JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, VOL.16, NO.6, DECEMBER, 2016
https://doi.org/10.5573/JSTS.2016.16.6.719

ISSN(Print) 1598-1657
ISSN(Online) 2233-4866

Implementation of IEEE 802.11ac Down-link MU-MIMO WLAN MAC using Unified Design Methodology

Chulho Chung[1], Yunho Jung[2], and Jaeseok Kim[1]

Abstract—**This paper proposes a unified medium access control (MAC) design methodology and presents the implementation of the IEEE 802.11ac down-link multi-user multi-input and multi-output wireless local area network MAC using the proposed design methodology. The proposed methodology employs unified code for both network simulation and system implementation. Because the unified code closely relates these two processes, the performance of the implemented MAC system can be estimated before implementation. The MAC architecture for an access point implemented using the proposed design methodology is verified on an ARM-based platform, and it is applied to a 65 nm CMOS library.**

Index Terms—**WLAN, IEEE 802.11ac, MAC, MU-MIMO, unified design methodology**

I. Introduction

IEEE 802.11ac is one of the latest amendments for very high throughput wireless local area networks (WLANs), i.e., at least 1 Gbps multi-user (MU) throughput and 500 Mbps single-user (SU) throughput, and the achievable physical layer (PHY) data rate is close to 7 Gbps. In particular, it employs a down-link MU multi-input and multi-output (DL MU-MIMO) scheme. The DL MU-MIMO scheme combined with frame aggregation enables an access point (AP) to simultaneously transmit multiple frames to multiple stations (STAs) via different spatial streams, increasing the MU throughput [1, 2].

For a medium access control (MAC) protocol, in general, existing design methodologies involve two independent steps: analyzing the network performance using the simplistic MAC model and implementing the MAC system. In this case, the consistency of results cannot be guaranteed because the code for each process differs considerably. Hence, there is a growing demand for a new design methodology that closely relates network simulation to system implementation.

There are several approaches to co-simulate systems over a network by combining a physical simulator (i.e., MATLAB, ModelSim) and a network simulator (i.e., OPNET, NS-2) [3, 4]. However, they integrate pre-implemented hardware code or simulation models with the network simulator and do not provide a means to evaluate network performance before hardware implementation. Therefore, this paper proposes a new methodology for designing the MAC protocol that closely correlates network simulation and system implementation, whereby the performance of the MAC system can be estimated before implementation. In addition, this paper presents the implementation results of a network simulator as well as the system hardware and software for IEEE 802.11ac DL MU-MIMO WLAN MAC using the unified design methodology.

The remainder of this paper is organized as follows. Section II provides a brief overview of IEEE 802.11ac

Manuscript received Aug. 27, 2015; accepted Dec. 2, 2015
[1] School of Electrical & Electronic Engineering, Yonsei University, Seoul, Korea.
[2] School of Electronics and Information Engineering, Korea Aerospace University, Goyang-si, Korea
E-mail : jaekim@yonsei.ac.kr

MAC. Section III presents the proposed unified design methodology. Section IV presents the software implementation for the network simulator and field programmable gate array (FPGA) as well as very large scale integration (VLSI) implementation results of the MAC system. We conclude this paper in Section V.

II. OVERVIEW OF IEEE 802.11AC MAC

One of the key enhancements of the MAC feature in IEEE 802.11ac is the transmission opportunity (TXOP) sharing mode which is a new enhanced distributed channel access (EDCA) TXOP mode added for MU-MIMO [5]. In this mode, when frames belonging to the granted access category (AC), namely primary AC, are transmitted via an MU-PHY convergence protocol (PLCP) protocol data unit (MU-PPDU), the frames belonging to other ACs (secondary AC) can also be included. The TXOP sharing mode only applies to an AP.

IEEE 802.11ac also enhances the frame aggregation of IEEE 802.11n. In order to improve MAC efficiency for higher PHY data rates, the maximum frame length is significantly extended; the aggregate-MAC service data unit (A-MSDU) length is extended up to 11,426 bytes, and the aggregate-MAC protocol data unit (A-MPDU) length is increased up to 1,048,575 bytes.

III. UNIFIED MAC DESIGN METHODOLOGY FOR DL MU-MIMO WLAN MAC

In general, the conventional MAC design process is divided into two steps: (1) designing the protocol behaviors and evaluating the performance of interworking with other network protocols, and (2) determining the system specification and implementing the system in accordance with the pre-defined specification. Fig. 1 shows the conventional design flow of the MAC protocol.

