DRAM Row Activation Energy Optimization for Stride Memory Access on FPGA-based Systems*

Ren Chen and Viktor K. Prasanna

Ming Hsieh Department of Electrical Engineering
University of Southern California, Los Angeles, USA, 90089
Email: {renchen, prasanna}@usc.edu

Abstract. FPGA-based systems consisting external memory have been extensively employed in data intensive applications such as signal, image, and network processing. Memory energy is a dominating factor in the overall system energy dissipation. In particular, when performing non-sequential memory access patterns, significant amount of energy is dissipated due to frequent memory row activations. In this paper, we consider the classic stride memory access pattern and evaluate the energy consumption in DRAM. Lower bounds on DRAM energy consumption are derived for this widely used memory access pattern which introduces row-wise writing and column-wise reading memory operations. To achieve the lower bounds of the DRAM energy consumption, we remap data onto DRAM for both row-wise and column-wise memory operations. This significantly reduces the latency brought by frequent DRAM row activations due to column-wise memory operations. We validate experimentally our analysis using 2-D FFT as a benchmark application on FPGA-based system. The experimental results demonstrate that our proposed optimizations result in 74.8%∼77.7% reduction in energy consumption of the overall system compared with the baseline for 1024× 1024, 4096× 4096, and 8192× 8192 points 2-D FFTs, respectively.

1 Introduction

Increasing density and integration of various hardware components, such as low power DSP IP cores and memory blocks, have made FPGAs an attractive option for implementation of various applications [7, 8]. Compared with general purpose processors, the state-of-the-art FPGAs have advantages in much lower power with high performance, especially for data intensive applications. FPGA-based systems usually employ DRAM as the external memory to store large data sets. Such systems have been widely used in various image and digital signal processing applications [6, 7]. In these applications, maximizing performance under energy dissipation constraints is a critical issue.

* This work was supported by the DARPA under grant HR0011-12-2-0023, and the U.S. National Science Foundation under grants CCF-1320211 and ACI-1339756. Equipment grant from Xilinx, Inc. is gratefully acknowledged.

Stride memory access patterns are widely used in data intensive applications such as 2-D FFT, matrix multiplication and convolution. When DRAM is accessed with stride memory addresses, it usually results in a large number of row activations. This increases both the data access latency as well as the power consumption. Most previous optimizations concentrate on reducing the latency due to DRAM row activations so that the DRAM bandwidth can be fully utilized. Mapping data on DRAM using a block or tiling data layout has been investigated by some researchers [5, 10]. However, energy efficiency is not considered in their works. In this paper, we analyze the energy performance of DRAM when accessed with stride memory addresses. We identify the key factors which impact the energy consumption introduced by DRAM row activations. Based on our analysis, we remap data onto DRAM through on-chip stride permutation. As a result, the latency between successive row activations on the same DRAM bank are overlapped. This leads to improved memory bandwidth utilization. Compared with using block data layout, our proposed data remapping approach reduces the DRAM row activation energy with minimal extra latency overhead. Furthermore, as each DRAM row activation introduces a much higher power consumption, we also develop a data prefetch approach to minimize the number of DRAM row activations. The proposed optimizations largely reduce the energy dissipation for accessing DRAM using strided memory addresses. The contributions in this paper are:

1. We analyze the energy performance of DRAM for stride memory access patterns. Lower bounds on the DRAM energy consumption are derived.
2. We propose a data remapping approach to reduce the latency introduced by DRAM row activation, thus improving the memory bandwidth utilization significantly.
3. We demonstrate significant energy efficiency improvement for the benchmark application. Up to 77.7% performance improvement is achieved compared with the baseline.

