International Journal of Recent Engineering Research and Development (IJRERD)
ISSN: 2455-8761
www.ijrerd.com || Volume 02 – Issue 06 || June 2017 || PP. 44-54

Efficient Multi Ported Memory Designs On FPGA For Data Access Policies In Image Processing

Rinku K R[#1], Ms. Anju Susan Varghese[#2]
[1]*Final Year M. Tech ECE, MET'S School of Engineering, Mala, Thrissur, Kerala*
[2]*Assisstant Professor, Department of ECE,MET'S School of Engineering, Mala, Thrissur, Kerala*

Abstract: The usage of BRAMs plays a key role in determining the performance of multi-ported memories on FPGA .Such memories are widely used for image processing applications. The FPGA dedicated for such applications require to support various data access policies for image processing algorithms. Normally, for each peculiar data access policy, the memory has to be redefined which is a tedious task. The memory so designed should be efficient since multi-ported memory designs are used. This paper introduces a new architecture which supports the data access policies for image processing algorithms by exploiting two reads one write (2R1W) memory as a 2R1W/4R memory building block, for a hierarchical design of 4R1W memory that is very efficient and uses considerably lesser area than conventional methods. Instead of redesigning the memory with change in scan orders in image processing this perspective initially sets a predetermined order that could easily support the required data access policy for image processing.
Keywords: Block RAM (BRAM), multi-ported memory data access policy, field programmable gate array (FPGA), image processing algorithm

I. INTRODUCTION

A field-programmable gate array (FPGA) is defined as an integrated circuit that could be redesigned by a customer or a designer at the field thus its specified as "field-programmable". Generally the configuration of FPGA is described with hardware description language(HDL), similar to ASIC. For example the Xilinx FPGAs introduces a Spartan FPGA which consist of programmable logic blocks in an array with lot of interconnects that are capable enough to wire these blocks. These logic blocks mentioned here are so designed for carrying complex functions, or for supporting simple logic gates. These logic blocks are also the occupants of memory which uses flip flops or other designed memory blocks.
Design specifications of FPGA includes:

- Configurable logic block count.
- Fixed function logic block count.
- Memory resource size,(embedded block RAM).

In Xilinx FPGAs, a Block RAM is merely defined as a dedicated dual-port memory which constitute of various kilobits of RAM. Each FPGA has several such blocks. Block RAM, or block memory, is actually RAM that is embedded within the FPGA ensuring data storage. The LUT (look Up Table) inside the logic blocks implements the necessary logic functions and thus its configurable. Multiport memory contributes multi access ports either to different processors or to the different parts of same processor. Bus mechanism is an example. These are widely used in VLSI systems for assisting various applications ,which includes usage as register files in processors , storage element for media and network related applications.

The multi access ports ensure the multiple access to the memory on FPGA which will be carried out simultaneously. This is advantageous for high speed processors, media and communication processors etc. Also this ability to support concurrent read and write requests is utilised by several digital designs on FPGA for attaining wider memory bandwidth. So as the name implies such memories could directly and independently execute read or write in preferable address locations through this multiple ports. Each applications in image processing varies according to the behaviour of the area it deals with and the designer who process the image data. FPGAs dedicated for image processing applications hence are to be configured to perform the desired task. As memory is inevitable in all implementation irrespective of the field its designing is very important and promising. An efficient memory design could always enhance the system performance to greater extend. Thus its important to combine an efficient memory design that could support image processing algorithms. Various data access policies for image processing algorithms normally determines the design of memory on a FPGA. According to change in these policies or scan order the memory is designed . There are normally 6 orders that is discussed later in this paper. They are various order that is used for scanning images for acquiring data in image processing applications. Section II describes the literature review which includes various related works that paved way for the emergence of this paper. The analysis of previous methods is done which finally evolves into

an idea that is proposed in section III followed . The detailed description and implementation of this proposed architecture is evaluated in section IV where the results are compared and analysed. Finally section V concludes the paper from the analysis made that could firmly support the idea of this paper.

II. RELATED WORKS

1.B.-C. C. Lai and J.-L. Lin [1] proposed an efficient architecture of increasing the read and write ports and integrating them for implementing efficient multiport memory designs. The replication method in [1], [2] and [4] increase the read ports by replicating the data on multiple BRAM cells. Though the logic is simple it uses considerable area for implementation. LVT approach for increasing the write ports utilizing a look up table for fetching latest updated data is also discussed in these papers. Paper [3] describes another conventional method that uses xor operations to encode and store the latest updated data by duplicating BRAMs which is different from LVT approach for write port increment. The increased area used for implementing these approaches limits the efficiency of FPGA memory and hence BDX, HBDX , BDRT methods in [11] put forward more efficient design for multi-ported memory by introducing an architecture in [1] . It uses 2R1W as a 2R1W/4R dual modes module as the building block for this design architecture and could be implemented for a hierarchial 4R1W memory and similarly extended for implementing 4RnW memory designs. This architecture combines HBDX method [11],[1] for increasing read ports and makes a 2R1W memory to act as dual mode either a 2R1W memory module or 4R , which is denoted as 2R1W/4R. 2R1W memory supports 2 read ports and 1 write port. But when working in 4R mode no write requests will accessed. Implementing a memory on FPGA that is dedicated for image processing is challenging as the memory here is to be redesigned according to the change in data access policies used [9]. This complication can be dealt if the addresses accessing a memory is pre determined in a specified order. Also the memory used here should be efficiently designed that uses lesser areas than previous designs. Combining these ideas of using an efficient memory design for multi port memory on FPGA that could assist the image processing applications without redesigning the memory with changes in data access policies used for image processing algorithms in scanning the data from an image ,the proposed architecture is introduced in the next section.

