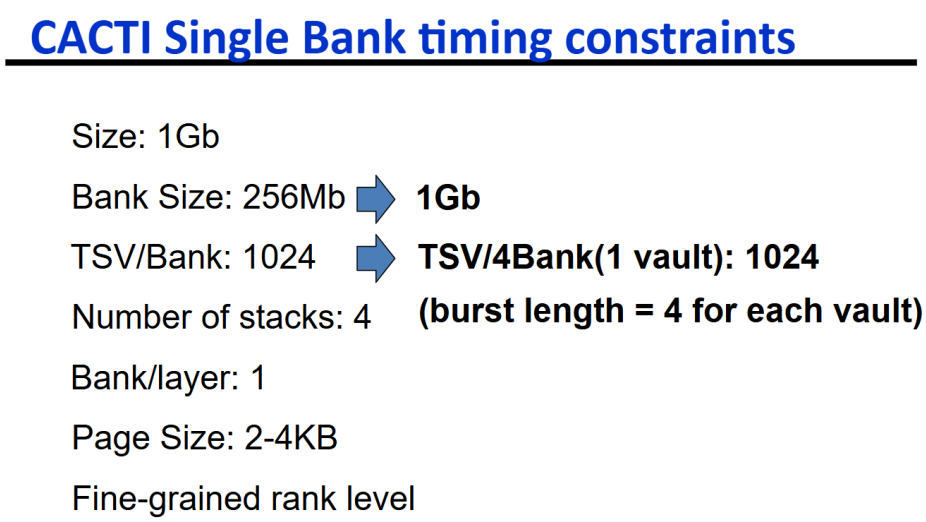
3D-DRAM-Analysis

Created	@October 16, 2024 6:21 PM
Class	DRAM Controller

Goal

- Single slice controller with point to point banks connections with multiple channels supports.
- Single channel single bank architecture with timing controls
- Analyze 1 channel 4 banks design & point to point design. Their Area, power,timing & performance to a source of random traces
- Refresh controls targeting single bank architecture choose and combine the merits of multiple refresh techniques together or think about a new application specific one
- New Specification to analyze.
- DDR model simply modify from senior's model



1. Bank modeling using CACTI

- Timing constraints:
  - Basic DRAM timing constraints:
    - Look at DDR3,DDR4 timing constraints for necessary constraints analysis and verification.
    - TSV constraints and delays
- Energy:
  - Read write energy consumption also idle energy consumptions.
  - TSV energy consumption
- Verification

- Remodeling of the 3D-DRAM of samsung 3D-DRAM architecture.
- Get the related parameters working and anaylze it
- See the mismatches, it is about 5% error

Calculate the BW,AE & Power consumpton

- Plot the Bandwidth, AE and power consumption using CACTI-3DD with a better configuration.

Find the important modeling parameters from papers

> In addition to the row buffer locality (RBL), bank level parallelism (BLP) has a significant impact on the DRAM bandwidth and energy utilization. Given that different banks can operate independently, one can overlap the latencies of the row precharge and activate operations with the data transfer on different banks. BLP enables high bandwidth utilization even if the RBL is not fully utilized provided that the accesses are well distributed among banks. However, frequently precharging and activating rows in different banks increase the power and total energy consumed.

- RBL & BLP tradeoffs is important [1], size of bank, number of bank also the size of row buffer is important design consideration. Row buffer locality is particulary important for 3D-DRAM

> **3D-stacked DRAM** is an emerging technology where multiple DRAM dies and logic layer are stacked on top and connected by TSVs (through silicon via) [13], [14], [15], [4]. TSVs allow low latency and high bandwidth communication within the stack without I/O pin count concern. Fine-grain rank-level stacking, which allows individual memory banks to be stacked in 3D, enables fully utilizing the internal TSV bandwidth [16], [13]. As shown in Figure 5, fine-grain rank-level stacked 3D-DRAM consists of N_{stack} DRAM layers where each layer has N_{bank} DDR3 DRAM banks, and each bank has its own N_{TSV}-bit data TSV I/O. Vertically stacked banks share their TSV bus and form a vertical rank (or sometimes referred as vault [13]). Hence, the overall system is composed of N_{bank} ranks where every rank has its own N_{TSV}-bit TSV bus and can operate independently.

which enables achieving much better timings [16]. Another interesting distinction of the 3D-stacked DRAM is that the banks within a rank can operate in pseudo-parallel, time multiplexing the shared TSV bus, contributing to the aggregate bandwidth additively by exploiting the high bandwidth TSVs [4]. As a result, row buffer misses in a bank become visible as reduced bandwidth in the corresponding rank which can only be amortized by fully utilizing opened row buffers. Hence, 3D-stacked DRAMs are more vulnerable to the strided access patterns.

Name	Configuration (3D-stacked DRAM) $N_{stack}/N_{bank}/N_{TSV}/R$(Kb)/Tech(nm)	tCL-tRCD-tRP-tTSV (ns)	Max BW (GB/s)
conf-D	4 / 8 / 256 / 8 / 32	12.2-7.89-16.8-0.68	178.2
conf-E	4 / 8 / 512 / 8 / 32	12.2-7.89-16.8-0.68	305.3
conf-F	4 / 8 / 512 / 8 / 45	14.2-9.23-19.1-0.92	246.7
conf-G	4 / 8 / 256 / 32 / 32	11.8-21.9-16.4-0.68	229.5

Table 1: 3D-stacked DRAM low level energy breakdown.

