

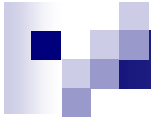
Circuit Design for MLC Flash:

Towards a semiconductor replacement for the hard drive

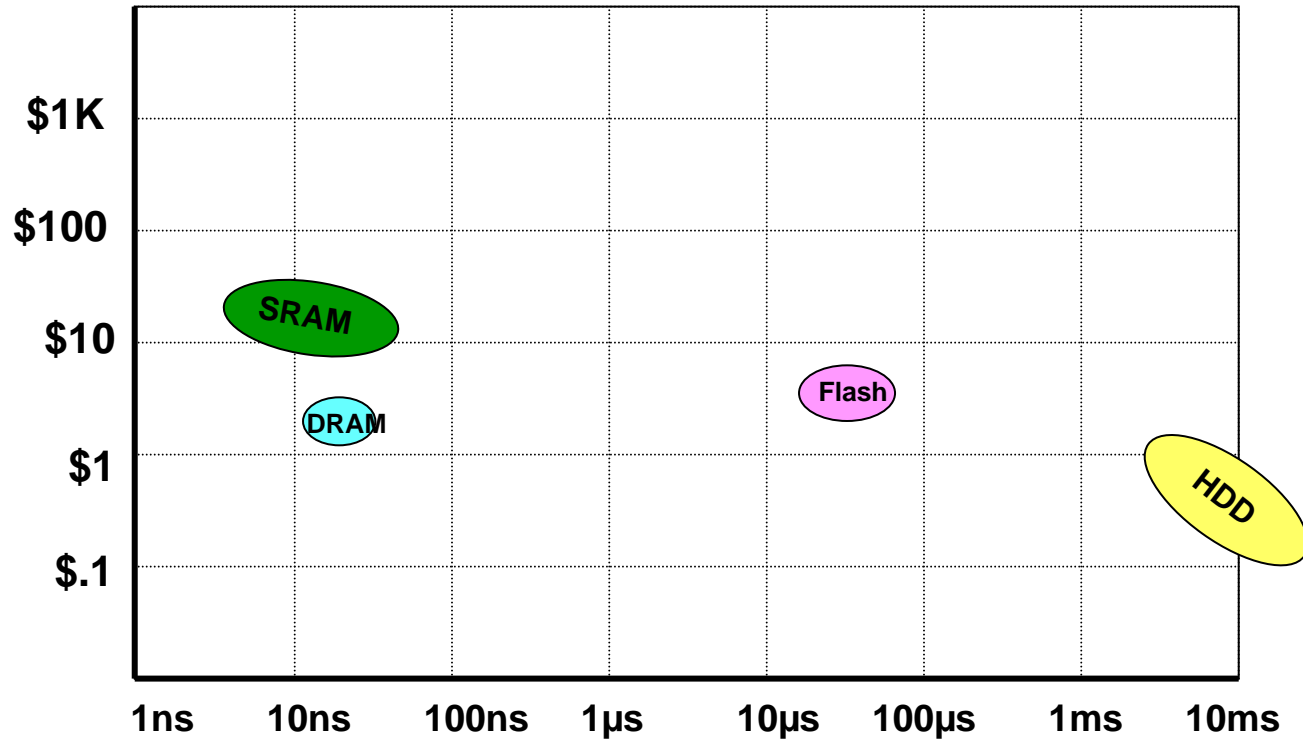
R. Jacob (Jake) Baker

<http://CMOSedu.com/jbaker/jbaker.htm>

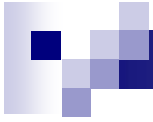
Abstract – Working towards dramatic increases in the density of Flash memory, this presentation addresses the practical design of sensing and programming circuits in a floating-gate (FG) CMOS technology using digital signal processing. The end-goal of the work is a higher-density, increased-reliability, memory using a Multi-Level Cell (MLC). A market overview is followed by a brief introduction to NAND Flash memory operation and the limitations inherent in increasing the density of Flash memory. Circuit design techniques are discussed. Simulation results are given along with suggested circuits and ways to minimize stress while increasing memory lifetime (both retention and endurance).



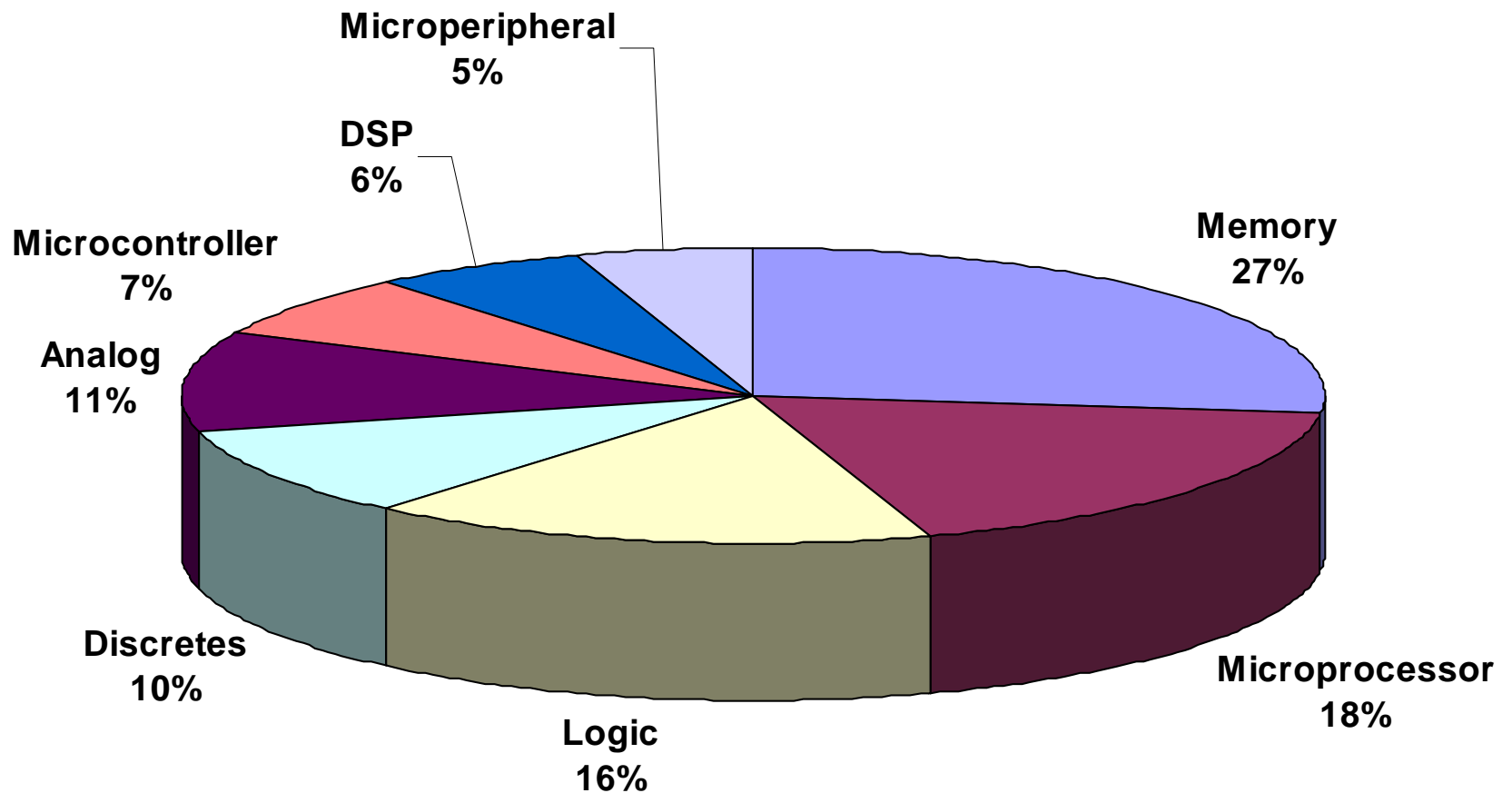
Memory Cost per Gbit



Access time (not burst rate)

