International Journal of Research Studies in Science, Engineering and Technology
Volume 1, Issue 9, December 2014, PP 126-133
ISSN 2349-4751 (Print) & ISSN 2349-476X (Online)

On Chip Permutation Network for Multiprocessor System on Chip

G. Priyanka*, K. Madan Mohan**

*PG Scholar, Electronics and Communication Engineering,
Intel Engineering College, AP, India
**Assistant Professor, Electronics and Communication Engineering,
Intel Engineering College, AP, India

Abstract: *This On-Chip Permutation Network for Multiprocessor System-On-Chip presents the silicon-proven design of a novel on-chip network to support guaranteed traffic permutation in multiprocessor system-on-chip applications the routing algorithm, the deflection routing is not energy-efficient due to the extra hops needed for deflected data transfer, compared to a minimal routing. Moreover, the deflection makes packet latency less predictable. The proposed network employs a pipelined circuit-switching approach combined with a dynamic path-setup scheme under a multistage network topology. The dynamic path-setup scheme enables runtime path arrangement for arbitrary traffic permutations. The circuit-switching approach offers a guarantee of permuted data and its compact overhead enables the benefit of stacking multiple networks Unlike conventional packet-switching approaches, our on-chip network employs a circuit-switching mechanism with a dynamic path-setup scheme under a multistage network topology. The dynamic path setup tackles the challenge of runtime path arrangement for conflict-free permuted data. The pre-configured data paths enable a throughput guarantee. By removing the excessive overhead of queuing buffers, a compact implementation is achieved and stacking multiple networks to support concurrent permutations in runtime is feasible.*

Index Terms: *Guaranteed throughput, multistage interconnection network, network-on-chip, permutation network, pipelined circuit-switching ,traffic permutation.*

1. Introduction

As VLSI technology becomes smaller, and the number of modules on a chip multiplies, on-chip communication solutions are evolving in order to support the new inter-module communication demands. Traditional solutions, which were based on a combination of shared-buses and dedicated module to module wires, have hit their scalability limit, and are no longer adequate for sub-micron technologies. Current chip designs incorporate more complex multilayered and segmented interconnection buses. More recently, chip architects have begun employing on-chip network-like solutions .This evolution of on-chip interconnects may evoke feelings networking old-timers. We believe that the considerations that have driven data communication from shared buses to packet-switching networks (spatial reuse, multi-hop routing, flow and congestion control, and standard interfaces for design reuse, etc.) will inevitably drive VLSI designers to use these principles in on-chip interconnects. In other words, we can expect the future chip design to incorporate a full fledged network-on-a-chip (NoC), consisting of a collection of links and routers and a new set of protocols that govern their operation.

In designing a NoC, one has to address all the classical networking issues. Addressing and routing schemes need to be devised in order to allow packets traversing the same links to be routed to diverse destinations. Names meaningful to applications (such as memory and I/O addresses) need to be translated into routing efficient labels. Since the timely delivery of certain types of traffic (or signals) on the chip is crucial for performance, support for multiple quality-of service (QoS) requirements is also essential. Similarly, a NoC should support network level congestion control in order to accommodate excessive traffic conditions. Where congestion control is employed, fairness issues need to be considered as well. One also needs to address reliability in the face of communication soft errors that may corrupt transmitted data this can be done using a combination of error-correction codes and retransmission mechanisms.

This paper presents a novel silicon-proven design of an on-chip permutation networkfics under arbitrary permutation. Unlike conventional packet-switching approaches, our on-chip network employs a circuit-switching mechanism with a dynamic path-setup scheme under a multistage network topology. The dynamic path setup tackles the challenge of runtime path arrangement for conflict-free permuted data. The pre-configured data paths enable a throughput guarantee. By removing the excessive overhead of queuing buffers, a compact implementation is achieved and stacking multiple networks to support concurrent permutations in runtime is feasible.

2. Related Work

Regarding the switching technique, packet switching requires an excessive amount of on chip power and area for the queuing buffers (FIFOs) with pre-computed queuing depth at the switching nodes and/or network interfaces. Regarding the routing algorithm, the deflection routing is not energy-efficient due to the extra hops needed for deflected data transfer, compared to a minimal routing. Moreover, the deflection makes packet latency less predictable; hence, it is hard to guarantee the latency and the in-order delivery of data. This paper presents a novel silicon-proven design of an on-chip permutation network to support guaranteed throughput of permutated traffics under arbitrary permutation.

