An FPGA Based Reconfigurable MAC Architecture for Universal Short Range Communication Networks

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

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

Abstract. The wireless universal short-distance network refers to a heterogeneous network that combines multiple wireless short-distance networks such as wireless local area networks and wireless personal area networks. Users can choose to use different multiple access protocols to access the network according to the characteristics of the service. This paper proposes an FPGA-based reconfigurable MAC architecture, which uses a combination of software and hardware to select the working mode of the universal MAC in the FPGA through ARM, so as to achieve fast switching between different networks. The proposed universal MAC module includes a control module based on finite state machine and a data frame transceiver module, it implements the function of switching between different networks by configuring the control module as different finite state machine of different MAC protocols. A universal MAC module combining ALOHA protocol and CSMA/CA protocol is designed and implemented in this paper. The simulation results show that the designed universal MAC module is equivalent to the CSMA/CA protocol in terms of resource utilization, and can be flexibly switching between ALOHA protocol and CSMA/CA protocol.

Keywords: Universal MAC · Reconfigurable · FPGA · Universal short range communication networks

1 Introduction

With the continuous development of network technology, wireless short-range networks gradually show its superiority. A variety of wireless network technologies such as Bluetooth, Wi-Fi, 2G, 3G, etc. greatly infiltrate people's lives, and bear more and more user traffic [1]. In the short-distance network, users have various needs, and users often need to switch networks according to different needs. Therefore, it is necessary to design a universal MAC for wireless short-range networks.

Wireless universal short-distance network refers to a heterogeneous network that combines multiple wireless short-distance networks such as wireless local

B. Li et al. (Eds.): IoTaaS 2019, LNICST 316, pp. 20–36, 2020.
https://doi.org/10.1007/978-3-030-44751-9_3

area networks and wireless personal area networks. Users can choose to use different multiple access protocols to access the network according to the characteristics of the service, which can effectively solve the above problem. So far, there have been few cases where the MAC protocol has been integrated through an integrated approach on a unified platform.

In the existing research on UMAC, Ref. [3] based on the European Omega project [2], introduced an integrated programming model of a Universal Media Access Controller (UMAC), which is capable of implementing TMAC and some InterMAC functions [3]. As shown in Fig. 1, this UMAC architecture has many features such as protocol agnosticity, high flexibility, ease of operation, scalability, and high efficiency. Compared with the traditional MAC, the flexibility is greatly improved, which provides the possibility of cross-layer optimization and function improvement of the MAC layer. At the same time, TMAC now provides customized information about link status and channel for better InterMAC functionality.

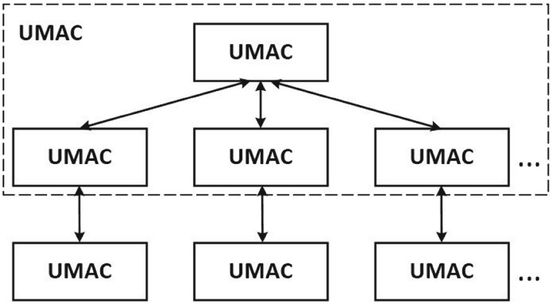


Fig. 1. Omega node architecture [4].

However, the establishment of the UMAC model in this article is implemented in Click using software programming language, which brings the problem of low efficiency.

In the case of program analysis and performance evaluation of the MAC protocol, the theoretical model is mathematically easy to handle, but usually lacks physical layer accuracy. The actual evaluation protocol can be achieved under actual PHY layer conditions through experimental prototyping and testing [5]. There are two types of experimental platforms based on commercial NICs and hardware-based customizations that can implement the actual testing of the MAC protocol, but there exist many problems such as design difficulty and high cost. The OpenMAC platform proposed in Ref. [3] is implemented on a hardware platform based on reconfigurable field programmable gate array (FPGA). The MAC protocol designed by C++ simplifies the prototyping process, thereby reducing the burden of the protocol designer on the hardware and timing. At the same time, the OpenMAC platform introduces a hardware/software partitioning concept based on a shared memory architecture, which simplifies the task, and reduces software latency by concentrating packet processing load on the HW MAC and preventing SW MAC to access to large data payloads. Therefore, in addition to the flexibility and reconfigurability brought by the hardware, the OpenMAC platform simplifies and accelerates the process of implementing the MAC protocol. The concept of hardware and software partitioning has great significance for the design of UMAC.

Compared with the rapid development of mobile device performance, the development of battery capacity is much slower. The energy consumption of the terminal needs to be solved [6]. The Green-T project in Ref. [4] optimizes energy consumption problem by effectively utilizing multiple radio interfaces to solve energy efficiency problems. This provides a reference for the idea of switching between multiple MAC protocols. At the same time, the implementation of the Green-T project still faces multiple challenges. For example, there is a need to balance the trade-off between performance and energy savings. For vertical switching scenarios, if the terminal is able to increase the investment when searching for available networks, then the terminal will have the opportunity to switch to a lower energy network, finding this trade-off is one of the goals of the Green-T project. At the end of the paper, a unified system architecture is proposed. It is hoped to achieve high flexibility and high resource utilization through OpenMAC architecture and energy-saving, flexible baseband physical interface design. This is also a major challenge in the design process of this paper.

