

Synergistic Integration of Asynchronous FIFO and FIR Filters: A Novel Framework for Advanced Digital Signal Processing

Chengtian Wang *

Electronics and Information Engineering, Nanjing Normal University, Nanjing, 210000, China

* Corresponding Author Email: 07200602@njnu.edu.cn

Abstract. This study rigorously investigates the synergistic integration of Asynchronous FIFO (First-In, First-Out) with FIR (Finite Impulse Response) filters, proposing an advanced framework for digital signal processing. It commences by elucidating the cardinal significance of Asynchronous FIFO and FIR filters within the modern landscape of digital signal processing. The architecture of the Asynchronous FIFO is given primacy, emphasizing its avant-garde address management facilitated by the incorporation of a "toggle" register in the empty module. Additionally, the manuscript offers a comprehensive analysis of assorted methodologies for embedding FIR filters into the Asynchronous FIFO. The preferred method involves the infusion of DSP algorithms into the FIFO for precise simulation and modeling. A salient feature of this research is the introduction of a novel technique that utilizes a moving average filter algorithm to refine data granularity. In conclusion, the research accentuates the expansive applicability of this amalgamated methodology, with potential usages in realms such as audio refinement, video noise diminution, and biomedical signal processing, underscoring its pivotal role across varied sectors of digital signal processing.

Keywords: FIR Filter; Asynchronous FIFO; DSP algorithm; Signal processing.

1. Introduction

In the field of modern digital signal processing, Finite Impulse Response (FIR) filters are a widely used type of filter in applications such as signal processing, communication systems, audio processing, and more [1]. Asynchronous FIFO is a memory component designed to buffer and synchronize data in situations where data production and consumption rates differ. It addresses the challenges of asynchronous data transfer and is extensively used in digital systems [2].

This paper aims to explore how to harness the potential of asynchronous FIFO and FIR filters by combining them to create a new digital signal processing solution. Specifically, this paper introduces the overall architecture of the asynchronous FIFO, conducts simulation, and introduces program innovations. It delves into the functionality and performance of asynchronous FIFO, compares three methods of FIR filter implementation, and discusses their advantages and disadvantages. Finally, the paper implements one of the methods to validate its effectiveness.

The significance of the proposed solution lies in the combination of asynchronous FIFO with FIR filters, providing a novel digital signal processing solution for applications in audio and video processing. In audio processing, this solution can be used for audio noise reduction, enhancing audio quality, audio equalization, and improving audio sound quality and effects. In video processing, the filter can be employed to remove noise from videos, reduce or eliminate motion blur, and enhance image and video quality. Additionally, this solution has potential applications in the field of medical signal processing, where it can be used to remove noise such as power interference and baseline drift, extract clear biomedical signals, and aid in accurate medical diagnosis. Organization of the Text

2. Theoretical Foundations

2.1. Asynchronous FIFO

FIFO, which stands for First in First Out, is a type of data storage and buffer that finds widespread applications in the digital circuit and data transmission domains. Unlike typical memory, FIFO has unique characteristics in its structure. It doesn't require external read and write addresses to control

data access; instead, it uses an internal counter to automatically increment addresses, enabling orderly data read and write following the "first in, first out" principle. This means that FIFO does not support random read and write operations and only allows reading and writing data in the order it enters. The inner structure of FIFO is shown in figure 1 [3].

