Project Report EE313, Winter 2009/10

Names: Laura Sharpless (ljs62)
Karthika Periyathambi (karthipd)

Design Objectives: To reduce the power consumption of the system and to increase frequency to 1.25 GHz without adversely affecting the power consumption**.**

1. **Summary of what ideas you explored. Please list the parts of the memory that you worked on for the project, and what you tried to do.**

- **Decoder:**
  - *Redesigned the decoder for 7:128 outputs (divided wordlines) using 3:8 predecoder*
  - *Decreased power by 7.5 times by optimizing delay for decrease in energy (marginal costs matching)*
  - *Used post-charge logic from SAE to create wordline pulses*

- **Array:**
  - *Used divided wordlines, each global wordline driving eight local wordlines*
  - *Divided the bitlines as well to have each bitline only loaded by 64 memory cells*
  - *Changed the bitline precharge voltage to 700mV*
  - *Moved the sense amps to the center of the array to allow for an array structure of 512x128*

- **SenseAmp:**
  - *Used the clocked sense amp, but changed the precharge capacitance sizes to reduce the read voltage to 700mV and reduce precharge power*
  - *Use 2:1 MUX to select the upper or lower half of the bitlines, so now use 512 sense amps in the array*
  - *Drive tri-state buffers with the sense amps to the output bus*

- **Bitline Precharge:**
  - *Reduced the precharge capacitance to 30 λ, reducing precharge power*
  - *Decreased the precharge voltage to 700mV instead of 1V*

- **Write Circuitry:**
  - *Designed a new write data signal that resets after a certain time to read mode, hence facilitating faster precharge*
  - *Reset is independent of input signal duration*

- **SAE generation:**
  - *Designed a inverter chain with variable capacitors to allow for delay variation after fabrication*
  - *Used post-charge logic to reset the sae after evaluation of sense amp time*

- *Generated a reset signal to create wordline pulses*

- **Signal generation:**

  - *Generated blpc and sapc signals as shadowed images of clk signal swinging between 700mV and ground*
  - *Generated sel <7:0> signal from last 3 address bits to enable the divided wordline architecture and to gate other related signals (like blpc, sae, sapc) for power saving*

- **Project Description Category 5:**

  - *Analyzed multiple power sources*
  - *Analyzed marginal cost of Vdd for the decoder*

2. Performance Results (give results in both ps, and FO4 delays FO4=24ps)

Clk (or address) -> WL rise **__190.8ps, 7.95 FO4_**

WL rise -> Bit Line split **__28.7ps, 1.2 FO4___**

Bit Line -> SenseAmp Out **__70.8ps, 2.95 FO4___**

SenseAmp Out -> Global Out **__49.4ps, 2.06 FO4___**

Total **__339.7ps, 14.15 FO4_**

Min Cycle Time **_____800 ps_____**

What constrained the minimum cycle time? Which phase was critical (e.g. bitline precharge, SAE evaluation, etc)?

We have designed our system with enough safety margins for each signal generation as mentioned below:

SIGNAL	REQUIRED	PROVIDED	SAFETY MARGIN
Wordline pulse width	70.8 ps	141ps	99.15 %
SAE rising	261.6 ps	296.5 ps	13.34 % (but variable capacitor programmability included that gives around 42 % margin)
SAE pulse width	49.4 ps	77.66 ps	57.21 %
Sense Amp sampling	36 ps	105.7 ps	193.6 %
BitLine Precharge	358.2 ps	400 ps	8 %
Total clock	2 * (max (339.7, 358.2))= 716.4 ps	800 ps	11.66 %

Hence, it is clear that we have ample safe margins and shouldn't run into any timing constraints. But, if we desire to decrease the clock time, then the constraint might be the bitline precharge which can be easily handled by increasing the transistor size to 50λ (slight increase in power) and that should help us increase our frequency even further.

3. Power @ f_{clk}=1000 MHz:

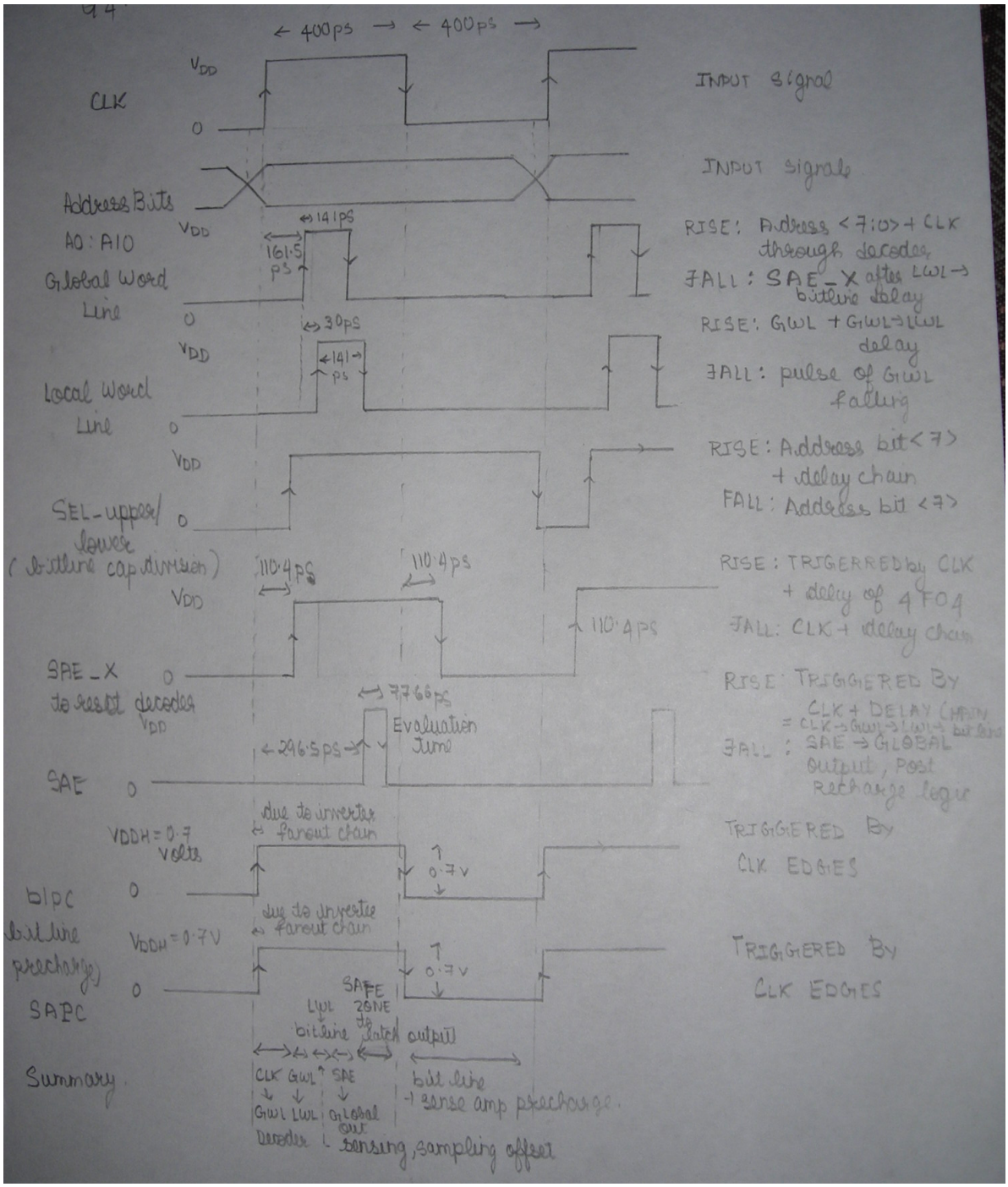
Decoder:	**Gates:** 0.273mW	**Clk:** 18uW	**Leakage:** 0.823uW
Memory Core:	**Gates:** 5.253mW	**Clk:** 43.7uW	**Leakage:** 1.109mW
Sense Ckt:	**Gates:** 1.477mW	**Clk:** 435uW	**Leakage:** .538mW
Final Bus Drive	**Gates:** 1.068mW	**Clk:** 0uW	**Leakage:** 80.9uW
Total:	**Gates:** 8.071mW	**Clk:**496.7uW	**Leakage:** 1.727mW

Explain how power scales with f_{clk}.

As power is calculated by the formula P = CV*Vf , we accept power to increase linearly with frequency. This is reflected in the measured values for our new frequency of 1.25 GHz. Leakage power does not vary much but, dynamic power scales linearly.