Reference [5] proposes a computer-aided wireless sensor network (WSN) evaluation and optimization platform, which can effectively help the simulator to perform network simulation [7]. The simulators for WSN in the existing work are mainly designed for software engineers in the communication field. It is difficult for experts in other fields to master their usage after short-term training, and it is difficult for the simulator to implement specific site simulation. The middleware designed in the article is independent of the specific target architecture. It solves the above problem by abstracting objects in the WSN. The operator only needs to define the characteristics and network of the node in the user interface, then the middleware can realize the process of automatic modeling, data conversion, simulator implementation and network optimization. With this middleware, anyone can easily perform network simulations.

This paper proposes an FPGA-based reconfigurable MAC architecture that uses hardware and software to encapsulate several MAC protocols (Multiple Access Protocols) in reconfigurable field-programmable gate arrays (FPGA). In the development board, the choice of these protocols is written into ARM using a software programming language. This enables flexible use of multiple protocols, and is suitable for coping with the increasing heterogeneity of the network. At the same time, this architecture has the characteristics of high flexibility, low resource consumption rate, and high speed compared to other existing projects.

This paper is divided into five chapters. The first chapter briefly introduces the research content of this paper from research background, main ideas and existing related work. The second chapter specifically designs the FPGA-based reconfigurable MAC architecture proposed in this paper. The third chapter details the design and implementation steps of the universal MAC module based on ALOHA protocol and CSMA/CA protocol, and analyzes its simulation results and performance in all aspects in the fourth chapter. The fifth chapter looks forward to the follow-up research of this article.

2 FPGA-Based Reconfigurable MAC Architecture

2.1 FPGA-Based Reconfigurable MAC Overall Architecture

The following figure shows the integrated programming model of FPGA-based reconfigurable MAC architecture (UMAC) proposed in this paper (Fig. 2).

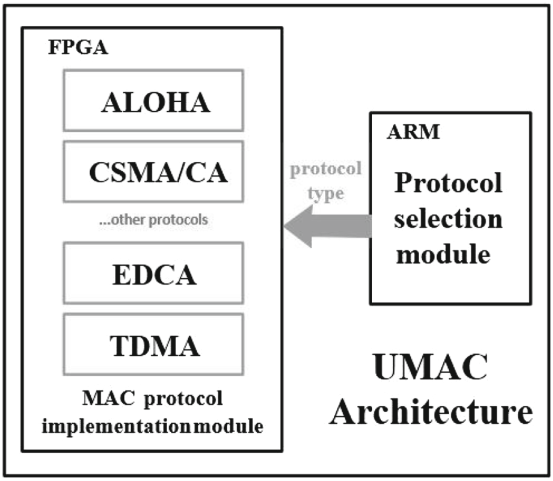


Fig. 2. UMAC Architecture.

The UMAC model is mainly divided into two parts: the protocol selection module and the protocol storage module.

(1) UMAC protocol selection module: the module is recorded into the ARM processor using software programming. The communication protocol is selected according to specific rules, and the selected protocol type is transmitted to the protocol storage module.
(2) UMAC protocol implementation module: this module is used to store the implementation flow of multiple communication protocols. The implementation of the protocol storage module is accomplished by encapsulating several MAC protocols (Multiple Access Protocols) in FPGA board using a hardware programming language.

This paper mainly discusses the UMAC protocol implementation module in the UMAC model.

2.2 UMAC Protocol Implementation Module

Figure 3 shows the system architecture of the MAC layer networking protocol module. The MAC layer networking protocol module is mainly divided into two parts: the MAC protocol module and the memory management module. The MAC protocol module mainly implements the logic function of the MAC protocol. The memory management module buffers and transmits data according to the requirements of the MAC protocol, and decides whether to send the data to the data group frame sending module or the upper layer module. In addition, the memory management module also supports for retransmission of completed data. The MAC protocol module is mainly divided into the sending module and the receiving module, and the memory management module is also divided into the sending memory management module and the receiving memory management module.

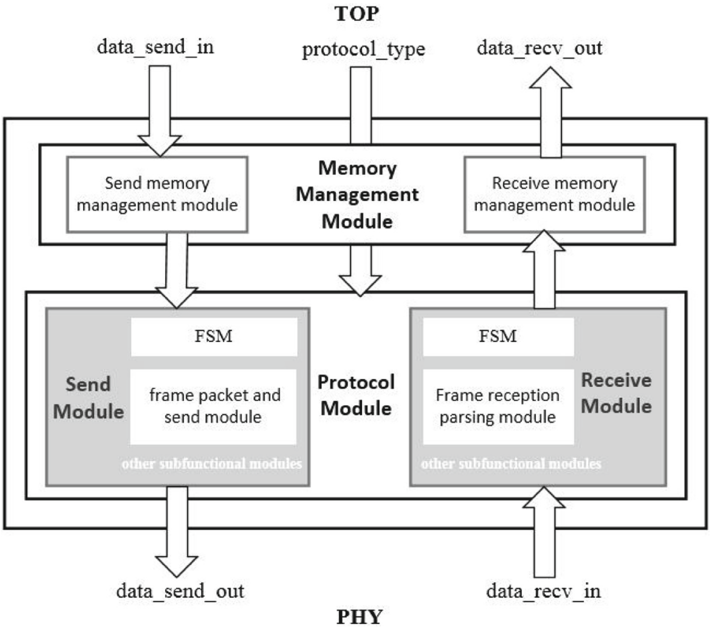

TOP
data_send_in
protocol_type
data_recv_out
Send memory management module
Memory Management Module
Receive memory management module
FSM
Send Module
frame packet and send module
other subfunctional modules
Protocol Module
FSM
Frame reception parsing module
Receive Module
other subfunctional modules
data_send_out
data_recv_in
PHY

Fig. 3. System architecture of the MAC layer network protocol module.