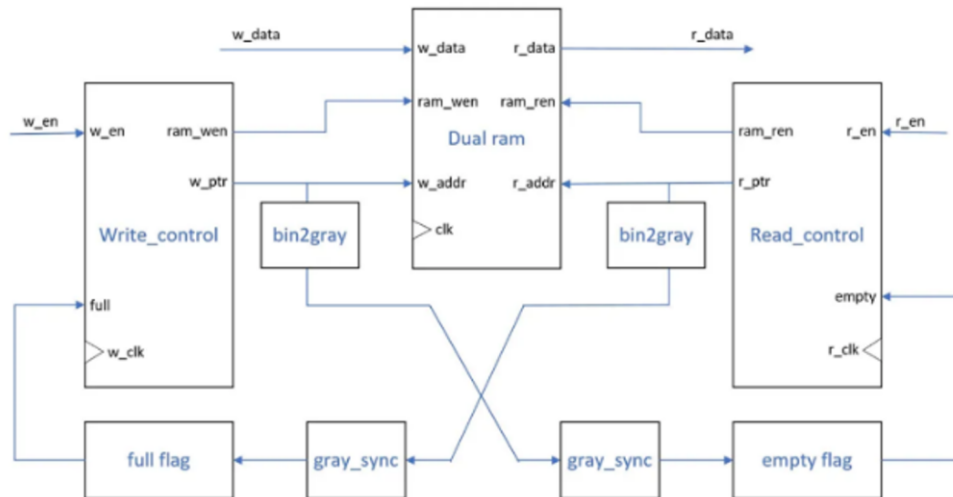


Fig 1. The inner structure of FIFO [3].

Asynchronous FIFO employs different clock signals for read and write operations to address the needs of asynchronous data transfer. The design and application of asynchronous FIFO primarily serve two critical purposes: Firstly, asynchronous FIFO enables data to be transferred between different clock domains. In digital systems, there can be issues with asynchronous clock frequencies and phase mismatches between clock domains. Asynchronous FIFO effectively facilitates the transfer of data from one clock domain to another, ensuring reliable data transmission and preventing data loss or errors due to clock asynchrony. Secondly, asynchronous FIFO supports the interconnection of data interfaces with varying data widths. This means that asynchronous FIFO can be used to connect modules with different data bit widths, making data transfer between them more flexible and compatible [4]. This is highly valuable in complex digital systems where different modules often have inconsistent data widths. Asynchronous FIFO acts as a bridge for data width conversion, ensuring proper data transmission.

In summary, FIFO and asynchronous FIFO are essential components in digital circuit design, providing reliable solutions for orderly data storage and asynchronous data transmission.

2.2. FIR Filters

Field-Programmable Gate Arrays (FPGAs) are endowed with structured internal logic matrices and a profusion of interconnection resources, rendering them particularly apt for intricate digital signal processing endeavors. In contrast to ubiquitous DSP chips, which predominantly operate based on serial computations, FPGAs exhibit enhanced parallelism and modularity. Through the adept application of rapid multiplication-accumulation algorithms within FPGAs, one can engineer high-velocity FIR digital filters [5].

Prior to the ingress of data into an FIR filter, it necessitates two sequential preprocessing stages: Initially, the signal must be subjected to an analog-to-digital transformation via a designated A/D apparatus, transmuting the continuous analog waveform into its discrete digital counterpart. To ascertain that the subsequent signal processing remains devoid of distortions, the sampling rate of said signal ought to comply with the Nyquist-Shannon sampling criterion. This typically mandates a sampling frequency approximately 4-5 times the zenith of the signal's inherent frequency [6]. A/D converters, especially those employing swift successive approximation techniques, are frequently favored. Irrespective of whether an FIR filter is crafted utilizing the multiply-accumulate paradigm or a distributive methodology, the resultant output invariably manifests as a consecutive data

sequence. To render this output perceptually coherent, it necessitates a subsequent digital-to-analog transmutation [7]. Thus, the terminal output from an FPGA-implemented FIR filter should be integrally interfaced with a D/A module.

3. Implementation of Asynchronous FIFO

3.1. Overall Architecture

The overall architecture of the asynchronous FIFO discussed in this paper is relatively clear and consists of seven main components: memory module, r2w module, w2r module, empty module, full module, fifo_tb module, and the fifo module.

