Cacti → technology node can't be specified
0.8 um fixed

↳ enhanced access and cycle time model for on-chip caches.

→ derive relatively simple equations that predict the access/cycle times of caches as a function of various cache parameters, process parameters, and array organization parameters.

→ A Hspice model of the cache was used to validate ~~their~~ analytical model.
→ This only shows that model matches Hspice model, it does not address the issue of how well the assumed cache model structure reflects a real cache design.

→ Code needs to be compiled using the make file provided.
→ time.c is the model.
→ transistor widths and process parameters are defined in def.h

→ chooses the organisation that gives the smallest access time.

→ Associativity in cache??

$\frac{32 \times 8}{32 \times 1}$

Date / /
Page No.

→ Number of Sets (S) → $\frac{\text{Cache Size (C)}}{\text{Block size (B)} \times \text{Associativity (A)}}$

→ Such an organisation could result in an array that is much longer in one direction causing either bitlines or wordlines to be very slow.
→ leads to longer than necessary access time.
→ N_{dwl} → indicates how many times the array has been split with vertical cut lines (creating more but shorter wordlines).
→ N_{dbl} → indicates how many times the array has been split with horizontal cut lines (causing shorter bitlines.
→ total subarrays → $N_{dwl} \times N_{dbl}$.

array organizational parameters

→ N_{spd} → how many sets are mapped to a single wordline.
→ N_{twl}, N_{tbl}, N_{tspd} → for tag array.

→ Circuit gets decomposed into many equivalent RC circuits, and using simple RC equations to estimate the delay of each stage.

Decoder
Sense Amplifiers
Comparators
Column MUX.
Wordline
Bitlines.
MUX Drivers
output Drivers.

* Cacti 2
↳ An integrated cache timing and power model.
→ Access time, cycle time, power.

→ ~~It~~ added support for full associative caches and for caches with multiple access ports

→ User can specify read/write ports, read only ports and write only ports

*cacti cache_size block_size assoc tech node.
read/write ports read ports write ports.

Cacti 3

→ timing, power and area model.
aspect ratio; efficiency.

→ fully independent banking of caches. This allows users to split the cache into a number of banks with its own set of address and data buses.

fully associative cache
$N_{sets} = N_{spd} = 1$
N_{blk} → can be varied.

·/cacti cache size cache_line assoc tech-node. read/write ports read ports write ports n-banks.

Cacti 4

→ leakage power added.
→ web interface.

H-tree ?

→ if requested no. of ports is equal to or less than the number of banks, we assume each bank only implements one read/write port.
→ can also model memories without tags

Date / /
Page No.

Cacti 5

→ dynamic power, access time, area, and leakage power of caches and other memories.

→ cache organisation assumed by cacti is now output graphically to assist users in understanding the output generated by cacti

Date / /
Page No.

main.cc → takes input file / command line arguements and passes control to cacti_interface

Linked to io.h (uca_org_t class)

io.h → linked to const.h, cacti_interface.h.

io.cc → io.h, area.h, basic_circuit.h, parameter.h, Ucache.h, nuca.h, crossbar.h, arbiter.h

→ InputParameter class
  ↳ parse_cfg func. → parses the cache.cfg file.
  ↳ display_ip() → displays the input parameters

powercomponents, powerDef + operator overloading

uca_org_t cacti_interface() → outputs final results.
  ↳ function overloading (one for file input and other for command input).
  uses a 'solve' function to solve the equations.

→ error_checking func.

→ output_data_csv → creates a csv file.
↳ output_UCA → prints the results.

cacti_interface.h → linked to const.h

↳ defines 3 classes

↳ min_values_t

↳ mem_array

↳ uca_org_t

→ ~~pat~~ defines powercomponents and powerDef classes

→ write types definition

→ defines creates Input Parameter class.

this will change.

uca_org_t class

↳ variables for data ~~arr~~ and tag array, access time, cycle time, area, area efficiency, power etc.

→ functions → find_delay, find_energy, find_area, find_cyc

mem_array class → wspd, Nwl, Nbl, area, power, delay of components.

