SPaC: A Segment-based Parallel Compression for Backup Acceleration in Nonvolatile Processors

Xiao Sheng Yiqun Wang Yongpan Liu Huazhong Yang
{ypliu,yanghz}@tsinghua.edu.cn
Tsinghua National Laboratory for Information Science and Technology
Dept. of EE Tsinghua Univ. Beijing, 100084, China

***Abstract*—Nonvolatile processor (NVP) has become an emerging topic in recent years. The conventional NV processor equips each flip-flop with a nonvolatile storage for data backup, which results in much faster backup speed with significant area overheads. A compression based architecture (PRLC) solved the area problem but with a nontrivial increasing on backup time. This paper provides a segment-based parallel compression (SPaC) architecture to achieve tradeoffs between area and backup speed. Furthermore, we use an off-line and online hybrid method to balance the workloads of different compression modules in SPaC. Experimental results show that SPaC can achieve 76% speed up against PRLC and meanwhile reduces the area by 16% against conventional NV processors.**

I. Introduction

Nonvolatile memory based computing system has become a hot topic in recent years[1–6]. Nonvolatile processor (NVP), which can instantly retain system state under power interruptions, has created new applications for low power system consisting of energy harvesting and fine-grained power management[7, 8]. Different from conventional processor, NVP backs up the states in local nonvolatile flip-flops[9] instead of external nonvolatile memory when sleep[7]. The distributed architecture provides NVP instant and ultra low-power sleep/wakeup features [10]. Therefore, many works have concentrated on the NVP implementations[11–13] with various nonvolatile materials in recent years.

The normal NVP replaces all the registers with nonvolatile ones for fully parallel data backup (full replacement architecture). However, it leads to more than 40% area overheads[13, 14]. An area efficient architecture (PRLC)[14] was proposed to reduce the area by using compression before storing. It can reduce the number of nonvolatile registers by 4 times. However, the compressing and decompressing operations are time consuming and deteriorate the sleep and wake-up speed.

In order to guarantee both the sleep/wakeup speed and area efficiency, we propose a segment-based parallel compression (SPaC) architecture. It is a tradeoff between the full replacement architecture and the centralized compressing architecture (PRLC). Our contributions are listed as the followings:

This work was supported in part by the NSFC under grant 60976032, High-Tech Research and Development (863) Program under contract 2013AA013201 and National Science and Technology Major Project under contract 2010ZX03006-003-01.
DATE'13, March 18-22, 2013, Grenoble, France.

1) This paper proposes an SPaC architecture to tradeoff the area overheads and the backup speed. It partitions the data stream into several segments and compresses them in parallel.
2) We proposed a hybrid off-line/online method to balance the workload of each segment to reduce the compressing time. An off-line algorithm is used to cluster the registers to achieve similar compressing time for each compression module, while an online method balances the dynamic variations during program execution.
3) We evaluate the SPaC and hybrid method in a nonvolatile 8051 processor. Several real programs are used to validate the effectiveness. Experimental results show that the SPaC improves the backup speed by up to 83% compared with PRLC and reduces the area by 16% against the full replacement architecture.

II. Motivation

In this segment we first introduce the proposed SPaC architecture by comparing it with PRLC and the full replacement architecture. We then evaluate SPaC and give out its design challenges.

II.A. Architecture Comparison

The conventional NVP uses the full replacement architecture to backup the system states. As Fig. 1 (a) has shown, each register is connected to a nonvolatile cell. The backup process is parallel and fast but leads to nontrivial area overheads due to a large number of nonvolatile cells. An area efficient NVP architecture (PRLC) in Fig. 1 (b)) uses a compression module (CM) to reduce the number of nonvolatile cells as well as the area. However, the CM compresses the data stream bit-by-bit causing longer backup time.

In order to achieve tradeoffs between area efficiency and backup speed, we proposed a Segment-based Parallel Compression (SPaC) architecture in Fig. 1 (c). We divide the system states into several segments and equip each segment with an individual CM. In SPaC, all segments are compressed in parallel to achieve a faster backup speed against PRLC. Meanwhile, SPaC can reduce the area against the full replacement approach. We will evaluate SPaC's area and speed behaviors as below.

II.B. Area and Backup Speed Behavior

Two key metrics of SPaC are area and backup speed. We will evaluate those metrics under different number of segments

M to show the motivation. The evaluation is based on an actual NVP platform[10] and we find some meaningful results in Fig. 2.

