Neuromorphic Integrated Bioelectronics - Fall 2025

BENG 216, UC San Diego

Homework 4: Due October 31

This final homework serves as preparation for the final project. You will complete this homework with your final project group, working together as you will for the final project. The first part is a proposal summary for your final project, defining the objective with a problem statement in neuromorphic integrated electronics and a proposed solution *in silico*. The second part is an exercise in teamwork for scalable and modular design, which will come very handy in executing your final project with optimal efficiency in coordinating the various parts to fit together seamlessly, while all working on your individual parts in parallel. For both parts of the homework, submit a single copy of your joint work, as a group submission over Canvas.

1. *Final project proposal summary* [20 points].

   Here we ask each group to submit a final project title and a one paragraph description of your final project. Define a problem, of your choice, that calls for the use of neuromorphic integrated electronics towards an efficient solution implemented in a custom silicon integrated circuit. Ideas and examples can be drawn from the lectures and homework which included silicon neural arrays, silicon retina, silicon cochlea, etc. You are welcome to explore projects that call for hybridization with other technologies beyond the CMOS substrate available in the 130nm CMOS technology, although we encourage you to stay within the limits of the CMOS technology unless you have means to extend the CMOS fabication through hybridization with other technologies, *e.g.,* deposition of electrode materials or MEMS structures in the Calit2/QI nano3 facility on campus. We expect most groups to stay with simple "vanilla" CMOS for projects that use the available MOS transistors for neuromorphic computing and bioelectronic interface circuits.

   This part of the homework should be submitted over the following google sheet, so that we can coordinate the projects with everyone in the class:

   https://docs.google.com/spreadsheets/d/1KEuRBBQZg_TynUflUi4a9yl89XfouiDAUztnn5vYVUw/edit?usp=sharing

2. *Extended silicon retina with Gray address decoders and multiplexers for random-access readout* [50 points].

   Here we extend the design experiences in previous homework by completing the layout and LVS verification of the 4×4 contrast-sensitive silicon retina of Homework 2, now including all circuits needed at the periphery of the pixel array for row and column multiplexed random-access readout. To facilitate your projects with greatest efficiency and minimal overhead in teamwork, we implement the multiplexer using a scalable, modular design with linear arrays of address decoders and switches at the row and column periphery of the pixel array, with each decoder cell pitch-matched to the pixel cells in horizontal and vertical directions. We start by designing the address decoders in scalable and efficient manner.

(a) *Address decoder*: An address decoder is a digital circuit that takes an n-bit address $A_{n-1}A_{n-2}\ldots A_0$, and that produces $N = 2^n$ outputs of which a single one is selected to be active high ("1") as coded by the address, and every other one is low ("0"). An address decoder serves as the digital building block interfacing to switches to implement a multiplexer. The purpose of a multiplexer is to select one out of N inputs at the output, as coded by the address. For an analog multiplexer the inputs are analog variables, and one of these is brought out to the output.

An example 3-bit binary code for an address decoder with $N = 8$ outputs is shown in the table on the right. Here, the output in the rightmost column represents the index of the single active output (*i.e.,* 3 corresponds to binary outputs 00010000). Produce the equivalent of this table for $n = 2$ with $N = 4$, and for $n = 5$ with $N = 32$.

