# Cross-Layer Design for IEEE 802.16-2005 System Using Platform-Based Methodologies

Li-chuan Tseng, Kuan-yin Chen, and ChingYao Huang

Institute of Electronics, National Chiao-Tung University, Hsinchu, Taiwan {lctseng.ee90,cgi0911.ee94g}@nctu.edu.tw
http://wintech.ee.nctu.edu.tw

Abstract. In this article, we present a cross-layer design scheme for 802.16-2005 system. The cross-layer design includes both MAC and PHY and is implemented through platform-based methods. In this design, we discuss how the cross-layer design affects radio resource management efficiency and system timing by examining system throughput. As for the platform-based design, we present a method that enables designers to estimate the allocation of hardware costs at an early design stage.

**Key words:** IEEE 802.16, Platform-Based Design, Cross-layer, HW/SW Co-Design, Performance Estimation

# 1 Introduction

As the demand of broadband wireless access (BWA) grows rapidly, people need new standards to provide high speed wireless transmission services. The OFDMA-based IEEE 802.16-2005 Standard (Mobile WiMAX) has become the most important candidates among all cellular technologies supporting BWA services. However, the system design becomes more challenging with higher system performance requirement. Designers are seeking for new design approaches to overcome strict constraints such as a standby time, hardware size, and etc.

With suitable architecture and optimal performance, our research objective is to discover new design approaches for the ratified Mobile WiMAX Standard [1] [2]. An example of system architecture was discussed in [3]. Algorithm design and performance simulation issues were discussed in [4].

An important issue for communication system implementation is the layer structure. Traditionally, a layered model such as OSI 7-layer structure separates the whole system into well defined layers. This is to facilitate system development, and provide compatibility between products from different vendors. In traditional layered designs, direct communications between layers are forbidden, only procedure calls and responses are allowed [5].

However, for modern high-speed BWA systems, the traditional approaches become an obstruction for designers to meet strict performance criteria. Lack of instant RF information in upper layers causes inefficiency of radio resource management (RRM), while procedure calls and responses bring unnecessary execution overheads. In view of this, designers began to choose cross-layer design

strategies, aiming to break inter- layer boundary and to enhance communication among layers. Many studies such as [5] [6] provide a theoretical view of cross-layer design.

In our studies, we define a reduced system architecture, and try to implement the system including MAC and PHY layers. The design and implementation of advanced MAC and PHY layers in mobile communication is challenging. Undoubtedly, the PHY layer functional blocks are mostly implemented as hardware (possibly on DSP). For the MAC part, since the functions become more complex, a pure-software design is not adequate, thus we need to partition the system into hardware and software parts. Traditionally, we design hardware part with FPGA and then port the software part onto the system such as [7] [8]. Now we are able to design HW and SW parts simultaneously with a platform-based strategy, in which the HW part can be modeled at the transaction level using systemC. This method significantly facilitates the software design since software can now be developed in parallel with hardware just after the hardware modules are defined. This also helps us to have better insight on the whole architecture and protocol issues.

The rest of the paper is organized as follows: First, we provide an overview of the WiMAX standard in Section II. In Section III, some cross-layer design issues are discussed. In Section IV, the proposed architecture for the MAC and PHY layer system are described. Section V includes a hardware-software co-design example and simulation results. Finally, conclusions are drawn in Section VI.

# 2 802.16-2005 MAC and PHY Overview

MAC Layer - The MAC layer of Mobile WiMAX can be roughly separated in to two planes, data plane and control plane, according to the functionality [1][2]:

### A. Data Plane

Data plane is responsible for forming protocol data units (PDU) from data packets, i.e. service data units (SDU) coming from upper layers. Construction of the data plane is based on the data flow between upper layers and the PHY layer, which can be classified into following steps:

# - Convergence Sublayer

including Packet Header Suppression (PHS) and Packet Classification. The packet classifier maps packets into various connections according to its service flow.

# - Fragmentation and De-fragmentation

For connections without ARQ support, the SDUs are subjected to further fragmentation and packing. For connections with ARQ support, the SDUs are reduced to fixed-sized blocks and are not apt for further changes.

#### - Header and Subheader

Headers and subheaders are appended to payload according to its contents and properties or control messages in between BS and MS.

# - CRC Computation

CRC field is generated for error detection. This is optional in IEEE 802.16-2004 but mandatory in 802.16-2005.

### - Framing

The PDUs are collected and concatenated, i.e. PDUs are packed into data bursts according to their destination.