Fig. 2(a) shows the normalized chip area under different segment number M. It represents the PRLC architecture when $M = 1$. The area is approximately linear to the number of segments, as more segments bring more additional CMs. In our experiments, the area savings from compression cannot compensate the area increasing of CMs when $M > 7$. Moreover, we adopt the run-length encoding (RLE) algorithm in SPaC and its compression ratio decreases when M increases due to shorter 0/1 continuous sequence. Therefore, M in SPaC is limited by those factors.

Fig. 2(b) shows the compression speed under different M. Generally speaking, larger M leads to deeper parallelism and fewer clock cycles. However, when $M > 6$, the speedup is trivial. We use three curves to indicate the average value and the upper/lower bound in Fig. 2(b). The variations come from the input changes at different backup points. If the variation is large, it can significantly degrade the backup speed.

In sum, SPaC will achieve a considerable promotion in the backup speed with reasonable area increasing compared with PRLC. We should find a good compromise point considering both area efficiency and backup speed.

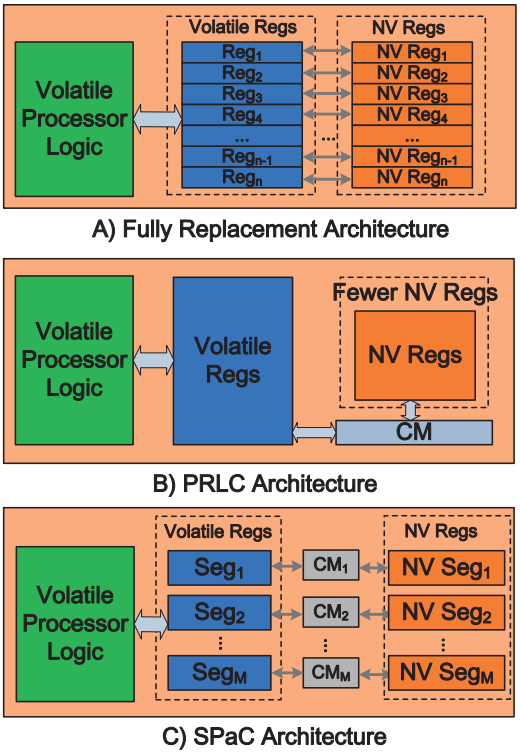

Fig. 1. Architecture Comparison

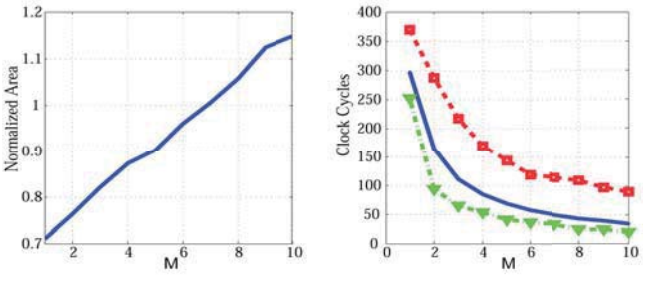

Fig. 2. Area and speed performance vs M

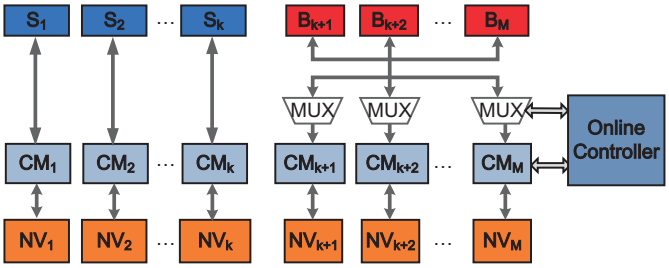

Fig. 3. SPaC structure

II.C. Design Challenges

The overall backup speed is determined by the slowest segment and strongly depends on the speed difference among all CMs. Intuitively, if we equalize the speed of all CMs, we can achieve the fastest compressing speed. However, the compressing speed of every CM varies at different backup points and it is difficult to realize the optimal case. Therefore, the key challenges of SPaC come from the speed differences among all CMs and the workload variations. It contains two major aspects: i) to find a partition strategy to equalize the average compressing speed of each CM, ii) to dynamically adjust the workload variations at different backup points.

