International Journal of VLSI System Design and Communication Systems

ISSN 2322-0929
Vol.01,Issue.04,
November-2013,
Pages:201-208

ww.semargroups.org
www.ijvdcs.org

Deadlock Recovery Technique in Bus Enhanced NOC Architecture

BHUKYA YUGANDHAR[1], B. MANASA[2], K. V. VARA PRASAD[3]

[1]PG Scholar, Dept of ECE, Dhruva College of Engineering & Technology, Choutuppal, Nalagonda, AP-INDIA,
E-mail: yugandhar.bhukya@gmail.com.

[2]Asst Prof, Dept of ECE, Dhruva College of Engineering & Technology, Choutuppal, Nalagonda, AP-INDIA.

[3]Asst Prof, Dept of ECE, Dhruva College of Engineering & Technology, Choutuppal, Nalagonda, AP-INDIA.

Abstract: Multiprocessor system on chip is emerging as a new trend for System on chip design but the wire and power design constraints are forcing adoption of new design methodologies. Researchers pursued a scalable solution to this problem i.e. Network on Chip (NOC). Network on chip architecture better supports the integration of SOC consists of on chip packet switched network. Thus the idea is borrowed from large scale multiprocessors and wide area network domain and envisions on chip routers based network. Cores access the network by means of proper interfaces and have their packets forwarded to destination through multichip routing path. In order to implement a competitive NOC architecture, in this paper we implement a parallel router which can support five requests simultaneously. Increase in the speed of processors has led to crucial role of communication in the performance of systems. As a result, routing is taken into consideration as one of the most important subjects of the Network on Chip architecture. Routing algorithms to deadlock avoidance prevent packets route completely based on network traffic condition by means of restricting the route of packets. This action leads to less performance especially in non-uniform traffic patterns. On the other hand True Fully Adoptive Routing algorithm provides routing of packets completely based on traffic condition. However, deadlock detection and recovery mechanisms are needed to handle deadlocks. Use of global bus beside NoC as a parallel supportive environment, provide platform to offer advantages of both features of bus and NoC. Design And Verify the functionality of the “Design and Verification Four Port Router for Network on Chip” IP core using the latest verification methodologies, Hardware Verification Languages and EDA tools and qualify the IP for Synthesis an implementation.90% of ASIC respins are due to functional bugs. In order to avoid the delay and meet the TTM, we use the latest verification methodologies and technologies and accelerate the verification process. The Design and Verification Plan is based on Verilog Hardware Verification Language. The methodology used for Verification is Constraint random coverage driven verification .As per our requirement you can develop the 5 ports or n ports.

Keywords: Network on Chip (NOC), Router, Deadlock.

I. INTRODUCTION

A. Router

A router is a device that forwards data packets across computer networks. Routers perform the data "traffic direction" functions on the Internet. A router is a microprocessor-controlled device that is connected to two or more data lines from different networks. When a data packet comes in on one of the lines, the router reads the address information in the packet to determine its ultimate destination. Then, using information in its routing table, it directs the packet to the next network on its journey. A data packet is typically passed from router to router through the networks of the Internet until it gets to its destination computer. Routers also perform other tasks such as translating the data transmission protocol of the packet to the appropriate protocol of the next network. The most familiar type of routers are home and small office routers that simply pass data, such as web pages and email, between the home computers and the owner's cable or DSL modem, which connects to the Internet(ISP). However more sophisticated routers range from enterprise routers, which connect large business or ISP networks up to the powerful core routers that forward data at high speed along the optical fibre lines of the Internet backbone. Routers may also be used to connect two or more logical groups of computer devices known as subnets, each with a different sub-network address. The subnets addresses recorded in the router do not necessarily map directly to the physical interface connections. The router forwards data packets between incoming and outgoing interface connections.

B. Applications of Router

When multiple routers are used in interconnected networks, the routers exchange information about destination addresses, using a dynamic routing protocol. Each router builds up a table listing the preferred routes between any two systems on the interconnected networks.

A router has interfaces for different physical types of network connections, (such as copper cables, fiber optic, or wireless transmission). It also contains firmware for different networking protocol standards. Each network interface uses this specialized computer software to enable data packets to be forwarded from one protocol transmission system to another. Routers may also be used to connect two or more logical groups of computer devices known as subnets, each with a different sub-network address. The subnets addresses recorded in the router do not necessarily map directly to the physical interface connections. A router has two stages of operation called planes:

1. Control plan
2. Forwarding plane

1. Control plane

A router records a routing table listing what route should be used to forward a data packet, and through which physical interface connection. It does these using internal pre-configured addresses, called static routes. A typical home or small office router showing the ADSL telephone line and Ethernet network cable connections.