↳ modify this if any component is added.

cacti_interface.cc → linked to area.h, basic_circuit.h, component.h, const.h, parameter.h, cacti_interface.h

↳ defines the functions of uca_org_t class.

find_delay, find_energy, find_cyc
→ return final values if value of for data and tag arrays are known.

* const.h
↳ defines constants for project
→ address bits
→ output bits
→ max. value of Ndwl, Ndbl, Nspd etc
→ threshold voltages.

Important
↳ 6 for transistors.
↳ needs to be changed for more transistors.

* area.h → linked to cacti_interface.h, basic_circuit.h
↳ creates 'Area' class
↳ func. for getting width, height and returning area

cell height, width
area = h * w

area.cc → linked to area.h, component.h, decoder.h, parameter.h, basic_circuit.h
→ has no code.

* basic_circuit.h → linked to const.h, cacti_interface.h

basic_circuit.cc
functions → powers, ispow2, _log2, logtwo, gate_C, gate_C_pass drain_C_, R_on, R_to_w

horowitz, pmos_to_nmos_sz_ration.
simplified_nmos_leakage, simplified_pmos_leakage,
cmos_Ileak

- log2 → log base 2 of num.
is_pow2 → check if num is power of 2
powers → to calculate ~~power~~ base exponent.

gate_c, gate_c_pass → same.
↳ return gate capacitance in farads.

⊖ Technology Parameters class ??

R_to_w → given resistance, returns transistor width in um.

p_mos_to_nmos_sz_ratio.
↳ sizing ratio.

horowitz → timing models for MOS circuits by Mark Horowitz.

cmos_Ileak, simplified_nmos_leakage, simplified_pmos_leakage
↳ gives off current for cmos, nmos and pmos respectively.

* parameter.h
Technology Parameter class
↳ parameters which are functions of certain device technology
↳ capacitance, resistance, V_{on}, I_{on}, I_{off}, C_{ox}, t_{ox}

→ 3 classes inside class
↳ DeviceType
↳ InterconnectType
↳ MemoryType

w_comp-inv-p1, w_comp-inv-p2, w_comp-inv-p3
w_comp-inv-nm1, w_comp-inv-n2,
w_comp-inv_n3
↳ meaning of parameters not clear !!!

time
64-173

Dynamic Parameter class
↳ Npd, Ndwl, Ndbl

parameter.cc
↳ defines functions for above classes like display for displaying parameters.

* component.h → linked to area.h, parameter.h

Classes → Crossbar, Bank, Component

↓

compute_gate_area,
compute_tx_width_after_folding
height_sense_amplifier.
logical_effort.
compute_diffusion_width.

component.cc → defines the above functions

logical_effort, ~~compute~~ height_sense_amplifier
→ need to understand these functions.

* technology.cc → linked to basic_circuit.h, parameter.h

→ 3 functions
- wire resistance.
- wire capacitance.
- init_tech_params.
  - line 376 to 382 → SRAM cell parameters.
  - for 90nm.
  - for 65nm → 574-580
  - for 45nm → 776-780
  - for 32nm → 976-980

defines technology parameters Vdd, gate length, Cox, Vth, etc.

* arbiter.h → linked to basic_circuit.h, cacti_interface.h, component.h, parameter.h, mat.h, wire.h.

Arbiter class → inherited from component class.

private variables → NTm1, PTm1, NTm2, PTm2, R, PTi, NTi, flit_size, NTtH, PTtH, TriS1, TriS2 → ??

arbiter.cc
↳ line 56 - 63 → ??

→ width of n-mos and p-mos transistors.
↳ used in leakage power computation.

* crossbar.h → linked to basic_circuit.h, cacti_interface.h, component.h, parameter.h, mat.h, wire.h.

Crossbar class → inherited from component class.

crossbar.cc → output_buffer, compute_power functions.

Date / /
Page No.

prints → flit size, width and height of array, dynamic and leakage power.

* Ucache.h → linked to area.h, router.h, nuca.h.

min_values_t class. → @ update_min_values function.

struct solution.
Calculate_time, solve, init_tech_params functions.

