International Journal for Modern Trends in Science and Technology, 9(02): 221-226, 2023

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST0902040
Available online at: http://www.ijmtst.com/vol9issue02.html

Design of Generic Mesochronous FIFO using DPRAM

Ch. Sree Harshitha[1] | B.Lakshmi[2]

[1]PG Scholar, Department of Electronics and Communication Engineering, GVPCEW, Visakhapatnam.

[2]Assistant Professor, Department of Electronics and Communication Engineering, GVPCEW, Visakhapatnam.

To Cite this Article
Ch. Sree Harshitha and B.Lakshmi. Design of Generic Mesochronous FIFO using DPRAM. International Journal for Modern Trends in Science and Technology 2023, 9(02), pp. 221-226. https://doi.org/10.46501/IJMTST0902040

Article Info
Received: 18 January 2023; Accepted: 23 February 2023; Published: 26 February 2023.

ABSTRACT

FIFO which stands for First in First Out which means whatever data is written first is read first. when the clock domains of write and read are of same frequencies then there is control on the data flow so, there is no loss of data. But when the write and read frequencies are different there is no control on the data flow due to this there is loss of data. But in the reality, it is difficult to match the read and write frequencies. so, in order to avoid the loss of data when different frequencies used for write and read, implement a FIFO between the two clock domains of write and read. This project is an implementation of First in first out algorithm which overcomes clock domain crossing. Designed module is tested against synchronous clock domains, mesochronous clock domains and proposed mesochronous clock domains. The parameters observed here are memory(461780KBytes), frequency(686.86MHZ) and delay(1.456ns). The delay is reduced by 82% when compared with mesochronous dual-clock FIFO.

KEYWORDS: - FIFO, Clock Domain, mesochronous FIFO.

1. LITERATURE REVIEW

A FIFO is a special type of buffer. The name FIFO stands for first in first out and means that the data written into the buffer first comes out of it first. There are other kinds of buffers like the LIFO (last in first out), often called a stack memory, and the shared memory. The choice of a buffer architecture depends on the application to be solved.

FIFOs can be implemented with software or hardware. The choice between a software and a hardware solution depends on the application and the features desired. When requirements change, a software FIFO easily can be adapted to them by modifying its program, while a hardware FIFO may demand a new board layout. Software is more flexible than hardware. The advantage of the hardware FIFOs shows in their speed.

Clocks having two different frequencies is called clock domain. Transferring data between two different clock domains is called clock domain crossing. Due to this there is a lossof data. In order to prevent that need to design FIFO between the two different clock frequency domains. FIFO stands for First in First Out is an interface between two clock domains. This will prevent the loss of data as it provides memory to store the data.

fw -- write clock frequency, fr -- read clock frequency.

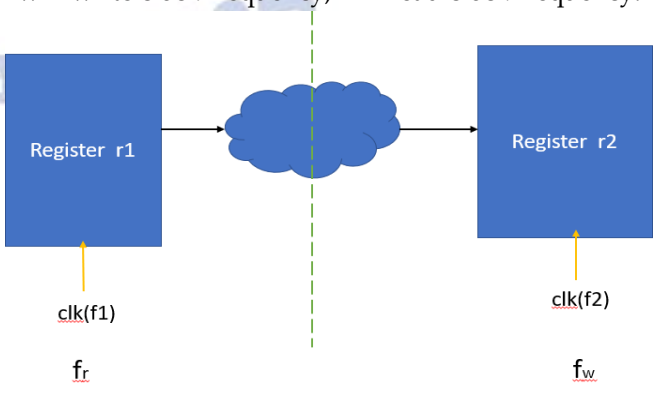

Figure 1: Clock Domain Crossing

So, if fr = fw then they are said to be synchronous and there will be no loss of data. But if fr < fw then they are not synchronized and there will be loss of data and also if fr > fw then also there will be loss of data but we get garbage data. In order get the synchronized data flow between two different clock domains then FIFO should be placed between them. The design of FIFO is done based on the FIFO_DEPTH so that it will store the data that is lost.

