http://ijea.jctjournals.com/ ISSN(ONLINE):2320 – 0804

Volume-11, Issue-10, October , 2022

International Journal of Engineering Associates (IJEA)

Page Number: 01-10

An Experimental Study on The Performance of A Tree-Shaped Mini-Channel Liquid Cooling Heat Sink

Raj Kamal Kishor Bharti[1],Prof. DevanderPatle[2]

[1]Research Scholar Very Large Scale Integration *(VLSI)*, [2]Department of Electronics and Communication

[1,2] School of Engineering

[1,2]Sri Satya Sai University of Technology & Medical Sciences, Sehore (M. P.)

rkkbharti@gmail.com,

***Abstract*—The application domain of System-On-Chips (SoC) includes mobile devices, end terminals, multimedia terminals, automotive, set-top-boxes, games, processors etc. The SoC design paradigm relies heavily on reuse of intellectual property cores, enabling designers to focus on functionality and performance of the overall system. This is possible if IP cores are equipped with highly optimized interface for plug and play insertion into communication architecture. To this purpose the virtual Socket Interface Alliance represents an attempt to set the characteristics of industry wide, thus facilitating the match of pre-designed software and hardware blocks from multiple sources. The SoC interconnect must be designed and optimized to support a heterogeneous mix of data paths which may inherently have widely varying performance characteristics. The fabric must reliably deliver the required throughput and hide latency for performance critical paths while simultaneously managing the flow of traffic for slower paths and ports requiring lower bandwidth. Thus the system bus as a whole must strike the appropriate between latency and throughput for the collection data paths. Optimizing around this balance is essential to minimizing power, performance, area (PPA) costs and avoiding an inefficient, over-designed SoC.**

***Keywords*—SOC, PPA, AMBA and QOS., etc.**

I. Introduction

Deep submicron technologies are making the integration of large IP blocks on same silicon blocks on same silicon die technically feasible. As a consequence, several heterogeneous cores can be combined through sophisticated communication architectures on same integrated circuit, leading to the development of flexible hardware platforms able to accommodate highly parallel computation. The application domain of these System-On-Chips (SoC). Includes mobile devices, end terminals, multimedia terminals, automotive, set-top-boxes, games, processors etc.

The SoC design paradigm relies heavily on reuse of intellectual property cores, enabling designers to focus on functionality and performance of the overall system. This is possible if IP cores are equipped with highly optimized interface for plug and play insertion into communication architecture. To this purpose the virtual Socket Interface Alliance represents an attempt to set the characteristics of industry wide, thus facilitating the match of pre-designed software and hardware blocks from multiple sources.

The SoC interconnect must be designed and optimized to support a heterogeneous mix of data paths which may inherently have widely varying performance characteristics. SOC must reliably deliver the required throughput and hide latency for performance critical paths while simultaneously managing the flow of traffic for slower paths and ports requiring lower bandwidth. Thus, the system bus as a whole must strike the appropriate between latency and throughput for the collection data paths. Optimizing around this balance is essential to minimizing power, performance, area (PPA) costs and avoiding an inefficient, over-designed SoC.

There are many other questions involved with optimization to allow for a balanced SoC: What is the best way to isolate and eliminate performance bottlenecks? How can load-balancing and quality of service (QoS) simultaneously be ensured? How will cache coherency impact the interconnect traffic – and system throughput?

The most likely adopted interconnect architecture for soc IP blocks is bus-based and consist of shared communication resources managed by dedicated arbiters that are in charge of serializing access request. This architecture s usually employs hierarchical buses and tends to distinguish between high performance system buses and low complexity and low speed peripheral buses. Many commercial on-chip architectures have been developed to support the connection of multiple bus segments in arbiter topologies, providing at the same time a moderate degree of scalability, Wishbone, Advance Microcontroller Bus architecture (AMBA) and Core Connect are relevant examples.

