ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Implementation of Virtual Channel in Network on Chip Router

Mohammad Shamshad Alam
Department of Electronics and Communication Engineering,
Nimra College of Engineering,
Ibrahimpatnam, Vijayawada, A.P-521456, India.

Md Shamshad Begum
Department of Electronics and Communication Engineering,
Nimra College of Engineering,
Ibrahimpatnam, Vijayawada, A.P-521456, India.

Abstract:

The on-chip communication requirements of many systems are best served through the deployment of a regular chip-wide network. This paper presents the design of a low-latency on-chip network router for such applications. We remove control overheads (routing and arbitration logic) from the critical path in order to minimize cycle-time and latency. Simulations illustrate that dramatic cycle time improvements are possible without compromising router efficiency. Furthermore, these reductions permit flits to be routed in a single cycle, maximizing the effectiveness of the router's limited buffering resources.

Introduction

The ability to fully exploit modern fabrication technologies is tempered by both physical and logical design complexity. The cost of this complexity suggests the reuse of design and verification effort wherever possible. This is often achieved by composing systems from commodity IP or by reusing custom blocks repeatedly in the same design. The relatively poor scaling of global interconnects and the need to achieve architectural performance gains in an energy-efficient manner, provide pressure to decentralise computation. Together these trends suggest a move towards an increasingly communication-centric view of processor and system architecture [16, 21, 15, 14]. One proposed solution to the problem of chip-wide communication is a network of top-level point-to-point communication channels [1, 8, 12] (See Figure 1). This highly regular wiring strategy aims to reuse a small number of highly optimised wiring layout and driver designs. As channel layouts are reused to create the network, effort in characterising delay, power and verifying signal integrity is minimised. The simple behaviour of the network also aids in predicting performance and ensuring correctness. In contrast, large bus based communication networks presenta complex verification task at every level. In addition, the limited ability to scale interconnect delays makes the presence of long global wires and buses increasingly undesirable Similar observations have already been made in the case of inter-chip and wider-area communication. While much of this work is applicable, some important differences exist [8]. In particular, on-chip designs exploit a far greater number of pins and wires, while inter-chip designs are often pin limited. In addition, while inter-chip router designs may exploit a large number of buffers, on-chip designs must aim to minimise buffer count in order to maximise the silicon real-estate available for computation. Area pressures, together with the need to minimise on-chip communication latencies, suggest the implementation of relatively simple on-chip routers.

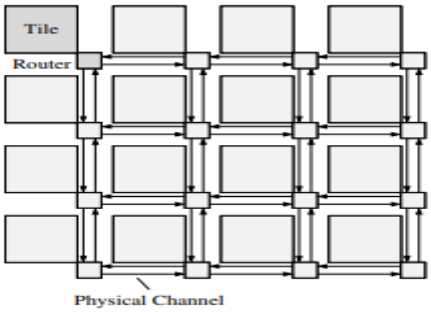


Figure 1. On-Chip Network. Each tile may contain identical logic, as in the case of a multiprocessor or tiled system, or simply represent a partitioning of a SoC design.

Cite this article as: Mohammad Shamshad Alam & Md Shamshad Begum, "Implementation of Virtual Channel in Network on Chip Router", International Journal & Magazine of Engineering, Technology, Management and Research, Volume 6 Issue 11, 2019, Page 50-56.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

This paper describes how router latency may be significantly reduced by hiding control overheads. Thecreation of a single-cycle architecture also reduces latency and maximises the impact of limited buffering resources. Simulation results illustrate that while these techniques offer dramatic cycle time reductions, they do not compromise router efficiency. Initial circuit-level simulations suggest a router cycle time of 12-FO4 delays1 plus clock overhead is possible. Previously published delay models have suggested similar router designs require three pipeline stages and a clock cycle-time of 20-FO4 delays.

Background

A network may be characterised by its topology, routing strategy and method of flow-control [5]. For simplicity we assume a mesh network (with bidirectional links) together with dimension-ordered (XY) routing2. The choice of flow control technique is guided by the need to minimise buffer requirements and latency in our on-chip network. Schemes that reserve buffer space or apply flow-control at the packet level, such as store-and-forward [20] or virtual-cut through [13], are unsuitable for these reasons. A wormhole-router provides the necessary fine-grained flow control, while the addition of virtual-channels [4, 6] aids in boosting performance and circumventing message-dependent deadlock. Furthermore, Quality-of-Service (QoS) enhancements are possible by prioritising the allocation of virtual-channels and switch bandwidth. The remainder of this section provides an overview of the architecture of a generic virtual-channel router.

