Reconfigurable Active Drive: An FPGA Accelerated Storage Architecture for Data-Intensive Applications

Teng Li, Miaoqing Huang, Tarek El-Ghazawi, H. Howie Huang

Department of Electrical and Computer Engineering, The George Washington University
801 22nd St, N.W., Washington, D.C., 20052, United States
{tengli,mqhuang,tarek,howie}@gwu.edu

***Abstract*— This paper presents an FPGA-based Solid State Drive (SSD) architecture for data-intensive applications called Reconfigurable Active Drive (RAD). In the RAD architecture, an FPGA utilizes parallel data transfer of flash memory to improve the data throughput for streaming applications. The storage architecture is designed to match the I/O bandwidth of the FPGA and is capable of fully parallelizing data access and data processing. RAD is modelled and evaluated using an SSD simulator with applications implemented on FPGAs. Compared with reconfigurable computing (RC) and pure software (SW) approaches, RAD achieves up to 159× and 541× speedups respectively in terms of the end-to-end throughput.**

I. INTRODUCTION

Over the years Field-Programmable Gate Arrays (FPGAs) have been widely used as application accelerators in a broad range of scientific applications such as cryptanalysis and image processing. By adding FPGAs to general-purpose computers, various Reconfigurable Computers (RCs) have been designed where FPGAs work as co-processors. The programmability of FPGA allows developers to develop customized applications in hardware, which provides better application performance than a software implementation running on traditional processors. However, when a very large amount of data is to be processed, the bandwidth of data transfer from and to the storage system becomes a limiting factor that affects the overall system performance. Therefore, it is necessary to design a new storage architecture that is capable of providing high-throughput data access for FPGAs. Prior research discussed the integration of co-processing units and hard drives. The main idea is to move data processing units close to hard drives. Acharya et al. [1] presented a stream-based programming model to allow "disklet" to execute efficiently and safely, where "disklet" is defined as the disk-resident code of an application. Similarly, Huston et al. [2] introduced a "Diamond" architecture to use "searchlet" code inside the storage devices to perform efficient filtering of large data collections. Smullen et al. [3] discussed trade- offs to offload computation along the I/O path to various processing components such as disk array controller and disk processing unit. Franklin et al. [4] proposed to move Chip Multiprocessors (CMPs) and FPGAs to hard drives. In this architecture, the FPGAs have been put between the disk controller and I/O bus while the CMP and FPGA compose one computing node. Furthermore, Netezza [5] has designed a server that uses an FPGA next to the hard drive to reduce data transfer volume over the network.

The emergence of NAND-based Solid-State Drives (SSDs) presents a great opportunity to remove the data access bottleneck in hard drives and match the data processing capability of FPGAs. Unlike traditional hard drives whose internal data transfer rate is proportional to the rotating speed, SSDs achieve much higher internal data access rate by exploiting parallelisms among NAND flash chips in the device. In this paper, we propose a new storage architecture, Reconfigurable Active Drive (RAD), which combines the high data processing rate of an FPGA and data access rate of an SSD. Data-intensive applications are ideal candidates for these architectures, including database processing, data mining, image processing, and satellite data processing. In the following sections, we first describe the RAD architecture and derive some performance parameters for our RAD model, followed by evaluations with simulators and FPGA devices. We use two representative scientific applications to evaluate the performance of the RAD model in terms of the end-to-end throughput.

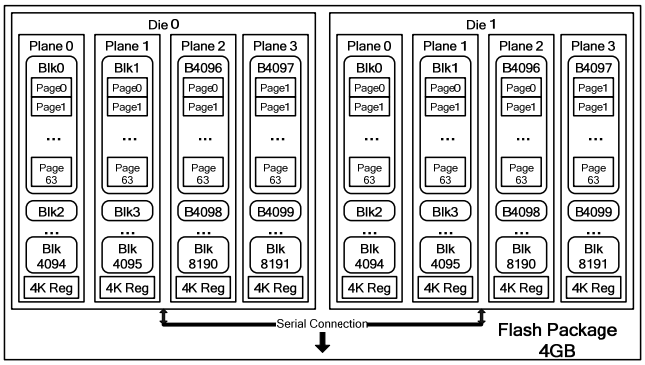

Fig. 1 The internal structure of Samsung 4GB K9XXG08UXM [6][7]

II. RAD ARCHITECTURE MODEL

A typical SSD architecture consists of an array of flash packages, a controller, and a host interface logic unit. A flash package (or chip) has multiple dies that consists of multiple planes. Each plane is equipped with a register, which shares the same size as one page. Data reads and writes are both carried out at the granularity of flash pages, while erases are done at the eraseblocks. An erase operation is needed before a page becomes available for writes. In Samsung K9XXG08UXM [6][7] shown in Fig. 1, it takes 100 μs to transfer one page (4 KB) of data from internal register to the I/O of the package. In other words, the maximum data access rate for a single chip is around 40 MB/s. In order to achieve higher data access rate, RAD can use an array of parallel flash chips to leverage the scalability of flash chips inside an SSD.

