GPLM: An 802.11ac-Capable Low-MAC Architecture for FPGA-based SDR Systems

Boyan Ding, Jun Liu, Haoyang Wu and Tao Wang
Peking University, Beijing, China
Email: {dboyan,juneliu,wuhaoyang,wangtao}@pku.edu.cn

***Abstract*—802.11 is a widely-used wireless communication standard today and is still under constant evolution. Two major enhancements of the standard, 802.11n and 802.11ac, boost the performance and quality of service, with the former getting support in nearly all commercial devices today and the latter gaining prevalence. However, there is currently no software-defined radio (SDR) system that is capable to support the whole 802.11ac protocol stack, especially the MAC layer, in real-time, limiting research and testing on these latest innovations. This paper presents GPLM, a Low-MAC architectural design for FPGA-based SDR systems that supports 802.11ac. We identify challenges imposed by new features in 802.11ac MAC layer, and make careful architectural choices to ensure both standard compliance and flexibility. The design and implementation of critical modules and the employment of software hardware co-design are detailed in this paper. Paired up with an existing work on the PHY layer implementation of 802.11ac, the implementation of GPLM is validated from various aspects.**

***Index Terms*—802.11, SDR, MAC, FPGA**

I. Introduction

Software-defined radio (SDR) provides users with flexible and configurable developing platform for wireless communication, upon which protocols and algorithms, either existing or new, can be prototyped and tested. A number of SDR platforms [1]–[5] are available for different needs, and many of them support IEEE 802.11 [6], one of the most commonly-used wireless communication standards today.

The increasing need for wireless communication speed and quality elicits several extensions to the original standard, with the most notable ones being 802.11n and 802.11ac. The former extension, published in 2009, greatly boosted the performance compared with original 802.11a/b/g, and is supported in nearly every commercial device today. While the latter 2013 extension improves even further, also gradually gaining prevalence. However, the development of SDR platforms doesn't quite catch up with the pace of the continuously evolving 802.11 standard. Although there have been works [7], [8] to support 802.11n/ac's physical layer in real time, works concerned with 802.11n/ac MAC, such as [9], [10], are still limited to software simulation. This not only limits the research in MAC itself, but also cripples the testing of PHY layer since a full-feature and real-time MAC is indispensable to providing a real workload to the physical layer.

The difficulty for real-time MAC layer in SDR platforms to catch up with the evolution of 802.11n/ac mainly comes from the following aspects. First, real-time implementation of MAC layer requires hardware acceleration, which has a harder development process than pure software. More importantly, the nature of MAC layer processing causes difficulty in hardware MAC development. The MAC layer contains complex control flow as dipicted in Fig. 1. More often than not, one single functional change in the MAC layer may affect the design of multiple modules. Finally, 802.11n/ac introduces changes that greatly affect framing procedure, invalidating many previous designs that are feasible to 802.11a/g.

In this paper we propose GPLM, an architectural design of real-time MAC layer for SDR systems that supports new features in 802.11ac. The design is based on analysis on new features of 802.11ac MAC protocol. Besides supporting 802.11ac, GPLM also takes flexibilty and easiness of programming into account. FPGA is chosen as the implementation platform for its wide use in SDR, and we have integrated our implementation with physical layer of existing work [8].

Our contributions in this paper can be summerized as:

- We analyze 802.11n/ac MAC protocols and identify the challenges they impose on MAC SDR design;
- We propose GPLM, an FPGA based Low-MAC architecture that can support 802.11ac MAC in real time. In our knowledge, this is the first SDR MAC design to really support the mandatory features in 802.11n/ac;
- We implement GPLM and integrate it with an existing 802.11ac PHY implementation [8], validating the design and testing the performance.

The remainder of this paper is organized as follows: Section II provides a detailed analysis of the new features in 802.11ac MAC and the challenge they impose. The design and implementation are presented in Section III and IV respectively. Section V evaluates GPLM from various aspects. Related works are discussed in Section VI. Finally, Section VII concludes this paper.

II. An analysis of 802.11ac MAC protocol

A. MAC extensions introduced in 802.11ac

We mainly focus on features that are critical to the performance of 802.11ac. 802.11n is also taken into account for backward compatibility. The main extensions introduced in 802.11n/ac include:

- *QoS extension.* Although not strictly part of either 802.11n/ac, it provides some fundamental basis for them, including prioritized categories (AC) for medium access and block acknowledgment (Block Ack).

