Multi-list Design and FPGA Implementation Method of OLSR protocol

Hongyu Zhang, Bo Li, Zhongjiang Yan(✉), and Mao Yang

School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China
chong@mail.nwpu.edu.cn,
{libo.npu,zhjyan,yangmao}@nwpu.edu.cn

Abstract. Ad Hoc network is a new networking technology that does not rely on preset communication facilities, and is one of the important components of the next generation wireless communication network system. The OLSR (Optimized Link State Routing) protocol is a classic proactive routing protocol in Ad Hoc networks. Most of the existing research is based on software, there has not been a case of using hardware to implement the OLSR protocol. The core of implementing OLSR protocol based on FPGA (Field Programmable Gate Array) is the design of memory management table. This paper proposes a multi-link list design and FPGA implementation method, which effectively avoids the conflicts that will arise when using memory management tables.

Keywords: Ad Hoc · OLSR · FPGA · Memory management

1 Introduction

Ad Hoc network is a new and developing network technology in the current wireless communication field [1], which has the characteristics of no central control node, multi-hop routing, and dynamic topology [2]. It can provide flexible and convenient communication without fixed infrastructure support, which has a very wide range of application scenarios in today's society. A good routing protocol design can improve the overall performance of Ad Hoc networks [3], which is one of the hot spots and difficulties in the research of Ad hoc networks.

The Optimized Link State Routing protocol for mobile Ad Hoc networks is an optimization of the classical link state algorithm tailored to the requirements of a mobile wireless LAN [4], which is a table-driven active protocol. In the protocol, adjacent nodes periodically exchange HELLO packets [5] to realize neighbor discovery and link detection. Then MPR (Multipoint reply) nodes periodically forward TC packets to spread topology information to the entire network. Finally, the route calculation is realized by dynamically establishing and updating network topology. Among them, the addition of MPR mechanism

B. Li et al. (Eds.): IoTaaS 2020, LNICST 346, pp. 474–483, 2021.
https://doi.org/10.1007/978-3-030-67514-1_38

reduces the occupation of bandwidth resources due to the transmission of control packet information [6], which greatly reduces the overhead of network transmission.

Most of the research on Ad Hoc network at home and abroad is based on software. Existing network simulation tools mainly include: OPNET, NS-2, GloMoSim, OMNET++, NS-3, etc. [7], there has not been a case of implementing OLSR protocol based on FPGA.

In the Ad Hoc network, by connecting the hardware and the embedded chip, the power consumption can be reduced while the operation speed is increased, and other operations can be performed at the same time [8]. In addition, using hardware to implement the OLSR protocol can establish calls and change the dynamic topology faster [9]. Therefore, the implementation of OLSR protocol based on FPGA can effectively improve the speed and reduce the power consumption, which will have a great practical use.

The core of implementing the OLSR protocol based on FPGA is the design of the memory management table. The main difficulty is to solve the conflicts caused by "multiple modules simultaneously read or write to the same port of the same memory management table", which are generated during the work of the memory management table based on the "RAM + FIFO" design concept.

2 Implementation Model of Multi-link List Based on FPGA in OLSR protocol

With reference to the introduction of the OLSR protocol in the rfc3626 standard document, this article designs and implements an FPGA-based OLSR protocol. During its implementation, the FPGA-based OLSR project is divided into five parts: packet sending module, packet receiving module, memory management module, expired entry deletion module, and time counter module. Each part can be further divided into several sub-modules, the relationship of them is shown in Fig. 1.

During the work of the OLSR protocol, nodes send and process HELLO or TC packets by the packet sending module and the packet receiving module, which can continuously update the local memory management tables to calculate the shortest path from this node to other nodes, and store them in the local routing table. In order to maintain the local routing table, the expired entry deletion module combines the time counter module to periodically read and delete expired entries in each memory management table. It is not difficult to find that as the core of the OLSR protocol design, the memory management table has close information interaction with every module. The working relationship between each memory management table and other function modules is shown in Fig. 2.

