Zewail City for Science and Technology
University of Science and Technology
Nanotechnology and Nanoelectronics Engineering

Digital Design of DDR5 Physical Layer

A Graduation Project
Submitted in Partial Fulfillment of
B.Sc. Degree Requirements in

Nanotechnology and Nanoelectronics Engineering

Prepared By

Mohamed Hosni Abdalmonem	201700450
Mohammed Waleed Hassan	201700392
Mohamed Mahmoud El-Desouky	201701302
Yasmein Khalil Abdelmeguid	201700110

Supervised By

Dr. Hassan Mostafa   Signature

------------------   --------------------

Fall 2021/2022

Acknowledgments

This project was funded and sponsored by Si-vision, Egypt and ONE lab at Zewail City of Science and Technology.

Abstract

This report aims to present the work done in the first semester for the graduation project entitled: Digital Design of DDR5 PHY under the supervision of Dr. Hassan Mostafa and sponsored by Si-vision. The report contains introduction and literature review on the topic, the standards used, and the project design and execution.

Table of Contents

List of Figures

List of Tables

1- Introduction

1.1 General introduction and literature review

There is a continuous need in the market for a faster, larger and lower power memories, and these demands are the reason for modern DRAM (Dynamic Random-Access Memory) advancements. DRAMs have evolved into SDRAM (Synchronous DRAM), DDR (Double Data Rate) SDRAM, DDR2 SDRAM, DDR3 SDRAM, DDR4 SDRAM, DDR SDRAM. With these advancements to DRAMs, and due to the complexity of memory controllers, a physical layer was needed to provide compatibility between MC and the newer DRAM versions. To begin, memory systems must be discussed.

1.1.1 Memory systems

Memory speed is an essential aspect for computer systems operation. The most important concept for the modern memory system advancement is memory hierarchy. Advanced memory hierarchies gather the performance of the fastest component, the cost per bit of the cheapest component, and the energy of the most energy-efficient component. The concept of memory hierarchy transformed the system design into a modular process. Consequently, enhance the development of each subsystem (disk, DRAM, cache) independently. However, the independence in designing and optimizing the memory subsystems is no longer efficient due to many problems that arose in the modern technologies such as: device physics, signaling protocols choice, choice of topologies to achieve signal integrity, concurrency, and problems related to communication such as queuing and scheduling algorithms. Currently, these problems dominate the design process although they were insignificant a decade ago. The interconnect physics in sub-micron technologies is the main challenge in current cache architectures, for modern DRAM designs are controlled by circuit-level constraints, while in disks on-board caching and scheduling policies dominate their performance. The main challenge is not in disks, DRAMs, or caches, but in the interaction between them and the connection methods. Consequently, the isolation concept in subsystems design is no longer a valid method. Instead, the memory system designer has to be familiar with the problems related to all levels of memory hierarchy whether it is disk, DRAM, or cache. An overview of memory subsystems will be explained further in the following subsections.

1.1.2 Cache

Cache relies on the locality of reference principle, apps' proclivity for referencing a predictable limited quantity of data within a specific time frame. Classification of storage devices is divided into access time and cost per bit factors, with faster storage technologies having a shorter access time and a higher cost per bit than slower storage technologies. The cache technology would typically cost more per bit, but the cache would only need to be large enough to hold the application's set of instructions in addition to data items.

Most application accesses will be satisfied out of the cache due to locality of reference, and so the access characteristics will be those of the cache most of the time: significantly faster and often requiring less energy than the bigger storage device behind the cache. A single cache could be made up of several different entities that work together. A symmetric multiprocessor's individual last-level caches, for example, might be thought of and managed as a dispersed yet logically unified entity. Some cache organizations, on the other hand, are multi-level hierarchies that are described as a single cache. Thus, a single "cache" can be labeled and studied as a collection of numerous storage units, and a monolithic cache can be labeled and analyzed as a collection of multiple, independent entities. Caches can be transparent; their actions are independent of the client's requests. Consequently, embedded algorithms are included in them for deciding which data will remain in it.

Another type of cache is scratch-pads, which are handled directly by the request making process by the client. Hybrid combinations from the two types can exist. Processor caches in general-purpose systems, file caches in most distributed file systems, and practically all types of web caches are transparent caches, because the "client" of a web cache is often the person using the web browser. Scratch-pads include the widely used register file and tag-less SRAMs found in practically all microcontrollers and digital signal processors. There are three main concepts regarding caches, cache’s organization, methods for content management, and methods for consistency management.

- The logical arrangement of data saved within the cache's context is the cache's structure. An operating system's buffer cache, for example, could be structured as an array of queues, with each queue weakly indicating the last time a saved item was referenced (e.g., queue1 holds items that have just arrived, queue2 holds items that have been recently referenced, queue3 holds items that haven't been referenced in a while, etc.); a solid-state cache could be arranged as a cluster of sets, each of which includes a fair number of cache blocks managed in a time-ordered way.
- The determination to cache or not caches a specific item at a specific time during execution is represented by content-management algorithms. These can be handled by the programmer and/or compiler during the design phase, or by application software and/or the cache itself during the run phase. The algorithms can be static, in which case the contents of the cache do not change significantly over time, or dynamic, in which case the elements of the cache may change dramatically from time - to - time.
- Algorithms for consistency management guarantee that the instructions and data that application software receives are really received. Continuity, like cache contents, can be handled by a range of aspects, including the operating system, application software, and the cache itself.

Computer programs’ behavior have been recorded to meet certain criteria regarding the memory access patterns. The memory-access patterns are found not to be random. Locality of reference refers to the phenomena of predictable, non-random memory access behavior. The behavior is named from the fact that memory accesses in a program tend to be localized in spatial and temporal:

- If the program refers to a data once, it will almost certainly refer to it again in the near future.
- If the program accesses data once, it will very certainly reference nearby data in the near future.

The first is known as temporal locality, whereas the second is known as spatial locality. Computer scientists have recently noticed a new form of event. Another type of behavior has emerged in the last decade as a result of the extensive usage of computer graphics algorithms and comparable algorithms in other disciplines, such as computer modeling, circuit emulation, HDL3 code interpretation, and so on. Because these programs often stroll down dynamic data types and access the exact data in almost the same sequence over and over (for example, each time a new video frame is formed), their behavior is essentially deterministic. The duration between 2 subsequent visits to a certain data, on the other hand, is long (hence, the program displays no specific temporal locality), and the data items are usually far apart in memory space (thus, the program shows no specific spatial locality). These programs do not demonstrate considerable locality, according to our existing notions of locality, but they do exhibit regular, predictable, and exploitable behavior. It's difficult to describe this type of behavior because it appeared to have covered everything in terms of time and location. As a result, it's simple to miss the fact that this type of conduct exists. But it certainly exists. This behavior is called algorithmic locality.

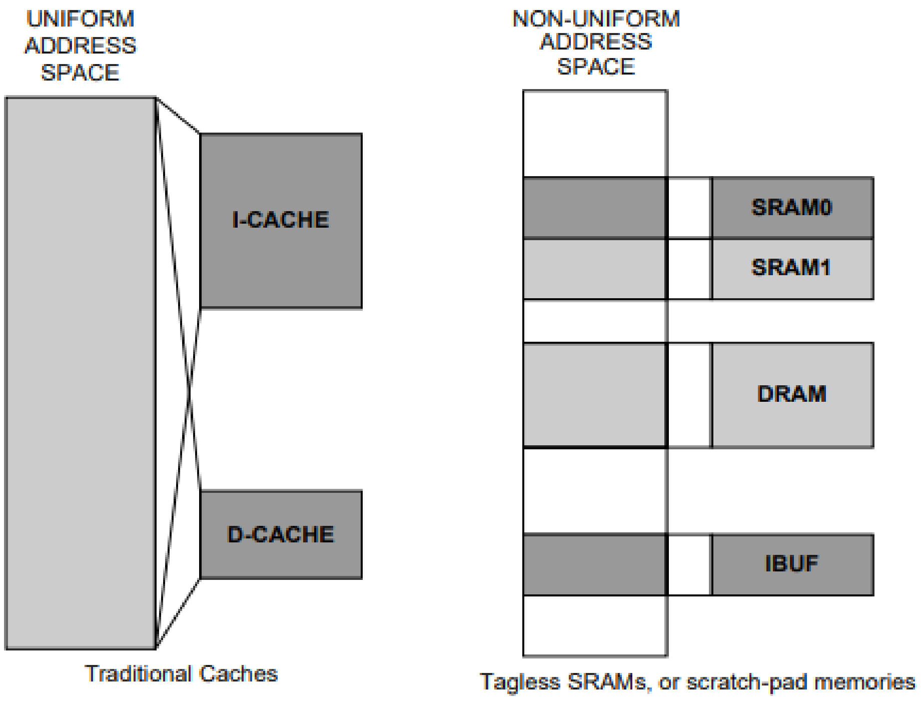


Figure 1. Scratch-pad SRAMs vs. transparent caches in a non-uniform domain.

1.1.3 DRAM

DRAM is the "computer memory" that can be acquired online or bought in a store. The universal memory module, a small computer board (a printed circuit board, or PCB) with a handful of chips linked to it, occurs in most systems in the manner shown in figure 2. The DRAM chips are the eight black rectangles on the displayed module: plastic packaging that each encapsulates a DRAM die (an extremely thin, fragile slice of silicon).

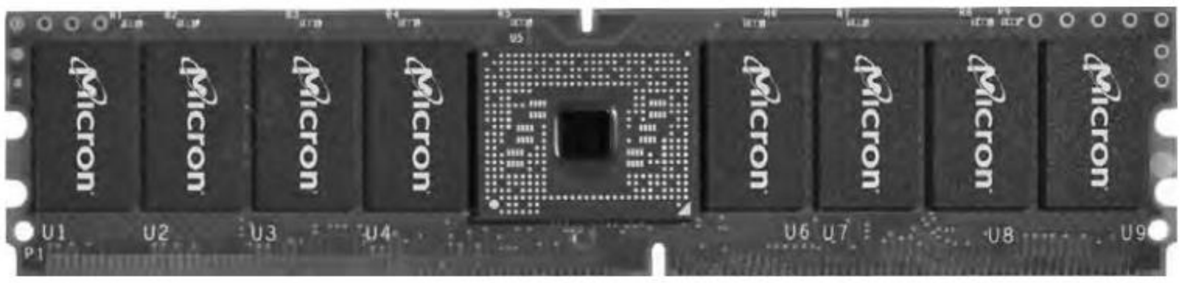


Figure 2. DIMM (dual in-line memory module) [1].

DRAM's location in a typical PC is depicted in figure 3. A memory controller normally connects an individual DRAM device to a CPU (i.e., a microprocessor) indirectly. The memory controller is part of the north-bridge chipset, which controls several microprocessors, the graphics coprocessor, communication with the south-bridge chipset (which controls all of the system's I/O functions), and the interface to the DRAM system in PC systems.The north- and south-bridge chipsets are no longer chipsets; they are generally performed as individual chips, and in certain devices, the functionality however are integrated into a single die. As DRAM is typically an external entity by default, the use of it, designing, and analyses should account for execution impacts which are frequently overlooked for the use, architecture, and evaluation of on-chip memories like SRAM caches and scratch pads. The following is some of the concerns that a design team should think about:

- Pins (capacitance and inductance).
- Signaling.
- Signal integrity.
- Packaging.
- Clocking and synchronization.
- Timing conventions

Inability to take these factors into account while developing a DRAM system will almost certainly result in a substandard and non-functional implementation.

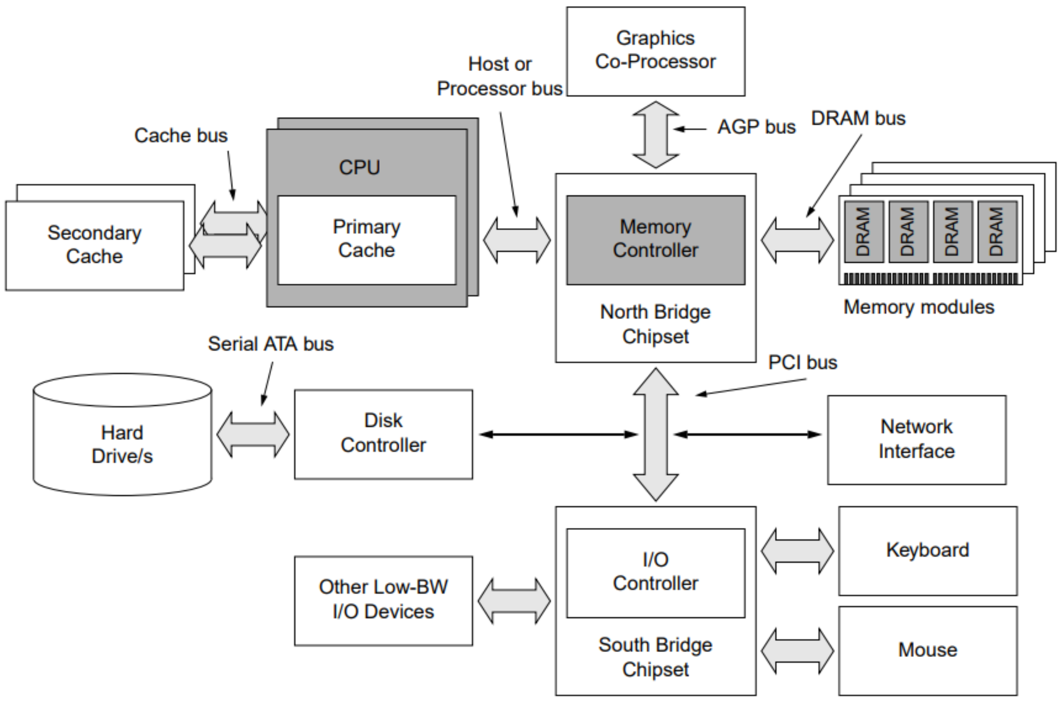


Figure 3. A common computer setup [1].

1.1.4 Disk

In the last few years, disk technology has evolved dramatically. It permits very complicated programs to be built without worrying about size limitations, while also liberating individuals from having to be concerned about what as well as how much data to store, by offering almost limitless on-line storage at extremely minimal prices. Whereas disk storage isn't always the fundamental drive of a computer, its rapid advancement figured prominently in propelling computer systems forward from their earlier years to where they are today.

Considering how limited recent computers can be if they had only dozens of megabytes of secondary storage instead of the tens of gigabytes, we now take for granted. The influence of disk-based storing nowadays has extended across computer systems, having begun with a double storage device for management accounting reporting. It is becoming more prevalent in our daily lives as embedded electronics in products such as streaming video, cameras, mp3 players, automotive navigation units, mobile phones, and so on. Although disk drives' core concepts stay the same, these applications necessitate a different focus and performance characteristics. Regardless of the fact that hard disks have become standard items, they are nevertheless a highly complicated electro-mechanical system incorporating years of highly developed studies across a wide range of disciplines. Mathematics, chemistry, material science, tribology, electrical & electronics engineering, mechanical engineering, computer programming, data science, and industrial science are just a few of the subjects they cover. This gives the foundation needed to look at some of the design difficulties and trade-offs that really can impact the function of hard disks and hard drive storage subsystems.

1.1.5 DRAM Organization

A signal DRAM chip includes multiple bank groups and each bank group includes multiple banks. Each bank has a standard number of DRAM arrays which are 4, 8 or 16, which is also identified as width of the column making DRAMs classified as x4, x8 or x16 based on the column width. Another note is that the width of the DQ data bus is the same as the column width, so we can say that the DRAMs are classified based on the width of the DQ bus. All the arrays in the same bank are fed the same address row address so the same word line is selected in each of the arrays, also the arrays are fed the same column address thus the same column address is selected in each of the arrays. At the end one cell is selected from each array but at the same position, and that’s how a byte is read from a DRAM chip. Opening and closing a row can take up to 18 cycles that’s why burst mode was introduced to allow reading a number of words simultaneously without needing to re-open the row again. This happens by reading multiple columns from an already opened row successively. Different generations of DRAM have different Burst length DDR4 has 8 and DDR5 has 16. Figure 4 shows the DRAM hierarchy and organization. However, after finishing the burst needed and the time for the row closing comes, the DRAM would need to be closed for a period of time before re-opening it. That is why banks were introduced to allow accessing another bank, and this method was called “Interleaving Memory Banks” as this method achieves higher bandwidth as the data bus uses frequency that’s higher than any one DRAM bank can support. The word banks introduce a set of independent memory arrays inside the DRAM device. Independent accesses to different DRAM arrays can occur in parallel, as each bank is an independent array. The array can be in different phases in row access cycles, that’s why Multi banks can operate independently or concurrently, can be activated independently, and precharged or refreshed in parallel Interleaving memory banks achieve high bandwidth as the data bus uses a frequency that’s higher than any one DRAM bank can support. Figure 5 shows the timing of bank interleaving. That’s why we use multiple independent memory banks as it allows other banks to be in a different read/write cycle than others. When the row is closed in bank 1, bank 2 row is ready to be read and buffered and this continues until we reach the final bank that’s when bank 1 is ready again to be used.

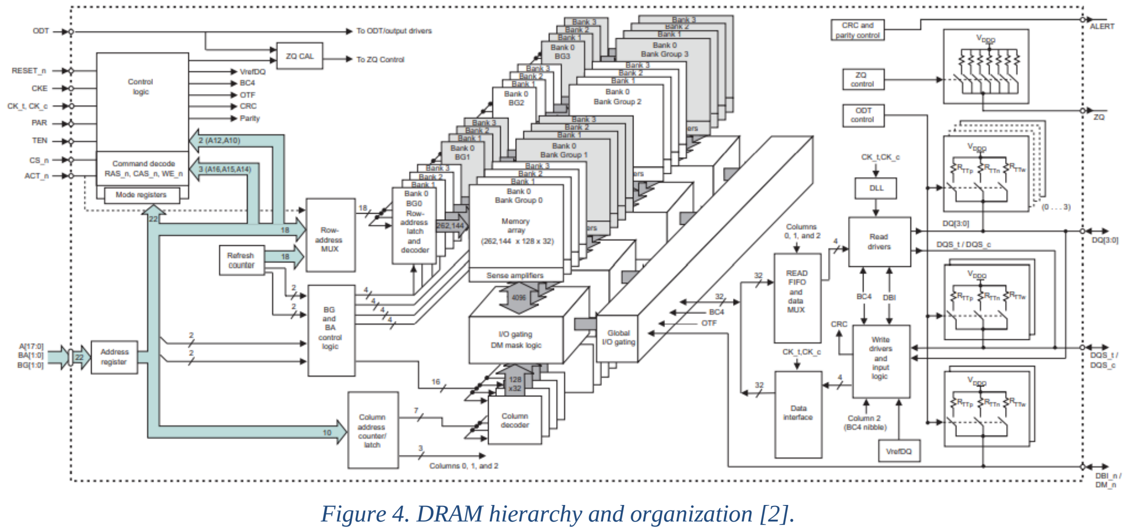


Figure 4. DRAM hierarchy and organization [2].

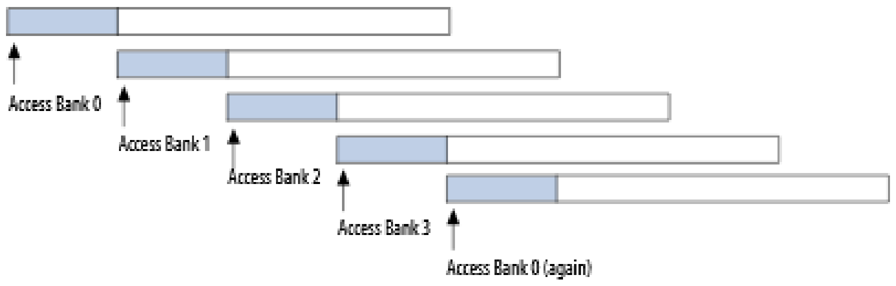
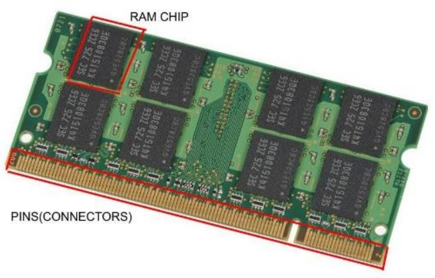


Figure 5. Bank interleaving timing.

The RAM chips are not individually mounted on the motherboard because of less capacity, hence in earlier times, several chips used to be soldered together and converted into “modules”(integrated circuit boards) and these modules were mounted over motherboard using “pins”(also known as connectors). Figure 6 shows the RAM DIMM modules layout.



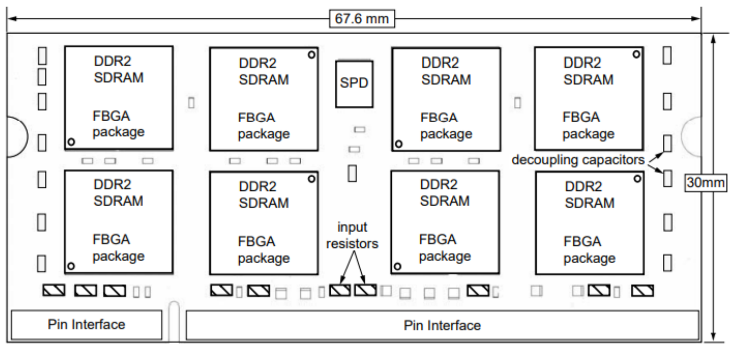
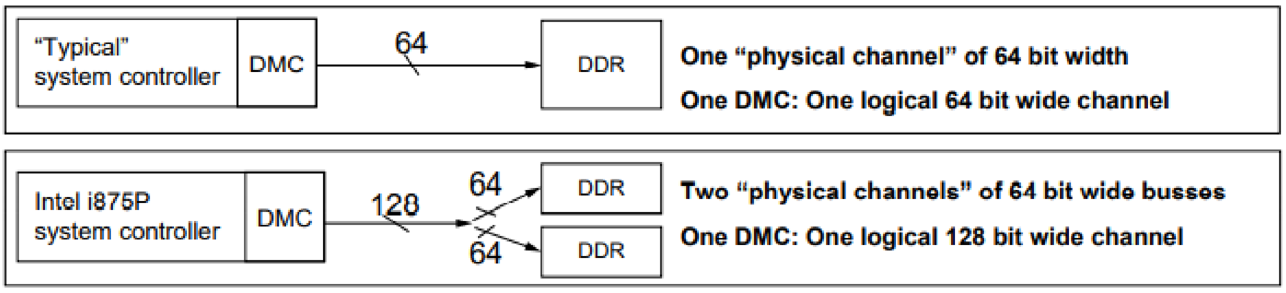


Figure 6. DIMM modules layout.

DRAM memory controller controls a single channel of memory as in figure 7. The typical system controller controls a 64-bit-wide channel. The system controller like intel i875P requires a matching pair of 64-bit wide memory modules to operate with 128-bit-wide data bus which is referred to as dual channel configuration, both memories operate in lockstep. But it can allow the use of mismatched pairs of memory modules in the different physical channels, however the multiples mismatched memory modules cannot be accessed concurrently, and only one channel of memory can be accessed at any given time.



“Typical” system controller
DMC
64
DDR
One “physical channel” of 64 bit width
One DMC: One logical 64 bit wide channel
Intel i875P system controller
DMC
128
64
64
DDR
DDR
Two “physical channels” of 64 bit wide busses
One DMC: One logical 128 bit wide channel

Figure 7. DRAM channels.

The memory system in figure 8 is populated with 2 ranks of memory systems. A rank is now used to denote a set of DRAM devices that operate in lockstep in response to a given command. Address and command busses are connected to every DRAM device in the system. The wide 64-bit data bus is partitioned into equal 16 bits and connected to different DRAMs, and chip select is inserted to select the needed Rank.

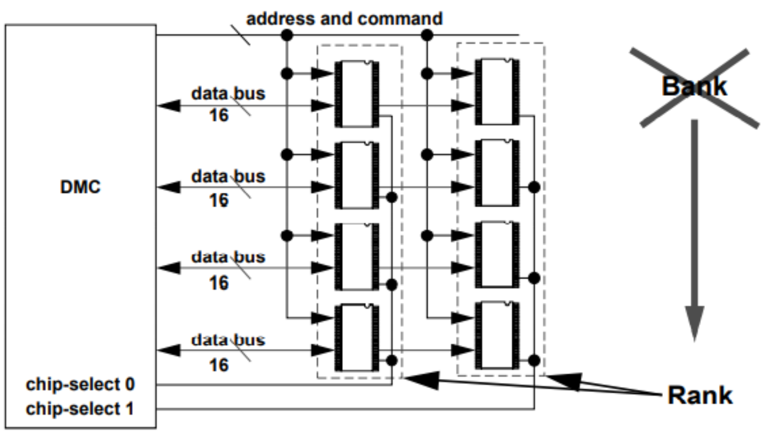


Figure 8. Memory system with 2 ranks.

DRAM is soldered down on a board. User Logic, DDR controller and DDR PHY are all part of the ASIC or FPGA. The interface between the user logic and the DDR controller is user-defined no need for standard. When the user logic makes a read/write request, it issues logical address, then the controller converts the logical address to physical address, then issues these commands are issued to the PHY. The controller and the PHY talk to each other over DFI standard interface. The PHY then does all the lower-level signalling and drives the DRAM. The interface between the DRAM and the PHY is specified in JEDEC standard. Figure 9 shows the PHY location in the memory system architecture.

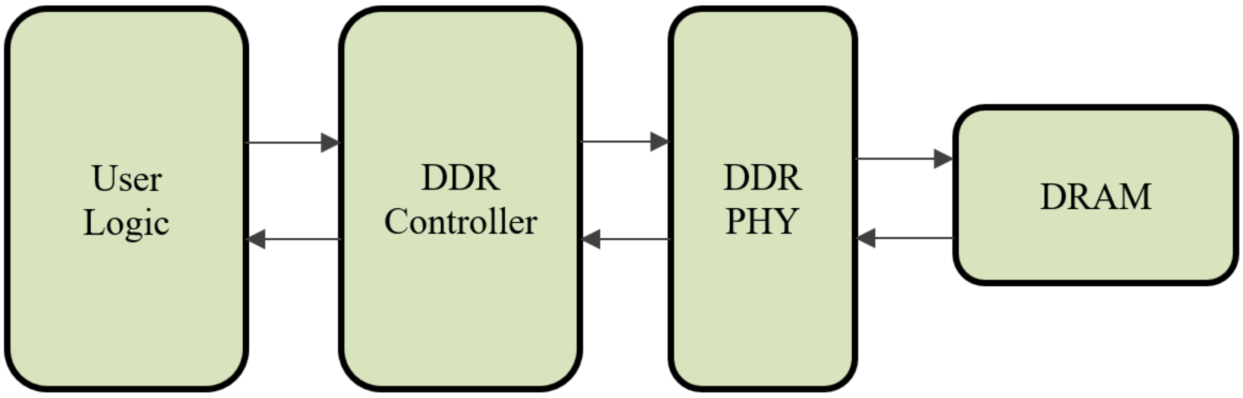


Figure 9. Memory system architecture.

1.2 *Problem definition*

The demand of faster, higher capacities and lower latencies DRAMs has increased recently to meet the CPU, and GPU workloads [3]. SDRAM latest generation DDR5 was released in 2020 and it has significantly superior performance compared to its latest predecessor as shown in table 1 [4].

As memories advance newer generations of hardware (processors and motherboards) are required to fully utilize its capabilities. Currently, only 12th generation Intel processors support DDR5 and none of AMD's processors support it as illustrated in table 2 [5]. To be able to support DDR5 SDRAM, a processor must have a Memory Controller (MC) and a Physical Layer (PHY) that can handle data rates and features illustrated in table 1. In this project we design, verify and implement the digital part of a Silicon Intellectual Property (IP) of a specific PHY implementation in collaboration with an industrial partner.

Table 1. DDR4 vs DDR5 [4].

Feature/Option	DDR4	DDR5	DDR5 Advantage
Data rates	1600-3200 MT/s	3200-6400 MT/s	Increases performance and bandwidth
V_{DD}/V_{DDQ}/V_{PP}	1.2/1.2/2.5	1.1/1.1/1.8	Lowers power
Internal V_{REF}	V_{REFDQ}	V_{REFDQ}, V_{REFCA}, V_{REFCS}	Improves voltage margins, reduces BOM costs
Device densities	2Gb-16Gb	8Gb-64Gb	Enables larger monolithic devices
Prefetch	8n	16n	Keeps the internal core clock low
DQ receiver equalization	CTLE	DFE	Improves opening of the received DQ data eyes inside the DRAM
Duty cycle adjustment (DCA)	None	DQS and DQ	Improves signaling on the transmitted DQ/DQS pins
Internal DQS delay monitoring	None	DQS interval oscillator	Increases robustness against environmental changes
On-die ECC	None	128b+8b SEC, error check and scrub	Strengthens on-chip RAS
CRC	Write	Read/Write	Strengthens system RAS by protecting read data
Bank groups (BG)/banks	4 BG x 4 banks (x4/x8) 2 BG x 4 banks (x16)	8 BG x 2 banks (8Gb x4/x8) 4 BG x 2 banks (8Gb x16) 8 BG x 4 banks (16-64Gb x4/x8) 4 BG x 4 banks (16-64Gb x16)	Improves bandwidth/performance
Command/address interface	ODT, CKE, ACT, RAS, CAS, WE, A<X:0>	CA<13:0>	Dramatically reduces the CA pin count
ODT	DQ, DQS, DM/DBI	DQ, DQS, DM, CA bus	Improves signal integrity, reduces BOM costs
Burst length	BL8 (and BL4)	BL16, BL32 (and BC8 OTF, BL32 OTF)	Allows 64B cache line fetch with only 1 DIMM subchannel.
MIR ("mirror" pin)	None	Yes	Improves DIMM signaling
Bus inversion	Data bus inversion (DBI)	Command/address inversion (CAI)	Reduces V_{DDQ} noise on modules
CA training, CS training	None	CA training, CS training	Improves timing margin on CA and CS pins
Write leveling training modes	Yes	Improved	Compensates for unmatched DQ-DQS path
Read training patterns	Possible with the MPR	Dedicated MRs for serial (userdefined), clock and LFSR -generated training patterns	Makes read timing margin more robust

Mode registers	7 x 17 bits	Up to 256 x 8 bits (LPDDR type read/write)	Provides room to expand
PRECHARGE commands	All bank and per bank	All bank, per bank, and same bank	PREsb enables precharging-specific bank in each BG
REFRESH commands	All bank	All bank and same bank	REFsb enables refreshing of specific bank in each BG
Loopback mode	None	Yes	Enables testing of the DQ and DQS signaling

Table 2. Processors that support DDR5 [5].

CPU Manufacturer	**Application**	**Generation**	**Launch plan**
Intel	Desktop	12th Gen	Q4-21
	Mobile	12th Gen	Q1-22
	Workstation	12th Gen	Q1-22
	Server	4th Gen Xeon	2H-22
AMD	Desktop	Zen4	2H-22
	Mobile	Zen3+	1H-22
	Workstation	Zen3+	2H-22
	Server	Zen4	2H-22

1.3 Project Objectives

The objective of the project documented in this thesis is to design, verify and implement a DDR5 PHY Utility Block IP in collaboration with system and design engineers at Silicon Vision. The system is shown below in figures 10 and 11. Figure 10 shows typical computer organization where the communication between memory and processor is highlighted in red. Figure 11 shows a breakdown of said system with more details where the PHY mediates this communication between the Memory Controller (MC) and the DRAM module with a different interface standard at each end. The MC and PHY interface is defined by DFI V.5.0 standard and the PHY DRAM interface is defined by JEDEC JESD79-5A standard (both standards are to be discussed with more details later in the thesis).

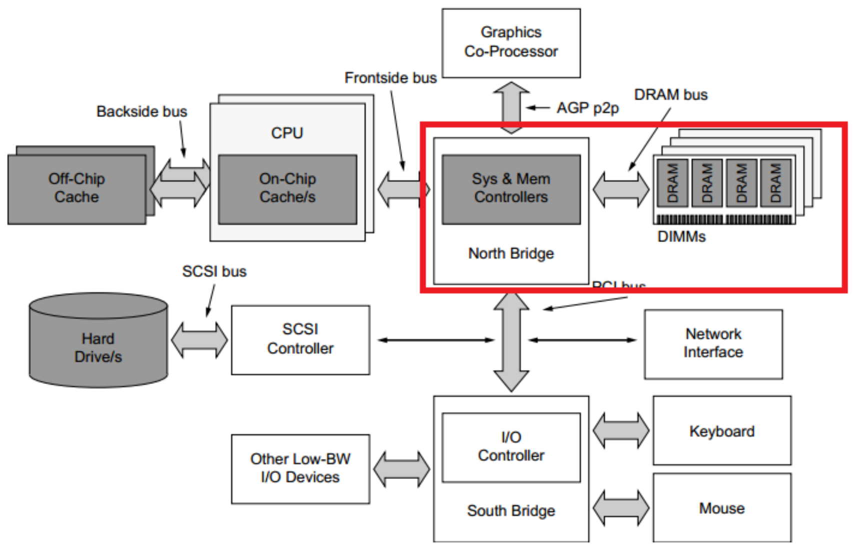


Figure 10. Standard Computer Organization.

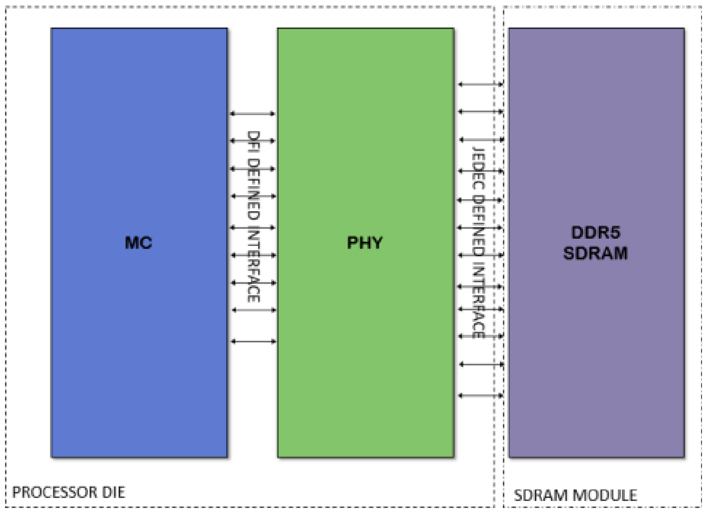


Figure 11. Processor memory interface.

To attain the project objective, the following series of steps was implemented each of which will be discussed properly in a separate section

1- Standards specification extraction

Both interface standards were first discussed with the system engineer to acquire knowledge about the company specific PHY IP implementation and to extract functional specification of the design. The main functionalities required for PHY IP.

2- System level architectural design and high-level modeling

After the main functional requirements were obtained, an architectural design of the system was carried out and each block was implemented as a high-level model using MATLAB.

3- Block level digital design, verification and optimization

After the system level design and verification of intended functionality, each block RTL was designed and tested against the high-level model and expected timing diagrams available in interface standards.

4- Physical Implementation

The system was implemented on simulated and implemented on FPGA and ASIC.

1.4 Functional Requirements

- Support DDR5 SDRAM Devices with multiple widths x4, x8, and x16.
- Support multiple frequency ratios.
- Support CRC (Cyclic Redundancy Check) validation.
- DFI and JEDEC compatible.
- Support Read (RD), Read with Auto Recharge (RDA), Refresh (REF), Refresh All (REFab), Mode Register Write (MRW), Mode Register Read (MRR), Pre-charge (PRE), and Activate (ACT).

1.5 Report organization

This thesis is organized as follows: The next part discusses certain sections of DFI V.5.0 and JEDEC JESD79-5A that were crucial during specification extraction phase of this project. Part three discusses a sample of the literature that relates to the memories and the current global market status. The remaining sections discuss project design, execution and evaluation considering numerous criteria of concern. Hence, Finally, a summary of possible future extension to our work is discussed alongside with conclusion.

2- Standards used

2.1 DFI [6]

DFI is a standard that defines the interface between the Memory controller (MC) and the DDRx Physical layer PHY. DFI divides that interface to 12 interface groups where an interface group consists of signals and parameters some of which may be programmable. The names and functionalities of each group interface is illustrated in Table 3.

Table 3. DFI interface groups [6].

Interface Group	Description
Command	Required to drive the address and command signals to the DRAM devices.
Write Data	Used to send write and receive valid read data across the DFI.
Read Data	
Update	Provides an ability for the MC or the PHY to initiate idling the DFI bus.
Status	Used for system initialization, feature support and to control the presence of valid clocks to the DRAM interface.
PHY Master	Used to allow the PHY to control the DFI bus.
Disconnect Protocol	Allows an ongoing handshake to be broken.
Error	Used to communicate error information from the PHY to the MC.
2N Mode	Uses the 2N function (also referred to as the geardown mode) of the DRAM
Low Power Control	Allows the PHY to enter power-saving modes.
MC to PHY Message	Used to send defined messages from the MC to the PHY.
WCK Control	Controls the WCK on, off and synchronization timing.

2.1.1 Clock Domains

DFI BUS operates at the same frequency as the MC, which is different from the frequency of operation of the DRAM itself. Some DRAMS have separate clocks for commands and data and others don't. Thus, DFI interface have 3 clock domains:

1. DFI Clock
2. DFI Command Clock
3. DFI Data Clock

For DRAMS that have the same clock for command and data both command and data clocks run at the same frequency which is 1x 2x or 4x of the DFI clock which is the MC clock as illustrated earlier. In this case the command and data clock frequency are half of the actual data rate (because DDR is operated at both edges). For DRAMS that have different clocks for command and data, the command clock runs on the same frequency as the MC (DFI clock) and data remains 1x 2x or 4x of that frequency. For 1x systems it's called Matched Frequency system (the three domains have the same frequency) else it is called frequency ratio system. Generally, the standard uses the term DFI clk for MC clk and DFI PHY for data clk. For DDR5 the DFI Command Clock and the Data Clock run at the same frequency, while the relation between them and the command clock is identified by the frequency ratio.

2.1.2 *Command interface and Functionality*

The command interface handles the transmission of signals required to drive the address and command signals to the DRAM devices. The concept of frequency ratio was introduced earlier. Now we introduce the concept of phase, for a frequency ratio of x there are x DFI PHY clock cycles occurring in one DFI clock cycle, these x cycles shall be named phase 0 and so on until phase x-1. phase-specific signals are enhanced with a suffix of “pN” that defines the signal value for each phase N of the DFI PHY clock.

2.1.3 *Read interface and functionality*

The read interface signals are shown in table 4.

Table 4. Read interface signals [6].

