FPGA Based Synchronous Multi-Port SRAM Architecture for Motion Estimation

Purnachand Nalluri[1,2], Luis Nero Alves[1,2], Antonio Navarro[1,2]

[1]Institudo de Telecomunicações,
Pólo-Aveiro, Campus Universitário de Santiago,
3810-193 Aveiro, Portugal.
[2]Departamento de Electrónica, Telecomunicações e Informática,
Universidade de Aveiro
Campus Universitário de Santiago
3810-193 Aveiro, Portugal.
nalluri@av.it.pt, nero@ua.pt, navarro@ua.pt.

Abstract

Very often in signal and video processing applications, there is a strong demand for accessing the same memory location through multiple read ports. For video processing applications like Motion Estimation (ME), the same pixel, as part of the search window, is used in many calculations of SAD (Sum of Absolute Differences). In a design for such applications, there is a trade-off between number of effective gates used and the maximum operating frequency. Particularly, in FPGAs, the existing block RAMs do not support multiple port access and the replication of DRAM (Distributed RAM) leads to significant increase in the number of used CLBs (Configurable Logic Blocks). The present paper analyses different approaches that were previously used to solve this problem (same location reading) and proposes an effective solution by using an efficient combinational logic to synchronously and simultaneously read the video pixel memory data through multiple read-ports.

1. Introduction

With the increased demand for multi-tasking and parallel processing, the modern applications for FPGA and ASIC based memory architectures cannot rely just on single port or dual port memories. The only possible solution for this is to increase the bus bandwidth and implement multiple port memories. Especially when considering synchronous memories, implementing multiple read port memory operation is required [1-2]. There are many applications where multiple read operations are highly required. For example, in robotic vision system, the object recognition system has to search many samples of live video frames and output one object that has minimum error. In reconfigurable vision systems, a shared memory is necessary to access the video content through multiple resources [1-2]. Similarly, in video compressions system, a shared memory is required to process many samples of video frame blocks in one clock cycle. The present paper focuses on the key role of multi-port SRAM (Static RAM) in motion estimation [3] application. Section 2 explains the memory architecture requirement in motion estimation. Section 3 compares the performance of some architectural solutions. Section 4 proposes a new architecture suited for FPGAs. Section 5 details

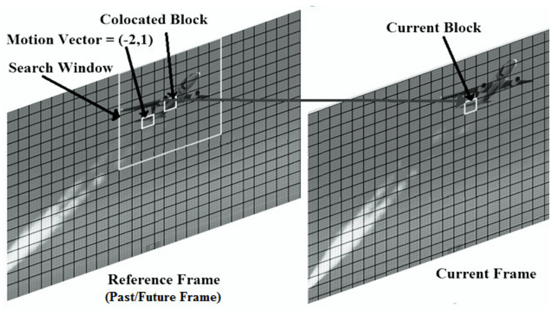

Fig. 1(a). Illustration of ME Process

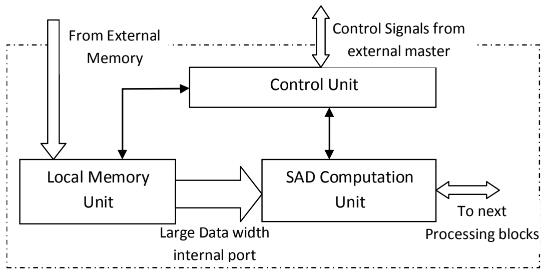
From External Memory
Control Signals from external master
Control Unit
Local Memory Unit
SAD Computation Unit
Large Data width internal port
To next Processing blocks

Fig. 1(b). Architecture for real time processing of Motion-Estimation

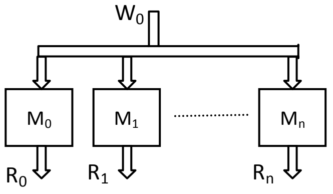

Fig. 2(a). Replicated Multiport Memory Architecture

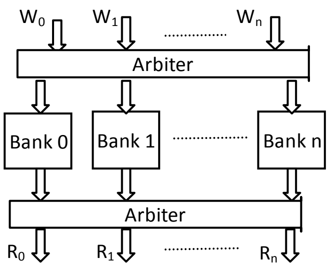

Fig. 2(b). Banked Multiport Memory Architecture

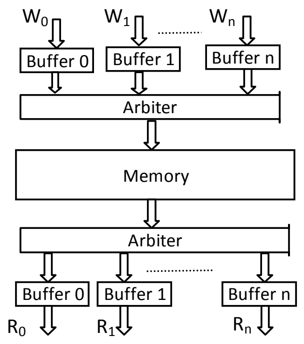

Fig. 2(c). Streaming Multiport Memory Architecture

the experimental results and followed by Section 6 with concluding remarks.