# - Interaction with Control Plane

Another source of PDU payloads are management messages. The decision of message parameters is in the control plane, while the data plane is responsible for accessing the parameters and turns them into PDU payloads. This takes complicated bit manipulations.

# B. Control Plane

# - Mobility Control

This part handles mobility issues of MSs, such as motion, switching among multiple cells, and power adjustments. This part includes the following functional blocks:

- 1. Cell Reselection and Handover.
- 2. Idle Mode and Sleep Mode.
- 3. Active Mode Power Management.
- 4. Channel Measurement.

# - Network Entry

Network entry part in Mobile WiMAX in responsible for initializing network connection between MS and BS. It is also responsible for handling new network entries if MSs wish to handover from one network to another.

#### Scheduling

The system will arrange transmission order among SDUs of different connections according to scheduling rules. Scheduling algorithms vary in accordance to different QoS classes.

### - Management Message Composer

The parameters needed in a management message are gathered by the composer, and then passed to the data plane in the form of C language structure.

PHY Layer - The mobile WiMAX system has an OFDMA-based physical layer:

#### - IFFT / FFT

The kernel of OFDM, also the part that occupies most hardware resources in PHY implementation.

# - Interleaving and randomization

To combat the burst error problem of time-varying wireless channel

# - Channel coding and digital modulation

PHY encodes and modulates the signal to reflect the changes of link performance

#### - Synchronization and Channel Estimation

implemented at the receiver side with the help of preamble and MAC management messages

# 3 Cross-Layer Issues

As discussed in [5] [6], the cross-layer control and optimization strategy can be categorized into several classes. For example, top-down approach, in which the lower layer is controlled by a higher layer; feedback approach, which is top-down controlled with some information feedback from the lower layer; integrated approach, in which two or more layers are considered together. While we have several choices for each layer, an optimization task is setup to find a jointly optimal solution. The following part describes some issues that need cross-layer solutions:

#### A. Radio Resource Management

The MAC layer makes decisions on many functions: scheduling, handover triggering, power control, coding-modulation scheme selection, etc. These decisions require PHY layer measurements such as CINR, RSSI, BER, and others. In many circumstances, layered signaling is just not efficient enough for radio resource allocation. Reference layered structure passes parameters layer-by-layer and simply takes too much time, so that signaling contrarily becomes a bottleneck for high-speed MAC design. In view of this, new interfaces shall be added between adjacent layers, and even non-adjacent layers to provide a quicker runtime signaling.

#### B. Performance Considerations

As the system become more complex, cross-layer signaling is no longer as simple as connecting adjacent layers. The interface must be efficient enough so that overhead is minimized and thus can provide high speed and high throughput. In mobile WiMAX, this is partially done by MAC management messages parameters defined in the protocol. However, the actual passing of parameters and data stream, through hardware pins or software variables, still needs to be designed carefully.

To solve the problems stated above, we will examine the problems from the architecture point of view. The main idea is that for those functional blocks which are closely interacted should be designed as an integrated module. Some examples are:

#### - Framing:

Framing includes selection of the coding-modulation scheme and bit-loading. These are conventionally categorized as MAC functions. FEC encoding and modulation are categorized as PHY functions. These functional blocks comprise the kernel of data transmission. In conventional layered structure, the MAC layer constructs a complete frame including several data bursts and necessary control fields, such as FCH and DL/UL-MAPs. After that, the

MAC layer passes the whole frame to PHY layer in the form of bit-stream. However, we shall show the problem of execution overhead in the later parts of this article.

# - Ranging/Handover:

In ranging or handover processes, a device detects the environment to adjust its transmission parameter or switch to another base station. They both include the cross-layer signaling that the channel condition (CINR or RSSI) is passed from PHY to MAC, and control information (e.g. power adjustment, HO command) is passed from MAC to PHY. The measurements and parameters passing can be more efficient if we design specific connections among related functional blocks.

# 4 System Architecture

# A. Proposed System Architecture

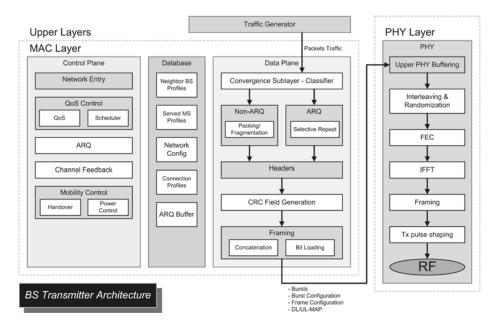


Fig. 1. Proposed System Architecture

Fig.1 is our proposed system architecture with cross-layer consideration, for the transmitting function of a base station (BS). The architecture includes two planes, the data plane and the control plane. Two planes are linked together by a public parameter database.

