# Optimizing Memory Power in Hybrid ARM-FPGA Chips With Lossless Data Compression

Peng Sun
Microelectronics Group
University of Bristol
Merchant Venturers' Building
Woodland Road Clifton
p.sun@bristol.ac.uk

Jose Nunez-Yanez
Microelectronics Group
University of Bristol
Merchant Venturers' Building
Woodland Road Clifton
j.l.nunez-yanez@bristol.ac.uk

## ABSTRACT
In current electronic systems the amount of power needed by the memory components can represent a large percentage of overall power requirements, and while modern DRAM memories offer very low idle power states, the reduction in active power is much more modest. Motivated by these observations, this paper presents a system architecture in which a hardware lossless data compressor/decompressor is connected to the application processor present in the same chip. The compressor increases the amount of time that the DRAM memories can remain in low power state by reducing the number of memory accesses and hence reducing the DRAM memory power consumption. The data compressor is instantiated in the programmable logic side of a ZYNQ device and is controlled by the ARM processors present in this chip moving data between the on-chip local memory and the off-chip DDR memory through the AXI interconnect. Memory active time and power are monitored in the board while different tests are run under the Linux operating system. The presence of the compressor enables the memory to move to a low power mode more frequently and it achieves an overall system power reduction of 12.4%. This figure includes the power overhead introduced by the presence of the compressor itself and it is limited by the efficiency of the low power modes of the considered DDR3 devices and data compressibility.

## Categories and Subject Descriptors
C.1.3 [**Computer System Organization**]: Other Architecture Styles – *Heterogeneous (hybrid) systems.*

## General Terms
Design

## Keywords
Hybrid-FPGA, Low Power, Energy Efficient

## 1. INTRODUCTION
Memory Compression was extensively researched during the past decade and brought important benefits to high performance computing [1], signal and image processing [2], personal mobile devices [3] and web servers [4]. Although increasing memory and bus bandwidth has been the main motivation of memory compression research, recent studies have demonstrated that this approach can also be exploited when the ultimate target is the energy (or power) minimization of a processor-based system. However, these studies either do not aim to reduce the DRAM power consumption, which accounts for a large percentage of overall power requirements in today's embedded system, or do not take full advantage of memory compression for power reduction. This paper proposes a lossless compressor-decompressor engine paired with a hardened processor present in the same chip. The compressor-decompressor engine uses a DMA (Direct Memory Access) technique to move data independently from processor activity. The device selected for this work is the ZYNQ [5] chip with the lossless data compression algorithm created in the PL (Programmable Logic) side of this hybrid chip. The DMA technique manages data transfer between the OCM (On-Chip Memory) and DDR through the ACP (Accelerated Coherence Port) interface available in ZYNQ devices. This design reduces the DDR active time and hence the system power consumption. All the experiments presented in this paper were processed using the ZC702 Evaluation Board [6] and all the source files are made available at OpenCores [7].

The remainder of this paper is organized as follows: Section 2 presents a review of related works. Section 3 reviews the lossless data compression algorithm and hardware employed in the proposed system. Section 4 analyses the techniques to power down the DDR memory during idle periods in which there is no application running in the ZYNQ board. Section 5 describes the hardware architecture of the design used to achieve the DDR power reduction. Section 6 compares the power consumption of the proposed design with that of a design without lossless data compression, and Section 7 concludes this paper.

## 2. RELATED WORK
Currently available techniques to reduce the system power consumption include *bus encoding* which reduces bus power by changing the format of the data transmitted on the processor-memory bus, and *memory organization* which changes the way data is stored in memory so that the address streams generated by the processor have low transition activity. Code and data compression schemes have been applied to memory-processor systems in order to optimize the system power consumption. Previous works on reducing RAM power requirements with memory compression can be categorized as either hardware based or software based.


*FPGAWorld'14*, September 9–11, 2014, Stockholm and Copenhagen.

