UNIVERSITY OF CALIFORNIA AT IRVINE
The Henry Samueli School of Engineering
Department of EECS

EECS189AB Final Report

Project title:

UCI-SAW

Faculty Mentor:

Dr. Ahmed Eltawil

Student name	Student ID
Ramin Mousavi	**29857233**
Brian Brown	**43112398**
Mohsen Rafizadeh	**35665479**

EECS189AB, FALL 2007 &WINTER 2008

Table of Contents:

Section	Page
Section 1: Introduction, Back Ground and Description of Project	**Page 4**
Abstract	**Page 4**
Background.	**Page 5**
Common Models for Power estimation	**Page 6**
Our Objective	**Page 7**
Section 2: Detailed description of work performed	**Page 8**
Part1: Investigation and Information gathering	**Page 8**
Major Leakage mechanism	**Page 11**
Gate Leakage	**Page 12**
Vt and Subthreshold Leakage	**Page 13**
Component Characterization	**Page 15**
Memory Cell	**Page 15**
Sense Amplifier	**Page 17**
Decoder	**Page 17**
Part 2: Algorithm, Execution and Results	**Page 18**
Part 2: UCI-SAW Website	**Page 20**
Perl	**Page 21**
HSPICE & Matlab	**Page 24**
Future Work	**Page 32**
Section 4: Non-Technical Constraints	**Page 33**
Special Thanks To	**Page 34**
References	**Page 35**

Table of Figures & Images

Figure 1, Moore's law	Page 4
Figure 2, Total Leakage power for 6-T SRAM	Page 5
Image 1, Samsung 105A SRAM	Page 6
Image 2, HP CACTI 4.20	Page 7
Figure 3, Understanding Leakage mechanism	Page 8
Figure 4, SRAM Stability	Page 9
Figure 5, Schematic for leakage path	Page 10
Figure 6, Major Leakage Mechanism	Page 11
Figure 7, Gate oxide Leakage	Page 12
Figure 8, Sub threshold leakage	Page 14
Figure 9, 6-T SRAM Memory cell	Page 15
Figure 9.2, SRAM Characterization	Page 16
Figure 10, Sense Amplifier	Page 17
Figure 11, Decoder	Page 18
Figure 12, Algorithm for UCI-SAW	Page 19
Image 3, UCI-SAW Screen shot	Page 20
Image 4, Data Page of UCI-SAW	Page 21
Image 5, Outputs file	Page 29
Figure 13, The breakdown of dynamic power for a 1KB SRAM	Page 30
Figure 14, The breakdown of leakage power for a 1KB SRAM	Page 30
Figure 15, The breakdown of total power for a 1KB SRAM	Page 31
Figure 16, The Comparison of results	Page 32

Section 1: Introduction, Back Ground and Description of Project

Abstract:

In the 21st century and inside the iPod era, memories are playing an essential role in most people's life and accessories. Static random-access memory (SRAM) is also a very a critical component in the wide range of microelectronics applications from consumer wireless to high-end workstation and microprocessor applications to portable MP3 players & all in one pocket organizers. The substantial increased demand for lighter, faster and cheaper portable electronic applications with longer battery life and therefore less need for recharge have fueled the need for new technologies that can give low standby power to the consumer while maintaining the high performance quality. [1, 2]. In this project, we propose a new procedure to calculate the power dissipation inside a SRAM and offer a new web application that could be a new frontier for all manufacturers and researchers in order to find power dissipation of different technologies of SRAM. Our approach will be a revolutionary model of calculation that is never been used before and could completely change the expectation and planning while designing a new SRAM. Decreasing transistor sizes has lead to a rapid increase not only of the total amount of static power dissipation, but also of the number of mechanisms causing static power. Probing logic gates for current and power evaluation used to be straightforward, and with switching power being the sole contributor we could even resort to estimates using switched capacitances, avoiding probing altogether.

The need for early design studies, which combines architectural timing simulation with power estimation, has become a critical part of design process especially considering that power is now a first class design constraint for portable devices.

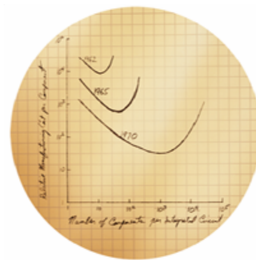


Figure 1

Back Ground:

Leakage power dissipation today significantly contributes to the total power dissipated in deep submicron VLSI chips. As leakage continues to increase in importance, accurate leakage power estimation is required to allow for good design trade-offs. This is especially true at higher design levels that are associated with a higher degree of design freedom, potentially leading to higher power and energy savings. On-chip SRAM memories have large sections that are idle for relatively long periods of time and, thus, dissipate considerable amounts of static power. Thus, sub threshold leakage remains the main contributor to total leakage. Figure 1 demonstrates the total leakage power for the 6-Transistor SRAM:

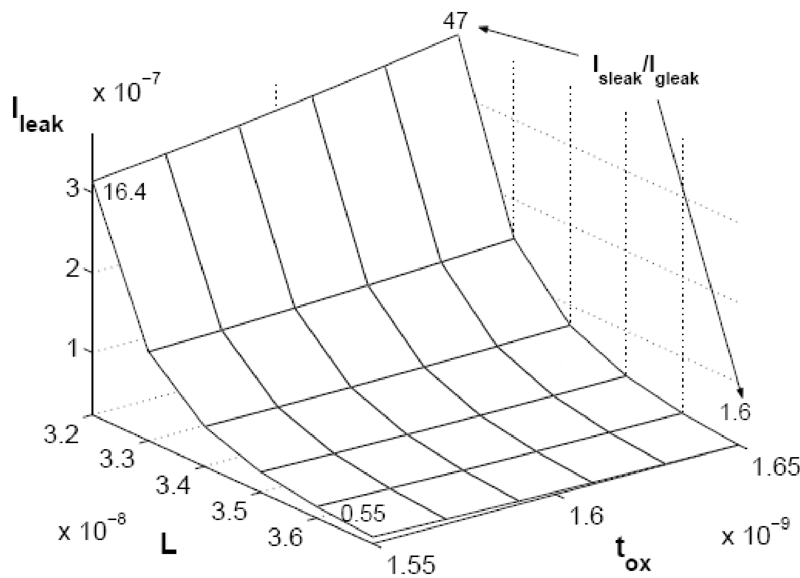


Figure 2
(Source: [7])

Static random access memory (SRAM) is a type of semiconductor memory. The word "static" indicates that the memory retains its contents as long as power remains applied, unlike dynamic RAM (DRAM) that needs to be periodically refreshed (nevertheless, SRAM should not be confused with read-only memory and flash memory, since it is volatile memory and preserves data only while power is continuously applied). SRAM is different from SDRAM, which stands for *synchronous DRAM* and is entirely different from SRAM, or with pseudo static RAM ([PSRAM]), which is DRAM configured to function, to an extent, as SRAM.

:

Image 1: SAMSUNG 105A SRAM

Common Models for Power estimation

Generally, there are two approaches for developing power models: analytical and empirical.

In analytical approach model, assumed parameters, relations, and formulas for the capacitance of important nodes help to calculate the power consumption.

In empirical model, result is obtained using circuit simulation often using the knowledge from the older technologies.

The good thing about analytical model is that it's fairly simple and quick, so very easy power consumption can be calculated; However due to high approximation and inaccuracy of the relations used in this approach, final results are usually up to 30% inaccurate.

Empirical way however, gives us very accurate information about the power consumption but it is extremely complicated. It is also an extremely time consuming process and it could take weeks before we can get any results. Usually in the middle this process, problems such as loss of data or change of environment could happen. In addition, by doing this, we cannot always use older technologies, which could be difficult, if we are dealing with a new technology.

Our Objective:

We are planning to develop a new model that is between two models that were mentioned before. This model, which is called simple circuit simulation based power modeling, combines an empirical and analytical approach. In this approach, we break the circuit to small blocks. Then we use the simplest cell of memory and simulate that simple circuit. This simple circuit is used like a brick to build bigger blocks of memory. In other words, once this simple cell is simulated, then we use our analytical knowledge to calculate the power consumption of a complete SRAM.

The major benefits of such approach is that while we get a real accurate results, close to 4% to be précised, we will spend a fraction of time to do the simulation since we are only simulating a simple cell.

Once we calculate an overall solution to obtain the power leakage of a SRAM, we will implement our technique in a website called UCISAW. This web site will allow any user to choose his desired technology and even upload their technology if it is not inside our data base and then within minutes get a graphical and numerical answer to the probability of power leakage in that specific SRAM technology. We are anticipating that this web site very quickly can replace HP CACTAI, which is a similar web site that uses purely analytical models to calculate the power consumption. Using UCISAW, could help designer have a better and more accurate design, and allow manufacturers to get up to 15% better yield.

Image2: HP CACTI 4.20

Section 2: Detailed description of work performed

Part1: Investigation and Information gathering

Understanding the specific leakage mechanisms that govern the cell and array leakage as a function of temperature and applied voltage is crucial to controlling the SRAM array standby power. For the present discussion, the leakage mechanisms are classified as being either parametric (intrinsic) or defect-related in nature. The SRAM array parametric standby leakage contributors include well isolation leakage [3], sub threshold device leakage [4], gate-oxide tunneling [5], reverse-bias diffusion leakage [6], and gate-induced drain leakage (GIDL) [7,8] for both n-FET and p-FET devices. In this paper, we review each of the parasitic components and their impact on the overall cell and array standby power. Therefore, to begin our process we are going to review and research every section of a 6-T SRAM and find and verify every parameter that causes the leakage.