2. Forwarding plane

The router forwards data packets between incoming and outgoing interface connections. It routes it to the correct network type using information that the packet header contains. It uses data recorded in the routing table control plane. Routers may provide connectivity within enterprises, between enterprises and the Internet, and between internet service providers (ISPs) networks. The largest routers (such as the Cisco CRS-1 or Juniper T1600) interconnect the various ISPs, or may be used in large enterprise networks. Smaller routers usually provide connectivity for typical home and office networks. Other networking solutions may be provided by a backbone Wireless Distribution System (WDS), which avoids the costs of introducing networking cables into buildings. All sizes of routers may be found inside enterprises. The most powerful routers are usually found in ISPs, academic and research facilities. Large businesses may also need more powerful routers to cope with ever increasing demands of intranet data traffic. A three-layer model is in common use, not all of which need be present in smaller networks.

II. ROUTER CORRECTNESS

In the context of our scheme, a correctly functioning router should ensure that a packet's integrity is maintained within or across routers. If a router guarantees that no flits are dropped and all body flits follow the head flit in a wormhole, then we consider it to be functioning correctly. Our definition of router's correctness is motivated by the fact that even with no guarantees of forward progress, if each router in the network ensures to hold these properties, all data in flight in the network is maintained in a coherent state and correct network state can be restored without any need of data duplication. In this section we describe our approach to ensure router correctness using either formal or runtime verification. Without loss of generality we discuss our ideas for a fairly complex 3-stage pipelined router that is input-queued and that uses virtual channel (VC) flow control, look ahead routing and switch speculation. A detailed schematic of this router is shown in Fig. 2. A router essentially ties together its data path components, such as input buffer, channel and crossbar, with a control plane that consists of input VC control (IVC), route computation unit(RC), VC allocator(VA), switch allocator (SA), output VC control (OVC) and flow control manager.

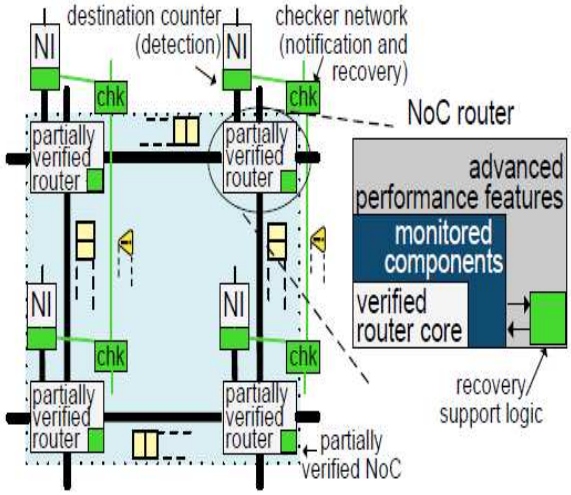

Fig.1. High-level overview of ForEVeR. A combination of router-level runtime monitoring and network-level detection and recovery scheme, along with component formal verification, ensures correct NoC operation

The control plane manages the error free flow of data from input channels to the output channels via input buffers and crossbar respectively. Since the data path components are fairly simple and can be easily verified, we specifically focus our verification effort on controlling logic. Moreover, it is well known that the most complex verification tasks arise from the interaction of concurrent components. In the framework of a router, the interactions between the concurrently operating VCs are handled by RC, VA and SA units. These units utilize information provided by the flow control mechanism, which is used to transmit buffer state information among neighboring routers. Other control units such as IVC and OVC operate mostly on a standalone basis, with information provided by the RC, VA and SA units, and hence can be formally verified using existing formal tools.

Based on the above observations, first a full formal proof of router correctness is attempted. If due to the complexity of the logic involved, formal methods fail to provide correctness guarantees, only parts of the router that can be easily handled by existing formal tools are verified. Then runtime hardware checker are used to protect the vital router components that handle the interactions among concurrent units, which keeps the area

cost low. In addition, provisions have to be made to prevent the NoC from entering an unrecoverable state; for example this might happen when a flit is dropped or corrupted before the monitoring hardware flags an error. In either case, operation of the router during recovery has to be formally verified.

A. Formal Verification

The verification process can be efficiently divided into two sub- goals: ensuring no flits are dropped and proving that body flits follow the header flit in a wormhole. To prove that the router does not drop flits, it is necessary to verify that all valid flits received through input channels are written into valid buffer entries, followed by the verification of first-in first-out functionality of the buffers. Finally, it should be proven that a flit read from the input buffer should get to some output channel, within a fixed number of clock cycles depending on the router pipeline.