Ucache.cc → linked to area.h, bank.h, basic_circuit.h, component.h, const.h, decoder.h, parameter.h, subarray.h, uca.h.

@ uses UCA class from uca.h.

functions → check_uca_org, check_mem_org, find_optimal_uca, filter_tag_arr, filter_data_arr

Date / /
Page No.

* uca.h. → linked to area.h, bank.h, component.h, parameter.h, htree2.h.

class UCA → inherited from Component class.

compute_delays, compute_power_energy()
↓ ↓
returns cathisetime. power for a bank of an array.

defined in uca.cc
uses functions from bank.h and bank.cc.
and from htree2.h.
for delays and power.

* bank.h. → linked to component.h, decoder.h, mat.h, htree2.h.

Bank class → inherited from component class

functions → compute_delays, compute_power_energy

Bank.cc
Bank constructor → read-write ports, exclusive read and write ports.
address bits, data input bits, data output bits.

compute delays → uses mat.compute_delays()
↳ in mat.h

compute_power_energy() → uses mat.compute_power_energy()
↳ computes dynamic and leakage power for read operation.

* mat.h → linked to component.h, decoder.h, wire.h, subarray.h.

Mat class → inherited from component class.

functions → compute_delays, compute_power_energy.

mat.cc
↳ line 92 → 2 transistors per cell??

line 364 → delay_fa_tag()
{ } function definition??
↳ line 490.

functions → width_write_driver_or_write_mux, compute_comparators_height, compute_bitline_ddly, compute_sa_ddly.

compute - subarray - out - drv.
compute - comparator - delay.

compute - power - energy.

* Subarray.h → linked to area.h, component.h, parameter.h.

Subarray class → inherited from component class.

variables → num_rows, num_cols,
cell
↳ (instance of Area class).

c_wl, c_bl → capacitances.

func. → get_total_cell_area
compute_C()
↳ for bitline and wordline capacitance.

cell.get_area() * num_rows * num_cols

* hTree2.h → linked to basic_circuit.h, component.h, parameter.h, assert.h, subarray.h, cacti_interface.h, wire.h.

HTree2 class → inherited from component class.

functions → in_hTree(), out_hTree(), limited_in_hTree(), limited_out_hTree(), input_cmd, output_buffer

Date / /
Page No.

nthree2.cc

input_nand → nand gate sizing calculation.

output_buffer → tristate buffer model consisting of not, nand, nor and driver transistors.

* decoder.h → linked to area.h, component.h, parameter.h,

Decoder class → inherited from component class.