Parameter	Value (pj/bit)	Reference
DRAM access (CAS)	2 - 6	[22, 34, 57]
TSV transfer	0.02 - 0.11	[22, 60, 7]
Control units	1.6 - 2.2	[4, 34]
SERDES + link	0.54 - 5.3	[34, 47, 41, 23, 37]

Table 2: 3D-stacked DRAM configurations.

Parameter	HI	MH	ML	LO
Vault (#)	16	8	4	2
Layer (#)	8	4	4	2
Link (#)	8	8	7	1
Link BW (GB/s)	60	40	40	40
Total TSV (#)	2048	2048	1024	512
Intern BW (GB/s)	860	710	360	90
Extern BW (GB/s)	480	320	280	40
Power (Watt)	45	30	25	12

[15]. Finally a cache flush is issued, before the acceleration starts, to ensure that the accelerator accesses the most recent copy in the memory stack to avoid coherency problems.

- Row buffer miss conflicts is expensive for typical 3D-DRAM archtiecture. Minimizing row buffer conflicts is important

1 KB for all the configurations. The unusual DRAM page size of 1 KB is a typical value for 3D-DRAM (from 256 byte [34] to 2 KB [58] are reported). Medium to high end configurations

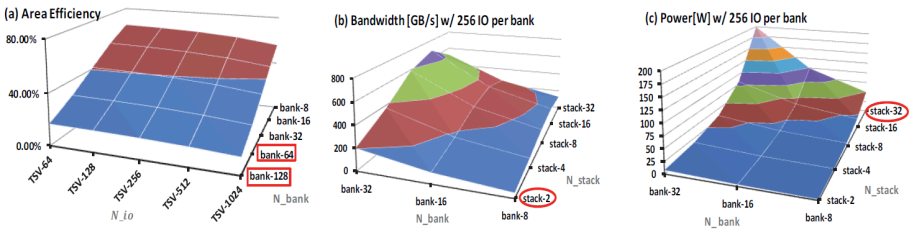


Fig. 7. 3D-DRAM Design Space Exploration for Area, Bandwidth and Power.

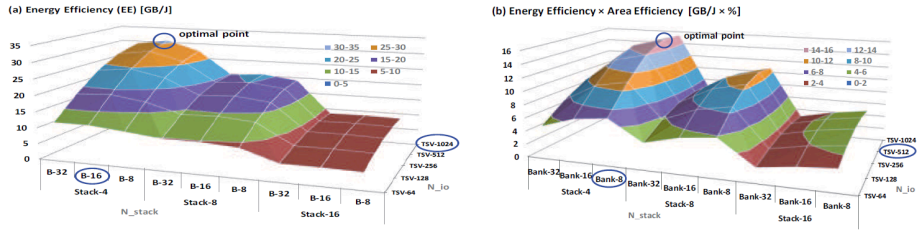


Fig. 8. 3D-DRAM Design Space Exploration for Energy Efficiency.

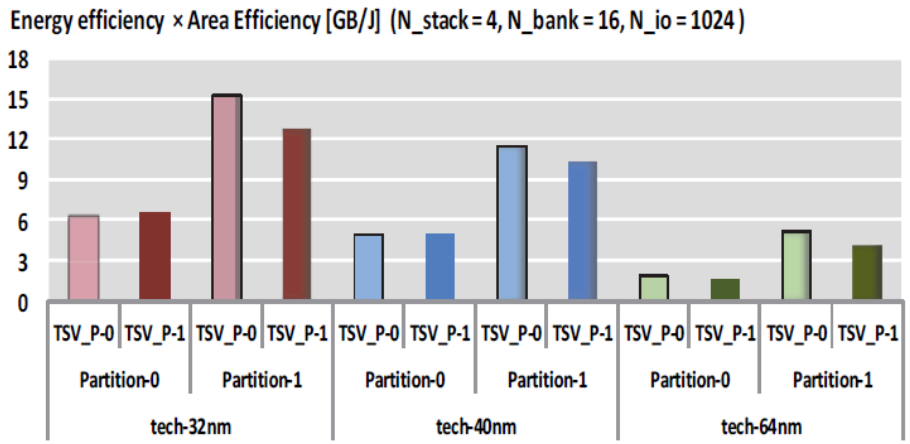


Fig. 9. More 3D-DRAM Design Space Exploration.

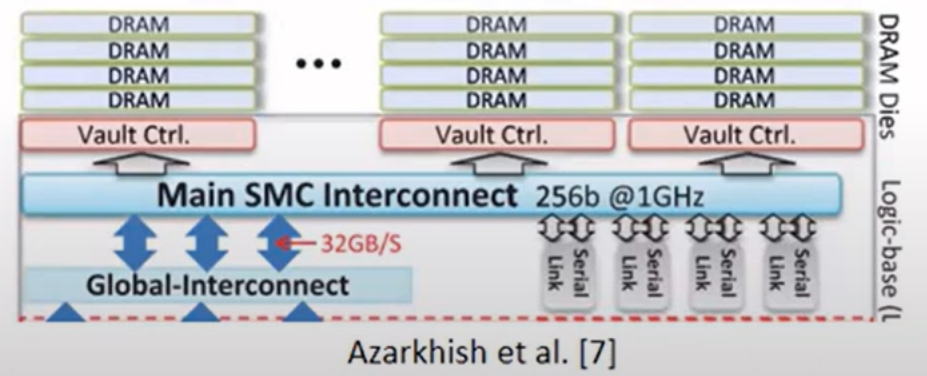
8Gb Memory Capacity with 16Kb Page_size; N_stack = 4; Tech_node: 32nm; TSV_P = 0; Partition = 1							
Bank_size	Bank_count	TSV per bank	AE	Bandwidth	Power	EE	EE × AE
(Mb)	#	#	%	GB/s	W	GB/J	GB/J × %
256	Bank-8	TSV-64	0.65	21.53	1.96	10.99	7.09
128	Bank-16	TSV-64	0.54	45.52	3.78	12.04	6.53
128	Bank-16	TSV-128	0.53	90.62	4.8	18.86	10.11
256	Bank-8	TSV-512	0.6	164.82	6.38	25.81	15.56
256	Bank-8	TSV-1024	0.56	312.75	11.39	27.46	15.38
128	Bank-16	TSV-512	0.5	350.33	12.22	28.67	14.32
128	Bank-16	TSV-1024	0.46	668.4	21.81	30.64	14.03