The performance evaluation of a protocol is carried out using network simulators such as NS-2, NS-3, or OPNET. A network simulator can simulate various network protocols and application environments; thus, it facilitates network development and performance analysis of the protocol to be implemented [6]. Because the source code used in a network simulator is simply modeled on the basis of the major functions of the

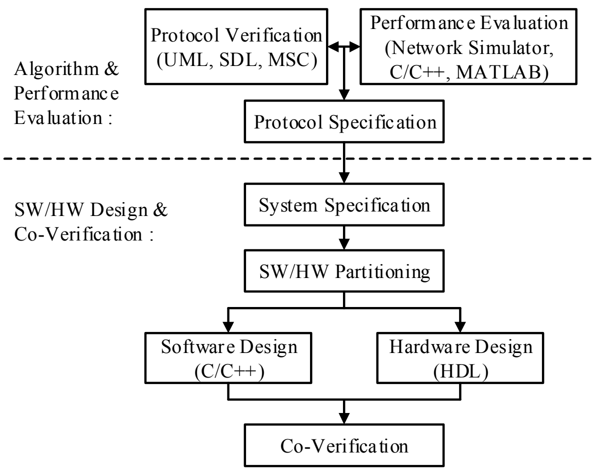

Algorithm & Performance Evaluation :
Protocol Verification (UML, SDL, MSC)
Performance Evaluation (Network Simulator, C/C++, MATLAB)
Protocol Specification
SW/HW Design & Co-Verification :
System Specification
SW/HW Partitioning
Software Design (C/C++)
Hardware Design (HDL)
Co-Verification

Fig. 1. Conventional MAC Design Flow.

protocol in general, it differs significantly from the code used for system implementation. Therefore, in conventional MAC implementation, the codes for the system software and hardware should be newly implemented after the target system specification is determined [7, 8]. Owing to the difference of each code for performance evaluation by network simulation and for system implementation, it is difficult to evaluate the performance of the system to be implemented before complete implementation of the system and network. Moreover, it is difficult to compare the results of network simulation and system evaluation.

In order to overcome this problem, a unified design methodology that uses closely correlated code for both protocol evaluation and system implementation is proposed. In the proposed methodology, first, the unified code for network simulation and system implementation is developed in C/C++ even though it can be generated from the specification and description language (SDL) because it is not an optimized code and does not reflect the system architecture [9-11]. The code includes every function of the MAC protocol described in this paper, including the functions to be implemented to hardware, as well as specialized functions for network simulation and system implementation. Because the implemented code reflects the processing speed of a specific target system, more accurate results can be obtained during performance evaluation. The design specification of the target system and the partitioning of the MAC software and hardware functions can be adjusted according to these results. The functional code of the hardware MAC component is converted into Verilog hardware

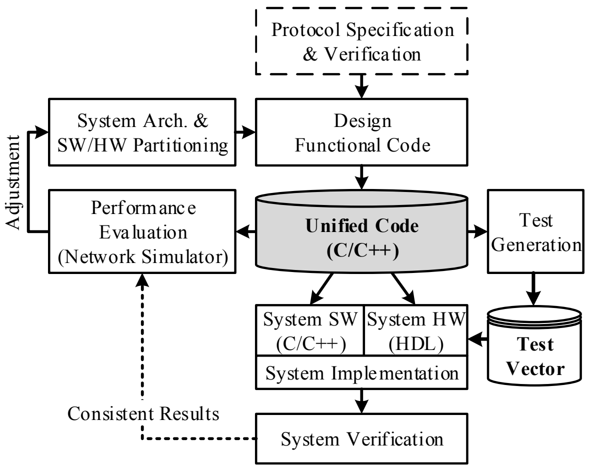


Fig. 2. Proposed Unified MAC Design Flow.

description language (HDL). By using the test vectors generated by the same unified code, the functionalities and output results can be comparable; hence, conformity can be check easily. The overall design flow of the proposed unified MAC design methodology is shown in Fig. 2. Because the unified code is used as the basis of all processes as opposed to Fig. 1, consistent results can be obtained from the performance evaluation and system verification, and the performance of the system to be implemented can be estimated.

In order to integrate the unified code with the network simulator, several considerations are required depending on the network simulator employed. In this study, the NS-2 was used for network simulation; the considerations are specific to the NS-2 simulator. However, they are also applicable to other network simulators in general.

The NS-2 simulator provides system tasks such as packet delivery functions and event timer functions. The event scheduler proceeds with the simulation by delivering the Packet class containing the event time to the process and by calling the related function at the specified time, which is set according to a specific event. In order to apply the implemented code to the NS-2 simulator, the following two factors should be considered: the Packet class and the protocol timers handled by the event timer.

