# 3D-SWIFT: a High-performance 3D-stacked Wide IO DRAM*

Tao Zhang, Cong Xu, Yuan Xie
Penn State University
{zhangtao, czx102, yuanxie}@cse.psu.edu

Ke Chen
Oracle Corporation
ke.c.chen@oracle.com

Guangyu Sun
Peking University
gsun@pku.edu.cn

## ABSTRACT

Wide IO has been standardized as a low-power, high-bandwidth DRAM for embedded system. The performance of Wide IO, however, is limited by the power constraint and unexploited fine-grained memory parallelism. In this work, we propose a novel architecture, 3D-SWIFT, that achieves high access parallelism by partitioning a memory bank into sub-banks with a fine access granularity, which takes advantage of 3D die-stacking. The power constraint is naturally eliminated by the fine-grained structure due to the reduced activation power. Moreover, we propose sub-bank autonomy and introduce corresponding management policies to enable an intelligent interface protocol. Thanks to sub-rank autonomy, the overhead of tracking huge concurrent accesses in the memory controller is significantly reduced, making our 3D-SWIFT architecture scalable for future memory systems. We evaluate our 3D-SWIFT and the results show that 3D-SWIFT can achieve 87.6% performance improvement compared to the state-of-the-art Wide IO.

## 1. INTRODUCTION

As an early adoption of 3D integration, 3D-stacked DRAM is a promising technology for overcoming the barriers in DRAM scaling, thereby offering an opportunity to break the "memory wall" with improved DRAM cell density (capacity) and wire routing resources (connectivity), as well as reduced wire length (latency and power) [1, 2, 3, 4, 5, 6]. In particular, Wide IO DRAM [1] has been standardized by JEDEC as a high bandwidth and low-power 3D DRAM for embedded SoC system. A Wide IO DRAM has four channels that are independent of each other. Each channel is 128-bit wide with single data rate. The Wide IO follows the low-power design methodology of LPDDR so that it removes the delay lock loop (DLL) and on-die termination (ODT) logic and applies lower supply voltage to achieve low static power.

Unfortunately, Wide IO does not take full advantage of the die-stacking since it has to comply with conventional DRAM structure where the bank size is large and the number of banks is small. In addition, the increasing number of channels in turn mandates more memory controllers (MCs) and interconnects between them, bringing in issues that may hinder its popularity. As 3D die-stacking eliminates (or at least alleviates) most of the physical limitations, it provides the opportunity to further enhance memory parallelism with novel memory architecture. In this work, we propose *3D-SWIFT* to enable a fine-grained memory architecture so as to improve performance in Wide IO. The contributions of our work can be summarized as follows.

- **Fine-grained 3D DRAM structure.** We find that the power constraint can suppress the performance of Wide IO DRAM. To eliminate the power constraint, a bank in 3D-SWIFT is further divided into multiple sub-banks and each sub-bank has the ability to independently serve a memory request and provide an entire cacheline. Once there is a memory access, only the target sub-bank is activated. In this way, the fine-grained access reduces the activation/precharge current and thus enables higher memory parallelism. To our best knowledge, this is the first work that takes into account power constraint in designing high-performance Wide IO.

- **Sub-bank "autonomy".** Leveraging the close-page row-buffer management policy, sub-bank autonomy is developed to combine the commands `RAS`, `CAS` and `PRE` as `REQ` to activate a sub-bank, carry on the data burst, and close the sub-bank *automatically*. We devise a packet-based interface protocol accordingly to simplify the memory transaction. Moreover, by making use of the rich routing resource on the logic die, a wide data bus is employed to deliver a full cacheline in one cycle.

- **Simplified memory controller design.** 3D-SWIFT takes into account the design complexity of MCs. By leveraging the sub-bank autonomy and packet-based protocol, a MC of 3D-SWIFT can be integrated into the processor die so that it is visible to the processor for the system-level optimization.

## 2. BACKGROUND AND MOTIVATION

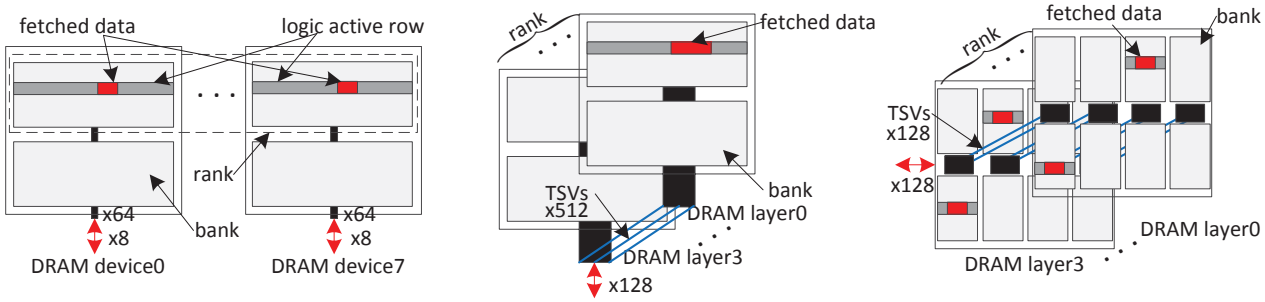
In conventional DDR*x* family, the pin count constraint is a major factor that limits the memory bandwidth due to the packaging limitation. The long off-chip memory bus implemented with the transmission line mandates the trade-off between a high operating frequency and the channel capacity because of the load and signal integrity. In addition, since all DRAM devices (chips) in a rank work in lockstep, DDR*x* exposes an extremely large logic row while only a small fraction of data is delivered for each memory access (see Figure 1a). Even though the data overfetching is beneficial for applications with high spatial locality, the fixed device association limits bank-level parallelism since few banks are visible to the MC.

