Design and implementation of multiport memories for re configurable devices

Arun S.Tigadi, Deepti. S.P, Hansraj Guhilot.

Abstract— The included block RAMs in the fabric have typically only two ports, so the multi-ported memories have become challenging to implement with FPGAs. Only by using logic elements or by combination of multiple block RAMs, the multi-ported memories can be designed. A new design is proposed in this paper called Live Value Table. This design has more number of read ports and write ports compared to other methods. The other method called XOR based approach is also introduced ,this requires more area compared to Live Value Table based approach.

Index Terms— FPGA , memory, multi-port , multipumping, LVT, XOR.

1 INTRODUCTION:

Designers increasingly use FPGAs to implement the larger and more complex systems-on-chip as these FPGAs continue to increase in their transistor density. Frequent sharing, communication, queueing and synchronization among distributed functional units and compute nodes are required by these complex designs. The construction of multi-ported memories of FPGA logic elements is inefficient and is one of the challenge . And the block RAMs with only two ports are provided by the FPGA substrate. However , the multi-ported memories with more than two ports must be SOFT that is constructed using logic elements and/or hard BRAMs.

Multi-ported memories have become challenging to implement withFPGAs as the block RAMs provided typically have only two ports. Here in this paper a thorough exploration of the design of FPGA based soft multi-ported memories is presented by evaluating conventional solutions to this problem and a new design is introduced here that combines block RAMs into multi-ported memories with more number of read and write ports called Live Value Table. And also another approach which is based on XOR operation which provides multi-ported memories that uses less logic than LVT design but with more block RAMs. Along with this LVT and XOR methods, other methods such as Replication, Banking and multi-pumping are designed and implemented. And their respective area, timing , power values of all methods are compared for different devices.

In VLSI systems, multi-ported memories are commonly used components such as register files in microprocessors, storage media or network applications. And through different ports, the content of a multi-ported memory can be accessed simultaneously. For processors like speed, media and communication processors, this multi-ported memory is especially valuable. Since all ports need to be verified, these multi-ported memories require more testing effort.

2 IMPLEMENTATION

2.1 Replication :

Replication is nothing but a process of making a replica or a copy of something. Here in this project it includes only one external write port with four number of external read ports. Only one single write port is connected to one of the four ports of each replicated RAM. This method is one of the simplest method to increase the number of read ports of a memory and also gives as many as copies of the memory as the read ports are required that is by keeping the common write address and data as shown in fig (a), the whole memory bank is replicated in order to provide more number of read ports. And to all the copies of this memory only one write port is routed to maintain them up to date. Hence this technique alone does not support for more than one write port. At the same address, all the bank replicas will be written with any data. Hence reading from any bank is equivalent. This replication requires more area and more power is consumed.

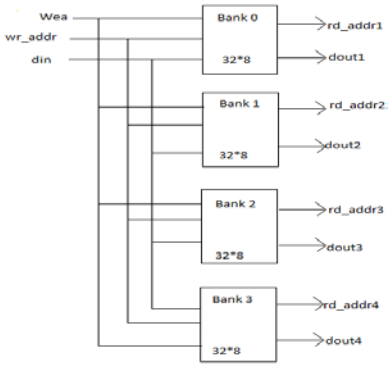


Figure 1(a) Replication

2.2 Banking :

This banking method is different compared to replication method. This method supports any number of read and write ports. Here the 32*8 memory location is divided among all multiple banks. As the four ports are considered, the 32*8 is divided into four 8*8 . That is each bank will have memory location of 8*8 as shown in above fig(b). These additional banks formed will support the additional read and write port, that is only one corresponding section can only be accessed by each write port and read port. But like replication, this method cannot access memory between the memory banks. Only the corresponding memory bank is accessed by its particular read or write port. Hence this design does not support the sharing between the ports. To create truly multi-ported memory, this method needs to be combined with any other methods. This is one of the simplest method to design. As compared to replication, this method requires less area space.