Signal	**From**	**Width**	**Description**
dfi_rddata or dfi_rddata_wN	PHY	DFI Data Width	Read data bus. This bus transfers read data from the PHY to the MC. Read data is expected to be received at the MC within tphy_rdlat cycles after the dfi_rddata_en signal is asserted.
dfi_rddata_cs or dfi_rddata_cs_pN	MC	DFI Physical Rank Width	DFI read data chip select. The polarity of this signal is the same as the polarity of the dfi_cs signal. This signal indicates which chip select is accessed or targeted for associated read data.
dfi_rddata_en or dfi_rddata_en_pN	MC	DFI Data Enable Width	Read data enable. This signal indicates to the PHY that a read operation to memory is underway and identifies the number of data words to be read. The dfi_rddata_en signal must be asserted trddata_en cycles after the assertion of a read command on the DFI command interface and remains valid for the duration of contiguous read data expected on the dfi_rddata bus. Ideally, there is a single dfi_rddata_en bit for each PHY data slice. The dfi_rddata_en [0] signal corresponds to the lowest segment of dfi_rddata signals.
dfi_rddata_valid or dfi_rddata_valid_wN	PHY	DFI Read Data Valid Width	Read data valid indicator. Each bit of the dfi_rddata_valid signal is asserted with the corresponding dfi_rddata for the number of cycles that data is being sent. The timing is the same as for the dfi_rddata bus. The width of the dfi_rddata_valid signal is equivalent to the number of PHY data slices. Ideally, there is a one-to-one correspondence between a dfi_rddata_valid signal bit and each PHY data slice. The dfi_rddata_valid[0] signal corresponds to the lowest segment of the dfi_rddata signals.

2.1.4 *Read Sequence*

The read sequence is as follows:

1. The read command is issued.
2. t_rddata_en cycles elapse. Then the rddata_en signal is asserted for the length of the data to be captured later on.
3. For contiguous read commands, rddata_en signal can be asserted for the total length of the 2 or more data streams.
4. Data is returned on the dfi_rddata bus.

5. The data is returned with the dfi_rddata_valid signal asserted.
6. The associated read data signal (dfi_rddata) is sent to the MC.

2.1.5 Read Examples

The following figures 12, 13, 14, 15 and 16 show different read examples from the DFI standard.

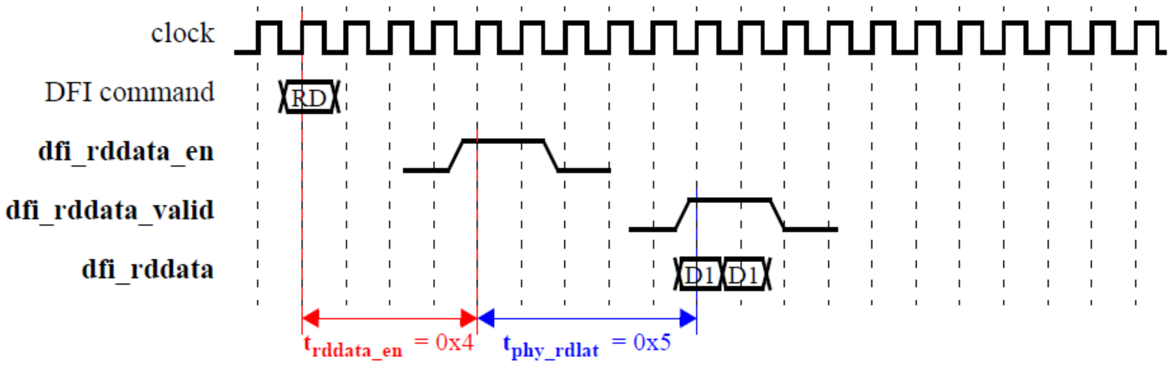


Figure 12. Single read transaction where the data is returned in less than the maximum delay.

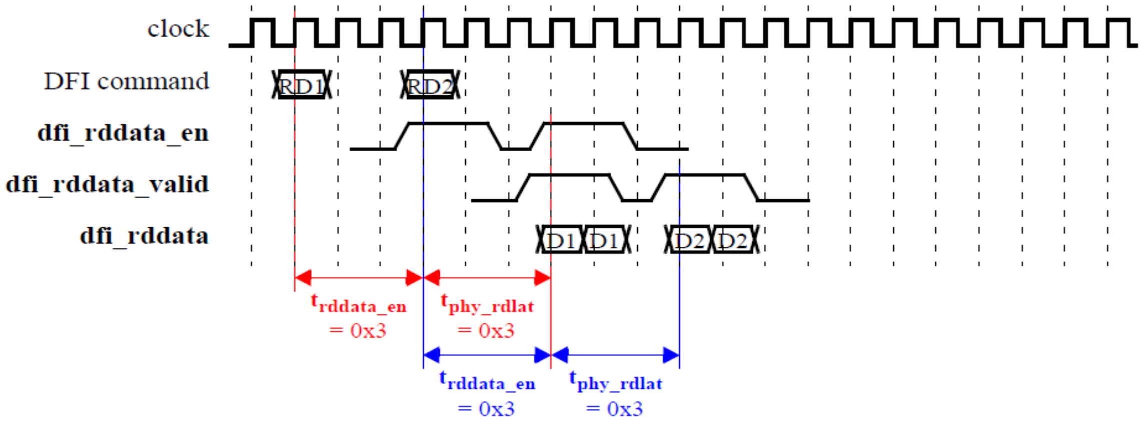


Figure 13. Two Independent Read Transactions (DDR2 Example).

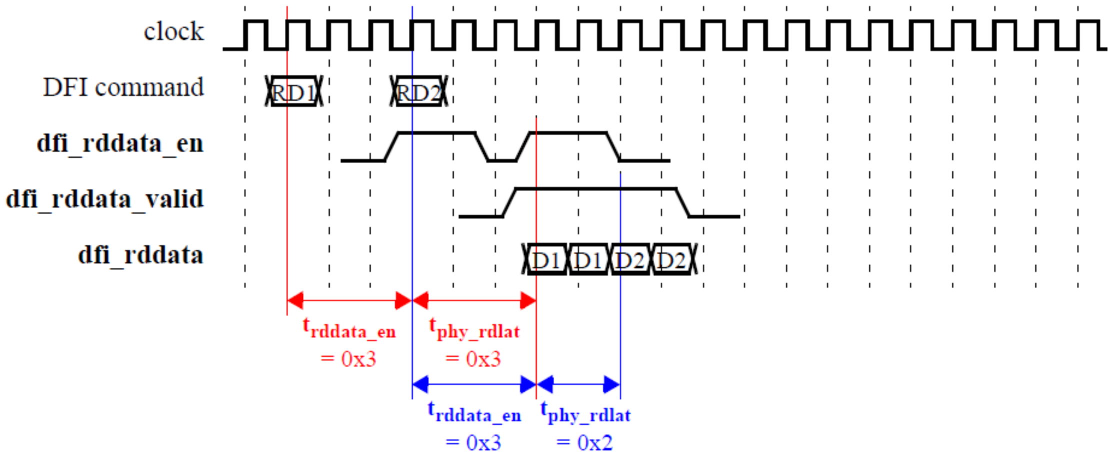


Figure 14. Two Independent Read Transactions (DDR2 Example).

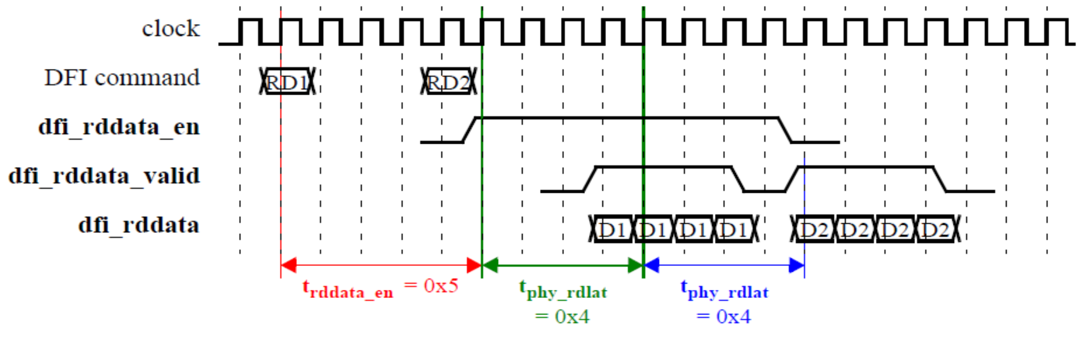
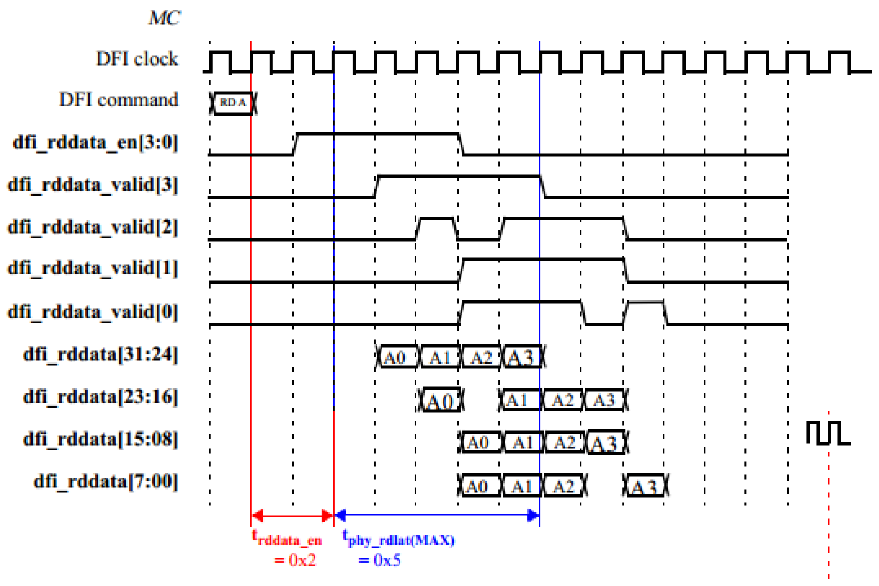


Figure 15. Two Independent Read Transactions (DDR3 Example)



.

Figure 16. DFI Read Data Transfer Illustrating dfi_rddata_valid Definition.

2.1.6 *Frequency Ratio Concept*

We have illustrated earlier that in DFI there are 3 separate clock domains namely DFI clock, DFI command and DFI Data. However, 2 of these domains run at the same frequency, depending on DRAM devices. If a DRAM device operates command and data at the same memory clock, then both run at the same frequency that is half the actual data transmission rate and that is 1x 2x or 4x of the DFI clock frequency. However, if DRAM device operates command and data at different clocks the command clock frequency is the same as the DFI frequency and they both are commonly referred to as DFI clk and the data clock could be 1x 2x 4x of their frequency and it is commonly referred to as DFI PHY clk. The MC communicates frequency ratio settings to the PHY on the dfi_freq_ratio signal. This signal is only required for devices using this frequency ratio protocol. We define phases in frequency ratio systems as the number of the clk cycle of the higher frequency that is occurring in the phase aligned lower frequency period as illustrated in figure 17. In DFI standard, the following statement is mentioned:

The MC and the PHY must operate at the same frequency ratio. The frequency ratio applies to the command and DFI data clock domain (PHY frequency ratio) or to the DFI data clock domain (data frequency ratio) only. DFI clock domain signals do not operate at a clock ratio, they are always DFI clock based. The definition of frequency ratio for MC is absurd and needs to be clearly illustrated.

The DFI specification supports the ability to send a unique command on each phase of the DFI PHY clock. To communicate this information to the PHY, the DFI specification defines commands for a frequency ratio system in a vectored format. The PHY must maintain this information to preserve the timing relationships between commands and clock data. Therefore, for frequency ratio systems, the command interface, the write data interface and the read data enable signal are all suffixed with a "_pN" where N is the phase number. As an example, for a 1:2 frequency ratio system, instead of a single dfi_address signal, there are 2 signals: dfi_address_p0 and dfi_address_p1. The operation of frequency ratio systems will be illustrated through several timing diagrams below.

The figure below illustrates how command interfaces would be interpreted by PHY. The signal dfi_we_n_p0 and the signal dfi_we_n_p1 are both high at the first DFI clock cycle thus in both phases for that particular cycle the dfi_we_n signal is high. The mapping is illustrated by blue and green arrows. Note how the behavior changed in the third DFI clock cycle where dfi_we_n_p1 is high and dfi_we_n_p0 is low; simply the associated phase0 of that cycle is zero while phase 1 is high in the DFI PHY clk domain.

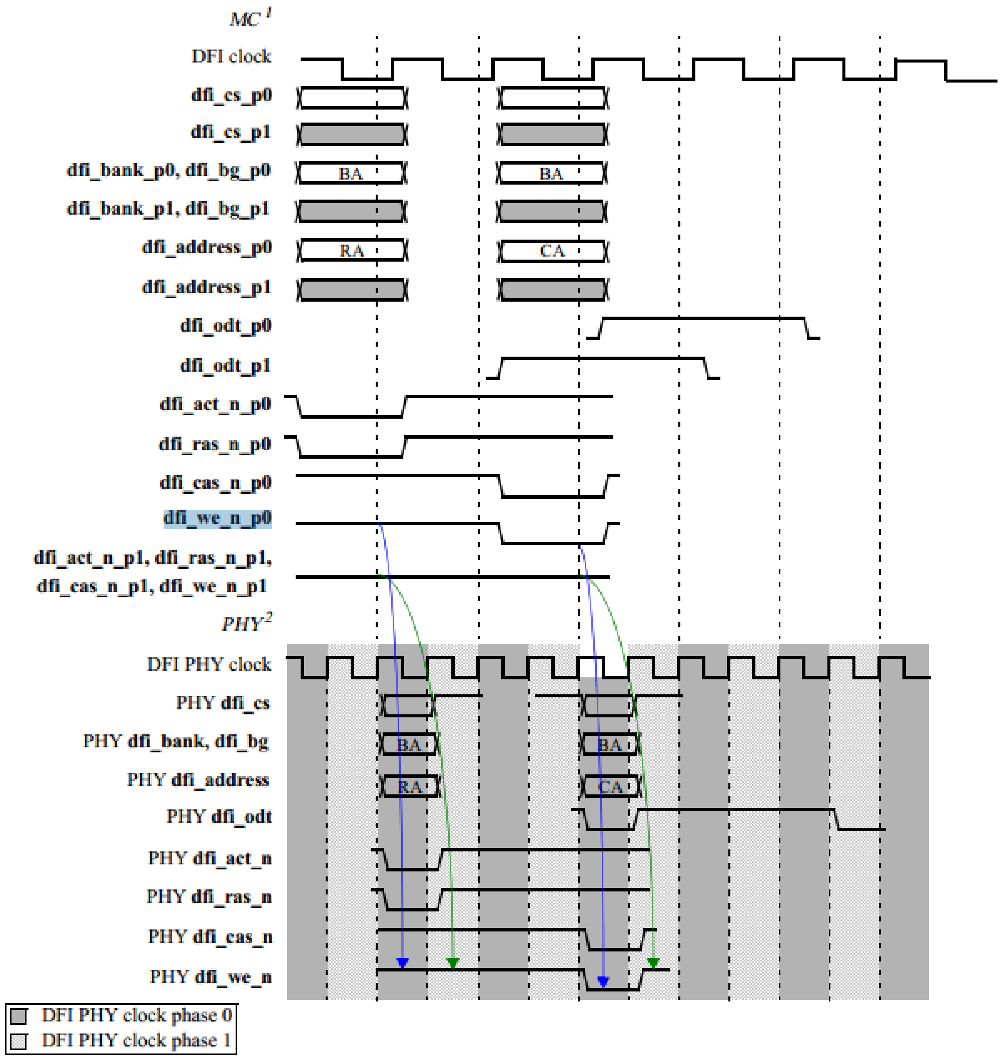


Figure 17. Frequency ratio system with 2 phases.

2.1.7 CRC

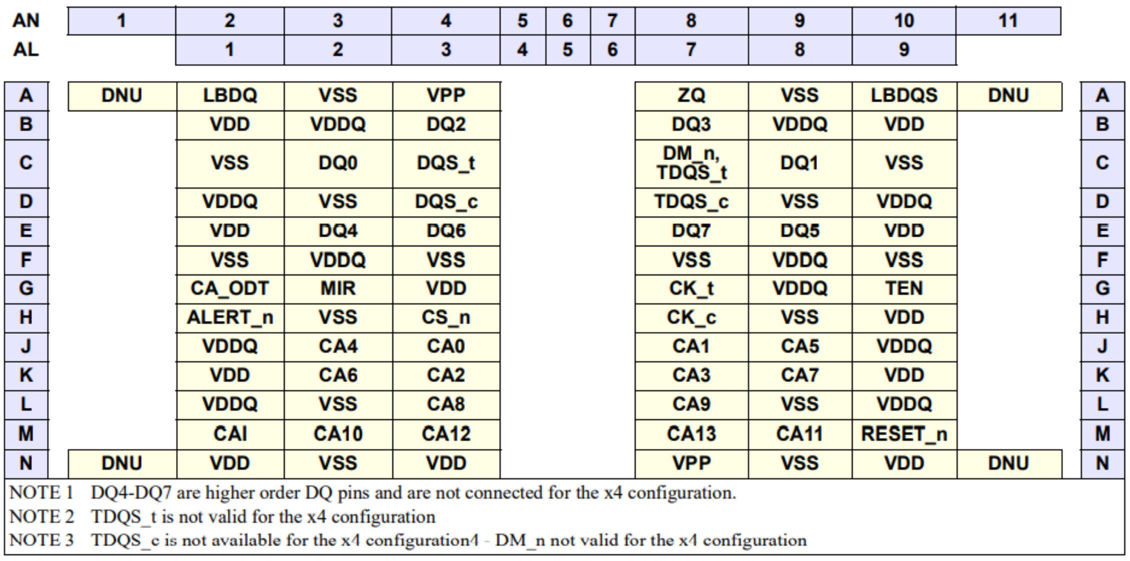
Table 5 shows the Phycrc_mode signal which indicates the CRC validation happens in PHY or MC.

Table 5. Write Data Interface Programmable Parameters.

Parameter	Defined By	Description
phycrc_mode	PHY	Sends CRC data as part of the data burst. • ’b0 = CRC code generation and validation performed in the MC. • ’b1 = CRC code generation and validation performed in the PHY.
phydbi_mode	PHY	Determines which device generates DBI and inverts the data. • ’b0 = DBI generation and data inversion performed in the MC. • ’b1 = DBI generation and data inversion performed in the PHY.

2.2 JEDEC [7]

2.2.1 Package



AN	1	2	3	4	5	6	7	8	9	10	11	
AL		1	2	3	4	5	6	7	8	9		
A	DNU	LBDQ	VSS	VPP				ZQ	VSS	LBDQS	DNU	A
B		VDD	VDDQ	DQ2				DQ3	VDDQ	VDD		B
C		VSS	DQ0	DQS_t				DM_n, TDQS_t	DQ1	VSS		C
D		VDDQ	VSS	DQS_c				TDQS_c	VSS	VDDQ		D
E		VDD	DQ4	DQ6				DQ7	DQ5	VDD		E
F		VSS	VDDQ	VSS				VSS	VDDQ	VSS		F
G		CA_ODT	MIR	VDD				CK_t	VDDQ	TEN		G
H		ALERT_n	VSS	CS_n				CK_c	VSS	VDD		H
J		VDDQ	CA4	CA0				CA1	CA5	VDDQ		J
K		VDD	CA6	CA2				CA3	CA7	VDD		K
L		VDDQ	VSS	CA8				CA9	VSS	VDDQ		L
M		CAI	CA10	CA12				CA13	CA11	RESET_n		M
N	DNU	VDD	VSS	VDD				VPP	VSS	VDD	DNU	N

NOTE 1 DQ4-DQ7 are higher order DQ pins and are not connected for the x4 configuration.
NOTE 2 TDQS_t is not valid for the x4 configuration
NOTE 3 TDQS_c is not available for the x4 configuration4 - DM_n not valid for the x4 configuration

Figure 18. Package of the DRAM.

2.2.2 Pinout Description

Table 6. Pinout Description.

Symbol	Type	Function
CK_t, CK_c	Input	Differential clock inputs.
CS_n	Input	provides for external Rank selection on systems with multiple Ranks.
DM_n, DMU_n DML_n	Input	Input data mask. Input data is masked when DM_n is sampled LOW coincident with that input data during a Write access.
CA[13:0]	Input	Command/Address Inputs

RESET_n	Input	Active Low Asynchronous Reset
DQ	Input/output	Data Input/Output: Bi-directional data bus
DQS_t, DQS_c DQSU_t, DQSU_c DQSL_t,DQSL_c	Input/output	Edge-aligned with read data, centered in write data For the x16, DQSL corresponds to the data on DQL0-DQL7; DQSU corresponds to the data on DQU0-DQU7.
TDQS_t, TDQS_c	Output	Termination Data Strobe
ALERT_n	output	If there is an error in CRC, then Alert_n goes LOW. During Connectivity Test mode, this pin works as input.
TEN	Input	Connectivity Test Mode Enable:
MIR	Input	Mirrored mode vs. Standard mode. SDRAM internally swaps even numbered CA with the next higher odd number CA.
CAI	Inout	internally inverts the logic level present on all the CA signals
CA_ODT	Input	ODT for Command and Address. Apply Group A if High and apply Group B if Low
LBDQ(Loopback)	Output	Training of DQ is done using the newly defined Loopback pin
LBDQS(Loopback)	Output	Training of DQS is done using the newly defined Loopback pin
RFU	Input/output	Reserved for future use
NC		No Connect
VDDQ	Supply	DQ Power supply 1.1V
VDD	Supply	1.1V
VSS	Supply	Ground
VPP	Supply	DRAM activating power supply 1.8V
ZQ	Reference	The pin is connected to 240ohm for reference

TDQS provides termination on both the TDQS and TDQS# balls that is equal to the termination selected on DQS and DQS# for signal integrity issues. For MIR, it is needed because of the placement of DRAM chips on the front and the backside of the DIMM PCB, so CA pins are switched to match the corresponding DRAM chip. For CAI, it is needed as the new DDR5 has double the rows of DRAM chips compared with previous versions so it inverts the CA bits of the upper row relative to the lower row as shown on the following figure resulting in reduction in power delivery noise (difference between source and destination power).

2.2.3 Addressing

Table 7. 8GB addressing in DDR5.

Configuration		**2 Gb x4**	**1 Gb x8**	**512 Mb x16**
Bank address	BG Address	BG0~BG2	BG0~BG2	BG0~BG1
	Bank Address in a BG	BA0	BA0	BA0
	# BG / # Banks per BG / # Banks	8 / 2 / 16	8 / 2 / 16	4 / 2 / 8
Row Address		R0~C15	R0~R15	R0~C15
Column Address		C0~C10	C0~C9	C0~C9
Page size		1KB	1KB	2KB
Chip IDs / Maximum Stack Height		CID0~3 / 16H	CID0~3 / 16H	CID0~3 / 16H

Chip ID determines the ID of a specific die in a 3D stacked DRAM chip. The page size is per bank, the number of bits loaded into the sense amplifiers when a row is activated.

$$PageSize = 2^{\#ColBits} * \left(\frac{Width}{8}\right)$$

$$TotalCapacity = \#Banks * 2^{\#Row\ Address\ Bits} * 2^{\#Col\ Address\ Bits} * Width$$

2.2.4 Commands

Table 8. DDR5 command truth table.

Function	**Abv.**	**CS**	**CA Pins**														**Note**
			CA 0	**CA 1**	**CA 2**	**CA 3**	**CA 4**	**CA 5**	**CA 6**	**CA 7**	**CA 8**	**CA 9**	**CA 10**	**CA 11**	**CA 12**	**CA 13**	
Mode Register Write	MRW	L	H	L	H	L	L	MRA 0	MRA 1	MRA 2	MRA 3	MRA 4	MRA 5	MRA 6	MRA 7	V	1,6
		H	OP 0	OP 1	OP 2	OP 3	OP 4	OP5	OP6	OP7	V	V	CW	V	V	V	
Mode Register Read	MRR	L	H	L	H	L	H	MRA 0	MRA 1	MRA 2	MRA 3	MRA 4	MRA 5	MRA 6	MRA 7	V	6,7, 13
		H	L	L	V	V	V	V	V	V	V	V	CW	V	V	V	
Mode Register Read **NT**	MRR-**NT**	L	H	L	H	L	H	V	V	V	V	V	V	V	V	V	7,14
		L	L	L	V	V	V	V	V	V	V	V	V	V	V	V	
Write	WR	L	H	L	H	H	L	BL*=L	BA0	BA1	BG0	BG1	BG2	CID0	CID1	CID2	3,5,7, 10,12
		H	C2	C3	C4	C5	C6	C7	C8	C9	C10	V	H	WR_P =L	V	CID3	
Write Auto Pre-charge	WRA	L	H	L	H	H	L	BL*=L	BA0	BA1	BG0	BG1	BG2	CID0	CID1	CID2	3,5,7, 10,12
		H	C2	C3	C4	C5	C6	C7	C8	C9	C10	V	L	WR_P =L	V	CID3	
Write **NT**	WR-**NT**	L	H	L	H	H	L	BL*=L	V	V	V	V	V	V	V	V	3,5,7, 10
		L	V	V	V	V	V	V	V	V	V	V	V	V	V	V	
Read	RD	L	H	L	H	H	H	BL*=L	BA0	BA1	BG0	BG1	BG2	CID0	CID1	CID2	3,5,7, 12
		H	C2	C3	C4	C5	C6	C7	C8	C9	C10	V	H	V	V	CID3	
Read Auto Pre-charge	RDA	L	H	L	H	H	H	BL*=L	BA0	BA1	BG0	BG1	BG2	CID0	CID1	CID2	3,5,7, 12
		H	C2	C3	C4	C5	C6	C7	C8	C9	C10	V	L	V	V	CID3	
Read **NT**	RD-**NT**	L	H	L	H	H	H	BL*=L	V	V	V	V	V	V	V	V	3,5,7
		L	V	V	V	V	V	V	V	V	V	V	V	V	V	V	

Two cycle commands have two Chip select values Low and High. CA5:BL*=L, the command places the DRAM into the alternate Burst mode described by MR0[1:0] instead of the default Burst Length 16 mode.

2.2.4 Two Cycle Commands

DDR5 DRAM commands ACT, WRP, WRPA and MRW are 2-cycle commands. The DRAM will not execute these 2-cycle commands if the CS_n is LOW on the 2nd cycle (command cancel). Figure 19 shows a 2-cycle command example.

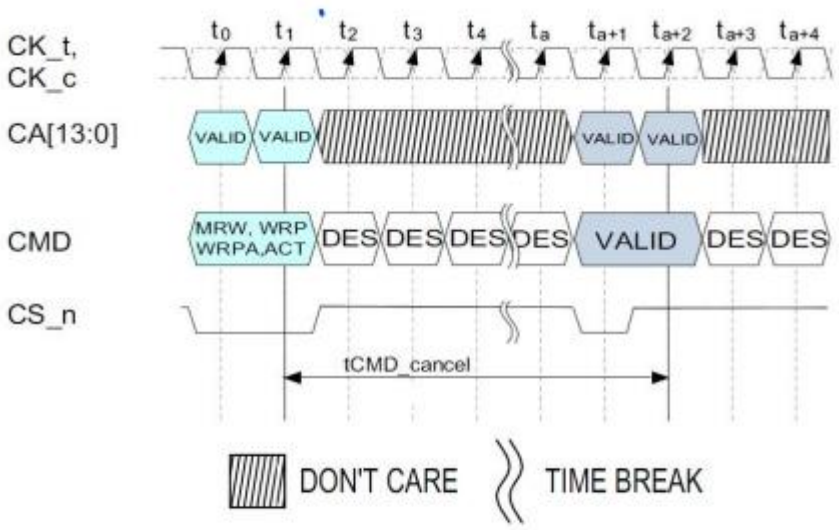


Figure 19. 2-cycle commands example.

2.2.5 Burst Length

The burst length is defined by bits OP [1:0] of Mode Register MR0. Table 9 shows the MR0 register and OP-code definitions. Burst length options include BC8 OTF, BL16, BL32 (optional) and BL32 OTF. On The Fly (OTF) means that the BL can be changed dynamically. C0 – C10 (Column Address Bits) are used to specify a starting column address. The ordering of accesses within a burst is determined by the burst length and the starting column address.

Table 9. MR0 Register and OP-Code Bit Definitions.

OP7	**OP6**	**OP5**	**OP4**	**OP3**	**OP2**	**OP1**	**OP0**
RFU			CAS Latency			Burst Length	

Function	**Type**	**OP**	**Description/Data**
Burst Length	R/W	OP[1:0]	**00=BL16, 01=BC8 OTF, 10=BL32, 11=BL32 OTF**
CAS Latency	R/W	OP[6:2]	**00000=22, 00001=24, 00010=26, 00011=28, 00100=30, 00101=32,**
			00110=34, 00111=36, 01000=38, 01001=40, 01010=42, 01011=44,
			01100=46, 01101=48, 01110=50, 01111=52, 10000=54, 10001=56,
			10010=58, 10011=60, 10100=62, 10101=64, 10110=66,
			10111-11111=RFU
RFU	RFU	OP[7]	RFU

2.2.6 Programmable Preamble & Post-amble

The strobe line may be used to signal the beginning and end of a burst of data. A read “preamble cycle" may be used to indicate the beginning of a read burst and a "post-amble cycle” may be used to signal the end of the read burst. A DDR SDRAM controller and associated memory may perform read and write operations synchronously based on a periodic signal transmitted over a “strobe' line. But due to high frequencies signal integrity issues between the strobe and data can occur leading to memory errors. Read Preamble is configured as 1tCK, 2tCK (two unique modes), 3tCK and 4tCK via MR8: OP[2:0]. Read Post-amble is configured as 0.5tCK or 1.5tCK via MR8: OP[6]. DQS shall have an option to drive early by x-tCK to accommodate different HOST receiver designs as controlled by the Read DQS Offset in MR40: OP[2:0] but this will be processed by the analog part of the PHY. Figure 20 shows a waveform of the usage of preamble signal.

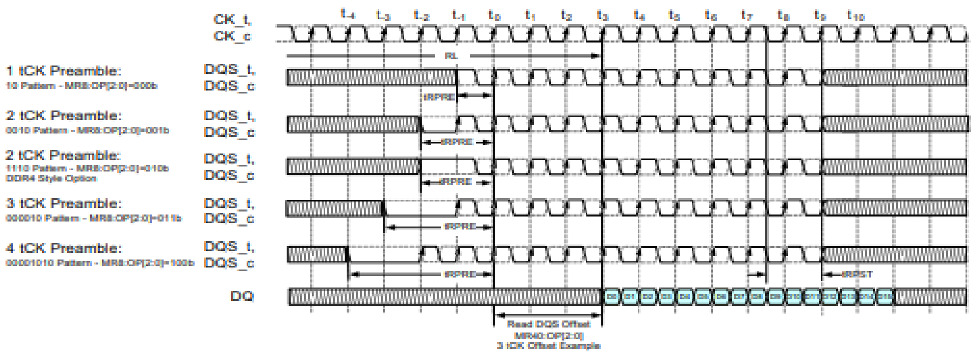


Figure 20. Preamble usage example.

2.2.7 Inter-amble

The post-amble and pre-amble size shall not force the host to add command gaps in the command interval just to satisfy post-amble or pre-amble settings. In Read to Read operations with tCCD(Column to column delay)=BL(Burst Length)/2, post-amble for 1st command and preamble for 2nd command shall disappear to create consecutive DQS latching edge for seamless burst operations. If the post-amble and pre-ambles overlap, the toggles take precedence over static preambles. Figure 21 shows an interamble example.

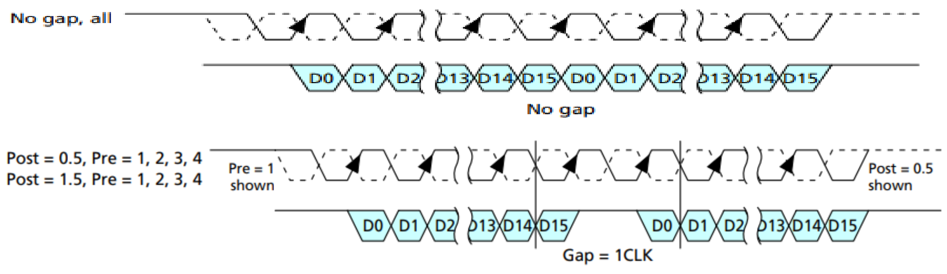

No gap, all
D0 D1 D2 D13 D14 D15 D0 D1 D2 D13 D14 D15
No gap
Post = 0.5, Pre = 1, 2, 3, 4
Post = 1.5, Pre = 1, 2, 3, 4
Pre = 1 shown
Post = 0.5 shown
D0 D1 D2 D13 D14 D15 D0 D1 D2 D13 D14 D15
Gap = 1CLK

Figure 21. Interamble example.

Inter-amble scenarios are based on the pre-amble, post-amble and the gap between these two given by table 10.

Table 10. Interamble scenarios.

tCCD	Preamble	Postamble	Interamble
min+1	1, 2, 3, 4	0.5	10
min+2	3	0.5	0010
	4	0.5	01010
	1, 2, 3, 4	1.5	01010

min+3	4	0.5	0001010
	3	1.5	0100010
	4	1.5	0101010
min+4	4	1.5	010001010

2.2.8 *Read Burst Operation*

During the READ or WRITE. MR0[1:0] is used to select burst operation mode. But it could be discarded and the default value of 16 is used in a particular read, a single read operation is shown below in figure 22.

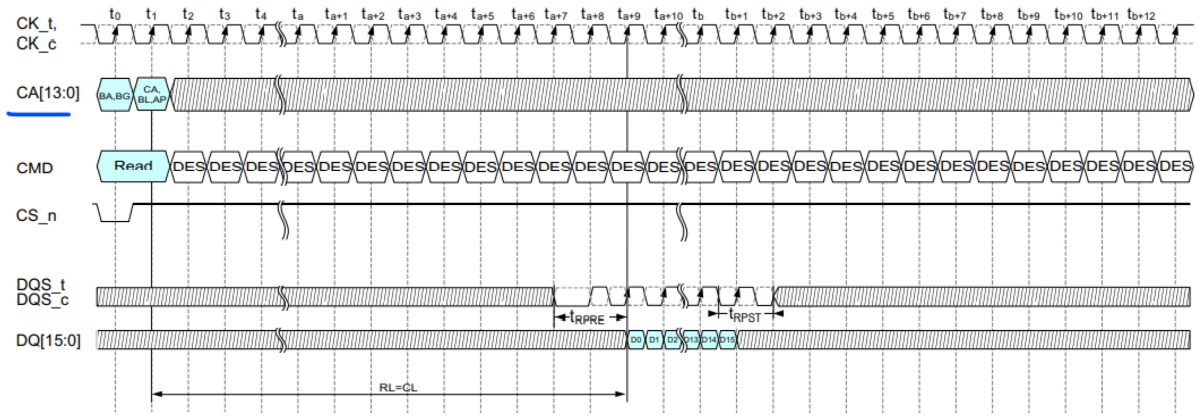


Figure 22: Read operation

In this example, read DQS offset timing is set to 0 clocks, BL is 16, Preamble = 2tCK - 0010 Pattern Preamble, 1.5tCK Postamble. Also note that DES commands are shown for ease of illustration; other commands may be valid at these times.

2.2.9 *Back-to-Back Reads Different Ranks*

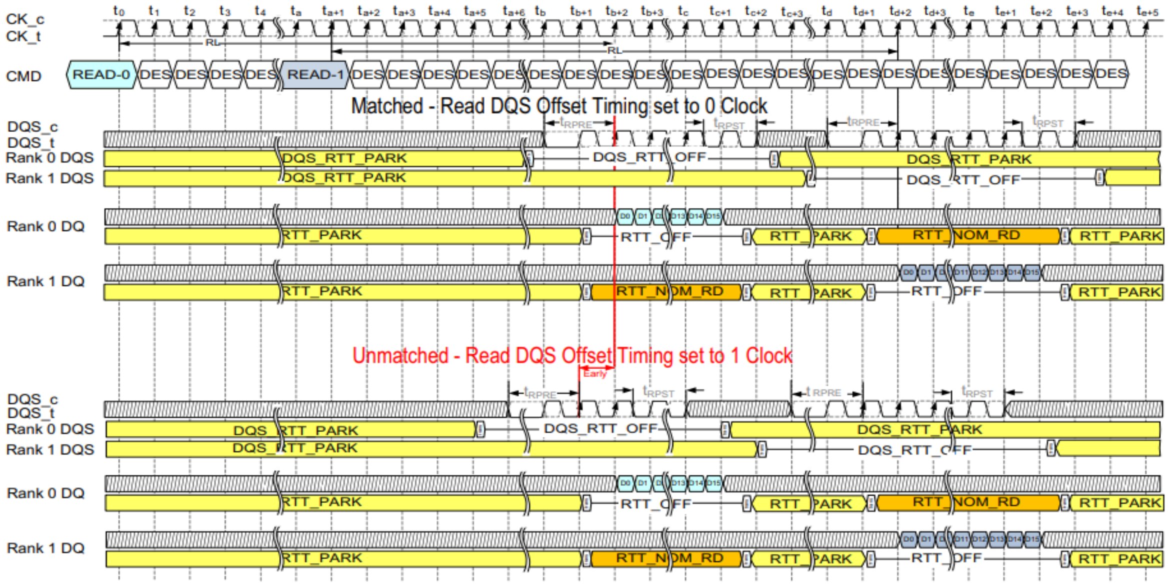


Figure 23. Back-to-back reads from different banks.

The figure illustrates two different examples, the top with the default setting for Read DQS Offset = 0 Clock, the lower with a 1 Clock setting. In the lower case, the DQS is started 1 clock earlier than normal with respect to RL (CL). Note that Read Latency was not violated and data of both reads were driven at the same time in both examples.

2.2.10 Burst Read Followed by Pre-charge

To illustrate this section, the following timing parameters first has to be defined:

- tRTP (Read to Precharge) → minimum internal Read to PRE timing
- tRAS → minimum ACT to PRE timing. (Time taken from ACTIVATE command for the DRAM to move data from row to sense amplifiers and restore the data again).
- tRP → minimum time after precharge. (Time needed for the DRAM to be precharged for another row access)
- tRC (Row Cycle) → tRAS + tRP (Read Cycle) between 2 Row accesses.

Note that new bank active command may be issued to the same bank when tRPmin and tRCmin are satisfied simultaneously. A read followed by pre-charge operation is shown in figure 24

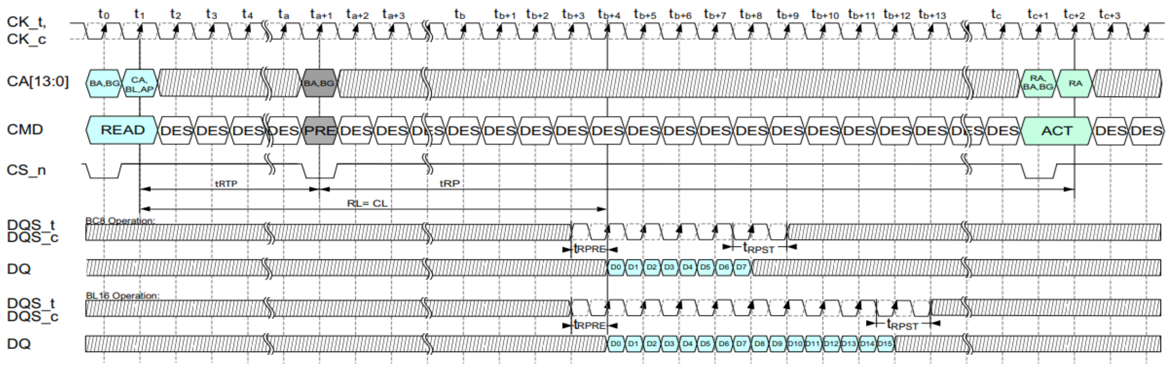


Figure 24:Read followed by pre-charge

Figure shows that pre-charge can happen before data gets out on the data bus as the row data already moved from sense amp to I/O gating which has a read data latch (tRTP).