As complexity of Soc increases, the communication architectures become performance bottleneck of the system. The performance of multiprocessor system depends more on efficient communication among processors and on the balanced distribution of computation among them, rather than CPU speed. For integration levels in orders of hundreds of processors on the same SoC, the most efficient and scalable solution will be the implementation of micro-networks of interconnects but below that limit bus-based communication architectures remain the reference solution of state of art microprocessor system because of lower design and hardware cost. These forces designers to push the performance of these architectures to limit within the architectural degrees of freedom made available by existing commercial bus standards

Memory access is strongly dependent on the processing sequence of memory transactions. On system bus the outstanding memory transaction issued by bus device often have consecutive address and same read write types. Under traditional bus arbitration schemes however outstanding transactions from different devices are most likely to be interleaved with each other, which incurs non sequential readdressing access as well as different R/W types access. Due to limited scheduling performance of memory controller, such sequences usually prevent the memory controller from accessing the memory effectively.

The arbitration process plays a crucial role in determining the performance of the system, at it is assign the priorities with which processors are granted the access to the shared communication resources. The increasing integration levels of SoC translate to increase of contention among the processing elements for the bus, and this might lead to real time violation of real time constraints and more in general to performance degradation. An efficient contention resolution scheme id therefore required to support real-time isochronous data flow associated with networking and multimedia data streams.

A. Arbitration

The arbiter is a electronic devices that allocate access to shared resources. Arbiter block plays important role in the SoC shared bus communication. The masters on a SoC bus may issue requests simultaneously and hence an arbiter is required to decide which master is granted for bus access. Bus Arbiter plays a vital role in handling the requests from the master and responses from slave (like Acknowledgement signal, Retry, etc). The main objective of arbitration algorithms is to ensure that only one master has access to the bus at any given time, all the other masters are forced to remain in the idle state until they are granted the use of the bus.

The arbiter has 2 schemes as follows.

1. Round Robin scheme
2. Fixed/Strict priority scheme

A particular scheme can be programmed as required. The round-robin scheme is about time-slicing that is we must fix a certain amount of time when each process must be executed. It is usually implemented using equal priority for simplicity. If the tasks have a relatively equal importance, then the round-robin works better, since all the tasks get a better chance of getting run; we avoid the situation where the task with the lowest priority hardly ever gets run, since there seems to always be another task with a higher priority. Imagine we need to read data from a number of sources.

The arbiter block plays important role in the SoC shared bus communication. The masters on a SoC bus may issue requests simultaneously and hence an arbiter is required to decide which master is granted for bus access. In many applications, masters may have real-time and/or bandwidth requirements on requests. A master with a real-time requirement demands its transactions accomplished within a fixed number of clock cycles. On the other hand, a master

with a bandwidth requirement must occupy a fixed fraction of total bandwidth of a bus. The arbitration algorithm could be implemented in a centralized or a distributed fashion. In a centralized arbitration scheme, the master side of the arbitration protocol instantiated in each master communicates with the slave side of the arbitration protocol instantiated in an additional arbiter component attached to the bus. In a distributed arbitration scheme, there is no slave side of the arbitration protocol and the master sides of the protocol in each master regulate accesses among themselves. Bus Arbiter plays a vital role in handling the requests from the master and responses from slave (like Acknowledgement signal, Retry, etc). The available arbitration protocols strive to optimize the contentions arising while different masters issue the request for using the bus at the same time. The arbitration algorithms must also be in an optimized manner to handle the contingencies. The main objective of arbitration algorithms is to ensure that only one master has access to the bus at any given time, all the other masters are forced to remain in the idle state until they are granted the use of the bus. The power utilized by the arbitration technique in different on-chip communication tends to vary significantly for a particular application. Thus it makes the comparison of arbitration algorithms a vital step in the SoC design.

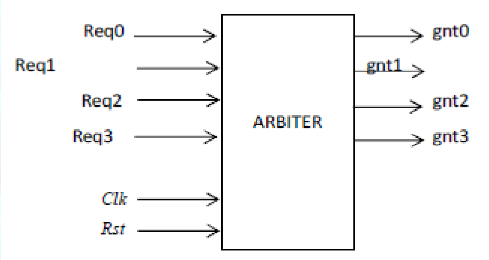


Fig. 1.1 Bus arbiter

The above Figure shows the basic block diagram of bus arbiter. Here for simplicity we are considering only four requests.