The cell designs nodes are similar in layout, each containing a segmented polysilicon word line strapped with the first level of metal and a shared ground contact between two adjacent cells. **Figure 3(a)** shows the cell layout for the 0.13-µm design optimized to achieve density and yield. The designs of the polysilicon, active silicon, and first level of metal were optimized using optical proximity correction (OPC) based on the aerial image modeling [10] shown in **Figure 3(b)**. The final cell design provides RAM density and preserves compatibility with the base logic process. **Figure 3(c)** is an SEM image taken of the cell region just prior to silicide processing; it shows the patterned diffusion and polysilicon regions in the cell that correspond to the drawn shapes in **Figure 3(a).**

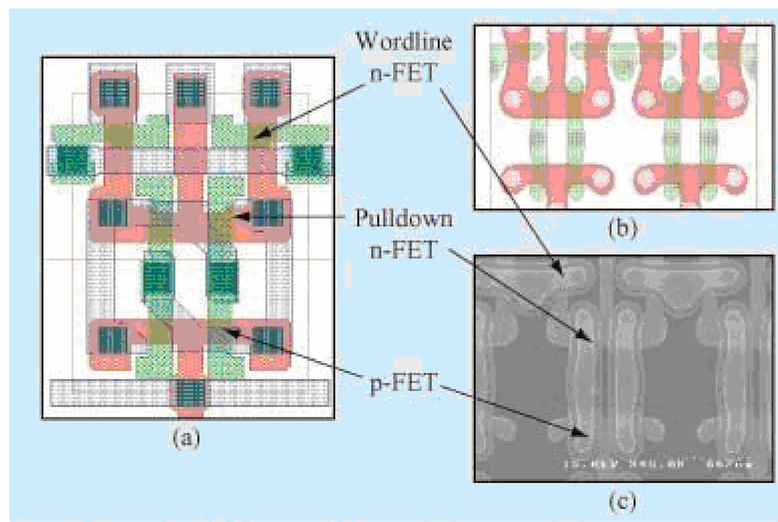


Figure 3: Understanding Specific Leakage Mechanism (From IEEE J. Solid-State Circuits 36)

SRAM cell
(a) Design; (b) simulation of cell; (c) SEM

Figures 4(a) and **4(b)** respectively show the measured and modeled (SPICE) butterfly curves for the cell, with two different threshold voltages at 25°C for the 0.18-µm cell design. The simulation and measurements were done with the word line held at high voltage (i.e., V_{DD}) and either internal node was ramped while the opposite node voltage was measured. The room-temperature static noise margin was measured to be ~500 mV for the high-V_t case and ~250 mV for the higher-performance devices with lower threshold.

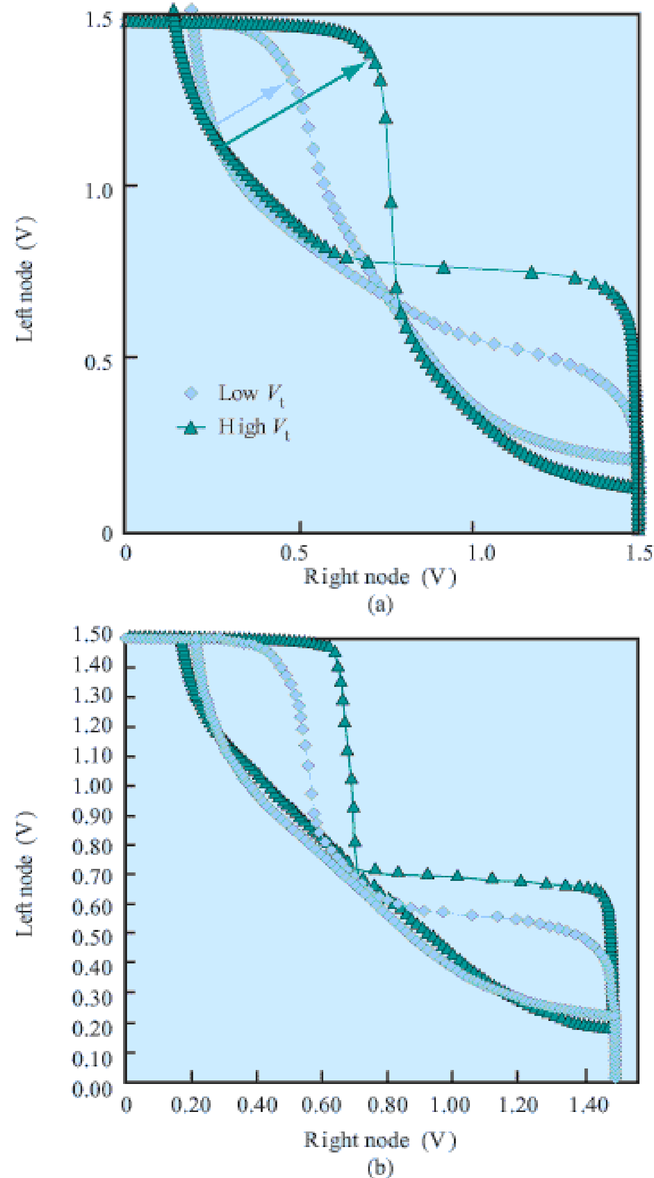


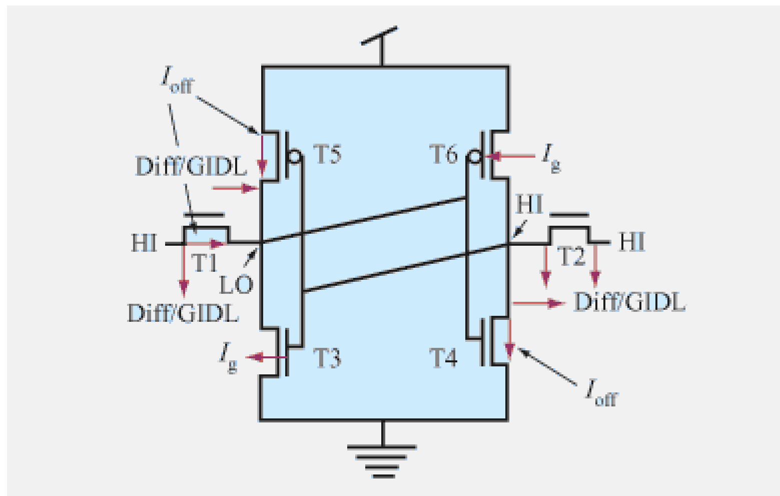
Figure 4: SRAM Stability
(Source: IEDM Tech. Digest, pp. 671–674 (1999))

To estimate the leakage associated with an array of cells, we first define the dominant leakage mechanisms and critical paths within the simple cell (the brick to our block). **Figure 5** shows schematically the significant parametric cell leakage paths and

mechanisms operating in the 6 Transistors SRAM. By accounting for each leakage, mechanism and calculating the leakage to the given cell dimensions, the total cell and/or array leakage can be calculated effectively and the expected array leakage can be estimated very adequately. In Figure 3, for the arbitrary state chosen, the internal node on the left side of the latch is maintained at ground while the node on the right is held at highest voltage V_{DD} by the operation of the cross-coupled latch.

The intent of this part of our research is to describe briefly the critical parametric leakage sources within the cell derived from this latched configuration corresponding to memory array standby mode. We need to do this because it is usually impossible to choose a specific state for all nodes inside the SRAM.

So as mentioned, below are the significant leakage paths in the 6T cell:



Schematic of leakage paths in a six-transistor SRAM cell on bulk silicon. HI = supply voltage (V_{DD}); LO = ground; Diff = diffusion leakage.

Figure 5:

(Source: Electron Device Lett. 8, 515)

The five dominant parametric mechanisms to be addressed are threshold voltage optimization, gate tunneling leakage, sub threshold leakage, reverse-bias diffusion leakage, and gate-induced drain leakage (GIDL).

Major Leakage Mechanism

Great deal of research is done on the leakage of different designs and after close focus on the 6-T design; the following are recognized as the most common mechanism of leakage [11]:

I1: pn junction leakage

I2: Sub threshold S-D leakage

I3: DIBL and contribution from SCE

I4: Gate-Induced-Drain Leakage (GIDL)

I5: Punch through current

I6: Narrow width effects

I7: Gate oxide leakage

I8: Hot carrier injection

Figure 6 demonstrates these 8 major mechanisms:

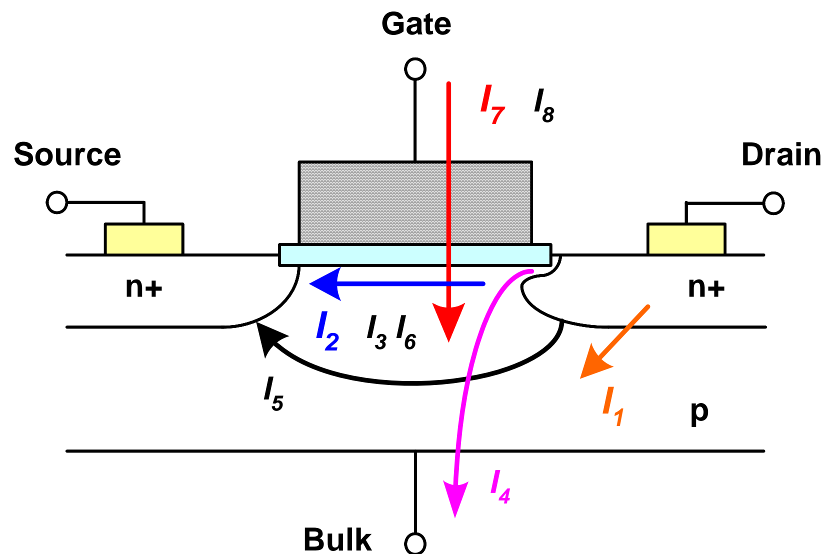


Figure 6: Major leakage mechanism in a transistor

Some of these mechanisms have a much more decisive impact on the total leakage of the transistor so in order to save time and still stay accurate in our results we narrowed our research and simulation to the three most impactful sources of the leakage in the transistor:

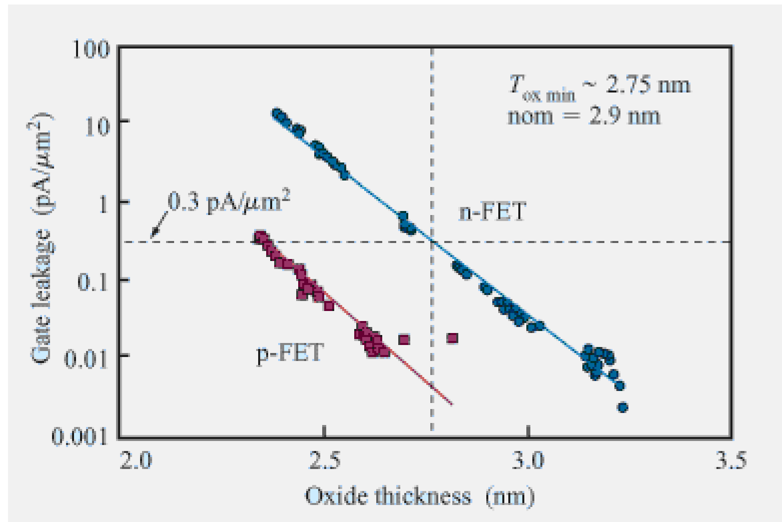
- Gate oxide leakage (I_g)
- Sub threshold leakage (I_{sub})
- pn junction reverse-bias leakage

