International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V70I4P231
Volume 70 Issue 4, 360-372, April 2022

Original Article

Hybrid State Analysis with Predictive Model Control on Memory Controller using Machine Learning Algorithms

Vijayalakshmi ch[1], Jaikaran Singh[2]

[1,2]*Department of ECE, LNCT University, Madhyapradesh, India.*

[1]vijji.lnctphd@gmail.com

Received: 13 February 2022 Revised: 20 March 2022 Accepted: 26 March 2022 Published: 30 April 2022

Abstract *- The design perspective of the memories and their implementation have become computational storage for all the different scenarios of applications governed by the current AI market. The feature of low latency devices or hardware for the application on Mobiles, laptops, etc., is implicated with AI technology with improved memory control and power modules encapsulating the output performance. One such model and structural changes have been implemented with the Hybrid model on low latencies with probabilities, also the memory bandwidth of the proposed controller and memory unit utilized in Wireless applications. With the feature of Area, power, and delay, our Design investigates the feature of reliability of the data storage on the memory model and its formulation approach for low latency. An intuitive approach to gate-level Design with flash memories is implicated with predictive memory array structures for the low area and power efficiency.*

Keywords *- Built-in self-test (BIST)**,** Computational Memory Architecture (CMA), Design for testability (DFT).Field-programmable gate array (FPGA), Processing Elements (PEs).*

1. Introduction

There was a huge growth in the want for laptop processing capability during the preceding decade. Because of the increasing need for computing capacity, heterogeneous systems, comprised of a range of high-performance processing devices, have been created in response to this demand. [1] Field-programmable gate array (FPGA) and other similar components are used with standard CPUs to create a high-speed network. It is possible to considerably increase the performance of computationally intensive applications by merging numerous Processing Elements (PEs), which provide a considerable speed boost. [2] When examining the numerous aspects of the design views and efficiency of the homogeneous system, the energy consumption of the homogeneous system would entail a significant application value. Reconfigurable computing has progressively risen in popularity since its introduction at the beginning of the previous decade to speed up computationally intensive applications. FPGAs are a particularly attractive choice if you're looking for a high-performance solution for the implementation and realization of the hardware of computationally intensive applications. The ability to program or reconfigure a chip to perform a different function for each application distinguishes it as programmable or reconfigurable. When used in conjunction with FPGAs capable of executing specialized application processors, the flexibility of general-purpose processors may be increased. The hardware designer must map hardware to FPGAs to customize the hardware for a specific application. With the selection of various domains, FPGA has emerged as a key component of sleek Design and scalable architecture. FPGAs have become more popular due to their ability to include a large number of parallel arithmetic and logical units and their high memory bandwidth[3]. When a multi-core CPU and FPGA devices [1–5] are combined, the goal is to increase the speed of computationally demanding applications. When developing FPGA-based total hardware accelerators, the construction of a reliable device for facts switch among the external memory and the hardware accelerator itself is the toughest task to deal with. Generally speaking, the time had to retrieve data from external memory is the important thing stumbling block in most FPGA-based systems.

Machine learning, deep learning, photograph/video processing, and big data are all examples of compute-intensive programs that need to be used when coping with big amounts of records (or facts). [4-5] Because FPGAs do not have sufficient on-chip memory to fulfil the workload, the records to be processed at the FPGA should be stored and retrieved from off-chip memory. It is much slower in the case of compute-extensive packages to get admission to information from external memory than utilizing FPGA-based total hardware accelerators. Consequently, the confined memory bandwidth has a detrimental influence on the overall performance of hardware accelerators, as previously stated.

2. Existing Model

2.1 ASP-MC

[16] This model from the design feature represents the specific scenario where a single application is only intended for the device. Considering the figure, we have the Read and write address feature with 64bits is the implication with the Data unit and control unit. To access the data governing the different process models with elements based on the condition aspects that interface the data and control unit. The process elements are key aspects for control and R/W operations for each set of core modules introduced for the application-specific processing.

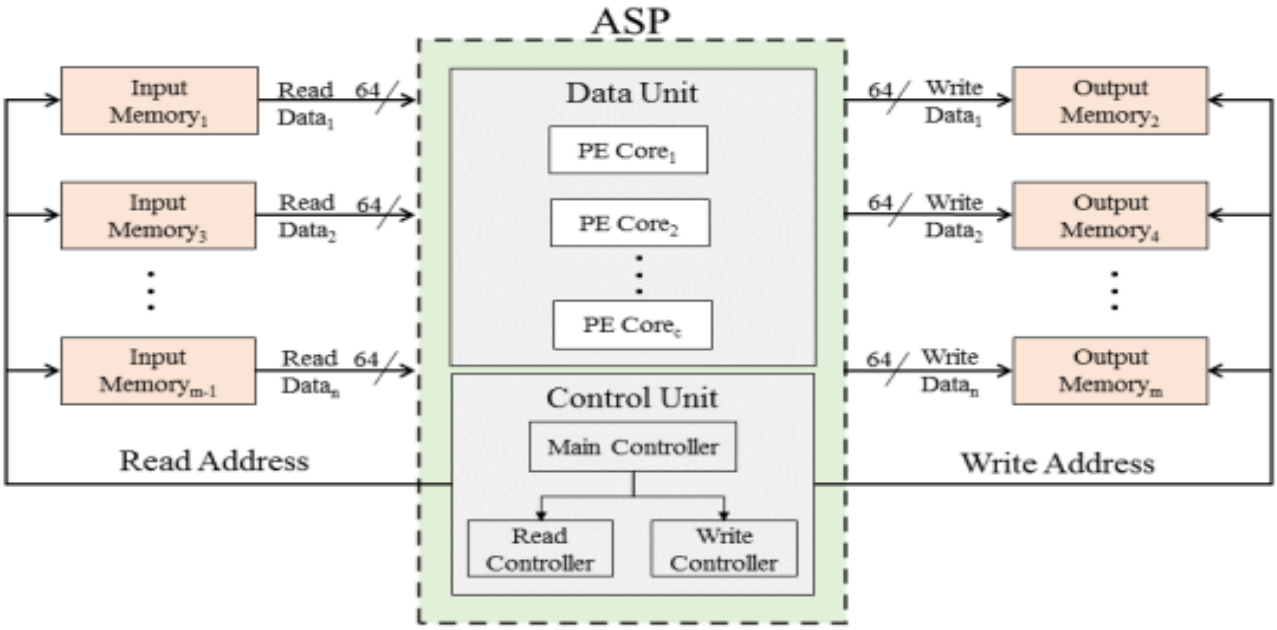

Fig. 1 ASP-MC operational diagram

[17] The output memory keeps the consequences produced after the FPGA processes the information. In the case of reading controllers, they're in charge of transmitting the alerts required to read data from many input memory locations at an identical time from the equal area. In a similar vein, the write controller sends a signal to the write controller, teaching it to write data to several output memory locations simultaneously. The main controller is in charge of controlling the Read and write controllers, respectively. In the case of the basic and sophisticated cores, ASP demonstrates that they can process 32-bit data, but the memory on the FPGA board can process 64-bit data. When using a single input and output memory, PE cores are had to method 64-bits of records from the input memory and write 64-bits of records to the output memory, assuming that our structure contains simply one input and one output memory. As an effect, c cores will be needed to investigate the information for a total of m memories, resulting in a complete of m memories. In this example, it's possible to have a 2:2 Memory model on the same laptop (m should be even). Figure 2 depicts the memory structure we utilized in our studies, with two inputs and outputs (right).