2. The Need for Multi-Read-Port Memory in Video Processing

For compressing video signal, a video encoder eliminates the temporal and spatial redundant information and encodes the error/residual information. For doing this, a video encoder typically uses predictive encoding techniques, so that residual information is further reduced. Motion estimation and motion compensation are the typical tools in a block-based video encoder that predicts and generates the video frames temporally. In a block based video encoder, each frame is divided into rectangular or square blocks of pixels, and hence prediction and other tools can be applied on each block. Especially when motion estimation is considered, the problem is very challenging since it is the most computationally expensive tool amongst all the tools in a video encoder.

Motion estimation is the process of searching the best matched block of a video frame in the past/future frame's ROI (Region of Interest, technically termed as search window) as shown in Fig.1(a) [3]. In order to find this best match, the ME algorithm uses matching criteria like SAD (Sum of Absolute Difference) or MSE (Mean Square Error) among others. For real-time applications, it is often required to apply parallel computation schemes. Instead of computing the SAD, block by block sequentially, computing a set of blocks in parallel makes the ME task faster. Hence in real-time, it is required to access many blocks of pixels data in parallel from local memory, and send them to the SAD estimator as shown in Fig.1 (b). It is here, where the major design challenges occur on how to implement the local memory without impairing overall performance.

3. Approaches for Implementing Multi-Read Port Memory

For writing the pixel block to local memory unit, the external architectural bus is constrained to either 32 or 64 bits. Hence, we do not need to optimize the writing procedure. Instead, a design will have a great necessity in implementing multiple reading procedures in this scenario. For implementing the read ports there are many possible approaches [4-6] as described in the following sections.

3.1 Memory Replication

Memory replication is the simplest way to implement multiple read ports. For reading the same memory location through 'n' multiple read ports, the memory is replicated 'n' times and the data is written to each of the write ports in parallel as shown in Fig.2(a). In case of FPGA BRAMs, for an application where one BRAM is required for one read port, the design has to replicate 'n' BRAMs for n read ports. The main disadvantage in this approach is the increase of system area and power.

3.2 Memory Banking

Memory banking technique divides memory capacity across smaller banks of physical memory. The totals read and write port widths are also divided equally among each bank. Multiple requests to the same memory banks are usually handled by an arbiter. This is somewhat similar to memory replication except that in each memory bank, the total memory is divided instead of replication, as shown in Fig.2 (b). However, the memory banking technique does not support multiple read operations, since in a given cycle only one read operation is possible. When the same memory bank is accessed by multiple sources, then each has to wait until its turn in arbiter is initiated.

3.3 Streaming Multiport Memory

In streaming multiport (also called stream-buffered multiport or multi-pumping) memory, the entire memory is provided with only one read port (and one write port), and each read requester is provided an internal register to hold the requested data as shown in Fig. 2(c). Multiple requests are arbitrated using an arbiter or a multiplexer, with a