2.2.11 Read Timing Illustration

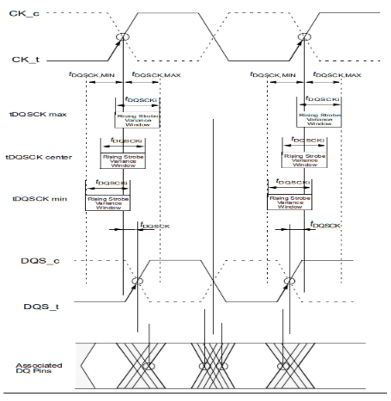


Figure 25:DQ bus timing

Note that data Strobe is synchronized with the clock but it has some deviation from the clock edge by a variance window. Also, previously tested that when the clock edge comes the strobe will change its signal either within max, center or min variance window. Moreover, the DQ is shown like that to show that there are parallel bits coming from different bank groups.

2.2.12 Read Burst Operation for Optional BL32 Mode

BL32 in BL32 OTF Mode is shown below in figure 26

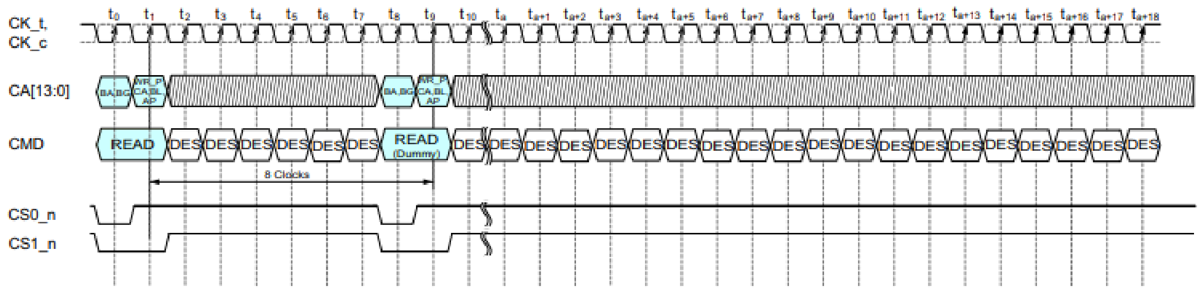


Figure 26: Read Timing for fixed BL32 and BL32 in BL32 OTF mode

Note that a dummy CAS command is required for the second half of the transfer of BL32 with a delay of 8 clocks (16 bits).

BL16 in BL32 OTF Mode is shown below in figure 27

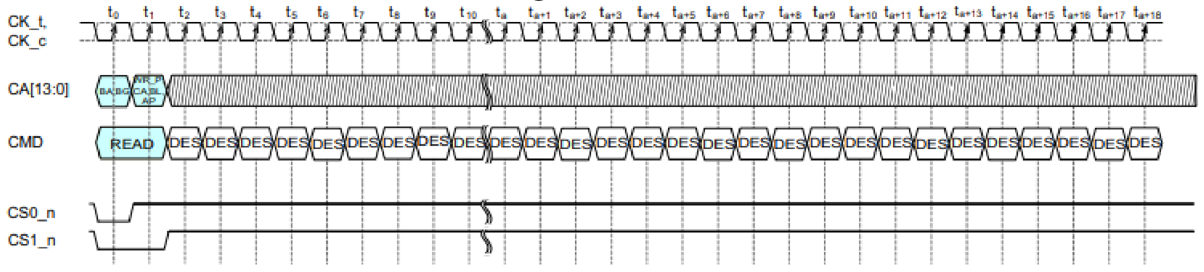


Figure 27: Read Timings for BL16 in BL32 OTF mode

The figure shows BL16 read operation when MR0 is programmed to use BL32 OTF mode. In this case, no dummy RD command is required as transfer size is BL16.DDR5 DRAM supports an optional fixed BL32 mode and optional BL32 OTF (On the fly) mode for x4 devices only.

2.2.13 Bank Group Read.

The minimum allowed timing between reads is dependent on where do the consecutive reads fall. Figures 28 and 29 illustrate that difference

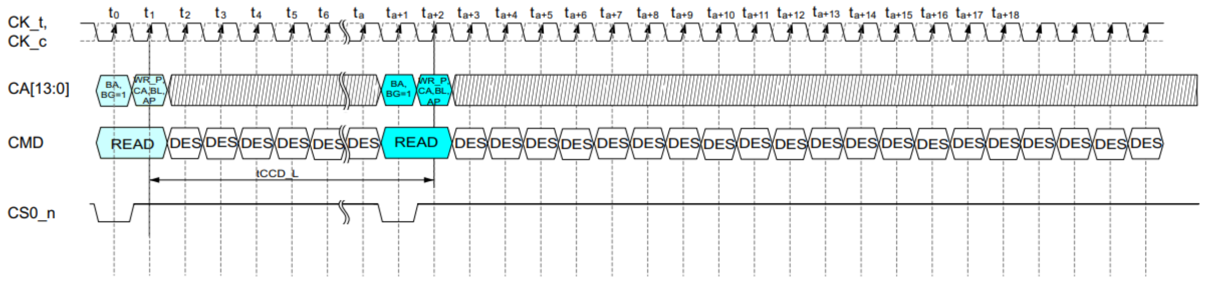


Figure 28: Read to Read to Same Bank Group for BL16 in BL32 OTF

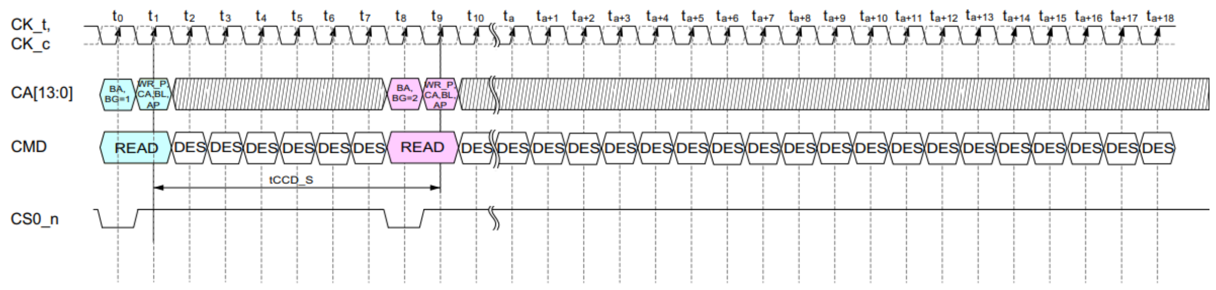


Figure 29:Read to Read to Different Bank Group for BL16 in BL32 OTF.

Note that bank accesses to different banks' groups require less time delay between accesses than bank accesses to within the same bank's group. Bank accesses to different bank groups require tCCD_S (or short) delay between commands while bank accesses within the same bank group require tCCD_L (or long) delay between commands.where tCCD_s is limited by burst length as both BGs can work in parallel. And tCCD_l=>limited by back access because they share the same resources.

2.2.14 Read with Auto Pre-charge

A read with auto pre-charge for fixed BL32 and BL32 in BL32 OTF Mode is shown in figure 30

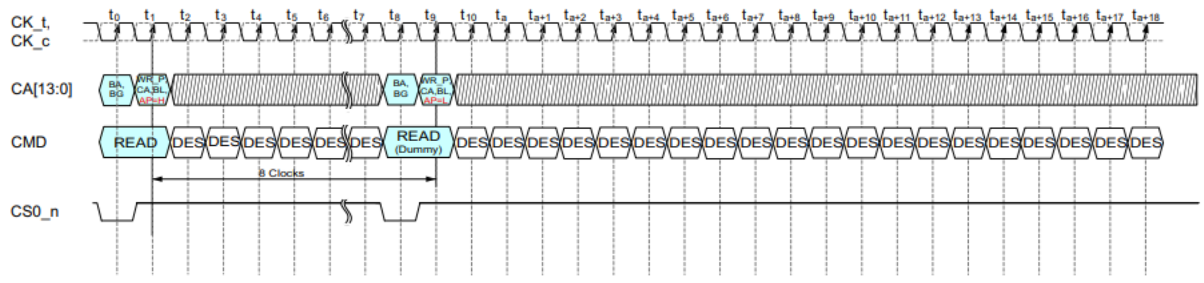


Figure 30: A read with auto pre-charge

Note that auto pre-charge is in the second read not the first. If it was in the first the row would get closed and no further reads can happen. Also, CA bits other than C10 and AP in the dummy CAS command are the same as the first CAS command.

2.2.15 Read and Write command interval

Table below illustrates the minimum command separation in case of consecutive read and write. This process is also illustrated in figures 31 and 32 below.

Table 11: Command Separation

Bank Group	Timing Parameter	Value
Same	Minimum read to write	CL - CWL + RBL/2 + 2tCK - (Read DQS offset) + (tRPST - 0.5tCK) + tWPRE
	Minimum write to read	CWL + WBL/2 + tWTR_L
	Write to read AP same bank	CWL + WBL/2 + tWTRA
Different	Minimum read to write	CL - CWL + RBL/2 + 2tCK - (Read DQS offset) + (tRPST - 0.5tCK) + tWPRE
	Minimum write to read	CWL + WBL/2 + tWTR_S

Note that write to read time is long when accessing the same bank group. But write to read time is short when accessing different bank group.

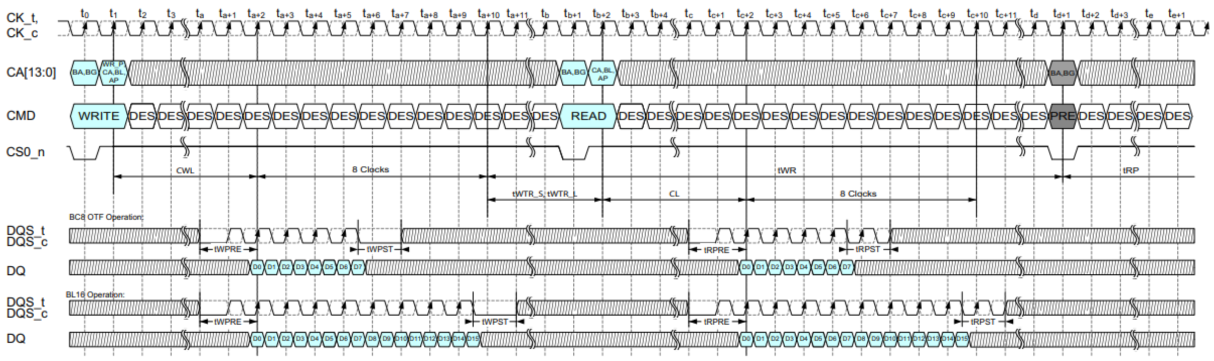


Figure 31:Timing diagram for write to read

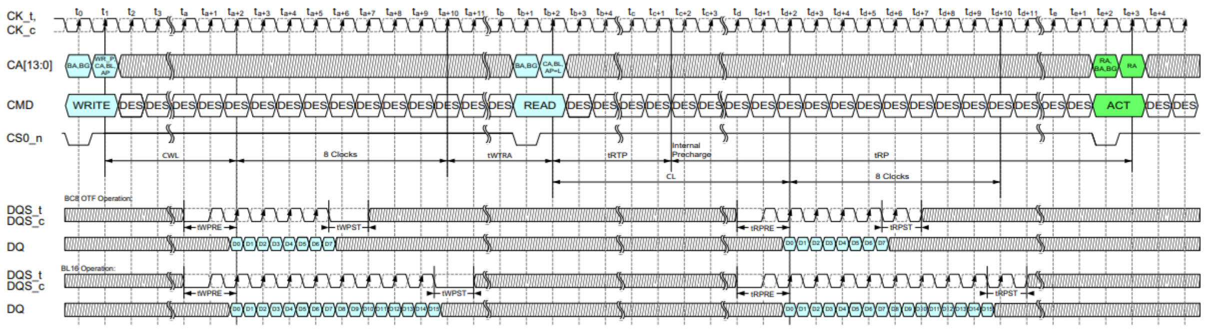
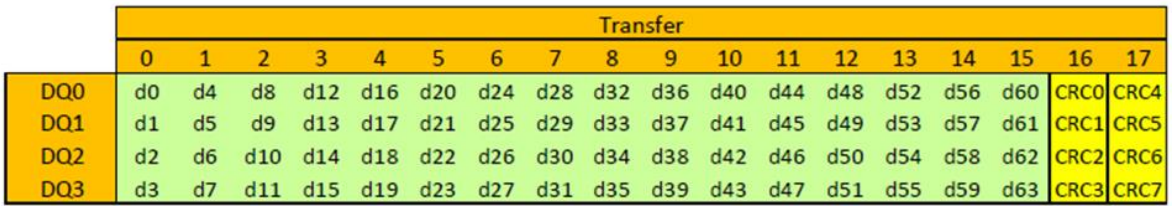


Figure 32:Timing diagram for write to read Auto Pre-charge in the same bank

2.2.16 CRC

DDR5 supports CRC for write and read operations. Read and Write CRC can be enabled by separate mode register bits. The CRC polynomial is the ATM-8 HEC, X^8+X^2+X+1, same used on DDR4. Refer to appendices for CRC pseudocode as provided by the standard.

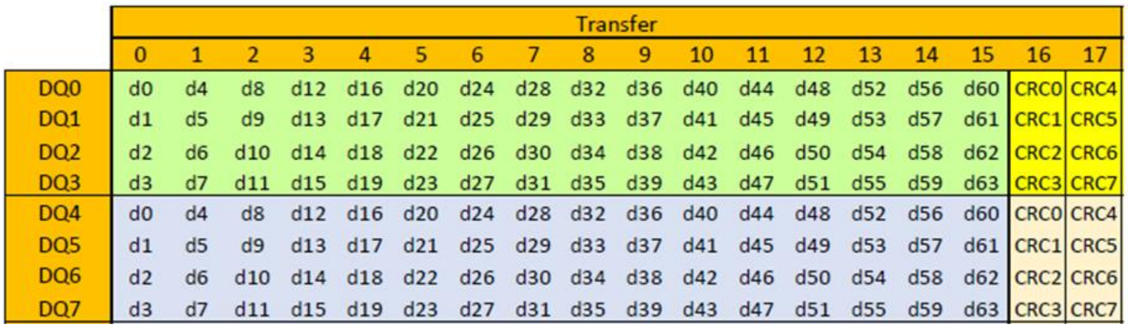
This bit mapping is common between write and read CRC operations and it differs based on DRAM width as shown below in figures 33,34,35.



	Transfer																	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
DQ0	d0	d4	d8	d12	d16	d20	d24	d28	d32	d36	d40	d44	d48	d52	d56	d60	CRC0	CRC4
DQ1	d1	d5	d9	d13	d17	d21	d25	d29	d33	d37	d41	d45	d49	d53	d57	d61	CRC1	CRC5
DQ2	d2	d6	d10	d14	d18	d22	d26	d30	d34	d38	d42	d46	d50	d54	d58	d62	CRC2	CRC6
DQ3	d3	d7	d11	d15	d19	d23	d27	d31	d35	d39	d43	d47	d51	d55	d59	d63	CRC3	CRC7

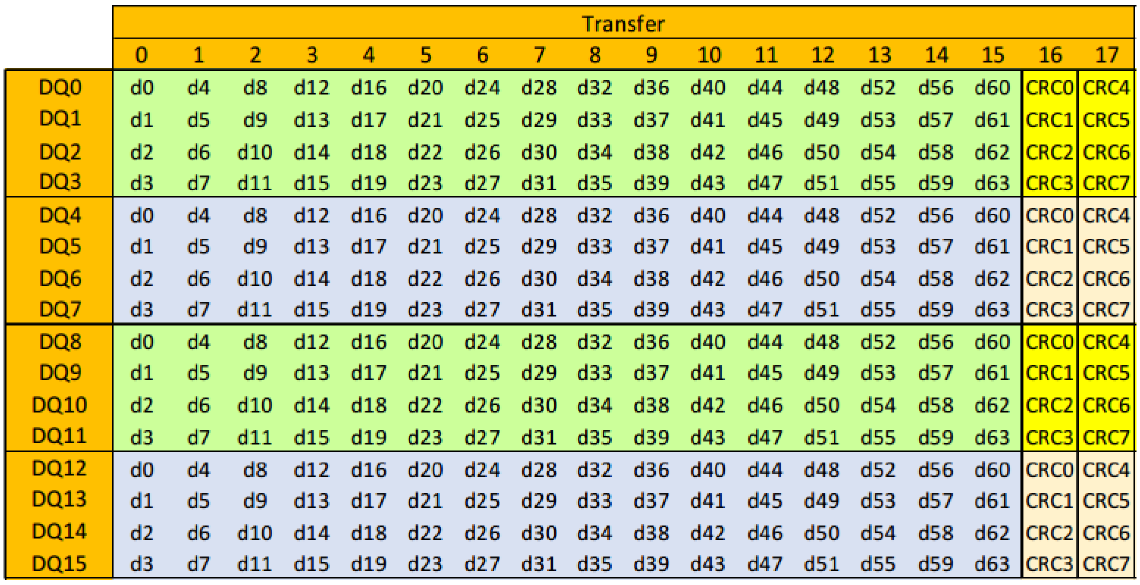
Figure 33: CRC Bit mapping for x4 device

This bit mapping is common between write and read CRC operations. x8 devices have two DQ nibbles and each DQ nibble has its own eight CRC bits to protect 64 data bits. Therefore, a x8 device will have two identical CRC trees implemented.



	Transfer																	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
DQ0	d0	d4	d8	d12	d16	d20	d24	d28	d32	d36	d40	d44	d48	d52	d56	d60	CRC0	CRC4
DQ1	d1	d5	d9	d13	d17	d21	d25	d29	d33	d37	d41	d45	d49	d53	d57	d61	CRC1	CRC5
DQ2	d2	d6	d10	d14	d18	d22	d26	d30	d34	d38	d42	d46	d50	d54	d58	d62	CRC2	CRC6
DQ3	d3	d7	d11	d15	d19	d23	d27	d31	d35	d39	d43	d47	d51	d55	d59	d63	CRC3	CRC7
DQ4	d0	d4	d8	d12	d16	d20	d24	d28	d32	d36	d40	d44	d48	d52	d56	d60	CRC0	CRC4
DQ5	d1	d5	d9	d13	d17	d21	d25	d29	d33	d37	d41	d45	d49	d53	d57	d61	CRC1	CRC5
DQ6	d2	d6	d10	d14	d18	d22	d26	d30	d34	d38	d42	d46	d50	d54	d58	d62	CRC2	CRC6
DQ7	d3	d7	d11	d15	d19	d23	d27	d31	d35	d39	d43	d47	d51	d55	d59	d63	CRC3	CRC7

Figure 34:CRC data bit mapping for x8 devices



	Transfer																	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
DQ0	d0	d4	d8	d12	d16	d20	d24	d28	d32	d36	d40	d44	d48	d52	d56	d60	CRC0	CRC4
DQ1	d1	d5	d9	d13	d17	d21	d25	d29	d33	d37	d41	d45	d49	d53	d57	d61	CRC1	CRC5
DQ2	d2	d6	d10	d14	d18	d22	d26	d30	d34	d38	d42	d46	d50	d54	d58	d62	CRC2	CRC6
DQ3	d3	d7	d11	d15	d19	d23	d27	d31	d35	d39	d43	d47	d51	d55	d59	d63	CRC3	CRC7
DQ4	d0	d4	d8	d12	d16	d20	d24	d28	d32	d36	d40	d44	d48	d52	d56	d60	CRC0	CRC4
DQ5	d1	d5	d9	d13	d17	d21	d25	d29	d33	d37	d41	d45	d49	d53	d57	d61	CRC1	CRC5
DQ6	d2	d6	d10	d14	d18	d22	d26	d30	d34	d38	d42	d46	d50	d54	d58	d62	CRC2	CRC6
DQ7	d3	d7	d11	d15	d19	d23	d27	d31	d35	d39	d43	d47	d51	d55	d59	d63	CRC3	CRC7
DQ8	d0	d4	d8	d12	d16	d20	d24	d28	d32	d36	d40	d44	d48	d52	d56	d60	CRC0	CRC4
DQ9	d1	d5	d9	d13	d17	d21	d25	d29	d33	d37	d41	d45	d49	d53	d57	d61	CRC1	CRC5
DQ10	d2	d6	d10	d14	d18	d22	d26	d30	d34	d38	d42	d46	d50	d54	d58	d62	CRC2	CRC6
DQ11	d3	d7	d11	d15	d19	d23	d27	d31	d35	d39	d43	d47	d51	d55	d59	d63	CRC3	CRC7
DQ12	d0	d4	d8	d12	d16	d20	d24	d28	d32	d36	d40	d44	d48	d52	d56	d60	CRC0	CRC4
DQ13	d1	d5	d9	d13	d17	d21	d25	d29	d33	d37	d41	d45	d49	d53	d57	d61	CRC1	CRC5
DQ14	d2	d6	d10	d14	d18	d22	d26	d30	d34	d38	d42	d46	d50	d54	d58	d62	CRC2	CRC6
DQ15	d3	d7	d11	d15	d19	d23	d27	d31	d35	d39	d43	d47	d51	d55	d59	d63	CRC3	CRC7

Figure 35: CRC data bit mapping for x16 devices

This bit mapping is common between write and read CRC operations. x16 devices have four DQ nibbles and each DQ nibble has its own eight CRC bits to protect 64 data bits. Therefore, a x16 device will have four identical CRC trees implemented.

2.2.17 Write CRC for x4, x8 and x16 devices

The following notes represent design constrains thus they are to be listed separately for highlighting purposes

- Write function can be enabled or disabled per each nibble independently in x8 devices, with separate enable MR bits.
- When at least one of two enable bits is set to ‘1’ in x8, the timings of write CRC enable mode is applied to the entire device (both nibbles).
- When only one write CRC is enabled, the DRAM does not check CRC errors on the disabled nibble, and hence the ALERT_n signal and any internal status bit related to CRC error is not

impacted by the disabled nibble.

- For x4 or x16, only one of two write CRC enable bit is used as defined in the MR table (Figure TBD). The unused write CRC enable bit is don't care in x4 and x16, i.e., MR50 OP [2] is set to low for x4 and x16 devices.
- The DRAM checks for an error in received code words per each write CRC enabled nibble by comparing the received checksum against the computed checksum and reports errors using the ALERT_n signal if there is a mismatch in any of nibbles.
- DRAM can write data to the DRAM core without waiting for CRC check for full writes. If bad data is written to the DRAM core, then the controller will retry the transaction and overwrite the bad data. Controller is responsible for data coherency.
- No write latency adder when write CRC is enabled.

2.2.18 Write CRC auto-disable

The following notes represent design constrains thus they are to be listed separately for highlighting purposes

- Write CRC auto-disable mode is enabled by programming the Write CRC auto-disable mode to enable bit MR50:OP[4] to '1'.
- When this mode is enabled, the DDR5 SDRAM counts the number of Write CRC error occurrences per device, regardless of configuration (x4, x8 or x16). When the number of Write CRC errors exceeds the Write CRC Auto-Disable Threshold (between 0 and 127) as programmed in MR51:OP[6:0], the DDR5 SDRAM disables Write CRC error checking of all nibbles and sets the Write CRC auto-disable status bit MR50:OP[5] to '1'.
- To exceed the Write CRC Auto-Disable Threshold, the number of Write CRC errors must occur within the Write CRC Auto-Disable Window described below.
- Unless the Write CRC auto-disable status bit is set, the Write CRC error counter is reset after the predetermined number of writes between 0 and 127, where 0 means an infinite window, as programmed in MR52:OP[6:0], so that the Write CRC error count will accumulate during each Write CRC Auto-Disable Window.
- Once the Write CRC auto-disable status bit is set, the write CRC error checking is not re-enabled at the end of the Write CRC Auto-Disable Window, even though the Write CRC error counter is reset below the threshold value.
- Write CRC error checking can be re-enabled by resetting the Write CRC auto-disable status bit MR50:OP[5] to '0'. This will reset the Write CRC error counter and restart the Write CRC Auto-Disable Window.
- Prior to changing the Write CRC Auto-Disable Threshold as programmed in MR51:OP[6:0] or the Write CRC Auto-Disable Window as programmed in MR52:OP[6:0], the host shall disable the Write CRC Auto-Disable mode, MR50:OP[4]=0.
- Once the updated values have been programmed in MR51 and/or MR52, Write CRC Auto-Disable

mode can be (re)enabled, MR50:OP[4]=1.

- Disabling the Write CRC Auto-Disable mode, if enabled, will reset the DRAM's Write CRC error counter and restart the Write CRC Auto-Disable Window. However, if the Write CRC auto-disable status bit had previously been set to '1', MR50:OP[5]=1, the host is required to set MR50:OP[5]=0 to resume error counting.
- Changes to the Write CRC auto-disable threshold (MR51) and window (MR52) settings are only allowed when the CRC Write auto-disable mode is disabled (MR50[4]=0).
- If the CRC auto-disable threshold is reached and the DDR5 SDRAM was already driving ALERT_n to low due to the current or a previous Write CRC error, then ALERT_n may be released upon satisfying CRC_ALERT_PW_min.
- When Write CRC auto-disable mode is disabled, MR50:OP[4] = 0, Write CRC error counters may remain at reset values even if Write CRC errors occur.

2.2.19 Read CRC for x4, x8 and x16 devices

The following notes represent design constrains thus they are to be listed separately for highlighting purposes

- The controller can check for an error in received code words per nibble by comparing the received checksum against the computed checksum and if there is a mismatch in any of nibbles then controller may retry the transaction.
- Read latency adder when read CRC is enabled depends on data rate as shown in Table below.

Table 12:Read CRC latency adder

Data Rate MT/s	**Read CRC Latency Adder (nCK)**
1980 MT/s ≤ Data Rate ≤ 2100 MT/s	0
2933 MT/s ≤ Data Rate ≤ 6000 MT/s	0
6000 MT/s < Data Rate ≤ 6400 MT/s	2
6800 MT/s < Data Rate ≤ 8400 MT/s	4

2.2.20 CRC Burst Order

The following notes represent design constrains thus they are to be listed separately for highlighting purposes

- When Write CRC is enabled, the CRC bits are calculated based on the sequential burst address order of the write data for the Write command. This sequential order is '0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F' in BL16, and '0,1,2,3,4,5,6,7,T,T,T,T,T,T,T,T' or

'8,9,A,B,C,D,E,F,T,T,T,T,T,T,T,T' in BC8 OTF.

- When Read CRC is enabled, the DDR5 SDRAM's CRC generator overrides the CA burst order bits C3 and C2 to '00', and CRC bits are calculated based on the sequential burst address order of the read data for the Read command. This sequential order is '0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F' in BL16, and '0,1,2,3,4,5,6,7,T,T,T,T,T,T,T,T' or 'T,T,T,T,T,T,T,T,8,9,A,B,C,D,E,F' in BC8 OTF.
- The override values do not modify the actual data burst ordering, and are only used for the CRC calculations. Actual data burst follows the burst order as indicated by C3 and C2 in the Read command.

2.2.21 *Write CRC error handling*

The following notes represent design constrains thus they are to be listed separately for highlighting purposes

- When DRAM detects CRC error on received code words in any of nibbles, then it drives ALERT_n signal to ‘0’ for TBD clocks.
- DRAM will set Write CRC Error Status bit in A[3] of MR50 to '1' upon detecting a CRC error. The Write CRC Error Status bit remains Group At '1' until the host clears it explicitly using an MRW command.
- The controller upon seeing an error as a pulse width will retry the write transactions. The controller understands the worst-case delay for ALERT_n (during init) and can back up the transactions accordingly or the controller can be made more intelligent and try to correlate the write CRC error to a specific rank or a transaction. The controller is also responsible for opening any pages and ensuring that retrying of writes is done in a coherent fashion.
- The latency to ALERT_n signal is defined as tCRC_ALERT

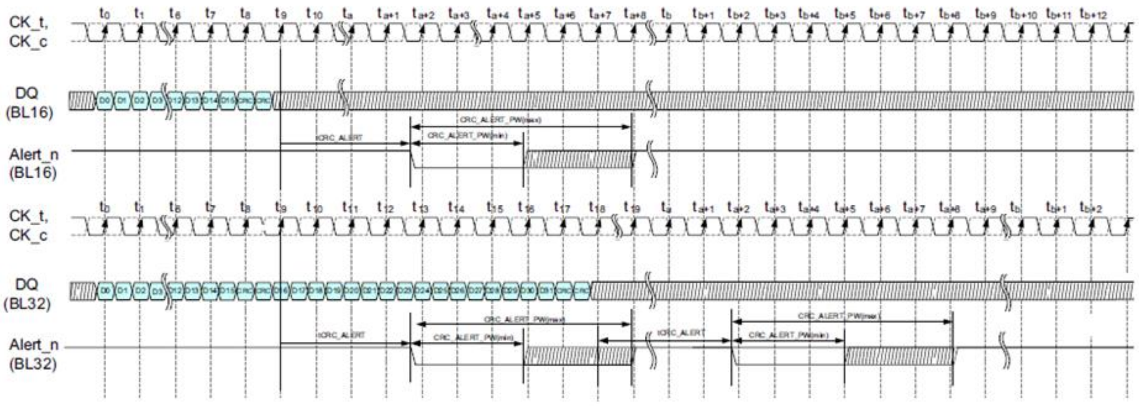


Figure 36:Error reporting timing diagram

Table 13: CRC error handling timing paramters

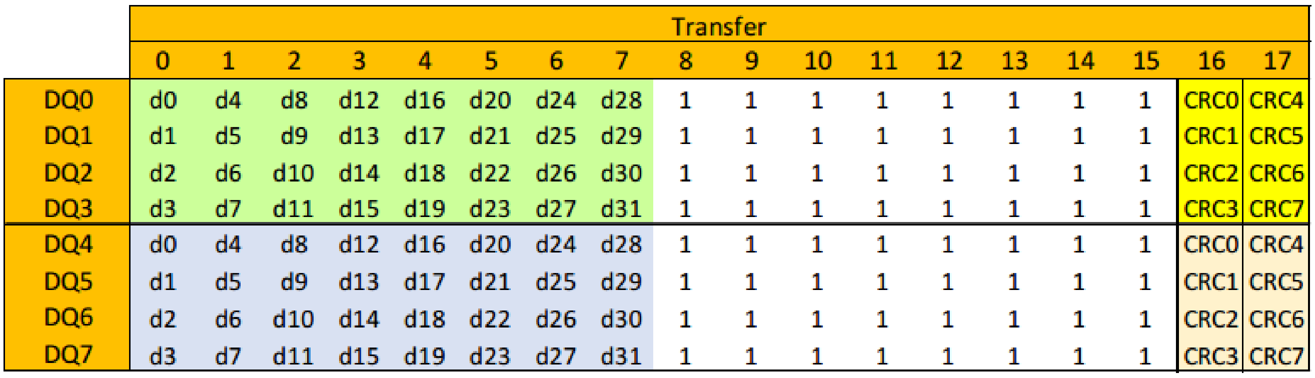
Symbol	Description	Min	Max	Unit
tCRC_ALERT	CRC Alert Delay	3	13	Ns
CRC_ALERT_PW	CRC Alert Pulse	12	20	nCK

2.2.21 CRC bit mapping in BC8 mode

The following notes represent design constrains thus they are to be listed separately for highlighting purposes

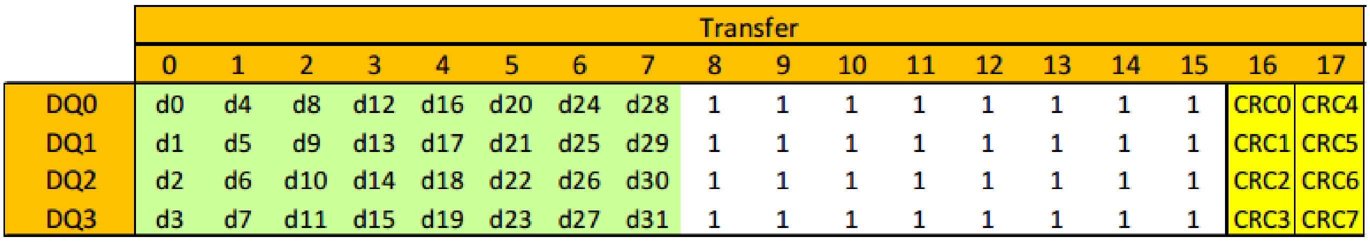
- CRC bits are always transferred on 17th and 18th UI, in BC8 mode.
- When read CRC is enabled during BC8 read, DQ bits are driven high and DQS is toggled by DRAM during the chopped data bursts.

- When write CRC is enabled during BC8 write, DQ bits must be driven high and DQS must be toggled by controller during the chopped data bursts.
- In BC8 mode, read CRC and write CRC bits are calculated with the inputs to the CRC engine for the chopped data bursts replaced by all '1's as shown in figures below.



	Transfer																	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
DQ0	d0	d4	d8	d12	d16	d20	d24	d28	1	1	1	1	1	1	1	1	CRC0	CRC4
DQ1	d1	d5	d9	d13	d17	d21	d25	d29	1	1	1	1	1	1	1	1	CRC1	CRC5
DQ2	d2	d6	d10	d14	d18	d22	d26	d30	1	1	1	1	1	1	1	1	CRC2	CRC6
DQ3	d3	d7	d11	d15	d19	d23	d27	d31	1	1	1	1	1	1	1	1	CRC3	CRC7
DQ4	d0	d4	d8	d12	d16	d20	d24	d28	1	1	1	1	1	1	1	1	CRC0	CRC4
DQ5	d1	d5	d9	d13	d17	d21	d25	d29	1	1	1	1	1	1	1	1	CRC1	CRC5
DQ6	d2	d6	d10	d14	d18	d22	d26	d30	1	1	1	1	1	1	1	1	CRC2	CRC6
DQ7	d3	d7	d11	d15	d19	d23	d27	d31	1	1	1	1	1	1	1	1	CRC3	CRC7

Figure 37: Bit mapping for CRC at BL=8 in x8 device



	Transfer																	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
DQ0	d0	d4	d8	d12	d16	d20	d24	d28	1	1	1	1	1	1	1	1	CRC0	CRC4
DQ1	d1	d5	d9	d13	d17	d21	d25	d29	1	1	1	1	1	1	1	1	CRC1	CRC5
DQ2	d2	d6	d10	d14	d18	d22	d26	d30	1	1	1	1	1	1	1	1	CRC2	CRC6
DQ3	d3	d7	d11	d15	d19	d23	d27	d31	1	1	1	1	1	1	1	1	CRC3	CRC7

Figure 38:Bit mapping for CRC at BL=8 x4 device

2.2.22 CRC bit mapping in BL32 mode

The following notes represent design constrains thus they are to be listed separately for highlighting purposes

- In BL32 mode, CRC bits are separately calculated for the first half and the second half of the data.
- CRC bits for the first half of the data are transferred on 17th and 18th UI, and CRC bits for the second half of the data are transferred on 35th and 36th UI.

2.3 SystemVerilog Language Standard [8]

Formally known as IEEE 1800-2017. It is utilized in hardware description and verification. We used SV for design for its numerous enhancements over classical Verilog. It is less user error prone due to removing previous Verilog ambiguities. It is also utilized in verification due to its powerful UML capabilities.

3- Market and Literature Review

Memory systems poses a real performance bottleneck for computing systems. Since 1980s and the memory-processor gap is on the rise as illustrated in figure below [9] which fuels the continuous effort to make a faster and denser memories.

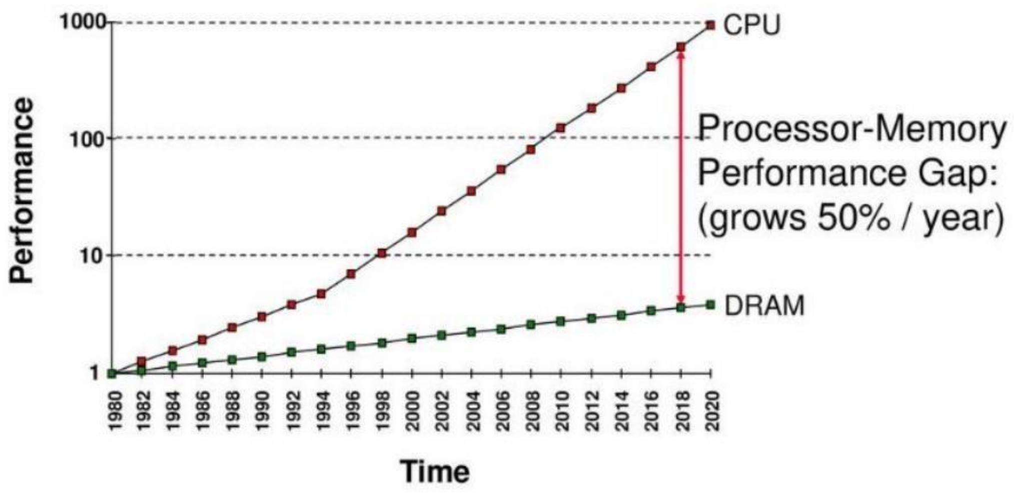


Figure 39:Processor Memory Performance Gap.

3.1 SDRAM Market

The SDRAM market can be segmented into three categories

3.1.1 DRAM Module Manufacturers

The global DDR market size in 2020 is 1.08 billion USD and it is expected to grow by 3x by 2027 with a CAGR of 18.3%. Currently, this market is segmented to DDR1, DDR2, DDR3, DDR4, DDR5until now DDR4 dominates the market with a share of 74% but DDR5 share is expected to rise significantly in the following years [10]. The key DDR5 SDRAM modules providers include:

- Micron Technology, Inc
- Hynix

3.1.2 Hardware that supports DDR5 Manufacturers

As illustrated in table 2 the main two key manufacturers of processors that support DDR5 are Intel and AMD. Currently AMD hasn't released any processors that support DDR5 and only 12th generation Intel processors support DDR5

3.1.3 IP Vendors

The discussed market above does not account for other IPs essential for implementing a functional system. In general, the processor memory interface is shown before requires a physical layer (PHY) that interfaces as well as a Memory Controller (MC). Both MC & PHY have numerous IPs that are compatible with DDR5 SDRAM modules. The key MC& PHY providers include:

- Cadence
- Faraday
- Synopsys
- Rambus

3.2 *History and Evolution of DRAM*

As illustrated previously, since 1970s the performance gap between the memory and processors kept rising and that called for architectural changes in the DRAM, the internal structure of DRAM (Arrays of rows and columns of 1 T cell with a capacitor) remained essentially unchanged, however the interface of the DRAM changed gradually. The changes were either targeting lower latency or higher throughput.

