✏️

CACTI-3DD

Created	@October 6, 2024 8:56 AM
Class	

Get the runnable CACTI-3DD from Sectored DRAM

```
https://github.com/CMU-SAFARI/Sectored-DRAM
```

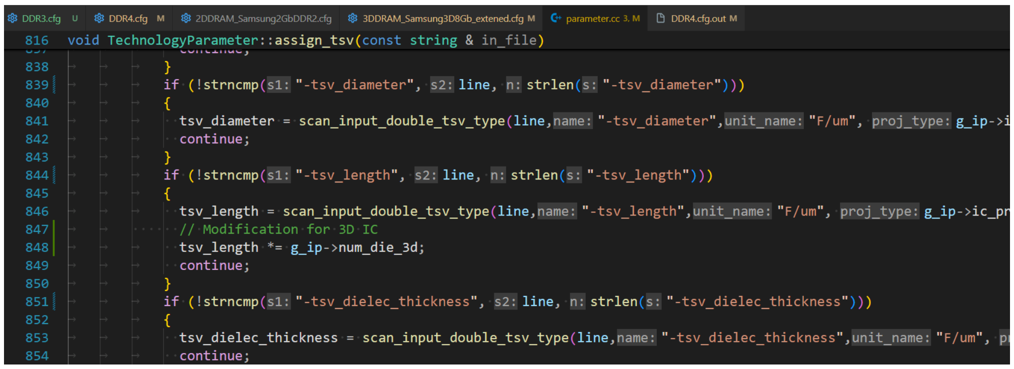
- Run the areapower.py to generate the runnable makeFile for cacti
- Modify the code containing the sectored DRAM modification, which are the sense amplifier modifications.

```
https://github.com/HewlettPackard/cacti
```

- Replace the mat.cc with the one from CACTI-3DD

Bug for modifications

- Modification for tsv_length must be modified to get correct timing constraints when inputing the stacking parameter.
- https://github.com/HewlettPackard/cacti/issues/2

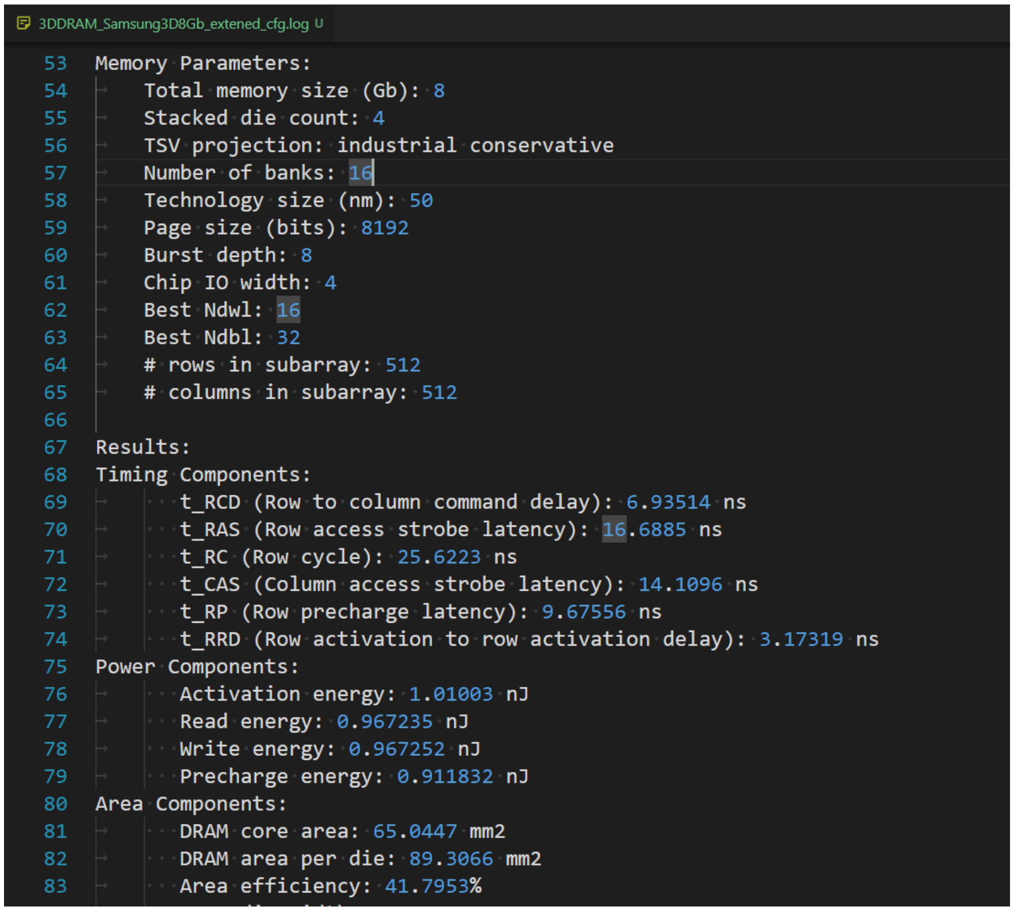


```
void TechnologyParameter::assign_tsv(const string & in_file)
            }
            if (!strncmp(s1: "-tsv_diameter", s2: line, n: strlen(s: "-tsv_diameter")))
            {
                tsv_diameter = scan_input_double_tsv_type(line,name: "-tsv_diameter",unit_name: "F/um", proj_type: g_ip->i
                continue;
            }
            if (!strncmp(s1: "-tsv_length", s2: line, n: strlen(s: "-tsv_length")))
            {
                tsv_length = scan_input_double_tsv_type(line,name: "-tsv_length",unit_name: "F/um", proj_type: g_ip->ic_pr
                // Modification for 3D IC
                tsv_length *= g_ip->num_die_3d;
                continue;
            }
            if (!strncmp(s1: "-tsv_dielec_thickness", s2: line, n: strlen(s: "-tsv_dielec_thickness")))
            {
                tsv_dielec_thickness = scan_input_double_tsv_type(line,name: "-tsv_dielec_thickness",unit_name: "F/um", pr
                continue;
```