Once we relocate the DRAM from off-chip to on-chip, multiple DRAM dies can be stacked together to increase the cell density. The Wide IO (Figure 1b) is applicable for memory bandwidth improvement for two reasons. First of all, the pin-out constraint is eliminated and the on-chip I/O bus replaces the long off-chip transmission lines. Moreover, the compact DRAM layout reduces

*This work is supported in part by SRC Grants, NSF (1017277, 1213052, and 1218867), NSF China (61202072), 863 Program of China (2013AA013201), and AMD Gift Grant.


*GLSVLSI'14,* May 21–23, 2014, Houston, Texas, USA.

http://dx.doi.org/10.1145/2591513.2591529.




(a) 2D DRAM data access (b) 3D Wide IO data access (c) 3D-SWIFT data access

Figure 1: Data access mechanisms in different memories. (a) DDR3: the data is distributed over all devices in a rank. Only a small fraction of row is fetched in each memory access; (b) Wide IO: the data is placed in a large row and no individual device exists. Still only a portion of data is fetched; (c) 3D-SWIFT: the width of fetched data matches cacheline width. The rest of a row is idle.

memory access latency as well as power consumption. Despite the benefits a Wide IO provides, some new problems emerges and need to be solved carefully:

• **More restricted power and current density constraints.** A new problem in Wide IO is the requirement of low-power design, which is critical for the success of 3D DRAM due to the increasing power density [7]. Since no DLL or ODT is employed, the background power is dramatically reduced and the burst and activation/precharge power now starts to dominate in the Wide I/O DRAM. As illustrated in Figure 1b, the entire row is activated but only a portion of data is fetched, which still indicates large power redundancy. In addition, considering the power supply challenges in 3D ICs [8, 9], the power constraint becomes more restricted in 3D context. Therefore, Wide IO has conservatively switched the restriction on memory accesses from four-bank activation window (tFAW) to two-bank activation window (tTAW) and extended the window width from 30ns to 50ns (the lower the better for performance) [1].

As shown in Figure 2, with a large tTAW constraint (tTAW=50ns), no further improvement is observed even if the bank-level parallelism is augmented as the bank number increases from 8 to 64. In other words, the power constraint can significantly suppress the bank-level-parallelism in a rank (channel), which has also been verified in [10] when close-page row buffer management policy is applied. Once the power constraint is eliminated (tTAW=0ns), the performance gain can be up to 44% when increasing the bank number from 8 to 64. Moreover, by getting rid of tTAW constraint, the bank-level parallelism can further provide 14% performance improvement. As a result, neglecting the power constraint and simply increasing the bank number or bank size in 3D DRAM is not a wise choice in terms of either performance or power. Instead, 3D DRAM design should carefully cope with the power constraint to make sure it does not incur performance degradation.

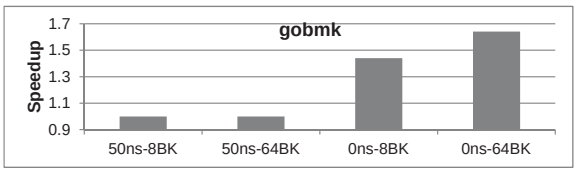



Figure 2: The impact of power constraint on 3D DRAM. The simulation is done with 8/64 banks(BK) and tTAW=50ns/0ns (see Section 5 for the details of simulation setups).

• **Increased design complexity of memory controller.** The second problem is that the increasing number of channels one Wide IO provides can result in higher design complexity of memory controllers. For example, eight independent channels have been implemented by Tezzaron [3]. While more channels can effectively improve the bandwidth, the need of multiple MCs induces noticeable area overhead when MCs are deployed in the processor die. These MCs are required to have either peer-coordination [11] or smart application assignment [12] to achieve better memory performance, which aggravates the design complexity. Alternatively, HMC [2] puts the memory controller on the logic layer, which leaves little room for a system designer to conduct further system optimization on the MC for the better communication with 3D DRAM [13].