III. SPaC Design

We first illustrate the SPaC diagram in this section. Later we proposed an offline and online hybrid method to deal with the speed difference of all CMs in SPaC.

III.A. SPaC Architecture

Fig. 3 shows the detailed diagram of SPaC with M segments. The registers are clustered into M segments denoted as $\{S_1, S_2, ...S_k, B_{k+1}, ..., B_M\}$ and each segment is connected to a CM module for parallel compression. To support online workload adjustment, we design a specific structure to allow a part of segments to allocate their data to CMs from other segments. We denotes such segments as B_i and other segments as S_i. Segments S_i usually have relative small speed variations and do not support data reallocation. To dynamically reduce the workload variations in segments B_i, we use a set of MUXs to allocate their input data to CMs from other segments. An online controller provides the selecting signals for MUXs based on the following algorithm. In real cases, the multiplexing usually leads to trivial area overheads because of relative small number of segments B_i. Besides, we have to add some prefix flags to record the data flow of each segment for data recovering operations.

III.B. Offline Partition Algorithm

The offline algorithm tries to determine the length of each segment so as to balance the compressing time in each corresponding CM. Supposing that we partition the system state vector V into M segments, the offline algorithm uses an iterative way to approach equal compressing speed in each segment. The offline algorithm is described in Algorithm 1.

Algorithm 1 Offline Algorithm

Input: $\mathbf{V}, \mathbf{M}, \mathbf{Var}_{th}, \mathbf{loop}_{th}$
Output: $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_M)$
Variable: $std, time, step, loop$
1: $\mathbf{S}_i = length(\mathbf{V})/\mathbf{M}$ for $i = 1, 2, \ldots, \mathbf{M}$; $std = \mathbf{S}_{th}$;
2: **while** $std \geq \mathbf{S}_{th}$ and $loop \leq \mathbf{loop}_{th}$ **do**
3: $time = CM(\mathbf{V}, \mathbf{S})$
4: $std = STD(time)$;
5: $step = ceil(std)$;
6: $\mathbf{S}(Indexof(\max(time))) = \mathbf{S}(Indexof(\max(time))) - step$
7: $\mathbf{S}(Indexof(\min(time))) = \mathbf{S}(Indexof(\min(time))) - step$
8: $loop = loop + 1$
9: **end while**

The input of offline algorithm includes the system state vector V, the number of segments M, the threshold of the standard deviation S_{th} and the loop limitation $loop_{th}$. The output $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_M)$ represents the length of each segment. We use the standard deviation of average compressing clock cycles in each segment to measure the variation. In Algorithm 1, std is the temporary standard deviation under the current partition S; $time = (t_1, t_2, , t_M)$ gives out the average clock cycles of all segments; $step$ is the max step value to change the vector length and $loop$ is the iterating number.

We use the equal partition as the initial S and set std to S_{th}. In each loop, we calculate the compressing time of each segment to get $time$ and its variation std. We find the segment with the maximum average clock cycles in $time$ and reduce its vector length by one step. Similarly, the opposite operation is performed to the segment with the minimum average clock cycles. We keep changing S until std is smaller than S_{th}, otherwise it will return an error message. In case of failure, we either reduce S_{th} or set a larger $loop_{th}$.

III.C. Online Balancing Algorithm

In order to decrease the dynamic workload variations, we proposed an online strategy for segments B_i. Fig.4 shows the structure of the multiplexing among the segments with large variations. Each B_i is divided into two parts. The lower L bits are denoted as BT_i), and they are connected to MUXs. The remaining parts are denoted as BH_i, and they are directly connected to CM_i. During the compression, the online controller monitors the complete state of each segment. Supposing B_s completes its compression and BT_i are not processed, the online controller switches the BT_i of the slowest segment to CM_s. The online algorithm is described in Algorithm 2.

The input of online algorithm is the process signal P_{EN} and the CM completion status vector $C = [C_1, C_2, \ldots, C_K]$. The output $SL = (SL_1, SL_2, \ldots, SL_K)$ is the select signals of the MUXs.

During the compressing operation, the P_{EN} signal is set to high to enable the online controller. When one of C_i is set to 1, it means the CM_i has completed the compression. The controller sets SL_i as the index of the slowest segment to multiplex its lower part to CM_i. This operation is repeated until the completion of all compression. In practice, we can partition the lower bits of these blocks into more parts and increase the multiplex degree for better performance.