Unlike conventional packet-switching approaches, our on-chip network employs a circuitswitching mechanism with a dynamic path-setup scheme under a multistage network topology. The dynamic path setup tackles the challenge of runtime path arrangement for conflict-free permuted data. The pre-configured data paths enable a throughput guarantee. By removing the excessive overhead of queuing buffers, a compact implementation is achieved and stacking multiple networks to support concurrent permutations in runtime is feasible. In our synthesis approach, we use accurate delay and power models for the network components (switches and links) that are obtained from layouts of the components using industry standard tools. The synthesis approach utilizes the floor plan knowledge of the NoC to detect timing violations on the NoC links early in the design cycle.

This leads to a faster design cycle and quicker design convergence across the high-level synthesis approach and the physical implementation of the design. We validate the design flow predictability of our proposed approach by performing a layout of the NoC synthesized for a 25-core CMP. Our approach maintains the regular and predictable structure of the NoC and is applicable in practice to existing NoC architectures.

A. The Switch and Network Taxonomy

The switch is the other important component in IIP and has a central function in NoC. Responsible for routing data packets, it implements the network (sending resource-to-receiving resource routing) and link layer (switch-to-switch routing) when receiving a data packet, the switch extracts the header information, makes routing decision based on the header information and current traffic load (to avoid congestion) and performs appropriate action (put the packet onto a link, delay the packet, drop the packet, etc.). So far, the NoC has been described as a communication network based on data packets and the high-level logic function of the switch is routing the packets.

For different network cores, different approaches may be used for data packet routing. In the following text, the traditional telecommunications network taxonomy (also apply on NoC), which determines the low-level architecture and Implementation of the switch will be studied.

A traditional telecommunications network either employs circuit or packet switching. A link in a circuit switched network can use either frequency-division multiplexing (FDM) or time division multiplexing (TDM) while packet switched networks are either virtual circuit (VC) networks or datagram networks. This classification can be generalized and apply on any network core, including NoC.

B. Packet Switching

Depending on the routing method, packet switched networks are divided into virtual circuit networks and datagram networks. The virtual circuit approach is connection-oriented and resembles the circuit switching. Both packet switched VC network and circuit switched network are suitable for uniform data traffic with long lifetime. For other bursty traffic, the connection management will tend to be computationally demanding and occupy a large portion of the bandwidth. They also require that the switches maintain the state information, resulting in more complex switch architecture and signaling scheme between switches. To reduce the switch complexity and therefore also the area overhead, datagram switching can be used. The datagram based switch is state and memory less, each packet is treated independently, with no reference to preceding packets. This approach more easily adapts to changes in the network such as congestion and dead links. However, it does not guarantee that packets with same source and destination will follow the same route. Consequently, the delay of packets with same source and destination may vary and packets may also arrive out of order, requiring buffering element at the receiving end. A datagram based switch implementation is described in.

C. Circuit Switching

A circuit switched network requires a dedicated end-to-end circuit (with a guaranteed constant bandwidth) between the transmitting and the receiving end. As the "circuit" is an abstract concept, most of the time, it is not a physical end to end wire, but can span over many links. In a telecommunications network, the circuit is typically implemented with either frequency division multiplexing or time-division multiplexing in each link. With FDM, the frequency spectrum of a link is shared among the connections across the link. For obvious reasons, the FDM is not suitable for NoC. For TDM on the other hand, time is divided into frames of fixed duration, and each frame is divided into a fixed number of time slots. When the network establishes a connection (or circuit) across a TDM link, the network dedicates a certain number of time slots in every frame to the connection. These slots are dedicated for the sole use of that connection, with some time slots available for use (in every frame) to transmit the connections data. The Ethereal Network on Chip developed at Philips Research is based on the time-division multiplexed circuit switching approach described above.

3. ON-CHIP NETWORK DESIGN

The key idea of proposed on-chip network design is based on a pipelined circuit-switching approach with a dynamic path-setup scheme supporting runtime path arrangement. Before mentioning the dynamic path-setup scheme, the network topology is first discussed. Then the designs of switching nodes are prespresented*A. On-Chip Network Topology* Clos network, a family of multistage networks, is applied to build scalable commercial multiprocessors with thousands of nodes in macro systems [7], [11]. A typical three-stage Clos network is defined as c(n,m,p)where _ represents the number of inputs in each of _ first-stage switches and _ is the number of second-stage switches. In practical MPSoCs [3]–[5], we proposed to use c(4,4,4)topology for the designed network (see Fig. 1). This network has a rearrangeable property [11] that can realize all possible permutations between its input and outputs.

