2011 International Conference on Computer Science and Information Technology (ICCSIT 2011)
IPCSIT vol. 51 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V51. 119

Design and Implementation of Lookup Table in Multicast Supported Router

Wenmin Hu[a,*], Hengzhu Liu[a],Zhonghai Lu[b], Axel Jantsch[b]

[a]School of Computer, National University of Defense Technology, Chansha, P.R. China
[b]Royal Institute of Technology, Stockholm, Sweden

Abstract. In this paper, a design and implementation of a lookup table are proposed, which facilitate the multicast supporting in Network-on-Chip (NoC). Our lookup table allows setting or clearing some bit in a table entry, clearing table entry and reading the content from the table. Base on this, our scheme could setup any-shaped multicast path by combining the sub-path incrementally.

Keywords: Network-on-Chip, multicast

1. Introduction

Since billions of transistors could be integrated in one die, multi-core architectures have become the mainstream for designing of the multiprocessor System-on-Chip (MPSoC). The communication becomes a bottleneck in improving the performance for MPSoC. Network-on-Chip (NoC) [1], [2], [3], [4], [5]emerged as a promising approach to solve the global interconnect problems of MPSoC.

As an important communication mode existing in MPSoC applications, multicast plays a key role in whole system performance. Traditional unicast-based router could implement multicast by replicating and sending copies to different destination nodes. However, it is inefficient, due to the long startup latency for some packets. Thus, some efficient multicast-based routers were proposed [6], [7], [8], [9]. Generally, routing approaches on NoC can be classified into two categories: *table-based* and *logic-based*. Since the table-based approach supports any-shaped routing path according to special requirement, it is widely used in multicast.

This work is partly based on the work in [9], which supports any-shaped path setup. Since paper [9] didn't show design of the lookup table in detail, we walk through the detailed design of lookup table, which is a key component in table-based multicast supported router.

The paper is organized as follows. In Section 2, a brief review of related work is presented. In Section 3, we introduce the architecture of our router as preliminaries. In Section 4, the design and implementation of the lookup table are discussed. In Section 5, we present the experiment result. Finally, conclusion and future work are given in Section 6.

2. Related Work

For table-based routing, Jerger et al. presented an efficient mutlicast routing scheme named Virtual Circuit Tree Multicasting (VCTM) [6]. VCTM constructs the multicast tree incrementally by sending several unicast setup packets to destinations. Each setup packet is routed using the Dimension-Ordered Routing (DOR) algorithm and the outgoing direction is stored in a table entry according to the identical ID [6]. Based on VCTM, Hu et al. proposed a approach to setup the multicast path in a optimized shape [9]. In their scheme, a two-period sub-path setup process is used to form the multicast tree incrementally. During the first period, the setup packet routes to a predetermined intermediate node just likes the normal unicast packet; in

**E-mail address:* huwenmin@nudt.edu.cn

the second period of sub-path setup, the packet routes to the destination and simultaneously updates the lookup table using the outgoing direction computed by the routing unit.

Since both [6] and [9] didn't introduce the implement of lookup table in detail, we give a HDL implementation of lookup table in detail.

3. Preliminaries

In this section, we first introduce the packet types supported by our router. Then, we gives the architecture of proposed router. Since we adopted lookahead routing, the packet format and architectures are different from the work in [9].

3.1. Packet Format

Each multicast forms a tree connecting the source with the destination set, which is identified by a Multicast ID number to each source node and its destination nodes combination. For a tree-based approach, a multicast packet travels along a common path until it arrives at the branch node, where it is replicated and forwarded to corresponding outgoing ports. Once a multicast tree is setup, the packet will be routed based on the multicast table (MCT) number at each router. At the source node, destination set content addressable memory (CAM) is integrated to record the destination set for multicast trees. Each entry is *n* bits vector (*n* represents the number of nodes on the NoC). If one bit is set, it means that the corresponding node is the destination node. A bit indicating whether the entry is valid is also included. At each router, MCT is partitioned to *n* sub-tables corresponding to each source node. Each sub-table has 16 entries or more.