Gate leakage

First, we tried to find an approximation or solution as how to derive the gate related leakage. After reading multiple papers about gate leakage on IEEE, the measured tunneling current through the gate dielectric turned out to be a function of oxide thickness for n-FET and p-FET devices. Gate-oxide tunneling leakage is observed to be roughly 1.5 orders of magnitude higher for the n-FET at the thickness and voltage conditions of interest. The gate leakage can become a significant contributor to the room-temperature cell leakage at thicknesses below 2.7 nm. For the arbitrary latched state chosen for Figure 4, the gate tunneling leakage mechanism is active for the n-FET (T3) and p-FET (T6) sites shown in **Figure 5**. For the purpose described in this paper, the leakage is generally found to be adequately modeled for a given voltage as a function of gate-oxide thickness from the empirical relationship

$$I_g(t_{ox}) = A_0 \exp(-B_0 t_{ox})$$

For both n-FETs and p-FETs. Because this mechanism is governed by quantum-mechanical tunneling, this mechanism is virtually temperature-independent; while other leakage mechanisms dominate at higher temperatures; this mechanism was found to establish the minimum gate-oxide thickness for the technology based on the established lower-temperature leakage targets.



gate-oxide tunneling current as a function of oxide thickness for both n-FET and p-FET.

Figure 7:

(Source: IEEE Trans. Components, Hybrids, Manuf. Technol. 12)

V_t and subthreshold leakage

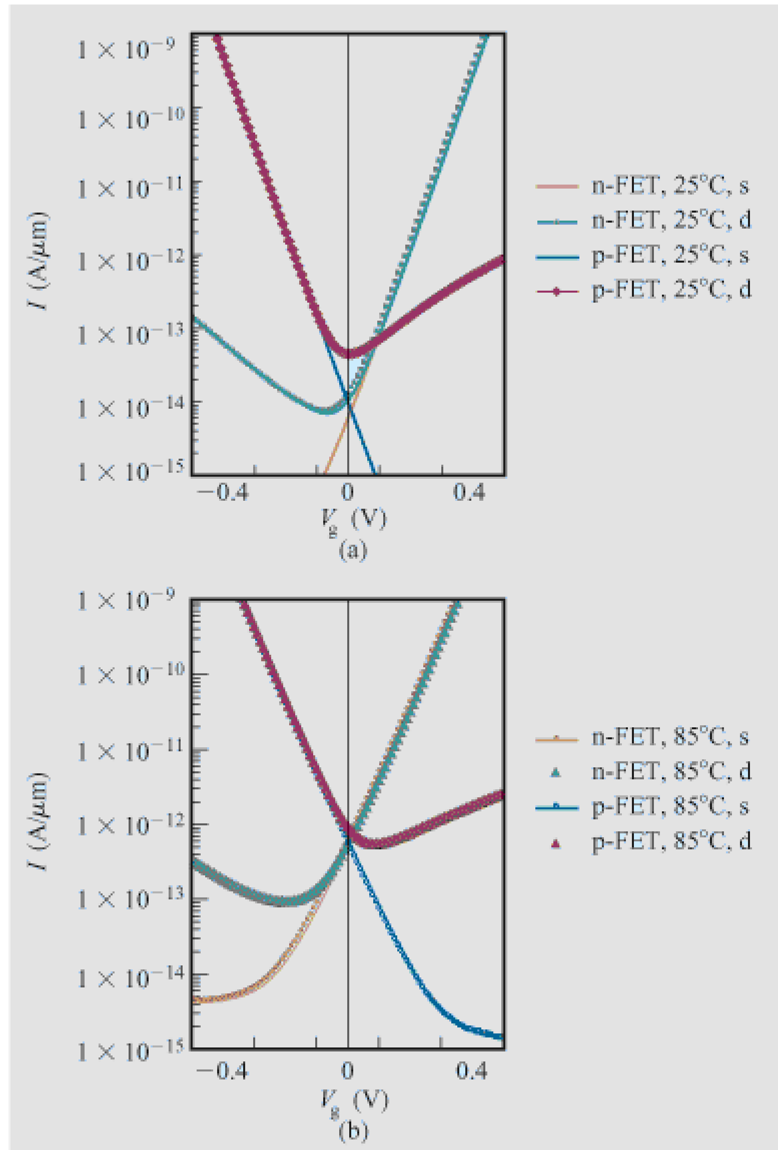
Next parameter to find is the Vt and the sub threshold voltage leakage. We can see the *I*–*V* characteristics with V_{ds} at 1.5 V at 25°C and 85°C in **Figures 8(a)** and **8(b)**. Unlike gate-oxide tunneling leakage, sub threshold device off-current leakage is strongly temperature-dependent and is typically the dominant leakage mechanism at higher temperatures. **Figure 8** shows three transistors in which this mechanism is actively contributing to the standby leakage when the SRAM array is in the standby state. Since it is most common for the bit lines to be held high (at V_{DD}) in standby mode, this is the mode shown for the sake of discussion. However, it is worth pointing out that if the bit lines were held low (at ground), there would still be three devices in the cell contributing to the off-state leakage, since the internal nodes of the SRAM cell are held in opposite states. The off-state leakage can be adequately characterized given the sub threshold slope parameter (B_1), an extracted parameter (A_1) and threshold voltage (V_t) for both the n-FET and the p-FET with the following relationship:

$$I_{off}(V_t, T) = C(T) A_1 \exp(B_1 V_t),$$

Where $C(T)$ is expressed as

$C(T) = 10^{(V_t/S) - [V_t - (T - 298)\gamma]/(T/298)S]}$, & A1 & A2 are the extracted parameters from the BSIM 4.0.

Where S is the sub threshold slope, γ is the slope of the V_t as a function of temperature and T is the temperature in degrees Kelvin.



subthreshold characteristics of the n-FET and p-FET at (a) 25°C and (b) 85°C (s = source, d = drain).

Figure 8:
(Source: IBM Ultra slim chip design library)

It is well known that the statistical variation in V_t will become an increasing concern as devices continue to scale [9, 10]. This is due not only to the physical dimension tolerances but also to statistical variations in channel dopant associated with the reduction in channel area. This variation can be compounded in the SRAM cell by V_t variations associated with overlay tolerances and corner-rounding effects due to aggressive scaling to achieve maximum density for the cell. Because of the exponential relationship of I_{off} with V_t, the contribution of the devices in the array with lower threshold voltage must be accounted for in calculating the overall array leakage. The array leakage increase that is associated with the variation in the standard deviation of V_t, such as , $\sigma(V_t)$, can be estimated by means of the following equation:

$$I_{ddk}[V_{td}, \sigma(Vt)] = \frac{\int_{V_{t1}}^{V_{t2}} \frac{1}{\sqrt{2\pi}\sigma(V_t)} e^{\{[-1/2\sigma(V_t)^2](V_{td}-\overline{V}_t)^2\}} [A_1 e^{(-B_1 V_{td})}] dV_{td}}{A_1 e^{(-B_1 \overline{V}_t)}},$$

Where $\overline{V}_t$ is the V_t mean, V_{td} is the device V_t, and A_1, B_1 are defined above.

Component Characterization

The process of reading and writing from and to a memory cell is accomplished by the pre-charge stage of all bit-lines (BL) to *Vdd* and then selecting the row and column using Row/column decoders for the given address. A word line and a pair of BL/BL are then selected to connect memory cell to sense amplifier.
It is clear that nodal capacitance is needed to get the right value of the dynamic power dissipation. For every array component, we chose to extract those nodal capacitances using a circuit simulator that establishes the operating point and DC capacitances. [12]

Memory cell

Dynamic power dissipation of a read is due to BL discharging currents through the accessed cell, while write dynamic power is due to BL discharging currents
Through the write circuits. During a read or write, cycle not only the selected memory cell dissipates power but other cells, sharing the same word line with the selected one, also dissipate power.