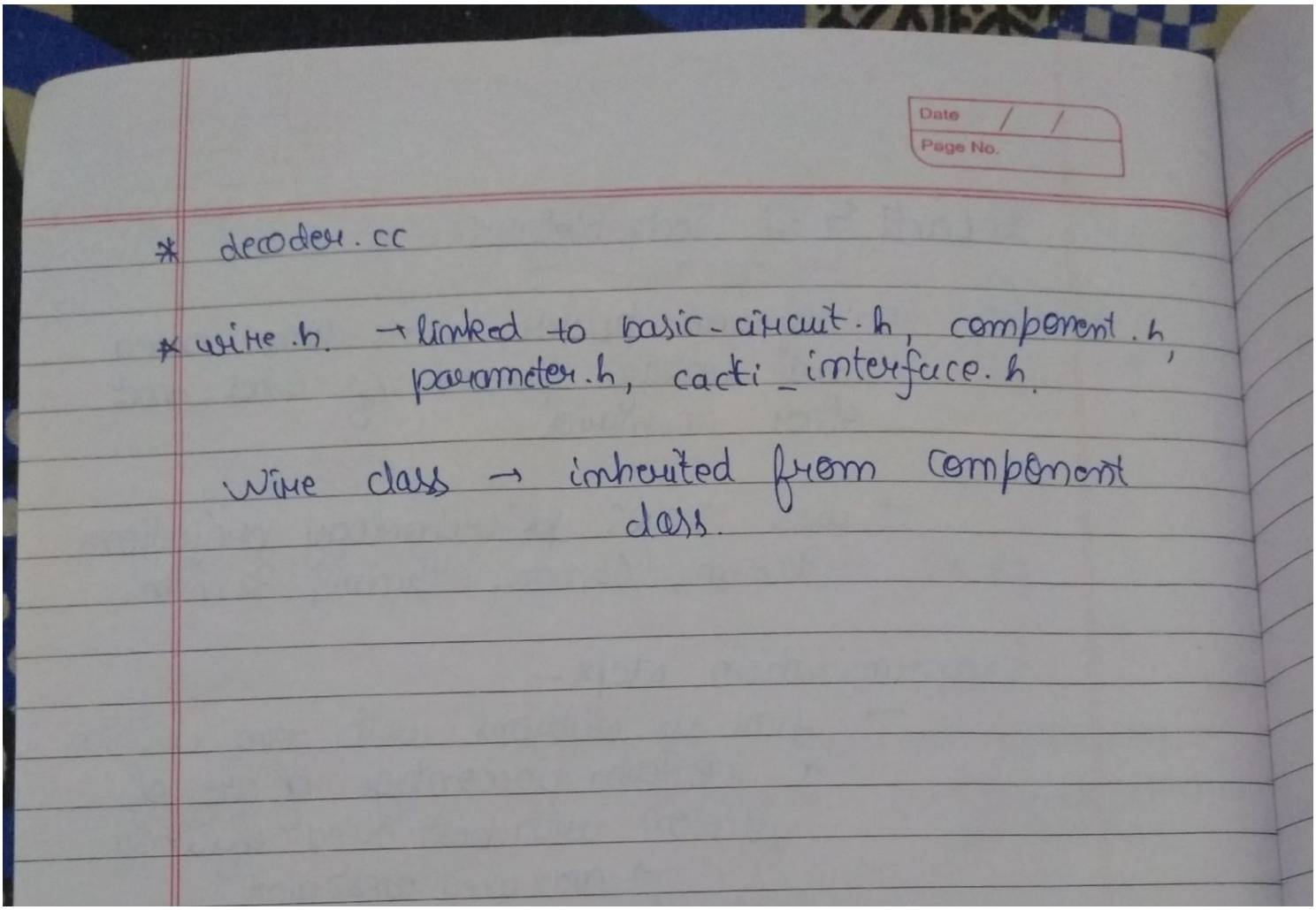
func. → compute_widths, compute_area, compute_delays
↳ returns outrise time.

PredecBlk class → inherited from component class.

PredecBlkDrv → inherited from component class

Predec class → inherited from component class.

Driver class


Date / /
Page No.
* decoder.cc
* wire.h → linked to basic_circuit.h, component.h, parameter.h, cacti_interface.h.
wire class → inherited from component class.

* Cacti 5.0 Tech Report

→ dynamic power, access time, area and leakage power of caches and other memories.

→ uses ITRS technology projections 90nm, 65nm, 45nm, 32nm.

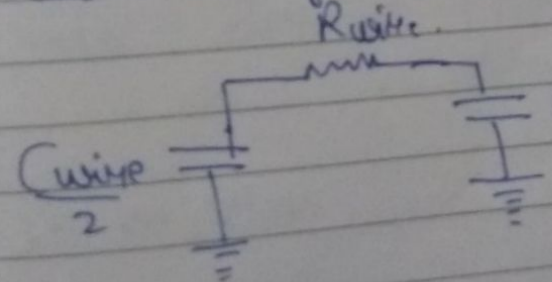
→ Optimization steps →

→ find all solutions with area within a certain percentage of area of solution with best area efficiency.
→ max area constraint.

→ find all solutions with access time that is within a certain percentage of best access time. solution in this reduced subset of solutions
→ max acc time constraint.

→ apply opt. function.

→ wire delay model.