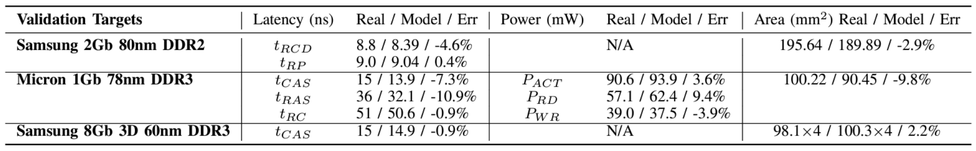
- Otherwise the CACTI-3DD cannot be run correctly

3D-DRAM Samsung.cfg modification

- https://github.com/HewlettPackard/cacti/issues/6
- Such that you can verify and get similiar results for the parameter of 3D-Samsung model timing constraints.


3DDRAM_Samsung3D8Gb_extened_cfg.log

```
Memory Parameters:
    Total memory size (Gb): 8
    Stacked die count: 4
    TSV projection: industrial conservative
    Number of banks: 16
    Technology size (nm): 50
    Page size (bits): 8192
    Burst depth: 8
    Chip IO width: 4
    Best Ndwl: 16
    Best Ndbl: 32
    # rows in subarray: 512
    # columns in subarray: 512

Results:
Timing Components:
       t_RCD (Row to column command delay): 6.93514 ns
       t_RAS (Row access strobe latency): 16.6885 ns
       t_RC (Row cycle): 25.6223 ns
       t_CAS (Column access strobe latency): 14.1096 ns
       t_RP (Row precharge latency): 9.67556 ns
       t_RRD (Row activation to row activation delay): 3.17319 ns
Power Components:
       Activation energy: 1.01003 nJ
       Read energy: 0.967235 nJ
       Write energy: 0.967252 nJ
       Precharge energy: 0.911832 nJ
Area Components:
       DRAM core area: 65.0447 mm2
       DRAM area per die: 89.3066 mm2
       Area efficiency: 41.7953%
```



Validation Targets	Latency (ns)	Real / Model / Err	Power (mW)	Real / Model / Err	Area (mm^2) Real / Model / Err
Samsung 2Gb 80nm DDR2	t_{RCD}	8.8 / 8.39 / -4.6%		N/A	195.64 / 189.89 / -2.9%
	t_{RP}	9.0 / 9.04 / 0.4%			
Micron 1Gb 78nm DDR3	t_{CAS}	15 / 13.9 / -7.3%	P_{ACT}	90.6 / 93.9 / 3.6%	100.22 / 90.45 / -9.8%
	t_{RAS}	36 / 32.1 / -10.9%	P_{RD}	57.1 / 62.4 / 9.4%	
	t_{RC}	51 / 50.6 / -0.9%	P_{WR}	39.0 / 37.5 / -3.9%	
Samsung 8Gb 3D 60nm DDR3	t_{CAS}	15 / 14.9 / -0.9%		N/A	98.1×4 / 100.3×4 / 2.2%