The inputs to the bus arbiter are

- Req0 - request signal generated from processor 1
- Req1 - request signal generated from processor 2
- Req2 - request signal generated from processor 3
- Req3 - request signal generated from processor 4
- Clk – clock signal
- Rst – reset signal

The outputs of the arbiter are

- Gnt0 – grant signal for processor 1 in order to acquire CPU& perform data transfer
- Gnt1– grant signal for processor 2 in order to acquire CPU& perform data transfer
- Gnt2 – grant signal for processor 3 in order to acquire CPU& perform data transfer
- Gnt3 – grant signal for processor 4 in order to acquire CPU& perform data transfer

B. ROUND ROBIN Arbitration

A round-robin token passing bus or arbiter guarantees fairness (no starvation) among masters and allows any unused timeslot to be allocated to a master whose round-robin turn is later but who is ready now. A reliable prediction of the worst-case wait time is another advantage of the round-robin protocol. The worst-case wait time is proportional to number of requestors minus one. The protocol of a round-robin token passing bus or switch arbiter works as follows. In each cycle, one of the masters (in round-robin order) has the highest priority (i.e., owns the token) for access to a shared resource. If the token-holding master does not need the resource in this cycle, the master with the next highest priority who sends a request can be granted the resource, and the highest priority master then passes the token to the next master in round-robin order. Here a BA is generated to handle four requests. Figure shows the Bus Arbiter (BA) block diagram for four bus masters. BA consists of a D flip-flop, priority logic blocks, an M-bit ring counter and M-input OR gates as shown in Fig. where M=4.

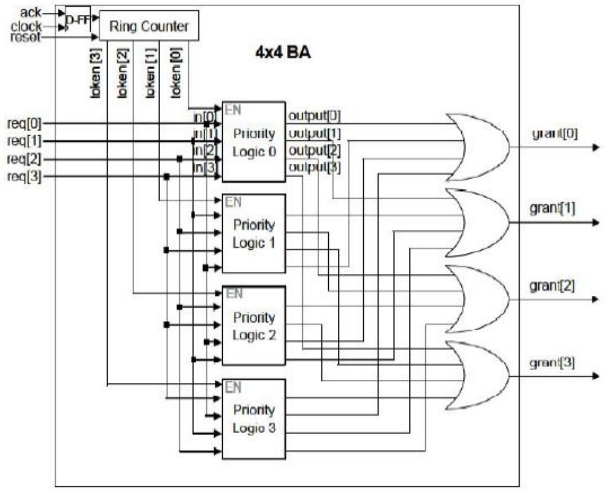


Fig 1.2 Logic Diagram of 4x4 Bus Arbiter

The priority of inputs are placed in descending order from in[0] to in[3] in the priority logic blocks (Priority Logic 0 through 3) shown in Fig. Thus, in[0] has the highest priority, in[1] has the next priority, and so on. To implement a BA, we employ the token concept from a token ring in a network. The possession of the token allows a priority logic block to be enabled. Since each priority logic block has a different order of inputs (request signals), the priority of request signals varies with the chosen priority logic block. The token is implemented in a 4-bit ring counter as shown in Fig. The outputs (four bits) of the ring counter act as the enable signals to the priority logic blocks. Thus, only one enabled priority logic block can assert a grant signal. The ack signal to the bus arbiter is delayed by one arbitration cycle by a D flip-flop as shown in Fig. The delayed ack signal pulls a trigger to the ring counter so that the content of the ring counter is rotated one bit. Thus, the token bit is rotated left each cycle, with 4’b1000 rotating to 4’b0001 in Fig. and the token is initialized to one at the reset phase (e.g., 4’b0001 for four-bit ring counter) so that there is only one ‘1’ output by the ring counter. In the round-robin algorithm, each master must wait no longer than (M- 1) time slots, the period of time allocated to the chosen master, until the next time it receives the token (i.e., highest priority). The assigned time slot can also be yielded to another master if the owner of the time slot has nothing to send. This protocol guarantees a dynamic priority assignment to bus masters (requestors) without starvation.

