Volume 7 Issue 2 (2019) 113-115 ISSN 2347 - 3258
International Journal of Advance Research and Innovation

Implementation of an Efficient Design of Multi ported Memory on FPGA

A Priya, P Thenmozhi
Department of Electronics and Communication Engineering, Gnanamani College of Technology, Tamil Nadu, India.

Article Info
Article history:
Received 13 May 2019
Received in revised form
20 May 2019
Accepted 28 May 2019
Available online 15 June 2019

Keywords: Field programmable gate array, memory, multi-port, chip

Abstract
The utilization of block RAMs (BRAMs) is a critical performance factor for multi ported memory designs on field programmable gate arrays (FPGAs). Not only does the excessive demand on BRAMs block the usage of BRAMs from other parts of a design, but the complex routing between BRAMs and logic also limits the operating frequency. This paper first introduces a brand new perspective and a more efficient way of using a conventional two reads one write (2R1W) memory as a 2R1W/4Rmemory. By exploiting the 2R1W/4R as the building block, this introduces a hierarchical design of 4R1W memory that25% 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 BRAM usage for 4R2W memory designs with 8K-depth. For complex multiport designs, the proposed BRAM-efficient approaches can achieve higher clock frequencies by alleviating the complex routing in an FPGA.

1. Introduction

As FPGAs (Field-Programmable Gate Arrays) continue to increase in transistor density, de-signers are using them to build larger and more complex systems-on-chip that require frequent sharing, communication, queueing, and synchronization among distributed functional units and compute nodes—for example, applications that include multiple FIFO buffers for moving data between clock domains. These mechanisms are often best implemented with multi-ported memories—memories that allow multiple reads and writes to occur simultaneously—since they can avoid serialization and contention.

As another example, processors normally require a multi-ported register file, where more register file ports allows the processor to exploit a greater amount of instruction-level parallelism (ILP) where multiple instructions are being executed at the same time. However, FPGA-based soft processors have so far exploited little ILP, limited mainly to simple instruction pipelines. This is partly due to the fact that multi-ported memories (register files, in this case) are particularly inefficient to implement on FPGAs.

II.Literature review

The register file is an expensive component in the design of any processor, especially, when considering the additional ports that are needed to support multiple data paths within a wide-issue VLIW processor[5]. In a recent work, these additional resources were used to dynamically reconfigure the register file to support a dynamically reconfigurable VLIW core. The design can be perceived as a single 8-issue, two 4-issue, or four 2-issue VLIW cores. Consequently, the multi-ported design can operate in different modes, namely as one, two, or four register files, respectively, corresponding to the active number of cores. The implementation of the register file design on FPGAs using Block RAMs still results in unused resources due to the coarseness of the Block RAMs.

Existing implementation methods[6,7] of multi-port register files (MPo-RF) in FPGAs are not scalable enough to deal with the increased number of ports due to higher logic area and power. While the usage of dedicated block RAMs (BRAMs) limits the designer to use only single read and single write port, slice based approach causes large resource occupation and degrades design performance significantly. Similarly, the conventional multi-pumping (MPu) approaches are not efficient enough due to increased combinational delay and area of huge multiplexers. In this paper, we propose a new design which exploits the banking and replication of BRAMs with efficient shift register based multi-pumping (SR-MPu) approach.

III. LVT-Based Multi-Ported Memories

The new approach to implement multi-ported memories on FPGAs that can exploit the strengths of all three conventional techniques for adding ports. Their approach comprises banks of replicated block RAMs, where a mechanism of indirection steers reads to read from the bank holding the most-recent value for a given location. Multi pumping is orthogonal to our approach, and can be applied to reduce the area of a memory in cases where a slower operating frequency can be tolerated, as we demonstrate later . They name our indirection mechanism the Live Value Table (LVT), since it tracks which bank contains the “live” or most-recently updated value for each memory location.

A new design for true multi-ported memories that capitalizes on FPGA block RAMs while providing (i) substantially better area scaling than a pure logic-based approach, and (ii) higher frequencies than the multi pumping approach. The key to our approach is a form of indirection through a structure called the Live Value Table (LVT), which is itself a small multi-ported memory implemented in reconfigurable logic similar to Figure 1.1. 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 holds the most recent or “live” write value.

The intuition for why an LVT-based design is more efficient even though the LVT is implemented in reconfigurable logic is because the LVT is much narrower than the actual memory banks since it only holds bank numbers rather than full data values—thus the lines that are de-coded/multiplexed are also much narrower and hence more efficiently placed and routed. 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. Additionally, an LVT-based design and multi pumping are complementary, and we will show that with multi pumping they can reduce the area of an LVT-based design by halving its maximum operating frequency. With these techniques they can support soft solutions for multi-ported memories without the need for expensive hardware block RAMs with more than two ports.