2 Background and Related Work

FPGA has abundant hardware resources including logic cells, interconnect and on-chip memory. DRAM is usually employed to store large data sets. DRAM bandwidth and capacity have been improving continuously during the last decades. Accordingly, memory energy consumption becomes a major contributor to the overall system power profile. When processing a DRAM access request, one of the DRAM banks needs to be activated so that a row in this bank could be opened to be accessed [2]. This activation process is called as the DRAM row activation. The latency between consecutive row activations to the same DRAM bank is normally 10x the DRAM clock period. This latency could be hidden by pipelining successive memory access requests using multiple DRAM banks. However, this approach is only efficient for sequential memory access pattern, where the memory address is incremented after every memory access. For *stride memory access* pattern, where the stride is the distance of successively accessed

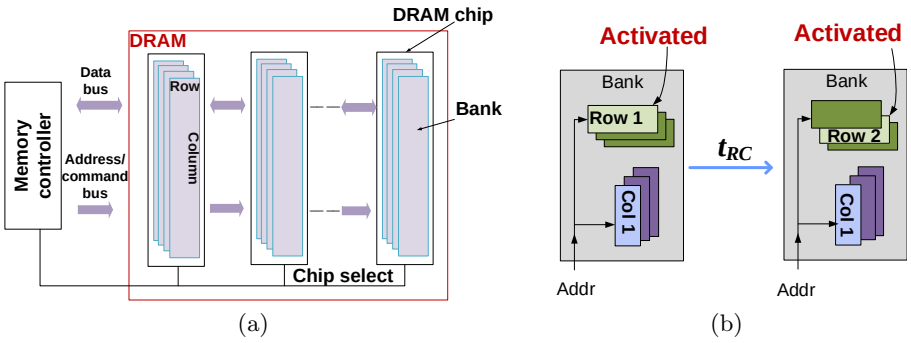

Fig. 1: DRAM access delay: a) Overview of DRAM organization, b) Successive accesses to the same bank and different rows

memory locations, this approach becomes infeasible when the same DRAM bank has to be accessed successively. In this case, a substantial latency is introduced by DRAM row activations.

Previous work are mainly focused on optimizing the latency and throughput rather than the energy dissipation of the external memory when considering stride memory access patterns. To reduce the considerable delay caused by stride memory access in DRAM, a tile data layout was employed to improve the external memory bandwidth utilization for 2-D FFT application in [5] . Their approach maximizes the DRAM bandwidth utilization while at the expense of an extra latency of $O(N^2)$ for $N \times N$ 2-D FFT. There are also other prior research works on effectively utilizing the DRAM bandwidth [11, 13], however, energy was not considered. To the best of our knowledge, all previous works do not consider energy efficiency for stride memory access. Some of them develop DRAM energy optimizations using algorithmic or architectural techniques for general purpose computing. Algorithmic techniques for reducing memory energy have been introduced in [12]. In [9], the authors proposed a conservative row activation scheme that reduces the memory power with negligible performance impact by allowing a memory rank to stay at the low-power mode longer. However, no specific memory access patterns are discussed.

3 Memory Energy Model

3.1 External Memory

We employ a high level model for external memory energy analysis and discuss lower bounds on the memory energy dissipation of algorithms [1]. The power model of the external memory (such as DRAM) is defined as:

Memory Organization An external memory is composed of several ranks, each including several chips. A DRAM chip is organized into banks. Each bank

consists of a two dimensional matrix of locations. At each addressable location ([Bank, Row, Column]), a fixed number of data bits (usually 8 bits) are located. The data word denotes the memory data width. The burst length denotes the minimum granularity at which external memory accesses are performed. The column index is used to locate a data word in one row.

Memory Row Activation A row of a bank needs to be activated before accessed. A read/write operation on an active row is defined as page-hit access. Page hit rate represents the percentage of page-hit access to the external memory. Once the row is activated, multiple accesses on that row can be performed with low latency. When a particular row is active, and a request is for a different row in the same bank, the active row must be closed and the new row must be opened. This incurs a latency penalty due to the time it takes to close the row and activate a new row.