In our scheme, incremental setup is adopted to simplify the process of multicast path building which is similar to VCTM. In VCTM, each setup packet is routed using the Dimension-Ordered Routing (DOR) algorithm and the routing result is stored in a table according to the identical ID. The sub-path begins from the source node and ends at one destination node. So for a certain source-destination set, the mulitcast path shape is determinate. However, in our approach, the process of each sub-path setup is divided into two periods (TPSS), where during the first period the setup packet just routes to the predeterminate intermediate node, and during the second period the packet routes to the real destination node with the routing results reserved into the lookup table in the router passing through. The intermediate node is the branch node of multicast tree, which is determined by path searching algorithms. The sub-path begins from the intermediate node and ends at one destination node. Multiple unicast setup packets can be injected into the network to setup the sub-path in parallel. When all are done, a multicast tree is constructed successfully.

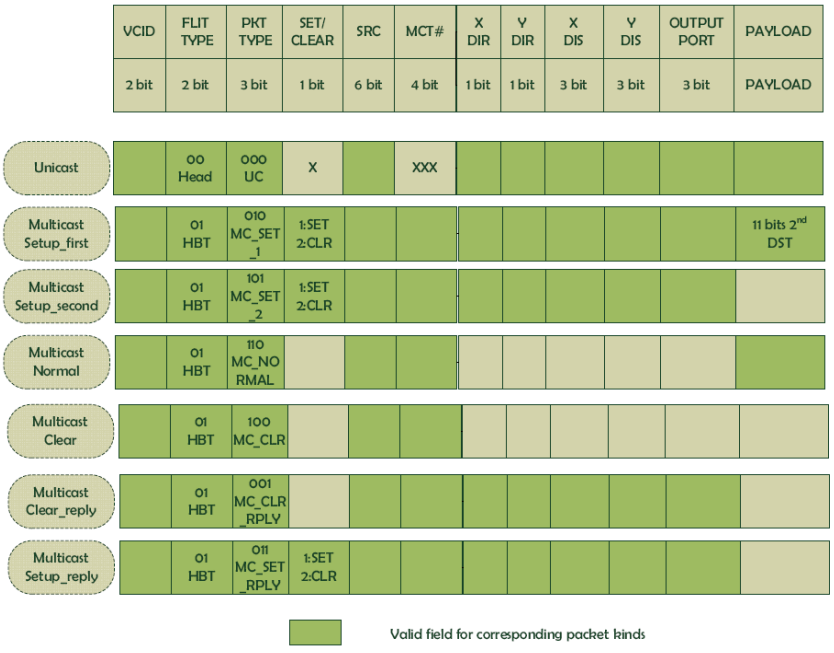

Fig. 1. Packet Format for all the packets supported in proposed router.

As shown in Fig. 1, the packets supported in our router are classified into four categories: *MC_SET_1*, *MC_SET_2* and *MC_SET_RPLY* are multicast setup; *MC_CLR* and *MC_CLR_RPLY* are multicast evicting; *MC_NORMAL* is multicast data packet; and *UC* is unicast data packet. The fields of packet are defined as follow: *X DIR* and *Y DIR* indicate the directions of destination node, *X DIS* and *Y DIS* are the Manhattan distance between the source node and destination node in X direction and Y direction. Since lookahead routing is adopted in our router, *OUTPORT* field stores the routing result of downstream router. *MCT#* is the id of multicast table entry. Source address is also encoded in *SRC*. *SET/CLEAR* indicates whether a sub-path is added to existing path or it is cut off. Four different flit types are supported by 2 bits field of *FLIT TYPE*: Head, body, tail and HBT.

