HDL Design and Verification of General Purpose Input Output (GPIO) Coprocessor

[1]Priyanka, [2]Manjit Kaur, [3]Gurmohan Singh
[1]Research Scholar, ACS Division, C-DAC, Mohali, India-160071
[2] Engineer, ACS Division, C-DAC, Mohali, India-160071
[3]Senior Engineer, DEC Division, C-DAC, Mohali, India-160071
[1]sharmapriyanka309@gmail.com, [2]manjeet@cdac.in, [3]gurmohan@cdac.in,

Abstract

One of the key features of a microprocessor is to interact with and control peripheral devices. These input/output interfaces may be DMA, UART, SPI, CAN, memories, ADCs/DACs, and interrupt controllers etc. In order to interface with such devices, almost all microprocessors have dedicated interfaces that can be programmed as input or output according to user applications. Nowadays, most widely used digital interface for data handling in microprocessor is General Purpose Input/ Output (GPIO) Coprocessor. GPIO coprocessors relieve microprocessor of responsibilities of handling various input/output interfaces and in the meantime perform other meaningful computations. In this research work, a GPIO coprocessor IP core has been designed and functionally verified. The performance analysis of GPIO has been performed using different priority implementation techniques. The priorities of the different peripherals that are interfaced with the GPIO interfaces are configurable using arbitration technique. The arbitration techniques used are static fixed priority, TDMA based, and centralized arbiter with dynamic priority. The GPIO IP core has been analyzed with and without an arbitration algorithm. The delay for executing with or without arbitration is computed. The design is described using Verilog HDL and verified using Xilinx ISim simulator. The delay for the GPIO module using centralized arbiter with dynamic priority is 10.557 ns. The delay computed for the static fixed priority is less as compared to TDMA based and centralized arbiter with dynamic priority i.e. 8.357 ns. The simulation results validates that by using suitable arbitration algorithm as per requirement of application, the performance of coprocessor IP core can be enhanced by 21%.

Keywords: GPIO, arbitration algorithm, TDMA, round robin, centralized arbiter.

1. Introduction

There are various applications that uses microcontroller hence to create adaptability in the field of communication, there is a need to provide flexibility in terms of data input and output transmission with microprocessor. Depending on the user's desire and application requirement, these pins available on a processor can be configured to be used to either accept input or provide output to peripheral devices. On these pins, the various data handling methods are implemented such as interrupt handling and ADC conversion. Most of the devices required a functionality through which they can communicate with the other components in the system using low speed interface pins. The General purpose input/output (GPIO) IP core is user-selectable general-purpose input/output controller. The GPIO pins can be programmed for I/O or bypass mode. Each single or multiple bits can be set or cleared independently. Each GPIO port can act as an interrupt source and has its own configuration options i.e. level sensitive, active high or low and status flag. The GPIO coprocessor provides a connection to the I/O pins of peripherals, can be 16 or 32 I/O channel. I/O pads can be connected with the channel to either input/output data registers of GPIO or any one of the selectable peripheral devices. Each I/O pad is provided with control and data signal by GPIO to select data direction and data of I/O signal to transmit or receive. For level sensitive interrupt capability the I/O pins can be

***© 2016 Vidya Publications.* Authors are responsible for any plagiarism issues.**

made active. Control registers of the GPIO controller enable the per channel control of the GPIO. Each GPIO pin is configurable to generate a CPU interrupt. The interrupt can be introduced on rising, falling, and both the edges of the GPIO signal. GPIO peripheral clock is synchronized with the edge detection logic module of the GPIO.
The remainder of this paper is organized as follows. Section 2 introduces the standard architecture and block diagram of the GPIO. The detailed description of arbitration techniques are discussed in section 3. Section 4 gives the simulation results and the performance analysis of the proposed GPIO module. Conclusion is covered in section 5.