The data plane is based on the data flow between upper layers and the PHY layer, with separated flows for ARQ and non-ARQ connections. The control plane is classified into various blocks according to functionality. Between the two planes there is a shared database. Parameters, BS/MS profiles and network configuration profile are stored in the database. Control plane message composer, data plane, and algorithm modules will access the shared database. The receiver side is roughly the inverse of the transmitting one, except that the PHY part should be a little more complex to include synchronization and channel estimation. Receiver architecture is not discussed in this article.

### B. Modulized Algorithm Development

To facilitate algorithm development, the algorithm functional blocks must be fully modularized and can be easily added or removed. This can be done by applying a standard interface, as illustrated in Fig.1. The standard interface comes in the form of a variable database, accessed by all algorithm modules, message composer and data plane. If necessary, new variables and algorithm modules can be added at ease.

# 5 Design and Implementation Isssues

### A. Design Flow

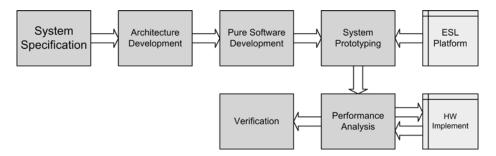


Fig. 2. Proposed System Design Flow

We implement the 802.16-2005 MAC and PHY layers in a platform-based ESL (Electronic System Level) design manner. The complete design progress starts from defining or understanding system specification, based on this, we have an architecture containing functional blocks with control and data signal flows. Then we implement the system with both hardware and software and then verify the design at final. The proposed design flow is described in Fig.2 and will be explained as follows.

To have a system view at a higher level, it is preferred to implement the system with pure software from the start. Algorithm development and layer interface issues are handled at this stage. We apply profiling in order to gain a

rough insight of system performance, and then considering the cross-layer cosimulation in order to ensure the function correctness.

After software development, to facilitate real-time operation, substitution of bottleneck functional blocks with HW implementation is a necessary means. For our communication system case, the PHY layer baseband processing is hardly to be kept in software due to its high computation power, so it will be transfer into a hardware model and than implemented with specific hardware architecture. The MAC data plane, especially the part dealing with SDUs, also contains computational operations that is routined and repeated, and is therefore more favorable for HW substitution. The control plane, which contains less frequently called functions and each requires variable amount of memory, is less favorable.

Note that the implementation of pure software does not need to have a complete control plane. An important benefit of ESL design is that the hardware and software can be developed simultaneously. This means, using hardware "models" with well-defined interfaces, we may refine the software without actual implementation (FPGA or ASIC) of hardware part. Take the scheduler as an example, we may use a simple FIFO queue at the beginning (in order to verify the hardware model), and applying more complex algorithms later.

In the example presented in Section VI, HW/SW co-simulation is done by building a processor based platform. The design tool used for platform construction is the ARM RealView SoC designer; like other similar tools, there is a canvas to draw a platform and simulate it.

### B. Effect of Cross-Layer Design on Timing

Besides radio resource issues, cross-layered design may bring great influence on hardware timing. In traditional discrete-layered designs, MAC and PHY threads are executed separately. The two layers are connected only through buffering and signaling. Only after the procedure of one layer is completed, it will then issue a procedure call and start another layer's procedure. However, buffering and signaling bring excessive memory accesses, which is the main reason for system throughput degradation. If a designer wishes to solve this problem, it would be better if the procedure sequence is re-arranged. By merging MAC and PHY layer into an optimized hardware module, we are able to reduce excessive memory accesses and thus improve system timing.

Here is an example illustrating how hardware timing is improved by merging both layers. Consider the sequential execution of packing- fragmentation, which is a function of MAC data plane, and then FEC encoding and modulation, which belongs to PHY layer. The sequence's goal is to generate data for n bursts, and then propagate these bursts through FEC coding and modulation.

Three execution schemes are examined:

- A. Discrete-layered execution, without any hardware optimization.
- B. Discrete-layered execution, with all PHY functions integrated into a hardware module.
- C. Cross-layered execution, with MAC and PHY functions integrated into a sin-

gle hardware module.