The memory module is responsible for implementing the data storage function of the FIFO. The r2w module and w2r module can track the read pointer and update the write pointer based on clock signals and reset signals. The empty module and full module are capable of detecting the empty and full states of the asynchronous FIFO and managing the read operations. The fifo_tb module serves as the testbench module, responsible for simulating clock signals, generating data, and controlling the FIFO. The most crucial component is the fifo module, which serves as the core of the asynchronous FIFO, integrating the other modules to achieve the complete functionality of the FIFO. The architecture of the asynchronous FIFO in this paper is illustrated in the following figure 2.

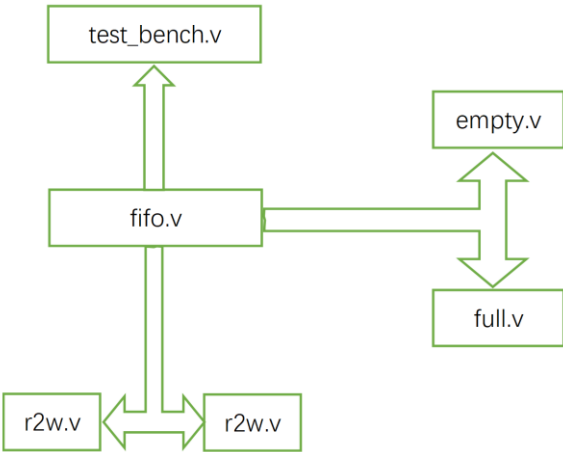


Fig 2. The architecture of the asynchronous FIFO in this paper (Photo/Picture credit: Original).

3.2. Simulation and modeling

The simulation and modeling of the asynchronous FIFO were conducted using the software "Vivado." The simulation results are displayed in the following figure 3.

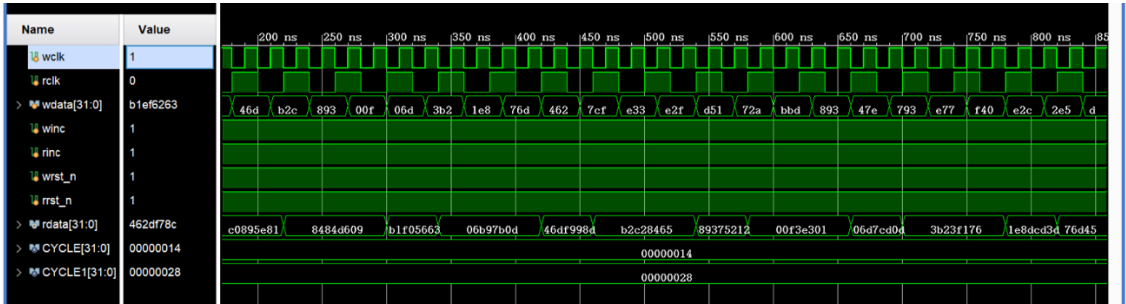


Fig 3. The simulation results (Photo/Picture credit: Original).

In the context of the simulation, the roles and nuances of each parameter can be elucidated as follows:

wclk: This represents the Write Clock Signal. Its primary function is to delineate the chronology of write operations, demarcating the precise temporal junctures when data ought to be inscribed.

rclk: Analogously, the Read Clock Signal, denoted as rclk, orchestrates the timing for read operations, indicating the epochs when data retrieval should ensue.

wdata: The Write Data Signal conveys the specific data intended for transcription into the asynchronous FIFO. Within the confines of this simulation, the wdata signal embodies the discrete data values designated for storage in the FIFO.

winc: As a Write Control Signal, winc exercises dominion over write functionalities. The actuation of the winc signal within our simulated environment metaphorically replicates the transcription of data into the FIFO.

rinc: The Read Control Signal, or rinc, governs read processes. Activating the rinc signal within the simulation analogously represents the data extraction procedure from the FIFO.

wrst_n: This parameter signifies the Write Reset Signal, characterized by its low-level logic association in the asynchronous FIFO's context. The actuation of wrst_n during the simulation can be interpreted as an initiation or rejuvenation of write functionalities.

rrst_n: Correlatively, the Read Reset Signal, symbolized as rrst_n, is affiliated with the low-level logic of the asynchronous FIFO. Within the simulation spectrum, the activation of the rrst_n signal orchestrates the initialization or recalibration of read mechanisms.

rdata: The Read Data Signal exemplifies the data being extricated from the asynchronous FIFO. In the simulated milieu, rdata mirrors the data values that have been procured from the FIFO.