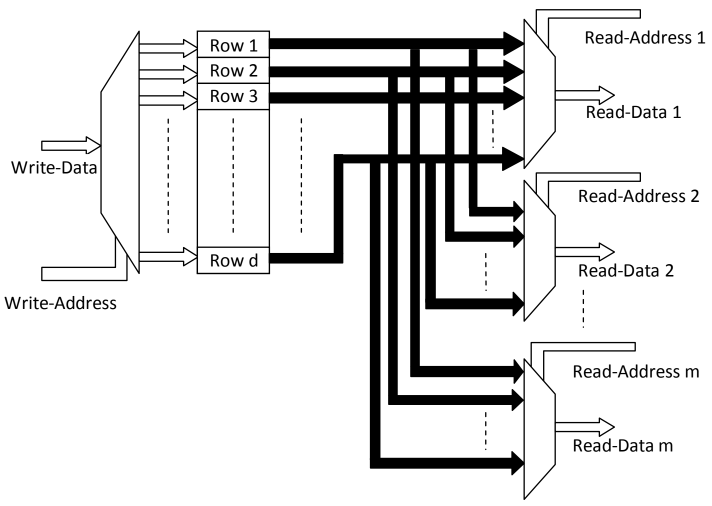

Fig. 3. Multiplexed Read Port (MRP) memory architecture for a single column of memory with 'd' rows and 'm' number of read ports

clock frequency that is multiple of external clock frequency and hence this scheme decreases the maximum external clock frequency and degrades system's performance as shown in Table 1 (Section 5).

3.4 Other Multiport Memory

Besides the aforementioned types, there are some other types of multi-porting techniques like bank-switching multiport memory, cached multi-port memory etc. In bank switching, an external latch is added to the memory controller, to select or switch between memory banks. In cache based multi-port memory, a local cache is provided for each read-port, and a cache coherence protocol is used to sync the cache contents. But in cache based porting, the read request may experience delay (variable delay) depending on the request sequence pattern and cache coherence protocol.

4. Proposed Technique

To handle multiple read transactions within the same clock cycle and without latching, each memory location can output data to all the read ports through a multiplexer circuit as shown in Fig.3. The select line for this multiplexer will be nothing but the read address port. Through this way the memory can have any number of read ports, independent of number of write ports.

As shown in Fig.3, the data is written to DRAM (Distributed RAM) column through one write port. Data is read through 'm' read-ports via multiplexer and each select line is the read address for that particular read operation. The advantage in this circuit is that all the address line can be selected in the same clock cycle and hence any number of memory locations can be read synchronously in one clock cycle. The second advantage is that the same memory location can be read through multiple reading ports and hence data can be shared with multiple ports without any extra cycles.

The memory architecture shown in Fig.3 is for one column of memory (or one column of block of pixels in a video frame), however, the proposed MRP (Multiplexed Read-Port) memory method can also be applied to an entire block of pixels in a video frame. Typically, the motion estimation block requires only luminance information, where each pixel is 8 bits wide for the luminance component. The write port is drawn from external memory which is typically constrained to 64 bits wide in modern FPGAs [7]. Hence each row could be 64 bits wide storing 8 pixels of information. Thus in one clock cycle, 8 pixels of memory is written, and the read operation will wait for the completion of the write operation.

Fig.4 shows the application of the proposed concept, for N pixel columns in a block of video frame. Data is written through one write port, and the corresponding memory column is selected using data and address select lines. At the output, the data is read through M read ports from each column. From all the NxM read ports, any number of read ports can be selected through the switch box. This is similar to memory banking, but the main difference is that here the designer can customize M, and can read multiple ports from each bank/column. But in memory banking, the read request is possible for only one memory location, in single cycle from each bank. Further, in the design shown in Fig.4, M need not be the same in each bank/column, and the designer has full flexibility to choose M, depending on the algorithm used. Usually, for H.264/AVC video standard, the maximum block size is 16x16 pixels and hence N will be equal to 2 (=16/8) in this case. For the latest video coding standard HEVC, the maximum block size is 64x64 and hence N will be

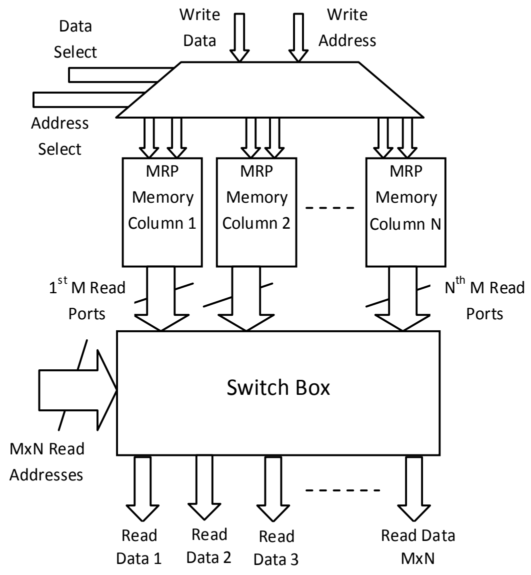

Fig. 4. Multiplexed Read Port (MRP) memory architecture for 'N' columns of memory with 'd' rows in each column and 'M' number of read ports in each column.

equal to 8 (=64/8) for this standard. Similarly, the memory depth 'd' is equal to 16 and 64 for H.264/AVC and HEVC, respectively.