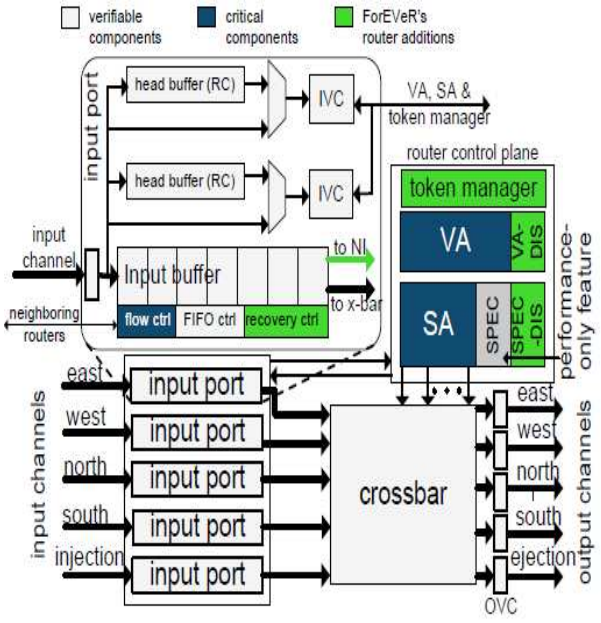

Fig.2. Router modifications in ForEVeR. VA, SA and flow control units are monitored by runtime checkers. To implement recovery, NoC router is augmented with VC and speculation disablers along with a token manager and a recovery FIFO controller

Verifying that a packet maintains its wormhole nature is more involved as now correctness has to be proven over an entire packet rather than a flit. Apart from proving that flits follow the head flit, formal methods should ensure that no other flit from any other packet meddles with the wormhole. Also, it is essential to verify that only valid flits are transmitted and that there is no flit/packet duplication within the router. This requires extensive verification of various router control components and their interactions. Table 1 summarizes the correctness goals for NoC router and the entailed properties that require proof. To illustrate the verification procedure of writing specification properties, we provide an example where we discuss in detail how a correctness property is divided into sub-properties. The property 'incoming valid flits written to IP buffer' is used as an example. This property holds if it can be verified that an incoming valid flit always has a valid VC tag and the corresponding VC buffer has a free slot (no overflow). Additionally, the flit contents should be written to the free slot of only the requested VC buffer. Finally, an incoming invalid flit should not be written to any of the VC buffers. Some router implementations maintain a separate header buffer corresponding to each VC buffer and thus similar properties should be verified for the header buffers, where instead of any valid flit, only valid header flits are considered.

TABLE 1: FORMAL VERIFICATION OF ROUTER CORRECTNESS

correctness goal	verified property	sub	time(s)
No dropped flit (datapath and standalone control units)	* incoming valid flit written to buffer * buffer behaves in FIFO manner * flit gets from buffer to OP channel	6 20 17	90 660 170
Packet keeps wormhole (complex interactions of concurrent components)	* only valid body flits follow head flit with no mixing between packets and no flit/packet duplication	42	2400

The number of sub-properties (sub), expressed as System Verilog Assertions (SVA), to represent each correctness property is also reported in Table 1. Finally, Table 1 states the time taken to verify each property processor running at 2.27 GHz and using 4GB of main memory. It should be noted that formal guarantees for starvation freedom in allocation schemes and proper functioning of the route computation module need not be provided, as the network level detection and recovery scheme efficiently handles these scenarios.

B. Runtime Verification

As mentioned earlier, due to the area overhead of runtime checkers, only components that handle interactions between the concurrently operating modules are monitored. These components are the hardest to verify and result in majority of hard-to-catch bugs. We also pointed out that the routing unit, VC allocator and switch allocator orchestrate the actions of input and output VCs and that the flow control unit interprets and communicates the control information between routers. Among these units, errors in the routing stage are not detrimental to our scheme as long as the other router functionalities are guaranteed to be correct. Thus the routing unit is not monitored at runtime. All other vital units are supervised for correct operation by runtime checkers, as shown in Fig.2. Once an error is flagged by these checkers, router level reconfiguration is performed that forces the router to a formally verifiable degraded mode, with minimum functionality to support network level recovery. This is followed by network level recovery initiation that we discuss in section B of III chapter.

1. Detection and Recovery

VC and switch allocator: A design flaw in VC allocator may give rise to various erroneous conditions, some of which are benign as they either do not violate our definition of router correctness or are effectively detected and recovered by our network level correctness scheme. Assignment of an unreserved but erroneous output VC to an input VC is an example of such a benign error, as in the worst case it may only lead to misrouting or deadlock. Starvation is another example that needs no detection or remedy at a router level. Critical errors arise when an unreserved output VC is assigned to two input VCs or an already reserved output VC is assigned to a requesting input VC. This situation will lead to packet mingling and/or packet/flit loss. Similar to VC allocator, a design flaw in switch allocator may or may not have an adverse affect on For- EVeR's operation. An error in switch allocator may send a data flit to a different direction than the corresponding header flit; it may also cause the same flit to be sent to multiple outputs; or multiple flits from different packets to be directed towards the same output at the same time. All these cases lead to packet data corruption and an un-recoverable network state. To monitor VC and switch allocators at runtime for corrupt behavior, we propose the use of Allocation Comparator (AC) unit that is a stripped down version of a similar unit that was proposed for soft error protection. The AC unit is purely combinational logic that performs all comparisons within one clock cycle. It simultaneously analyses the state of VC and switch allocators for duplicate or invalid assignments. If an error is flagged, all VC and switch allocations of the previous cycle are invalidated. Flits traversing the crossbar just after the error is flagged are discarded at the output. To avoid losing flits due to this invalidation/discard operation, an extra storage slot per VC buffer is reserved for use during such emergencies. To implement this, VA, SA and crossbar units are modified to accept invalidation command from the AC.