The NS-2 simulator exchanges information between layers using the Packet class. The Packet class generally includes information such as event ID, length of event, transmission time, and protocol headers. However, an actual data is usually not included in it. Because the

```
Class Packet: public Event {
Packet*          mac802_11ac_pkt[ ];    (for user0~3)
Unsigned int     mac802_11ac_numpkt; (for user0~3)
Unsigned char*  mac802_11ac_data;      (for user0~3)
Unsigned int     mac802_11ac_length;  (for user0~3)}
```

Fig. 3. Modified Packet class of NS-2 (packet.h).

implemented MAC code has to process actual data streams, the pointer of the data array should be included. In addition, a single Packet class should contain multiple Packet class pointers to support frame aggregation adopted in IEEE 802.11ac. Fig. 3 shows the modified Packet class. After information of the Packet class (i.e., frame length and transmission time) is updated, the Packet class is delivered to other layers using packet delivery functions such as "downtarget_" and "uptarget_".

The NS-2 simulator provides timer functions by using the event scheduling mechanism. By employing these timer functions, it is possible to implement the MAC protocol timers and the latency time reflecting system characteristics such as MAC/PHY processing time, which are difficult to implement using only software.

In order to adopt the unified code for system development, it is necessary to consider the target system specification, system interface of data input/output and control signals, and test vector generation. This paper presents the implementation of IEEE 802.11ac MAC on an embedded processor-based system. The MAC software is processed by the firmware and connected to the MAC hardware through a bus interface and an interrupt signal line. Therefore, the interrupt handling procedures and read/write operations for the memories and internal registers of the MAC hardware should be implemented as separate functions in order to easily convert the hardware component of the unified code into Verilog HDL.

The MAC hardware is divided into two components: the data processing engine and the controller. The conformity of the data processing engine is verified by comparing the input/output vectors generated from the unified code for each block. The controller consists of finite state machines. The identity of the unified code and the converted code can be verified by comparing their state transitions. Implementation of the state transition condition as separate functions facilitates verification and conversion of the code.

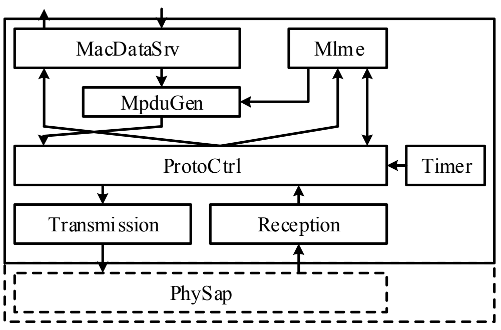


Fig. 4. Architecture of IEEE 802.11ac MAC unified code.

IV. Implementation of IEEE 802.11ac MAC

First, the unified code is developed in C/C++. Because the developed code reflects the considerations for network simulation and system implementation, as discussed in the previous section, it can be easily applied to the NS-2 simulator and the system implementation.

1. Software Implementation for Network Simulator

The architecture of the IEEE 802.11ac MAC code implemented using the proposed methodology is shown in Fig. 4. The unified code includes PHY service access point (SAP) to process a data frame through PHY.

The MacDataSrv block receives an MSDU to be transmitted from logical link control (LLC) and delivers the successfully received MSDU to LLC. The functions are verified by the vectors corresponding to the test scenario used in network simulation.

The Mlme block operates management functions, i.e., connection, power management, and link adaptation, and generates management frames that are used for these functions. Further, it stores information about the parameter values and statuses for MAC processing.

The MpduGen block generates an MPDU, and stores it in transmission queues, existing as a linked list, with information for transmission. This block also applies the A-MSDU scheme by judging whether an A-MSDU can be generated when the MPDU is being stored.

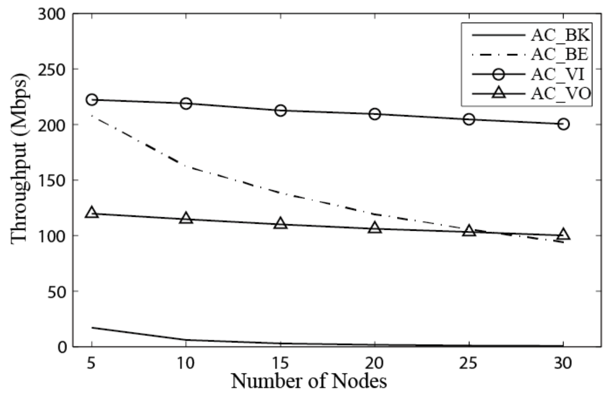
The ProtoCtrl block is in charge of the core MAC function, i.e., making decisions regarding channel access and frame transmission. It monitors the channel status and controls channel access for frame transmission, which is known as EDCA. This block also makes decisions regarding the use of request-to-send (RTS)/clear-to-send (CTS), fragmentation, frame aggregation, and TXOP sharing by analyzing the information of frames that are ready to be transmitted. The response of the received frame, such as an acknowledgment, is also accomplished by this block. Furthermore, this block restores the frame to the transmission queue or discards it after determining whether it is to be retransmitted owing to an error.

