Design Of Multiport Memory For Consumption Of Less Energy

Chinna Thambi[a], Durai[b], Pattamuthu[c], Vivek Anand[d]

[a,b,c]Student, Electronics and Communication Engineering Department, National Engineering College, K. R. Nagar, Kovilpatti - 628503
[d]Assistant Professor, Electronics and Communication Engineering Department, National Engineering College, K. R. Nagar, Kovilpatti -628503

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 23 May 2021

Abstract: The utilization of Block RAMs (BRAMs) makes a critical performance factor for multiport memory excessive demand on block of BRAMs. The BRAMs usage from other parts of a design limits the operatind frequency, as does the dynamic routing between BRAMs and logic. A more effective way of using a standard two reads one write (2R1W) memory as a 2R1W/4R memory has been introduced with the introduction of a brand new perspective of BRAM. Exploitation of the 2R1W/4R as the building block, gives introduction to 4R1W memory hierarchical design that requires 25% fewer BRAMs when compared to the previous approach of 2R1W module duplication. With the 2R1W/4R memory and the hierarchical 4R1W memory, more read/write ports can be added. The efficient approches of BRAM can achieve higher clock frequencies for complex multiport designs by reducing the complexity of routing in an FPGA. From the proposed 2R1W/4R memory to the hierarchical 4R1W memory, more read/write ports memories have the capacity to expand. In this work, design of multiport memory are carried out using the register bank and the communication are sent to the processor. The results are once again stored back to the multiport memory. The simulation results are observed using the model sim software

1. Introduction

Data can be moved between logic slices using the versatile routing channels. With the addition of executing logic operations, the slices can be used as storage elements such as register files, flip-flops, or other memory modules. The market for in-system memory modules is rising as digital systems become more complex. A large number of slices will be required to synthesise a large number of memory modules, resulting in an inefficient design. BRAMs are more costly than the storage module created by slices. Although achieving higher operating frequencies, the device is both space and power efficient. In certain instances, an FPGA is used. There are many BRAMs of the same specification. Multiported memories, which allow multiple concurrent reads and writes, are commonly used in various digital designs on FPGAs to achieve high memory bandwidth. The one write port and two read ports. is needed for the register file of an FPGA-based scalar MIPS-like soft processor needs [3]. More access ports are needed by multiple-instruction processors. The multiple

concurrent accesses is allowed by the shared cache system between multiple soft processors on FPGA . A routing table in a network switching function will need to allow multiple accesses to support multiple requests from different ingress ports. Alternative ways to accommodate multiple accesses

include time multiplexing and task scheduling. However, these schemes typically result in more complicated designs with numerous corner cases, necessitating additional verification effort. The design of a system that requires concurrent data accesses can be significantly simplified by the generic multiported memory module.

Increasing logic depth limits the maximum operating frequency. Designers would have to go above and beyond the basic specification if they want to provide a storage module that supports more concurrent access ports than the existing BRAMs. As compared to multiported memory designs that only use slices, less total equivalent area when achieving higher frequencies have been shown by previous methods with BRAMs.

Due to the lack of BRAMs, designers would have to synthesise storage modules with slices, which would consume a large number of slices while also limiting the design's total operating frequency.

The main contributions of the paper are listed below. The first section of this paper describes a novel method for using a 2R1W module as either a 2R1W or a 4R module, known as a 2R1W/4R memory. Second, this paper proposes a hierarchical 4R1W memory architecture that takes advantage of the versatile user mode of the proposed 2R1W/4R memory. This hierarchical 4R1W architecture consumes 25% less BRAMs than the [9] solution, which duplicates the 2R1W module.

When compared to XOR-based and live value table (LVT)-based designs from previous methods, the proposed designs would minimize BRAM consumption by up to 53 percent and 69 percent, respectively, for 4R2W memory designs with 8K-depth. The higher clock frequencies for complex multiported designs by reducing the difficulty of routing in an FPGA could be achieved by the proposed BRAM-efficient approaches

2. Related Works

The implementation of multiport memory makes the design technique requirement into two types: increasing read ports and the write ports. By replicating data through multiple BRAMs, replication , allows for multiple read ports. While this method uses low-complexity control logic, it does necessitate the use of a large number of BRAMs.