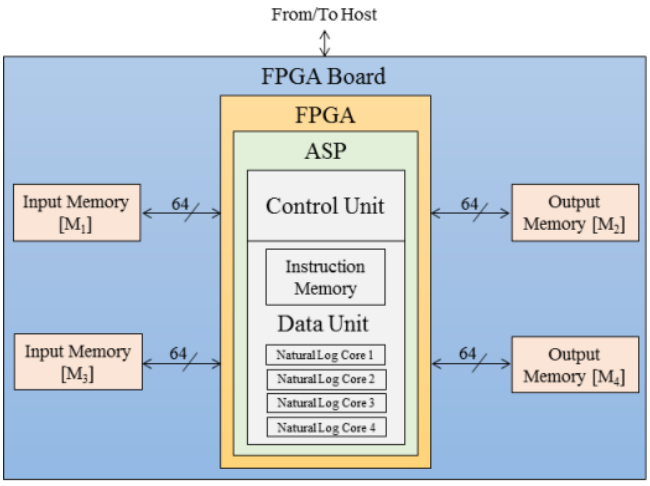

Fig. 2 Memory architecture that we used in our research

The one-input, one-output memories structure is required to develop the multiple, two-data memory architecture [17]. Four Natural Logarithm function cores are contained inside the two-input, two-output memory architecture frame. There are many procedures performed by it, including taking four 32-bit floating-point values from the

input ram and, after they have been processed, outputting four 32-bit floating-point values to the output memory after they have been processed.

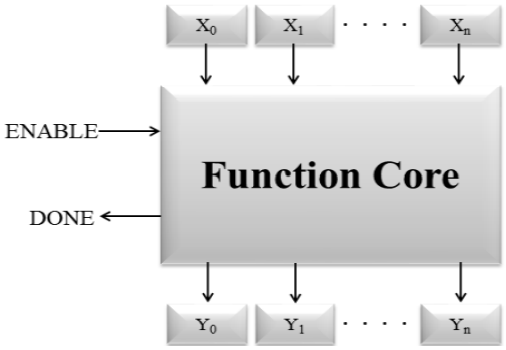

Fig. 3 The traditional floating-point operation core

[20] The consequence is that it is adept at accessing data from and into memory up to 128 bits in size. Compared to the one-input, one-output memory design, the multiple, separate memory architecture delivers two times the speed up. Specifically, data processing speed is doubled when a memory architecture with two inputs and two outputs is used compared to a system with one input and one output. The four-core architecture requires twice the amount of hardware to execute all memory interfaces and lower overall processing time by a factor of two. With a floating-point value of 1, the traditional floating-point operation core is shown in Figure 3.

[21] The functionality of the design model where our feature has one or more floating-point inputs of single precision and one or more floating-point outputs of single precision; is true for all of the functions in our project. The Enable and Done pins provide a basic structured control interface that is easy to use. The ASP can communicate with any of the cores with relative ease. When valid data is available on the inputs, the Enable Pin indicates this, and when valid data is present on the outputs, the Done Pin indicates this. The rudimentary floating-point arithmetic function cores are concatenated together to form more advanced floating-point arithmetic function cores. It is possible to produce a basic netlist that defines the connections between the primitive function cores by using a tool called FUNCOREGEN, which was developed in our lab and is available online. Using a set of input parameters, Fun-core-gen automatically creates the hardware description of basic function cores that could be directly built from the hardware description generated by the software.

2.2 DDR-RAM

[22] The data rate doubling of the synchronous dynamic ram memory is emphasized in this design model, which utilizes low power and high-speed requirements to achieve this. To work on such a unique model design, it was necessary to address certain requirements for the design circuit to function at low power and high speed independently. In accordance with the complexity of the designed controller, encoder, and decoder used on the suggested design model, as indicated in the figure, the condition would vary.

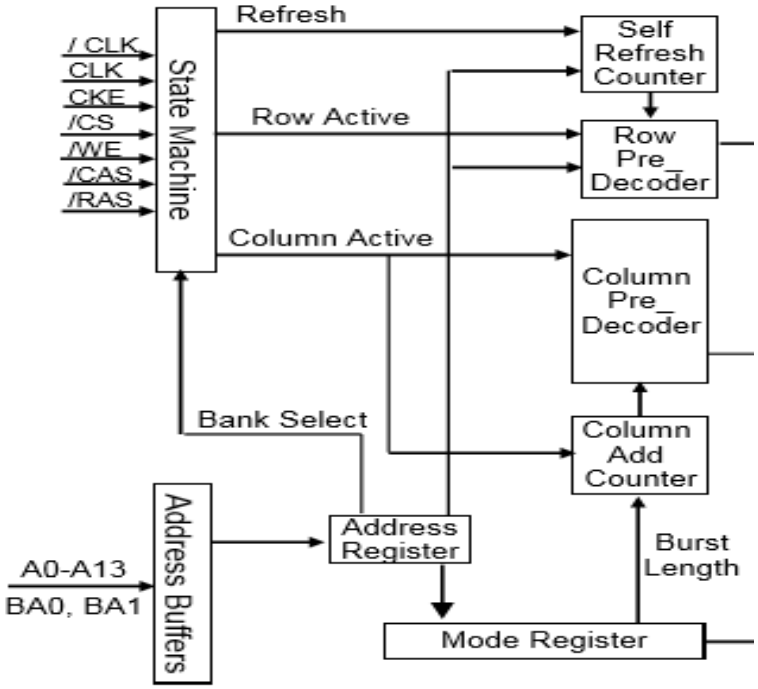

Fig. 4 Representing the Component Diagram for initiating the Low power model for DDR RAM