978-1-5386-7646-2/19/$31.00 ©2019 IEEE

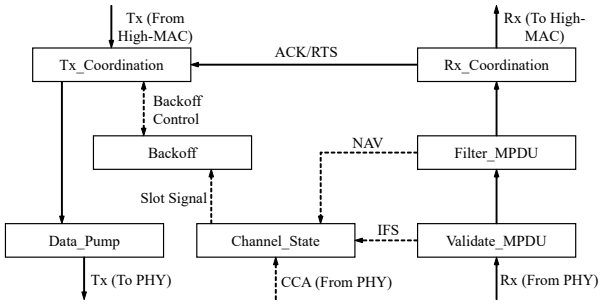

Fig. 1. Low-MAC functional diagram

- *HT extension.* Two frame aggregation schemes, A-MSDU and A-MPDU, are introduced in 802.11n. They enable longer frames in physical layer to make better use of its higher throughput. A-MSDU groups frame early in MAC processing, while A-MPDU aggregates frames before they enter physical layer. The latter is more frequently used and requires Block Ack to function.
- *VHT extension.* It provides enhancements based on HT extensions to adapt to longer aggregated frames and MU-MIMO in 802.11ac. The format of A-MPDU delimiter is also changed.

New features included in these extensions have to be supported in 802.11ac-capable SDR MAC layers. While some of them only need little change in hardware, others require more effort, posing challenges on MAC layer design.

B. MAC model of SDR systems

Before listing challenges of 802.11n/ac features on SDR MAC, we first discuss the MAC model of SDR systems and how the above features map into the model.

Most MAC layer implementations of SDR system follow the model of commercial WLAN NIC and can be devided into two parts [11]: High-MAC (Upper MAC) that handles high-level data and management (MLME) services and Low-MAC (Lower MAC) that is responsible for timing-critical protocol control and low-level transmission and rececption. Fig. 2 depicts this division.

Of the two parts, the Low-MAC is more important and challenging in SDR architecture. This is because Low-MAC needs hardware acceleration and requires more design and implementation effort. While its counterpart, the High-MAC is often implemented in pure software, since it is less demanding in timing and loosely-coupled with Low-MAC.

In typical SDR MAC implementations with 802.11a/g support, functions that should be implemented in Low-MAC include: 1) Timing functions: clear channel assessment (CCA) and backoff; 2) Control frame (ACK and RTS/CTS) generation; 3) Frame transmission and reception, including frame check sequence (CRC32) appending and checking. Besides above, frame retransmission is often implemented in Low-MAC to improve performance.

Among the new features of 802.11ac, some features can be easily implemented in High-MAC. While others impacts greatly on the Low-MAC architecture. Take the two frame

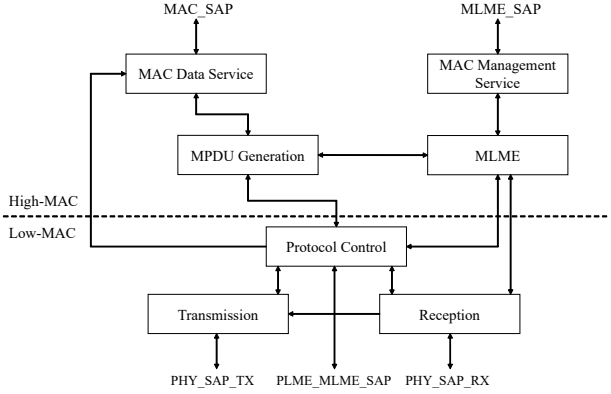

Fig. 2. The model of SDR MAC

aggregation schemes: A-MSDU and A-MPDU (Fig. 3) as an example. In the former form, aggregated frames are encapsulated in a single MAC header and is considered a single frame in 802.11 MAC processing. So aggregation can take place in High-MAC, and is often implemented this way. A-MPDU, on the other hand, is quite the opposite. The frames (MPDUs) are aggregated *after* all processing of traditional 802.11 MAC, and are treated as individual frames in the receiver's MAC layer. Thus, paired up with Block ACK, which reports the status of individual A-MPDU subframes, A-MPDU mandates a hardware Low-MAC implementation.