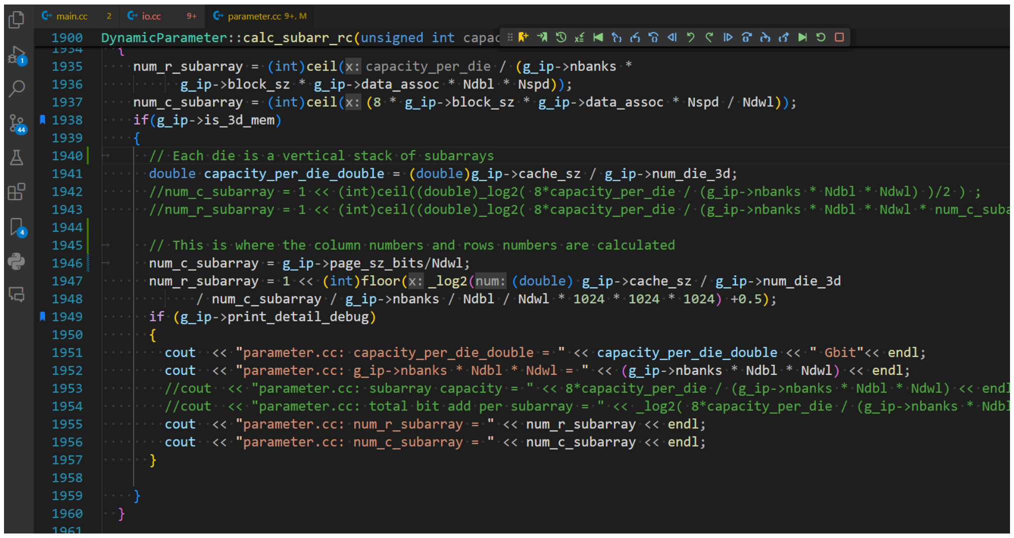
TABLE III

- The CACTI-3DD does not support 60nm tech, thus uses 50nm, can notice that it is actually quite accurate.

CACTI-IO Technical Report

- https://escholarship.org/content/qt6f6904sc/qt6f6904sc_noSplash_d5a14fc9a279c9b4c6c4b353a7e61ad5.pdf?t=rri97w

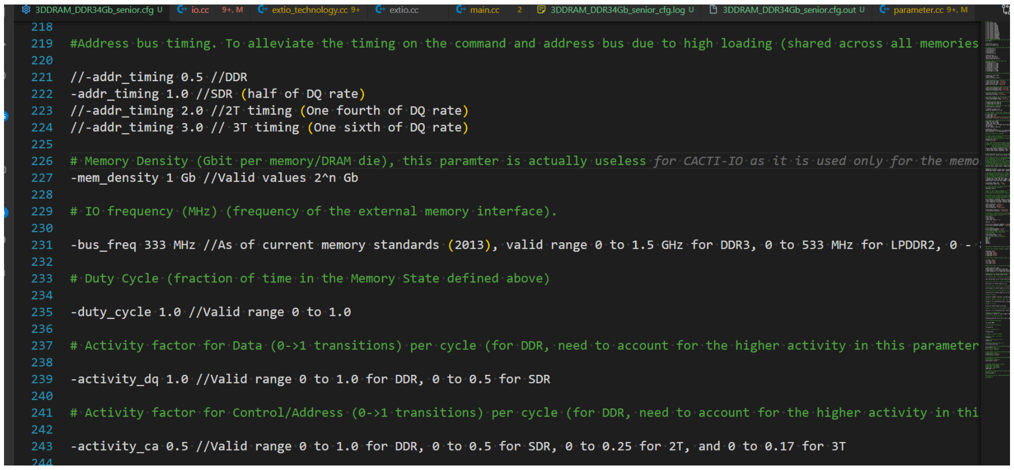
Row, Column array number



```
DynamicParameter::calc_subarr_rc(unsigned int capac
    num_r_subarray = (int)ceil(x: capacity_per_die / (g_ip->nbanks *
            g_ip->block_sz * g_ip->data_assoc * Ndbl * Nspd));
    num_c_subarray = (int)ceil(x: (8 * g_ip->block_sz * g_ip->data_assoc * Nspd / Ndwl));
    if(g_ip->is_3d_mem)
    {
      // Each die is a vertical stack of subarrays
      double capacity_per_die_double = (double)g_ip->cache_sz / g_ip->num_die_3d;
      //num_c_subarray = 1 << (int)ceil((double)_log2( 8*capacity_per_die / (g_ip->nbanks * Ndbl * Ndwl) )/2 ) ;
      //num_r_subarray = 1 << (int)ceil((double)_log2( 8*capacity_per_die / (g_ip->nbanks * Ndbl * Ndwl * num_c_suba

      // This is where the column numbers and rows numbers are calculated
      num_c_subarray = g_ip->page_sz_bits/Ndwl;
      num_r_subarray = 1 << (int)floor(x: _log2(num: (double) g_ip->cache_sz / g_ip->num_die_3d
            / num_c_subarray / g_ip->nbanks / Ndbl / Ndwl * 1024 * 1024 * 1024) +0.5);
      if (g_ip->print_detail_debug)
      {
        cout  << "parameter.cc: capacity_per_die_double = " << capacity_per_die_double << " Gbit"<< endl;
        cout  << "parameter.cc: g_ip->nbanks * Ndbl * Ndwl = " << (g_ip->nbanks * Ndbl * Ndwl) << endl;
        //cout  << "parameter.cc: subarray capacity = " << 8*capacity_per_die / (g_ip->nbanks * Ndbl * Ndwl) << endl
        //cout  << "parameter.cc: total bit add per subarray = " << _log2( 8*capacity_per_die / (g_ip->nbanks * Ndbl
        cout  << "parameter.cc: num_r_subarray = " << num_r_subarray << endl;
        cout  << "parameter.cc: num_c_subarray = " << num_c_subarray << endl;
      }

    }
  }
```