2.3 Design Units Description For The SRAM-Memory Design Model

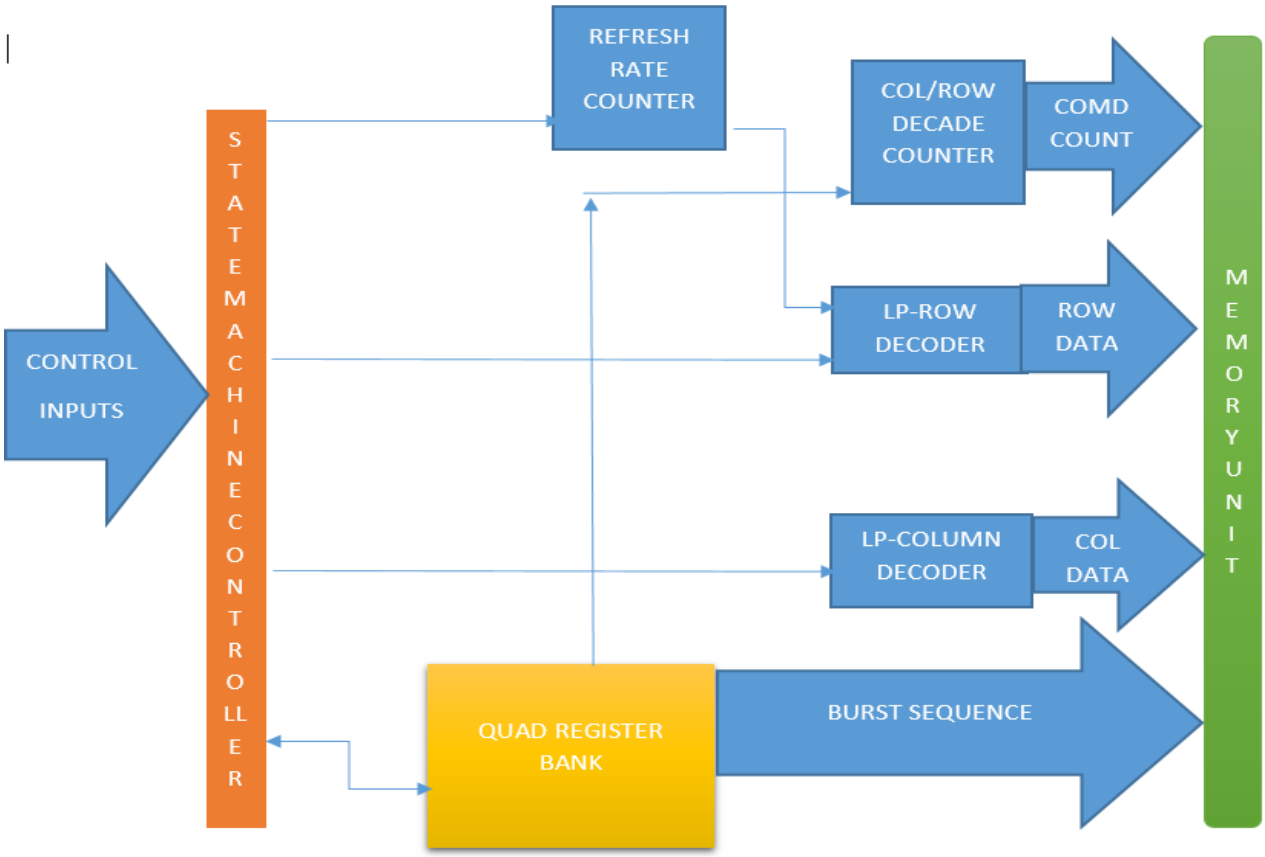

Fig. 5 SDR SDRAM controller design block diagram

The SDR SDRAM Controller comprises four main modules: the SDRAM controller, the control interface, the direction, and the information interface modules. The SDRAM controller module is the highest-level module, and it is responsible for launching the three lower-level modules and bringing the whole plan together. Directions and associated memory addresses are acknowledged by the control interface module from the host, disentangling the order and forwarding the request to the order module. In response to directions and addresses received from the control interface module, the direction module generates the best feasible directions to the SDRAM. During the WRITEA and READA directions, the information interface module is in charge of handling the specified information tasks.

2.4 Low Power Analysis on DDR_SDRAM

- To provide perfect synchronization on the design Unit to impart better data synchronization.
- Efficient decoder scheme to provide less time to execute the command and perform synchronization and fast data transfer

As a result, our Design uses three different techniques to initiate the low-power synchronized model to analyze the instability observed as a result of the use of synchronization based on Toggle, Gray, and FIFO Synchronizer, all of which have an impact on the relationship of Metastability. Since our Design understands that Metastability is directly related to the meantime failures, which can be predicted accurately depending on the Flip flop utilized, we can leverage this knowledge. Compared to conventional FF, the Design focuses on Dual-D FF, which has superior and higher PAD parametric criteria than the Design.

2.5 Flow Diagram for Simplest Memory Controller

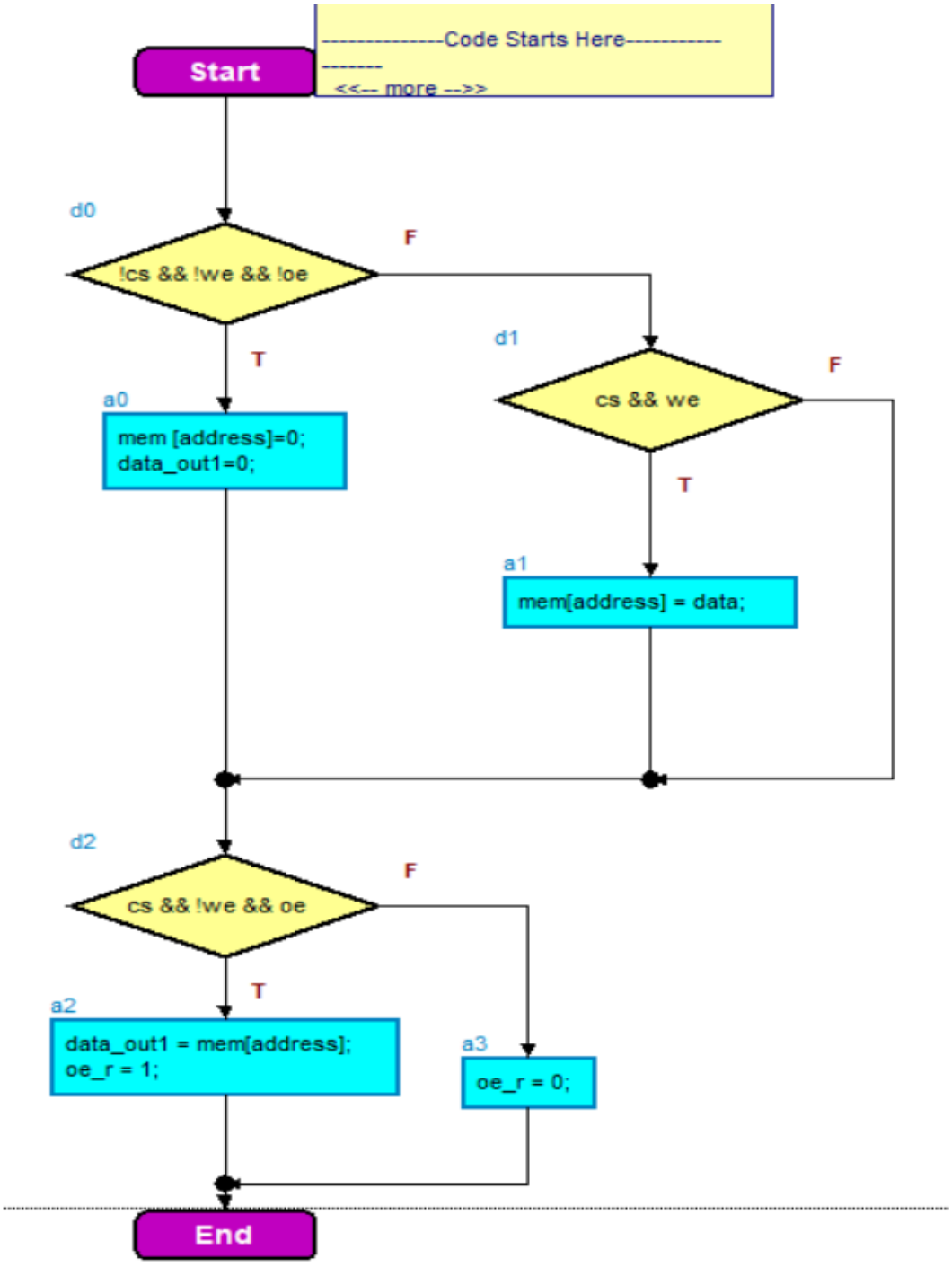

Fig. 6 The Simplest Memory Controller Flow Diagram

Because of this, the Design utilized a finite state model to represent the LP-DDR RAM in accordance with the design requirements. It is anticipated that our proposed study would offer outstanding performance opportunities in both Design and the algorithms used to enhance the low-power capabilities. In this case, the DDR RAM would store the controlling changes seen from the control signals used in modelling the state diagram changes to take advantage of better and quicker reaction times produced on each cycle of the program's execution.