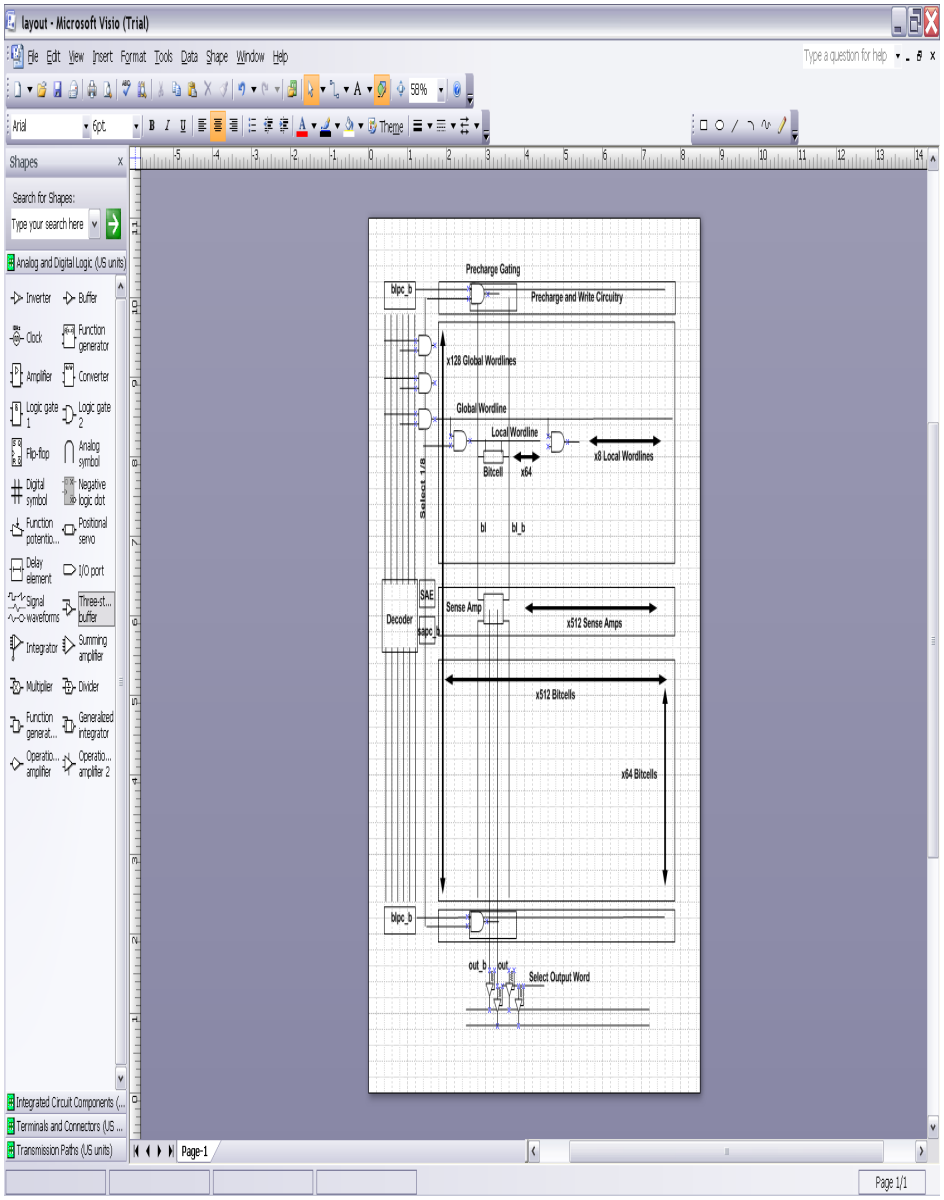
Ex: Decoder dynamic power at 1.25GHz = 0.365 mW (1.34x power at 1GHz)

4. Overall Timing Diagram:



5. Basic Floorplan:

Please draw a floorplan of the chip indicating where all the parts are located. Rough size estimates for the memory (at least the memory array).



The overall memory array should be around 512x32λ wide and 128x20λ tall, so it has a ratio of 2:1. This does not include the extra space needed for the local wordline decoders every 64x32λ on every row, or the sense amps located in the middle of the array.

6. Design Description

Decoder

Objectives:

- To decrease the power consumption of the decoder
- To re-model it to suit the divided word line architecture
- To generate a pulsed word line to decrease power consumption in bit lines
- To check the marginal costs of performance/energy for the gates

Implementation:

Pulsed Wordline

There were two options to use post-charge logic to set the wordline pulse:

1. To reset the gates after the dynamic inverter to 0 after time period=70 ps (time needed for bitlines to reach 150mV difference). It doesn't make sense to add the reset to NAND as that would result in stack of 3 NMOS: highly undesirable (due to sizing, parasitic capacitances and saturation effects). Hence, the option is to make the Inverters NOR and feed them with the reset signal. But this would result in larger fanouts seen as the latter stages have bigger sizes and hence would increase the power.
2. To add the reset stage prior to dynamic inverter. This would result in
   - Lesser branches to which the reset signal should go
   - Lesser sizes and lesser load for the reset signal
   - Lesser power consumption as the initial stages hardly contribute to the power
   - Dynamic inverter can be removed as the wordline falls before the clock, hence we needn't be bothered about the clk to wordline fall delay being twice. This saves immense clock power.

   Based on the above conclusions, we decided to convert the 2nd inverter in the chain to a NOR gate with corresponding sizing fed by a reset signal SAE_X which is fed from the SAE signal generation circuitry.

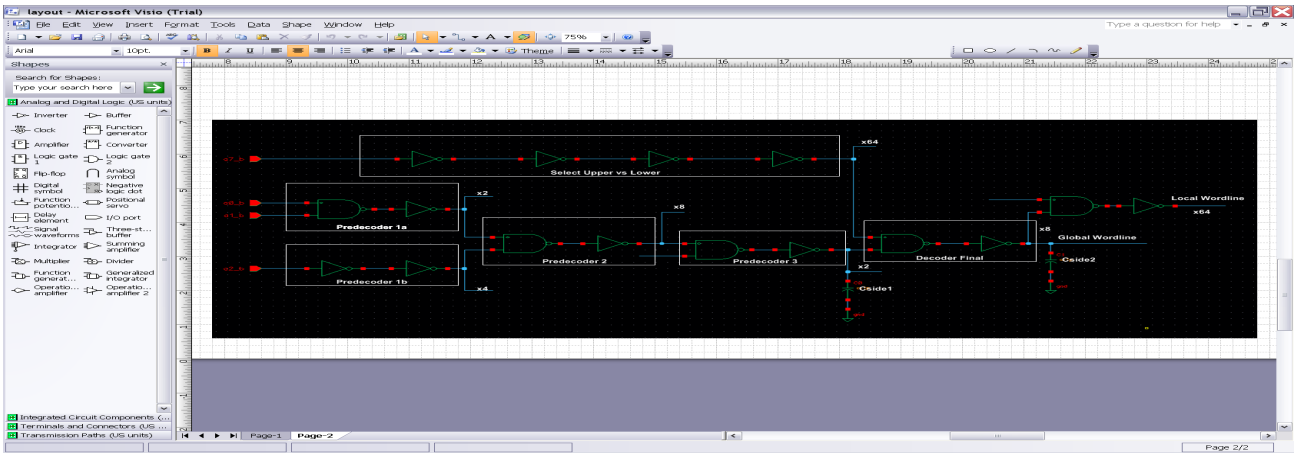
Wordline for the Divided Wordline Architecture

Now, the global word line GWL is feeding 8 LWL (local word lines). Assuming the LWL have a input capacitance of Cgl, Load on GWL = 8 * Cgl + 512 * Cwire / memory cell. Now we have a chain with 2 more stages and 2 points of side loads, one for memory cell capacitance; other is for the 64 branches travelling along the breadth of the memory SRAM.

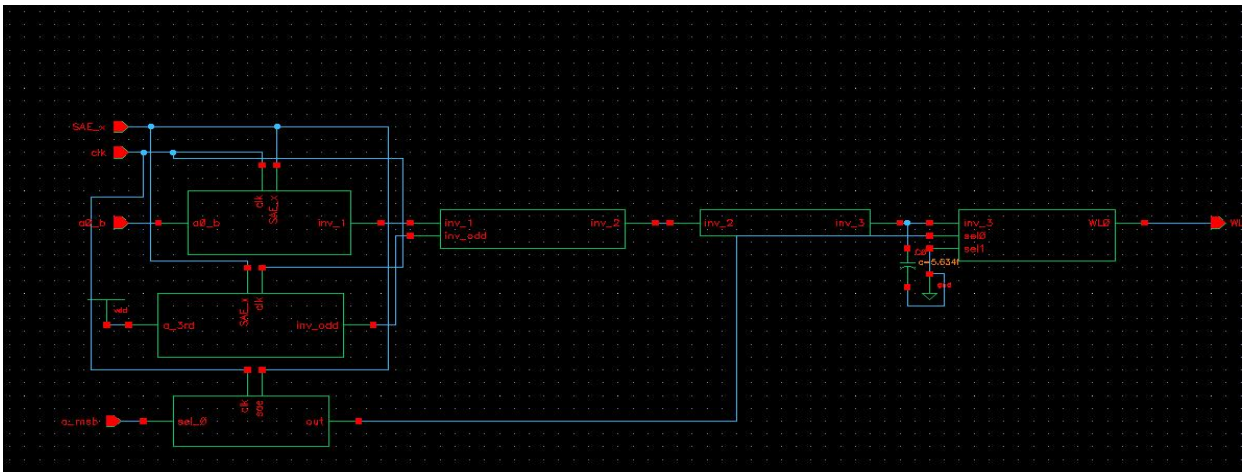
Stage Decision

We need 128 GWL instead of 256. Hence using the previous decoder with dummy input leads to higher delay and more power. Hence I tried to re-design the wordline with 3:8 Predecoder followed by a stage that combines these to give 64

lines followed by a stage that uses the address's MSB to branch them into 128, selecting either upper half or lower half accordingly.



This is the same technique used to split the bitline capacitance and same thing accordingly; so the wordlines can be generated as required for the either half. This removal of 1 signal leads us with 6 address bits and hence easy 3:8 analysis for the predecoder.