Frames that have been selected for transmission by the ProtoCtrl block are delivered to PHY by the Transmission block. Frame check sequence (FCS) is computed simultaneously, based on the 32-bit cyclic redundancy code (CRC), and added at the end of every MPDU. When the A-MPDU mechanism is used, this block generates an A-MPDU subframe including an MPDU delimiter with 8-bit CRC and conveys it to PHY.

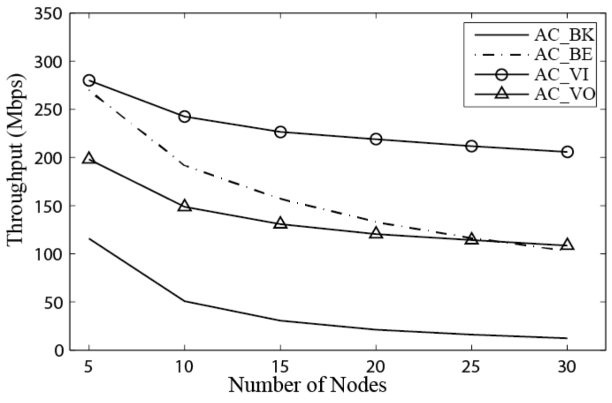
The Reception block analyzes the PLCP service data unit (PSDU) received from PHY. On receiving an A-MPDU, first, it divides PSDU into units of MPDU. Each MPDU is checked for errors and duplication. The successfully received MPDU is delivered to the MacDataSrv block and the statuses are notified to the ProtoCtrl block for the response procedure.

The Timer block includes the timer functions of the MAC protocol, such as slot, inter-frame space (IFS), response timeout, RTS timeout, network allocation vector (NAV), and time synchronization function (TSF). They are applied to the event timers of the NS-2 simulator.

In order to evaluate the network performance by the implemented network simulator, the simulation environments are configured as follows. Each station including AP has saturation traffic belonging to four ACs (i.e., AC_BK (background), AC_BE (best effort), AC_VI (video), and AC_VO (voice)) and the EDCA parameters, such as arbitration IFS (AIFS), contention window (CW), and retry limit, are set to $AIFS_i$ = [79,61,43,34]µs, $TXOPLimit_i$ = [0,0,3008,1504]µs, CWmin = 31, CWmax = 1023 and RetryLimit = 7. The variable bit rate traffic is generated with variable length (average 1,200 bytes). The secondary ACs for the TXOP sharing mode are arranged in descending order of priority, and traffic belonging to different ACs is destined to different destination STAs. Each station has two antennas while an AP has eight



(a) without TXOP sharing mode



(b) with TXOP sharing mode

Fig. 5. Saturation throughput for each AC w/wo TXOP sharing mode.

antennas and supports TXOP sharing mode. The channel bandwidth is set to 80 MHz and the modulation and coding scheme index is set to 9, i.e., the PHY data rate is 780 Mbps for each user and up to 3.12 Gbps for MU.

Fig. 5 shows the saturation throughput of each AC of the overall network according to the TXOP sharing mode. The throughput of each AC increases significantly with the TXOP sharing mode because a secondary AC can be transmitted via MU-PPDU.

2. System Implementation of IEEE 802.11ac MAC

The IEEE 802.11ac MAC architecture for an AP, which supports the TXOP sharing mode with DL MU-MIMO (up to 3-stream MIMO for SU and 8-stream MIMO for MU), implemented using the proposed methodology is partitioned into two components: MAC software and MAC hardware. The functions of each component are classified in order to satisfy the performance requirements and to efficiently execute its newly adopted features. Accordingly, the MAC hardware performs protocol time-related functions such as accessing a channel and transmitting frames at the exact time after the appropriate IFS including the response procedure. The MAC hardware also performs repeated processor- and bus-intensive functions such as calculating CRC and detecting and filtering duplicated MPDU. The software and hardware MAC functions are summarized in Table 1.

Table 1. Partitioned functions of IEEE 802.11ac MAC

MAC Software	MAC Hardware
- Management frame generation - MPDU generation - MSDU (de) aggregation - Tx/Rx queue management - TXOP scheduling (TXOP sharing) - (De) fragmentation - Retransmission - Multi-rate support	- Channel access function - Frame transaction - MPDU (de) aggregation - Fragmentation - Frame check sequence - Duplication detection - Protocol timer

The functional codes for MAC software are extracted from the unified code and ported to the ARM firmware. System tasks specific to the NS-2 simulator are replaced by ARM-specific functions. Further, the functional codes for MAC hardware are converted into Verilog HDL, and system tasks specific to the NS-2 simulator, such as protocol timers, are modified for the hardware implementation.

