Application Specific Cache Design Using STT-RAM Based Block-RAM for FPGA-Based Soft Processors

Hyunwoo Park¹, Hyun So¹, and Hyukjun Lee¹a)
¹ Department of Computer Engineering, Sogang University
35 Baekbeom-ro, Mapo-gu, Seoul, Korea
a) hyukjunl@sogang.ac.kr

Abstract: As many emerging applications use FPGAs for acceleration (e.g., deep learning, data mining), designing highly-optimized application-specific soft processors on FPGAs gets much attention. Cache is an important component of the soft processor, which is built from Block-RAMs (BRAMs) in FPGAs. SRAM based BRAMs suffer from high static power consumption and area penalty, which prevents implementing large caches with high associativity. STT-RAM based BRAM may be a good solution to these issues. However, existing cache design with SRAM-based BRAMs for soft processors or SRAM and STT-RAM hybrid cache design in conventional processors is not suitable for the cache with STT-RAM based BRAMs. In this paper, we propose a BRAM allocation method that can effectively implement highly set-associative caches whereas reducing the impact of long delays and power consumption of write operations in STT-RAM. Using our framework, we show that the optimal size of STT-RAM based BRAM is 1KB with 64-bit IO width for soft-processor cache and the proposed cache structure reduces power and area on average by 55.3% and 76.9%, and reduces runtime by up to 15.6%. Supporting diverse sizes and associativity enables application specific optimization of a cache. In addition, we show that a hybrid cache with SRAM and STT-RAM is not recommended for the soft processor.

Keywords: FPGA, Block-RAM, STT-RAM, Cache, Soft Processor

Classification: Integrated Circuits

References

[1] P. Yiannacouras, et al.: “Application-specific customization of soft processor microarchitecture,” Proceedings of the ACM/SIGDA 14th international symposium on Field programmable gate arrays (2006) 201 (DOI: 10.1145/1117201.1117231).
[2] Altera. Nios II Gen2 Processor Reference Guide (2015) https://www.altera.com/en_US/pdfs/literature/hb/nios2/n2cpu-niiv1gen2.pdf
1 Introduction

A Field-Programmable Gate Array (FPGA) is a programmable hardware device which gains popularity for implementing various accelerators and provides both better performance and power consumption than CPU in many emerging applications. FPGA-based soft processors get much attention recently because both processor and cache can be customized for different applications and optimization metrics [1]. Cache is a crucial component in the soft processor, which is built from BRAMs. Since applications implemented on FPGAs are diverse, the architecture of BRAM is not optimized for a specific application. As a result, BRAM is tiled and the tiles are arranged in several columns in an FPGA. For instance, a BRAM tile has 36Kb capacity (32Kb data and 4Kb parity) with 72-bit IO width (64 data and 8 parity) in Xilinx 7 series FPGA [10].

There are two commercial soft processors, Xilinx Microblaze [3] and Altera Nios II [2]. These commercial soft processors currently support only a direct-mapped cache to avoid large area penalty and power consumption incurred from supporting set-associativity using SRAM-based BRAMs.

To avoid external memory accesses in soft processors, it is desirable to have large caches with high associativity. STT-RAM is a natural fit for...
BRAMs in FPGAs. In general, STT-RAM consumes small static power and has better area density but is known for its long access latency and high energy consumption of a write operation [4].

Table I extracted from NVSim [4] compares these values for SRAM and STT-RAM based BRAM. The last column shows the benefits of using STT-RAM over SRAM.

Table I. Comparing SRAM and STT-RAM (4KB with 64-bit IO width)

| Parameter        | SRAM  | STT-RAM |
|------------------|-------|---------|
| Area ($um^2$)    | 13556.4 | 4895.82 |
| Read latency (ns)| 0.271 | 1.450 |
| Write latency (ns)| 0.271 | 5.380 |
| Read energy ($pJ$)| 13.635 | 12.364 |
| Write energy ($pJ$)| 11.047 | 17.378 |
| Leakage Power (mW)| 8.766 | 1.007 |

The read/write operation of STT-RAM in Table I shows worse performance and energy consumption (except for leakage power) than SRAM. Unfortunately the architecture of a conventional SRAM-based BRAM cache [3] is not adequate to overcome the large performance loss and energy consumption of writes to STT-RAM. Moreover, applying SRAM and STT-RAM hybrid caches used in conventional processors [11] has to be re-evaluated for soft processors because maintaining both SRAM and STT-RAM BRAM tiles are costly.