C. Challenges of 802.11n/ac features on SDR MAC

As mentioned before, the challenges of 802.11n/ac mainly come from the introduction of A-MPDU and Block Ack, and can be summerized as follows:

- *Complexity of A-MPDU.* A-MPDU makes data-processing of 802.11 MAC considerably more complex. In addition to appending CRC32 checksum, frames are grouped and separated with 4-octet A-MPDU delimiters, involving quite a few details such as zero-padding between frames. Greater throughput of 802.11ac also sets a higher performance requirement for real-time processing of A-MPDU. Without these in mind, previous architectures can hardly meet either the functional or the performance requirement.
- *Data manipulation in Block Ack and other features.* In the original 802.11a/g MAC, frames generated in Low-MAC (ACK, RTS/CTS) have mostly fixed format except MAC address field, so most previous solutions use fixed-function modules to generate these frames. In 802.11ac, more kinds of frames with a greater degree of variance are generated from Low-MAC, with Block ACK being a notable example. Thus, the old way is no longer viable in 802.11ac and a more generic means of frame generation should be devised in the new architecture.
- *Retaining programmability.* The purpose of SDR is to facilitate prototyping and testing, so programmability is always a primary goal. There are two aspects of programmability here: less developing effort and easier customization. However, previous functional and performance requirements often tend to make design complex

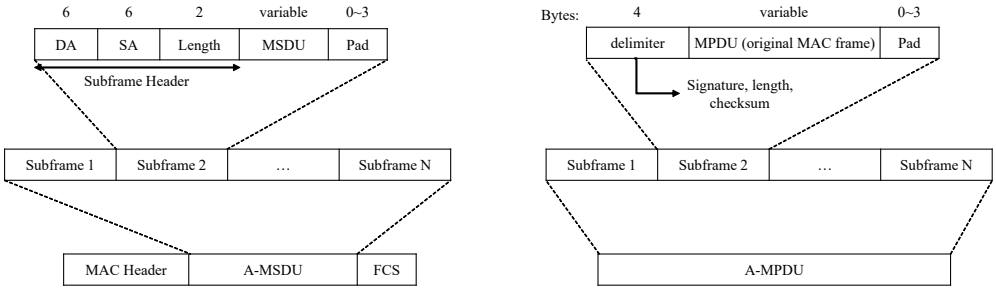

Fig. 3. Comparison of aggregation schemes: A-MSDU and A-MPDU

and fixed-function, hurting programmabilty. So retaining reasonable programmability under such requirements is not a trivial task and requires careful design.

Any Low-MAC implementation for SDR systems that supports 802.11ac should settle these challenges in its design and implementation. Our approaches to these challenges will be discussed in the following two sections.

III. Design of GPLM

In order to properly address the challenges stated above, several design choices are carefully made, constituting the highlight of GPLM design. The complexity of A-MPDU is tackled with descriptor FIFO design and HLS written processing module; a versatile memory architecture ensures the feasibility of Block ACK and other data manipulation; a high degree of programmabilty is achieved with SW/HW co-design techniques including the employment of processor-accelerator architecture and HLS. Details of the design and implementation are discussed in this and the following section.

A. Overall Architecture

The Low-MAC of GPLM consists of a microprocessor (MCU) and several custom accelerators implemented with programmable logic as peripherials. The microprocessor is responsible for general control flow operations and configurations, running real-time operating system (RTOS). While accelerators implement individual logics that require high throughput (framing/deframing), low latency (carrier sensing) or implement specialized functions that are not feasible for the microprocessor. The architecture is shown in Fig. 4.

The approach GPLM adopts is an example of HW/SW co-design, effectively combining the versatility of software and the efficiency of programmable logic. Users only need to tend to the hardware design flow when customizing those specialized logic. While at other times, only adaptation in software is necessary.

This approach also enables modular design. Logics that have to be implemented as hardware are divided into independent modules. Each hardware module is wrapped with standard interfacing logic, with most control interfaces exposed as memory-mapped registers on AXI4 bus [12] and streaming data intefaces implemented with AXI-Stream or FIFO, all of which are standard and widely-used interfaces. Besides Verilog, we also support high-level synthesis (HLS) in the development of modules (section IV-B). These efforts simplify

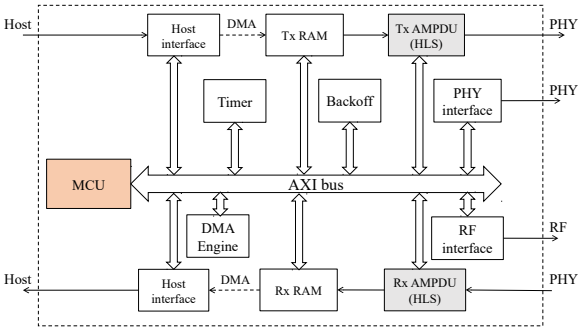

Fig. 4. GPLM's Low-MAC architecture

hardware development for Low-MAC by providing a standard framework for accelerator logic and interfaces. Users can develop or debug any accelerator independently with ease.