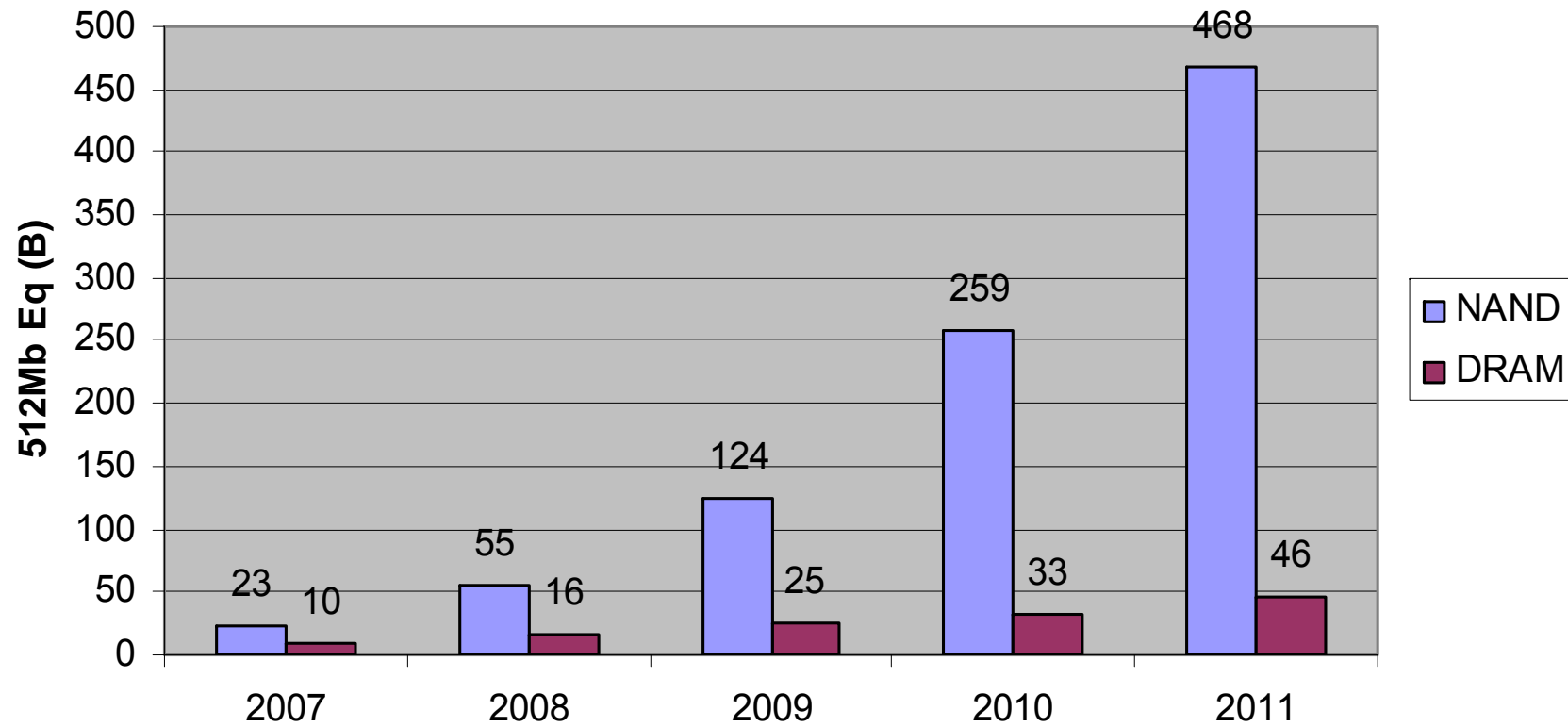


Semiconductor market shares by product

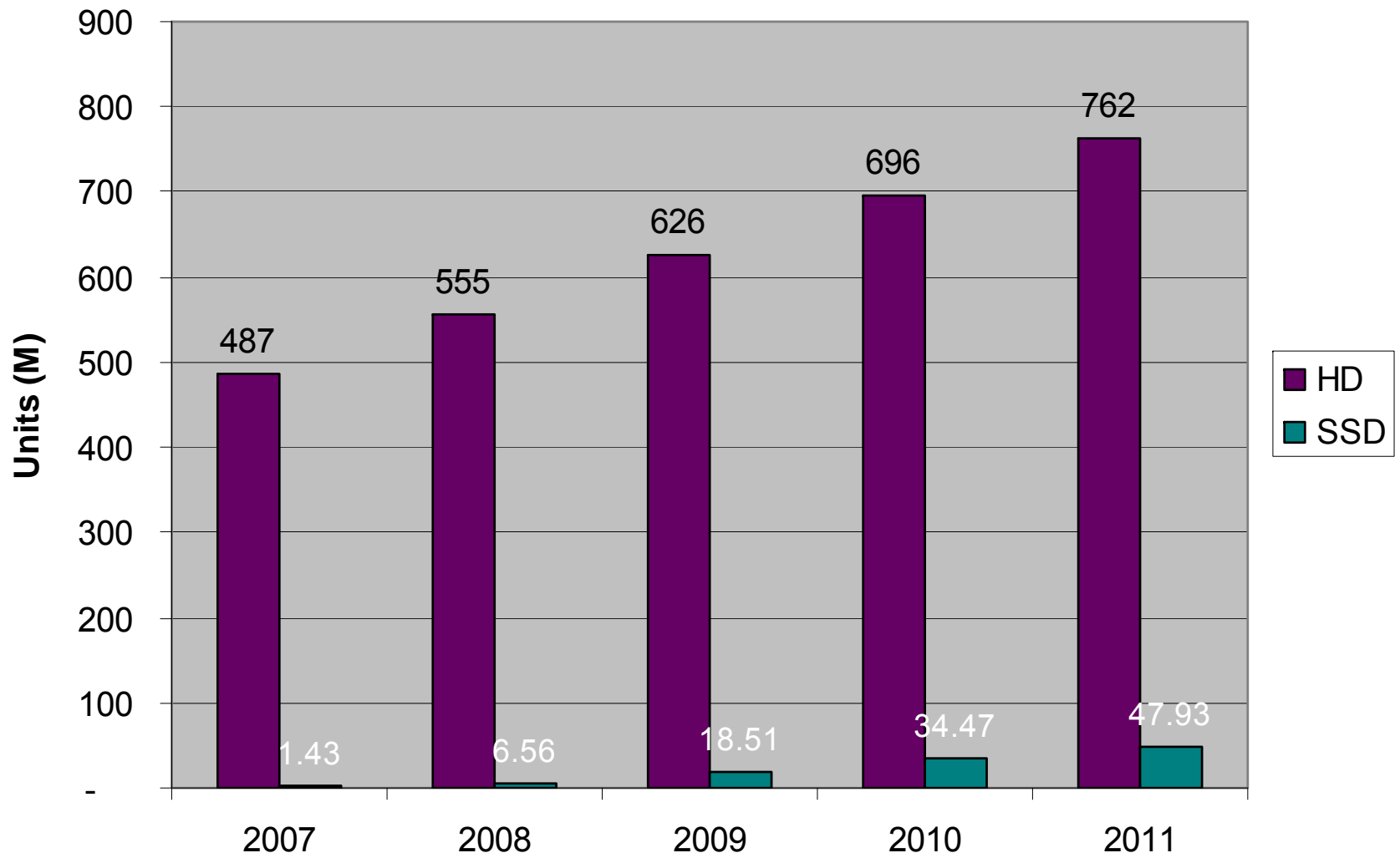


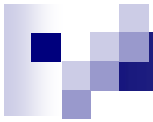


DRAM - NAND Trends

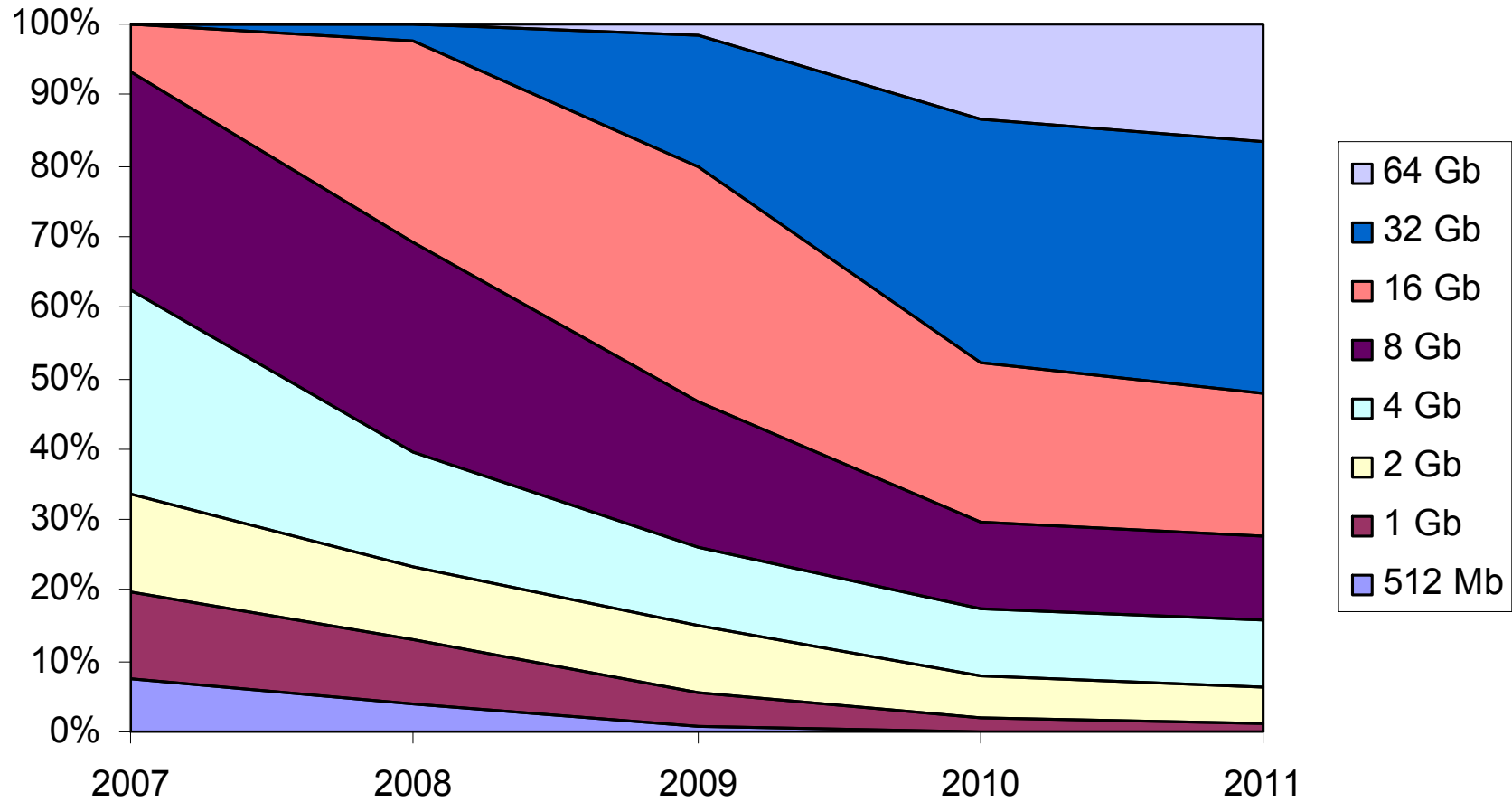


HDD- SSD Trends





NAND Density Transitions



What is Flash Memory?

- An MOS transistor with the addition of a Floating gate.
 - Used to shift the device's threshold voltage, the information we are storing.