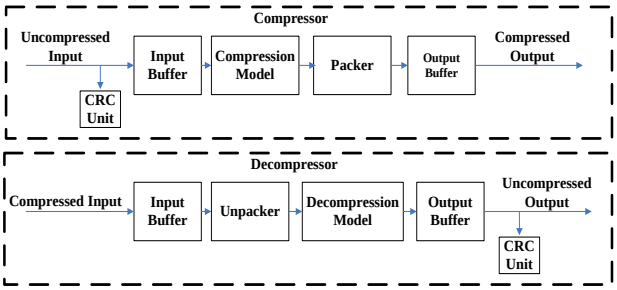


Fig. 1 The Architectural Overview of XMatchPRO

1) *Hardware-Based Approach:* Systems designed under the hardware-based approach are implemented with, and rely on, special-purpose hardware which performs compression and decompression at both the instruction and data level. At the instruction level, code compression techniques [8] store instructions in compressed format and de-compress them during execution by using code compression techniques. Power minimization at this level is achieved by reducing the energy requirements of either I-caches [9][10] or program memory [11][12] through instruction compression. However in this case, only a small part of the main memory, which stores programmes, is being compressed and all data access the memory in their original, uncompressed form. At the data level, the main memory compression techniques [13]-[15] implement a hardware compressor/de-compressor between the cache and RAM. Compared with the compression at the instruction level where only a decompressor engine is required, compression at the data level requires both compressor and decompressor engines during the execution. Memory compression at the data level enables energy saving in two ways. First, storing data in compressed form requires a smaller number of memory accesses to retrieve or write the same amount of information [16]. Second, bus traffic is reduced [17]. However, besides establishing both compression and decompression processes on-line, which would affect the system performance, the design in [16] was tested on SRAM. DRAM thanks to its cost effectiveness is being used as main memory in almost all computers, laptops, embedded systems etc., and since SRAM is only used on components like cache, TLB etc., where fast access is strictly required, the results produced by this design could not be generalized. Moreover, although Shafiee et al. in achieved considerable memory energy reduction [17], the method used in the paper utilized the spare spaces made available by memory compression for extra codings. This design therefore did not take full advantage of memory compression to achieve memory power reduction.

2) *Software-Based Approach:* The software-based approach focuses mainly on swap compression [18] and compressed caching [19]. However, instead of reducing system power consumption, the software-based approach has the main design goals of improving system performance and targeting general-purpose systems with hard disks. Therefore, none of these works has been evaluated on embedded systems for which power consumption and performance are critically important, and few of them provide analysis of their effects on memory power reduction.

Our own work follows the hardware-based approach to exploit data-level memory compression which is interface with software running under a Linux operating system. Different from previous work, our design acts as a DMA master and it investigates how data-level memory compression can be fully utilized to reduce the number of DRAM memory accesses (or memory active time counted in clock cycles), hence reducing the DRAM power consumption. XMatchPRO, a lossless data compression algorithm developed in [20], was selected in this research due to its high throughput and full exploitation of the in-RAM data regularities via its internal parallel architecture. Moreover, the compressed data, although with different lengths, are automatically packed together into streams of 32-bit words that can then be stored in external DRAM. These properties make XMatchPRO a perfect candidate to access the memory efficiently. The recent availability of devices that embed a hardened processor system and a FPGA fabric means that it is possible to closely couple the compressor and processor in a real chip and measure the effects in overall system power which constitutes the main aim of this paper.

**Table 1 Memory Power Mode**

| DDR3 Power Mode | LPDDR2 Power Mode |
|---|---|
| Active | Active |
| Idle | Idle |
| Self-Refresh | Self-Refresh |
| | Temperature Compensated Self-Refresh |
| | Partial Array Self-Refresh |
| | Deep Power Down |

## 3. XMATCHPRO OVERVIEW

XMatchPRO is a dictionary-based lossless data compression hardware which can achieve good compression rates and high throughput. It uses a parallel dictionary of previously seen data in it attempts to match or partially match the current data element with an entry in the dictionary. Compression is achieved by replacing repeated phrases with references to the dictionary; these replaced code words are smaller than the phrase itself. The dictionary is fully adaptive and is built simultaneously to the compression process. The detailed algorithms and architectures are presented in [20]. This section gives an overview of XMatchPRO before moving on to our own design architecture.

At the architecture level, Fig. 1 shows the design architecture of XMatchPRO, which contains a compressor and a decompressor channel. During compression, the compressor receives uncompressed input words through an input buffer and activates the compression model, which generates the required code words. The code words are combined successively into fixed 32-bit width words by the packer and are written to external memory through a write buffer. The decompressor is responsible for the reverse process, in which data is read from the external memory and the required dictionary references are generated to allow the decompressed data to be created. To optimize compression efficiency, XMatchPRO allows partial matches of different widths together with run-length coding when the same dictionary location is hit two or more times. The compression ratio in this paper is defined as the ratio of input bits over the output bits and XMatchPRO achieves a typical 2:1 compression ratio at throughput of 400Mbytes/s.