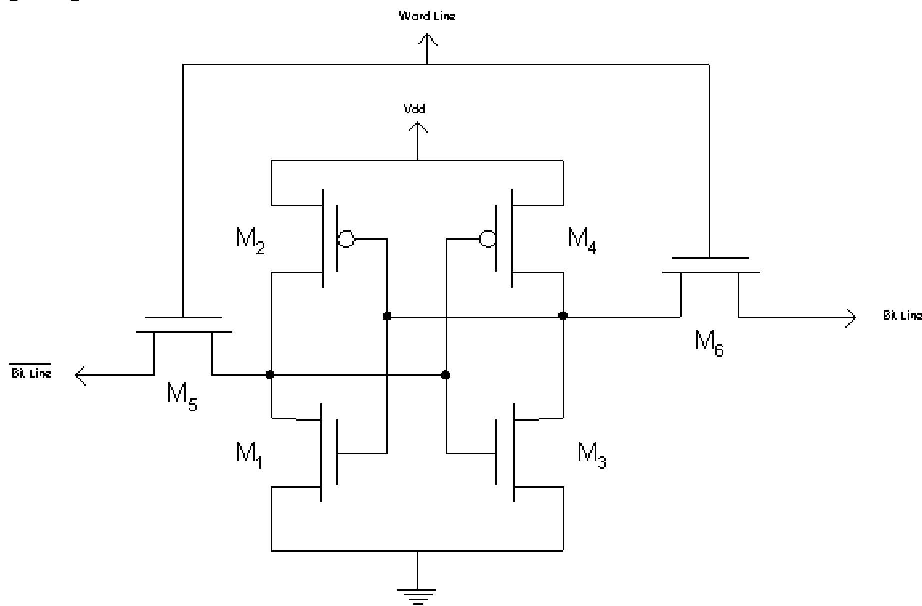


Figure9: 6-T SRAM memory cell

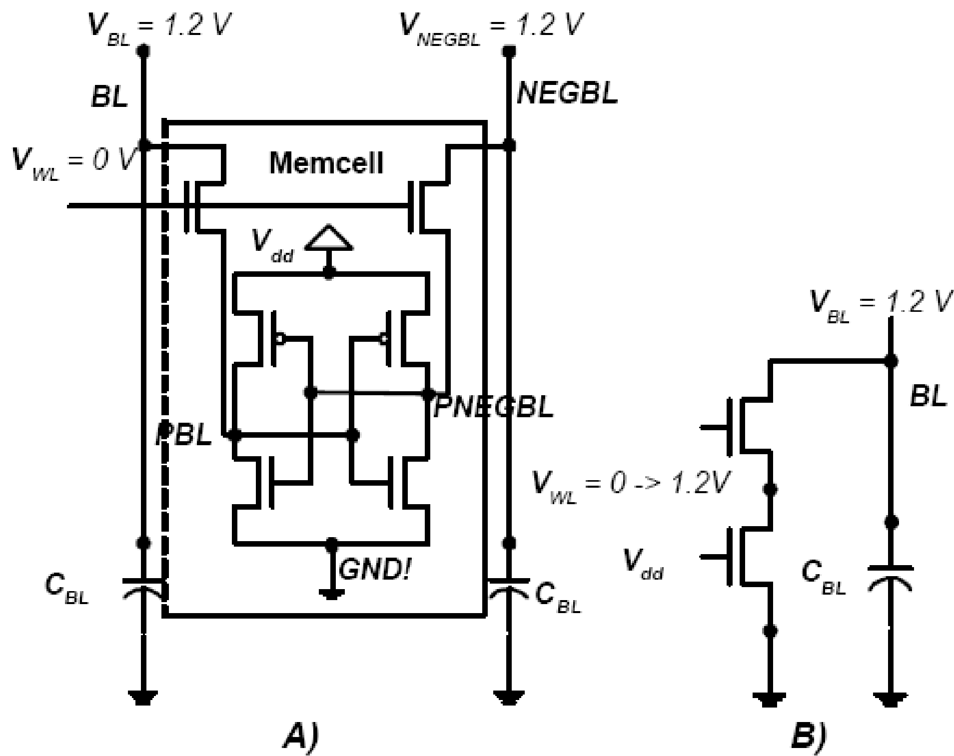


Figure 9.2: (a) Characterization of a SRAM cell, (b) Hspice configuration for Power estimation

(Source: [12])

The characterization for a memory cell is done by performing a circuit-level DC simulation for a single cell connected to a pair of BL and ST to quantify several leakage components.

The dynamic power dissipation can be accurately estimated using

$$P_{switching} = \alpha\, C_L\, V_{dd}^2\, f_{clk} = \alpha\, C_L\, \Delta V\, V_{dd}\, f_{clk}$$

Given bit line capacitance, clock frequency, bit line voltage swing and supply voltage.

The 'passive read' power can also be estimated by using [12]:

$$\Delta V_{BL}^{pass.read} = \Delta T_{wordline} \frac{I_{bitline_discharge}}{C_{\ BL}}$$

Sense Amplifier

Power dissipation of a sense amplifier consists of leakage and dynamic components. Dynamic power is due to the current that discharges BL of the SA from (Vdd- Vsens) to zero.

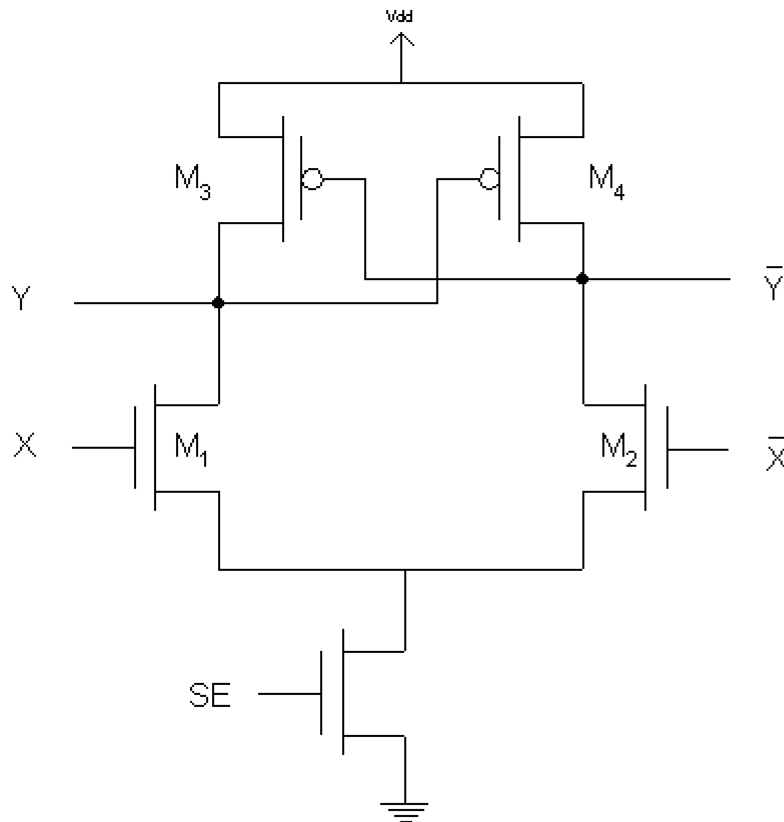


Figure 10: Cross-Coupled Sense Amplifier

Address Decoders

In this paper, we use 2-4 row/column decoders of the architecture that is similar to the one used in CACTI [2] for cache delay estimation.
Figure below shows a simple 2-4 row/column decoder:

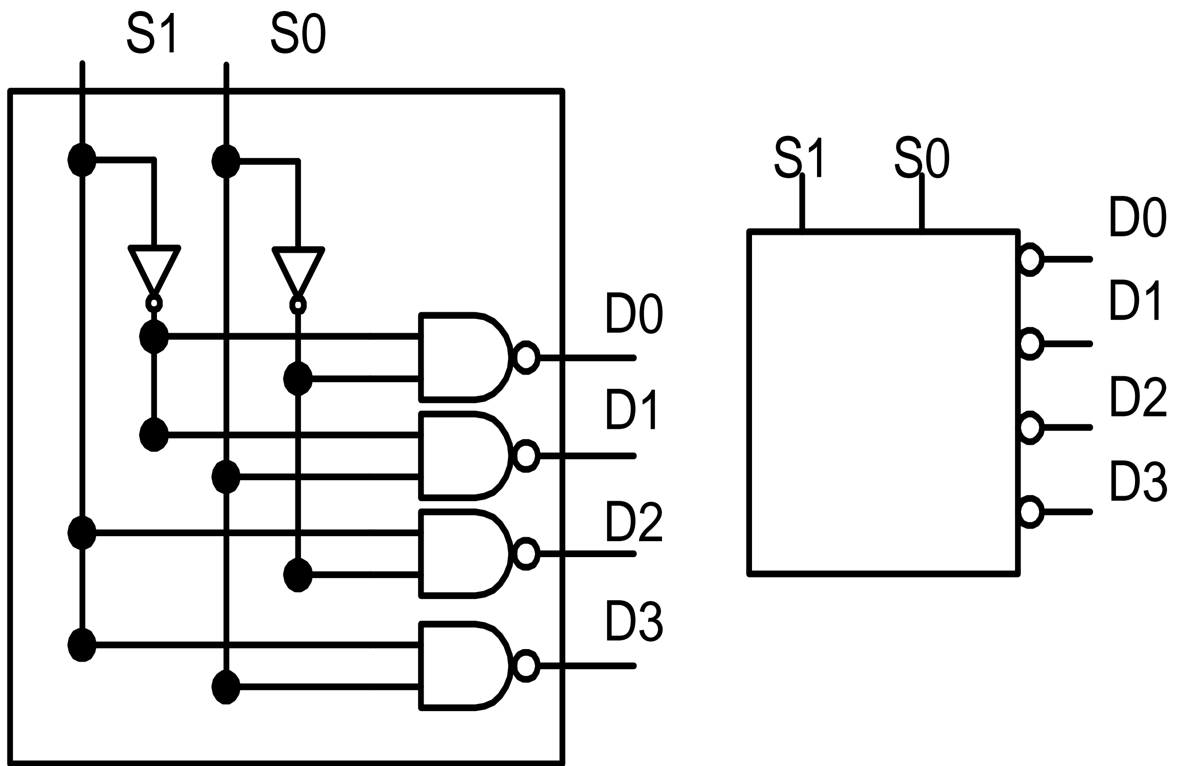


Figure 11: 2 to 4 decoder

Part 2: Algorithm, Execution and Results

Once all major components of a SRAM cell were identified, an algorithm was put in place to execute the hybrid model, in a similar way to the operation of the HP CACTI.

We needed an interface that received the parameter data in regards to the technology from the user, and then some sort of an operating system that activates the simulation using these data, and then a way to send the data back to the user.