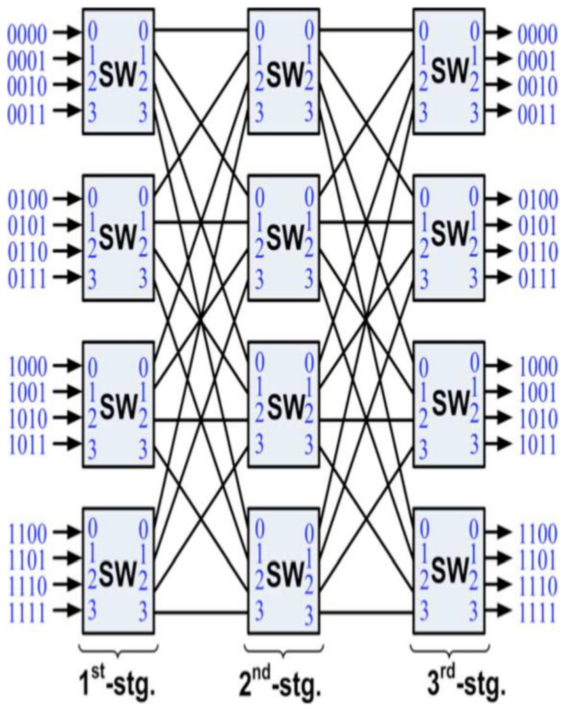


Fig. 1. *Proposed on-chip network topology with port addressing scheme*

The choice of the three-stage Clos network with a modest number of middle-stage switches is to minimize implementation cost, whereas it still enables a rearrangeable property for the network. A pipelined circuit-switching scheme is designed for use with the proposed network. This scheme has three phases: the *setup*, the *transfer*, and the *release* [2], [9]. A dynamic path-setup scheme supporting the runtime path arrangement occurs in the setup phase. In order to support this circuit-switching scheme, a switch-by-switch interconnectioFig. 2.

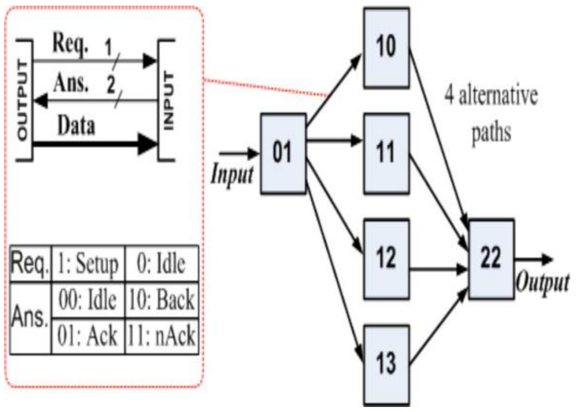


Fig. 2. *Switch-by-switch interconnection and path-diversity capacity*

The bit format of the handshake includes a 1-bit *Request (Req)*and a 2-bit *Answer (Ans)*. Req=1is used when a switch requests an idle link leading to the corresponding downstream switch in the setup phase. The req=ois also kept during data transfer along the set up path. A req=0 denotes that the switch releases the occupied link. This code is also used in both the setup and the release phases. An Ans=01(Aek) means that the destination is ready to receive data from the source. When the Ans=01 propagates back to the source, it denotes that the path is set up, then a data transfer can be started immediately. An Ans=11(n Aek) is reserved for end-to-end flow control when the receiving circuit is not ready to receive data due to being busy with other tasks, or overflow at the receiving buffer, etc. An Ans=10 (*Back*) means that the link is blocked. This *Back* code is used for a backpressure flow control of the dynamic path-setup scheme, which is discussed in the following subsection. *B. Dynamic Path Setup to Support Path arrangement* A dynamic path-setup scheme is the key point of the proposed design to support a runtime path arrangement when the permutation is changed.