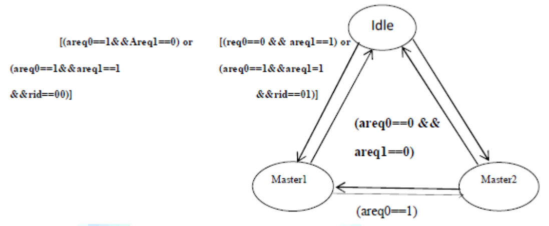


Fig. 1.3 State Diagram for Bus Arbiter (round robin)

State diagram model is used for modelling reactive or event driven embedded systems whose processing behaviour are dependent on state transitions. The model describes the system behaviour with ‘states’, ’events’, ’actions’ & ‘transition’. State is a representation of a current situation. An event is an input to the state. The event acts as stimuli for state transition. Transition is the movement from one state to another. Action is an activity to be performed by the state machine. Here we are considering only two masters are requesting for bus access. It consists of three states-

- Idle
- Master1
- Master2

areq0 & areq1 are the requests generated from master1 & master2 respectively. If master1 requests for cpu access then areq0 is set to one & areq1 signal is forced to be in idle state. State transition of master1 takes place from idle to master1 state. After the completion the data transfer if in case areq1 is set the transition takes from master1 to master2. If areq1 is not set then transition from master1 to idle state occurs. If master2 requests for cpu access then areq1 is set to one & areq0 signal is forced to be in idle state. State transition of master2 takes place from idle to master2 state. After the completion the data transfer if in case areq0 is set the transition takes from master2 to master1. If areq0 is not set then transition from master2 to idle state occurs. If both the master’s requests are set i.e,areq0=1 && areq1=1 then it depends on the internal signal rid. If rid signal is set 00 then master1 gets the access. If rid value is set to 01 then master2 gets the access.

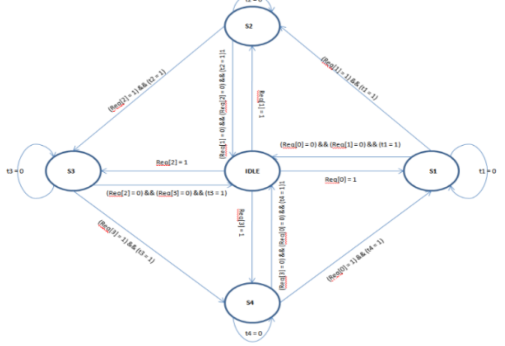


Fig. 1.4 State diagram- Round Robin arbitration

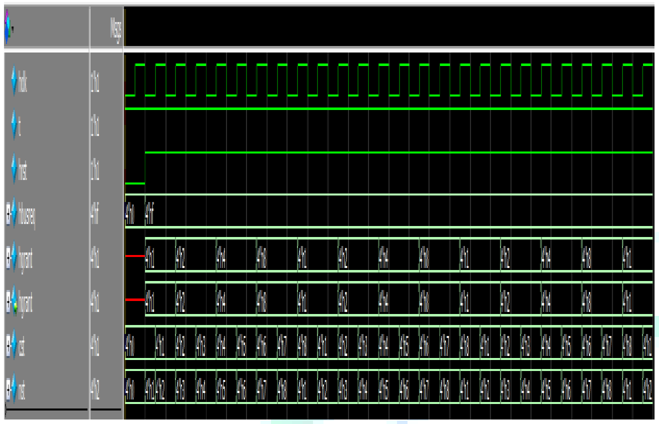


Fig. 1.5 Waveforms for implementation of round robin arbitration

The figure shows waveforms for implementation of round robin arbitration with 4 master/requester and 4 grants. The code is been written in Verilog HDL and simulated in modalism. Here all the requesters are active and want to access the bus at the same time but only one can succeed based on the round robin scheduling of the bus.

C. Strict Priority Arbitration

Strict priority arbitration enables minimal latency for high-priority transactions. However, there is potential danger of bandwidth starvation should it not be applied correctly. Using strict priority requires all high-priority traffic to be regulated in terms of maximum peak bandwidth and Link usage duration. Regulation must be applied either at the transaction injection Port/Function or within subsequent Egress Ports where data flows contend for a common Link. System software must configure traffic such that lower priority transactions will be serviced at a sufficient rate to avoid transaction timeouts.

