3D LOGIC-MEMORY INTEGRATION FOR HIGH PERFORMANCE EMBEDDED SYSTEM AND RECONFIGURABLE COMPUTING

By

Yangyang Pan

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: ELECTRICAL ENGINEERING

Approved:

____
Tong Zhang, Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

December 2010
(For Graduation December 2010)

© Copyright 2010
by
Yangyang Pan
All Rights Reserved

CONTENTS

LIST OF TABLES . . . iii

LIST OF FIGURES . . . iv

ACKNOWLEDGMENT . . . vi

ABSTRACT . . . vii

1. INTRODUCTION . . . 1
   - 1.1 Overview of 3D Integration Technology . . . 1
   - 1.2 Summary of Motivation and Contributions . . . 2
     - 1.2.1 3D Memory Integration with VLIW . . . 2
     - 1.2.2 3D Enabled DRAM-Based FPGA . . . 3
2. Improving VLIW Processor Performance using Three-Dimensional (3D) DRAM Stacking . . . 7
   - 2.1 3D DRAM Stacking in VLIW DSPs . . . 7
     - 2.1.1 VLIW Processor Basics . . . 7
     - 2.1.2 DRAM Architecture and 3D DRAM Design . . . 9
     - 2.1.3 3D DRAM L2 Cache . . . 11
   - 2.2 Evaluation . . . 14
3. Design of DRAM-Based FPGA in 3D FPGA-Memory Integrated Computing Systems . . . 17
   - 3.1 Background and Prior Work on 3D FPGA . . . 17
     - 3.1.1 FPGA Basics . . . 17
     - 3.1.2 Prior work on 3D FPGA . . . 18
   - 3.2 DRAM-based FPGA Enabled by 3D Memory Stacking . . . 19
     - 3.2.1 On-Chip DRAM Refresh Cost Analysis Methodology . . . 22
     - 3.2.2 Case Study I: DRAM Cells Use Parasitic Capacitance . . . 26
     - 3.2.3 Cost Study II: DRAM Cells Use Embedded Capacitors . . . 29
   - 3.3 Performance Evaluation . . . 30
4. CONCLUSIONS . . . 36

Literature Cited . . . 37

LIST OF TABLES

2.1 Configuration parameters used in Trimaran [35] 13

3.1 Resistance and capacitance parameters at 45nm node. 26

3.2 Cost analysis summary for the case study when using parasitic capacitance within each DRAM cell. 29

3.3 Cost analysis summary for the case study when using embedded capacitor within each DRAM cell. 30

3.4 DRAM cell size vs. retention time when DRAM cells use parasitic capacitance. 34

LIST OF FIGURES

2.1 Example VLIW processor with 4 clusters and each cluster contains 4 ALUs. . . . 7

2.2 Illustration of (a) Conventional system without 3D stacking, (b) straightforward 3D DRAM stacking, and (c) migrating on-chip L2 cache into 3D DRAM domain. . . . 8

2.3 Estimated results of (a) footprint, (b) access latency, and (c) energy consumption for an example conventional 1Gb DRAM using different design approaches at 65nm technology node based on ITRS projection. 11

2.4 (a) Illustrated top view of a 3D DRAM with inter-sub-array 3D partitioning, and (b) illustration of a 3D sub-array set with 4 layers. . . . 12

2.5 Comparison of access latency of 2D SRAM, 2D single-V_{th} DRAM, and 2D multi-V_{th} DRAM at 65nm node. . . . 14

2.6 Comparison of 2MB L2 cache access latency as we increase the number of DRAM dies. . . . 14

2.7 Normalized IPC Improvement of the two 3D VLIW DSP design options over the baseline. . . . 15

3.1 Illustration of island-style FPGA architecture. . . . 17

3.2 Area breakdown of various components within one FPGA Tile. . . . 18

3.3 Illustration of DRAM-based FPGA enabled by 3D memory stacking (for the purpose of clarity, the figure does not show the TSVs for connecting heterogeneous memory blocks with FPGA die). . . . 20

3.4 Illustration of on-chip DRAM cell structure based on (a) parasitic capacitance and (b) embedded capacitor. . . . 21

3.5 Distributed DRAM cells within one FPGA tile. . . . 22

3.6 Distributed RC models for (a) word-lines and (b) bit-lines. . . . 23

3.7 Illustration of the DRAM refresh cost analysis flow diagram. . . . 25

3.8 Three types of devices used in FPGA interconnects. . . . 31

3.9 Critical path delay reduction, (a)(c)(e): DRAM cells use parasitic capacitance; (b)(d)(f): DRAM cells use embedded capacitors. . . . 32

3.10 Simulated critical path delay reduction vs. DRAM cell size over several benchmarks. 34

3.11 Estimated area and dynamic power consumption reduction vs. DRAM cell size. 35

ACKNOWLEDGMENT

I take this opportunity to truly thank my advisor, Professor Tong Zhang, for his continuing guidance, support and encouragement. His enthusiasm and expertise in the area of signal processing, coding algorithms, VLSI architectures and various memory data storage technology help me better understand the subtle facets of this work. His rigorous attitude toward science stimulates me throughout the writing of thesis.

I would also like to thank my labmates, Mr. Guiqiang Dong, Mr. Qi Wu, Mr. Yiran Li, Mr. Wei Xu, Mr. Ningde Xie, Mr. Shu Li, Mr. Peter Sumberac and Mr. Kalyana Sundaram Venkataraman in our group. Their advice and ideas are invaluable and let me walk forward. Working with these colleagues has not only been a good learning experience, but also a great pleasure.

Most important, I must thank my parents for their endless love and support all through my life.

ABSTRACT

Three-dimensional (3D) integration has recently become a topic of great interest because of its many compelling advantages such as increased performance, reduced power, small form factor, flexible heterogeneous integration, and reduced overall costs. We proposed two approaches to improvement the system performance by 3D Memory-Logic integration. The first approach studies the potential of using emerging 3D integration to improve embedded VLIW computing system. We focus on the 3D integration of one VLIW processor die with multiple high-capacity DRAM dies. Our proposed memory architecture employs 3D stacking technology to bond one die containing several processing clusters to multiple DRAM dies for a primary memory. The 3D technology also enables wide low-latency buses between clusters and memory and enable the latency of 3D DRAM L2 cache comparable to 2D SRAM L2 cache. These enable it to replace the 2D SRAM L2 cache with 3D DRAM L2 cache. The die area for 2D SRAM L2 cache can be re-allocated to additional clusters that can improve the performance of the system. From the simulation results, we find 3D stacking DRAM main memory can improve the system performance by 10%∼80% than 2D off-chip DRAM main memory depending on different benchmarks. Also, for a similar logic die area, a four clusters system with 3D DRAM L2 cache and 3D DRAM main memory outperforms a two clusters system with 2D SRAM L2 cache and 3D DRAM main memory by about 10%.

The second approach is the emerging three-dimensional (3D) integration technologies can naturally meet this demand by enabling 3D FPGA-memory integration. As FPGAs enter increasingly diverse and high-end applications, large on-chip storage capacity with high memory bandwidth becomes increasingly indispensable. The 3D stacked memory may hold both embedded memory blocks visible to the users and multiple sets of FPGA configurations. We propose and study a DRAM-based FPGA design strategy enabled by such 3D FPGA-memory integration. In current design practice, FPGAs do not use on-chip DRAM cells for configuration data storage mainly because on-chip DRAM self-refresh involves destructive DRAM read

operations. This problem can be solved if we use the 3D stacked memory as primary FPGA configuration data storage and externally refresh on-chip DRAM cells. In this work, we study such DRAM-based FPGA design and investigate involved design issues, and employ the VPR tool set to demonstrate that DRAM-based FPGAs can noticeably reduce FPGA die area and hence improve speed and dynamic power consumption performance, compared to their SRAM-based counterparts.

1. INTRODUCTION

1.1 Overview of 3D Integration Technology

In general, 3D integration refers to a variety of technologies which provide electrical connectivity between stacked multiple active device planes. Various 3D integration technologies are currently pursued and can be divided into three categories:

1. *3D packaging technology*: It is enabled by wire bonding, flip-chip bonding, and thinned die-to-die bonding [36]. As the most mature 3D integration technology, it is already being used in many commercial products, noticeably in cell phones. Its major limitation is very low inter-die interconnect density (e.g., only few hundreds of inter-die bonding wires) compared to the other emerging 3D integration technologies.

2. *Transistor build-up 3D technology*: It forms transistors layer by layer, on polysilicon films, or on single-crystal silicon films. Although a drastically high vertical interconnect density can be realized, it is not readily compatible to existing fabrication process and is subject to severe process temperature constraints that tend to degrade the circuit electrical performance.

3. *Monolithic, wafer-level, back-end-of-the-line (BEOL) compatible 3D technology*: It is enabled by wafer alignment, bonding, thinning and inter-wafer interconnections [51]. Realized by through TSVs, inter-die interconnects can have very high density, e.g., Patti [53] demonstrated the feasibility of fabricating TSVs with 4μm pitch.

Wafer-level BEOL-compatible 3D integration appears to be the most promising option for high-volume production of highly integrated systems. From an IC design perspective, it can provide several distinct advantages, including *integration of disparate technologies*, *massive inter-die bandwidth* and *wire-length reduction*. Therefore, this work focuses on the use of wafer-level BEOL-compatible 3D integration technology to design 3D integrated DRAM.

1.2 Summary of Motivation and Contributions

1.2.1 3D Memory Integration with VLIW

Three-dimensional (3D) integration has recently become a topic of great interest because of its many compelling advantages such as increased performance, reduced power, small form factor, flexible heterogeneous integration, and reduced overall costs [52]. As one of its most promising real-life applications, 3D integration provides a viable solution to address the well-known *memory wall* problem [55] in computing systems, i.e., by stacking multiple high-capacity DRAM dies with one or more processor dies and enabling massive inter-die interconnect bandwidth, 3D processor-DRAM integrated systems can have drastically reduced memory access latency and increased memory access bandwidth. This has been well recognized by the computer architecture community and many recent work [37, 38, 40, 48–50] have explored and well demonstrated the potential of 3D processor-DRAM integrated computing systems.

Most prior work on 3D processor-DRAM integration focus on general-purpose multi-core high-end computing systems, which nevertheless leaves embedded computing systems largely unaddressed. This work attempts to fill the missing link by investigating the use of 3D DRAM stacking in DSP (digital signal processor), in particular VLIW (very long instruction word) DSPs. VLIW DSPs (e.g., Philips' TriMedia and TI's TMS320C64X) are being widely used in communication and multimedia applications that exhibit abundant instruction-level parallelism (ILP). Due to the data intensive nature of most communication and multimedia applications, VLIW DSPs face an equally critical memory wall problem as those general purpose processors. Extensive prior work have been done to design and optimize on-chip cache hierarchy of VLIW DSPs (e.g., see [33, 41, 42, 54]).

This work studies the potential of applying 3D DRAM stacking to improve VLIW DSP performance. The most straightforward option is to directly stack existing VLIW DSP die with 3D DRAM dies, which can directly reduce the on-chip cache cache miss penalty and hence improve the system performance. Beyond this straightforward design option, we can further leverage the stacked 3D DRAM to improve the VLIW DSP computation parallelism. Most existing VLIW DSPs contain

a certain number of clusters, where each cluster may has its own L1 cache and all the clusters share on-chip L2 cache. On-chip L2 cache typically has a relatively large capacity and hence tends to occupy a large portion of VLIW DSP die. If we could migrate on-chip L2 cache into the stacked 3D DRAM, we may be able to put more clusters on the DSP die and hence improve the computation parallelism without increasing DSP die size. In this regard, stacked 3D DRAM becomes heterogeneous and covers more than one level along the memory hierarchy. As a result, stacked 3D DRAM cannot use homogeneous commodity DRAM dies and we must further customize 3D DRAM circuit and architecture design. This work presents a 3D DRAM design solution that can readily support the evaluation of heterogeneous 3D DRAM. We further use Trimaran [35] simulator to evaluate the above two options of using 3D DRAM stacking in VLIW DSP under various media benchmarks.

