IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 14, Issue 4, Ser. I (Jul.-Aug. 2019), PP 51-56
www.iosrjournals.org

Multiport SRAM with BIST and Low Power DLL Design on FPGA Implementation

[1]Puneet Bharadwaj [1], [2]Sunil Kumar Shah [2]
[1]ECE, GGITS/RGPV, India
[2]ECE, GGITS/ RGPV, India
Corresponding Author: Puneet Bharadwaj

Abstract: *The clock alignment circuits must provides a phase resolution better than 40ps, produces worst case long - term jitter of less than 250ps peak -to peak. DLL are preferred unconditional stability, lower phase-error accumulation and faster locking time. The operating ranges is 400MHz-1.4Ghz, and 3.52mW is consumed at 1.4Ghz.The measured locking time ranges from 16 to 25 cycles over entire operating frequency ranges.*
Keywords: *Multiport RAM, DLL, field-programmable gate Array(FPGA),Block RAM*

Date of Submission: 22-07-2019 Date of acceptance: 07-08-2019

I. Introduction

FIELD-PROGRAMMABLE gate arrays (FPGAs) are broadly used in fast prototyping of multipart digital systems. FPGAs contain programmable logic arrays, usually referred to as slices. Slices can be configured into different digital logic functions. In addition to implementing logic operations, slices can also be used as storage space 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 system on-system memory modules. Synthesizing a large number of memory modules would consume a significant amount of slices, and could therefore result in an inefficient design. The overuse of slices could also lead to a limiting factor on the maximum size of a system that can be prototyped on an FPGA. To efficiently support the system's on-chip memory, modern FPGAs deploy block RAMs (BRAMs) that are hardcore memory blocks integrated within an FPGA to support efficient memory usage in an integrated design. An FPGA usually deploys numerous BRAMs with the same requirement. For example, Xilinx Virtex-7 xc6vlx75tff484-3 FPGA contains 8 word BRAMs, and each BRAM can be configured in two port mode or dual-port mode. Multiport memories, which allow multiple coexisting reads and writes, are frequently used in various digital designs for 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 greater quantities of instructions require even more access ports. The common cache system among multiple soft processors on FPGA should support multiple simultaneous accesses. Time multiplexing and task scheduling are the primary solutions to support multiple accesses. However, these schemes would usually cause more complex designs with numerous corner cases and therefore requiring extra efforts of verification. Having a nonspecific multiport memory module can greatly ease the design when a system necessitates the provision of parallel data accesses. Designs of multiport memories that leverage BRAMs have been proposed to attain superior utilization of FPGA resources as well as system performance. BRAMs in an FPGA can support two access ports that can be used as either a read or a write port. This fundamental specification requires extra effort if designers would like to implement a storage module that supports more concurrent access ports than the existing BRAMs. Compared with the multiport memory designs that only make use of slices, the previous approaches with BRAMs have established less total equivalent area while achieving desired frequencies. The limited quantity and capacity of BRAMs have led to serious concerns for designers when implementing multiport memories on an FPGA. Using an unnecessary amount of BRAMs for multiport memory could seriously restrict the usage of BRAMs for other parts of a design. Our previous work introduced a two reads, one write (2R1W) module based on XOR-encoded values and achieved better performance with fewer BRAMs. Nevertheless, these designs still occupy more than half of the available BRAMs on some modern FPGAs. This issue becomes pertinent when the design requires huge internal storage capacity. Therefore, having a more capable design of multiport memories remains the primary design concern for modern digital systems on FPGAs. First, we must prototype a brand new method using a 2R1W module as either a 2R1W or a 4R module, denoted as 2R1W/4R memory. Secondly, by exploiting the adaptable usage mode of the proposed 2R1W/4R memory, this paper proposes a hierarchical design of 4R1W memory. This hierarchical 4R1W design requires 25% fewer BRAMs than the alternative approach of duplicating the 2R1W module used. Memories with more read/write ports can be extended from the proposed 2R1W/4R memory and the

hierarchical 4R1W memory. For complex multiport designs, the proposed BRAM-efficient approaches can achieve higher clock frequencies by alleviating the complex routing in an FPGA. For 4R3W memory with 8 word depth, the proposed design can save 53% of BRAMs while at the same time improve the operating frequency by 20%. This paper is organized as follows: a discussion of previous design approaches of multiport memory on FPGA followed with a proposition for designing 2R1W/4R memory and the hierarchical 4R1W memory that achieve more efficient multiport memories on FPGAs.

