ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

BRAM-based Multiported Memory Designs on FPGA

M.Supriya[1], Dr.N.Rajesh Kumar[2]

PG Scholar, Dept of VLSI Design, KPR Institute of Engineering and Technology, Coimbatore, India[1]

Associate Professor, Dept of ECE, KPR Institute of Engineering and Technology, Coimbatore, India[2]

Abstract: FPGAs offer an attractive platform to build multi-ported memories. Multi-ported memories are used in modern designs on FPGAs (Field Programmable Gate Arrays). Block RAMs (BRAMs) are broadly used for multi-ported memory designs on FPGA. This paper first introduces 2R1W memory as 2R1W/4R memory; hence 4R/1W requires fewer BRAMs than 2R/1W. Compared with existing technique like Hierarchical Bank division with XOR design (HBDX) and Bank division with remap table (BDRT) with 2R1W/4R the proposed technique combines HBDX and BDRT and uses 2R1W/4R as 8R1W with 8K depth; as the memory capacity increases. For complex Multi-ported designs, the proposed BRAM approach can achieve higher clock frequencies. For 8R/1W the design requires fewer BRAMs compared with 4R/1W.

Index Terms: Block RAM (BRAM), FPGA (Field Programmable Gate Arrays), Hierarchical Bank division with XOR design (HBDX) and Bank division with remap table (BDRT), Clock frequency.

INTRODUCTION

FPGAs (Field Programmable Gate Array) have been used in fast prototyping and FPGAs contain programmable arrays, usually referred to as slices [1] which can be configured into different logic functions. As FPGAs increase in size, designers use them to build larger systems-on-chip that require frequent data sharing, communication, and synchronization among distributed functional units. FPGAs are adopted for various design purposes [2]. FPGAs usually implement multiple BRAMs with the same specifications. Multi-ported memory, which allows multiple concurrent, reads and writes, is used in various designs. Any design that requires a memory with more than two ports must be built out of logic elements or by combining multiple blocks RAMs. The challenge of constructing a multi-ported memory out of FPGA logic elements is inefficient [3]. Multi-port memory is widely used as the shared memory in multi-processor systems. Although the current FPGA design tools can automatically synthesize the multi-ported memory by configuring slices, it has demonstrated to be considerably inefficient in terms of utilization of slices. The increasing depth also becomes a design limit to the maximum operating frequency. The fixed specification of BRAMs requires extra effort if designers like implement a storage module that requires more ports then the existing BRAMs. When compared with the designs that only utilize slices, the approaches with BRAMs have demonstrated less total equivalent area while achieving higher frequencies. Occupying too many BRAMs for multi-ported could seriously block the usage of BRAMs for other parts of a design. Multi-ported memory is broadly used in modern digital designs.

RELATED WORKS

Previous researches have proposed several approaches to enable multi-ported memories on FPGA by using the two-port BRAM. The design methods can be split into two parts, increasing the read ports and increasing the write ports. There are other works focusing on enabling multiple accesses for specific types of storage elements, such as register files [7]–[9].

Replication [4] is one of the most common techniques used to increase the read ports. 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. However, this technique alone cannot support more than one write port since the single common external write port must be routed to each block RAM, using up its second port, to keep it up-to-date. As shown in fig 1(a), the data in memory M_0 is replicated to other banks (M_1 to M_{m-1}) in order to support multiple concurrent reads R_0 to $R_{m-1.}$ The main advantage of replication is its simplicity without requiring additional control logic; however, it needs m times the number of memory modules, where m is the number of read ports that would be supported by the multi-ported memory design.

Live Value Table (LVT) is proposed by [4] to support multiple write ports. Essentially, the LVT allows a banked design to behave like a true multi-ported design by directing reads to appropriate banks based on which bank

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

holds the most recent or live write value. An LVT based design also leverages block RAMS, which implement memory more efficiently, and has an operating frequency closer to that of the block RAMs themselves. Fig. 1(b) shows an example of multi-ported design that enables n concurrent write requests. The memory is replicated n times into banks M_0 - M_{m-1}. Each bank supports one write port, since multiple writes updates different memory addresses, an additional block Live Value Table is implemented.