2. SYSNCHRONOUS FIFO

Synchronous FIFO is a first in first out queue in which there is a single clock pulse for both data write and read. In synchronous FIFO the read and write operations are performed at the same rate. The signals wt_en, wt_addr and wt_clk are used to write the data into the memory with respect to the write reset. The signals rd_en, rd_addr and rd_clk are used to read the data from the memory with respect to the read reset.

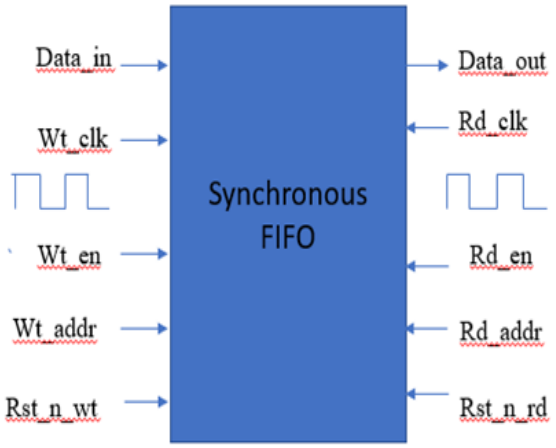

Figure 2: Diagram of Synchronous FIFO

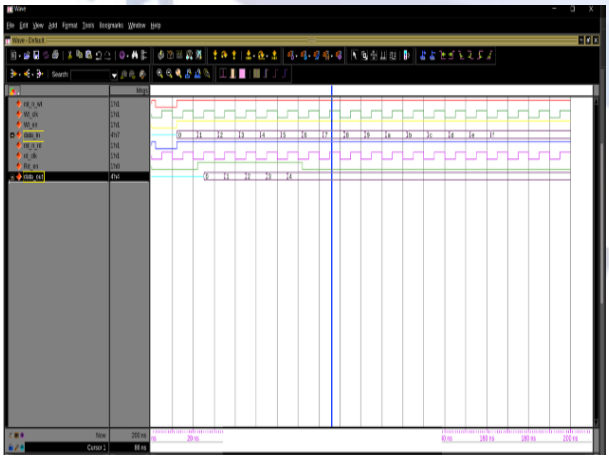

Figure 3: Waveform of Synchronous FIFO

The observation from the output is that there is no loss of data as both write and read frequencies are same and the phases of both the clock signals are same.

As, the write and read clocks are of same frequency the read has to wait till the completion of write operation. so, there may be a chance for the occurrence of delay. So, to reduce that is go for Mesochronous FIFO.

3. MESOCHRONOUS FIFO

In Mesochronous FIFO there is a single clock pulse for both data write and read but there are different in phases. The read and write operations are not performed at the same rate. The signals wt_en, wt_addr and wt_clk are used to write the data into the memory with respect to the write reset. The signals rd_en, rd_addr and rd_clk are used to read the data from the memory with respect to the read reset.