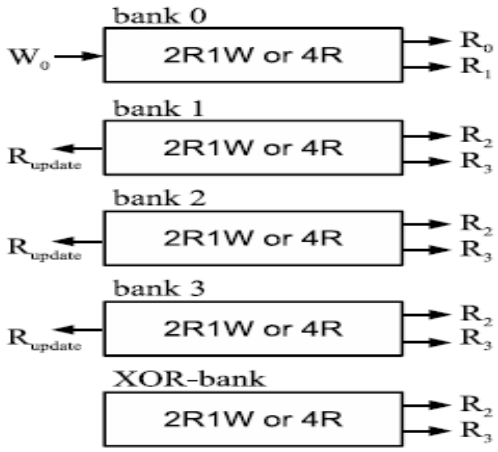

Fig. 1: 2R1W/4R FPGA Bank

1) This new way of using the 2R1W module is denoted as 2R1W/4R. This *hybrid* module can support either 2R or 1Wor 4R.

Corresponding Author,
E-mail address: riyaarunriya@gmail.com

2) The write request *W0* stores the data directly to the target data bank, and reads all the data at the same offset from the other data banks

4.Proposed System

FPGAs 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 hard core 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. Multi ported memories, which allow multiple concurrent reads and writes, are frequently used in various digital designs on FPGAs to achieve high memory bandwidth. For example, the register file of an FPGA-based scalar MIPS-like soft processor requires one write port and two read ports. Processors that issue multiple instructions require even more access ports. The shared cache system among multiple soft processors on FPGA should support multiple concurrent accesses. A routing table in a network switching function would also need to enable multiple accesses in order to support multiple requests from different ingress ports .

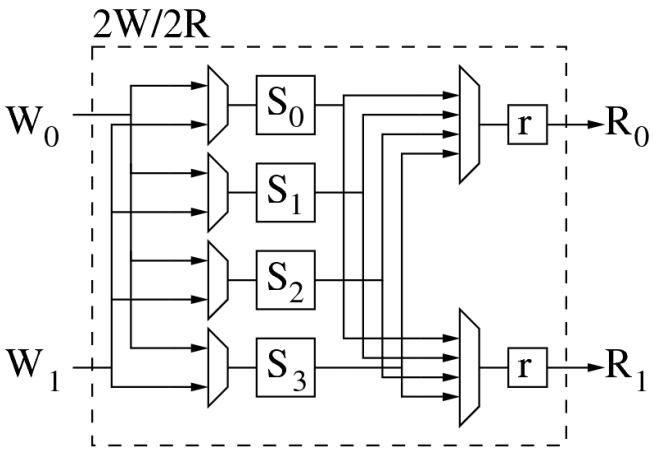

Fig. 2: Proposed 2W/2R Architecture

- The memory is composed of 2W/2R memory (constructed via replication of block RAMs).
- Each write port writes to its own bank, and each read port can read from any of all the banks via its multiplexer.
- Driving the multiplexer of the read port to select the output of the proper block RAM bank

5. Results

Simulation results are given by figure 3,4 and 5.

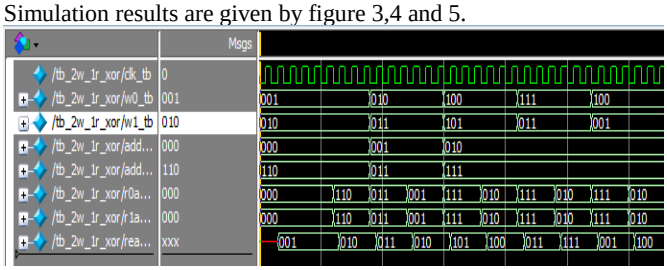

Fig.3: 2 write 1 read simulation waveform

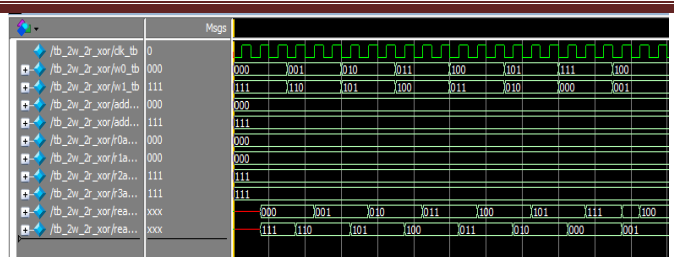

Fig.4: 2write 2 read simulation waveform

5.1 Synthesis Results

Flow Status	Successful - Fri Mar 02 13:11:15 2018
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	memory_2w__1r
Top-level Entity Name	memory_2w_1r
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	203 / 33,216 (< 1 %)
Total combinational functions	203 / 33,216 (< 1 %)
Dedicated logic registers	54 / 33,216 (< 1 %)
Total registers	54
Total pins	22 / 475 (5 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Fig.5: Area report for 2w 1r

