International Journal For Technological Research In Engineering
Volume 4, Issue 7, March-2017 *ISSN (Online): 2347 - 4718*

DESIGN MULTI-PORT MEMORY FOR FPGA VIA BIST

Trupti Rangari[1], Dr. Ashok Chandak[2]
[1]P.G.Student, [2]HOD
[1,2]Department of Electronics & Telecommunication Engineering
[1]Dhole Patil College of Engineering, Pune, India, [2]Cusrow Wadia College of Engineering, Pune, India

Abstract: Now a day's many applications need multi-ported memories to increase the process speed and allow write and read memory banks from multiple ports at the same time. Most FPGAs only support dual port RAM, and we must build by ourselves multi-port RAM from available logic elements and RAM building blocks of FPGAs when the number of port exceeds 2. In this paper, the conventional approaches for implementing multi-ported memory with BIST. Which help to reduce the complexity, and thereby decrease the cost and reduce reliance upon external (pattern-programmed) test equipment. BIST reduces cost. The area detail is also given to help the designer defining which blocks affect significantly to the total area of design, and to help designer deciding which blocks should be optimized. This work also helps the multi-ported memory designer choosing which approach is suitable for the specific application. There is no need to perform wait operation for conventional multi port operation.
Keywords: Xilinx, Multi-ported memory, FPGA, Live Value Table, BIST

I. INTRODUCTION

Multi-ported memory is broadly used in modern designs on FPGAs. However, the excessive demand on BRAMs to implement multi-ported memory on FPGA would block the usage of BRAMs for other parts of a design. This issue becomes a serious concern especially for designs that require huge internal storage capacity. This paper proposes a BRAM efficient scheme on increasing read ports and write ports. When compared with previous works, the proposed multi-ported memory can reduce requirement on BRAMs with only minor frequency degradation. Field programmable gate arrays have been broadly used in fast prototyping of complex digital systems. FPGAs contain programmable logic arrays, usually referred to as slices. Slices can be configured into different logic functions. The flexible routing channels can support data transferring between logic slices. In addition to implementing logic operations, if needed, the slices can also be used as storage elements, such as flip-flops, register files, or other memory modules. Due to the increasing complexity of digital systems, there is a growing demand for in-system memory modules. Synthesizing a large number of memory modules would consume a significant amount of slices, and would therefore result in an inefficient design. The excessive usage of slices could also pose a limiting factor to the maximum size of a system that can be prototyped on an FPGA. To more efficiently support the in-system memory, modern FPGAs deploy block RAMs (BRAMs) that are hardcore memory blocks integrated within an FPGA to support efficient memory usage in a design. Compared with the storage module synthesized by slices, BRAMs are more area and power efficient while at the same time achieving higher operating frequencies.

II. LITERATURE REVIEW

Implementations for FPGA-based multi-ported memories have only recently been formally described and studied; the conventional approaches here. A straightforward approach is to construct a multi-ported memory using logic elements—for example Altera's adaptive logic modules (ALMs)—enjoying flexibility but at a heavy cost in area and performance. Replication enables constructing a memory with any number of external read ports, but can support only a single external write port that must be connected to one of the two ports of each replicated BRAM. Banking divides the read and write ports across multiple separate BRAMs, supporting concurrent read and writes but fragmenting and isolating the data across banks. The Live Value Table (LVT) approach ugments a banked approach with a table that uses output multiplexers to steer reads to the most recently-updated bank for each memory address. The LVT approach improves significantly on the area and speed of comparable designs built using ALMs, although the internal LVT table itself scales somewhat poorly, can consume a lot of area, and usually becomes the critical path. Finally, Multipumping can be applied to any memory design to multiply its read and write ports by operating that memory at a multiple of the external clock frequency. Multipumping reduces the area required for a memory with a certain number of ports, but also reduces its maximum achievable external operating frequency. To implement a multi ported memory on an FPGA, two types of design techniques are required, namely increasing read ports and increasing write ports. Table I lists the techniques proposed by previous works for multi ported memories on FPGAs. The approach of replication enables multiple read ports by replicating the data on multiple BRAMs. This technique uses low complexity of control logic, but requires excessive usage of BRAMs. LVT, which is implemented by synthesizing slices on FPGA, enables multiple write ports by duplicating BRAMs and tracking which BRAM stores the latest value of an address. The other approach to increase write ports is referred to as XOR-based. Different from LVT, which uses a table to track the location of the latest value, the XOR-based design duplicates BRAMs and encodes the stored data with XOR operations. The target data can be retrieved by applying the XOR again. In general, the XOR-based approach can achieve a higher operating frequency, but requires more BRAMs than the LVT