It shows the generation of the GWL as LWL schematic is included in the array picture. The sub-circuits predec_1b and sel_inv are used for buffering the lone signals to match their co-ones which pass through NANDs and Inverters and ramp them up to face the desired capacitance. Using Matlab code: the sizes are:

- Cside1 = 64 * Height of cell * Wire cap = 5.632 f F (Each line traverses 64 rows till it reaches the required place)
- Cside2=512 * Width of cell * Wire cap = 72.0896 fF (Each global word line row is spanned over 512 SRAM cells as our array is 512 *128)
- Cload = 512 * (4+4) *λ * Cg / 8 = 15.7696 fF

So here Cload and Cside2 exchange positions as Cside 2 dominates. Using a MATLAB code to optimize the iterative loop these are the sizes obtained:

Stage	Logical Effort	Branching	Size	PMOS	NMOS
Clocked NAND	0.61	1	11.85	6	12
NOR (reset signal)	1.2795	2	55.3868	48	7
NAND	0.81	1	61.7079	25	36
INV	0.833	8	217.199	174	43
NAND	0.81	1	92.9242	38	55
INV	0.833	2	327.0756	262	65
NAND	0.81	1	468.300	193	276
INV	0.833	8	468.300	1103	276
NAND	0.81	1	201	103	98
INV	0.833	1	805	690	115

As seen, theoretical power calculations gave us 0.4015 mW, and delay of 176.1875 ps

Marginal analysis for lower power

The power can be reduced even further by performing marginal analysis on the decoder stages. As local wordline sizing was not very high and we ignore it during the power convergence as it leads to increased complexity. The major stages are stage 8 and 7 which consume high power. So, using the theory of compensating little delay at minima delay point for decrease in power consumption:

Marginal Cost for stage i = (dE/dW)/(dD/dW)

$$\frac{\partial Energy / \partial W_i}{\partial Delay / \partial W_i} = \frac{\alpha \cdot C_g \cdot W_i \cdot Vdd^2}{\tau_{inv}(EF_{i-1} - EF_i)}$$

By iterating over different sizes for the latter stages, we can try matching the energy for the fanout differences and hence try matching the marginal costs accordingly. We utilized two MATLAB codes, one which optimizes the marginal costs for the last four stages while the other attached below tries to recalculate the sizing for the entire chain in order to rather than compensating only for last stage; it matches the costs and delay compensation in each stage in proportion to their power consumption contribution.

New sizes and theoretical power and delay calculations are show below. The code was written for 10 % more delay.

Stage	Old PMOS	Old NMOS	PMOS	NMOS
Clocked NAND	6	12	6	12
NOR (reset signal)	48	7	40	6
NAND	25	36	18	25
INV	174	43	101	25
NAND	38	55	18	26
INV	262	65	105	26
NAND	193	276	39	56
INV	1103	276	498	124

Theoretical delay = 160.805 + 30 = 190.805 ps (10% more)
Theoretical power saving = 0.241 mW
Theoretical Power Savings = 39.80 %

Matlab Code for Power Optimization

```
%Original file after pulse generation: POWER OPt GWL new arch for decoder
%New Decoder Design

m_min=100;
s1_min=0;
s2_min=0;
cin_min=0;
delay_min=0;
LE1=0.61;
LE2=1.2795;
LE3=0.81;
LE4=0.833;
LE5=0.81;
LE6=0.833;
LE7=0.81;
LE8=0.833;

cin1=12;
cs=182.857;
cload=3948.57;

optimum_delay=29.2375;
power_optimum_delay=1.10 * optimum_delay;
pe1_const=2*8*LE1*LE2*LE3*LE4*LE5*LE6;
pe2_const=LE7*LE8;
a=pe1_const/cin1;
b=pe2_const;

for cin7=10:0.1:700
    s1=power((a*(cs+2*cin7)),1/6);
    s2=power(b*cload/cin7,0.5);

    cin8=cload*LE8/s2;
cin7=cin8*LE7/s2;
cloads=cs+2*cin7;
cin6=cloads*LE6/s1;
cin5=cin6*LE5/s1;
cin4=8*cin5*LE4/s1;
cin3=cin4*LE3/s1;
cin2=cin3*2*LE2/s1;
cin1=cin2*LE1/s1;
```

```
wn8=round(cin8/5);
wp8=round(0.8*cin8);
wn7=round(cin7*1.43/2.43);
wp7=round(cin7/2.43);
wn6=round(cin6/5);
wp6=round(cin6*0.8);
wn5=round(cin5*1.43/2.43);
wp5=round(cin5/2.43);
wn4=round(cin4/5);
wp4=round(cin4*0.8);
wn3=round(cin3*1.43/2.43);
wp3=round(cin3/2.43);
wn2=round(cin2/7.62);
wp2=round(cin2*6.62/7.62);
wn1=round(cin1);
wp1=round(cin1/1.85);

cg=1.4;
cj=1.2;
y1=(2*wp1+wn1)*cj/wn1/cg;
y2=cj/cg*(wp2+2*wn2)/(wp2+wn2);
y3=(2*wp3+wn3)*cj/(wp3+wn3)/cg;
y4=cj/cg;
y5=cj/cg*(2*wp5+wn5)/(wp5+wn5);
y6=cj/cg;
y7=(2*wp7+wn7)*cj/(wp7+wn7)/cg;
y8=cj/cg;
    delay=6*s1+2*s2+LE1*y1+LE2*y2+LE3*y3+LE4*y4+LE5*y5+LE6*y6+LE7*y7+LE8*
y8;
    m=abs(delay-power_optimum_delay);
    if m<m_min
        m_min=abs(m);
        s1_min=s1;
        s2_min=s2;
        cin_min=cin7;
        delay_min=delay;
    end
end
s1_min
s2_min
m_min
cin_min
delay_min

s1=s1_min;
s2=s2_min;
cin7=cin_min;

cin8=cload*LE8/s2;
cin7=cin8*LE7/s2
cloads=cs+2*cin7;
cin6=cloads*LE6/s1
cin5=cin6*LE5/s1
cin4=8*cin5*LE4/s1
cin3=cin4*LE3/s1
cin2=cin3*2*LE2/s1
cin1=cin2*LE1/s1

wn8=round(cin8/5)
wp8=round(0.8*cin8)
wn7=round(cin7*1.43/2.43)
wp7=round(cin7/2.43)
```

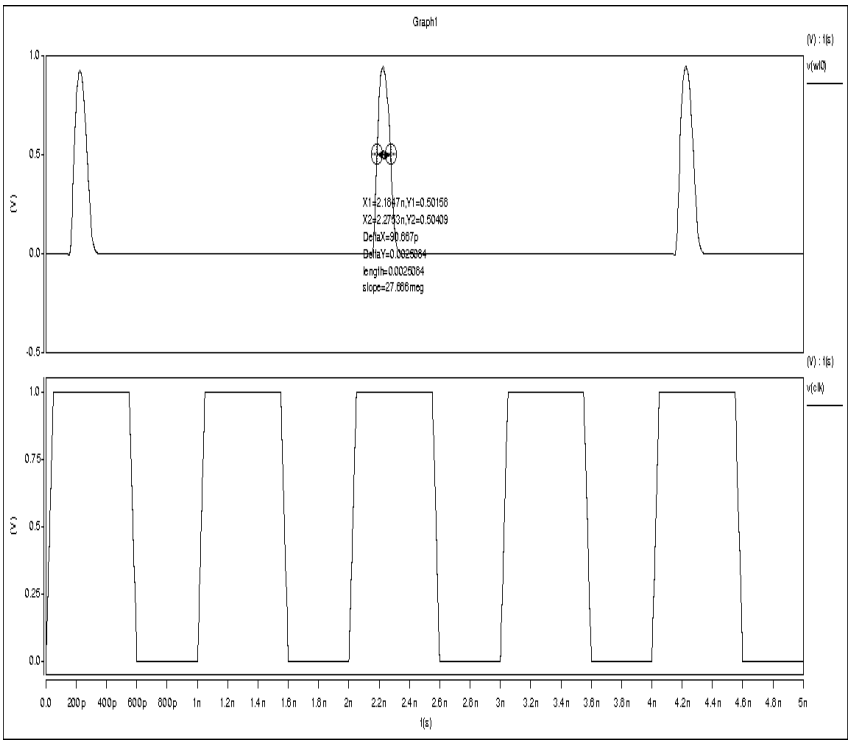
```
wn6=round(cin6/5)
wp6=round(cin6*0.8)
wn5=round(cin5*1.43/2.43)
wp5=round(cin5/2.43)
wn4=round(cin4/5)
wp4=round(cin4*0.8)
wn3=round(cin3*1.43/2.43)
wp3=round(cin3/2.43)
wn2=round(cin2/7.62)
wp2=round(cin2*6.62/7.62)
wn1=round(cin1)
wp1=round(cin1/1.85)

cg=1.4;
cj=1.2;
r=cj/cg;
y1=(2*wp1+wn1)*cj/wn1/cg;
y2=cj/cg*(wp2+2*wn2)/(wp2+wn2);
y3=(2*wp3+wn3)*cj/(wp3+wn3)/cg;
y4=cj/cg;
y5=cj/cg*(wp5*2+wn5)/(wp5+wn5);
y6=cj/cg;
y7=(2*wp7+wn7)*cj/(wp7+wn7)/cg;
y8=cj/cg;
delay=6*s1+2*s2+LE1*y1+LE2*y2+LE3*y3+LE4*y4+LE5*y5+LE6*y6+LE7*y7+LE8*y8

power=4*((wp4+wn4+r*(2*wp3+wn3))+(wp5+wn5+r*(wp4+wn4)))+2*((wp6+wn6+r*(wp
5+wn5))+(2*(wp7+wn7)+cs+r*(wp6+wn6)))+(wp8+wn8+r*(2*wp7+wn7)+cload+r*(wp8
+wn8))

MC8 = cin8/((LE7*(cin8/cin7))-(LE8*cload/cin8))
MC7 = cin7/((LE6*(cloads/cin6))-(LE7*cin8/cin7))
MC6 = cin6/((LE5*(cin6/cin5))-(LE6*cloads/cin6))
MC5 = cin5/((LE4*(cin5/cin4))-(LE5*cin6/cin5))
MC4 = cin4/((LE3*(cin4/cin3))-(LE4*cin5/cin4))
MC3 = cin3/((LE2*4*(cin3/cin2))-(LE3*cin4/cin3))
```

After wiring up the circuit using the above sizes, this is the resulting waveform obtained:



Power:

Theoretical Calculations:

The activity factors gets halved for the predecoder stages as the sel line will always be high for one case and hence we need only one of the 64 outputs to go up. Also, clock power is negligible.