A more better design has to be implemented that could overcome the limitations of the earlier design. This HBDX,BDRT approaches [1],[11] proposes a more methodical design to implement multiported memories on FPGA. The approach facilitate multiple reads with XOR operations, while multiple writes can be achieved using additional BRAMs, this is different from replication method. The main memory architecture is similar to that of previous work and it includes a remap table to track the location of the correct data. Highlighting the main architecture, the approach contributes an innovative design of using a 2R1W module as either a 2R1W or a 4R module, denoted as a 2R1W/4R memory. By applying the 2R1W/4R, this paper utilizes the dual mode and implements a hierarchical XOR-based design of 4R1W memory that significantly reduces the BRAMs than previous designs. Efficient memories supporting multiple read/write ports can be supported implemented by extending the proposed 2R1W/4R memory and the hierarchical 4R1W memory.

A. *HBDX- Hierarchical Bank Division With XOR*

HBDX can be applied to support more than two read ports ,since BDX needs to use multiple 2R1W memories , that results in increase in BRAM usage. HBDX uses a 2R1W module as a 2R1W/4R module.

1)2R1W/4R (An Efficient Two-Mode Memory)- This hybrid module supports either 2R and 1Wor 4R and uses the same design as the 2R1W module introduced in the previous section. Fig. 1demonstrates the procedure of two modes.

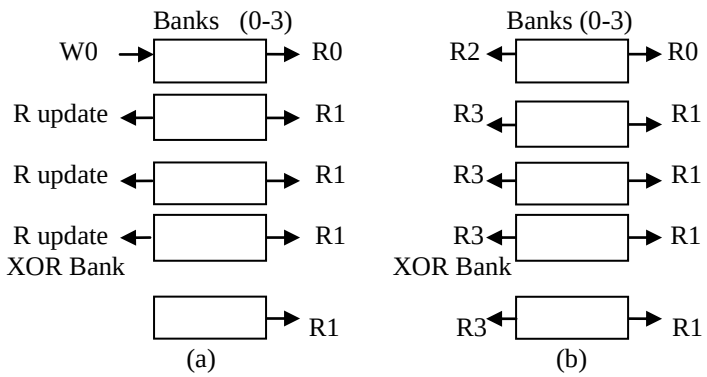


Fig.1 Two modes of 2R1W/4R module which is implemented with four data banks and one XOR-bank. (a) 2R1W mode. (b) 4R mode.[1]

Fig. 1(a) shows the 2R1W mode. For a single write request *W*0, this design is capable of supporting up to 2 conflicting reads. *W*0 writes the data directly to the target data bank, and at the same time reads all the data from the same offset of the other data banks (*R*update) such that it can update the XOR-bank. Fig. 1 (b) shows the 4R mode. This mode works only when no write requests exists.In this case, it can support up to four reads. Considering the worst case when all the reads (*R*0 to *R*3) are accessing bank 0, *R*0 and *R*2 access bank 0 directly. Simultaneously, *R*1 and *R*3 recovers the required data by XOR-ing the values at the same offset of other data banks along with the XOR-bank.

2) HBDX Designs With 2R1W/4R Module.

A more BRAM-efficient design can be achieved using HBDX, which adopts a hierarchical structure that combines the 2R1W to obtain a 4R1W apart from replicating the 2R1W module. For enhancing the design, HBDX here exploits the 2R1W/4R design as the basic building module for implement a 4R1W module. Fig.2 given below shows a 4R1W memory design using the HBDX method utilizing 2R1W module as building block. As previously discussed a 2R1W module can be used as either a 2R1W or a 4R module.The HBDX exploits this versatility in establishing an efficient 4R1W design. Considering one of the worst cases where all the 4R and 1W are accessing bank 0. *R*0 and *R*1 reads directly from bank 0. *R*2 and *R*3, reads the same offset from the other data banks (banks 1–3) and the XOR-bank for recovering the target data. *W*0 uses pipeline architecture that stores the data directly to bank 0, simultaneously the data at the same offset of the other data banks are read and XOR-ed at the same cycle.

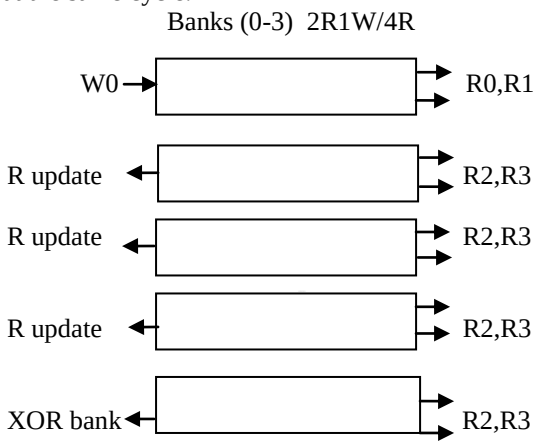


Fig. 2. 4R1W memory using HBDX.[1]

mode, but other banks are in the 4R mode. The HBDX-based 4R1W memory is highly BRAM efficient than duplicating the 2R1W module

B. Bank Division With Remap Table (BDRT)

BDRT is an approach to increase write ports proposed in priorly.