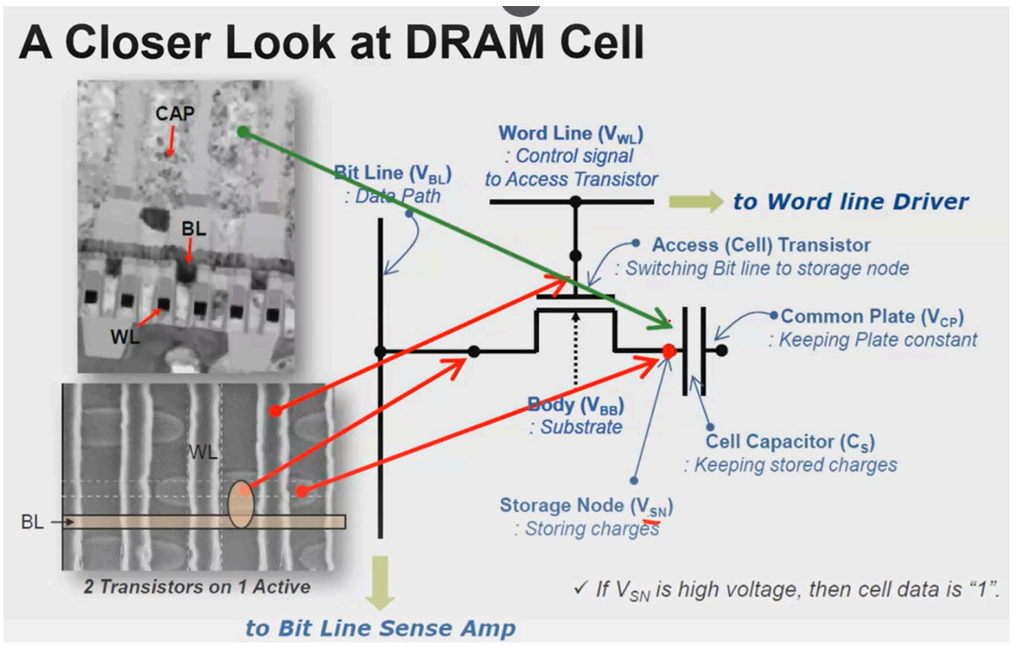
- These are for subarrays calculations and output the correct subarray number in parameter.cc
- Senior uses 128×128, adjust it to 128×128. 1Gb per die.