3.2.1 *Structural modifications targeting throughput*

Before interfaces standardization (until mid-70s) most DRAMs were clocked. Basically, DRAM commands and addresses were directly driven by periodic clock signal. After mid-70s the DRAM interface was standardized to be asynchronous. The timing diagram shown in figure below shows a typical read process in an asynchronous DRAM. A read command invokes the row activation via the de-assertion of RAW Address Strobe (RAS) -the bar over the name to affirm that it is an active low signal- during that time RAS signal is de-asserted first the raw address is driven on the address bus hence Column Address Strobe (CAS) is de-asserted and the column address is driven on the address bus hence both RAS and CAS are asserted and valid data is driven on DQ bus [1,10].It is worth noting that each read command reads only 1 bit at a time so if it is desired to read 2 adjacent bits (same raw two adjacent columns), 2 separate read commands must be issued.

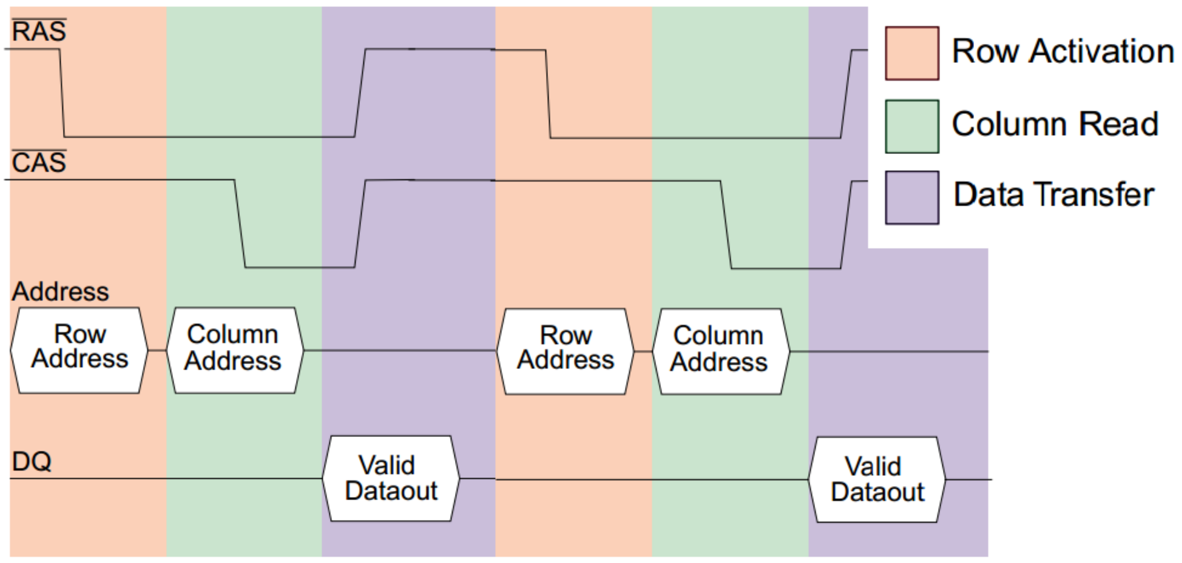


Figure 40:First generation of Asynchronous DRAMS Read process

Generally, in computing it is often the case that when you need to access data from a certain location in memory you would probably need to access data that is stored in a physically close location from the previous location which is known as spatial locality [1,9]. Because generally the number of rows is around 3 orders of magnitude higher than number of columns, raw accessing is actually the bottle neck of memory accessing in terms of time. Thus, the introduction of Fast Page Mode (FPM DRAM) is extremely intuitive, that is holding the raw open and accessing multiple columns of the same raw. The timing diagram of FPM DRAM is shown in figure below. As shown RAS signal is held low while CAS signal got de asserted and re asserted three times and after each de assertion the accompanying address is driven on the address bus hence in this case for the same raw access three bits are accessed (assuming here we are dealing with one array bank). After FPM is was desired to reduce time between column reads which was implemented in Extended Data Out (EDO DRAM) by adding memory elements (latches) between sense amplifiers and output pins. The timing diagram of EDO DRAM is shown in figure below.

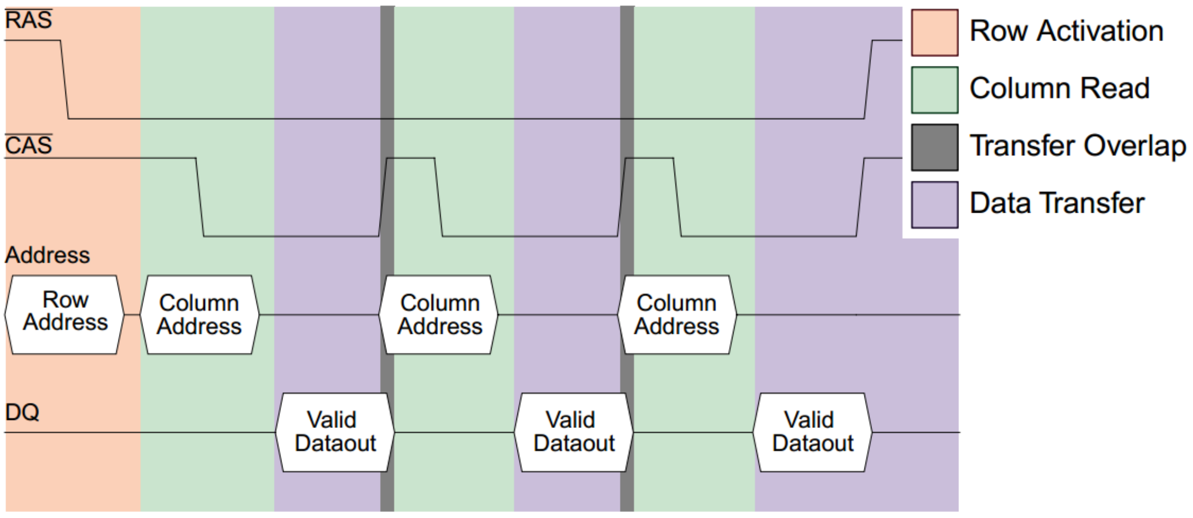


Figure 41:FPM DRAM Read process

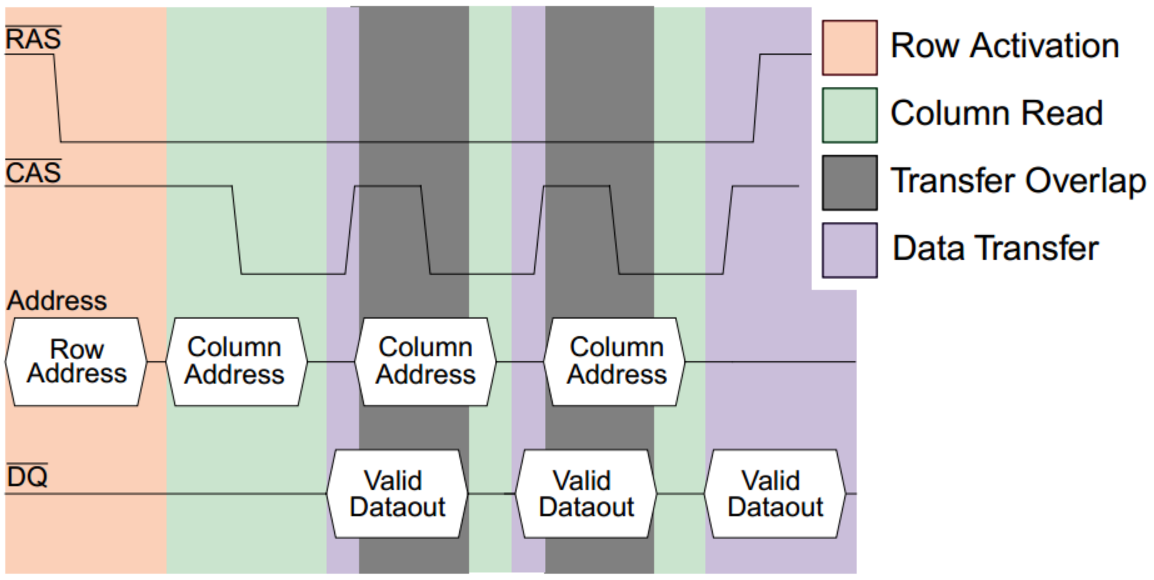


Figure 42:EDO DRAM Read Process

The addition of memory element increased the overlap time between column addressing and data driving hence enabling CAS signal to get de asserted again faster. After these structural modifications the concept of spatial locality in memory read requests induced the revolutionary concept of bursting that is if a desired location is to be read the addresses next to it (same raw adjacent columns) are to be read and stored in a buffer for the same read operation. This concept was implemented in Burst Mode Extended Data Out (BEDO DRAM) where a counter was added before the column muxing action such that for a given “open” row the initial column address is accepted and hence an increment in the column number is achieved and the results are stored in a buffer. The timing diagram of BEDO DRAM is shown in figure below. First RAS signal is de-asserted and raw address is driven on address bus then CAS signal is de-asserted and the column address is driven on the address bus hence it is re-asserted and de-asserted three times with no column addresses driven on the address bus and hence four valid data packets are driven of DQ bus. At each de-assertion and re-assertion of CAS signal the said counter increments the address of the column by 1 hence no new column address is needed [1,11].

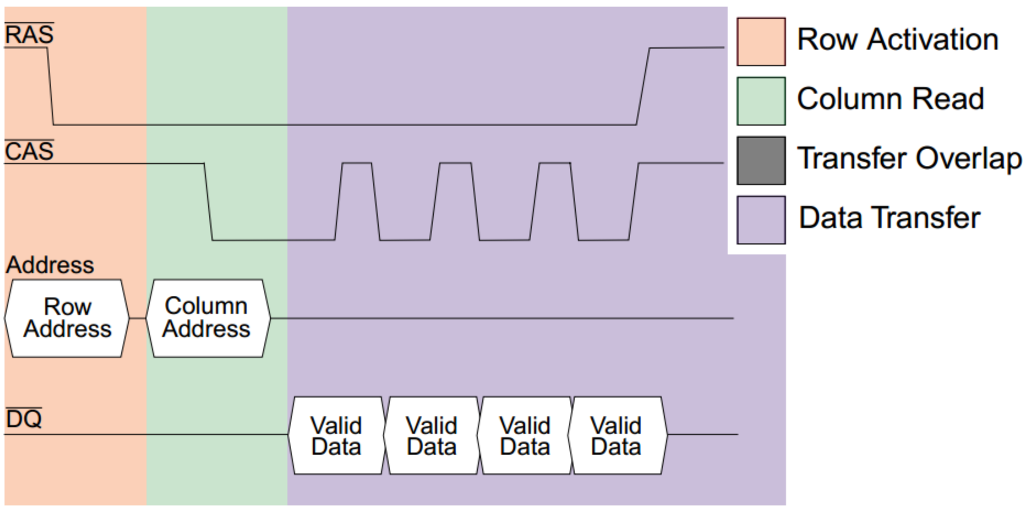


Figure 43:BEDO DRAM Read Process

This revolutionary interface reduced the minimum read time with 30% for practically almost no extra hardware (a counter per array). Then in 1990 in International Solid State Circuits Conference, IBM proposed toggle mode DRAM as an option for future DRAMs where Toggle mode DRAMs drives data to and from a DRAM on both edges of a high-speed data strobe rather than transferring data on a single edge of the strobe hence doubling data rate at the same bus frequency and the name "toggling" comes from the fact that interface stays the same however the DRAM toggles between two time interleaved output buffers each of which runs with half the rate. IBM toggling mode DRAM inspired changes and for lower latencies the DRAM interface is reverted back to synchronous to regularize requests arrival. The timing diagram of the synchronous DRAM is shown in figure below First RAS signal is de-asserted and raw address is driven on address bus then CAS signal is de-asserted and the column address is driven on the address bus hence four valid data packets are driven of DQ bus at positive edge of the clock signal.

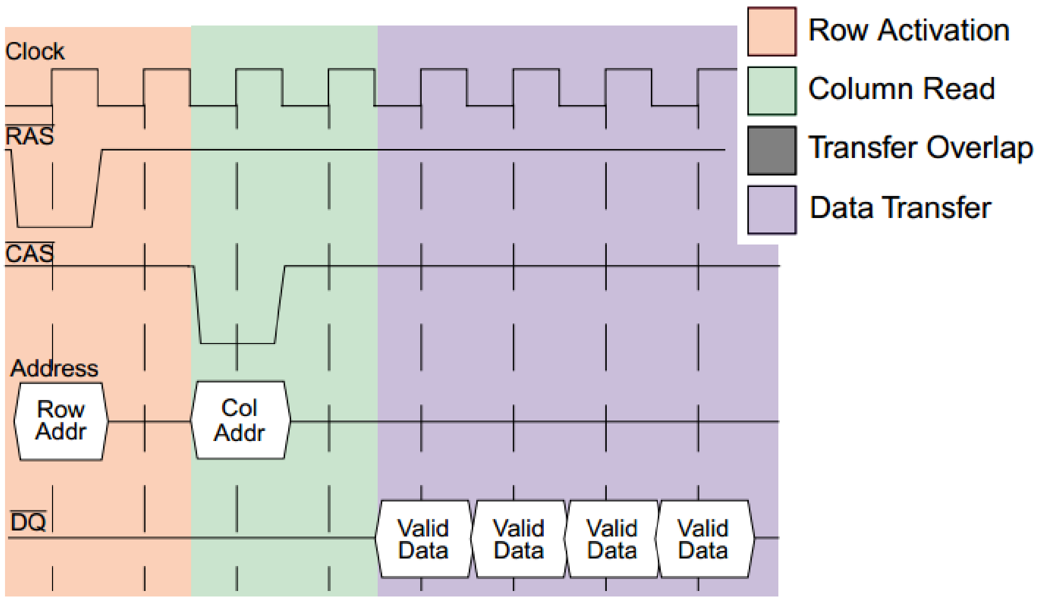


Figure 44:SDRAM Read Process

IBM toggling principle inspired DDR SDRAM where data is driven on both edges of the clock the read process of DDR SDRAM is shown in figure below+5. First RAS signal is de-asserted and raw address is driven on address bus then CAS signal is de-asserted and the column address is driven on the address bus hence four valid data packets are driven of DQ bus at each edge of the clock signal. DQS signal was introduced for synchronization between MC and DRAM.

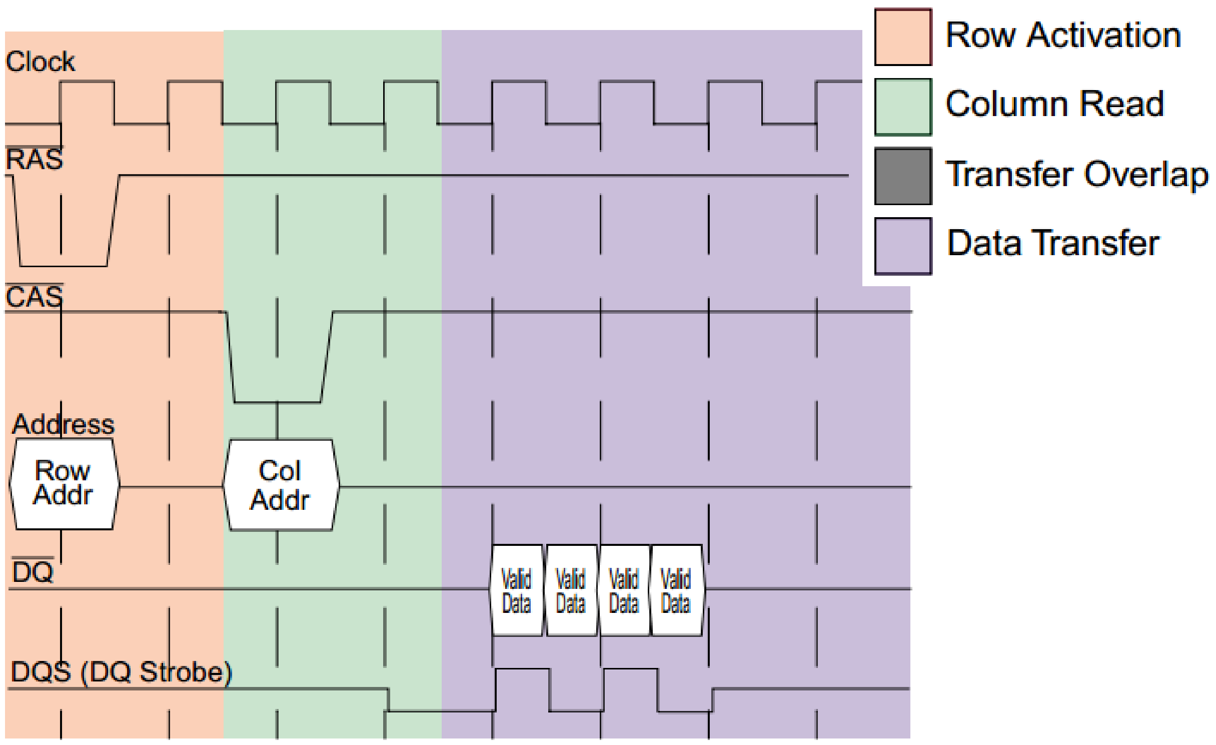


Figure 45:DDR SDRAM Read Process.

Figure below summarizes modifications targeting throughput of DRAM

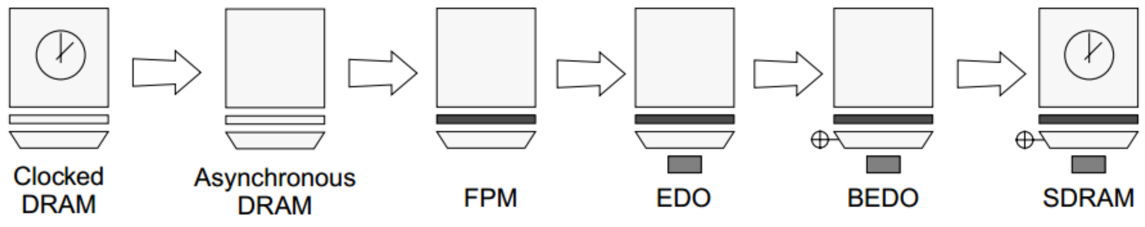


Figure 46:history of DRAM.

3.2.3 *Modifications targeting throughput*

There has been attempts to build a DRAM with lower latencies either by having faster hardware or via caching (adding SRAM to DRAM) from these trials we mention the following example to demonstrate the general idea. Enhanced SDRAM (ESDRAM) is similar to EDO DRAM discussed above, Enhanced SDRAM adds an SRAM latch (6T cell) to the DRAM core, but whereas EDO adds the latch after the column mux, ESDRAM adds it before the column mux. Therefore, the latch is as wide as a DRAM page instead of the desired word length as in EDO which adds expensive overhead in area (6T per column) However this allows earlier overlap of activity that is shown above in figure 42.

3.2.3 *Evolution of DDRx*

The state of the Art DDR SDRAM (DDR5) can achieve up to 6400MT/s with a density of up to 64Gb which is a huge enhancement in speed and capacity in comparison with its predecessor DDR4 as mentioned above. Table below summarizes SDRAMS performance evolution.

Table 14:SDRAMS performance evolution.

	Data rate (MT/s)	Transfer rate (GB/s)	Voltage
DDR1	266-400	2.1-3.2	2.5
DDR2	533-800	4.2-6.4	1.8
DDR3	1066-1600	8.5-14.9	1.35
DDR4	2133-3200	17-23.2	1.2
DDR5	3200-6400	23.2-51.2	1.1

3.3 Required Tools and Technical approaches

In this project we aim to build the Digital circuitry of DDR5 PHY IP hence we require the following tools to build a state-of-the-art system in partnership with Si-Vision. RTL simulations tool such as Synopsys VCS alongside with open-source editor such as Visual Studio Code, FPGA tool forprototyping and finally lint checking tool such as Synopsys Spyglass Lint. The Techniques we will use include:

- In Architecture Optimization
- Parallelism
- Pipelining
- Block level optimization techniques
- RTL modeling (Not HLS)
- One hot encoding for FSMs
- Deep pipelining when applicable

The technical approach we intend to use in the design of the project include:

- Enhancing our background in digital design via Silicon Vision proposed training material
- Reading DFI & JEDEC standards independently for specification extraction
- and architectural description
- Reviewing the specifications and with the responsible system engineer at Silicon Vision.
- Developing and verifying RTL for Components of the design
- System integration
- System verification
- Hardware emulation

4- Project Design

4.1 Project purpose and constraints

The main goal of this project is designing a fully functional, optimized, and synthesizable physical layer (PHY) IP for DDR5 with behavioral simulations on the top level and individual modules level for testing followed by hardware testing using FPGA prototype. The IP is constrained with the communicating standards (JEDEC & DFI) specifications and requirements.

4.2 Project technical specifications

The specifications required for DDR5 PHY to operate under the standard conditions are:

- DFI compliant
- JEDEC JESD79-5A compliant
- Supports data rates up to 6400 Mbps
- Support multiple frequency ratios (1:1, 1:2, 1:4)

4.3 Design alternatives and justification

Alternatives are hard and soft IPs from IP companies such as Synopsys, Cadence, Micron, and Faraday. Those IPs have hundreds of engineers working on their design, verification, and implementation; hence, their prices are very high and are dependent of the customer interests.

4.4 Description of the selected design

The design proposed is the implementing only the digital part in the DDR5 physical layer. The hierarchy of the physical layer is shown in figure below.

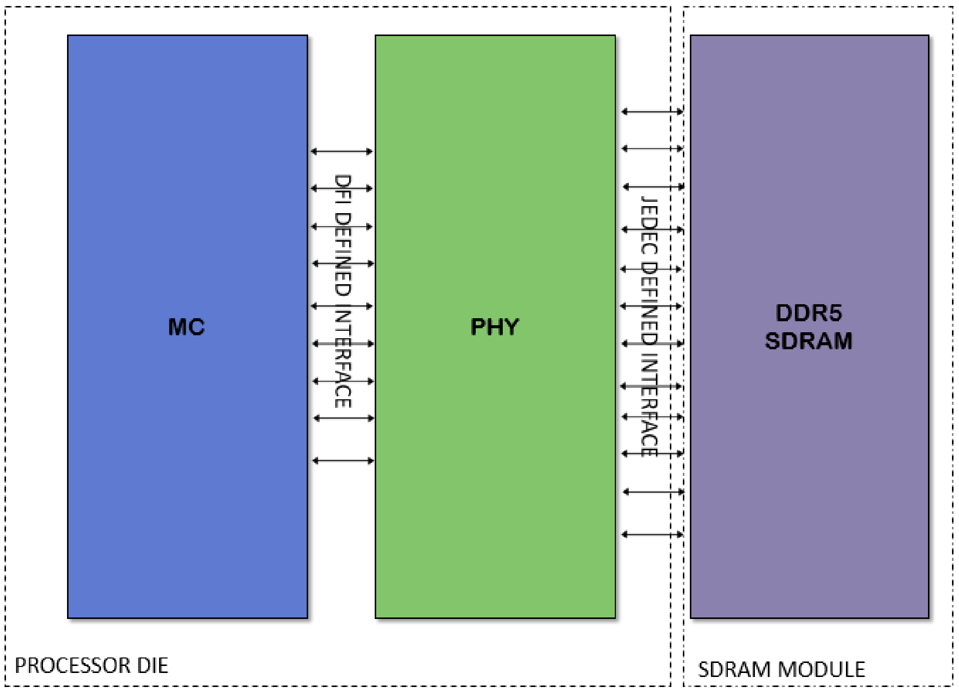


Figure 47:Memory Systems Architecture.

A detailed interface of the digital part of our design is shown in figure below.

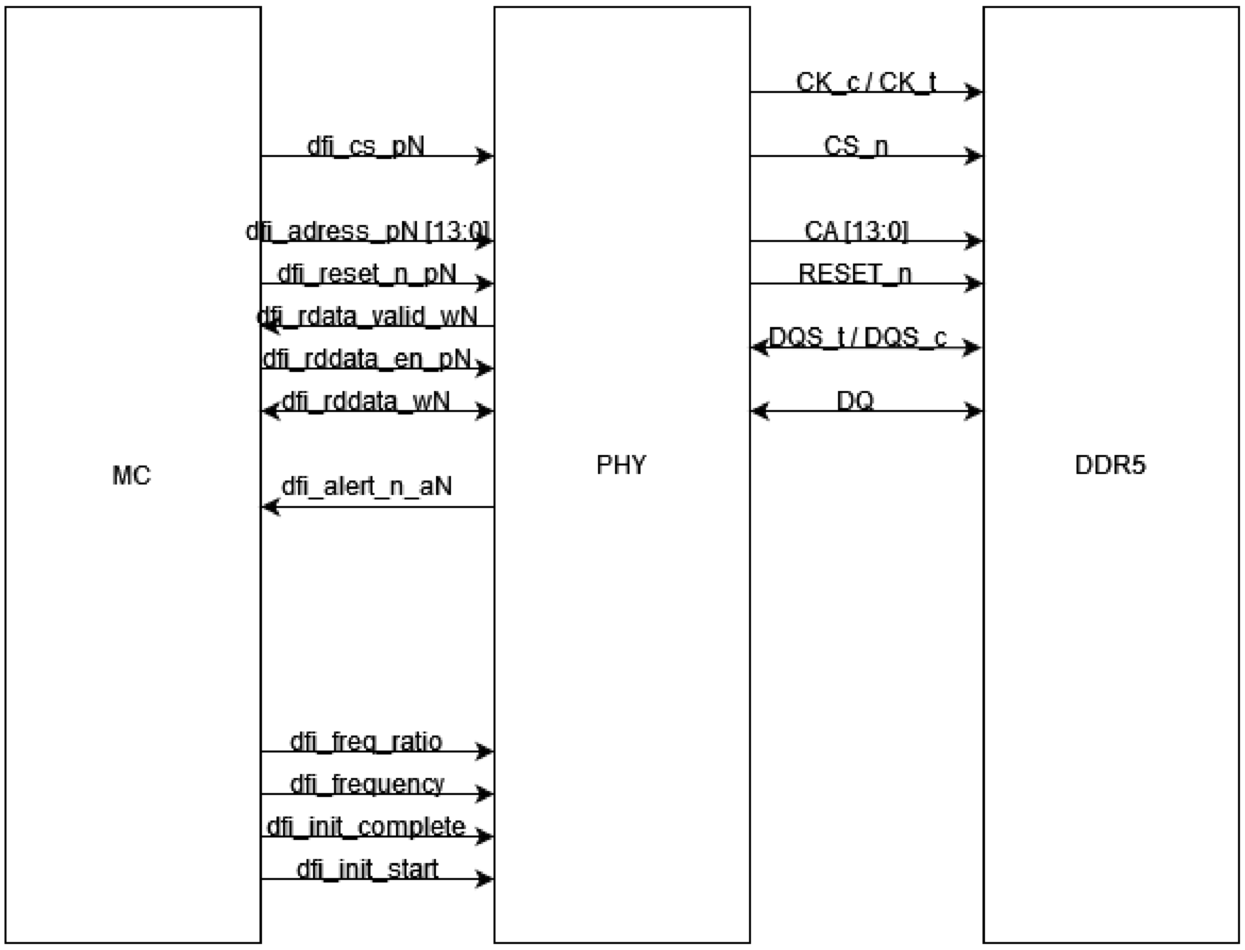


Figure 48:Design interfaces as extracted from the standard

The digital part submodules will contain:

- Data manager module
- Read manager module
- Timing Register File
- Frequency Ratio manager
- Command Address (CA) manager
- Mode registers
- CRC validation block

4.5 Block diagram and functions of the subsystems

4.5.1 Architectural description

The block diagram is shown in figure below. The functionality of the submodules is shown in table 15.

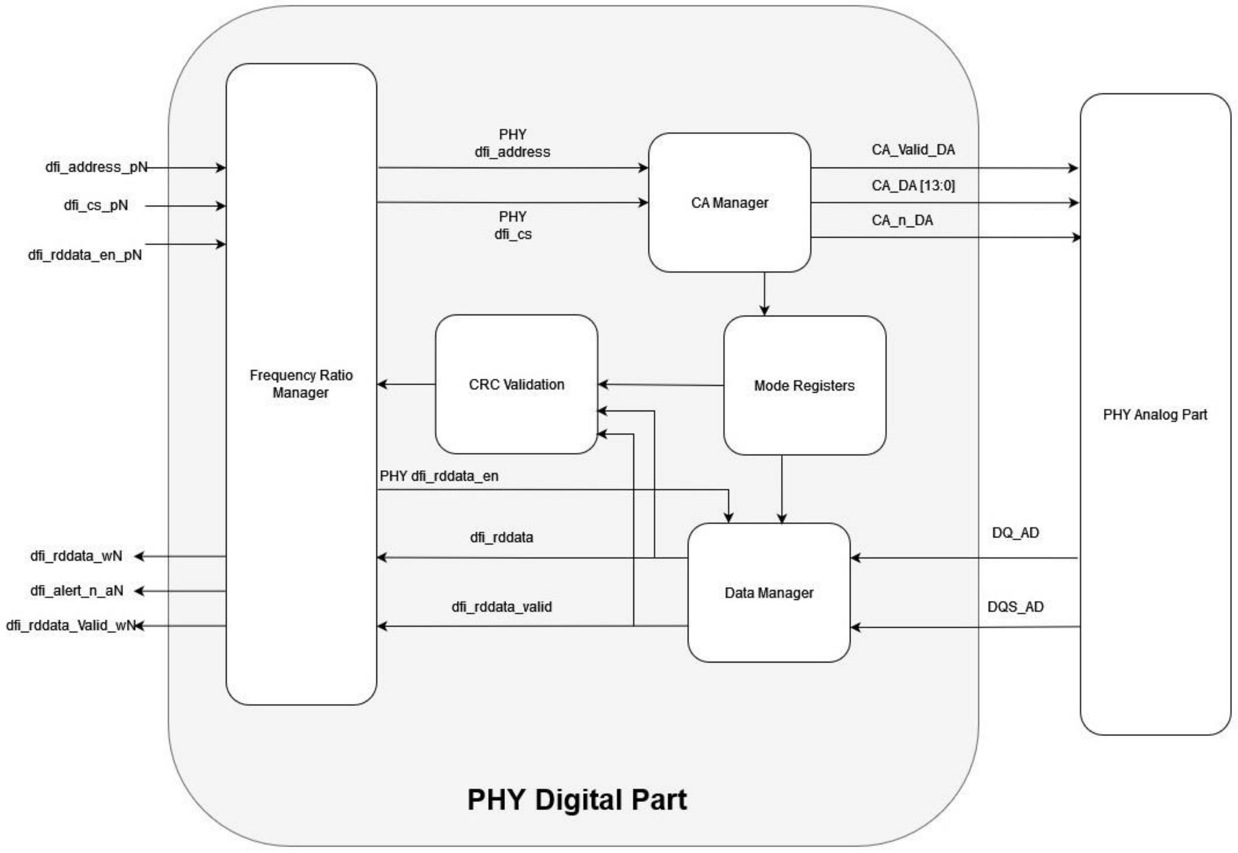


Figure 49:Block diagram of the proposed design.

Table 15:Submodules inputs, outputs, functionality, and parameters.

Block Name	Inputs	Outputs	Functionality	Parameters
Frequency Ratio Manager	dfi_address_pN dfi_cs_pN dfi_rddata_en_pN dfi_alert_n dfi_rddata_valid dfi_rddata	dfi_alert_n_aN dfi_rddata_valid_wN dfi_rddata_wN PHY dfi_rddata_en PHY dfi_address PHY dfi_cs	1- Serializes inputs coming from MC to PHY based on the dfi_freq_ratio parameter and on the DFI PHY clock. 2- Deserializes inputs coming from PHY to MC based on the dfi_freq_ratio parameter and on the DFI clock.	dfi_freq_ratio[2:0]
CA Manager	PHY dfi_address PHY dfi_cs	CA_Valid_DA CA_DA [13:0] CS_n_DA	1- Distinguish between a 1Cycle and a 2Cycle command. 2- Assert a valid signal (CA_Valid_DA) that	**Tctrl_delay**

			follows the validity of CA_DA. 3- During MRW operation, it saves values such as BL, read crc enable, and preamble settings. 4-adds delays when needed 5-discard BL of MR0 when needed	
Read Manager	PHY dfi_rddata_en DQ DQS	pre_rddata_valid pre_rddata	1- Detect the preamble. 2- Detect the end of data based on the BL. 3- Sample the data based on DQS. 4-Assert a valid signal (dfi_rddata_valid) along with the dfi_rddata.	Read CRC enable phycrc_mode BL Preamble settings
CRC validation	DQ pre_rddata_valid	dfi_alert_n	1- Starts the operation when pre_rddata_valid is asserted. 2- Validates DQ 3- Activates dfi_alert _n when an error occurs 4- Assert a valid signal (pre_rddata_valid) along with the pre_rddata.	Mode Registers: PHY_CRC_mode read_CRC_enable

The associated signals characteristics are shown in table 16 below.

Table 16: Signals widths

SIGNAL	WIDTH
dfi_freq_ratio	3 bits
dfi_frequency	5 bits
dfi_rddata	2* DRAM Width
dfi_rddata_valid	1
dfi_rddata_en	1
dfi_alert_n	1
dfi_reset_n_pn	1
dfi_address_pN	14
dfi_cs_pN	(physical rank no.)
dfi_init_complete	1
dfi_init_start	1
PHY dfi_cs	(physical rank no.)
PHY dfi_rddata_en	1
PHY dfi_address	14
ReadFlag	1
DQ_AD	DRAM Width
DQS_AD	1
pre_rddata	2*DRAM Width

pre_rddata_valid	1
CA_Valid_DA	1
CS_n_DA	(physical rank no.)
CA_DA	14
dfi_rddata_valid_wN	1
dfi_alert_n_aN	1
dfi_rddata_wN	DRAM Width

To illustrate the operation of the proposed block diagram. Figures below developed with the block diagram to show the waveforms for Read, Read with CRC, Back-to-Back Reads, back-to-back reads with CRC, and 1:2 frequency ratio operations respectively.

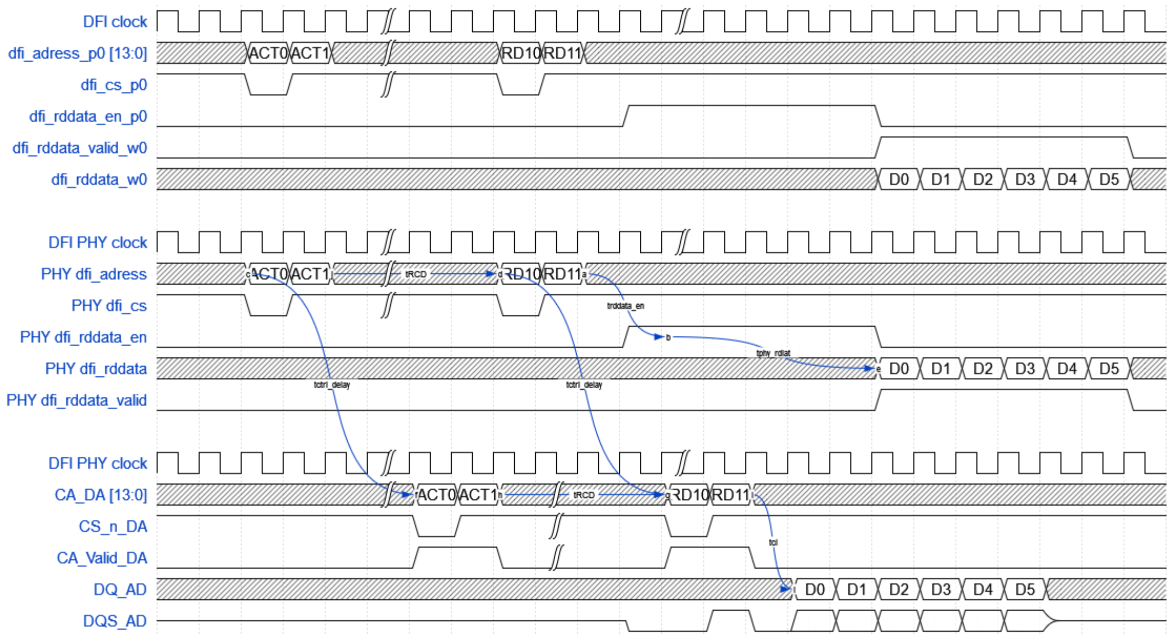


Figure 50:Standard Read operation Developed Waveforms.

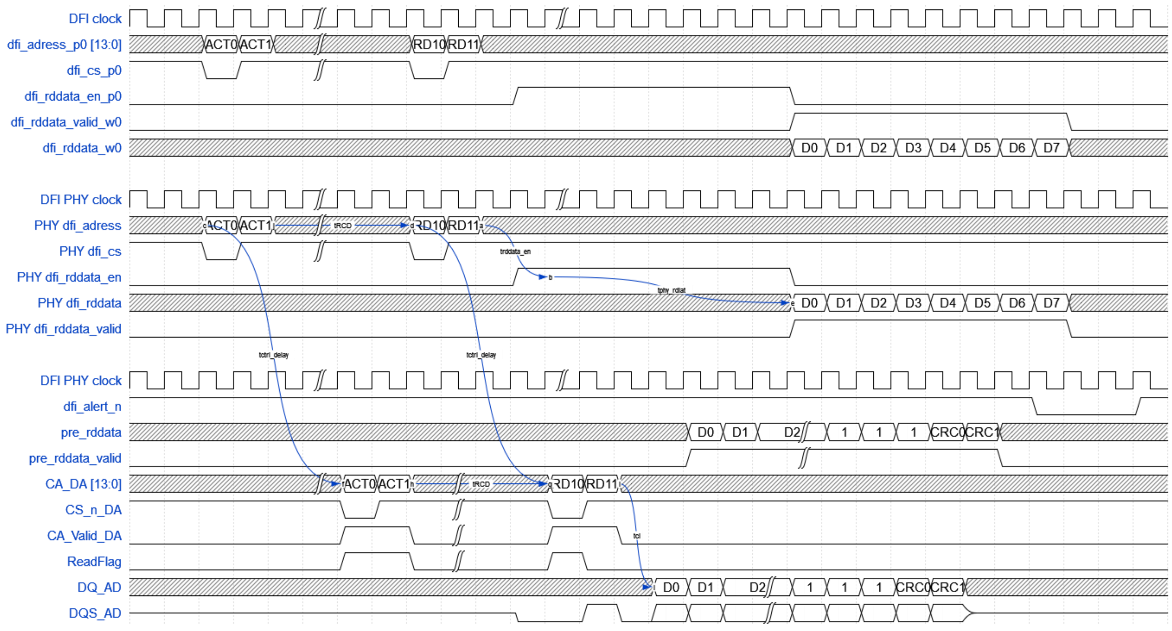


Figure 51:Read with CRC operation Developed Waveforms

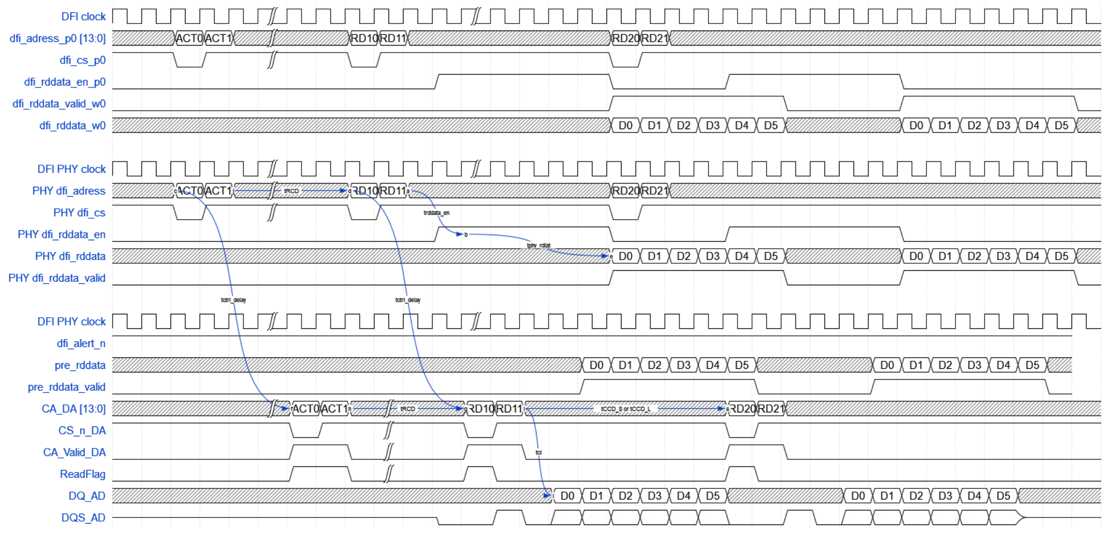


Figure 52:Back-to-Back Reads operation Developed Waveforms.