## 3. 3D-SWIFT–A NOVEL WIDE IO DRAM

As shown in Figure 3, 3D-SWIFT re-maps a 2D DRAM that has nine devices (eight for data and one for ECC) and eight banks per device into the 3D-stacked DRAM. With respect to the process optimization, 3D-SWIFT separates control and interface logics from DRAM cells and put them in the logic layer as other state-of-the-art 3D DRAM does. Note that 3D-SWIFT is extensible when more ranks are employed. Those ranks can be either stacked vertically or placed horizontally as neighbors. Like HMC, multiple 3D-SWIFTs can be populated on the interposer to increase the memory capacity.

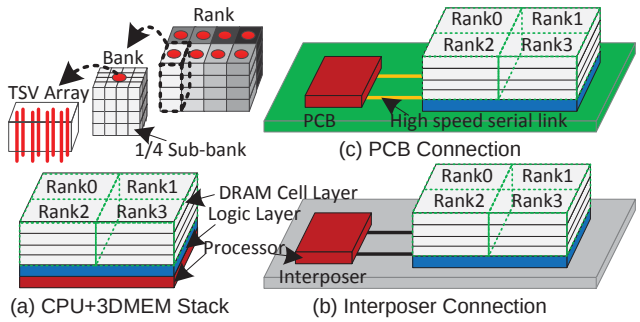



Figure 3: 3D-SWIFT memory subsystem and possible applications. (a) 3D-SWIFT directly stacks on the top of CPU; (b) 3D-SWIFT is connected to CPU by interposer; (c) 3D-SWIFT is placed on PCB with high speed links. Multiple 3D-SWIFTs can be populated to provide large memory capacity (not shown).

### 3.1 Fine-grained Memory Architecture

Without loss of generality, we use an example DRAM design with specific configurations to demonstrate the 3D-SWIFT design. To enable the fine-grained memory access, a 128M-bit (16K×8K) bank is further split into 16 identical sub-banks so that each sub-bank is sized by 16384×512 to provide a full cache line. As a consequence, a device with eight banks has 128 sub-banks and totally 2,048 sub-banks are available in 4 channels within the four layers. As multiple DRAM layers are provided, each sub-bank can be further folded [14]. For example, when there are four layers, only

four sub-banks belonging to the same bank are on one layer. Since a sub-bank can serve a memory request independently, 3D-SWIFT can significantly improve the memory parallelism. In particular, *TAW constraint can be eliminated* since ideally 3D-SWIFT allows as many as 2×16=32 sub-banks to be activated in pipeline due to the reduction of active row size[1], which always holds with 1/cycle request rate and tRC row cycle (in this work, tRC=36ns=15cycles). Furthermore, the 16 identical sub-banks guarantee that 3D-SWIFT can provide sustained bandwidth even in the the worst case, where all requests access the same bank but no sub-bank conflict occurs.

## 3.2 Sub-bank Autonomy

In traditional DDR*x* protocol, MC and DRAM work in a **master-slave** manner. As a master, MC must send various commands, including `RAS` (row activation), `CAS` (column read/write) and `PRE` (precharge), to order the target DRAM bank to complete a data transaction. These commands can only be issued under various timing constraints (tRAS, tRCD, tRP, etc.). As MC is a queuing system, commands can be rescheduled to maximize row buffer hit rate and/or bank-level parallelism. The growing design complexity of the scheduler, however, incurs large hardware overhead and makes MC error-prone.

To offload MC's complexity, 3D-SWIFT employs the sub-bank autonomy, in which each sub-bank can automatically go through the state transition loop without intervention from MC. To enable the sub-bank autonomy, one sub-bank should be aware of the timing stamp to complete the state transition. Thanks to the deterministic access latency listed above, a transition generator can be deployed to easily signal the sub-bank when to move. As a result, a new **client-server** relation is established between MC and 3D-SWIFT: Whenever there is a new memory request, MC (client) only needs to send the request to 3D-SWIFT and then wait for the sub-bank's response. Once a sub-bank (server) detects a request, the sub-bank carries on the transition and completes the data transfer automatically. As a result, 3D-SWIFT eliminates the complicated and area-consuming scheduler in MC.

## 3.3 Packet-based Interface Protocol

With respect to the sub-bank autonomy, a simple packet-based interface protocol is developed to simplify MC's interface design. As shown in Figure 4, MC sends a `REQ` with a full memory address and read/write signal to 3D-SWIFT. After tRCD+CL cycles, MC should receive the data on the data bus if it is a read, or put the write data on the data bus in the case of a write. Multiple memory requests can be issued in pipeline as long as these requests do not cause sub-bank conflict, which happens once a request reference to a busy sub-bank. As a result, even though overfetching is disabled, a series of back-to-back requests can be issued to mimic the open-page overfetching. In addition, the request pipeline can also be used to support a DMA transfer that usually has a larger data size than a cacheline, in which case MC generates multiple memory requests in burst to 3D-SWIFT.

To enable the sub-bank autonomy, the automatic state transition shown in Figure 4 must be designed so that the sub-bank knows how to move forward to complete data transaction. A dedicated transition generator is deployed as a timer to signal the sub-rank to complete transition (see Figure 5). According to the cycle number of tRC, (15cycles), a 4-bit counter is sufficient to count the cycle number in the transition generator. Whenever one of the control signals "Data phase", "Pre phase" or "Idle" is asserted, the sub-bank automatically moves to the next state till the time it returns to idle.