Overview of a Virtual-Channel Router

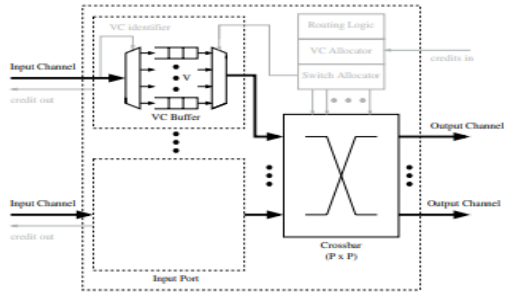


Figure 2. Virtual-Channel Router

Figure 2 illustrates the major components of a generic virtual-channel router. The router has P input ports and P output ports, supporting V virtual-channels (VCs) per port. Virtual-channel flow control exploits an array of buffers at each input port. By allocating different packets to eachof these buffers, flits3 from multiple packets may be sent in an interleaved manner over a single physical channel. This improves both throughput and latency by allowing blocked packets to be bypassed.

1. Routing. The first flit of a new packet arrives at the router. The routing field is examined and a set of valid output virtual-channels upon which the packet can be routed is produced. The number of output VCs produced by the routing logic will depend on the routing function. Possibilities range from a single output VC to a number of different VCs potentially at different physical channels (i.e. adaptive routing). The selection of an output VC can also be influenced by the class of the packet to be routed. Packets from particular classes will often be restricted to travelling on a subset of virtual-channels to avoid message-dependent deadlock. A common practise is to provide separate request and reply virtual-networks.

2. Virtual-Channel Allocation. An attempt is made to allocate an unused VC to the new packet. A request is made for one of the virtual-channels returned by the routing function. Allocation involves arbitrating between all those packets requesting the same output VC.

3. Switch Allocation. Each packet maintains state indicating the availability of buffer space at their assigned output VC. When flits are waiting to be sent, and buffer space is available, an input VC will requestaccess to the necessary output channel via the router's crossbar. On each cycle the switch allocation logic matches these requests to output ports, generating the required crossbar control signals.

4. Crossbar Traversal. Flits that have been granted passage on the crossbar are passed to the appropriate output channel.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Input Buffer and Bypass

Each new incoming flit is stored in the VC buffer designated by its VC identifier. This identifier is appended to every flit in the previous router stage. If the VC buffer is empty and the flit is able to access the crossbar immediately, a bypass path is required to expedite its journey.

Routing Logic

In order for virtual-channel and switch allocation to take place the routing function must first be evaluated to determine which virtual-channel(s) at which output port(s) the packet may request. To ensure that this computation does not lie on the router's critical path, the computation may be performed in the previous router in preparation for use in the next. The idea that the route may be calculated one step ahead of where it is required was first employed by the SGI routing chip [10] and is known as look-ahead routing.

Virtual-Channel Allocation

Peh and Dally detail the complexity of both virtual-channel (VC) allocation and switch-allocation logic in [19]. The following two sections provide a brief overview of these schemes. The complexity of VC allocation is dependent on the range of the routing function. In the simplest case, where the routing function returns a single VC, the allocation process simply consists of a single arbiter for each output VC. As any of the input VCs may request any output VC, each arbiter must support PV inputs. If the router function returns multiple output VCs restricted to a single physical channel, an additional arbitration stage is required to reduce the number of requests from each input VC to one. The winning request at each virtual channel buffer then proceeds to the second stage as described above. The complexity of such a scheme is illustrated in Figure 3. The routing function determines the output port and VCs that may be requested prior toVC allocation. A VC which is free to be allocated is then selected by the first stage of arbitration. The result of this first stage of arbitration is a request for a single VC at a particular output port. This request is subsequently sent to the appropriate second stage arbiter. While this scheme does not guarantee to allocate all free output VCs to potential waiting input VCs in a single cycle, there is no performance penalty as only one flit may be sent per cycle on an output channel. In the most general case where the routing channel may return any of PV VCs, the number of inputs to the first stage of arbiters must now be increased from V to PV. In this case some performance degradation may be expected as the scheme makes little effort to perform a good matching of requests to free output VCs.