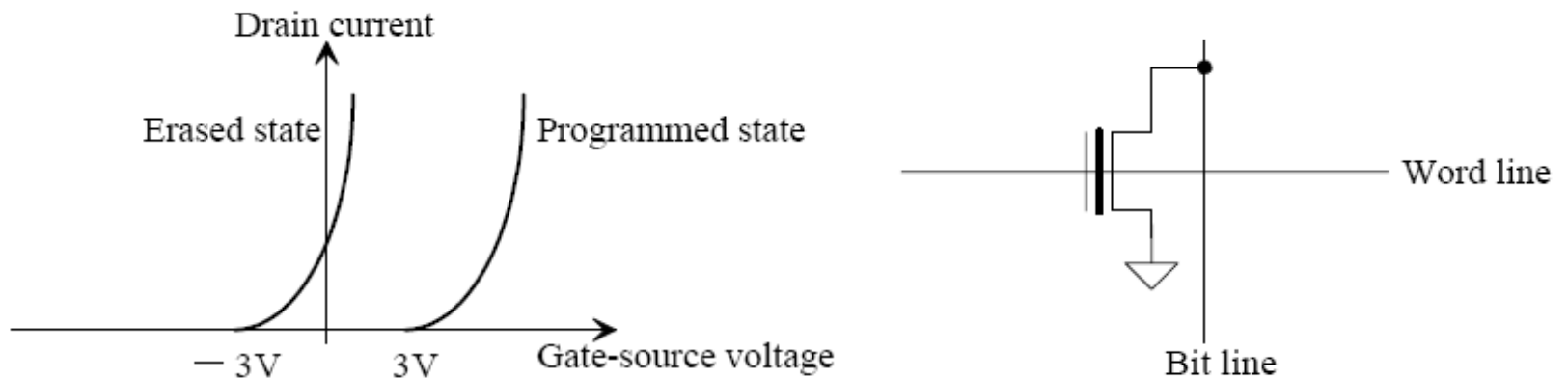


Figure 16.60 Programmed and erased states of a flash memory.

Programming and Erasing

- Devices can be placed in series to increase density
 - Hence the name NAND

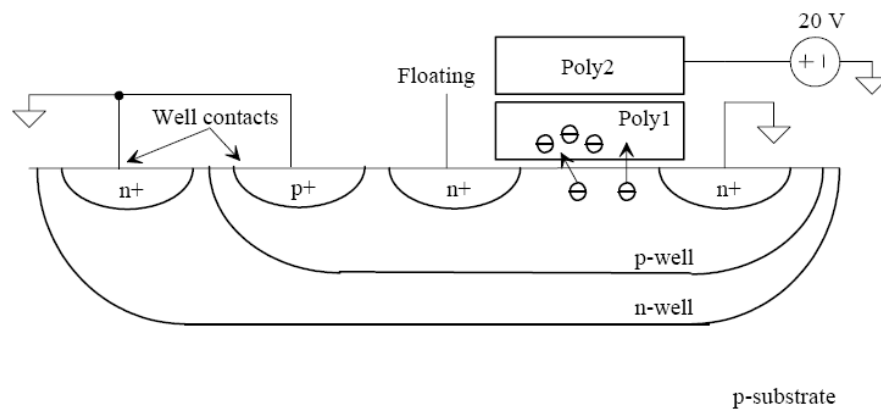


Figure 16.58 FNT of electrons from the p-well to a floating gate to increase threshold voltage (showing programming).

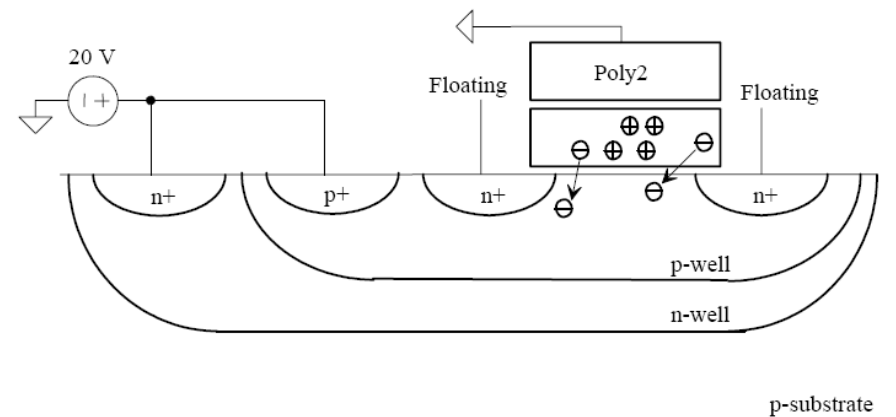
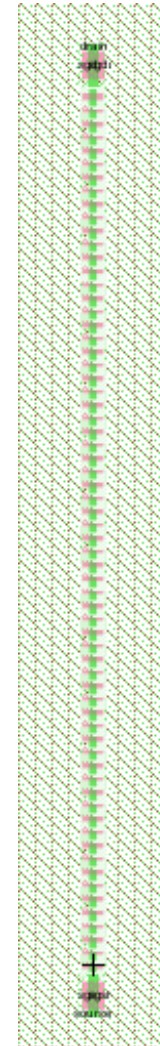
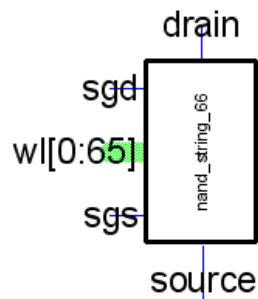
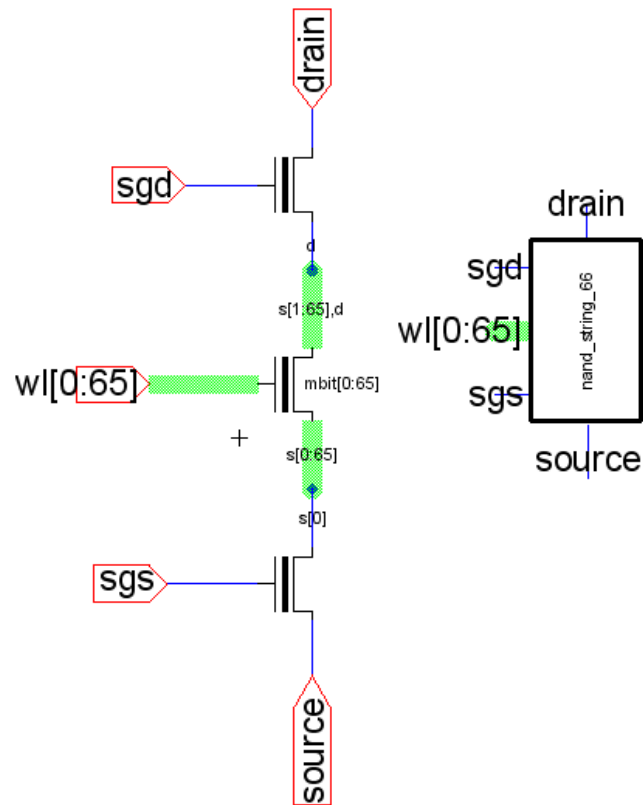


Figure 16.59 FNT of electrons from the floating gate to p-well to decrease threshold voltage (showing erasing).

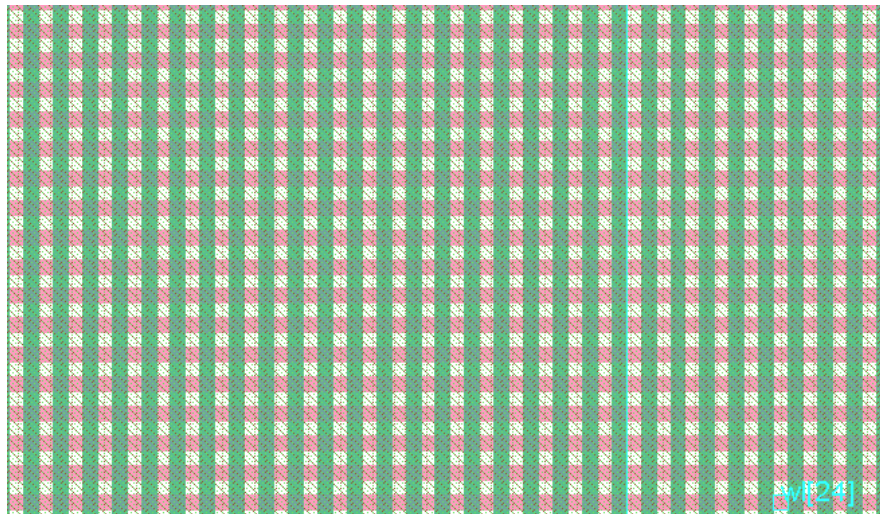
NAND string



Layout

Array Layout

- Flash memory density
 - Potential to store multi-level cells (good)
 - Retention and endurance (number of times the memory can be written) are issues



Approaches $4F^2$

NAND Parasitics

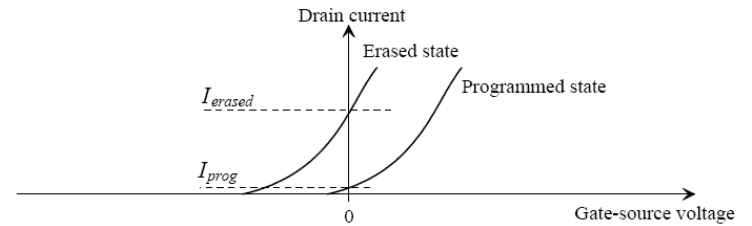
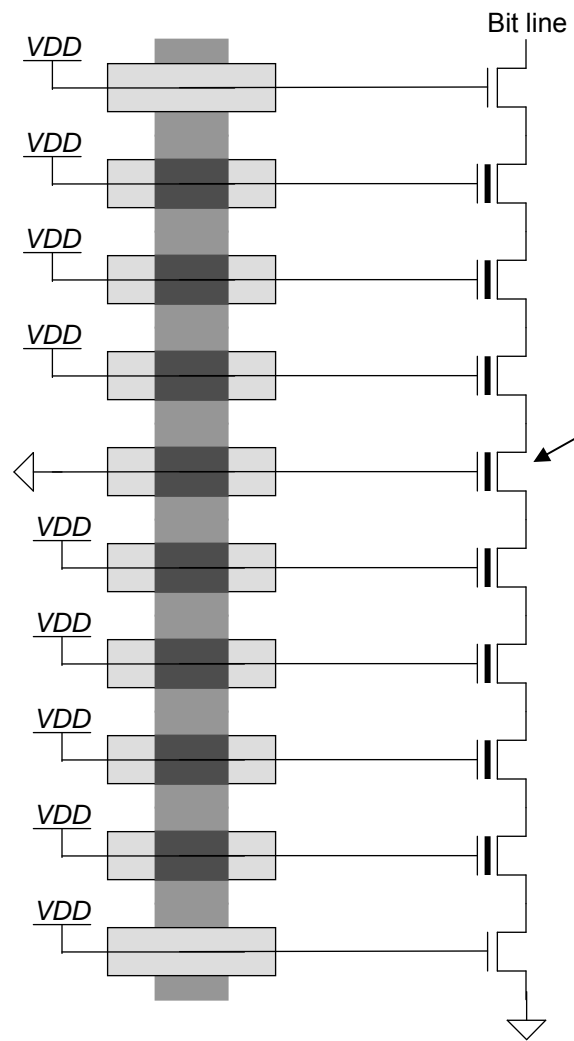
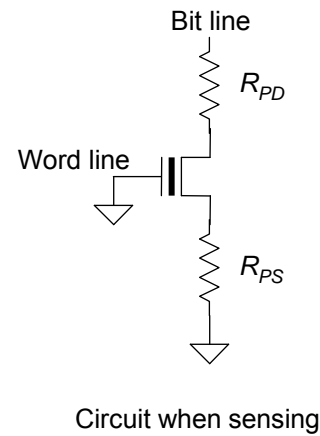


Figure 16.64 Expanded view showing erased and programmed IV curves.

