Proceedings of CONF-MCEE 2026 Symposium: Advances in Sustainable Aviation and Aerospace Vehicle Automation
DOI: 10.54254/2755-2721/2026.KA29371

Design and Application of Data Recording Buffer Based on AXI Bus in System-On-Chip (SOC)

Runyang Chen

School of Physics and Electronic Engineering, Hanshan Normal University, Chaozhou, China
cnekhk@uest.edu.gr

Abstract. The integration of numerous heterogeneous Intellectual Property (IP) cores operating in different clock domains in complex System-on-Chip (SoC) architectures necessitates robust cross-clock domain communication. In this paper, we focus on the design and application of a data recording buffer based on the Advanced eXtensible Interface (AXI) bus to address associated data transfer challenges. The study provides an in-depth analysis of the buffer's evolutionary trajectory, compares key architectural designs, and examines its role in ensuring data integrity and enhancing system throughput. Future development trends and potential advancements for AXI-based buffering mechanisms are also discussed.

Keywords: AXI Bus, SOC, data recording buffer

1. Introduction

Owing to the increasing complexity of system-on-chip (SoC) architectures and the widespread availability of diverse intellectual property (IP) cores capable of performing specialized functions, SoC designers frequently integrate numerous IP cores into a single chip. As these cores often operate under different clock domains and perform heterogeneous tasks, reliable data transfer between them necessitates robust cross-clock domain communication. To address synchronization challenges and ensure data integrity, data buffering mechanisms—such as data recording buffers—are employed to temporarily store data during domain transitions. These buffers effectively enhance system throughput and minimize communication errors. This paper presents an in-depth analysis of the evolutionary trajectory of data recording buffers tailored for the Advanced eXtensible Interface (AXI) bus, compares key architectural designs, and discusses their future development trends and potential advancements. Summarizing and reviewing the commonly used techniques in the Buffer design is the main goal of this thesis. Among them, the AXI bus is a commonly used technology. Through AXI, the host can link multiple slaves, which enables the bus to connect up to ten different buffers at most. Meanwhile, based on the AXI system, the design of direct memory access (DMA) becomes more efficient [1]. DMA control enables direct high-speed data transfer between devices and memory, effectively enhancing data transmission efficiency. Moreover, DMA can also be used on different bus architectures, significantly enhancing the universality and compatibility of buffer design. With the advancement of technology, the CPU operates at a faster speed. During the transmission process, it needs to wait for instructions from the transmission equipment, and the CPU also needs to spend additional time coordinating the timing between data transmission devices and

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.KA29371

handling issues such as data transmission formats. Moreover, when a large amount of data needs to be transmitted, there may be situations of excessive load, increased energy consumption and latency. These problems limit the performance of the CPU and reduce the overall efficiency and reliability of the system. However, buffers that can perform DMA can effectively solve this problem [2]. Buffer typically uses RAM as its storage medium. However, when reading and writing in RAM, different read/write sequences are required, so it is necessary to switch between different modes of the AXI bus to adapt. Meanwhile, due to the limited size of a single RAM module, in the actual design of the SoC buffer, careful consideration needs to be given to how to store the overflow data [3]. Currently, one relatively efficient method for SoC design is to generate a powerful IP soft core by interconnecting multiple custom IP cores through the AXI4 bus, and then embed it into an FPGA chip to form a unique SoC system. Similarly, by using modular IP cores to build the soft core of the buffer, it is possible to quickly integrate the buffer into the designs of various SoCs [4]. Even better, the Block Random-Access Memory (BRAM) technology has matured. We can easily use BRAM to build the RAM required for our buffer design. The BRAM usage method is flexible, including simple pin control and bus read/write operations. It is compatible with common single-port, dual-port and block RAMs, and can be configured with various read/write modes [5].

Based on the above technology, the development of the buffer has become simpler and more efficient. This paper presents an in-depth analysis of the evolutionary trajectory of data recording buffers tailored for the AXI bus, compares key architectural designs, and discusses their future development trends and potential advancements.

2. Theoretical basis based on AXI

2.1. AXI bus

AXI) is an efficient, high-throughput, and low-latency bus protocol developed by Advanced RISC Machines Ltd (ARM) [1]. It constitutes a protocol within the Advanced Microcontroller Bus Architecture (AMBA) family, which is developed by Arm. AMBA is the main communication protocol in various SoCs and Field-Programmable Gate Arrays (FPGAs), and the AXI bus protocol was first proposed in the AMBA3.0 protocol and has become an important part of the AMBA protocol. Both the previously commonly used Advanced High-speed Performance Bus and Advanced Peripheral Bus are compatible with this Bus, and can support low-power memory access and out-of-order transactions. This strong compatibility makes it widely used in communication between various SoCs. The AXI bus protocol is structured around three primary components: master, slave, and infrastructure. The protocol employs a design that separates address, control, and data channels to enhance efficiency and parallelism. At the same time, AXI also supports outstanding transactions. Its duplex is extremely flexible; a device can be used as a master when data is transmitted, and can be used as a slave when configured [1]. In addition, the performance of the AXI bus can achieve up to 762 MB / s read and write throughput [2].