CYCLE: This intrinsic parameter delineates the span of a singular clock cycle. Predominantly, CYCLE is appropriated for the semi-period of the write clock signal, facilitating the emulation of its periodic oscillations.

CYCLE1: Similarly, CYCLE1 stands as an inherent parameter demarcating the duration of a singular clock cycle. This parameter is specifically allocated for the semi-period of the read clock signal, aiding in the simulation of its rhythmic alterations.

3.3. Program innovation

In the empty and full modules, when reading data, it is necessary to retrieve the next data item from the FIFO, which involves address management [8]. This paper chooses to add a register called "toggle" to implement an alternating method for address management. Taking the empty module as an example, the innovative aspect of the program is explained in figure 4.

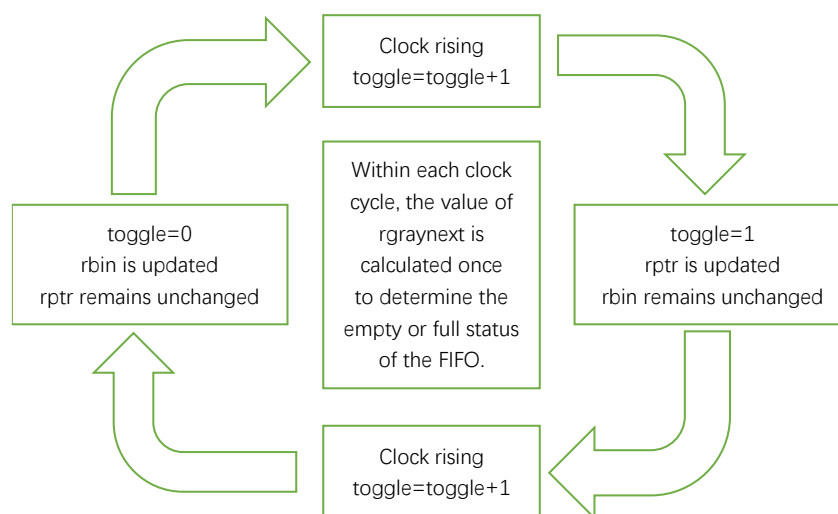


Fig 4. The innovative aspect of the program (Photo/Picture credit: Original).

As is mentioned above, the primary purpose of the 'toggle' register is to switch the update mechanism for the read address. It determines whether, in each clock cycle, the binary representation 'rbin' or the calculation of the next read address value should be updated. This alternating update mechanism is crucial for the proper functioning of the FIFO, ensuring the correct updating of addresses.

The 'toggle' register is a single-bit register with only two states: 0 and 1. When the asynchronous reset signal 'rrst_n' is low, the 'toggle' register is reset to 0, ensuring a clear initial value in the reset state. On each rising edge of the clock, the next action is determined based on the current state of the 'toggle' register. If 'toggle' is 0, the read address 'rbin' should be updated to the next address value, and the read pointer 'rptr' is set to the next gray code address value 'rgraynext.' Similarly, if 'toggle' is 1, the same update operation is performed.

Currently, the two states of the 'toggle' register in this article do not have a direct impact on the basic functionality of the asynchronous FIFO. However, this register opens up more possibilities for future research and development. For instance, it can be used to switch to real-time filtering modes, allowing dynamic adjustment of filters to accommodate signals with different spectral characteristics. It can also facilitate real-time adjustments to trigger parameters, such as cutoff frequencies, filter coefficients, or gain adjustments. Furthermore, when combined with multiple channels, the 'toggle' register can support multi-channel real-time filtering, allowing each channel to independently switch filter modes or parameters.