Cell whose value we are sensing



Parasitic Resistances

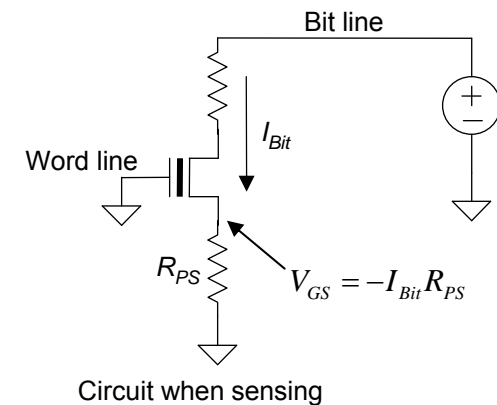
- By holding the bitline at a constant voltage the effects of R_{PD} are (mostly) eliminated
 - The scheme discussed here holds the bit line at an average voltage value. There is, however, a benign voltage ripple on the bitline.
- The source parasitic, R_{PS} , directly affects the MOSFET's gate-source voltage and thus the margins are also affected
 - Margins are limited by R_{PS}

Margin limitations



Varying R_{SD}

- Suppose $20 \text{ nA} \leq I_{Bit} \leq 1 \text{ } \mu\text{A}$ and that the maximum variation allowed in V_{GS} is 20 mV
 - Result is $R_{PS} < 20 \text{ k}\Omega$
 - This is a significant limitation!
 - If the on-resistance is 5 k Ω of a device then we can only use 4 devices in series
 - Note that the sensing circuit should have a resolution of 2 nA or so for good margins
 - Also note that this will require adjusting the programmed threshold voltages by <50 mV



Let's Fail

- This scheme will never work! Putting 64 FG MOSFETs in series will make, even with a perfect sensing circuit and no noise, storing 64 levels (5 bits) on each FG impossible because of the variability of R_{PS}
- Sensing with a resolution of 2 nA or changes in $V_{GS} < 20$ mV! No way!
- Programming with better than 50 mV resolution! Come-on! That's only 50 electrons change on the floating gate!

Let's Succeed

- If 64 FG MOSFETs in series is too many then use fewer, say 32 or 16
 - Note that using 2-bits (4 levels) with 32 FG devices gives a density of 64-bits/block of memory
 - Using 4-bits (16 levels) with 16 FG devices gives a density of 64-bits/block in almost half the space
 - Can we do better?

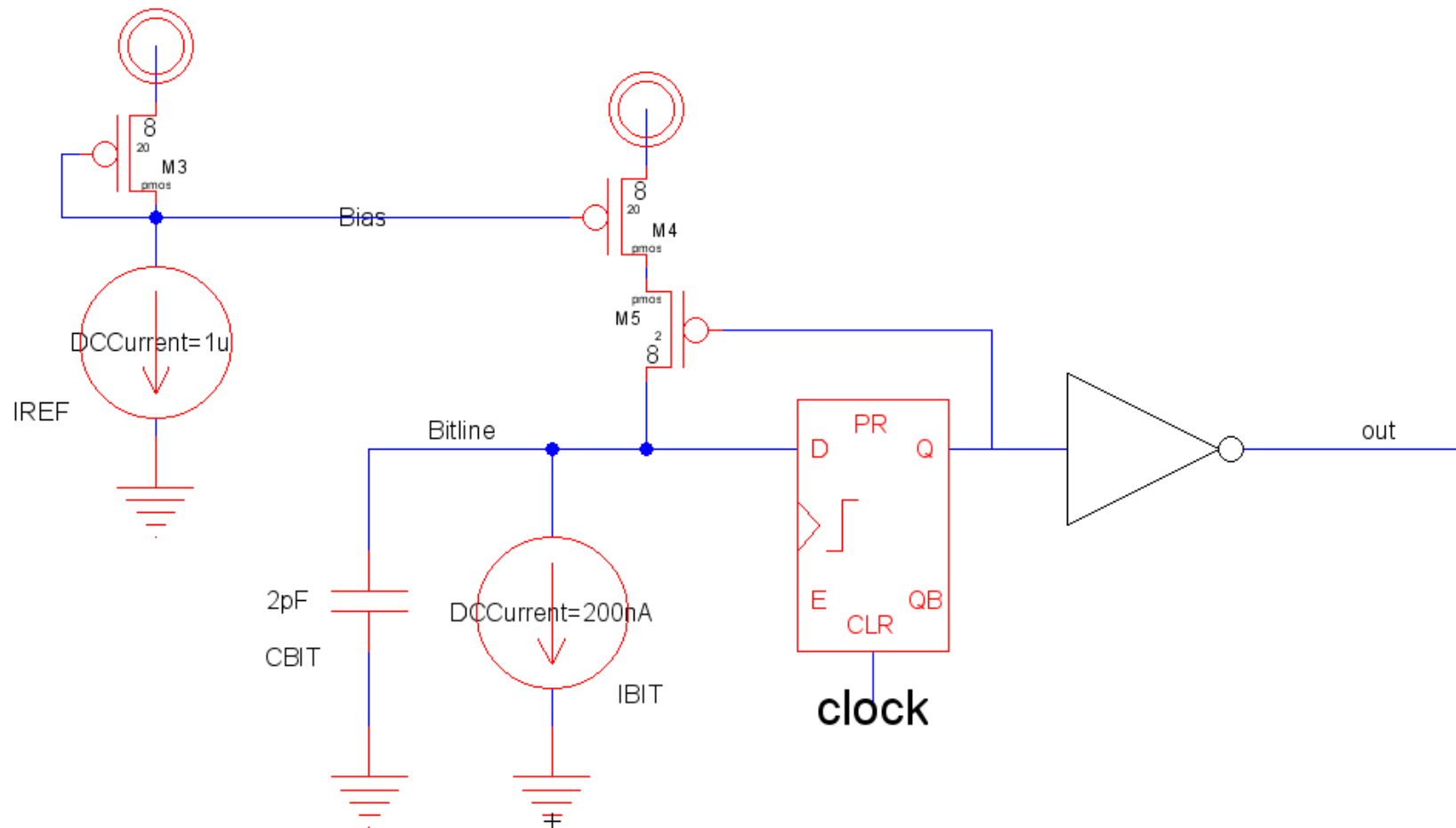
Let's Succeed – cont'd

- Use 64 FG MOSFETs in series but require cell be programmed from the bottom up
 - Makes the error associated with R_{PS} correlated and thus it subtracts out
- For significant increases in density, say 5- or 6-bits per cell, we need a topology that can evaluate what's possible
 - We can with what's proposed here!

