12

Design and Implementation of WiMAX Baseband System

Zhuo Sun, Xu Zhu, Rui Chen,
Zhuoyi Chen and Mingli Peng
Beijing University of Posts and Telecommunications, China

1. Introduction

Design and implementation of a wireless communication system protocol stack on the hardware platform are challenging tasks, which are seldom mentioned in published results currently. In fact, the work aims at achieving the predetermined function and performance based on the specific hardware resource, which involves how to design the software architecture according to hardware, how to choice the suitable algorithms and program them optimally on programmable DSP or embedded processor, etc. Typically, there are two ultimate application modes for a communication protocol stack: dedicated ASIC or programmable DSP. However, before the protocol stack is formed into dedicated ASIC, the task of protocol or algorithms implementation and testing on the programmable DSP should be finished primarily. Therefore, the chapter will focus on the topic of development of the WiMAX PHY/MAC protocol on a multi-core DSP platform.

The contents of this chapter will be organized into three sections:

The first section, we will discuss the overall software and hardware architecture of a WiMAX TDD baseband system, and what are the most import considerations in this design phase.

The second section, we address on the topic of developing the WiMAX PHY protocol on the programmable multi-core DSP platform.

The third section, it is about the design and implementation of WiMAX MAC protocol on the embedded processor, on which embedded Linux OS is running.

2. Hardware platform for WiMAX system

2.1 Baseband system

2.1.1 Requirements

Wireless systems designers need to meet a number of critical requirements including processing speed, flexibility, and time-to-market, all of which ultimately drive the hardware platform choice.

- Processing throughput

WiMAX and LTE broadband wireless systems have significantly higher throughput and data rate requirements than W-CDMA and cdma2000 cellular systems. To support these high data rates, the underlying hardware platform must have significant capability of processing throughput. In addition, advanced signal processing techniques such as Turbo coding/decoding, and front-end functions including fast Fourier transform/inverse fast Fourier transform (FFT/IFFT),beam-forming, MIMO, crest factor reduction (CFR), and digital pre-distortion (DPD) are computationally intensive and require several billion multiply and accumulate operations per second.

- Flexibility

WiMAX is a relatively new market and is currently in the initial development and deployment stages. Similarly, 3GPP LTE is being defined and will go through numerous revisions before being finalized. While there are many competing mobile broadband technologies, such as WiMAX, LTE, and UMB, their common thread is OFDMA-MIMO (Parssinen et al., 2000). In this current scenario, having a flexible and reprogrammable product is necessary to provide a standards-agnostic or multi-protocol base station. Systems offering this flexibility can significantly reduce the capital expenditures and operating expenditures for wireless infrastructure OEMs and operators while alleviating risks posed by constantly evolving standards.

- Cost-Reduction Path

A valuable lesson learned from designing and deploying 3G systems is the importance of establishing a long-term cost-reduction strategy in the beginning. Evolving WiMAX and LTE standards are expected to stabilize. For OEM sand service providers to remain competitive in the marketplace, the cost of the final product eventually will be more important than flexibility. Choosing the right hardware platform also provides a seamless cost-reduction path for production volumes, saving millions of dollars in engineering costs that would otherwise be incurred by system redesign.

2.1.2 Generic hardware architecture

The goal of LTE and WiMAX is to provide a high-data-rate, low-latency and packet-optimized radio-access technology supporting flexible bandwidth deployments. In addition, new network architecture is designed with the goal to support packet-switched traffic with seamless mobility, quality of service and minimal latency.

Due to the demand of high performance of high data-rate, low-latency and reduced delays, the choice of LTE hardware platform is a challenging job. The general architecture of the mobile communications system is depicted as the figure below.

The architecture can be divided into three parts: radio frequency (RF), intermediate frequency (IF) and digital baseband. The mainstream design of receiver proceed as follows: sampling the analog signal in the intermediate frequency, then down-converting the digital signal to baseband, finally demodulating the digital baseband signal. The transmitter is just opposite of the receiver (Dohler et al., 2005). Moreover, there may be CFR (Crest Factor Reduction) and DPD (Digital Pre-Distortion) modules before converting to analog signal.