Stage		Switching factor	Cload(λ)	Cparasitic(load)
Clock power		Negligible	-	-
Predecoder_1a		Negligible	-	-
Predecoder_1b		Negligible	-	-
Predecoder_2	NAND	2	126	61
	INV	2	352	126
Predecoder_3	NAND	1	131	62
	INV	1	372.85	131
sel_inv		1	930.77	263
Decoder_final	NAND	1	622	134
	INV	1	3948.57	622
Total	(λ)	-	6961.19	1349
	(fF)	-	214.404	35.6136
Power	mW	-	-	0.25 mW

Power from spice simulations:

Stage	Switching Factor	Dynamic +Leakage (mW)	Leakage (nW)	Total = (switching * dynamic) + (no. of instances * leakage) (mW)	Theoretical Power (mW)
Predecoder2	2	0.02046	70.58	0.04092	0.019659 * 2
Predecoder3	1	0.02100	100.0	0.021	0.020613
MSB signal inverter chain	1	0.04271	118.8	0.04269	0.035611
Final decoder stage	1	0.1693	343.8	0.1693	0.160732
Clock	1	0.00185	0.0232	0.0185	-
Total	-	-	-	0.29243	0.25

Simulations:

For frequency = 1 GHz:

CLK - > GWL delay = 161.5 ps
GWL- > LWL delay = 29.3 ps
Total delay = 190.8 ps
Pulse width of Wordline = 140.14 ps
Total power (GWL) = 0.292 mW

For frequency = 1.25 GHz

Total Power = 0.365 mW

Bitline Precharge

Objectives:

- To decrease the power consumption of the circuit
- To ensure read and write margins are not badly affected
- To check the precharge time fits in the 0.8 ns clock cycle

Implementation:

Decrease the 80 λ transistors to a lower value

The 80 λ transistor causes huge power consumption for the pre charge circuit as power = 256 * 3 * 80 λ * Cg * Vdd * Vdd * f

Even with the divided wordline architecture, the power is decreased by 8 times which is a major improvement. But power can also be reduced by decreasing the precharge capacitance. Using dc sweep, we find that:

Precharge Transistor Size decreases -> Precharge Time Increases.

Decreasing to even 10 λ is not a problem with read margin; but writing 0 and then precharging back is not feasible within 400 ps (new clock cycle = 0.8 ps).
So, we came up with a ***write circuitry*** that resets the write signal to 1 after some time period. Once a zero is written pulling the line to 1 back will just cause non destructive read and hence facilitate faster precharge and equalization of bitlines.

Write Circuitry

The aim is to reset the inverter which write zero to one after time = 200 ps by which 0 is written. This time is the safe maximum limit obtained after schematics verification. So, we decided to use the 'SAE' signal generated to reset the inverter as it goes high approximately the same time.

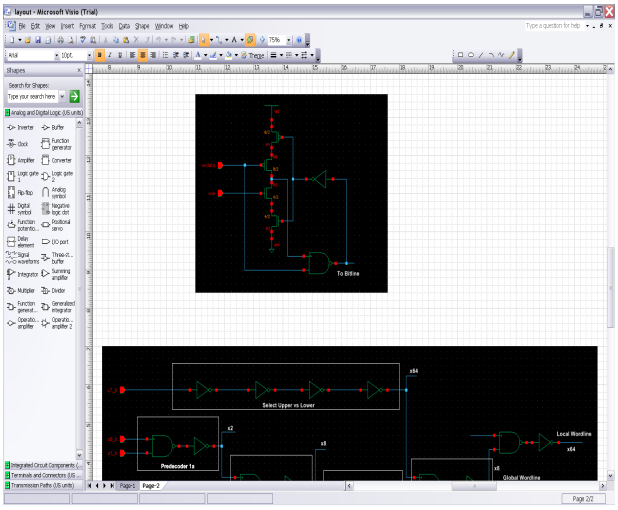
Another problem, is reset should be independent of the input signal as wrdata available time may be longer than reset loop time i.e.

Input Signal Duration > 2*Reset Loop time,

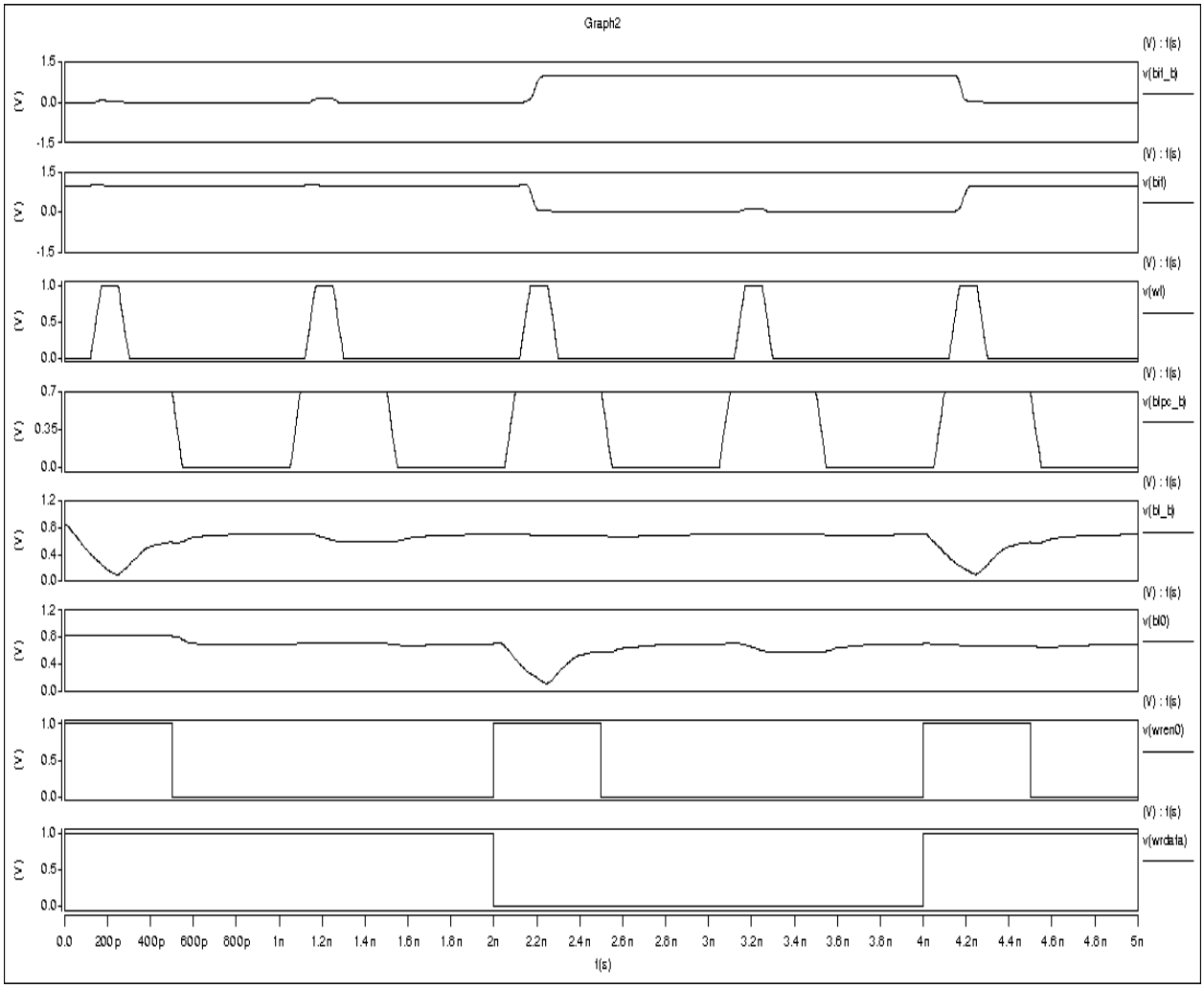
Hence we added a pmos in series to allow the signal to rise back only when input is low. The nmos triggered by SAE causes reset to happen only when SAE goes high i.e. after 200 ps (approx).

Using this logic, the precharge and write circuitry was modifies and simulated results of which are shown below:

The rewired circuitry:



The timing signals generated that prove that memory cell can be written and the precharge can occur within 400 ps:



Calculations:

Hand Calculations:

Only 1/8th of 512 array is active at a time and blpc signal oscillates between 0.7 V and ground.

C_{blpc_b}= (30*3)*0.0308fF/λ =2.772fF

Vdd = 0.7

N = 512/8 = 64

P_{blpc_b}= Vdd^2*f*N*C_{blpc_b}

At f = 1 GHz:

P_{blpc_b}= **0.0869 mW**

At f = 1.25 GHz (Tclk = 0.8ns)

P_{blpc_b}= **0.1086 mW**

Spice simulations:

P_{blpc_b}= **0.0884 mW**

Hence, there is clear agreement between the data that power has decreased by 95.42 %

Bitlines Precharged to 700mV

Objectives:

- Reduce the power dissipated during a read

Implementation:

- We reduced the bitline precharge voltage to 700mV from 1V, and this helped reduce the memory core power. The power dissipated during a read is given by:

$$P_{av} = (num\ bitcells) I_{dsat} V_{dd} t_{wl\ high} / t_{clk\ period}$$

Since the bitlines are at a lower voltage, Idsat will decrease, decreasing the overall power. Also, for reads, Vdd will be 700mV instead of 1V, which also decreases the power. Coupled with our reduced number of bitcells on each bitline and our pulsed wordlines, the overall power should be significantly reduced.