2. GPIO Architecture

Basic architecture of GPIO is consisting of four main building blocks as shown in Fig. 1 [3]:

1) GPIO registers
2) APB
3) Interface to external I/O cells and pads
4) Auxiliary inputs

GPIO Register

Multiple software accessible registers are present in GPIO IP core. Mostly registers have same bandwidth as the number of general purpose input output signals and that are generally from 0-31 bits. The operation of each GPIO signal can be programmed using these registers.

APB

APB belongs to the logic family of AMBA protocol. It provides a good interface that is optimized for reduced complexity of interface and minimum power consumption.

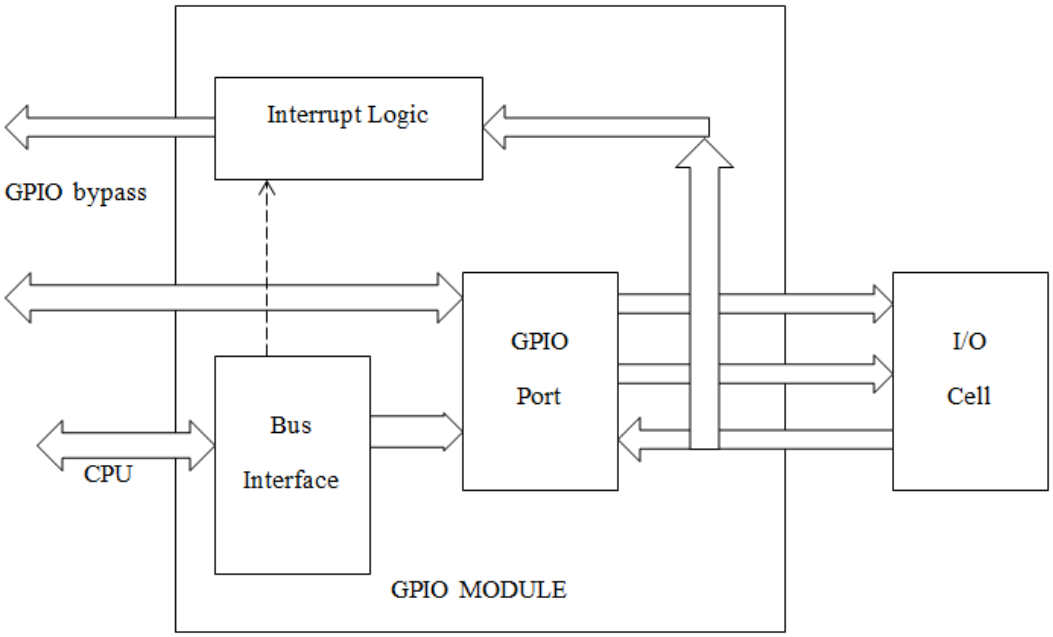

Figure 1: GPIO Block Diagram [3]

Interface to external I/O cells and pads

It connects the GPIO core to external cells and pads [3]. The interface between processor and the external peripherals is provided by this module.

Auxiliary inputs

Auxiliary inputs can bypass the outputs of the GPIO. They are used to multiplex other on chip peripherals to the GPIO pin.

Interrupt logic

© 2016 Vidya Publications. **Authors are responsible for any plagiarism issues.**

The GPIO can be configuring to handle any type of the interrupt. It could be edge sensitive or level sensitive. To make it to sense the coming interrupt at the clock edge, the values of the registers are compared with the predefined value of the register [3]. If the set rising edge is *'1'* and clear rising edge is *'0'*, it means the pin is sensitive for the positive edge interrupt signals. If the set falling edge is *'1'* and clear falling edge is *'0'*, it means the pin is sensitive for the negative edge interrupt signal.
Some of the important features are GPIO signals are user programmable and its range is from 1 to 32, Several GPIO cores can be used in parallel for more I/O pins, Full synchronous design, the input data is synchronized and the direction of pin is programmable.