Flow control: To safeguard against flow control errors, a hardware monitor is inserted to detect buffer overflow errors. Additionally, to avoid dropped flits, input buffers are equipped with two emergency slots per VC. On receiving a flit at buffer full condition, indicating an overflow, the downstream router tells the upstream router to switch to a slightly modified version of ACK-NACK flow control the second emergency slot is reserved for a possible in-flight flit during this upstream signaling. The modified ACKNACK flow control eliminates the need for negative acknowledgements and re-ordering ability at the downstream router. This is achieved by stopping further transmission on the link until an acknowledgement is received for a previously transmitted flit. The flit awaiting acknowledgement is re-transmitted every two cycles (round trip latency of the links), before being dropped on receiving an acknowledgement. This scheme, though detrimental for performance, is extremely simple and can be implemented with little modification to the existing flow control mechanism. In addition, the router works in this mode only during recovery, switching back to its high performance mode after recovery is complete. Note that to safeguard against all errors at most two emergency slots per VC buffer are required. Since buffers are usually designed as circular FIFOs, this scheme entails only slight modifications to the buffer full logic.

2 Degraded Mode

When a bug is detected by hardware monitors, the router switches to a degraded mode with formally verified execution semantics, by either disabling complex units or replacing vital ones with simpler spare counterparts. This mode is equipped with bare-minimum features to support the network level recovery that is initiated immediately after discovering a bug. To prevent the NoC routers from servicing new packets in probable erroneous state, all packet level operations such as route computation and VC allocation are disabled during recovery, as discussed in section B of III chapter. Similarly advance "performance only" features such as switch speculation and prioritizing mechanisms are disabled. Since stuck packets have to be drained out of NoC routers, it still requires the switch allocator and flow control manager to work properly. To this end, the router reconfigures to use a spare simple arbiter that polls each input VC for switch allocation. Similarly, flow control switches to an acknowledgement based mode to prevent flit loss as discussed in above sub section 1. The resulting degraded router has significantly less concurrency and thus can be verified to function correctly.

III. NETWORK CORRECTNESS

With router correctness guaranteed, we need a network level solution that ensures forward progress in the NoC system. More specifically it should efficiently detect and recover from design errors that inhibit forward progress in the network (deadlock, live lock and starvation) and misrouting errors. To this end, ForEVeR adds a lightweight and verifiable checker network that works concurrently with the original NoC, providing a reliable fabric for transfer of notifications and recovered packets during detection and recovery phases respectively. Our checker network should be a simple, low latency optimized network that can consistently deliver notifications before the actual packets arrive through the primary network. We, therefore, leverage the single cycle latency, packet-switched routers of, organized as a ring network.

In the detection phase, each packet sent on primary network is accompanied by a corresponding notification over the checker network, both directed to the same destination. Each destination maintains a count of expected future packet deliveries through the primary network, decrementing the count on receiving a packet from the primary network. A distributed detection scheme monitors the counter values for zeros, initiating recovery on not observing a zero value during the entire check epoch of certain cycles. During the recovery phase, in-flight packets are recovered from the primary network, and reliably transmitted through the checker network. Fig.1

shows a baseline NoC augmented with the checker network. Interactions between the NoC router and checker network are handled by the NI unit, which also houses the detection and recovery initiation logic.

A. Detection

All design errors that inhibit forward progress result in packet(s) jammed within the network, and thus our detection mechanism should be designed to detect such scenarios. Moreover, it should be simple enough to be implemented with small area overhead and minimal changes to the existing infrastructure. To this end, we use notification messages travelling via the reliable checker network as the means for destinations to keep a count of the future packet deliveries. A bug in the primary network will always lead to an unaccounted packet at the destination, and thus the counter value will never go to zero, under the assumption that notifications always reach the destination before their counterpart packets. Therefore, our distributed detection scheme flags an error if it does not observes a zero counter value at any particular destination inside an entire check epoch. Fig.3 depicts the working of our distributed detection scheme. Counting logic is added to the NI to keep a count of number of expected packets at destination nodes, as shown in Fig.2. A timer monitors the counter value for zeros during entire check epoch length, failing which recovery is triggered. With proper size of the check epoch, this simple scheme is effective in catching bugs as we show in our experimental results and it can be implemented with lightweight counting logic. On the other hand, misrouting errors that do not cause deadlock or live lock are detected at destinations by analyzing the routing information carried by header flit.