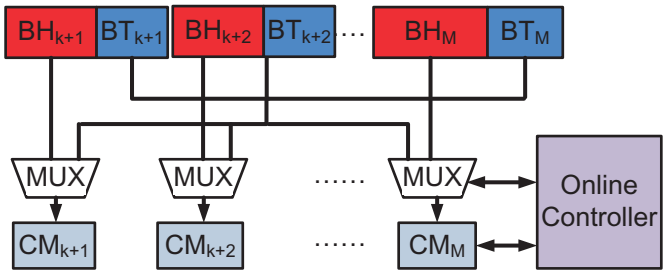

Fig. 4. Online multiplex structure

Algorithm 2 Online Algorithm

Input: $\mathbf{P}_{EN}, \mathbf{C} = \mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_K$
Output: $\mathbf{SL} = \mathbf{SL}_1, \mathbf{SL}_2, \ldots, \mathbf{SL}_K$
1: $Compressstart, \mathbf{P}_{EN} = 1$
2: **while** $\mathbf{P}_{EN} == 1$ **do**
3: **if** $\mathbf{C}_i == 1$ **then**
4: $\mathbf{SL}_i = indexof(min(C)), \mathbf{C}_i = 2$
5: **end if**
6: **end while**

IV. Experiments

In the experiments, we evaluated the SPaC architecture based on an actual NVP[10] with four real benchmarks. The timing information is extracted from Modelsim and the area values are from Synopsys Design Compiler under SMIC 0.13um CMOS process.

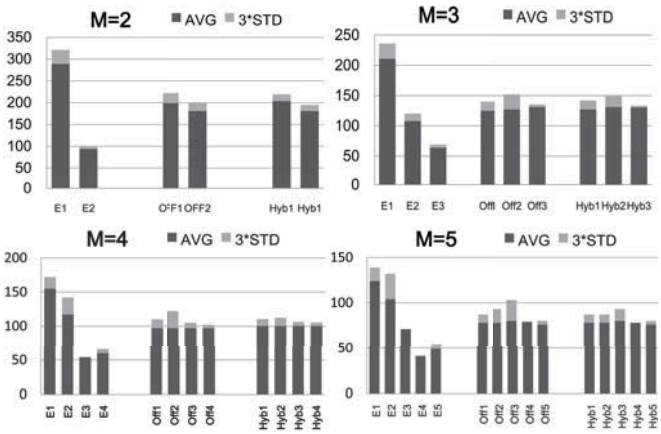

Fig. 5. Performance comparison between equally partition and offline strategy

We compare metrics of an NVP using equal partition, offline only partition and hybrid method under different M in Fig. 5 and table I. In Fig. 5, each bar represents a segment. The dark bar represent the average compressing clock cycles, while the light one shows its variation. We can see that the offline algorithm balances the workload of different segments

TABLE I
SPEED PERFORMANCE COMPARISON BETWEEN EQUALLY PARTITION,OFFLINE ONLY AND OFFLINE ONLINE HYBRID

M	Equally partition			Offline only				Offline Online hybrid				
	Average	3*Std	Sum	Avg	Var	Sum	Reduction/%	Avg	3*Std	Sum	Var Reduction/%	Overall Reduction/%
2	288.9	31.8	320.7	199.2	22.5	221.7	30.8	203.8	15.6	219.4	30.6	1
3	211.1	24.9	236	130.6	25.4	156	33.8	130.2	17.7	147.9	30.3	5.1
4	155.4	25	180.4	97.2	24.9	122.1	32.3	100.2	17	117.2	31.7	4
5	124.4	28	152.4	80.2	23.1	103.3	32.2	82.3	16.1	98.4	30.3	4.7
6	101.8	18.4	120.2	67.7	22.7	90.4	24.7	67.5	13.6	81.1	40	10.2
7	96.1	24	120.1	58.1	15.9	74	38.3	59.5	11.5	71	27.6	4

effectively while the online algorithm further decreases the variations. As table I shows, the offline only partition can improve the speed performance by 32% compared to the equal partition. The online strategy further reduces the variations by average 31.7% and improves the overall speed performance by up to 10%.