After data arriving from the upper layer, it first enters the sending memory management module for buffering, and then transmits to the sending module of the MAC protocol with the protocol type signal transmitted by the protocol selection module. The transmitting finite state machine control module in the sending module schedules the corresponding multiple function modules to cooperate according to the protocol type signal, and finally completes data encapsulation and sends the data to the PHY layer.

After the data of the PHY layer enters the receiving module of the MAC protocol, the receiving finite state machine control module in the receiving module also cooperates according to the protocol type signal to coordinate the corresponding multiple function modules to work together, completes the receiving and parsing of the data and transmits it to the receiving memory management module for buffering, then finally outputs to the upper layer.

3 Design and Implementation of General MAC Module Based on ALOHA Protocol and CSMA/CA Protocol

In the design of the UMAC protocol implementation module, this paper cites the FPGA design and implementation method of complex communication system based on flow chart proposed in Ref. [7], which is in response to the problem that existing communication protocol and algorithm design is very complicated and lack of general FPGA design and implementation method [8]. FPGA design and implementation methods based on flow chart significantly reduce FPGA development time. The method is mainly divided into the following three parts. Firstly, a modular architecture for control and data phase separation of FPGA is designed [9,10]. Secondly, a flow chart based finite state machine design method is proposed. Finally, a method for packaging a finite state machine into a reusable IP core in FPGA is designed. According to this flowchart-based FPGA design method, this paper takes the simple UMAC protocol implementation module including ALOHA protocol and CSMA/CA protocol as an example to verify and analyze the performance of the UMAC model envisaged in this paper.

First of all, according to the workflow of ALOHA protocol and CSMA/CA protocol, this paper draws its flow chart and divides the state, and then determines its state transition diagram.

3.1 ALOHA Protocol

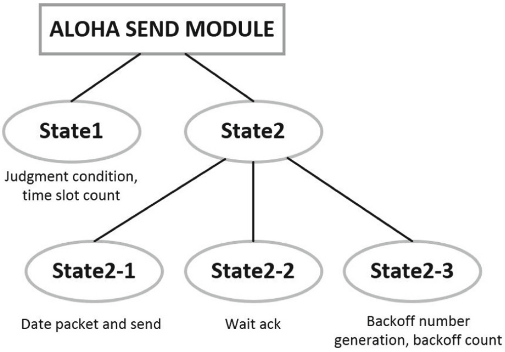


Fig. 4. ALOHA send module tree.

Sending Module. According to the workflow of the ALOHA sending module, combined with the method described in Ref. [7], the tree diagram of the ALOHA protocol sending module shown in Fig. 4 can be obtained. The single functional module consists of five parts:

1. A fixed length data frame encapsulation transmitting module;
2. Time slot division counting module;
3. Random backoff generation counting module;
4. ACK sending module;
5. NAK sending module;

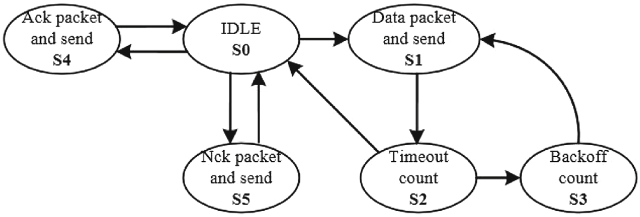


Fig. 5. ALOHA send module main control module finite state machine.

According to the transmission flow chart and the module tree diagram, the state machine transition diagram of the finite state machine as shown in Fig. 5 can be further drawn. The slot division counting module and the data buffer module are not shown in the figure, but should be added in the actual process to control data transmission and data frame buffering. The transition conditions between states are shown in the transition matrix shown in the figure below (Fig. 6).

$$\begin{pmatrix} 0 & \text{There is data to send and new time slot start} & 0 & 0 & \text{Ack send enable coming} & \text{Nck send enable coming} \\ 0 & 0 & \text{Data send over} & 0 & 0 & 0 \\ \text{Ack coming} & 0 & 0 & \text{Count timeout or nck coming} & 0 & 0 \\ 0 & \text{Backoff over} & 0 & 0 & 0 & 0 \\ \text{Ack send over} & 0 & 0 & 0 & 0 & 0 \\ \text{Nck send over} & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 6. ALOHA transmitter module finite state machine state transition matrix.

According to the state transition diagram and the state transition matrix, the state machine control module [2] of the ALOHA transmitting module and other sub-function modules can be determined.