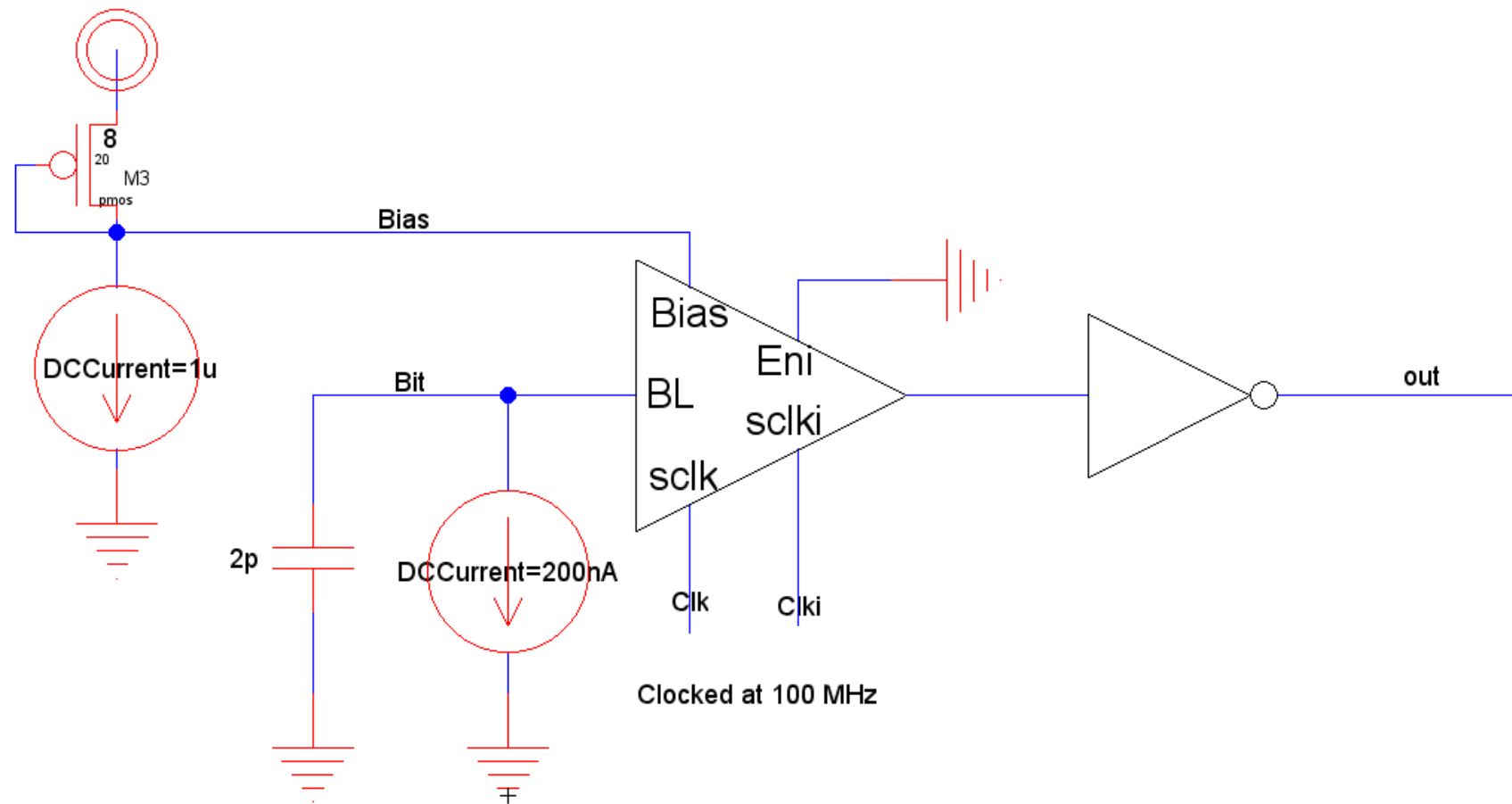
Let's Succeed – cont'd

- Is it practical to try and sense < 20 mV changes in a device's threshold voltage?
 - Absolutely not if using the current sensing circuits
 - Cell phones sense < 20 μ V analog signals. How?
 - Cell phones, or any sensitive measuring instrument, use narrowband filtering to remove noise from the signal
 - Here, we'll use a counter which behaves like a low pass filter to provide excellent sensitivity
 - The cost for going from an analog sense to a digital sense is larger sense-amp area
 - Increasing the sensing time causes the bandwidth to become narrower with the result of better sensitivity

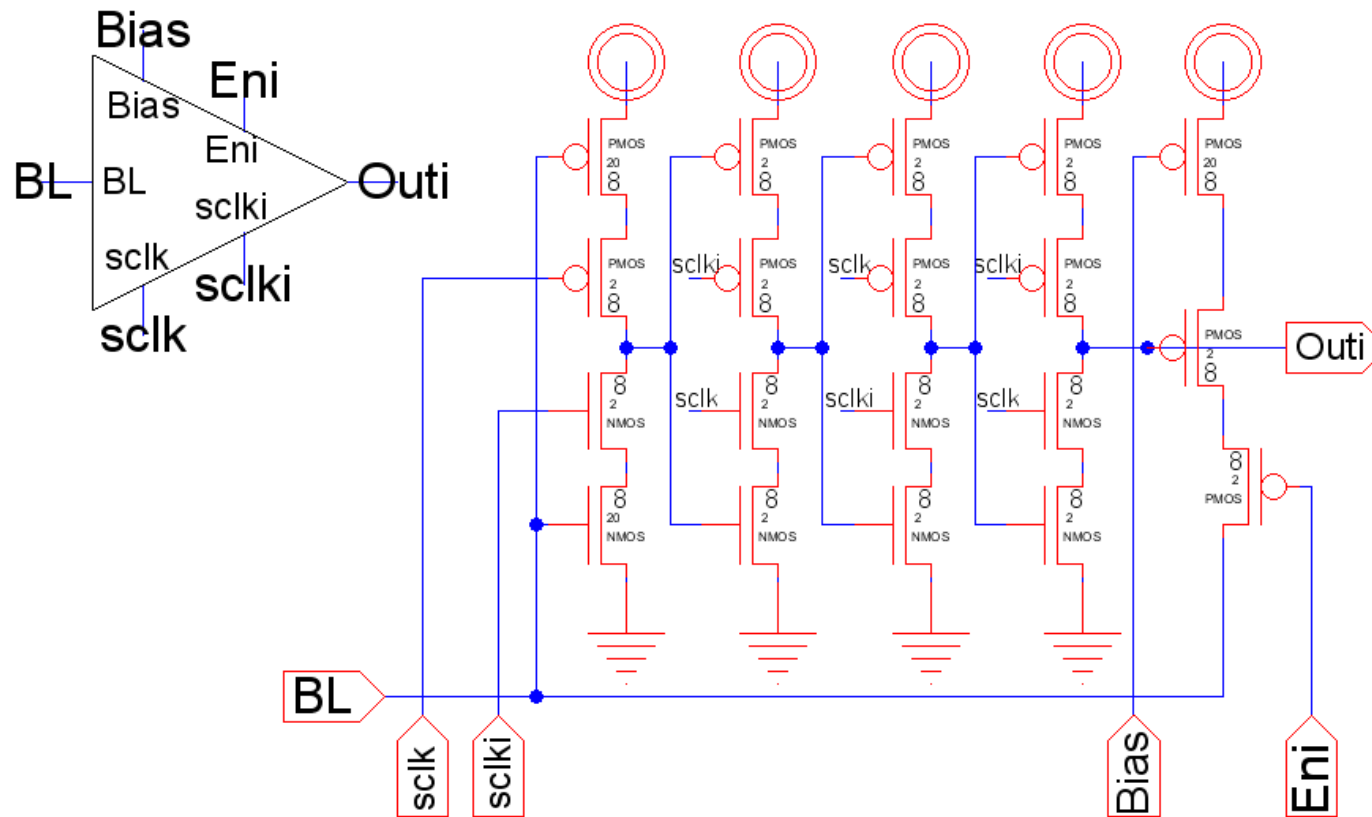
Sensing using Delta-Sigma Modulation – Basic Idea



Sensing using Delta-Sigma Modulation – Cont'd



Circuit Details



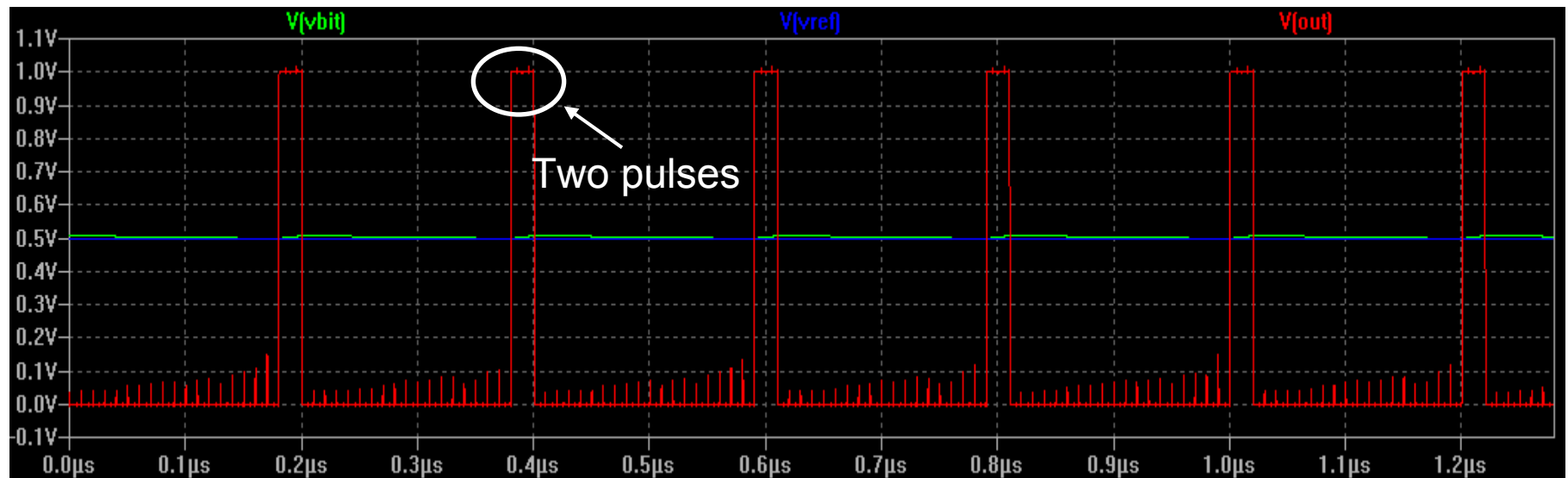
Operation

- Basic sensing circuitry consists of
 - A comparator (here two DFFs)
 - Metastability is a concern, hence why we use two DFFs
 - Bitline moves around switching point of DFFs
 - Three transistors
 - A counter (not shown in previous slide)
- The feedback action keeps the bitline at an essentially fixed voltage
- Output is a running average of the cell's current

$$\frac{I_{cell}}{I_{REF}} = \frac{\text{Number of times DSM output goes high}}{\text{Total number of times DSM is clocked}}$$

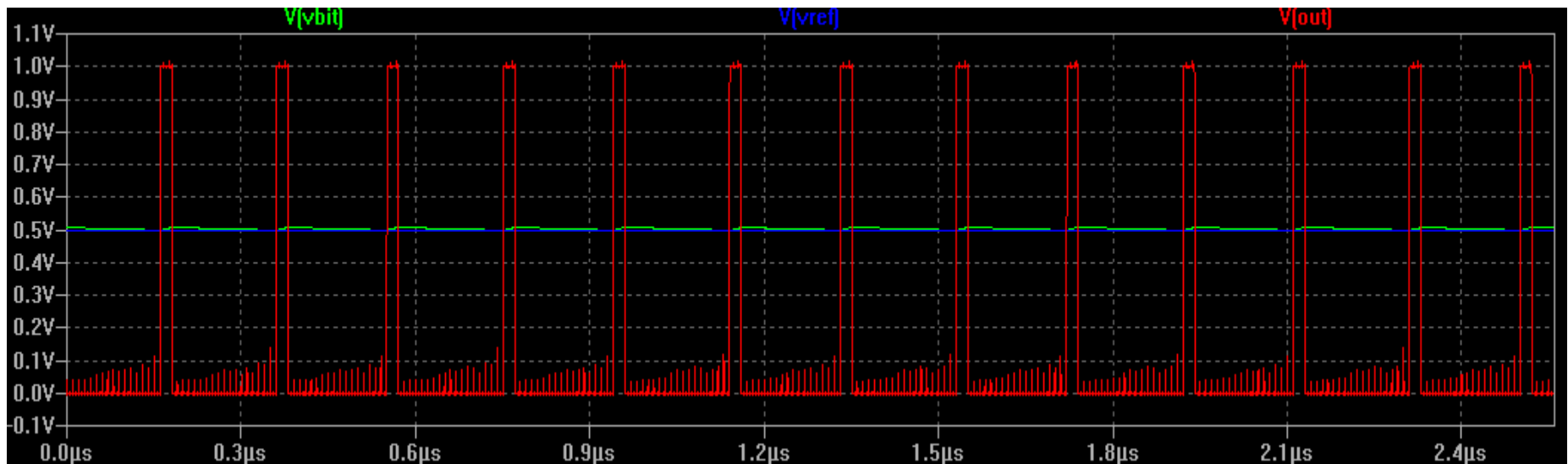
Examples

- Sensing with $I_{REF} = 1,000 \text{ nA}$ and $I_{cell} = 100 \text{ nA}$
 - We get 12 output highs out of 128 clock cycles
 - $100\text{nA}/1,000\text{nA} = 0.1$
 - Our sensing circuit gives $12/128 = 0.094$
- Note! The resolution of the sense is $1 \mu\text{A}/128 = 7.8 \text{ nA}$
 - Increasing sense time (number of times we clock the DSM) decreases the resolution of the sense (which is good!)