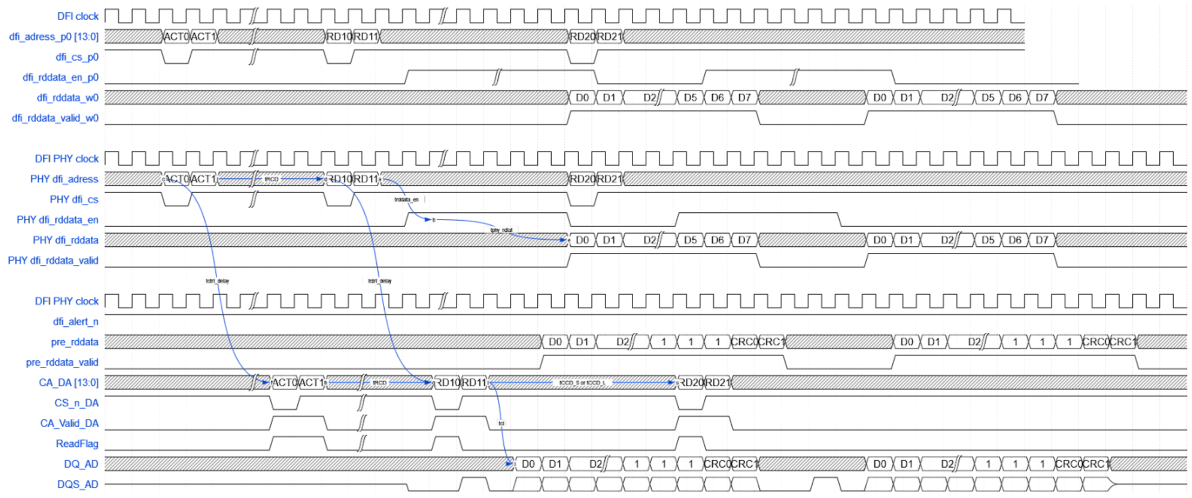


Figure 53:Back-to-Back Reads with CRC operation Developed Waveforms

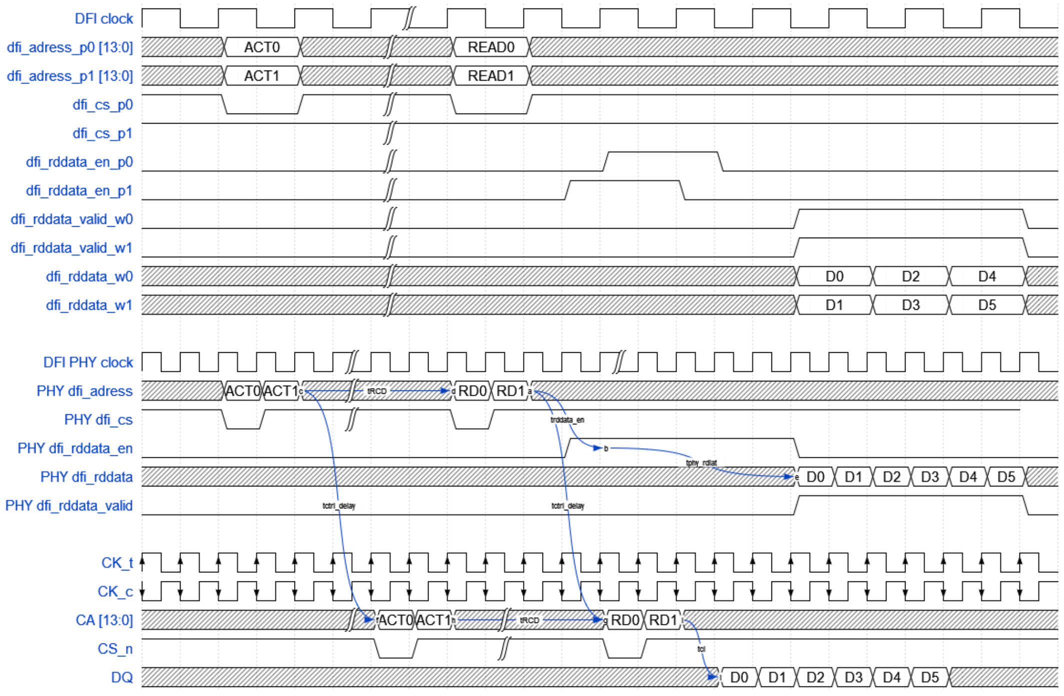


Figure 54:1:2 Frequency ratio Read operation Developed Waveforms.

4.5.2 CA Manger

Now we will discuss each block separately to furtherly discuss its specific implementation. Command address manger is the block that handles command bus, in DDR5 no classical RAS and CAS signals as in all previous generation but instead a 14-bit command address bus is directly mapped to DRAM command bus from MC. In this PHY specific implementation the following functionalities are required from the block that handles command address bus

1- Distinguishing between a 1Cycle and a 2Cycle command.
2- Asserting a valid signal (CA_Valid_DA) that follows the validity of CA_DA.
3- During MRW operation, it saves values such as BL, read crc enable, postamble and preamble settings.
4- Discard BL of MR0 when needed

A block diagram of CA manger is shown in figure 55.

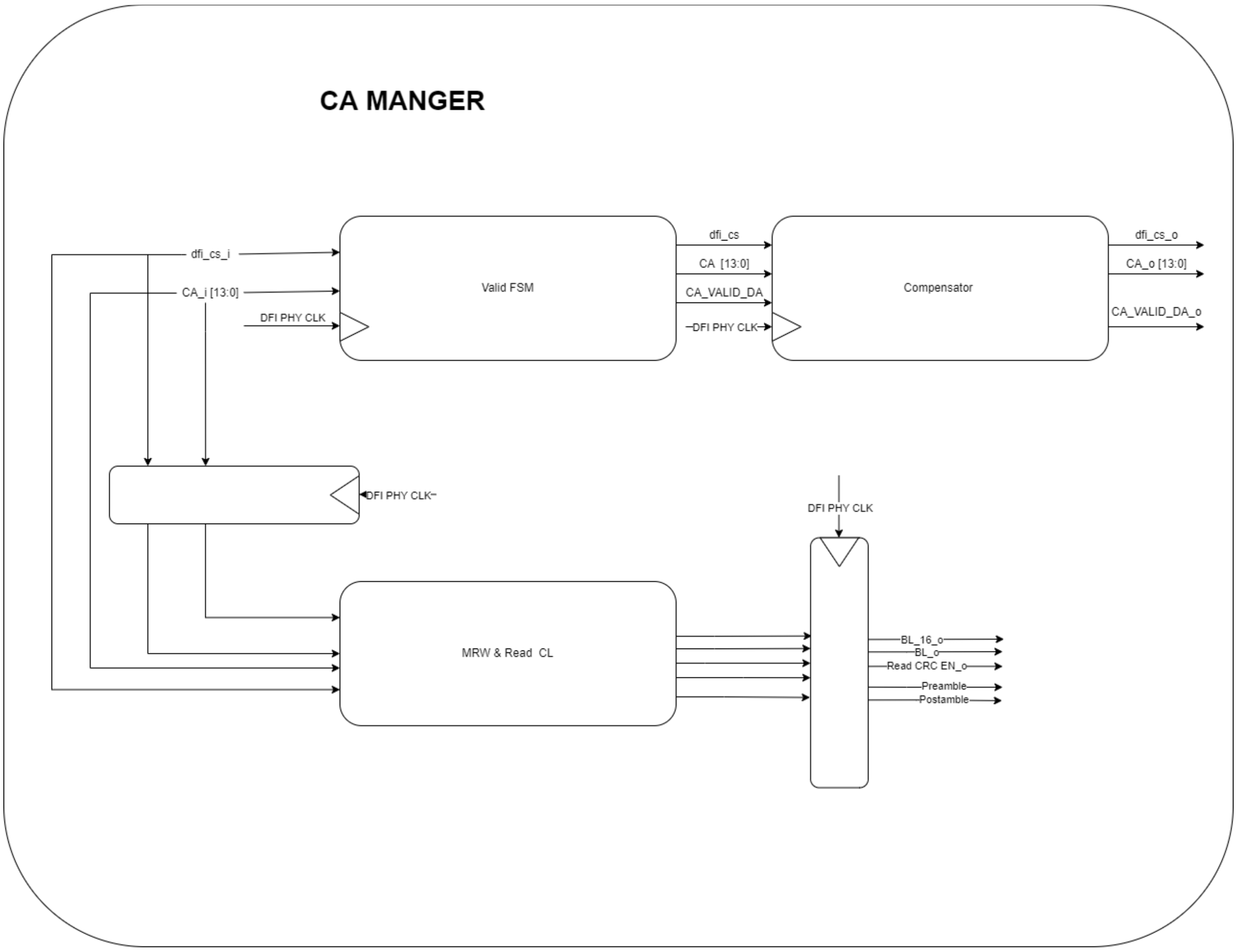


Figure 55. CA Manger Block Diagram

The valid FSM serves functionality 1 and 2 while MRW & Read CL serves functionality 3 and 4. Figure 56 shows the state diagram of CA valid FSM.

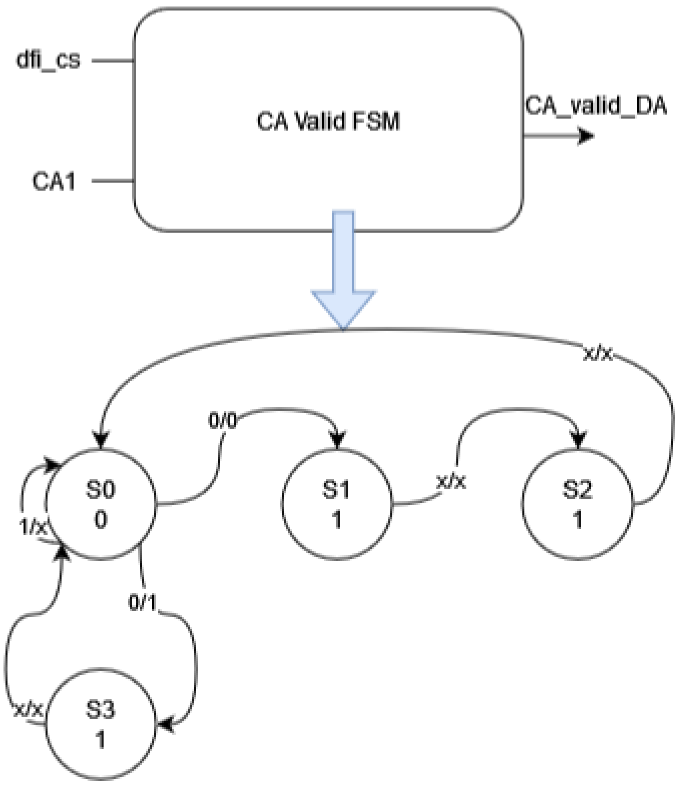


Figure 56. CA valid FSM state diagram

S0 is the state where the CA is idle. S1 is the state when the first cycle of a two-cycle command is in progress. S2 is the state when the second cycle of a two-cycle command is in progress S3 is the state when the first cycle of a one cycle command is in progress. The outputs are driven accordingly. The CL of MRW and Read is summarized in the table below

Table 17. MRW & Read CL

dfi_address_0			**dfi_address_1**							
CA[0:4]	CA [5:12]	cs	CA0	CA1	CA2	CA3	CA4	CA5	CA6	CA7
1_0100	0011_0010	1	CRC							
	0000_0000	1	BL							
	0000_0100	1	Preamble						Post	
	0010_1000	1	DQS offset							
x_xxxx	xxxx_xxxx	0	Don’t overwrite							

Where dfi_address_0 is command address during the first cycle of a two-cycle command and dfi_address_1 is command address during the second cycle of a two-cycle command. A high-level model for the block was implemented using MATLAB and it can be found in appendices figures 57 58 59 60 shows the desired functionality in MATLAB.

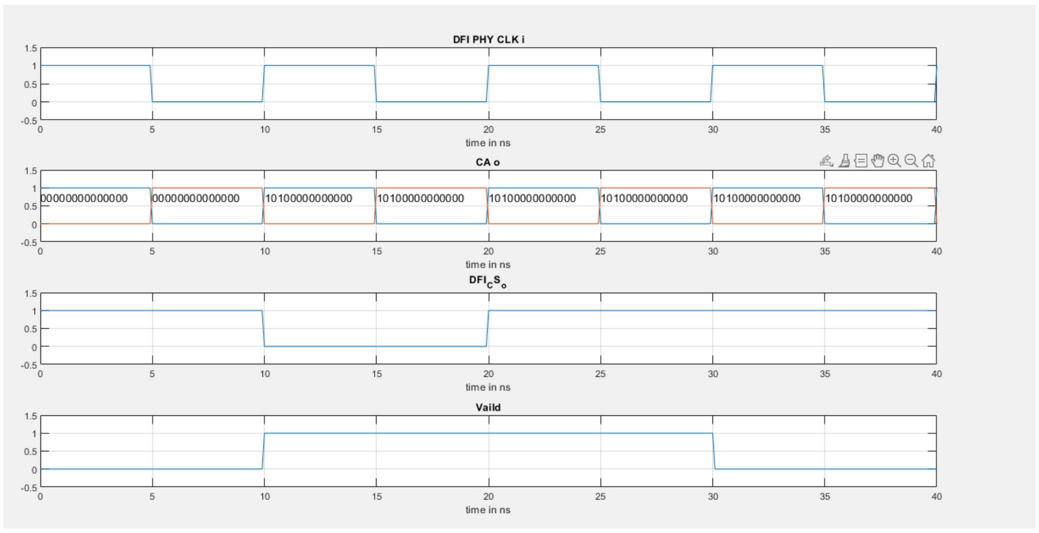


Figure 57. Assertion of valid signal for two cycle command.

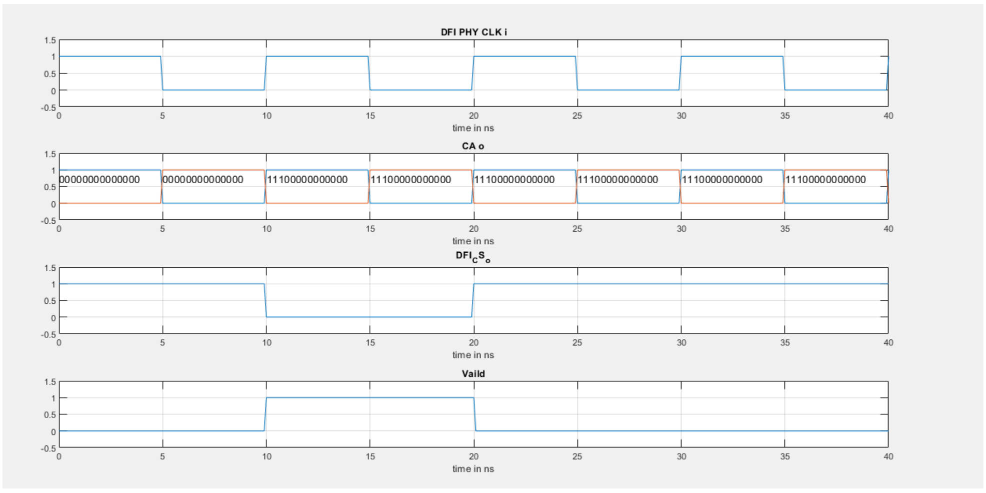


Figure 58. Assertion of valid signal for one cycle command.

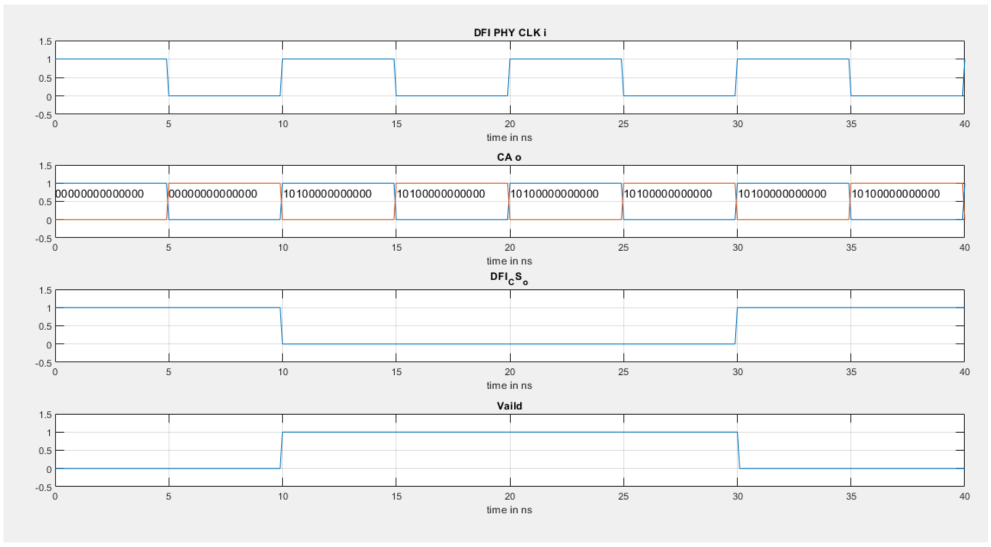


Figure 59. Assertion of valid signal for canceled two cycle command

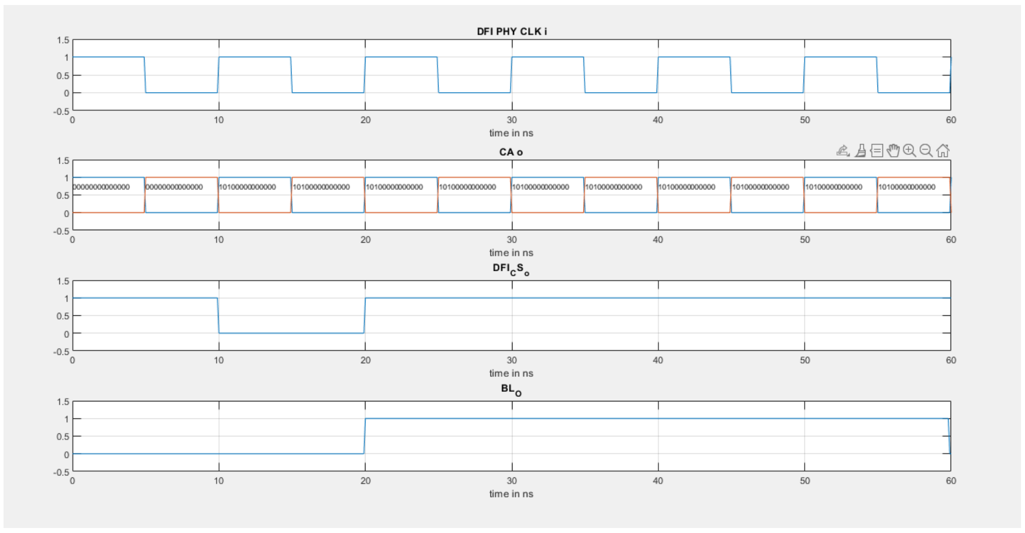


Figure 60. MRW that changes BL (only bit one shown)

After ensuring that the design was valid it was implemented as an RTL using System Verilog the code was hence synthesized using Synopsys Design Compiler with target PDK nangate 45nm and it was also synthesized on Quartus the following simulation highlights the main functionality of the block.

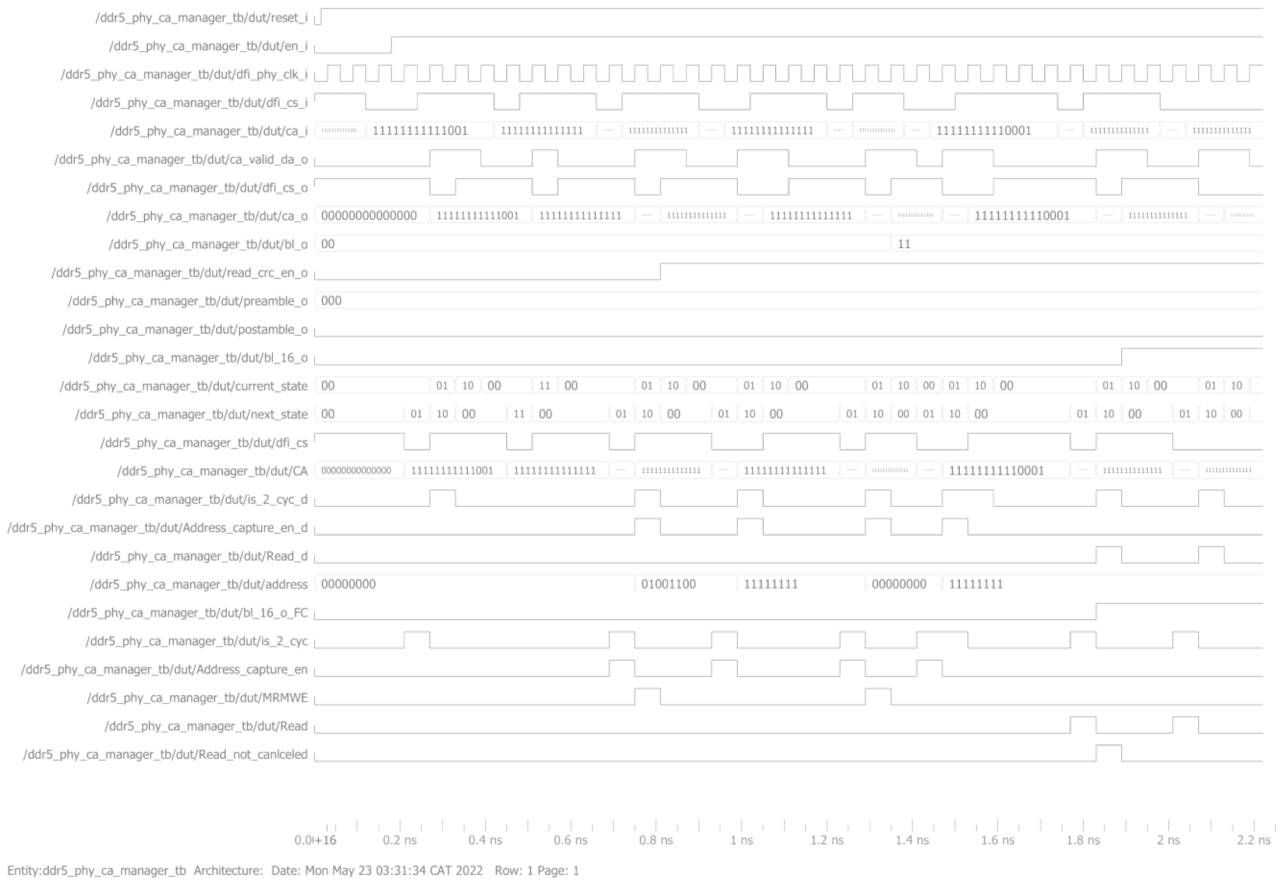


Figure 61. CA manger RTL simulation

As shown a series of MRW commands was sent targeting value changes at registers of interest some of them were canceled and hence did not result in a value change and some was valid and hence resulted in a value change observe read_crc_en at 0.8ns for instance or bl_o at 1.4 ns. Finally, a read operation was driven on CA bus and the BL was set to default of 16 hence the signal bl_16_o was asserted at 1.8 ns. Note how ca_valid_da_o signal is asserted differently for 1 and 2 cycle commands.

4.5.3 Data Manger

The functional requirements of the data manger include: detecting the preamble patterns, detecting the end of data based on the BL, sampling the data based on DQS**,** and asserting a valid signal (dfi_rddata_valid) along with the dfi_rddata. The block generates a valid signal along with the data, and this valid signal is sent directly to the MC through the frequency ratio manager. Also, the block saves the sends the settings of the CRC validation to the CRC validation block with each read data. The block's interface is shown in table 18. The block diagram of the data manager is shown in figure 62.

Table 18. Data manager interface.

Inputs		**Output**	
Signal	**Size (bits)**	**Signal**	**Size (bits)**
Preamble Settings	3	Data valid signal	
Burst Length	2	Data	
Post amble settings	1	2*device_width	
Read data enable	1		
Data Strobe	1		
Data	2*device_width		
Clock	1		
Reset	1		
Enable	1		
Read CRC Enable	1		
Phy crc mode	1		

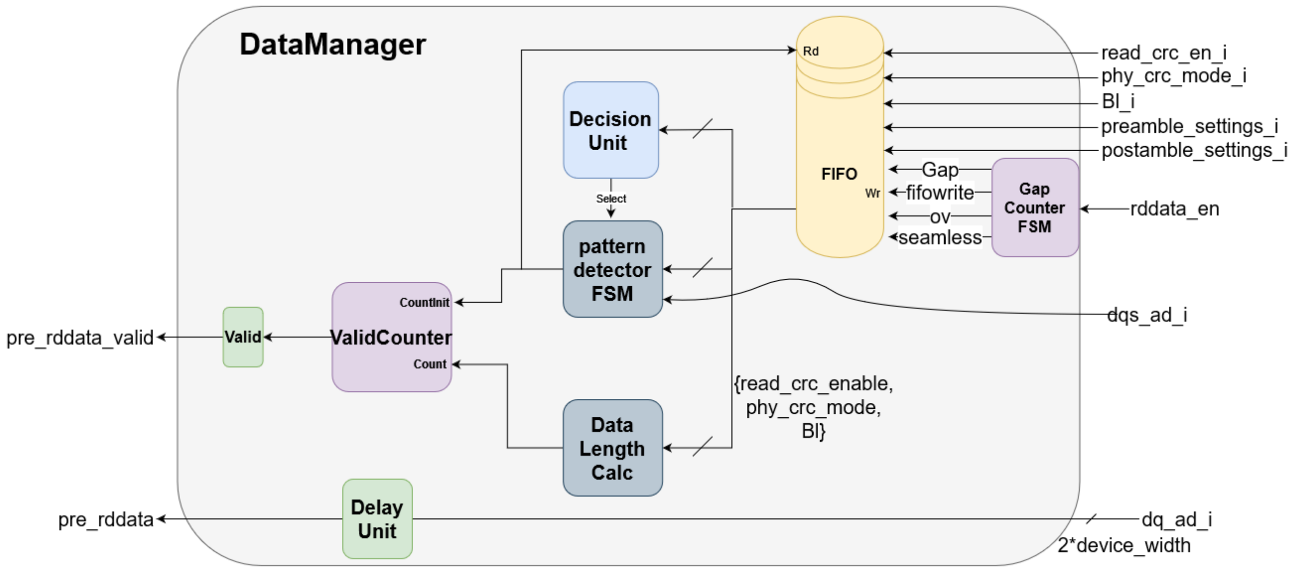


Figure 62. Data manager block diagram.

The Gap Counter contains both an FSM and a counter. The FSM is used to control the signals such as the gap counter reset signal, a gap valid signal and the FIFO write signal. The Counter calculates the gap. The operation is depicted in the figure 64.

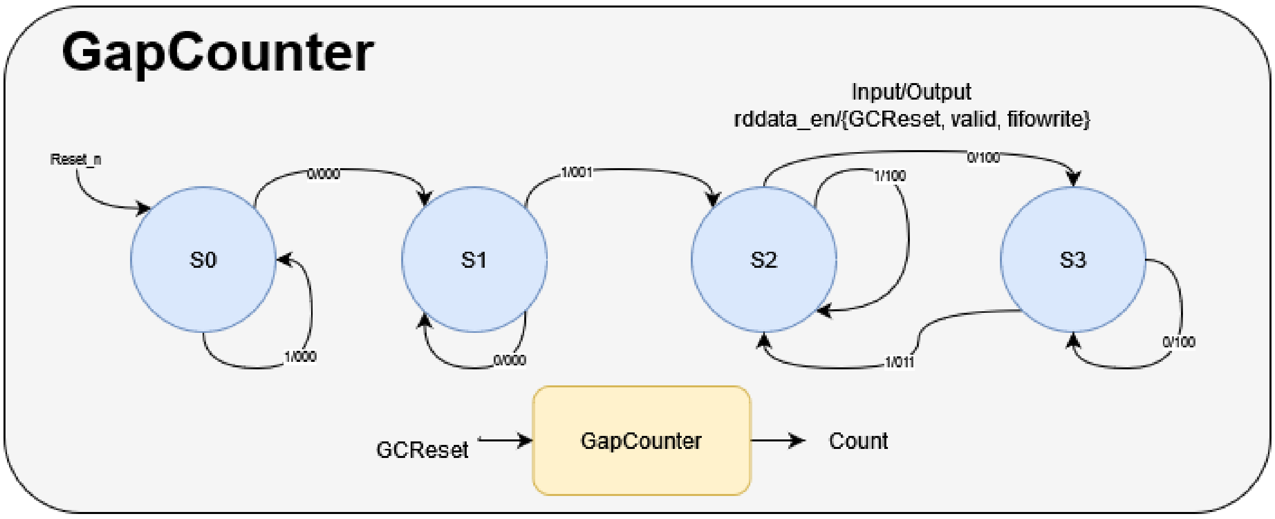


Figure 63. Gap counter FSM.

The Gap Counter calculates the gap between the read commands by calculating the gap between the positive edges of the read data enable signal. Also, it generates a FIFO write command with each posedge of the read data enable meaning that the settings are written into the FIFO with each read command. An OV signal is used to indicate an over flow of the counter. When the gap counter FSM detects a (0 1) pattern in the rddata_en signal it de-asserts the reset signal so the counter starts counting the gap until a (0 1) pattern is detected again so the count becomes valid and it resets the counter. Also, the FSM generates a valid signal along with the gap. The FIFO counter and corresponding pointers are shown in figure 64.

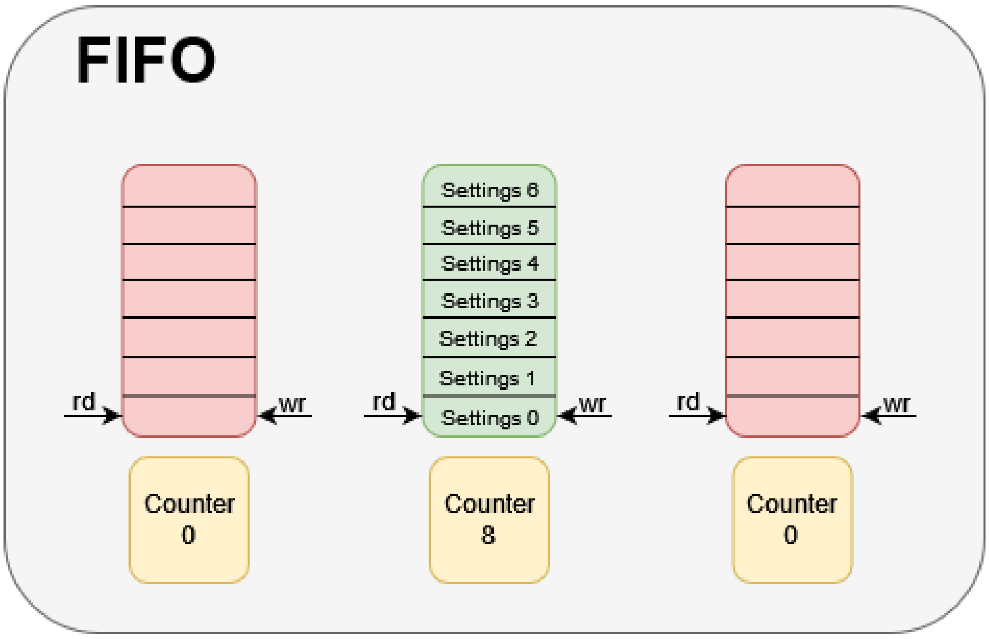


Figure 64. FIFO counter and pointers.

A fifo is needed in the design so as to reserve the settings of different reads. The read Command Latency (CL) of a DDR5 SDRAM has a min of 22 cycles and a max of 66 cycles so multiple read commands can be issued with different settings before any data is present. Tccd_min = 8 cycles, TCL_max = 66 cycles. This means that 9 read commands with different settings can come before the first read data is out. The read signal is issued once the previous pattern is detected and also an FSM is needed so that it can be issued with the first read operation. The CountCalc block generates the number of cycles that the valid needs to be asserted based on the BL, and CRC settings. The specific table for the block is described below. Also, the block gives a valid signal for two read data if a seamless condition is applied. Table 19 show the different cases for the gap counter.

Table 19. Gap counter cases.

readCRCen	phyCRCmode	Seamless	BL	Count
0	0	0	8	4
0	1	0	8	4
1	1	0	8	4
1	0	0	8	9
x	x	1	16	16
0	0	0	16	8
0	1	x	16	8
1	1	x	16	8
1	0	x	16	9

The ValidCounter block counts the number of cycles up to the CountCalc issued value to generate the valid signal once the pattern is detected. This module is composed of an intermediate register that takes the value of the gap when the valid is high. A mux that selects either the incoming data is a pre or interamble then forwards this information to the pattern detector block for the detection phase.

The pattern detector block is respobsible for detecting multiple different patterns according to the configuration of the PHY. The block should assert the signal pattern_detected for 1 clock cycle after detecting a pattern. The block diagram of the pattern detector is shown in figure 65. The signals are clk_i which is the input clock signal, reset_n_i which is the input reset signal that is active low, en_i which is an enable signal for the internal FSM. When low, the module state is not changing. The DQS_i signal is the input serial data that will contain the pattern or not. The signals pre_amble_sett_i, post_amble_sett_i, gap_i, pre_or_inter are configuration signals that determine which pattern the module is going to detect as illustrated in table 20. The pattern_detected signal is an output signal that gets asserted when the configured pattern is detected. The block is diveded into 2 main blocks, setting generator and generic fsm, the setting generator should take the the configuration from the PHY, specifically from the CA manager, then provide the FSM with the required information about the pattern that will be detected. Then, the generic fsm accordingly detects that pattern and asserts the “pattern detected” signal. The brute force approach to implement the generic fsm is to have multiple FSMs, one for each pattern in the standard, and the generic fsm choses which one to be active. This approach’s FPGA results are shown in figure 66. Alternatively, the generic FSM was implemented as one FSM which is reconfigurable for each pattern. This FSM will need a signal that provide the pattern itself, another signal that provide the last state of the pattern, and lastly an array that determine which state the FSM will return to when a wrong bit is detected. The generic fsm diagram is shown in figure 67 and the setting/configuration generator is cases are shown in table 20. Also, the results of the FPGA implementation after using this approach is shown in figure 68. As shown in the results, the generic design use much less flipflops and hence is better in terms of area and power. Lastly a simulation of one case is shown in figure 69. The output state is 7 instead of 5 as 7 is the gray code number for 5.

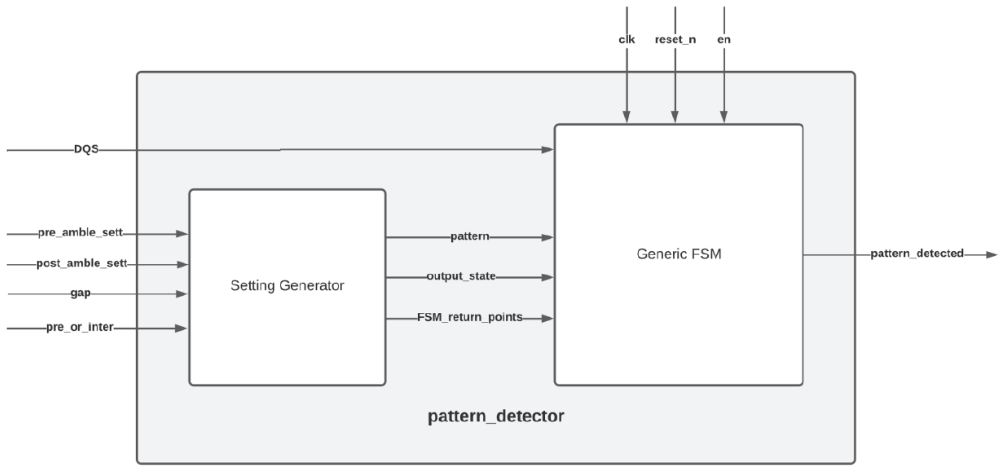
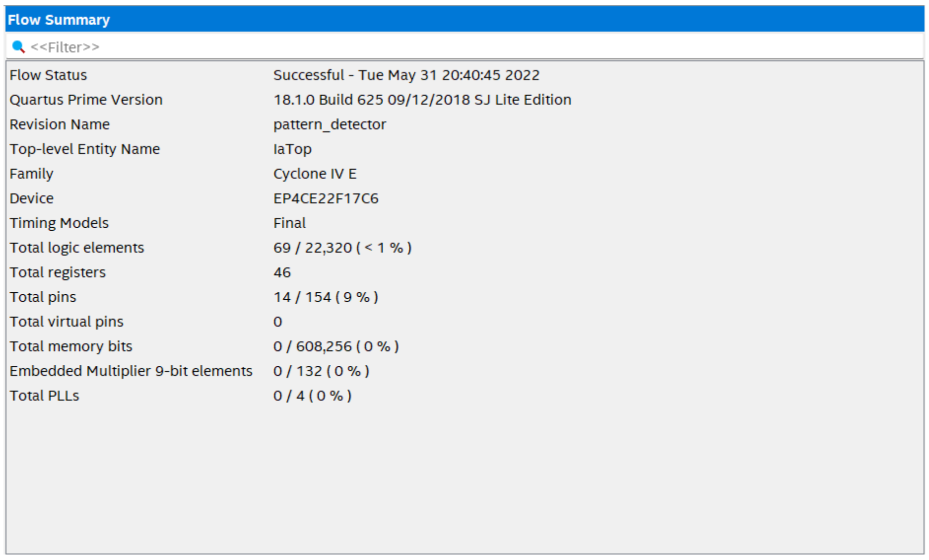


Figure 65. Pattern detector block diagram.