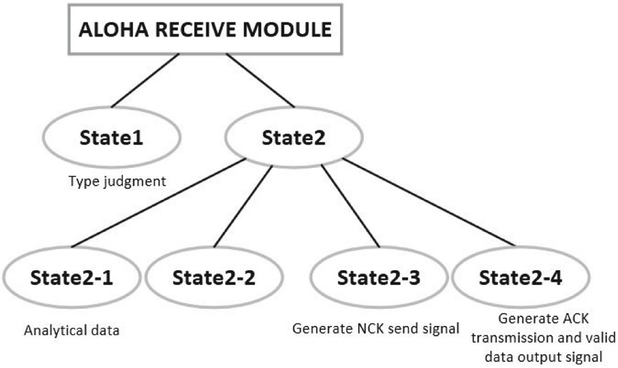


Fig. 7. ALOHA receive module tree.

Receiving Module. In the same way, according to the workflow of the ALOHA receiving module, combined with the method described in Ref. [7], the tree diagram of the ALOHA protocol receiving module shown in Fig. 7 can be obtained. The functional module is composed of two parts:

1. Data type judgment;
2. Data frame parsing module;

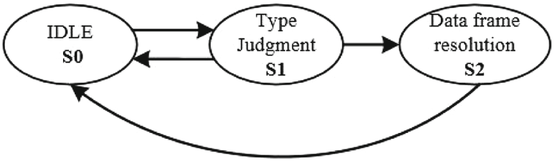


Fig. 8. Receive module main control module finite state machine.

A finite state machine state transition diagram as shown in Fig. 8 can be further drawn according to the reception flow chart and the state division tree diagram. The data cache module is not shown in the figure, but it should be added in the actual process. The transition conditions between states are shown in the transition matrix shown in the figure below (Fig. 9).

$$\begin{pmatrix} \text{Data coming} & 0 & 0 \\ \begin{array}{c}\text{Datatype is}\\ \text{not data}\end{array} & \begin{array}{c}\text{Datatype is}\\ \text{data}\end{array} & 0 \\ \begin{array}{c}\text{Data analysis}\\ \text{completed}\end{array} & 0 & 0 \end{pmatrix}$$

Fig. 9. ALOHA receive module finite state machine state transition matrix.

According to the state transition diagram and the state transition matrix, the state machine control module and other sub-function modules of the ALOHA receiving module can be determined.

3.2 CSMA/CA Protocol

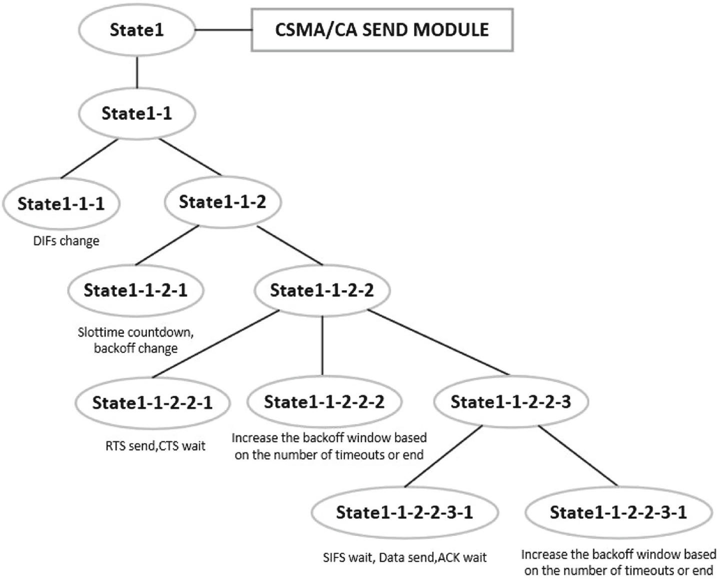

State1
CSMA/CA SEND MODULE
State1-1
State1-1-1
DIFs change
State1-1-2
State1-1-2-1
Slottime countdown, backoff change
State1-1-2-2
State1-1-2-2-1
RTS send,CTS wait
State1-1-2-2-2
Increase the backoff window based on the number of timeouts or end
State1-1-2-2-3
State1-1-2-2-3-1
SIFS wait, Data send,ACK wait
State1-1-2-2-3-1
Increase the backoff window based on the number of timeouts or end

Fig. 10. CSMA/CA send module tree.

Sending Module. Similarly, according to the workflow of the CSMA/CA transmission module, combined with the method described in Ref. [7], the tree diagram of the CSMA/CA protocol transmission module shown in Fig. 10 can be obtained.

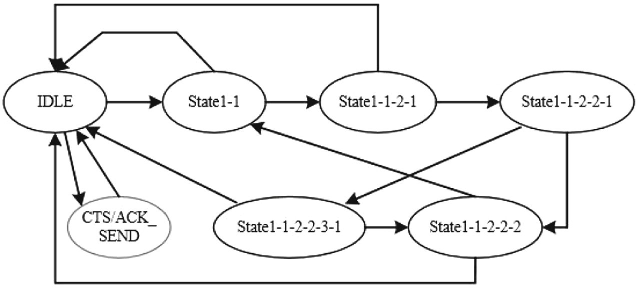


Fig. 11. CSMA/CA send module main control module finite state machine.