The MAC software and hardware architectures for the IEEE 802.11ac system implemented using the proposed methodology are shown in Fig. 6. It can be seen that the MacDataSrv block is solely software-operated, whereas the Transmission block is solely hardware-operated. The other blocks are operated by both MAC software and MAC hardware. The MAC software components of these blocks support the operations of the corresponding blocks of the MAC hardware components.

The hardware component of the MpduGen block contains frame buffers that store the last elements of the transmission queue in the software component of the MpduGen block. The buffer is implemented in a circular queue with a small capacity; it can support multiple frame transmission and A-MPDU transmission efficiently.

The hardware component of the ProtoCtrl block has a channel state monitor to check whether the medium is busy. The independent backoff procedure for each AC is invoked after the corresponding IFS to transmit a frame when the channel is found to be busy or after a failed

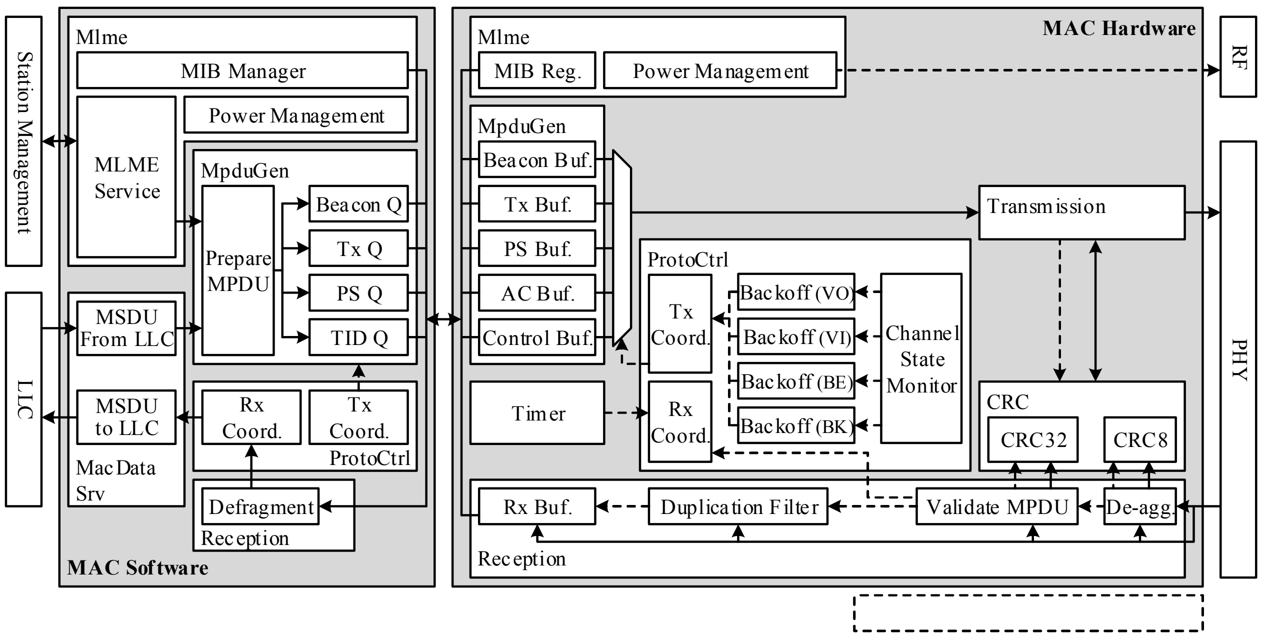


Fig. 6. Software and hardware architecture of the implemented IEEE 802.11ac MAC system (AP).

transmission. A virtual collision handler resolves internal collisions among the ACs. The hardware component of the ProtoCtrl block conducts the transmission procedure according to information about the useof RTS/CTS, fragmentation, frame aggregation, and TXOP sharing set by the software component of the ProtoCtrl block. The response procedure is determined and accomplished by the hardware; the response frame must be transmitted after a short IFS (16 μs) immediately following the reception. The software component of the ProtoCtrl block determines the frame exchange sequences during the obtained TXOP by the backoff procedure, and it calculates the value of the duration field of each MPDU. Furthermore, it performs the retransmission procedure.