[1]In fact, the number of concurrent active sub-banks in 3D-SWIFT is less than 16 because of the 2-cycle data burst and the read/write turn-around overhead.

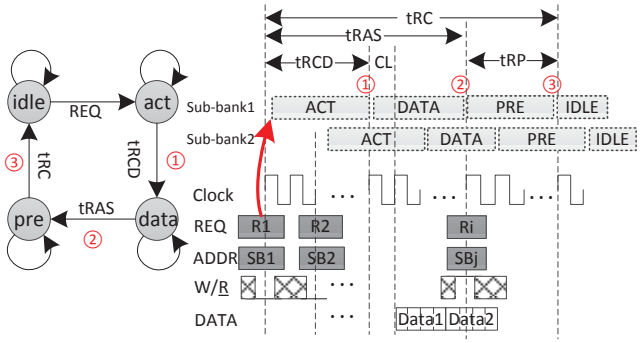



Figure 4: Packet-based interface protocol. Memory controller only issues REQ to initiate a memory access. The sub-bank can complete the data transfer and precharge to close the row automatically.

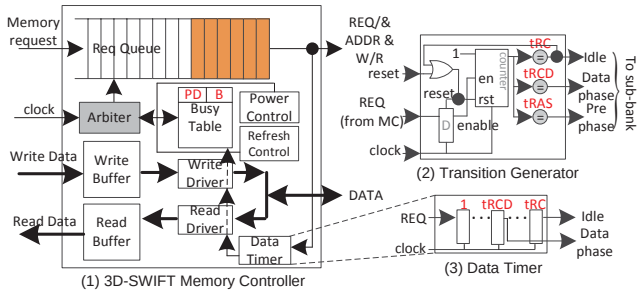



Figure 5: Memory controller design in 3D-SWIFT. (1) Memory controller; (2) Transition generator; (3) Data timer. The length of the counter and shifter are determined by the cycle number of tRC.

## 3.4 The Design of Memory Controller

As mentioned, the proposed packet-based interface protocol can significantly reduce the design complexity and area overhead of MC. Unlike a conventional MC that needs to maintain a bank status table for guarding the timing constraints, a 3D-SWIFT MC (Figure 5) only needs to affirm the freedom from sub-bank conflict with simple logic. In addition, command queues are removed to improve area efficiency. Considering the negligible footprint of the data timer, 3D-SWIFT is able to reduce MC's area and make it simpler for verification.

Different from HMC, of which the MC is transparent to the user, the smaller MC in 3D-SWIFT can be integrated into the processor die with few pin-outs. As a consequence, processor designers can apply system-level optimization on the "visible" MC. We believe this is meaningful for the wide adoption of 3D-SWIFT.

**Busy Table.** As MC is responsible to avoid sub-bank conflict, a busy table is deployed to track sub-banks' status. As shown in Figure 5, one table entry has two bits: 1) the "busy" bit tells whether the sub-bank is idle ('0') or busy ('1'); and 2) the "PowerDown" bit indicates if the sub-bank is in PowerDown mode. Since the size of busy table is relatively small (4,096bits), a 512B multi-port (4R4W) register file is simply employed. The area and power of the busy table is estimated based on [15].

**Data Timer** As mentioned in Section 3.2, MC and 3D-SWIFT have the agreement that they know exactly when to put the data on the data bus. Different from a sub-bank that gets the information from the transition generator, MC completes the data transfer with the assistance of a data timer. The data timer is in fact a bit shifter

Table 1: Hardware Implementation Summary

| Name | Type | Area | Power |
|---|---|---|---|
| Rank decoder & Device decoder | 5-32dec | 58.92μm$^2$ | 9.40μW |
| Bank decoder | 3-8dec | 13.41μm$^2$ | 2.05μW |
| Subbank decoder | 4-16dec | 27.87μm$^2$ | 4.68μW |
| Transition generator | counter | 55.75μm$^2$ | 18.71μW |
| Data timer | shifter | 93.25μm$^2$ | 35.85μW |
| Row address register | register | 101.61μm$^2$ | 35.92μW |
| Busy table [15] | regfile | 0.02mm$^2$ | 25mW |

that has tRC bits. Figure 5 illustrates the function of the data timer. A valid request sets the left-most bits to indicate a data transfer is scheduled. Once the "Data phase" is asserted, MC knows there should be a data transfer in the following cycle. The "Idle" phase notifies MC to clear the busy bit in busy table. Two data timers are used to distinguish the data read and write. They also help MC meet the bus turn-around constraint.