In the AXI bus, the basic unit of data transmission is called a burst. A burst consists of multiple transfers; the number of transfers is called the burst length, and the number of bytes contained in a transfer is called the burst width. At each handshake, the host declares the address of the burst to be transmitted, and an additional transfer is transmitted at the end of the burst to indicate that the burst has been transmitted. This is the basic transmission mode of the AXI bus. In addition, AXI bus represents three special transmission modes: outstanding mode, out-of-order mode, and interleaving mode. As mentioned above, the basic transmission mode of AXI needs to wait for the completion of all transfers and transmission of the previous burst to transmit the next burst. However, in the

DOI: 10.54254/2755-2721/2026.KA29371

outstanding mode, the AXI bus is allowed to output multiple addresses to the address bus at one time without waiting. The out-of-order mode allows multiple bursts corresponding to the same address to be transmitted at the same time without waiting for the burst transmission of the previous address to be completed. Interleaved transmission is to allow arbitrary exchange of data in transfer units. Lai Zhexiang and Huang Shizhen have pointed out that if the device can be compatible with these three special modes at the same time, the efficiency of data transmission will be greatly improved [3].

Compared with other bus protocols in the AMBA protocol, the biggest feature of the AXI bus protocol is that it can be compatible with the interfaces of AHB and APB, and separates the channels of reading and writing data, supporting more addressing methods. The AXI4 protocol is the latest AXI bus specification. The protocol was introduced by ARM in 2010, incorporating notable improvements over the previous AXI3 standard. It has made many upgrades to the AXI3 protocol. The most important upgrade is to improve the support for disorderly transmission. High-speed devices that allow post-initiation operations can take precedence over low-speed devices that initiate operations first to complete data transmission, which greatly reduces waiting time and system delay. Coupled with five separate read and write channels, its operating frequency and bandwidth are almost extreme [4].

2.2. Block Random-Access Memory (BRAM)

BRAM is the most commonly used customizable high-speed RAM resource within FPGAs, which has been used in the design of First-In-First-Out (FIFO) buffers in many fields, such as high-efficiency low-density decoding, optimization of asynchronous FIFO cycle delay characteristics, etc [5].. In order to reduce latency and improve system performance, using Static Random-Access Memory (SRAM) to configure BRAM is a good solution. SRAM is a kind of RAM. It uses transistors to store data. Due to its non-volatile nature under constant power, SRAM enables persistent data storage. This characteristic simplifies buffer management procedures and thereby enhances operational speed. Although there are also schemes to replace SRAM with Non-Volatile Memory (NVM), the performance of NVM is not as good as SRAM in terms of reading speed and energy consumption. In addition, in the past five years of research, RATHI N and others have devoted themselves to reducing the power consumption of SRAM cells. They have compared the power consumption, delay, area, energy, and stability parameters of different RAMs such as 6T, 7T, 8T, 9T, and 10T, and believe that 6T and 7T SRAMs should be preferred for applications that require lower power consumption and faster read speeds [6].

2.3. FIFO

FIFO is a commonly used read and write strategy in the design of a cache. Li Yao pointed out that in the practical application of FPGA, the design and implementation of FIFO has a direct impact on the stability and efficiency of data transmission. In the paper published by Li Yao in July 2025, many feasible solutions to the problem of FIFO empty and full have been summarized [7].

The FIFO scheme is divided into asynchronous FIFO and synchronous FIFO, and the main difficulty in the design of asynchronous FIFO is how to prevent the trigger from being metastable and how to judge the empty state. Among them, the metastability is one of the urgent problems to be solved, it refers to the signal to maintain a period of time before the arrival of the clock edge or to maintain a period of time after the arrival of the clock edge, during this period of time on the trigger signal if it does not reach the trigger value, the signal will be in a state between 0 and 1, known as

DOI: 10.54254/2755-2721/2026.KA29371

Metastability. The empty state is read by the pointer. When the asynchronous FIFO is empty, the address pointer is read and written [7].

In the same position, this situation may be caused by the following two cases. First, after the system is initialized, the read and write pointers are set to the starting position. Second, the system completes the reading operation of the last data item in the asynchronous FIFO, which causes the reading pointer to catch up with the writing pointer. On the contrary, when the asynchronous FIFO reaches a full state, although the read and write pointers also point to the same storage unit, this only occurs when the external system continuously writes data to the asynchronous FIFO until the write pointer coincides with the read pointer. Because both of these states appear when the read pointer and the write pointer point to the same position in RAM, the key to determining these two states is to distinguish whether the reading pointer overtakes the writing pointer or the write pointer is ahead of the read pointer [7].