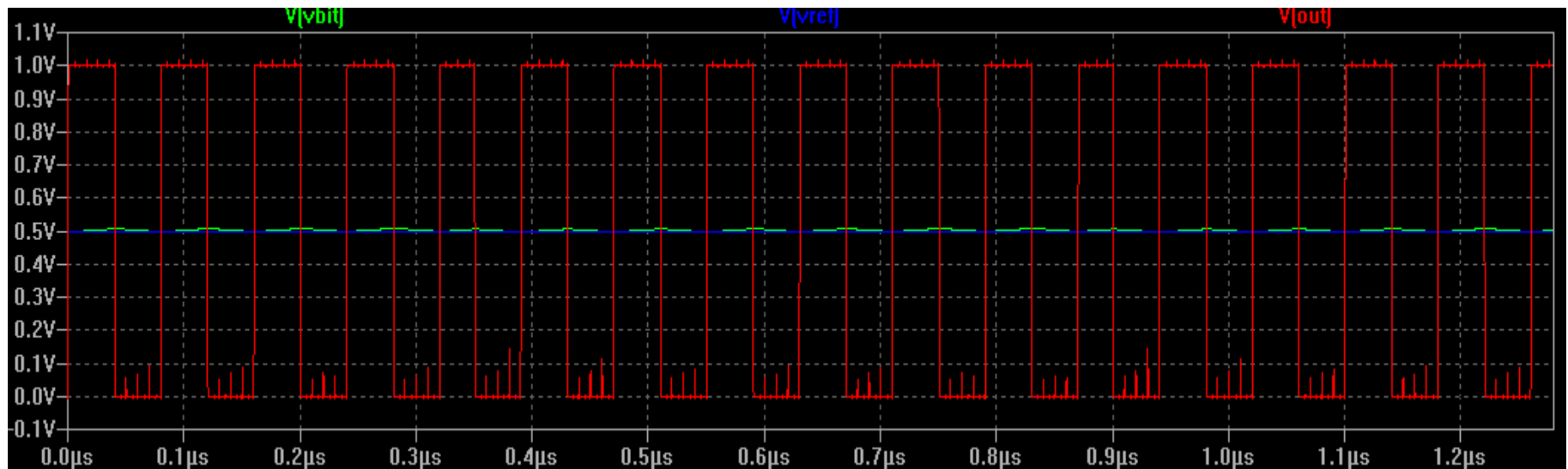
Resolution limitations

- If we increase I_{cell} to 105 nA, we get the same output data (for a 128 clock cycles)
- Doubling the sense time gives the data seen below
 - Now the output of the DSM goes high 26 times out of 256
 - $105/1,000 = 0.105$ is the signal value
 - $26/256 = 0.102$ is the sensed value
- Note how we can easily trade off time for SNR

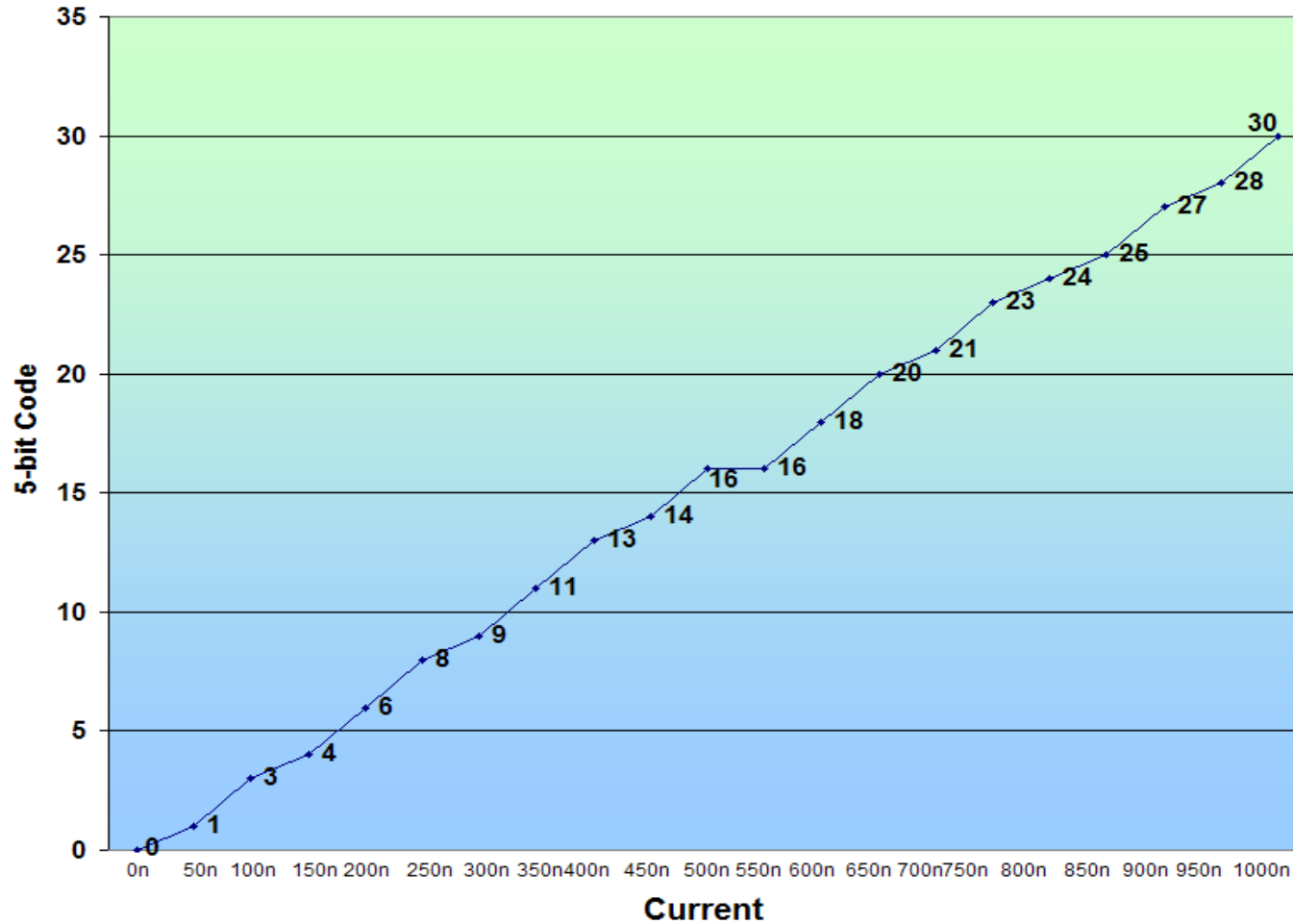


Another Example

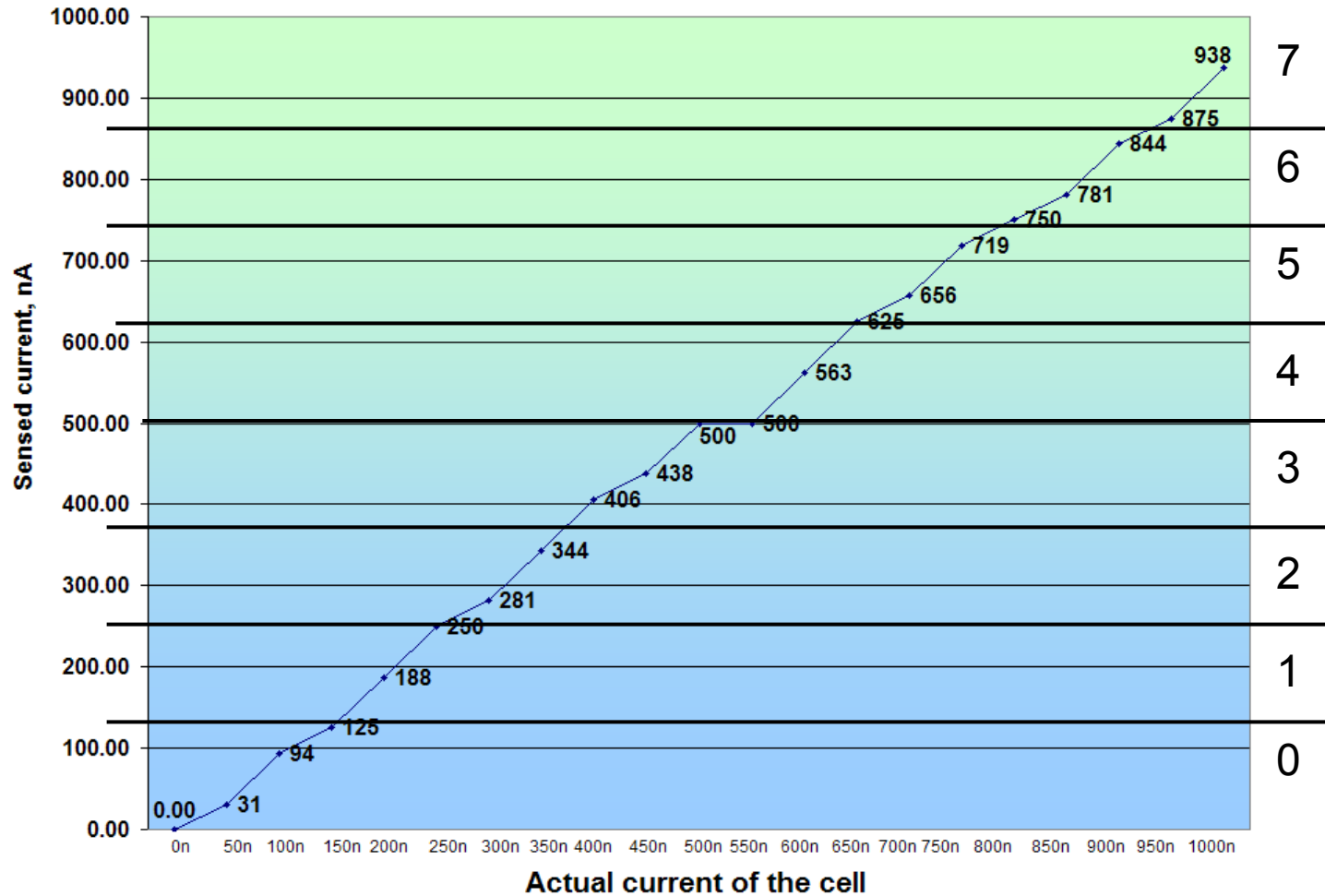
- If I_{cell} is increased to 500 nA we get output data that is essential half ones and half zeroes as seen below
- At this point, the basic idea should be understood