## 4. LOW POWER DDR STATE

The ZYNQ System on Chip memory controller supports both DDR3 and LPDDR2 (Low-Power-DDR2). JEDEC Solid State

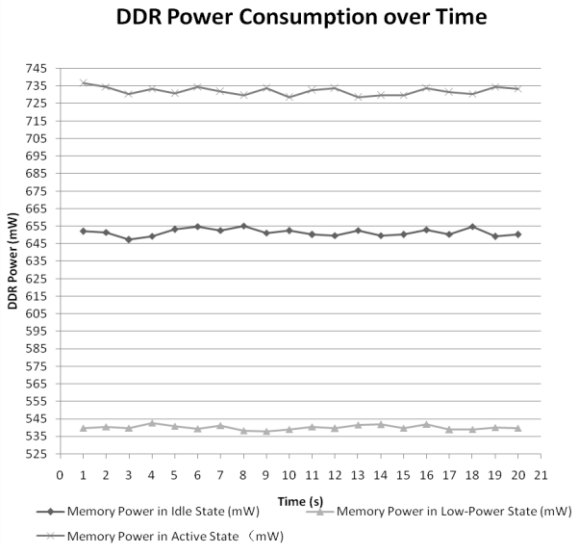


Fig. 2 DDR Power Consumption in different memory states

Technology in [21] gives a comprehensive comparison between the two. Besides using a 1.2V voltage supply compared to the 1.5V power required for DDR3, additional power savings of LPDDR2 come from extra low power modes as shown in Table 1. The temperature-compensated self-refresh mode enables DRAMs to refresh less often at low temperature. Moreover, compared to the self-refresh mode applied on DDR3, LPDDR2 can be programmed to enter a partial array self-refresh mode, a technique which allows refresh operations to perform not across the full memory cell arrays but only within specific banks where data retention is required. Furthermore, LPDDR2 offers a deep power down mode, which sacrifices all memory content for power reduction. Any command that arrives while the DRAM is in deep power down mode is stored in the Content Addressable Memory (CAM) and is processed after deep power down exit and DRAM re-initialization.

Our current board ZC702 includes DDR3 type memory. Although this has been used for all the experiments, except the deep power down mode in which all the memory contents are sacrificed, the results should be extendible to LPDDR2 memory, which offers deeper power down states. Before describing the hardware design to reduce the power of DDR3 when active, this section describes some techniques to power down the DDR3 in its idle period, when no application is running.

The memory controller in the ZYNQ device allows users to clock-gate the DRAM and hence reduces power in the memory. When this feature is enabled, the DDR PHY is allowed to stop the clocks going to the DRAM. Moreover, the DDR memory controller can dynamically use pre-charge power down mode to reduce the power consumption during the idle period. All these features are effective only when the DDR is in self-refresh mode. In this mode, DRAM contents are maintained even when the DDR controller core logic is fully powered down, hence allowing the DDR clocks to be stopped. During self-refresh mode, software must ensure that no transactions arrive.

Fig. 2 shows the measured DDR power reduction effect for 20 seconds when the memory is clock-gated. The memory is being read and written continuously in the active state, doing nothing in the idle state and being clock-gated in the low-power state. According to Fig. 2, more than 100mW DDR power is reduced from the idle state when the DDR is clock-gated, and an average power reduction of 17.6%, from 651.46mW to 537.03mW, is achieved. The power values were taken by software written to monitor various power rails of a ZC702 evaluation board. The details of the experiment setup are described in Section 6. Therefore, when there is no application running on board, the DDR memory can be clock gated so that most of the DDR dynamic power consumption is saved.