3. Proposed Memory Control Architecture with Low Latency

The design estimation and its features on control architecture are estimated with different performance factors, and one such factor is Low latency. This factor majorly controls the design access speed from memory to bus and other related subsystems that are interlinked. Low latency implicates less delay and hence better speed. On the contrary, multiple algorithms have been established implicating the design aspects of the controller for high speed and low power applications. In the case of understanding the low latency problem, our model has been improvised with a predictive heuristic approach on the data path and data bus for each set of communication processes leading to memory output bandwidth. This model consists of four modules that are implemented and scaled with 8 bit, 16 and 32 simultaneously, as mentioned in figure 7.

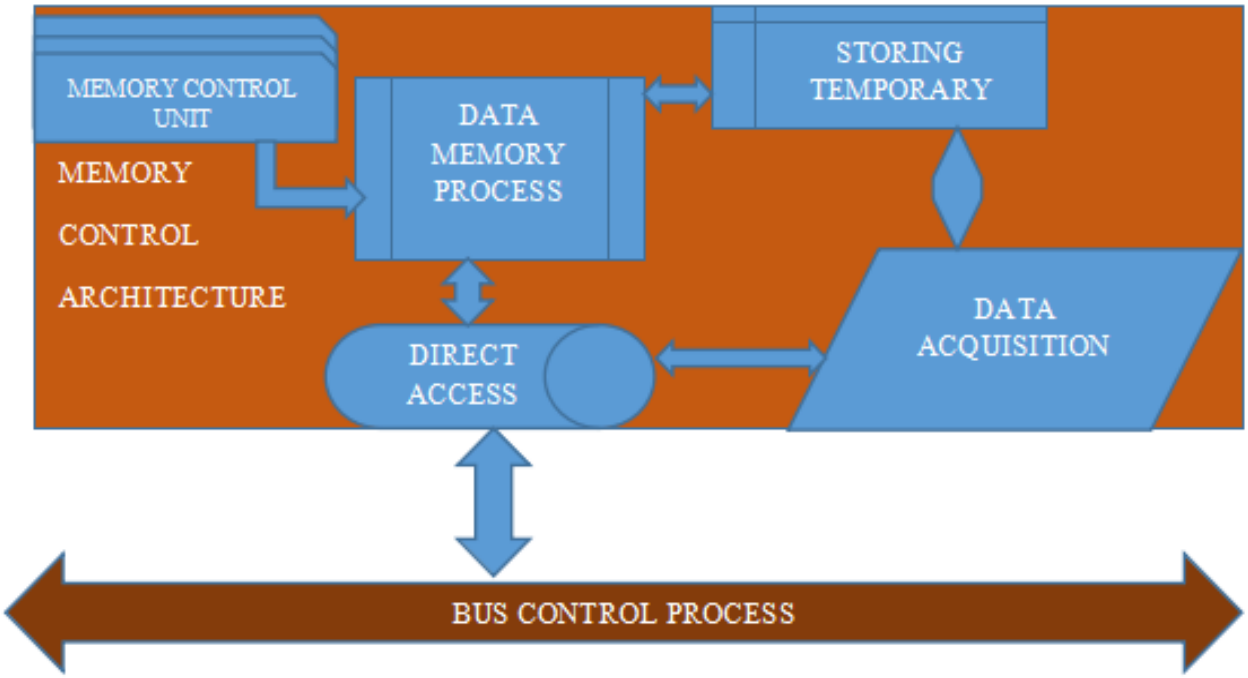

Fig. 7 Representing the Proposed Block diagram for the control process architecture for memory

Figure 7 comprises the control unit, memory process, direct access, data acquisition, with a process control scenario where the design platform for the working scenario is controlled with MCU (memory control unit) designed using HFSPM (hybrid finite-state predictive memory). This predictive concept of the memory consists of the concurrent FSM with predictive algorithms for state analysis and state control data. These two features are estimated by implicating the algorithm as mentioned below:

3.1 Procedure

- Initiate the MPC (model process control) satisfying set of constraints as low latency for the memory controller.
- Implement the process architecture for Memory operations with commands and data feature initialization
- Acquire a new set of specifications according to the access for different external devices.
- Encapsulate the different input commands and data sequences for memory and bus operations.

3.2 Algorithm1

- Input and output width of 16 bits is initiated to process the two-stage address and data transfer.
- Bus width of 24 bit is chosen for direct data transfer.
- To initiate the input and output criteria with a Write address, read address, enables, busy, ready, bank address, clocks and its enables, data-padding with low and high values.
- An improved interface model is implemented with both manual and auto features for real time applications

3.3 PSEUDO Code for Commands and Address

```
Sd_ram_bank_addr=(state[4])?  Sd_ram_bank_addr_r  : command[2:1];

Addr_main  = (state[4] | state == INIT_LOAD) ? addr_read :
  { {SDRADDR_WIDTH-11{1'b0}}, command[0], 10'd0 };
```