In doing so, we needed a process in place that can help save time as well as maximize the performance so we can send the results back to the user, thus the following procedure was developed:

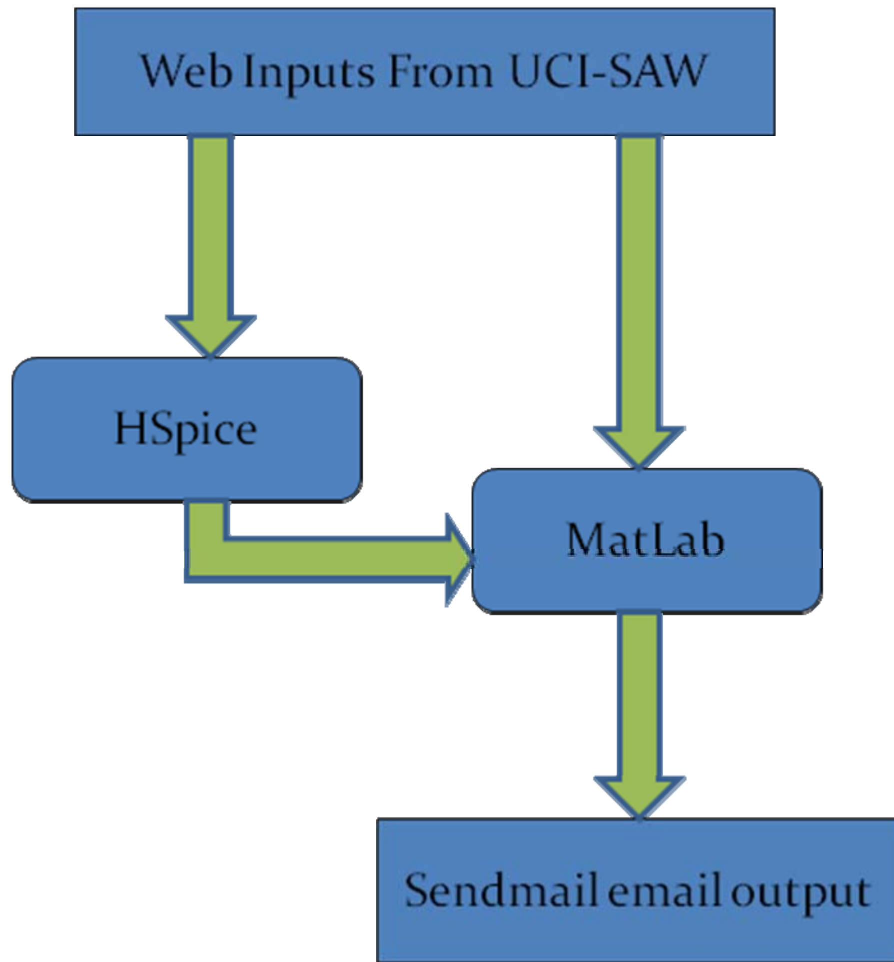


Figure 12: an algorithm of process for UCI-SAW

As it is shown in the algorithm, we needed four components.

First, we needed an interface between user and model, which led to the creation of the UCI-SAW website. Then we needed an operation system, which for all practical purposes Perl programming was used to develop this part of the project. Perl placement in the process is identified with green arrows in the **Figure 12**.

Simulation and calculation was done using the HSPICE and Mat lab.

After the simulation was done, the results were sent to the user using the Perl again.

UCI-SAW website

Located on the Malibu server, a web site is designed at the location *wscl.eng.uci.edu\ucisaw2.* This is the interface between the user and the model program.

Image 3 demonstrates UCI-SAW website:

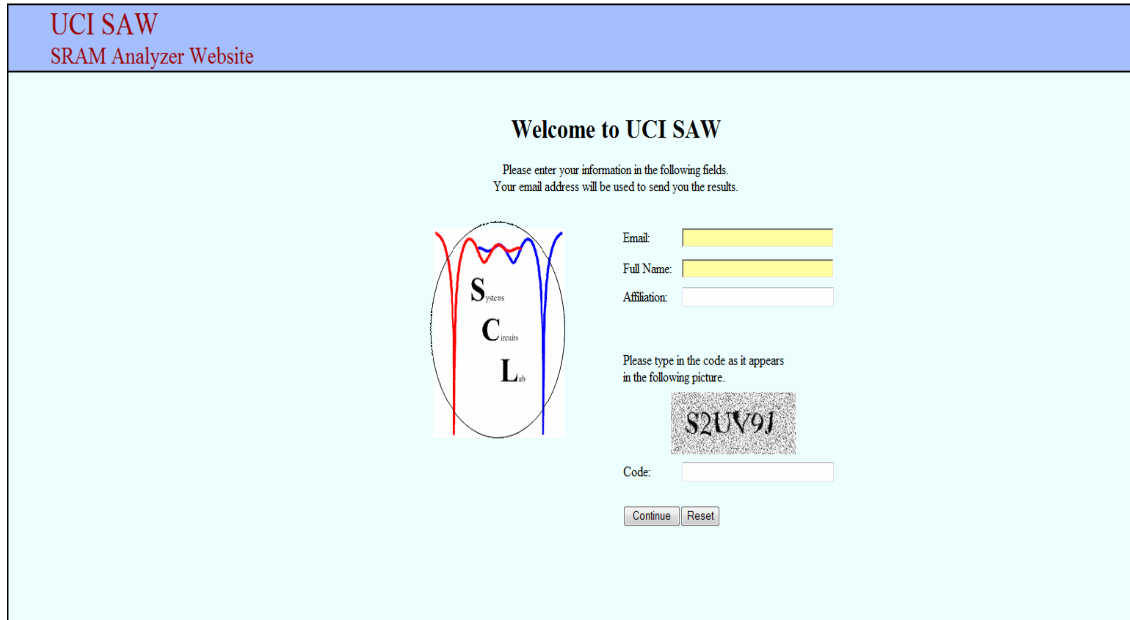


Image 3: UCI-SAW screen shot

Once personal information was entered, the user was directed to the data page where he can enter the parameters for the technology that he wants to do a power simulation for.

Here the user gets to enter information such as size, number of banks, word line and temperature. In addition, there is a capability of uploading the ptm file, which would be a very useful option for the designers who have their own technology files.

There is also a tab for advance option, which allows the user to enter more detailed information in regards to their technology.

Image 4 is a screen shot of the data page of the web site:

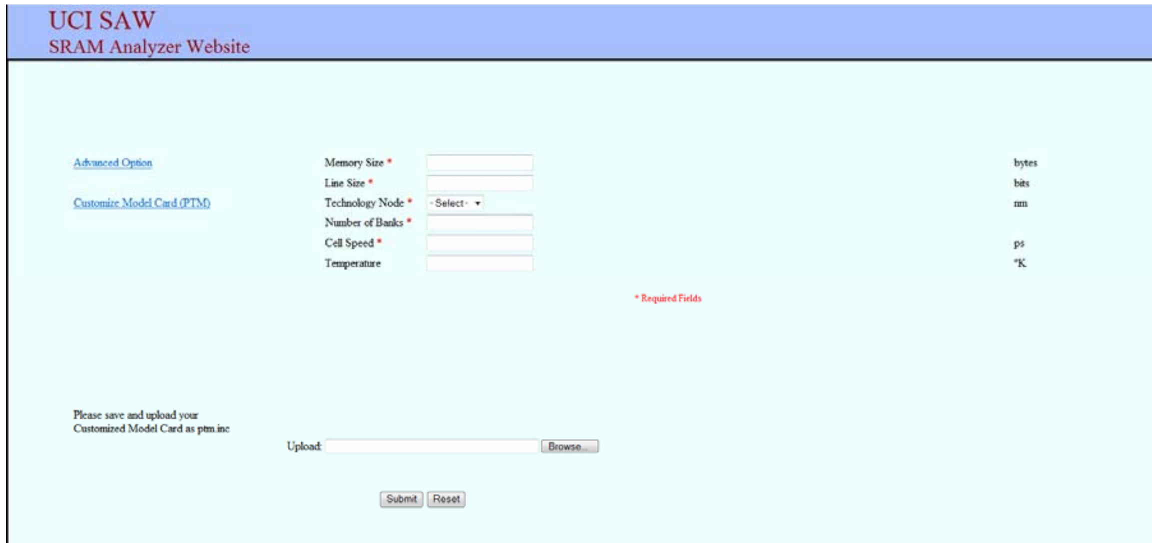


Image 4: Data page of the UCI-SAW website.

Once all of this information is entered into the website, two files are created inside the website in a sub directory named after the user email address:

First file, which is named personal.txt, will contain user's information and the second file that is named data.txt is containing parameters entered by the user. After all this is done, it is time for Perl file to be called and start the simulation process.

Perl

What is Perl?

According to www.perl.org , Perl, sometimes referred to as Practical Extraction and Reporting Language, is an interpreted programming language with a huge number of uses, libraries and resources. Arguably, one of the most discussed and used languages on the internet; it is often referred to as the Swiss army knife, or duct tape, of the web.

Perl was first brought into being by Larry Wall circa 1987 as a general-purpose UNIX scripting language to make his programming work simpler. Although it has far surpassed his original creation, Larry Wall still oversees development of the core language, and the newest version, Perl 6.

How did we use perl?

We used a Perl file to send the parameter from the data.txt to the Hspice, call the Hspice file as well as the Mat lab file, and initiate the simulation. In addition, once the simulation is completed the Perl generates an email to the user that contains the outputs from the Mat lab calculation information. Perl can be optimized to set an automation procedure in a way that, right after creation of the data.txt, the process starts to work. Further optimization will be the subject for additional research.

Below is the perl file used for the UCI-SAW project

```
#!/usr/bin/perl
use CGI ':standard';
#opens and extracts data values from web site.
open (SAW_DEF, "data_values.txt") or die "I couldn't get at log.txt";
            @data = <SAW_DEF>;

@memory_size = @data[0];

@line_size = @data[1];

@technology_node = @data[2];

@number_of_banks = @data[3];

@cell_speed = @data[4];

@temperature = @data[5];

######these prints can be removed
print "@memory_size\n";
print "@line_size\n";
print "@technology_node\n";
print "@number_of_banks\n";
print "@cell_speed\n";
print "@temperature\n";

close SAW_DEF;

#opens and extracts personal data from web site.
open (SAW_PER, "personal_values.txt") or die "I couldn't get at log.txt";
            @pers = <SAW_PER>;
@email_address = @pers[0];


##########this print can be removed
```

```
print "@email_address\n";

close SAW_PER;
use MIME::Lite;
#@email_address = 'uci.saw@gmail.com';

$msg = MIME::Lite->new(From    => 'uci.saw@gmail.com',
                 To      => "@email_address",
                 Subject => 'SRAM UCI SAW',

                 Type    => 'multipart/mixed');
$msg->attach(Type        => 'image/jpeg',
         Path        => '/home/sawproject/testfolder/my_file.jpg',

         Filename    => 'my_file.jpg');
$msg->attach(Type        => 'TEXT',

         Data        => 'Your Requested SRAM Simulation Results from UCI SAW Project');
$msg->send( );           # default is to use sendmail(1)
```

Hspice & Matlab

Hspice was used to do the simulation part of the project.

Perl calls the Hspice & Matlab files, and Hspice starts the simulation for a single SRAM cell using simple DC simulation and then sends the extracted parameters to calculate the dynamic and leakage power as well as total power dissipation for the specific technology.