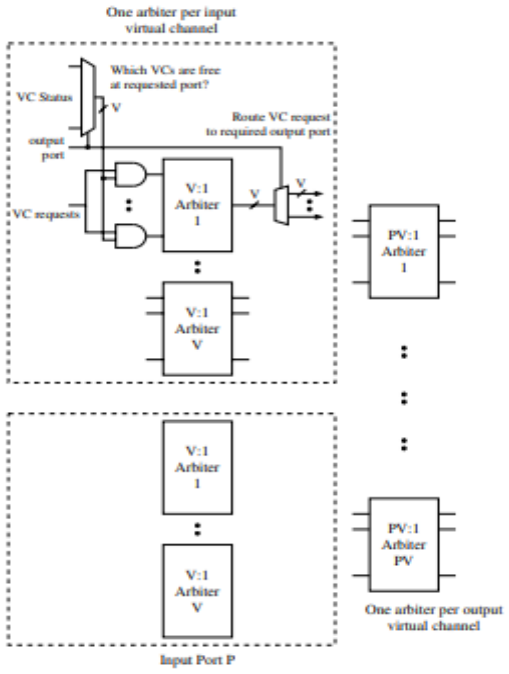


Figure 3. Arbitration complexity of a virtual channel allocator. In this example, the routing function returns a single output port and one or more VC requests.

Switch Allocate

Individual flits arbitrate for access to physical channels via the crossbar on each cycle. Arbitration may be performed in two stages [19]. The first reflects the sharing of a single crossbar port by V input virtual-channels, this requires a V-input arbiter for each input port. The secstage must arbitrate between winning requests from each input port (P inputs) for each output channel. The scheme is illustrated in Figure 4. The request for a particular output port is routed from the VC which wins the first stage of arbitration. In order to

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

improve fairness, the state of the V-input arbiter is only updated if the request is also successful in the second stage of arbitration. We assume this organization wherever multiple stages of arbitration are present. This switch allocator organization may reduce the number of requests for different output ports in the first stage of arbitration, resulting in some wasted switch bandwidth.

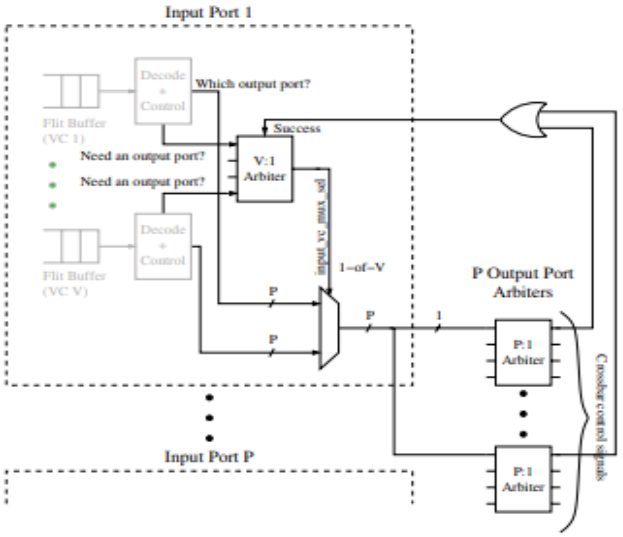


Figure 4. Switch Allocation Logic

Speculative Switch Arbitration

Virtual-channel flow control as discussed performs VC allocation and switch allocation sequentially. This guarantees that only packets that have successfully obtained an output VC from the VC allocator can make requests for their desired output channel. Peh and Dally [19] describe how this dependency may be relaxed if we speculate that a waiting packet will successfully be allocated an output VC. In this way both VC and switch allocation can be performed in parallel. To avoid a negative impact on performance the switch allocator in the speculative design must prioritise non-speculative requests over speculative ones. This is achieved by implementing two switch allocators, one handling speculative requests (from packets that arealso requesting a VC be to allocated) and another for non-speculative requests (from packets which have already been allocated a VC). Only when no non-speculative requests are granted for a particular output port are successful speculative requests granted. In the case that a speculative request is granted we must ensure that the VC has been allocated and it is capable of receiving a new flit (has free buffer space) before the flit is actually sent. Fortunately, such checks may be performed in parallel with crossbar traversal.