4. Flow Diagram for Proposed Memory Fifo Model

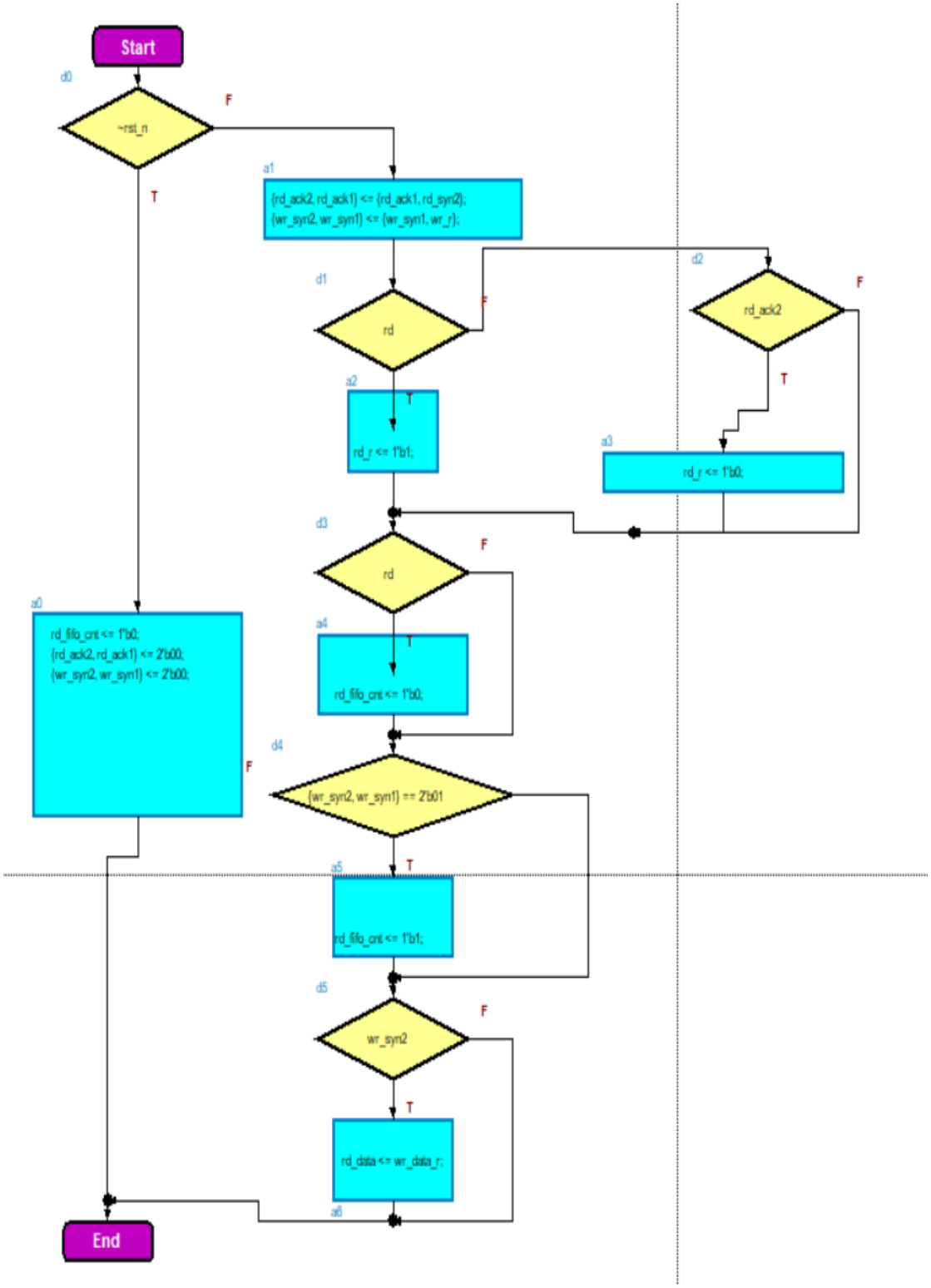

Fig. 8 Representing the Dual-FIFO model and its implementation using read-write conditions

4.1 Algorithm2

1. Input and output widths of 16, 32 and 64 bits are initiated to process the two-stage address and data transfer.
2. Bus width of 48 bit is chosen for direct data transfer.
3. To initiate the input and output criteria with a Write address, read address, enables, busy, ready, bank address, clocks and its enables, data-padding with low and high values.
4. The hybrid Predictive Model interface model is implemented with manual and auto features for real-time applications.

4.2 Hybrid Predictive Algorithm

4.2.1 RFR

A technique that uses "boosting" (adjusting the weight of an observation) (which creates subsets of data from training samples, chosen randomly with replacement). Bagging is a method that Random Forest employs.

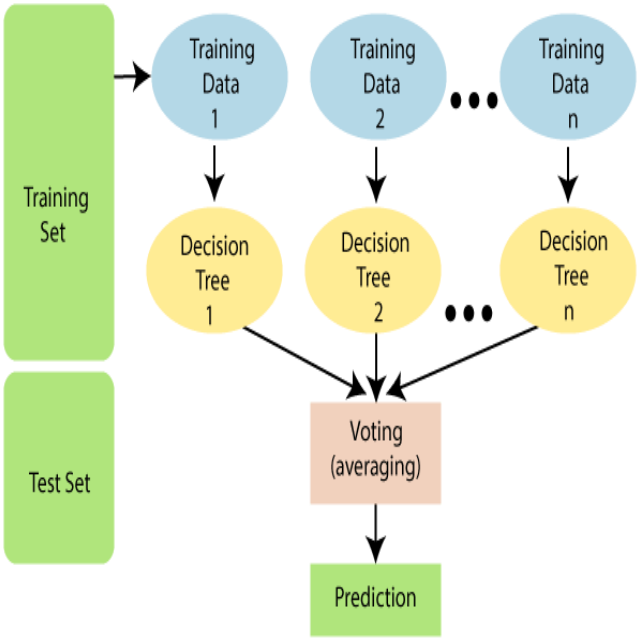

Fig. 9 Representing the RFR algorithm for implementing the Training models and test

4.3 K-Means

One of the most popular high-speed algorithms is K-means, which finds data points based on similarities and places them in distinct groups. This method is used in the clustering algorithm. K-means attempts to identify and group the common features among all the people and groups. This is extremely tough to accomplish with one million individuals.

Pseudo code:

Input:

$D = \{t_1, t_2, t_3 \ldots T_N\}$ *set of elements that have to predicted*

K *−−− Be the design clusters formed*

Output:

Initiate values for $m_1, m_2, m_3 \text{---} m_k$ defining the centroids of each element Estimate the distance d for each element and centroid based on the distance algorithms

Finally, compare the distance of each element and centroids based on the proposed threshold values.

4.4 Probability Solution

On this predictive logic, our Design has to estimate the latency of each of the modules and its functional capacity that governs the Design. The functional features are estimated with the modules utilized and implemented based on Algorithms such as RFR and Kmeans, implicating the mathematical formulations for calculating predictive latency values estimated based on the formula as mentioned below:

4.4.1 Formulations

Let a random probability function (RPF) govern the latency property as defined by the processing time for each module governing the Design. Let P(M) defines the RPF for each module utilzied as:

$$P(M)_t = m1 * P(m1)_{t1} + m2 * p(m2)_{t2} + \ldots mn * p(mn)_{tn} \quad (1)$$

Here $m1, m2, and\ m3 \ldots mn$ implicating the modules and their probabilities as the additive function. To calculate the module's latency and features, our Design has presented the controller, read and Write modules, Temporary storage (Dual-FIFIO), and a predictive interface for latency calculation from the perspective of in and out of the counters for data initializations.