Flow Status	Successful - Fri Mar 02 13:16:08 2018
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	memory_2w__1r
Top-level Entity Name	memory_2w_2r
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	154 / 33,216 (< 1 %)
Total combinational functions	154 / 33,216 (< 1 %)
Dedicated logic registers	64 / 33,216 (< 1 %)
Total registers	64
Total pins	31 / 475 (7 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Fig.6: Area report for 2w 2r

6.Conclusions

The proposed design of an efficient BRAM-based multi ported memory is going to design on FPGA. The existing design methods require significant amounts of BRAMs to implement a memory module that supports multiple read and write ports. Occupying too many BRAMs for multi ported memory could seriously restrict the usage of BRAMs for other parts of a design. This paper proposes techniques that can attain efficient multi ported memory designs. This paper introduces a novel 2R1W/4R memory. By exploiting the 2R1W/4R as the building block, this paper proposes a hierarchical design of 4R1W memory that requires 33% fewer BRAMs than the previous designs based on replication. Memories with more read/write ports can be extended from the proposed 2R1W/4R memory and the hierarchical 4R1W memory.

References

[1] F Anjam, M Nadeem, SWong. A vliw softcore processor with dynamically adjustable issue-slots. In Field-Programmable Technology (FPT), 2010 International Conference on, 12, 2010, 393 –398.

[2] F Anjam, S Wong, F Nadeem. A multiported register file with register renaming for configurable softcore vliwprocessors. In Field-Programmable Technology (FPT), International Conference on, 12, 2010, 403 –408.

[3] R Carli. Flexible MIPS Soft Processor Architecture. Technical report, Massachusetts Institute of Technology,Computer Science and Artificial Intelligence Laboratory, 6, 2008.
[4] B Fort, D Capalija, Z Vranesic, S Brown. A Multithreaded Soft Processor for SoPC Area Reduction. In IEEE Symposium on Field-Programmable Custom Computing Machines, 4,2006, 131–142.
[5] AK Jones, R Hoare, D Kusic, J Fazekas, J Foster. An FPGA-based VLIWprocessor with custom hardware execution. In International Symposium on Field-Programmable Gate Arrays, 2005.
[6] CE LaForest, JG Steffan. Efficient Multi-ported Memories for FPGAs. In Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays, FPGA New York, NY, USA, 2010. ACM, 10, 2010, 41–50,.
[7] N Manjikian. Design Issues for Prototype Implementation of a Pipelined Superscalar Processor in Programmable Logic. In PACRIM 2003: IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 1(8), 2003, 155–158.
[8] R Moussali, N Ghanem, MAR Saghir. Supporting multithreading in configurable soft processor cores. In CASES '07: Proceedings of the 2007 international conference on Compilers, Architecture, and Synthesis for Embedded Systems, New York, NY, USA, ACM, 2007, 155–159,.
[9] DA Patterson, G Gibson, RH Katz. A case for redundant arrays of inexpensive disks (raid). Proceedings of ACM SIGMOD international conference on Management of data, 1988.
[12] M Saghir, R Naous. A Configurable Multi-ported Register File Architecture for Soft Processor Cores. ARC: Proceedings of International Workshop on Applied Reconfigurable Computing, Springer-Verlag 3, 2007, 14–25.
[10] MAR Saghir, ME Majzoub, P Akl. Datapath and ISA Customization for Soft VLIWProcessors. In ReConFig: IEEE International Conference on Reconfigurable Computing and FPGAs, 9, 2006, 1–10.
[11] H Wong, V Betz, J Rose. Comparing fpga vs. custom cmos and the impact on processor microarchitecture. Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays, FPGA ' ACM, New York, NY, USA, 11,2011, 5-14.
[12] PYiannacouras, JG Steffan, J Rose. Application-specific customization of soft processor microarchitecture. FPGA: Proceedings of ACM/SIGDA 14th international symposium on Field Programmable Gate Arrays, pages, New York, NY, USA, 2006, 201–210.