- Our core power would have been further reduced if we had decreased the sram cell Vdd to 700mV as well, but we weren't able to run the necessary noise margin analysis to fully analyze the effects of lowering Vdd by 300mV, so this change hasn't been included.

Power Calculations:

The new value of Idsat with the bitlines held at 700mV was obtained through simulations: 12.37µA.

$$P_{av} = \frac{64 * 12.37\mu A * 700mV * 140ps}{1000ps} = 110\mu A$$

This value is 1.2% of the original power dissipated during a read in the memory array.

Divided Wordlines / Bitlines

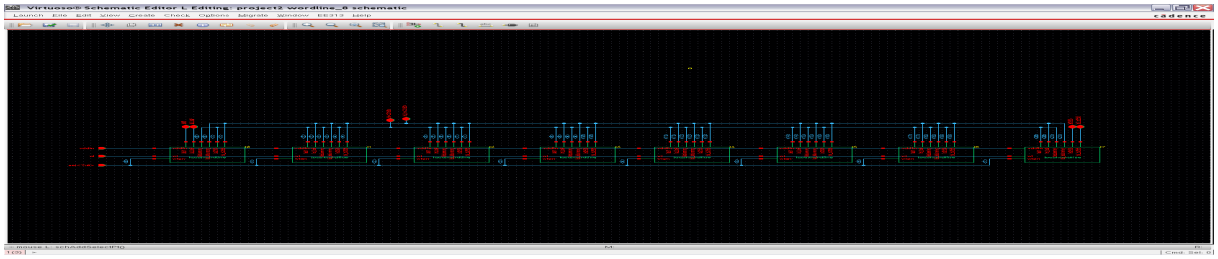
Objectives:

- To decrease capacitance on the wordline by using divided wordlines, speeding up reads and decreasing the overall bitline capacitance that switches with each read.
- To decrease bitline capacitance and reduce read time.

- Use the select signals from the divided wordlines to gate the precharge signals and sense amp enable signals, reducing overall power consumption.

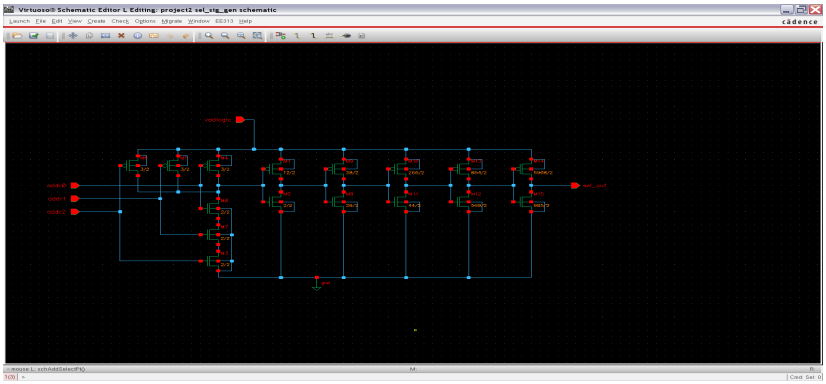
Implementation:

- We divided the bitlines into quarters by placing the sense amps in the center of the array and setting the array up to be 512 x 128 instead of 256 x 256. This gave us eight groupings of 64 bits horizontally (eight local wordlines), with each bitline having a load of 64 bits. So the length of the bitlines was quartered since they only extend halfway down the array now, instead of the full height, with the height of the array being halved.
- A large part of changing the architecture was determining the select signals and fanning them out to drive the necessary number of gates. We select which word to read by looking at the lower three address bits. These are fed into a decoder which creates eight one-hot signals to drive the local wordline decoders. Even though the power of reading the bit cells decreases (bitlines precharged to 700mV, which reduces Idsat), the capacitance introduced by the select logic is fairly large, which doesn't mean that our power reduction is as great as anticipated.

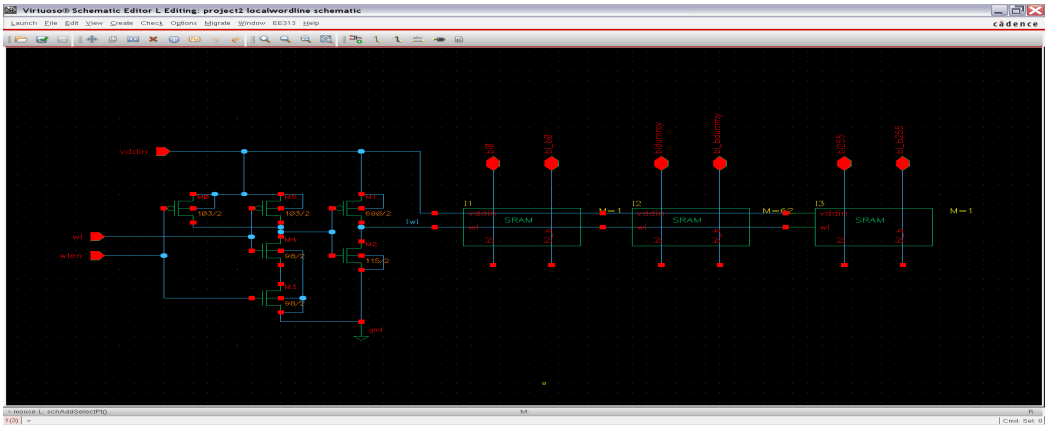


- But this architecture was very effective at reducing the read timing from the wordline rising to the bitline falling by 150mV. After the global wordline goes high, the local wordline takes 28.7ps to rise, and after that, the bitline takes 70.8ps to fall by 150mV. This is a rather large reduction from the 172ps delay in the original array.

Power Calculations:



Total switched capacitance: 8709λ gate cap = 268fF (select signal generation, [shown above] x 8)



128* (103+98+690+115) λ + 2*(2*60+2*1440) λ = 134768 λ
= 4.15fF (local wordline decoders [shown above] + blpc gating)

Power = ½*αCVdd²f= ½*2*4.418pF*1*1GHz = 4.418mW

In this, we are ignoring the power due to charging and discharging the bitlines, calculated above, since it is such a small fraction of the power dissipated by select signal fanout.

Simulation Results:

Running at 1GHz, our total power dissipation was 5.253mW and the static power dissipation was 1.109mW. So the total dynamic power dissipation is 4.144mW.

Sense Amps

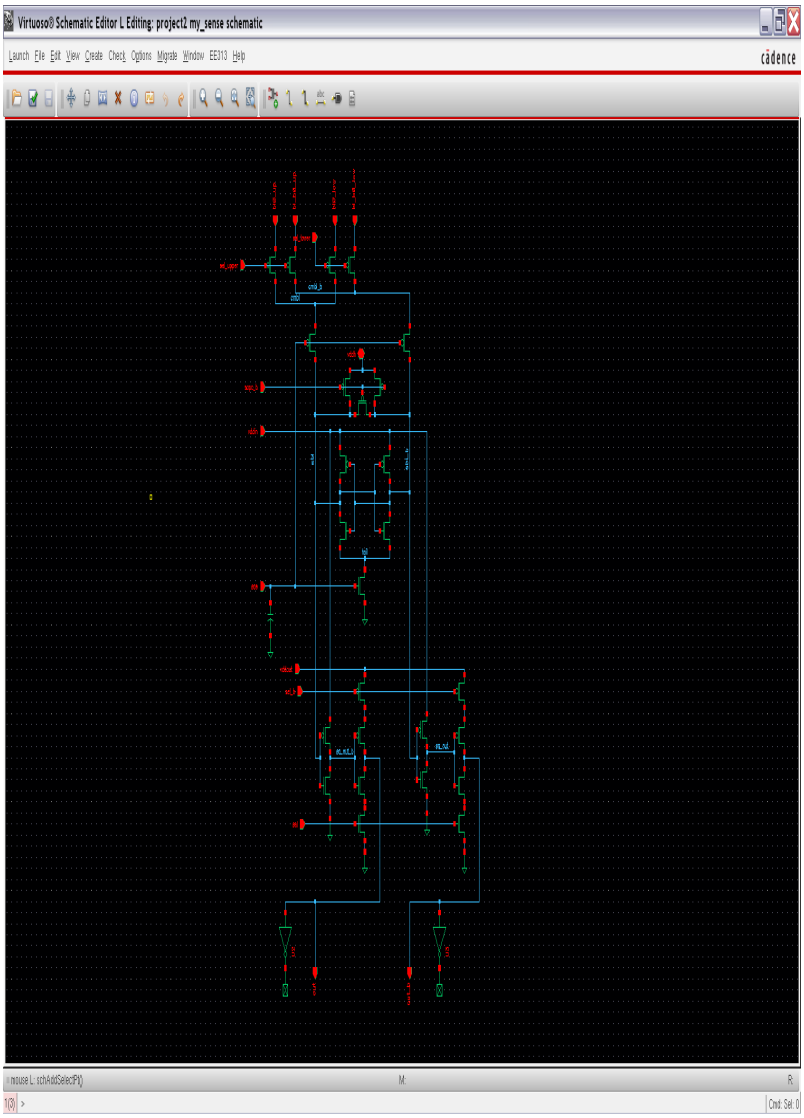
Objectives:

- Maintain low power even though we have increased the total number of sense amps by a factor of eight
- Drive increased output load due to relocation of the sense amps

Implementation:

- We gated both sae and sapc_b with sel<7:0> so that only sense amps whose bitlines had evaluated would actually evaluate, and only sense amps which had evaluated would be precharged again.
- We changed the 4-input mux to a two input mux to take in signals from the upper and lower halves of the array, instead of 4 different bitlines.