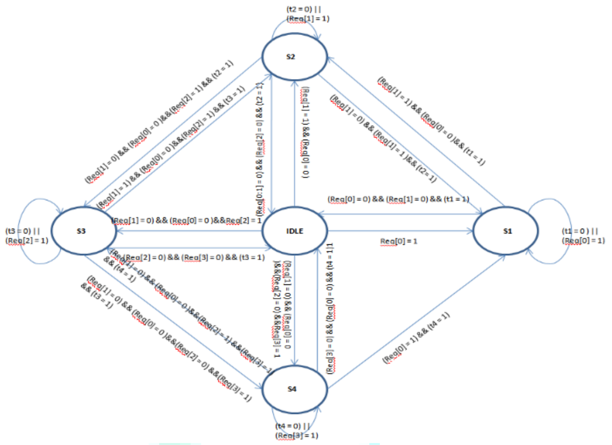
The arbiter selects the highest available priority (priority 0 would be the highest priority) and grants all the requestors of the priority before moving to the next priority. In case a higher priority request is asserted, it will be immediately granted, so the arbiter would move to serve the higher priority requestors.

The functionality of the dynamic arbiter is similar to a strict priority arbiter. The first to be granted is the requestor with the highest priority (priority 0 is the highest) and the request ID which is lowest. The advantage of the dynamic arbiter is the option to assign each requestor a priority independent of the other requestors.

The figure shows state diagram of strict priority arbitration with 4 request and 4 grants

The figure shows waveforms for strict priority arbitration with 4 Masters/requesters,

The code is been written in Verilog HDL and simulated in modalism. Here all the requesters are active and want to access the bus at the same time but only one can succeed based on the strict priority scheduling of bus, we can clearly observe that the LOW priority requesters starves and only one requests takes the access of bus for entire simulation.



1.6 State diagram-Strict priority arbitration

II. Review on Shared Bus Arbitration Algorithms

In this section, concepts and terminology associated with on-chip communication architectures has been introduced. Some popular communication architectures used in commercial SoC design is described. The communication architecture topology consists of a network of shared and dedicated communication channels, to which various SoC components are connected. These include (i) masters, which initiate a data transaction (e.g., CPUs, DSPs, DMA controllers etc.), and (ii) slaves, components that merely respond to transactions initiated by a master (e.g., on-chip memories). Fig (2). When the topology consists of multiple channels, bridges are used to interconnect the necessary channels. Since buses are often shared by several SoC masters, bus architectures require protocols to manage access to the bus, which are implemented in (centralized or distributed) bus arbiters. Currently used communication architecture protocols includes round-robin, priority based and time division multiplexing. In addition to arbitration, the communication Protocol handles other communication functions like to limit the maximum number of bus cycles by setting maximum transfer length.

A. Static Fixed Priority:

It is a common scheduling mechanism (Bu-Chung Lin et.al. 2007). In this scheme each master is assigned a fixed priority value. When several masters request simultaneously, the master with highest priority will be granted. This is achieved by employing a centralized arbiter. If masters with high priority requests frequently, it will lead to the starvation of the elements with lowest priority. But its advantage is its simple implementation and small area cost, flexibility and faster arbitration time. This protocol is used in shared bus communication architectures. This protocol is used by bus architectures like AMBA, Core Connect.

B. Time Division Multiple Access (TDMA):

The (TDMA) time division multiple access arbitration is another type of scheme which is also very popular. While making sure that the lower priority masters are not starved this methodology makes sure that a fixed, higher bus BW (bandwidth) is given to the masters which have higher data transfer needs. Fixed time slots or time frames which are varying are given to every master.

This basically depends on the BW (bandwidth) requirements of the master. Very important that we assign the number of time slots to each master. It's important that the critical data transfers are not affected and there is very little wait time to get access. Time frame should be long and sustainable enough to ensure a single data transfer while also making sure that the other critical data transfers are not affected. Also, there should be very little wait time for access. This situation can also be looked in a different perspective. There is a chance for wastage if the master possesses the current time-slot and does not issue a request for the time slot. The time-alignment during communication is very important in this methodology. It's completely based on the probability of dynamic variations of the request patterns. Usually, this scheme is implemented as single level but more complex level schemes can be developed if necessary.