The Transmission block assembles the PLCP information, MAC header, and payload, which are stored in the frame buffers and registers in the MpduGen block, into a PSDU and delivers it to PHY. To support the TXOP sharing mode with DL MU-MIMO when MAC acts as an AP, this block includes four parallel Datapump sub-blocks that can transmit MPDUs contained in the frame buffer of each AC to up to four users simultaneously. Each sub-block delivers PSDU to PHY via a dedicated data path that supports the maximum 2.34 Gbps PHY data rate for SU. Whereas all four sub-blocks are activated when MAC acts as an AP, only one sub-block is activated when MAC acts as an STA. The block

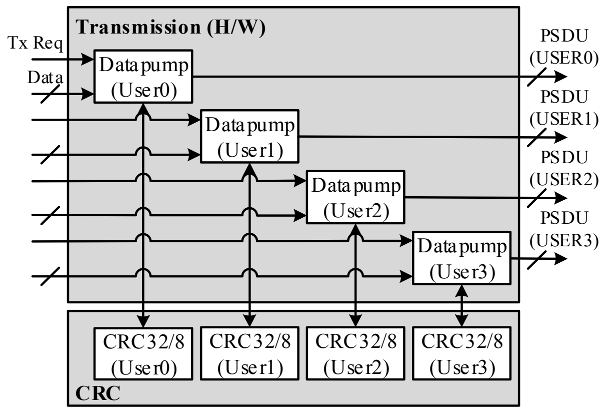


Fig. 7. Block diagram of the Transmission block (H/W).

diagram of the Transmission block is shown in Fig. 7.

The hardware component of the Reception block de-aggregates the PSDU received from PHY on receiving an A-MPDU. The de-aggregation process checks whether the first four octets constitute the MPDU delimiter as expected. If the MPDU delimiter is valid, i.e., 8-bit CRC and the delimiter signature are confirmed, then, the MPDU is extracted from the received A-MPDU. If the MPDU delimiter is not valid, it skips forward by four octets and checks for validation repeatedly. After de-aggregation, the Reception block continues processing units of MPDU. The hardware component of the Reception block includes the Duplication Filter sub-block that maintains a cache of recently received MPDUs

Table 2. FPGA synthesis reports of the proposed IEEE 802.11ac MAC hardware (AP)

Target Device	Xilinx XC5VLX330
Target Clock Frequency	40 MHz
Dual Port RAMs (RAM32M)	24
Block RAM (RAMB36)	65
Total LUTs	9237

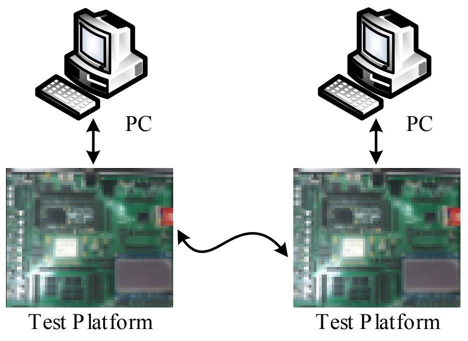
and performs comparisons with the received MPDU to discard the duplicated MPDU. In order to reduce overhead of the system bus, only the successfully received and non-duplicated MPDU is delivered to the software component of the Reception block through the reception buffer. The software component of the Reception block stores and reorders the received MSDUs. Then, the completed MSDUs are delivered to the upper layer.

The designed MAC software is compiled for the firmware of the target ARM processor, and the MAC hardware is applied to the target FPGA. Table 2 summarizes the implementation details. The required clock frequency of the MAC hardware is 320 MHz to support a maximum PHY data rate of 3-stream MIMO for SU (2.34 Gbps) at a single Datapump sub-block; however, owing to the operating speed limit of the test platform, i.e., interface between ARM processor and FPGA, the target clock frequency of FPGA is scaled down to 40 MHz, and the system is tested under this clock frequency.

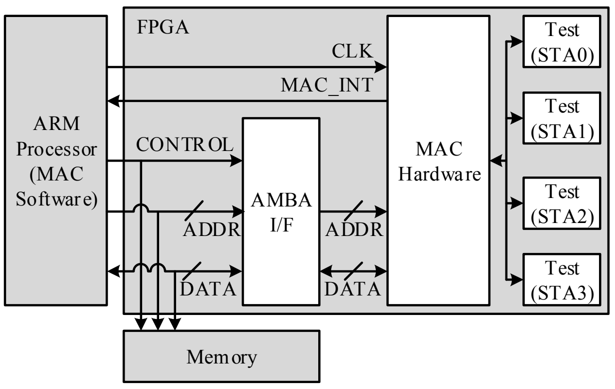
Fig. 8 shows the FPGA-based test system architecture and the block diagram of the test platform. Two test platforms are used to verify the operations of the designed IEEE 802.11ac MAC system. These platforms are wired-connected to each other for establishing a peer-to-peer connection. The operations of the designed IEEE 802.11ac MAC system are verified by generating MSDUs in the ARM firmware, which are delivered to the MAC hardware. Test patterns are generated from the unified code, and the validity of implementation is confirmed by comparing the results.