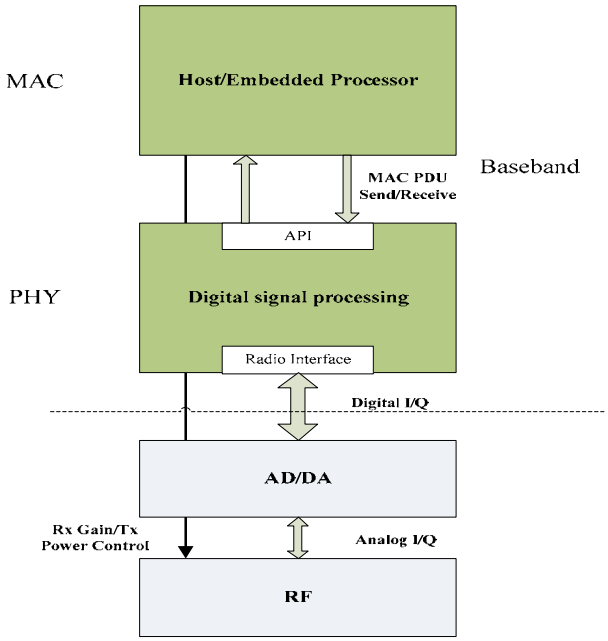


Fig. 1. The general hardware architecture of mobile communications system

The process in IF contains NCO (numerical controlled oscillator), CIC filter, half-band filter and FIR filter. All of the modules mentioned above have such a simple structure that they can be easily implemented in an FPGA. Because of the high data-rate of IF digital data, it is impossible to implemented in DSP. For example, first decimation filter in a digital wireless receiver, typically, is a CIC filter, operating at a sample rate of 50-100MHz. At these rates any DSP processor would find it extremely difficult to do anything. However, the CIC has an extremely simple structure, and implementing it in an FPGA would be easy. A sample rate of 100MHz should be achievable, and even the smallest FPGA will have a lot of resource available for further processing. In addition, the latest mobile communications system has employed MIMO technology, which means there will be two or more antennas. As the fact of introducing MIMO, the parallel sampling data will be processed simultaneously. Therefore, it is general to choose FPGA or ASIC as the processor in IF instead of DSP. For example, AD6654 is an IF to baseband receiver, with programmable decimating FIR filters, interpolating half-band filters and CIC filters built-in (Fathi, 2004).

Baseband is usually divided into two parts. One of them is digital signal processing, which is used for implementing PHY protocol. The other part is microprocessor used for implementing MAC and lower protocol (Goldfarb et al., 2000). The microprocessor is generally selected within ARM or Power PC processor. However, the choice of baseband digital signal processing solution is various. Generally there are four choices which are ASIC, DSP, FPGA and DSP+FPGA (Jusslia et al., 2001). Application-specific integrated circuit (ASIC), is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use. Digital signal processors (DSP) are a specialized form of microprocessor, while FPGAs are a form of highly configurable hardware.

- ASIC

According to circuit functions and performance requirements, ASIC design needs to select circuit form, the device structure, process plan and design rules to minimize chip area, lower design cost and shorten the design cycle, and finally brings forward the correct and reasonable mask layout. Nevertheless, the disadvantages of full-custom design can include increased manufacturing and design time, increased non-recurring engineering costs, more complexity in the computer-aided design (CAD) system. Moreover, Due to the changing demand of mobile communications system, the equipment has to upgrade one day, but ASIC cannot upgrade flexibly. When the hardware platform does not meet the requirements, all of the equipment must be replaced. As a result, the cost of upgrading is very expensive.

- DSP

A digital signal processor (DSP) is a specialized microprocessor with an optimized architecture for the fast operational needs of digital signal processing. Digital signal processing algorithms typically require a large number of mathematical operations to be performed quickly and repetitively on a set of data. Signals (perhaps from audio or video sensors) are constantly converted from analog to digital, manipulated digitally, and then converted again to analog form. Many DSP applications have constraints on latency; that is, for the system to work, the DSP operation must be completed within some fixed time, and deferred (or batch) processing is not viable (Parssinen et al, 1999).

- Multi-core DSP

Multi-core processing is the technology or group of technologies that companies like Intel and IBM are betting will replace Instruction Level Parallelism and the clock rate ratchet: dual and quad core systems for desktop applications are already in volume production. As we have seen, in many cases the controlling factor in device performance has moved from the ability to complete computation to the ability to move data. Well designed multi-core architectures allow data stores from registers to main memory to be distributed throughout the system, in whatever way makes most sense for the application. In fact, in multi-core architectures the communications fabric can substitute for memory accesses by allowing direct communication between the processing elements. If matched to the task in hand, such an infrastructure can therefore intrinsically help to overcome any restrictions imposed by the need to move data.

- FPGA

The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare). FPGAs can be used to implement any logical function that an ASIC could perform. The ability to update the functionality after shipping, partial re-configuration of the portion of the design and the low non-recurring engineering costs relative to an ASIC design (notwithstanding the generally higher unit cost) offer advantages for many applications.