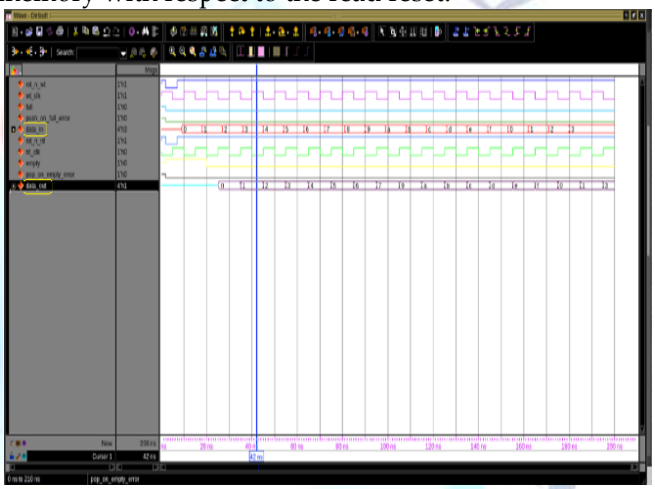

Figure 4: Waveform of Mesochronous FIFO

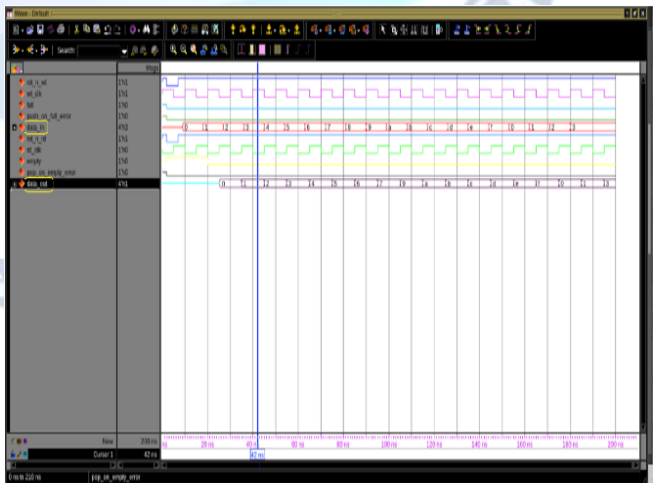

Figure 5: Diagram of Mesochronous FIFO

From the output it is observed that there is no loss of data as both write and read frequencies are same. But, the phases of both the clock signals are different. And the delay obtained here is less compared to the synchronous FIFO. As, the write and read of having different phases the read operation can be done without waiting for the completion of the write operation by this the delay is reduced. The delay is further reduced with the Proposed Mesochronous FIFO.

4. PROPOSED MESOCHRONOUS FIFO

An Mesochronous FIFO refers to a FIFO where the data value is written to the FIFO at a different rate and data values are read from the same FIFO at a different rate, both at the same time. The reason for calling it Mesochronous FIFO, is that the read and write clocks are not synchronized.

The basic need for a Mesochronous FIFO arises when we are dealing with systems with different data rates. For the rate of data flow being different, we will be needing Mesochronous FIFO to synchronize the data flow between the systems. The main work of an Mesochronous FIFO is to pass data from one clock domain to another clock domain.

The architecture consists two paths, namely data path and control path. Data Path through which the data the data is passed and processed. Control path it controls the data path.

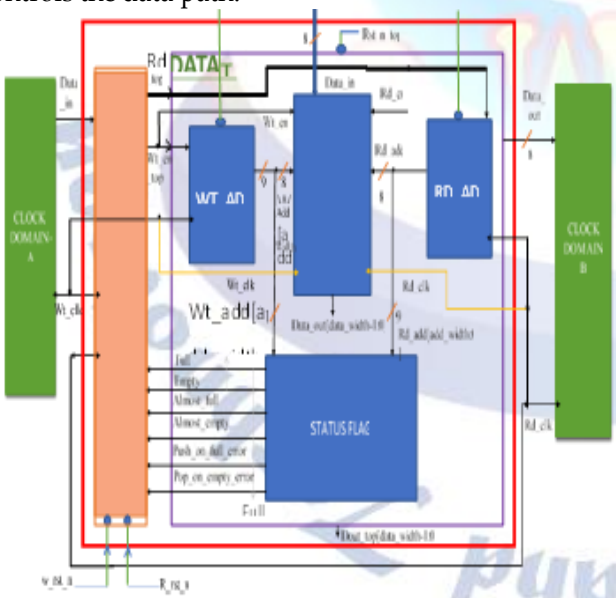

Figure 6: Architecture of Proposed Mesochronous FIFO

a) DATA PATH:

The data path is in which the data is passed with respect to the control signals write enable, write clock, read enable, and read clock generated by the control path based the signals from the status flag. The data is written into the DPRAM based on the address generated by the write address generator and read address generator. The data path contains 4 blocks. Theyare:

1. DPRAM Block
2. Write Address Generation Block
3. Read Address Generation Block
4. Status Flag Block

1. DPRAM:

This stands for Dual Port Random Access Memory. The ports that are included are: Write Port, Read Port. The data is written into the memory with Wt_en(write enable), Wt_clk(write clock) for the give Wt_address(write address). The data is read from the memory with the help of Rd_en(read enable), Rd_clk (read clock) from the given Rd-address. The port refers to where the transaction of data should take place. Write port refers the transaction of data is write that is data is written into the DPRAM with respect to the write address which specifies in what location the data should be written. The read port refers to the transaction od data is to read that is data is read from the particular location of DPRAM with respect to the read address which specifies from what location the data should be read.