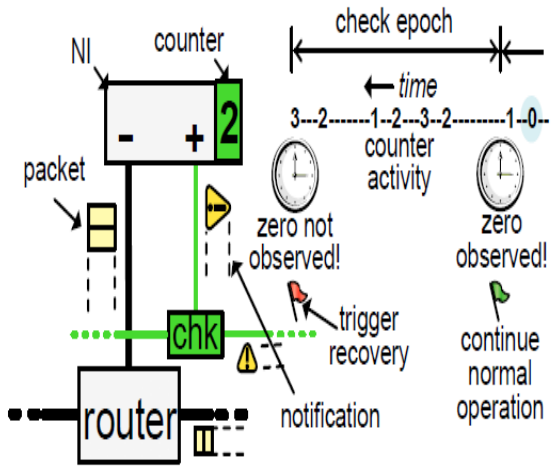

Fig.3. ForEVeR's detection scheme. Each destination tracks the notification counters for zero values. Recovery is triggered if zero is not observed during the entire check epoch at any destination

B. Recovery

When an error is reported either by the router level runtime monitors or by the network level detection scheme, the NoC enters a unified recovery phase, consisting of network drain step followed by a packet recovery step. In network drain phase, the network is allowed to operate normally to drain its in-flight packets for a preset amount of time, with the exception of switching the erroneous routers to a degraded mode if recovery was initiated by router level checkers. During this phase, new packets are not injected into the network, as shown in Fig. 4(a). Recovery is aborted at the end of network drain if all destinations receive the packets they were expecting, indicating a false positive due to the limited accuracy of the detection scheme. It should be noted that false positives, though a performance hit in absence of errors, do not affect the correctness of the system.

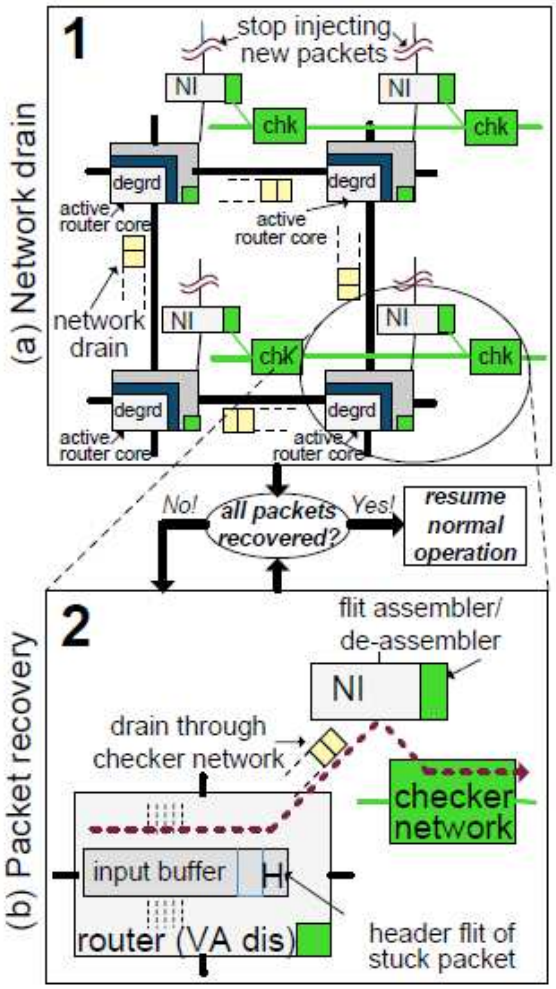

Fig.4. ForEVeR recovery process. Network drain is followed by packet recovery until all primary network packets are recovered

The network then enters packet recovery, where we try to recover packets that are stuck within the network. To this end, a token is circulated through all routers in the NoC via the checker network, and NoC routers can operate only when they hold this token. In addition, VC allocators of all the NoC routers are disabled to prevent them from

processing new data packets from neighboring routers. When a router receives the token, it examines the fronts of its VC buffers in a serial manner, looking for packet headers. In case of a successful search, the packet is retrieved and sent over the checker network as shown in Fig. 4(b). Since vital router functionalities for packet drain are still active (even in the degraded mode), the entire packet can be safely diverted to its destination through the checker network. Once the token has circulated through all primary routers, the entire process of packet recovery is repeated until either each destination receives all the pending packets or no more packets are retrieved, in which case a design bug has slipped through our scheme. To enable the ForEVeR scheme, NoC routers are augmented with certain simple units, as shown in Fig.2. First, a token manager is added to the routers to manage token passing. In addition, virtual channel (VC) allocation disabler (VC-DIS) and switch speculation disabler (SPEC-DIS) are included to prevent routers from processing new packets during the packet recovery phase and to keep the ForEVeR operation simple and easily verifiable for correctness. The recovery operation is implemented with very little overhead, making use of the router's existing functionalities to drain out packets from their buffers, with the help of the FIFO recovery controller.