- FPGA-DSP Co-Processing

FPGA and DSP represent two very different approaches to signal processing – each good at different things. There are many high sampling rate applications that an FPGA does easily,

while the DSP could not. Equally, there are many complex software problems that the FPGA cannot address. Another advantage of co-processing is reconfigurable features of FPGA, which means that engineers can quickly build and modify the design architecture. Moreover, FPGA supports the integration of other components (such as Serial Rapid IO transceiver, PCI Express interfaces, glue logic and low-rate control task), which reduces overall system cost and power consumption. In addition, the integration of so many interfaces is valuable for scalability, which meets the changing demand of mobile communications system. As a result, the ideal system is often to split the work between FPGAs and DSPs.

2.1.3 Communication between MAC and PHY

In the following discussion, we adopt the picoChip PC7205 development platform as our baseband hardware platform, in which integrated one of the multi-core DSP processor (PC205) and one piece of FPGA. The PC205 process also includes an embedded ARM926EJ processor that could implement the MAC functions.

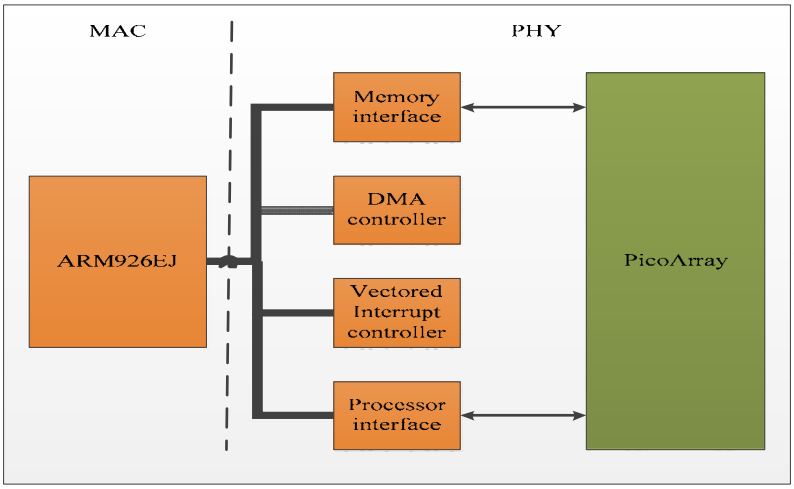


Fig. 2. PC205 block diagram

The PC205 microprocessor interface is designed for communications with a processor. No specific processor family is assumed and data can be exchanged over 8, 16 or 32-bit wide data bus. The processor interface is used for communication between MAC and PHY.

The Processor interface supports two basic transaction types

1. Single (GPR) – Reading or writing one word at a time.
2. Burst (DMA) – Reading or writing words at the same rate as the microprocessor proc clock.

GPR accesses allow access to the majority of memory mapped registers and services within the processor interface, GPR accesses can only be used for single read / write accesses. Typically, GPR is used for transmit control signals whose amount of data is small such as automatic power control (APC) signal between MAC and PHY.

DMA Accesses are primarily used for the efficient movement of data to and from the picoArray. Generally we use DMA to transmit the bulk data such as wireless frame between MAC and PHY. Fig.3 shows the DMA channel configured for write access. A FIFO buffers the data written from the microprocessor to the selected DMA channel. The FIFO output is connected to the picoBus and data is transferred to the internal array elements by using a get command from within the software.

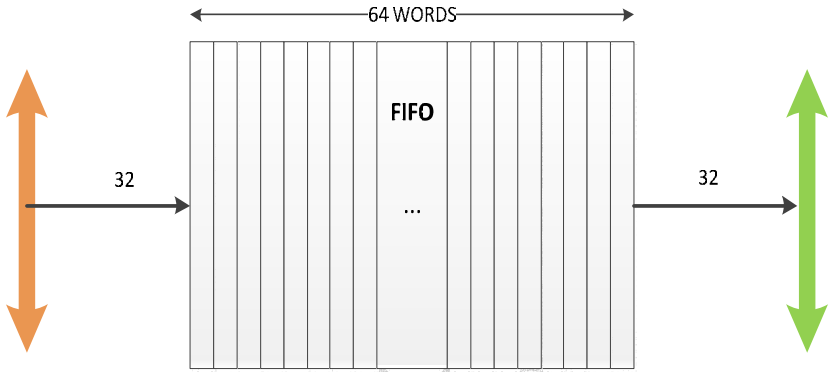


Fig. 3. DMA channel configured for write access

The PC205 support 3 DMA transfer mechanisms.