2. Read Address Generation:

In this Read address generation block the Read address is generated with the Rd_en (Read enable) and Rd_clk(Read clock). The active low reset which resets the Read address generation block. The address generated from this will determine from which location the datashould be read.

3. Write Address Generation:

In this write address generation the write address is generated with the Wt_en mean write enable and Wt_clk mean write clock. The active low reset which resets the write address generation block. The write address generated by this will tell in which location the data shouldbe written.

4. Status Flag:

The input of status flag block is Wt_add[add_width:0] and Rd_add[add_width:0]. It will tell the status of Wt_add (Write address) and Rd_add (Read address) that is full, empty, almost_full, almost_empty, pop_on_empty_error, push_on_full_error. Full condition when the msb of write address and read address in different and all others bits are equal then it is considered as full. When the write address and read address are equal then the condition is empty. When the write address meets the threshold it will give the almost_full condition. When the read address meets the threshold, it will give almost_empty. When the FIFO is completely read when you are trying to read the memory further then the pop_on_empty_error will occur. When the FIFO is completely filled when you are trying to write into the memory further then the push_on_empty_error will occur.

Figure 7: Waveform of Data Path

b) CONTROL PATH:

It gets the inputs Wt_clk that is write clock and data_in from clock domains –A and Rd_clk that is read clock from clock domain - B. The outputs of status flag that is full, empty, almost_full, almost-empty, pop_on_empty_error, push_on_full_error is given as input to the control path. It contains two active low resets one from writing and another for reading. The control path will generate the outputs Wt_en (write enable) and Rd_en (read enable). When the write enable is activated then data is written into the FIFO. When read enable is activated then the data is read from the FIFO. When full is generated from status flag then the write enable will not active. When the empty is generated then the write enable is activated which allows to write into the memory and the read enable is de-activated.

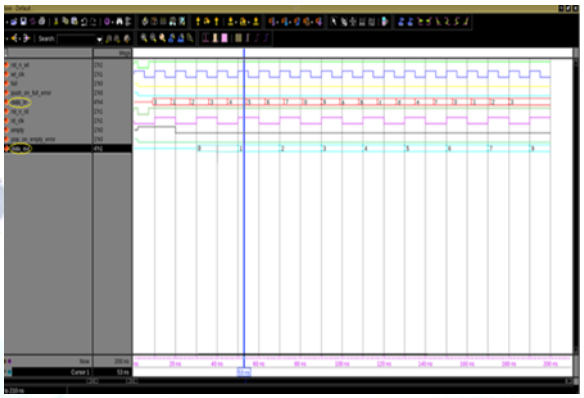

Figure 8: waveform of control path

The waveform of the Mesochronous FIFO is shown below:

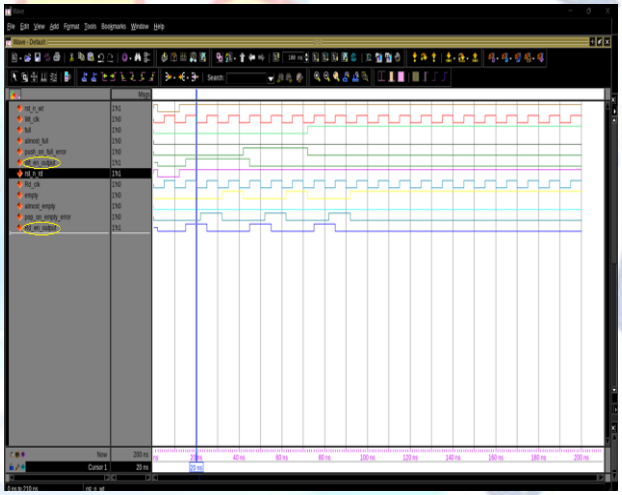

Figure 9: Waveform of Proposed Mesochronous FIFO

5. COMPARISON OF RESULTS

The comparison of results for the Synchronous FIFO, Mesochronous FIFO and Proposed Mesochronous FIFO are done. The parameters that are compared are memory, frequency, power, phase and delay. From the result, it is observed that synchronous FIFO is having less memory and low power consumption compared with others and as for delay the proposed mesochronous FIFO is having less.