In the process of buffer design, high-speed asynchronous FIFOs are often used. At this time, the handshake synchronization efficiency of the FIFO will largely affect the transmission speed. The use of Gray code pointers solves this problem. Gray code, or typical binary Gray code, is a set of integers where any two adjacent codes differ by only one binary bit. Gray code encoding technology is reliable and can minimize errors. It uses absolute encoding, and the code has single-step self-complement, reflection, and cyclic characteristics. When randomly accessing numbers, its cyclic and single-step characteristics eliminate the possibility of catastrophic errors. These characteristics make inversion very convenient. A Gray code counter is a counter used to transfer count results between the clock domains of asynchronous FIFOs. In some cases, the counter needs several milliseconds to take the next step, or it needs to count a large amount of data before making a judgment. Therefore, the counter's range needs to be larger. If a high operating frequency is required, the Gray code counter will generate glitches, causing deviations. To accelerate multi-bit Gray code counters and make their operation stable, we divide the large counter into a combination of several small counters [8].

3. Performance analysis and discussion

3.1. Performance indicators

Throughput is an important performance indicator that represents how much data a Buffer can successfully deliver per unit of time. If the burst length of a single burst is larger, the number of handshakes will be fewer, the bus will be more utilized, and consequently, the throughput will be higher. The greater the throughput, the better the high-speed transmission performance of the buffer. Since the AXI bus is often used in high-speed communication, the throughput of the buffer is a very important performance indicator. In complex SoCs, throughput is restricted by various factors, so the throughput size is limited. A system mainly comes from aspects such as data generation rate and system performance [9]; In addition to considering general performance factors, another system is also subject to strict timing constraints of real-time systems [10]. These different constraints lead to differences in buffer throughput in terms of design goals and implementation methods.

3.2. Configurability

Both systems reflect the good compatibility of AXI bus and Direct Memory Access (DMA). Numerical experiments demonstrate that adopting a multi-objective optimization approach reduces energy consumption by 15%-20% compared to single-objective throughput maximization, with a

DOI: 10.54254/2755-2721/2026.KA29371

minimal throughput loss of merely 3%-5% [1,9]. This result is partly attributed to the dynamic Buffer configuration capability of DMA, which allows the system to achieve a trade-off between performance and energy consumption. In a player or camera scenario, DMA needs to continuously move large chunks of data from memory to the decoder/encoder. The Buffer parameters (such as burst length and interval) of DMA can be configured to optimize the data flow rate, reduce the playback lag or recording delay, and reduce the increase in energy consumption caused by Buffer overflow. When the network interface receives a large number of packets, the DMA can write the packets directly to the memory or hardware buffer. By dynamically configuring Buffer size and DMA transmission frequency, the network throughput can be significantly improved, while avoiding packet loss and retransmission energy consumption caused by insufficient Buffer. All these show that Buffer based on AXI bus has high configurability when used with DMA.

3.3. Power consumption comparison

Energy consumption is also an important performance indicator for evaluating buffers. Since SoCs are frequently used in various mobile devices, it is necessary to reduce their energy consumption during use. Using random access buffer transaction management can reduce the power consumption of the buffer to a certain extent. In addition, appropriately lowering the clock frequency during idle time is also an option. Y. Alaouchiche et al. described a method that simultaneously optimizes the two conflicting goals of production throughput and energy consumption while meeting the total storage capacity constraints of the production system [11]. They modeled the problem as a dual-objective optimization problem, maximizing production throughput and minimizing energy consumption. Three processing methods are mentioned in the text, namely the Weighted Sum Method: assigning weights to two objectives and transforming the multi-objective problem into a single-objective optimization. By adjusting the weights, different trade-off schemes (such as high throughput priority or low energy consumption priority) are generated. Epsilon-Constraint Method: Set one of the objectives (such as energy consumption) as a constraint condition (such as energy consumption $\leq \varepsilon$), and optimize the other objective (throughput). Explore the Pareto frontier by adjusting the ε value. And NSGA-II: Evolutionary algorithms are adopted to simultaneously optimize two objectives, generating a Pareto optimal solution set. Each solution represents a buffer allocation scheme, achieving a balance between throughput and energy consumption. At the same time, they also mentioned the Pareto optimal solution and trade-offs, pointing out that through numerical experiments, the algorithm generates a set of non-dominated solutions (Pareto frontiers), and each solution corresponds to a buffer allocation strategy. Choose the solution based on actual needs. If the production line is sensitive to latency (such as just-in-time manufacturing), a high-throughput solution can be selected. If the energy cost is high, a low-energy consumption solution can be chosen [11,12].