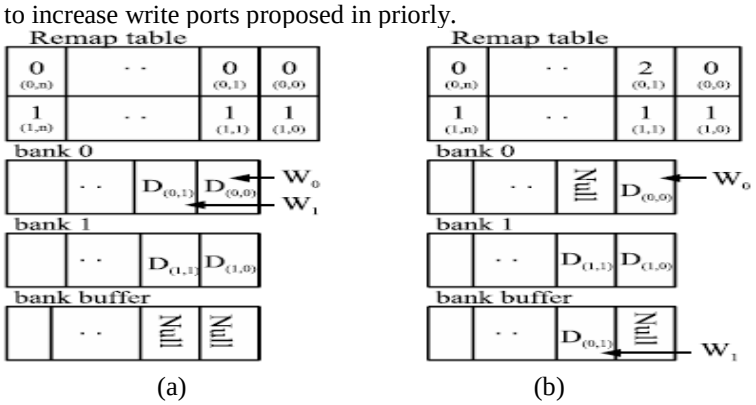


Fig.3. A 1R2W memory implemented with BDRT technique. (a) According to the remap table, both *W*0 and *W*1 access bank 0. The null entries in BRAMs do not store any valid data.(b) Final state of the multiported memory after *W*0 and *W*1.[1]

Main difference from LVT method is that , BDRT avoids replicating the whole memory space thus supports multiple writes using additional BRAMs along with a remap table to track the location of the latest updated data. Fig.3 is an example of the design for a 1R2W memory. From the illustration we can see that there are 2 data banks (banks 0 and 1), one bank buffer, and a remap table. The Null entries in a memory bank indicates that the entries that do not have any valid data. While accepting *W*0 and *W*1, these requests will priorly look up the remap table to locate the correct BRAM that stores the latest updated data. As per the remap table, *W*0 and *W*1 are, accessing address 0 and address 1 in bank 0. Now ,one write , *W*0 will store the data directly to address 0 in bank 0. *W*1 thereby gets directed to the bank buffer with a null entry in offset 1 . Simultaneously , the remap table gets updated as it has to reflect the modified location of *W*1, as in Fig.3 (b). The address 1 of the remap table gets updated with the value 2 which is the identification number of the bank buffer. Thus this state of remap table implies that the data of address 1 is presently stored in the bank buffer.Thus we can see that this BDRT method utilizes only smaller storage space as compared to the previously discussed LVT approach. The extra cost of BDRT comprises of the additional registers required to implement the remap table. The number of extra registers is given by,

# of registers for remap table

$= ([\log2(\#DataBanks + 1)] - 1) \times MemoryDepth.$ (1)

where **MemoryDepth** denotes the total number of memory entries in the memory space and *#DataBanks* implies the number of data banks in the multiported memory.

III. PROPOSED SYSTEM

A. HBDX- BDRT integrated architecture

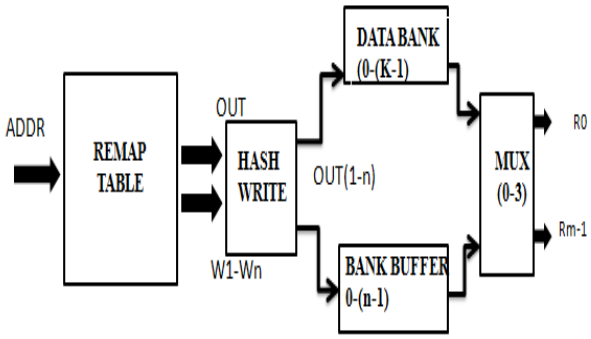


Fig. 4 . Architecture required to integrate HBDX and BDRT for a *mRnW* memory [1].

This architecture [1] proposes an architecture that implements the design of a multiported memory that integrates HBDX and BDRT, that supports a *m*R*n*W memory. This memory architecture is splitted into *k* data banks, in which BDRT, supports all the writes and thereby requires total $n - 1$ bank buffers. A hash mechanism is introduced that distributes the writes to banks.