According to their functions, peripherial modules in Fig. 4 can be divided into the following categories: interface modules that provide connection with other components (i.e., host PC, PHY, RF) to form a complete SDR system; control modules that are responsible for Low-MAC control logics; memory subsystem that holds the frame data in Low-MAC; and A-MPDU processing modules that handle aggregation. Among these, the *memory subsystem* and *A-MPDU processing modules* stand out in GPLM because they play important roles in solving the challenges discussed in Section II and require novel design. The design of memory subsystem modules are detailed in the following subsections.

B. Memory Subsystem

GPLM's memory subsystem consists of two parts that are similar in structure, one for transmission, and the other for reception. Fig. 5 depicts the architecture of either part of the memory subsystem. The following discussion in this section applies to both Tx and Rx part unless explicitly specified.

Either part of the memory subsystem contains a dual-port random access memory (RAM) used to hold frame data. The main reason to use RAM instead of other memory primitives, such as FIFO, is the consideration of programmability. RAM provides easier and more flexible ways to manage and manipulate frame data, which is crucial to generation of Block ACK and other custom frames.

The flexibilty is further ensured by arrangements of the ports. One port of the RAM is connected to the AXI bus with an adaptor IP, which means both host interface and microprocessor can easily access frame data through this port.

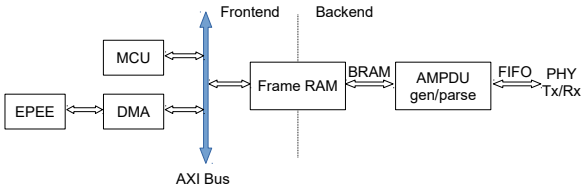

Fig. 5. Memory subsystem of GPLM's Low-MAC

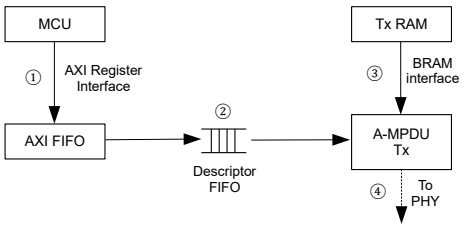

Fig. 6. Tx A-MPDU Processing

Normal frame data are transferred between RAM and host interface module (a wrapper around PCIe, USB or Ethernet IP that provides host access) through a DMA engine, which guarantees high throughput with little microprocessor interaction. Meanwhile, the microprocessor can access frame data in the same manner as accessing normal memory. Thus generating Block ACK or other custom frames are as easy as writing to main memory. The other port is provided exclusively for A-MPDU generation or parsing modules in the Tx or Rx side.

An extra bonus of this memory subsystem design is the extensive usage of common IP provided by FPGA design tools. Except host interface and A-MPDU processing modules, all other components in Fig. 5 can be found in IP repository in tools such as Vivado [13]. This substantially reduces developing effort of GPLM.

C. A-MPDU Processing

The complexity of A-MPDU in 802.11ac comes from the fact that frames are aggregated *after* all other MAC layer processing in the transmitter and should be deaggregated and processed individually in the receiver. Thus both process must happen in real time in Low-MAC. Moreover, various details and backward compatibility with legacy 802.11 or 802.11n have to be taken into account.

Our solution to this complexity is to define an *A-MPDU descriptor queue* as the control interface between the microprocessor and A-MPDU processing module. A legacy or A-MPDU frame can be represented as one or more *A-MPDU descriptor(s)* where one descriptor represents a legacy frame or a A-MPDU subframe. The A-MPDU descriptor queue can be implemented with the AXI-Stream FIFO IP.

In the transmitter, after deciding what MPDUs are aggregated, the microprocessor writes a set of A-MPDU descriptor into the descriptor queue. The A-MPDU generation module reads descriptors from the queue and fetch frame data from Tx RAM to aggregate frames. The aggregated frame is then sent to physical layer for transmission. This process is shown in Fig. 6. The A-MPDU deaggregation module in the receiver, on the other hand, reads received frame data from physical layer and deaggregates A-MPDU into single MPDUs, checking the validity of each subframe in the process. The content of MPDUs is written into the Rx data RAM while descriptors is written in the descriptor queue. The microprocessor further processes the received frames accroding to the descriptors.