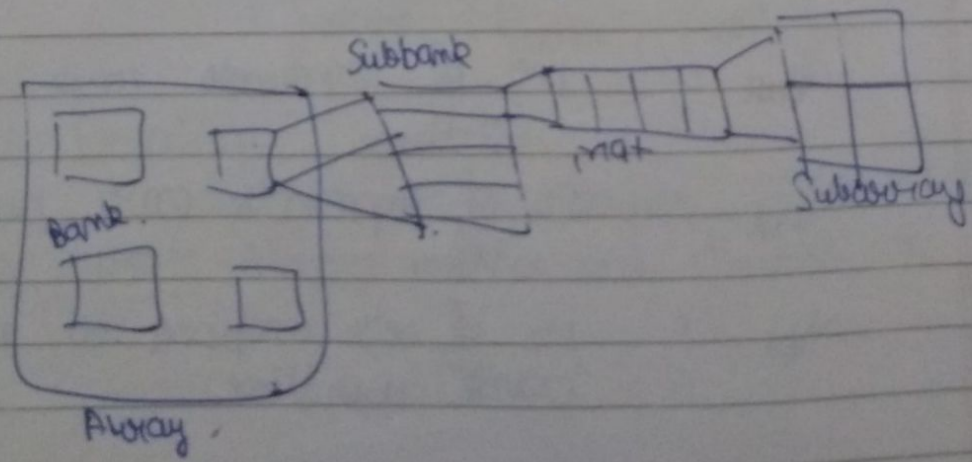


Ecc → Error correction code.
→ a variable to ~~incorporate~~ specify the number of data bits per ecc bit
→ default value = 8

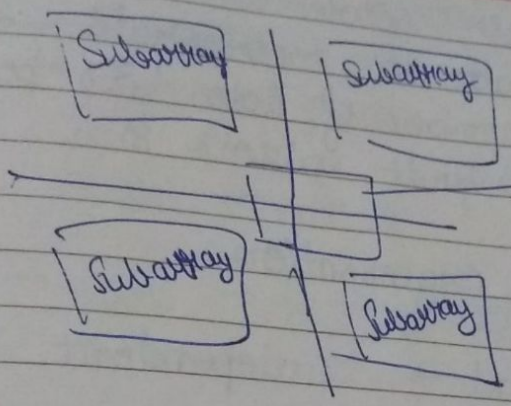
Data Array Organisation

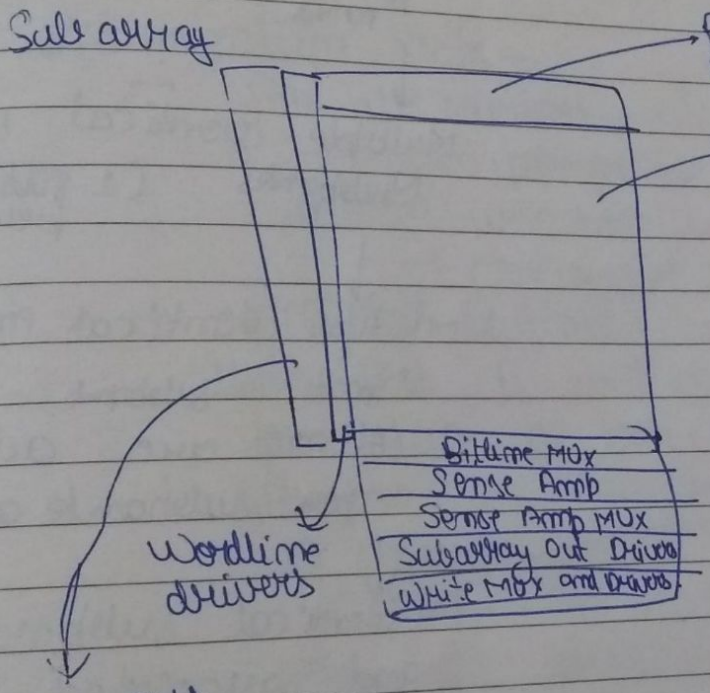
Highest level → Independent Banks
N_{banks}.
↓
Multiple identical subbanks.
$N_{subbanks}$ (1 subbank activated per access).
↓
Multiple identical mats
$N_{mats-in-subbank}$.
(All mats are activated per subbank access).
↓
4 identical subarrays and associated predecoding logic.