- We lowered the super bitline precharge voltage to 700mV, to match our bitline precharge voltage.
- We added a pair of inverters to the outputs of the sense amps to drive the actual output, since we have to drive a wire that runs the length of the array, and feeds an output load of 150λ at the bottom of the array. Since we are using divided wordlines, it no longer makes sense to interleave the sense amps. So we have eight sense amps connected to a bus with tri-state drivers, selected by sel<7:0>. Each bus runs the length of the array (512x32λ) and then drives the 150λ output at the bottom of the array. So the total output capacitance is 402λ, meaning that we need a single gate, the tri-state buffer, between the sense amp output and the sram output. The modified sense amp schematic is shown below. Since we only add one gate to the output driver, this doesn't increase our overall delay significantly.



Power:

Our measurements from simulation are 1.297mW power dissipated in the cross coupled inverter pair and the output inverters, with .18mW dissipated due to precharging the super bitlines. The static power dissipated is .538mW and 1.02uW, respectively. So the total dynamic power dissipated is .939mW at 1GHz, and 1.17mW at 1.25GHz

This overall power dissipation is about what we expected, but it makes sense since we have eight times the number of sense amps as previously, and there will be a lot of static leakage current, but our dynamic power is close to the original power dissipation, .99mW. Overall, the dynamic power is very close to the original sense amp power, even though we are using eight times as many, because we are gating the precharge and enable signals.

The power dissipated due to the output drivers is rather small, 80.9uW static and 99.5uW total power, so the output drivers dissipate very little dynamic power, 18.4uW, at 1GHz.

SAE Generation

Objectives:

- To generate the Sense Amplifier Enable SAE signal
- To generate the SAE to drive the modified write circuitry
- To generate the SAE_X signal to reset the word line pulse
- To decrease the power consumption
- To match the variability due to random variation
- To create the SAE signal as pulse of shorter duty cycle

Implementation:

Design and choice of gates

We had two choices to replicate the wordline and bitline delay

1. Replica bit cell with variable number of memory cells connected and discharge one of them when the replica wordline goes high and then mimic the wordline delay using a chain of inverters.
2. Use a series of inverters to model the whole delay

Analyzing the options we see that the advantages of these options are:

- Option 1 gives more accurate random mismatch variation of the bitline delay as we are mimicking using the replica bitline and memory cells. While the inverter chain, as discussed in lecture, needs to have 50% margin to match the variations.
- Option 2 leads to lower power and hence optimization of circuit

- Option 2 is better in case wordline to bitline delay is less, which in this case is 3-4 FO4 inverters, hence can be modeled easily.
- Adding *programmability* to the inverter chain using *variable capacitors* minimizes the problem of variation mismatch. To include programmability, we used 'nmos4 and pmos4' models with drain and source shorted to a variable voltage vg and vg_b respectively. Varying vg will keep the flexibility in delay matching. This is proved by the hpsice simulation results:

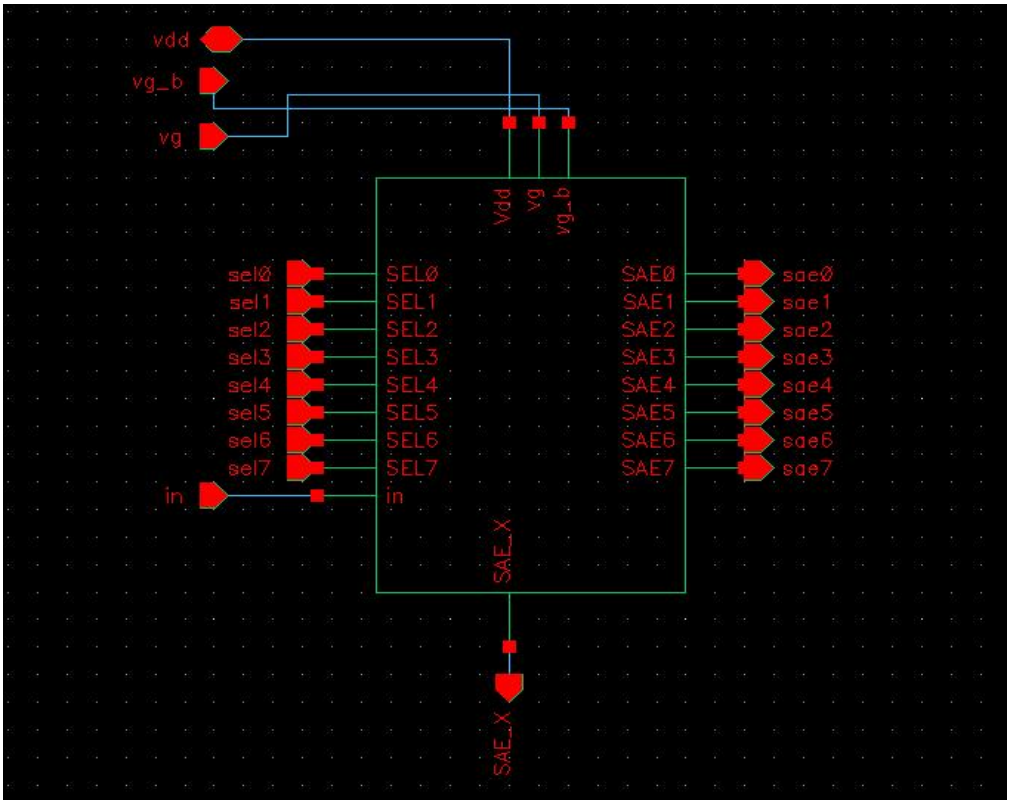
  Vg= control for nmos = ctrl
  Vg_b=control for pmos = vdd-ctrl
  delay = CLK to SAE delay
  delay1 = CLK to SAE_X delay (feeding the reset of decoder)
  Pulse width = pulse width of SAE

```
$DATA1 SOURCE='HSPICE' VERSION='B-2008.09-SP1 32-BIT'
.TITLE 'example spice deck'
 ctrl             delay            pulsewidth       delay1

    0.             2.965e-10        7.761e-11        1.104e-10
  5.000e-02        2.936e-10        7.668e-11        1.089e-10
    0.1000         2.906e-10        7.549e-11        1.073e-10
    0.1500         2.876e-10        7.434e-11        1.057e-10
    0.2000         2.845e-10        7.323e-11        1.041e-10
    0.2500         2.815e-10        7.210e-11        1.025e-10
    0.3000         2.785e-10        7.098e-11        1.009e-10
    0.3500         2.757e-10        6.983e-11        9.938e-11
    0.4000         2.729e-10        6.873e-11        9.794e-11
    0.4500         2.703e-10        6.781e-11        9.654e-11
    0.5000         2.681e-10        6.689e-11        9.529e-11
    0.5500         2.662e-10        6.605e-11        9.425e-11
    0.6000         2.646e-10        6.549e-11        9.343e-11
    0.6500         2.634e-10        6.494e-11        9.273e-11
    0.7000         2.622e-10        6.448e-11        9.209e-11
    0.7500         2.610e-10        6.399e-11        9.146e-11
    0.8000         2.599e-10        6.353e-11        9.086e-11
    0.8500         2.588e-10        6.312e-11        9.029e-11
    0.9000         2.578e-10        6.267e-11        8.976e-11
    0.9500         2.569e-10        6.230e-11        8.928e-11
    1.0000         2.562e-10        6.195e-11        8.886e-11
```

This demonstrates the flexibility we have in designing the SAE signal.

Visualizing the symbol and designing from the scratch, below is the symbol view giving rough idea of inputs and outputs to be generated:



Loads on SAE

Each SAE_i has to feed the following loads, where I ranges from 0 to 7 to select the particular group of signals:

- Sense amp has PMOS (sampler) and NMOS (tail mos which onsets the evaluation stage) Load = 12*2 + 8 = 32λ
- Write circuitry: the nmos in bottoms to reset the write signal
  Load 2 = 6 *2λ (two as both blpc and blpc_b need it)
- Wire capacitance : 64 memory cells
  Load 3 = 64 * 32 * 0.2 /1.4 =292.57λ

  Total load = (512 / 8) * (32 +12) + 292.57= 3108.57λ

Using FO4 inverters, we can scale out to this load using the last stages and previous stages can be designed as

PMOS = 3W=24λ

NMOS= W = 8λ

Next stage sizing can be *SAME* as the remaining 3 load can be adjusted in the sizing of the variable capacitors which are PMOS: 72 and NMOS: 24. This saves a lot of power. The saturation factor of 1.42W and 1.67W for NMOS and PMOS stack has been incorporated accordingly. Also, base size has been chosen as PMOS = 24 λ and NMOS = 8 λ to help in scaling later. The remaining 3 factor of FO4 has been done in the variable capacitors using NMOS of 24 λ and PMOS of 72 λ.

- ***Specifications:***

  From hspice, delay of FO4 inverter = 25 ps (approx)

  CLK to GWL delay = 161.5 ps
  GWL to LWL delay = 29.3 ps
  LWL to Bitline delay = 63 ps
  Total shift required (including 10 % margin) = 279.18 ps
  No. of inverters = 12 to 14

  Pulse width = evaluation time = spl-> out delay in Sense Amp circuit
  = 65 ps (approximately)
  Reset duration = 3 inverters

  Delay of SAE_X = lwl to bitline delay + safety margin = 100 ps (safe zone)

  Plug out point = 4 inverters roughly

- ***MUXing into 8 branches:***

  Using the sel <7:0> from the 3 address bits, the SAE needs to be split into SAE<7:0>; the branching can be taken care while scaling up the later stages. Also the PMOS stack velocity saturation effect can be taken into effect by using 1.656, and instead of a proper NAND using series PMOS solves the purpose.