II. Built In Self Test

2.1Built in Self Repair Solution Review of DWL and DBL Architecture

The main objective of these techniques is to reduce dynamic power in MPTIPORT SRAM by falling of signal bit line capacitance. Let us assume 6 SRAM is divided into several bit lines. These bit lines are then divided into sub bit lines in order to decrease bit line capacitance. The fundamental architecture of the DWL technique is shown in Fig. 1 for an m × n memory array. Here, the memory array has m words (W0−Wm−1) and n columns (C0−Cn−1). The memory row is partitioned into b row blocks (each contains n/b memory cells) and b switching gates which are added to each row in the memory. Every switching gate is controlled mainly by two inputs—the row select line and the row bank select (RBS) line. The n/b columns proscribed by the same RBS line (RBSi, $0 \leq i \leq b-1$) constitute a row bank. For example, the memory array in Fig. 1 contains b row banks (RB0 − RBb−1). Likewise, as shown in Fig. 1, the memory column is partitioned into a column blocks (each contains m/a memory cells) and a number of switching transistors which are added to memory column. All switching transistors can be described by the column bank select (CBS) lines (CBSi), $0 \leq i \leq a-1$. The m/a memory rows can be described by the same CBS line from a column bank (CB0−CBa−1). The alienated structure does not

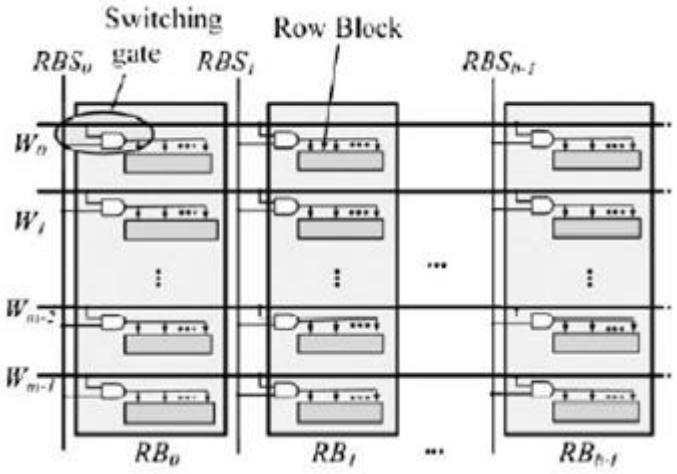


Fig. 1 Divided word line architecture

fault in tolerant applications. However, adding additional rows and columns into the memory array must bemitigated through DWL and DBL techniques. Redundancy analysis assigns a new content accessible memory (CAM) to addresses in the faulty block.

2.2 Memory Architecture for Repairable Multiport RAM

The mixture architecture can be generated by applying both DWL and DBL architectures in a memory array. The fundamental diagram of the memory arrangement can be partitioned into two row banks (RB0 and RB1) and two column banks (CB1 and CB2). There are three SR sand SCs which are also added and partitioned into blocks. The divided array is an overlapped area of a row bank and a column bank ($DA_{(i,j)}$, where is the row bank index, $0 \leq i \leq b-1$, and j is the column bank index, $0 \leq j \leq a-1$). For example, in Fig. 2, four DAs (DA00, DA01, DA10, and DA11) are contained. The basic diagram below is of DLL based multiport mixed architecture. In the diagram, cross marks within the memory array indicate faulty memory cells which are supposed to be replaced with the spare row blocks or spare columns blocks. Each row block/column block in the unnecessary rows/unnecessary columns can be used to repair the faulty row block/column block in the same row bank/column bank. Such auxiliary row blocks/column blocks are called local spare row blocks/local spare column blocks. When making an allowance for group faults, the predicament becomes more severe. By overcoming gathered faults in the global spare row block global spare column block one can reduce the area overhead.