Crossbar

In the architecture illustrated in Figure 2 each input port is forced to share a single crossbar port even when multiple flits could be sent from different virtual-channel buffers. This restriction allows the crossbar size to be kept small and independent of the number of virtual-channels. Dally [6] and Chien [2] suggest that providing a single crossbar input for each physical input port will have little impact on performance as the data rate out of each input port is limited by its input bandwidth. While simulation results indicate some advantage in providing larger crossbars (see Figure 8) this is often unrealistic as crossbar implementations scale very poorly. A more effective use of area may simply be to increase the size or number of VC buffers.

The Free Virtual Channel Queue

The first stage of arbitration in the virtual-channel allocator ensures each VC makes a single request for a output VC. The requests are generated as a product of the routing function and a VC status mask, indicating the availability of free VCs at a particular output port. An alternative is to simply queue free VC identifiers and provide a mask with a single bit set (indicating the free VC at the head of the queue), thus avoiding the need to arbitrate between multiple free VCs. A separate queue is provided for each output port and for each virtual-network (traffic-class), e.g. two queues per output port to provide request and reply networks. The scheme effectively removes the need for arbitration by predetermining the order of grants.

VC Allocation Logic (V:1) Arbiters

Requests made for the same output VC from the same input port aarbitrated by P groups of V-input arbiters at each output port. Grant-enable signals are precompiled regardless of the state of the VC (whether it is free or

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

not). This is safe as each arbiter is dedicated to a particular output VC and requests will only be made if the VC is free. In the case where no requests are made, all the grant-enable signals for the arbiter may be asserted. This environment is safe since at most one new flit may be received per cycle at one input port.

VC Allocation Logic (P:1) Arbiters

These arbiters face the same problem as the second stage P-input arbiters in the speculative switch allocator. If no request is present on the preceding clock cycle it cannot easily be determined from which input port the next flit will be received. Again we may proceed by asserting all grant-enable signals and aborting granted operations in the case that two or more requests are subsequently received. Note that the reorganisation of the monolithic PV-input arbiters as a tree arbiter simplifies the precipitation of grant signals.

Analysis of Dependencies/ Critical Path

Illustrates the dependencies within our optimized router design. Virtual-channel flit FIFOs are assumed to be able to receive a flit in one clock cycle ready for use in the next. The case where the flit is needed on the same cycle is handled by a bypass. The fast allocator is used to generate VC and switch grant signals from the precompiled grant enables. The presence of precompiled grant-enables at the start of the clock cycle means that the logic required to generate the crossbar and crossbar input multiplexer control signals becomes trivial. Cases where the fast allocator produces invalid control signals are quickly detected and the associated operations aborted (in these cases valid control signals are guaranteed to be generated on the next clock cycle). The permitted grants and existing requests are then used to calculate the request signals guaranteed to be present on the next cycle (of course new requests may also be made as new flits arrive). The permitted grants are also used to update the state of the matrix arbiters. Once the requests present on the next cycle have been computed and updated VC buffer state information is available, grant enables for the next cycle may be computed. One concern is the need to update VC buffer state information prior to precompiling the grant-enable signals for the P:1 non-speculative switch arbiters. One possibility is to recomputed grant-enable signals using the older state before it is updated. Unfortunately, the buffer state of multiple output VCs assigned to VCs at a single input port may be updated in a single cycle. This prevents us from setting all grant-enable signals safely. Although this could be done if we are able to abort grants in the case that two or more requests are subsequently received. In the simulations that follow we assume that this dependency is not on the router's critical path and may be tolerated. In our implementation we have adopted a simple on/off channel flow control mechanism which simplifies the logic needed to maintain the buffer state. Such a scheme would be less desirable if the router did not operate in a single cycle. Initial results from preliminary extracted layout (180nm technology) suggest that the design will operate at our target cycle time of 12 FO4 delays plus clock overhead. This is approximately twice the tile frequency in our planned system. In our test network each flit carries 64-bits of data and routers are placed 1mm apart. All signal transitions (in each output channel and the crossbar) are in the same direction during evaluation avoiding worse-case crosstalk. Typical case communication delays between routers are within 2 FO4 delays. Inter-wire capacitance values for communication channels were calculated using Quick Cap [11]. The precipitation of grant-enable signals is essential in meeting our cycle time. Our best 5-input matrix-arbiter designs have a typical latency of approximately 3 FO4 delays. The complete control logic takes the majority of the clock cycle in the optimized design, although almost none of this is now on the critical path.