## 5. HARDWARE ARCHITECTURE

The compressor/decompressor and control logic are shown as the shaded component in Fig. 3 and were created on the Programmable Logic side of the ZYNQ System on Chip with all interfaces necessary to communicate with the Processing System side through ACP interconnect. Solid arrows show control from master to slave and data flow in both directions. Data is transmitted, as shown by the dotted arrows, between the OCM

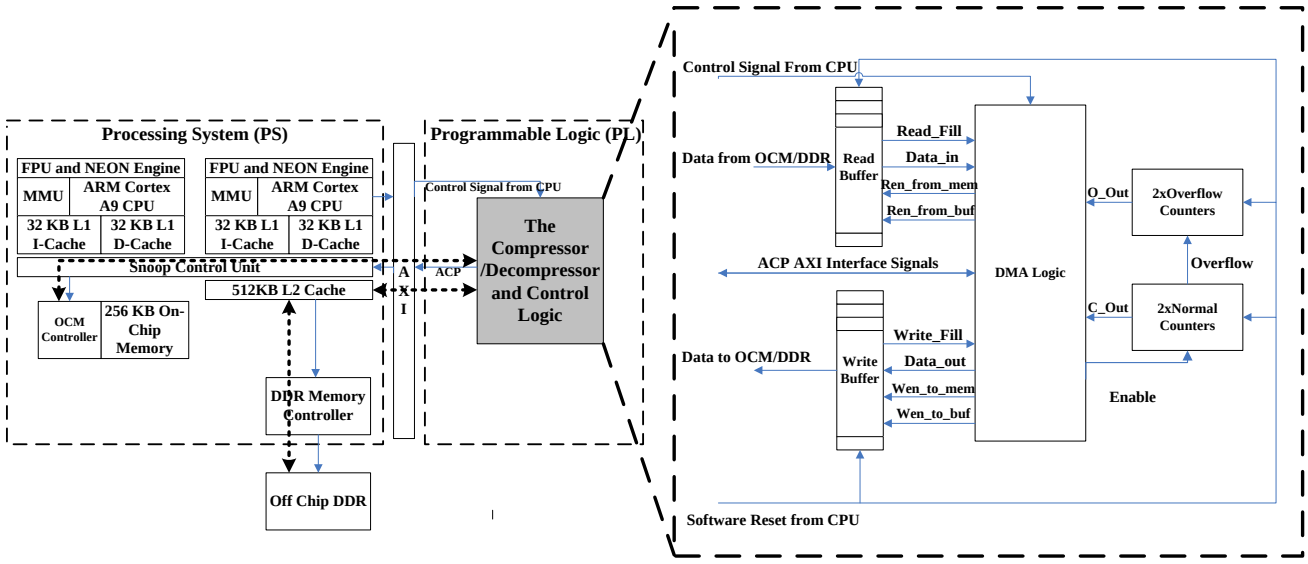


Fig. 3 Design Architecture without XMatchPRO

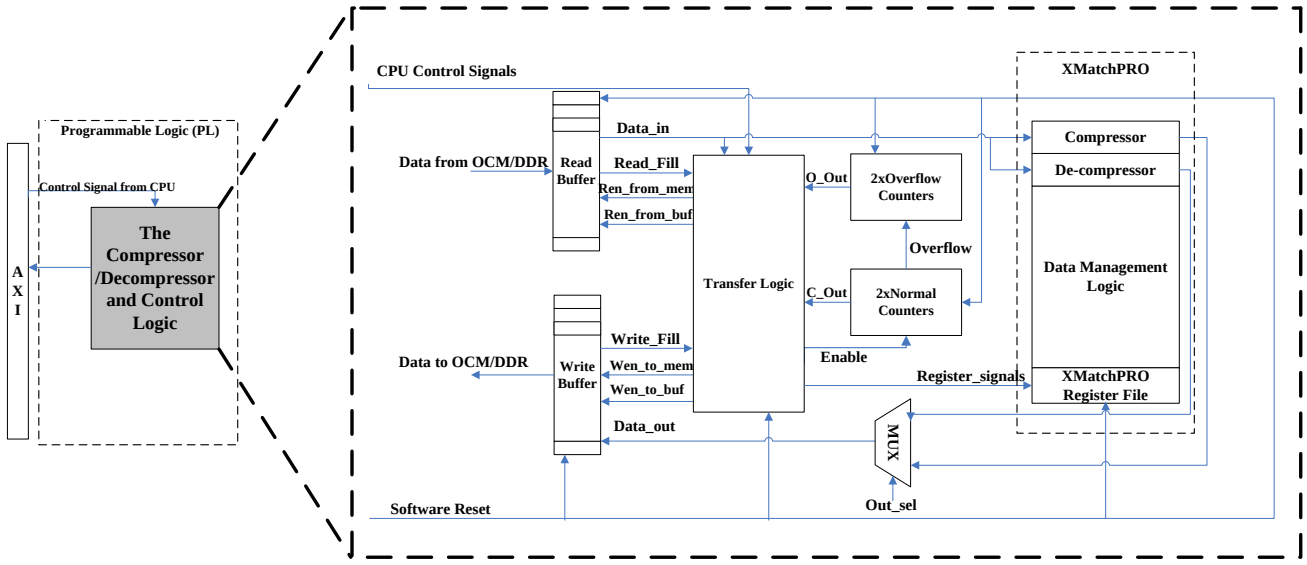


Fig. 4 Design Architecture with XMatchPRO

and the off-chip DDR.