III. Design Flow

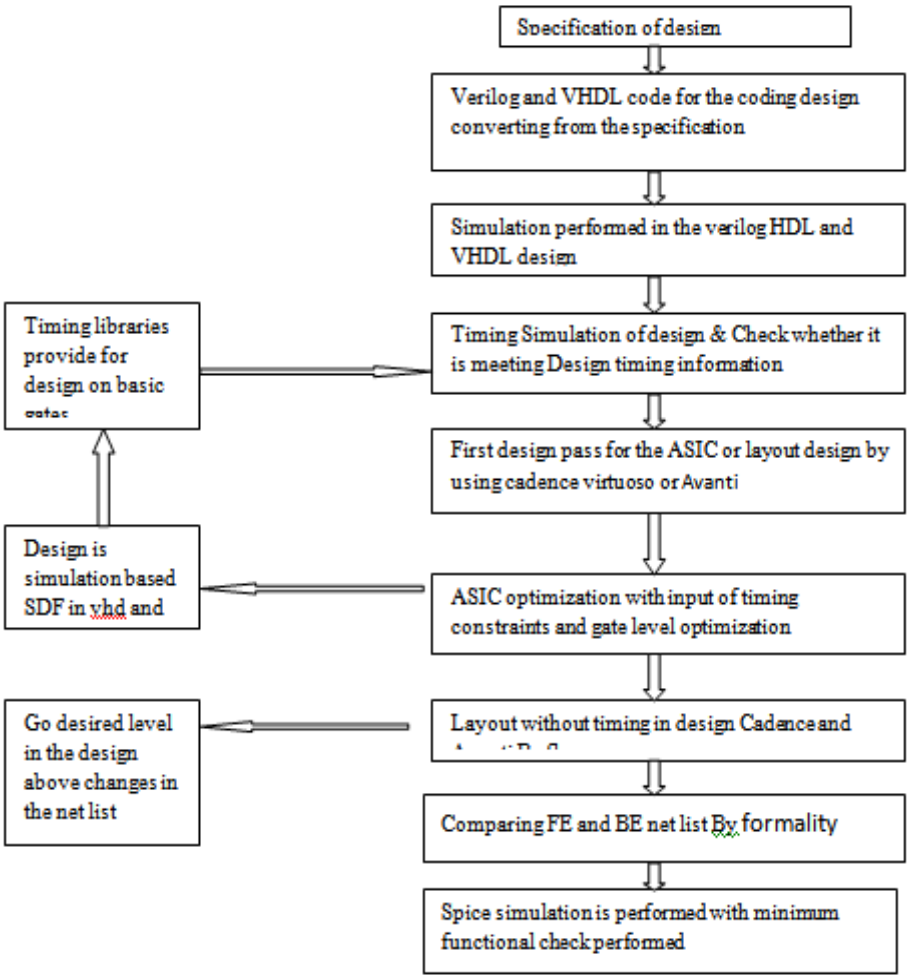
3.1Flow Chart 1

The use of conventional design flow starts from the specification of design written in text format and it is converted into Verilog and VHDL code which is used to simulate the behavior. The design is inserted into the design compiler tool where code modifies the design according to the synthesis design guidelines and then applies the proper design constraints. From there we generate a netlist for invigilation. The next step is to perform static timing determination or static timing analysis. In the primetime analyzer, the pass timing analysis is used in the BE design and then Cadence, an Avanti design tool. Here the Asic design forms a layout. Afterwards netlist utilizes the primetime analysis tool to complete timing analysis by comparing the netist of FE and BE. This is helpful if any design changes are made in any of the steps of the flow chart. Afterwards, spice simulation is performed with a series of minimum functional checks.