1.2.2 3D Enabled DRAM-Based FPGA

Since its advent in the mid 1980s, FPGAs (field-programmable gate array) have played an increasingly important role in numerous real-life computing systems and continue to expand into ever diversified applications. As FPGAs enter more high-end applications such as digital signal processing, data communication, and image/video compression and processing, a large on-chip storage capacity with high memory bandwidth becomes increasingly indispensable. As a result, modern FPGA devices have been steadily incorporating more on-chip SRAM storage capacity and sophisticated heterogeneous memory system architectures. Nevertheless, integration of embedded memories tend to sacrifice FPGA on-chip logic and routing resources. By vertically stacking and connecting logic circuit die(s) and memory die(s) together with through silicon vias (TSVs), emerging three dimensional (3D) integration technologies [1] apparently provide an exciting opportunity to equip FPGA devices with an integrated memory system with substantially higher storage capacity and flexibility, without sacrificing FPGA logic and routing resources. Because 3D integration is inherently subject to thermal, testing, and yield issues, 3D integration of a single high-performance logic die with multiple memory dies appears to be the most viable option, at least in the near future.

This work is interested in FPGA design under a 3D FPGA-memory integration framework, where a signal FPGA die is integrated with one or more memory dies. It is clear that, since FPGA devices are interconnect-rich in nature and TSVs tend to reduce available routing space (to ensure high manufacturability, TSV size is typically in the range of at least a few μm or tens of μm [1]), there cannot be too many TSVs between the FPGA die and memory dies. Apparently, on-chip SRAM blocks such as embedded Block RAMs in Xilinx FPGA devices can be readily migrated to 3D stacked memory without incurring too many TSVs. Meanwhile, most FPGAs today use distributed on-chip SRAM cells to store the data for configuring both logic components and interconnects. If all these SRAM cells for configuration data storage are also moved to 3D stacked memory, it will demand the fabrication of hundreds of thousands TSVs with very small pitch, which can result in significant FPGA device performance degradation and TSV fabrication difficulty as pointed out in [2]. Hence, FPGA configuration data storage should still remain on the FPGA die in 3D FPGA-memory integrated computing systems. Nevertheless, due to the relatively large size of SRAM cells, SRAM-based configuration data storage tends to occupy a large percentage of FPGA silicon die area. Moreover, SRAM is more vulnerable to process variations than logic circuits [3, 4] and hence may not be able to leverage the technology scaling to its full extent.

Under the 3D FPGA-memory integration framework, this work investigates the potential of using on-chip DRAM cells instead of SRAM cells for the storage of FPGA configuration data. Although a DRAM cell tends to occupy much less silicon area than an SRAM cell, DRAM cells are not being used in current design practice mainly because of the destructive nature of DRAM read operations. Because of the inherent charge leakage issue, DRAM cells must be periodically refreshed. If we use on-chip DRAM self-refresh (i.e., read the content out from on-chip DRAM cells and subsequently write them back), destructive DRAM read process will temporarily destroy DRAM cell content and hence interrupt normal FPGA operations. It is very intuitive that, for 3D FPGA-memory integrated computing systems, the on-chip DRAM cells can be externally refreshed by 3D stacked memories, instead of using on-chip DRAM self-refresh. As a result, on-chip DRAM cells for configuration

data storage will never be explicitly read, hence the normal FPGA operations will not be interrupted. The 3D stacked memory provides the primary storage of one or multiple sets of FPGA configurations, and the on-chip DRAM cells simply hold one set of FPGA configuration. The benefits of using 3D stacked memory to store multiple configurations sets and allow dynamic reconfiguration has been recently discussed in [5].

Due to the smaller DRAM cell size, DRAM-based FPGA can reduce on-chip configuration data storage silicon area overhead and hence improve FPGA performance metrics such as speed and power consumption. In spite of these apparent potential benefits, DRAM-based FPGAs under the framework of 3D FPGA-memory integration clearly involve two design issues: (i) The 3D stacked memory should drive a set of word-lines and bit-lines on the FPGA die through TSVs to periodically refresh the on-chip DRAM cells. If too many TSVs are required, it may noticeably reduce FPGA on-chip routing resource. (ii) Periodic on-chip DRAM refresh may incur non-negligible energy consumption overhead. Both the number of TSVs and DRAM refresh energy consumption depend on the retention time of on-chip DRAM cells, which further depends on how DRAM cells are implemented. In this work, we consider two different strategies for implementing these on-chip DRAM cells: (1) Each DRAM cell simply depends on parasitic capacitance to hold the charge. (2) Each DRAM cell contains one explicitly fabricated capacitor enabled by commercially available embedded DRAM technologies [6, 7]. These two DRAM cell implementation options represent a trade-off between fabrication cost and potential performance benefits, e.g., although the first design option obviates the extra cost induced by explicit capacitor fabrication, it tends to result in a larger cell size and short retention time.

In this work, we elaborate on the analysis of cost induced by periodic on-chip DRAM refresh with respect to the number of TSVs and power consumption for both DRAM cell implementation options. To quantitatively evaluate the involved trade-offs and potential performance benefits, we use a 45nm FPGA consisting 80×30 tiles as a test vehicle and employ the widely used FPGA modeling tool set VPR [8]. Through detailed SPICE simulations and VPR modeling, we show

that such a DRAM-based FPGA design strategy can reduce FPGA footprint by up to 30% and FPGA dynamic power consumption by over 13%. The first DRAM cell implementation option (i.e., DRAM cells use parasitic capacitance) incurs a total 0.085W power consumption cost and around 27,000 TSVs for the use of DRAM-based FPGA; while the second DRAM cell implementation option (i.e., DRAM cells use embedded capacitor) only incurs about total 0.02W power consumption cost and around 9,000 TSVs in return for its higher fabrication cost. We further evaluate the critical path delay reduction potential by carrying out 20 MCNC benchmarks [9], which shows an average of 20% reduction can be achieved.

This work is organized as following: Section 3.1 presents the background of FPGA and 3D integration technologies and reviews prior work on exploiting 3D integration in FPGA design. We describe the proposed DRAM-based FPGA design and elaborate on cost analysis with case studies in Section 3.2, and present further performance evaluation and comparison results in Section 3.3.

2. Improving VLIW Processor Performance using Three-Dimensional (3D) DRAM Stacking

2.1 3D DRAM Stacking in VLIW DSPs

2.1.1 VLIW Processor Basics

VLIW architecture are characterized that each instruction specifies several independent operations. This is compared to RISC (reduced instruction set computer) instructions that typically specify one operation and CISC (complex instruction set computer) instructions that typically specify several dependent operations. VLIW processors typically contain several identical processing clusters, where each cluster consists a collection of execution units, such as integer ALUs, floating-point ALUs, load/store units, and branch units, as illustrated in Fig. 2.1. All the clusters can execute several operations in one instruction as the same time.

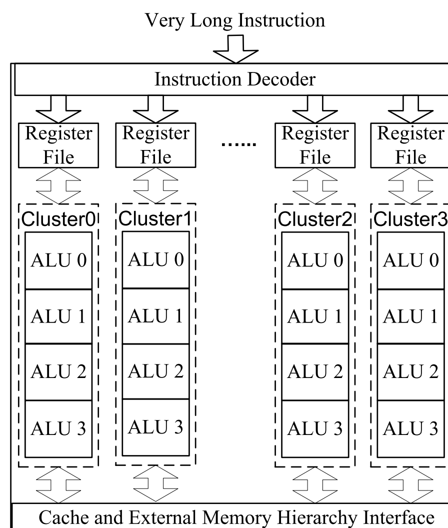


Figure 2.1: Example VLIW processor with 4 clusters and each cluster contains 4 ALUs.

Because VLIW processors can readily exploit the ILP and most signal processing applications have abundant ILP, most existing DSPs inherently use VLIW

[1]This chapter previously appeared as: Yangyang Pan and Tong Zhang,“Improving VLIW Processor Performance Using Three-Dimensional (3D) DRAM Stacking”, 20th IEEE International Conference on Application-specific Systems, Architectures and Processors, pp.38-45, 2009

architectures. VLIW DSPs rely on compiler to explicitly exploit parallelism, which is different from RISC and CISC that rely on hardware to discover parallelism on-the-fly. Compiler schedules the operations based on expected architectural latencies, including the instruction execution time and the memory access time. Because of the data intensive nature of most signal processing applications, VLIW DSPs demand careful design and optimization of cache and memory hierarchy, which has been extensively studied [33, 41–43, 54]. It is very intuitive that VLIW DSP and high-capacity DRAM integration enabled by the emerging 3D integration technologies can further improve the memory hierarchy performance and hence improve the overall system performance.

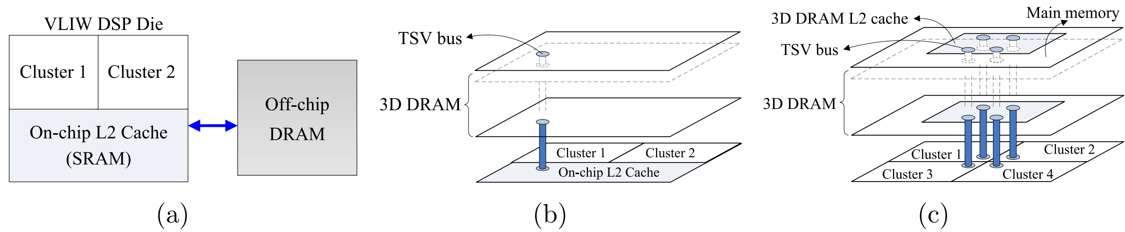


Figure 2.2: Illustration of (a) Conventional system without 3D stacking, (b) straightforward 3D DRAM stacking, and (c) migrating on-chip L2 cache into 3D DRAM domain.

As illustrated in Fig. 2.2(a), a conventional VLIW DSP chip with shared on-chip L2 cache connects with an off-chip DRAM, where each L2 cache miss may result in significant penalty due to the long latency of off-chip data access. As pointed out earlier, we have two options to explore the design of VLIW DSPs with 3D DRAM stacking:

1. The most straightforward design option is to directly stack a VLIW DSP die with 3D DRAM, as illustrated in Fig. 2.2(b). This can be considered as moving the off-chip DRAM into the package. Clearly, such a straightforward design option can directly reduce the on-chip L2 cache miss penalty due to the much less processor-DRAM access latency enabled by the 3D integration. Moreover, since 3D integration can enable a massive inter-die interconnect bandwidth through TSVs, the cache-memory communication bandwidth can be largely increased, which can further improve the overall system performance.

2. Beyond the above straightforward design option, we further study the potential of migrating the shared on-chip L2 cache into the 3D DRAM domain. As illustrated in Fig. 2.2(c), this makes it possible to put more clusters on the VLIW DSP die, which can directly increase the system parallelism and potentially improve the overall computing system performance without increasing the chip footprint. In this context, the 3D DRAM has a heterogeneous structure and covers two levels of the entire memory hierarchy. Because L2 cache demands very short access latency, the 3D DRAM L2 cache should be particularly customized in order to achieve a comparable access latency as its on-chip SRAM counterpart.

To facilitate the evaluation of the above two design options, we first present 3D DRAM design strategies that can exploit the TSVs to improve overall DRAM system performance and enable a heterogeneous cache-memory integration in the 3D DRAM domain.

2.1.2 DRAM Architecture and 3D DRAM Design

In current design practice, high capacity DRAMs have a hierarchical architecture consisting of banks, sub-banks, and sub-arrays. With its own address and data bus, each DRAM bank can be accessed independently from the other banks. The main purpose of using a multi-bank DRAM structure is to enable the memory access pipelining to hide the access latency induced by activation and pre-charge. Each bank is divided into sub-banks, and the data are read/written from/to one sub-bank during each memory access. Each sub-bank is further divided into sub-arrays, and each sub-array contains an indivisible array of DRAM cells surrounded by supporting peripheral circuits such as wordline decoders/drivers, sense amplifiers (SAs), and output drivers etc. Reference [45] gives detailed discussions on modern DRAM circuit and architecture design. DRAM cells, peripheral circuits, and interconnect, including H-tree outside/inside banks and sub-banks and the associated buffers, altogether determine the overall DRAM performance, including silicon area, access latency, and energy consumption. As the technology scales, interconnect tends to play an increasingly important role, particularly for high capacity DRAMs [45].