Fig. 10. Potential Optimal 3D DRAM Design Points.

Table 4 — HBM Channel Addressing

Legacy Mode [6]					Notes
Density per Channel	**1 Gb**	**2 Gb**	**4 Gb**		
Prefetch Size (bits)	256	256	256		1,3
Row Address	RA[12:0]	RA[13:0]	RA[13:0]		
Column Address	CA[5:0]	CA[5:0]	CA[5:0]		1
Bank Address	BA[2:0]	BA[2:0]	BA[3:0]		8
Page Size	2 KB	2 KB	2 KB		1,3
Refresh	8K/32 ms	8K/32 ms	8K/32 ms		
Refresh Period	3.9 us	3.9 us	3.9 us		
Density Code	0001	0010	0011		11

- 256 bit prefetch per memory read and write access
- BL = 2 and 4
- 128 DQ width + optional ECC pin support/channel
- Legacy mode and Pseudo Channel (PC) mode operation; 64 DQ width for PC mode
- Differential clock inputs (CK_t/CK_c) for command/address
- Double data rate (DDR) command/address. Row Activate commands require two cycles, all other commands require one cycle
- Semi-independent row and column command interfaces allowing Activates/Precharges to be issued in parallel with Read/Writes.
- Data referenced to unidirectional differential data strobes RDQS_t/RDQS_c and WDQS_t/WDQS_c. One strobe pair each per DWORD
- Up to 8 channels / device
- Channel density of 1 Gb to 16 Gb
- 8, 16, 32 or 48 banks per channel; varies by device density/channel
- Bank grouping supported
- 2 KB page size per channel
- DBIac support configurable via MRS
- Data mask for masking write data per byte
- Self refresh modes
- I/O voltage 1.2 V
- DRAM core voltage 1.2 V, independent of I/O voltage
- Unterminated data/address/cmd/clk interfaces
- Temperature sensor with 3-bit encoded range output



Timing & Power information

- 1 Gb memory size with 4 layers, each bank being 256Mb
- Note to make the timing constraints compatible to Ramulator, must recalculate it with the DRAM's data rate and clock frequency

Summary

- More number of banks are needed to ultilize BLP
- Row buffer conflicts hinders performance greatly in 3D-DRAM
- EE, AE, BW are important considerations for 3D-DRAM design
- Typcial refresh interval is 32ms for 3D-DRAM, thus the tREF & tRFC is determined by this parameter.
- Row buffer size is typcially 1KB or 2KB to balance the EE, power consumption and cost is an important issue

2. Ramulator2

- Understand how the timing constraints slots means by comparing it with the DDR3,DDR4 timing constraints and parameters.
- Modify the density calculation, address mapping, banks number also removes the bank groups.
- Look at SMLA to see how they model the desired architecture of 4 TSV architecture and how they model the basic 3D-DRAM to prevent from making mistakes
- DRAM sizes calculation, row size, col size and its dq meaning.

Why Is `dq` Included in the Density Calculation?

1. **Determining the Total Storage Capacity**:
   - To find the total storage capacity of the DRAM, you need to understand how much data can be stored per row/column configuration across all banks and bank groups.
   - The `dq` value indicates how many **bits** (or data lines) are associated with each column. When you address a specific row and column, `dq` tells you how much data is actually stored or accessed.
   - If each column can store `dq` bits, then to calculate the total capacity, you must **multiply** by `dq`.
2. **Storage Per Column Access**:
   - Consider a simplified example: If a DRAM has 4 banks, 1,024 rows, 1,024 columns, and each column holds **8 bits** (`dq = 8`), then the total capacity would be:

$$4 \text{ banks} \times 1024 \text{ rows} \times 1024 \text{ columns} \times 8 \text{ bits}$$

   - Here, the `8 bits` comes from `dq`, indicating that each column can store 8 bits of data. Without this, you would only know the number of **cells** but not the total bit capacity.

Summary

The inclusion of `dq` in the density calculation is essential because it converts the organizational structure (banks, rows, columns) into **actual data capacity**. Each combination of row and column points to a data chunk of `dq` bits, so multiplying by `dq` ensures you calculate the **total storage capacity** accurately. Without `dq`, th alculation would only give the total number of addressable locations, not the actual amount of data that can be stored.