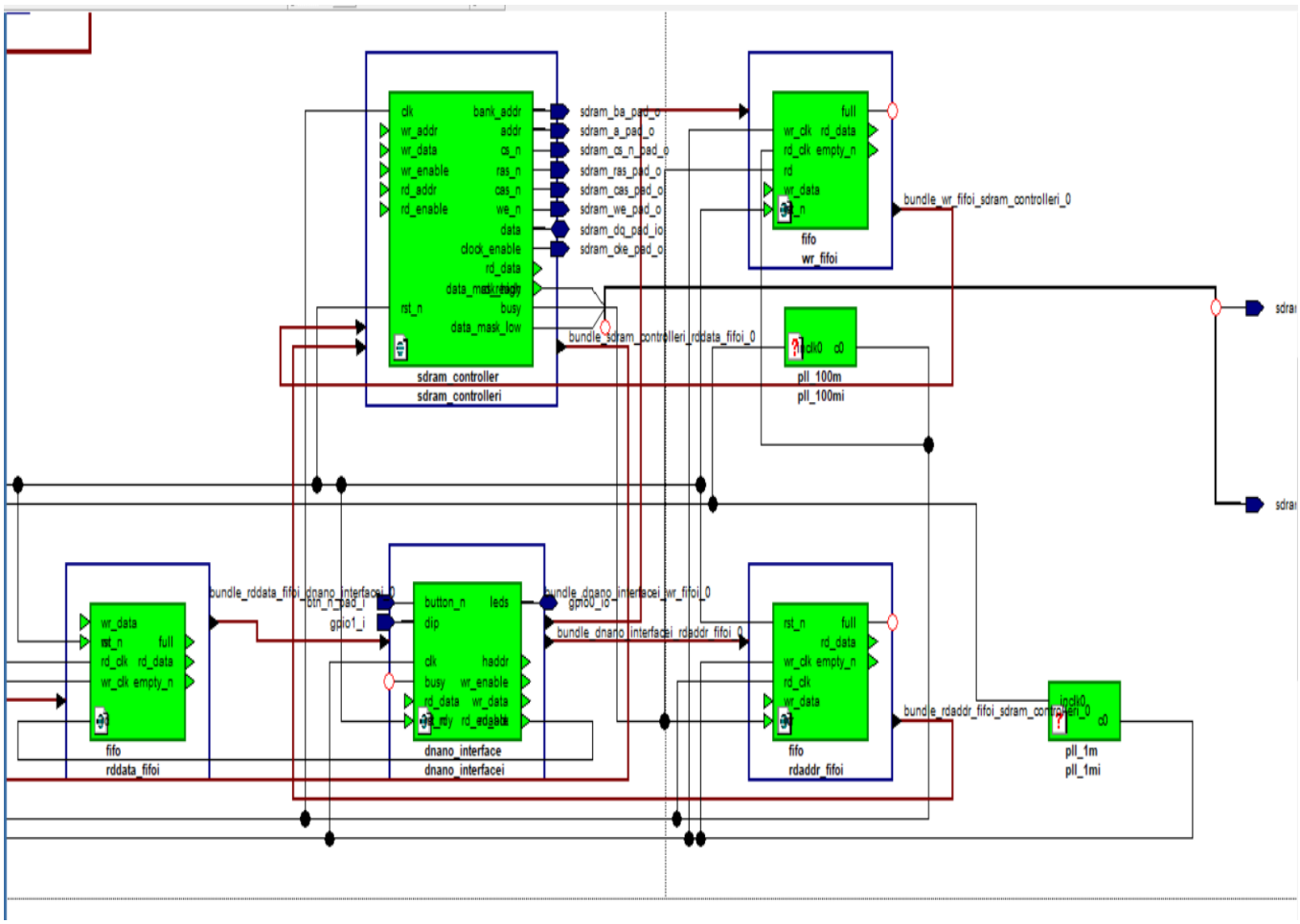

Fig. 10 Representing the Block diagram for Memory controller based on Hybrid Predictive algorithm

Let M_x be the interface implicating the design characteristics as lactency calculations for each input as

Here $M_x$canc be measure with the clock input and its time of setup and hold time as measured:

$$M_x = T_{sx} * ClK_x + T_{hx} * (P - CLk_{sx+hx}) \quad (2)$$

$$p(m_x) = T_{sx} * \left(p(CLK_x)\right) + p(P - CLk_{sx+hx}) * p(T_{hx}) \quad (3)$$

The value $p(T_{hx})$defines the output time probability, which will be dependent on each module connected to the interface. These formulations have been improvised with a probability factor between $\frac{p}{1-p}, to\ \sum_{i=1}^{n}(a_i * CLK_x - b_i * CLK_{sx+hx}$the K-means and RFR conditions mentioned above based on pseudo-code.

4.5 PSEUDO Code

```
state_cnt_nxt = 4'd0;
command_nxt = CMD_NOP;
if (state == IDLE)
     // Monitor for refresh or hold
     if (refresh_cnt >= CYCLES_BETWEEN_REFRESH)
       begin
       next = REF_PRE;
       command_nxt = CMD_PALL;
       end
     else if (rd_enable)
       begin
       next = READ_ACT;
       command_nxt = CMD_BACT;
       end
     else if (wr_enable)
       begin
       next = WRIT_ACT;
       command_nxt = CMD_BACT;
       end
     else
       begin
       // HOLD
       next = IDLE;
       end
```

Fig. 11 Representing the PSUEDO code for IDLE state

```
if (!state_cnt)
  case (state)
    // INIT ENGINE
    INIT_NOP1:
      begin
      next = INIT_PRE1;
      command_nxt = CMD_PALL;
      end
    INIT_PRE1:
      begin
      next = INIT_NOP1_1;
      end
    INIT_NOP1_1:
      begin
      next = INIT_REF1;
      command_nxt = CMD_REF;
      end
    INIT_REF1:
      begin
      next = INIT_NOP2;
      state_cnt_nxt = 4'd7;
      end
    INIT_NOP2:
      begin
      next = INIT_REF2;
      command_nxt = CMD_REF;
      end
    INIT_REF2:
      begin
      next = INIT_NOP3;
      state_cnt_nxt = 4'd7;
      end
```

Fig. 12 Representing the States for Refresh for 6.5 us as per the design characteristics