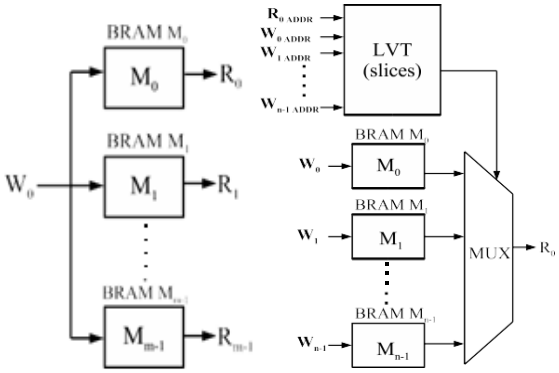

Fig. 1(a) Replication. (b) LVT Technique.

Combination of Replication and LVT can design a multi-ported that supports m reads and n writes. Fig. 2 shows the design which requires a total number of m*n memory modules. The design combines the techniques of replication and LVT to support two reads (R0 and R1) and two writes (W0 and W1). The multiple writes are handled by the LVT, while the multiple reads can be serviced by the replicated BRAMs.

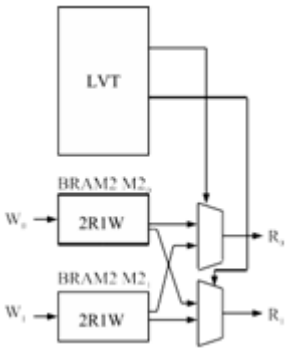

Fig. 2 Combination of replication & LVT Technique.

An **XOR-based approach for 2W/1R** proposed in **[**5**]** is a way to increase write ports. Fig. 3 illustrates an XOR based memory design that support two simultaneous writes W_0 and W_1 and one read R_0. The XOR based design encodes the stored data by using XOR operations. The XOR based multi-ported memory can achieve higher frequency by eliminating the logic path from LVT to output multiplexors. However, it also adopts the replication method to increase read ports. A challenge for the XOR design is that each write requires reading as well, since the write value XOR the old value of that location from the other bank. Hence the design requires a column of BRAMs that is as wide as one less than the number of write ports, to provide sufficient internal read ports to support writing.

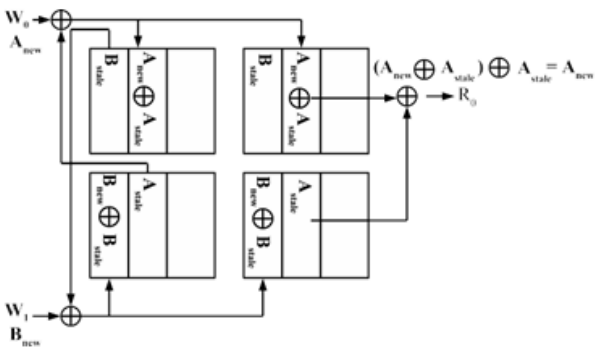

Fig. 3 XOR based memory design that can support 2W/1R.

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

Combination of Replication and XOR-based approach support two reads and two writes. Fig. 4 shows the data flow of 2R/2W. R0 reads both the values A_{stale} and $A \oplus A_{stale}$ from address 2 at two BRAM2 modules, and recovers A by XOR-ing these two values. The operation is given by equation (1) and (2).

$$R_0 = (A \oplus A_{stale}) \oplus A_{stale} = A \quad (1)$$

$$R_1 = (B \oplus B_{stale}) \oplus B_{stale} = B \quad (2)$$

W_0 reads value C_{stale} from three BRAMs at the bottom of Fig. 4, and updates the encoded value. Therefore the design in Fig. 4 needs six BRAM2 modules to provide sufficient internal read ports and support all the data accesses.

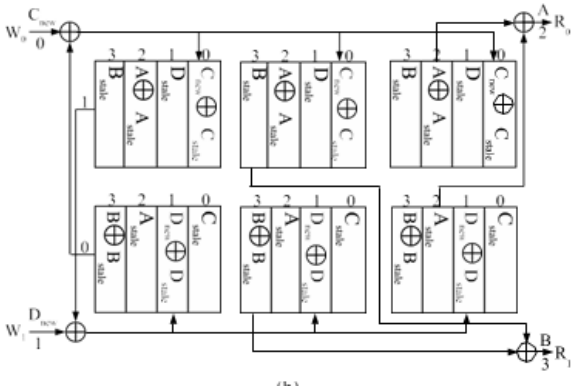

Fig. 4 Combination of Replication and XOR-based approach.