3. Arbitration Algorithms

An arbiter is required for sharing the resources among many requesting devices. This property of the arbiter is used to increase the performance of GPIO to handle the various interrupts by the peripherals on the GPIO pins. When all the peripherals start requesting the access to the controller simultaneously, then, there should be some logic which will decide which will be granted access the controller first. The arbiter takes the requests from the peripherals at the rising edge of the clock and then decides which master will be granted to take the access [13]. The arbitration play very significant role to determine performance of the bus based system, as it assign priorities with which processor grant the access to the shared communication resources. Round robin arbitration is generally used for scheduling. Priority arbiters are used as building block for other types of arbiter. In a simple priority arbiter, each requesting device is assigned with a fixed priority and the arbiter with the highest priority is granted access to the buses. A complex arbiter can reorder the incoming request in the desired priority, run these scrambled requests through a simple priority arbiter and then decode the grant which come out. Various arbiters are used for different application. Arbiter with minimum delay is preferred over others if priority is not the point of concern. But if the importance of the peripheral depends upon their priority value then in that case priority based arbitration algorithms are preferred.
In the recent years, the need for effective implementation of arbiter has increased due to the increased number of on chip peripherals. The main function of any arbiter is the arbitration that is to resolve the conflicting requests for the same source [13]. In the round robin arbiter, the problem of conflicting requests is resolved in a directional or cyclic order. The round robin algorithm is simply the form of time slicing or scheduling. It allows each peripheral to access the controller for an equal share of time. The turn of each peripheral is fixed in a particular manner i.e. circular. A token is moved through the peripherals. It stays with the peripherals for a fixed interval of time [12].The wastage of clock cycle is more in the round robin because the peripherals have token even when they doesn't required the access to the processor. But problem of bus starvation is solved by only round robin arbitration algorithm. So, this algorithm is preferred where every peripheral is requires equal treatment. But, it is so fair due to which sometimes the decline in the bus efficiency may also occurs. The disadvantage of the fixed priority algorithm is mainly faced in the case of heavy traffic condition. In heavy traffic, the peripheral with the lowest priority will not get grant easily. Dynamic arbitration algorithm is used to overcome the disadvantage of fixed priority and round robin algorithm. in this algorithm the priority can be changed at run time, while it is not possible with fixed arbitration and round robin.
The major shortcoming of this algorithm is that it is very busy system, no limit is present here that how long a low priority peripheral may need to wait until it access grant. The delay offered by the dynamic arbitration is more as compared to the round robin and fixed priority arbiter because of the extra time taken by it to make decisions. The arbitration algorithm mainly concludes the achievable throughput and delay performance of

***© 2016 Vidya Publications.* Authors are responsible for any plagiarism issues.**

processors. The complexity of any arbitration algorithm is determined by the total number of bits.

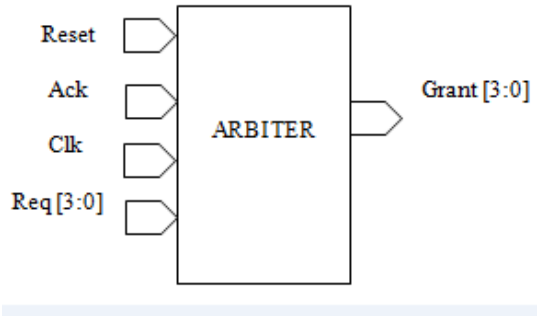

Figure 2: Example of Arbitration [13]

The Fig. 3 shows the flow and working of the dynamic arbiter. The policy of the arbitration refers to the algorithm or logic by which the arbiter decides to give the grant or access to the bus for requesting devices when multiple masters request the bus simultaneously. It also decides what to do when none of the masters accessing the bus. The performance of the system mainly depends upon the design of the communication architecture.