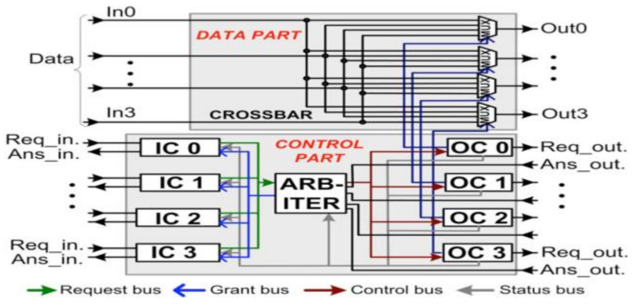
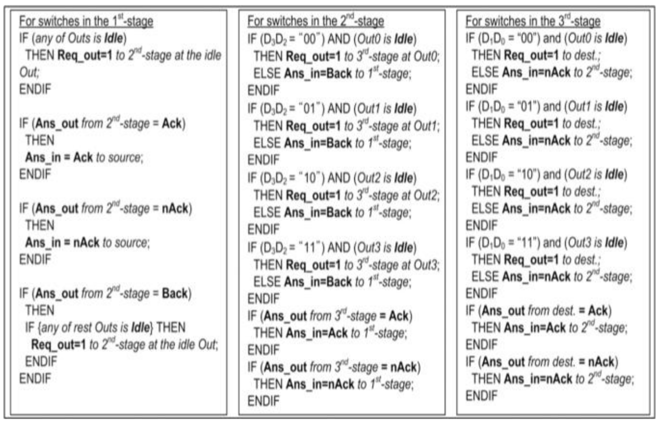


Fig. 3. *Common switch architecture*

Each path setup, which starts from an input to find a path leading to its corresponding output, is based on a *dynamic probing* mechanism. The concept of probing is introduced in works [2], [9], in which a probe (or setup flit) is dynamically sent under a routing algorithm in order to establish a path towards the destination. Exhausted profitable backtracking (EPB) [12] is proposed to use to route the probe in the network work.A path arrangement with *full permutation* consists of sixteen path setups, whereas a path arrangement with *partial permutation* may consist of a subset of sixteen path setups.

A question is that can the proposed EPB-based path setups used with the Clos C (4,4,4)realizen with its handshake signals is proposed, as shown in possible *full permutations* between itinputs and outputs? As proofed in works [11], [13], the three-stage Clos network C(m,n,p)is rearrangeable if m>n In the proposed network of C(4,4,4) m=n=4 so it is r*earrangeable*. There always exists an available path from an idle input leading to an idle output. By the *Exhaustive Property* of EPB as proofed inwork [12], the EPB-based path setup completely searches all the possible paths within the set of path diversity between an idle input and idle output. Directly applying the Exhaustive Property of the search into rearrangeable C(4,4,4)shows that the EPB-based path setup can always find an available path within the set of four possible paths between the input and the idle output. Based on this EPB-based path-setup scheme, it is obvious that the path arrangement for full (as well as partial) permutation can always be realized in the proposed network with C(4,4,4)topology. As designed in this network, each input sends a probe containing a 4-bit output address to find an available path leading to the target output. During the search, the probe moves forwards when it finds a free link and moves backwards when it faces a blocked link. By means of non-repetitive movement, the probe finds an available path between the input and its corresponding idle output.



```
For switches in the 1st-stage
IF (any of Outs is Idle)
  THEN Req_out=1 to 2nd-stage at the idle
Out;
ENDIF

IF (Ans_out from 2nd-stage = Ack)
  THEN
  Ans_in = Ack to source;
ENDIF

IF (Ans_out from 2nd-stage = nAck)
  THEN
  Ans_in = nAck to source;
ENDIF

IF (Ans_out from 2nd-stage = Back)
  THEN
  IF {any of rest Outs is Idle} THEN
    Req_out=1 to 2nd-stage at the idle Out;
  ENDIF
ENDIF
```

```
For switches in the 2nd-stage
IF ($D_3D_2$ = "00") AND (Out0 is Idle)
  THEN Req_out=1 to 3rd-stage at Out0;
  ELSE Ans_in=Back to 1st-stage;
ENDIF
IF ($D_3D_2$ = "01") AND (Out1 is Idle)
  THEN Req_out=1 to 3rd-stage at Out1;
  ELSE Ans_in=Back to 1st-stage;
ENDIF
IF ($D_3D_2$ = "10") AND (Out2 is Idle)
  THEN Req_out=1 to 3rd-stage at Out2;
  ELSE Ans_in=Back to 1st-stage;
ENDIF
IF ($D_3D_2$ = "11") AND (Out3 is Idle)
  THEN Req_out=1 to 3rd-stage at Out3;
  ELSE Ans_in=Back to 1st-stage;
ENDIF
IF (Ans_out from 3rd-stage = Ack)
  THEN Ans_in=Ack to 1st-stage;
ENDIF
IF (Ans_out from 3rd-stage = nAck)
  THEN Ans_in=nAck to 1st-stage;
ENDIF
```

```
For switches in the 3rd-stage
IF ($D_1D_0$ = "00") and (Out0 is Idle)
  THEN Req_out=1 to dest.;
  ELSE Ans_in=nAck to 2nd-stage;
ENDIF
IF ($D_1D_0$ = "01") and (Out1 is Idle)
  THEN Req_out=1 to dest.;
  ELSE Ans_in=nAck to 2nd-stage;
ENDIF
IF ($D_1D_0$ = "10") and (Out2 is Idle)
  THEN Req_out=1 to dest.;
  ELSE Ans_in=nAck to 2nd-stage;
ENDIF
IF ($D_1D_0$ = "11") and (Out3 is Idle)
  THEN Req_out=1 to dest.;
  ELSE Ans_in=nAck to 2nd-stage;
ENDIF
IF (Ans_out from dest. = Ack)
  THEN Ans_in=Ack to 2nd-stage;
ENDIF
IF (Ans_out from dest. = nAck)
  THEN Ans_in=nAck to 2nd-stage;
ENDIF
```