EXISTING APPROACHES

Bank division with XOR (BDX) is an approach to increase read ports proposed in [6]. Unlike the method used in [3], BDX avoids replicating the storage elements of the whole memory space.

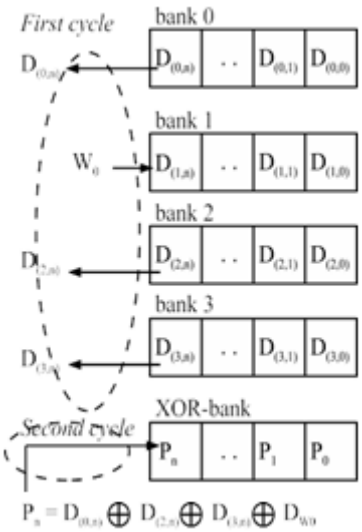

Fig. 5 BDX Technique that supports a write request in two-cycle pipeline architecture.

The XOR-Based approach in [5] uses XOR operations to increase write ports by storing the data coherence between memory modules. Fig. 5 shows an example of a 2R1W memory implemented with BDX approach. At the first cycle W_0 writes the data $D_{(1,n)}$ are read and XOR-ed. At the second cycle, the XOR-ed value will be written back to the XOR bank at offset n. The 2R1W memory introduced previously only needs an additional XOR-bank that requires the same storage size of each data bank. In this case, assume all the data banks are of equal size. This design can process one write request every cycle.

Hierarchical Bank Division with XOR (HBDX) approach is a new architecture to increase read ports. This approach applies a new perspective of using a 2R1W modules as a 2R1W/4R module.

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

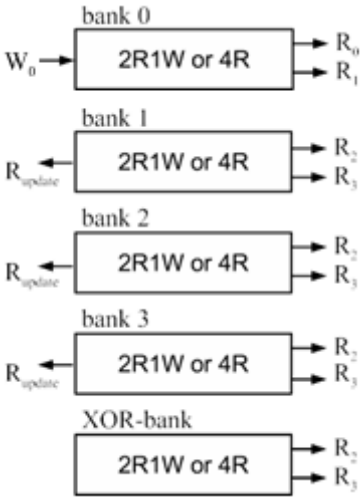

Fig. 6 HBDX 4R1W implemented with 2R1W/4R modules.

Fig. 6 illustrates a 4R1W memory design by using the HBDX scheme. The HBDX will utilize this versatile usage mode to achieve a more efficient 4R1W design.

Integration of HBDX and Bank Division with Remap Table (BDRT) uses HBDX and BDRT to implement an mRnW memory. Fig. 7 shows an implementation of mRnW memory. This memory architecture is divided into k data banks. Based on BDRT, to support all the writes, there reqire total n-1 bank buffers. A hash mechanism is added to distribute the writes to banks.

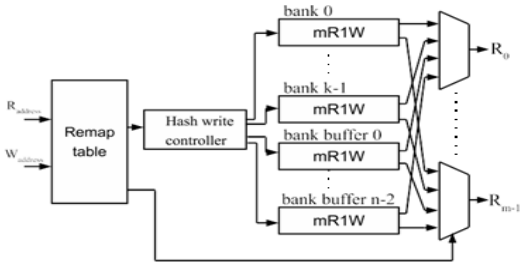

Fig. 7 Architecture that integrates HBDX and BDRT .

Integration of HBDX and BDRT for 2R1W/4R approach shown in Fig. 8 is used to implement memory with k data banks and using 2R1W/4R as the building block. The 4R can be supported by exploiting the 4R mode of the 2R1W/4R module. 2R1W/4R module cannot support any write requests when it is servicing more than two reads.

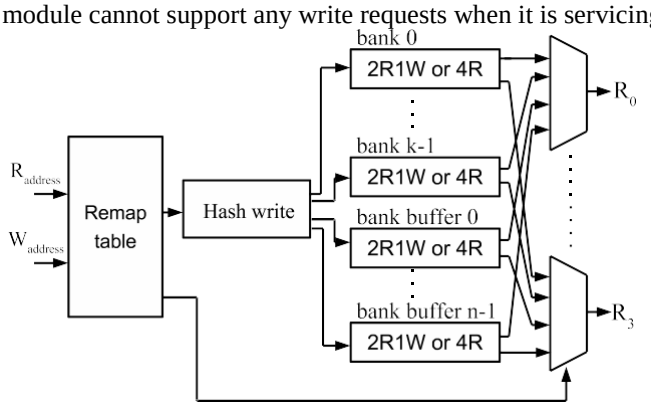

Fig. 8 integration of HBDX and BDRT that applies 2R1W/4R modules as building blocks.