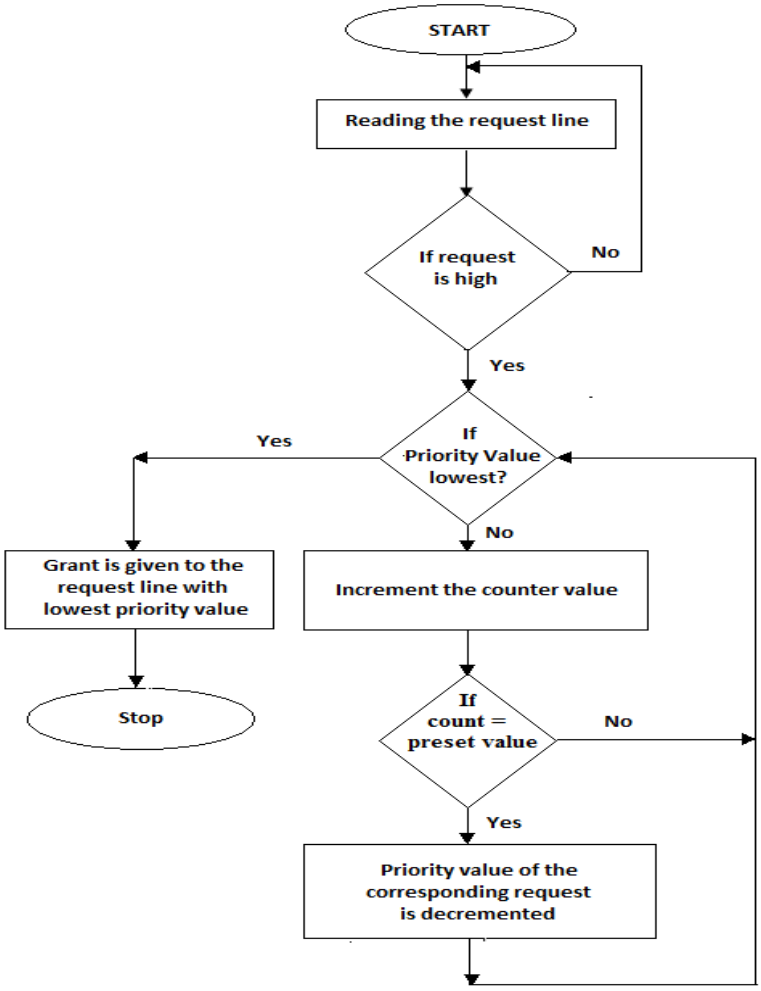

Figure 3: Dynamic Arbitration Flow [12]

© 2016 Vidya Publications. **Authors are responsible for any plagiarism issues.**

4. Results and Analysis

The simulation results of designed GPIO IP core show the functional working of this module. Four different peripherals are taken as an interrupt source. These peripherals generate interrupt request to the GPIO as per their requirement. Four I/O pins of GPIO are configured for the use of peripherals and another eight I/O pins are used to transfer data of 8-bit. First of all, the synchronization is performed for all the peripherals. The synchronization between the peripheral and the processor is essential. After the synchronization, the direction of the I/O pins is configured. They can be used as input as well as output. Four *'int_req'* signals are taken which represents the interrupt generated by the corresponding peripheral. Four data registers of 8-bit are used to store the data to be transferred between GPIO and peripherals. The *'data_out_gpio'* is assigned with the output data register. As soon as any peripheral requests for the buses by generating interrupt to the I/O pin, then the grant is given to a peripheral according to the decision of the arbitration algorithm. The three different algorithms are used to analysis the performance of the GPIO. The main difference between these is the time delay. The three algorithms used are fixed priority arbitration, TDMA based arbitration and dynamic priority based centralized arbitration.