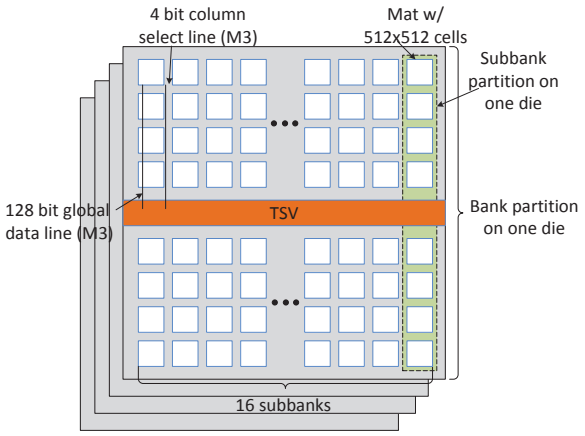



Figure 6: The Floorplan of 3D-SWIFT. A sub-bank consists of 32 mats and each mat is 256Kb (512×512).

## 4. DESIGN OVERHEAD ANALYSIS

Since DRAM is very cost-sensitive and area is the major contributor to the memory cost, we strive to minimize the area overhead of our sub-bank design in 3D-SWIFT. In this section, we elaborate the floorplan of the 3D-SWIFT with sub-bank design in addition to the description in Section 3.

**Sub-bank DRAM Floorplan.** In general, our goal is to increase the data width per mat to output all bits in the sense amplifier (i.e., row buffer) without introducing extra metal layer or area overhead on wire interconnect. Figure 6 shows the 3D-SWIFT floorplan inside a bank which consists of 16 sub-banks. As shown, each sub-bank has 8 mats on one layer, with a total size of 8Mb over four layers. Note that we keep a consistent mat size with the same 512 wordlines and 512 bitlines as in the conventional DRAM design. To minimize the overhead on intra-bank wiring incurred by the sub-bank partitioning, we leverage similar design as in the prior fine-grained 3D DRAM design [14].

In the conventional 2D DRAM, since each mat outputs 4 data bits, there are 512/4=128 column select lines per mat. Note that the numbers of column select lines (M3) and global data lines (M3) in a mat are complementary, meaning that the product of the two numbers is in fact the total number of bitlines of a mat. Alternatively, in 3D-SWIFT we can switch the usage of the wires and have 128 data lines and 4 column select lines. This is easily accomplished by flipping the direction of the three-state driver that connects the column select line and data line (the driver input is the sense amplifier output). In this way, the output of a mat is increased by 128/4=32 times without incurring any area overhead or wire routing overhead. In order to provide the full 512 bits from each mat's sense amplifier to feed an entire cache line, we use an internal burst length of 4.

Simply switching the column select line and global data line is at the risk of increasing the data line access latency, because the original column select lines are usually densely routed above the mats and are relatively narrow and slow. Specifically, the new global data line has a pitch of only 8F (while F is feature size=45nm). Fortunately, because we can partition the bank and sub-bank into different die layers (connected with TSVs), the length of the global dataline (equal to the height of the bank) is effectively shortened by four folds. In addition, we can layout the TSVs in the middle ground between the mats and share the TSVs between the half sub-banks on two sides as shown in Figure 6, which further reduces the data line length by half. In this way, the wire latency and power consumption is effectively reduced to 1/8 of the original.

**Area Overhead.** Based on the above-mentioned design, we basically eliminate the data wire routing overhead. The area overhead now comes from two parts: 1) the address/command buses and the row drivers routed to the individual sub-banks because they need to be accessed independently. 2) The TSV layout overhead. Here we assume that all sub-banks in a bank share a common 128-bit wide TSV data bus. We modified CACTI-3DD [16] to model the sub-banked 3D DRAM with the aforementioned configurations, and compare the results against a Wide IO design with the same memory capacity and mat size. The results show that our design has only a negligible 1.6% area overhead with each DRAM die size as 29.3mm$^2$.

Specifically for the TSV overhead part, according to ITRS' projection [17], the size of TSV will quickly shrink to 2-4μm. Based on the published 3D DRAM prototype [5, 6], the TSV used in this work is set as 2μm wide, 6μm high, and has 4μm pitch. The TSV count is calculated as follows. In one bank, there are 128 TSVs for data delivery. In addition, one redundant TSV is inserted to address the TSV reliability issue [18]. Given 14-bit row address and 5-bit control signals, one bank needs 147 TSVs with a total number of 4,704 TSVs employed in each layer, which consumes about 0.075mm$^2$ as one TSV takes 16μm$^2$.

In addition, the wire model in CACTI-3DD is used to calculate the bus delay, where 1mm wire introduces 0.087ns delay and TSV has 0.03ns delay at 45nm technology. Since the RC delay is proportional to the wire length, the longest path, which crosses four layers and goes from the corner to the center, is around 5.35mm and thus introduces 0.58ns delay. Compared to the 2.5ns clock period, the signal propagation delay does not incur extra bus latency (in cycle).

**Control Logic Overhead.** Except the busy table, all control logics in the 3D-SWIFT and MC have been implemented and verified by ModelSim. All designs are further synthesized by Design Compiler with TSMC 45nm technology for the area and power analysis. Table 1 shows the the synthesis results.