For scheme A and B, the execution sequence can be expressed by the following pseudo code:

```
MAC_Procedure()
for( i <= n_burst )</pre>
       while( burst[i] is not full )
           fragmentation/packing();
   PHY_Procedure ();
}
PHY_Procedure()
    retrieve_MAC_parameters();
    upper_PHY_buffer();
    for( i <= n_burst ) FEC_encoding( burst[i] );</pre>
    for( i <= n_burst ) Modulation( burst[i] );</pre>
}
   For scheme C, the pseudo code expression is:
Integrated_Procedure()
    for( i <= n_burst )</pre>
        while( burst[i] is not full)
            fragmentation/packing();
retrieve_MAC_parameters();
    FEC_encoding( burst[i] );
    Modulation( burst[i] );
}
```

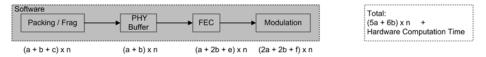
From the pseudo codes, we observe that in scheme A and B, there are three loop structures that each of them has n iterations, thus giving 3n conditional branches. In scheme C, there is only one loop structure, giving n conditional branches. Under massive amount of loop execution, this could bring obvious performance difference.

Also to quickly estimate system performance, we model the system's memory access behavior with the following assumptions:

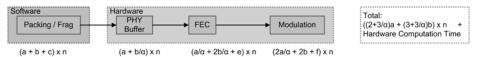
- Time needed for reading a burst's bit load from system RAM, denoted by a.
- Time needed for writing a burst's bit load to system RAM, denoted by b.
- Time needed for reading a burst's bit load from hardware cache, denoted by  $a/\alpha$ . Note that  $\alpha$  is the average speedup ratio between system RAM and hardware cache access.

- Time needed for writing a burst's bit load to hardware cache, denoted  $b/\alpha$ .
- Number of bursts, denoted by n.
- Computation time for packing/fragmentation, FEC encoding and modulation respectively, denoted by c, e and f .

#### Scheme A: Discrete MAC and PHY Design without HW Optimization



### Scheme B: Discrete MAC and PHY Design with PHY HW Optimization



# Scheme C: MAC and PHY Cross-Layered Design with Integrated HW Optimization



Fig. 3. Execution Sequence and Memory Access Time Estimation for Three Schemes

Note that a burst's bit load would increase through FEC encoding. Here we assume the code rate is 1/2, and therefore the bit load after FEC process is double of that before encoding. Also we focus on memory access time in this part; the acceleration within each functional blocks, i.e. changes in execution time c, e and f are not considered.

Refer to Figure 3, we can derive approximate memory access time for each execution scheme, shown as follows:

```
- Scheme A: (5a + 6b) \times n

- Scheme B: ((2+3/\alpha)a + (3+3/\alpha)b) \times n

- Scheme C: ((1+3/\alpha)a + (2+3/\alpha)b) \times n
```

Since the access speed of hardware cache is much faster than that of system RAM, we can assume that  $\alpha$  has a large value, and therefore we have further approximation shown as follows:

```
- Scheme A: (5a + 6b) \times n

- Scheme B: (2a + 3b) \times n

- Scheme C: (a + 2b) \times n
```

From the estimation stated above, hardware integration of MAC and PHY can dramatically reduce memory access time by almost 75%, thus improve system throughput.

#### B. Predict HW behavior with HW/SW Partition

The most exciting advantage of platform-based design is that we can develop hardware and software part simultaneously after a proper interface is defined by rough software and some hardware models. However, the interface can still be defined without this platform-based simulation, so why do we need it? The answer is that it provides us with more precise information of hardware behavior, including ports, cycle count, before actual hardware tape-out. But some information, like the cycle-accurate hardware execution time, is still unavailable until Verilog designs are completed. If a designer needs to know more about the performance evaluation in the early stage, a systematic way should be developed.

We formulate this problem with the following factors: For the system before HW/SW optimization, we have:

- $C_{Si}$ : Software execution cycle count of the i<sup>th</sup> functional block, per instance.
- U<sub>i</sub>: How many times the ith functional block is referenced.

# B. After HW/SW optimization, we have:

- C<sub>H</sub>: Target total cycle count of all hardware modules.
- C<sub>Hi</sub>: Cycle count of the ith functional block, after it is implemented with hardware.

The designer sets up a fixed goal for  $C_H$  according to the timing constraint, and then allocates a target  $C_{Hi}$  value for each functional block which will be implemented with hardware. The goal is now to find out a combination of  $C_{Hi}$  values that minimizes the system cost, say hardware area, under a fixed constraint of total  $C_H$ .