Below is the HSPICE file used for the project:

```
****************************************************************************
****************************************************************************

* SRAM cell

******************** Parameters ************************************

* Parameters

.param pvdd=    1.2
.param pRise =  40ps
.param pFall =  40ps
.param pCycle = 500ps
.param pCycle2= '2*pCycle'
.param pCycle4= '4*pCycle'
.param pCycle8= '8*pCycle'
.param pDelay1= 'pCycle-20e-12'
.param pDelay5= '5*pCycle-20e-12'
.param pWidth=  'pCycle-20e-12'
.param pLow =    0.0
.param pHigh =  'pvdd'

* voltage sources

vGnd          gnd!      0 dc 0
vdd        vdd!      0 dc pvdd
vPhi       Phi       0 PU pLow pHigh pDelay1 pRise pFall pWidth pCycle2
vWL        WL                0 PU pLow pHigh pDelay1 pRise pFall pWidth pCycle2
vdataIn    dataIn    0 PU pLow pHigh pDelay5 pRise pFall pWidth pCycle8
vdataInBar dataInBar 0 PU pLow pHigh pDelay1 pRise pFall pWidth pCycle8
vwrite     write     0 dc 0

******************* Controls **********************************

.tran 20ps 4ns UIC sweep DATA= cornerInfo
.IC v(L1)=0 v(R1)=pvdd
.dc vds 0 5 .1 vgs 0 5 1

.Data cornerInfo MER
 FILE='./data.txt' pvdd=1 pvta_sl=2 pvta_nl=3
.ENDDATA

.option post

* measure statements
.param tdval=5n tstop=200n
.meas tran tdelay trig v(1) val=2.5 td=tdval rise=2
+targ v(2) val=2.5 fall=2
.meas tran vmax max v(2) from=tdval to=tstop
.meas tran vmin min v(2) from=tdval to=tstop
.meas tran trise trig v(2) val='vmin+.1*vmax' td=tdval
+rise=1 targ v(2) val='.9*vmax' rise=1
```

```
.meas tran tfall trig v(2) val='.9*vmax' td=tdval
+fall=2 targ v(2) val='vmin+.1*vmax' fall=2
.measure tran inv_delay TRIG v(IN) VAL='supply/2' TD=20ns RISE=1
+ TARG v(OUT) VAL='supply/2' TD=20ns FALL=1
.MEAS TRAN max_current MAX I(Vdd)


********************* circuit ************************************

* global nodes

.global vdd!
.global gnd!


* instance cell sram1 used for Read write analysis
* The cell initially stores '0' on node L1
* read delay, read a 'zero' on node L1
* delay from wl=pvdd/2 to BLT1 drop to pvdd*0.85

.param pvta_sl=0;
.param pvta_nl=0;

xSRAM1  L1  R1  BLT1 BLC1 Phi WL dataIn dataInBar write sram delvton3=pvta_nl delvton4=pvta_sl

*********************** read margin testbench**************************************************
Vdd vdd 0 2.5
Vin VIN 0 dc 2.5
xinv Vin Vout vdd 0 my_inv
M5 vblbar vdd Vout 0 nmos l=0.24u w=0.36u
Vblbar vblbar 0 2.5
.dc Vin 0 2.5 0.01
.option post
.END


************************* Write margin testbench******************************
Vdd vdd 0 2.5
Vin VIN 0 dc 2.5
xinv Vin Vout vdd 0 my_inv
M6 vbl vdd Vout 0 nmos l=0.24u w=0.36u
Vbl vbl 0 0
.dc Vin 0 2.5 0.01
.option post
.END


********************** SRAM subcircuit *****************************

.macro sram L R BLT BLC Phi WL dataIn dataInBar write delvtop1=0 delvtop2=0 delvton1=0 delvton2=0 delvton3=0 delvton4=0

  * 1ohm resistor to measure vdd current for single cell
   Rdummy vdd! vdd1 1



  * Write and precharge circuitry

  mP0  BLC          Phi         vdd1   vdd1          pmos l=0.032u w=2.304u
  mP1  BLT          Phi         vdd1   vdd1          pmos l=0.032u w=2.304u
  mN0  BLT          write       net43  gnd!          nmos l=0.032u w=4.608u
  mN1  net43        dataInBar gnd!   gnd! nmos l=0.032u w=4.608u
  mN2  BLC          write    net49  gnd!   nmos l=0.032u w=4.608u
  mN3  net49        dataIn    gnd!   gnd!   nmos l=0.032u w=4.608u


  * 6T SRAM cell

  mpr R L  vdd1 vdd1 pmos l=0.032u w=0.108u delvto=delvtop1
  mpl L R  vdd1 vdd1 pmos l=0.032u w=0.108u delvto=delvtop2
```

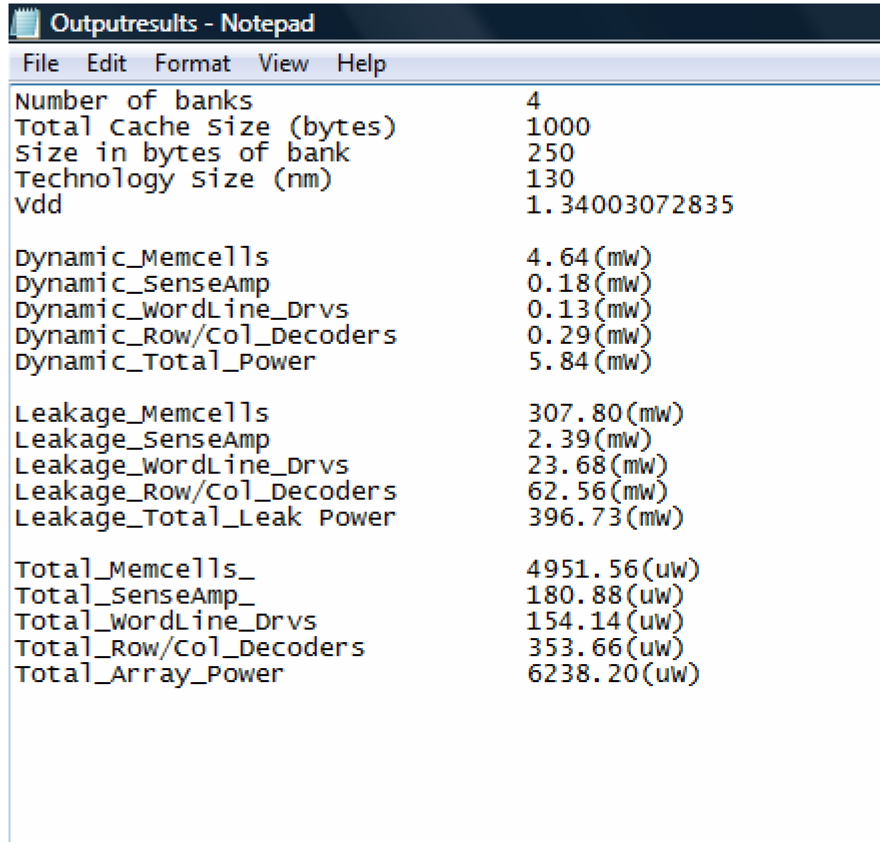
```
mnr R L  gnd! gnd! nmos l=0.032u w=0.108u  delvto=delvton1
msr R WL BLC  gnd! nmos l=0.032u w=0.072u delvto=delvton2
mnl L R  gnd! gnd! nmos l=0.032u w=0.108u  delvto=delvton3
msl L WL BLT  gnd! nmos l=0.032u w=0.072u delvto=delvton4


* Latch Type Sense Amplifier

msN1 vdataOut    BLT        com     gnd! nmos l=0.032u w=4.608u
msN2 vdataOutBar BLC        com     gnd! nmos l=0.032u w=4.608u
msP3 vdataOut    vdataOutBar vdd1    vdd1 pmos l=0.032u w=2.304u
msP4 vdataOutBar vdataOut    vdd1    vdd1 pmos l=0.032u w=2.304u
mSEN com        EN          gnd!    gnd! nmos l=0.032u w=4.608u


* 2 to 4 row-decoder

mdP1 vdd1       A0        invOutput1 pmos l=0.032u w=2.304u
mdN1 invOutput1  AO        gnd!      nmos l=0.032u w=4.608u
mdP2 vdd1       A1        invOutput2 pmos l=0.032u w=2.304u
mdN2 invOutput2  A1        gnd!      nmos l=0.032u w=4.608u
mdN3 vdd1       invOutput1 D0        nmos l=0.032u w=4.608u
mdN4 D0         invOutput2 gnd!      nmos l=0.032u w=4.608u
mdN5 vdd1       invOutput2 D1        nmos l=0.032u w=4.608u
mdN6 D1         A0        gnd!      nmos l=0.032u w=4.608u
mdN7 vdd1       A1        D2        nmos l=0.032u w=4.608u
mdN8 D2         invOutput1 gnd!      nmos l=0.032u w=4.608u
mdN9 vdd1       A1        D3        nmos l=0.032u w=4.608u
mdN10 D3        A0        gnd!      nmos l=0.032u w=4.608u


* 2 to 4 col-decoder

mdP3 vdd1       A0        invOutput1 pmos l=0.032u w=2.304u
mdN11 invOutput1  AO       gnd!      nmos l=0.032u w=4.608u
mdP4 vdd1       A1        invOutput2 pmos l=0.032u w=2.304u
mdN12 invOutput2  A1       gnd!      nmos l=0.032u w=4.608u
mdN13 vdd1      invOutput1 D0        nmos l=0.032u w=4.608u
mdN14 D0        invOutput2 gnd!      nmos l=0.032u w=4.608u
mdN15 vdd1      invOutput2 D1        nmos l=0.032u w=4.608u
mdN16 D1        A0        gnd!      nmos l=0.032u w=4.608u
mdN17 vdd1      A1        D2        nmos l=0.032u w=4.608u
mdN18 D2        invOutput1 gnd!      nmos l=0.032u w=4.608u
mdN19 vdd1      A1        D3        nmos l=0.032u w=4.608u
mdN20 D3        A0        gnd!      nmos l=0.032u w=4.608u



* 64 more transistors on each side of the bit line to represent the load of the other sram cells

*64 T on BLC

mBLCN0  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN1  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN2  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN3  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN4  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN5  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN6  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN7  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN8  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN9  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN10 BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN11 BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN12 BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN13 BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN14 BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN15 BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
```

```
mBLCN16  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN17  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN18  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN19  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN20  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN21  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN22  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN23  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN24  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN25  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN26  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN27  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN28  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN29  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN30  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN31  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u

mBLCN32  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN33  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN34  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN35  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN36  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN37  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN38  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN39  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN40  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN41  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN42  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN43  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN44  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN45  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN46  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN47  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u

mBLCN48  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN49  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN50  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN51  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN52  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN53  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN54  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN55  BLC gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLCN56  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN57  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN58  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN59  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN60  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN61  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN62  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLCN63  BLC gnd! vdd1 gnd! nmos l=0.032u w=0.072u


*64 T on BLT


mBLTN0  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLTN1  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLTN2  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLTN3  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLTN4  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLTN5  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLTN6  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLTN7  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
mBLTN8  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLTN9  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLTN10 BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLTN11 BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
mBLTN12 BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
```

```
  mBLTN13 BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN14 BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN15 BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u

  mBLTN16  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN17  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN18  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN19  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN20  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN21  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN22  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN23  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN24  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN25  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN26  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN27  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN28  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN29  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN30  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN31  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u

  mBLTN32  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN33  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN34  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN35  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN36  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN37  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN38  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN39  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN40  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN41  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN42  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN43  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN44  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN45  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN46  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN47  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u

  mBLTN48  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN49  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN50  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN51  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN52  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN53  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN54  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN55  BLT gnd! gnd! gnd! nmos l=0.032u w=0.072u
  mBLTN56  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN57  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN58  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN59  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN60  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN61  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN62  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u
  mBLTN63  BLT gnd! vdd1 gnd! nmos l=0.032u w=0.072u

.eom


************************ include files ********************************

.inc ./bsim4_32nm_bulk_models.inc


.end
```