Listing 1
SIMPLIFIED HLS SOURCE OF TX A-MPDU MODULE

```
void tx_ampdu_gen(
    hls::stream<txdescdata_t> &desc_din,
    data_t tx_ram[TXBUFFER_DEPTH],
    hls::stream<txfifo_t> &dout)
{
    txdescdata_t desc_data;
    txdesc_t desc;
    addr_t addr;
    do {
        desc_din >> desc_data;
        parse_tx_desc(desc_data, desc);
        addr = desc.addr;
        if (desc.ampdu)
            dout << mpdu_delim(desc);
        for (int i = 0; i + 3 < desc.flen; i += 4){
            dout << tx_ram[addr++];
            // ... CRC calculation here
        }
        output_tail_crc();
        for (int i = 0; i < desc.padcnt; i++)
            dout << empty_mpdu_delim();
    } while (!desc.last_mpdu);
}
```

IV. IMPLEMENTATION OF GPLM

A. Platform and Dependencies

We implement GPLM on Xilinx 7-series FPGA. Currently VC707 and ZC706 evaluation boards are supported. The ARM processor core is used on ZC707 (Zynq) platform, and we use Microblaze IP as the microprocessor on VC707.

Besides utilizing IPs provided by Vivado design tool mentioned in Section III, we also use an updated version of EPEE [14] in the host communication interface module, providing PCIe or USB 3.0 interface with the host PC.

Physical layer and RF frontend are also needed to form a complete SDR platform. We integrate the works from [8] and [4] into our system for 802.11ac and 802.11a/g support respectively. They are used in the evaluation in Section V.

B. HLS-based A-MPDU Processing

It is previously mentioned that A-MPDU processing in 802.11ac is complicated and constitutes one of the challenges in Low-MAC design and implementation. While the design of A-MPDU processing modules is discussed in Section III, its implementation is still not easy. In implementing A-MPDU processing modules, we take advantage of High-Level Synthesis (HLS) to convert algorithmic description of the modules in C++ into HDL design. We found that HLS can meet our requirements in resource and efficiency and greatly reduce the programming effort in A-MPDU processing modules compared with traditional HDL such as Verilog.

TABLE I
RESOURCE UTILIZATION OF 802.11AC SYSTEM

Versions	LoC	Resource		Throughput (100MHz clock)
		LUT	FF	
Verilog 802.11a/g	377	340	193	0.8Gbps
HLS 802.11a/g	141	462	325	0.8Gbps
HLS 802.11ac	238	669	532	1.6Gbps

1) Difference between HLS and HDL: When designing hardware modules with HDL, developers should design the timing, state machine, interface logic besides the intended functionality of the module. The source code is often long and complex with all of the above mixed together. In HLS, however, the source code contains only algorithmic description of the module. The HLS compiler, on the other hand, takes the responsibility of resolving other details. Although HLS is not suitable for all hardware design problems and might lead to suboptimal design, it is an ideal choice for the case of A-MPDU processing module.

2) A-MPDU Processing in HLS: Take the Tx A-MPDU generation module as an example. A simplified version of the source code for this module is shown in Listing 1.

The arguments of `tx_ampdu_gen` correspond with the interfaces of the A-MPDU generation module and the function body describes its logic. This plain C++ code is enough to produce logically correct A-MPDU processing module, however, some tuning is needed to ensure ideal performance. For example, by applying "pipeline" directive on the CRC calculation loop, the performance can increase by nearly 33%.

3) Analysis on HLS: To evaluate the effect HLS brings to the development of A-MPDU processing module, we made a comparison among three implementations using either traditional Verilog or HLS, shown in Table I.

The third row represents the HLS implementation of A-MPDU processing module in the transmitter. Due to the lack of Verilog implementation of A-MPDU processing module, we choose its corresponding module in 802.11a/g—the checksum appending module from [4], which is much simpler in functionality. We also reimplement an equivalent of the latter module in HLS to ease comparison.

As is indicated in the table, two HLS implementations need considerably less lines of code, even for more functionally complex 802.11ac modules. The difference between the two HLS modules lies in more complicated A-MPDU processing logic and more efficient checksum calculation, which doubles the throughput to fit 802.11ac better. The downside of HLS is that it takes more FPGA resources (LUT and FF in Table I). But considering the convenience it brings to development, the moderate increase in resource consumption is acceptable.

C. Microprocessor Control

The control program that is executed on GPLM's microprocessor core (Microblaze or ARM) is responsible for the general control flow and the configuration of Low-MAC. The control of Low-MAC consists of several individual execution flows (Tx, Rx, Backoff, etc.) and each of them has rigid requirement on timing.