For the memory management table, this article adopts the idea of one-way linked list, and uses two storage structures of true dual-port RAM and FIFO to

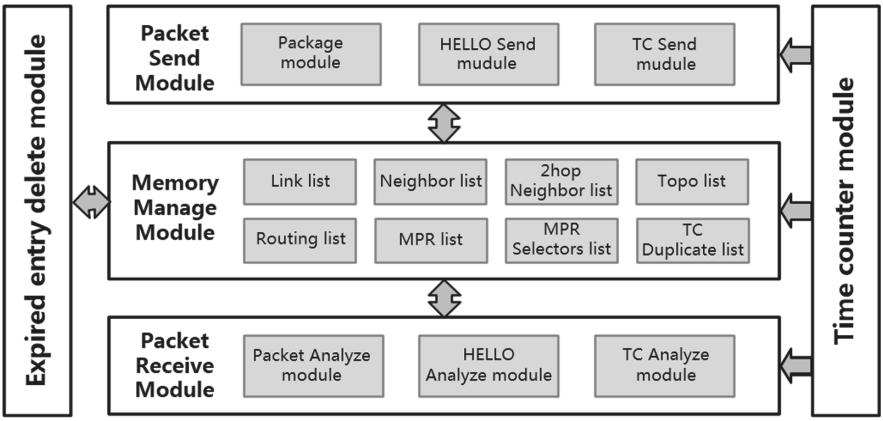

Fig. 1. OLSR routing algorithm module division.

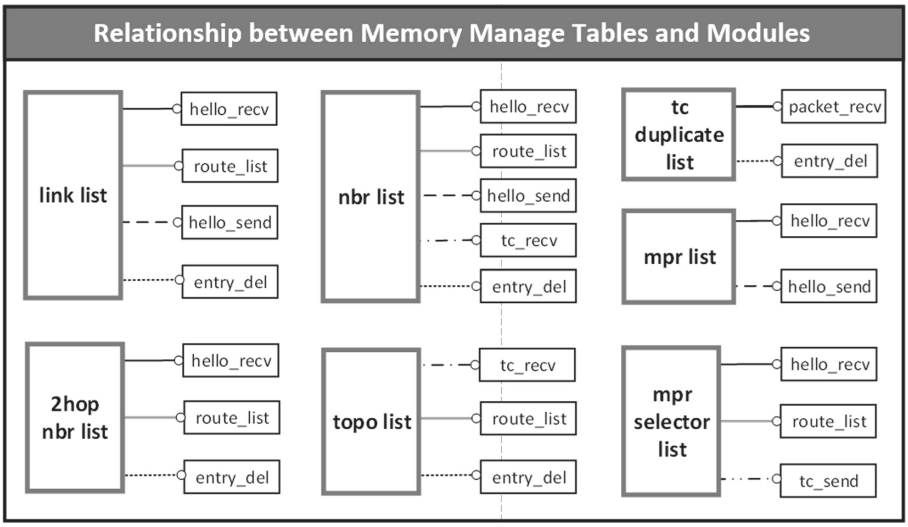

Fig. 2. Relationship between tables and modules.

complete the design. Each storage unit in RAM contains three parts of information: header, key data of this storage table, and address of the previous entry. FIFO is used to store the valid RAM entry addresses. Based on such table structure, referring to the processing idea of the one-way linked list, the modules can add, query, delete, traverse and update the memory management table entries through the read and write operations on the RAM A or B ports.

With reference to Fig. 2 and 3, it can be seen that the memory management table whose main body is true dual-port RAM needs to meet the read and write requirements of multiple modules at the same time. Then, when the read or write

demand exceeds the available number of 2, there will be conflicts that have a significant impact on the result. Therefore, in the implementation of the FPGA-based OLSR protocol, the biggest difficulty is how to solve the conflict problem that occurs when "multiple modules in the OLSR project simultaneously read or write to the same port of the same table" in the case of "less RAM ports with more demand".