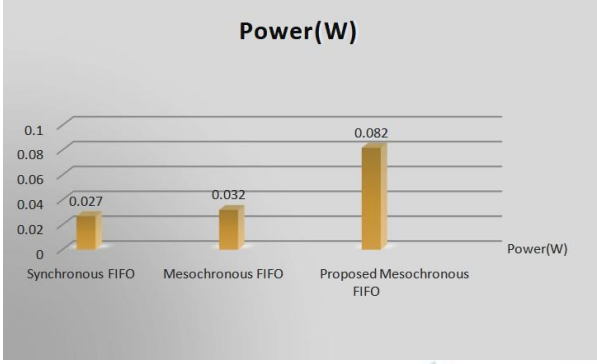

Figure 10: Comparison of Power

And from this power plot, it is observed that the synchronous FIFO is having less power consumption.

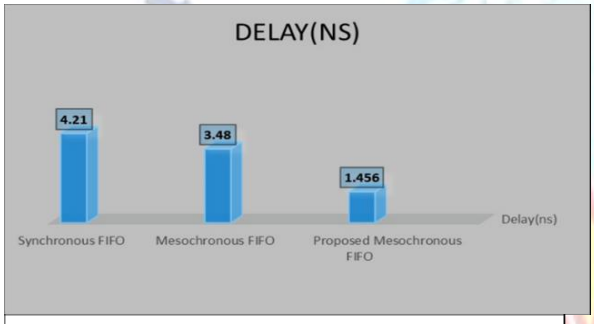

Figure 11: Comparison of Delay

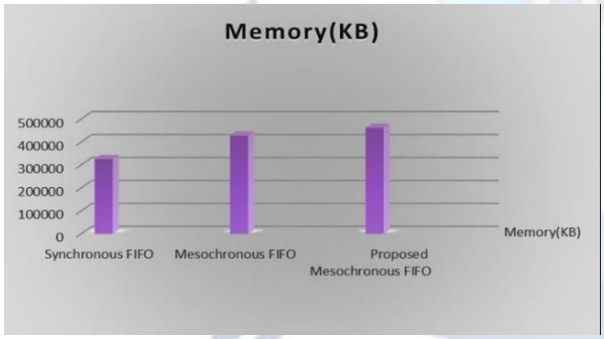

Figure 12: Comparison of Memory

And from this plot, it is observed that proposed mesochronous FIFO is having less delay when compared with others.

And from this plot, it is observed that synchronous FIFO is consuming less memory that the other two.

Table 1: Comparison of Results

Parameters	Synchronous FIFO	Mesochronous FIFO	Proposed Mesochronous FIFO
Frequency(MHZ)	F_r =100 F_w=100 (F_w=F_r)	F_r =100 F_w=100 (F_w=F_r)	F_r =75 F_w=100 ($F_w \neq F_r$)
Phase	$\emptyset_r = \emptyset_w$	$\emptyset_r \neq \emptyset_w$	$\emptyset_r \neq \emptyset_w$
Delay(ns)	4.21	3.48	1.456
Memory(KB)	325706	428100	461780
Power(w)	0.027	0.032	0.082

From the result, it is observed that synchronous FIFO is having less memory and low power consumption compared with others and as for delay the proposed mesochronous FIFO is having less.