4. Implementation of FIR Filters

4.1. Feasibility Study

In the exploration of implementing FIR filtering through asynchronous FIFO, this chapter focuses on three possible methods, each with its unique advantages and disadvantages when applied to specific engineering projects. Therefore, this chapter will delve into the feasibility of these methods to better understand their characteristics.

4.1.1. Adding DSP Algorithms Inside the Asynchronous FIFO

The advantage of this approach lies in its compact design. By embedding the filter directly within the FIFO, it reduces the consumption of hardware resources, making it suitable for resource-constrained systems [9]. Additionally, its control logic is relatively simple because filter operations are synchronized with FIFO read and write operations, reducing control complexity.

However, this approach also has its disadvantages, most notably the introduction of significant delay since filter operations need to be performed before data enters or leaves the FIFO. Furthermore, changing the filter type or parameters would require modifications to the hardware design of the FIFO, which limits flexibility.

4.1.2. Adding DSP Algorithms at the Output Ports

The advantage of this method is that it doesn't require modifications to the hardware design of the FIFO. Different filters can be added at the output ports to meet specific requirements, offering a degree of flexibility [10].

The main drawback of this approach is the increased delay, for the same reasons mentioned earlier. Additionally, to enable filters to operate at different rates, this method requires additional data buffering, which adds complexity to the system.

4.1.3. Using Two FIFOs and Adding DSP Algorithms in the Second FIFO

This method is more flexible as it separates filter operations from the FIFO, allowing for greater control over filter behavior, including dynamic switching of filter type or parameters. It is relatively easy to maintain because buffering and filtering functions are separated, making it easier to identify the source of faults when they occur [11].

The disadvantage of this method is primarily on the hardware side. It requires additional hardware resources to implement two FIFOs and necessitates ensuring data synchronization between the two FIFOs, which increases system complexity and hardware resource consumption.

4.2. Simulation and modeling

In this experiment, the paper adopted the approach of "adding DSP algorithm internally to the asynchronous FIFO" to implement a filter. Here, a simulation of a moving average filter was carried out using the algorithm shown in figure 5.

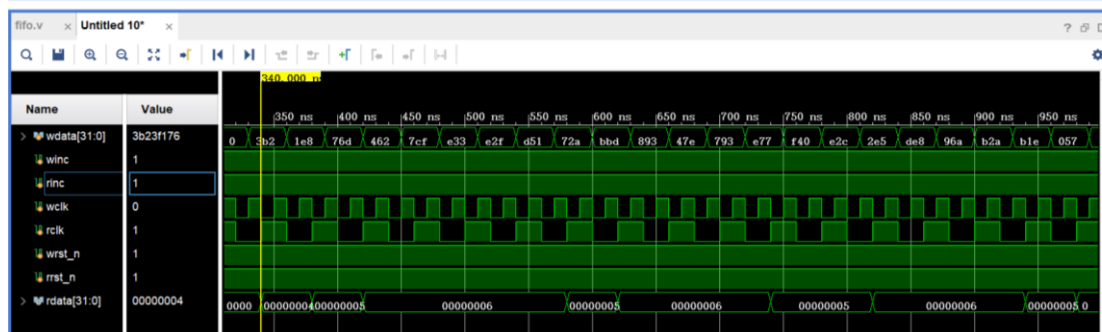


Fig 5. Simulation result of a moving average filter (Photo/Picture credit: Original).

The program in this paper utilizes the moving average filtering algorithm to smoothen the input data stream. Moving average filtering can reduce high-frequency noise and fluctuations in the data, resulting in a smoother output. A brief explanation of how the program implements the moving average filtering algorithm is as follows:

First, a register is added to store the moving average value, initialized to 0. Then, the output signal is assigned to `rdata`, which is used to store the data after smoothing.

As shown in the figure above, the register's value is updated on the rising edge of the clock signal. If the `rst_n` signal is 0, indicating that the reset signal is at a low level, the register is reset to 0, restarting the moving average calculation. If the `rempty` signal is 0, indicating that there is available data in the FIFO, the moving average algorithm is executed. Finally, the assignment statement is used to assign the value to `rdata`, completing the simulation.