In this paper, we propose a BRAM allocation method for STT-RAM based cache. We show that the proposed STT-RAM based cache is superior to its SRAM counterpart. The major contributions are summarized as follows. First, we develop a framework for evaluating STT-RAM based BRAM cache for FPGA based soft processors. Using this framework, energy, power, and runtime of the cache built from STT-RAM based BRAMs can be estimated for various applications and cache configurations. In addition, the tool enables designing an application specific cache. Second, we propose a BRAM allocation method for cache, which exploits density and low static power consumption of STT-RAM. When the optimal size of STT-RAM based BRAM tile (1KB with 64-bit IO width) is used, power and area are reduced by 55.3% and 76.9% respectively on average compared with the baseline SRAM based cache. In addition, when the same die size is used as SRAM, the runtime decreases by up to 15.6%.

Finally, we show that a hybrid cache with SRAM and STT-RAM is not recommended for the soft processor.

2 Proposed BRAM Allocation Method for Cache

Cache consists of data and tag array. The number of BRAM tiles to implement the tag and data array depends on the BRAM allocation method for
cache. We propose an allocation method to optimize for both area and power by exploiting the unique characteristics of STT-RAM based BRAM.

![Fig. 1. Cache access patterns: (a) Cache hit (b) Cache miss](image)

![Fig. 2. BRAM allocation for 2-way set-associative cache: (a) Conventional scheme, (b) Proposed BRAM allocation scheme](image)

For that, two optimization goals are set. First, in Fig. 1(a) and (b), the candidate words of all cache ways are read simultaneously in the beginning of cache accesses regardless of misses. For minimum power consumption, we should minimize the accesses of BRAM tiles in the set-associativity cache. Second, the words in a single cache block should be read (replacement) and written (block fetch) in parallel to minimize the penalty for the miss case (i.e. minimizing the impact of long latency of writes to STT-RAM is crucial to minimize the miss penalty).

To satisfy the first goal, we pack words from as many ways as possible in one BRAM tile as shown in Fig. 2(b). In that example, two words from two ways are packed into one BRAM tile. As shown in equation (1), \( [BLK_{act \_data}] \) BRAM tiles must be activated (read) concurrently in the data array during cache accesses where \( A_s \) and \( IO_{BLK} \) are associativity of a cache and IO width of a BRAM tile respectively.

To satisfy the second goal, different words from the same set should be able to be read or written at the same time. Four different words in a single cache block are distributed over four different BRAM tiles in Fig. 2(b). We prioritize packing words from different ways in a tile over packing words from
the same cache block because the former is a far more frequent.

In general, \( BLK_{data} \) BRAM tiles are necessary to accommodate whole data array to satisfy two goals. The exact equations are shown in equation (2) where \( L_8 \) represents cache line size and \( Row_{BLK} \) represents the number of rows in a BRAM tile. Allocation for the tag array is almost same as the data array except for the fact that tag array bits are padded for the IO width of power of 2. \( BLK_{act,tag} \) (equation (3)) and \( BLK_{tag} \) (equation (4)) represent the number of BRAM tiles required for one set of a tag array and for the whole tag array of a cache respectively.

\[
BLK_{act, data} = \left\lceil \frac{Word \_ Size \times A_8}{IO_{BLK}} \right\rceil \quad (1)
\]

\[
BLK_{data} = \left\lceil \frac{L_8 \times 8 \times A_8}{IO_{BLK}} \times \left\lceil \frac{\# of sets}{Row_{BLK}} \right\rceil \right\rceil \quad (2)
\]

\[
BLK_{act, tag} = 2^{\left\lceil \log_2 \left( Tag \_ Entry \_ Size \times A_8 \right) \right\rceil} / IO_{BLK} \quad (3)
\]

\[
BLK_{tag} = \left\lceil BLK_{act, tag} \times \left\lceil \frac{\# of sets}{Row_{BLK}} \right\rceil \right\rceil \quad (4)
\]

To show the difference between a conventional method and our proposed one, two methods for the data array of a 4 KB 2-way set-associative cache using 2KB and 64-bit IO width BRAM tiles are illustrated in Fig 2(a) and (b) respectively. \( L_8 \) is 16 bytes.