B. Extend To 4RnW Memory With 2R1W/4R Module

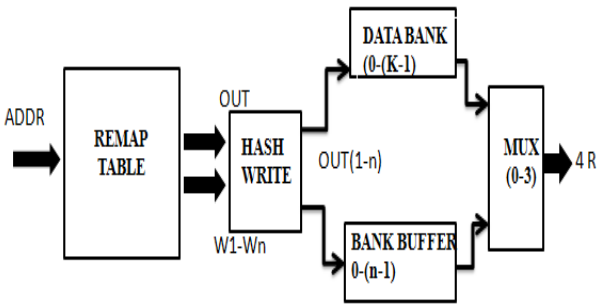


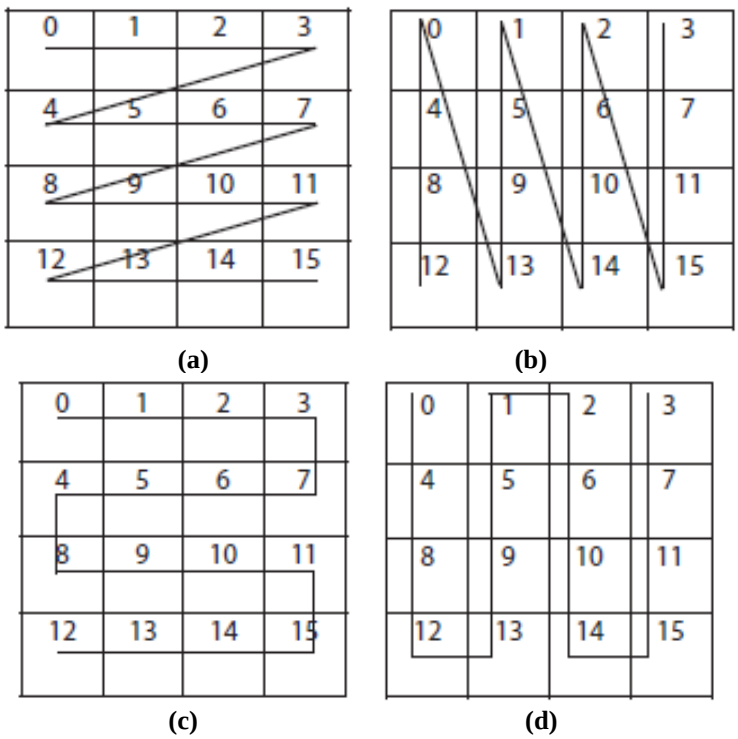
Fig.5 *4RnW* memory integrating HBDX and BDRT, utilizing 2R1W/4R modules as building blocks [1].

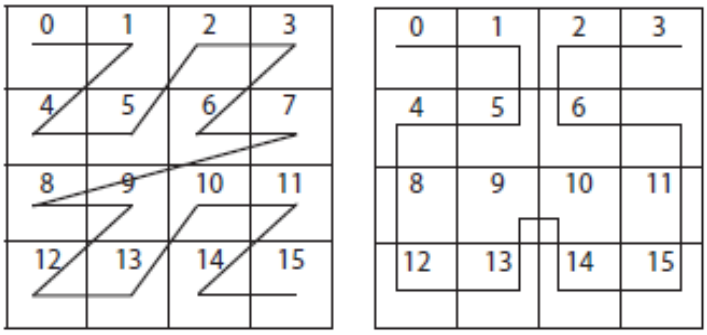
In order to make this extension more efficient here we propose [1] a wise way of implementing a 4R*n*W memory using the design 2R1W/4R module that was discussed before . Though, it functions in dual mode ,a 2R1W/4R module is generally a 2R1W memory block considering design cost and performance. Fig.5 above shows a 4R*n*W memory implemented with *k* data banks that makes use of 2R1W/4R as the building block. The four reads can be easily achieved by the 4R mode of the 2R1W/4R module. Somehow 2R1W/4R module will not be able to support any write requests while servicing more than two reads. In order to support *n* write requests, this design makes use of *n* extra bank buffers , for write requests.

C. Data Access Policies

In FPGA , for most applications like signal processing, medical imaging, aero-space and defense systems, image processing computer vision, speech recognition, cryptography, bio informatics and growing range of other areas memory plays an important role. The processors that facilitates these application normally make use of old memory approaches, which is differentiated as on-chip and off-chip memories. The on-chip memories support smaller data processing applications whereas off-chip memories supports complex data processing applications. The on-chip memory which is implemented inside FPGA will not have any external connections on the circuit board and hence provides highest throughput and thereby achieves maximum access speed from memory .The on-chip memory is subjected to continuous increment , thereby exhibiting enhanced performance of portable devices and high- performance processors for future generations . On-chip memory is available based on the applications that utilises it. Random Access Memory(RAM) is the on-chip memory widely used. Static Random Access Memory(SRAM) as we know is expensive ,less complex, easier to control ,faster and consumes less power when compared to Dynamic Random Access Memory(DRAM).The content of the DRAM cells are to be periodically refreshed and thus there will be an increment in power consumption. Due to above mentioned facts, on-chip SRAM memory is preferred for image processing applications on FPGA.

The prominent problem faced by image processing applications for accessing the data from memory is the scan order or data access policies . Scan order demonstrate the proficiency of scanning the data for image processing algorithms. Mainly used image scan orders are Row, Column, Row prime, Column prime, Morton and Peano–Hilbert., where the data access policy named Peano–Hilbert is considered to be complicated to implement [9]. Row scan or Raster scan is the scan order preferred by most of the image sensors. This is because the mentioned scan order is very useful when the applications demands that all the pixels have to be transmitted out of a chip without any possible extra spatial processing. In case of requirement of spatial processing such as mean (average) or median computation, a provision for multiple additional storage elements are done (digital or analog) outside or inside the pixel. It is notable that since row scan transmits pixels serially row after row a more better approach is put forward by Morton and Peano–Hilbert scan order ,where the pixels are sequentially read and are concentrated in blocks .





(e) **(f)**

Fig.6 (a) Row access, (b) Column access (c) Row prime access, (d) Column prime access, (e) Morton access, and (f)Peano-Hilbert access. [9]

Block clusters such as 2×2, 4×4, 8×8 can be extracted easily by , transmitting the pixels in the block one after another.