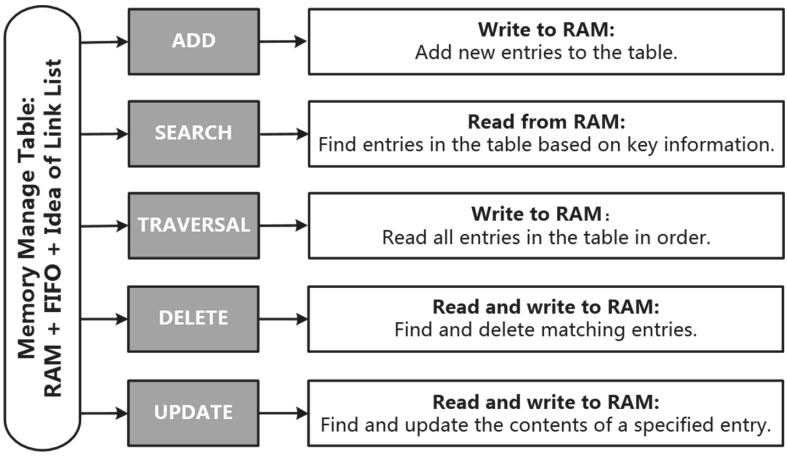

Fig. 3. List function and meaning.

Taking the neighbor table with the most associated modules as an example, the relationship among RAM port, function and module is shown below. As Fig. 4 shows, the arrow on the left indicates whether the A/B port of RAM is read or written when using a function of the table, and the right line connects the function and the module using this function.

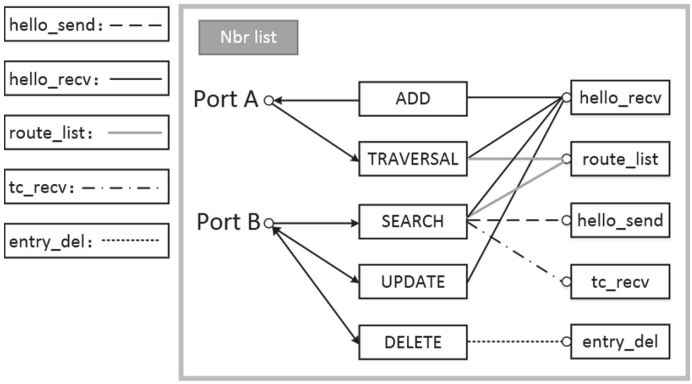

Fig. 4. Port-function-module relationship diagram of the neighbor table.

The port read and write status of the neighbor table RAM is shown in Table 1. It can be seen that the number of read and write requirements of each module in the olsr project for this table greatly exceeds the number of read ports that a true dual-port RAM can provide. In this way, when the situation of "HELLO receive module and route list module use the traversal function of A port at the same time" or "route list, HELLO send, TC receive module use the search function of B port at the same time" occurs, there will be conflicts which can affect the working process of the OLSR protocol.

Table 1. Neighbor table port supply and demand.

Port	Supply	Demand
Port A: read	1	2
Port A: write	1	1
Port B: read	1	6
Port B: write	1	2

3 Solution for Conflicts in the Use of Memory Management

In order to solve the conflict problem that occurs when "multiple modules in the OLSR project simultaneously read or write to the same port of the same table", the following two aspects are considered to improve the memory management module of the original OLSR project.

3.1 Sort Out Timing Relationship

Sort out the timing relationship of the modules associated with each table, and put modules with "clear timing relationship" on the same port. Take the neighbor table as an example. After modifying according to this idea, the "ram port-function-module" relationship changed from the left to the right in Fig. 5. At this time, the function on the RAM port A is only related to the HELLO receive module, and the HELLO receive module has a sequential relationship in the use of these functions, so there is no possibility of conflict. On this basis, if the possibility of modules associated with port B working simultaneously is eliminated, the conflict problem of the use of the neighbor table can be effectively solved.