Finally, when results were created, the perl program would send the matlab output file, which is a text file, to the user.

Below is a screen shot of an example simulation:




```
Number of banks                   4
Total Cache Size (bytes)          1000
Size in bytes of bank             250
Technology Size (nm)              130
Vdd                               1.34003072835

Dynamic_Memcells                  4.64(mW)
Dynamic_SenseAmp                  0.18(mW)
Dynamic_WordLine_Drvs             0.13(mW)
Dynamic_Row/Col_Decoders          0.29(mW)
Dynamic_Total_Power               5.84(mW)

Leakage_Memcells                  307.80(mW)
Leakage_SenseAmp                  2.39(mW)
Leakage_WordLine_Drvs             23.68(mW)
Leakage_Row/Col_Decoders          62.56(mW)
Leakage_Total_Leak Power          396.73(mW)

Total_Memcells_                   4951.56(uW)
Total_SenseAmp_                   180.88(uW)
Total_WordLine_Drvs               154.14(uW)
Total_Row/Col_Decoders            353.66(uW)
Total_Array_Power                 6238.20(uW)
```

Image 5: The outputs file to be sent to a user

In order to an analysis and some verification we ran a test on a 130 nano meter technology SRAM, with the VDD=1.2 V and the CLK frequency of 400 MHZ and temperature of 300 K, and then below we are showing the results of this simulation.

In **Figure 13**, the dynamic power of this SRAM for a 1K SRAM is broken down into different components. **Figure 14** shows the breakdown of the leakage power for the same 1K SRAM .

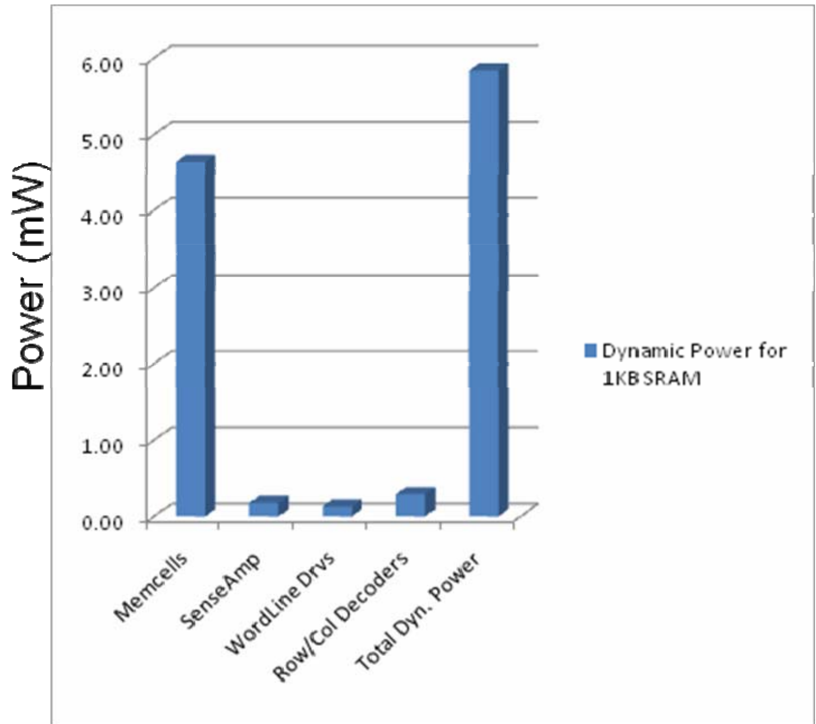


Figure 13: The breakdown of dynamic power for a 1KB SRAM

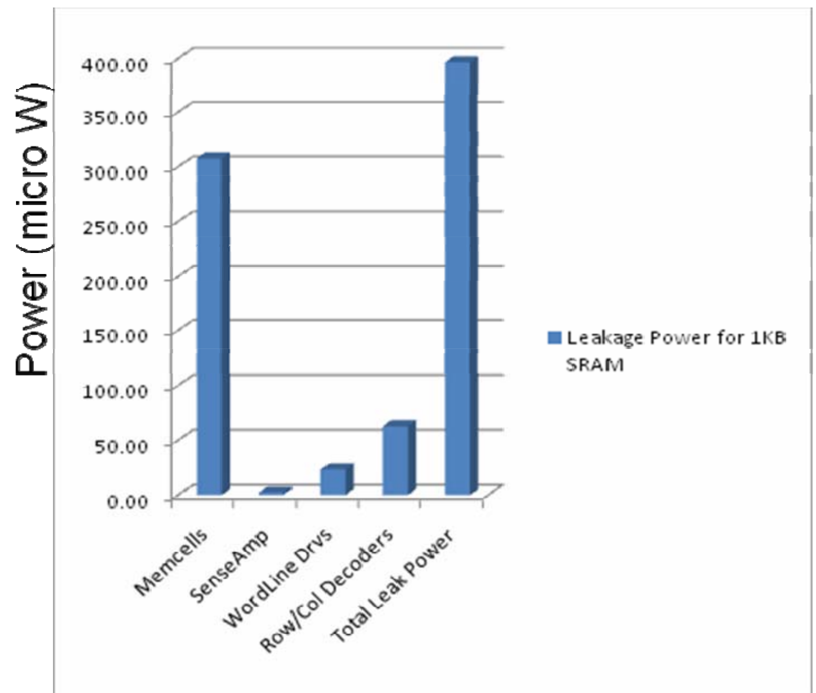


Figure 14: The breakdown of the leakage power for a 1KB SRAM

Finally, **Figure 15** demonstrates the total power for a 1K SRAM. What is noticeable is that most of the power used in a SRAM is in the memory cell. This emphasizes the need for better memory designs with less power dissipation.

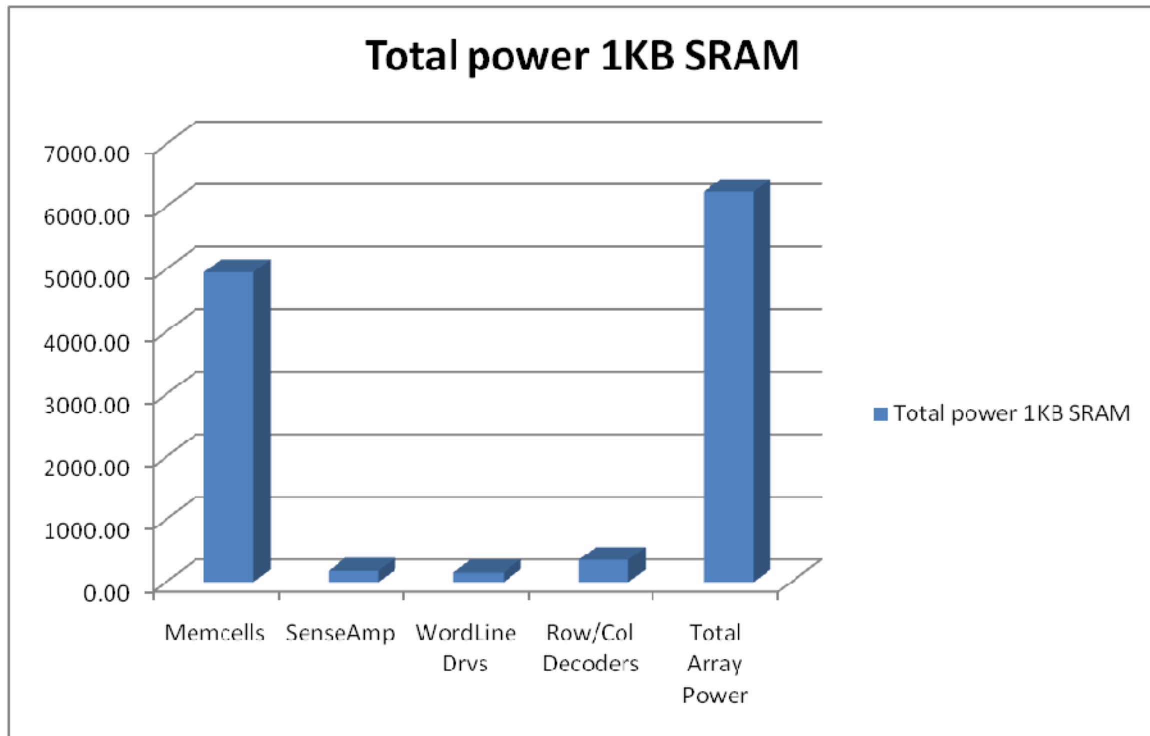


Figure 15: The total power breakdown for a 1K SRAM

Then to verify our results we did a test with same exact input parameters through HP CACTI 4.20 and also used the stated results from reference [13] to measure and verify the accuracy of our results.