According to the transmission flow chart and the module tree diagram, the finite state machine state transition diagram shown in Fig. 11 can be further drawn. The transition conditions between states are shown in the transition matrix shown in the figure below (Fig. 12).

$$\begin{pmatrix}
0 & \text{There is data to send and channel is idle} & 0 & 0 & 0 & 0 & \text{CTS/RTS need to be send} \\
\text{Channel is busy} & 0 & \text{DIFS=0} & 0 & 0 & 0 & 0 \\
\text{Channel is busy} & 0 & 0 & \text{BACKOFF=0} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \text{time out} & \text{CTS receives right} & 0 \\
\text{The number of timeouts exceeds the maximum} & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{Ack receive right} & 0 & 0 & 0 & \text{time out} & 0 & 0 \\
\text{CTS/ACK send over} & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Fig. 12. CSMA/CA transmitter module finite state machine state transition matrix.

According to the state transition diagram and the state transition matrix, the state machine control module and other sub-function modules of the CSMA/CA transmission module can be determined.

Receiving Module. Similarly, the receiving module tree diagram shown in Fig. 13 can be made according to the working process of the receiving module, and then the state transition diagram is made. As shown in Fig. 14, the jumping conditions between the states are showed in state transition matrix of Fig. 15.

According to the state transition diagram and the state transition matrix, the state machine control module and other sub-function modules of the CSMA/CA receiving module can be determined.

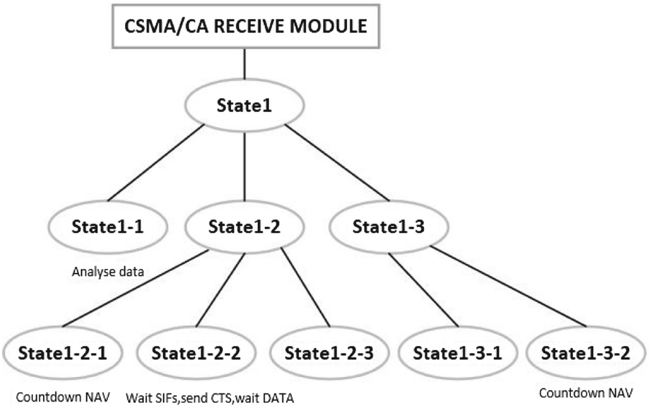


Fig. 13. CSMA/CA receive module tree.

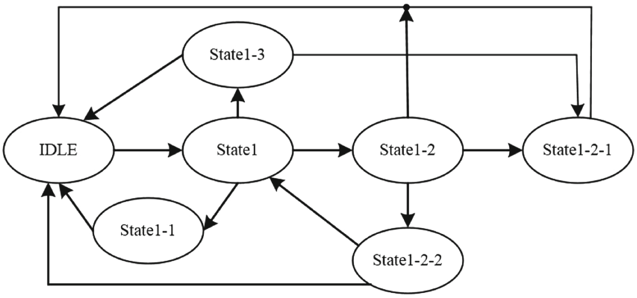


Fig. 14. Receive module main control module finite state machine.

$$\begin{pmatrix}
\text{Frame is coming} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \text{type is data} & 0 & \text{type is rts} & \text{type is cts/ack} \\
\text{Nav=0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \text{Sifs wait over} & 0 & 0 & 0 & 0 \\
\text{Data wait over} & \text{Data is coming} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \text{data receive over and data is valid} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{Rts receive over and rts is valid} & 0 & \text{rts is valid and send to this node} & \text{rts is valid but not send to this node} & 0 & 0 & 0 & 0 & 0 \\
\text{cts/ack receive over and rts/ack is valid} & 0 & \text{cts/ack receive over but not rts/ack is valid} & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Fig. 15. Receive module finite state machine state transition matrix.

3.3 UMAC Protocol Implementation Module Based on ALOHA Protocol and CSMA/CA Protocol

The expected function of the UMAC protocol implementation module based on ALOHA protocol and CSMA/CA protocol designed in this paper is: the current working protocol can be selected according to the input protocol type signal. If the protocol type is 0, the ALOHA protocol is used for data interaction. When the protocol type is 1, the CSMA/CA protocol is used for data interaction. The overall workflow of the module is shown in Fig. 16.

After the memory management module transmits the data and the protocol type signal to the protocol module, it first enters the state machine control module of the sending module. The module selects a corresponding protocol transmission state machine according to the protocol type signal, and connects the state and state transition conditions included in the protocol to the state machine IP core, so that the involved sub-module completes the data transmission according to the protocol.