Fig. 4. *Probe routing algorithms designed to route probe in each stage.*

The EBP-based path-setup scheme is designed with a set of probe routing algorithms as mentioned later in Fig. 4. The following example describes how the path setup works to find an available path by using the set of path diversity shown in Fig. 2. It is assumed that a probe from a source (e.g., an input of switch01) is trying to set up a path to a target destination (e.g., an available output of switch 22)First, the probe will non-repetitively try paths through the second-stage switches in the order of 10-11-12-13 Assuming that the link 0-10 is available, the probe first tries this link (req=1) and then arrives at switch 10 If link 10-22 is available, the probe arrives at switch 22 and meets the target output. An Ans=Aek then propagates back to the input to trigger the transfer phase. • If link 10-22 is blocked, the probe will move back to switch 01 Ans=Back and link 01-10 is released (Req=o) From switch 01, the probe can then try the rest of idle links leading to the second-stage switches in the same manner. By means of moving back when facing blocked links and trying others, the probe can dynamically set up the path in runtime in a conflict-avoidance manner. *C. Switching Node Designs* Three kinds of switches are designed for the proposed on-chip network.

These switches are all based on a common switch architecture shown in Fig. 3, with the only difference being in the probe routing algorithms. This common architecture has basic components: INPUT CONTROLs (ICs), OUTPUT CONTROLs (OCs), an ARBITER, and Fig. 5. (a) FIFO-based test wrappers supporting (b) end-to-end source-synchronous data transfer scheme. a CROSSBAR. Incoming probes in the setup phase can be transported through the data paths to save on wiring costs. The ARBITER has two functions: first, cross-connecting the Ans_Outs and the ICs through the *Grant bus*, and second, as a referee for the requests from the ICs. When an incoming probe arrives at an input, the corresponding IC observes the output status through the *Status bus*, and requests the ARBITER to grant it access to the corresponding OC through the *Request bus*. When accepting this request, the ARBITER cross-connects the corresponding Ans_Out with the IC through the *Grant bus* with its first function. With the second function, the ARBITER, based on a pre-defined priority contention when several ICs request the same free output. After this resolution, only one IC is accepted, whereas the rest are answered as facing a blocked link (i.e., similar to receiving an Ans=Back).The IC is implemented with finite-state machine (FSM). The probe routing algorithm and the operation of the switches are controlled according to this FSM implementation in the ICs [9]. The probe routing algorithms and their corresponding handshake signals are given in Fig. 4. In order to support the probing path setup, ICs are implemented with different probe routing algorithms depending on its switch stage.

