OVERCOMING THE MEMORY WALL

FINAL REPORT

By
Jennifer Inouye
Paul Molloy
Matt Wisler

ECE/CS 570
OREGON STATE UNIVERSITY
Winter 2012
Contents

1. Introduction .................................................................................................................. 3

2. Background .................................................................................................................. 5

3. 3D Stacked Memory ................................................................................................. 8

4. Performance Comparison of 2D and basic 3D Structures .................................... 10

5. Architectural Changes Associated with 3D Stacked Memory ............................. 13

6. Thermal Analysis ...................................................................................................... 20

7. Conclusion .................................................................................................................. 25

8. References .................................................................................................................. 26
Abstract
Due to the Memory Wall one of the more promising solutions to this is to stack the memory creating what is called 3D DRAM using TSV’s. Some of the benefits of stacking DRAM is that it will be more compact, reduce wire length, and thus speed up performance. It has been studied that there is at least 2x the speed improvement in comparison to 2D DRAM. This report will go more in depth about the structure of 3D DRAM, the performance and benefit improvements of the different designs and ways to optimize it. In addition, challenges with heat will be discussed and proposed designs to solve this.

Introduction
Recent studies have shown that processor performances are increasing at a rate of about 50%/year while DRAM performance is only increasing at about 10% per year leaving a large gap difference between the two.

![Processor-DRAM Memory Gap (latency)](image)

**Figure 1.1** Processor-DRAM Memory Gap showing growing trend of increasing processor performance over DRAM performance over time.¹

This becomes a problem because as the gap increases, the performance can no longer be increased just by adding more processors. Currently, video encoding is the most memory hungry applications² and the frame rate of a video is 30 frames/sec. Just processing 30 frames alone and segmenting them takes approximately 15 minutes on a four core processor with 24 GB of RAM. Each processor was only running at approximately 50-60% and so the number of processors was not an issue, but there was an issue with the memory access time. And as frame
rates increase along with resolution, this time can extend into hours and even months to process just for a single second in a video. This is not scientifically proven, but just by experience. And over time more devices are headed in the direction of becoming more compact yet still having the same functionality, but performing at a faster rate than that of the larger versions in addition to storing more data.

Figure 1.2 Due to bandwidth and the memory wall increasing the number of processors will not continually increase the performance of computer.³

With knowing where the one of the biggest problem occurs is within the memory, we can study what affects the performance and what changes would need to be made in order to do this. There are many suggestions as to how to modify the DRAM of course there are many drawbacks as well and thus people have been researching 3DRAM in order to solve this.

Recent researchers have proposed to vertically stack the memory on top of one another, creating a 3d integrated circuit. There are many studies done as to how to design this and the challenges and benefits of each, which will be mentioned later in this report. As of recent, Intel has created a Hybrid Memory Cube (HMC) at IDF, which is said to produce up to seven times the energy efficiency of DDR3’s. IBM and Micron have also teamed up together to produce cubes using layers of DRAM connected by vertical conduits called through-silicon vias (TSVs)⁴. The TSV technology creates

Figure 1.3 Intel HMC
micron-sized holes through the chip silicon vertically instead of just horizontally, creating a much denser architecture. The study of 3DRAM extends to applications of GPU’s as well as one study had suggested stacking the GPU’s.

The rest of the report will elaborate on ......

**Background**

We will first take a look at the Dynamic random-access memory (DRAM) in order to first understand what it is before we can explain the new design proposals and how the old structure can improve. DRAM is a type of random-access memory that stores a bit of data using a transistor and capacitor pair, which together make up the memory cell. The capacitor can be either charged or discharged; these two states are represented by the two values of a bit, 0 and 1. The transistor acts as a switch that lets the control circuitry on the chip read the capacitor’s state of charge or change it. Since capacitors leak charge, the information eventually fades unless the capacitor’s charge is refreshed periodically, which is considered volatile, because their state is lost or reset when power is removed from the system. Because of this refresh requirement, it is called *dynamic* memory. Below is a figure of different types of DRAM’s.