Due to the limited bandwidth of the checker network, each primary network flit is transmitted as several checker packets. During recovery, only one router is transmitting its stuck packets to a single destination at a time, greatly simplifying the disassembling/assembling process. To send the entire primary network flit as multiple checker packets, the channel of the checker network is augmented with head and tail indicators. The flit with head indicator carries the destination address and reserves an exclusive path between the source and one particular destination. All intermediate valid flits traversing the ring network are ejected at the same destination till a flit is received with a tail indicator, in which case the process repeats itself on transmission of another flit with a head indicator. Moreover, all transmissions on the checker network during recovery occur in the same (clockwise) direction to avoid wormhole overlap of two packets. In our evaluation system with 64 nodes, the checker network channel is 8 bits wide (6-bit address, 2-bit head-tail indicators). Thus each 64-bit primary network flit takes 12 checker networks packets (1 head, 11 body/tail) to transfer.

C. Verification of Recovery Operation

All components involved in the detection and recovery processes must be formally verified to guarantee correct functionality. Verification of the detection mechanism involves ensuring the correct functioning of the counting and timer logic at NIs and due to the simplicity of the logic involved this makes up for a trivial verification task. Formally verifying the recovery operation is more involved and requires two major tasks: first, verifying the checker network functionality; and second, verifying the interaction between checker and primary network during recovery, to ensure proper restoration of erroneous packets.

1. Checker network: It should be verified that the checker network correctly delivers all packets to their respective destinations within a bounded time. To this end, this correctness goal was partitioned into three sub-properties: eventual injection (inj_prop), guaranteeing injection of a waiting packet into the network; forward progress (fw_prop) ensuring that packets progress on a path towards their destination; and timely ejection (ej_prop) that guarantees packet ejection at correct destination.

2. Interaction with primary network: The primary network's units that interact with the checker network to salvage stuck packets from the primary routers must be ensured to function properly. During recovery, primary routers work in a rudimentary mode by disabling all complex hardware units not involved in the recovery process, such as the VC allocators and SWspeculators, thus making the verification task tractable. First, it is verified that the checker network could extract a complete packet from an individual primary router's VC buffer (rec_prop), leaving it empty (rec_emp_prop). The complement of this property is also validated (not_rec_prop) to check that only valid packets are extracted from the primary network. This was followed by checking for fairness and exclusivity among the primary routers while undergoing recovery (fair_ex_prop), ensuring that packets are salvaged from one router at a time.

TABLE 2: FORMAL VERIFICATION OF FOREVER'S RECOVERY OPERATION

correctness goal	verified property	time(sec)
Checker network correctness	inj_prop	8
	fw_prop	156
	ej_prop	86
Interaction with primary network	rec_prop	15
	rec_emp_prop	10
	not_rec_prop	46
	fair_ex_prop	29

Table 2 summarizes the correctness goals for ForEVeR's recovery process and the time required to prove the detailed properties.

IV. SIMULATION RESULTS

The work consists of simulation of bus, detection mechanism and recovery by means of forwarding flits of deadlocked message on the bus. For evaluation of the results, we use Noxim simulator. These results are based on 2-Dimension 4x4 mesh topology. The packets length is between 4 to 10 flits. It is simulated under non-uniform traffic loads including First Matrix Transpose, Butterfly and Bit Reversal. We compare our deadlock recovery technique - true fully adaptive recovery with bus - with three International Journal of VLSI design &

Communication Systems (VLSICS) Vol.3, No.4, August 2012 7 different routing algorithms XY, Minimal West First and Odd-Even. As shown in figure 4, we compared average of packets latency and throughput metrics with increasing packet injection rate. Our proposed deadlock recovery mechanism with increase packets injection rate keep the value of latency of packets in lower than other routing algorithms in First Matrix Transpose traffic pattern. And with better use of routes between sources and destinations the average of throughput is increased, compared to other routing algorithms the average of Packet Latency and Throughput in First Matrix Transpose traffic pattern. However in figure 5, we compared the latency of packets in Butterfly and Bit Reversal traffic patterns. According to lower amounts of average latency, our proposed mechanism has provided more increase, packet injection rate, as compared to other routing algorithms. (a) Average of Packet Latency in Bit Reversal traffic pattern. (b) Average of Packet Latency in Butterfly traffic pattern. Figure 6 depicts the Block Diagram of FSMTB Simulation Waves.