Most prior work on 3D processor-DRAM integration assume the 3D DRAM is realized by directly stacking several commodity DRAM dies. Intuitively, although this option requires almost no changes on the DRAM circuit and architecture design, it may not be able to exploit the potential benefit of 3D integration to its full extent. The so-called "true" 3D DRAM design approach, which has been assumed in [49], leverages TSVs to completely separate the fabrication of DRAM cell arrays and DRAM peripheral circuits, which ideally can reduce the memory footprint and access latency. However, with this very aggressive design strategy, the pitch of TSVs must be comparable with the DRAM wordline and bitline pitches. Hence, as the DRAM technology scales down, this tends to put an increasingly stringent constraint on TSV pitch, which will result in very significant manufacturability challenges.

With the objective to relax the TSV fabrication constraints, we preset a 3D DRAM architecture design that use a coarse-grained inter-sub-array 3D partitioning, in which only the address and/or data I/O wires of each memory sub-array associate with TSVs and the entire memory sub-array including cell array and peripheral circuits remain exactly the same as in 2D design. Fig. 2.4(a) illustrates the top view of a 3D memory with inter-sub-array 3D partitioning, which looks exactly the same as its 2D counterpart except that each leaf is a 3D sub-array set. Let n denote the number of memory dies being integrated. Hence each 3D sub-array set contains n individual 2D sub-arrays, and each 2D sub-array contains the memory cells and its own complete peripheral circuits such as decoders, drivers, and SAs. Within each 3D sub-array set, all the 2D sub-arrays only share address and data input/output through TSVs, which clearly requires a much lower number of TSVs than the intra-sub-array 3D partitioning. The global address/data routing outside and inside each band is simply distributed across all the n layers through TSVs, as illustrated in Fig. 2.4(b). Let N_{data} and N_{add} denote the data access and address input bandwidth of each 3D sub-array set, and recall that n denotes the number of memory layers. Without loss of generality, we assume N_{data} and N_{add} are divisible by n. As illustrated in Fig. 2.4(b), the N_{data}-bit address bus and N_{add}-bit data bus are uniformly distributed across all the n layers and is shared by all the n 2D sub-arrays within the same 3D sub-array set through a TSVs bundle.

Compared with direct commodity DRAM die stacking, the major difference is that the DRAM global address/data routing is shared among DRAM dies to reduce the routing area overhead in each DRAM die. Since global routing tends to play an importance role in determining the overall DRAM performance, this approach can achieve considerable performance gain. Meanwhile, compared with the ”true” 3D design strategy, this approach has a much relaxed constraint on TSV pitch and requires much less number of TSVs. For the purpose of evaluation, we modified CACTI 5, the latest version of the widely used memory modeling tool CACTI [28], to support the proposed inter-sub-array 3D partitioning strategy. Using a 1Gb DRAM as an example, we estimate the area, latency and energy of such 3D DRAM at the 65nm technology node, as shown in Fig. 2.3. For the purpose of comparison, we further consider conventional 2D DRAM and 3D die packaging of separate DRAM dies without the use of TSVs (referred to as 3D die packaging). The results clearly demonstrate the noticeable performance gain of such inter-sub-array 3D partitioning for 3D DRAM architecture design.

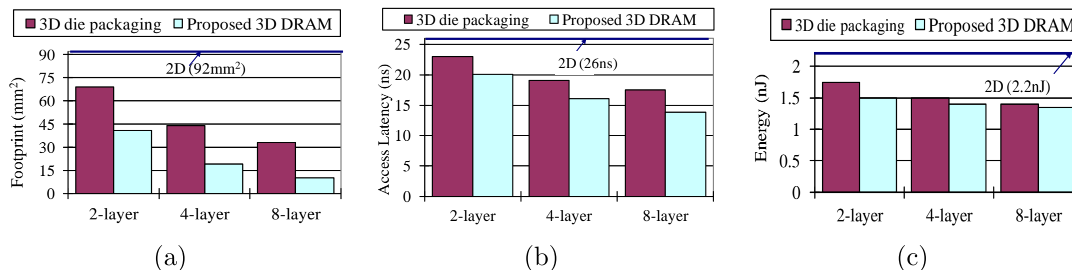


Figure 2.3: Estimated results of (a) footprint, (b) access latency, and (c) energy consumption for an example conventional 1Gb DRAM using different design approaches at 65nm technology node based on ITRS projection.

2.1.3 3D DRAM L2 Cache

The 3D VLIW architecture configuration as shown in Fig. 2.2(c) migrates the on-chip L2 cache into the 3D DRAM domain. Since L2 cache access latency plays a critical role in determining the overall computing system performance, one may intuitively argue that, compared with on-chip SRAM L2 cache, 3D DRAM L2 cache may suffer from much longer access latency and hence result in significant

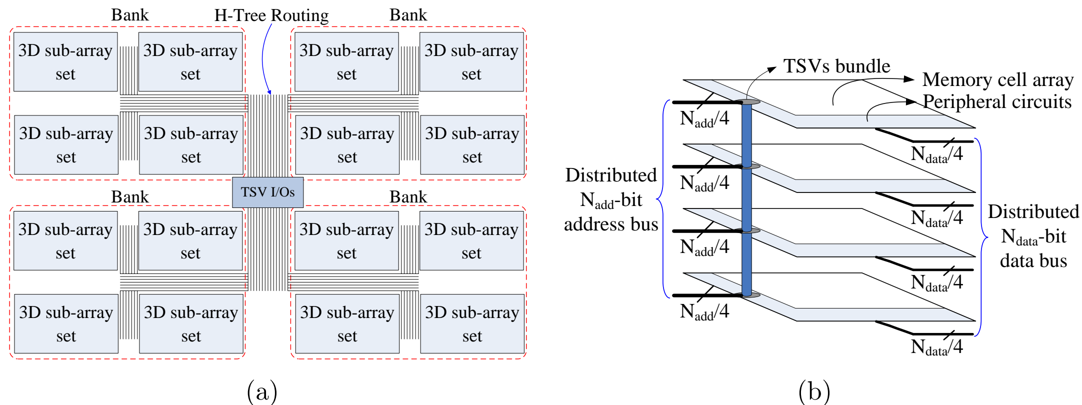


Figure 2.4: (a) Illustrated top view of a 3D DRAM with inter-sub-array 3D partitioning, and (b) illustration of a 3D sub-array set with 4 layers.

performance degradation. In this section, we show that this intuitive argument may not necessarily hold true. In particular, as we increase the L2 cache capacity and the number of DRAM dies, the 3D DRAM L2 cache may have a comparable latency to SRAM L2 cache.

Commercial DRAM is typically much slower than SRAM mainly because, being commodity, DRAM has been always optimized for density and cost other than speed. The speed of DRAM can be greatly improved by using two approaches, including: (i) We can reduce the size of each individual DRAM sub-array to reduce the memory access latency at the penalty of storage density. With shorter lengths of wordlines and bitlines, a smaller DRAM sub-array can directly lead to reduced access latency because of the reduced load of the peripheral decoders and bitlines. (ii) We can adopt the multiple threshold voltage (multi-V_{th}) technique that has been widely used in logic circuit design [44], i.e., we still use high-V_{th} transistors in DRAM cells to maintain a very low cell leakage current, while use low-V_{th} transistors in peripheral circuits and H-tree buffers to reduce the latency. Such multi-V_{th} design is not typically used in commodity DRAM since it will increase leakage power consumption and, more importantly, complicate the DRAM fabrication process and hence incur higher cost. Moreover, as we increase the L2 cache capacity, the global routing will play a bigger role in determining the overall L2 cache access latency. By using the above presented 3D DRAM design strategy, we can directly reduce the latency incurred by global routing, which will further contribute to reducing 3D

DRAM L2 cache access latency compared with 2D SRAM L2 cache.

Table 2.1: Configuration parameters used in Trimaran [35]

Frequency	200 MHz
Number of DRAM dies	6
Die size	58.5 mm^2
Parameter of one cluster	ALU number: 4; register file: 2KB multi-ported; area size: 2.77mm^2; technology: 90nm
2D SRAM instruction cache for each cluster	4KB, 2 way, 16 byte blocks; access latency:1.26ns
2D SRAM data cache for each cluster	4KB, 2 way, 16 byte blocks; access latency:1.26ns
Shared 2D SRAM L2 cache	256KB, 2 way, 32 byte blocks; access latency: 2.43ns; area: 5.69mm^2; technology: 90nm
Shared 3D DRAM L2 cache	512KB, 2 way, 32 byte blocks; access latency: 4.12ns; area: 0.08mm^2; technology: 65nm
Shared Main memory	1GB, 4KB page size; access latency: 22.5ns; technology: 65nm

To evaluate the above arguments, Fig. 2.5 shows the comparison of access latency of 2D SRAM, single-V_{th} 2D DRAM, and multi-V_{th} 2D DRAM, under different L2 cache capacities, at 65nm node. The results show that, as we increase the capacity of L2 cache, multi-V_{th} DRAM L2 cache already excels its SRAM counterpart in terms of access latency, even without using 3D to further reduce global routing delay. This essentially agrees with the conclusions drawn from a recent work on embedded SOI DRAM design [39] that shows embedded SOI DRAM can achieve shorter access latency than its SRAM counterpart at the capacity as small as 256KB.

Fig.2.6 further shows that, when we use the 3D DRAM design strategy presented above to implement a 2MB L2 cache, the access latency advantage of multi-V_{th} 3D DRAM over 2D SRAM will further improve as we increase the number of DRAM dies. This is mainly because, as we stack more DRAM dies, the footprint and hence latency incurred by global routing will accordingly reduce.

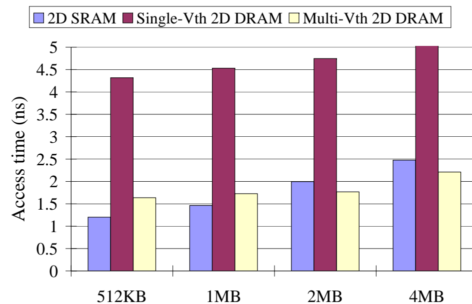


Figure 2.5: Comparison of access latency of 2D SRAM, 2D single-V_{th} DRAM, and 2D multi-V_{th} DRAM at 65nm node.

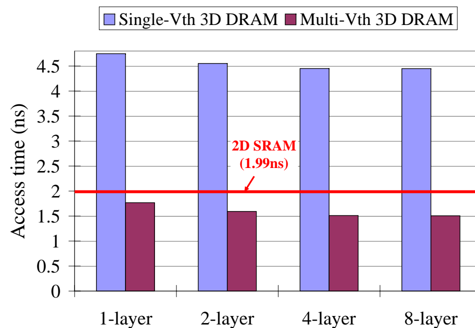


Figure 2.6: Comparison of 2MB L2 cache access latency as we increase the number of DRAM dies.

2.2 Evaluation

To evaluate the above presented two 3D VLIW DSP architecture design options as illustrated in Fig. 2.2(b) and Fig. 2.2(c), we use the Trimaran simulator [35] to carry out simulations. Trimaran is an integrated compilation and performance monitoring infrastructure and covers HPL-PD architecture [46]. We further integrate the memory subsystems of M5 [34] to strengthen the memory system simulation capability of Trimaran. For the 3D VLIW processor and DRAM integration, we assume the VLIW processor and 3D DRAM are designed using 90nm and 65nm technologies, respectively. For the VLIW processor, each cluster contains 4 ALUs, 2KB multi-ported register file, and private 4KB instruction cache and 4KB data cache. All the clusters share one 256KB L2 cache.