When the CPU needs to write data to the DDR, it writes it first to the On-Chip Memory, which has a smaller access latency compared to the DDR, (1 clock cycle vs hundreds of clock cycles) and sends control signals to the compression logic in the PL. Then this compressor logic reads data from the OCM, processes and transfers it to the L2 cache and hence to the off-chip DDR through the ACP. Finally, this compressor logic sends a signal back to the CPU indicating the completion of the transfer. A reverse mechanism is used when the CPU needs to read data from the DDR. The decompressor logic, after being activated by the CPU, then transfers data from the DDR to the OCM for the CPU to read.

Since memory active time and power consumption are measured and compared with and without lossless data compression, we have developed two different versions:

## 5.1 Design Without XMatchPRO

In this case, data is directly transferred from one memory location to another without compression or decompression. Therefore, the control unit only contains a read buffer, a write buffer, 32-bit counters which measure the main memory active time and the total time duration when a benchmark is running in clock cycle and DMA logic which manages all data transfer, as shown in Fig. 3.

The Read Buffer is a 256x32-bit duplex buffer which can perform simultaneous read (from memories) and write (to the DMA). This buffer is read and written from address 0 again when overflowed. The design of the Write Buffer uses the same logic. Similar to the Read Buffer, the Write Buffer content can be filled (by the DMA Logic) and sent (to the memory) simultaneously.

Whenever the CPU wants to activate the DMA Logic for data transaction, it first sends a burst read request with data sizes and source memory address to enable reading from the memory to the read buffer. The DMA Logic is activated and outputs data to the Write Buffer when the number of 32-bit words in the Read Buffer reaches a pre-defined threshold value. The Write Buffer starts to output data to memories when the number of 32-bit words in it reaches a pre-defined threshold value. Moreover, acknowledge signals are sent from the DMA Logic to the CPU indicating the completion of the data transfer.

## 5.2 Design With XMatchPRO

In this case, data transferred from OCM to DDR are sent through a compressor, and those transferred from DDR to OCM are sent through a decompressor, so that the DDR always stores the data in compressed states. Besides all the components discussed in the previous section, a compressor and a decompressor are now added to the original design, as shown in Fig. 4.

While other components remain the same, as described in Part 5.1, the Read and Write Buffer are connected directly to the compressor and decompressor. Whenever the CPU wants to activate a data transfer, it first sends compressor/decompressor-choice signals and write corresponding compressor/decompressor registers to enable XMatchPRO. Depending on which engine is chosen, compressor or decompressor, XmatchPRO automatically produces compressed data to the DDR through the compressor or produces uncompressed data to the OCM through the decompressor. After then, the rest process is the same as described in Part 5.1. Data saved in the DDR are now in compressed form and the data saved in the OCM remain uncompressed.

# 6. POWER COMPARISON

The hardware design was tested by running six software benchmarks, each contains artificial data that was generated to obtain certain compression ratios in order to measure the power savings and trade-offs regarding compressor utility, under Linux OS on ZC702 Evaluation Board. ZC702 is bootable by an SD card which contains both the hardware design as well as a Linux Operating System and which communicates with a host desktop through Secure Shell (SSH). Whenever running a benchmark, power-analysis software is running simultaneously with it, recording and printing power values used by various sections of the system. This power-analysis software calculates and outputs

