Yet Another Multi-Port Memory for FPGAs Using XOR and LVT Methods

Kevin Townsend
kt@krtownsend.com

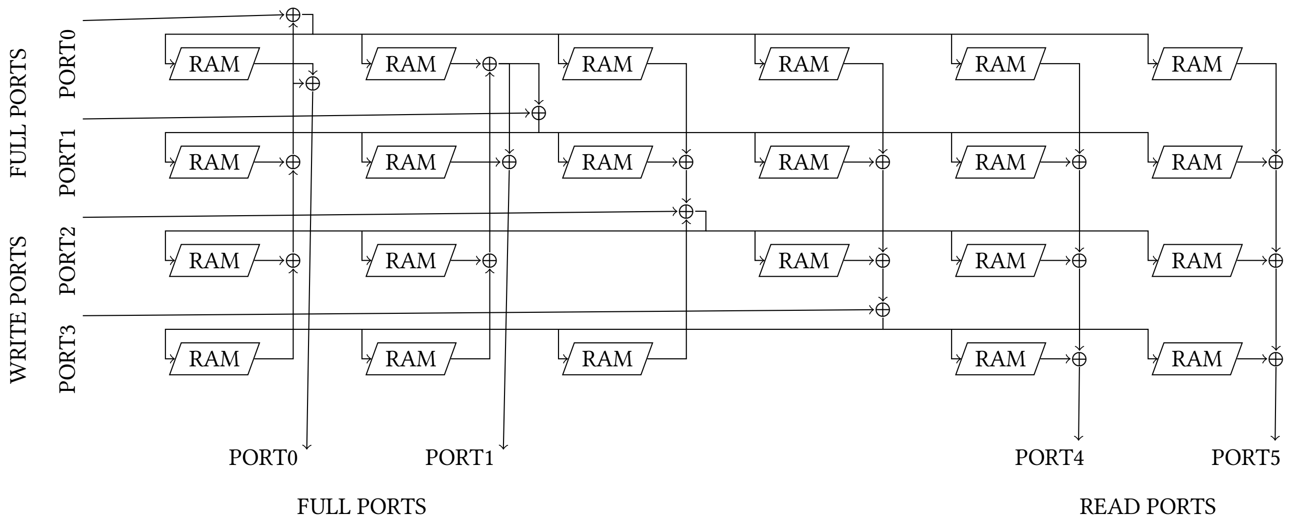

Figure 1: A multi-port memory with 2 full ports, 2 write-only ports and 2 read-only ports.

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.

This paper presents a novel and efficient architecture for creating live value table memory using an XOR-based scheme, emphasizing its bidirectional capabilities . By combining the principles of a banked memory design with the logical properties of the XOR operation, we address the limitations of traditional Live Value Table (LVT) approaches, particularly concerning resource overhead for deep memories and multi-port support. The proposed architecture eliminates the need for the control logic and multiplexing associated with a conventional LVT by storing XOR-encoded data across multiple memory banks. A key contribution of our work is the demonstration of a bidirectional memory system, where data can be written to and retrieved from any port, and memory access patterns can be reversed. This is achieved by exploiting the associative and commutative properties of XOR, allowing any data entry to be reconstructed by XORing the corresponding entries from all memory banks. The result is a high-throughput, coherent multi-ported memory that is particularly well-suited for implementation on Field-Programmable Gate Arrays (FPGAs). We evaluate the architecture's performance and resource utilization, showing that it uses significantly less logic and can achieve higher frequencies for deep memory configurations compared to LVT-based designs. This makes the XOR-based bidirectional live value table a compelling alternative for applications requiring high-performance, flexible memory access.

ACM Reference Format:
Kevin Townsend. 2025. Yet Another Multi-Port Memory for FPGAs Using XOR and LVT Methods. In *Proceedings of International Symposium on Field Programable Gate Arrays (FPGA '26).* ACM, New York, NY, USA, 3 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 between 1 to 3 read ports.

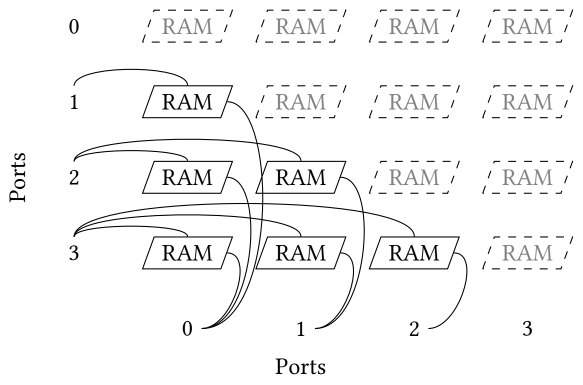

Figure 2: Multiport memory created with dual-port memories.

All 3 vendors support block memory with 2 full ports. None of the vendors support memories with more than 2 full ports.

Although this limitation is problematic for designs requiring multiple ports particularly write or full ports, we show that these resources make it possible to achieve high throughput quad and octal full port memories.

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 XOR memory

We propose a simple generalization to XOR memories presented in previous work[1]. 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.

The number of RAMs needed is:
$(W + F)(W + F + R) - W$
which expands to:
$W^2 + 2WF + F^2 + WR + FR - W$

4 Live Value Table Memory

XOR memories can be used by themselves, however a live value table (LVT) may be more efficient.

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

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.

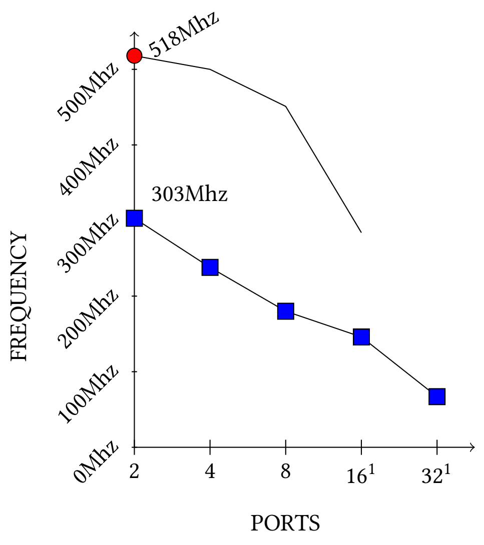

Figure 3: Frequency of design.

Table 1: Synthesis results for different port counts

Ports	LUTS	FF	BRAM	Max Frequency
2	127	0	1	303Mhz
4	640	0	6	238Mhz
8	3,012	0	28	180Mhz
16[1]	16,544	0	120	146Mhz
32[1]	85,728	0	496	67Mhz

Table 2: Synthesis results for different port counts for pipelined design

Ports	LUTS	FF	BRAM	Max Frequency
2	111	26	1	518Mhz
4	708	64	6	500Mhz
8	3,092	152	28	451Mhz
16[1]	16,800	448	120	284Mhz
32[1]	86,608	1,502	496	XMhz

5 Implementation

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.

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

[0]To reduce the number of IO ports and fit the design on the FPGA we used a wrapper for the multi-port memory for designs with 16 and 32 ports.

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

[1] Charles Eric Laforest, Zimo Li, Tristan O'rourke, Ming G. Liu, and J. Gregory Steffan. 2014. Composing Multi-Ported Memories on FPGAs. *ACM Trans. Reconfigurable Technol. Syst.* 7, 3, Article 16 (Sept. 2014), 23 pages. doi:10.1145/2629629

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