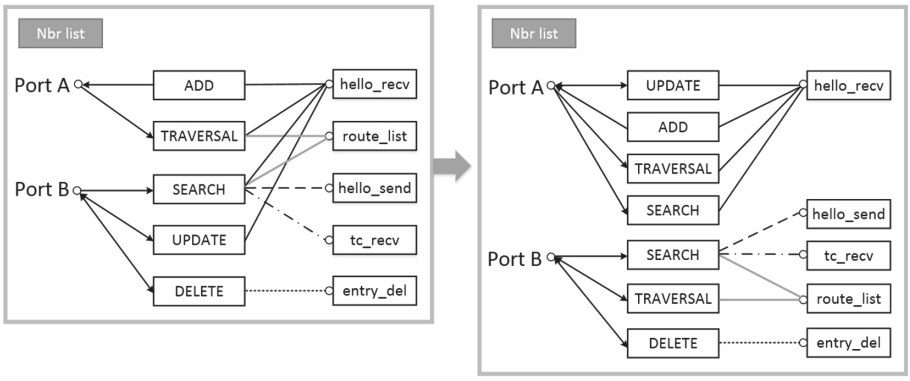

Fig. 5. Port-function-module relationship conversion after time's sorting.

3.2 Add Timing Relationship

If the timing relationship between modules is not clear, add a timing relationship as appropriate. Consequence: bring some performance loss. Considering that it is necessary to ensure that the expired entries can be promptly deleted and the routing table can be accurately updated after the node is moved, the working order of each module is adjusted to:

(1) Periodically start the expired entry deletion module according to HELLO cycle.
(2) After deleting the expired entries, send the HELLO packet and the TC packet successively.
(3) Start the calculation of the routing table.

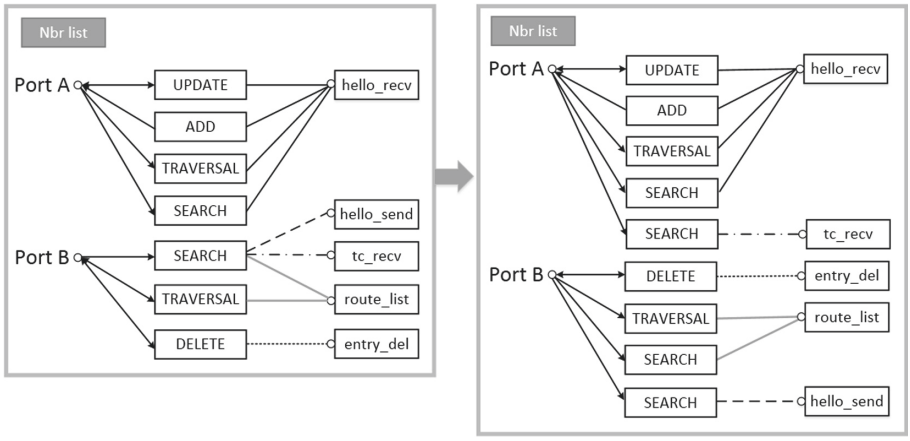

Fig. 6. Port-function-module relationship conversion after adding time relationship.

In this way, the entry delete, route list, HELLO send, and TC receive modules have a sequential relationship, which can basically solve the conflict of port usage of each table. At this time, the port-function-module relationship of the neighbor table can be further transformed as shown in Fig. 6.

For the neighbor table at this time, the three modules associated with port B no longer have the possibility of conflict. Just add the waiting sequence for the HELLO receive and TC receive modules to the read function of port A, all conflicts that may occur in the use of the neighbor table will be resolved. The situation of other tables is simpler than that of neighbor tables. Conflicts can be resolved under the above ideas.

4 Simulation

In this paper, Verilog language is used to design the OLSR project, and the simulation of OLSR project is completed based on virtex-7 device and xc7vx485tfg1157-1 device.