5. Simulation Results

5.1 Data Reading

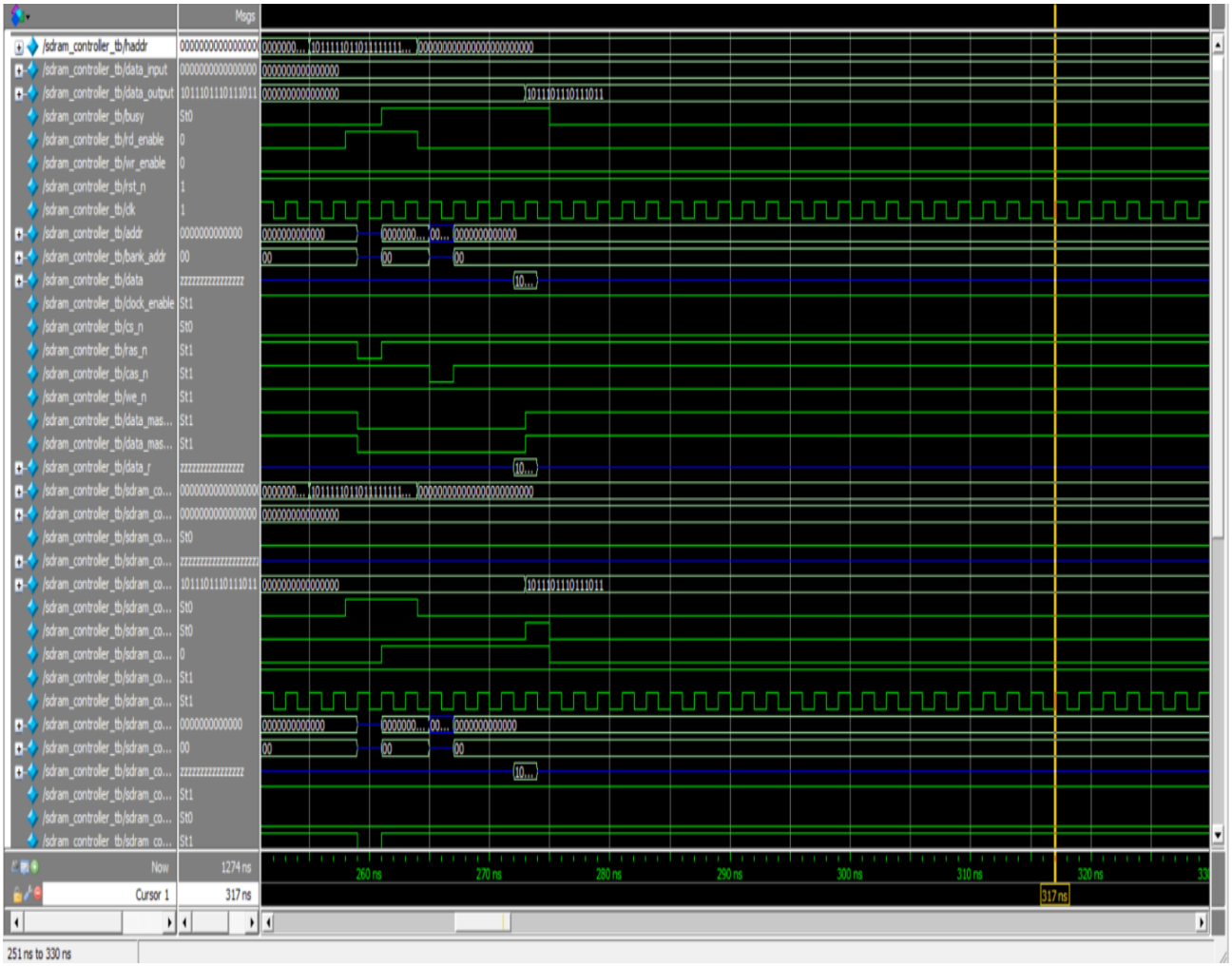

Fig. 13 Representing the Data reading for the input sequence value

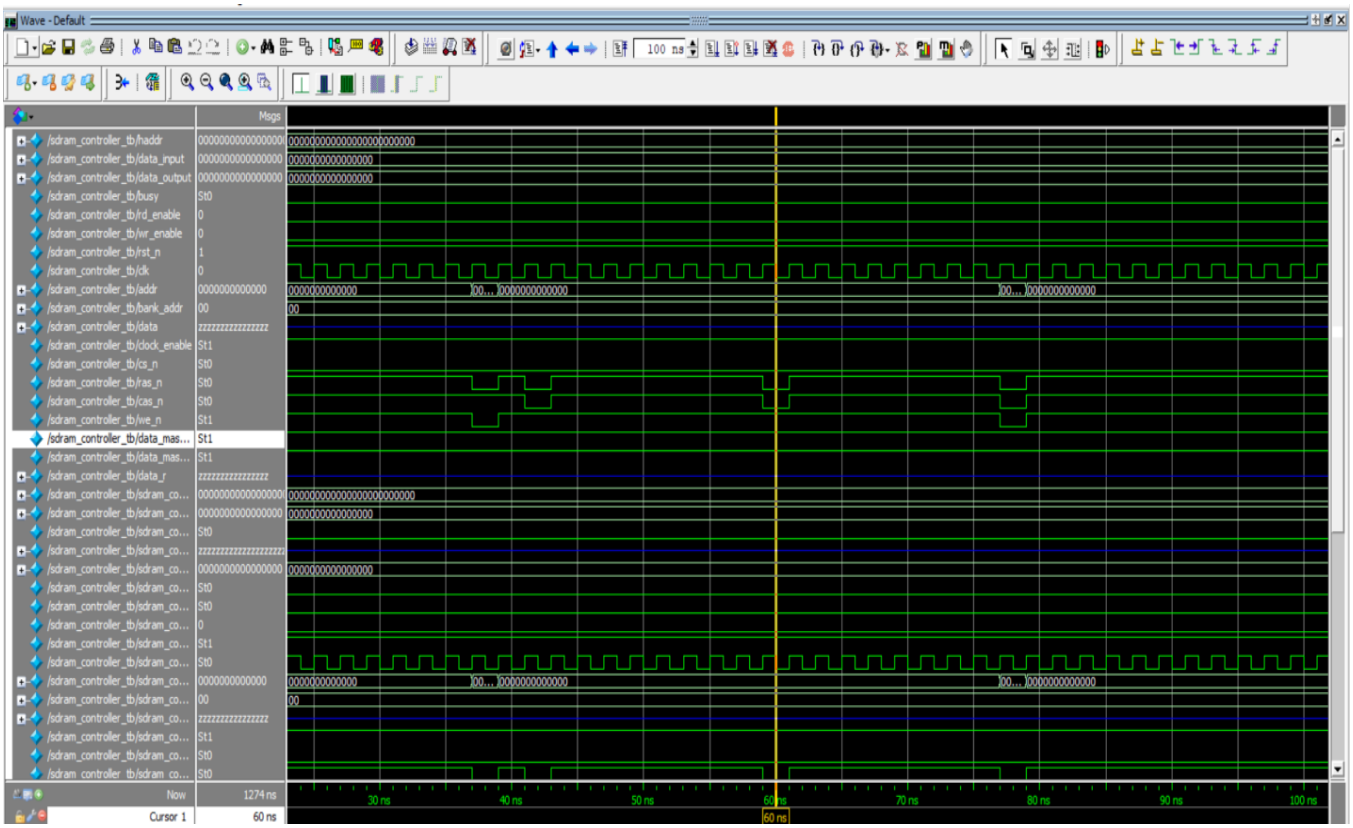

Fig. 14 Representing the data written for the input sequence

Figures 13 and 14 implicate the data reading and writing for the different sequences of the input that have been implicated in the figure representing the model sim output. For each address, our Design was estimated with input values from the test bench as represented with the pseudo-code:

```
initial
begin
   #6 rst_n = 1'b0;
   #9 rst_n = 1'b1;

   #140 haddr = 24'heebced;
   data_input = 16'd6733;

   #5 wr_enable = 1'b1;
   #10 wr_enable = 1'b0;
   haddr = 24'd0;
   data_input = 16'd0;

   #100 haddr = 24'heebced;
   #2 rd_enable = 1'b1;
   #7 rd_enable = 1'b0;
   haddr = 24'd0;

   #9 data_r = 16'haaa;
   #5 data_r = 16'h4ede;

   #10000 $finish;
end
```

Fig. 17 Representing the pseudo-code for the Proposed Controller model

6. Systhesis Results

6.1 Area

Table 1. Representing the Area report for the proposed memory controller (two algorithms)

Resource	Utilization	Available	Utilization %
LUT	235	133800	0.18
FF	201	267600	0.08
IO	26	500	5.20
BUG	1	32	3.12

6.2 Delay

Table 2. The timing report for the design unit for worst slack for both setup and hold times

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	94.574 ns	Worst Hold Slack (WHS):	0.121 ns	Worst Pulse Width Slack (WPWS):	49.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	308	Total Number of Endpoints:	308	Total Number of Endpoints:	202

All user specified timing constraints are met.