Using a 5-bit Counter (32 clock cycles or 320 ns sense time)



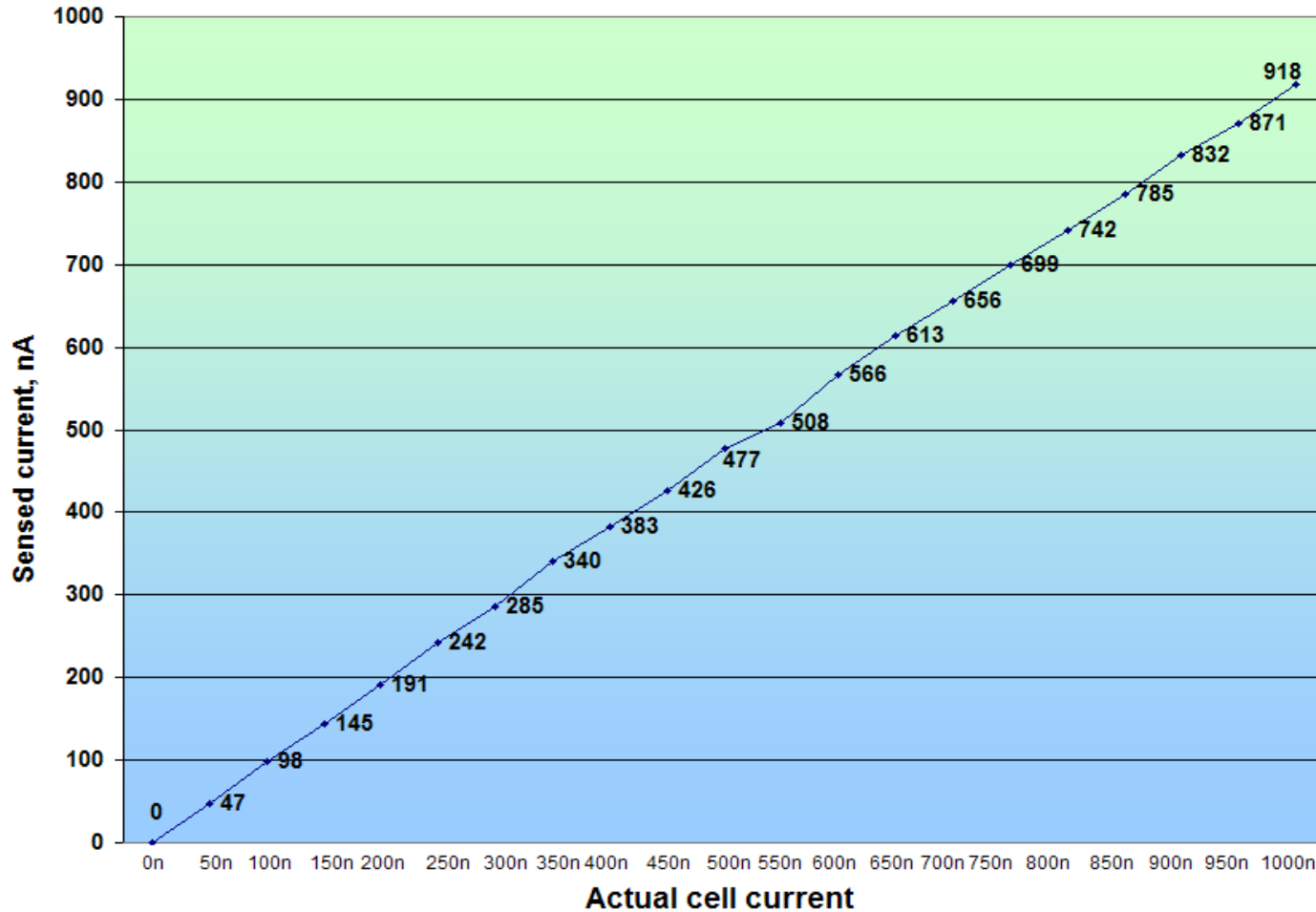
Again, using a 5-bit Counter (32 clock cycles or 320 ns sense time) with current on the y-axis



Comments on using a 5-bit counter

- Theoretical resolution is 31.25 nA ($=1,000\text{nA}/32$)
 - Practical values stored should be $\gg 50$ nA for safe margins
 - Should be able to get 3-bits (8-levels in MLC) where the resolution required is 125 nA (see the levels drawn on the previous slide)
 - Note that the actual value of the counter output is NOT important! Rather, the linearity of the sense is the important parameter.
- Big results follow big ideas
 - Let's see what happens with an 8-bit counter

Using an 8-bit Counter (256 clock cycles or 2,560 ns sense time)

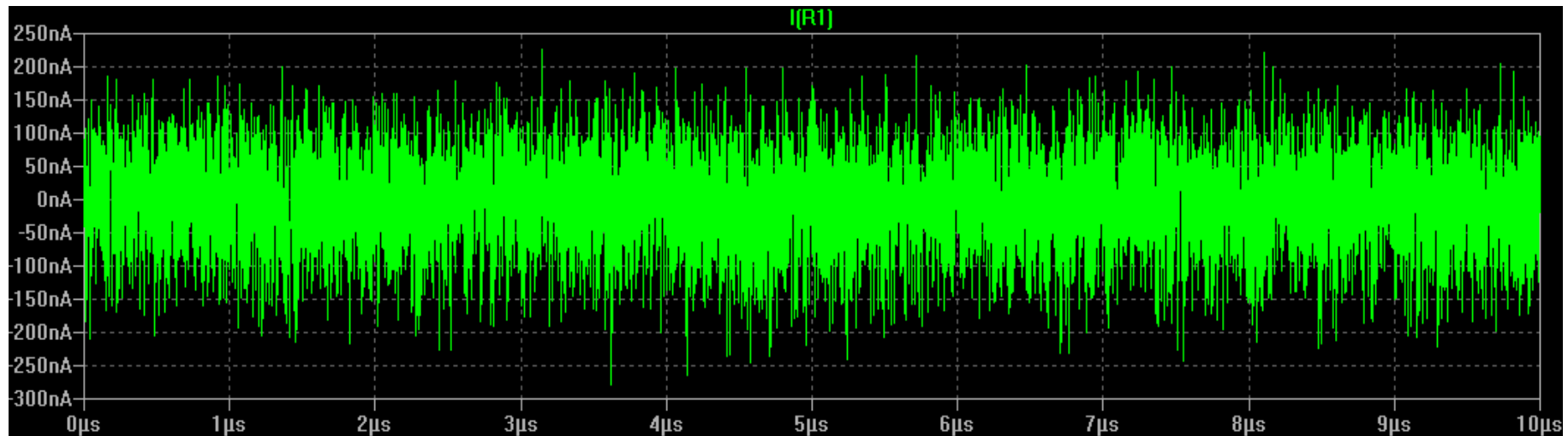


Comments on using an 8-bit counter with 2,560 ns sense time (256 clock pulses)