![Figure 2.1 From top to bottom: DIP, SIPP, SIMM (30-pin), SIMM (72-pin), DIMM (168-pin), DDR DIMM (184-pin).](image)

Over the years is has been observed that microprocessor speeds are increasing, but not at the same rate that memory access latencies have decreased and so this causes what is called the *Memory Wall*. One source states this increase in microprocessor speed to be roughly 60% per year, while memory access times have improved by less than 10% per year. This has been a concern as devices are expected to be more compact and powerful with all the capabilities such as camera functionality, storing pictures, music, and storing of other data. The amount of data is said to be doubling every 18-24 months. This data must then be transmitted, stored, organized, and processed in real-time. Several modifications to the DRAM have been made to try to reduce this increasing gap including adding multiple levels of caches, and designing processors to prefetch and tolerate latency.
There are three components, which impact the performance and dependence on one another, which are interface, architecture, and controller policies, which will be gone over in further detail later in this report.

DRAM is widely used because it is the most cost effective solid-state storage device, and because of this changes made must be compatible across all applications and not just targeted at computers.

**Main Memory Architecture**

Traditional 3D-stacked implementations have used traditional 2D DRAM organizations, and simply stacked them on top of a single-layer processor. While this does provide a significant benefit in the form of reduced wire delay and power, it does not take full advantage of the many possibilities provided by a 3D organization.

To better describe why a 2D architecture is insufficient and to provide a basis for the architectural changes that will be made in the 3D architectures, the specifics of 2D DRAM architecture will now be discussed.

The general architecture for DRAM is an array of single-transistor bitcells accompanied by logic to access those bits (refer to Error! Reference source not found. (a)).

![Figure 2.2 (a) Organization of the memory hierarchy starting from L2 cache. (b) Details of one memory rank, (c) Details of one memory bank](image)

From left to right in the Error! Reference source not found. (a). above we see there is the L2 cache, which holds copies of data from the most frequently used main memory. A cache reduces the average latency a CPU would take to access memory. Then there is the miss status handling register, which keep track of cache misses. The memory controller (MC), manages data going to and from memory by reading and writing to it as well as controlling the refreshes.

When a miss occurs in the L2 cache, it requires accessing memory to satisfy this request. It must then proceed to the MSHER to note this miss. The request is then forwarded to the memory controller to access the memory. This request must wait in the memory request queue if there are other requests made previous to it. A scheduler such as first in first out (FIFO) exists to determine which request to be sent to memory first. When the request is ready, the MC forwards the physical address to be read or written as well as manages the timing of other signals.

DRAM array are divided into ranks. For each DRAM there are about one or two ranks per module as seen in Figure 2.2(b). Ranks are divided into banks, which consist of 2d arrays of
bitcells. When a read request is made, bits from the physical address are used to select the rank, bank, and row to read the data from.

![Diagram showing the selection of rank, bank, and row from a 32-bit physical address.]

It then takes the data from the row and latches it to the row buffer, which is then sent back to the MC to send back to the processor. The row buffer allows for subsequent accesses to bypass the array-read process. The data must eventually be written back to the bitcell array after any read and contents within the array must be refreshed periodically since these contents are capable of leaking. Based on this generic process of a DRAM we can see that the primary area of concern in terms of speed of memory requests are: 1) the number of clock cycles that elapse between a processor’s request for data and the arrival of the first line of the requested data at the processor input pins (latency); and 2) the rate at which the subsequent data lines are returned after the first line arrives (bandwidth). Figure 2.3 below illustrates this terminology.

![Diagram illustrating latency vs. bandwidth terminology.]

Latency here is defined to be latency of a specific access or the average latency of all accesses in a given workload.
Other factors to consider that affect the performance would be the number of memory accesses that can be pipelined for a specific application, the frequency required for refreshing the dynamic cells, and if a one or more banks are capable of being refreshed while one is being accessed.

Ways to leverage this latency would be to increase the number of ranks, banks, and rows per bank. The more banks and ranks there are, the more you can run in parallel in terms of the number of simultaneous open pages/rows that the memory system can support. Though more ranks require more DRAM modules, and by increasing the number of banks requires more row decoders, sense amplifiers, column muxes, and row buffers, which limits the area a bank can occupy. On the other hand, smaller arrays could lower the latency because of the lower capacitance associated with the word and bit lines. Though, it is stated that by either increasing or decreasing the size of the array increases the amount of die area required per bit. Thus cost and performance increase together when ideally we would want performance to increase while cost decreases. In addition, increasing the row buffer entries per bank would increase the frequency of lower-latency row-buffer hits. One could also increase the width or clock speed of buses, which would increase the bandwidth for transferring the data, but this is limited by the pin counts on both ends of the components as well as area requirements on the motherboard for the printed circuit board (PCB) traces.