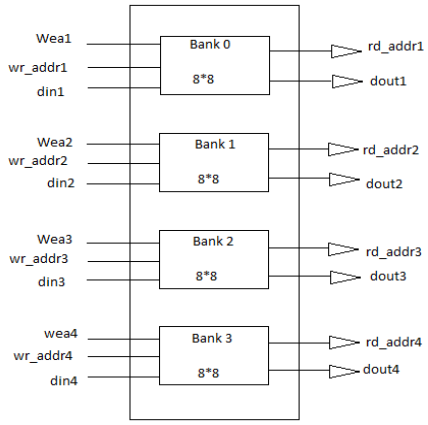


Figure 1(b) Banking

2.3 Multipumping :

To control the processing of the memory location access, this method makes use of an external clock which is the multiple speed of the system clock. This design requires multiplexers and registers to hold the address and data temporarily , of the pending reads and writes. Here there is no doubling or sharing of memory like in replication and banking, there is only one single large memory. Before and after the memory, there are registers and multiplexers to multiplex the read or write operations. The write address is written in the memory, before writing, the write address are being multiplexed using the select lines of multiplexer. All the write addresses are written with priority, the first write address in figure(c) has the first priority, so the remaining three write address ports have the registers which holds the temporary address and data of the pending ports. The address which are stored in memory are read using the select lines of the Dmux and are read at the corresponding output.This method has main advantage that it can simultaneously read and write the data. Also this approach requires less space. The main drawback of this method is with the increase in number of ports, the external operating frequency gets reduced.

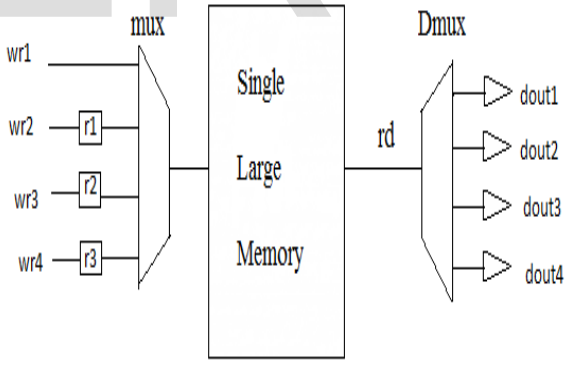


Figure 1(c) Multipumping

2.4 Live Value Table:

The LVT is a new design which is introduced for the true multi-ported memories that provides the more better area scaling compared to other approaches. And

has the higher frequencies compared to multipumping method. This LVT is nothing but a small multi-ported memory. This is somewhat similar to replication. Writing is exactly like replication, but it has a separate LVT module which stores the latest vaues. For each write address, the LVT stores the bank number which holds the most recent data.

An LVT based multi-ported memory makes use of different replication of banks for each write address port but each bank has many read address ports. All the read address are accessed parallel by all the banks. And the LVT module uses the multiplexers that helps to read out the correct bank number which it holds the most recently written bank number for each of the address. A banked design is allowed by this method to behave like a true multi-ported memory by which appropriate banks are directed by reads based on that bank holds the most latest or live write data. This LVT module has a same memory depth as that of the depth of the memory bank. This is one of the main cause of more area space. Compared to the actual memory banks, the LVT is much narrower as rather storing the whole data, it stores or holds only the bank numbers.

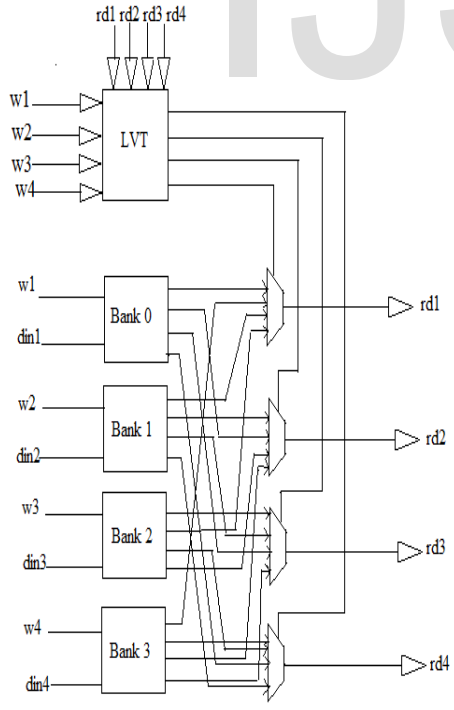


Figure 1(d) Live Value Table

So this LVT based multi-ported memories is more efficient compared to other approaches. The only basic idea of this design is that each read port should be able to connect to the most latest updated written bank for the memory location given. There are two challenges which are caused in this LVT based design by the togetherness of the LVT module itself and the multiplexers which are at the output side. They are : critical path is constituted and for implementation requires more number of logic elements. These challenges are avoided by the method which is explained in the next section.

2.5 XOR -Based Design:

This XOR (exclusive OR) is the logical operation. The outputs are true only when the two inputs have different values. When both the inputs are of same values then the output results false.