Memory density parameter(Not used)



```

#Address bus timing. To alleviate the timing on the command and address bus due to high loading (shared across all memories

//-addr_timing 0.5 //DDR
-addr_timing 1.0 //SDR (half of DQ rate)
//-addr_timing 2.0 //2T timing (One fourth of DQ rate)
//-addr_timing 3.0 // 3T timing (One sixth of DQ rate)

# Memory Density (Gbit per memory/DRAM die), this paramter is actually useless for CACTI-IO as it is used only for the memo
-mem_density 1 Gb //Valid values 2^n Gb

# IO frequency (MHz) (frequency of the external memory interface).

-bus_freq 333 MHz //As of current memory standards (2013), valid range 0 to 1.5 GHz for DDR3, 0 to 533 MHz for LPDDR2, 0 -

# Duty Cycle (fraction of time in the Memory State defined above)

-duty_cycle 1.0 //Valid range 0 to 1.0

# Activity factor for Data (0->1 transitions) per cycle (for DDR, need to account for the higher activity in this parameter

-activity_dq 1.0 //Valid range 0 to 1.0 for DDR, 0 to 0.5 for SDR

# Activity factor for Control/Address (0->1 transitions) per cycle (for DDR, need to account for the higher activity in thi

-activity_ca 0.5 //Valid range 0 to 1.0 for DDR, 0 to 0.5 for SDR, 0 to 0.25 for 2T, and 0 to 0.17 for 3T
```

Ndwl,Ndbl,Nspd & its relationship with subbanks



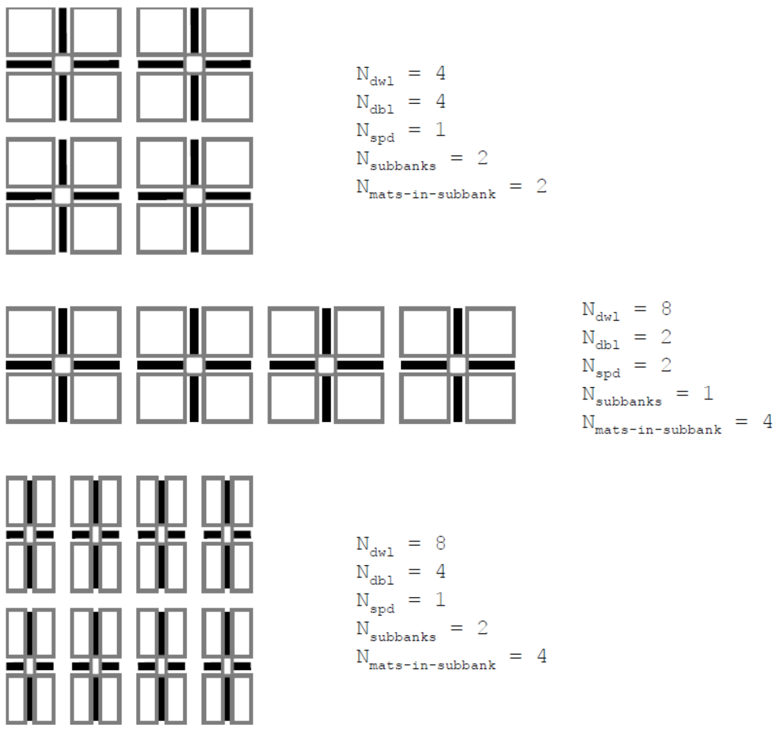
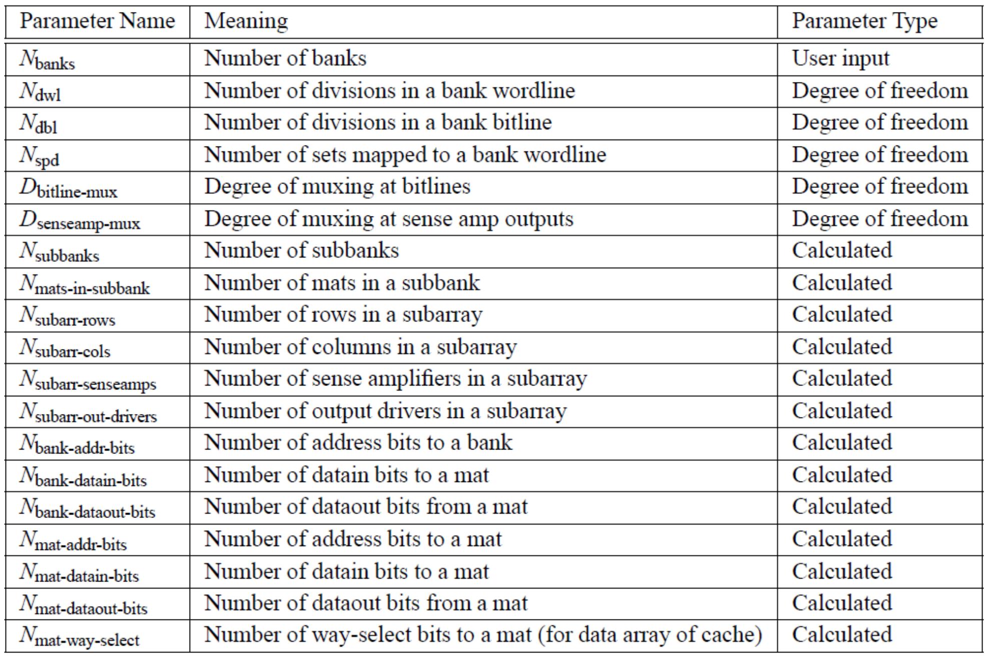