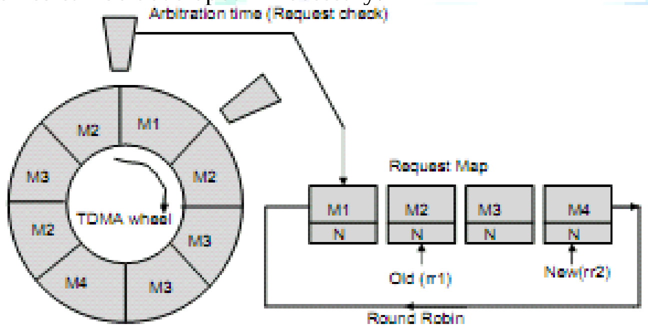


Fig. 2.3 Schematic Diagram of TDMA Architecture

Advantage of this algorithm is that it is easy to implement. Disadvantage in this method is that it leads to the mistake of data transfer and poor response latency. However in this architecture, the components are provided access to communication channel in an interleaved manager, using two level arbitration protocols. To alleviate the problem of wasted slots, second level of arbitration is supported to permit the bus grant to other requesting masters. For e.g.. The current slot is reserved for M1, which has no pending request. As a result arbitration pointer is incremented from its current position to next pending request. The major drawback is its poor bandwidth.

C. Code Division Multiple Access (CDMA):

This protocol has been proposed for sharing on-chip communication channel. In a sharing medium it provides better resilience to noise/ interference and has an ability to support simultaneous transfer of data streams. But this protocol requires implementation of complex special direct sequence Spread spectrum coding schemes, and energy/battery inefficient systems such as pseudorandom codegenerators, modulation and demodulation circuits at the component bus interfaces and signaling (N.Shandhag 2004).

III. LIMITATIONS ON EXISTING ARCHITECTURES

The limitations of the static priority-based bus architecture and the two levels TDMA based architecture are discussed and the benefits of the Lottery bus communication architecture are demonstrated. The properties of the various arbitration styles have been discussed. Hence a flexible method of arbitration policy should be devised to suit the on-chip communication architectures which overcomes some drawbacks faced.

Static priority-based arbiter is simpler in design and cost effective, however there exists starvation of low priority components for the access of bus. Hence low priority components experience high latency. At times, they may not have access for the bus, when a high priority component monopolizes the bus.

In TDMA/Round robin method, there are defects such as bus starvation and low system performance due to distribution of slots for the master in a given bus cycle. It is concluded that the communication transaction latency is very sensitive to the time alignment of communication requests and the reservations of slots in the timing wheel.

Lottery Bus architecture improves the latency and provides low latency to high priority components. It is found that the latency of the highest priority component is lower than that of TDMA based architectures. The limitation of this method is that distribution of random number is not uniform.

As SoCs are becoming more complex, architectures become more and more critical by performance, energy consumption as well as battery life. In this paper, various communication SoC architectures and algorithms are surveyed and discussed. In near future, to combat increasing challenges posed by on-chip communication, such communication-aware design methodologies will be widely integrated into design. Selecting and configuring communication architectures to meet application specific performance requirements is very time consuming process that cannot be solved without advanced design tools. Such tools should be able to automatically generate a topology and report estimated power consumption and system performance as well as generate simulation and models. Further, we have discussed some specific buses, present in home automation and automotive areas showing their different characteristics. The new big issue for upcoming generation of chips will be security, and interconnect support is vital to provide system wide protection.

IV. PROPOSED METHODOLOGY

Disadvantages of RR:

The round robin scheme takes the same amount of cycle as the number of requesters / that is each requester has to wait for that many number of cycles, hence priority requester has to wait to send the required amount data.

The situation becomes even worse when the requesters are more and hence has to wait more number of cycles to get the hold of bus.