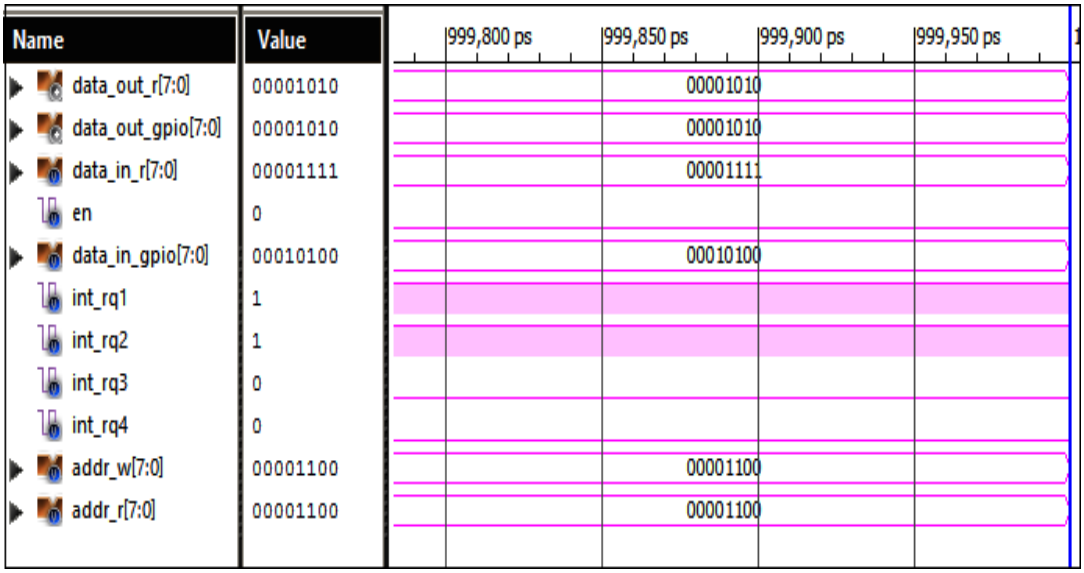

Figure 4: Simulation Waveform of GPIO Module

The Fig. 4 shows the simulation results of the GPIO. The *'data_in_r'* signal shows the value stored in the register for transmission, in this case it is *'15'*. The signal *'data_in_gpio'* shows the value which is to be transmitted over the buses of the processor by the peripheral and the *'data_out_gpio'* is the signal which shows the value of the data output of the GPIO module. The *'int_rq1'*, *'int_rq2'*, *'int_rq3'* and *'int_rq4'* are the input signals used by the peripherals to generate the interrupt signals. As Fig. 4 shows the *'int_rq1'* and *'int_rq2'* are high it means that the peripheral 1 and 2 are requesting for the bus access. Now after that the arbiter will provide the bus access to one of the requesting peripheral and then its value will be given to the *'data_out_gpio'*.

The Fig. 5 shows the simulation results of the GPIO. The *'data_in_r'* signal shows the value stored in the register for transmission, in this case it is *'20'*. The signal *'data_in_gpio'* shows the value which is to be transmitted over the buses of the processor by the peripheral and the *'data_out_gpio'* is the signal which shows the value of the data output of the GPIO module. The *'int_rq1'*, *'int_rq2'*, *'int_rq3'* and *'int_rq4'* are the input signals used by the peripherals to generate the interrupt signals. As the Fig.5 shows the

***© 2016 Vidya Publications.* Authors are responsible for any plagiarism issues.**

'int_rq2' and *'int_rq3'* are high it means that the peripheral 2 and 3 are requesting for the bus access. Now after that the arbiter will provide the bus access to one of the requesting peripheral and then its value will be given to the *'data_out_gpio'*.