4. Conclusion

In summary, Buffer designed based on AXI bus has the advantages of strong compatibility, large throughput and high efficiency. At the same time, its excellent configurability makes it suitable for use in the SoC design of different neighborhoods, such as player, camera mobile device, server, network router and other high-performance embedded systems. In addition, it can also be used in the interface between the processor and memory to reduce memory access conflicts and improve system performance. The burst transmission and unaligned data transmission characteristics of AXI Buffer make multimedia data transmission efficient in video codec, image processing, audio processing and

DOI: 10.54254/2755-2721/2026.KA29371

other multimedia processing. For example, in a video codec, AXI Buffer can optimize the transmission of video frame data, reduce codec delay, and improve real-time performance. AXI Buffer can also be developed into a variety of IP cores because of its flexibility.

Research on AXI Buffer can promote the development of many industries, including but not limited to enhancing the fineness of media product signals. Accelerate the SoC design for smart facilities such as automobiles and smart homes; Accelerate the development of SoCs such as manufacturing industrial terminal devices that are more convenient for network control. It can also effectively reduce the energy consumption during SoC communication and optimize the battery life of mobile smart devices such as smartphones and tablet computers. In addition, in the SoC of new communication technologies, Buffers also play a very important role. For instance, in the confirmation mode of the Radio Link Control (RLC) layer, this mode uses a separate buffer for transmission and retransmission, while taking into account the behavior of the physical channel. The Buffer plays a crucial role in data temporary storage and retransmission management here, ensuring reliable data transmission over unreliable wireless channels.

At present, the Buffer design based on AXI is exceedingly mature. The use of Gray code, DMA, BRAM, and FIFO technology makes the use of the buffer more stable and efficient. At present, AXI bus is one of the most commonly used communication protocols in SoC design, which makes the Buffer designed based on AXI bus have high compatibility. The high configurability of the AXI protocol also allows us to design flexible Buffer IP cores. This flexibility will make the subsequent development of various SoCs easier and easier, reduce the cost of SoC development, improve the stability and economy of SoC design, and be very helpful for the further development of SoC design. Meanwhile, developing a buffer based on AXI can also serve as an excellent research topic for students to learn and understand the usage and design of technologies such as AXI bus and FIFO, making SoC design easier to learn.

References

[1] Pu, J. Li, G. (2012) Design and implementation of DMA controller based on AXI bus. Design and implementation of DMA controller based on AXI bus, 24(02), 174-177.
[2] Ren, X. Yang, Y. Wang, L. Yan, D. Zhang, H. Li, F. Li, L. (2025). Research Progress in DMA Controllers Based on Different Buss Protocols, 50(02), 117-126.
[3] Lai, Z. Huang, S. (2025) Design and Simulation of an Efficient DMA Controller Based on the Interleaved Transmission Mode in AXI, 33(13), 65-68.
[4] Mo, J. (2018) Design and verification of IP cores interconnection model based on AXI4
[5] Yi, M. Zhang, J. Lu, Y. Liang, H. Ma, L. A Test Vector CODEC Scheme Based on BRAM-Segmented Synchronous Table Lookup, 47(09), 1-11.
[6] Chen, M. Zhang, J. Wang, Y. Mai, M. Qiao, F. Design of Low-Power On-Chip Cache for Visual Perception Systems on the Edge, 47(09), 1-10.
[7] Li, Y. (2025) Research on the Implementation Scheme of Asynchronous FIFO, 26(07), 143-145.
[8] Z. Hao, L. Liu and B. Tian, (2023) The Principle and Applications of Asynchronous FIFO, 2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, pp. 277-279
[9] A. G. Shenoy, A. S, B. P L, A. Sangeeth, C. R. Jose and A. D. Pai, (2024) High Speed Data Transfer from External Interface to Memory using FIFO and AXI DMA, 2024 Global Conference on Communications and Information Technologies (GCCIT), BANGALORE, India, pp. 1-7
[10] Z. Jiang, N. Audsley, D. Shill, K. Yang, N. Fisher and Z. Dong, (2021) Brief Industry Paper: AXI-InterconnectRT: Towards a Real-Time AXI-Interconnect for System-on-Chips, 2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS), Nashville, TN, USA, pp. 437-440
[11] Y, Alaouchiche. Y, Ouazene. F, Yalaoui. (2022) Multi-Objective Optimization of Energy-Efficient Buffer Allocation Problem for Non-Homogeneous Unreliable Production Lines. 10, 3320-333

DOI: 10.54254/2755-2721/2026.KA29371

[12] A. Karnam, J. John, K. C. Joshi, G. Exarchakos, S. H. de Groot and I. Niemegeers, (2025) Reliability Modeling for Beyond-5G Mission Critical Networks Using Effective Capacity, 2025 IEEE Wireless Communications and Networking Conference (WCNC), Milan, Italy, pp. 1-6