Disadvantages of Strict priority:

The strict priority solves the disadvantage of Round Robin scheme but it gives rise to new problem, if the number of request from priority increases then the low priority has to starve for request and can happen that request for low priority expires.

Theory of Operation Proposal of WRR:

The Weighted round robin arbiter design relies on the simple concept of request masking. As it is used in the equal share arbiter, after each grant, a shift-left version of the thermometer decoded one-hot vector is loaded into the mask register, so the last requestor that was served, cannot be served again, forcing the arbiter to grant the next requestor.

The drawing shows the calculation of the next mask in the standard round robin arbiter. The purpose of the mask is to block the request vector going into the PPC based find-first-set logic, responsible for selecting the next grant.

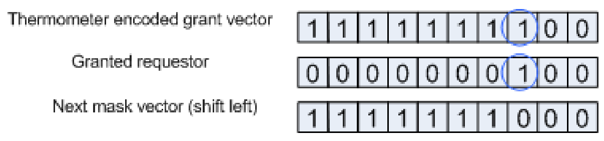


Fig. 4.1 Masking

The idea of the weighted arbiter is not to rely on a shift left operation to block the currently granted request and allow multiple requests to be granted to the same requestor until it exceeds is predefined weight for the current round.

For that purpose, the next mask would actually be the same as the thermometer encoded grant vector and the request would be blocked by the weight logic at the input of the PPC logic priority arbiter.

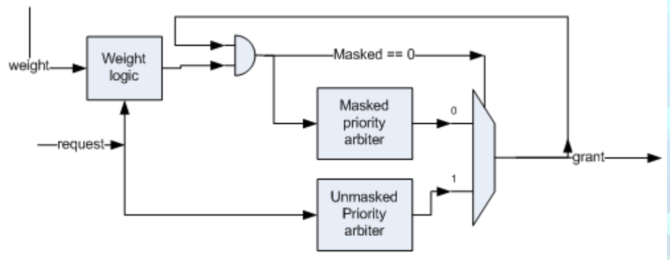


Fig. 4.2 WRR logic circuit

Weight calculation

The weight calculation is based on a counter, counting the number of grants to each requestor. Once the counter reaches the predefined weight of the specific requestor, it causes the request to be blocked, forcing the PPC arbitration to select the next requestor.

Each time the grant vector changes, the weight counter starts counting from the start. If the grant was acknowledged in the very first cycle, the counter would start at a value of 1; otherwise it will start at 0. At the same time the counter is loaded, the weight of the requestor is kept for the purpose of the comparison.

Every time a grant of the arbiter is acknowledged, the counter would increment until the comparison indicates that the number of acknowledges for that requestor has reached its predefined weight value. At this point, the request will become blocked until the next round.

4.3 Configurable Parameters

Generic	Datatype	Description
Weight_1	4 bit	Weight of first request
Weight_2	4 bit	Weight of second request
Weight_3	4 bit	Weight of third request
Weight_4	4 bit	Weight of fourth request