A_2	A_1	A_0	OUT
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

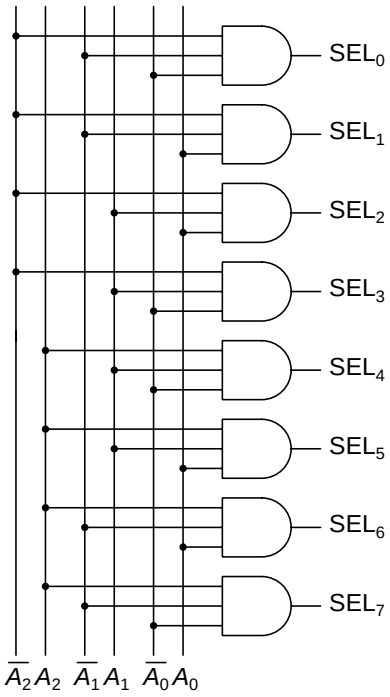
(b) *Gray code address decoder*: For greatest efficiency in multiplexing we will implement Gray rather than standard binary codes for the address decoders. In the Gray code, only a single bit in the address flips for every transition from one address code to one of its two neighbors. That minimizes power consumption during serial scanning, going through addresses in sequential order, since power in CMOS digital circuits is directly proportional to the number of bit flips per clock cycle.

An example 3-bit Gray code is shown on the right, implementing the same $N = 8$ address decoder as shown above, except with codes appearing in a different order such as to ensure single-bit transitions. Produce the equivalent of this table for for $n = 2$ with $N = 4$, and for $n = 5$ with $N = 32$.

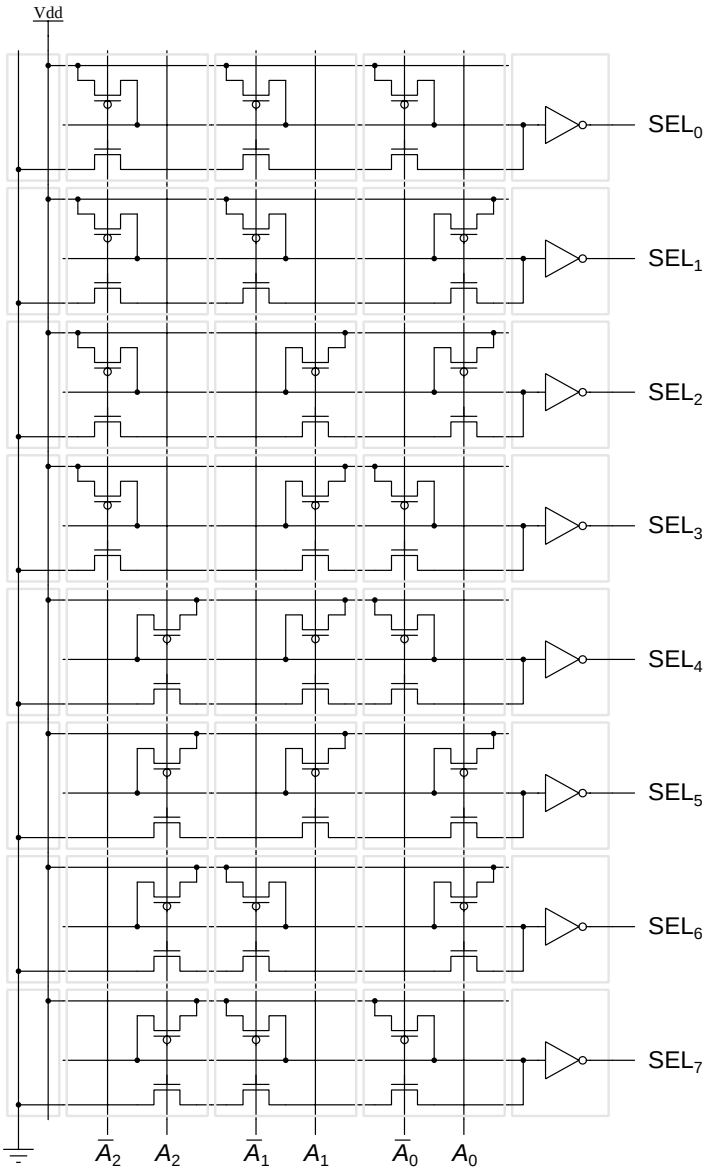
A_2	A_1	A_0	OUT
0	0	0	0
0	0	1	1
0	1	1	2
0	1	0	3
1	1	0	4
1	1	1	5
1	0	1	6
1	0	0	7

(c) *Scalable, modular CMOS address decoder*: Either the binary or Gray code address decoder is implemented by a bank of N n-input AND gates, each activating one of the outputs for a specific code configuration of the address bits, by tapping the corresponding lines with either the address bit A_j or its complement $\overline{A_j}$ at each of its inputs, as shown on the right.