Subarray will be the part where cell and its peripheral circuitry will be defined.



Mat





N_{dwl} → no. of segments in a bank wordline.

N_{dbl} = no. of segments in a bank bitline.

N_{spd} = No. of sets mapped to each bank wordline.

→ Each predecode block has two levels.
  → the first level is composed of one 2-4 decode unit and one 3-8 decode unit.
  → at second level, 4 outputs from 2-4 decoder and 8 outputs from 3-8 decoder are combined using 32 NAND2 gates.

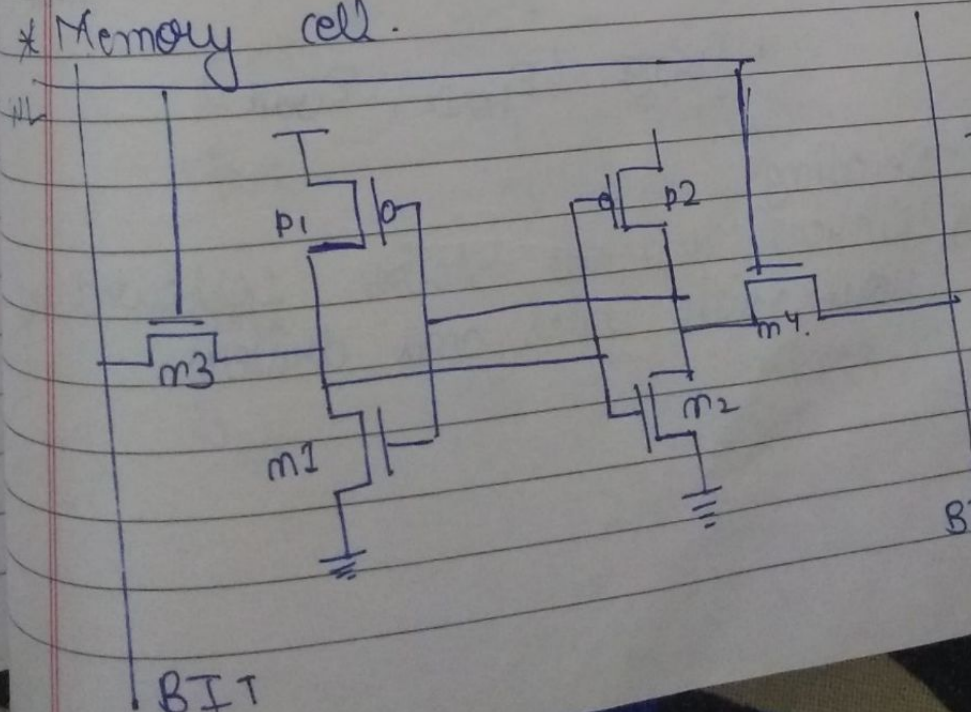
$R_{Predec-output-wire} = L_{Predec-output-wire} R_{unit-length-wire}$

$C_{Predec-output-wire} = L_{Predec-output-wire} C_{unit-length-wire}$

↓
max. length amongst lengths of predecode output wires.

→ sizing of gates in each circuit path is calculated using the method of logical effort.

* Memory cell.



→ 1 ported 6T SRAM cell.

$N_{subbanks} = \frac{N_{dbl}}{2}$

$N_{mats\text{-}in\text{-}subbanks} = \frac{N_{dwl}}{2}$.

Wire Modelling

$R_{wire} = L_{wire} R_{unit\text{-}length\text{-}wire}$.

$C_{wire} = L_{wire} C_{unit\text{-}length\text{-}wire}$.

L_{wire} → length of wire.

$R_{unit\text{-}length\text{-}wire} = \alpha_{scatter} \frac{\rho}{(thickness - barrier - dishing) * (width - 2 * barrier)}$

$C_{unit\text{-}length\text{-}wire} = \varepsilon_0 \left(2M\varepsilon_{horiz} \frac{thickness}{spacing} + 2\varepsilon_{vert} \frac{width}{ILD_{thick}}\right) + fringe(\varepsilon_{horiz}, \varepsilon_{vert})$.