PROPOSED DESIGN

For efficient usage of clock, multi-pumping is introduced shown in Fig. 9. 8 bit comparator is used which compares the two input signal and provide one output signal. Multi-pumping can bring about a useful reduction in area if the speed of the original memory is significantly higher than required by the surrounding system. The LVT approach

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 11, November 2017

can also support the implementation of multi-ported memories having bidirectional ports. The main advantage of this method is minimization of area and simultaneous write and read can be performed comparing the existing approaches.

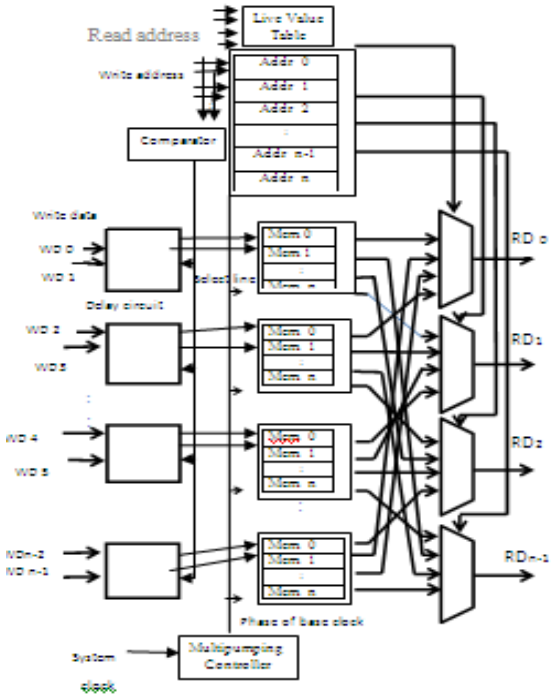

Fig. 9 Combination of HBDX and BDRT for 8R

CONCLUSION

This paper proposes efficient BRAM-based multi-ported memory designs on FPGAs. The existing approaches require significant amounts of BRAMs to implement a memory that supports multiple read and write ports. This paper proposes technique that can attain efficient multi-ported memory designs. Compared with previous approaches, the proposed design can respectively reduce the area. For complex multi-ported designs, the proposed BRAM-efficient approaches can achieve higher clock frequencies by alleviating the complex routing in an FPGA.

REFERENCES

[1] Xilinx 7 series FPGAs Configurable Logic BlockUserGuide, http://www.xilinx.com/support/ documentation/user_guides/ug474-7series-CLB.pdf.
[2] R. Sindhu and V.K. Prasanna, "Fast regular expression matching using FPGAs," in proc. 9th Annu. IEEE Symp. Custom Comput. Match. (FCCM), Mar. 2001, pp. 227-238.
[3] C.E. LaFaorest and J.G.Steffan. "Efficient Multi-ported memories for FPGAs". In Proceedings of the 18th annual ACM/SIGDA international symposium on Field Programmable gate arrays,FPGA 10, pages 41-50, New York,NY, USAA, 2010. ACM.
[4] Charles Eric, LaForest, Zimo Li, Tristan O'rourke, Ming G.Liu, and J.Gregory Steffan, "Composing multi-ported memories on FPGAs," in proceedings of the ACM treansactions on Reconfigurable Technology and systems (TRETS), vol.7, no.3, 2014.
[5] 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
[6] J.L. Lin and B.C. Lai, "BRAM efficient multi-ported on FPGA," in porc. Int. symp. VLSI Design, Autom. test (VLSI.DAT) Apr.2015, pp. 1-4.
[7] G.A. Malazgirt, H.E. Yantir, A. Yurdakul, and S. Niar, "Application specific multi-ported memory customization in FPGAs," in proc. IEEE Int. Conf. Field Program. Logic Appl. (FPL),Sep. 2014,pp.1-4
[8] 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.
[9] H. E. Yantir, S. Bayar, and A. Yurdakul, "Efficient implementations of multi-pumped multi-port register files in FPGAs," in Proc. EuromicrConf. Digit. Syst. Design (DSD), Sep. 2013, pp. 185–192.