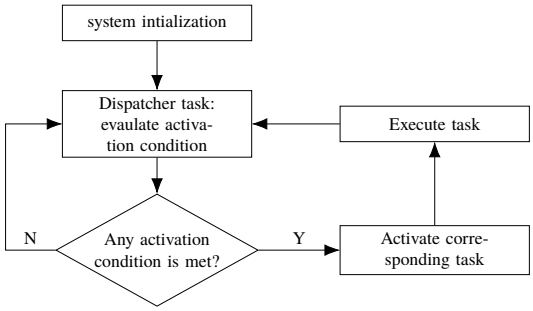

Fig. 7. Task dispatch on MCU

TABLE II
RESOURCE UTILIZATION OF GPLM ON VC707

Resource	GPLM		Full 802.11ac	
	Util	%	Util	%
LUT	46742	15.40	141703	46.67
LUTRAM	3408	2.61	4425	3.38
FF	45910	7.56	164818	27.14
BRAM	158	15.34	322	31.26
DSP	4	0.14	264	9.43

To meet the requirement in timing while maintaining a clear program structure to facilitate customization. GPLM utilizes FreeRTOS [15], an open-source real-time operating system. Each control flow is divided into one or more FreeRTOS tasks which implement a part of the control flow without sleep or polling with dead loop. Each task also has an "activation condition" to judge whether the task is ready to run. A low-priority dispatcher task evaluates the tasks' activation condition and activate the corresponding task when condition is met. This procedure is depicted in Fig. 7.

V. EVALUATION

A. Experiment Setup

We use Xilinx VC707 evaluation board as our FPGA platform on which GPLM and corresponding physical layer are built. The design is built with Vivado Design Suite 2016.4, including the high-level synthesis (HLS) compiler and the software development environment (SDK). Ubuntu 16.04 is used on both development and experiment PCs. The host PC and FPGA are connected with USB 3.0 connection provided by EPEE communication library. We use ADRV9371 as RF front-end, which, coupled with physical layer of Tick, provides 80MHz 802.11ac PHY implementation.

B. Resource Utilization

Table II lists the resource consumption of the standalone GPLM project and a complete 802.11ac system with physical layer from Tick and RF. The names in the first column, including LUT (look up table), FF (flip-flop), etc., stand for various kinds of logical resource on FPGA.

According to the tables, GPLM has a modest consumption of FPGA resources, only occupying a small fraction compared with the 802.11ac physical layer in Tick. This ensures enough space for physical layer implementations and/or other applications on FPGA.

TABLE III
ACK LATENCY OF GPLM SYSTEM

	GPLM (ACK)	GPLM (Block ACK)	Original Tick (ACK)
Latency (μs)	20.96	21.45	19.70

C. Functional and Performance Evaluation

1) Functional Verification: Multiple levels of verification were made on the system design. Core functional modules, such as A-MPDU processing module, went through functional simulation to ensure correct behavior. Then a Low-MAC system project were built with a loopback physical layer for overall functional testing of the whole MAC layer. Finally, we integrated the Tick's 802.11ac physical layer into the GPLM project and verify the design with live communication in the air. The final version of the GPLM project passed functional verifications and can support both the legacy 802.11a/g and the newer 802.11n/ac.

2) Latency: We use the interval between a data frame and its corresponding ACK by the GPLM+Tick to evaluate the processing latency of GPLM platform. The time interval is measured from the RF trace data of a third Tick system. Table III lists the ACK latency of the new 802.11ac system. The latency of the original Tick [8] is also listed as a reference.

According to the table, the latency of legacy ACK of GPLM increased by 1.26μs compared to the original MAC. But GPLM has its own advantages: 1) It supports A-MPDU and Block ACK in 802.11n/ac, which the original system cannot. 2) Compared to the original system, more components in Low-MAC processing can be done software, such as ACK frame format, making the system configurable.

We also measured the ACK latency of commercial WiFi NICs. Traces are collected with Intel's IWL7265 card on Linux in monitor mode and analysed with Wireshark. The average legacy ACK latency is 18.8μs, while the average Block ACK latency is 23.3μs. Thus the ACK latency is generally on the same level of commercial products.

3) Throughput: We tested the communication throughput of GPLM with Tick's 802.11ac physical layer under various conditions. All tests uses the MCS6 (64-QAM, 3/4 coding rate) coding scheme because it achieves the best balance between coding effectiveness and robustness (The limitation of PHY decoding of Tick [8] causes performance degradation in higher MCSs). The throughput data are all measured in Mbps.

Table IV and V list the measured throughputs of legacy frame and A-MPDU frame transmission respectively. A comparison with Intel IWL7265 NIC is listed in Table VI.

As can be seen in Table IV, in legacy transmission, the throughput increases significantly as the data frame becomes longer. Nevertheless, the effiency of the longest frame is still a small fraction of the 263.3Mbps theoratical value. This is caused by the overhead of 802.11ac communication.