4.1 Functional Simulation and Analysis

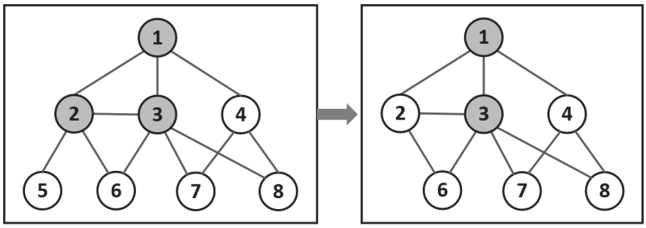

Fig. 7. Simulation scenario of OLSR.

The functional verification scenario of the OLSR project is shown in Fig. 7. According to the protocol content, when nodes 1 to 8 communicate as the neighbor relationship shown on the left, nodes 1, 2, and 3 can be selected as MPR nodes, and their respective routing tables are established. After a period of time (as shown on the right), node 5 leaves and other nodes update their memory management tables. At this time, only nodes 1 and 3 are selected as mpr nodes. This scenario can cover all workflow branches under the OLSR protocolso it has generality.

In the simulation results shown in Fig. 8, there are 7 entries in the routing table of node 1, including one-hop routes to nodes 2, 3, 4 and two-hop routes to nodes 5, 6, 7, 8, which is same as expected. According to the provisions of the agreement on the content of the routing table entry, the symbol ① in the Fig. 8 refers to the address of the local node as 1111aaaa, ② refers to the routing table

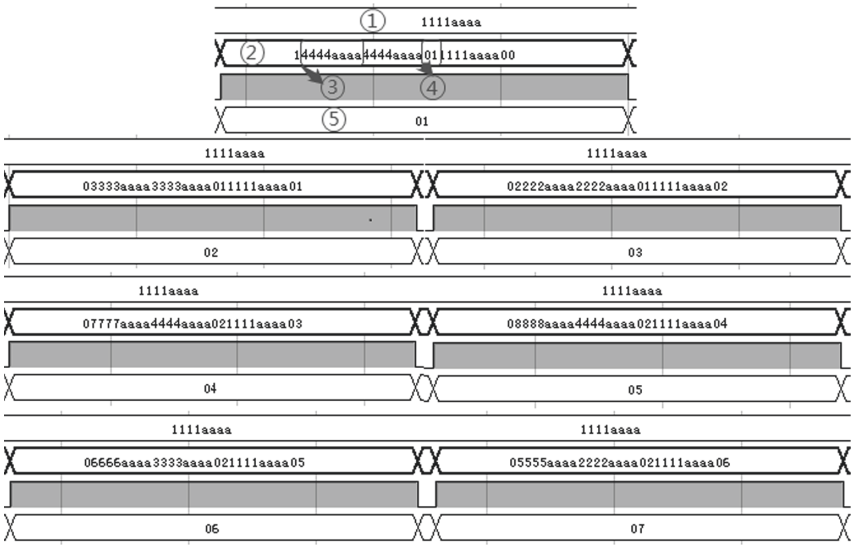

Fig. 8. Routing table of node 1.

entry records the ④ hop route from this node to the node with the address ③, ⑤ refers to the sequence number of the entry in the routing table.

In the scenario on the left side of Fig. 7, the contents of the routing table and mpr table of the 8 nodes are shown in Table 2, and all match the results estimated according to the OLSR protocol.

Table 2. Routing table of 8 nodes.

Node	1hop-route	2hop-route	3hop-route	mpr list
1	2/3/4	5/6/7/8	None	2/3
2	1/3/5/6	4/7/8	None	1/3
3	1/2/6/7/8	4/5	None	1/2
4	1/7/8	2/3	5/6	1
5	2	1/3/6	4/7/8	2
6	2/3	1/5/7/8	4	2/3
7	3/4	1/2/6/8	5	3
8	3/4	1/2/6/7	5	3

Waiting for a period of time, stop the communication between node 5 and other nodes, the scene changes to the right side of Fig. 7. After 2 HELLO sending cycles, the routing tables of 8 nodes are updated correctly. At this time, the storage tables of node 5 are empty, that of the other 7 nodes no longer contain

node 5, and the mpr table no longer contains node 2, which are all in line with expectations. So far, the function verification of FPGA-based OLSR project is successful.