Synthesised slices on FPGA implemented LVT allows multiple write ports by duplicating BRAMs and monitoring which BRAM stores the most recent value of an address. The other choice is to use an XOR-based [2] method to increase write ports. It focuses on architectural solutions for achievement of multiple accesses for a general memory that accepts requests one time and returns results the next. From multiported memory users, memory design details can be completely hidden. Enabling multiple accesses to different types of storage components is focused by other research, such as register files. They allow concurrent reads using a method similar to replication, but write conflicts by renaming the registers with software like a compiler or an assembler is avoided by them. The sections that follow will go into application of this method and design problems in greater depth. A more general debate is faciliated by the following paragraphs refer to a standalone memory module used as a building block for the integration of memory unit as a memory bank. A memory card is usually made up of many banks. Banks share the memory depth or memory space. When memory system is build on FPGAs, the entire memory space is supported by using BRAM. In banks, BRAMs may be used to increase memory space or access bandwidth. Replication is a popular technique used to increase read ports. By replicating the data to many BRAMs , this technique allows several read ports to be accessible. When there are multiple reads, in order to reach the target data without interfering with the other reads each one will be directed to a different BRAM. When a write to an address occurs, the write must be forwarded to each BRAM, and the data must be modified to the address in each BRAM that corresponds to the write.

The main advantage of replication is straightforward and no need of complex control logic; however, it must replicate m times the number of memory modules provided by the target multiported memory design, where m is the target multiported memory design's number of read ports. This technique allows multiple concurrent writes by making use of multiple repeated BRAMs.

Multiple read and write ports must be available at the same time in a general multiported memory. As a result, the previous sections' read and write techniques should be integrated into a design to handle multiple read and write ports. This section shows how these strategies can be combined to construct a multiported memory architecture using real-world examples.

3. Proposed Design

Unlike the replication methods mentioned , the approach proposed in this paper allows for multiple reads using XOR operations and multiple writes using additional BRAMs. This paper proposes a novel method for using a 2R1W module as a 2R1W or 4R module, known as a 2R1W/4R memory. This paper proposes a hierarchical XOR-based 4R1W memory architecture that uses less BRAMs than previous designs by using the versatile usage mode of the 2R1W/4R.

A. Methods for Increasing Read Ports:

To increase read ports it proposes a strategy called Bank Division with XOR. Unlike the methods used before, BDX avoids replication of the storage elements of the entire memory space. with the usage of XOR operation multiple reads can be provided in BDX It's important to note that BDX is not the same as the XOR-based architecture described before which stores encoded data to maintain data coherence between memory modules and uses XOR operations to increase write ports. BDX uses XOR operations to increase read ports by retrieving the target data from the encoded value.

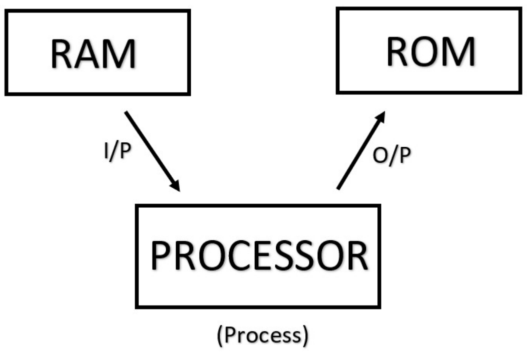
B. Techniques for Increasing the Number of Write Ports:

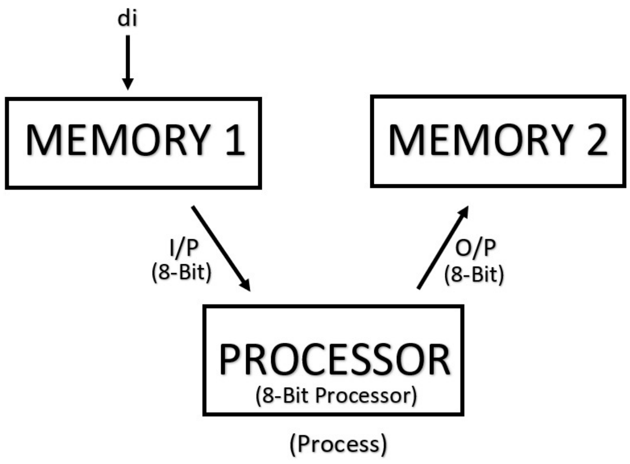
Unlike the LVT designs used, Bank Division with Remap Table enables multiple writes with the usage of additional BRAMs and a remap table to keep track of where the most recent data is stored, rather than replicating the entire memory space. The following equation gives the number of extra registers needed, where MemoryDepth is the total number of memory entries in the memory space.