We estimate the cluster silicon area based on [47] that reports a silicon implementation of VLIW DSP at 0.13um technology node similar to the architecture being considered in this work. In [47], one cluster, consisting of 6 ALUs and 16KB

single-ported register file, occupies about 3.2mm^2, and the entire VLIW DSP contains 16 clusters and occupies 155mm^2 (note that besides the 16 clusters, it contains two MIPS CPUs and many other peripheral circuits). Accordingly, we estimate that one cluster with 4 ALUs and 2KB multi-ported register file is about 2.77mm^2 at 90nm technology node, the 256KB L2 cache is 5.69mm^2, and the entire VLIW processor die size is 58.5mm^2. Therefore, when we move the 256KB L2 cache into 3D DRAM domain, we can re-allocate the silicon area to two additional clusters. We have that 58.5mm^2 die size with 6 stacked DRAM dies can achieve a total storage capacity of 1GB with 128-bits output width at 65nm node. Table. 3.1 lists basic configuration parameters used in our simulation. We use the conventional system

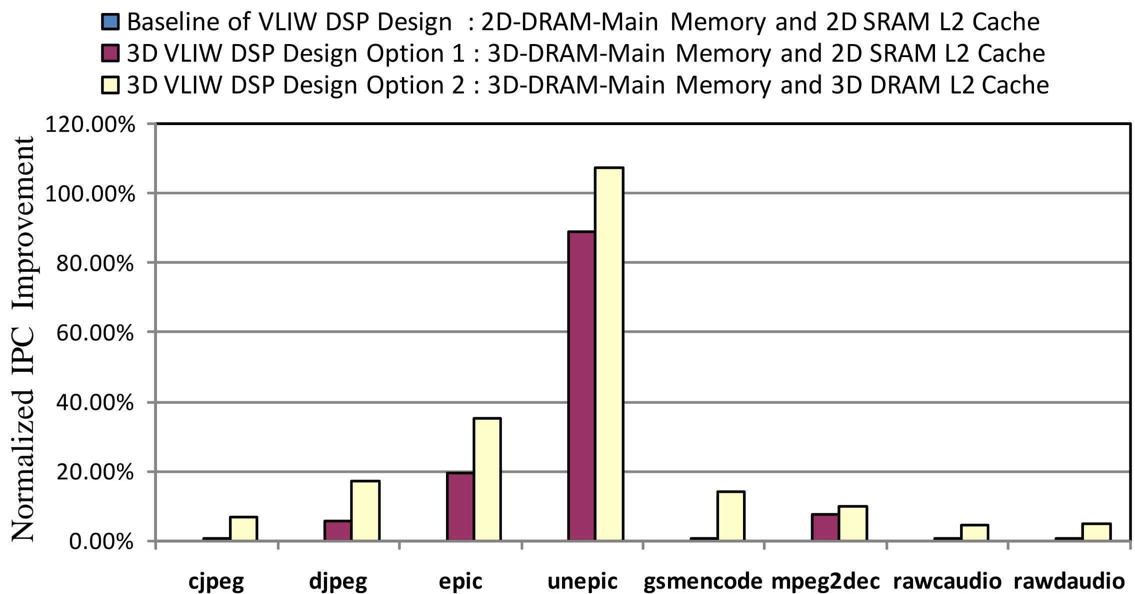


Figure 2.7: Normalized IPC Improvement of the two 3D VLIW DSP design options over the baseline.

architecture with off-chip DRAM as shown in Fig. 2.2(a) as a baseline configuration, where the number of clusters is 2 and the L2 cache size is 256KB. The latency of off-chip main memory access is set to 500 cycles. For the first 3D VLIW DSP design option as shown in Fig. 2.2(b), the 1GB main memory is directly stacked with the processor die. For the second 3D VLIW DSP design option as shown in Fig. 2.2(c), the 256KB L2 cache is migrated into 3D DRAM and we put two more clusters into the processor die without increasing the die area. Using the Trimaran simulator and

a variety of media benchmarks, we evaluate the effectiveness of these two 3D design options in terms of IPC (Instructions Per Cycle). The results clearly demonstrate the potential performance advantages of the 3D VLIW DSP design options. The first design option with direct 3D DRAM stacking can lead to 10%∼80% IPC improvement over the baseline scenario. Compared with direct 3D DRAM stacking, the second design option can further improve up to 10% IPC by migrating L2 cache into 3D domain and increasing computational parallelism.

3. Design of DRAM-Based FPGA in 3D FPGA-Memory Integrated Computing Systems

3.1 Background and Prior Work on 3D FPGA

3.1.1 FPGA Basics

Fig. 3.1 illustrates the island-style FPGA logic fabric architecture, being used in latest Xilinx Virtex FPGAs, which contains a mesh of reconfigurable logic blocks (LBs) connected through reconfigurable switch boxes (SB) and connections boxes (CB). FPGA fabric is conventionally partitioned into an array of tiles, and each tile contains one LB, one SB, and two CB, as shown in Fig. 3.1.

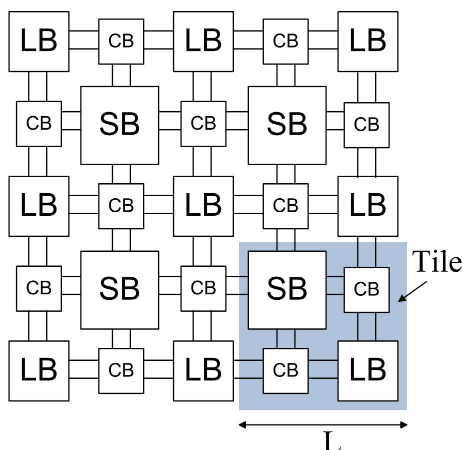


Figure 3.1: Illustration of island-style FPGA architecture.

In Virtex devices, each LB contains four slices, and each slice mainly contains two 4-input look up tables (LUTs) and two flip flops. Since CBs and SBs inevitably induce non-negligible extra routing latency, FPGAs typically implement interconnects with different lengths in order to seek appropriate trade-offs between routing configurability and latency performance, e.g., one FPGA may contain four different types of interconnects that span one, two, four, and six tiles, respectively. The functions of all the reconfigurable logic and routing blocks are controlled by

[2]This chapter previously appeared as: Yangyang Pan and Tong Zhang, "DRAM-Based FPGA Enabled by Three-Dimensional (3D) Memory Stacking", ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), Feb. 2010

on-chip configuration data storage memory cells. In current design practice, on-chip configuration data storage is realized using either SRAM or Flash memory, where SRAM-based FPGAs dominate the market while Flash-based FPGAs are typically geared to specialized markets such as military and aerospace. SRAM-based FPGAs need at least 5 or 6 transistors to store each configuration bit, hence SRAM-based on-chip configuration data storage tends to occupy a significant percentage of the total FPGA die area.

For the purpose of demonstration, we use the well-known VPR tool set [8] with the buffer sharing model to estimate the area breakdown of various components within each FPGA tile. Following the discussions in [10], we estimate that one SRAM cell has a 9.15 minimum-width transistors size. Let interconnects spanning one, two, four, and six tiles are referred to Single, Double, HEX-4, and HEX-6 interconnects, respectively, and we assume the ratio among these four different types of interconnects are 14%:20%:18%:48%. We assume each LB has 16 input and 8 output signals, and set the routing channel width as 72 and the connection box connectivity as 36. Using the VPR tool set, we estimate the area of the LBs and routing resources (RR) including SBs, CBs and buffers, and the results are shown in Fig. 3.2. SRAM cells for configuration data storage occupy about 52% of the title area, which essentially agrees with the results presented in [2,11]. This quantitative study directly motivated us to consider the possibility of using DRAM cells to replace SRAM cells as on-chip configuration data storage.

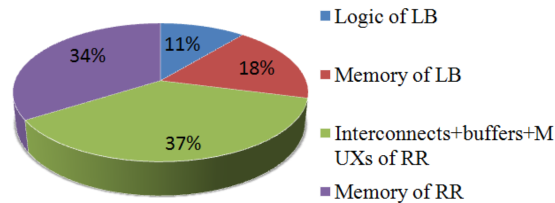


Figure 3.2: Area breakdown of various components within one FPGA Tile.

3.1.2 Prior work on 3D FPGA

How to exploit 3D integration technologies in FPGA design has recently attracted many attentions, where most prior work targeted the 3D stacking of several

FPGA dies and aimed to leverage TSVs to reduce FPGA routing latency and hence improve FPGA performance. An early work is the Rothko 3D FPGA architecture presented and evaluated in [16] based on the wafer-stacking technology presented in [17]. Most recent work on 3D FPGA architecture design assumed the use of wafer-level BEOL-compatible 3D integration technology. Under the framework of 3D integration of several FPGA dies connected with TSVs, placement and routing techniques aware of TSVs have been investigated in [18–21], and new 3D-aware switch box design has been addressed in [22, 23]. The authors of [24] studied the interconnect requirement for such 3D FPGAs. The authors of [25] analyzed the impact of heterogeneous 3D FPGA architectures with different partitioning strategies on the die area. The authors of [26] proposed a 3D nFPGA architecture using 3D and CMOS-nano hybrid techniques.

A monolithically 3D stacked FPGA design has been presented in [2], where the memory cells for configuration data storage are moved to a higher silicon layer. Relying on 3D transistor build-up, such monolithic 3D stacking can provide extremely high die-to-die interconnect via bandwidth to support the 3D FPGA presented in [2]. Nevertheless the technology itself is still in a very early stage of research, compared with the wafer-level 3D stacking being used in this work. The authors of [5] proposed to use 3D stacked DRAM for configuration data caching, which can allow very fast dynamic FPGA reconfiguration.

3.2 DRAM-based FPGA Enabled by 3D Memory Stacking

As pointed out in Section 3.1.1, SRAM-based FPGA configuration data storage tends to incur a large silicon area overhead. Although a DRAM cell can be much smaller than an SRAM cell, DRAM cells are not used to realize on-chip FPGA configuration data storage in current design practice. DRAM cells are inherently subject to charge leakage, hence each DRAM cell must be periodically refreshed to compensate for charge leakage. A straightforward periodic refresh process as in normal DRAM has to first read data out from DRAM cells and then write them back. However, such a straightforward periodic DRAM self-refresh process cannot be used in FPGAs because destructive DRAM read operations will temporarily

destroy DRAM cell content and hence interrupt the normal FPGA operations.

Under the framework of 3D FPGA-memory integration in which embedded memory blocks migrate to the 3D stacked memory, we aim to explore the practical potential of leveraging 3D stacked memory to enable DRAM-based FPGAs, as illustrated in Fig. 3.3. The key is to use a separate 3D stacked memory to externally

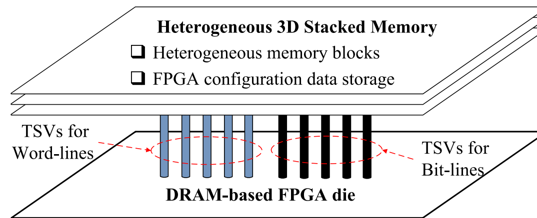


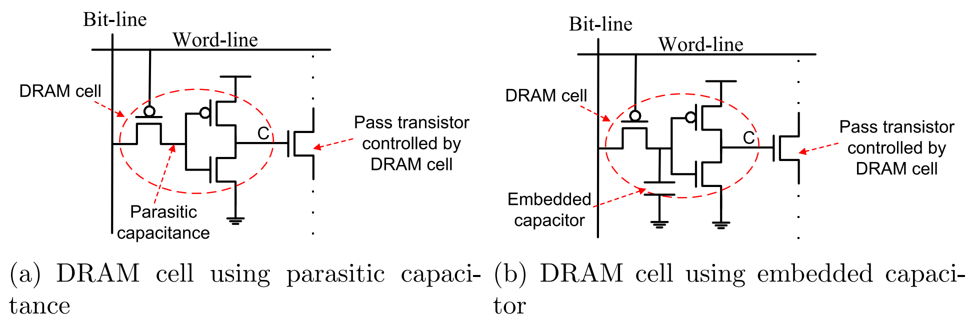
Figure 3.3: Illustration of DRAM-based FPGA enabled by 3D memory stacking (for the purpose of clarity, the figure does not show the TSVs for connecting heterogeneous memory blocks with FPGA die).

refresh the DRAM cells on the DRAM-based FPGA die, instead of using on-chip DRAM self-refresh. Being either SRAM, DRAM, or nonvolatile Flash memory, the 3D stacked memory fabric has a heterogeneous structure, consisting of heterogeneous memory blocks migrated from FPGA die and memory for storing one or more multiple sets of FPGA configurations. The on-chip DRAM cells simply hold one set of FPGA configuration. Distributed over the FPGA die, all the on-chip DRAM cells are refreshed through arrays of word-lines and bit-lines that are driven by the 3D stacked memory through TSVs, as illustrated in Fig. 3.3. Since we no longer need to explicitly read the on-chip DRAM cells, such DRAM external-refresh will not interrupt the normal FPGA operations.