Here we introduce another factor,  $\gamma_i$ , which is a characterization factor of the ith functional block. The factor indicates how easy the functional block could be boosted with a compact sized hardware implementation. As an example, functions such as FFT and IFFT can be accelerated with Butterfly structure, which is relatively small in size. Therefore they have higher  $\gamma_i$  values.

Then the *cost function* can be defined as:

$$J' = \sum_{i} \frac{C_{Si}}{\gamma_i C_{Hi}} \tag{1}$$

J' has the same trend with required hardware area, and therefore is a good indicator of hardware area needed for the functional block.

Our goal is to minimize J', subject to the constraint

$$\sum_{i} C_{Hi} Ui \le C_H \tag{2}$$

where CH is a fixed target, we solve the optimization problem of minimizing J' with Lagrange Multiplier Method, shown as follows:

$$J = J' + \lambda (\sum_{i} C_{Hi} U_i - CH) \tag{3}$$

where  $\lambda$  is the Lagrange Multiplier. We seek for the minimum value of cost function by taking partial derivation with respect to CHi, and let it equal to zero:

$$\frac{\partial J}{\partial C_{Hi}} = -\frac{C_{Si}}{\gamma_i C_{Hi}^2} + \lambda U_i = 0 \tag{4}$$

Solving equation 4, we have:

$$C_{Hi}^2 = \frac{C_{Si}}{\lambda \gamma_i U_i} \tag{5}$$

from which we can allocate each functional block an adequate target cycle count, and minimize the corresponding hardware area cost. This is a reasonable allocation since:

- For a functional block which costs more software cycle originally, we allocate more hardware cycle.
- For a functional block which is used more frequently, we allocate less hardware cycle to reduce the overall hardware cycle, at the expense of increasing area or power.

With similar procedures, designers are also able to compute the allocation of hardware resource subjected to other criteria such as hardware power.

Here we consider a simplified implementation of WiMAX MAC and PHY layers. The pure software version needs about 14.6 M cycles to complete a frame, which equals to 146ms with a fully utilized processor running at 100 MHz. This is the estimation result without any aid of application-specified hardware or DSP modules, and is much higher than the standard-specified 5ms goal. Therefore, designers have to consider optimizing some functional blocks with hardware.

According to the estimation method described in equation 1 to 4, Table 1 shows the profiling results of two heavy loaded modules, IFFT and FEC Encoder, and their corresponding parameter settings. A simulation period of 20 frames is applied.

Function	$U_i$	$C_{Si}$	$\gamma_i$
		313511	
(2) FEC Encoder	320	192995	1

Table 1. Profiling Result

Setting  $C_H = 600,000$  cycles allows the frames to be generated on time (processor not fully-used). Using equation 2 and 5, we allocate  $C_{H1} = 400$  cycles, and  $C_{H2} = 1000$  cycles.

# 6 Conclusion

In this paper, we presented a cross-layer design methodology of Mobile WiMAX MAC layer, involving both hardware and software designs. By combining MAC and PHY layer, designers are able to reduce excessive memory access and looping, thus improve system performance. Also, in a HW/SW co-design platform, we can allocate target cycle counts for each HW component according to some rules so that the cost is minimized.

# Acknowledgment

This work is sponsored by Institute for Information Industry, Taiwan.

# References

- IEEE Standard for local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems, IEEE Std. 802.16, 2004.
- IEEE Standard for local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1, IEEE Std. 802.16, 2005
- 3. Taesoo Kwon et al., "Design and Implementation of a Simulator Based on a Cross-Layer Protocol between MAC and PHY Layers in a WiBro Compatible IEEE 802.16e OFDMA System", IEEE Commun. Mag., Dec 2005, pp. 136-146.
- 4. M. Settembre et al., "Performance Analysis of an Efficient Packet-Based IEEE 802.16 MAC Supporting Adaptive Modulation and Coding", Proceedings of the Seventh IEEE International Symposium on Computer Networks, 2006.
- Vineet Srivastava, "Cross-Layer Design: A Survey and the Road Ahead", IEEE Communications Magazine, December 2005.
- Qi Wang et al., "Cross-Layer Signaling for Next-Generation Wireless Systems", Wireless Communications and Networking, WCNC 2003.
- H. Holisaz et al., "Hardware Accelerator IP-Core for Wireless 802.16 MAC", IFIP International Conference on Wireless and Optical Communications Networks, 2006.
- Nak Woon Sung, "HW/SW Co-designed Implementation of IEEE 802.16 TDMA MAC for the Subscriber Station", Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science, 2005.