The RAD architecture in Fig. 2 is based on typical SSD internal components [6] and uses an FPGA between the SSD

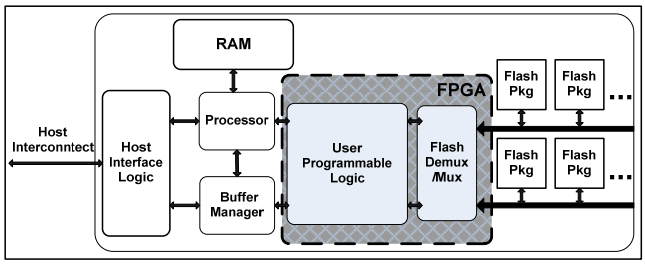

Fig. 2 The RAD architecture [6]

controller and the flash package array. Some SSD components such as flash multiplexers and control logics to the SSD controller can also be implemented on the FPGA. The customized hardware application are developed using the user programmable logic.

In general, scientific applications that are implemented on FPGAs require data access rate up to several GB per second. As shown in Fig. 3, the FPGA device in RAD is directly connected to a multiplexer. The FPGA can access each row of flash package in parallel. While an FPGA is always operating at a much higher frequency than the I/O of flash package array, the multiplexer is used to select different rows of flash package in an interleaved way to maximize the performance. For the flash package in the RAD Model, $\boldsymbol{R_f}$ is defined as the size of the data register. Here, we assume that the size of data register is the same as the size of one page. $\boldsymbol{T_f}$ is defined as the time for serial data access from the data register. Using these two parameters, the maximum data access rate from a single flash chip is $\boldsymbol{R_f/T_f}$ Bytes/Second. If we define $\boldsymbol{F_{HW}}$ as the frequency of the FPGA, $\boldsymbol{W_{HW}}$ and $\boldsymbol{W_f}$ as the number of bytes on the I/O side of the FPGA and of the flash package, the processing throughput of the FPGA is $\boldsymbol{W_{HW}*F_{HW}}$ and the required number of flash chips is $\boldsymbol{(T_f*W_{HW}*F_{HW})/R_f}$. The number of flash package the FPGA could access at one time is determined by the number of columns, $\boldsymbol{W_{HW}/W_f}$, and the number of rows is $\boldsymbol{(T_f*F_{HW}*W_f)/R_f}$. Given the number of rows and columns, theoretically, the RAD model can set up the flash package array to avoid the I/O bottleneck. In most current SSDs, the performance limitation is the device's interfaces, i.e., to read and write via the single port (the read/cache registers) and SATA. It is feasible for RAD to increase the level of concurrency and deliver higher data access rate. Fusion-IO [8] SSD is a good example. In addition, RAD increases the I/O width of the flash package array to match that of the FPGA. As a result, RAD is able to completely pipeline the data access and processing stages.

III. Performance Evaluation of RAD Model

A. Test Environment Setup

In order to evaluate the performance of the proposed RAD model, we utilized the DiskSim Simulation Environment [6][9] from Carnegie Mellon University and Microsoft Research to simulate the top level SSD data transfer rate. In the flash package matrix, the number of columns depends on FPGA I/O width and the number of rows (ways of interleaving) depends on the I/O bandwidth of a single flash. Using the following

TABLE I
Parameters used in RAD model and example values

	Parameters			
	Description	**Symbol**	**Unit**	**Example Values**
Flash Package in RAD	Data Reg Size	R_f	Byte (B)	4096 B
	Access Time to Data Reg	T_f	Second (S)	0.0001 S
	I/O Width	W_f	Byte (B)	1 B
FPGA in RAD	Frequency	F_{HW}	Hertz (Hz)	200 MHz
	I/O Width	W_{HW}	Byte (B)	16 B