As the MC of 3D-SWIFT can be integrated into the processor die, it does not consume any area in 3D-SWIFT. The main area overhead caused by the control logic is from the dedicated row address register. Given 2,048 sub-banks in 3D-SWIFT, the row address register consumes 0.21mm$^2$ area in total (four layers). Similarly, the area overhead of transition generator is 0.12mm$^2$, if each sub-bank also has a dedicated generator. In fact, the number of transition generators can be reduced to the value of tRC (in cycle) by sharing the generators among all sub-banks, as at least one

Table 2: Simulation Platform Configuration

| System | |
|---|---|
| Cores | 4, ALPHA, out-of-order |
| CPU Clock Freq. | 3 GHz |
| LDQ/STQ/ROB Size | 32 / 32 / 128 entries |
| Issue/Commit Width | 8 / 8 |
| L1-D/L1-I Cache | 32kB / 32kB 2-way<br>2-cycle latency |
| D-TLB/I-TLB Size | 64 / 48 entries |
| L2 Cache | Shared, Snooping, 4MB, LRU<br>8-way, 15-cycle latency |
| **Memory** | |
| 2D | JEDEC-DDR3, 2GB, 2 ranks<br>8 banks($\times$8), 64-bit bus,<br>800MHz (1.6GHz DDR)<br>tRAS-tRCD-tRP-tTAW: 28-10-10-24 |
| 3D-WIDE | 128-bit/channel, 4-channel, 400MHz<br>tRAS-tRCD-tRP-tTAW: 20-8-8-20 |
| 3D-SWIFT | 128-bit/channel, 4-channel, 400MHz<br>tRAS-tRCD-tRP-tTAW: 20-8-8-**0** |

Table 3: Benchmark Classification

| | #**Benchmarks(MPKI)** |
|---|---|
| *H* | [1]bzip2(45.89), [2]bwaves(36.62), [3]zeusmp(19.37), [4]gobmk(37.69),<br>[5]sjeng(33.51), [6]lbm(26.44), [7]STREAM(34.43) |
| *M* | [8]milc(5.13), [9]cactusADM(7.85), [10]leslie3d(8.33),<br>[11]libquantum(6,94), [12]wrf(7.33), [13]astar(1.01) |
| *L* | [14]soplex(0.16),[15]gamess(0.12), [16]gromacs(0.14),[17]sphinx3(0.08),<br>[18]gcc(0.12),[19]hmmer(0.08),[20]GemsFDTD(0.001),[21]namd(0.04) |
| Mix*H* | mix1: bzip2, bwaves, zeusmp, lbm<br>mix2: lbm sjeng, gobmk, stream<br>mix3: bzip2, sjeng, zeusmp, stream<br>mix4: zeusmp , bwaves, gobmk, lbm |
| Mix*M* | mix1: milc, cactusADM, leslie3d, wrf<br>mix2: astar, libquantum, wrf, milc<br>mix3: cactusADM, leslie3d, astar, libquantum<br>mix4: wrf, libquantum, leslie3d, astar |
| Mix*L* | mix1: gcc, gamess, gromacs, namd<br>mix2: GemsFDTD, soplex, hmmer, sphinx3<br>mix3: gcc, soplex, gromacs, sphinx3<br>mix4: GemsFDTD, gamess, hmmer, namd |

transition generator is free after a row cycle. However, the shared transition generators induce the internal multiplexing logic and distribution network. Therefore, we insist on the dedicated transition generator. Accounting for the address decoder, the area overhead caused by control logic circuit is 0.35mm$^2$, which is only 1.2% of one layer.

# 5. EXPERIMENT

## 5.1 Evaluation Methodology

In this work, we adopt gem5 [19] as our simulation platform. We modified the DRAMSim2 [20] and successfully integrated it to gem5 as the DRAM model. The SPEC2006 [21] and STREAM [22] benchmarks are employed as multi-programmed benchmarks. Normalized Instructions-Per-Cycle (IPC) is used as the speedup criteria. Table 2 shows the gem5 setup during the simulation. In this work, two memory systems are developed as our reference models. **2D** is the base model that simulates a commodity JEDEC DDR3-1600 SDRAM. **3D-WIDE** is a Wide IO DRAM model that leverages the wider memory bus to deliver the burst data. For the fairness of comparison, even though 3D-WIDE has four independent channels, all channels are used as ranks and connected to a single MC. In addition, the data bus in 3D-WIDE runs at 400MHz with double data rate, which is envisioned as the next generation Wide IO. FR-FCFS scheduling scheme [23] is used in 2D and 3D-WIDE, to maximize the row buffer hit rate, whilst FCFS is deployed in 3D-SWIFT due to the close-page policy. The key timing parameters are shown at the bottom of the table. Note that the tTAW of 3D-SWIFT is 0 to indicate that it eliminates the power constraint.