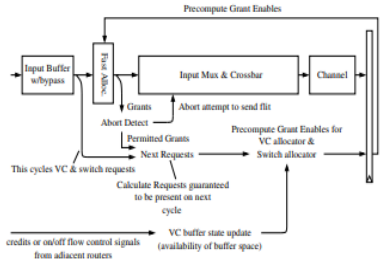


Figure 5. Data dependencies within our single-cycle optimized router architecture

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Simulation Results

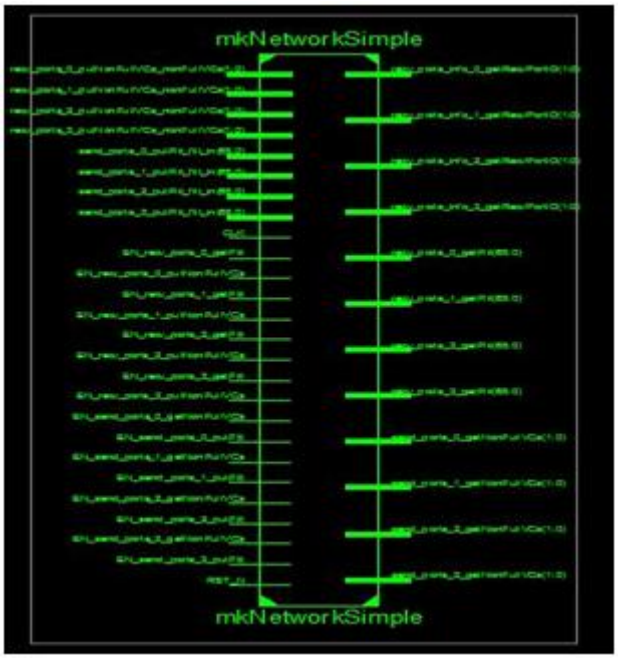
A parameterized network model was constructed using HASE (Hierarchical Architectural Simulation Environment) [3]. The underlying simulation system is multi-threaded and event-driven. Each tile or node generates packets with random destinations. Packets aregenerated at a constant rate and queued until they are able to enter the network. The interval between the creation of individual packets is random (geometric distribution) to prevent packets being injected into the network synchronously.

Network latency is measured from the time the first flit is created to the time the last flit in the packet is received at its destination, including any time spent buffered at the source node. Each node injects 1000 packets into the network and performance statistics are gathered after an initial warm-up period of 100 packets/node. The network is an 8x8 mesh, each router has 5 input and 5 output ports. Packets are 5 flits in length. In all simulations we assume a single cycle router implementation.

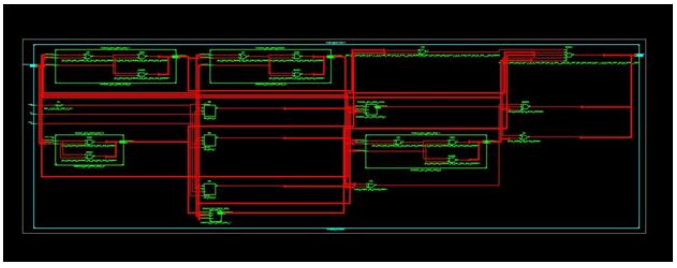
For a range of buffer sizes and virtual-channel configurations. An initial inspection of the results shows that all but the Parallel-NoSpec model have very similar performance characteristics. At closer inspection and perhaps surprisingly the Sequential scheme does not necessary outperform the parallel schemes. This behavior is the result of two effects. Firstly, the speculative switch allocator prioritizes packets during switch allocation that have held a VC for at least one cycle. This can be modelled in the sequential case, slightly improving performance. Secondly, in the case of the speculative allocator two requests from each input port may be considered after the first stage of arbitration.

This potentially increases the chance of finding a more complete matching of waiting flits and ready output ports. Performance could be potentially improved further in the parallel schemes by ensuring speculative requests are only made if at least one free VC is available at the required output port.