For greatest simplicity and for almost[a] optimal results in operational efficiency in the CMOS circuit implementation of the address decoder, we adopt a scalable, modular architecture as an $n \times N$ array of two-transistor cells as shown below. Each CMOS cell in the decoder array contains one series nMOS transistor and one parallel pMOS transistor implementing one of n inputs in a distributed n-input NAND gate, followed by a single inverter.



[a] In principle, more optimal results may be obtained with a tree-based hierarchical design, especially for larger-size decoders ($n > 5$; $N > 32$) for which propagation delays of n-input nMOS series NAND chains become a performance limiting factor, which can be rectified by buffering intermediate stages in the NAND chain.

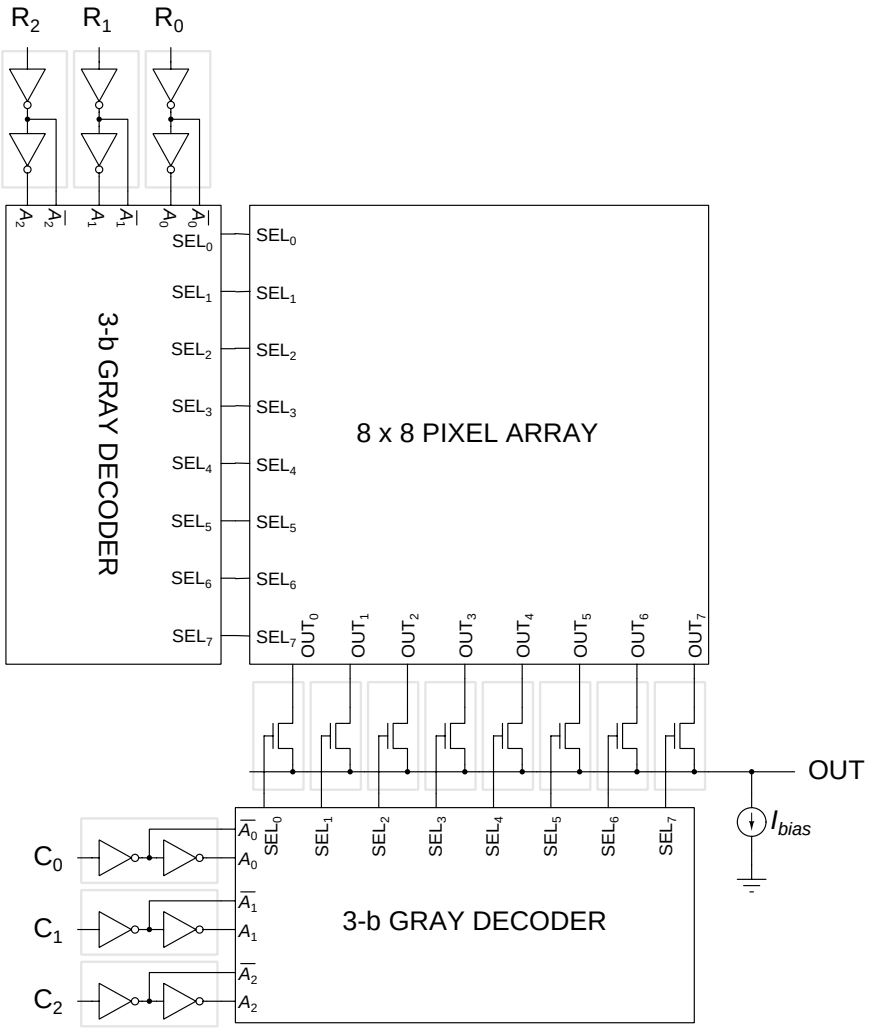


Notice in the diagram how the arrangement of the cells and the way they interface with each other is both *modular* and *scalable*: even though each output SEL_j has a unique code, its AND gate is realized using instances of the same few cells (denoted with light-shaded outlines in the diagram), with their inputs and outputs properly aligned along vertical and horizontal directions on a regular grid to coordinate the data flow. The example shown above is for a Gray code $n = 3$, $N = 8$ address decoder, but any (reasonable) size decoder can use exactly the same cells, just with a different number of instances in a $n \times N$ array, and on the same regular grid, without the need for any new layout beyond what is already within the cell instances.

Complete the layout and LVS verification of a Gray code $n = 2$, $N = 4$ address decoder as a 2×4 array of decoder cell instances. Consistent with the above diagram, your decoder cell layout should have mirror symmetry on the lines for the address A_j and its complement $\overline{A_j}$, so that inversion of polarity in the address selection by a decoder cell is accomplished by flipping the cell instance across its mirror axis. We expect your layout to have all "0" cell instances in the array flipped with respect to the "1" cell instances. You may use same standard nMOS 130nm

Skywater process using the `nfet_1v8` model for the nMOS transistors and the `pfet_1v8` model for the pMOS transistors in the `sky130_fd_pr_main` core library, and use short-length large-width device sizing (*e.g.,* $L = 180$ nm and $W = 1$ μm) for greatest bandwidth and least dynamic power of the digital logic implementation.