- ***Generating SAE_X:***

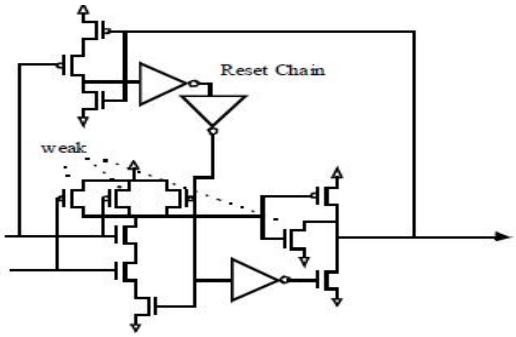
  The SAE_X is fed to the second gate of the predecoder_1a (2 instances) and the inverter chain of predecoder_1b (2 instances) and select inverter (2 instances for upper and lower bit).

  Total load = 2*{ (40+6)*(2 gates) + (20+3)+(13+4)} = 264 λ

  Again as we need the reset after 90 ps (wordline to bitline delay) the pin can be drawn after 4th or 5th stage and fanned out to drive the desired load accordingly.

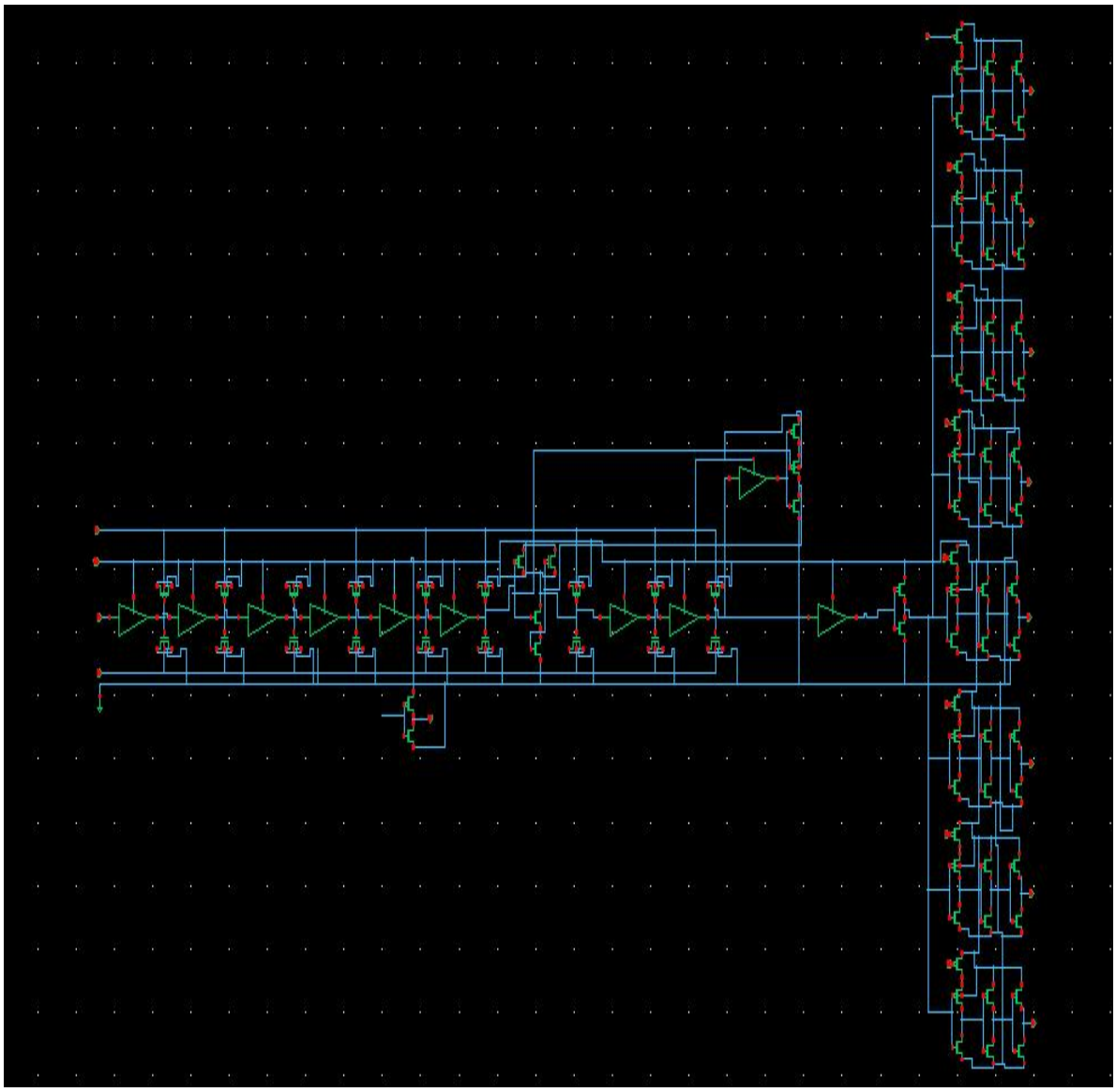
- ***Reset logic:***

  To generate a shorter pulse independent of input clock pulse we used the post charge logic discussed in lecture:

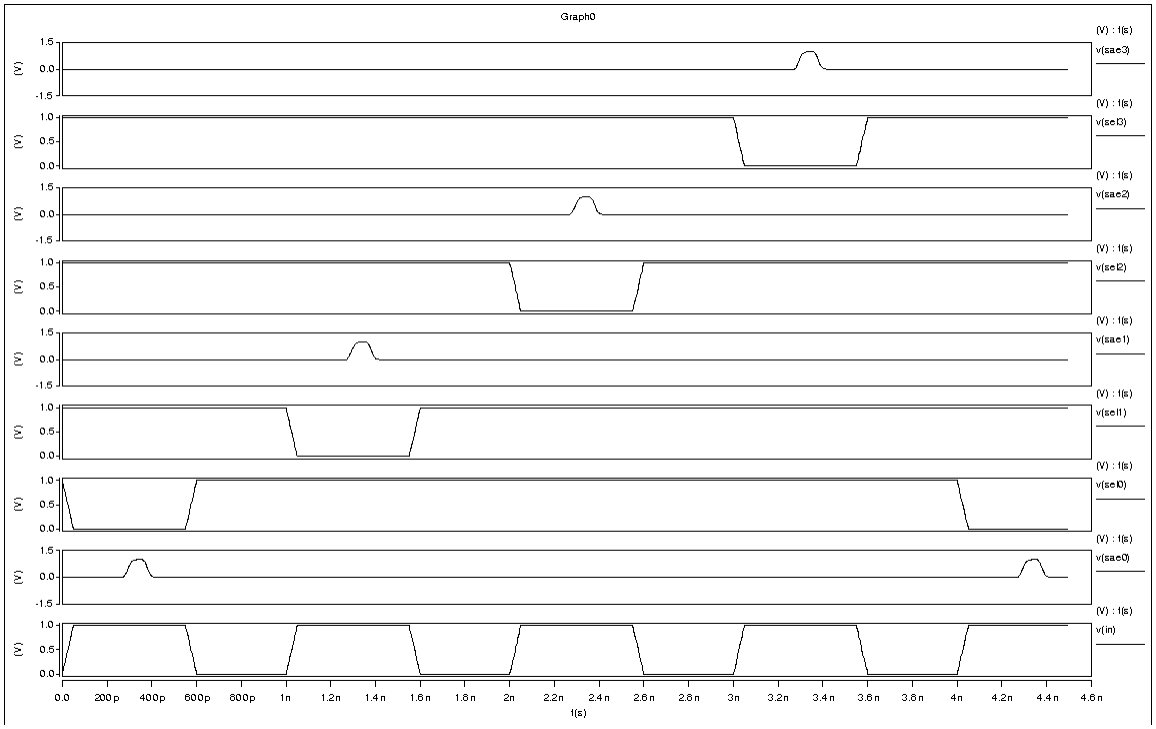


The middle part in the schematic clearly indicates the reset logic. While implementing this logic, the sizing has been taken care to maintain FO4 and to reset accordingly.

Our schematic is:



Using all of these and generating the signals we get:



This shows roughly the waveforms obtained for toggling between sel <0> to sel <3> and the obtained sae<0> to sae<3> in pulsed form outputs.