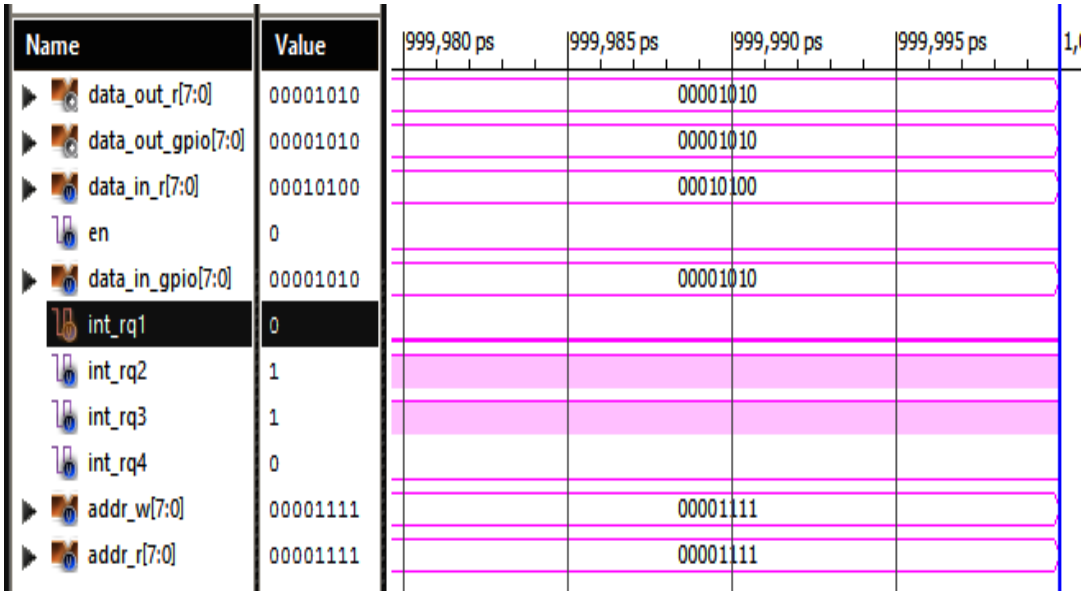

Figure 5: Simulation waveform of GPIO module with more than one requests

Now, this GPIO module is used with the different arbiters and the path delay of the module is calculated for each arbiter. The grant signal decided by each arbiter is different according to the algorithm used for that particular arbiter. The data signals are of 8 bit and the interrupt signals are of 1-bit. If any one of the interrupt signals is high it means that particular peripheral is requesting for the buses. Then the next job is performed by the arbiter. According to the algorithm of arbiter, grant is given to the one peripheral. Then the data that peripheral have to send is moved to the *'data_out_gpio'*. This data is further transmitted over the buses of the processor.

In Fig. 6, the simulation result of the dynamic priority based arbiter has been shown. In this case two peripherals are requesting at the same time i.e. peripheral 2 and peripheral 4. As per algorithm, the grant is given to the peripheral 2 because according to the algorithm this peripheral has the counter value equals to the preset value and hence its priority value is lowest at that moment. Hence, it is granted access to the buses.

The delay value of this arbiter is maximum i.e. 14.640 ns. The reason for this large value is the time taken by the arbiter to make decision is more as compared to others.

***© 2016 Vidya Publications.* Authors are responsible for any plagiarism issues.**

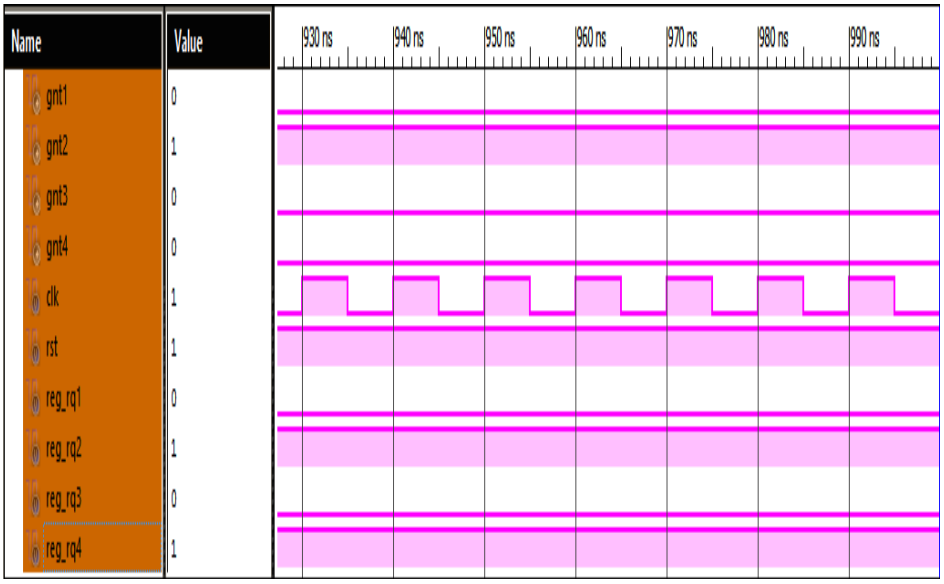

Figure 6: Simulation waveform of dynamic priority based arbiter for multiple requests