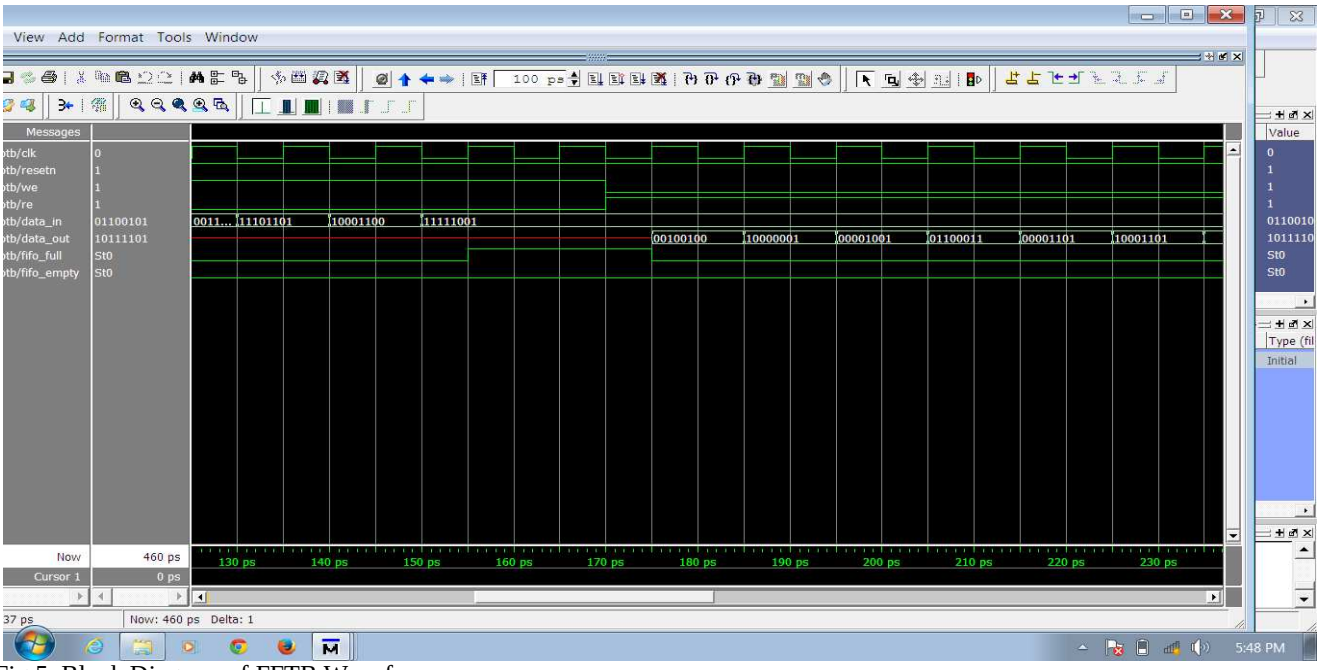

Fig.5. Block Diagram of FFTB Waveform

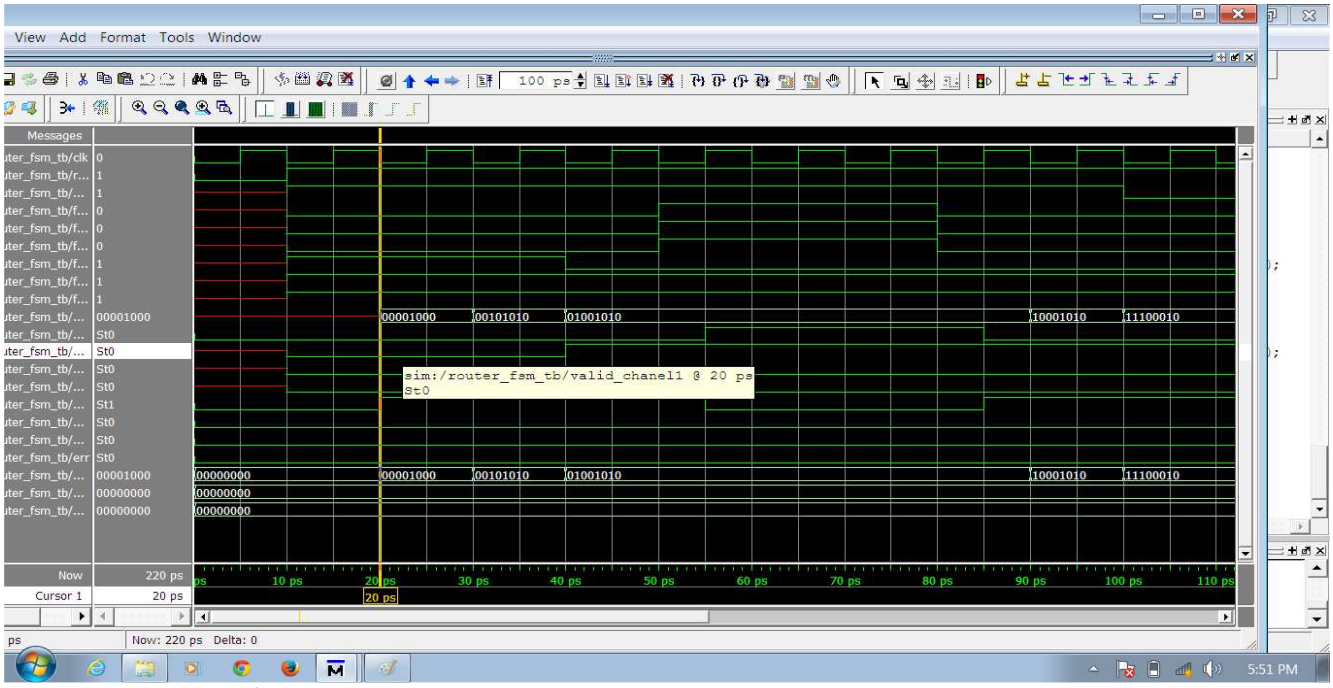

Fig.6. Block Diagram of FSMTB Simulation Waves

V. CONCLUSION

Increase in the speed of processors has led to important role of communications in interconnection networks. The restrictions that deadlock avoidance routing algorithms apply on the routing of packets prevent the packet to be routed completely base on network traffic condition. The True Fully Adaptive Routing algorithm provides packets routing completely base on traffic condition. A bus adjacent NoC improve the performance of network and provides the bus advantage beside NoC. The simulation results are shown, this bus suitable for deadlock recovery. International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, No.4, August 2012 8 According to deadlock rarely occurrence, when the network is not close or beyond saturation if flexible routing algorithm is used, this bus is applicable for broadcast and multicast operations, system management, delay sensitive signals and etc. With increase of packet injection rate, the network tends to saturation. With increase of packet injection rate, the network tends to saturation. Therefore latency of packets in reaching to destination will severely increase with respect to algorithms adaptation and traffic patterns. Adding virtual channels in each direction in routers can increase network throughput. Also uses of two virtual channels per physical channel have been shown to be enough to reduce probability of deadlock to very small values. Our future objections are discussion of the effect of virtual channels on average of throughput and packets latency in the architecture of network on chip with enhanced bus.

VI. REFERENCES

[1] J. Duato and S. Halamanchili, Interconnection Networks and Engineering approach, Morgan Kaufmann Publishers, 2003. Pinkston and S. Warnakulasuriya, "On Deadlocks in Interconnection Networks", Proc. 24th International symp. Computer Architecture, 1997.