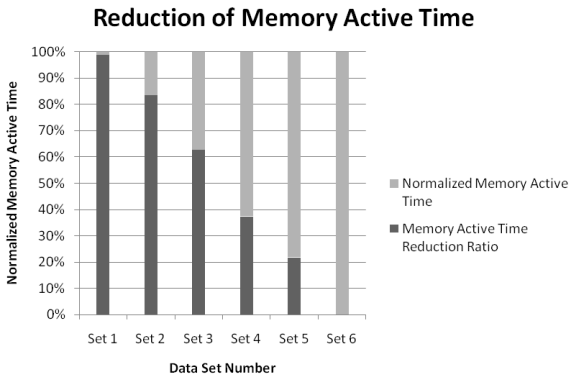


Fig. 5 Reduction of Memory Active Time

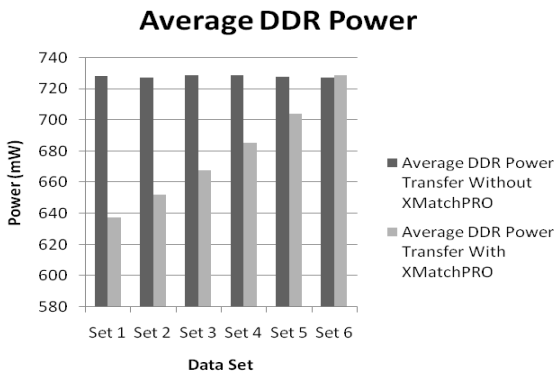


Fig. 6(a) Reduction of DDR Power due to Memory Compress

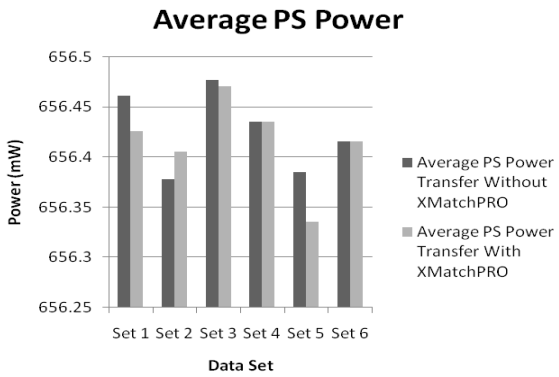


Fig. 6(b) Processing System Power Consumption

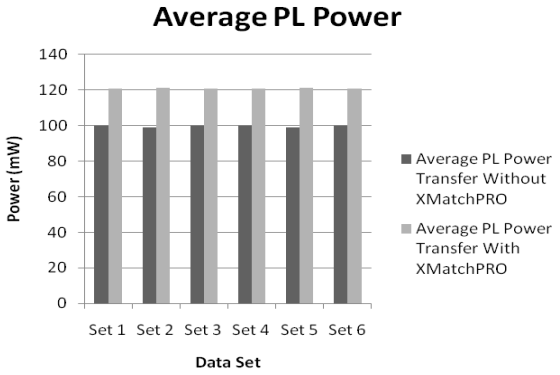


Fig. 6(c) Programmable Logic Power added by XMatchPRO

power consumptions for PS, PL, Block Memory, DDR and External IO by monitoring various power rails on ZC702.

Our power consumption comparison is based on data transaction with and without lossless data compression. Six data sets, each with a size of 16kB, were chosen with different levels of compressibility. Data Set 1 has a compression ratio higher than 100:1, and Data Set 6 is not compressible. Sets 2 to 5 were chosen with data compression ratios of 5.6:1, 2.7:1, 1.6:1 and 1.2:1 respectively. The compression ratio of data is defined as the ratio of the input bits over the output bits. Hence a higher compression ratio indicates a better compressibility of data. Therefore, six tests were performed, with each test transferring one of the six data sets from the OCM to the DDR and back from the DDR to the OCM 500 times so that the duration was long enough for the power analysis software to capture the power value. Every time when the same set of data moved back into the OCM from the DDR, the CPU checks that the received data is the same as the data previously stored in the OCM. Any mismatch of data terminated the transfer loop with an error message printed out.

Fig. 5 presents the reduction ratio of memory active time for every Data Set. It shows that the reduction on memory active time is proportional to the compressibility of the data, and a good compression ratio can significantly reduce the memory active time. Therefore, Data Set 1 shows a large reduction of memory active time (dark shaded) due to its large compression ratio and the normalized memory active time (light shaded) is very small.

Fig. 6 shows the comparison results in terms of average DDR power, PS power, PL power and total power. Even the switch between active and idle period of the memory is sometimes too fast for the software to capture, the average DDR power can be calculated, using the two timer values provided by the counters in the control logic, by averaging the total memory energy consumption during both active and idle period throughout the duration of the test. Average power consumed by the rest of the system was obtained by averaging all values from the power measurement software. Average total power was obtained by adding the Averaged DDR, PS, PL and all other components' power consumption.