We also compare the performance evaluation results of the network simulation and the implemented MAC system. To evaluate the throughput of the implemented MAC system under a DL MU-MIMO environment with a single test platform, the MAC hardware is connected to four test modules modeled to perform only the MAC functions related to the response procedure. Only DL traffic is generated and delivered from the MAC hardware (AP) to the test modules (multiple STAs), whereas the other parameters are set to the same values as those in the network simulation. Because the clock frequency of the test platform is scaled down, the results of the test platform are multiplied by 8 for comparison with the results of the network simulation.

Table 3 presents a comparison of throughput results for the network simulation and the implemented system from the proposed methodology and throughput results from the conventional methodology, e.g. AC_BE. Because the number of antennas is set to two for each STA, the AP with DL MU-MIMO can utilize more



(a) Test system architecture



(b) Block diagram of test platform

Fig. 8. Test system architecture and block diagram.

Table 3. Throughput comparison of the proposed methodology and the conventional methodology (AC_BE)

	Proposed Design Methodology			Conventional Design Methodology	
# of Users	System	Sim.	Diff.	Sim.	Diff.
1	221.0 Mbps	223.2 Mbps	1.0 %	235.3 Mbps	6.5 %
2	218.3 Mbps	216.6 Mbps	0.8 %	233.7 Mbps	7.0 %
3	506.8 Mbps	513.0 Mbps	1.2 %	539.0 Mbps	6.4 %
4	612.5 Mbps	620.7 Mbps	1.3 %	649.9 Mbps	6.1 %

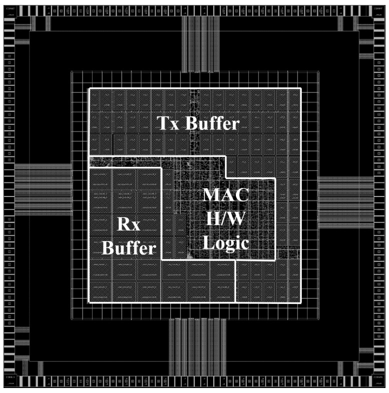


Fig. 9. Layout of implemented IEEE 802.11ac MAC hardware (AP).

Table 4. Comparison of the implementation results

Design	Paper	Ref.[12]	Ref.[13]	Ref.[14]
CMOS Tech.	65 nm	0.18 µm	0.18 µm	40 nm
Clock Frequency	320 MHz	max. 83 MHz		
Core Area	4.8 mm^2		60.8 mm^2	46 mm^2
Gate Count	0.10 M	0.12 M	3.56 M	
Memory	33.5 KB			
Function	MAC (AP)	MAC	MAC/ PHY/ Analog	MAC/ PHY/ Analog
Through-put	5.1 Gbps (MU)		150 Mbps	1.1 Gbps
Supported Standard	11e/a/n/ac, 3-stream (SU), 8-stream (MU) MIMO	11a	11e/a/g/n, 2-stream 3x3 MIMO	11e/a/b/g/ n/ac, 3-stream MIMO

antennas as the number of STAs increases, and the throughput is improved. The system evaluation results are lower than the throughput results of the conventional methodology because of the system processing delay but in good agreement with the results of the network simulation by the proposed methodology (within 1.5 % of differences). The differences of the results by the conventional methodology from the results of the system evaluation are much greater (more than 6 %). Thus, these data show that the performance of the implemented MAC system can be estimated in advance by using the proposed methodology.

The MAC hardware is also applied to 65 nm CMOS technology. Fig. 9 shows the VLSI layout of the implemented MAC hardware and the comparison with previously published work is summarized in Table 4. Only MAC function has been employed in [12], whereas PHY and analog circuits have also been included in [13, 14]. PHY is the largest component of a digital circuit. The comparison results show that the design proposed in this paper has a similar gate count but supports enhanced MAC features and AP functionality, such as DL MU-MIMO and the TXOP sharing mode; moreover, it achieves a much higher throughput up to 5.1 Gbps for MU.

V. CONCLUSION

This paper proposed a unified MAC design methodology for estimating the network performance of a MAC system prior to its implementation. In contrast to the conventional design approach, the proposed methodology employs identical code for both network performance evaluation and system implementation. The proposed methodology was adopted for implementing the IEEE 802.11ac DL MU-MIMO WLAN MAC system, which is applied to an ARM-based test platform and 65 nm CMOS technology. Its validity was confirmed via comparative evaluation of the network simulation and the implemented system.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2015R1A2A2A01004883).

REFERENCES

[1] R. Van Nee, "Breaking the Gigabit-per-second barrier with 802.11AC," *Wireless Communications, IEEE,* Vol.18, No.2, pp.4-4, 2011.

[2] Y. Jung, et al., "7.7 Gbps Encoder Design for IEEE 802.11ac QC-LDPC Codes," *Journal of Semiconductor Technology and Science,* Vol.14, No.4, pp.419-426, 2014.