Memory Energy External memory energy is determined by its access latency and memory power. Power components of a memory bank consist of P_{acc} and P_{non_acc}. P_{acc} denotes the power consumed dynamically by memory read or write operations. P_{non_acc} is composed of memory activate power and background power, which are independent on memory access activities. P_{non_acc} is determined by the memory power mode. For example, DRAM has multiple power modes including active, refresh, and power down modes. P_{non_acc} in active, refresh, or power down mode is denoted as P_{act}, P_{ref}, and P_{pdn} respectively.

Besides, several time constraints are involved when accessing DRAM; time constraints used in this work are described here (latency values for the time constraints mentioned here are for 2Gb DDR3 SDRAM [1])

t_{RP}	precharge the long wires before switching to the next row ($\approx$ 15 ns)
t_{RAS}	minimum time between issuing an activation and issuing a precharge ($\approx$ 25 ns)
t_{RC}	minimum time between issuing two successive activate commands in a single bank ($= t_{RP} + t_{RAS} \approx$ 40 ns)
t_{RRD}	minimum time between successive activate commands to different banks ($\approx$ 8 ns)

We use E_{row} to denote the energy consumption of read and write operations to a row within a memory bank. E_{row} can be computed as $E_{row} = (P_{non\_acc} + P_{acc}) \times t_{RC}$. Fig. 1b shows the process when executing successive accesses to different rows in the same bank. A minimum time of t_{RC} is required between issuing two successive activate commands in a single bank. We define effective energy E_{eff_row} as $E_{row} \times r$, where r denotes the data utilization rate within the bank row. r decreases significantly when executing non-consecutive memory access patterns, such as stride memory access pattern where the memory address increases with a stride m after every memory access cycle. If assuming the number of data words in a bank row is w, r can be calculated as $\lceil w/m \rceil$when $m < w$, and r would be constantly $1/w$ when $m \geq w$.

4 Energy Optimizations

4.1 Throughput balanced design

A complete FPGA design usually consists of a computation kernel and an external memory. The external memory is composed of several DRAM chips, each of which has a fixed output data width (usually 16 bits). The number of DRAM chips to be in active state is determined by the memory bandwidth required by the computation kernel on FPGA. For example, if given a computation kernel running at 100 MHz requires a memory bandwidth of 3.2 GB/s, a DRAM chip with 16 data pins running at 800 MHz can be employed to meet the bandwidth requirement [1]. Note that more memory bandwidth can be obtained by using multiple DRAM chips in parallel, however, more memory energy will be consumed correspondingly. At run-time, unused DRAM chips should be set to precharge power-down mode to save energy. In Section 3.1, we derived energy consumption equations for accessing n data words on b memory banks. Similarly, the maximum throughput achieved in DRAM can be obtained as the ratio of n to the total amount time required for accessing the n data words. We can vary the data parallelism p of the on-chip computation kernel with operating frequency of f to balance the throughput between the DRAM and the computation kernel.

4.2 Data remapping and data prefetching

When data are written sequentially and then read using strided addresses, successive DRAM row activations to the same DRAM bank is needed. Strided addresses are a sequence of addresses incremented with a stride. When performing such a DRAM access pattern, a 2-D data array is first created by the computation kernel and then written onto the DRAM with row-wise memory write operations. The computation time and the DRAM access time are usually overlapped through pipelining. After that, the computation kernel needs to fetch the data array with column-wise read operations. Therefore, the memory address is increased with a stride equals to the width of the data array after every memory access cycle. As a result, DRAM row activations may require to be executed frequently, thus introducing a significant delay especially for large size data arrays.