5. CONCLUSION

In order to synchronize the data flow between the systems with different clocks the Mesochronous FIFO is used. By implementing the generic Mesochronous FIFO, the loss of data can be avoided when different frequencies and phases are used for write and read. In this the implementation of synchronous, mesochronous and Proposed Mesochronous can be done without any loss of data. The delay is reduced to 1.456ns. The code is implemented and verified in Verilog. The tool used is Questa sim and the parameters observed here are power, memory, frequency and delay.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] The Mesochronous Dual-Clock FIFO Buffer, Dimitrios Konstantinou, Anastasios Psarras, Chrysostomos Nicopoulos, and Giorgos Dimitrakopoul, Jan. 2020. T. Chelcea and S. M. Nowick, "Robust interfaces for mixed-timing systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 8, pp. 857–873, Aug. 2004.

[2] A. M. S. Abdelhadi and M. R. Greenstreet, "Interleaved architectures for high- throughput synthesizable synchronization FIFOs," in Proc. 23rd IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC), May 2017, pp. 41–48.

[3] P. Teehan, M. Greenstreet, and G. Lemieux, "A survey and taxonomy of GALS design styles," IEEE Des. Test Comput., vol. 24, no. 5, pp. 418–428, Sep./Oct. 2007.

[4] J. Ax, N. Kucza, M. Vohrmann, T. Jungeblut, M. Porrmann, and U. Rückert, "Comparing synchronous,

mesochronous and asynchronous NoCs for GALS based MPSoCs," in Proc. IEEE MCSoC, Sep. 2017, pp. 45–51.

[5] J. Ax, N. Kucza, M. Vohrmann, T. Jungeblut, M. Porrmann, and U. Rückert, "Comparing synchronous, mesochronous and asynchronous NoCs for GALS based MPSoCs," in Proc. IEEE MCSoC, Sep. 2017, pp. 45–51.

[6] W. J. Dally and J. W. Poulton, Digital Systems Engineering. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[7] R. Ginosar, "Metastability and synchronizers: A tutorial," IEEE Des. Test Comput., vol. 28, no. 5, pp. 23–35, Sep./Oct. 2011.

[8] R. W. Apperson, Z. Yu, M. J. Meeuwsen, T. Mohsenin, and B. M. Baas, "A scalable dual-clock FIFO for data transfers between arbitrary and haltable clock domains," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 10, pp. 1125–1134, Oct. 2007.

[9] D. Verbitsky, R. R. Dobkin, R. Ginosar, and S. Beer, "StarSync: An extendable standard-cell mesochronous synchronizer," Integration, vol. 47, no. 2, pp. 250–260, Mar. 2014.

[10] D. Ludovici, A. Strano, D. Bertozzi, L. Benini, and G. N. Gaydadjiev, "Comparing tightly and loosely coupled mesochronous synchronizers in a NoC switch architecture," in Proc. NOCS, May 2009, pp. 244–249.

[11] D. Ludovici, A. Stranoy, G. N. Gaydadjievx, L. Beniniyy, and D. Bertozziy, "Design space exploration of a mesochronous link for cost-effective and flexible GALS NOCs," in Proc. IEEE DATE, Mar. 2010, pp. 679–684.

[12] S. Saponara, F. Vitullo, R. Locatelli, P. Teninge, M. Coppola, and L. Fanucci, "LIME: A low-latency and low-complexity on-chip mesochronous link with integrated flow control," in Proc. 11th EUROMICRO Conf. Digit. Syst. Design Archit., Methods Tool, Sep. 2008, pp. 32–35.

[13] A. Edman and C. Svensson, "Timing closure through a globally synchronous, timing partitioned design methodology," in Proc. 41st Design Autom. Conf., Jul. 2004.

[14] M. Ghoneima, Y. Ismail, M. Khellah, and V. De, "Variation-tolerant and low-power source- synchronous multicycle on-chip interconnect scheme," VLSI Des., vol. 2007, Mar. 2007, Art. no. 95402.

[15] I. Loi, F. Angiolini, and L. Benini, "Developing mesochronous synchronizers to enable 3D NoCs," in Proc. Design, Autom. Test Eur., Mar. 2008, pp. 1414–1419.

[16] F. Vitullo et al., "Low-complexity link microarchitecture for mesochronous communication in networks-on-chip," IEEE Trans. Comput., vol. 57, no. 9, pp. 1196– 1201, Sep.