Flow Summary

<<Filter>>

Flow Status	Successful - Tue May 31 20:40:45 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	pattern_detector
Top-level Entity Name	laTop
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	69 / 22,320 (< 1 %)
Total registers	46
Total pins	14 / 154 (9 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 66. FPGA implementation results of the initial design.

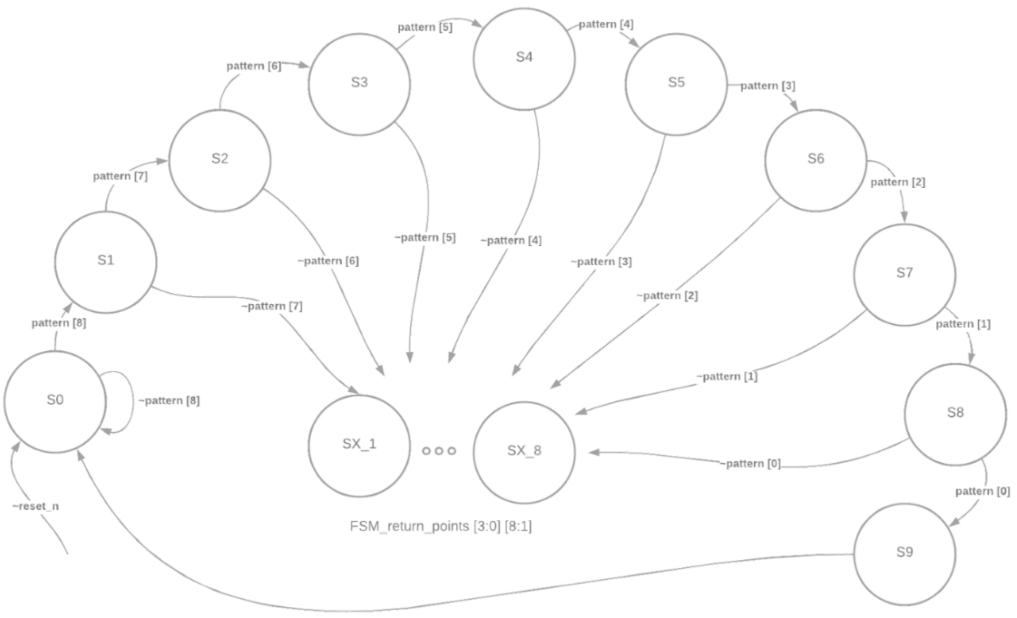
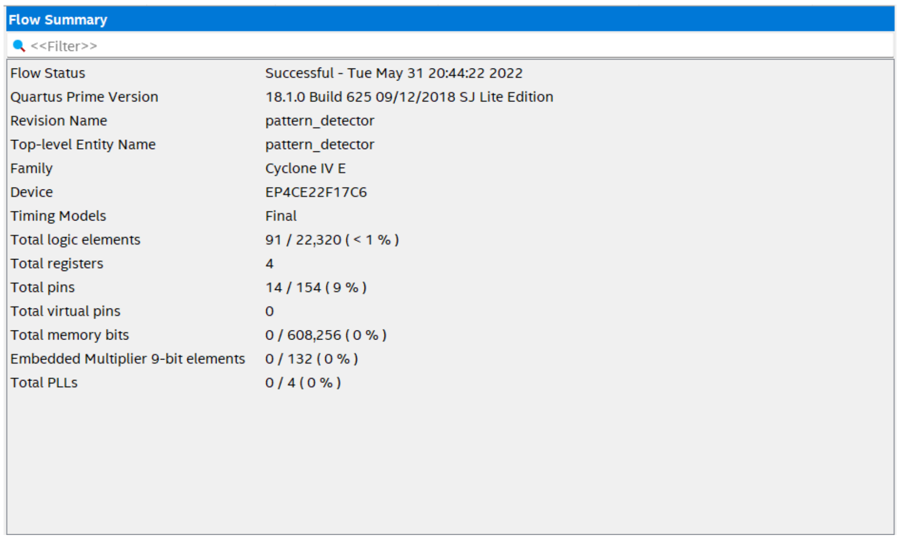


Figure 67. Generic FSM state diagram.

Table 20. Cases of the setting/configuration generator for the pattern detector.

INPUTS				OUTPUTS		
pre_or_inter	pre_amble_sett_i [2:0]	post_amble_sett_i	gap_i [4:0]	pattern_o [8:0]	output_state_o [3:0]	FSM_return_points_o [3:0] [8:1]
pre amble						
1	000	x	xxxx	10_0000000	S2	S0, S1, S0, S0, S0, S0, S0, S0, S0
	001			0010_00000	S4	S0, S0, S2, S0, S0, S0, S0, S0, S0
	010			1110_00000	S4	S0, S0, S0, S3, S0, S0, S0, S0, S0
	011			000010_000	S6	S0, S0, S0, S0, S4, S0, S0, S0, S0
	100			00001010_0	S8	S0, S0, S0, S0, S4, S0, S2, S0, S0
inter amble						
0	xxx	0	1001	10_0000000	S2	S0, S1, S0, S0, S0, S0, S0, S0, S0
	011	0	1010	0010_00000	S4	S0, S0, S2, S0, S0, S0, S0, S0, S0
	100	0	1010	01010_0000	S5	S0, S1, S1, S0, S0, S0, S0, S0, S0
	xxx	1	1010	01010_0000	S5	S0, S1, S0, S0, S0, S0, S0, S0, S0
	011	1	1011	0100010_00	S7	S0, S1, S0, S2, S2, S1, S0, S0, S0
	100	0	1011	0001010_00	S7	S0, S0, S0, S3, S0, S2, S0, S0, S0
	100	1	1011	0101010_00	S7	S0, S1, S0, S1, S0, S1, S0, S0, S0
	100	1	1100	010001010_	S9	S0, S1, S0, S2, S2, S1, S0, S1, S0



Flow Summary

<<Filter>>

Flow Status	Successful - Tue May 31 20:44:22 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	pattern_detector
Top-level Entity Name	pattern_detector
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	91 / 22,320 (< 1 %)
Total registers	4
Total pins	14 / 154 (9 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 68. FPGA implementation results of the final design.

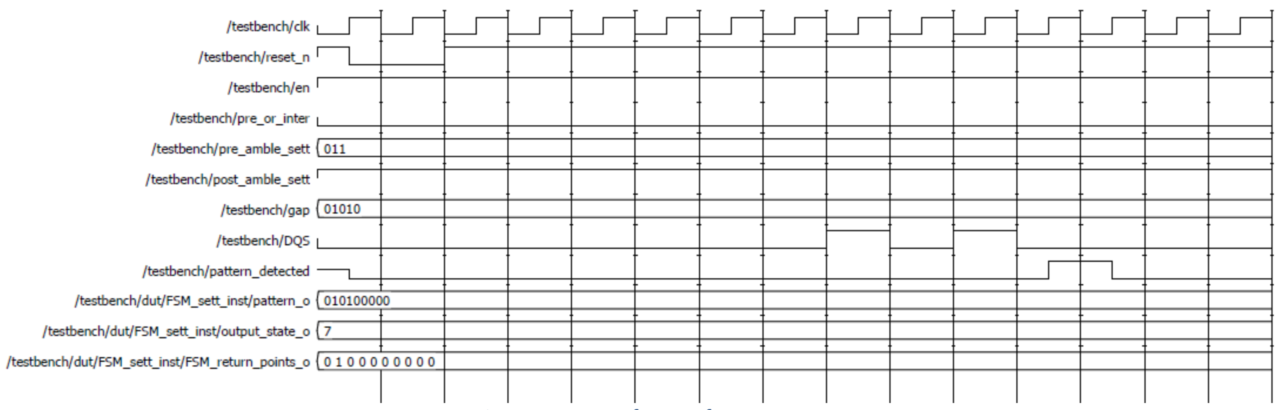


Figure 69. Waveforms of test case.

4.5.4 *Frequency Ratio Manger*

This block is divided into two subblocks, serializer and deserializer. The serializer function is to serialize the input data according to the frequency_ratio value whether it is 1:1 (transferred as it is), 1:2 (serialized with double the frequency), or 1:4 (serialized with four times the frequency). Each clock cycle of *dfi_clk* input data enters the serializer and the data is serialized according to the *dfi_phy_clk* ratio to the *dfi_clk*. The block relies on a case statement deciding the maximum value of the counter, which is the 2-level multiplexer selector which selects what data to be transferred on the output serial channel. The two clocks are edge-aligned, produced from the same PLL so there is no need for CDC (FIFO or bit synchronizer). The block works only at positive edge clock of both *dfi_clk* and *dfi_phy_clk* as to avoid working on data received on negative edge of *dfi_clk* which corresponds to positive edge of *dfi_phy_clk* in case of frequency ratio 1:2 or 1:4. The deserializer function is to deserialize the output data to be out in parallel each dfi_clk cycle positive edge on each of the output data channels according to the frequency_ratio value. The output data is received on the deserializer and on the next positive edge clock cycle of *dfi_clk* is transferred to the output channels. The data should follow the concept of data rotation mentioned in JEDEC standard. Which requires data to rotate on the output channels. For example, if frequency ratio is 1:2 and the data is output on first two channels then the following data will be output on the last two channels. A finite state machine determines which channels to select for the output according to the last output and the frequency ratio value. The data is shifted in internal shist registers and is output according to frequency ratio and last state. Table 21 explaines the functionality of the block. Table 21 simmarizes the functionality of the block. Figure 70 shows the block diagram of the serializer while figure 71 shows the block diagram of the deserializer. The block was then verified and figure 72 show that the serializer is working for a frequency ratio of 1:1. Figure 73 then show the results for the serializer with a frequency ratio of 1:4. Finally, figures 74 and 75 show the results of the deserializer using a frequency ratio of 1:1 and 1:4 respectively.

Table 21. Frequency ratio manager functionality.

Inputs	Outputs	Functionality	Parameters
dfi_address_pN [14 bits] dfi_cs_pN [1 bit] dfi_rddata_en_pN [1 bit] dfi_alert_n [1 bit] dfi_rddata_valid [1 bit] dfi_rddata [8 bits]	dfi_alert_n_aN [1 bit] dfi_rddata_valid_wN [1 bit] dfi_rddata_wN [8 bits] PHY dfi_rddata_en [1 bit] PHY dfi_address [14 bits] PHY dfi_cs [1 bit]	1- Serializes inputs coming from MC to PHY based on the dfi_freq_ratio parameter and on the DFI PHY clock. 2- Deserializes inputs coming from PHY to MC based on the dfi_freq_ratio parameter and on the DFI clock.	dfi_freq_ratio [1:0]

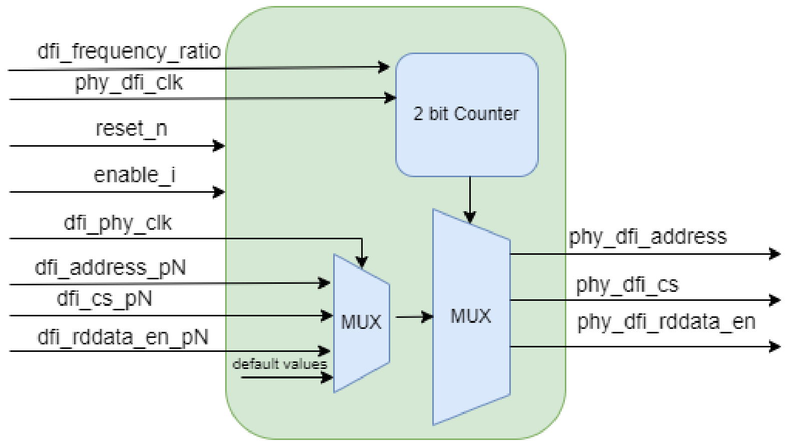


Figure 70. Serializer block diagram.

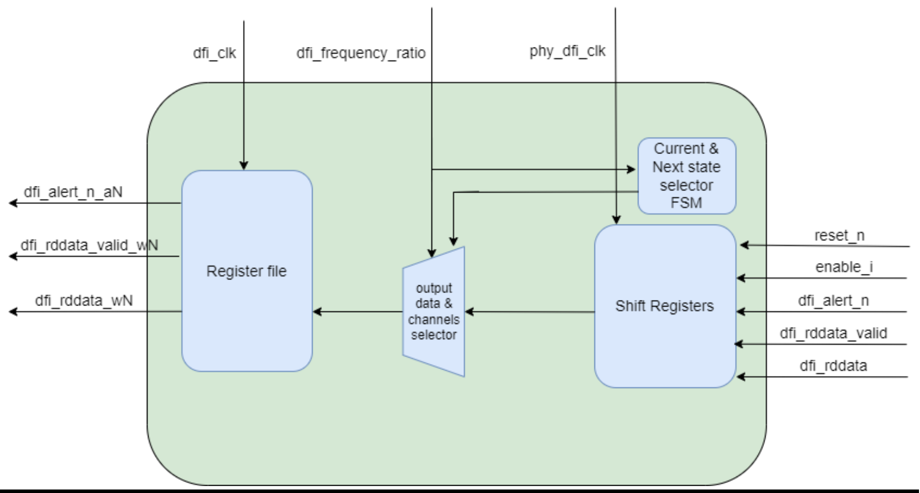


Figure 71. Deserializer block diagram.

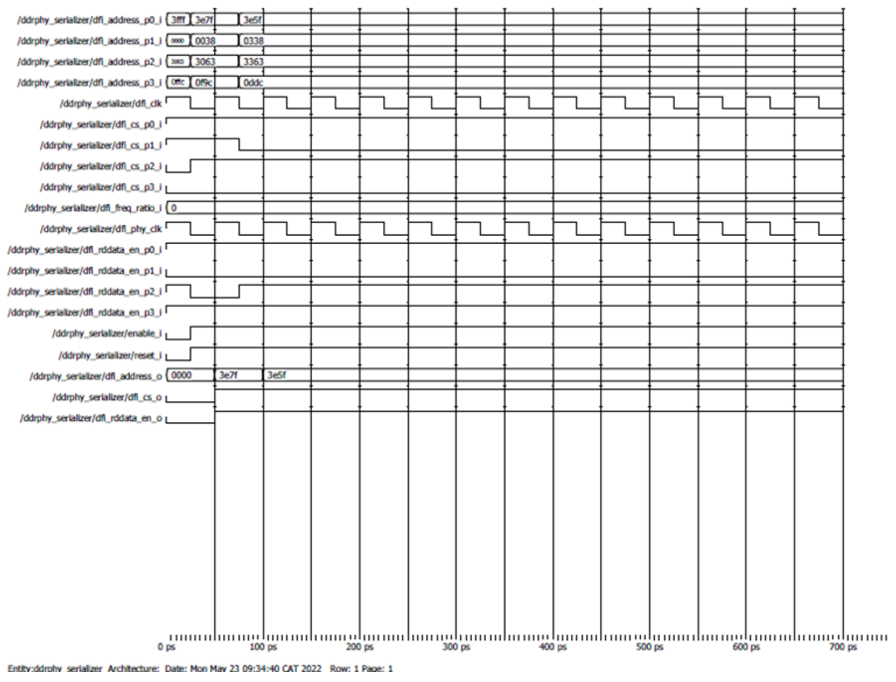


Figure 72. Serializer simulation for frequency ratio 1:1.

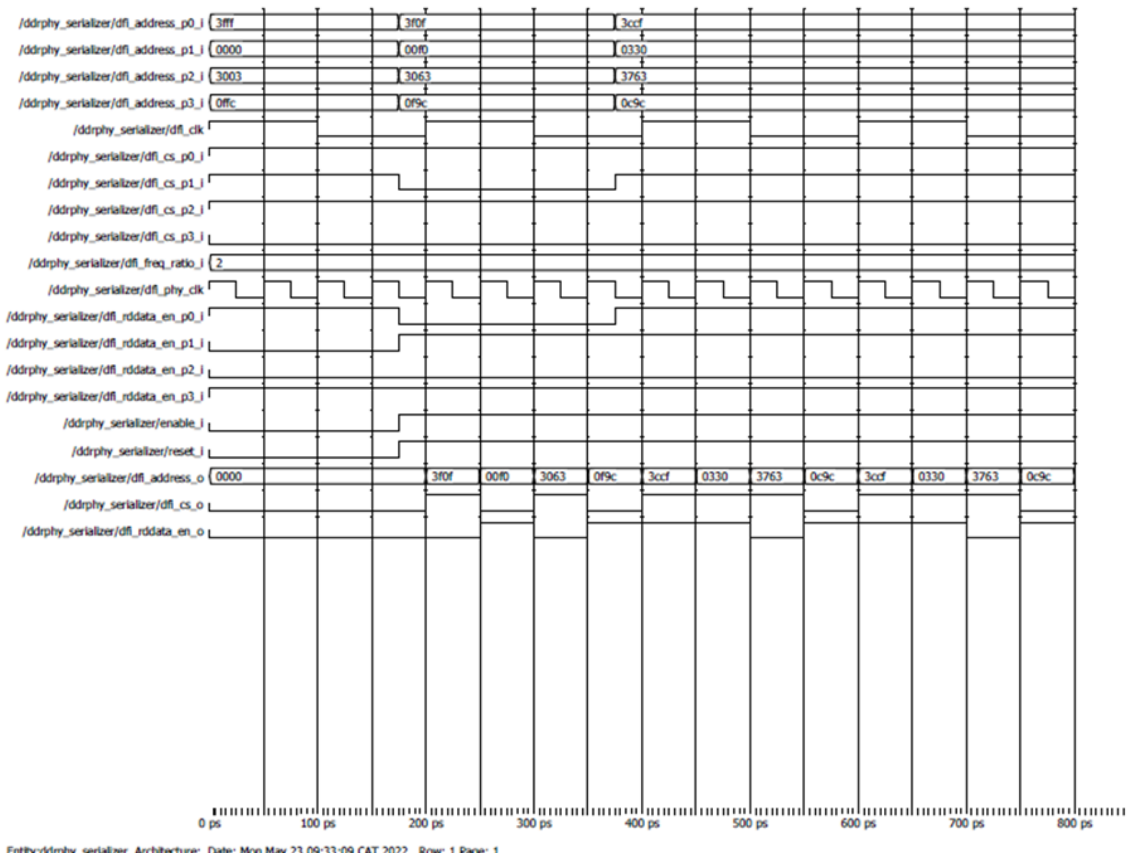


Figure 73. Serializer simulation for frequency ratio 1:4.

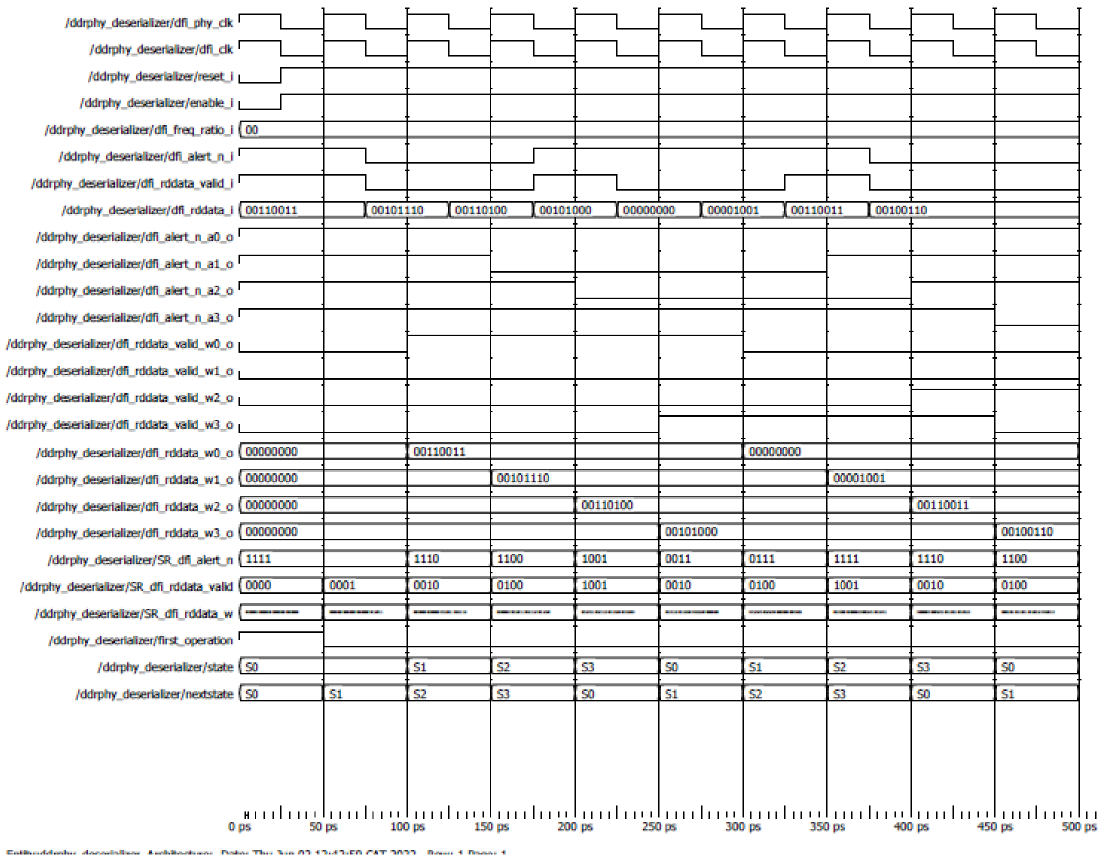


Figure 74. Deserializer simulation for frequency ratio 1:1.

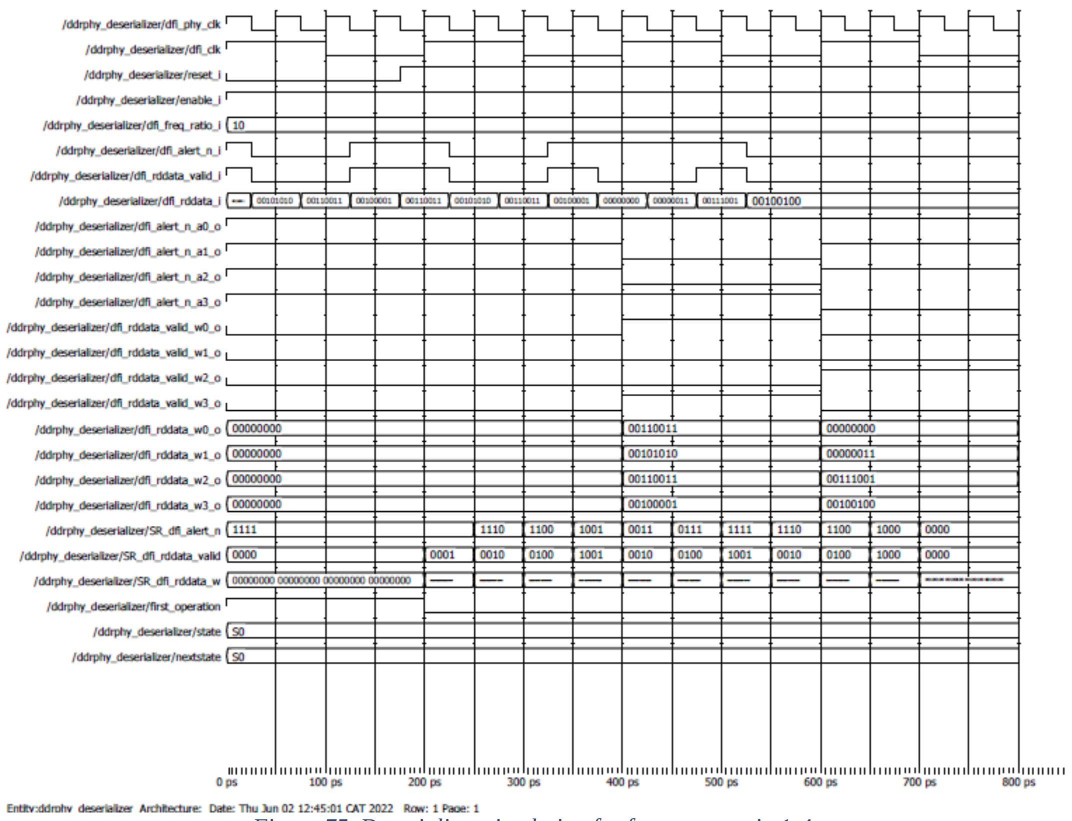


Figure 75. Deserializer simulation for frequency ratio 1:4.

4.5.5 CRC

CRC validation block function is to receive nine packets each of one byte; the first eight bytes represent the data and the last byte represents its CRC. In case of burst length of eight, then the first four packets are data and the following four packets are ones, followed by the ninth CRC byte. The CRC Validation block works on nine packets of data whether the burst length is eight or sixteen. The block is expected to receive continuous data, and for each nine packets of data, outputs a *dfi_alert_n* signal for one clock cycle. The data is transferred to sixty four bits *D* reg to start generating CRC bits and then compare it with the received ninth packet and outputs *the dfi_alert_ n* signal for one clock cycle. *The dfi_alert_n* signal is high by default, when a mismatch between received CRC byte (ninth packet) and generated CRC bits from the first eight packets, then error is been detected in the received data which consequently drive the *dfi_alert_n* signal low for one clock cycle. In this simulation, two consecutive 9 packets were received in the CRC Validation block. The first ta received is sent with correct CRC byte hence, dfi_alert_n remains high as default. However, the second data packets received are sent with wrong CRC byte which consequently drives the dfi-alert_n signal low for one clock cycle. Table 22 summarizes the functionality of the CRC block. The block diagram is then shown in figure 76. Finally, a simulation is shown in figure 77.

Table 22. CRC functionality.

Inputs	Outputs	Functionality	Parameters
pre_rddata [8 bits] [8 bytes of data followed by a byte of CRC] pre_rddata_valid [1 bit]	dfi_alert_n [1 bit]	1- Starts the operation when pre_rddata_valid is asserted. 2- Validates DQ 3- Activates dfi_alert _n when an error occurs	phycrc_mode [1 bit] Read CRC enable [1 bit]

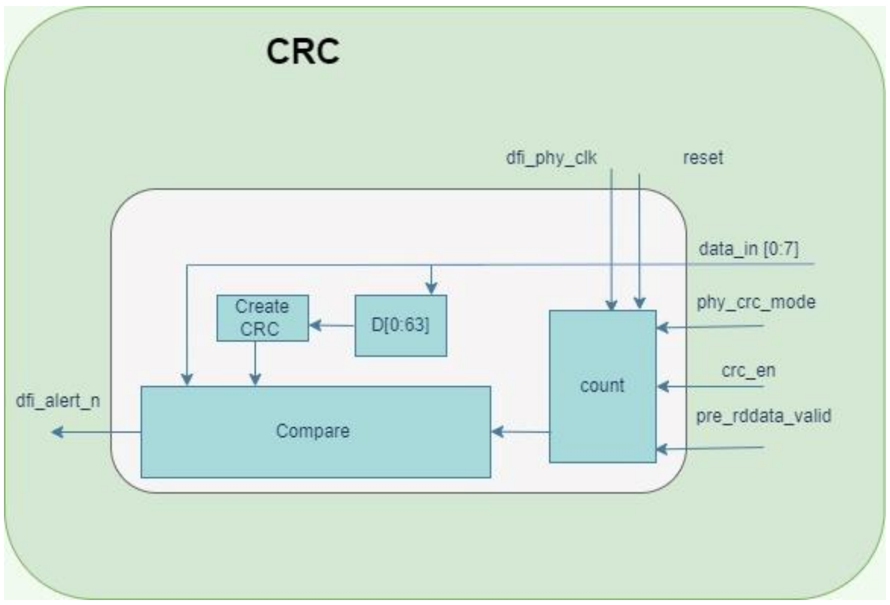


Figure 76. CRC block diagram.

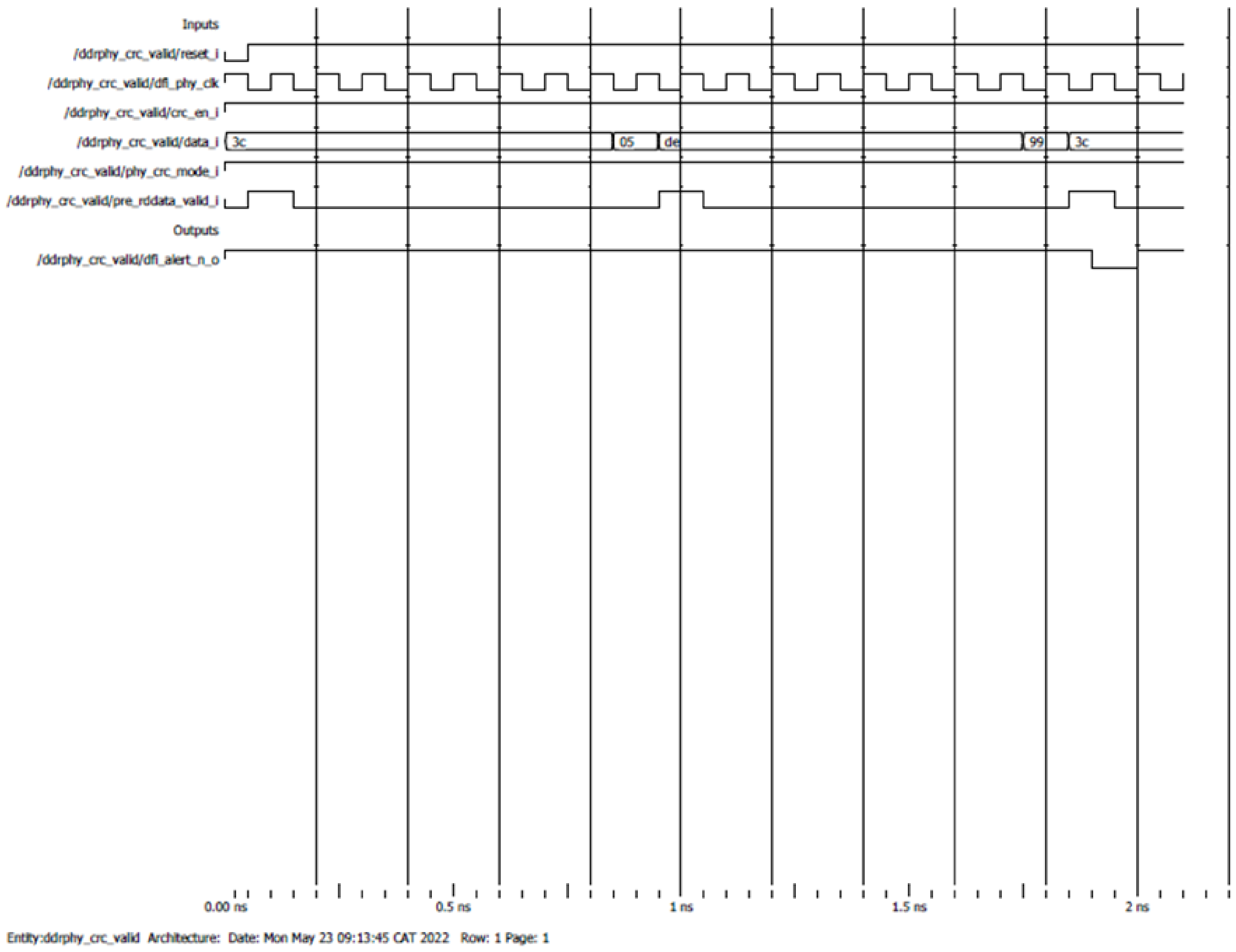


Figure 77. CRC simulation waveforms.

5- Project Execution

5.1 Project Tasks and Gantt chart

Tasks:

- Si-Vision VLSI training for basics and needed knowledge.
- Literature review on DRAMs and DDR (1,2,3,4).
- FCS block design (RTL).
- Standards (JEDEC & DFI) reading, analyzing, and summarizing.
- PHY architecture extraction and deduction from standards requirements.
- PHY modules design (RTLs).
- PHY block behavioral simulations (top level and separate modules level).
- Hardware testing (FPGA implementation).
- Thesis documentation.

The Gantt Chart is shown in figure 78.

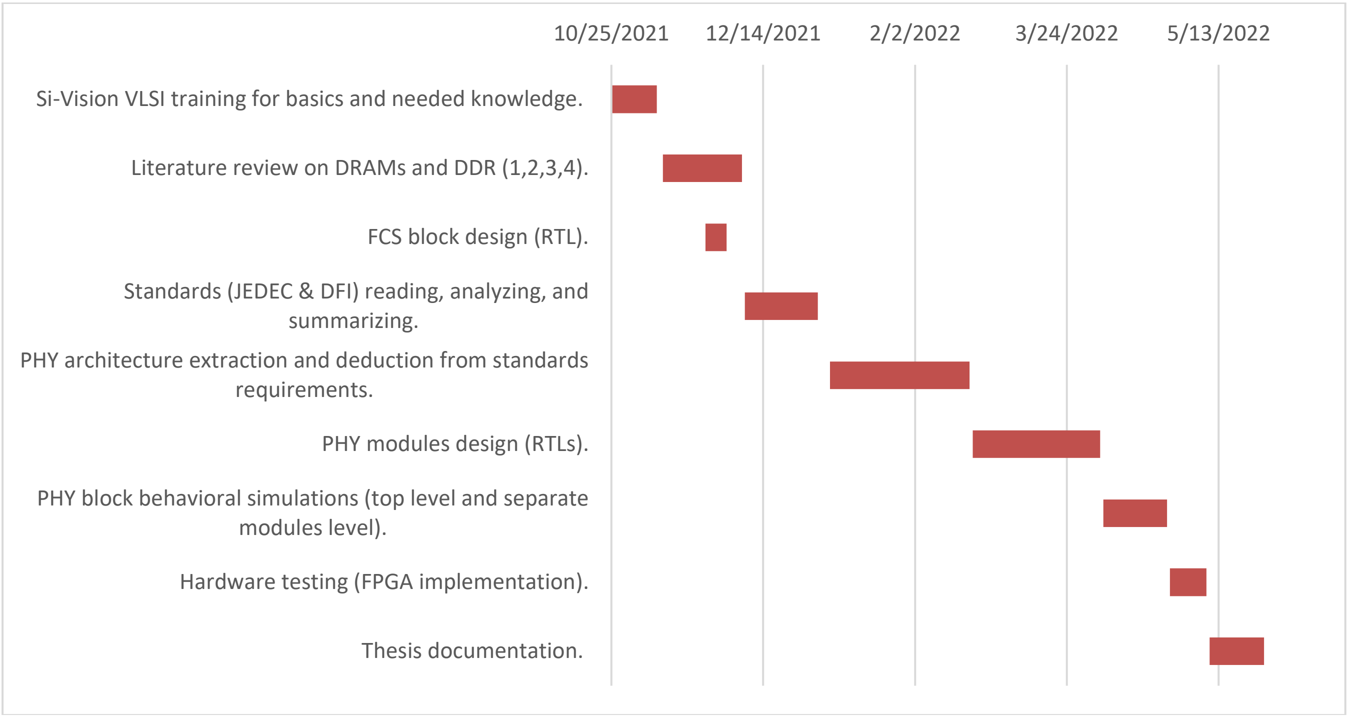


Figure 78. Gannt Chart.

5.2 Description of each subsystem

The project manufacturing can be done as an Application Specific Integrated Circuit (ASIC) and first a porotype on FPGA can be done. A simplified ASIC flow flowchart is shown in figure 79. However, it is very costly to have a state-of-the-art ASIC chip manufactured. The masks used in the fabrication process costs are high. A chip on 35 nm technology node may cost 1 million dollars. Consequently, we will go with the FPGA prototyping. A FPGA flow flowchart is shown in figure 80. FPGA protype will not cost us any money as the FPGAs are provided by UST at Zewail City.

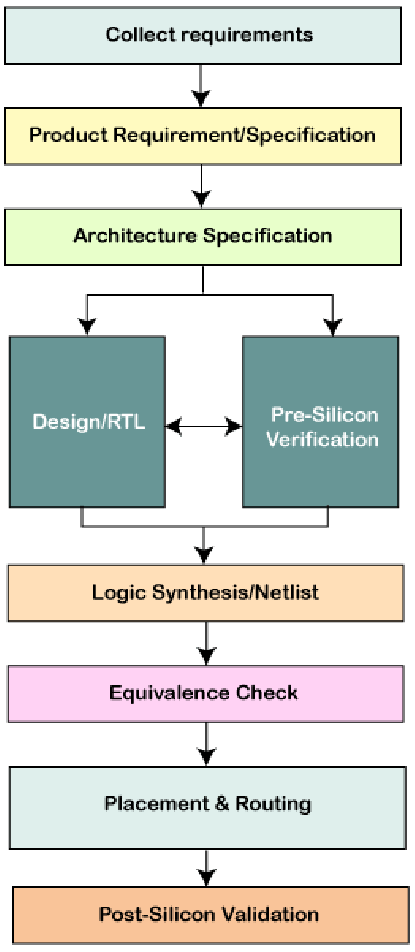


Figure 79. ASIC Design Flowchart.

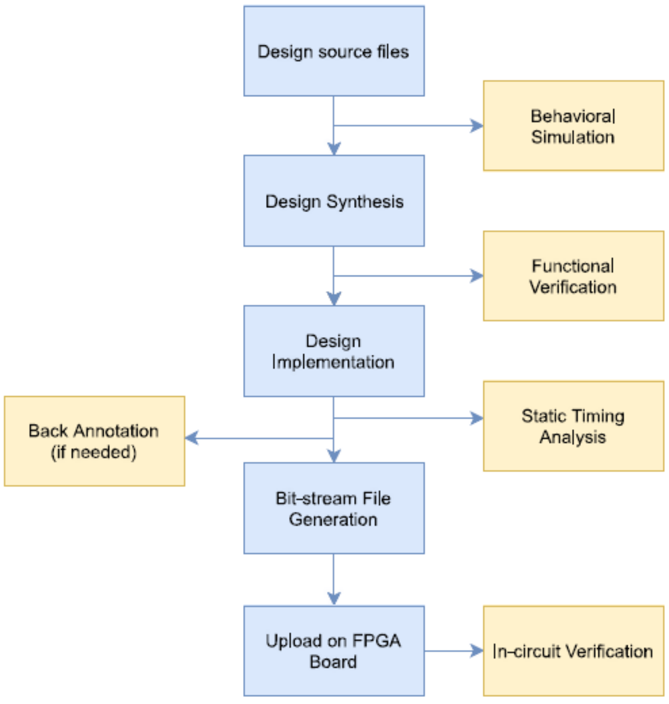


Figure 80. FPGA Design Flowchart.

5.3 Standard Usage in Project Execution

The project is a direct implementation of the DFI V.5.0 and JESD standards as the memory controller (MC) and PHY interface is defined by DFI V.5.0 standard while the PHY DRAM interface is defined by JEDEC JESD79-5A standard. Hence, the 2 standards were used in the project execution in 4 phases: First, in the identification of the features of the system in which the standards were the main reference. Second, in the design process of the architectures and the individual blocks as a reference for the specifications. Third, in the verification of the outputs of the whole integrated system to verify it is compatible with the standards of the DDR5 physical layer. Fourth, the SystemVerilog standard was used when describing the hardware as SystemVerilog is Hardware Description Language (HDL) that was used in this project.

5.4 Project Testing and Evaluation

The project verification is done using dynamic methods. A simulation of the standard waveforms has been done and verified. Also, the project was implemented as FPGA and as ASIC and then tested. Simulation and hardware testing passed the specifications illustrated before.