TPSS is executed by routing packet *MC_SET_1* and *MC_SET_2*. During the first period, the setup packet *MCT_SET_1* routes like unicast packet until it reaches the intermediate node. On arriving at the intermediate node, the first 11 bits of *PAYLOAD* is replicated to the field of *X DIR*, *Y DIR*, *X DIS*, *Y DIS* and *OUTPORT* to form the new destination while the PACKET TYPE is also changed to *MC_SET_2*. TPSS enters the second period. When the packet traverses a router, the routing result is used to update multicast table entry corresponding to the combination of *SRC* field and *MCT#* field. Once the *MCT_SET_2* reaches destination, a multicast reply packet (*MC_SET_RPLY*) is sent to the source. The other setup packets can be injected into network without waiting for the reply of the former setup packet. Each branch of the tree can be built simultaneously. When the source node receives the replies of all the destination nodes, the setup is completed.

When a multicast destination set is missed in CAM and there is no free entry to be utilized, a used multicast tree is needed to be evicted. Only the source node has the right to evict the multicasts tree. *MC_CLR* packet will be routed by looking up MCT just like normal multicast data packet (*MC_NORMAL*). After getting the outgoing ports, the corresponding table entry will be cleared in next cycle. When the *MC_CLR* packet sinks at the destination node, the destination node will generate a reply packet (*MC_CLR_RPLY*). Once the source node receives all the reply packets, the multicast tree is evicted.

3.2. router architecture

In this research, we use the wormhole router due to its small buffer requirement and high throughput. Fig. 2 shows the architecture of the proposed router. It has five input ports, each of which contains four Virtual Channels (VCs). A register file with five read ports and one write port is integrated into the router to store outgoing ports for the multicast packet. Both unicast and multicast follow the pipeline stages: buffer write/routing computation (BW/RC), switch allocation/virtual channel allocation (SA/VA), switch traversal/line traversal (ST/LT). We use look ahead routing [10] to compute the output port for the next router and store it into the *OUTPUT PORT* field of the head flit. Output directions of head flit in current router is achieved by selecting one from *OUTPUT PORT* field and *OP2* from the multicast table. The criterion for output direction selecting is the packet type. For example, *MC_NORMAL* and *MC_CLR* use OP2 as the output directions while others use the *OUTPUT PORT* field in their head flit. *MCTSG* shown in Fig. 2 is the logic block to generate operation signals to multicast table according to some fields of head flit (*FLIT TYPE*, *PKT TYPE*, *SRC*, *MCT#*, *SET/CLEAR*). The operations to multicast table involve reading outgoing ports from multicast table (*MC_NORMAL*, *MC_CLR*), setting some bit in one entry of multicast table (*MC_SET_2*), and clearing all the bits in one entry of multicast table *(MC_CLR*).

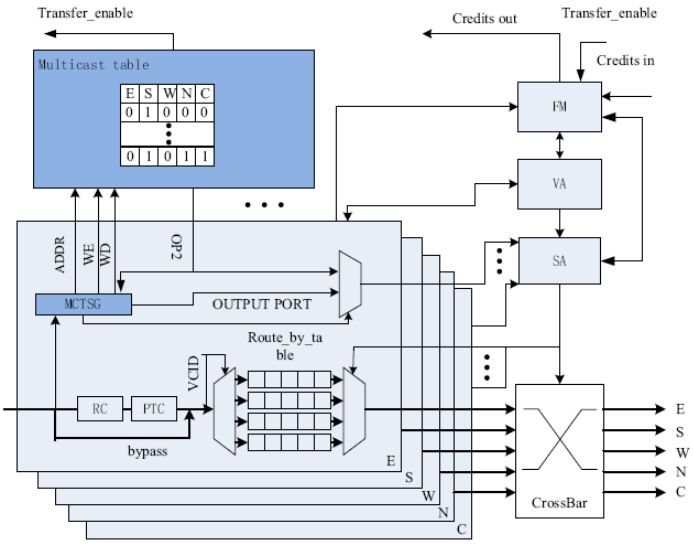

Fig. 2. Router Architecture

For the multicast packet routing, the result may contain multi-port. The flit is replicated to one port at one ST/LT stage when successfully getting the grant in SA/VA. Only when the flit is successfully transferred to all the destination ports, can the flit be deleted from the buffer. To keep the state of each multicast flit, the input virtual channel (VC) reserves a separate VC state register and buffer pointers. It is necessary to forward and control the pipeline stage by using the state register and buffer pointers. If the port belongs to the RC results, then its state register will be set to advance to SA/VA, otherwise the state will be idle. The buffer pointers contain a head pointer for an input VC, and five read pointers for all the destination ports.