1. Basic downlink Host processor to picoArray
2. Basic uplink picoArray to Host processor – not practical
3. HWIF_UL picoArray to Host processor

In the mode of Basic downlink, host processor initiates transfer. The process is as followed.

- Open a transport session
- Configure the transport session
- Start the transport session
- Write DMA data
- Close the transport session

In the mode of basic uplink, host processor initiates transfer, but the processor has to pre-assumes data size, which is not practical.

In the mode of HWIF_UL, PicoArray uses handshake mechanism to indicate data size through GPR registers, and uses interrupt to initiate transfer through ITS register. The process is as followed.

- Open and setup an interrupt clearing transport session used for clearing down interrupts.
- Open and setup a HWIF_UL transport session for uplink DMA.
- Poll for an event indicating the date available.
- Read DMA data
- Close the transport session

2.2 Connecting baseband to RF

2.2.1 Connecting to RF by using FPGA

Generally there are two kinds of interface signals between baseband and RF, which are data and control signals respectively. Data signal are usually 16 bits complex format. However, the control signals may adopt one communication protocol such as SPI, I2C and so on. Moreover, some of the protocols may be changed for the sake of implementation. The interface of DSP may not support all the protocols. As the result, it is the right way to introduce one piece of FPGA between the baseband and RF for flexibility and scalability. We can write the suitable protocols for almost all interfaces in FPGA.

In addition, for the sake of power consumption, more and more DSPs have chosen 1V and 1.8V as power supply of the core and interfaces respectively. But the other device may take 3.3V as the power supply of the interfaces. It is obvious that electrical characteristics don`t matched between the different interfaces. Nowadays, most of FPGA have more than one bank, and each bank can be supplied different power. We can use some of banks with 1.8V power as the interfaces with DSP, while the other banks with 3.3V power as the interfaces with some other device. As a result, it is easy to change the logic level.

2.2.2 Automatic gain control (AGC) and power control

Automatic gain control (AGC) Automatic gain control (AGC) is an adaptive function found in many electronic devices. In a digital communication receiver strong signals that fall outside the narrowband digital filter bandwidth, but inside the analog IF translator bandwidths, can overload or saturate the A/D converter. This results in the generation of in-band IMD products and can result in significant degradation of the desired signal. If large signal levels are detected at the A/D converter, the receiver gain may have to be re-distributed by reducing the pre-conversion analog gain and increasing the digital gain to maintain the desired signal output level. This will, however, reduce the desired signal-to-quantization noise ratio.

- Power Control

In the WiMAX system, there are two mechanisms for power control, which are open loop power control and closed loop power control respectively. It is necessary to have closed loop power control while open power control is optional. Closed loop power control means that the base station (BS) controls the transmission power of the mobile station (MS). The MS transmission power is controlled in order to avoid exceeding the BS`s total receiving power from an antenna. In the WiMAX standard, other uses of it are not defined (i.e., the uplink TPC algorithm is vendor specific).

- AGC and Power Control signal design

Usually RF device has the specific module for receiving gain and transmission power adjusting, which is controlled by voltage signals. Therefore, baseband just outputs direct current signals with variable amplitude to RF. Typically we can obtain the direct current signals with low rate DAC, but the interface between baseband and RF have to increases parallel lines used for transmitting the digital. Here we introduce a simple method to generate the direct current signals.

We can make use of Pulse Width Modulation (PWM) signal and a RC low-pass filter to generate the direct current signal. When the PWM signal duty ratio is 100 percent, the amplitude of direct current signal equals to the amplitude of PWM signal. When the PWM signal duty ratio is 50 percent, the amplitude of direct current signal equals to the half amplitude of PWM signal. The direct current signal is approximately linear with the duty cycle. In this case, it is necessary to use two digital signal lines for receiving gain and transmission power control.

2.2.3 Extern GPS synchronization signal

The IEEE 802.16 standard calls for the use of global positioning system (GPS) receivers to provide the precise time reference for synchronization of WiMAX networks. This operation is performed both during the startup and periodically in order to maintain the alignment with the external PPS pulse.

Briefly, the algorithm follows these steps. The controller of synchronization starts searching for the first PPS pulse while discarding the RX samples. Then it stalls the PHY while waiting for the PPS pulse and sends DL dummy complex samples. Once received the PPS pulse, after 100 ms, the controller starts passing the DL complex samples. For each frame period, the frame synchronization module receives the frame start indication and decides when a frame adjustment is required for maintaining the alignment with the external pulse.