The demand for DRAM’s are high due to the low cost of manufacturing it thus many are produced at a time. In order to ensure for successful products, DRAM devices typically include some redundant bits in the architecture in case a single cell or line fails. Thus when arrays are reduced in size, this would increase the area penalty, which would mean that more redundant bits would be required, and therefore the area required for these bits are increased. Device cost is inversely related to die area exponentiated to a constant power. Thus any suggested solution which increases the area increases the cost at a high rate, which might make this product too costly to buy.

One of the reasons performance decreases is due to the processor core waiting during the DRAM latency for critical data in order to proceed with the execution. Again, increasing bandwidth by expanding bus width would increase the cost. One could instead increase the bus speed instead to increase the bandwidth, but this would create high complexity and higher power consumption.

3D stacked Memory

One of the most promising solutions to this is 3d memory, which is basically dram dies stacked on top of one another connected by vertical separations. 3d memory would for devices to continually shrink in size with the same or more functionality. Cost would decrease as large chips would be split up into smaller dies thus improving the yield while decreasing fabrication costs. Circuit layers can be built on different processors or wafers instead of being restricted to a single wafer. Because wires can be sent through the dies instead of wiring everything horizontally, this would reduce the wire length, which would reduce circuit delay. In addition, this would reduce power consumption and thus reduce heat generation, extend battery life and
ultimately lower the cost of operation. Many vertical vias can be sent through these layers thus increasing the width of buses between functional blocks in different layers.

Challenges with 3D memory include the risk for defects during manufacturing, how heat will be dissipated within the stack, ensuring that the design is taking full advantage of what 3D integration has to offer, how testing is to be conducted on separate independent dies while maintaining a reduction in cost and high overall yield, impact of TSV on the area footprint as they occupy both the device and metal layers, which result in placement and routing obstacles. Last of all, if a supplier of one part fails to produce at the same rate as others, this will slow down the production of the whole product.

More details of the benefits and challenges based on heat will be discussed further in this report.

The continuation of Moore’s law by conventional CMOS scaling is becoming more and more challenging, requiring huge capital investments. 3D Packaging with 3D TSV interconnects provides another path towards the “More than Moore,” with relatively smaller capital investments. As current scaling trends require enormous investments only affordable to a select few, Moore’s Law reaches its limit and 3D integration becomes inevitable. This solution is the natural evolution for the IC industry; it is the convergence of performance, power and portability. The economic and technical improvements in performance, power, form factor, time-to-market and cost will drive the use of 3D systems going forward. 3D ICs with TSV are being widely developed around the world for two reasons. First of all, heterogeneous integration of logic, memory, graphics, power and sensor ICs requires it since these different functions cannot be integrated into a single chip. Secondly, 3D ICs with TSVs offer improved electrical performance due to the short interconnect and ultra-high number of connections between layers to address the perceived engineering limits in leakage and electrical performance of CMOS ICs beyond 11-16nm. The 3D TSV ICs have superior advantages over the alternative package choices. By combining the performance and power of system-on-chip (SOC) with the functionality and time-to-market advantages of system-in-package (SiP), TSV offers the best of both for achieving very high densities for memory and logic.

As processors become faster, latency differences between the processor and memory access is increasing even more day by day. This brings in the need for faster memories. Also, many processor architectures now have 3 levels of cache on-chip and the energy consumption for off chip memory access is considerably high. The problem with having memory on the same die as the user logic is that it has to be compatible with the user logic (in the same process technology). This is inefficient since usually user logic is designed in a smaller feature size than the memory. Also, having so many memories increases the die footprint. By going 3D, we alleviate these issues. The logic and memory can be on separate wafers, thereby reducing complexity and number of process steps. They can be individually processed in different feature sizes or even on heterogeneous materials. Chip to wafer bonding even allows stacking together chips of different sizes with the base die being the largest.

As we have multiple layers of memory, larger memories with smaller foot-prints can be implemented, which means there is a huge increase in density. In 3D memories, one layer is generally devoted to the controller and interface components. This means that there is more
Performance Comparison of basic 2D and 3D CPU-memory structure

There are countless aspects to be considered when one is trying to compare the performance between traditional 2D memory and CPU structures against a 3D structure. Many of these aspects are discussed in this survey including their individual potential performance improvements. This section provides a baseline comparison between a traditional 2D architecture and a basic 3D equivalent. [1]

This simulation compares nearly identical 2D and 3D designs with the only differences being the main memory speed and L2 cache external bus width and speed as seen in Table 4.1. The 2D design's main memory is off chip operating at 200Mhz with the L2 cache external bus operating at a fixed 200Mhz with 8 byte width. The 3D processor and memory structure can be seen in Figure 4.1. Here the CPU and L1 cache are on a single layer with the L2 cache stacked above. The main memory resides in the layers above the L2 cache. For this simulation, these structural improvements alone allow for improved bandwidth between the L2 cache and main memory. Here the L2 main memory bus operates at the CPU's core clock frequency and is evaluated for varying widths of 8-128 bytes. Additionally the main memory speed is improved by 2.5x operating a 500Mhz.