For read and write hit cases, both allocation methods perform the same number of operations. However, for a block read and write upon misses, (a) requires four cycles whereas (b) requires only one cycle. As a result, 3 and 5 BRAM tiles (including the tag array) are required for (a) and (b) respectively. The allocation method (b) may require more STT-RAM based BRAM tiles than (a) for small cache sizes. However, thanks to high density of STT-RAM, area decreases for high associativity and large cache sizes, which overcomes the drawbacks of STT-RAM and leads to performance improvement.

3 Results

We build a framework to design and evaluate the cache, which consists of SimpleScalar [5] and NVSim [4]. SimpleScalar is modified to reflect the behavior of the proposed allocation scheme. Along with clock frequency (100, 200, 300MHz), latencies of cache operations and external memory accesses are adjusted. The baseline soft processor has the same configuration as commercial ones [2] [3] which supports a single-level direct-mapped cache and inorder execution. 5 and 9 applications in MiBench [6] and SPEC CPU 2006 [7] respectively are selected to evaluate the cache. Using simulation results from SimpleScalar and power, area, runtime values for BRAM configurations from NVSim, metrics such as runtime, area, and power are measured. We use Commercial FPGAs containing BRAMs with a tile size of 4KB and 64-bit IO width to build a baseline SRAM-based direct-mapped cache.
3.1 Evaluating power, area and runtime

We evaluate metrics for the proposed BRAM allocation method. The optimal BRAM tile size is 1KB with 64-bit IO width. Fig. 3 shows the power and area reduction normalized to the same sized direct-mapped cache made of SRAM-based BRAMs at 100 MHz. It is assumed that a SRAM based cache uses the conventional method for BRAM allocation.

Power reduction can be up to 89.0% or 55.3 % on average over different cache sizes and associativity. Large dynamic power reduction is achieved because our proposed allocation method reduces the amount of memory reads/writes to BRAMs by 3.3%/20.1%. In addition, static power consumption is much smaller with STT-RAM based BRAMs, which leads to the significant gain.

Area reduction of the STT-RAM is shown in Fig. 3(b). Reduction is 76.9% on average whereas the maximum reduction is 87.5%. In particular, the area of set-associative cache constructed with STT-RAM based BRAMs is even smaller than that of the direct-mapped cache constructed with SRAM based ones.

Runtime reductions averaged over applications are also normalized to that of SRAM based one. Average reductions are 4.67%(100 MHz), 3.95%(200 MHz), and 3.95%(300 MHz). Overall, runtime reduction improves because a soft processor typically does not wait for cache write operations and the frequency of write operations (replacement and block fetch) to BRAMs is smaller than the one for read operations by about a factor of three thanks to our proposed allocation methods. Thus, long write latency barely impacts the execution time of applications.

3.2 Evaluating Application Specific Cache Design

A Wide range of associativity and sizes supported by the proposed allocation scheme make it possible to design an optimized application specific cache. In Fig. 4, if we want to choose a cache design satisfying a 25 sec runtime deadline while optimizing power consumption, 8KB 2-way set-associative cache can be chosen. If performance is the only optimization goal, 16KB 8-way
set-associative cache can be a candidate cache with 15.6% performance improvement.

3.3 Evaluating Hybrid Cache

Recent studies reveal that a hybrid cache with SRAM and STT-RAM is superior to SRAM or STT-RAM only caches [11]. For the cache of a soft processor, however, this is not necessarily true. Since the soft processor runs at 100 MHz to 300 MHz and has the smaller memory bandwidth requirement compared to state-of-the-art processors, replacing a portion of the STT-RAM based cache with SRAM-based BRAM tiles does not improve performance and only increases static power consumption. Assuming one way out of eight ways is replaced with SRAM in STT-RAM cache, Fig. 5 shows severely increased static power consumption and area. Thus a hybrid cache is not recommended in soft processors.

4 Conclusion

In this paper, we propose a BRAM allocation method for constructing STT-RAM caches for the FPGA based soft processor. Our framework can be used to optimize specific metrics of the cache. Using the framework, we show the proposed STT-RAM based BRAM cache can reduce power and area by 55.3% and 76.9% respectively on average. When the same die size is used, up to 15.6% performance improvement is achieved.