D. Address Generator

The storage sequence and image block retrieval in Image processing algorithms, are patterned firmly. Hence the memory allocation is performed by the address generator in two ways:

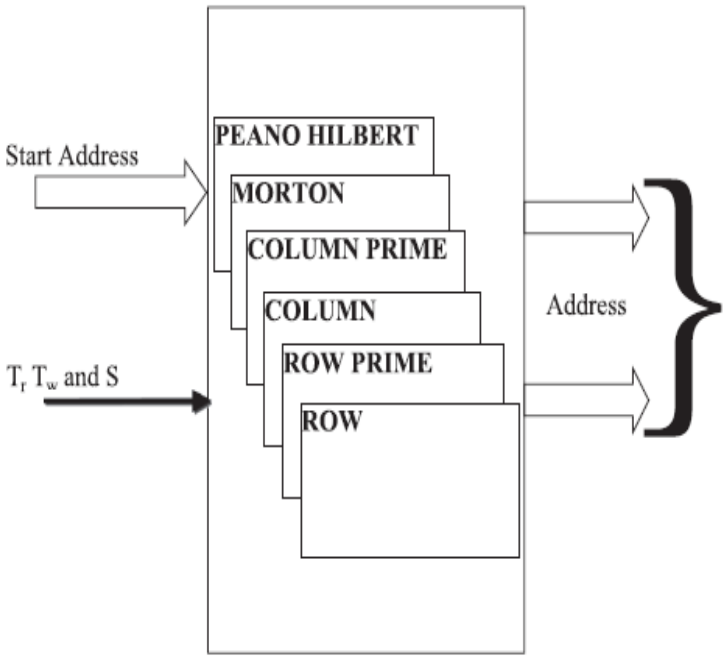


Fig. 7. Address generator [9]

(a) Primarily the incoming data is stored or written to the corresponding consecutive locations and the data to be consumed is read out in the specified order.
(b) Data received is priorly written in pre determined pattern such that the reading process proceeds in the specified sequential locations.

Each memory location is represented by four bits..Raster is formed as a results of line by line scanning and hence row order scanning is named as raster scan order. .Here the current and the next consecutive pixel values are physically next in memory.

E. Proposed Architecture

The architecture mentioned in the previous section III (A),(B) in Fig. 4,5 supports efficient memory design that can reduce the BRAM usage while implementing multi ported memories. The above discussions highlights the importance of an address generator that is specifically designed to support several data access policies (Fig.6) for image processing.

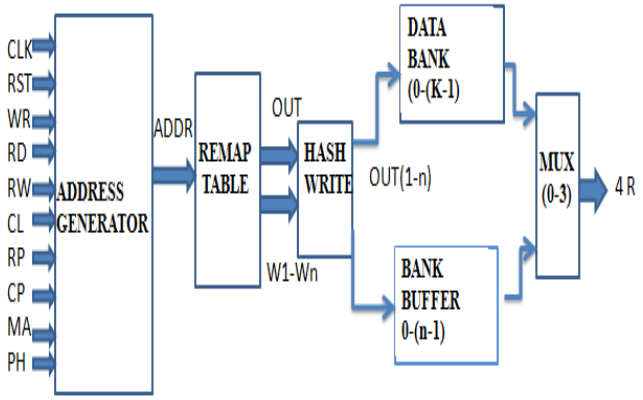


Fig.8 Modified 4RnW memory design for image processing .

Combining these two will contribute a more efficient memory design that is highly efficient and uses minimal area for implementing an image processing application. Here we implement a 4RnW memory for supporting data access policies in image processing algorithms, such that memory redesigning is not required with change in scan order. The address generator is so designed that it support dual port memories and we know that BRAM is dual port memory. The architecture here supports 4RnW memory that consist of address generator that can generate the address initially in the pre determined order according to the enables (RW, CL, RP ,CP,MA,PH) corresponding to different data access policies discussed before. When RST (reset) is one and any of the data access order is enabled , for the clock it for the image processing application. Since the address is initially generated in a required order that supports specific data policy the processing of image could be done more efficiently by exploiting the efficient multi ported memory architecture following it. Since the design has multi ported memory, data can be easily accessed.

IV. RESULT ANALYSIS AND SIMULATION

Methods	Total gate counts	Occupied slices
HBDX_BDRT_2	2188	32
HBDX_BDRT_4	1920	42
Modified HBDX_BDRT_2	5085	170
Modified HBDX_BDRT_4	5079	181

Table 1

Within the limitation of doing this project in the personal laptop with model sim 6.3f and Xilinx 8.1 ISE the comparison could be generated for number of gates used that implies the area and also the slice count to compare the slice utilization for 4R2W and 2R2W memory designs using architecture in Fig.4 [1] and Fig. 8. That is existing and proposed designs. Last 2 rows in the table shows the modified proposed architecture including address generator for supporting data access policies for image processing. Comparing the 2R2W memory design implementation with 4R2W memory existing design [1] we can see that as the number of ports is increasing there is decrease in gate count and thereby area and correspondingly the slice count is increasing. Similarly for the modified architecture proposed in this paper as the number of ports are increasing the gate count is decreasing with increase in slice utilization which implies the efficiency and enhanced performance. Thus this can be efficiently used in image processing applications that will decrease the area considerably. The results are generated only for a portion of design implemented using the prescribed approaches. While

implementing multi ported memory on a FPGA completely for specific application using this approaches larger variations could be attained that will decrease the BRAM usage considerably.