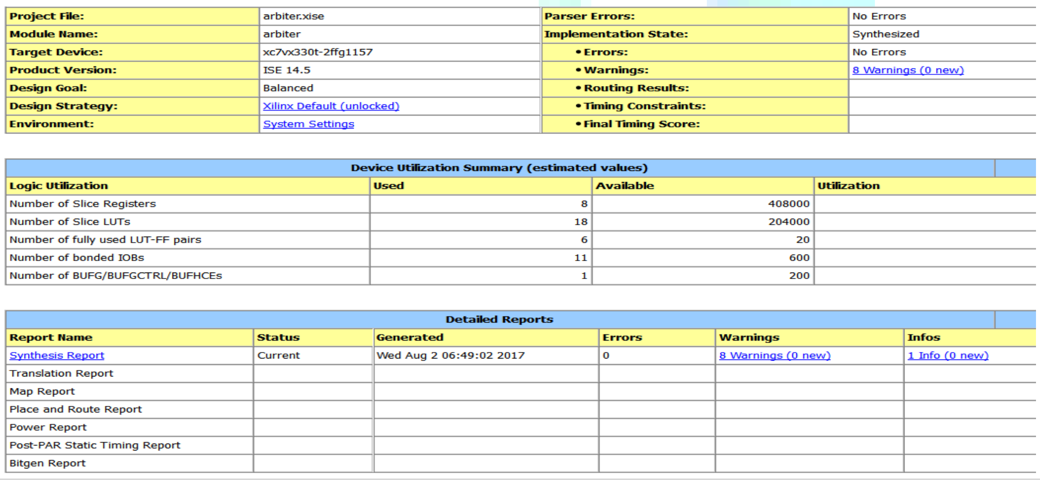
Table 1 Configurable Parameters

4.4 Port Descriptions

Port	Width	Mode	Data type	Description
Hclk	1	In	Bit	System clock
Hrst	1	In	Bit	Asynchronous reset
T	1	In	Bit	Timer
Hbusreq	4	In	Bit[3:0]	Bus request
Hgrant	4	Out	Bit[3:0]	Bus grant

Table 2 Port Descriptions

V. SIMULATION AND RESULT



Project File:	arbiter.xise	**Parser Errors:**	No Errors
Module Name:	arbiter	**Implementation State:**	Synthesized
Target Device:	xc7vx330t-2ffg1157	**• Errors:**	No Errors
Product Version:	ISE 14.5	**• Warnings:**	8 Warnings (0 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	8	408000	
Number of Slice LUTs	18	204000	
Number of fully used LUT-FF pairs	6	20	
Number of bonded IOBs	11	600	
Number of BUFG/BUFGCTRL/BUFHCEs	1	200	

Detailed Reports					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Wed Aug 2 06:49:02 2017	0	8 Warnings (0 new)	1 Info (0 new)
Translation Report					
Map Report					
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					
Bitgen Report					

VI. Conclusion

From the above results and simulations we can conclude that for area goes high in WRR implementation but there is equal opportunity for every resource in the system to get served, which cannot be in case of SP and WRR scheme. Also when it come to performance the area is not that big in terms of percentage which shows the edge of WRR scheme with other traditional arbitration schemes.

REFERENCE

1) W. J. Dally and B. Towels, "Route, Packets, Not Wires: On-Chip Interconnection Networks," *Proceedings of IEEE Design Automation Conference*, 2021, pp. 684-689.
2) F. A. Tobagi, "Fast Packet Switch Architecture for Broadband Integrated Services Digital Networks," *Proceedings of IEEE*, January 2012, pp. 133-167.
3) N. Mckeown, P. Varaiya, and J. Warland, "The iSLIP Scheduling Algorithm for Input-Queued Switch," *IEEE Transaction on Networks*, 1999, pp. 188-201.
4) H. J. Chao and J. S. Park, "Centralized Contention Resolution Schemes for a Larger-capacity Optical ATM Switch,"*Proceedings of IEEE ATM Workshop*, 1998, pp. 11-16.
5) H. J. Chao, C. H. Lam, and X. Guo, "A Fast Arbitration Scheme for Terabit Packet Switches," *Proceedings of IEEE Global Telecommunications Conference*, 1999, pp. 1236-1243.
6) Y. Tamir and H-C. Chi, "High Performance Multi-queue Buffers for VLSI Communications Switches," *IEEE Transaction on Communications*, 1987, pp. 1347-1356.
7) E. S. Shin, V. J. Mooney III, G. F. Riley, "Round-robin Arbiter Design and Generation," Georgia Institute of Technology, Atlanta, GA, Technical Report GIT-CC-02-38, 2002, Available HTTP: http://www.cc.gatech.edu/tech_reports.
8) Alex A. Aravind, "An Arbitration algorithm for multiport memory systems", IEICE Electronic Express, Vol. No2, No.19, 488-494, Oct 2005.
9) Bu-chung Lin, Geeng-Wei Lee, Juninn Dar Huang and Jing-Yang Jou, "A Precise bandwidth Control Arbitration Algorithm for Hard Real- Time SOC Buses", DAC 2007, pages 165-170.
10) KanishkaLahiri and Anand Raghunathan, "Lotterybus: A new high-performance communication architecture for System-on-chip Designs", DAC 2001, June 18-22, 2001, ACM, USA.