(d) *Complete* 4×4 *silicon retina with random-access, time-multiplexed readout*: Now we are ready to proceed with completing the 4×4 silicon retina design for Homework 2 with address decoders and readout circuits at the periphery. For time-multiplexed readout of the silicon retina 2-D signal in the pixel array, we implement a 2-dimensional analog multiplexer with row and column decoders selecting a single pixel in the array at any given time. The analog signal being read out is the buffered voltage output of the source follower inside the active pixel sensor, and the analog multiplexer brings out a single pixel's source followed output to the array periphery for random-access readout as selected by the row and column address decoders. Shown below is an 8×8 pixel array example, with 3-b Gray row ($R_2\, R_1\, R_0$) and column ($C_2\, C_1\, C_0$) decoders.



Complete the layout and LVS verification of the 4×4 silicon retina design for Homework 2 with random-access analog readout using Gray-code address selection and analog multiplexing of the source followed output of the selected pixel. As in the example shown above, include CMOS inverter instances pitch-matched at the array periphery to buffer the address bits A_j and produce their complements $\overline{A_j}$.

3. *A warm-up exercise in scalable, hierarchical, modular design and layout* [30 points].

   Extend your complete layout and LVS verification of your silicon retina design, including address decoders and readout circuits at the periphery, from the 4×4 pixel array of Problem 2 to a 32×32 array. For greatest consistency in results, and for your sanity, it is critical that you accomplish this transition in a single step without doing new layout or schematic entry, by doing no more than just instancing existing cells at the larger array size. As you will come to realize, it pays to adhere strictly to principles of scalable structured design in order to get the greatest results in efficiency, with the least amount of net design time, and effectiveness, with the least risk of human error and inconsistencies. See the supplementary 520.492 lecture notes at https://isn.ucsd.edu/courses/492/slides/week9.pdf for a tutorial on the fundamental principles of hierarchical modularity and scalability for structured and testable design.

Submission Guidelines: For this homework you are required to work with your final project team sharing configuration of the EDA tools from a single account, and submit one copy for your team. Singleton teams for the final project, if any, may join another team to complete this homework.

Fill in Part 1 of your homework on the class google sheet as directed above, and submit Part 2 and 3 as a single PDF over Canvas/Gradescope, one submission per group. Scanned handwritten notes are fine, and so are printouts of screenshots of the layout materials.