Figure 10: Different partitions of a bank.

In order to calculate the optimal organization based on a given objective function, like earlier versions of CACTI [30, 38, 40, 47], each bank is associated with partitioning parameters N_{dwl}, N_{dbl} and N_{spd}, where N_{dwl} = number of segments in a bank wordline, N_{dbl} = number of segments in a bank bitline, and N_{spd} = number of sets mapped to each bank wordline.



Parameter Name	Meaning	Parameter Type
N_{banks}	Number of banks	User input
N_{dwl}	Number of divisions in a bank wordline	Degree of freedom
N_{dbl}	Number of divisions in a bank bitline	Degree of freedom
N_{spd}	Number of sets mapped to a bank wordline	Degree of freedom
$D_{\text{bitline-mux}}$	Degree of muxing at bitlines	Degree of freedom
$D_{\text{senseamp-mux}}$	Degree of muxing at sense amp outputs	Degree of freedom
N_{subbanks}	Number of subbanks	Calculated
$N_{\text{mats-in-subbank}}$	Number of mats in a subbank	Calculated
$N_{\text{subarr-rows}}$	Number of rows in a subarray	Calculated
$N_{\text{subarr-cols}}$	Number of columns in a subarray	Calculated
$N_{\text{subarr-senseamps}}$	Number of sense amplifiers in a subarray	Calculated
$N_{\text{subarr-out-drivers}}$	Number of output drivers in a subarray	Calculated
$N_{\text{bank-addr-bits}}$	Number of address bits to a bank	Calculated
$N_{\text{bank-datain-bits}}$	Number of datain bits to a mat	Calculated
$N_{\text{bank-dataout-bits}}$	Number of dataout bits from a mat	Calculated
$N_{\text{mat-addr-bits}}$	Number of address bits to a mat	Calculated
$N_{\text{mat-datain-bits}}$	Number of datain bits to a mat	Calculated
$N_{\text{mat-dataout-bits}}$	Number of dataout bits from a mat	Calculated
$N_{\text{mat-way-select}}$	Number of way-select bits to a mat (for data array of cache)	Calculated

Table 1: Organizational parameters of a data array.

Area Efficiency , Bandwidth & Energy

To calculate the bandwidth of 3D-DRAM (like HBM, HBM2, or HBM3), you can use the following formula:

$$\text{Bandwidth} = \text{Data Rate} \times \text{Bus Width} \times \text{Number of Channels}$$

Explanation of Terms:

1. **Data Rate**: This is the speed at which data can be transferred per pin, typically measured in gigabits per second (Gbps). For example, HBM2 might have a data rate of 2 Gbps to 3.2 Gbps per pin.
2. **Bus Width**: This refers to the width of the data bus per channel, typically 128 bits (or 16 bytes) for HBM technologies.
3. **Number of Channels**: The number of independent channels available in the memory module. HBM2 usually has 8 channels, while HBM3 might have more.