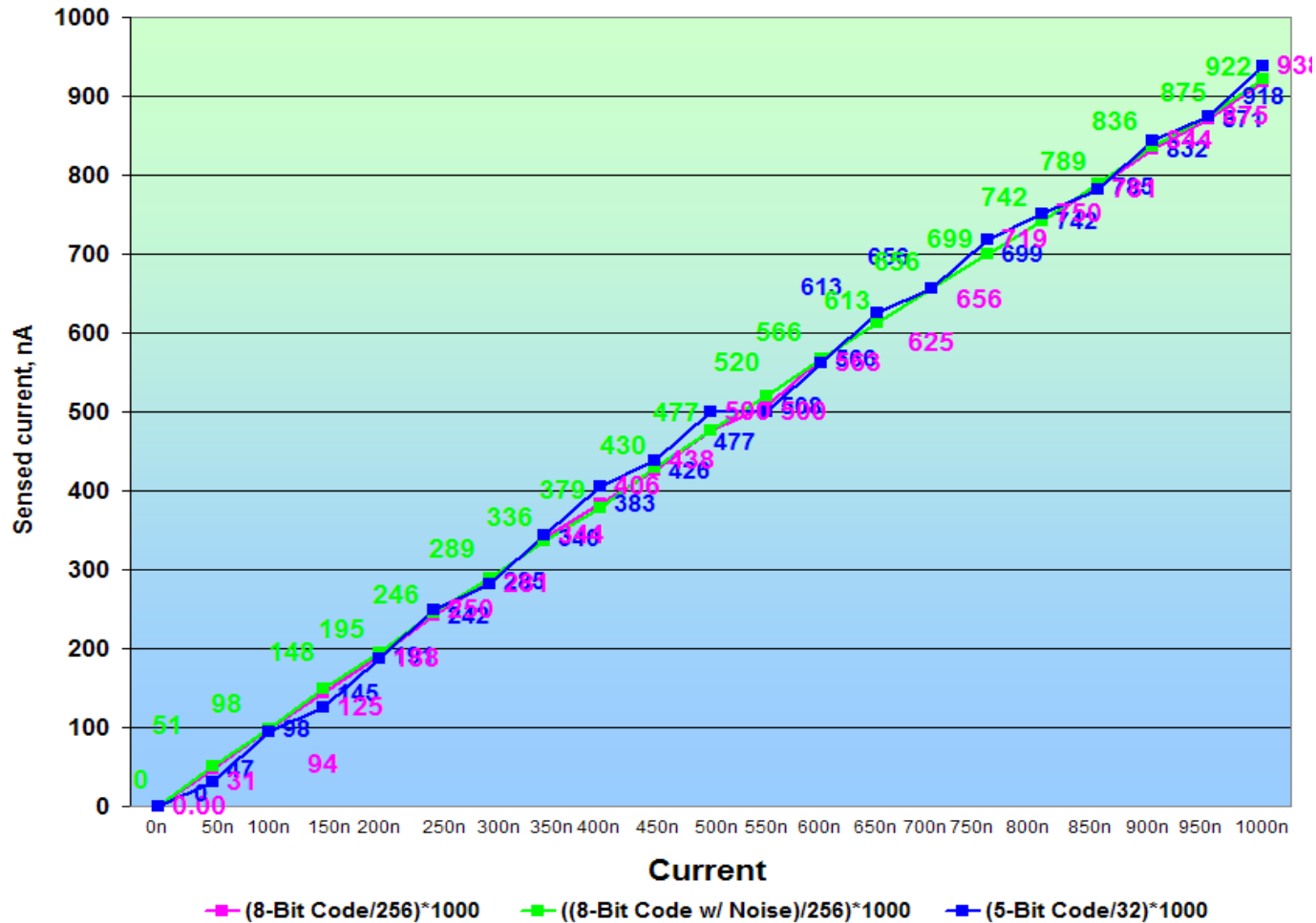
- Theoretical resolution is 3.9 nA (=1/256)
 - Practical resolution would be >10 nA for safe margins
 - Should be able to get to 5-bits (32-levels in MLC) where the resolution required is 31.25 nA
 - Again, the linearity of the sense is the important parameter.
 - This is all fine and good, but what about real life where the signals are obscured with noise?
 - Let's feel the power of DSP!

Adding Noise

- Let's add a white noise source with a 400 nA peak-to-peak amplitude.
 - Our first input current is 50 nA
 - $\text{SNR} < 1$



Comparison with noise (8-bit codes vary by, at most, 4 counts or 16 nA)



What happens if we need better SNR?

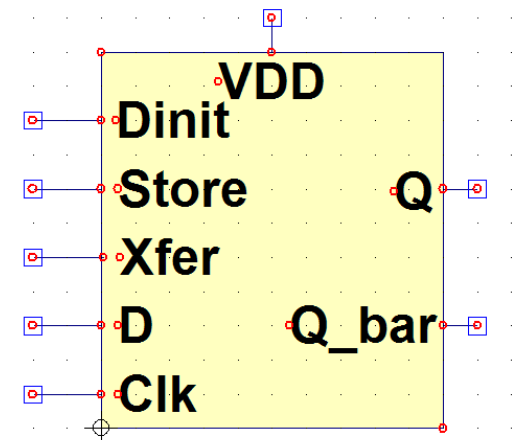
- Increase SNR by sensing longer
 - Decreases resolution (good)
 - Again, time can be traded-off for SNR for increasing the density

Summary

- A practical technique for increasing the storage density in FG Memory was presented
- Discussions and concerns were given
- Uses digital signal processing to increase SNR
- The following slides present a few more circuit design concerns and comments

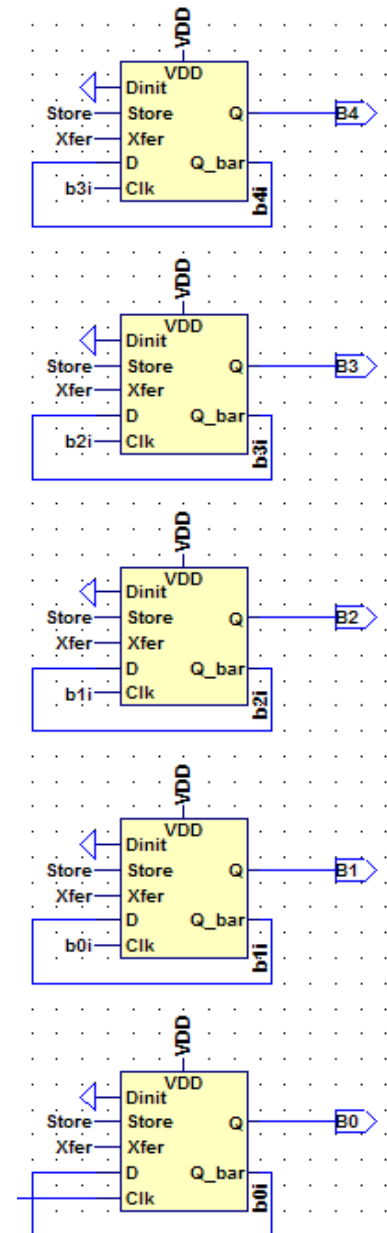
The Counter: Some Comments

- A simple D-FF can be used
 - Need to be able to store the desired programmed code
 - We store the value into the register by asserting the Store signal
 - Asserting the Xfer signal transfers the stored signal into the latch
 - With both Store and Xfer low, the circuit behaves like a D-FF and thus it can be used as part of a counter
 - Asserting the Xfer signal again transfers the stored value back into the D-FF (this operation is required for the program-verify operation)



The Counter, cont'd

- Can be used to center the stored value for the best noise tolerance
 - For example, if we write a code of 1011 0 (5 bit stored code) when programming we can say that the cell is programmed when the 8-bit counter code hits 1011 0100 (note there are many very trivial ways to implement these schemes)
 - When reading the data any counter output code ranging from 1011 0111 to 1011 0000 gives the same result (we have +/- 4 counts of noise)
 - Again, a larger counter (more time) can be used if we find that we aren't getting a high enough SNR for a given density



Programming (Program/Verify PV)

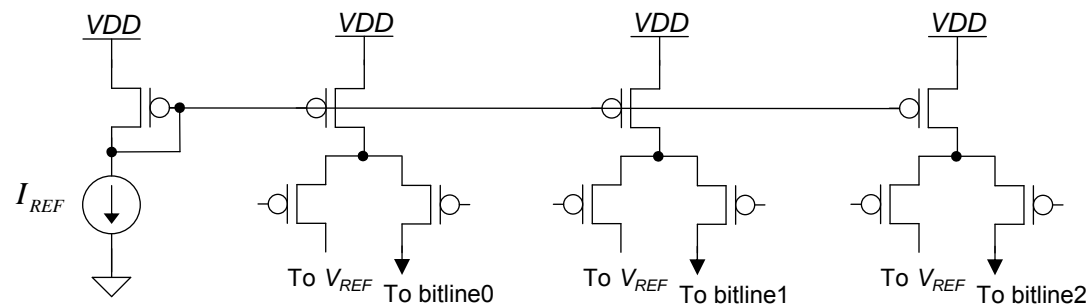
- We'll have to ramp up the wordline in very small increments during programming to achieve a large number of stored levels
 - Again, we can set the counter up so that when the MSB goes high or low, it signifies that the cell is programmed and then holds the bitline high to avoid additional FNT during the remaining program times (again, there are other very simple ways to detect that the cell has been programmed)
 - Performing the PV in this slower fashion, and thus more precisely, may result in an increase in the number of times that we can write to the cell
 - This may also allow for thinner oxides and lower voltages to be used (I would suspect there will be new technology developments if the programming is less aggressive)

Programming times

- It may take several ms to write a page of data for very high density designs
- If we go with 4-bits/cell (16 levels) and 32k of column lines then when we write 16kB (say a page) in a ms is this too slow?!
 - At 100 MHz writing 32k, 4-bit words, takes 320 us or 0.32 ms
 - Memory can be segmented into banks so that while we are writing in one bank we are reading or writing in one or more other banks
- Note that the way we designed the counter we use it for the storage of the 32k, 4-bit words
 - The DSM and counter can be set at the bottom (and/or top of the array) and staggered with a data bus running over the tops of the counters (layout and control should be straightforward)
 - The column address decoder then enables a specific column to control the bus (put data on or take data off)
 - Note that our previous schematic of the counter bit with storage didn't show the tristate connection to the bus or the address enable input (again, all very straightforward to implement)

Some Circuit Design Comments

- Reference current external to the array can be implemented in many ways
 - FG MOSFET
 - Self-biased reference
 - It's important that the reference current tracks the memory cells' currents with temperature



The Current Source, cont'd

- Current sources are DC circuits
 - Matching is important, must use large area devices (this is critically important in nanometer CMOS processes)
 - Read previous bullet again because using minimum length or width devices will result in failure
 - Currents don't need to be exactly the same (match) but they can't be zero or some small value
- A single switch can be used instead of two switches
 - Reduces power
 - Simpler control from comparator
 - The problem is the voltage across the current source changes (gives rise to a data dependent error which may be fine for all but really high densities)
 - As seen on the previous slide we can steer the current to a reference voltage to keep the voltage across the current source constant (earlier we "faked it" by using a dummy MOSFET)
 - We can also use a cascode current source to minimize the effects of finite current source output resistance

An Alternative to the Current Source

- Replace current source with a switched-capacitor resistor
 - Won't track the cell's behavior with temperature
 - May not matter if the data is encoded prior to storage (the data changes are stored, not the actual data)
 - Even lower power
 - No bias source needed
 - Very simple
- Note that if the bitline is shorted to a voltage source, we lose the “sigma” (the sense is reset)
 - The DSM sensing circuit should be clocked for some time before enabling the counter (important)