Compared with SRAM-based FPGAs, such a DRAM-based FPGA design strategy enabled by 3D memory stacking can largely reduce the on-chip configuration data storage silicon area overhead, which can be directly translated to higher speed and/or lower dynamic power consumption. Moreover, it is clear that such 3D FPGA-memory integration design approach can support high-speed dynamic FPGA reconfiguration, as discussed in [5]. On the other hand, this proposed DRAM-based FPGA design strategy is subject to certain implementation overheads. The periodic on-chip DRAM cells refresh may incur non-negligible extra energy consump-

tion. Hence, the refresh circuits and the energy cost must be carefully designed and analyzed. Moreover, the area overhead induced by TSVs for driving word-lines/bit-lines should be also carefully taken into account. Clearly, these overheads heavily depend on the retention time of on-chip DRAM cells, which further depends on the implementation of DRAM cells.

In this work, we consider two different DRAM cell implementation strategies as illustrated in Fig. 3.4. The DRAM cell structure shown in Fig. 3.4(a) relies on parasitic capacitance of the inverter and PMOS pass transistor to hold the charge. Although it does not incur any fabrication cost, the inverter and/or PMOS pass transistor should have a relatively large size to ensure a sufficient cell retention time. In contrast, like embedded DRAM, the DRAM cell structure shown in Fig. 3.4(b) uses an explicitly fabricated embedded capacitor. Because the NMOS pass transistor controlled by each DRAM cell may induce a non-negligible amount of gate tunneling leakage, we should use a minimum-sized inverter in between the explicitly fabricated capacitor and the NMOS pass transistor, as shown in Fig. 3.4(b), in order to ensure a sufficient DRAM cell retention time. Although the use of embedded capacitor can lead to a (much) longer retention time and reduce the cell size, it inevitably incurs fabrication cost due to the four to six additional masking steps [27].



(a) DRAM cell using parasitic capacitance

(b) DRAM cell using embedded capacitor

Figure 3.4: Illustration of on-chip DRAM cell structure based on (a) parasitic capacitance and (b) embedded capacitor.

In the remainder of this section, we will discuss how to analyze the cost induced by DRAM refresh in the above DRAM-based FPGA design, and carry out quantitative analysis for both DRAM cell implementation options through representative case studies.

3.2.1 On-Chip DRAM Refresh Cost Analysis Methodology

In the following, we first discuss how to estimate the period of each DRAM refresh operation, based on which we further discuss how to estimate the refresh energy consumption and the total number of TSVs. In this work, we assume the DRAM cells within each FPGA tile are distributed along the two perpendicular middle lines and four sides of the FPGA tile as shown in Fig. 3.5. We model DRAM

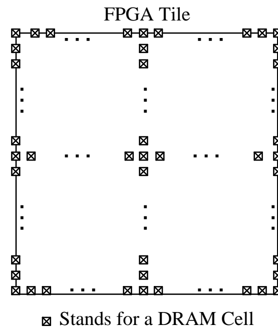


Figure 3.5: Distributed DRAM cells within one FPGA tile.

word-lines and bit-lines as distributed RC networks following the approach used in CACTI DRAM modeling tool [28], as illustrated in Fig. 3.6(a) and (b). In the word-line RC model as shown in Fig. 3.6(a), R_{dr} is the equivalent word-line driver resistance, C_{dr} is the driver diffusion capacitance, C_g is the gate capacitance of the access transistor in each DRAM cell, and R_w and C_w are the wire resistance and capacitance between two adjacent DRAM cells along each word-line. In the bit-line RC model as shown in Fig. 3.6(b), R_{dr} and C_{dr} are the equivalent driver bit-line driver resistance and diffusion capacitance, R_b and C_b are the wire resistance and capacitance between two adjacent DRAM cells along each bit-line, C_d is the diffusion capacitance of each DRAM cell access transistor, and R_{cell} and C_{cell} are the equivalent resistance and capacitance of the selected DRAM cell on each bit-line.

Let N_w and N_b denote the number of DRAM cells on each word-line and bit-line, and τ_{wl} and τ_{bl} denote the worst-case delay of word-lines and bit-lines in the

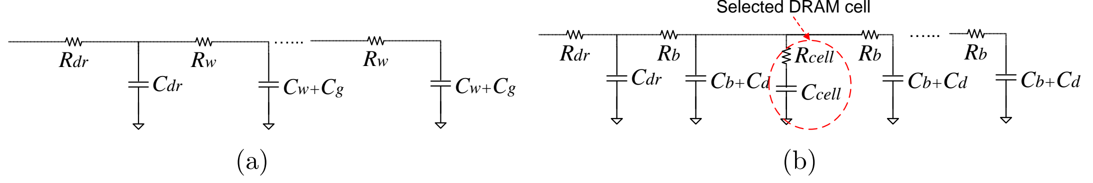


Figure 3.6: Distributed RC models for (a) word-lines and (b) bit-lines.

absence of buffers. Using Elmore delay formula [29], we have

$$\begin{aligned}\tau_{wl} &= R_w(C_w + C_g)N_w(N_w + 1)/2 + N_w R_{dr}(C_w + C_g) \\ &\quad + R_{dr}C_{dr}, \\ \tau_{bl} &= R_b(C_b + C_d)N_b(N_b + 1)/2 + N_b R_{dr}(C_b + C_d) \\ &\quad + R_{dr}C_{dr} + (R_{dr} + N_b R_b + R_{cell})C_{cell}.\end{aligned}$$

Let Δl_w and Δl_b denote the distance between two adjacent DRAM cells along each word-line and bit-line, and R_{metal} and C_{metal} denote the word-line/bit-line metal wire resistance and capacitance per unit length. We have $R_w = R_{metal}\Delta l_w$, $C_w = C_{metal}\Delta l_w$, $R_b = R_{metal}\Delta l_b$, and $C_b = C_{metal}\Delta l_b$.

Apparently, we can use the well-known buffer insertion technique to reduce the word-line and bit-line propagation delay. We note that it is possible to insert buffers along bit-lines since we only refresh (i.e., write to) the on-chip DRAM cells that will not be read at all. Let N_{buff}^w and N_{buff}^b denote the number of buffers inserted along each word-line and bit-line, respectively. Let τ_{dec} denote the extra delay induced by word-line decoders and τ_{buff} denote the delay of each buffer. Let τ_{read} denote the access time to read the data from the 3D stacked memory. We can estimate the total time required for one refresh cycle as

$$\begin{aligned}T_{cycle} &= \tau_{wl}/(N_{buff}^w + 1) + \tau_{bl}/(N_{buff}^b + 1) \\ &\quad + (N_{buff}^w + N_{buff}^b) \cdot \tau_{buff} + \tau_{dec} + \tau_{read}.\end{aligned}$$

Moreover, in order to meet the DRAM cell retention time constraint, we could further partition all the DRAM cells on the FPGA die into a certain number of banks, and all the banks are refreshed in parallel. Let N_{bank} denote the number

of DRAM banks. Recall that each word-line and bit-line connect with N_w and N_b DRAM cells, hence each bank has N_b word-lines and N_w bit-lines. During each cycle, within each bank all the DRAM cells on one word-line are refreshed and all the banks are refreshed simultaneously, hence the time for refreshing all the DRAM cells once can be estimated as $T = N_b \cdot T_{cycle}$. The total number of TSVs equals to the total number of on-chip DRAM word-lines and bit-lines, which can be calculated as $N_{bank}(N_w + N_b)$.

The DRAM cell refresh energy consumption can be estimated as follows. As shown in Fig. 3.4(a), a PMOS transistor is used as the pass transistor in each DRAM cell. We set the turn-on gate voltage of the PMOS pass transistor as a negative voltage V_{neg}. According to the distributed RC model of word-lines and bit-lines as shown in Fig. 3.6, during each refresh cycle, the word-line energy consumption within each DRAM bank can be estimated as

$$E_w = (N_w(C_w + C_g) + C_{dr})(V_{dd} - V_{neg})^2.$$

Regarding to bit-line energy consumption, let P_0 and P_1 denote the probability of each DRAM cell holds a logic 0 and logic 1, we have that the bit-line energy consumption during each refreshing cycle in each DRAM bank can be estimated

$$E_b = P_0 N_w (C_{dr} + N_b(C_b + C_d) + C_{cell}) V_{dd}^2.$$

Notice that we use the probability P_0 in the above calculation due to the use of an inverter within each DRAM cell. Besides the energy consumed on the word-lines and bit-lines, the inverters within DRAM cells may also consume a non-negligible amount of leakage energy, denoted as E_{leak}, because the inverter input voltage may degrade due to the charge leakage at the storage node. Finally, we should also take into account of the energy consumed when we read the configuration data from the 3D stacked memory, which is denoted as E_{read}. Therefore, the overall energy cost can be estimated as

$$E_{cost} = E_w + E_b + E_{leak} + E_{read}.$$

Apparently, the DRAM refresh should be sufficiently fast to meet the DRAM cell retention time constraint, which further depends on cell implementation. Furthermore, as pointed out earlier, in order to meet DRAM cell retention time constraint, on-chip DRAM cells may need to be partitioned into several banks that can be refreshed in parallel, and cells within each bank should be arranged with appropriate word-line and bit-line sizes. Hence, we should carry out quantitative cost estimation following the flow diagram as illustrated in Fig. 3.7.

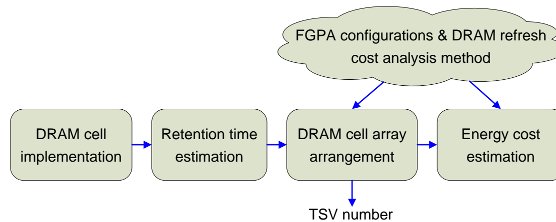


Figure 3.7: Illustration of the DRAM refresh cost analysis flow diagram.

We use a 45nm FPGA consisting of 80×30 tiles as a test vehicle to demonstrate the above discussions. Following the FPGA configuration of the example presented in Section 3.1.1, we keep the ratio of Single, Double, HEX-4 and HEX-6 interconnects as 14%:20%:18%:48%. Each LB has 16 input and 8 output signals, and the routing channel width is 72 and the connection box connectivity is 36. The FPGA works at a supply voltage of 1.0V and the DRAM word-line/bit-line supply voltage is also 1.0V. To reduce the DRAM cell leakage current, we use low-power transistors with a high threshold voltage to implement the DRAM cells. The pass NMOS transistor controlled by each DRAM cell (as shown in Fig. 3.4(a)) is a high-performance transistor with a low threshold voltage in order to minimize FPGA routing latency overhead. In this study, we use the low power and BISM4 45nm PTM transistor models [30], i.e., low-power NMOS and PMOS transistors have threshold voltages of 0.62V and −0.58V, respectively, and high-performance NMOS transistors use metal-gate/high-K and have threshold voltage of 0.34V. Based on the PTM model at 45nm and ITRS [31], we have the resistance, diffusion capacitance and gate capacitance of both low-power and high-performance transistors, and R_{metal} and C_{metal}, as listed in Table. 3.1.

In the following, we present the case studies on estimating DRAM refresh cost

Table 3.1: Resistance and capacitance parameters at 45nm node.