Table 4.1 System configurations
In order to fully compare these structural differences 3 different benchmarks were used with varying levels of memory use. The first application *mcf* is highly memory intensive and is expected to see the greatest improvement in performance with the 3D design. Additionally a moderately memory intensive application *parser* is used and a non-memory intensive application *twolf* is also evaluated.

The first simulation results can be seen in Figure 4.2 for the memory intensive *mcf* application. This simulation provides execution time for an increasing L2 cache size. The 3D design is compared with a varying bus width starting at the same 8 byte width of the 2D design progressing up to the same width of a L2 cache line of 128 bytes. It is quickly seen that the bandwidth improvement for the 3D design greatly reduces execution time a memory intensive application. Even with the same 8 byte bus width there is still a 2x improvement for the 3D design. Further improvement can be realized when increasing the bus width, this improvement reaches a limit however as it approaches the length of an L2 cache line (128 bytes).
An additional comparison is made in the inset of Figure 4.2 when looking at the performance improvement for increased L2 cache size. This value corresponds to the value of the negative slope for each comparison in the major portion of Figure 4.2. This shows that as L2 to main memory bandwidth is improved, the execution time is less dependent on L2 cache size. So effectively the L2 cache becomes less and less valuable because there the penalty for accessing main memory becomes increasingly small.

Keep in mind that the drastic improvement shown above is only for the memory intensive *mcf* application. When comparing performance between the two designs other applications must be considered since not all applications have the same level of memory use and thus will not show the same performance improvements. Figure 4.3 shows the improvement if execution time when increasing the CPU clock frequency with a fixed L2 cache size of 1MB and bus width of 8 bytes for both the 2D and 3D designs. As expected the performance improvements with the 3D design are highly dependent on the memory demands for a given program. The most memory intensive application *mcf* sees the greatest improvement with more than 2x speed up. When looking at the moderate memory requirements of *parser* only minimal performance improvements are seen. When evaluating an application that has very little memory use such as *twolf* there is no improvement in execution time.
In conclusion it was found that there are drastic improvements to be made to the performance of memory intensive applications even with the modest 3D design considered for this study. However it cannot be expected that improvements to memory bandwidth will drastically improve performance of all applications. Additionally the comparisons made here did not take into account how the thermal effects of a 3D CPU-memory structure may have on performance. However these thermal effects were considered for additional simulations of this study. These thermal effects must be considered when comparing 2D and 3D structures, and has been included in the thermal effects section of this survey.
Architecture Changes Related to 3D-Stacked Memory

This section goes into detail regarding the specific architecture of a 3D-stacked memory system. It describes the architecture of traditional systems to the extent that is necessary for an uninformed reader to understand the importance of the newer 3D architectures. Then it delves further into the proposed 3D architectures, at differing levels of granularity: from ranks to transistors.

In particular, this section discusses the changes to the main memory architecture, the cache architecture, and processor architecture that a 3D-stacked system brings to light.

3D Memory Hierarchy Architecture Changes

While the current 2D DRAM architecture is sufficient in a 2D system, there are a myriad of improvements that may easily be made when considering 3D-stacked memory architecture. This section is a discussion of these improvements, their advantages and their limitations.

Increase Memory Bus Width

The first and easiest of these is simply increasing the memory bus width. In a 2D system, the memory bus is severely limited by pin count and real estate. In a 3D system though, through-silicon-vias (TSVs) make it possible to easily have thousands if not millions of connections between the processor and main memory. Thus, increasing the memory bus to the maximum usable by the L2 cache, the size of a cache line, is the first logical step. On most systems this is 64 bytes; this does not come close to utilizing the potential bandwidth that is available in a 3D system.