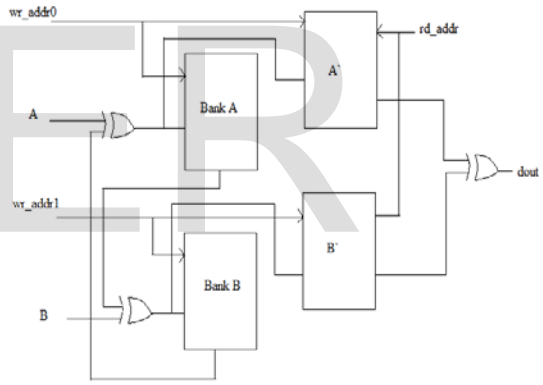


Figure 1(e) XOR Based Design

The truth table for this operation is shown below:

Inputs		Outputs
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

The XOR operation has the following properties:

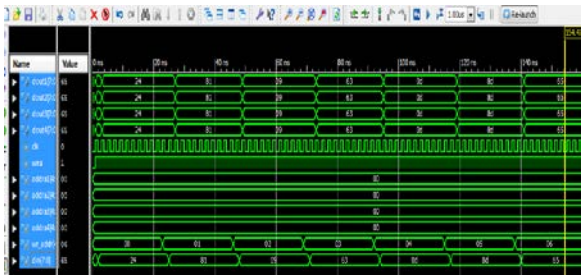
- A ^ 0 = A
- B ^ B = 0
- A ^ B ^ B = A

The third property indicates that the two values A and B can be XORed together , and by XORing the result A^B with B , the value A can be recovered. Here a simple memory 2W/1R is constructed as shown in figure(e). As illustrated in the figure, there are two BRAMs as the write ports each has its own bank. And to both the BRAMs each writes are copied that is the same value is written in all the BRAMs of the corresponding locations. The value A is stored by the Wo write port to some location, now this value A is XORed with the value of bank B of the corresponding location as of bank A. Similarly, the value B is stored by W1 write port to lower location of bank B, this value B is XORed with the value of the corresponding location of bank A. The main goal of this design for the read port is that it should only consist of the XOR values of the memory bank outputs. And this goal is achieved by storing the XORed values in their respective separate banks that is A` and B` as shown in the above design. And at the end, the stored values are XORed again. So that should result in recovering the value of A.

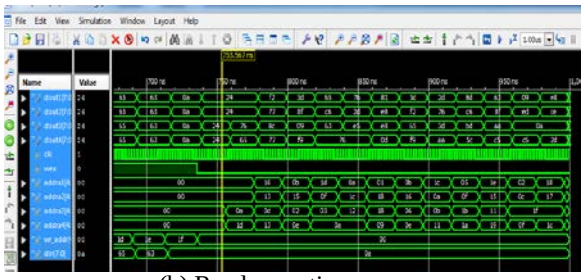
3. RESULTS :

The results part includes the simulation and chipscope results of all the five methods.

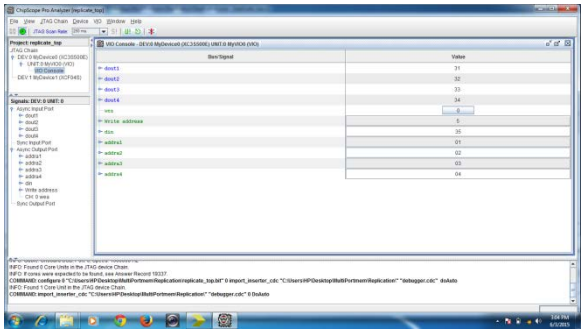
3.1 Replication:



(a) write operation



(b) Read operation

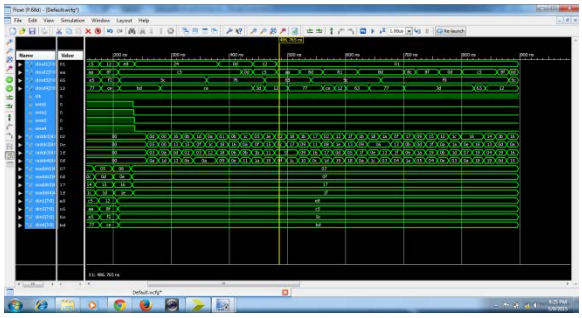


(c) Chipscope results

The above screenshots are the simulation results of the replication which includes write and read operations. When the write enable is 1, the data din is written in to write address as shown in figure. And when write enable becomes zero, the address are read in output i.e dout. The same read and write operations are shown for all methods below in the screenshots. And this also includes screenshot of chipscope results.