Technology	Low Power 45nm	BSIM4 45nm
R_{on}/unit-width	$237.5K\Omega$	$61.5K\Omega$
C_g/unit-width	$70.4 \times 10^{-18}F$	$53.24 \times 10^{-18}F$
C_d/unit-width	$55.5aF$	$55.7aF$
R_{metal}	$1.12 \times 10^{7}\Omega/m$	
C_{metal}	$1.9 \times 10^{-10}F/m$	

for both cases where either parasitic capacitance or embedded capacitors are used in DRAM cell implementation.

3.2.2 Case Study I: DRAM Cells Use Parasitic Capacitance

In this case, each DRAM cell contains one access PMOS pass transistor and one CMOS inverter. The parasitic capacitance at the internal node acts as the DRAM cell capacitor, as illustrated in Fig. 3.4(a). Clearly, the size of the PMOS pass transistor and inverter within each DRAM cell determines the trade-off between the overall DRAM cell size and DRAM cell retention time: As we increase the size of the PMOS pass transistor and/or inverter, the DRAM cell size will increase and hence reduce the silicon area saving by using DRAM cells to replace SRAM cells for configuration data storage, meanwhile the parasitic capacitance will proportionally increase, leading to a longer DRAM cell retention time and hence demanding less frequent DRAM cell refresh. In this specific case study, we set the two NMOS transistors with minimum size, the PMOS transistor with 1.3× minimum size, and the PMOS pass transistor with 4× minimum size. Hence, according to the area estimation method used in VPR [8], each DRAM cell has 4.65 minimum-width transistors size.

As shown in Fig. 3.7, given the specific DRAM cell implementation, we should first estimate the DRAM cell retention time. As the charge leakage continues to reduce (or increase) the inverter input voltage within each DRAM cell, the inverter will conduct an increasing amount of static leakage current from the power supply to ground. As the inverter input voltage becomes closer to the half of the power supply voltage, the static power consumed by the inverters may become non-negligible and even noticeably increase the overall FPGA power consumption. Hence, the selection

of appropriate DRAM cell retention time, and hence DRAM cell refresh frequency, should carefully take into account of such static power consumption issue. In this specific case study, we define the DRAM cell retention time as the period during which the inverter input drops from 1V to 0.81V or increases from 0V to 0.4V, which makes the inverter output (i.e., the node C that controls the pass transistor as shown in Fig. 3.4(a)) increases from 0V to 0.012V or keeps at almost 1V, respectively. When the inverter input is 0.81V and 0.4V, SPICE simulation shows that the inverter static leakage current is 450pA and 36nA, respectively. By carrying out further SPICE simulations based on the above configurations, we have that the DRAM cell retention time is 100μs in this case study.

According to the flow diagram shown in Fig. 3.7, after we determine the DRAM cell retention time, we should decide the DRAM cell array arrangement, i.e., how many banks and how many word-lines/bit-lines each bank has? Simulations using the VPR tool set show that each FPGA tile contains 864 DRAM cells, and each FPGA tile has the size of 44μm$\times$44μm. As illustrated in Fig. 3.5, within each FPGA tile, all the DRAM cells uniformly distribute along three horizontal and three vertical lines, leading to 144 DRAM cells per line within each FPGA tile. Hence we set each group of 144 DRAM cells belong to the same word-line. As pointed out above, this work assumes that the FPGA die contains an array of 80$\times$30 tiles (i.e., there are total $80 \times 30 \times 864 = 2073600$ (about 256K Byte) DRAM cells on the FPGA die). To meet the DRAM retention time constraint, we partition all the DRAM cells on the FPGA die into 45 banks (i.e., N_{bank} =45). We set each bit-line connects with 90 DRAM cells (i.e., N_b =90), which means there are total 90 word-lines in each bank. Hence, each word-line drives 2073600/(90$\times$45)=512 DRAM cells (i.e., N_w =512), which means there are total 512 bit-lines in each bank. Since each FPGA tile has the size of 44μm$\times$44μm and all the 144 DRAM cells along each line belong to the same word-line, we could estimate that the distance between two adjacent DRAM cells on each word-line is 44/144=0.3μm (i.e., Δl_w=0.3μm), and two adjacent DRAM cells on each bit-line is half of the FPGA tile length, i.e., Δl_b=22μm. We put two buffers on each word-line and bit-line (i.e., $N_{buff}^w = N_{buff}^b = 2$). Moreover, we use CACTI [28] to estimate that the read access time from a 3D stacked 256KB SRAM

memory is 5.58ns (i.e., $\tau_{read} = 5.58ns$) and the read energy consumption is 2.92nJ (i.e., $E_{read} = 0.57nJ$).

Calculations using the analysis method presented in Section 3.2.1 and above parameters show that refreshing each word-line takes T_{cycle} =11ns, leading to a total 44.7μs for refreshing all the DRAM cells once. This can well satisfy the 100μs DRAM cell retention time as estimated above. With the above DRAM cell array arrangement, we need to fabricate 27090 TSVs to drive all the word-lines and bit-lines on the FPGA die, which should be readily accomplished by wafer-level 3D integration technologies. In comparison, for the monolithic 3D FPGA architecture presented [2], the number of TSVs equals to the number of on-chip memory cells (i.e., $80 \times 30 \times 864 = 2073600$ in this case), which is two orders of magnitude larger than what this proposed design approach requires.

According to the flow diagram shown in Fig. 3.7, the last step is to estimate the total energy cost. Assuming P_0 and P_1 are both 0.5, we estimate that a total 2.8μJ (including energy consumed by word-lines/bit-lines and 3D memory read) will be dissipated for refreshing all the DRAM cells once. Hence, assuming a DRAM refresh period of 60μs, the overall DRAM refresh power consumption is 0.047W. Regarding to the leakage energy consumed by the inverters within DRAM cells, as pointed out above, the static leakage current reaches 36nA (denoted as I_{leak1}) and 450pA (denoted as I_{leak0}) if its input voltage increases from 0V to 0.4V or reduces from 1V to 0.81V. From our SPICE simulations, the inverter static leakage current increases approximately linearly as its input voltage changes. Hence, the static power consumption for each inverter is *upper bounded* by

$$P_{inv} = N_{DRAM}(P_0 I_{leak0} + P_1 I_{leak1}))V_{dd}/2,$$

where N_{DRAM} denotes the total number of inverters in the FPGA. Hence, based upon the above parameters, we have that the static power consumed by all the inverters within DRAM cells is upper bounded by 0.038W. This leads to a total 0.085W power consumption cost induced by the use of DRAM-based FPGA. Table 3.2 summarizes the results of the above case study in the case of using parasitic capacitance within each DRAM cell.

Table 3.2: Cost analysis summary for the case study when using parasitic capacitance within each DRAM cell.

Technology	45nm
FPGA tiles	80×30
DRAM cell retention Time	100μs
DRAM refresh power consumption	0.047W (60μs refresh cycle)
Inverters static power consumption	0.038W (upper bound)
Number of TSVs	27090

3.2.3 Cost Study II: DRAM Cells Use Embedded Capacitors

If each DRAM cell contains an explicitly fabricated capacitor as in the commercial embedded DRAM, we set the two NMOS transistors and the PMOS pass transistor with minimum size and the PMOS transistor with 1.3× minimum size. The footprint of the embedded capacitor with 20fF is about 1.4 × minimum size [10]. Hence, according to the area estimation method used in VPR [8], each DRAM cell has 4.1 minimum-width transistors size. According to the flow diagram shown in Fig. 3.7, we should first estimate the DRAM cell retention time. SPICE simulations show that, during the period of 800μs, the inverter input voltage can only either increase from 0V to 850μV or decrease from 1V to 0.995V. Hence, we choose the DRAM cell retention time as 800μs. The inverter leakage current is 8.1nA and 7.71pA when its input is 850μV and 0.995V, respectively.

After we determine the DRAM cell retention time, we should decide the DRAM cell array arrangement. Because of the much longer DRAM cell retention time, we partition all the 256KB DRAM cells on the FPGA die into 9 banks (i.e., N_{bank} =9), set each bit-line connects with 450 DRAM cells (i.e., N_b =450), and each word-line drives 450=512 DRAM cells (i.e., N_w =512). Since each FPGA tile has the size of 44μm×44μm and all the 144 DRAM cells along each line belong to the same word-line, we could estimate that the distance between two adjacent DRAM cells on each word-line is 44/144=0.3μm (i.e., Δl_w =0.3μm), and two adjacent DRAM cells on each bit-line is half of the FPGA tile length, i.e., Δl_b =22μm. We put two buffers on each word-line and bit-line (i.e., $N^w_{buff} = N^b_{buff} = 2$). Calculations using the analysis method presented in Section 3.2.1 and above parameters

show that refreshing each word-line takes T_{cycle} =59ns, leading to a total 240μs for refreshing all the DRAM cells once. This can well satisfy the 800μs DRAM cell retention time constraint. With the above DRAM cell array arrangement, we need to fabricate 8658 TSVs to drive all the word-lines and bit-lines on the FPGA die.

To estimate the energy cost, similar to the above discussions in Section 3.2.2, we assume P_0 and P_1 are both 0.5, and accordingly estimate that 4.4μJ will be dissipated for refreshing all the DRAM cells once. Assuming a refresh period of 400μs, we have that the overall DRAM refresh power consumption is 0.011W. Again, similar to the discussions in Section 3.2.2, we have that the static power consumed by all the inverters within DRAM cells is upper bounded by 0.008W. This leads to a total 0.019W power consumption cost induced by the use of DRAM-based FPGA. Table 3.3 summarizes the results of the above case study in the case of using parasitic capacitance within each DRAM cell. Comparing the data listed in Tables 3.2 and 3.3, it is clear that, in return for the fabrication cost incurred by explicitly fabricating capacitors, we can largely reduce the number of TSVs and energy consumption cost in this case.

Table 3.3: Cost analysis summary for the case study when using embedded capacitor within each DRAM cell.

Technology	45nm
FPGA tiles	80×30
DRAM cell retention Time	800μs
DRAM refresh power consumption	0.011W (400μs refresh cycle)
Inverters static power consumption	0.008W (upper bound)
Number of TSVs	8658

3.3 Performance Evaluation

In this section, we show that the proposed DRAM-based FPGA design scheme can noticeably improve the FPGA performance over its traditional SRAM-based counterpart. We consider both the scenarios discussed in the above, i.e., DRAM cells are implemented based on either parasitic capacitance or embedded capacitors. We keep the same DRAM cell and FPGA configurations as in the above case

studies. Using the VPR tool set, we estimate that the FPGA die area can be reduced by 28.3% and 30.1% when parasitic capacitance and embedded capacitors are being used, respectively. Such FPGA footprint reductions can directly translate to interconnect length reduction. Because interconnects play a very important role in determining FPGA speed and power consumption performance, it is reasonable to expect that DRAM-based FPGAs could achieve noticeable gains in terms of FPGA speed and dynamic power consumption compared with their SRAM-based counterparts.