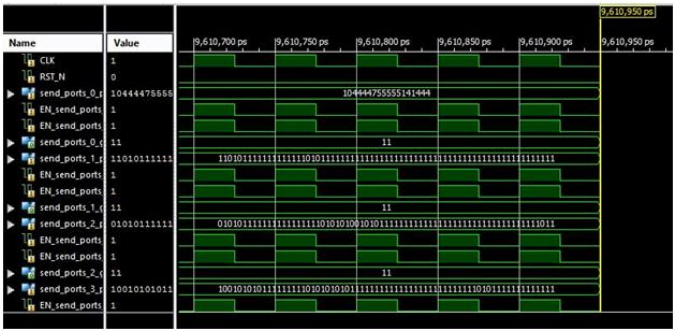
RTL DIAGRAM



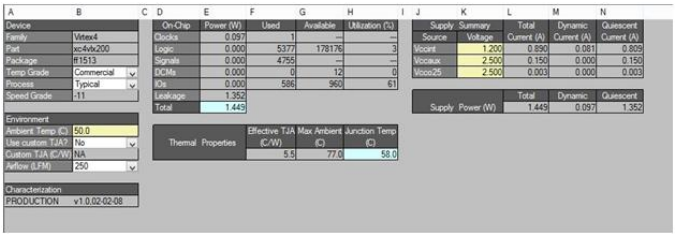
INTERNAL BLOCK DIAGRAM



Simulation results



Power



ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Area

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	792	178,176	1%	
Number of 4 input LUTs	5,105	178,176	2%	
Number of occupied Slices	2,873	89,088	3%	
Number of Slices containing only related logic	2,873	2,873	100%	
Number of Slices containing unrelated logic	0	2,873	0%	
Total Number of 4 input LUTs	5,377	178,176	3%	
Number used as logic	3,425			
Number used as a route-thru	272			
Number used for Dual Port RAMs	1,680			
Number of bonded IOBs	586	960	61%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Average Fanout of Non-Clock Nets	4.50			

Delay

```
    ----------------------------------------
    Total                     10.466ns (6.391ns logic, 4.075ns route)
                              (61.1% logic, 38.9% route)

=========================================================================


Total REAL time to Xst completion: 31.00 secs
Total CPU time to Xst completion: 30.80 secs
```

CONCLUSION

This paper proposed method to improve the performance of NoC routers. This is approach to significantly reducing the clock cycle of on-chip routers. Simulation results shown that the critical path is reduced significantly without compromising router efficiency by performing these two operations (VC allocation and SA) in parallel. Flip-flop is used in this router are 1074 which are large in numbers as compare to the other routers architecture, but the frequency is maximum so that the network latency is reduced, and performance is increases.

Future Work: Here we saw that the flip- flops are used so much so that area is more utilize. Our future plan for this is to find the best solution for buffer architecture, so that we reduce the number of flip-flops and also improvement in the crossbar switch for fast arbitration.

References

[1] Mostafa S. Sayed, A. Shalaby, M. El-SayedRagab, Victor Goulart, “Congestion Mitigation Using Flexible Router Architecture for Network-on-Chip”2012 IEEE.

[2] L.Rooban, S.Dhananjeyan“Design of Router Architecture Based on Wormhole Switching Mode for NoC” International Journal of Scientific &Engineering Research Volume 3, Issue 3, March-2012.

[3] Anh T. Tran and Bevan M. Baas NoCTweak: a Highly Parameterizable Simulator for Early Exploration of Performance and Energy of Networks On-Chip Technical Report, VLSI Computation Lab, ECE Department, and UC Davis July 2012.

[4] U. Saravanakumar, R. Rangarajan and K. Rajasekar Hardware Implementation of Pipeline Based Router Design for On-Chip Network intact journal on communication technology, December 2012, volume: 03, issue: 04.

[5] Ye Lu, John McCanny, SakirSezer “Generic Low Latency Noc Router Architecture for FPGA Computing Systems “„Journal of IEEE , Page no. 82 – 89, 978-1-4577- 1484-9 , 2011 21st International Conference on Field Programmable Logic and Application IEEEs.

[6] Son Truong Nguyen Shigeru Oyanagi “The Design of On-the-fly Virtual Channel Allocation for Low Cost High Performance On-Chip Routers”2010 IEEE.

[7] Everton A. Carara, Fernando G. Moraes FLOW ORIENTED ROUTING FOR NOCS 978-1-4244-6683-2/10/$26.00 ©2010 IEEE.