Table 3. Utilization results.

	Name	Power (W)	Utilization (%)
Dynamic (0.299 W)	Clocks	0.088	29
	Signals	<0.001	<1
	Logic	<0.001	<1
	BRAM	0.211	70
	I/O	<0.001	0
Device static	0.250 W		

4.2 Performance Simulation and Analysis

There are two power consumption estimation modes. One is the vector mode, which needs to provide SAIF or VCD files; the other is the non-vector mode, which only needs to provide simple parameters. Considering the estimated speed, Xilinx recommends selecting SAIF files in vector mode. At the same time, in order to obtain more accurate power consumption estimates, the performance analysis below is based on functional simulation after implementation.

The total on-chip power consumption is 0.549 W and the module power consumption is 0.299 W, which is shown in Table 3.

The resource usage of each part on the chip is shown in Table 4. Add clock constraint file for the OLSR project, and the highest clock frequency can be estimated to be 96.395 MHz.

Table 4. Utilization results.

Resource	Utilization/Available	Utilization (%)
LUT	17887/303600	5.89
LUTRAM	118/130800	0.09
FF	21854/607200	3.60
BRAM	73/1030	7.09
IO	229/600	38.17

In summary, using hardware to implement the OLSR protocol takes up very few hardware resources and consumes very low power. The integrated circuit can reach a higher clock frequency and a faster calculation speed.

5 Summary

This article proposes a multi-link list design and FPGA implementation method in OLSR protocol, which can solve the conflict problem that occurs when "multiple modules simultaneously read or write to the same port of the same table" in the case of "less RAM ports with more demand". Through platform simulation, the correctness of the method was verified, and the performance of resource utilization and power consumption of the project was analyzed. Since this method sacrifices the accuracy of the route calculation in a short time, the update of the correct route in the mobile scenario will be delayed by 2 HELLO cycles, the maximum clock frequency also needs to be increased. This research will continue to improve in the future, try to test the OLSR project in the multi-node scenario, and strive for lower power consumption.

Acknowledgement. This work was supported in part by Science and Technology on Avionics Integration Laboratory and the Aeronautical Science Foundation of China (Grant No. 201955053002), the National Natural Science Foundations of CHINA (Grant No. 61871322, No. 61771392, No. 61771390, and No. 61501373), and Science and Technology on Avionics Integration Laboratory and the Aeronautical Science Foundation of China (Grant No. 20185553035).

References

1. Xie, W.: Improvement and implementation of OLSR protocol in ad hoc network. Ph.D. dissertation, South China University of Technology (2010)
2. Zarakovitis, C.C., Ni, Q., Spiliotis, J.: Energy-efficient green wireless communication systems with imperfect CSI and data outage. IEEE J. Sel. Areas Commun. **34**(12), 3108–3126 (2016)
3. Xu, Y.: Ad hoc network performance analysis and routing technology research. Ph.D. dissertation, Xidian University (2014)
4. Clausen, T.: Optimized link state routing protocol, Rfc (2003)
5. Li Chen, R.Y., Pingjun Pan, M.H.: Ad hoc high dynamic routing protocol simulation and research. Mod. Defense Technol. **43**(249)(5), 121–125+174 (2015)
6. Chen, Y.: Comparative study and simulation analysis of wireless ad hoc network routing protocols OLSR and AODV. Comput. Knowl. Technol. **014**(8), 22–24 (2018)
7. Shu, W.: Performance simulation of OLSR protocol based on NS-3. Shandong Ind. Technol. **000**(005), 229–229 (2016)
8. Rathinam, A., Natarajan, V., Vanila, S., Viswanath, A., Guhan, M.S.: An FPGA implementation of improved AODV routing protocol for route repair scheme. IEEE Computer Society (2008)
9. Ramakrishnan, M., Shanmugavel, S.: FPGA implementation of DSDV based router in mobile adhoc network. In: India Conference (2006)