The protocol overhead includes preambles, inter-frame spaces (IFS), ACK frames, etc. With the advance of 802.11 protocols, the time used for data transmission drastically decreases. For a normal frame transmission, the overhead can be hundreds of microseconds. Meanwhile, in 802.11ac with 80MHz bandwidth, the data duration of a 1500 byte (the common ethernet length) frame in MCS6 modulation is only 48 microseconds. So only a small fraction of air time is used transmit the actual data. Thus, aggregation, which can increase the amount of data in a single frame by merging several frame into one, becomes important in 802.11n/ac to unleash the full potential of high-speed physical layer.

TABLE IV
THROUGHPUT OF LEGACY FRAME TRANSMISSION

Frame length (bytes)	1000	1500	2000	3000	4000
Throughput (Mbps)	18.98	24.21	34.68	48.58	58.30
Theoratical %	7.2	9.2	13.2	18.4	22.1

TABLE V
THROUGHPUT OF A-MPDU FRAME TRANSMISSION

MPDU length \ aggr. size	x3	x4	x5
1000	35.93	39.32	42.81
2000	59.68	74.78	84.11
4000	91.26	121.93	120.60

TABLE VI
THROUGHPUT OF LEGACY FRAME TRANSMISSION

	Real (Mbps)	**Theoretical (Mbps)**	**Efficiency**
GPLM legacy	58.3	263.3	22.1%
GPLM A-MPDU	121.93	263.3	46.3%
Intel NIC	222.31	360.0	61.8%

Whereas in Table V, although at the same frame length, the A-MPDU transmission performance isn't as great as that of legacy transmission caused by the overhead of fragmented data exchange between host PC and FPGA (solving this will be our future work), we still get impressive performance boost when frame and aggregation length get larger. The last 4000x5 case is an exception, in which the frame is too long and the variation of channel lower the chance of successful decoding, offseting the improvements in longer frames.

As for the data in Table VI. The IWL7265 was connected to a 40MHz band 2x2 MIMO 802.11ac WiFi network (so the throughput would be nearer to our situation). And the throughput was obtained from iperf3 UDP test. The Intel NIC gets nearly double throughput than GPLM+Tick. There are mainly two reasons contributing to the phenomenon. The first reason is that the decoding performance of Intel NIC is much better than Tick's physical layer, and it can successfully use MCS9 modulation scheme, with a much higher theoratical throughput compared with MCS6 used in our system. And this is outside the scope of MAC layer. While GPLM's MAC layer is designed for SDR programming so it is normal to be less efficient than commercial NIC. Taking these two reasons into account, the performance of GPLM+Tick can be considered reasonable and there is still space for improvement.

VI. RELATED WORKS

SDR platforms. Recent years have witnessed the emerge of several SDR platforms with different characteristics and use-

cases. They can be divided into two types: pure software and hardware-accelerated.

Pure software implementations focus on programmability and are mostly used in simulation or low-speed testing due to limitation in throughput and latency, GNU Radio [1] is a typical example. Sora [3] uses SIMD instructions to accelerate 802.11a/g's PHY layer computation. However, the latency between CPU and device still makes real time MAC impossible.

Other works, including [2], [4], [5], [8], take advantage of hardware accelerators, mostly FPGA, and have the potential to realize real-time communication. However, none of them fully supports 802.11n/ac. WARP [2] and Tick [8] claim to have support for 802.11n and 802.11ac respectively. But their focuses are mainly in the physical layer and both lack the features, even mandated ones, of 802.11n/ac.

Independent works on SDR MAC. There also exist works that focus on SDR MAC implementation. [11] leverages the reverse-engineered MAC processor of Broadcom NIC and explores its programmability. It is able to support TDMA instead of the original CSMA/CA in 802.11. [16] is a SystemC implementation of MAC layer that highlights low latency and low jitter for industrial use. THUMP [17] uses compiler technique to convert a custom meta-language to MAC design on WARP [2]. These works are all based on 802.11a/g model and none of them supports the new features of 802.11n/ac.

Studies on 802.11n/ac MAC. Due to the lack of real time SDR support of the 802.11n/ac MAC, studies on these protocols are conducted on commercial product or offline simulation. [9] builds a software simulation model that supports both frame aggregation (A-MPDU and A-MSDU) and block acknowledgment as mandated in 802.11n. Studies on 802.11n/ac MAC mainly focus on the performance impact introduced by new extensions. [18] and [10] studies the influence of frame aggregation in 802.11n and 802.11ac respectively. However, due to their simulation nature, how their results match real-world situations can't be guaranteed.