7. Comparition Results

Table 3. Representing the overall comparison report with the proposed two algorithms and with the existing Design of the progressive algorithm

S.no	Parameters	Existing Design with a progressive algorithm	Proposed algorithm 1 for memory control	Proposed algorithm 2 for memory control
1	AREA	22.56	8.32	8.9
2	POWER	3.345	2.43	1.153
3	DELAY	286 (ns)	187 (ns)	148 (ns)
4	LATENCY	128 ns	65ns	58 ns

From a design perspective, our Design has depicted different scenarios of the algorithms and their procedure to implicate the Area, power, and delay with latency using hybrid prediction algorithms as mentioned in section 4. As Table 3 provides the comparison table, 61% of the area improvement has been observed with the proposed algorithms. As per the power factor, there is not much difference since the Design desires high-speed configurations with delay and latency. Since employed with machine learning predictive models, algorithm 2 provides maximum performance factors for delay and latency.

8. Conclusion

This Design on the memory controller provides a new step-up challenge to implicate the low latency for each module, and its overall factor based on the circuitry developed. Both algorithms have proved a consistent change to implicate a test and train scenario for the control and latency factors that govern the design performance. Each algorithm representation and its implementation with PADL factors are mentioned in Table 3. Algorithm 1 has better features in all aspects compared to Existing for the area only, even though for algorithm 2, the Design implicates a hybrid predictive approach based on RFR and K-means for calculating latency for each module.

SCOPE

The implementation of the Memory controller is a Hybrid approach to the two scenarios of the Design, which full-fills

The design latency only while the other factors of performance aren't considered. With design structure accuracy, routing models, place,ment features, an FPGA would require a deep learning approach for each set of memory control.

References

[1] Vanya Gupta, Garima Singh and Abhijit Asati, BIST Architecture for Combinational Circuit, 530th International Academic Conference on Engineering Technology and Innovations (IACETI). (2019) 6-7.

[2] Joseph P Elsa and Antony P Rony, VLSI Design and Comparative Analysis of Memory BIST Controllers, International Conference on Computational Systems and Communications (ICCSC). (2014) 17-18.

[3] Charles E. Stroud, A Designer's Guide to Built-In Self-Test in, Kluweracademic Publishers (KAP). 19 (2002).

[4] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch and O. Mutlu, Improving 3D NAND Flash Memory Lifetime by Tolerating Early Retention Loss and Process Variation, Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems New York NY USA. (2018) 106-106.

[5] S. Mittal and J. S. Vetter, A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems, IEEE Trans. Parallel Distrib. Syst. 27(5) (2016) 1537-1550.

[6] L. Zuo, C. Zambelli, R. Micheloni and P. Olivo, Solid-State Drives: Memory Driven Design Methodologies for Optimal Performance, Proc. IEEE. 105(9) (2017) 1589-1608.

[7] K. Mizoguchi, T. Takahashi, S. Aritome and K. Takeuchi, Data-Retention Characteristics Comparison of 2D and 3D TLC NAND Flash Memories, 2017 IEEE International Memory Workshop (IMW). (2017) 1-4.

[8] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch and O. Mutlu, Improving 3D NAND Flash Memory Lifetime by Tolerating Early Retention Loss and Process Variation, Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems New York NY USA. (2018) 106-106.
[9] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers and Z. Zhang, High-Level Synthesis for FPGAS: From Prototyping to Deployment, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 30(4) (2011) 473-491.
[10] S. S. Jadhav, C. Gloster, C. Doss, Y. Kim and J. Naher, A Remote Field-Programmable Custom Computing Machine Accelerator, 2019 IEEE 9th Annual Computing and Communication Workshop And Conference (CCWC). (2019) 0513-0520.
[11] (2011). H. Devos, J. Van Campenhout, I. Verbauwhede and D. Stroobandt, Transactions on High-Performance Embedded Architectures and Compilers III, Constructing Application-Specific Memory Hierarchies on FPGAS. [Online]. Available: Http://Dl.Acm.Org/Citation.Cfm?Id=1980776.1980790.
[12] S. S. Jadhav, C. Gloster, C. Doss, Y. Kim and J. Naher, Autorare: An Automated Tool for Generating FPGA-Based Multi-Memory Hardware Accelerators for Compute-Intensive Applications, 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC). (2018) 1-8.
[13] B. Betkaoui, D. B. Thomas, W. Luk and N. Przulj, A Framework for FPGA Acceleration of Large Graph Problems: Graphlet Counting Case Study, Field-Programmable Technology (FPT) 2011 International Conference On. (2011) 1-8.
[14] Data Sheet S28HS512T / S28HS01GT S28HL512T / S28HL01GT 512-Mb (64-MB) 1-Gb (128-MB) HS-T (1.8-V) HL-T (3.0-V) Semper Flash with Octal Interface. (2018).
[15] X. Guo et al., Fast Energy-Efficient Robust and Reproducible Mixed-Signal Neuromorphic Classifier Based on Embedded NOR Flash Memory Technology, IEEE IEDM. (2017) 151-154.
[16] H. Om'mani et al., A Novel Test Structure to Implement a Programmable Logic Array Using Split-Gate Flash Memory Cells, ICMTS. (2013) 192-194.
[17] S. Jain, A. Ranjan, K. Roy and A. Raghunathan, Computing in Memory with Spin-Transfer Torque Magnetic RAM, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 26(3) (2018) 470-483.
[18] G. H. Loh, 3D-Stacked Memory Architectures for Multi-Core Processors, SIGARCH Comput. Archit. News. 36(3) 453-464 (2008).
[19] V. K. Chippa, S. T. Chakradhar, K. Roy and A. Raghunathan, Analysis and Characterization of Inherent Application Resilience for Approximate Computing, Proc. 50th Annu. Design Autom. Conf. (DAC). (2013) 1-9.
[20] S. Venkataramani, S. T. Chakradhar, K. Roy and A. Raghunathan, Approximate Computing and the Quest for Computing Efficiency, Proc. 52nd Annu. Design Autom. Conf. (DAC). (2015) 120:1-120:6.
[21] Q. Xu, T. Mytkowicz and N. S. Kim, Approximate Computing: A Survey, IEEE Des. Test. 33(1) (2016) 8-22.
[22] K. Cho, Y. Lee, Y. H. Oh, G.-C. Hwang and J. W. Lee, EDRAM-Based Tiered-Reliability Memory with Applications to Low-Power Frame Buffers, Proc. Int. Symp. Low Power Electron. Design (ISLPED). (2014) 333-338.

Vijayalakshmi chintamaneni is a research scholar at lnct university, Bhopal. She received her PG degree in VLSI & Embedded Systems from JNTU Kakinada in 2012. She has 8 years of teaching experience from 2008-to 2018. She received MBA in finance from IGNOU, New Delhi. She published more than 22 papers in various national & international journals and conferences. She is a life member of ISTE, IETE. She is awarded as state-level best service personnel in teaching for the year 2019 by little champs academy of India. She published 2 books in IOT & FOG Computing domains.

Dr. Jaikaran Singh was awarded the Ph.D. degree in Electronics and Communication Engineering by Rajiv Gandhi Technical University (Technical University of MP Government) Bhopal, MP India, in 2016. He passed his M.Tech. Degree in ECE with honour in 2009. He has more than 22 years of teaching experience. Dr Singh has published 2 books and more than 100 research papers in various National/International Journals/conferences**.** He has guided more than 30 M.Tech. Students and having more than 6 Life membershipS of professional bodies. He has organized/conducted more than 50 technical events.