## 5.2 Performance Analysis

**Single-core Simulation**. Firstly, we characterize the benchmark in the single-core simulation. We run each SPEC2006 CPU benchmark with reference input size for 100 million instructions to warm up the cache and another 100 million instructions for the statistics. According to the miss per kilo instructions (MPKI) of last level cache (LLC), we classify the benchmarks into three categories. The symbols *H*, *M*, and *L* stand for the applications that have high (>10), medium ([1, 10]), and low memory intensity (<1), respectively. Only eight benchmarks are selected as the representatives of L class because the rest have almost the same results. Table 3 lists the classification and the corresponding MPKIs. All selected benchmarks are numbered as shown in the table.

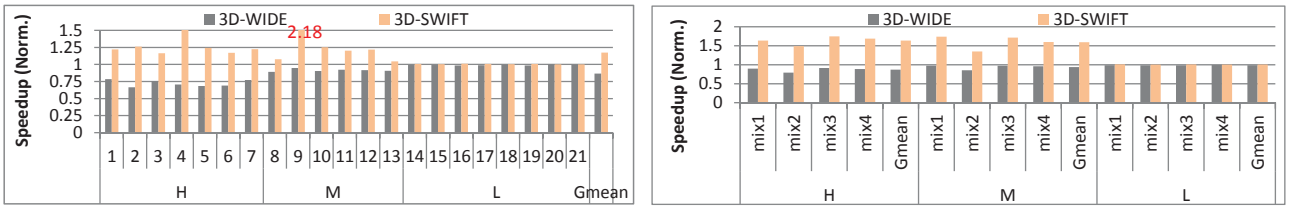
The performance result of signle-core simulation is shown in Figure 7a. 3D-WIDE experience 13.5% performance drop on average. The performance degradation of 3D-WIDE mainly stems from its larger access latency, which adversely affects the memory parallelism. The results from H and M benchmarks indicate that 3D-WIDE has poor performance with memory-intensive applications. In contrast, 3D-SWIFT only has small performance loss in some H and M benchmarks. On average 3D-SWIFT achieves 17.8% and 38.4% performance improvement over 2D and 3D-WIDE, respectively.

In particular, `cactusADM` has 2.18X improvement with 3D-SWIFT. Note that `cactusADM` has absolutely random accesses as the row buffer hit rate is 0. Therefore, 3D-SWIFT can fully take advantage of sub-bank activation to maximize the memory concurrency. This observation indicates that 3D-SWIFT is more promising in a multi-core system in which the intensive memory requests are more likely to be random due to the interference among applications.

**Four-Core Simulation**. We randomly select benchmarks from each category for four-core simulation. The mixed benchmarks are listed at the bottom of Table 3 and Figure 7b presents the results. Similar to the single-core simulation, 3D-SWIFT performs better in H and M benchmarks. In general, 3D-SWIFT achieves 64.7% (58.2%) and 87.6% (67.3%) improvement over 2D, and 3D-WIDE, respectively for H-Mix (M-Mix) benchmarks. The application interference that destroys the data locality and makes the memory request more random is the main reason for the performance improvement, as observed in [24].

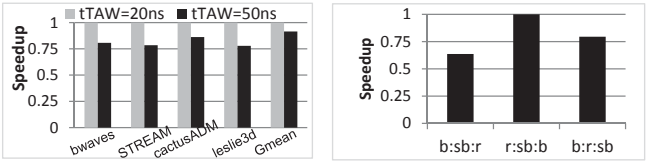
**Impact of Power Constraint (tTAW)**. As mentioned, the tTAW becomes larger (50ns) in [1]. We intentionally apply a 20ns tTAW to 3D-WIDE to see the impactof tTAW. As shown in Figure 8a, tTAW has significant impact on the memory-intensive applications. Particularly, STREAM has 21.5% performance drop compared with larger tTAW. On average, 8.5% performance degradation is observed, which results from the more restricted power constraint on Wide IO. As a result, 3D DRAM design should carefully cope with the power constraint and 3D-SWIFT is one of potential solutions.

**Address Mapping**. As shown in Figure 8b, we evaluate the impact of three address mapping schemes: "bank(b):subbank(sb):row(r)", "b:r:sb", and "r:sb:b" and compare it to the baseline "r:b:sb" which maximizes the sub-bank-level parallelism. First of all, there is no difference to prioritize the sub-bank-level (r:b:sb) or bank-level parallelism (r:sb:b) since both levels can be accessed in parallel. In contrast, both b:r:sb and b:sb:r introduces significant performance drop. The reason is these two schemes suppress the sub-bank-level




(a) Single-core (b) Four-core

Figure 7: Performance results of single-core and 4-core simulation. All results are normalized to 2D DRAM design.




(a) Impact of tTAW (b) Impact of address mapping

Figure 8: Sensitivity Study. (a) Impact of tTAW; (b) Performance with different address mapping schemes. "r"–row, "b"–bank, "sb"–sub-bank.

parallelism. In particular, b:sb:r is the worst case since it maps a large continuous memory space to a single sub-bank, where lots of sub-bank conflicts are fired so that following memory requests must stall and access the sub-bank sequentially. As a result, the overall performance drops by 33.5% and some applications even experiences 75% degradation. Therefore, 3D-SWIFT adopts r:b:sb as the solution.

