of the example that we have built.



You can use this finished example to check how everything is connected, and try to do it in a similar way.

Revision history

Oct 2006: Tobi Delbruck slight edits to this document, packaged zip
June 2006: Luis Camunas wrote this document during visit from Linares-Barranco lab, IMSE