Table 1: Comparison of the delay values of all the arbiters

Arbitration Algorithm	Combinational path delay (ns)
Fixed Priority	8.357 ns
TDMA Based	10.883 ns
Random Priority	10.557 ns
Dynamic Priority	14.640 ns

As shown in Table 1, the delay value for the fixed priority algorithm is lowest because here the priorities are fixed already so the arbiter does not take much time to make decisions. While for the dynamic priority based algorithm yields more delay. But the facility to alter the priorities of the peripherals at run time is only one advantage of this arbiter. More priority is given to the frequently requesting peripheral.

***© 2016 Vidya Publications.* Authors are responsible for any plagiarism issues.**

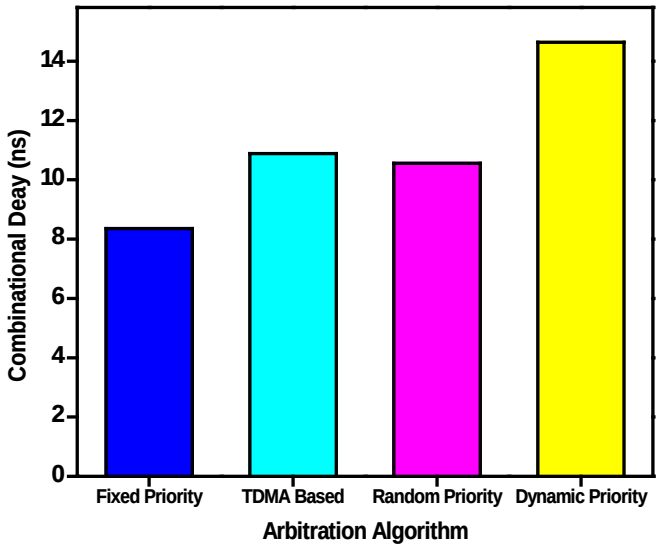

Figure 7: Comparison of delay values of the arbiters

The bar graph in Fig. 7 shows the comparison between delays shown by different arbiters. The delay value for fixed priority arbiter is small as compared to others and the delay shown by the dynamic priority based arbiter is the highest.

5. Conclusion

The GPIO coprocessor IP core was described in Verilog HDL. Firstly, the GPIO module is designed to interface the peripherals with the processor. The different registers are defined for the storage of the data to be transmitted over the buses of the processor. Interrupt generator is designed and configured to detect the edge sensitive interrupts. Synchronization logic is designed to synchronize the different peripherals with the processor. After the synchronization, decision of granting the access of bus is performed by the arbitration algorithm. The performance analysis of GPIO has been performed using different priority implementation techniques. The priorities of the different peripherals interfaced with the GPIO interfaces are configurable using arbitration technique. The arbitration techniques used are static fixed priority, TDMA based, and centralized arbiter with dynamic priority. The GPIO IP core has been analysed with and without an arbitration algorithm. The delay for executing with or without arbitration is computed. A significant reduction in the delay is observed by using the different arbitration algorithms for the designed GPIO IP core.

REFERENCES

[1] Y. Jianfei, L. Zhaolin, W. Chipin, Z.Qingwei and C.Jiajia, “Design of a Configurable General-Purpose Input/output with Event-Capture,” *in Proc. of Second Pacific-Asia Conf on Circuits, Communications and System (PACCS)*, vol. 1, pp. 335-338, Aug. 1- 2, 2010.

[2] S. kim, k. H. cho and B. Min, “An efficient GPIO block design methodology using formalized SFR description,” *in Proc. of International SoC Design Conf (ISOCC)*, pp.84-87, Nov. 17-18, 2011.