A reasonable thought, therefore, is that the cache line size must be increased. The theoretical limit without software changes is 4KB, a page. In a traditional 2D layout, such large cache lines are impractical due to the number of cycles required to fetch the cache line. 3D-stacked memory removes this barrier by providing a potential bandwidth high enough to fill a 4KB cache line in a single memory cycle. The larger cache line would also reduce miss rate, given a large L2 cache and a moderate amount of spatial locality in the application. However, it turns out that simply increasing the cache line size is not a valid solution. The access time of the L2 cache increases linearly with cache line size, which negates most of the benefits of a large cache line size and precludes its use.

Increase Memory Bus and Controller Frequency

Another necessary optimization for a 3D-stacked system is the increase in clock frequency for the memory bus and the memory controller. In a 2D system, the memory controller doesn’t need to schedule requests any faster than the DRAM can service them. Therefore, when the main memory latency is drastically reduced by moving to a 3D-stacked memory system, the clock frequency of the relevant memory units must be increased to compensate for the change.
This change does not provide great performance increase by itself; it is simply required in order to take advantage of the lower memory access latencies. It was suggested by C. Liu et al in their paper, *Bridging the Processor-Memory Performance Gap with 3D IC Technology*, that because wire latency was reduced by around 60%, the memory controller and bus frequencies could then be doubled to take advantage of this.

This architecture optimization and the previous one, increasing the memory bus width, are both considered to be elemental, and are part of the 3D-base system that will be compared against with other optimizations.

**Layer Separation and Optimization**

Until this point, all of the improvements introduced by 3D-stacked memory architecture are still inherently 2-dimensional. However, it is possible to split functional blocks across multiple layers in a 3D system. For example, a DRAM bank consisting of rows of bitcells and separate peripheral logic (row decoder, sense amps, row buffer, and column select), can be split between two layers, separating the memory from the logic.

A proposed architecture suggests four layers of DRAM memory and a layer of DRAM peripheral logic on top of the processor. Ranks would be split across silicon layers, in order to reduce wire length and capacitance. This is shown in Figure 5.1(b). This is compared to Figure 5.1(a), which shows traditional 2D DRAM ranks stacked on top of a processor. The advantage obtained by separating the memory and logic is that it provides the ability to optimize each layer for a separate purpose using different process technologies. In this architecture, the layer of DRAM memory can be implemented in NMOS technology optimized for density, while the logic layer can be implemented in CMOS technology optimized for speed.

Optimizing particular silicon layers for a specific purpose can be very effective: splitting ranks across the layers and using different optimization processes for particular layers improved memory access time by 32%, as Loh mentioned in his article, *3D-Stacked Memory Architectures for Multi-Core Processors*. 
Figure 5.1: DRAM stacked on top of a processor in a (a) traditional 2D DRAM rank setup, or (b) splitting ranks across layers to isolate peripheral logic on a separate layer.

**Increasing Number of Ranks and Memory Interfaces**

An additional way to take advantage of the extra bandwidth that is available in a 3D-stacked system is to increase the number of ranks. This is a relatively simple architectural change; it involves a reorganization of memory into a greater number of smaller arrays. This is beneficial because each smaller array of memory would have a relatively smaller access time. The only reason it isn’t done in traditional 2D systems is because of the extra pins and connections required, which are plentiful in a 3D system.

Other than increasing ranks, it is also possible to take advantage of the greater bandwidth by increasing the number of memory controllers. This introduces additional logic that needs to fit into the system, but it is possible to reduce the arbitration logic of each individual controller (reduce the number of scheduler entries proportionally) so that increasing the number of controllers does not provide a significant downside.

The main benefit of adding memory interfaces is to provide a significant level of memory parallelism. While this may be overkill for some applications if they are not able to exploit the parallelism, four memory controllers each with a 64 byte memory bus can provide a huge amount of memory bandwidth, greatly improving performance for memory-intensive applications.

While it would be possible to connect each memory controller to each rank of DRAM memory, and to each L2 cache bank, this creates some avoidable connection overhead. In particular, a prior study [citation] suggested that by altering the L2 bank granularity to match the granularity of a DRAM rank, each memory controller can be assigned to a few L2 cache banks and the associated DRAM ranks. By doing this, the connections between the memory controllers and the L2 cache and DRAM are greatly simplified, while maintaining a high level of bandwidth utilization. Additionally, decoding logic is greatly
reduced by having each memory controller refer to only a fraction of the total main memory. Figure 5.2(b) shows this hierarchy. This study was conducted to measure the performance with relation to the number of ranks and memory controllers. The various cases that are measured are found in Figure 5.2(a).