As previously mentioned, any-shaped multicast path is supported in our NoC by injecting multiple setup packets to form the path incrementally. *MC_SET_1* needs to be changed to *MC_SET_2* at the intermediate node. So packet type convert logic (PTC) is integrated into the router too. As shown in Fig. 2, PTC is set after the RC. In PTC module, if *MC_SET_1* 's *X DIS* field and *Y DIS* field are 0, it would be changed to MC_SET_2 . The first 11 bits of PAYLOAD is also copied to *X DIR, Y DIR, X DIS, Y DIS* and *OUTPUT PORT*. This routing information is for the downstream router which is computed at the source node, so it is not necessary to compute the output direction due to the change of destination.

4. Implementation of lookup table

In order to support the functions aforementioned, lookup table should support these operations: setting some bit in one entry, clearing whole entry and reading whole entry. These operation signals were generated by MCTSG, as shown in Fig. 3. Signals *SET/CLEAR, PKT TYPE, FLIT TYPE, OUTPUT*

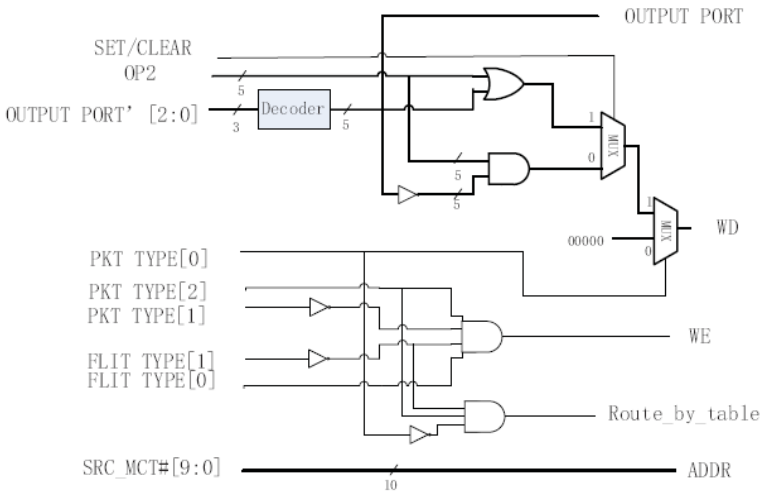

Fig. 3. Logic for MCTSG

PORT' and *SRC_MCT#* directly connect to the bits in packet stored in register. *OUTPUT PORT* is the outgoing direction computed by the upstream router, which is decoded from *OUTPUT PORT'* via 3-5 decoder. The OP2 are the outgoing directions read from the lookup table according *SRC* and *MCT#* . *SET/CLEAR* is used to select one operation from setting or deleting the corresponding bit in lookup table. WD is the final value written into the lookup table when the write enable signal *WE* is set. The write operation is enabled under two conditions: the packet type is *MC_SET_2*, where the WD selects the output from multiplexer controlled by SET/CLEAR; the packet type is *MC_CLR*, where the value 0 is selected as WD. *Route_by_table* is set only when the packet type is *MC_NORMAL* (110) or *MC_CLR* (100), where the flit type is HDT (01) or Head (00).