- Learn how to use the debug.h in Ramulator2 to display the information, do the analysis first, later try to spot the problem.

tREFT_BASE & t_RFC modifications for 3D-DRAM

```
// Refresh timings
// tRFC table (unit is nanosecond!), modify the DRAM timing tRFC according to the density
constexpr int tRFC_TABLE[3][6] = {
            // 256Mb  1Gb     2Gb     4Gb     8Gb     16Gb
    [0]={   [0]=60,[1]=110,[2]=160,  [3]=260,  [4]=360,  [5]=550}, // Normal refresh (tRFC1)
    [1]={   [0]=40,[1]=80,[2]=110,   [3]=160,  [4]=260,  [5]=350}, // FGR 2x (tRFC2)
    [2]={   [0]=20,[1]=60,[2]=90,    [3]=110,  [4]=160,  [5]=260}, // FGR 4x (tRFC4)
};
```

To calculate `tREFI` for a 3D-DRAM with a 32ms refresh interval, you need to understand how the refresh interval is distributed across the refresh cycles. The `tREFI` value represents the time interval between two refresh commands.

Steps to Calculate tREFI

1. **Total Refresh Interval**: The total time within which all rows must be refreshed. In this case, it is 32ms (milliseconds).
2. **Number of Rows**: The total number of rows in the DRAM module. This is typically determined by the DRAM organization.
3. **tREFI Calculation**: Divide the total refresh interval by the number of rows to get the refresh interval time per row.

Example Calculation

Assume the DRAM has 8192 rows (which is a common number for many DRAM modules).

1. **Total Refresh Interval**: 32ms = 32,000,000ns (since 1ms = 1,000,000ns)
2. **Number of Rows**: 8192 rows

[tREFI = \frac{32,000,000 \text{ ns}}{8192 \text{ rows}} = 3906.25 \text{ ns}]

So, `tREFI` would be approximately 3906.25 nanoseconds.

Debugging

```
// tREFI(base) table (unit is nanosecond!)
int tREFI_BASE =[](int density_Mb) -> int{
    switch (density_Mb) {
        case 256:   return 15625;
        case 1024:  return 3907;
        case 2048:  return 7800;
        case 4096:  return 7800;
        case 8192:  return 7800;
        case 16384: return 7800;
        default:    return -1;
    }
}(density_Mb: m_organization.density);
```

- Since the refresh interval is related to the size of the DRAM! Thus the refresh table must also be set according to let DRAM knows how much time are needed for the refresh command.
- The Refresh period should be scaled down, due to fewer number of cells within 1 single bank
- Refer to the refresh interval of different paper, also the refresh interval for different 3D-DRAM

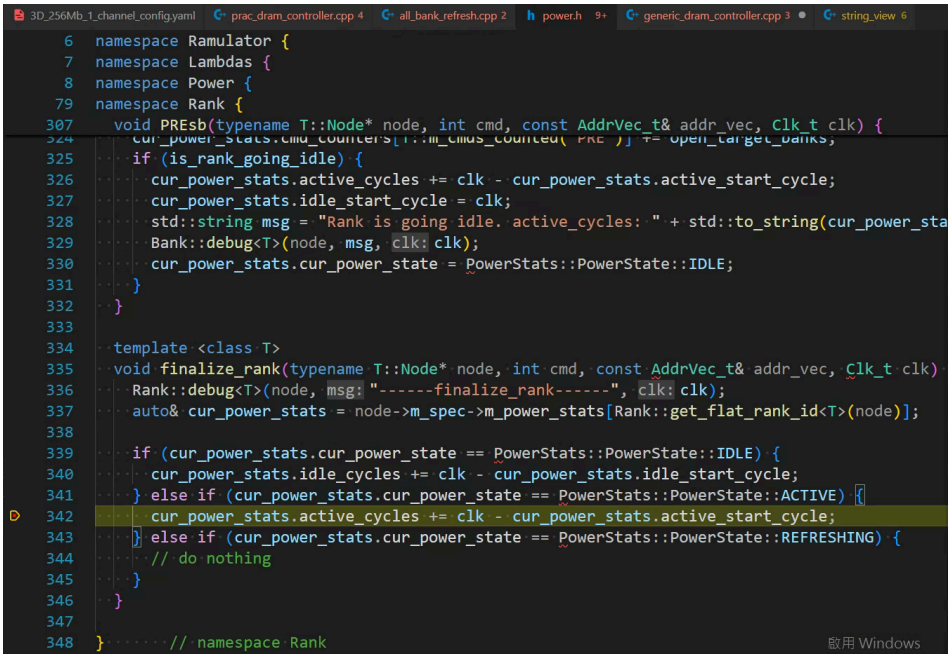
```
int m_internal_prefetch_size = [](int density_Mb) -> int {
    switch (density_Mb) {
        //! This is related to density of bank, the refresh interval, must be modified to reflect
        //! the correct value
        case 256 :  return 128;
        case 1024:  return 128;
        default:    return 8;
    }
}(density_Mb: m_organization.density);
```

- Modifying the internal prefetch size to 128

Analyze how Refresh works & how refresh energy is calculated

- The refresh power is calculated and updated within power.h, also the refresh is checked and entered within the controller for periodic refreshes
- Controller would issue refresh command once the refresh clk == m_clk

> There are three major parts that contribute to the power consumption of DRAM, background power, active power, and refresh power. Background power comes from peripheral circuitry energy consumption. Low power mode disables some peripheral circuitry to reduce background power consumption. Active power is only consumed while servicing a memory request. To service a memory request, the target row must be first activated and precharged. Since a DRAM cell is composed of an access transistor and a leaky capacitor, the DRAM is required to be refreshed periodically and thereby dissipating its refresh power. According to our evaluation, refresh power accounts up to 25–27% of the total energy consumption in a 32 GB DRAM device, and this percentage will even increase as the DRAM size increased in the future [3]. Therefore, refresh power reduction becomes a crucial issue in the future.
>
> Refreshing all rows at the same time leads to long refresh latency. Therefore, to avoid long refresh latency, the total rows in a bank are divided into 8K groups [13]. It is evenly distributing the refresh latency among runtime. That is to say, 8K auto-refresh commands are issued to refresh a subset of rows in 64 ms interval. As the example shown in Figure 3, 8K auto-refresh commands are issued every 64 ms/8K = 7.8 us.