In summary, this program applies the moving average filtering algorithm to smoothen the input data, reducing noise and fluctuations, and the smoothed data is output as `rdata`. The effectiveness of the algorithm depends on the `ASIZE` parameter, which defines the window size of the moving average. A larger window size results in a more noticeable smoothing effect but slower response.

5. Conclusion

The central focus of this investigation pertained to the theme, "Implementation of FIR Filters via Asynchronous FIFO." The study commenced with the successful orchestration and simulation of an asynchronous FIFO, employing the Verilog hardware description language within the Vivado environment. A novel register architecture was innovatively integrated, not merely fulfilling the preliminary objectives but also establishing a foundational scaffold for potential augmented functionalities in subsequent research undertakings.

Subsequently, the research delineated three prospective methodologies for the implementation of FIR filters within the asynchronous FIFO paradigm. Upon meticulous consideration of parameters such as intricacy of implementation and temporal investments, the methodology emphasizing the "Incorporation of DSP Algorithms within the Asynchronous FIFO" was selected for preliminary deployment. From a computational standpoint, this methodology adeptly instantiated the FIR filter. Nonetheless, given the absence of tangible demonstrations and juxtapositions against alternative methodologies, the exhaustive actualization of the FIR filter remained elusive within this research endeavor. Exploratory discussions also touched upon prospective applications, encapsulating domains like auditory signal analysis, visual data processing, and biomedical signal interpretation.

Conclusively, while the study efficaciously simulated the core theme, the absence of practical manifestations intimates potential gaps or oversights, necessitating further empirical investigations in the future.

References

- [1] Halak B, Chiu H C. Modified Micropipeline Architecture for Synthesizable Asynchronous FIR Filter Design. arXiv preprint arXiv:1603.04627, 2016.
- [2] Chen Y, Tsividis Y. Signal processing in continuous time using asynchronous techniques. 2017 51st Asilomar Conference on Signals, Systems, and Computers. IEEE, 2017: 1605-1609.
- [3] Xu G S. Design of Reconfigurable FIR Filter System Based on FPGA. Advanced Materials Research, 2012, 433: 4669-4674.
- [4] Ghazanfari L S, Airoldi R, Nurmi J, et al. Design of a reconfigurable multi-core architecture for streaming applications with a case study on performance evaluation of FIR-filters. Proceedings of the 2013 9th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME). IEEE, 2013: 237-240.
- [5] Murali A, Kakarla H K, Anitha Priyadarshini G M. Improved design debugging architecture using low power serial communication protocols for signal processing applications. International Journal of Speech Technology, 2021, 24: 291-302.
- [6] Chen Y, Zhang X, Lian Y, et al. A continuous-time digital IIR filter with signal-derived timing and fully agile power consumption. IEEE Journal of Solid-State Circuits, 2017, 53(2): 418-430.
- [7] Chen M, Han J, Fang D, et al. An ultra-low-power and area-efficient baseband processor for WBAN transmitter. 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference. IEEE, 2013: 1-4.
- [8] Jinding G, Yubao H, Long S. Design of high-speed sampling and adaptive filtering system. 2011 Fourth International Conference on Intelligent Computation Technology and Automation. IEEE, 2011, 2: 506-508.
- [9] Marr B, Karl J, Lewins L, et al. An Asynchronous Dataflow Signal Processing Architecture to Minimize Energy per Op. 2013 IEEE 19th International Symposium on Asynchronous Circuits and Systems. IEEE, 2013: 50-57.
- [10] Chen Y, Tsividis Y. Signal processing in continuous time using asynchronous techniques. 2017 51st Asilomar Conference on Signals, Systems, and Computers. IEEE, 2017: 1605-1609.
- [11] Chang I J, Park S P, Roy K. Exploring asynchronous design techniques for process-tolerant and energy-efficient subthreshold operation. IEEE Journal of Solid-State Circuits, 2010, 45(2): 401-410.