Calculations:

Hand calculations (done previously):

Total delay = 279.18 ps
Pulse width = 65 ps
Delay of SAE_X = 100 ps

From spice simulations:

For the case: Vg =0V
Vg_b=Vdd
f=1GHz

Total delay = 296.5 ps
Pulse width = 77.66 ps
SAE_X delay = 110.4 ps
Power dyn+leakage = 0.4385 mW
Power leakage = 0.003544 mW
Power dynamic = 0.4349 mW

For T=0.8ns, f=1.25 GHz

Power dynamic = 0.53 mW

Signal Generation

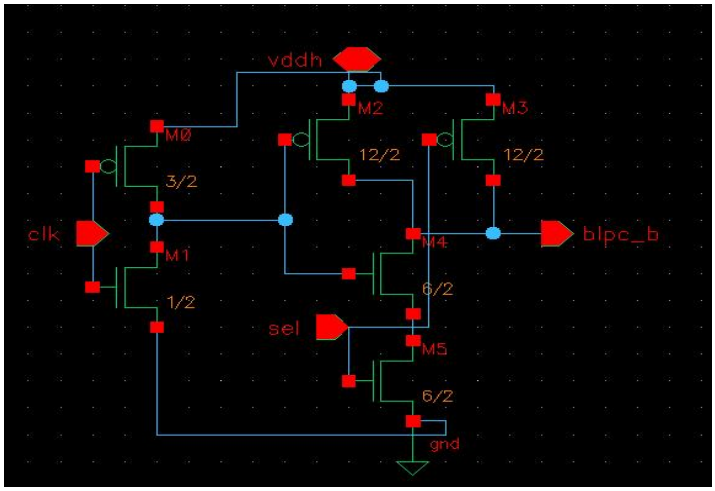
Objectives:

- To generate the blpc (Bit Line Pre-Charge) signal
- To generate the sapc (Sense Amp Pre-Charge) signal
- To generate the sel <7:0> using 3:8 decoder

Implementation:

Here, we start by designing the blpc generation. Remaining signals can be generated similarly. So, the steps involved are:

- Estimate the load to be driven:
  In this case, the blpc needs to drive (40+20) = 60 λ for each half of the upper or lower section according to select signal.
- Model with respect to CLK:
  We would prefer blpc to closely shadow clk signal without much delay as the sum of clk->WL + wl-> bitline + bitline->sense amp output + sa output -> global output is less than clk/2 = 0.4ns
  So, our inverters should not cause much delay.
- Determine the swing of the signal:
  This is 0.7 volts to ground for blpc and sapc as discussed previously. Then you should change the voltage supplies of the inverters accordingly to get the desired output voltage swing. But, this may affect the noise margin and switching point. Hence careful analysis needs to be done.
- Use FO4 invertors to optimize between desired drive load, delay needed and using the select signal keeping into track the velocity saturation model.
  Employing these techniques, blpc was generated.



Here an inverter followed by NAND has been used. SAPC generation is almost similar to blpc as it precharges at the same time as bitlines. Only difference in the load capacitance seen by it and hence corresponding sizes of inverter and NAND gate differs. Sel<7:0> varies in the sense that is independent of the clock and is restricted by input capacitance of 24 λ on the address input.

Calculations:

Hand Calculation:

Total load = (12+6 +60) * (2 circuits for each half of the bitline)
Total parasitic = (3+1+12*2+6)*2
Total cap=5.7024 f F

At f= 1GHz power = 5.7024 uW
At f=1.25 GHz power = 7.128 uW

At f=1.25 GHz:
Spice simulation: Dynamic = 3.525 uW
Leakage = 2.708 nW

This clearly proves that generation circuitry for blpc and similarly for sapc consumes negligible power. Hence they can be neglected.

On the other hand, sel<7:0> and other signals splitting locally near the memory core consumes high power due to large size of capacitances involved. This power is included in the simulations for memory core power and has been verified in that section.

Power Supplies

Our design uses 3 separate (potentially two, if SAE doesn't need to be tuned) voltage sources. In a real design, this could be problematic due to distributing the power across the chip, having three separate power grids. The problem is slightly simplified by

the 700mV supply only being needed for the bitline precharge and sense amp precharge circuits. The power grid wouldn't have to cover the entire chip, but that could complicate layout significantly.

The third voltage source, connected to variable capacitors (source, drain connected nmos and pmos), must be tunable and precise, since it controls timing for sae, which is, although not critical, important to have precise timing for. This voltage would only need to be distributed to a small area of the chip, mainly the sae generation circuitry.

But, overall, having three separate power sources could be problematic on a system level too, significantly increasing the complexity and cost of the SRAM and system in the end.

Category 5: *Matching marginal cost of Vdd for the decoder*

As, calculated theoretically,

$$R = K\,(Vdd - Vth)/Vdd \text{ [where K = proportionality factor]}$$

$$Delay = C_1\,(Vdd - Vth)/Vdd$$

$$Energy = C_2\,Vdd^{*}\,Vdd$$

$(dDelay/dVdd)/(dEnergy/dVdd) = C_3/Vdd^3$; approximately.

Hence matching this ratio (C_3 varies), we can optimize the decoder even further.

```
$DATA1 SOURCE='HSPICE' VERSION='B-2008.09-SP1 32-BIT'
.TITLE 'example spice deck'
 vdd    delr     st2r     st3r    st4r     ppsel    ppdec2    ppdec3    pfindec
  (V)     ps       ps       ps      ps       uW       uW        uW         uW

1.00   161.5    43.81    25.55   55.27     42.72    20.43      1.00       169.5
0.95   163.9    46.94    26.60   51.35     42.69    20.48      1.02       165.9
0.90   170.7    50.85    28.12   49.97     42.65    20.53      1.06       150.5
```

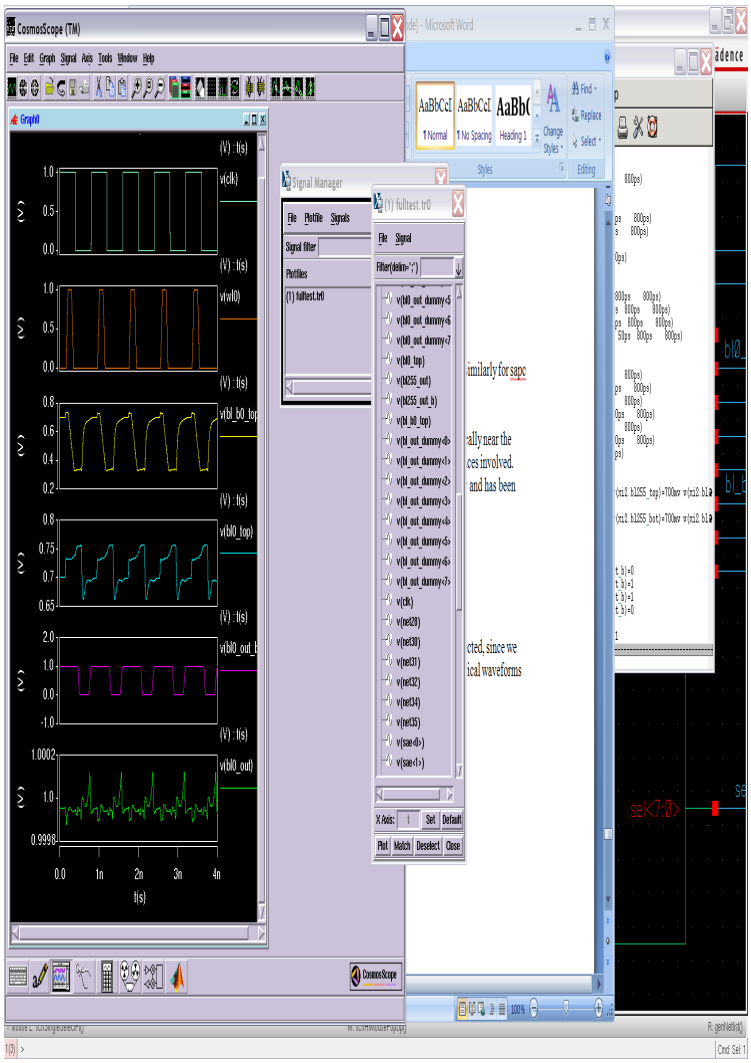
As is clear from the simulation data,

As Vdd decreases, delay increases and total power decreases. The marginal costs were calculated for various sizings and corresponding power/delay ratio was calculated in MATLAB. But, since the power was already low at 0.3 mW and delay was more critical, so the idea was dropped to match the marginal costs.

Overall Design

Simulations:

When we connected the parts together, they were worked as expected, since we had simulated each with the input/output timing from the others. The critical waveforms are shown below: the clock signal, wordline triggered from the clock, bitline and bitline_b, and the outputs from the sense amp.



7. Analysis – The memory you wished you designed

The current design was done with the main intention of decreasing power consumption. Comparing our design to the original design we get the following data for our new frequency of 1.25 GHz:

Stage	Old Design	New (f=1GHz)	New (f=1.25GHz)	Percentage reduction
Decoder	2.03 mW	0.273 mW	0.365 mW	82.019 %
Bit Line Precharge	1.93 mW	0.087mW	0.109mW	94.37 %
Memory core power + Generation circuitry (SAE)	9.16 mW + negligible	5.16 mW + 0.37 mW	5.92 mW + 0.53 mW	29.5%
Sense Amp Power + Drivers	1.18 mW	2.545 mW	2.957 mW	2.5% increase
Total Power	14.3 mW	8.435 mW	9.881 mW	30.9%

This clearly proves the superiority of this design over the previous one. We were able to increase frequency by 20% and decrease power by 30%. But nevertheless, there are still areas where this design can be improved. Given more time, we would look at improving the following things:

- Sense Amp Sizing: The sense amp sizing, besides the pre-charge part, hasn't been modified at all. We could change the sizing to enable faster evaluation.
- Low Vt: We haven't used low threshold voltage transistors, which when used for the tail nmos would help us to play with voltage supplies without worrying as much about noise margins.
- Sizing of memory cell: Changing the ratio of the pass transistor to the nmos will help to increase Idsat (Weff = W (1+L1/L2) increases) and thereby decrease delay. But, the read margin would have to be analyzed as the nmos gets weaker.
- Voltage sources: Except for precharge and tuning sae, we have not played with voltages of the decoder and memory cell. By changing the supply voltage of the decoder and memory cells, we could increase delay slightly and further decrease energy. But, this would also involve noise margin analysis to ensure that the memory cells are both writable and readable.
- Using a specific signal for the precharge circuitry, instead of precharging the bitlines for the entire negative edge of the clock. This would have allowed us to further reduce our frequency without having any adverse effects on other timing or delay. It also would have decreased our power further, since the precharge time would be shorter.

If those areas were explored and implemented, then the design would have given even better performance for decreased power consumption.