TABLE II
OVERALL PERFORMANCE COMPARISON

Architecture		Area Reduction/%	Speed Up/%
Conventional		0	-
PRLC		30.4	0
SPaC	**M=2**	25.8	42.2
	M=3	21	63.1
	M=4	16.3	71.6
	M=5	11.6	76.6
	M=6	7.0	80.8
	M=7	2.2	83.1

Table II shows the overall area reduction and speed improvement of the SPaC architecture. It makes a tradeoff between the full replacement architecture and PRLC. By choosing an optimal M, we can satisfy the need of different applications or systems. When $M = 4$, SPaC can achieve up to 71% speed improvement against the PRLC and remains 16% area reduction against the full replacement architecture at the same time.

V. CONCLUSIONS

In this paper, we proposed an SPaC architecture to achieve both area efficiency and fast backup speed in an NVP. It consists of an off-line partition and an online adjustment algorithm. The off-line partition algorithm accelerates the backup speed by 32% and the online algorithm provides additional 10% speed gain. Experimental results show that the SPaC improves the backup speed by up to 83% compared to PRLC and reduces the area by 16% against the full replacement architecture. Our future work will focus on the hardware implementation of SPaC and its potential applications.

REFERENCES

[1] X. Wang, Y. Chen, H. Li, D. Dimitrov, and H. Liu, "Spin torque random access memory down to 22 nm technology," *Magnetics, IEEE Transactions on*, vol. 44, 11 2008.

[2] Y. Chen, X. Wang, H. Li, H. Liu, and D. Dimitrov, "Design margin exploration of spin-torque transfer ram (spram)," in *Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on*. IEEE, 2008.

[3] A. K. Mishra, X. Dong, G. Sun, Y. Xie, N. Vijaykrishnan, and C. R. Das, "Architecting on-chip interconnects for stacked 3d stt-ram caches in cmps," in *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*. IEEE, 2011.

[4] W. Zhang, N. K. Jha, and L. Shang, "Design space exploration and data memory architecture design for a hybrid nano/cmos dynamically reconfigurable architecture," *ACM J. Emerging Technologies in Computing Systems*, vol. 5, no. 4, pp. 17.1–17.27, Nov. 2009.

[5] C. J. Xue, Y. Zhang, Y. Chen, G. Sun, J. J. Yang, and H. Li, "Emerging non-volatile memories: opportunities and challenges," in *Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, ser. CODES+ISSS '11, 2011, pp. 325–334.

[6] D. Liu, T. Wang, Y. Wang, Z. Qin, and Z. Shao, "Pcm-ftl: A write-activity-aware nand flash memory management scheme for pcm-based embedded systems," in *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, 29 2011-dec. 2 2011, pp. 357 –366.

[7] M. Zwerg, A. Baumann, and R. e. Kuhn, "An 82ua/mhz microcontroller with embedded feram for energy-harvesting applications," in *ISSCC 2011*, feb. 2011, pp. 334 –336.

[8] Y. Liu, Y. Wang, H. Long, and H. Yang, "Lifetime-aware battery allocation for wireless sensor network under cost constraints," *IEICE Transactions*, vol. 95-B, no. 5, pp. 1651–1660, 2012.

[9] J. Wang, Y. Liu, H. Yang, and H. Wang, "A compare-and-write ferroelectric nonvolatile flip-flop for energy-harvesting applications," in *Green Circuits and Systems (ICGCS), 2010 International Conference on*. IEEE, pp. 646–650.

[10] Y. Wang, Y. Liu, S. Li, and X. Sheng, "A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops," in *ESSCIRC 2012*, 2012.

[11] Rohm Co., Ltd., "Rohm Demonstrates Nonvolatile CPU," *Website:http://techon.nikkeibp.co.jp/english/NEWS_EN/20071004/140206/*.

[12] W. Yu, S. Rajwade, S. Wang, B. Lian, G. Suh, and E. Kan, "a non-volatile microcontroller with integrated floating-gate transistors," in *Proceedings of the 5th Workshop on Dependable and Secure Nanocomputing*. ACM Press, 2011, pp. 1–4.

[13] X. Guo, E. Ipek, and T. Soyata, "Resistive computation: avoiding the power wall with low-leakage, stt-mram based computing," in *2010 Proceedings of the 37th annual international symposium on Computer architecture*. ACM Press, 2010.

[14] Y. Wang, Y. Liu, and Y. Liu, "A compression-based area-efficient recovery architecture for nonvolatile processors," in *Design,Automation and Test in Europe (DATE 2012)*, 2012.