## 6. CONCLUSION

In this work, we propose a fine-grained 3D-stacked Wide IO DRAM architecture–3D-SWIFT, to exploit high memory parallelism for improving performance and power efficiency. Sub-bank autonomy and packet-based interface protocol are devised to simplify the MC design. The experiment results show that 3D-SWIFT can achieve 64.7% and 87.6% performance improvement than conventional 2D DRAM and 3D Wide IO, respectively. The results indicates the promising future of 3D-SWIFT.

## 7. REFERENCES


[1] JEDEC Solid State Technology Association, "JEDEC Standard: Wide I/O Single Data Rate Specification," http://www.jedec.org/standards-documents/results/jesd229, Dec. 2011.

[2] J. T. Pawlowski, "Hybrid Memory Cube," *HotChip'11*, Aug. 2011.

[3] Tezzaron, "Octopus 8-Port DRAM for Die-Stack Applications," http://www.tezzaron.com/memory/Octopus.html, 2010.

[4] U. Kang, H. Chung, S. Heo, S.-H. Ahn, H. Lee *et al.*, "8Gb 3D DDR3 DRAM Using Through-Silicon-Via Technology," in *ISSCC'09*, Feb. 2009, pp. 130–131.

[5] T. Zhang, Y. Xie *et al.*, "A Customized Design of DRAM Controller for on-chip 3D DRAM Stacking," in *CICC'10*, Sep. 2010, pp. 1–4.

[6] D. H. Kim, K. Athikulwongse, M. Healy, M. Hossain, M. Jung *et al.*, "3D-MAPS: 3D Massively Parallel Processor with Stacked Memory," in *ISSCC'12*, Feb. 2012, pp. 188–190.

[7] M. Ghosh and H.-H. S. Lee, "Smart Refresh: An Enhanced Memory Controller Design for Reducing Energy in Conventional and 3D Die-Stacked DRAMs," in *MICRO'40*, Dec. 2007, pp. 134–145.

[8] P. Jain, D. Jiao, X. Wang, and C. H. Kim, "Measurement, Analysis and Improvement of Supply Noise in 3D ICs," in *VLSI Circuits Digest*, Jun. 2011, pp. 46–47.

[9] M. B. Healy and S. K. Lim, "Power-Supply-Network Design in 3D Integrated Systems," in *ISQED'11*, Mar. 2011, pp. 223–228.

[10] B. Jacob, S. W. NG, and D. T. Wang, *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann, 2007.

[11] Y. Kim, D. Han, O. Mutlu, and M. Harchol-Balter, "ATLAS: A Scalable and High-performance Scheduling Algorithm for Multiple Memory Controllers," in *HPCA'16*, Jan. 2010, pp. 43–54.

[12] S. P. Muralidhara, L. Subramanian, O. Mutlu, M. Kandemir, and T. Moscibroda, "Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning," in *MICRO'44*, Dec. 2011, pp. 374–385.

[13] HPCWire, "NVIDIA Bill Dally Talks 3D Chips and More at GTC."

[14] D. H. Woo, N. H. Seong, and H.-H. S. Lee, "Pragmatic Integration of an SRAM Row Cache in Heterogeneous 3-D DRAM Architecture Using TSV," *TVLSI*, pp. 1–13, Dec. 2011.

[15] T. Shah, "Fabmem: A Multiported RAM and CAM Compiler for Superscalar Design Space Exploration," *Thesis*, 2010.

[16] K. Chen, S. Li, N. Muralimanohar, J. H. Ahn, J. Brockman, and N. Jouppi, "CACTI-3DD: Architecture-level Modeling for 3D Die-stacked DRAM Main Memory," in *DATE'12*, Mar. 2012, pp. 33–38.

[17] International Technology Roadmap for Semiconductors, http://www.itrs.net/, 2009.

[18] A.-C. Hsieh, T. Hwang, M.-T. Chang, M.-H. Tsai, C.-M. Tseng, and H.-C. Li, "TSV Redundancy: Architecture and Design Issues in 3D IC," in *DATE'10*, Mar. 2010, pp. 166–171.

[19] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi *et al.*, "The gem5 Simulator," *Computer Architecture News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[20] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A Cycle Accurate Memory System Simulator," *Computer Architecture Letters*, vol. 10, no. 1, pp. 16–19, Jan.–Jun. 2011.

[21] Standard Performance Evaluation Corporation, "SPEC2006 CPU," http://www.spec.org/cpu2006.

[22] J. D. McCalpin, "STREAM Benchmark," http://www.cs.virginia.edu/stream.

[23] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory Access Scheduling," in *ISCA'27*, Jun. 2000, pp. 128–138.

[24] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi, "Rethinking DRAM Design and Organization for Energy-constrained Multi-cores," in *ISCA'37*, Jun. 2010, pp. 175–186.