C. Using Read/Write Techniques Together:

This section depicts the architecture of a multiported memory with the incorporation of Hierarchial Bank Division with XOR design and Bank Division with Remap Table.

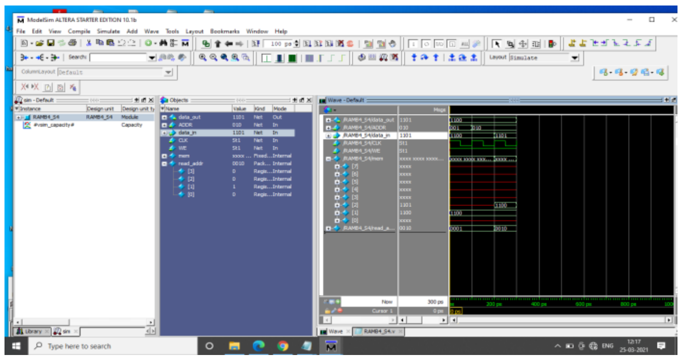
4. Block diagram



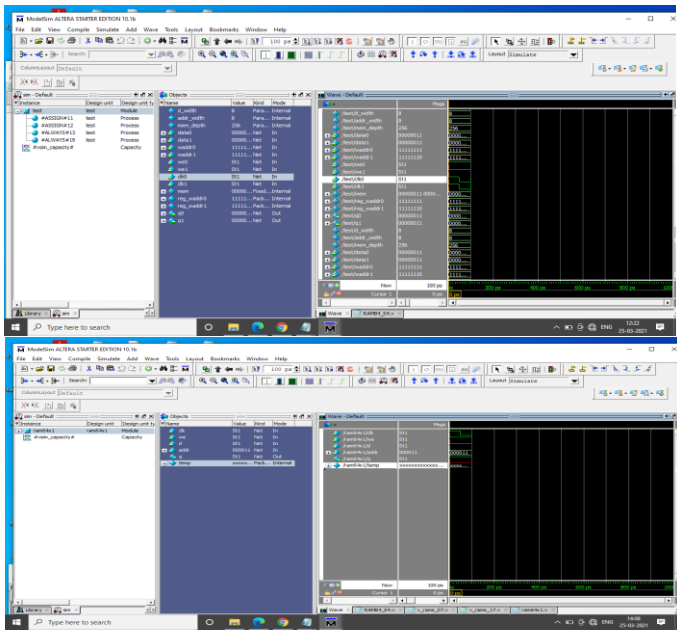


For starters, we're working with single-ported memory. The key disadvantage of single port memory is its wide area footprint, which is limited by the complexity of current multi-port memory designs. As a result, we've decided to go with dual-port memory. With proper bank organisation, a multiported design will achieve a 16 percent reduction in BRAM, a 21 percent increase in frequency, and a 27 percent reduction in slice utilisation.

4. Output



Register bank output



Output waveform of the Processor

There are two memories to be taken. We're going to put a processor in the centre. It's possible that the processor is an adder or a half adder. A processor is given two inputs (A and B). The output is saved to another memory after further processing. Multiport memory is the term for this. For storing the output, one or two memories are used.

5. Results And Discussion

This paper proposes efficient BRAM-based multiported memory designs for FPGAs. Current design methods necessitate a large number of BRAMs to implement a memory module with several read and write ports. This paper proposes a hierarchical 4R1W memory architecture that uses the 2R1W/4R building block and needs 33% less BRAMs than previous replication-based designs. When compared to XOR- and LVT-based methods, the proposed designs would reduce BRAM consumption by up to 53 percent and 69 percent, respectively, for 4R2W memory designs with 8K-depth. This paper also demonstrates how crucial it is to use the correct bank organisation when creating a memory template. With proper bank organisation, a multiported design will achieve a 16 percent reduction in BRAM, a 21 percent increase in frequency, and a 27 percent reduction in slice utilisation.

References

1. 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.
2. 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.
3. 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.
4. 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
5. 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
6. 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.
7. 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.File Formats For Graphics.
8. 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.
9. J.-L. Lin and B.-C. C. Lai, “BRAM efficient multi-ported memory on FPGA,” in Proc. Int. Symp. VLSI Design, Autom. Test (VLSI-DAT), Apr. 2015, pp. 1–4.