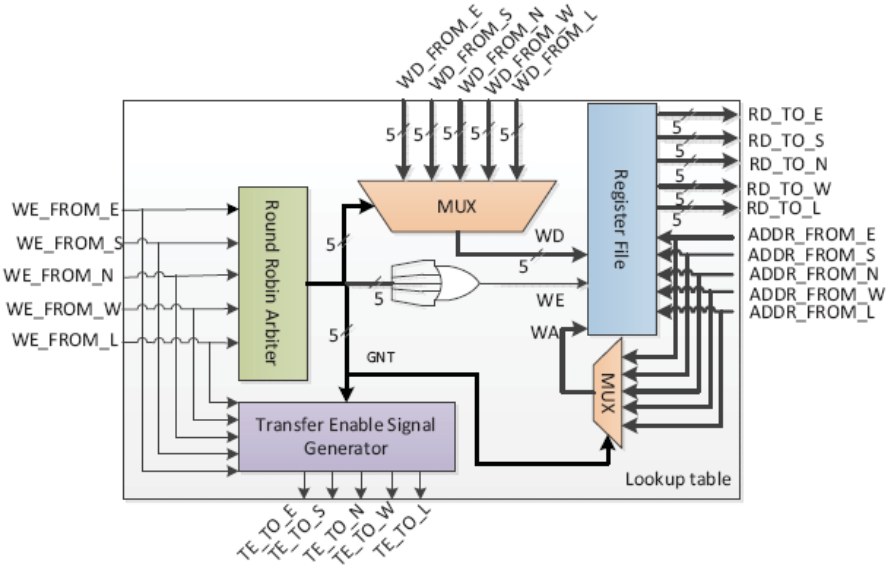

Fig. 4. Structure for Lookup Table

Fig. 4 shows the structure of lookup table, which is composed of multiplexer, register file, arbiter and transfer enable signal generator. There are five read ports in register file, which are corresponding to five input ports. Since the writing operation is not frequent, only when the entries is updated can it be executed, just one write port is integrated in register file. Hence, concurrent write requests from input ports must be arbitrated by round robin arbiter to avoid conflict. The one getting the grant writes the value to the register file, while other requests are hold by set the transfer enable signal as 0. Once the transfer enable signal is 0, the corresponding upstream router's output port is locked, and no new packet arrives at the input port. The register storing the setup packet is also locked and the write request is hold.

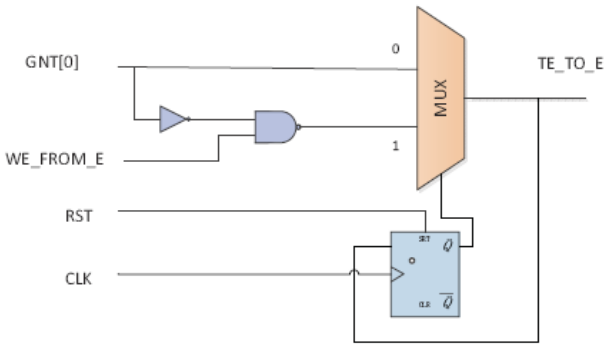

Fig. 5. Logic for Transfer Enable Signal Generator (input port E)

Fig. 5 shows the logic in Transfer Enable Signal Generator (TESG) for input port E. Since the logic for other ports is the same, here are not shown in detail. The transfer enable signal (TE_TO_E) is initiated as 1.

Only when the write enable signal (WE_FROM_E) is 1 and fails to get the write grant, where the GNT [0] is 0, the TE_TO_E will be reset to 0, then the register in input port E is locked. If the round robin arbiter gives the grant of writing after some cycles, the TE_TO_E will be set as 1, the register is unlocked.

5. Experiment

To evaluate the function of the lookup table, we develop the HDL model. Fig. 6 shows the waveform from the simulation by ModelSim. Two writing requests, WE_FROM_L and WE_FROM_E are initiated simultaneously. However, only the WE_FROM_L get the grant, and the value of WD_FROM_L (00101) is to be written to registers[1] in next cycle according to the ADDR_FROM_L. The write request WE_FROM_E holds for one cycle, and gets the grant. Hence, the value of WD_FROM_E (00001) is to be written to registers[0] in next cycle. The read function is also correct, where the RD_TO_E and RD_TO_L get the value in the same cycle, where the register is updated.