6- Simulation results of the integrated design

The full design was integrated and simulated using the cases in the standards. As seen in the following simulations, the test begins with the assertion of 2 two-cycle commands. The first one for setting the burst length and the second for setting the preamble and postamble settings. Then the data returns with the valid aligned to it. Figure 81 shows a test case for a pre-amble of 10 and an inter-amble of 10. Figure 82 shows a test case for a pre-amble of 00010 and an inter-amble of 0010. Figure 83 shows a test for a pre-amble of 00001010 and an inter-amble of 01010. Figure 84 shows a test for a pre-amble of 000010 and an inter-amble of 01010. Figure 85 tests for a pre-amble of 00001010 and an inter-amble of 0001010. Figure 86 tests for a pre-amble of 000010 and an inter-amble of 0100010. Figure 87 tests for a pre-amble of 00001010 and an inter-amble of 0101010. Figure 88 tests for a pre-amble of 000010 and an inter-amble of 010001010. Figure 89 tests for a pre-amble of 1110 and a valid CRC of 11010111. Figure 90 tests for a pre-amble of 1110 and an invalid CRC of 11010111. Lastly, figures 91 and 92 test the system with frequency ratios of 1:2 and 1:4 respectively with CRC enabled, burst length of 8, the crc sent wrong in the first one then correct in the second one.

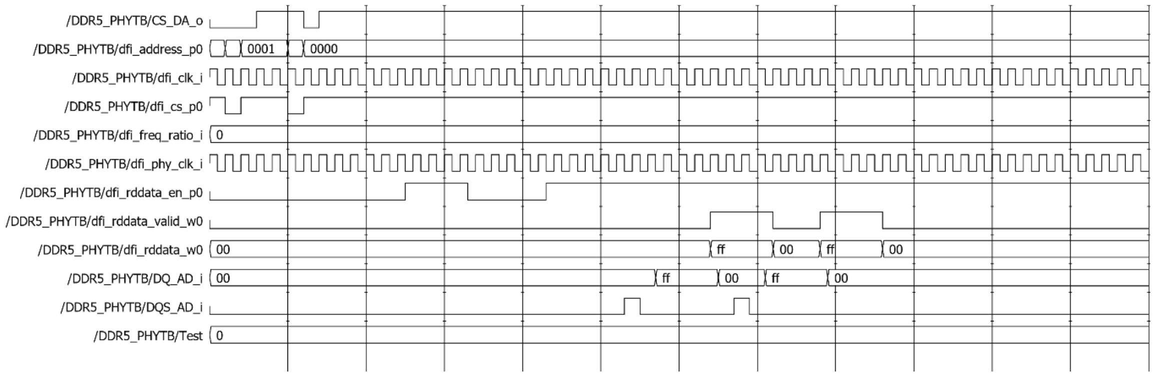


Figure 81. Pre-amble of 10 and an inter-amble of 10 testcase.

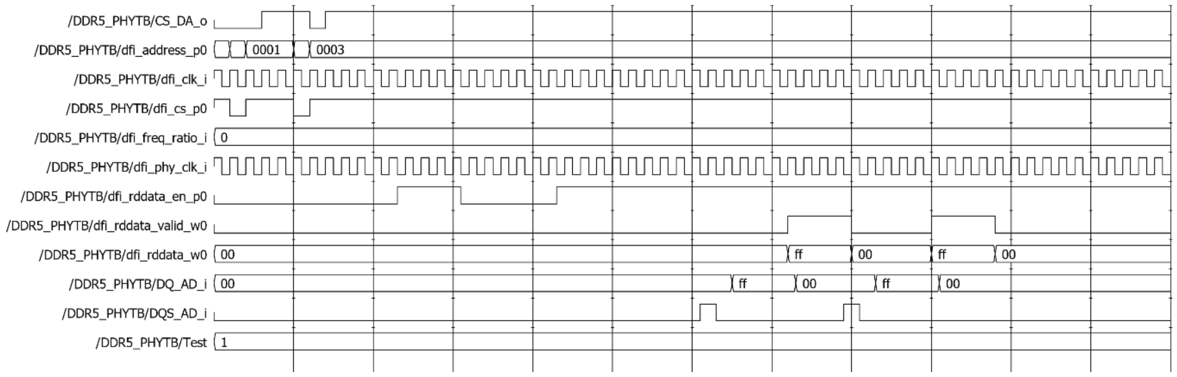


Figure 82. Pre-amble of 00010 and an inter-amble of 0010 testcase.

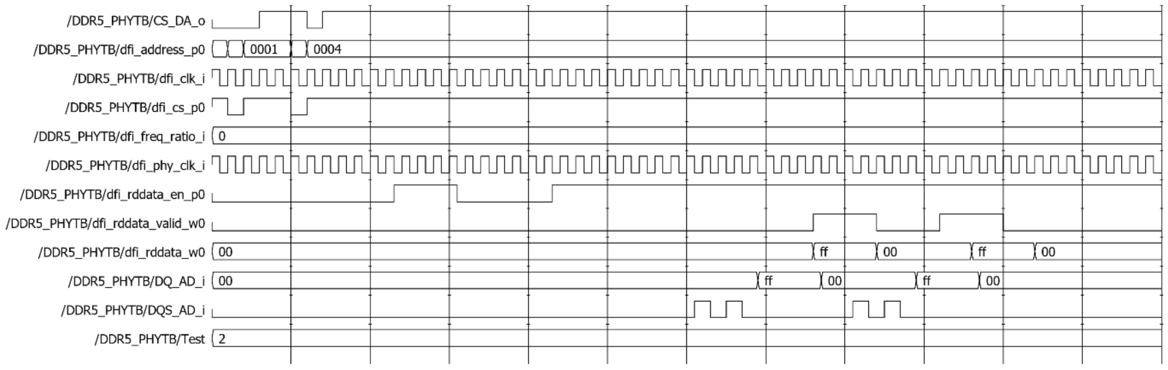


Figure 83. Test for a pre-amble of 00001010 and an inter-amble of 01010

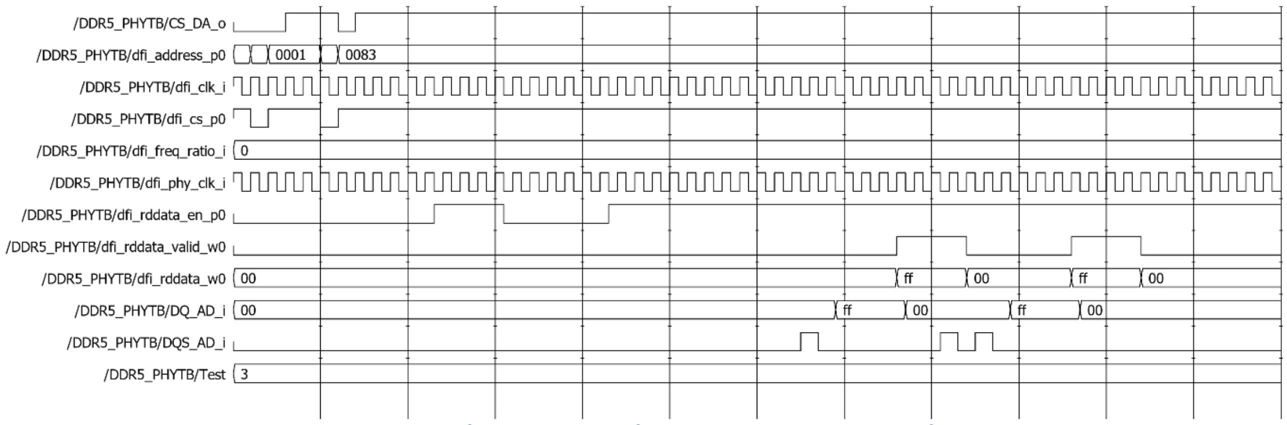


Figure 84. Test for a pre-amble of 000010 and an inter-amble of 01010.

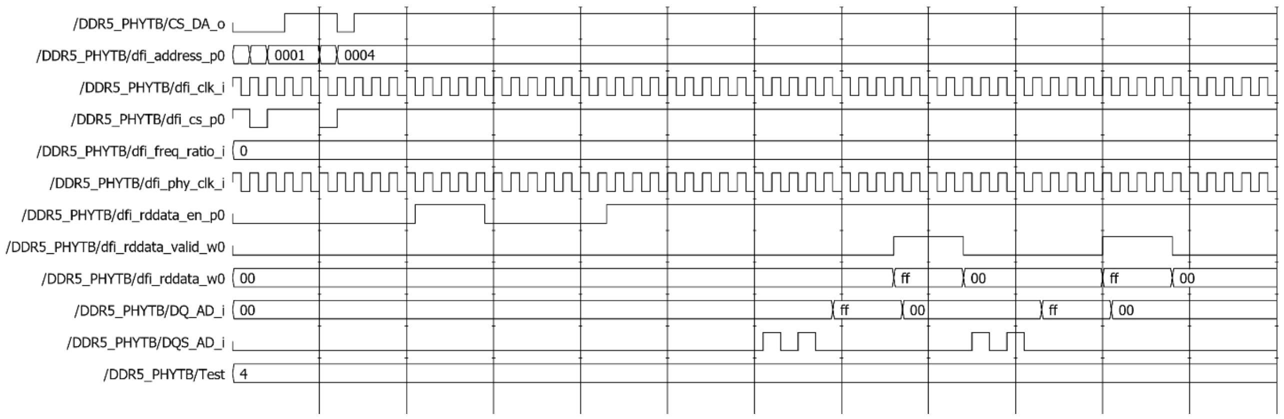


Figure 85. Test for a pre-amble of 00001010 and an inter-amble of 0001010.

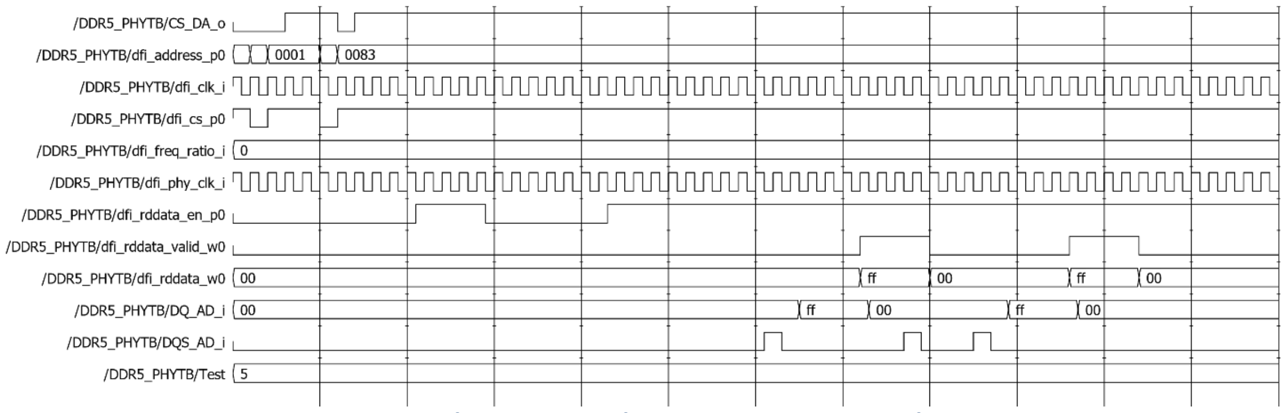


Figure 86. Test for a pre-amble of 000010 and an inter-amble of 0100010.

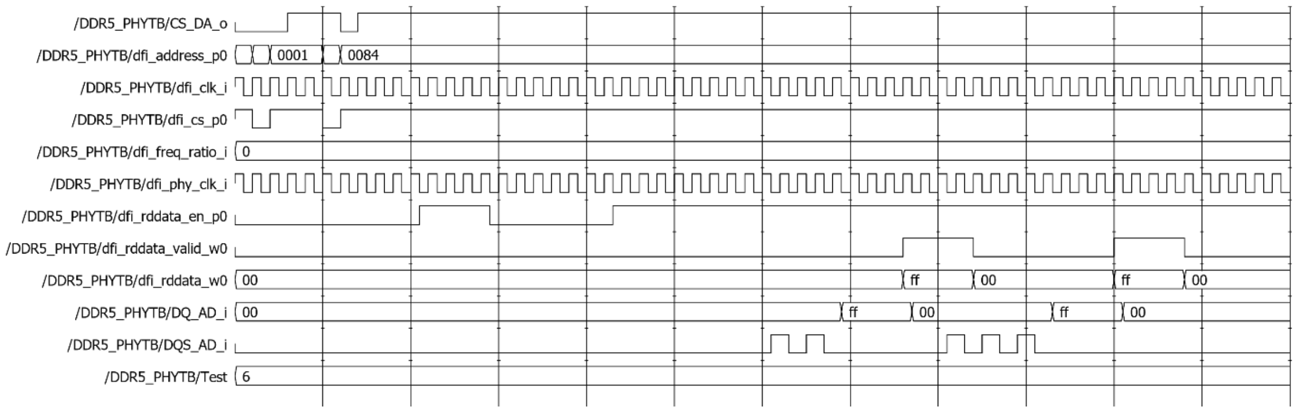


Figure 87. Test for a pre-amble of 00001010 and an inter-amble of 0101010.

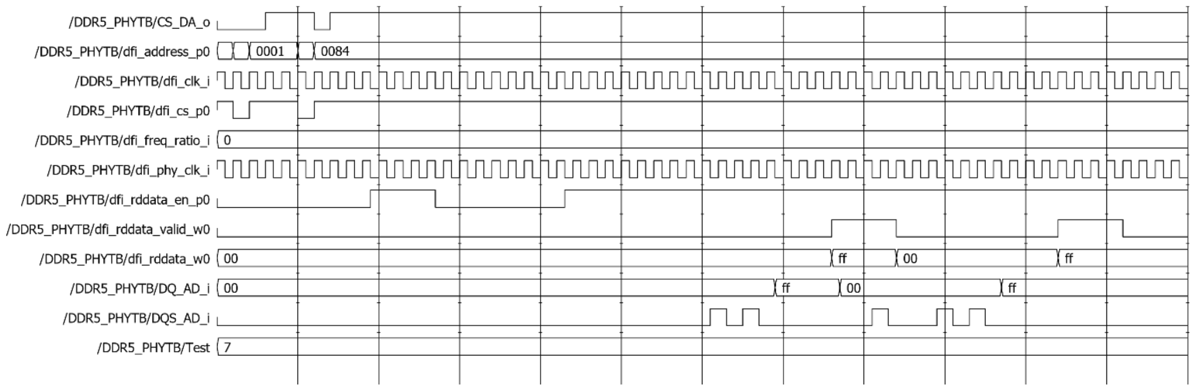


Figure 88. Test for a pre-amble of 000010 and an inter-amble of 010001010.

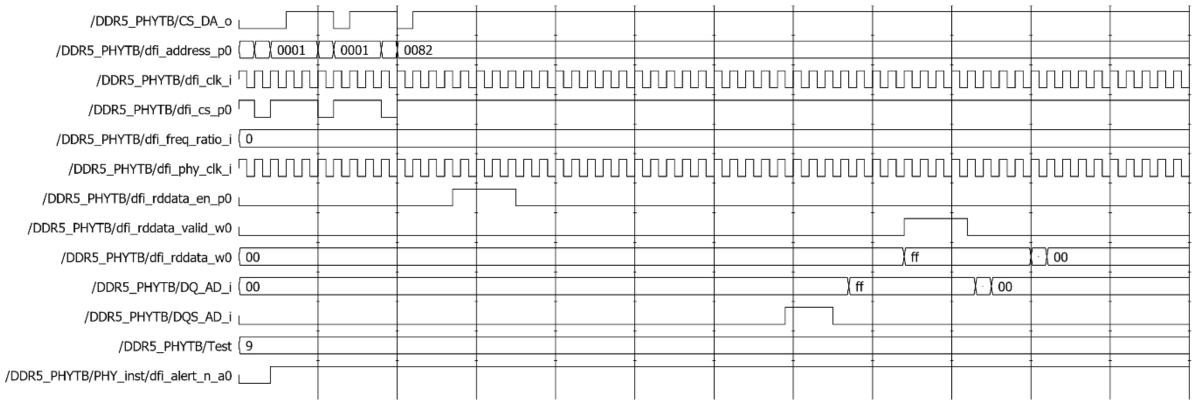


Figure 89. Test for a pre-amble of 1110 and a valid CRC of 11010111.

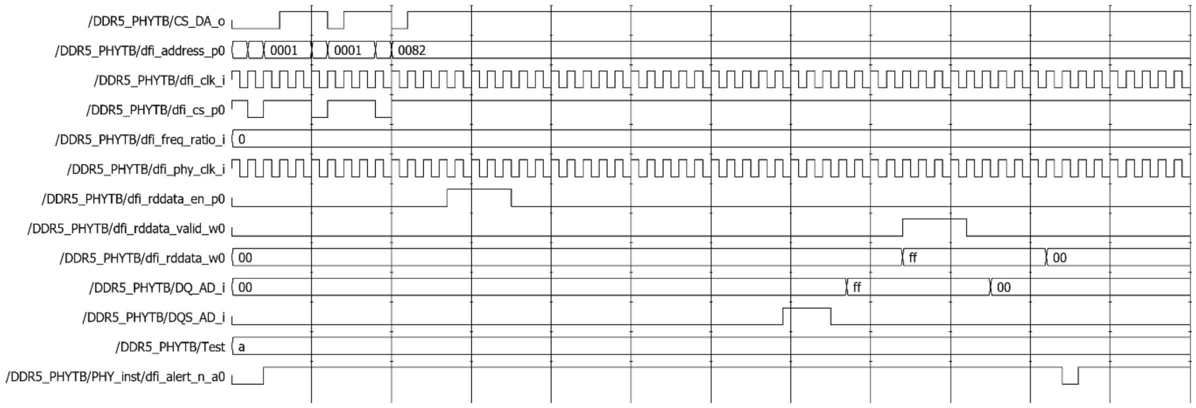


Figure 90. Test for a pre-amble of 1110 and an invalid CRC of 11010111.

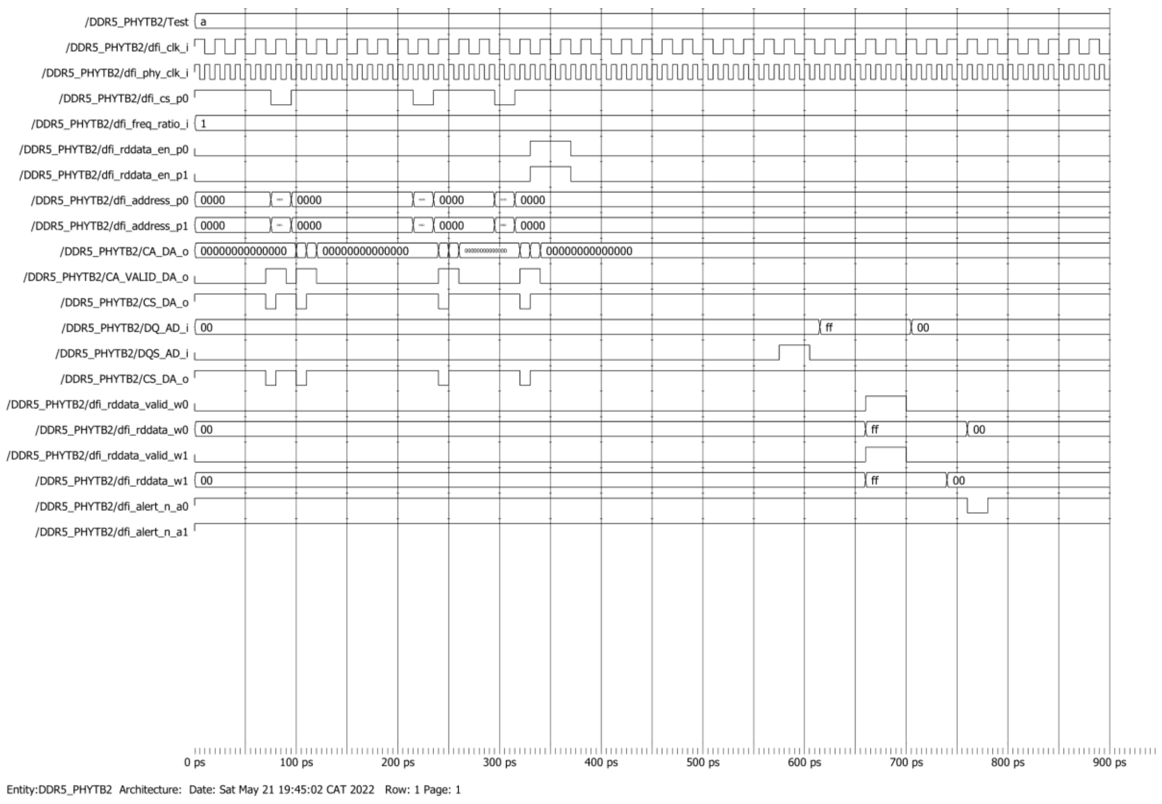


Figure 91. Testcase for frequency ratio 1:2 with CRC.

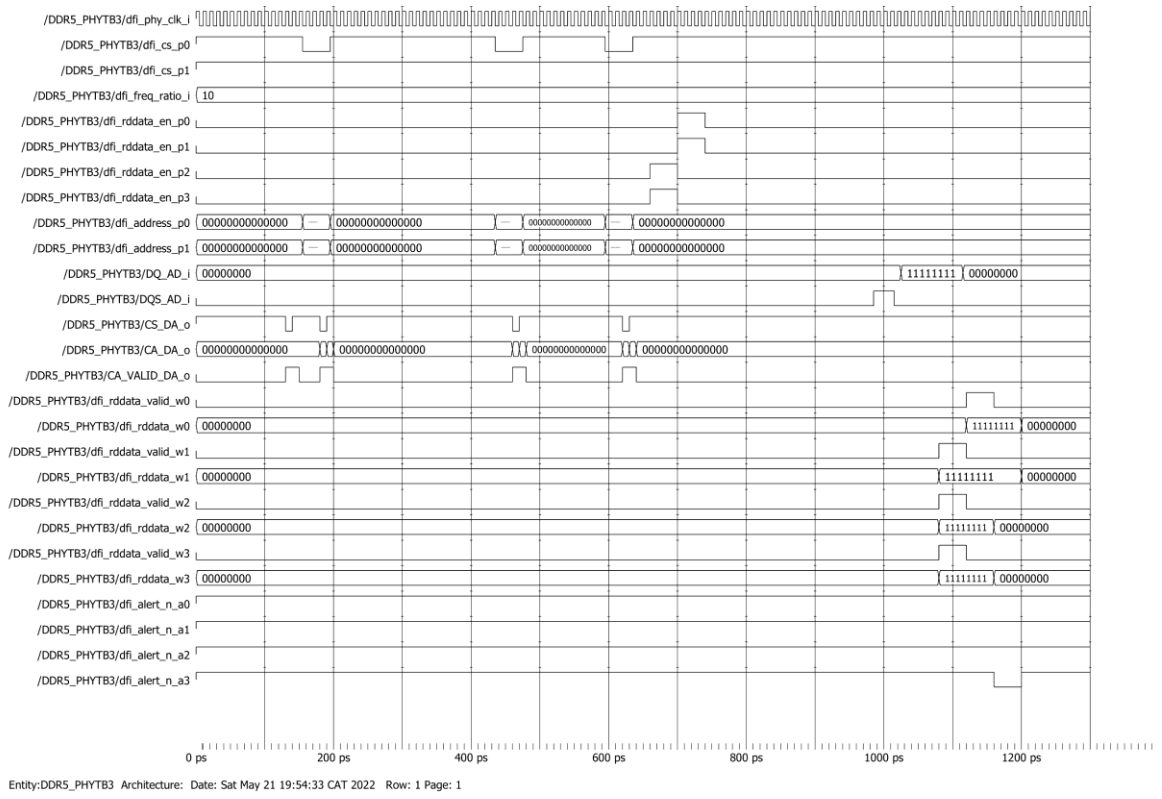
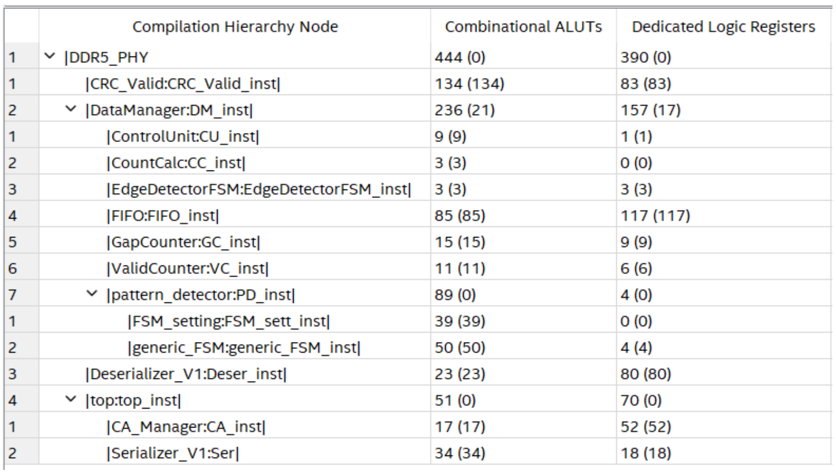


Figure 92. Testcase for frequency ratio 1:4 with CRC.

7- Hardware Implementation

7.1 FPGA

FPGA implementation was done using Intel Quartus tool and on the EP4CE22F17C6 Intel FPGA. Implementation results are shown in figure 93. After the implementation, a timing simulation was done. Timing simulation include the parasitic delays estimated of the physical circuit to the simulation using Standard Delay Format (SDF) files. The results of the timing simulation on FPGA are shown in figure 94. It is shown that the output signals are delayed after the clock positive edge due to the physical delays. It also shows that the block is functioning correctly after physical implementation. Figure 95 is a zoomed image of figure 94 in the region with the data coming from the DRAM to the PHY. The figure shows that the data is from 1 to 8 and it is correctly sent to the memory controller on different busses with valid signals correctly asserted as the frequency ratio here is 1:4.



	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers
1	⌄ |DDR5_PHY	444 (0)	390 (0)
1	|CRC_Valid:CRC_Valid_inst|	134 (134)	83 (83)
2	⌄ |DataManager:DM_inst|	236 (21)	157 (17)
1	|ControlUnit:CU_inst|	9 (9)	1 (1)
2	|CountCalc:CC_inst|	3 (3)	0 (0)
3	|EdgeDetectorFSM:EdgeDetectorFSM_inst|	3 (3)	3 (3)
4	|FIFO:FIFO_inst|	85 (85)	117 (117)
5	|GapCounter:GC_inst|	15 (15)	9 (9)
6	|ValidCounter:VC_inst|	11 (11)	6 (6)
7	⌄ |pattern_detector:PD_inst|	89 (0)	4 (0)
1	|FSM_setting:FSM_sett_inst|	39 (39)	0 (0)
2	|generic_FSM:generic_FSM_inst|	50 (50)	4 (4)
3	|Deserializer_V1:Deser_inst|	23 (23)	80 (80)
4	⌄ |top:top_inst|	51 (0)	70 (0)
1	|CA_Manager:CA_inst|	17 (17)	52 (52)
2	|Serializer_V1:Ser|	34 (34)	18 (18)

Figure 93. FPGA implementation results.

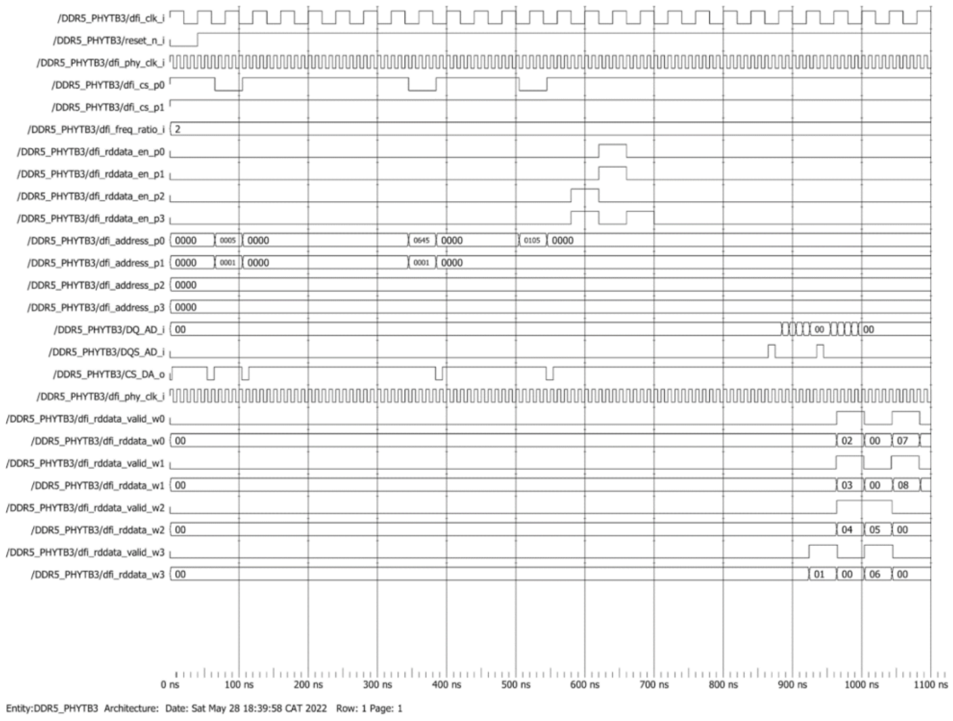


Figure 94. FPGA timing simulation.

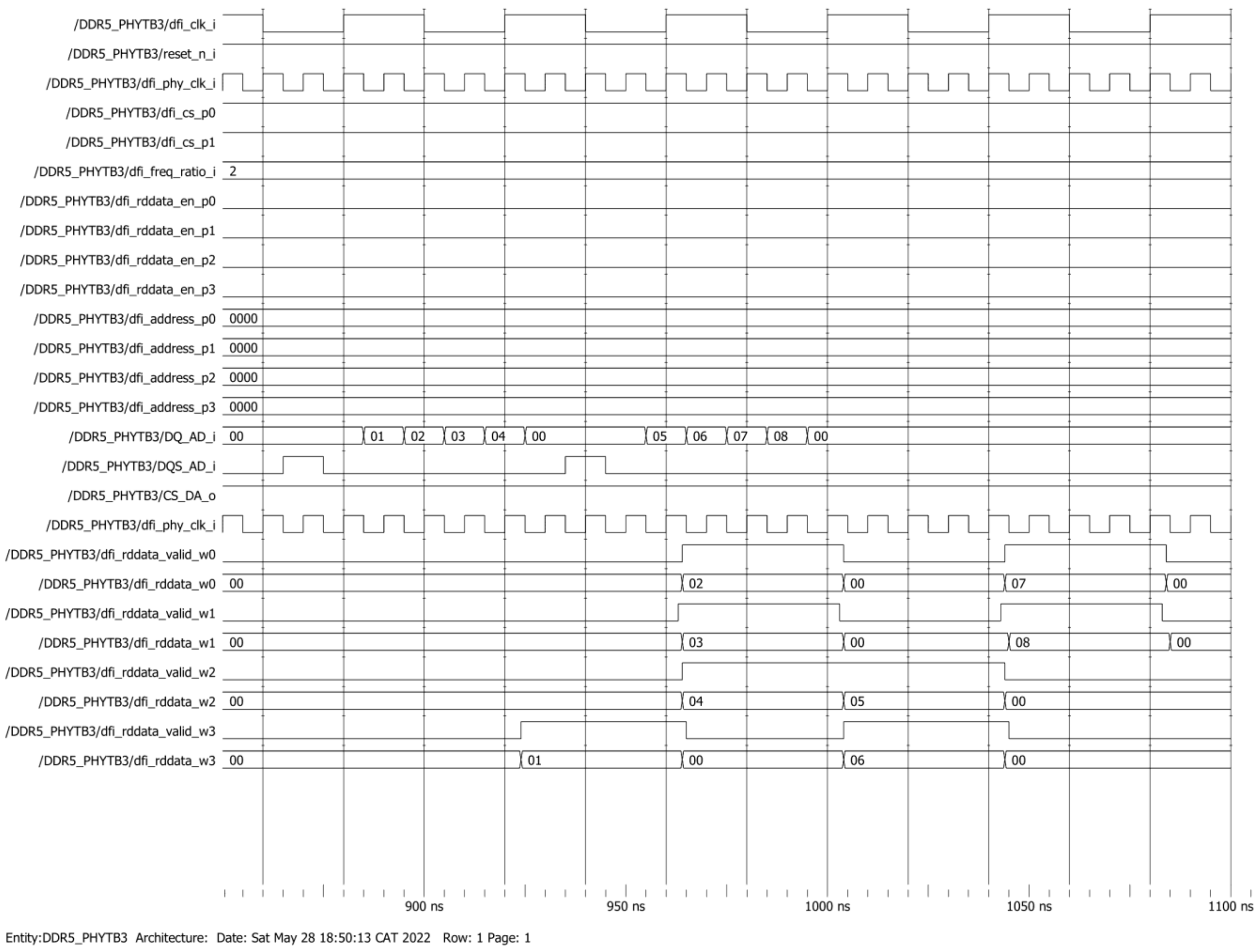
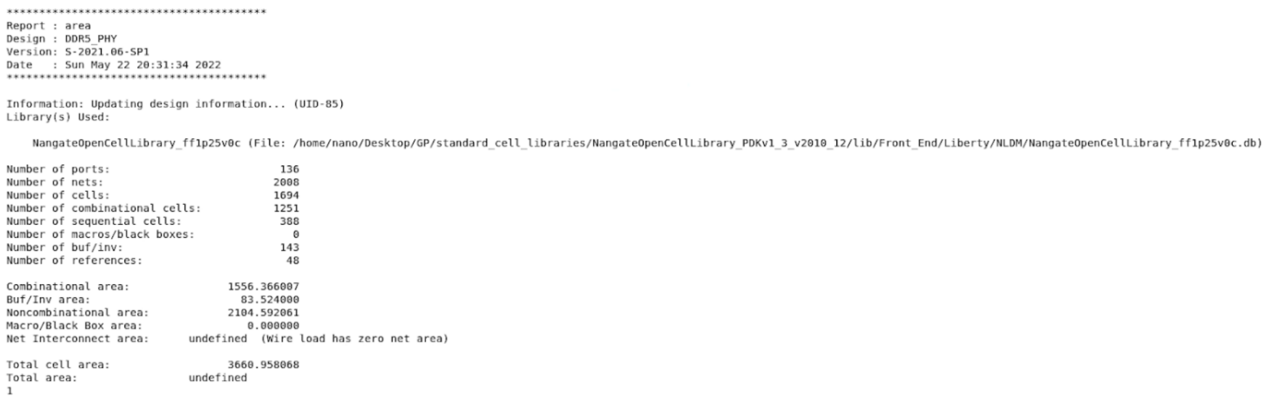


Figure 95. FPGA timing simulation data forwarding.

7.2 ASIC

The ASIC implementation include synthesis, floorplaning, powerplaning, placement, clock tree synthesis, routing, finishing and physical verification. Synthesis is the process of transforming the HDL design into a gate-level netlist, given all the specified constraints and optimization settings. Logic synthesis is the process of translating and mapping RTL code written in HDL which we wrote using the SystemVerilog language into technology specific gate level representation. The mapping allows the place and route tools afterwards to be able to get the required connection between the circuits, the recommended placement for each circuit to ease the routing process, and the added parasitics due to the routing process. As a result, the tool iterates to determine the best solution that makes the chip able to satisfy the constraints specified during the synthesis and PnR processes. The constraints set during the synthesis is critical as it is the guide the tool uses to respect all the timing and area specifications needed in the design. Also, the technology used is a key player in the synthesis process, as the synthesis is the first step to know how your design will react with the real technology that will be used in manufacturing. Consequently, the technology node and the standard cells library used during the synthesis critically affects the synthesis process. In this project, we used the nangate 45nm open-source technlogy library. Synopsys Design Compiler (DC) was used for synthesis, IC-Compiler (ICC) was used for placement and routing, Formality for formal verification, and PrimeTime was used for timing verification.

The synthesis output results are shown in figure 96. After the successful synthesis process, the placement and routing processes are done. Firstly, floorplanning is done to choose the most suitable place for each sub block to reduce the side effects of placement and routing, such as, clock skew, added capacitance and resistance, and having unrouteable cells. Floorplaning and powerplaning are shown in fiugre 97. The design dimensions are 144 um * 144 um with a core utilization of 25%. The IR drop of the powerplan after optmization is shown in figure 98. The IR drop less than 18.46 mV which is less than 2% of the power supply that is 1.2 V. The congestion map of the floorplan is shown in figure 99 which is acceptable to begin the placement process.



```
****************************************
Report : area
Design : DDR5_PHY
Version: S-2021.06-SP1
Date   : Sun May 22 20:31:34 2022
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    NangateOpenCellLibrary_ff1p25v0c (File: /home/nano/Desktop/GP/standard_cell_libraries/NangateOpenCellLibrary_PDKv1_3_v2010_12/lib/Front_End/Liberty/NLDM/NangateOpenCellLibrary_ff1p25v0c.db)

Number of ports:                          136
Number of nets:                          2008
Number of cells:                         1694
Number of combinational cells:           1251
Number of sequential cells:               388
Number of macros/black boxes:               0
Number of buf/inv:                        143
Number of references:                      48

Combinational area:               1556.366007
Buf/Inv area:                       83.524000
Noncombinational area:            2104.592061
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                  3660.958068
Total area:                 undefined
1
```

Figure 96. Synthesis results report.

