RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Exercise 5:
Current Mode DAC

Prof. Dr. P. Fischer

Lehrstuhl für Schaltungstechnik und Simulation
Uni Heidelberg

The Goal

- In this longer exercise you should design a 6 (or 8) Bit Digital-Analogue-Converter which generates a variable current.
  - To be more precise, it has a 6 bit wide digital input D⟨5:0⟩ and should generate 0...63 µA depending on this value

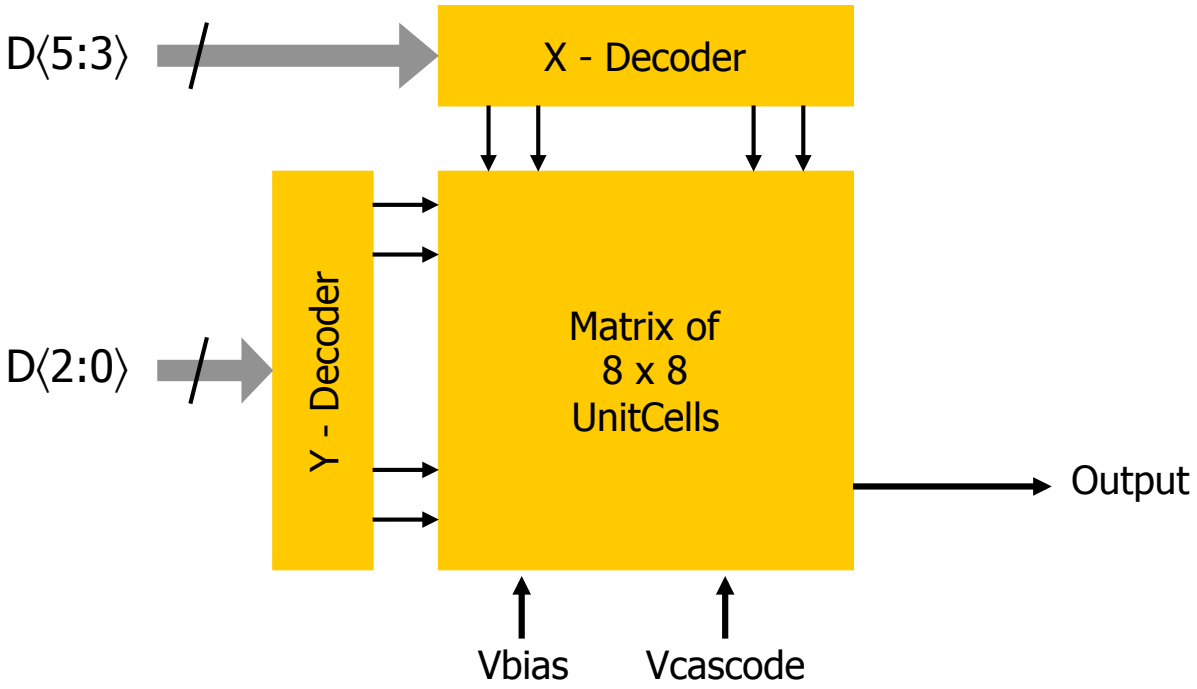
- You have to
  - Understand the design
  - Produce a schematic hierarchy
  - Simulate it
  - Make a compact, DRC free and LVS clean layout

RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

THE DESIGN

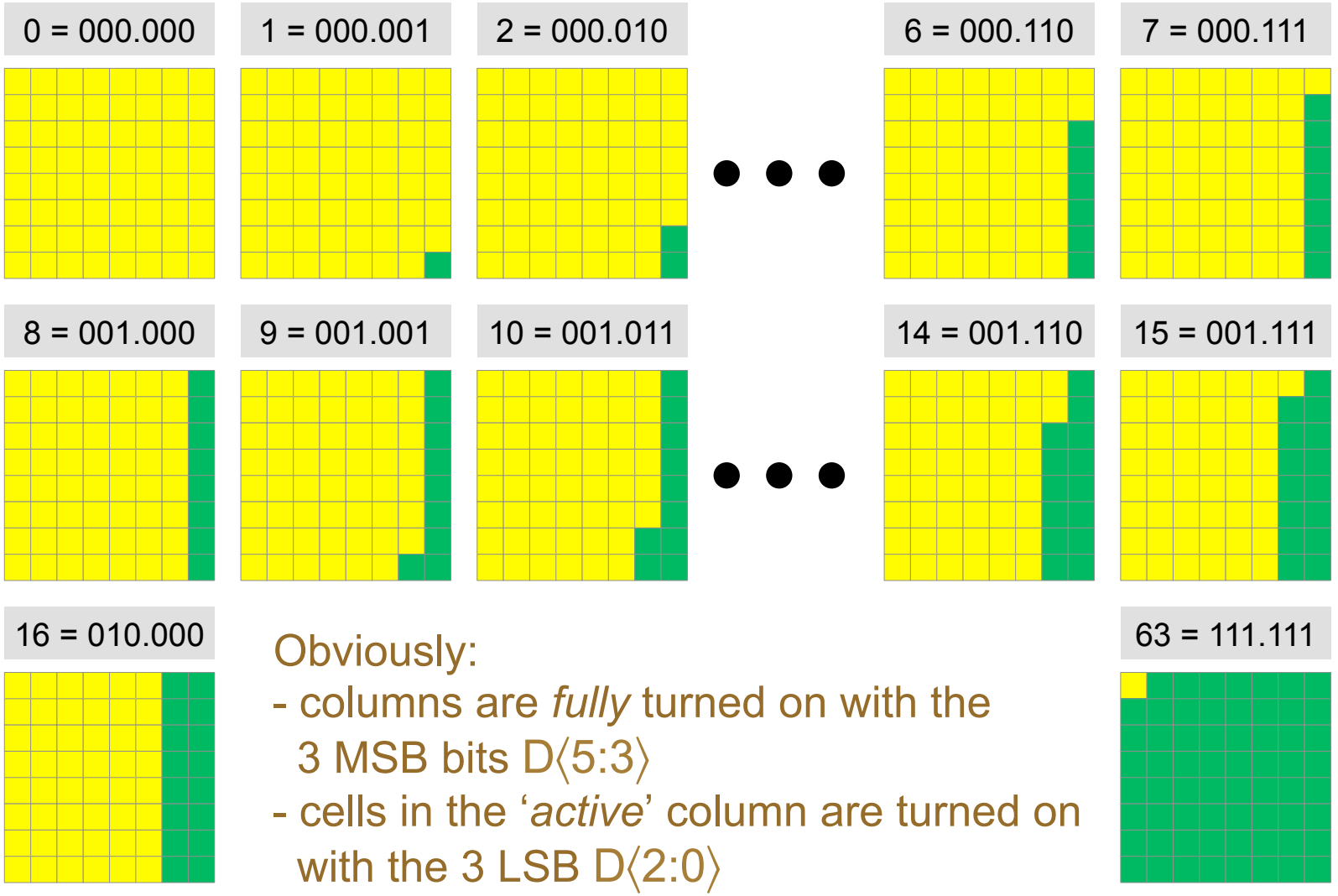
The Circuit (for a 6 Bit DAC)

- The output is the sum of currents from 63 *UnitCells*
- Each *UnitCell* can send a unit current I_0 = 1 µA or nothing
- We arrange the unit cells in a matrix of 8 x 8 cells
- Control signals running horizontally and vertically turn on the cells one after each other (as a function of input code)
  - These signals are generated in two *decoders*



Turing on the Cells

- In a first version, the cells are turned on like this:



Obviously:
- columns are *fully* turned on with the 3 MSB bits D⟨5:3⟩
- cells in the '*active*' column are turned on with the 3 LSB D⟨2:0⟩

Implementing This

- We define 3 control signals:
  - ColOn (vertical) turn on all cells in the column
  - ColSel (vertical) selects one column as 'active column'
  - Row (horizontal) used to turn on cells in the *active* columns