approach. Note that this paper focuses on architectural solutions to achieve multiple accesses for a general memory that takes requests at the current cycle and returns results in the next cycle. Users of the multiported memory can be completely ignorant of the details of memory designs. There are other works focusing on enabling multiple accesses for specific types of storage elements, such as register files . They enable concurrent reads with an approach similar to replication, but avoid write conflicts by renaming the registers with software approaches, such as compiler or assembler. These approaches, which tackle specific storage functions and involve effort of users, are not in the scope of this paper. The following sections will provide more in-depth discussions about implementations and design concerns of these techniques. To facilitate a more general discussion, the following paragraphs use a memory bank to refer to a standalone memory module used as a building block to implement a memory system. A memory system usually consists of multiple banks. The memory space, also referred to as memory depth, is distributed across the banks. When designing a memory system on FPGAs, a BRAM can be used to support the complete memory space. BRAMs can also be deployed as banks to enable larger memory space or higher access bandwidth.

III. METHODOLOGY

Efficient design of multi ported memory using HBDX (Hierarchical Bank Division with XOR) and BDRT (Bank Division with Remap Table). This project first introduces a brand new perspective and a more efficient way of using a conventional two reads one write (2R1W) memory as a 2R1W/4R memory. By exploiting the 2R1W/4R as the building block, this project introduces a hierarchical design of 4R1W memory that requires fewer BRAMs than the previous approach of duplicating the 2R1W module. Memories with more read/write ports can be extended from the proposed 2R1W/4R memory and the hierarchical 4R1W memory. Compared with previous xor-based and live value table-based approaches, the proposed designs can, respectively, reduce the BRAM usage for 4R2W memory designs with 8K-depth. A brand new perspective and a more efficient way of using a conventional two reads one write (2R1W) memory as a 2R1W/4R memory is introduced. A new architecture hierarchical bank division with XOR (HBDX) is introduced to increase read ports. A new architecture bank division with remap table (BDRT) is introduced to increase write ports. HBDX and BDRT are integrated to design a multi ported memory.