Row-Decoding

↳ two row predecode blocks followed by row-decode gates and drivers.

Read [7] → for SRAM cell size and widths.

Peripheral Circuitry

↳ Sense Amplifier → clock-latch based sense amplifier.

↳ Bitline and Sense Amp. MUXes.

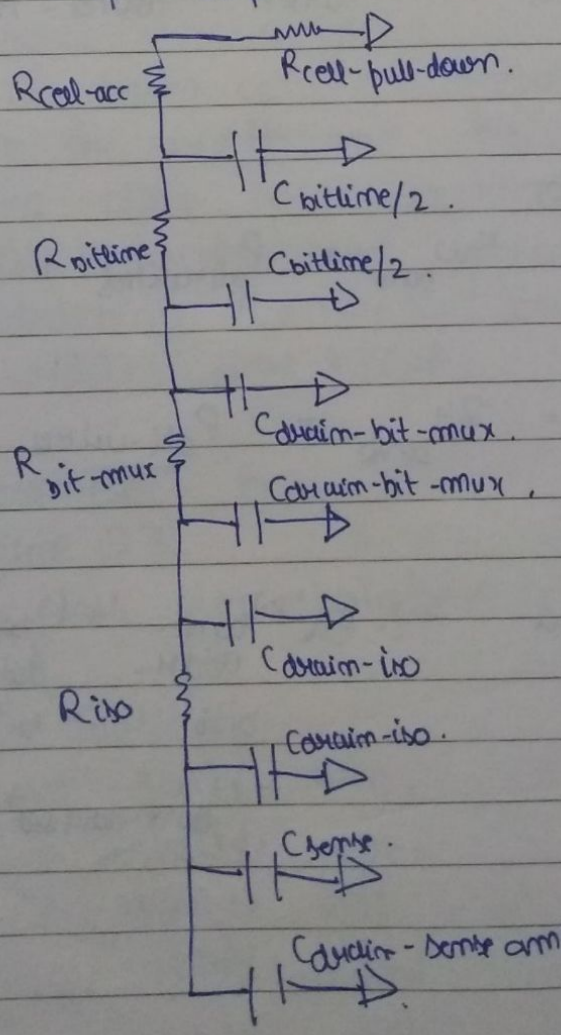
↳ NMOS based MUX.

↳ Precharge and Equalization Circuitry.

~~↳ fixed widths for NMOS.~~

fixed widths for NMOS from [20]

Circuit Model of b/w Memory cell and Sense Amp. Input.



[26] → analysis of latch-based sense amplifier and equations for sensing delay under different assumptions. (1)

* Area Model

total diffusion width → eqn 14 page 27.

Area of data array = $H_{data-arr} \; W_{data-arr}$

$P_{all-wires}$ = $P_{wire} \; N_{wires-routed-to-banks}$

pitch

for 8 banks.

$W_{data-arr} = 4W_{bank} + P_{all-wires} + 2\frac{P_{all-wires}}{4}$.

$H_{data-arr} = 2H_{bank} + \frac{P_{all-wires}}{2}$.

$N_{wires-routed-to-banks} = 8(N_{bank-addr-bits} + N_{bank-datain-bits} + N_{bank-dataout-bits} + N_{way-select-signals})$

$N_{banks\ in\ horizontal\ dim^n}$ $\geq 2 \times N_{banks\ in\ vertical\ dim^n}$

Page 30 :-

Area of Mat equations.

~~eq^n (32) and (3~~

** eq^n 33 and 34 uses height and width of memory cell

eq^n 38 → $H_{subarr-bitline-peri-dim}$ → will be changed.

* area.h, area.cc → returns area given the width and height, Area class has functions to set width, height and get_area() for returning the area based on saved width and height.

Nothing has to be changed in these files.

* parameter.h.

line 123

class MemoryType {

public:

double b_w;
double b_h;
double cell_a_w;
double cell_pmos_w;
double cell_nmos_w;
double Vbitbar;

will probably will be changed.

memory cell related

line 164-173

parameter cc → has a function definition for display() → to display the 3 parameters.