- Notice there is an order of magnitude difference in energy, investigate the energy correctly, check how the CACTI gets its energy consumption values?

```
/*********************************************
*                Organization
*********************************************/
int m_internal_prefetch_size = [](int density_Mb) -> int {
    switch (density_Mb) {
      //! This is related to density of bank, the refresh interval, must be modified to reflect
      //! the correct value
      case 256 :   return 128;
      case 1024:   return 128;
      default:     return 8;
    }
}(density_Mb: m_organization.density);

inline static constexpr ImplDef m_levels = {
    "channel", "rank", "bankgroup", "bank", "row", "column",
};
```

- prefetch_size here is used to define the basic address mapping in the linear address mapper.

Plugins

- CMD counts

```
Controller:
    impl: Generic
    Scheduler:
        impl: FCFS
    RefreshManager:
        impl: AllBank
    RowPolicy:
        impl: OpenRowPolicy
        cap: 4
    plugins:
        - ControllerPlugin:
            impl: CommandCounter
            path: ./cmd_records/1Gb_1ch.cmds
            commands_to_count:
                - REFab

AddrMapper:
    impl: RoBaRaCoCh
```

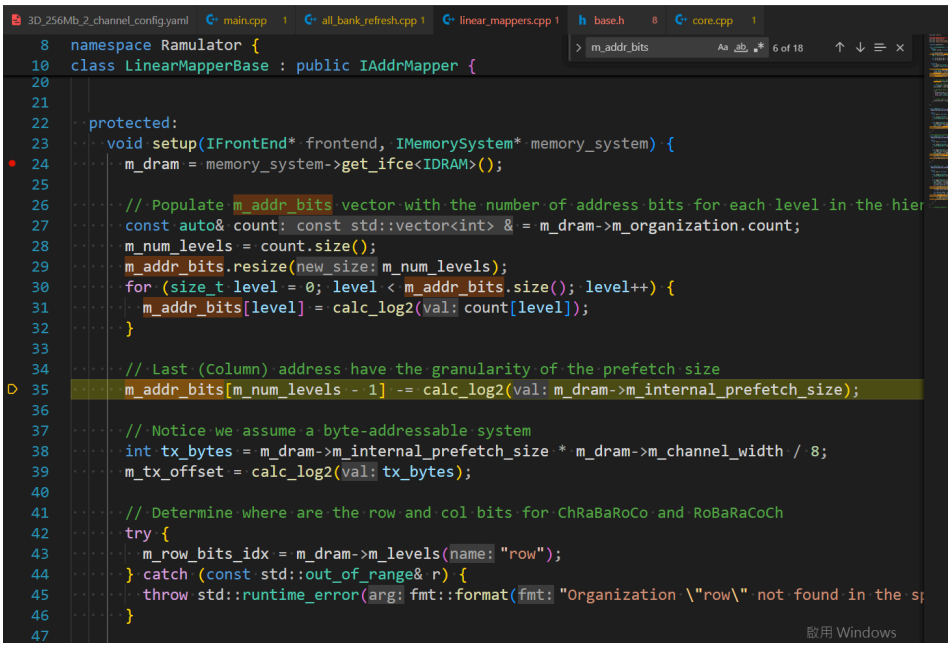
Sanity check

- Look at the request buffer and trace to ensure that the memory controller is doing the right thing
- Look at how refresh works within the memory controller later tries to modify it.

Using traces

- Modified PARSEC & Rodinia for references
- PRIM MLP workload exist within, can try to extract it out to speedup the process

Address mapping



- Note that the column index must be subtracted from the prefetch size.

Check for in order and out of order for multiple banks connections also how the frontend interacts with the memory system, THEY MUST BE IN ORDER

- The detail descriptions of the system must be described and plot out.
- Check if the LD,ST can returns the memory call back function, or perhaps modify the frontend through the call back function system.

3. DRAMPower

- Modify the DRAM power parameter to the analyzed 3D-DRAM modeling adding TSV and energy components into consideration.
- From the modified refresh interval, use DRAM power to estimate the energy consumption from traces.
  - See if DDR3,DDR4 power models working or not

There are five types of IDD current values that we measure: (1) idle: IDD2N, IDD3N; (2) activate and precharge: IDD0, IDD1; (3) read and write: IDD4R, IDD4W, IDD7; (4) refresh: IDD5B; and (5) power-down mode: IDD2P1.

4.1 Idle (IDD2N/IDD3N)

We start by measuring the idle (i.e., standby) current. JEDEC defines two idle current measurement loops: (1) IDD2N, which measures the current consumed by the module when *no* banks have a row activated; and (2) IDD3N, which measures the current consumed by the module when *all* banks have a row activated.