A. Simulated output waveforms

Modified 4R2W memory design

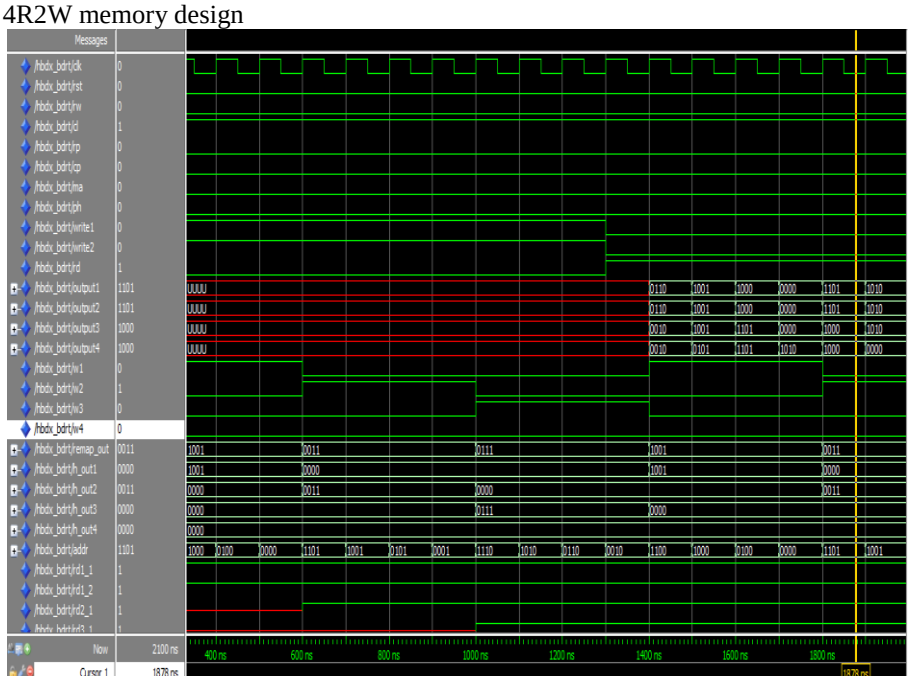


Fig.9 Address generator- Column access.

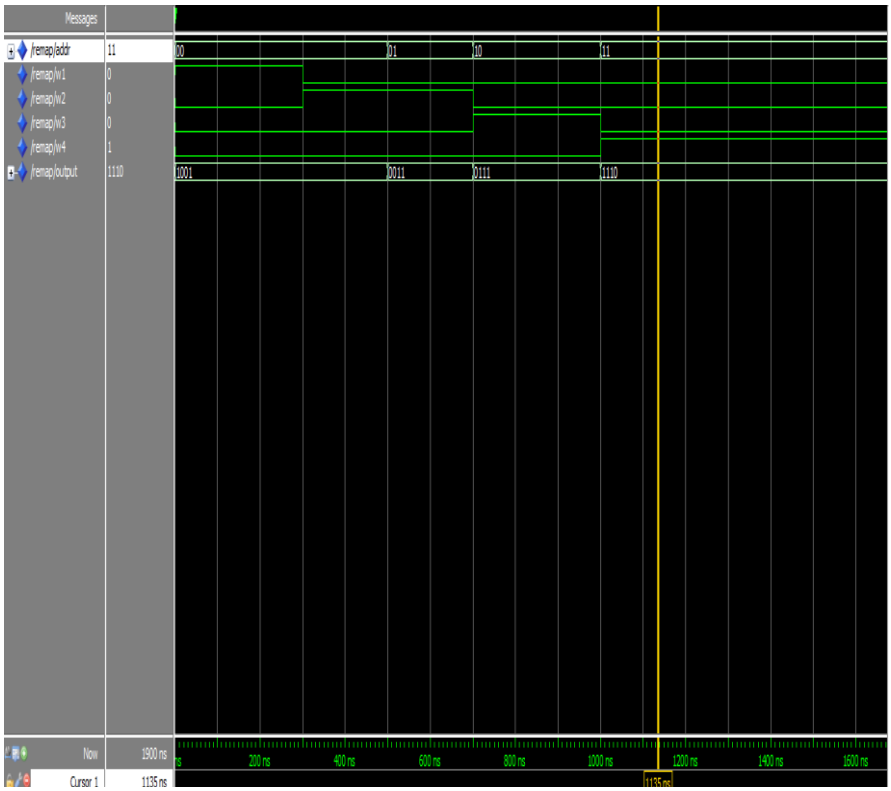


Fig.10 Remap table

The simulation output waveforms for each respective block in Fig.8 is shown. The address generator generating the specified scan order three out of six is given here. Later the output is the incoming inputs to the remap table.