* Delay Modeling → access time
→ random cycle time.

Access time
↳ T_{mat} → includes time taken by row-decoder path, bit-mux decoder path, sense amp decoder path.

$$T_{bitline} = \begin{cases} \sqrt{2 T_{step} \frac{V_{DD}-V_{th}}{m}}, & T_{step} \leq 0.5 \frac{V_{DD}-V_{th}}{m} \\ T_{step} + \frac{V_{DD}-V_{th}}{2m}, & T_{step} > 0.5 \frac{V_{DD}-V_{th}}{m} \end{cases}$$

from [28].

T_{step} → uses $R_{cell-pull-down}$, $R_{cell-acc}$.
↳ will change

$$m = \frac{V_{DD}-V_{th}}{2 T_{step}}$$

Random Cycle Time

↳ limited by wordline and bitline delays.

↳ will not change except $V_{bitline-swing}$

* Power Modeling

$V_{bitline-swing} = 2V_{sense}$.

↳ this may change.

↳ dynamic energy

↳ leakage power.

eqn 75 and 76 may change as $V_{bitline-swing}$ may change.

↳ $W_{inv-nmos}$, $W_{inv-pmos}$ → width of transistors

eqn 91, 92 → related to power consumed by cells.

$P_{mem-cell} = V_{DD} I_{mem-cell}$ (1)

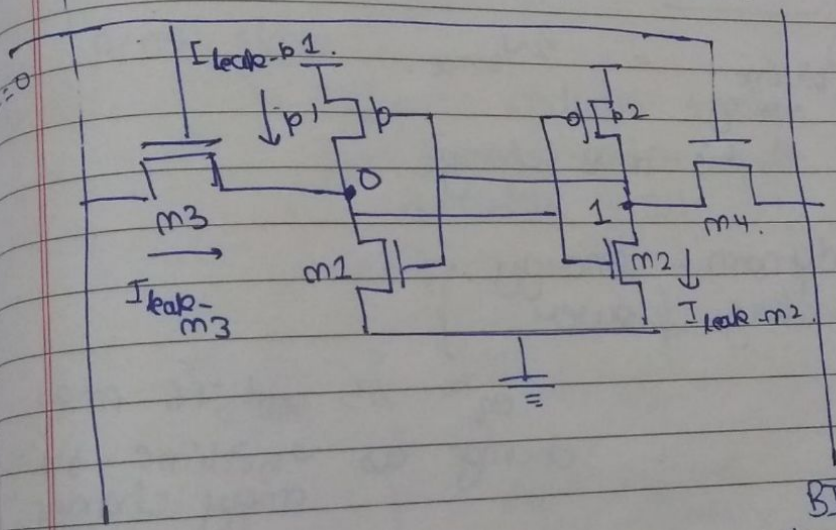
$I_{mem-cell} = I_{p_1} + I_{n_2} + I_{n_3}$.

$I_{p_1} = W_{p_1} I_{off-pmos}$.

$I_{n_2} = W_{n_2} I_{off-nmos}$.

$I_{n_3} = W_{n_2} I_{off-nmos}$.

These will change.



pre-charged to VDD.

* Technology Modelling

→ Technology projections from ITRS

→ 65, 90, 65, 45, 32 nm → nodes.

Three Device Types
- → High performance (HP)
- → Low Standby Power (LSTP)
- → Low operating Power (LOP).

can be set for data array cell-type, data-array peripheral type, tag array cell-type and tag array peripheral type in configuration file.

Table-5 page 39
↳ technology data for SRAM cell
- ↳ Area of SRAM cell — $A_{sram-cell}$
- ↳ $w_{sram-cell-acc}$ — width of SRAM cell access transistor
- ↳ $w_{sram-cell-pd}$ — width of of SRAM cell pull-down transistor
- ↳ $w_{sram-cell-pu}$ — Pull up transistor.
- ↳ $AR_{sram-cell}$ — Aspect Ration of cell.

line 575-579 for 455nm node in technology.cc

→ MASTAR can be used to obtain device data for scaling models and assumptions that are different from those of ITRS.