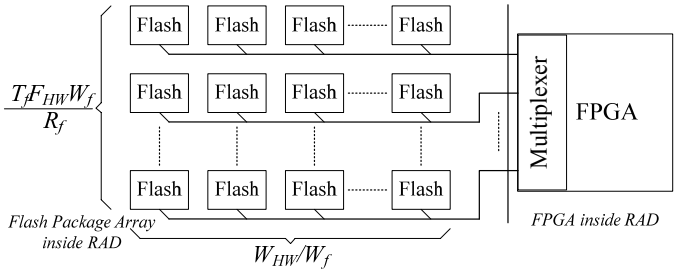

Fig. 3 The flash package architecture of RAD model

flash parameters: (1) page access time of 25 μs, (2) page program time of 200 μs, (3) block erase time of 1.5 ms, (4) model page sizes of 4 Kbytes, (5) block sizes of 64 pages and (6) bus width of 1 Byte, we establish the SSD model composed of N rows and 1 column of single flash packages in DiskSim, which represents N-way of interleaving. We will call it N× 1 SSD model in the following sections.

Once we obtain the I/O bandwidth of the $N\times 1$ SSD model, $\boldsymbol{BW_{N\times 1}}$, from the simulation in Section *III.B*, we can calculate the total bandwidth of the "$\boldsymbol{N\times W_{HW}/W_f}$" SSD in RAD model as $\boldsymbol{W_{HW}/W_f \times BW_{N\times 1}}$ based on the hardware application I/O width. Here we assume that the bus bandwidth would scale with the bus width theoretically. Thus, if an application with 16-byte I/O width is to be run on the FPGA, 16 $N\times 1$ SSD models will be connected to the FPGA in order to expand the I/O width to 16 Bytes. This setup will render the total bandwidth of the flash array in RAD to be $16\times \boldsymbol{BW_{N\times 1}}$. Also, we set both bus and controller delays to zero in DiskSim, which means the bandwidth calculated from DiskSim is the one directly from the flash array. We plan to further investigate the impacts of the bus and controller in the future.

B. DiskSim Simulation Results

Simulation results from DiskSim are based on several trace files collected using Blktrace [10] that records the block I/O of hard drives on the OS level. DiskSim provides output results including the average request size and number per second, which we use to calculate the SSD data transfer rate. Using the original hard drive trace on 1×1 SSD model, simulation results show only 12.37MB/s for reading rate and 6.83MB/s for writing rate. To investigate potential performance gains, we increase the request size per trace from 4 blocks to 512 blocks in the trace file in order to saturate the SSD model bandwidth while the block size is 512 bytes. As shown in Fig. 4(a), the new data reading and writing rate for 512-block requests is 24.61MB/s and 7.54MB/s respectively for the 1×1

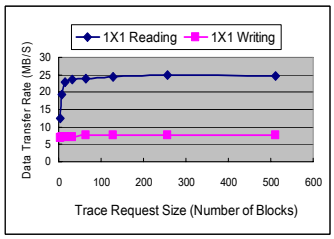

Fig. 4(a) 1 × 1 SSD model data rate

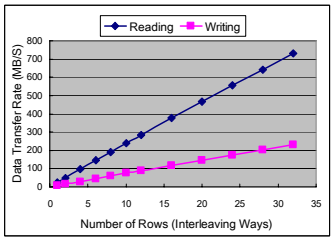

Fig. 4(b) N × 1 SSD model data rate

SSD model with 1 Byte I/O width.

Using the optimized trace files with request size of 512 blocks per trace, we have simulated both the reading and writing data rates of our $N \times 1$ SSD model by increasing N starting from 1 to 32. As shown in Fig. 4(b), the number of ways for interleaving a 1×1 SSD model (single flash) increases linearly with the simulated read and writing data rates from DiskSim.

C. RAD Model Performance Analysis

In the RAD architecture, data access and data processing can be fully overlapped. If we divide the end-to-end throughput into three logical phases, raw data reading, FPGA processing, and result data writing, the slowest phase is the bottleneck. Note that the FPGA processing rate is application dependent. In this paper, we select two applications, Data Encryption Standard (DES) using a processing core of 16-byte I/O and Sobel Edge Detection (SED) using a processing core of 8-byte I/O. We use FPGA devices on two reconfigurable computers (RCs) to collect the FPGA processing rate. DES is implemented on Virtex4LX200 FPGA of SGI RC100 running at 200MHz, while SED is on Virtex2VP50 FPGA of Cray XD1 running at 199MHz. The actual FPGA processing rates of both applications are listed in TABLE II. Based on our RAD model, DES needs 16 columns and SED needs 8 columns of flash array to match the application I/O width on the FPGA side. From the way we calculate the total bandwidth in Section *III.A*, by matching the actual FPGA processing rate, we are able to calculate the required data rate for the $N \times 1$ SSD model, which are 72.96MB/s for DES and 133.12MB/s for SED respectively. According to the writing rate (slower than reading) we have simulated in Fig. 4(b), we derive the number of N to be 10 for DES and 18 for SED. The corresponding flash matrix and RAD reading/writing bandwidth are also listed in TABLE II.