According to Fig. 6(a), there is an average DDR power reduction of up to 12.4%, from 728.6mW to 637.8mW, when running Data Set 1. This amount of power is the DDR dynamic power saved due to the reduction of memory active time. However, while transfer with XMatchPRO does not affect the PS power as shown in Fig. 6(b), XMatchPRO itself adds an average of 22.4mW to the PL power, shown in Fig. 6(c). (The power values in Fig. 6(b) and Fig. 6(c) do not have any meaningful variance when measurement accuracy is accounted.) This additional power consumption is introduced since more hardware is used in the PL side. Therefore, if the compressibility (represented by the compression ratio) of data is worse than 1.25:1, the reduction of the DDR power is smaller than the 22.4mW power introduced by XMatchPRO, and the system will suffer from a net increasing in average total power, as shown in Fig. 6(d) when running Data Set 6. However, the research results in [20] show an average compression ratio of

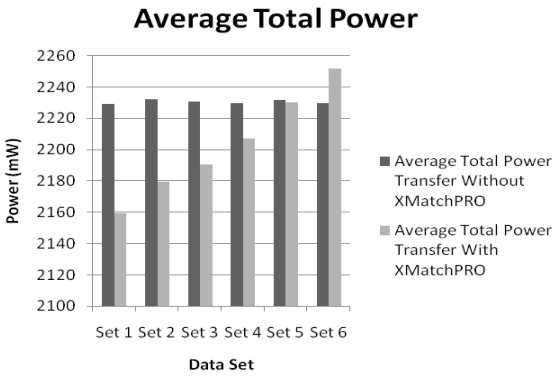


Fig. 6(d) Net Average Total Power of the System

2:1 for typical main memory data used in an engineering environment. In conclusion, the compressor/decompressor achieves up to 12.4% memory power reduction by increasing the amount of time that the DRAM memories can remain in idle state when running an application, and the power down techniques described in Section 4 achieves a further 17.6% memory power reduction by putting the memory from the idle state to the low power state when there is no application running. Therefore, in a perfect system where XMatchPRO runs so fast that the memory can always be put in the low-power state, a 26.3% reduction of the memory power can be achieved by putting the memory directly to the low power state from the active state.

Table 2 shows a summary of the hardware implementation details. The hardware design with XMatchPRO utilizes 2% more registers and 13% more LUTs. Both designs achieve the same frequency.

**TABLE 2**
**Implementation Details**

| Complexity | Without XMatchPRO | With XMatchPRO |
|---|---|---|
| Frequency | 200MHz | 200MHz |
| Register Use | 18% | 20% |
| LUT Use | 20% | 33% |
| Slice Use | 53% | 65% |
| I/O Use | 4% | 4% |
| BUFG Use | 6% | 6% |

## 7. CONCLUSION

This paper has presented the integration of a lossless data compression engine in the FPGA fabric of a hybrid ZYNQ chip. The close coupling of a compressor/decompressor engine with the processing functions running in the Cortex A9 processor means that the processor can use a simple API to request that data compressed in the main memory be decompressed and stored in the on-chip tightly coupled memories. Similarly uncompressed data in the on-chip memory can be compressed and moved to external DDR. In both cases there is a significant power advantage possible by maintaining external DDR memory in a power down state for longer. Our experiments used a DDR3 type memory to illustrate this advantage, and future work will include data with LPDDR type memories that have more effective power down states and stand to benefit more from the proposed technique. In the considered memory chips, the experimental results show, when running an application, up to 12.4% reduction of DDR power is achieved taking into account the additional power required in the PL side due to the presence of the compression/decompression engine. The level of power reduction is closely related to the compressibility of memory data, but other research [20] has shown that a 2 to 1 compression ratio is typical. In addition, when there is no application running, clock-gating the memory can achieve a 17.6% DDR power reduction as shown in Section 4. Future work will involve using the compression technology with real applications run on the Cortex A9 processors to investigate the power and performance benefits when both data and instructions are maintained in a compressed state in external DDR memories and will also considered LPDDR memory types with more sophisticated power modes.

## 8. ACKNOWLEDGEMENTS

We acknowledge with gratitude the support of EPSRC with the ENPOWER (Elastic and Non-Stationary POWER) project.