[3] U. Hatnik, et al., "Using ModelSim, Matlab/ Simulink and NS for simulation of distributed systems," *Parallel Computing in Electrical Engineering, 2004. PARELEC 2004, International Conference on,* pp.114-119, Sep., 2004.

[4] W. Li, et al., "Co-simulation platforms for co-design of networked control systems: An overview," *Control Engineering Practice,* Vol.23, pp.44-56, 2014.

[5] C. Zhu, et al., "MAC enhancements for downlink multi-user MIMO transmission in next generation WLAN," *Cosumer Communications and Networking Conference (CCNC), 2012 IEEE,* pp.832-837, 2012.

[6] A. Ben Hassouna, et al., "A model for deploying an opportunistic MAC protocol in NS-2," *Sciences of Electronics, Technologies of Information and Telecommunicationns (SETIT), 2012 6th International Conference on,* pp.604-611, Mar., 2012.

[7] Y. Kim, et al., "MAC implementation for IEEE 802.11 wireless LAN," *ATM (ICATM 2001) and High Speed Intelligent Internet Symposium, 2001. Joint 4th IEEE International Conference on,* pp.191-195, 2001.

[8] Y. Cai, et al., "Novel design and implementation of IEEE 802.11 medium access control," *Communications, 2004 and the 5th International Symposium on Multi-Dimensional Mobile Communications Proceddings. The 2004 Joint Conference of the 10th Asia-Pacific Conference on,* pp.278-282, Vol.1, Aug., 2004.

[9] Y. C. Yeow, et al., "Design and implementation of 802.11 medium access control protocol using SDL," *Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication., 2005 13th IEEE International Conference on,* pp.5-5, Nov., 2005.

[10] Z. Stamenković, et al., "MAC protocol implementation in RF-MIMO WLAN," *Electronics, Circuits, and Systems, 2009. ICECS 2009. 16th IEEE International Conference on,* pp.303-306, Dec., 2009.

[11] C. Chung, et al., "Implementation of IEEE 802.11n MAC using design methodology," *Journal of Korea Information and Communication Society,* Vol.34, No.4, pp.360-367, 2009.

[12] Z. Yang, et al., "Design and verification of high-throughput IEEE 802.11 MAC-layer hardware IP with FPGA platform," *Journal of the Chinese Institute of Engineers,* Vol.33, No.4, pp.551-562, 2010.

[13] P. Petrus, et al., "An integrated draft 802.11n compliant MIMO baseband and MAC processor," *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2007 IEEE International,* pp.266-602, 2007.

[14] M. He, et al., "A 40nm dual-band 3-stream 802.11a/b/g/n/ac MIMO WLAN SoC with 1.1Gb/s over-the-air throughput," *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International,* pp.350-351, 2014.

Chulho Chung received the B.S., M.S., and Ph.D. degrees in electrical and electronic engineering from the Yonsei University, Seoul, Korea, in 2003, 2009, and 2016, respectively. He is currently a senior engineer in Device Solution Division, Samsung Electronics Co. Ltd., Korea. His research interests include the algorithm and implementation of MAC layer for the wireless multi-media communication system such as WLAN and WPAN.

Yunho Jung received the B.S., M.S., and Ph.D. degrees in department of electrical and electronic engineering from Yonsei University, Seoul, Korea, in 1998, 2000, and 2005, respectively. From 2005 to 2007, he was a senior engineer in the Wireless Device Solution Team, Communication Research Center, Telecommunication Network Division, Samsung Electronics Co. Ltd., Suwon, Korea. From 2007 to 2008, he was a research professor at Institute of TMS Information Technology, Yonsei University, Seoul, Korea. He is currently an associate professor in the School of Electronics and Information Engineering, Korea Aerospace University, Goyang-si, Korea. His research interests include the signal processing algorithm and SoC/VLSI implementation for the wireless communication systems and image processing systems.

Jaeseok Kim received the B.S. degree in electronic engineering from Yonsei University, Seoul, Korea in 1977, M.S. degree in electrical and electronic engineering from KAIST, Daejeon, Korea in 1979, and Ph. D degree in electronic engineering from RPI, NY, USA in 1988. From 1988 to 1993, he was a Member of Technical Staff at the AT&T Bell Lab., USA. He was Director of VLSI Architecture Design Lab. of ETRI from 1993 to 1996. He is currently a professor of the electrical and electronic engineering department Yonsei University, Seoul, Korea, and also serves as director of IT SoC Research Center. His current research interests include communication (WLAN, UWB, 4G/5G) SoC design, high performance Multimedia IP (H.264, HEVC, ISP) design, and SoC Platform architecture.