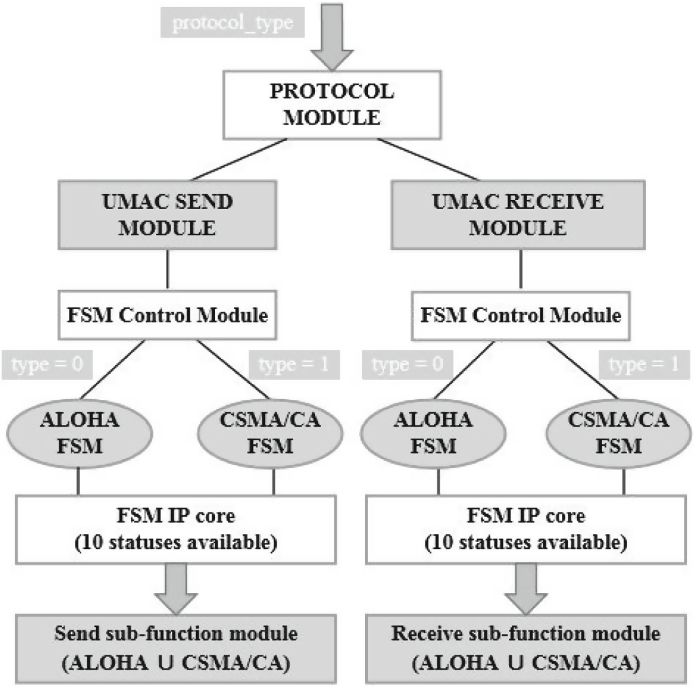


Fig. 16. UMAC Protocol Architecture.

Similarly, when data arrives, the memory management module transmits the data and protocol type signals that need to be received to the protocol module, and then first enters the state machine control module of the receiving module. The module selects a corresponding protocol receiving state machine according to the protocol type signal, and connects the state and state transition conditions included in the protocol to the state machine IP core, so that the involved submodule completes the data reception according to the protocol.

The overall module code architecture is shown in Fig. 17, where the subfunction module section contains all the sub-function modules of a single ALOHA project and CSMA/CA project.

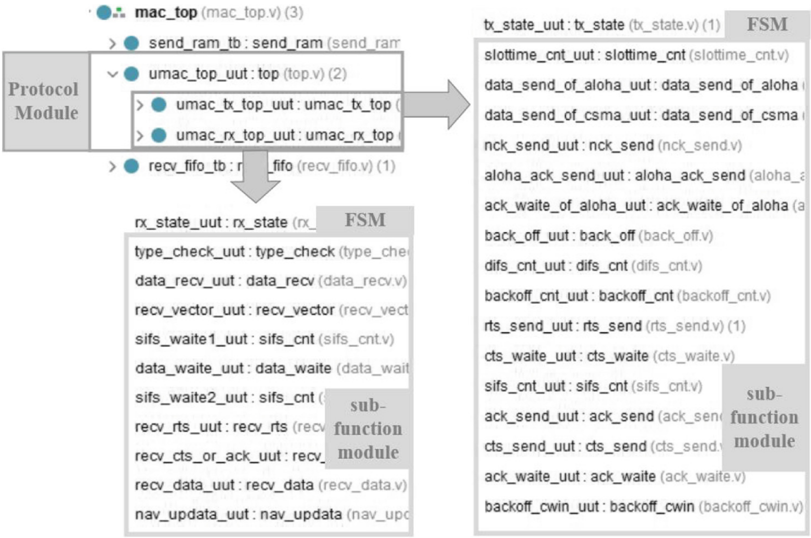

mac_top (mac_top.v) (3)
send_ram_tb : send_ram (send_ram
Protocol Module
umac_top_uut : top (top.v) (2)
umac_tx_top_uut : umac_tx_top (
umac_rx_top_uut : umac_rx_top (
recv_fifo_tb : r fifo (recv_fifo.v) (1)
rx_state_uut : rx_state (rx_
FSM
type_check_uut : type_check (type_che
data_recv_uut : data_recv (data_recv.v)
recv_vector_uut : recv_vector (recv_vect
sifs_waite1_uut : sifs_cnt (sifs_cnt.v)
data_waite_uut : data_waite (data_wait
sifs_waite2_uut : sifs_cnt (
recv_rts_uut : recv_rts (recv
recv_cts_or_ack_uut : recv_
recv_data_uut : recv_data (recv_data.v)
nav_updata_uut : nav_updata (nav_upd
sub-function module
tx_state_uut : tx_state (tx_state.v) (1)
FSM
slottime_cnt_uut : slottime_cnt (slottime_cnt.v)
data_send_of_aloha_uut : data_send_of_aloha (
data_send_of_csma_uut : data_send_of_csma (
nck_send_uut : nck_send (nck_send.v)
aloha_ack_send_uut : aloha_ack_send (aloha_
ack_waite_of_aloha_uut : ack_waite_of_aloha (a
back_off_uut : back_off (back_off.v)
difs_cnt_uut : difs_cnt (difs_cnt.v)
backoff_cnt_uut : backoff_cnt (backoff_cnt.v)
rts_send_uut : rts_send (rts_send.v) (1)
cts_waite_uut : cts_waite (cts_waite.v)
sifs_cnt_uut : sifs_cnt (sifs_cnt.v)
ack_send_uut : ack_send (ack_sen
cts_send_uut : cts_send (cts_send.
ack_waite_uut : ack_waite (ack_waite.v)
backoff_cwin_uut : backoff_cwin (backoff_cwin.v)
sub-function module

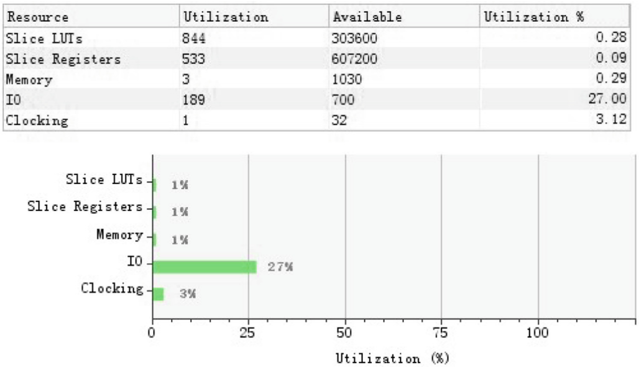
Fig. 17. Code Architecture.