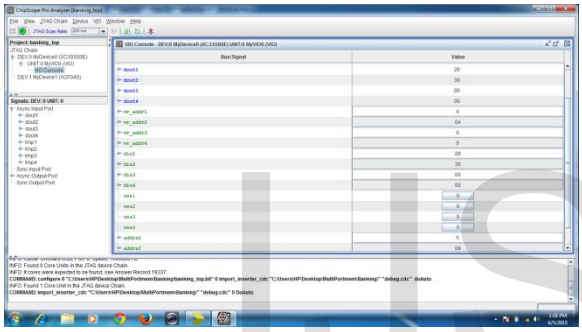
3.2 Banking :

(a) write operation

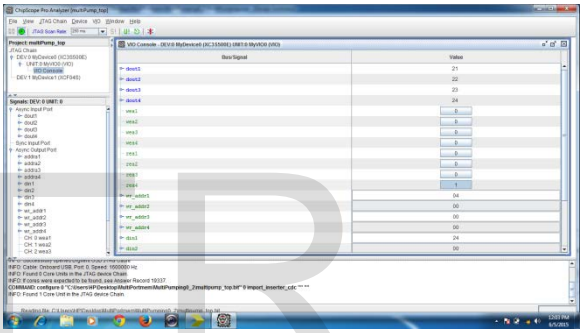


(b) read operation

(b) read operation



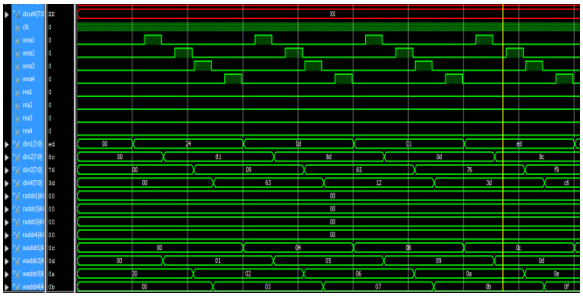
(c) Chipscope Results



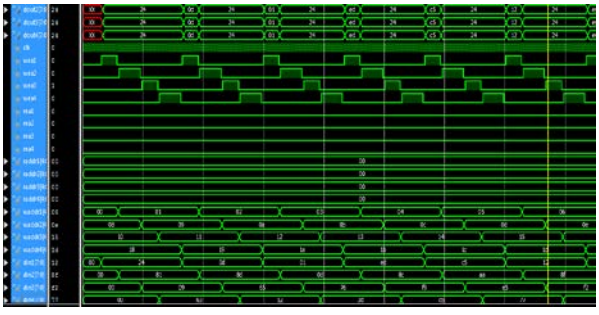
(c) Chipscope Results

3.3 Multipumping :

3.4 Live Value Table :



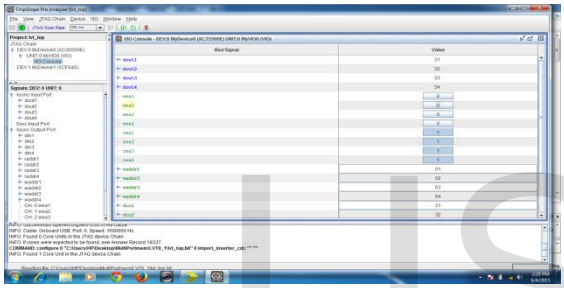
(a) write operation



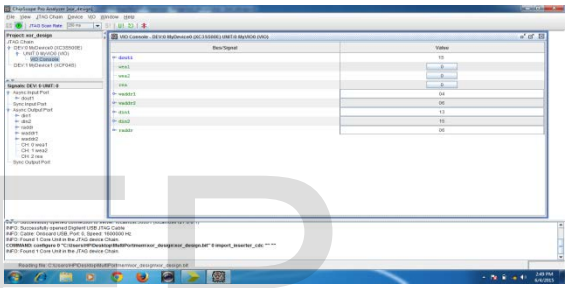
(a) write operation

(b) read operation

(b) read operation



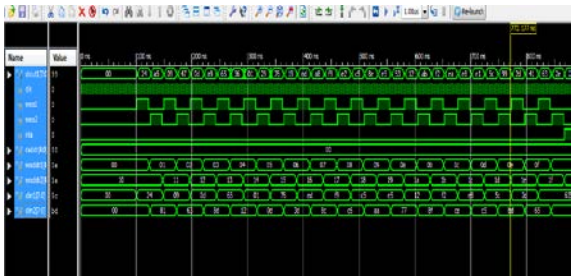
(a) Chipscope result



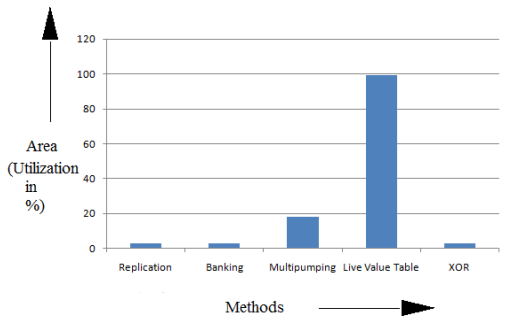
(b)Chipscope result

GRAPHS :

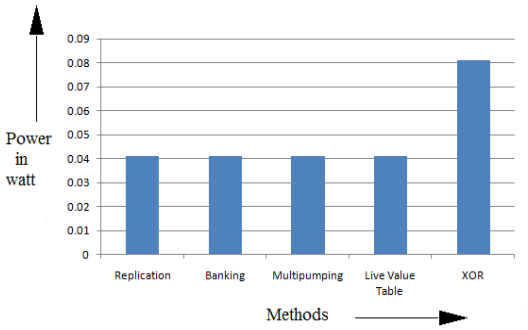
3.5 XOR :



(a) Write operation



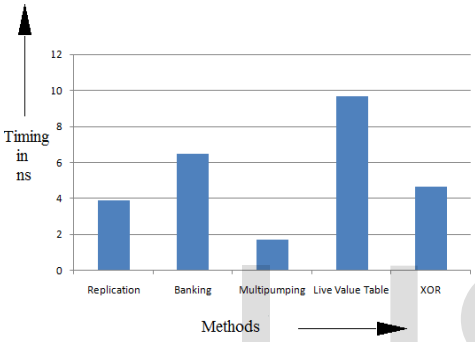
(a) Area utilization versus all methods



(b) Variation of Power for all methods

Depth	Slices	BRAMs
2	2	4
4	4	4
8	6	4
16	8	4
32	10	4
64	10	4

(a) Replication



(c) Variation of execution time for all methods

Depth	Slices	BRAMs
2	2	4
4	4	4
8	6	4
16	6	4
32	6	4
64	6	4

(b) Banking

The above graphs includes the plotting of area, power,timing for all the different methods for the spartan 3E family. It includes variation of area utilization, power and run time for all the five methods.

As the depth of the memory increases, the number of slices increases or remains same sometimes with each increase in depth as shown in the tables below and above