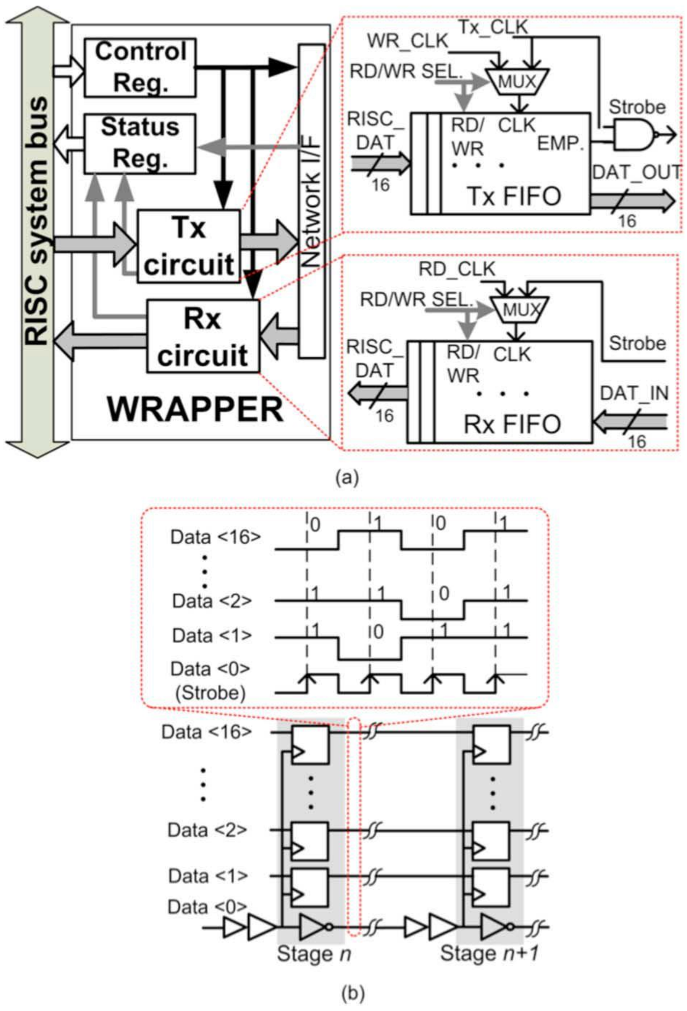


Fig. 5. (a) *FIFO-based test wrappers supporting (b) end-to-end source-synchronous data transfer scheme*

The probe contains the 4-bit address of the destination, i.e., $D_{3,}D_{2,}D_{1,}D_0$ (see Fig. 1 for the addressing scheme). The three routing algorithms for the switches in the first, the second, and the third stages are detailed in Fig. 4. In the first stage, the switch tries the free outputs in a non-repetitive manner (e.g., outputs 0-1-2-3). This implementation avoids repetitively searching the same path that may result in a live-lock. The second- and third-stage switches rely on the two most significant bits $D_3D_2$and the two least signification bits $D_{1,}D_0$ of the destination address, respectively, to route the probe. As can be seen from Fig. 4, depending on the availability of the desired output or the feedback (i.e., the signal *Ans*) from the downstream switch, the IC in a given switch will change its FSM state and reply to the upstream switches accordingly.

The OCs work as re-timing stages for the commands from ARBITER placed on the *Control bus* and control the CROSSBAR. The CROSSBAR is a 4_4 full-connecting matrix designed with output multiplexers. The ICs and the ARBITER are clocked with the rising and the falling edges of the clock, respectively. By this implementation, probing is dynamically processed by the switch in one clock cycle basis. As denoted in Fig. 3, the control part of switches performs the dynamic EPB-based path setup, whereas the data part simply provides configured paths for guaranteed circuit-switched data. This meets the target of designing the circuit-switched switches to support EPB-based path setup in network. To validate if the designed network works as desired, a test bench is applied to test the capability of realizing full permutation with sixteen path setups. To avoid a path setup interfering with others during the search and incurring a rearrangement of existing paths, a delay is set between the path setups launched one-by-one in a sequence in the test bench. This is to ensure that the previous path setup is completed before a new one is launched. As calculated based on the path-diversity graph shown in Fig. 2, the worst-case path setup needs 14 steps (hops) of moving its probe back and forth to search for a path. Each step of moving the probe needs two cycles, as derived from therule, resolvescycle-accurate design model. Hence, we set the delay to a value of 28 cycles. Arranging a full permutation requires 448 cycles to complete. By this setting, we simulate and validate the success of the design in arranging over ten different sets of 10 000 random full permutations.