4 Simulation Results and Performance Analysis

This chapter carries out specific simulation verification on the UMAC protocol implementation module based on ALOHA protocol and CSMA/CA protocol designed in Sect. 3.3. The module was written on the Virtex-707 development board of Vivado 2015.2 using verilog language, and the correctness of its function was verified by simulation test.

Now we prove the performance of the UMAC model by comparing the resource consumption of the module with the single MAC protocol module and the UMAC protocol.

By analyzing the resource consumption of the after comprehensive designed ALOHA project that implements the ALOHA protocol, the resource utilization table shown in the following figure can be obtained (Fig. 18).



Resource	Utilization	Available	Utilization %
Slice LUTs	844	303600	0.28
Slice Registers	533	607200	0.09
Memory	3	1030	0.29
IO	189	700	27.00
Clocking	1	32	3.12

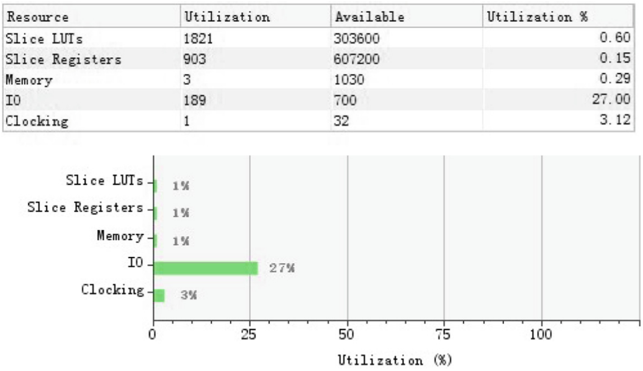
Fig. 18. Resource Utilization Percentage of ALOHA.

Similarly, the resource consumption analysis of the after comprehensive designed CSMA/CA project can be obtained as shown in Fig. 19.

The resource utilization of the UMAC protocol implementation module designed and implemented in this paper is shown in Fig. 20.

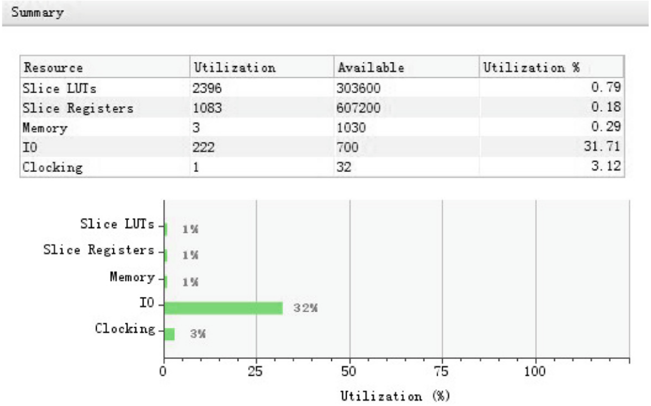
Thus, we can get a comparison of resource utilization as shown in the Fig. 21.

By comparing the resource consumption rates of various aspects of ALOHA project, CSMA/CA project and UMAC protocol in the comparison table, it can be seen that although the consumption rate of the UMAC protocol implementation module in Slice LUTs is twice that of the ALOHA project, it is slightly lower than that of the CSMA/CA project. The situation of Slice Registers is similar. Although the UMAC protocol implementation module is slightly higher than the ALOHA project, it is still lower than the CSMA/CA project. At the same time, the consumption of the UMAC protocol implementation module in IO is slightly higher than that of the ALOHA and CSMA/CA projects, because



Resource	Utilization	Available	Utilization %
Slice LUTs	1821	303600	0.60
Slice Registers	903	607200	0.15
Memory	3	1030	0.29
IO	189	700	27.00
Clocking	1	32	3.12

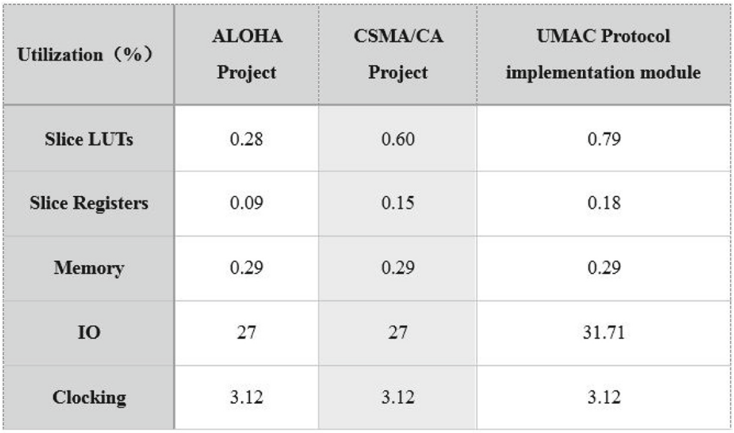
Fig. 19. Resource Utilization Percentage of CSMA/CA.