[2] Zhang and L. Hou, "Comparison Research between XY and Odd-Even Routing Algorithm of a 2- Dimension 3x3 Mesh Topology Network-on-Chip", Proc. Global Cong. on Intelligent Systems, pp. 329-333, 2009.

[3] A. de mello and L. Ost et al, "Evaluation of Routing Algorithms on Mesh Based NoCs", Technical Report Series, N. Faculdade De Informatics Pucrs - Brazil, 2004.

[4] M. D. Grammatikakis and P. Mattheakis, "Automated Recovery from Protocol Deadlock – Test Bench Manual", 2008.

[5] A. de mello and L. Ost et al, "Evaluation of Routing Algorithms on Mesh Based NoCs", Technical Report Series ,N. 04, Faculdade De Informatica Pucrs - Brazil, 2004. [5] M. D. Grammatikakis and P. Mattheakis, "Automated Recovery from Protocol Deadlock – Test Bench Manual", 2008.

[6] P. Lopez, J. M. Martinez and J. Duato, "A Very Efficient Distributed Deadlock Detection Mechanism for Wormhole Networks", Proc. High Performance Computer Architecture Workshop, pp. 57-66.

[7] E. Baydal, P. L´opez and J. Duato, "A Simple and Efficient Mechanism to Prevent Saturation in Wormhole Networks", Proc. Int1 Parallel and Distributed Processing symp. , pp. 617-622, 2000.

[8] E. Baydal, P. Lopez and J. Duato, "A Congestion Control Mechanism for Wormhole Networks", Proc. Ninth Euromico Workshop on parallel and Distributed Processing, pp. 19-26, 2001.

[9] Anjan K. V. and T. M. Pinkston, DISHA: "A Deadlock Recovery Scheme for Fully Adaptive Routing", Proc. 9th International Parallel Processing Symp. , pp. 573-543.

[10] J. M. Martinez-Rubio, Pedro Lopez, and Jose Duato, "A Cost-Effective Approach to Deadlock Handling in Wormhole Networks", IEEE Trans. on Parallel and Distributed Systems, Vol. 12, No. 7, pp.716-729, 2001.

[11] M. Mirza - Aghatabar, A. Tavakol, H. Sarbazi-Azad, A. Nayebi , An Adaptive Software-based Deadlock Recovery Technique, Proc. 22nd International Conf. on Advanced Networking and Applications- workshops, pp. 514-519, 2008.

[12] J. H. Ki, Z. Liu, and A. A. Chien, "Compression less routing: A framework for adaptive and fault tolerant routing," Proc.21st International Symp. On Computer Architecture, pp. 289–300, 1994.