Figure 5 shows the average current measured during the IDD2N loop. We use *box plots* to show the distribution across all modules from each vendor. Each box illustrates the quartiles of the distribution, and the whiskers illustrate the minimum and maximum values. We make two key observations from this data. First, *there is non-trivial variation in the amount of current consumed from module to module for the same DRAM vendor.* The amount of variation is different for each vendor, and the range normalized to the datasheet current varies from 14.7% for Vendor A to 37.5%

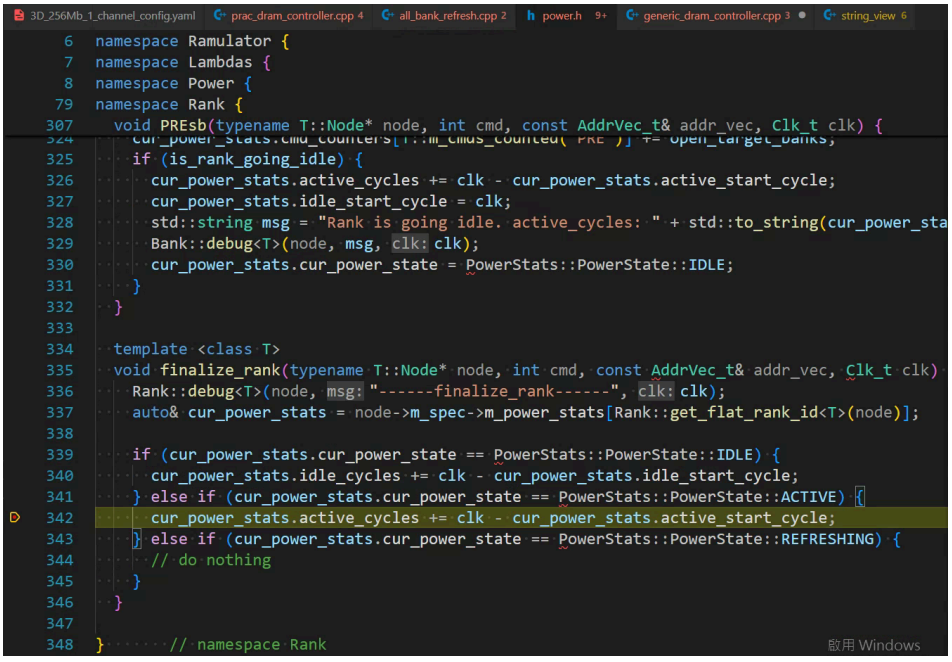
- This is obtained from the VAMPIRE paper. DRAM Vendors uses worst case current to calculate their power consumption in idle & active phase.

4. Refresh Power

- The calculation of refresh power through refresh counts, understands how refresh mechanism is implemented within Ramulator2
- See how the refresh power is calculated within the Ramulator
- Consider Ideal & extreme refresh power to do the analysis.
- Seperate out the total time spent in refresh and the total energy spent for refresh

Power.h

- Calculates the idle cycles, active cycles then approximate the energy for a certain rank



5. Architecture of Garbage Collector Refresh controller

- Considerations
  - Table stored in DRAM IO logic die or in the memory controller?
  - SRAM bit map on DRAM logic die or controller? RBR or OBR, refer to 3D-DRAM with OS page
  - Command leading to additional power consumption and bandwidth influence?
  - Dedicated garbage collection command or adding simply a bit field parrallelizing it?
- See modeling as a plugin or modeling as a whole DRAM system
- See how all_bank_refresh is modeled, modify it to mask the refresh through command accesses

CBR, ROR differences[3]

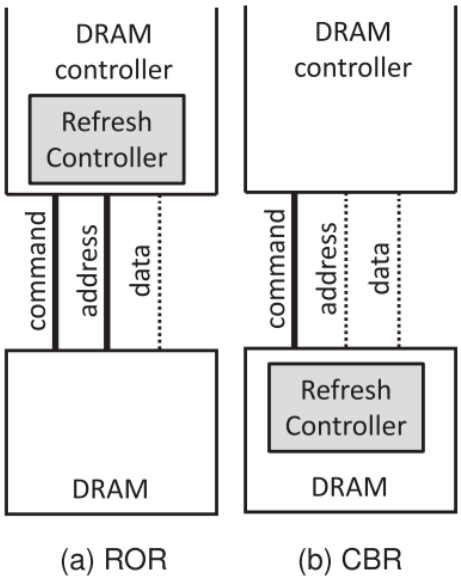


Fig. 3. Conceptual comparison of ROR and CBR refresh policies. (a) The ROR policy has a refresh counter on the DRAM controller and uses the command and address interfaces. (b) By contrast, CBR has a refresh counter on the DRAM device and uses only the command interface.

- Using CBR & ROR has different effect, CBR should modify the refresh controller. Had better use CBR, and implement this onto the DRAM logic layer. Approximate the area of this type of SRAM
- Support CBR is a much better idea for modifying other DRAM. Add an additional logic to block the refresh of a certain command

RART(Row address register table)[4]

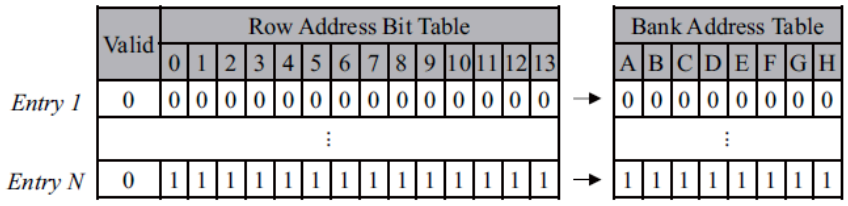


Fig. 2. Structure of RART(for a 512 MB DRAM device). The RART consists of one row and one bank address bit table, which are matched each other.