Example Calculation:

Suppose you have an HBM2 stack with the following specs:

- **Data Rate**: 2 Gbps
- **Bus Width**: 128 bits (16 bytes)
- **Number of Channels**: 8

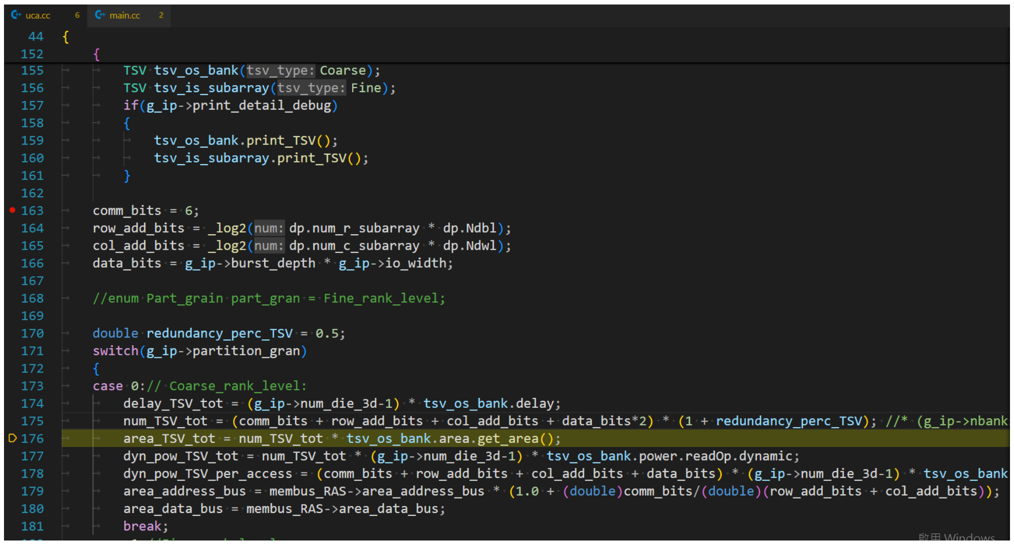
To calculate the data rate of a customized 3D-DRAM, you can use the following approach:

$$\text{Data Rate} = \frac{\text{Clock Frequency} \times \text{Data Transfers Per Clock Cycle}}{\text{Bus Width}}$$

Explanation of Terms:

1. **Clock Frequency**: The operating frequency of the DRAM, typically measured in megahertz (MHz) or gigahertz (GHz). This determines how many times data can be transferred per second.
2. **Data Transfers Per Clock Cycle (MT/s)**: This depends on the technology:
    - **DDR (Double Data Rate)**: Transfers data twice per clock cycle (once on the rising edge and once on the falling edge).
    - **QDR (Quad Data Rate)**: Transfers data four times per clock cycle.
    - If it's a customized 3D-DRAM, decide how many transfers you want per clock cycle.
3. **Bus Width**: The width of the data bus per channel, measured in bits. This determines how much data is moved per transfer. Typical 3D-DRAM configurations use 128-bit buses, but you can customize this.

Number of TSV



```
{
    {
        TSV tsv_os_bank(tsv_type: Coarse);
        TSV tsv_is_subarray(tsv_type: Fine);
        if(g_ip->print_detail_debug)
        {
            tsv_os_bank.print_TSV();
            tsv_is_subarray.print_TSV();
        }

    comm_bits = 6;
    row_add_bits = _log2(num: dp.num_r_subarray * dp.Ndbl);
    col_add_bits = _log2(num: dp.num_c_subarray * dp.Ndwl);
    data_bits = g_ip->burst_depth * g_ip->io_width;

    //enum Part_grain part_gran = Fine_rank_level;

    double redundancy_perc_TSV = 0.5;
    switch(g_ip->partition_gran)
    {
    case 0:// Coarse_rank_level:
        delay_TSV_tot = (g_ip->num_die_3d-1) * tsv_os_bank.delay;
        num_TSV_tot = (comm_bits + row_add_bits + col_add_bits + data_bits*2) * (1 + redundancy_perc_TSV); //* (g_ip->nbank
        area_TSV_tot = num_TSV_tot * tsv_os_bank.area.get_area();
        dyn_pow_TSV_tot = num_TSV_tot * (g_ip->num_die_3d-1) * tsv_os_bank.power.readOp.dynamic;
        dyn_pow_TSV_per_access = (comm_bits + row_add_bits + col_add_bits + data_bits) * (g_ip->num_die_3d-1) * tsv_os_bank
        area_address_bus = membus_RAS->area_address_bus * (1.0 + (double)comm_bits/(double)(row_add_bits + col_add_bits));
        area_data_bus = membus_RAS->area_data_bus;
        break;
```

- The TSV bit width is determined by the burst depth * io_width, usually burst depth is 2 though, as for HBM
- These information are for a single bank