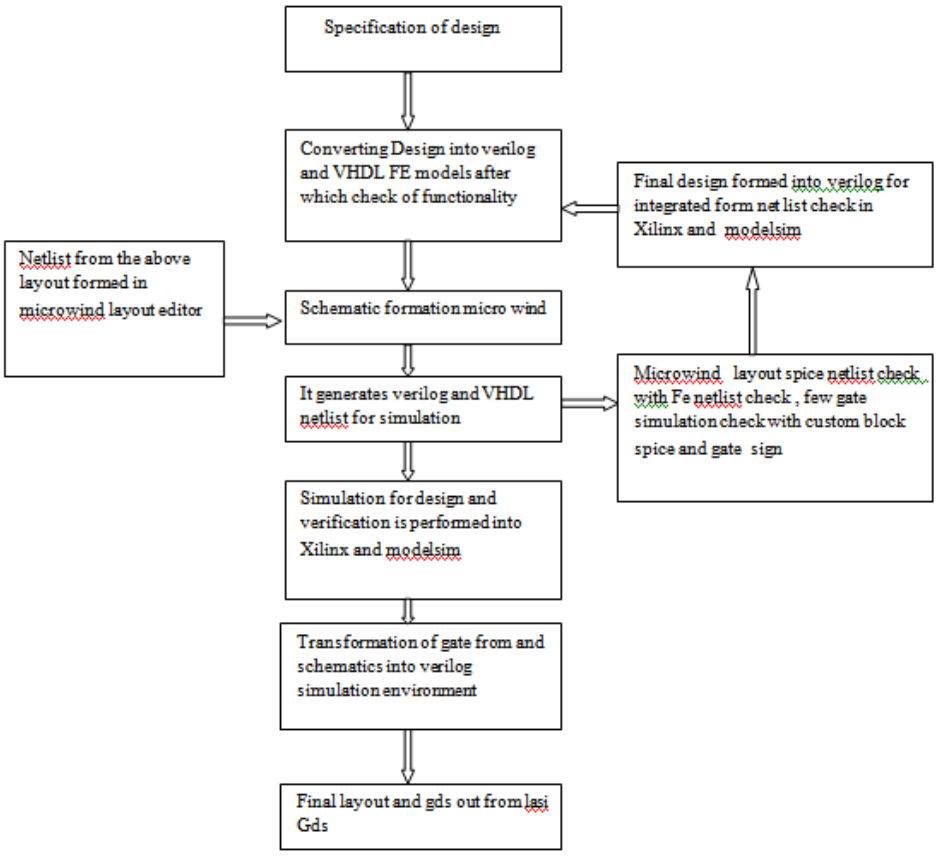
3.2 Flow Chart 2

We have a change in convention design flow using different design tools. Here we used modelsim and Xilinx for the behavior simulation. Xilinx uses timing and power information and then the lasi layout tool outputs the design. We have also used mircowind and LTspice for mixed digital and analog design.

Flow Chart 1



Flow Chart 2



IV. Phase Blending Circuit

However, it is not sufficient to use equal-sized inverters for the blending. A simple model can be used for determining the ideal relative sizes of the two phase-blending inverters to ensure that the phase of the Qcd lies directly between that of Qc and Qd. The model is an approximation of two inverters with two simple switched current sources sharing a common resistance-capacitance (RC) load. For two rising edge input signals separated in time by td, the model yields the equation

$$vQcd(t) = vdd + RI[w.u(t)\left(e - \left(\frac{t}{rc}\right) - 1\right) + (1-w)u(t-td)(e - \left[\frac{t-td}{rc}\right] - 1]$$

where R is the total resistive load, C is the output capacitance, I is the total pull down current of the phase-blending inverter, u(t) is the unit step function, and w is the phase blending inverter relative size ratio where W = WC/(WC+WD) which is the ratio of the devices width in the inverter C to the total device width in both inverter C and D which is the sum of the two decaying exponential terms.