The below tables shows the number of slices and BRAMs used for different depths for all the five methods :

Depth	Slices	BRAMs
2	42	1
4	44	1
8	45	1
16	51	1
32	57	1
64	57	1

(c) Multipumping

Depth	Slices	BRAMs
2	285	4
4	300	4
8	319	4
16	324	4
32	334	4
64	337	4

(d) LVT

Depth	Slices	BRAMs
2	18	4
4	19	4
8	20	4
16	24	4
32	25	4
64	25	4

(e) XOR

4. Conclusion :

The Live Value Table and XOR based approaches have been introduced which are fastest among the other conventional approaches like replication, banking and multipumping. These conventional methods either have inefficient area or slow. So the Live Value Table and XOR approaches are introduced that behaves like a true multi-ported design.

REFERENCES :

[1] Implementing Multi-Port Memories in ProASICPLUSDevices.http://www.actel.com/documents/APA_MultiPort_AN.pdf, July 2003. Application Note AC176, Accessed Sept. 2009.

[2] Mercury Programmable Logic Device Family Data Sheet.http://www.altera.com/literature/ds/ dsmercury.pdf, Jan 2003. Version 2.2, Accessed Sept. 2009.

[3] Stratix III Device Handbook Volume 1, Chapter 4: TriMatrix Embedded Memory Blocks in Stratix III Devices.http://www.altera.com/literature/hb/stx3/stx3_siii51004.pdf,May 2008. Version 1.8, Accessed Sept. 2009.

[4] Advanced Synthesis Cookbook: A Design Guide for Stratix II, Stratix III, and Stratix IV Devices.http://www.altera.com/literature/manual/stx_cookbook.pdf, July 2009. Version 5.0, Accessed Nov. 2009.