We use the VPR tool set to estimate the critical path delay of DRAM-based FPGA using the 20 MCNC benchmarks [9]. As specified in VPR, three different types of devices are used along interconnects, including single pass transistor, buffer, and tri-state buffer. In this study, we set the transistor size of these three types of devices as illustrated in Fig. 3.8. We use the BISM4 45nm model and SPICE

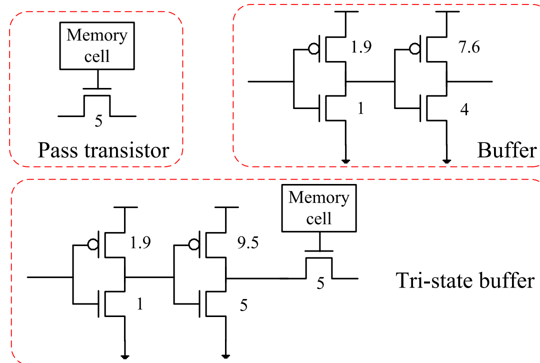
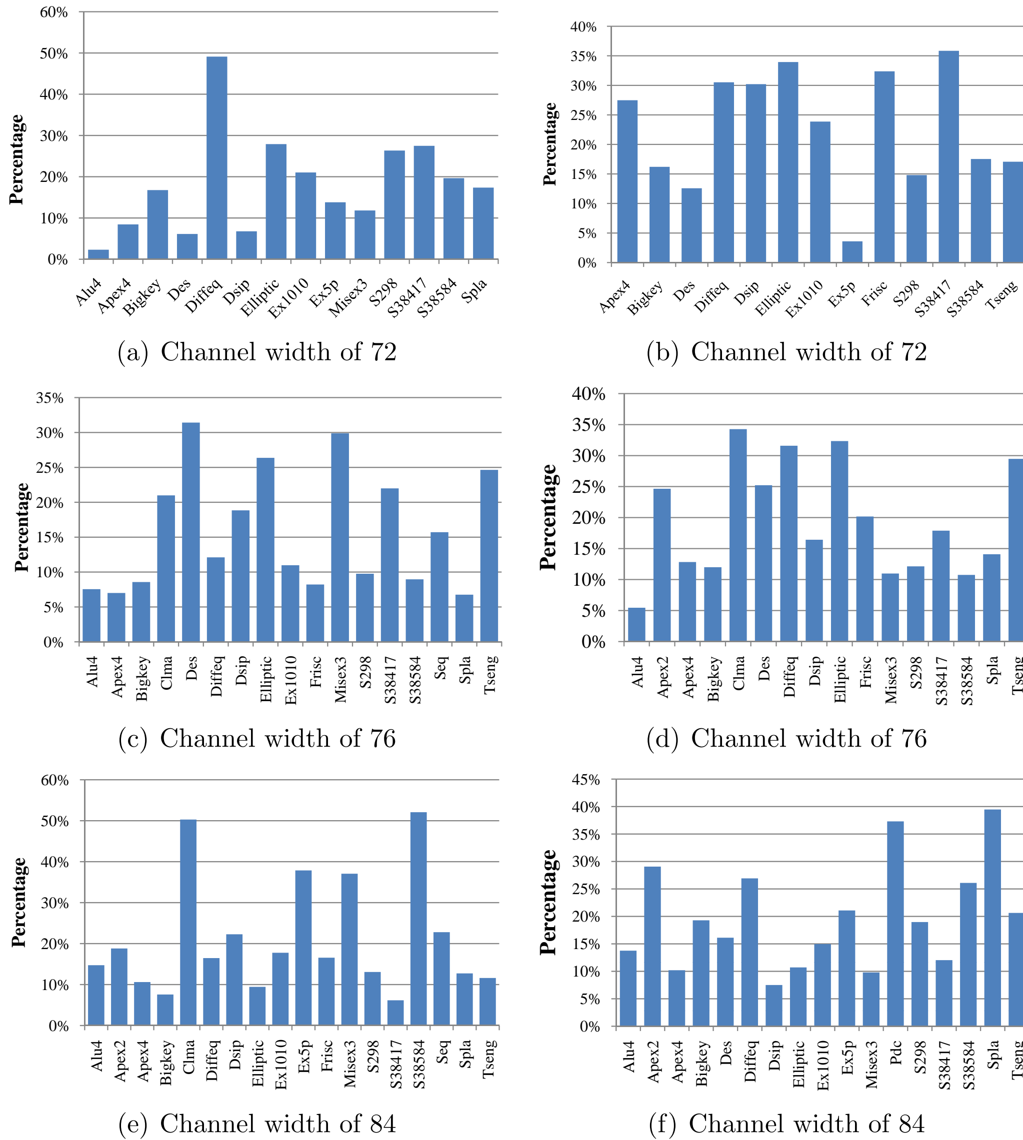


Figure 3.8: Three types of devices used in FPGA interconnects.

simulations to estimate the delay characteristics of all these types of devices. We apply the method used in VPR to derive the delay of the LBs based on BISM4 45nm model. To cover a wide range of FPGA routing channel width, we consider three different routing channel widths, including 72, 76, and 84. Fig. 3.9 show the simulated critical path delay reduction percentage compared to the SRAM-based FPGA. The results clearly show that DRAM-based FPGAs can noticeably improve the speed performance over their SRAM-based counterparts consistently over all the benchmarks.

It is well known that power consumption of CMOS integrated circuits mainly



(a) Channel width of 72 (b) Channel width of 72

(c) Channel width of 76 (d) Channel width of 76

(e) Channel width of 84 (f) Channel width of 84

Figure 3.9: Critical path delay reduction, (a)(c)(e): DRAM cells use parasitic capacitance; (b)(d)(f): DRAM cells use embedded capacitors.

consist of dynamic and static power consumption. Although the percentage of static power consumption in FPGA devices tend to increase as the technology scales down, dynamic power consumption still plays an important role in FPGA devices [32]. In this study, we focus on the dynamic power consumption saving potential when DRAM-based FPGA is being used. FPGA dynamic power consumption consists of three major components, including logic block power consumption P_{LB}, interconnect power consumption P_{int}, and clock network power consumption P_{clk}. Let φ

represent the average interconnect switching activity factor, C_{net} represent the overall interconnect capacitance, C_{clk} represent the overall clock network capacitance. The overall FPGA dynamic power consumption can be expressed as

$$\begin{aligned} P &= P_{LB} + P_{int} + P_{clk} \\ &= P_{LB} + \varphi V_{DD}^2 C_{net} f_{clk} + C_{clk} V_{DD}^2 f_{clk}. \end{aligned}$$

Compared with its SRAM-based counterpart, a DRAM-based FPGA should have almost the same logic block dynamic power consumption P_{LB}. Due to the reduced tile area and hence reduced interconnect length, a DRAM-based FPGA should have less interconnect power consumption and clock network power consumption. Let ξ_{int} and ξ_{clk} denote the power consumption reduction factor of interconnect and clock network of DRAM-based FPGA over its SRAM-based counterpart. Let ϕ_{LB}, ϕ_{int} and ϕ_{clk} denote the percentage of the dynamic power consumption of LBs, interconnect, and clock networks, respectively. Clearly we have $\phi_{LB} + \phi_{int} + \phi_{clk} = 1$, and the overall FPGA dynamic power consumption can be reduced by

$$\xi = 1 - (\phi_{LB} + \xi_{int}\phi_{int} + \xi_{clk}\phi_{clk}).$$

As pointed out above, these two DRAM cell implementation options can reduce the FPGA footprint by 28.3% and 30.1%, respectively. This leads to about 15.4% and 16.4% reduction of interconnect capacitance. Hence, we can estimate that the power consumption reduction factors ξ_{int} and ξ_{clk} as 84.6% when parasitic capacitance is being used and 83.6% when embedded capacitors are being used. Meanwhile, according to [11, 32], the representative values of ϕ_{LB}, ϕ_{int}, and ϕ_{clk} are 15%, 65%, and 20%, respectively. This suggests that we may expect an average of about 13.1% and 13.9% of FPGA dynamic power consumption reduction when parasitic capacitance and embedded capacitors are being used, respectively.

As pointed out earlier, in the case of using parasitic capacitance within each DRAM cell, the size of transistors within DRAM cells directly determines the trade-off between FPGA die area reduction and DRAM refresh cost. In the above case study, we choose the DRAM cell size of 4.65 minimum-width transistor size. In

this work, we further quantitatively investigate how different DRAM cell size may impact this trade-off. In particular, we adjust the transistor size to obtain three other DRAM cell sizes, including 3.65, 4.15, and 5.15 minimum-width transistor size. Accordingly, we estimate their retention based on the same criterion used in the case study I in Section 3.2.2, as shown in Table 3.4.

Table 3.4: DRAM cell size vs. retention time when DRAM cells use parasitic capacitance.

DRAM Cell Size (Minimum-width)	Retention time
3.65	69μs
4.15	86μs
4.65	100μs
5.15	118μs

Moreover, we use VPR tool set to estimate the critical path delay reduction achieved by using different DRAM cell sizes, as shown in Fig. 3.10. The general

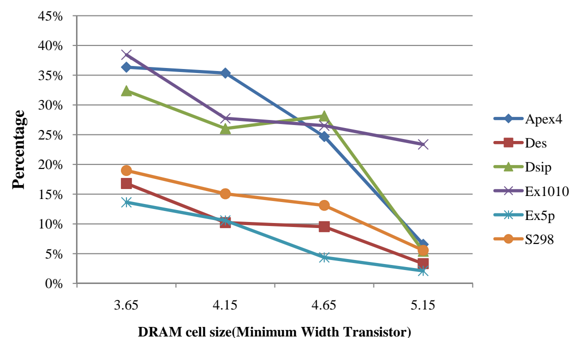


Figure 3.10: Simulated critical path delay reduction vs. DRAM cell size over several benchmarks.

trend shown in Fig. 3.10 can be very intuitively justified: as we increase the DRAM cell size, the FPGA footprint will increase and hence it will result in less potential for critical path delay reduction. Again, following the analysis flow diagram as shown in Fig. 3.7, we derived appropriate DRAM cell array arrangements and accordingly estimate that we need to fabricate 39240, 31140, 27090, and 19620 TSVs when DRAM cell has a 3.65, 4.15, 4.65, and 5.15 minimum-width transistor size, respectively. Subsequently, we estimate the overall energy cost is 0.132W, 0.105W,

0.085W, 0.07W when DRAM cell has a 3.65, 4.15, 4.65, and 5.15 minimum-width transistor size, respectively. Fig. 3.11 further shows the estimated FPGA die area and dynamic power consumption reduction, compared with an SRAM-based counterpart.

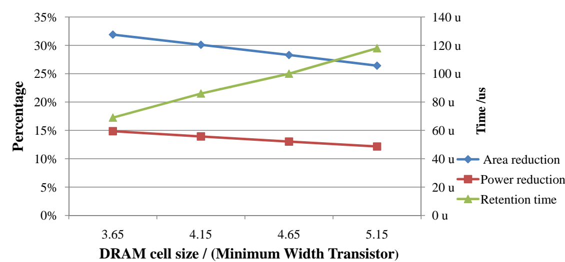


Figure 3.11: Estimated area and dynamic power consumption reduction vs. DRAM cell size.

4. CONCLUSIONS

In this work, we present the potential of 3D stacking technology for implementing VLIW embedded computing systems. Through the proposed method to design 3D DRAM based main memory and L2 cache, we decrease the latency of the 3D DRAM cache and main memory greatly.The ability to tightly couple large amounts of memory to the clusters through wide and low-latency interconnects can greatly reduce system complexity and create opportunities to implement 3D DRAM main memory and L2 cache. The 3D stacking system has significantly improved the IPC by 10%∼80% than conventional 2D system depending on the characteristic of different benchmarks. With the access latency of 3D DRAM L2 cache comparable to the 2D SRAM L2 cache, we found the 2D SRAM L2 cache die area can be replaced by additional clusters. For an area-equivalent VLIW computing system configurations, replacing the traditional on 2D SRAM-based L2 cache with 3D DRAM L2 Cache to re-allocate two more clusters increase the IPC by about 10%. Also, reduced cluster clock frequency enable it easy to simplify the architecture, such as using shorter pipelines with reduced forward logic. For future work, we plan to expand our research space on power consumption of VLIW embedded computing systems. As 3D integration technologies are quickly maturing and entering commercial market, 3D FPGA-memory integration appears to be a viable and economic option for FPGA devices to better serve many high-end memory-hungry applications. This work studies the potential of a DRAM-based FPGA design strategy enabled by this promising 3D FPGA-memory integration paradigm. DRAM cells are not being used in FPGAs for configuration data storage because on-chip DRAM self-refresh inevitably interrupts normal FPGA operations. Under the framework of 3D FPGA-memory integration, the 3D stacked memory may not only hold the embedded memory blocks visible to the users in the run time but also hold one or more FPGA configuration sets. Therefore, 3D stacked memory can externally refresh on-chip DRAM cells to obviate DRAM self-refresh. In this work, we investigate such DRAM-based FPGA design and address the on-chip DRAM cell design and

DRAM refresh cost analysis. Using 45nm FPGA design as a test vehicle and VPR tool set, we show that such DRAM-based FPGAs can largely reduce the FPGA footprint, which can further translate into significant speed and energy efficiency improvement, compared to SRAM-based FPGAs.