V. Figures And Tables

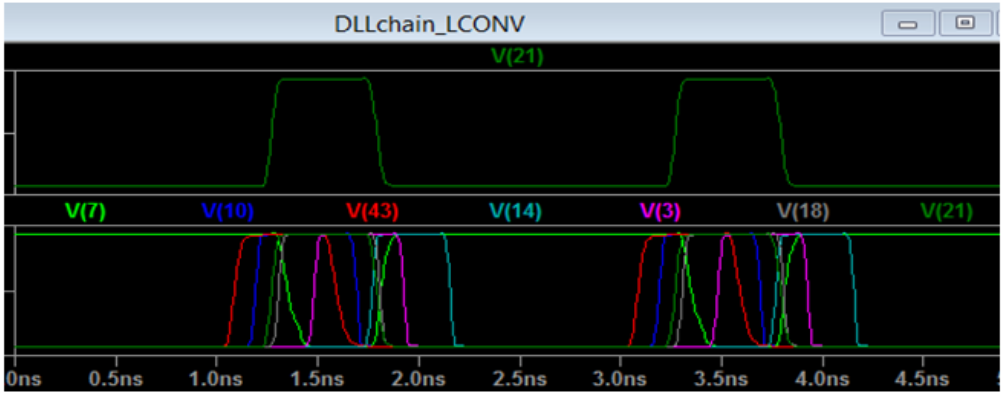


Fig. 2 Time shifting of generated frequency to match input waveform

Architecture	Digital feedback	CSD	SAR Algorithms	Cyclic-locking loop	Cyclic-locking loop (proposed)
Process	0.13um	90nm	0.13um	65nm	65nm
Supply	1.2V	1.2V	1.2V	1.1V	1.1V
Operating freq	400-800Mhz	6.7-1.24Ghz	30MHz-1Ghz	400-800Mhz	800Mhz-1.4Ghz
Locking time	75-374cycles	15cycles(w/RDL)	<= 42cycles	38-41cycles	<=25cycles