Summary

Resource	Utilization	Available	Utilization %
Slice LUTs	2396	303600	0.79
Slice Registers	1083	607200	0.18
Memory	3	1030	0.29
IO	222	700	31.71
Clocking	1	32	3.12

Fig. 20. Resource Utilization Percentage of UMAC.



Utilization（%）	ALOHA Project	CSMA/CA Project	UMAC Protocol implementation module
Slice LUTs	0.28	0.60	0.79
Slice Registers	0.09	0.15	0.18
Memory	0.29	0.29	0.29
IO	27	27	31.71
Clocking	3.12	3.12	3.12

Fig. 21. Resource Utilization Comparison.

the IO here is not constrained to the corresponding pin. If it is constrained to the pins by multiplexing, the IO port can be greatly reduced. In addition, the resource consumption in Memory and Clocking is the same. Therefore, we can conclude that the resource consumption rate of the UMAC protocol implementation module designed in this paper is slightly higher than that of a single protocol implementation project, but it is much smaller than the sum of a single protocol project, that is, the UMAC protocol implementation module can reduce the resource consumption rate.

Based on the above analysis, it can be seen that the UMAC protocol implementation module has the characteristics of high flexibility and low resource consumption rate while realizing the switchable functions of ALOHA and CSMA/CA protocols. This paper takes the simple UMAC protocol implementation module including ALOHA protocol and CSMA/CA protocol as an example to verify the performance of the UMAC model envisaged in this paper. If the state machine can be deeply integrated with multiple protocols, more interprotocol switching can be achieved, which greatly reduces the resource consumption rate. This will be the future direction of this paper.

5 Conclusion

In this paper, an FPGA-based reconfigurable MAC architecture is proposed for network heterogeneity. The combination of software and hardware is adopted to select the working mode of the general MAC in the FPGA through ARM, so as to achieve fast switching between different networks. In order to verify the feasibility of the above ideas, this paper designs and implements a general MAC protocol implementation module combining ALOHA protocol and CSMA/CA protocol, which can quickly switch between ALOHA protocol and CSMA/CA protocol by controlling an external signal. The simulation results show that the module is equivalent to the CSMA/CA protocol in terms of resource utilization, which indicates that the module has the characteristics of high flexibility, high speed and low resource consumption rate.

Acknowledgement. This work was supported in part by the National Natural Science Foundations of CHINA (Grant No.61771392, 61771390, 61871322, 61501373 and 61271279), the National Science and Technology Major Project (Grant No. 2015ZX03002006-004 and 2016ZX03001018-004), and Science and Technology on Avionics Integration Laboratory (Grant No. 20185553035).

References

1. O'Brien, D.C., et al.: Home access networks using optical wireless transmission. In: IEEE International Symposium on Personal (2008)
2. Javaudin, J., Bellec, M., Varoutas, D., Suraci, V.: Omega ICT project: towards convergent gigabit home networks. In: 2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1–5, September 2008
3. Loeb, H., Liß, C., Rckert, U., Sauer, C.: UMAC a universal MAC architecture for heterogeneous home networks. In: 2009 International Conference on Ultra Modern Telecommunications Workshops, pp. 1–6, October 2009
4. Gallego, F.V., Alonso-Zarate, J., Liss, C., Verikoukis, C.: OpenMAC: a new reconfigurable experimental platform for energy-efficient medium access control protocols. IET Sci. Meas. Technol. **6**(3), 139–148 (2012)
5. Antonopoulos, A., et al.: Green-T: enabling techniques for energy efficient mobile terminals. In: 2012 IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), pp. 206–210, September 2012
6. Liang, Y., Ying, R., Liu, P.: Efficient middleware for network evaluation and optimization in wireless sensor network design. In: 2013 IEEE International Symposium on Circuits and Systems (ISCAS 2013), pp. 1749–1752, May 2013

7. Hangchao, J.: Design and implementation of a hybrid frequency hopping multiple access protocol for wireless ad hoc network. Master's thesis, Northwestern Polytechnical University (2018)
8. Burgun, L., Dictus, N., Lopes, E.P., Sarwary, C.: A unified approach for FSM synthesis on FPGA architectures. In: Proceedings of Twentieth Euromicro Conference. System Architecture and Integration, pp. 660–668, September 1994
9. Mihhailov, D., Sudnitson, A., Tarletski, K.: Web-based tool for FSM encoding targeting low-power FPGA implementation. In: International Conference on Microelectronics (2010)
10. Sudnitson, A., Mihhailov, D., Kruus, M.: Advanced topics of FSM design using FPGA educational boards and web-based tools. In: 2010 East-West Design Test Symposium (EWDTS), pp. 514–517, September 2010