Literature Cited

[1] J.-Q. Lu, "3-D Hyperintegration and Packaging Technologies for Micro-Nano Systems," *Proceedings of the IEEE*, vol. 97, pp. 18–30, Jan. 2009.

[2] M. Lin, A.E. Gamal, Y.C. Lu, and S. Wong, "Performance benefits of monolithically stacked 3-D FPGA," *IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems*, vol. 26, pp. 216–229, February 2007.

[3] Y. Nakagome, M. Horiguchi, T. Kawahara, and K. Itoh, "Review and future prospects of low-power RAM circuits," *IBM Journal of Research and Development*, vol. 47, pp. 525–552, Nov. 2003.

[4] R. Heald and P. Wang, "Variability in sub-100nm SRAM designs," in *Proc. of International Conference on Computer-Aided Design*, 2004, pp. 347–352.

[5] A. Cevrero, P. Athanasopoulos, H.P. Afshar, P. Brisk, Y Leblebici, P. Ienne, and M. Skerlj, "3D configuration caching for 2D FPGAs," in *Proc. of the ACM/SIGDA international symposium on Field programmable gate arrays*, Feb 2009, pp. 286–286.

[6] R.E. Matick and S.E. Schuster, "Logic-based eDRAM: Origins and rationale for use," *IBM J. Res. & Dev.*, vol. 49, pp. 145–165, Jan. 2005.

[7] G. Wang, K. Cheng H. Ho, J. Faltermeier, W. Kong, H. Kim, and J. Cai, " A $0.127\mu m^2$ High Performance 65nm SOI Based embedded DRAM for on-Processor Applications," in *Proc. of International Electron Devices Meeting (IEDM)*, Dec. 2006, pp. 1–4.

[8] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Norwell,MA:Kluwer, 1999.

[9] S. Yang, "Logic synthesis and optimization benchmarks user guide version 3.0," 1991.

[10] S.S. Lye, J.E. Barth, P.C. Norum, J.P. Rice, L.R. Logan, and D. Hoyniakr, "Embedded Dram: Technology platform for the Blue Gene/L chip," *IBM Journal of Research and Development*, vol. 25, May 2005.

[11] V. George, *Low energy Field Programmable Gate Array*, Ph.D dissertation, UC Berkeley, CA, 2000.

[12] T.A.C.M. Claasen, "An Industry Perspective on Current and Future State of the Art in System-on-Chip (SoC) Technology," *Proceedings of the IEEE*, vol. 94, pp. 1121–1137, June 2006.

[13] K. Banerjee, S.J. Souri, P. Kapur, and K.C. Saraswat, "3-D ICs: a Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration," *Proceedings of the IEEE*, vol. 89, pp. 602–633, May 2001.

[14] M. Crowley, A. Al-Shamma, D. Bosch, M. Farmwald, and L. Fasoli, "512 Mb PROM with 8 Layers of Antifuse/Diode Cells," in *IEEE Intl. Solid-State Circuit Conf. (ISSCC)*, 2003, p. 284.

[15] S-M Jung, J. Jang, W. Cho, J. Moon, and K. Kwak, "The Revolutionary and Truly 3-Dimensional 25F2 SRAM Technology with the Smallest S3 (Stacked Single-Crystal Si) cell, $0.16\mu m^2$, and SSTFT (Stacked Single-Crystal Thin Film Transistor) for Ultra High Density SRAM," in *Proc. of Symposium on VLSI Technology*, June 2004, pp. 228–229.

[16] M.L. Waleed, W.M. Meleis, M.M. Vai, and P. Zavracky, "Rothko: A three dimensional FPGA," *IEEE Design and Test of Computers*, vol. 15, pp. 16–23, Jan-Mar. 1998.

[17] P. Zavracky, M. Zavracky, D.P. Vu, and B. Dingle, "Three dimensional processor using transderred thin film circuits," *U.S. Patent Application 08-531-177*, 1997.

[18] C. Ababei, H. Mogal, and K. Bazargan, "Three-dimensional place and route for FPGAs," in *ASP-DAC 05: Proc. of the 2005 Conference on Asia South Pacific Design Automation*, 2005, pp. 773–778.

[19] R. Manimegalai, E. Soumya, V. Muralidharan, B. Ravindran, V. Kamakoti, and D. Bhatia, "Placement and routing for 3D-FPGAs using reinforcement learning and support vector machines," in *Proc. of International Conference on VLSI Design*, 2005, pp. 451–456.

[20] K. Siozios, K. Sotiriadis, V. Pavlidis, and D. Soudris, "A software-supported methodology for designing high-performance 3D FPGA architectures," in *Proc. of International Conference on Very Large Scale Integration*, 2007, pp. 54–59.

[21] C. Dong, S. Chilstedt, and D. Chen, "Variation aware routing for three-dimensional FPGAs," in *Proc. of IEEE Computer Society Annual Symposium on VLSI*, 2009, pp. 298–303.

[22] A. Gayasen, V. Narayanan, M. Kandemir, and A. Rahman, "Designing a 3-D FPGA: Switch Box Architecture and Thermal Issues," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, pp. 882–893, July 2008.

[23] G. Wu, M. Shyu, and Y. Chang, "Universal Switch Blocks for Three-Dimensional FPGA Design," in *Proc. of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 1999, pp. 254–259.

[24] A. Rahman, S. Das, A. Chandrakasan, and R. Reif, "Wiring requirement and three-dimensional integration technology for field programmable gate arrays," *IEEE Trans. on VLSI Systems*, vol. 11, pp. 44–54, 2003.

[25] R. Le, S. Reda, and R.I. Bahar, "High-Performance, Cost-Effective Heterogeneous 3D FPGA Architectures," in *Great Lakes Symposium on VLSI*, Boston,MA,USA, May 2009, pp. 254–259.

[26] C. Dong, D. Chen, S. Tanachutiwat, and W. Wang, "Performance and power evaluation of a 3D CMOS/nanomaterial reconfigurable architecture," in *Proc. of the IEEE/ACM international conference on Computer-aided design*, 2007, pp. 758–764.

[27] S. Natarajan, S. Chung, L. Paris, and A. Keshavarzi, "Searching for the dream embedded memory," *IEEE Solid-State Circuits Magazine*, vol. 1, pp. 34–44, Summer 2009.

[28] *CACTI: An integrated cache and memory access time, cycle time, area, leakage, and dynamic power model*, http://www.hpl.hp.com/research/cacti/. Last accessed on Aug 7, 2010.

[29] W.C. Elmore, "The transient analysis of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics*, vol. 19, Jan 1948.

[30] Predictive Technology Model, http://www.eas.asu.edu/˜ptm. Last accessed on May 14, 2009.

[31] Semiconductor Industry Association, *The International Technology Roadmap for Semiconductors (ITRS)*, http://www.itrs.net/reports.html. Last accessed on June 8, 2009.

[32] L. Shang, A.S. Kaviani, and K. Bathala, "Dynamic Power Consumption in Virtex-II FPGA Family," in *ACM Int. Symposium on FPGAs*. 2002, pp. 157–164, ACM Press.

[33] S. G. Abraham and S. A. Mahlke. Automatic and efficient evaluation of memory hierarchies for embedded systems. In *32nd Annual International Symposium on Microarchitecture(MICRO-32)*, pp. 114–125, 1999.

[34] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The m5 simulator: Modeling networked systems. *IEEE Micro*, Vol.26, pp. 52–60, 2006.

[35] L. N. Chakrapani, J. Gyllenhaal, W. W. Hwu, S. A. Mahlke, K. V. Palem, and R. M. Rabbah. Trimaran: An infrastructure for research. In *in Instruction-Level Parallelism. Lecture Notes in Computer Science*, pp.32-41, 2004.

[36] T. Claasen. An Industry Perspective on Current and Future State of the Art in System-on-Chip (SoC) Technology. *Proceedings of the IEEE*, pp. 1121–1137, June 2006.

[37] P. Emma and E. Kursun. Is 3D chip technology the next growth engine for performance improvement? *IBM Journal of Research and Development*, pp. 541–552, Nov. 2008.

[38] B. B. et al. Die Stacking (3D) Microarchitecture. In *Proc. of Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 469–479, 2006.

[39] J. B. et al. A 500 MHz random cycle, 1.5 ns latency, SOI embedded DRAM macro featuring a three-transistor micro sense amplifier. *IEEE Journal of Solid-State Circuits*, pp. 86–95, Jan. 2008.

[40] T. H. K. et al. PicoServer: Using 3D Stacking Technology to Enable a Compact Energy Efficient Chip Multiprocessor. In *Proceedings of the 12th Symp. on Architectural Support for Programming Languages and Operating Systems*, 2006.

[41] J. A. Fisher, P. Faraboschi, and C. Young. *Embedded Computing: A VLIW Approach to Architecture, Compilers and Tools*. Morgan Kaufmann, 2004.

[42] E. Gibert, J. Sanchez, and A. Gonzales. Effective Instruction Scheduling Techniques for an Interleaved Cache Clustered VLIW Processor. In *Proceedings of the 35 th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 123–133, 2002.

[43] E. Gibert, J. Sanchez, and A. Gonzales. Local Scheduling Techniques for Memory Coherence in a Clustered VLIW Processor with a Distributed Data Cache. In *Proceedings of the International Symposium on Code Generation and Optimization*, pp. 193–203, March 2003.

[44] M. Horowitz, D. Stark, and E. Alon. Digital circuit design trends. *IEEE Journal of Solid-State Circuits*, pp. 757–761, April 2008.

[45] K. Itoh. *VLSI Memory Chip Design*. Springer, 2001.

[46] V. Kathail, M. S. Schlansker, and B. R. Rau. Hpl-pd architecture specification: Version 1.1. Technical report, Hewlett-Packard Company, 2000.

[47] K.Khailany, T. Williams, J. Lin, E. P. Long, M. Rygh, D. W.Tovey, and W. J.Dally. A Programmable 512 GOPS Stream Processor for Signal, Image, and Video Processing. *IEEE JOURNAL OF SOLID-STATE CIRCUITS*, 43, January 2008.

[48] C. C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari. Bridging the Processor-Memory Performance Gap with 3D IC Technology. *IEEE Design and Test of Computers*, pp. 556-564, November. 2005.

[49] G. Loh. 3D-stacked memory architecture for multi-core processors. In *Proceedings of the 35th ACM/IEEE Intl. Conf. on Computer Architecture*, June 2008.

[50] G. Loh, Y. Xie, and B. Black. Processor Design in 3D Die-Stacking Technologies. *IEEE Micro*, pp. 31–48, May-June 2007.

[51] J.-Q. Lu, T. Cale, and R. Gutmann. Wafer-level three-dimensional hyper-integration technology using dielectric adhesive wafer bonding. *Materials for Information Technology: Devices, Interconnects and Packaging (Eds. E. Zschech, C. Whelan, T. Mikolajick)*, pp. 386–397, Springer-Verlag (London) Ltd, August 2005.

[52] J.-Q. Lu, K. Rose, and S. Vitkavage. 3D Integration: Why, What, Who, When? *Future Fab International (http://www.future-fab.com/)*, pp. 25–27, July 2007. Last accessed on Nov 26, 2008.

[53] R. Patti. Three-dimensional integrated circuits and the future of system-on-chip designs. *Proceedings of the IEEE*, pp. 1214–1224, June 2006.

[54] Z. Wu and W. Wolf. Design study of shared memory in vliw video signal processors. In *Proceedings of the 1998 International Conference onParallel Architectures and Compilation Techniques*, pp. 52–59, Oct 1998.

[55] W. A. Wulf and S. A. McKee. Hitting the MemoryWall: Implications of the Obvious. *Computer Architecture News*, pp. 20–24, March 1995.