4. Results and Conclusion

Simulated Environment

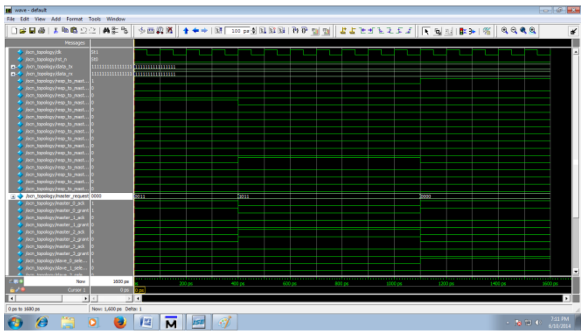


Fig 6.1. *simulations result*

5. Conclusion & Future Scope

The various challenges faced by researchers in SoC design forced them to look for new alternatives which paved the way for Network-on-chip technology. The NoC is a vast and emerging research area that is still in its initial stages. The NoC area has a significant influence in the design of next generation SoC or multicore architectures. In our project we went through the various research aspects of NoC and details of network topology and routing algorithms were explored. We tried to contribute in the research of NoC by exploring the design space of NoC routers which is a dominant component of the network. After analyzing the three architectures we concluded that the CDMA router architecture performs better than the other two. It has constant delay, constant latency, high throughput. Moreover it has concurrent transmission which gives it more flexibility over the other two architectures and it is less error prone. But it has more area than the other two.

The ultimate goal of this project is to develop a 4x4 NoC. The work conducted so far is the first part of the whole project. Future work includes the extension of the router architectures and to construct an efficient NoC. The FPGA implementation of the NoC will also be done.

References

[1] Gordon E Moore, "Cramming more components onto integrated circuits," *Electronics*, Vol. 38, No. 8. (19 April 1965), pp. 114-117.

[2] International Technology Roadmap for Semiconductors, report 2012.

[3] Resve Saleh, Shahriar Mirabbasi, AlanHu, Mark Greenstreet, Guy Lemieux, Partha Pratim Pande, Cristian Grecu, and Andre Ivanov, "System-on-Chip: Reuse and Integration," *Proceedings of the IEEE*, vol. 94, no. 6, Jun. 2006.

[4] R. Rajsuman, *System-on-a-Chip Design and Test*, Boston, MA: Kluwer, 2000.

[5] Tobias Bjerregaard and Shankar Mahadevan, "A Survey of Research and Practices of Network-on-Chip," *ACM Computing Surveys*, vol. 38, no. 1, pp. 1-51, 2006.

[6] Luca Benini and Giovanni De Micheli, "Networks on chips: a new SoC paradigm," *Comput.*, vol. 35, no. 1, pp. 70-78, 2002.

[7] Rickard Holsmark and Magnus Hgberg, "*Modelling and Prototyping of a Network on Chip*," Master of Science Thesis, 2002 Electronics.

[8] Muhammad Ali, Michael Welzl, and Martin Zwicknagl, "Networks on Chips: Scalable Interconnects for Future Systems on Chips," *4th European Conference on Circuits and Systems for Communications*, pp. 240-245, 2008.

[9] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Fault tolerant algorithms for network-on-chip interconnect," *Proceedings. IEEE Computer society Annual Symposium on VLSI*, pp.46-51, Feb. 2004.

[10] Israel Cidon and Idit Keidar, “Zooming in on network-on-chip architectures,”*16th international conference on Structural Information and Communication Complexity*, 2009.

[11] Xinan Zhou, “*Performance evaluation of network-on-chip interconnect architectures*,” Master of Science Thesis, 2009 Electrical Engineering.

[12] A. Jantsch and H. Tenhunen. *Networks on Chip,* Kluwer Academic Publishers, 2003.

[13] W.J. Dally and B. Towles “Route packets, not wires: On-chip interconnection networks,” *Proc. DAC*, 2001.50

[14] David Atienza, Federico Angiolini, Srinivasan Murali, Antonio Pullini, Luca Benini, and Giovanni De Micheli, “Network-on-Chip design and synthesis outlook,” *INTEGRATION, the VLSI journal* vol. 41, pp. 340–359, 2008.

[15] Davide Bertozzi and Luca Benini Xpipes “A Network-on-Chip Architecture for Gigascale Systems-on-Chip,” *IEEE CIRCUITS AND SYSTEMS MAGAZINE*, SECOND QUARTER, pp. 18-31, 2004.