VII. Conclusion

Wireless protocols such as 802.11 are constantly evolving, with the recent 802.11ac gaining prevalance. However, the development of SDR platforms doesn't quite catch up, especially in the case real-time MAC layer implementation.

The work presented in this paper, GPLM, offers a architectural solution for 802.11ac MAC layer support for FPGA-based SDR systems. In designing GPLM, the challenges of 802.11ac MAC support are identified, namely *A-MPDU support*, *flexible data manipulation* and *programmability*. To solve these challenges, GPLM proposes various architectural designs, including the microprocessor and accelerator architecture, the organization of memory subsystem and descriptor queue as control interface to A-MPDU processing modules. HLS and RTOS-based microprocessor control framework are also employed to ease programming in GPLM's implementation. We make resource and functional evaluations on GPLM, integrating it with existing SDR physical layers that supports 802.11a/g or 802.11ac.

Acknowledgment

The work is funded by National Key Research and Development Plan of China (2017YFB0801702) and key research project of National Natural Science Foundation (No. 61531004).

References

[1] E. Blossom, "GNU radio: tools for exploring the radio frequency spectrum," *Linux journal*, vol. 2004, no. 122, p. 4, 2004.

[2] A. Khattab, J. Camp, C. Hunter, P. Murphy, A. Sabharwal, and E. W. Knightly, "WARP: a flexible platform for clean-slate wireless medium access protocol design," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 12, no. 1, pp. 56–58, 2008.

[3] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, and G. M. Voelker, "Sora: high-performance software radio using general-purpose multi-core processors," *Communications of the ACM*, vol. 54, no. 1, pp. 99–107, 2011.

[4] T. Wang, G. Sun, J. Chen, J. Gong, H. Wu, X. Li, S. Lu, and J. Cong, "GRT: a reconfigurable SDR platform with high performance and usability," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 4, pp. 51–56, 2014.

[5] J. Zhang, X. Zhang, P. Kulkarni, and P. Ramanathan, "OpenMili: a 60 GHz software radio platform with a reconfigurable phased-array antenna," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 2016, pp. 162–175.

[6] IEEE Standards Association and others, "802.11-2012-IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std*, vol. 802, 2012.

[7] J. Declerck, P. Raghavan, F. Naessens, T. Vander Aa, L. Hollevoet, A. Dejonghe, and L. Van der Perre, "SDR platform for 802.11n and 3-GPP LTE," in *2010 International Conference on Embedded Computer Systems (SAMOS)*. IEEE, 2010, pp. 318–323.

[8] H. Wu, T. Wang, Z. Yuan, C. Peng, Z. Li, Z. Tan, B. Ding, X. Li, Y. Li, J. Liu *et al.*, "The Tick programmable low-latency SDR system," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. ACM, 2017, pp. 101–113.

[9] H. Loeb and C. Sauer, "A modular reference application for IEEE 802.11n wireless LAN MACs," in *IEEE International Conference on Communications*. IEEE, 2009, pp. 1–5.

[10] B. Bellalta, J. Barcelo, D. Staehle, A. Vinel, and M. Oliver, "On the performance of packet aggregation in IEEE 802.11ac MU-MIMO WLANs," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1588–1591, 2012.

[11] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, and F. Gringoli, "Wireless mac processors: Programming mac protocols on commodity hardware," in *Proceedings of IEEE INFOCOM*. IEEE, 2012, pp. 1269–1277.

[12] "AMBA Specifications – ARM," https://www.arm.com/products/silicon-ip-system/embedded-system-design/amba-specifications.

[13] "Vivado Design Suite," https://www.xilinx.com/products/design-tools/vivado.html.

[14] J. Gong, T. Wang, J. Chen, H. Wu, F. Ye, S. Lu, and J. Cong, "An efficient and flexible host-FPGA PCIe communication library," in *24th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2014, pp. 1–6.

[15] R. Barry, "FreeRTOS-a free RTOS for small embedded real time systems," 2006.

[16] K. Tittelbach-Helmrich and Z. Stamenkovic, "Hardware implementation of a medium access control layer for industrial wireless LAN," in *IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, 2016, pp. 1–6.

[17] X. Zhang, J. Ansari, G. Yang, and P. Mähönen, "Trump: Efficient and flexible realization of medium access control protocols for wireless networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2614–2626, 2016.

[18] D. Skordoulis, Q. Ni, H.-H. Chen, A. P. Stephens, C. Liu, and A. Jamalipour, "IEEE 802.11n MAC frame aggregation mechanisms for next-generation high-throughput WLANs," *IEEE Wireless Communications*, vol. 15, no. 1, 2008.