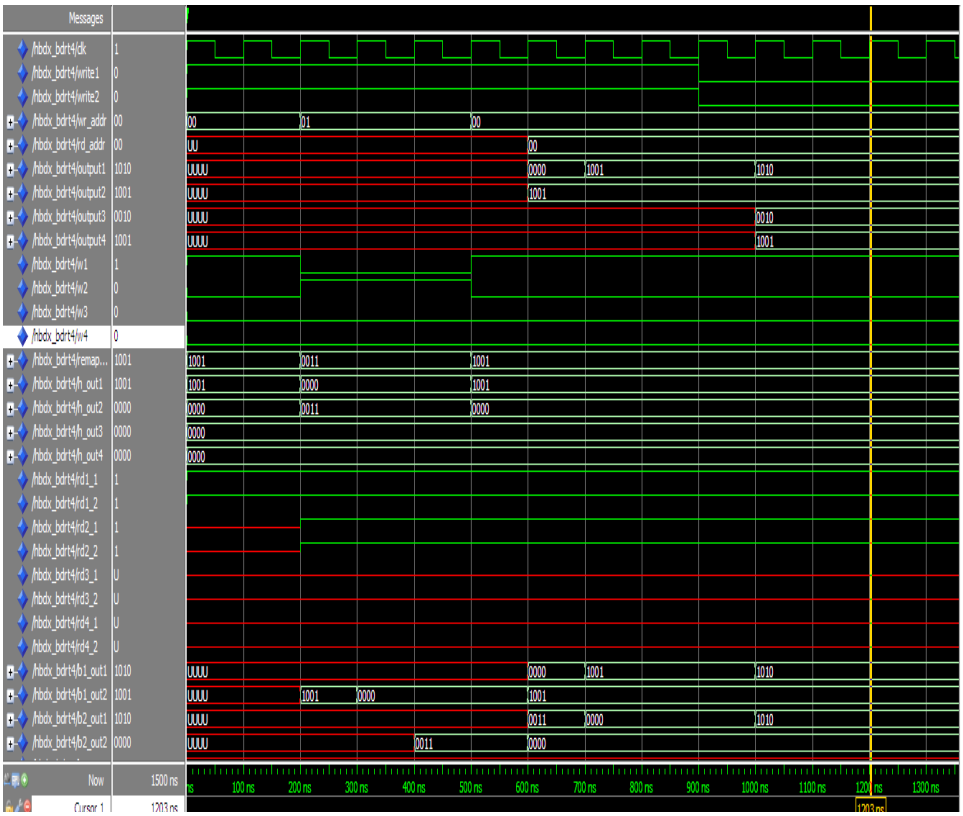


Fig.11 Hash write control

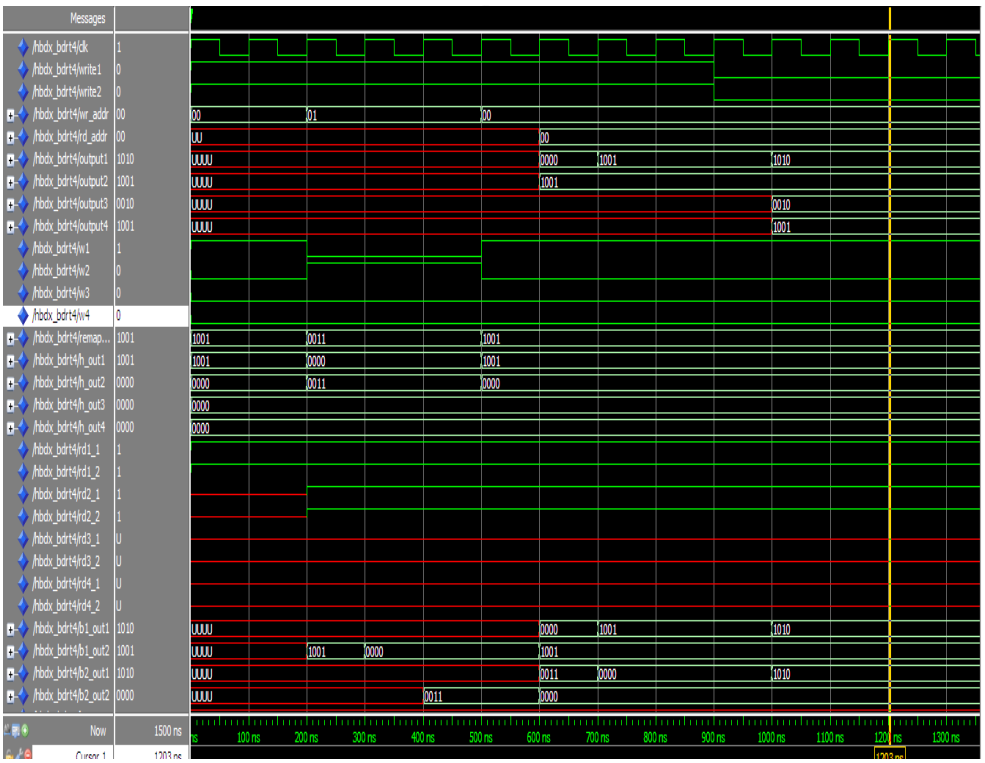


Fig.12 HBDX_BDRT_4R2W

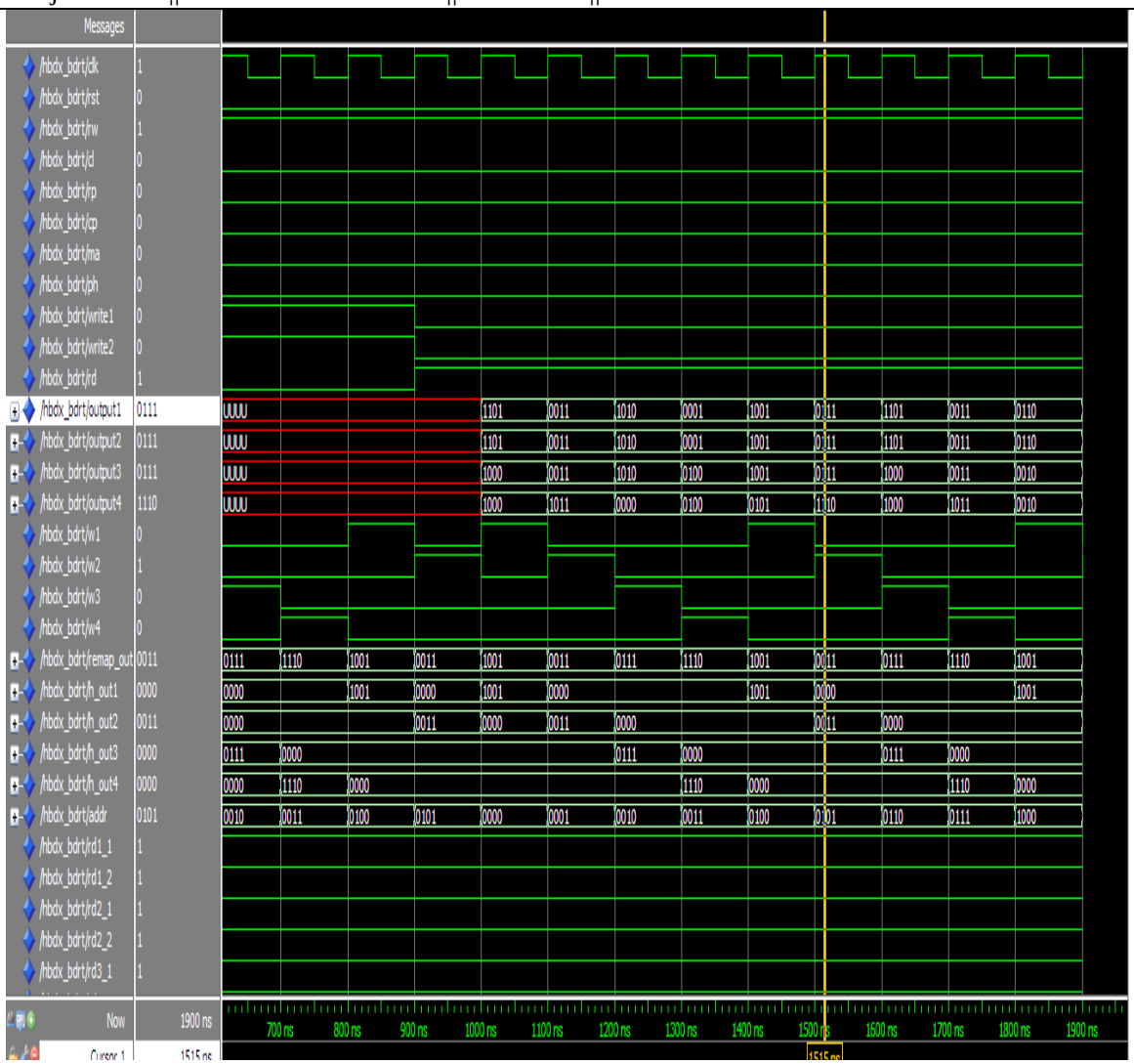


Fig.13 HBDX_BDRT_4R2W with Address generator final output

According to the requests fed to the address generator that is whether to access a memory to write or read data from a specified address the normal HBDX-BDRT integrating architecture proceeds. Remap table tracks the location of the address that gets updated with the latest data. The hash write mechanism distributes the write signals correspondingly to the memory modules and finally 4R output is generated, that is 4 reads and n writes are possible with this proposed design.

V. CONCLUSION

An efficient design for multi ported memory in FPGA has always been a promising factor. As technology evolves tremendous changes newer designs that could compensate the limitations of older designs are being introduced. Facing the challenges and modifying the existing designs to achieve a better more efficient and reliable design still continues. Here older approaches are being modified using several newer techniques in the designs proposed in the paper. 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. The proposed design architecture integrating HBDX and BDRT approaches considerably decreases area and contributes an efficient multi port memory design implementation on FPGA. The modification that added an Address generator to the proposed design made the architecture more customized for image processing applications as the addresses that supports specific data access policies for image processing algorithms are generated initially that can be used for image processing. The scan order varies accordingly and the efficient architecture along with this intelligent modification results in an efficient approach of implementing memory on an FPGA that supports image processing.

REFERENCES

[1] B.-C. C. Lai and J.-L. Lin" Efficient Designs of Multiported Memory on FPGA",IEEE,Apr.2016.

[2] C. E. 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.

[3] 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.

[4] 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.

[5] 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

[6] 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

[7] 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.

[8] 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.

[9] P.Deepa, C.Vasanthanayaki, "FPGA based efficient on-chip memory for image processing algorithms,"in Microelectronics Journal, 43 (2012),pp. 916–928.

[10] 10 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.

[11] 11 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.