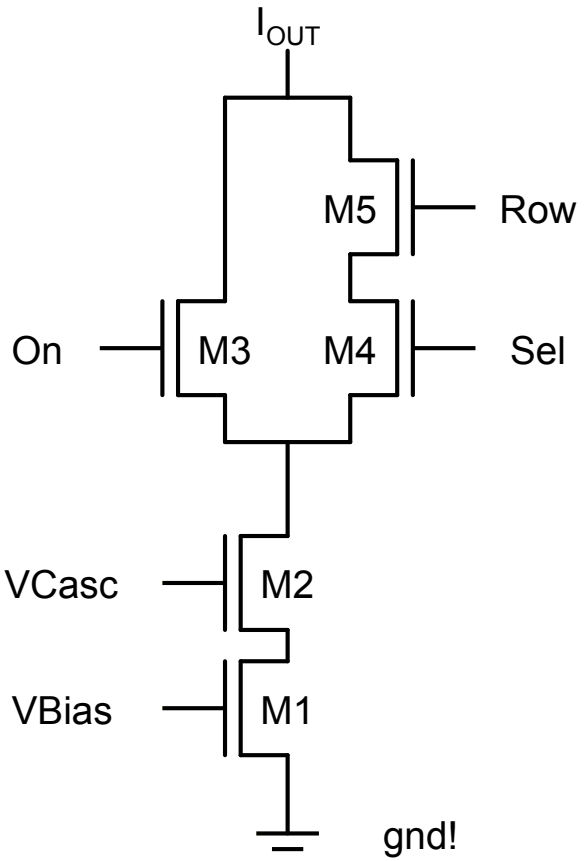
- Row:
  - Is a thermometer code generated from D⟨2:0⟩
  - Is generated in a Y-decoder
  - 8 signals run horizontally

- ColOn / ColSel
  - Are generated from D⟨5:3⟩
  - Also thermometer code from X-decoder
  - ColSel ⟨i⟩ = ColOn ⟨i-1⟩ !!!

D⟨5:3⟩	0	1	2	3
ColSel ⟨0⟩	1	X	X	X
ColOn ⟨0⟩	0	1	1	1
ColSel ⟨1⟩	0	1	X	X
ColOn ⟨1⟩	0	0	1	1
ColSel ⟨2⟩	0	0	1	X
ColOn ⟨2⟩	0	0	0	1

The Unit Cell

- Schematic:
  - M1 is a (long) current source
  - M2 is a (short) cascode
  - M3-M5 are (minimal) switches
  - Connect all bulks to gnd!
  - (In the layout, you may want to split M1 in two series connected devices to have more freedom to arrange the MOS. You then have to modify the schematic accordingly.)



Exercise DAC_A: The Unit Cell

- Create schematic & symbol of the unit cell
  - Use for instance L=1µ for the current source, minimal L for all other MOS and W=0.44µm for all MOS
- Generate a second, 'Bias' cell, which implements a current mirror so that the current source MOS generates 1µA in saturation
  - You can also re-use to UnitCell by connecting it correctly
- Simulate both cells
  - Put the voltage sources, stimuli,... In a separate schematic *SIM_UnitCell*! Do not put sources in the cells you use later!
  - Use a supply voltage of VDD=1.8V
  - Set VCasc to 1.8V to start with
  - Connect I_{OUT} to VDD
  - Verify that you can turn the current on and off with the 3 control signals

Exercise DAC_B: Finding the Cascode Voltage

1. When the cell is enabled, sweep the Cascode voltage
   - Find the minimal voltage before the output current drops because M1 gets out of saturation
2. Sweep the output voltage and find out down to which voltage the DAC works.
   - What is the current change ?
   - For a full scale current of 63 such cells: Will the change in current due to finite output conductance be below the current for 1 bit (the Least Significant Bit = LSB current)?
3. The current mirror only matches well, if the same cascode is also included in the *BiasCell*.
   - In order to make sure that any changes in size of M1 or M2 are consistent in the *UnitCell* and the *BiasCell*, use a new symbol & schematic for M1 + M2, and use this in *UnitCell* and *BiasCell* (or use *UnitCell* in the *BiasCell*)

The Thermometer Decoder

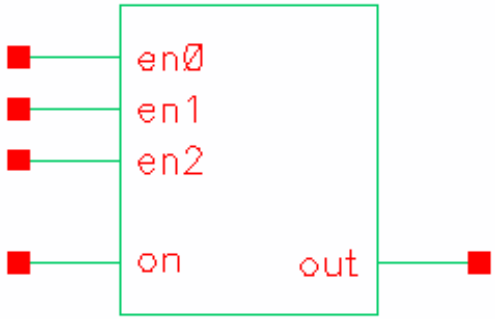
- The thermometer decoder does the following:

D	Out⟨0⟩	Out⟨1⟩	Out⟨2⟩	Out⟨3⟩	Out⟨4⟩	Out⟨5⟩	Out⟨6⟩	Out⟨7⟩
0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0
4	1	1	1	1	0	0	0	0
5	1	1	1	1	1	0	0	0
6	1	1	1	1	1	1	0	0
7	1	1	1	1	1	1	1	0

- We use a simple, systematic (but slow) implementation:
  - Out ⟨6⟩ = (D==7)
  - Out ⟨5⟩ = (D==6) | Out ⟨6⟩
  - **Out ⟨4⟩ = (D==5) | Out ⟨5⟩**
  - Out ⟨i-1⟩ = (D==i) | Out ⟨i⟩
- Note that one signal propagates from stage to stage

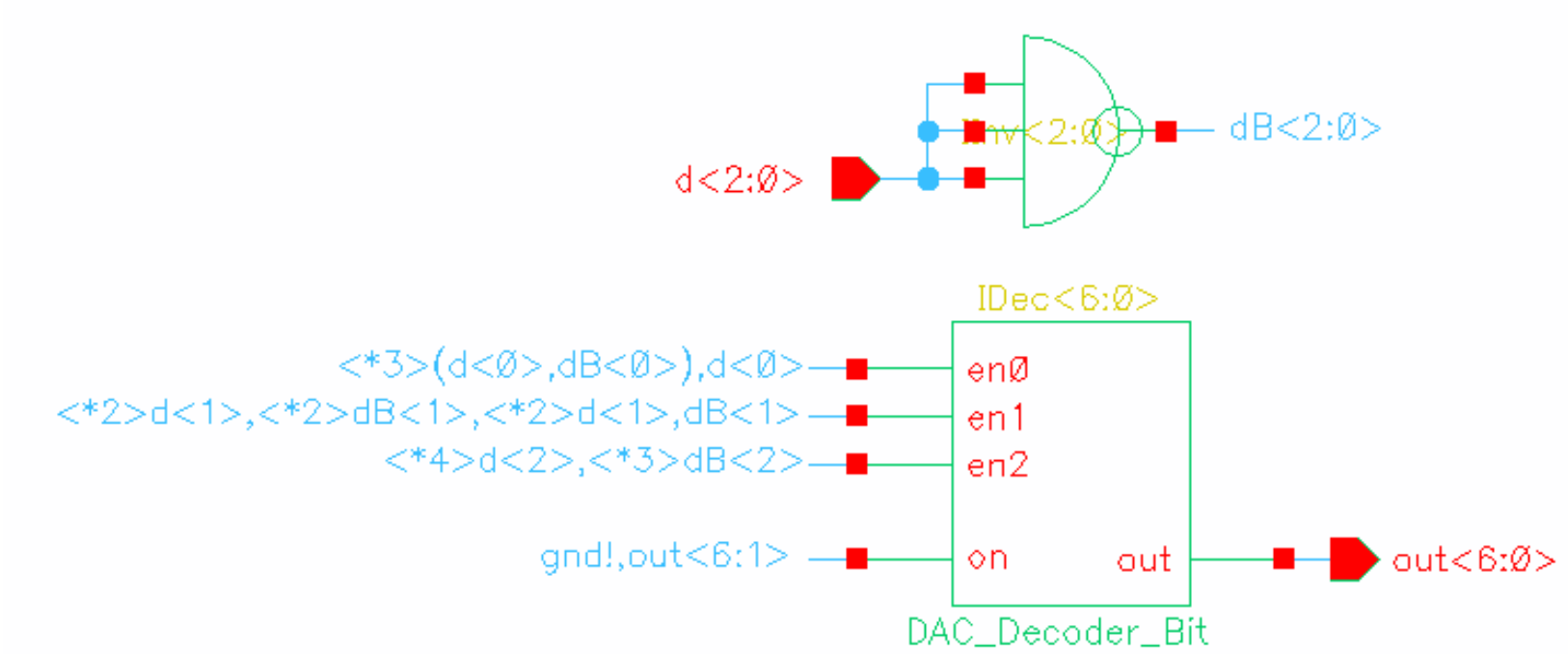
Exercise DAC_C: The *Decoder_Bit* Cell

- The OR2 function in the equations on the previous page cannot be implemented directly. Convert the equation so that we can use a NAND2.
- Create a cell *Decoder_Bit* for one bit of the decoder
  - To propagate the Out-Signal from stage to stage, you need an inverter
  - The comparison of D with 1,2,3... can be done with a 3 input NAND gate. You will need to invert some input signals (different in each stage).
  - Overall, you need a NAND3, a NAND2 and an INV. To reduce effort, you can only use a NAND3 (and connect 2 or 3 inputs to get NAND2 and INV).
- As a hint, here is my symbol of *Decoder_Bit*:



Exercise DAC_D: The Decoder

- Now implement the full decoder using the *Decoder_Bit* cell
- Note that you only need to produce 7 bits!
- As a hint, here is my schematic of *Decoder*:



Exercise DAC_D: Simulating the Decoder

- Create Symbol/Schematic *DAC_Stimulus* of a cell which generates increasing 6 bit binary numbers
  - Use 6 vpulse sources with increasing pulse widths / frequencies
  - Use 100ns per value, for instance
  - It is a goof idea to use a design variable for the unit time
- Simulate the Decoder for all 8 possible input values
  - Create a new simulation schematic also for this!
  - Use 3 bits of your *DAC_Stimulus*

Exercise DAC_E: A Column

- Create a Symbol/Schematic *DAC_Column_8* which contains 8 *UnitCell*s.
- This cell has
  - Analogue inputs *VBias* and *VCasc* and output *IOut*
  - Control inputs *ColOn* and *ColSel*
  - A bus of 8 control inputs *Row⟨7:0⟩*
- Connect the thermometer decoder and simulate the cell.
  - The current should increase in equal steps

Exercise DAC_F: The Matrix

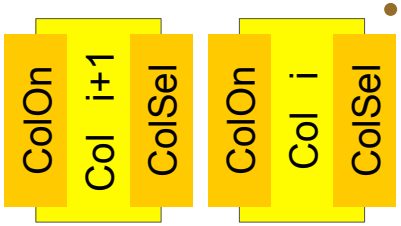
- Now make a Symbol/Schematic *DAC_Matrix_8x8* which contains 8 instances of *DAC_Column_8*.
  - When a column i is selected, the previous column (to the right) must receive its *ColOn* signal. We therefore only need *ColSel* signals for the columns. The last column does not need a *ColOn* signal, as it is never completely enabled.
  - This cell therefore has one more 8 bit control bus *ColSel⟨7:0⟩*
  - Connect the *ColOn* signals correctly internally!

- Connect two thermometer decoders

- Simulate everything using you *DAC_Stimulus* cell

Exercise DAC_G: Layout of the *UnitCell*

- Make a Layout of the *UnitCell* using the previous MOS dimensions
- Try to run *VBias, VCasc* and *Row* horizontally in Poly and all other signals vertically in M2.
  - If you run a vertical signal on the left and right cell edge, you can use the left one as *ColOn* and the right one as *ColSel*.
    If the next column is overlapped, the *ColOn* of the right cell will be automatically connected generated by the *ColSel* of the left cell, as required...



- Avoid a too asymmetric layout (high and small) so that the matrix is roughly square (or at most 2:1 aspect ratio)
  - You may want to split the current source *M1* in two serial connected devices
- If you are motivated, try several topologies....

Exercise DAC_H: The Matrix

- Assemble first the *Column_8* and then *Matrix_8_8*
  - Put a Bounding Box into the layout of *UnitCell* so that the cells just touch on the next hierarchy level.
  - The multiple *VBias* and *VCasc* nets are not connected (yet) in the layout, but they are in the schematic. You must therefore put net labels on ***all*** *VBias* and *VCasc* nodes.
  - When you make the *Matrix_8x8* cell, you can copy most labels from the existing *Column_8* cell.

Exercise DAC_I: Matrix Border

- Make a small cell which connects all Bias lines and add it to *Matrix_8_8*

Exercise 5J: Layout of the Decoder

- Make a compact layout of the NAND3. Consider that you have to connect 3 of these cells in *Decoder_Bit*. The three address bits and their inverses will have to run across and you must be able to connect them later.
- Make a layout of *Decoder_Bit* and of *Decoder* itself
- Connect two decoders to the matrix.
  - Use one or two fanouts to connect the 8 lines to the matrix