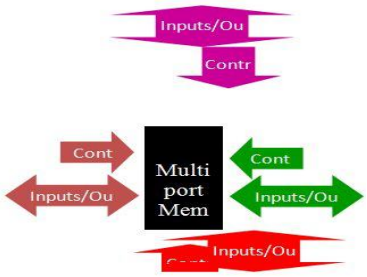

Fig.1 Block Diagram Design Multi-Port Memory for FPGA Via BIST

Here Bolck Diagram of Design Multi-Port Memory For FPGA Via BIST implies multiport input ouput operation using different techniques like technique used for write/read/read-write with achievements. In these paper rmain approach is to achieve multiple operations with BIST analysis . The integrated multi ported memory design can provide higher throughput than the time-multiplexing (TMX) based designs. The throughput of different designs for 4R2W memory. XOR-based (4R2W) is the XOR-based 4R2W design proposed in this project. TMX(2R1W) uses the replication-based 2R1W design and applies the time multiplexing scheme to achieve 4R2W. Here we assume the time-multiplexing scheme induces zero latency overhead and can run at the same clock frequency as the original 2R1W design. Therefore, for TMX(2R1W), it takes two cycles to serve four reads and two writes. The XOR-based designs run at slower clock rates than the TMX(2R1W) designs. However, XOR-based designs can achieve higher total throughput than the time-multiplexing designs. To attain the most benefit, users need to properly choose between designs that support multiple reads with and without mapping tables. As demonstrated previously, multiport designs with mapping tables can more efficiently utilize the BRAMs, but would suffer from lower operating frequencies due to more complex routing. The timing issue is further aggravated when the size and complexity of the multiport design approaches the capacity of the target FPGA. Therefore, users would prefer designs with mapping tables when the target FPGA contains abundant slices and relatively scarce BRAMs. If the number of slices becomes a design constraint, users should favor designs without mapping tables in order to avoid the possible performance degradation due to insufficient slices and congested connections. A proper bank-organization can enable further performance enhancement. A different bank organizations results in disparate performance. Refining the bank organization would attain performance enhancement and worth further exploration. The current designs still require table-lookups to support multiple writes. A design with mapping tables would limit the maximum operating frequency. We are currently developing a technique to support multiple writes without mapping tables. This technique would avoid the issue of congested routing and potentially enhance the overall performance of the multiport designs.

IV. CONCLUSION

XOR-based and LVT-based approaches, the proposed designs can reduce BRAM usage for 4R2W memory designs with 8K-depth. For complex multi ported designs, the proposed BRAM-efficient approaches can achieve higher clock frequencies by alleviating the complex routing in an FPGA. For 4R3W memory with8K-depth, the proposed design can save no. of BRAMs while at the same time enhance the operating frequency. This report also demonstrates the importance of applying an appropriate bank organization in a memory design. It is shown that a multi ported design with proper bank organization could achieve a BRAM reduction, a higher frequency, and lower slice

utilization. The results present great potential of future design refinement that could be achieved by optimizing the bank organizations.

REFERANCES

[1] Xilinx. 7 Series FPGAs Configurable Logic Block User Guide, accessed on May 30, 2016. [Online].Available:http://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

[2] Xilinx. Zynq-7000 All Programmable SoC Overview, accessed on May 30, 2016. [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[3] J.-L. Lin and B.-C. C. Lai, "BRAM efficient multi-ported memory onFPGA," in Proc. Int. Symp. VLSI Design, Autom. Test (VLSI-DAT),Apr. 2015, pp. 1–4.

[4] G. A. Malazgirt, H. E. Yantir, A. Yurdakul, and S. Niar, "Application specific multi-port memory customization in FPGAs," in Proc. IEEE Int.Conf. Field Program. Logic Appl. (FPL), Sep. 2014, pp. 1–4.

[5] C. E. Laforest, Z. Li, T. O'Rourke, M. G. Liu, and J. G. Steffan, "Composing multi-ported memories on FPGAs," ACM Trans. Reconfigurable Technol. Syst., vol. 7, no. 3, Aug. 2014, Art. no. 16.

[6] H. E. Yantir and A. Yurdakul, "An efficient heterogeneous register file implementation for FPGAs," in Proc. IEEE Int. Parallel Distrib. Process.Symp. Workshops (IPDPSW), May 2014, pp. 293–298.

[7] H. E. Yantir, S. Bayar, and A. Yurdakul, "Efficient implementations of multi-pumped multi-port register files in FPGAs," in Proc. Euromicro Conf. Digit. Syst. Design (DSD), Sep. 2013, pp. 185–192.

[8] C. E. LaForest, M. G. Liu, E. Rapati, and J. G. Steffan, "Multi-ported memories for FPGAs via XOR," in Proc. 20th Annu. ACM/SIGDA Int.Symp.Field Program. Gate Arrays (FPGA), 2012, pp. 209–218

[9] C. E. LaForest and J. G. Steffan, "Efficient multi-ported memories for FPGAs," in Proc. 18th Annu. ACM/SIGDA Int. Symp.Field Program.Gate Arrays, 2010, pp. 41–50.