To overlap this latency, a state-of-the-art approach remaps data in blocks by on-chip permutation. Their approach minimizes the number of row activations, however, at the expense of $\mathrm{O}(N^2)$ latency for $N \times N$ 2-D data array. Our DRAM energy model introduced in Section 3 indicates that the energy consumption of column-wise read operations can be reduced to the lower bound. In the state-of-the-art approach, m is reduced to 0 unnecessarily [5]. Instead, we reduce m to be lower than $wb\frac{t_{RRD}}{t_{RC}}$ through on-chip data permutation. As a result, the extra latency is reduced from $\mathrm{O}(N^2)$ to $\mathrm{O}(N)$. After data remapping, both the row-wise write and column-wise read operations are executed using stride memory access patterns. Therefore, for row-wise read operations, the generated 2-D data

array is written with a stride m, which is $\left\lceil wb\frac{t_{RRD}}{t_{RC}} \right\rceil$. After that, the column-wise write operations also need to be executed with the stride m. As a result, the latency of both row-wise and column-wise memory operations are minimized. Using this data remapping scheme, we are able to overlap the latency between successive memory accesses on the same DRAM bank. As a result, the utilization rate per DRAM row activation is improved significantly at the expense of extra memory resource.

5 Experimental Results and Analysis

We use the classic 2-D FFT as the benchmark application for experimental evaluation. The 1-D FFT kernel proposed in [6] is employed in our implementation. Using this kernel, we first build a baseline 2-D FFT architecture based on the row-column 2-D FFT algorithm [7]. Then we optimized the baseline architecture through the proposed data remapping and prefetching schemes. We compare the energy efficiency of the optimized 2-D FFT architecture and the baseline architecture. Both designs achieve a sustained throughput of at least 3.2 GB/s for 1024×1024, 4096×4096 and 8192×8192 points 2-D FFT.

5.1 Experimental Setup

The experiments were conducted on a state-of-the-art FPGA-based MPSoC platform Zynq-7000 with 512 MB DDR3 memory [4]. The FPGA device on Zynq-7000 is Xilinx Artix 7. The experiments were performed using Xilinx Vivado 2014.2 development tools including the Vivado Power Analysis tool used for determining the power dissipation [3]. The designs were verified by post place-and-route simulation. The input matrices for the simulation were randomly generated and had an average switching activity of 50%, which is a pessimistic estimation. All of the reported results are post place-and-route simulation results. We used the VCD (Value Change Dump Format) file as input to the Xilinx Vivado Power Analyzer to produce accurate power dissipation estimation.

5.2 Performance Comparison

We compare performance using two metrics. The first is energy per point, which refers to power dissipated per unit throughput (points per second) in performing 2-D FFT. Another performance metric is energy efficiency (or power efficiency) which is defined as the number of complex operations performed divided by the total energy consumption (GFLOPS/W). In order to give a complete view of the energy efficiency of the entire 2-D FFT architecture, we evaluate the static power dissipation of the architecture and calculate the energy efficiency including the static power dissipation. According to our power simulation results, the static power of the Artix-7 is 0.085W for the optimized and baseline architectures, regardless of how much resources are used and the operating frequency.

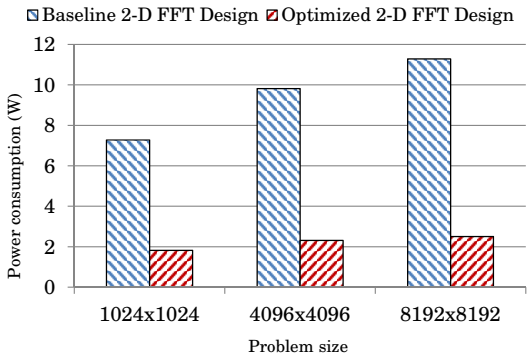

Fig. 2: Power performance comparison

Table 1: Energy performance of the optimized 2-D FFT architecture and the baseline

	Baseline architecture		Optimized architecture		
FFT size	Energy per point (nJoule)	Energy efficiency (GFLOPS/W)	Energy per point (nJoule)	Energy efficiency (GFLOPS/W)	Energy efficiency improvement
1024×1024	18.2	2.09	4.6	8.31	3.9x
4096×4096	24.5	1.89	5.8	8.01	4.2x
8192×8192	28.2	1.79	6.3	8.06	4.5x