[3] B. Patel and B. Tarpara, “Design and Implementation of General Purpose Input Output (GPIO) Protocol,” *International Journal of Scientific Engineering and Applied Science (IJSEAS)*, vol. 1, no. 3, pp. 478-483, Jun. 2015.

***© 2016 Vidya Publications.* Authors are responsible for any plagiarism issues.**

[4] B. Patil, Anuradha J. P. and Mrs. Shanthi V A, "Design and Development of Verification Environment to Verify GPIO Core using UVM," *International Journal of Scientific and Research Publications*, vol. 5, no. 6, Jun. 2015.

[5] L.V. Raju, B. K. V. Prasad,A.L.G.N.Aditya, A.Jhansi Rani and D.N .Dilip Kumar, "Functional Verification of GPIO Core Using OVM," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 2, pp.534-537, May 5, 2012.

[6] S. Mohan and A. Joseph, "A Dynamic Priority Based Arbitration Algorithm," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 3, no. 1, pp. 232-234, Jun. 2013.

[7] L.Nazir and R. N.Mir,"Performance Analysis of various scheduling algorithms using FPGA Platforms," *in Proc. of Int. IEEE Conf VLSI Systems, Architecture, Technology and Applications*, pp. 1-4, Jan. 8-10, 2015.

[8] F. Pe'trotand D. Hommais, "A generic programmable arbiter with default master grant," *in Proc. of Int. IEEE Symposium on Circuits and Systems*, vol. 5, pp. 749- 752, May 28-31, 2000.

[9] Y. J. Huang, C. M. Ko, and H. C.Teng, "Design and Performance Analysis of A Reconfigurable Arbiter," *WSEAS Trans. on Electronics*, vol. 5, no. 4, pp. 132-141, Apr. 2008.

[10] K.T. Teja and P.V.S. S.Tarun, " Implementation of Round Robin Arbiter using Verilog," *International Journal of Advanced Research in Electrical,Electronics and Instrumentation Engineering*, vol. 4, no. 10, pp. 8078-8080, Oct. 2015.

[11] M. H. khan, P. podder, M .M. rahman,T. rahman and T.zaman, "design of a round robin arbiter on resource sharing," *in Proc. of 8th IRF International Conf* , pp. 94-96, May 4, 2014.

[12] T. K. Gauttam, R. Agrawal and S. Sharma, " Arbiter Design Using Verilog for Switching to Communicate in Between Multiple Resources," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol.3, no..3, pp. 3-11, Aug. 2013.

[13] R. Deb and Dr.Rajrajan," Speed efficient implementation of round robin arbiter design using VERILOG," *International Journal of Enhanced Research in Science Technology & Engineering*, vol. 2, no. 9, pp. 1-9, Sep. 2013.

[14] ShashidharR., Sujay S.N. and Pavan G.S. , "Implementation of Bus Arbiter Using Round Robin Scheme," *International Journal of Innovative Research in Science Engineering and Technology*, vol. 3, no. 7, pp. 14937-14944, July 2014.

[15] C. H. Pyoun, C. H. Lin, H. S. Kim and J. W. Chong, " The efficient bus arbitration scheme in soc environment," *in Proc. of The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications*, 2003.

[16] P.Srinivasan, A.Olugbon, A.Ahmadinia, A. T.Erdogan and T.Arslan, "Power Analysis of Arbitration Techniques for AMBA AHB based Reconfigurable System-on –Chip," *in Proc. of IEEE Conf NORCHIP*, pp. 227-230, Nov. 2006.

[17] Z. Jianmin and S.shengyu, "Design and implementation of General Purpose Interface Controller GPIO_WB IP Core," *Microelectronics & Computer*, vol. 21, pp. 194-198, 2004.

[18] D.Gajski, A. C. H. Wu, V. Chaiyakul, S. Mori, T. Nukiyama and P. Bricaud, "Essential Issues for IP Reuse," *in Proc. of IEEE Conf on design automation ASP-DAC*, pp. 37-42, Jan. 9, 2000.

***© 2016 Vidya Publications.* Authors are responsible for any plagiarism issues.**