The reference claims to be 98% accurate, and we know that HP CACTI is between 10 to 30% inaccurate. Our results seem to be about 12.5 % in accurate or in other words 87.5% accurate comparing to the reference data. This is very promising considering there are some parameters that we are overlooking due to time constraints. The **Figure 16,** compares UCI-SAW results versus HP CACTI and reference paper for the same set of input parameters.

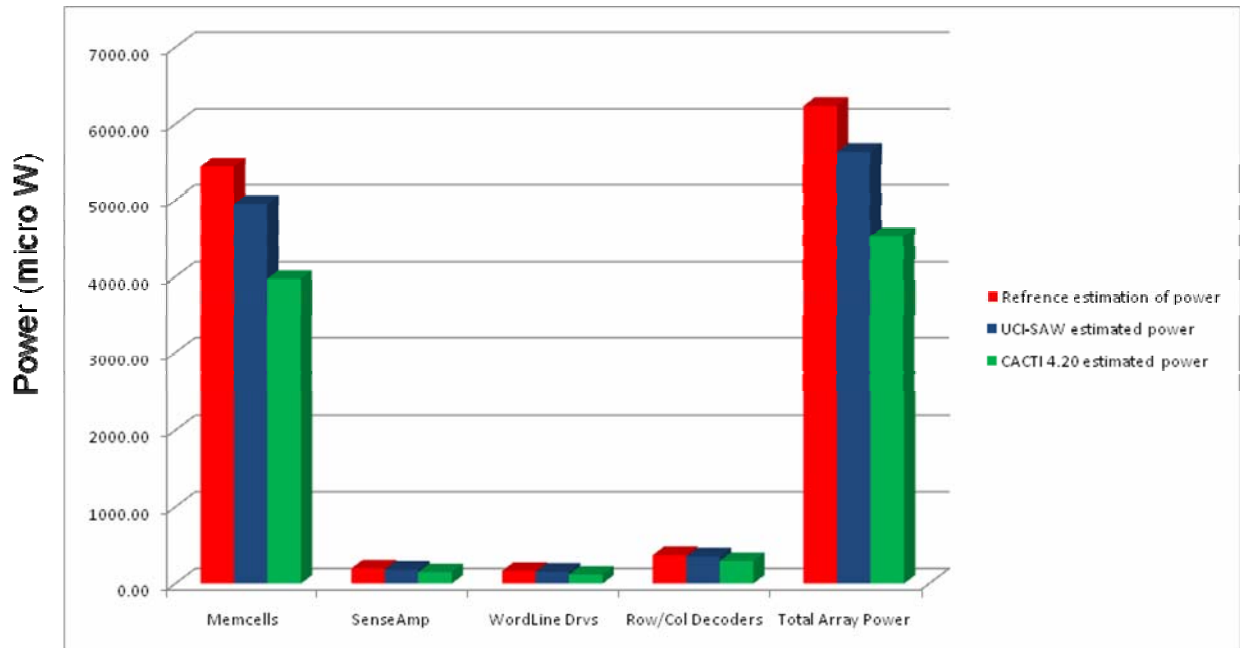


Figure 16: The comparison of results of UCI-SAW, HP CACTI and Reference data

Future work

Now that the groundwork is done for this tool, there some great opportunities that calls for further research and work.

On the investigation side, there is a lot of room for research in other leakage mechanism and their effect on the total power dissipation. On the implementation side, the web site can be optimized for better and faster performance, perl can be improved for real automation of the process and HSPICE code could be investigated for better components net list as well better parameter extraction.

In balance with time constraints and limit on the knowledge, a lot of effort went on this project and there is a great deal of commitment to continue the work on this project in the future.

Section 4: Non-Technical Constraints

As outlined in the course syllabus, two of the five suggested constraints must be argued for each project. For this project, two most influential constraints are economy and manufacturability. We now describe the impact of each of these as well as how we will address them.

When it comes to designing new technology or for this matter making a substantial change to an existing technology, the most important constraints is economy. Without proper budget, no researcher would spend any time to verify his or her ideas. Time has shown that companies that have invested in new ideas, while taking a chance on some capital, which usually is not that high, are getting the upper hand to be the first to use innovative technology. May be this is the reason that fortune technology company always have a research and development division that are always looking for innovative researchers with unique and different ideas. With high demand of electronic applications that use SRAM technology, it seems to be very beneficial for companies if they can use a new model for the design of SRAM chips.

Therefore, while it may look expensive to invent a completely new approach on design study of power consumption, especially while companies have an approximately achieved 81 % yield ratio, the idea that they can achieve up to 93% yield ratio makes any company think twice before rejecting the request for appropriate funding of this model. A quick dollar for dollar return evaluation suggests that any company will profit much higher from successful outcome of this project than what they invested in it.

2nd and in this case a more difficult constraints facing such project is manufacturability. Manufacturing chips is a very expensive process and it's because of this that many of even big companies that are leading the technology don't have their own manufacturing plant. Many factors such as need for new and efficient machinery and well as maintenance, safety and high standards discourage these companies to have any idea manufactured in real life. Many ideas that may look good on paper have never been implemented due to the high cost of manufacturing. In other words, sometimes some solutions seem to be very helpful to improve the performance of a technology but once the cost of manufacturing is calculated, it shows that is much more profitable to stay with the old technology. Low yield ratio is one of the most important factors once it comes to make decision about building a chip.

Our project however, proves not only analytically, but also empirically that some major changes could be considered in the manufacturing that could have a major impact in the yield ratio.

Our vision is to deliver a tool that not only replaces HP CACTAI, but also change the approach of all researchers and designer toward the realistic power consumption of the SRAM.

SPECIAL Thanks to:

Professor Ahmed Eltawil, for giving us the opportunity to be a part of this great project. Moreover, for his support and help all throughout this project.

Amin Khajeh Djahromi, for meeting with us every week and dedicate his time to help us and give us great direction.

Professor Minh Q. Do, for his great contribution to this field of research.

References

1. C. C. Wu, C. H. Diaz, B. L. Lin, S. Z. Chang, C. C. Wang, J. J. Liaw, Ch. H. Wang, K. K. Young, K. H. Lee, B. K. Liew, and J. Y. C. Sun, " Ultra-Low Leakage 0.16 µm CMOS for Low-Standby Power Applications," *IEDM Tech. Digest*, pp. 671–674 (1999).
2. C. H. Diaz, M. Chang, W. Chen, M. Chiang, H. Su, S. Chang, P. Lu, C. Hu, K. Pan, C. Yang, L. Chen, C. Su, C. Wu, C. H. Wang, C. C. Wang, J. Shih, H. Hsieh, H. Tao, S. Jang, M. Yu, S. Shue, B. Chen, T. Chang, C. Hou, B. K. Liew, K. H. Lee, and Y. C. Sun, " A 0.15 µm CMOS Foundry Technology with 0.1 µm Devices for High Performance Applications," *Digest of Technical Papers, Symposium on VLSI Technology*, 2000, pp. 146–147.
3. A. Chatterjee, J. Esquivel, S. Nag, I. Ali, D. Rogers, K. Taylor, K. Joyner, M. Mason, D. Mercer, A. Amerasekera, T. Houston, and I. C. Chen, " A Shallow Trench Isolation Study for 0.25/0.18 µm CMOS Technologies and Beyond," *Digest of Technical Papers, Symposium on VLSI Technology*, 1996, pp. 156–157.
4. F. S. Shoucair, "Scaling, Subthreshold, and Leakage Current Matching Characteristics in High-Temperature (25°C–250°C) VLSI CMOS Devices," *IEEE Trans. Components, Hybrids, Manuf. Technol.* **12**, No. 4, 780–788 (December 1989).
5. W.-C. Lee and C. Hu, "Modeling CMOS Tunneling Currents Through Ultrathin Gate Oxide Due to Conduction- and Valence-Band Electron and Hole Tunneling," *IEEE Trans. Electron Devices* **48**, No. 7, 1366–1373 (July 2001).
6. T. Wang, L.-P. Chiang, N.-K. Zous, C.-F. Hsu, L.-Y. Huang, and T.-S. Chao, " A Comprehensive Study of Hot Carrier Stress-Induced Drain Leakage Current Degradation in Thin-Oxide n-MOSFETs," *IEEE Trans. Electron Devices* **46**, No. 9, 1877–1882 (September 1999).
7. J. Chen, T. Y. Chan, I. C. Chen, P. K. Ko, and C. Hu, " Subbreakdown Drain Leakage Current in MOSFET," *IEEE Electron Device Lett.* **8**, 515–517 (1987).
8. T. Wang, T. E. Chang, C. M. Huang, J. Y. Yang, K. M. Chang, and L. P. Chiang, " Structural Effect on Band-Trap-Band Tunneling Induced Drain Leakage in n-MOSFET's," *IEEE Electron Device Lett.* **16**, No. 12, 566–568 (1995).
9. A. J. Bhavnagarwala, X. Tang, and J. Meindl, "The Impact of Intrinsic Device Fluctuations on CMOS SRAM Cell Stability," *IEEE J. Solid-State Circuits* **36**, No. 4, 658–665 (2001).
10. K. Takeuchi, "Channel Size Dependence of Dopant-Induced Threshold Voltage Fluctuation," *Digest of Technical Papers, Symposium on VLSI Technology*, 1998, p. 72.
11. Minh Q. Do, Per Larsson-Edefors, and Mindaugas Drazdziulis, "Current Probing Methodology for Static Power Extraction in Sub-90nm CMOS Circuits", , , Technical report No. 2007-07, Department of Computer Science & Engineering, Chalmers University of Technology, Göteborg, Sweden, May 2007
12. Minh Q. Do, Mindaugas Drazdziulis, and Per Larsson-Edefors, "Architecture-Level Power Estimation and Scaling Trends for SRAM Arrays ", , Proceedings of 2006 Swedish System-on-Chip Conference, Kålmorden , Sweden , May 4-5 2006.
13. Minh Q. Do,"Accurate Leakage-Conscious Architecture-Level Power Estimation for SRAM-based Memory Structures", PhD Thesis No 31D (ISBN: 978-91-7291-968-6), Department of Computer Science & Engineering, Chalmers University of Technology, Göteborg, Sweden, May 2007