By considering the energy consumed by both the DRAM and the 1-D FFT kernel, we compare the performance of the two target 2-D FFT architectures using the two metrics. Fig. 2 shows the power performance of the both architectures. By reducing the DRAM power consumption for stride memory access patterns, the power performance of the optimized 2-D FFT architecture is improved significantly. Table 1 shows the energy performance comparison between the baseline 2-D FFT architecture and the optimized 2-D FFT architecture. It shows that the entire optimized 2-D FFT architecture achieves 8.31, 8.01 and 8.06 GFLOPS/W in energy efficiency for 1024×1024, 4096×4096 and 8192×8192 points 2-D FFT when considering both dynamic power and static power. The energy consumption per point is reduced by 74.8%, 76.4.7%, 77.7% for 1024×1024, 4096×4096 and 8192×8192 points 2-D FFT, respectively. Comparing the results of the energy consumption per point in the 1-D FFT kernel and the entire 2-D FFT architecture, we observe that the optimization for DRAM access makes a major contribution to the improvement of energy performance.

6 Conclusion

In this paper, we evaluated the energy performance of DRAM on FPGA-based system for stride memory access. Lower bounds on DRAM energy performance were derived for these specific memory access patterns. We developed a data remapping technique to rearrange the data layout on the DRAM so that the latency between successive DRAM row activations can be overlapped. The access delay due to DRAM row activation is significantly reduced, thus saving significant amount of energy consumed by the entire FPGA system. To illustrate our proposed optimization, we selected 2-D FFT kernel as the benchmark application. The experimental results show that the optimized design outperforms the baseline in energy efficiency due to much lower energy consumption in DRAM. In the future, we plan to build a design framework targeted at energy efficient signal processing kernels, to enable automatic energy optimizations addressing memory and communication energy.

References

1. DDR3 SDRAM System-Power Calculator. `http://www.micron.com/-/media/Documents/Products/Power%20Calculator/DDR3_Power_Calc.XLSM`
2. Micron DRAM. `http://www.micron.com/products/dram/`
3. Vivado design suite user guide: design flows overview. `http://www.xilinx.com/support/documentation/`
4. Xilinx Zynq board. `http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html`
5. Akin, B., Milder, P., Franchetti, F., Hoe, J.: Memory Bandwidth Efficient Two-Dimensional Fast Fourier Transform Algorithm and Implementation for Large Problem Sizes. In: Proc. of IEEE FCCM '12. pp. 188–191 (April 2012)
6. Chen, R., Le, H., Prasanna, V.K.: Energy efficient parameterized FFT architecture. in Proc. of IEEE International Conference on FPL (2013)
7. Chen, R., Park, N., Prasanna, V.K.: High throughput energy efficient parallel FFT architecture on FPGAs. In: Proc. of IEEE International Conference on HPEC (2013)
8. Chen, R., Prasanna, V.: Energy and memory efficient mapping of bitonic sorting on fpga. In: Proc. of ACM/SIGDA International Symposium on FPGA (2015)
9. Fang, K., Zhu, Z.: Conservative row activation to improve memory power efficiency. In: Proceedings of ACM ICS '13. pp. 81–90. ACM, New York, NY, USA (2013)
10. Park, N., Hong, B., Prasanna, V.: Tiling, block data layout, and memory hierarchy performance. Parallel and Distributed Systems, IEEE Transactions on 14(7), 640–654 (July 2003)
11. Rixner, S., Dally, W.J., Kapasi, U.J., Mattson, P., Owens, J.D.: Memory Access Scheduling. SIGARCH Comput. Archit. News 28(2), 128–138 (May 2000)
12. Singh, M., Prasanna, V.K.: Algorithmic techniques for memory energy reduction. In: Experimental and Efficient Algorithms, pp. 237–252. Springer (2003)
13. Son, Y.H., Seongil, O., Ro, Y., Lee, J.W., Ahn, J.H.: Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations. In: Proc. of ISCA '13. pp. 380–391. ACM, New York, NY, USA (2013)