- Implement the RART onto the DDR4 peripheral side to allow easier refresh operations.

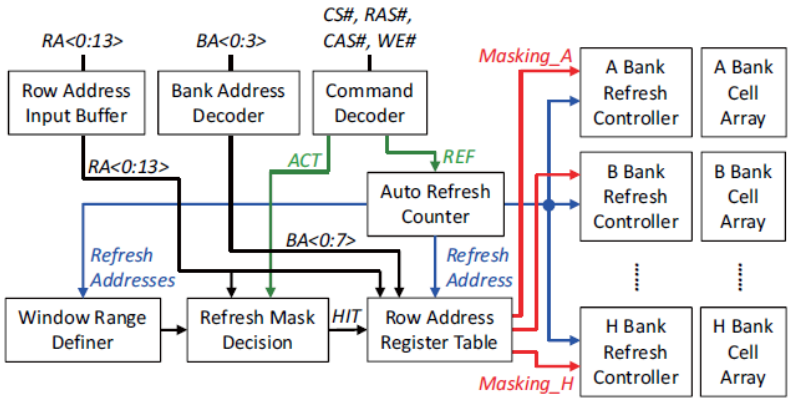
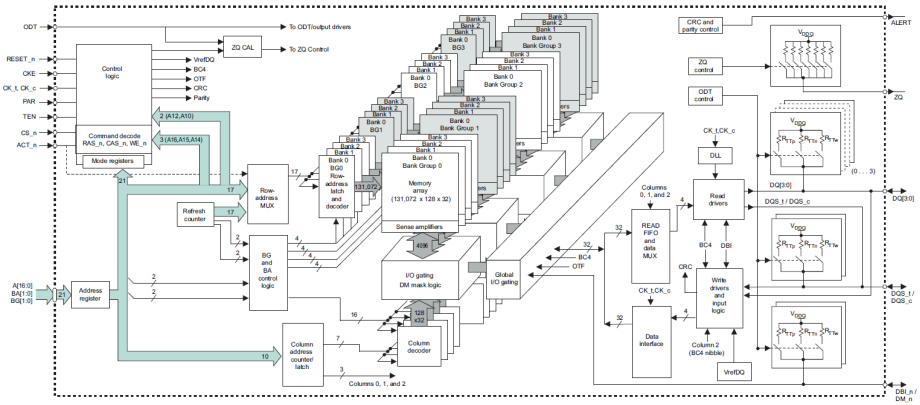


Fig. 3. Overall architecture of TWW. RART stores the row addresses not to be refreshed and it is most important for low cost overhead.

- Modify the command decoder, adding another bits indicating the row & bank needed to be mark no refresh.

DDR4 inner structure

Figure 2: 2 Gig x 4 Functional Block Diagram



- Modify the refresh counter, add an additional logic controlling the RART of the DDR4.
- The refresh logic goes through the RART extract the needed information about which row to refresh
- The RART would be updated under two condition, the present of Garbage bit. The write command toward that certain page of a certain bank.

7. MCPAT

- Some paper uses MCPAT to calculate the cost of memory controller on the logic die. Estimate the cost and efficiency using this tool.

