Yet Another Multi-Port Memory for FPGAs

Kevin Townsend
kt@krtownsend.com

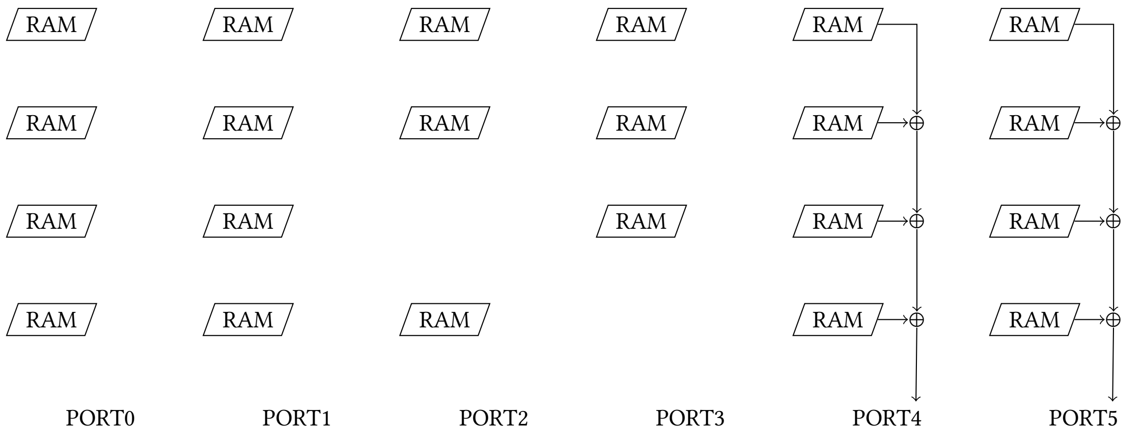

Figure 1: Seattle Mariners at Spring Training, 2010.

Abstract

We propose a simple extention to XOR memories presented in previous work. In this paper we generalize the XOR memory. We explore the resource usage on Xilinx/AMD FPGAs.

ACM Reference Format:
Kevin Townsend. 2025. Yet Another Multi-Port Memory for FPGAs. In *Proceedings of International Symposium on Field Programable Gate Arrays (FPGA '26).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Motivation

As computation needs keep increasing with Moore's Law one way to keep up has been specialized architectures. FPGAs provide a way to implement architectures without taping out an ASIC. However, the limitations of FPGA resources means some creativity is needed to map designs to FPGAs. This paper explores the limitation of FPGAs in the fact that FPGA memories have a limited number of ports. Specifically we look at creating memories with more than 2 ports.

The 3 major FPGA vendors implement distributed memory (small memories) and block memory (large memories) differently, however they share some characteristics.

All 3 vendors support distributed memory configurations with 1 full port and up to 3 read ports.

All 3 vendors support block memory with 2 full ports.

2 Source Code

We provide all of the source code used in implementation and testing our design at github.com/kevintownsend/mpm. We tested our design with Verilator and implemented the design with Vivado (Xilinx/AMD).

3 Introduction

We propose a simple generalization to XOR memories presented in previous work. XOR memories work by using the a xor b xor b = a property. We add bidirectional ports and analyze the perforamance of distributed memory and block memory versions of this design. We also present applications for these memories.

We also present a LVT memory that utizazes distrubuted memory xor live value table.

4 Live Value Table Memory

In (TODO citation) the live value table was implemented with registers. Previous work used distributed memory (TODO citation). However they did not use bidirectional xor ports in their implementation.

We create a LVT memory using the technique described in (TODO cititaion). TODO: describe technique.

We show we utilize x% less resources than LVT and I-LVT.

5 Impractical Designs

We obviously wanted to explore impractical designs with large numbers of ports. We say impractical because of the high resource usage of XOR and LVT memories at high port counts. N**2 for XOR and N*(N-1)/2 for LVT. However we were able to synthesize a 16 port memory.

FPGA '26, Seaside, CA
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

Without write delay the design runs at Xmhz (x% of max). With write delay and pipelining the design runs at Xmhz.

6 Conclusion

Several solutions to the port limit on FPGAs other than what is presented here. For example multi-pumping and banking. multi-pumping is the process of reducing the clock speed to increase the number of ports. For example a 300Mhz single port memory can handle 2 150Mhz ports. Banking requires stalls and routing logic due to the segmented memory. Our design uses replication and some creativity XOR and LVT to create multiple ports.

References

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009