5. FPGA Synthesis Results

The proposed MRP method was implemented in verilog and synthesised using Virtex-5 FPGA [7]. The row width for each memory location is chosen as 64, since the external bus is also configured to 64, and thus able to write 8 pixels of data in one clock cycle. The memory depth 'd' is configured as 16, and the memory columns 'N' is chosen as 2, which is suitable for a 16x16 block size pixels. For evaluation, 'M' is chosen as 5 implying that 5 concurrent read operations are possible from each memory column.

The proposed method is compared with the other methods – memory replication method, memory banking method and streamed memory buffer method. Table 1 shows the comparison results. The results show that MRP method has highest maximum operating frequency, with slight increase in number of CLBs (LUT-FF pairs) used in FPGA. The memory replication method has the highest resource usage, due to increase in the number of slice registers. The stream buffered method has the lowest maximum operating frequency due to the combinational logic of arbiter switching used. The memory banking method has a better resource usage and maximum operating frequency, but the very purpose of multi-porting is not solved with memory banking, since each memory bank can be simultaneously read only once.

	Memory Replication	**Memory Banking**	**Stream Buffer**	**MRP**
#Slice LUTs	4573 (6%)	2154 (3%)	2427 (3%)	4543 (6%)
#Slice Registers	12672 (18%)	4096 (5%)	5260 (7%)	4480 (6%)
#LUT-FF pairs	16121 (23%)	5126 (7%)	6301 (9%)	7899 (11%)
Max.Frequency (MHz)	329.96	331.71	273.09	335.21

Table 1. Virtex-5 FPGA synthesis results for different memory architectures.

6. Conclusions

This paper presented a multiple port read memory architecture. The propose architecture was synthesized using VIRTEX-5 FPGAs. The synthesis results show that the proposed MRP (Multiplexed Read Port) memory architecture maintain a good balance between number of FPGA resources used and maximum frequency. This architecture can be used for video processing applications like motion estimation, intra prediction or in other applications where simultaneous read operation from a common shared memory is required.

References

[1] J.M. Perez, P. Sanchez, M. Martinez, "High memory throughput FPGA architecture for high-definition Belief-Propagation stereo matching", IEEE Int. Conf on Signals Cir & Sys, pp. 1-6, Nov. 2009.

[2] S. Chang, B.S. Kim, L.S. Kim, "A Programmable 3.2-GOPS Merged DRAM Logic for Video Signal Processing", IEEE Trans. on Circuits and Systems for Video Technology, vol.10, No.6, Sept. 2000.

[3] N. Purnachand, L.N. Alves, A. Navarro, "Fast Motion Estimation Algorithm for HEVC", IEEE ICCE-Berlin Conf 2012, Berlin, pp. 34-37, Sept 2012.

[4] H. Zhao, H. Sang,T. Zhang, Y. Fan, "GEMI: A High Performance and High Flexibility Memory Interface Architecture for Complex Embedded SOC", IEEE CSSE Conf 2008, pp. 62-65, Dec 2008.

[5] W. Ji, F. Shi, B. Qiao, H. Song, "Multi-port Memory Design Methodology Based on Block Read and Write", IEEE ICCA Conf 2007, May 2007.

[6] M. Saghir, R. Naous, "A Configurable Multi-ported Register File Architecture for Soft Processor Cores", Int. Workshop on Appl. Reconfig. Computing, Springer-Verlag - pp. 14-27, March 2007.

[7] Xilinx Virtex-5 User Guide ver. 5.4, March 2012. "http://www.xilinx.com/support/documentation/user_guides/ug190.pdf",