For comparison, we have run DES and SED on SGI and Cray, respectively, with both RC and software implementations and measured the end-to-end throughputs. Our results in TABLE III show that RAD achieves a significant gain in terms of the end-to-end throughput compared with both RC and software implementations. The remarkable speedups of up to 541× on the RAD architecture are due to the fact that data access and data processing are fully pipelined and the hardware implementation on FPGA is leveraged for data-intensive applications. In contrast, while the RC platform takes the advantage of hardware implementation, it is not able to overlap data access and data processing completely.

TABLE II
FPGA PROCESSING RATE AND RAD BANDWIDTH OF TWO APPLICATIONS

	DES	**SED**
Actual FPGA Processing Rate Collected on RCs	1.14GB / S *Collected on SGI*	1.04GB/S *Collected on Cray*
Required Data Rate for the N × 1 SSD model	72.96MB/S	133.12MB/S
Required Flash Matrix	10 × 16	18 × 8
RAD Reading Bandwidth	3.72GB/S	3.33GB/S
RAD Writing Bandwidth	1.18GB/S	1.04GB/S

TABLE III
PERFORMANCE COMPARISON AND SPEEDUPS OF TWO APPLICATIONS

	DES		**SED**	
RAD end-to-end throughput	1.14GB/S		1.04GB/S	
RC end-to-end throughput	23.08MB/S *Collected on SGI*		14.13MB/S *Collected on Cray*	
SW end-to-end throughput	7.34MB/S *Collected on SGI*		1.97MB/S *Collected on Cray*	
Speedup (RC | SW)	**50.6**×	**159.0**×	**75.4**×	**540.6**×

IV. CONCLUSIONS

In this paper, an FPGA accelerated storage architectural model, RAD, is presented. The model defines an FPGA as a reconfigurable accelerator residing within an SSD and exploits potential performance gains of the data processing from the SSD. The RAD has been modeled using an SSD simulator and evaluated from the performance standpoint with existing FPGAs and two scientific applications. The results demonstrate that RAD outperforms both RC and software implementation approaches. Future work will explore the RAD architecture and perform evaluations on a wide range of applications.

REFERENCES

[1] A. Acharya, M. Uysal, and J. Saltz, "Active disks: programming model, algorithms and evaluation," *ACM SIGPLAN Notices*, vol. 33, no. 11, pp. 81--91, Nov. 1998.

[2] L. Huston, R. Sukthankar, D. Hoiem, and J. Zhang, "SnapFind: Brute force interactive image retrieval," in *Proc. the Third International Conference on Image and Graphics (ICIG'04)*, Dec. 2004, pp. 154--159.

[3] C. W. Smullen, IV, S. R. Tarapore, S. Gurumurthi, P. Ranganathan, and M. Uysal, "Active storage revisited: the case for power and performance benefits for unstructured data processing applications," in *Proc. the 2008 conference on Computing Frontiers (CF'08)*, May 2008, pp. 293--304.

[4] M. Franklin, R. Chamberlain, M. Henrichs, B. Shands, and J. White, "An architecture for fast processing of large unstructured data sets," in *Proc. the IEEE International Conference on Computer Design (ICCD'04)*, Oct. 2004, pp. 280--287.

[5] *Streaming Analytics with the Netezza Performance Server Appliance,* Netteza, Inc., Dec. 2007.

[6] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse and R. Panigrahy, "Design Tradeoffs for SSD Performance," in *Usenix Annual Technical Conference* (USENIX '08), June 2008, pp. 57--70.

[7] *256M x 8 Bit / 128M x 16 Bit NAND Flash Memory*, Samsung, Inc., Aug. 2003.

[8] Fusion-I/O, http://www.fusionio.com.

[9] J. S. Bucy, J. Schindler, S. W. Schlosser, G. R. Ganger, "The DiskSim Simulation Environment Version 4.0 Reference Manual," May. 2008

[10] J. Axboe, A. D. Brunelle, "Blktrace User Guide," Feb. 2007.