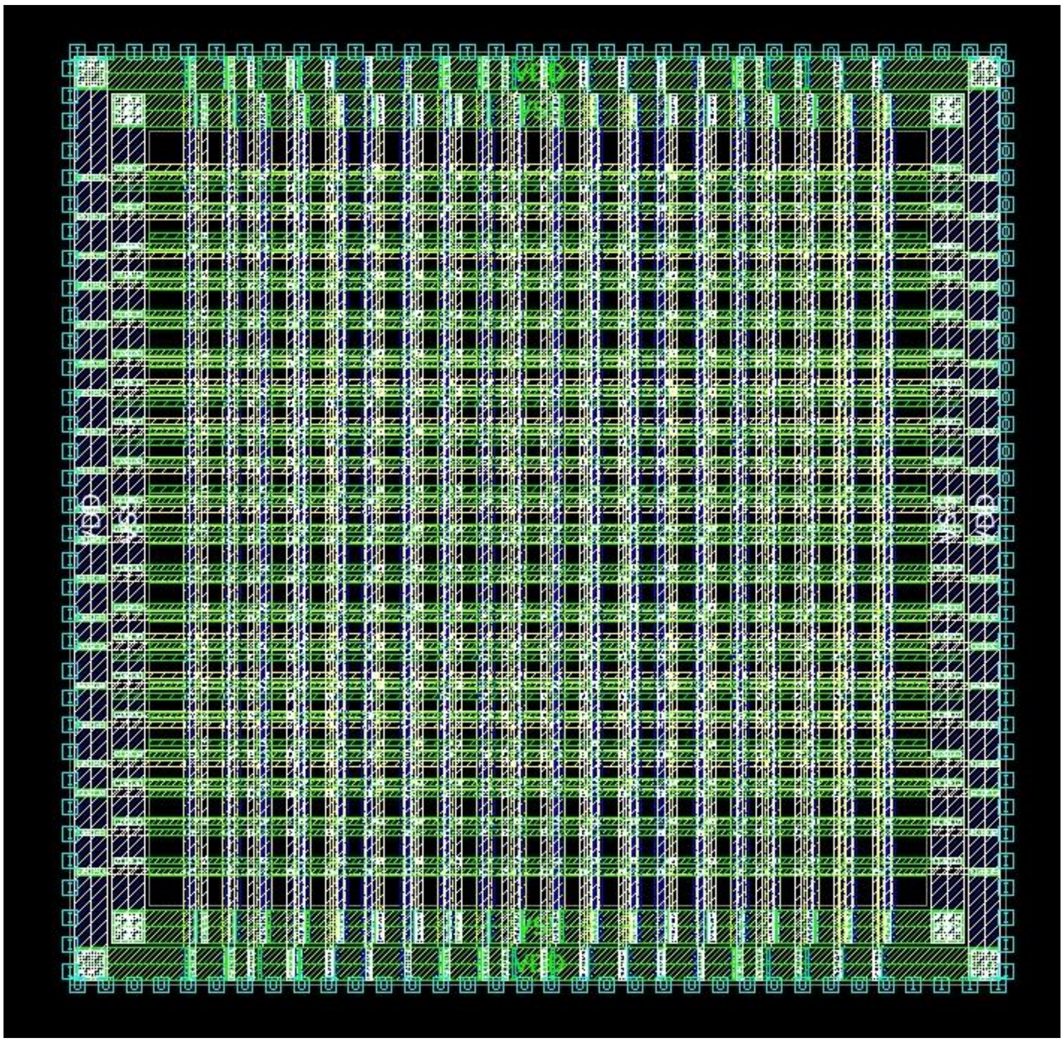
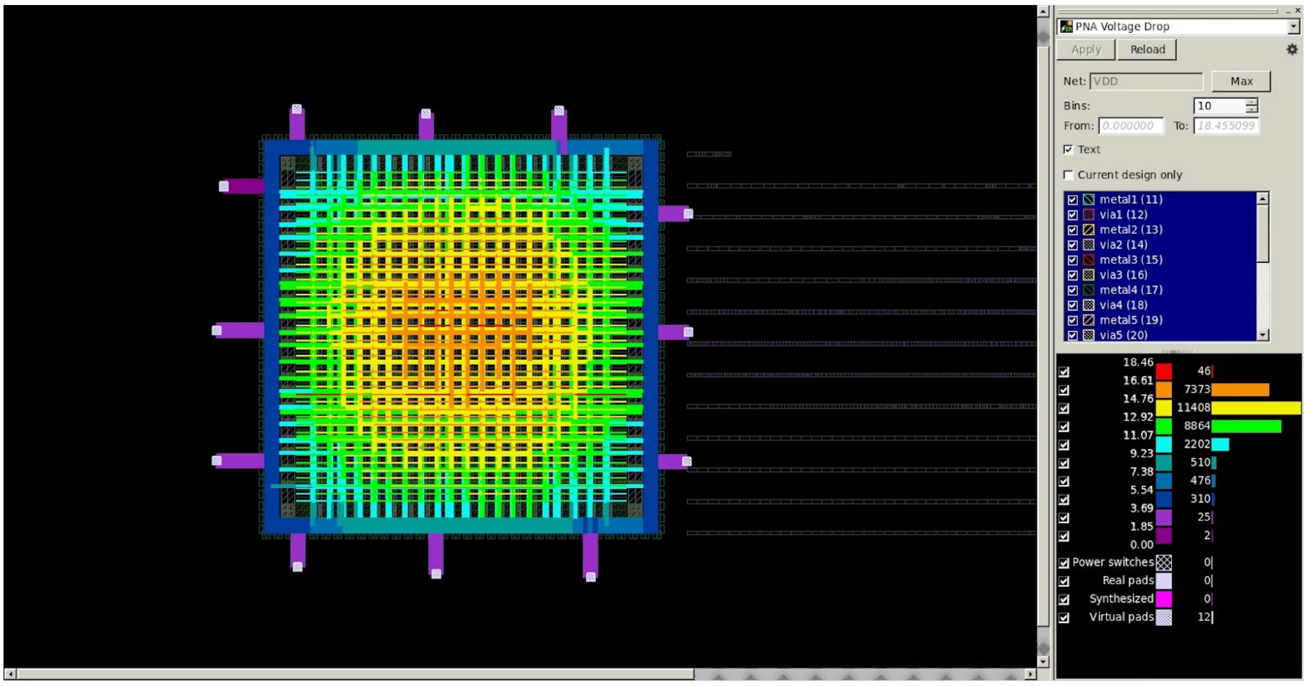


Figure 97. Floor and power planning.

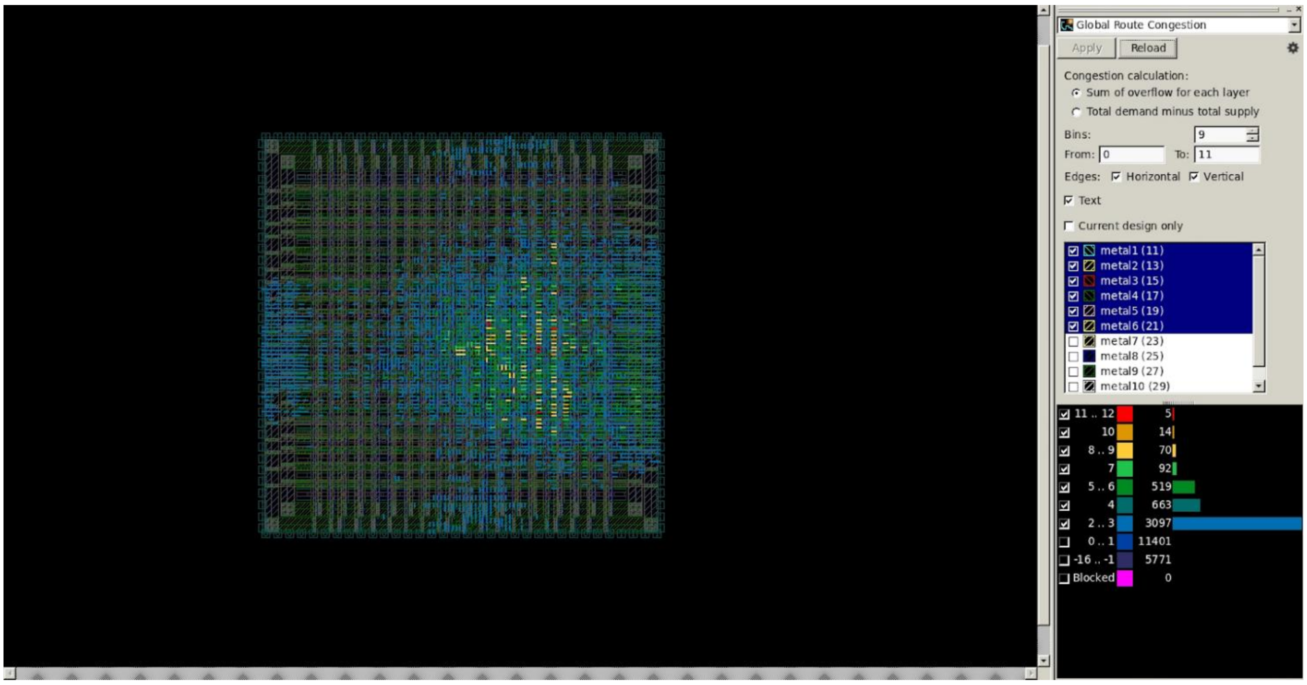


Net: VDD
Bins: 10
From: 0.000000 To: 18.455099
☑ Text
☐ Current design only

☑ metal1 (11)
☑ via1 (12)
☑ metal2 (13)
☑ via2 (14)
☑ metal3 (15)
☑ via3 (16)
☑ metal4 (17)
☑ via4 (18)
☑ metal5 (19)
☑ via5 (20)

☑	18.46 – 16.61 🟥	46
☑	16.61 – 14.76 🟧	7373
☑	14.76 – 12.92 🟨	11408
☑	12.92 – 11.07 🟩	8864
☑	11.07 – 9.23	2202
☑	9.23 – 7.38	510
☑	7.38 – 5.54	476
☑	5.54 – 3.69 🟦	310
☑	3.69 – 1.85 🟪	25
☑	1.85 – 0.00 🟪	2
☑	Power switches ▨	0
☑	Real pads	0
☑	Synthesized	0
☑	Virtual pads	12

Figure 98. Powerplanning IR drop.



Congestion calculation:
(● Sum of overflow for each layer)
○ Total demand minus total supply
Bins: 9
From: 0 To: 11
Edges: ☑ Horizontal ☑ Vertical
☑ Text
☐ Current design only

☑ metal1 (11)
☑ metal2 (13)
☑ metal3 (15)
☑ metal4 (17)
☑ metal5 (19)
☑ metal6 (21)
☐ metal7 (23)
☐ metal8 (25)
☐ metal9 (27)
☐ metal10 (29)

☑	11 .. 12 🟥	5
☑	10 🟧	14
☑	8 .. 9 🟨	70
☑	7 🟩	92
☑	5 .. 6 🟩	519
☑	4	663
☑	2 .. 3 🟦	3097
☐	0 .. 1 🟦	11401
☐	-16 .. -1	5771
☐	Blocked	0

Figure 99. Floorplan congestion map.

The placement effort specified in this project is to target having less congestion and to optimize the timing of the cells, in order to be aware of placing the cells in locations that eventually allows the design to meet the required timing constraints. The placed design is shown in figure 100. The congestion map of the placed design is shown in figure 101 which is better than the floorplan congestion. Then, the routing step is performed by routing the power nets firstly, followed by the detailed routing, choosing to optimize the congestion and to reach timing closure. The routed design is shown in figure 102 and showning that there are no violations in DRCs or timing. The timing will be checked again later using PrimeTime. Layout vs Schematic (LVS) check is shown in figure 103. This check is nessacary to ensure that the physical implementation is equivelant to the netlist provided from the synthesis tool. This step checks if there a short between two signals or if there is an open signal or anyting that is functionaly different than the netlist. Finally, filler cells are added the design was finalized. Figure 104 shows the final layout of the design. The quality of results are then shown in figure 105 showing that there are no violating paths. Finally, timing closure was tested using PrimeTime and succeeded as shown in figure 106.

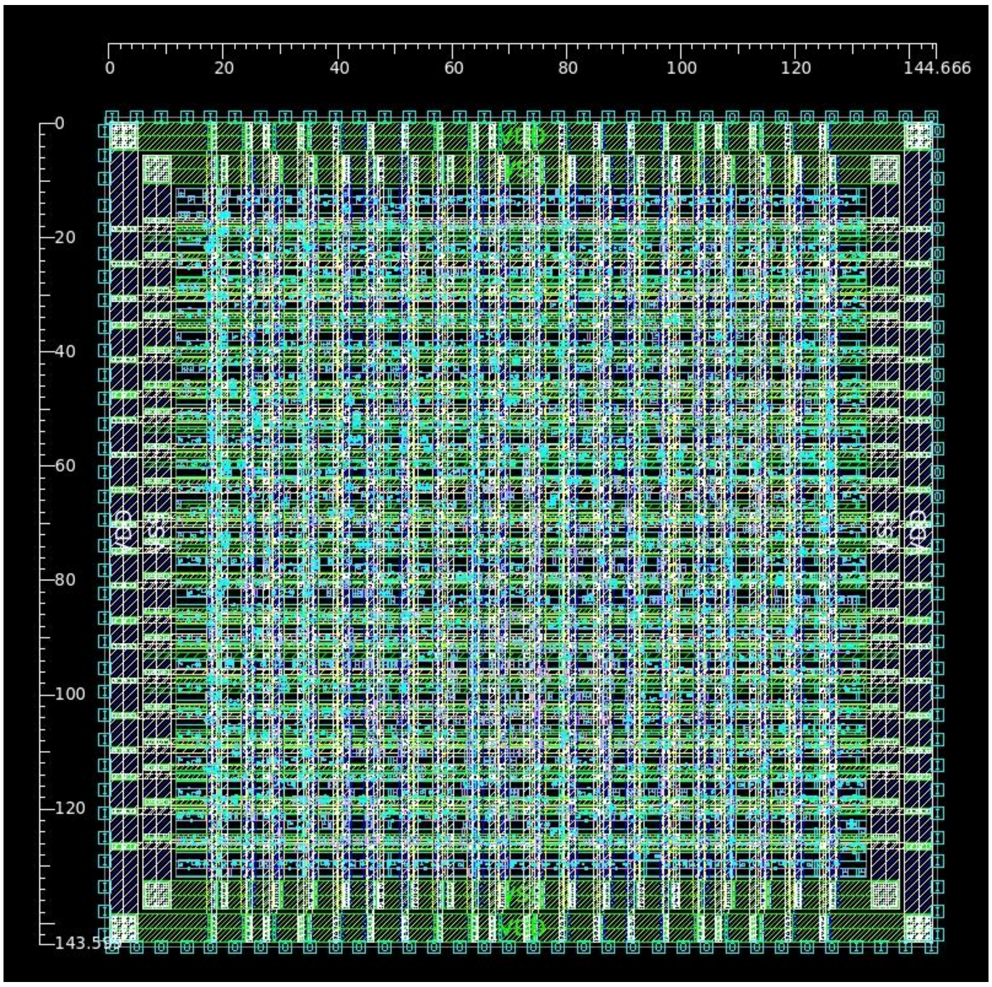
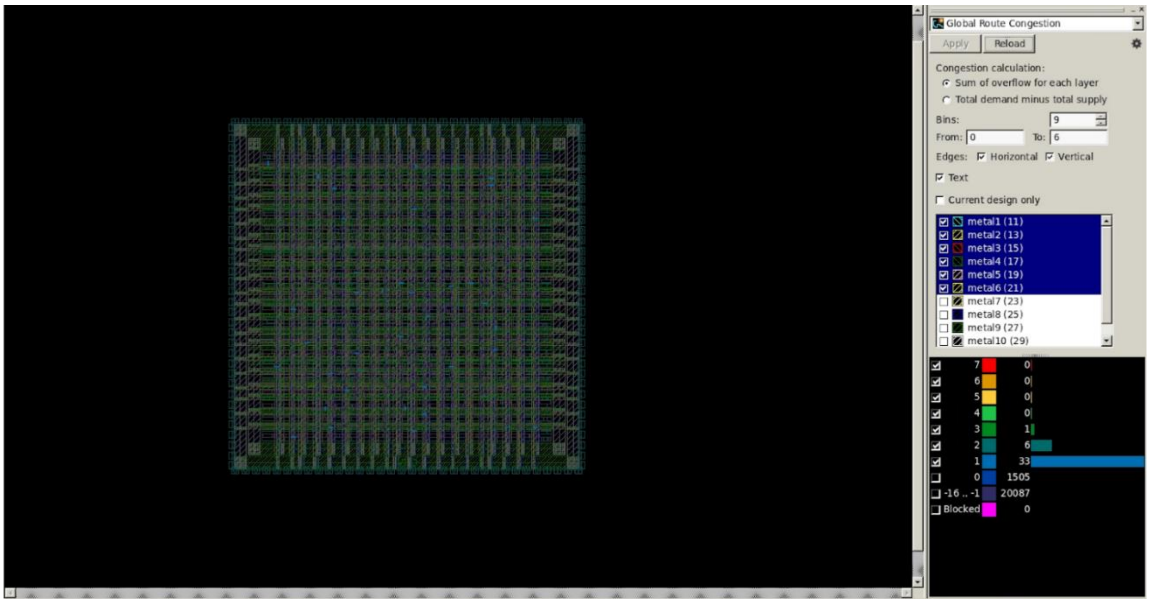


Figure 100. Placed design layout.



Global Route Congestion

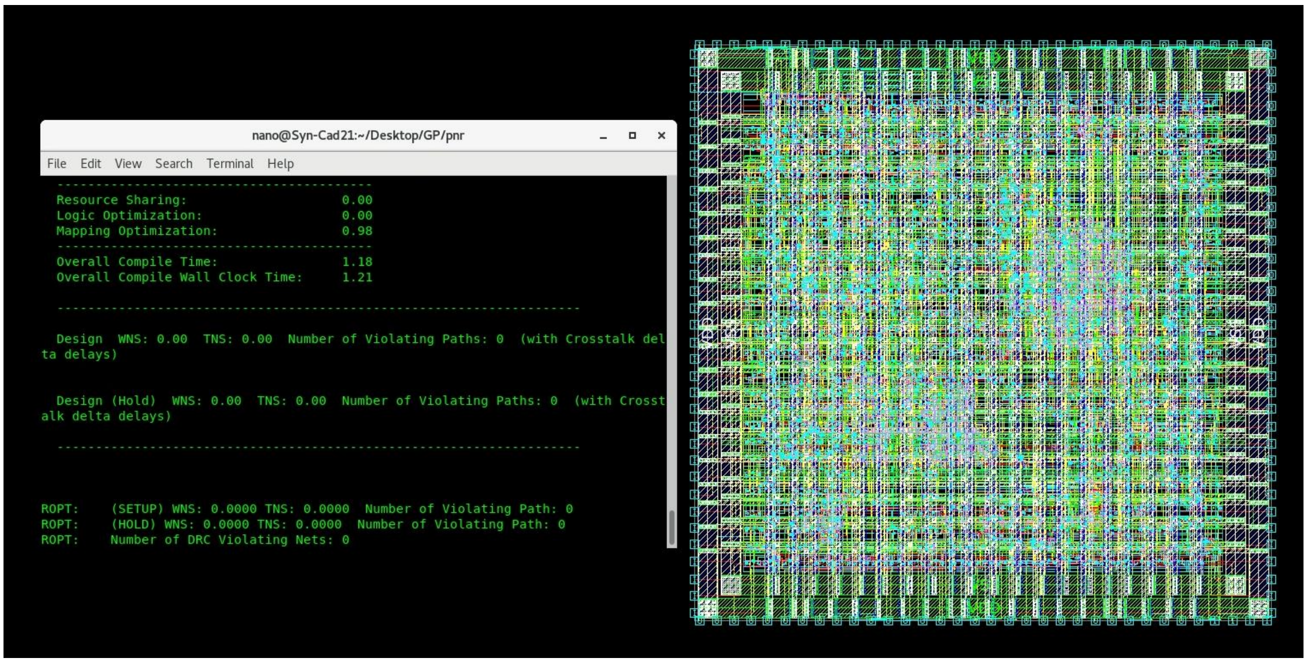
Congestion calculation:
- [✓ Sum of overflow for each layer]
- Total demand minus total supply

Bins: 9
From: 0 To: 6
Edges: ☑ Horizontal ☑ Vertical
☑ Text
☐ Current design only

- ☑ metal1 (11)
- ☑ metal2 (13)
- ☑ metal3 (15)
- ☑ metal4 (17)
- ☑ metal5 (19)
- ☑ metal6 (21)
- ☐ metal7 (23)
- ☐ metal8 (25)
- ☐ metal9 (27)
- ☐ metal10 (29)

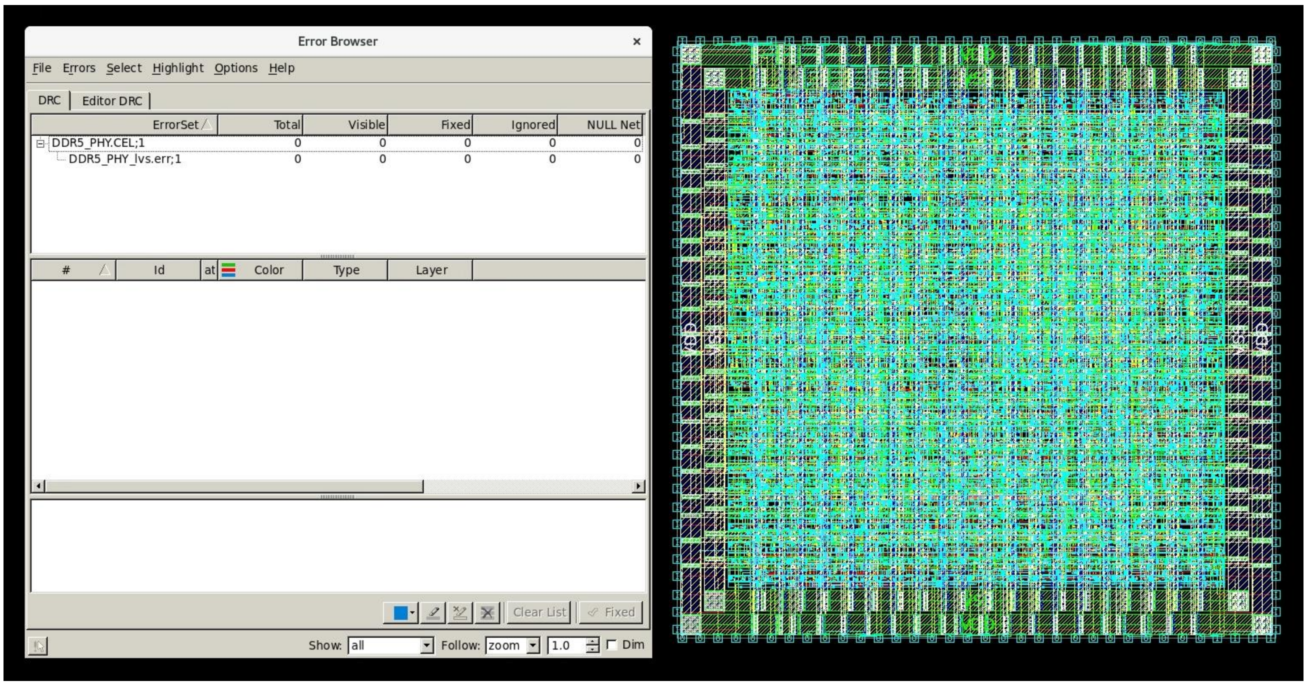
☑	7	🟥	0
☑	6	🟧	0
☑	5	🟨	0
☑	4	🟩	0
☑	3	🟩	1
☑	2	🟦	6
☑	1	🟦	33
☐	0	🟦	1505
☐	-16 .. -1	🟪	20087
☐	Blocked	🟪	0

Figure 101. Placement congestion map.


nano@Syn-Cad21:~/Desktop/GP/pnr

```
  -----------------------------------------
  Resource Sharing:                      0.00
  Logic Optimization:                    0.00
  Mapping Optimization:                  0.98
  -----------------------------------------
  Overall Compile Time:                  1.18
  Overall Compile Wall Clock Time:       1.21

  --------------------------------------------------------------------

  Design  WNS: 0.00  TNS: 0.00  Number of Violating Paths: 0  (with Crosstalk del
ta delays)


  Design (Hold)  WNS: 0.00  TNS: 0.00  Number of Violating Paths: 0  (with Crosst
alk delta delays)

  --------------------------------------------------------------------



ROPT:    (SETUP) WNS: 0.0000 TNS: 0.0000  Number of Violating Path: 0
ROPT:    (HOLD) WNS: 0.0000 TNS: 0.0000  Number of Violating Path: 0
ROPT:    Number of DRC Violating Nets: 0
```

Figure 102. Routing and violations.



ErrorSet	Total	Visible	Fixed	Ignored	NULL Net
DDR5_PHY.CEL;1	0	0	0	0	0
DDR5_PHY_lvs.err;1	0	0	0	0	0

#	Id	at	Color	Type	Layer

Figure 103. LVS check.

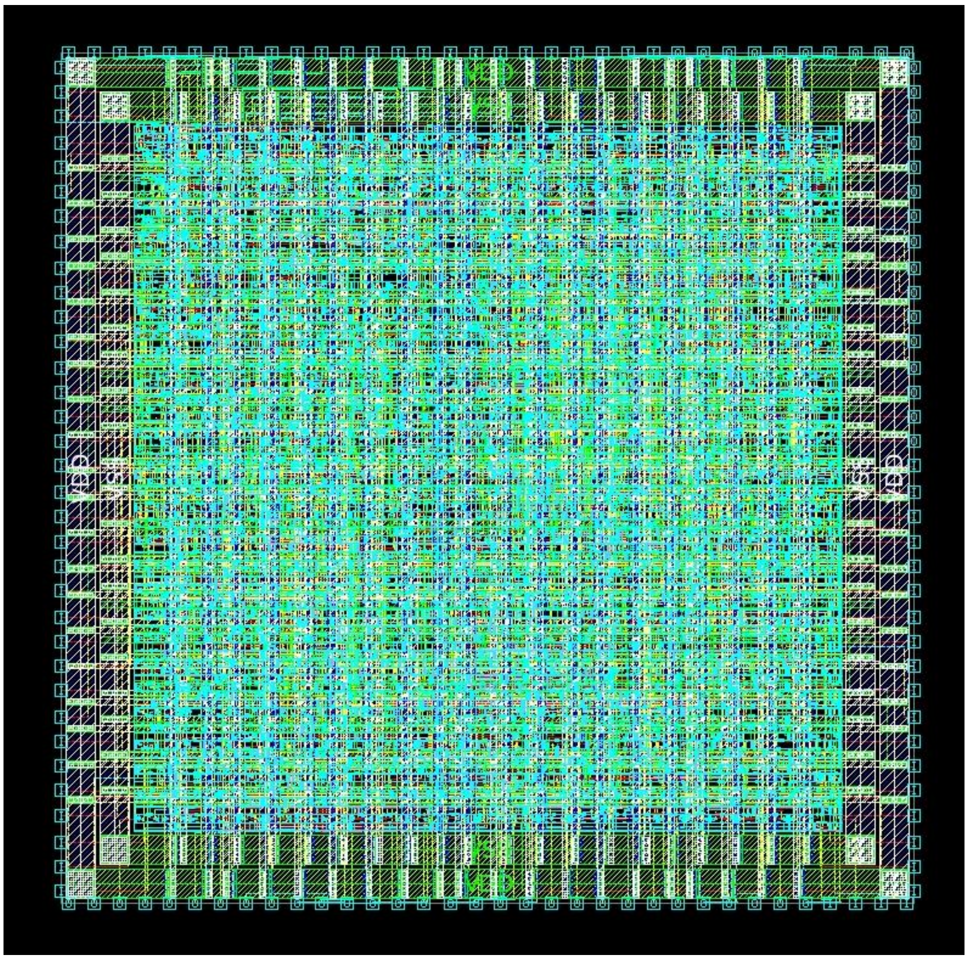
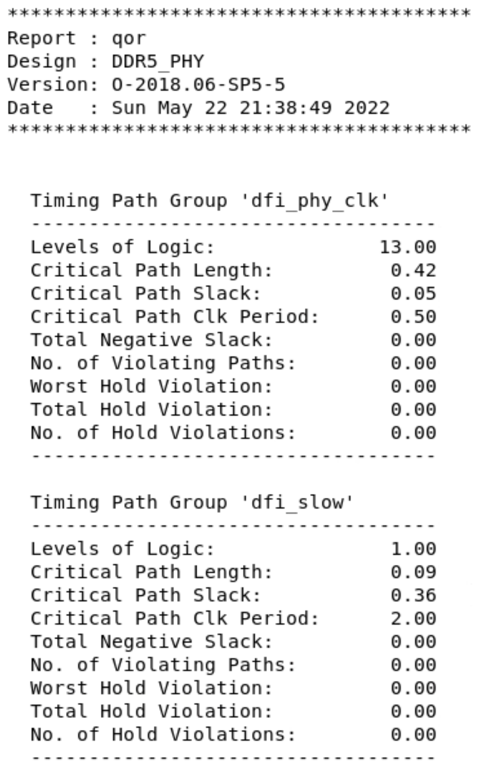
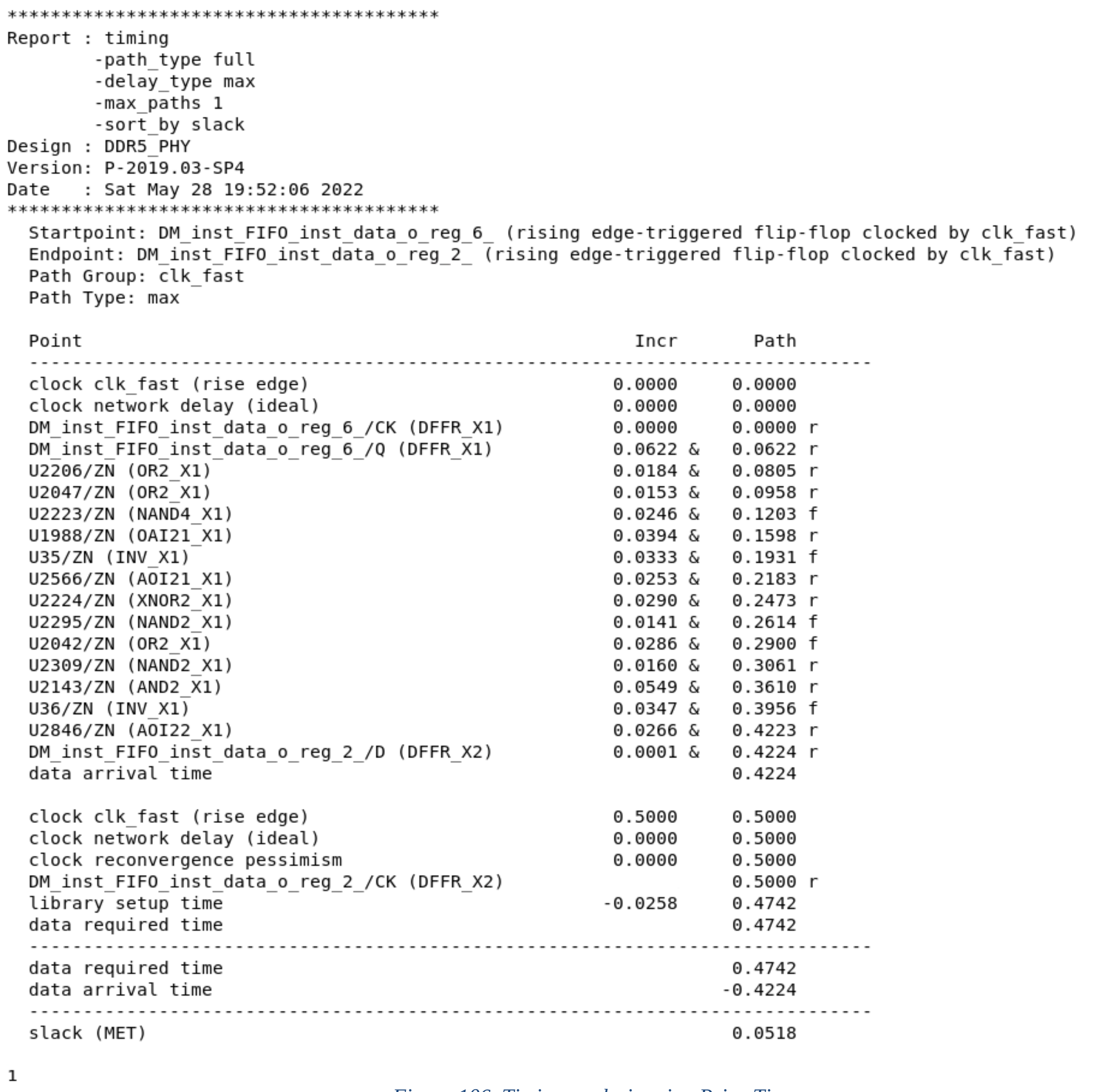


Figure 104. Final layout.



```
****************************************
Report : qor
Design : DDR5_PHY
Version: O-2018.06-SP5-5
Date   : Sun May 22 21:38:49 2022
****************************************


  Timing Path Group 'dfi_phy_clk'
  -----------------------------------
  Levels of Logic:              13.00
  Critical Path Length:          0.42
  Critical Path Slack:           0.05
  Critical Path Clk Period:      0.50
  Total Negative Slack:          0.00
  No. of Violating Paths:        0.00
  Worst Hold Violation:          0.00
  Total Hold Violation:          0.00
  No. of Hold Violations:        0.00
  -----------------------------------

  Timing Path Group 'dfi_slow'
  -----------------------------------
  Levels of Logic:               1.00
  Critical Path Length:          0.09
  Critical Path Slack:           0.36
  Critical Path Clk Period:      2.00
  Total Negative Slack:          0.00
  No. of Violating Paths:        0.00
  Worst Hold Violation:          0.00
  Total Hold Violation:          0.00
  No. of Hold Violations:        0.00
  -----------------------------------
```

Figure 105. Quality of results.



```
****************************************
Report : timing
        -path_type full
        -delay_type max
        -max_paths 1
        -sort_by slack
Design : DDR5_PHY
Version: P-2019.03-SP4
Date   : Sat May 28 19:52:06 2022
****************************************
  Startpoint: DM_inst_FIFO_inst_data_o_reg_6_ (rising edge-triggered flip-flop clocked by clk_fast)
  Endpoint: DM_inst_FIFO_inst_data_o_reg_2_ (rising edge-triggered flip-flop clocked by clk_fast)
  Path Group: clk_fast
  Path Type: max

  Point                                                    Incr       Path
  ---------------------------------------------------------------------------
  clock clk_fast (rise edge)                             0.0000     0.0000
  clock network delay (ideal)                            0.0000     0.0000
  DM_inst_FIFO_inst_data_o_reg_6_/CK (DFFR_X1)           0.0000     0.0000 r
  DM_inst_FIFO_inst_data_o_reg_6_/Q (DFFR_X1)            0.0622 &   0.0622 r
  U2206/ZN (OR2_X1)                                      0.0184 &   0.0805 r
  U2047/ZN (OR2_X1)                                      0.0153 &   0.0958 r
  U2223/ZN (NAND4_X1)                                    0.0246 &   0.1203 f
  U1988/ZN (OAI21_X1)                                    0.0394 &   0.1598 r
  U35/ZN (INV_X1)                                        0.0333 &   0.1931 f
  U2566/ZN (AOI21_X1)                                    0.0253 &   0.2183 r
  U2224/ZN (XNOR2_X1)                                    0.0290 &   0.2473 r
  U2295/ZN (NAND2_X1)                                    0.0141 &   0.2614 f
  U2042/ZN (OR2_X1)                                      0.0286 &   0.2900 f
  U2309/ZN (NAND2_X1)                                    0.0160 &   0.3061 r
  U2143/ZN (AND2_X1)                                     0.0549 &   0.3610 r
  U36/ZN (INV_X1)                                        0.0347 &   0.3956 f
  U2846/ZN (AOI22_X1)                                    0.0266 &   0.4223 r
  DM_inst_FIFO_inst_data_o_reg_2_/D (DFFR_X2)            0.0001 &   0.4224 r
  data arrival time                                                 0.4224

  clock clk_fast (rise edge)                             0.5000     0.5000
  clock network delay (ideal)                            0.0000     0.5000
  clock reconvergence pessimism                          0.0000     0.5000
  DM_inst_FIFO_inst_data_o_reg_2_/CK (DFFR_X2)                      0.5000 r
  library setup time                                    -0.0258     0.4742
  data required time                                                0.4742
  ---------------------------------------------------------------------------
  data required time                                                0.4742
  data arrival time                                                -0.4224
  ---------------------------------------------------------------------------
  slack (MET)                                                       0.0518

1
```

Figure 106. Timing analysis using PrimeTime.

8- Cost Analysis

8.1 The Cost

The cost of JEDEC JESD79 is 369$, the cost of Zynq UltraScale+ MPSoC ZCU106 is 100$. The standard is provided by Si-Vision Co. and the board is provided by Zewail City.

8.2 Manufacturability

Our design can be manufactured as an ASIC (Application Specific Integrated Circuit) by mapping it to a specific technology and requesting a tape-out. However, what we will do is to download it into an FPGA fabric and test it on silicon as FPGAs provide smaller time to market and lower cost.

8.3 Social and Economic Impact

Memories have been the bottle neck in almost every application that needs high speed comuations and the requirement for faster, lower in power, and larger memories have been increasing rapidly. With the introduction of the newest generation DDR5, every device will need the physical layer compatible with DDR5 such as mobile phones, PCs, Smart cars and smart TVs. All these applications will be faster leading to a more comfortable life.

9- Conclusion and Future Work

In conclusion we started the project by reviewing the latest development in DDR SDRAMs from generation to another, and we made a documentation about this literature review. Then, we summarized both the JEDEC JESD79-5A and DFI v5 standards and identified the key features to be included in out DDR5 PHY that will assure the compatibility between the MC and the DD5 SDRAM. We finished a timing diagram of the system that includes an activate along with a read command with a number of options that identified important blocks that we included in out block diagram. In our future work we will write RTLs to describe our PHY then we will start our behavioral simulation of a PHY to debug our system, after which we will be ready to implement the system on FPGA for testing and emulation.

References

[1] Jacob, B., Wang, D., & Ng, S. “Memory systems: cache, DRAM, disk,” 2010.

[2] “Micron enables pervasive, data-driven experiences,” *Micron*. [Online]. Available: https://www.micron.com/. [Accessed: 18-Jun-2022]

[3] Kim, Y.-H., Kim, H.-J., Choi, J., Ahn, M.-S., Lee, D., Cho, S.-H., Lee, J.-B. “A 16Gb Sub-1V 7.14Gb/s/pin LPDDR5 SDRAM Applying a Mosaic Architecture with a Short-Feedback 1-Tap DFE, an FSS Bus with Low-Level Swing and an Adaptively Controlled Body Biasing in a 3rd-Generation 10nm DRAM,” *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, 2021, doi:10.1109/isscc42613.2021.9366050

[4] “DDR5 SDRAM,” Micron. [Online]. Available: https://www.micron.com/products/dram/ddr5-sdram. [Accessed: 19-Jun-2022]

[5] A. T. Inc, “APACER industrial - the most reliable storage and memory for industries,” *What Sets DDR5 Memory Modules Apart - 7 Key Specification Differences of Industrial DDR5 RDIMM - News & Events - Apacer for Industrial – Leader in industrial SSD and DRAM module*, 14-Apr-2022. [Online]. Available: https://industrial.apacer.com/en-ww/NEWS/What-Sets-DDR5-Memory-Modules-Apart--7-Key-Specification-Differences-of-Industrial-DDR5-RDIMM. [Accessed: 19-Jun-2022]

[6] “DDR PHY Interface (DFI),” DFI 5.0 Specification, Cadence Design Systems, Inc, APRIL 27, 2018.

[7] “JEDEC STANDARD, JESD79-5A (Revision of JESD79-5, JULY 2020),” JEDEC SOLID STATE TECHNOLOGY ASSOCIATION, October 2021.

[8] “IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language,” in IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012), vol., no., pp.1-1315, 22 Feb. 2018, doi: 10.1109/IEEESTD.2018.8299595.

[9] J. L. Hennessy and D. A. Patterson, Computer Architecture: A quantitative approach, Sixth edition. Beijing Shi: Ji xie gong ye chu ban she, 2019.

[10] “DDR SDRAM Market: Global Industry Analysis and Forecast (2021-2027) Trends, Statistics, Dynamics, Segmentation by Memory Type, Processor, Verticals, and Region,”

[11] P. Jacob, Synchronous DRAM Architectures, Organizations, and Alternative Technologies. 2022.