Table 1 Performance Summary and Comparison

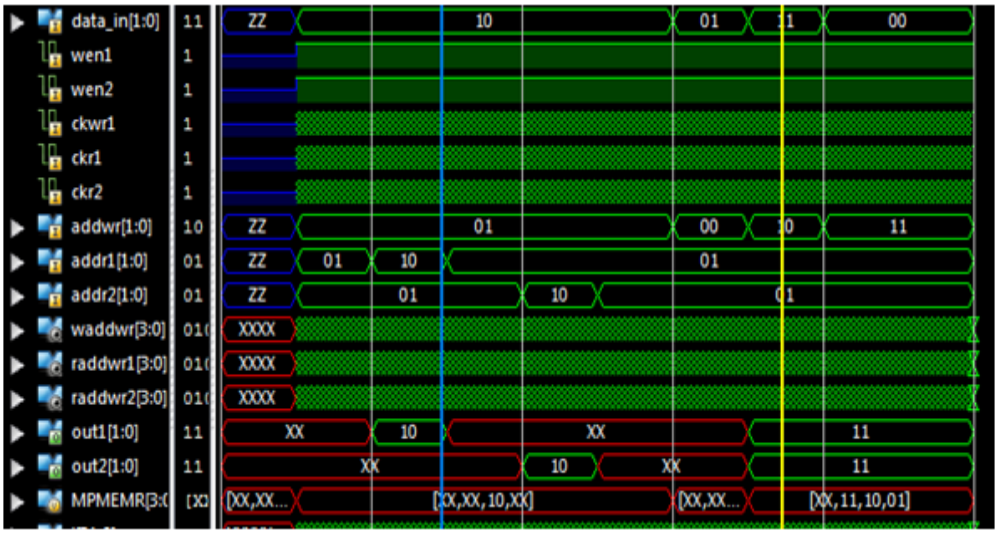


Fig. 3 Memory4x2 simulation result

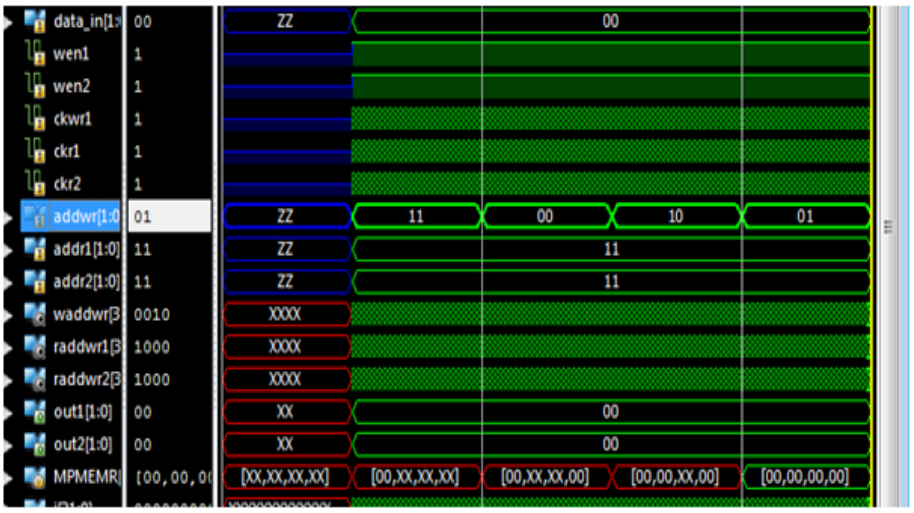


Fig. 4 Memory4x2 simulation result

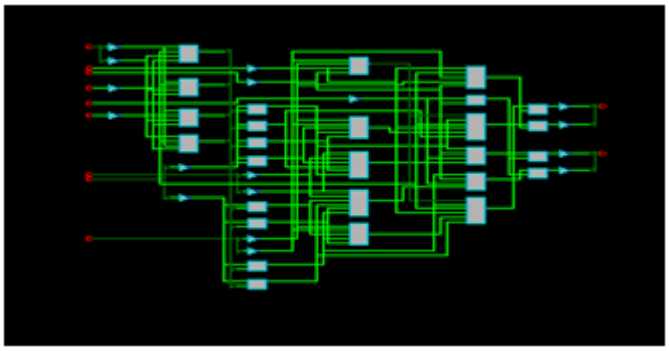


Fig 5 FPGA schematic of memory 4x2

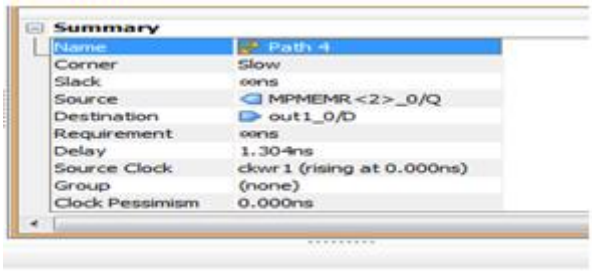
Table 2 of utilization fpga building bocks

Part: xc6vlx75tff484-3

Resource	Utilization	Available	Utilization
Register	8	93120	1%
LUT	11	46560	1%
Slice	6	11640	1%
IO	17	240	7%
OLOGICE1	4	360	1%

Implemented Timing

Table 3 Access time of memory 4x2



VI. Conclusion

In this paper we compared data generated from 40 to 25 cycles of incoming frequency. We found the maximum value of access time in multiport RAM design by implementing bubble sort. This paper proposes a design technique for integrating DLL, BIST, and multiport RAM. We achieved low power DLL operation at 1.1 V and a system clock frequency of 1.4 GHz. In the future we would want to explore different sorting techniques in multiple computer algorithms.

References

[1]. 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.
[2]. 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.
[3]. C. E. Laforest, Z. Li, T. O'Rourke, M. G. Liu, and J. G. Steffan, "Composingmulti-ported memories on FPGAs," ACM Trans.Reconfigurable Technol. Syst., vol. 7, no. 3, Aug. 2014, Art. no. 16.
[4]. 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
[5]. 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
[6]. 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.
[7]. H. E. Yantir and A. Yurdakul, "An efficient heterogeneous register file implementation for FPGAs," in Proc. IEEE Int. ParallelDistrib. Process. Symp. Workshops (IPDPSW), May 2014, pp. 293–298.
[8]. H. E. Yantir, S. Bayar, and A. Yurdakul, "Efficient implementations of multi-pumped multi-port register files in FPGAs,"inProc.Euromicro Conf. Digit. Syst. Design (DSD), Sep. 2013, pp. 185–192.
[9]. 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.

Puneet Bharadwaj" Multiport SRAM with BIST and Low Power DLL Design on FPGA Implementation" IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) 14.4 (2019): 51-56.