## 9. REFERENCES

[1] M. Ekman and P. Stenstrom, "A Robust Main-Memory Compression Scheme," Proc. of 32nd International Symposium on Computer Architecture, pp.74-85, June 2005.

[2] Ahmed A. Aqrawi and Anne C. Elster, "Bandwidth Reduction Through Multithreaded Compression of Seismic Images," IEEE International Symposium on Parallel and Distrubuted Processing, pp.1730-1739, May 2001.

[3] M. Kato and C-T. Lo, "Power Consumption Reduction in Java-enabled, Battery-powered Handheld Devices through Memory Compression," IEEE Inernational Symposium on Consumer Electronics, pp.1-6, June 2007.

[4] V. Beltran, Jordi Torres and E.Ayguade, "Improving Web Server Performance Through Main Memory Compression," 14th IEEE International Conference on Parallel and Distrubuted System, pp. 303-310, Dec 2008.

[5] http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf, "Zynq-7000 All Programmable SoC Technical Reference Manual," UG585 (v1.7), February 2014.

[6] http://www.xilinx.com/support/documentation/boards_and_kits/zc702_zvik/ug850-zc702-eval-bd.pdf, "ZC702 Evaluation Board for the Zynq-7000 XC7Z020 All Programmable SoC User Guide," UG850 (v1.3), June 2014.

[7] http://opencores.org/project.xmatchpro, XmatchPro Lossless Data Compressor project maintained by J.L.Nunez-Yanez.

[8] C.Lefurgy, P.Bird, I.Cheng and T.Mudge, "Code Density Using Compression Techniques," Proc. Of the 30th Annual International Symposium on MicroArchitecture, pp.194-203, December, 1997.

[9] H. Lekatsas, J.Henkel, and W.Wolf, "Code Compression for Low Power Embedded System Design," in Proc, Design Automation Conference, 2000, pp. 294-299.

[10] L.Benini, A.Macci, A.Nannarelli, “Cache-Code Compression for Energy Minimization in Embedded Processors,” ISLPED-01, pp.322-327, 2001.

[11] Y.Yoshida, B.Y.Song, H.Okuhata, T.Onoye, I.Shirakawa, “An Object Code Compression Approach to Embedded Processors,” ISLPED-97, pp.265-268, Augst, 1997.

[12] L.Benini, A.Macii, E.Macii, M.Poncino, “Selective Instruction Compression for Memory Energy Reduction in Embedded Systems,” IEEE/ACM Proc. of International Symposium on Low Power Electronics and Design, pp. 206-211, 1999.

[13] R.B.Tremaine *et al.* “IBM Memory Expansion Technology,” IBM Journal of Research and Development, vol.45, no.2, 2001.

[14] G.Pekhimenko, V.Seshadri, Y.Kim, H.Xin, O.Mutlu, M.A.Kozuch, P.B.Gibbons and T.C.Mowry, "Linearly Compressed Pages: A Low-Complexity, Low-Latency Main Memory Compression Framework," proc. of MICRO, 2013.

[15] J.Ziv and A.Lempel, "A Universal Algorithm for Sequential Data Compression," IEEE Transactions on Information Teory, Vol.23, No.3, pp.337-343, 1997.

[16] L. Benini *et al.* “Hardware-assissted Data Compression for Energy Minimization in Systems with Embedded Processors,” in Proc. Design, Automation&Test in Europ, 2002.

[17] A.Shafiee, M,Taassori, R.Balasubramonian and A.Davis, "MemZip: Exploring Unconventional Benefits from Memory Compression," in proc of 20th International Symposium on High Performance Computer Architecture, Feburary 2014.

[18] T.Cortes, Y.Becerra, and R.Cervera, “Swap Compression: Resurrecting Old Idears,” in Software-Practice and Experience Journal, no.30, pp.567-587, June 2000.

[19] F.Douglis, “The Compression Cache: Using online Compression to Extend Physical Memory,” in Proc. USENIX Conf., 1993.

[20] J.L.Nunez-Yanez, S.Jones, “Gbit/s Lossless Data Compression Hardware,” IEEE transactions on Very Large Scale Integration (VLSI) Systems, 2003, vol.11, issue.3, pp.93-98.

[21] “Low Power Double Data Rate 2 (LPDDR2),” JEDEC Solid State Techonology Associateion, 2010, retrieved 2010-12-30.