Table 1. Area and Power Breakdown

	Lookup table						MCTSG
entries	32	64	128	256	512	1024	
Area(um^2)	7211	13599	26840	52757	104071	199785	83.26
Power(mW)	3.77	7.01	13.41	26.05	50.99	117	0.023

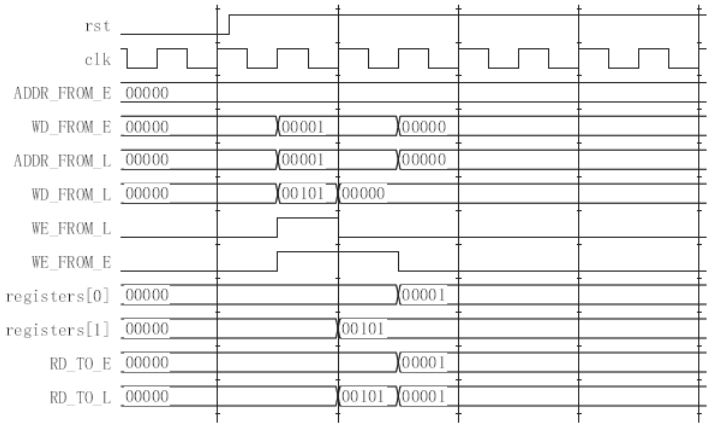

Fig. 6 Waveform for lookup table

Our lookup table and MCTSG have been synthesized using a CMOS stand-cell technology library from Chartered Semiconductor Manufacturing. Table 1shows the synthesis results with 90-nm CMOS stand-cell technologies. The logic synthesis of lookup table is made by setting the target working frequency to 1GHz.

6. Conclusions

In this paper, the design and implementation of a lookup table have been proposed, which facilitate the multicast supporting in Network-on-Chip (NoC). Our lookup table allows setting or clearing some bit in a table entry, clearing table entry and reading the content from the table. Base on this, our scheme could setup any-shaped multicast path by combine the sub-path incrementally.

For future work, we would like to research the dynamic organization of lookup table, which could downsize the lookup table.

7. Acknowledgement

This work is supported by the National Science Foundation of China, under grant No.60970037, and Doctor Program Foundation of Education Ministry of China, under grant No. 20094307110009 and No. 20114307130003.

8. References

[1] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. *In Proceedings of the Design Automation and Test in*

Europe Conference, February 2004.

[2] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, et al. The Raw microprocessor: A computational fabric for software circuits and general purpose programs. *IEEE Micro*, 22(2):25–35, 2002.

[3] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, et al. Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture. *In Proceedings of the 30th annual international symposium on Computerarchitecture*, February 2003.

[4] P. Guerrier and A. Greiner. A generic architecture for on-chip packet switched interconnections. *In Proceedings of the Design, Automation and Test in Europe Conference*, March 2000.

[5] E. Rijpkema, K. Goossens, J. Dielissen A. R˘adulescu, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. *IEE Proceedings: Computers and Digital Technique*, 150(5):294–302, September 2003.

[6] N. E. Jerger, L. S. Peh, and M. Lipasti. Virtual circuit tree multicasting: A case for on-chip hardware multicast support. *In Proceedings of the 35th annual international symposium on Computer architecture*, June 2008.

[7] F. A. Samman, T. Hollstein, and M. Glesner. Multicast parallel pipeline router architecture for network-on-chip. *In Proceedings of the Design, Automation and Test in Europe Conference*, March 2008.

[8] S. Rodrigo, J. Flich, J. Duato, and M. Hummel. Efficient Unicast and Multicast Support for CMPs. *In Proc. 41st IEEE/ACM Int'l Symp. Microarchitecture*, pages 364–375, 2008.

[9] W. Hu, Z. Lu, A. Jantsch, and H. Liu. Power-efficient tree-based multicast support for networks-on-chip. *In Proceedings of 16th Asia and South Pacific Design Automation Conference (ASP-DAC11)*, January 2011.

[10] M. Galles. Scalable Pipeline Interconnect for Distributed Endpoing Rouing: The SGI SPIDER Chip. *In Proc. Hot Interconnect*, pages 141–146, 1996.