8. PARSEC Workload

- PARSEC

9. Multi-refresh(Zero mapping to combat VRT)

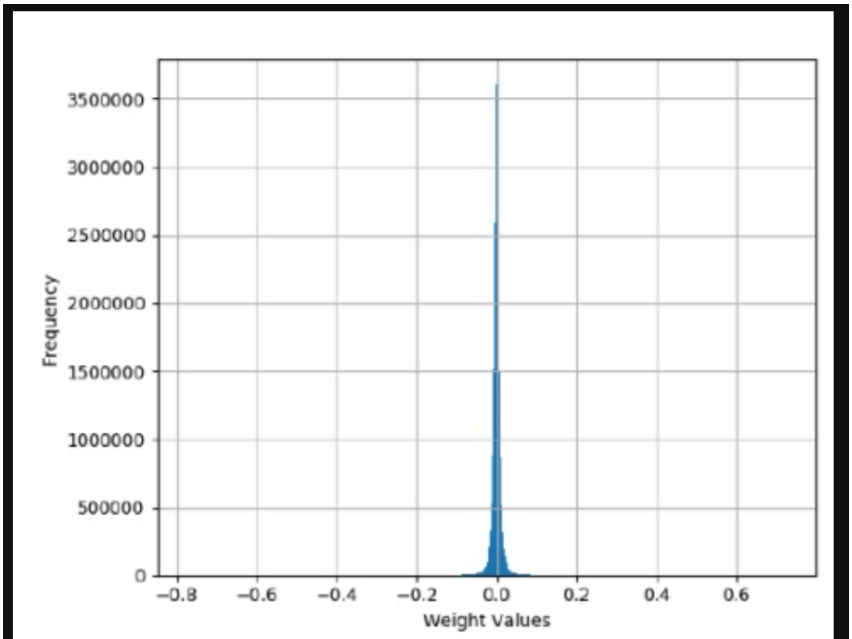
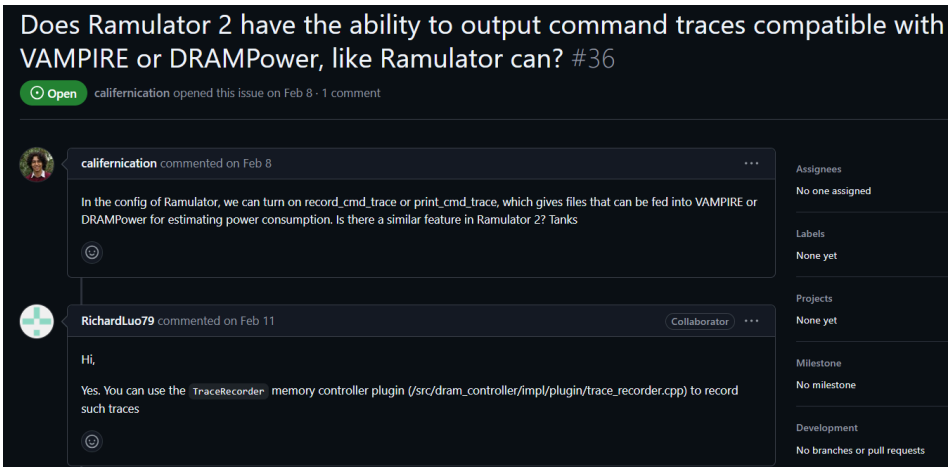
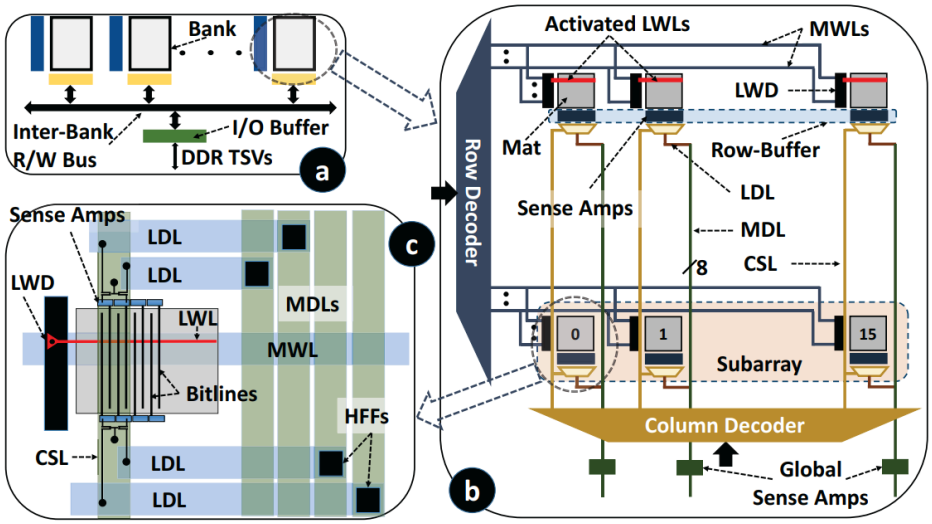


Figure 1: Histogram of frequencies of the weights in the Llama2 7B layer 0 Wq matrix.



Subarray Level Analysis



- Usually subarray is refered to as a collection of mats in normal Bank configuration, however in CACTI this is not the case.

$$N_{\text{subbanks}} = \frac{N_{\text{dbl}}}{2} \quad (1)$$

$$N_{\text{mats-in-subbank}} = \frac{N_{\text{dwl}}}{2} \quad (2)$$

Figure 10 shows different partitions of the same bank. The partitioning parameters are labeled alongside. Table 1 lists various organizational parameters associated with a data array.

- In CACTI, subbanks is the same as subarray in normal contents, thus to perform analysis, must sweep through the number of subbanks to observe the changes to the whole system

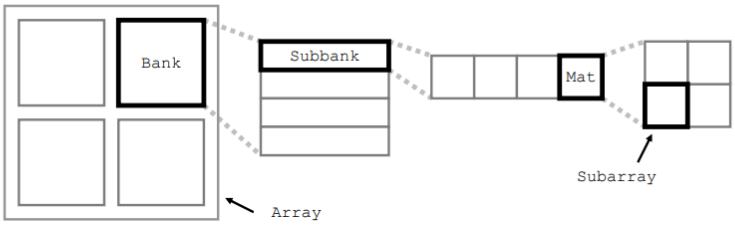


Figure 3: Layout of an example array with 4 banks. In this example each bank has 4 subbanks and each subbank has 4 mats.

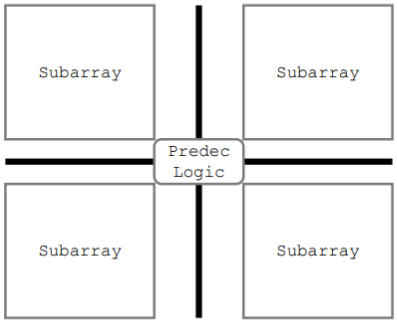


Figure 4: High-level composition of a mat.

- CACTI reference, in this example a mat contains 4 memory arrays(subarrays),

References

[1] Understanding the Design Space of
DRAM-Optimized Hardware FFT Accelerators

[2]

GitHub - CMU-SAFARI/Sectored-DRAM: A new DRAM substrate that mitigates the excessive energy consumption from both (i) transmitting unus
A new DRAM substrate that mitigates the excessive energy consumption from both (i) transmitting unused data on the memory channel and (ii) activating a dispro

https://github.com/CMU-SAFARI/Sectored-DRAM/tree/main

[3] **EXTREME: Exploiting Page Table for Reducing Refresh Power of 3D-Stacked DRAM Memory**

[4] Timing Window Wiper : A New Scheme for Reducing Refresh Power of
DRAM

[5] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, *CACTI 5.0: An Integrated Cache Timing, Power, and Area Model*, HP Laboratories, Tech. Rep. HPL-2007-167, 2007.