3. Physical layer implementation

3.1 Introduction of PHY

Considering PHY implementation, the main functions of PHY layer may comprise API, control, transmit-path, receive-path and synchronization/radio interface for both BS and MS entities (LAN/MAN Standards Committee of the IEEE Computer Society et al., 2008). Fig.4depicts the relationship between these function blocks.

API:

The API provides an interface between the PHY and MAC. Its function is responsible for:

- The physical transfer of data to and from the MAC
- Buffering of data to and from the MAC
- Error checking and diagnostics on the data
- Interpreting data from the MAC and generating internal control data for other functions in the PHY
- Interpreting data from the PHY and parsing into data for the MAC

Control:

The control function is responsible for distributing control data originating in the API around the other functions of the PHY. Its most important task is to ensure that each of the data-path functions has access to the control data that it needs to process the data-path data it is currently working on. Secondly it orchestrates the collection of measurements from the PHY and provides routes for diagnostic information to be accessed by the MAC or other external processes.

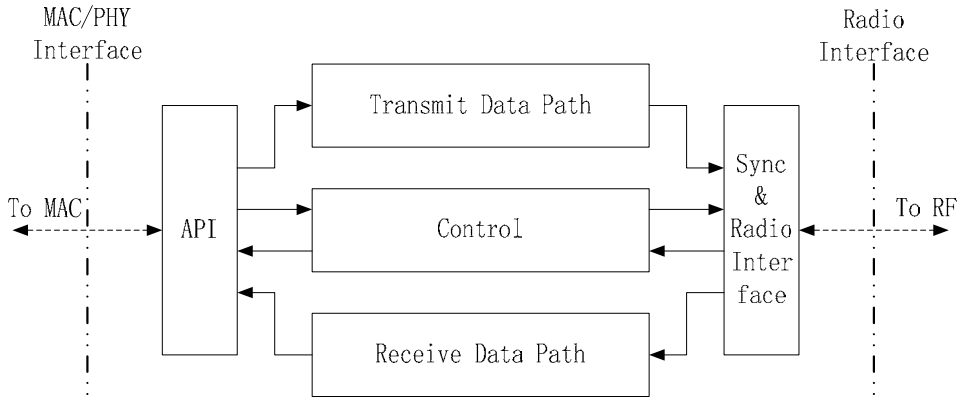


Fig. 4. Main function blocks of PHY Layer

Transmit-path:

This block provides the main data path for transmit in the PHY, which comprises the following stages:

- Unpacking of data into FEC blocks, Encoding, including randomization, FEC, interleaving, repetition---FEC block
- Data modulation: The conversion from bit stream to QPSK, 16QAM, 64QAM symbols---ConstPack block
- OFDMA zone processing, including pilot and preamble generation subcarrier permutation, subcarrier scrambling and zone boost---BurstZoneblock&AntEnc block
- Antenna processing, including IFFT, cyclic prefix, Peak-to Average reduction and transmitting filtering---FrondEnd block

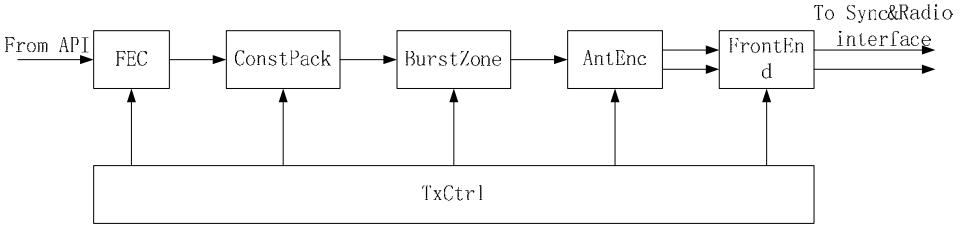


Fig. 5. BS transmit process

Receive-path:

This block provides the main data path for receive in the PHY, which including:

- Time and frequency synchronization process---MsRxAcq block &MsRxTf block
- Antenna processing with receiving filtering (Foschini, 1996), ALC, cyclic prefix removal and FFT---MsRxTf block

- OFDMA zone receive, with AAS/MIMO (Mugenet al., 2007) processing and buffering, subcarrier descrambling and depermutation, pilot extraction, CPE and frequency compensation, channel estimation and equalization, constellation demaping, channel state compensation and MRC---MsRxSym block &MsRxMap block
- Decoding: including derepetition, deinterleaving, depuncturing, H-ARQ (Lin and Yu, July 1982), FEC decoding, derandomising and repacking of user data into PDUs---MsRxBurstChain block

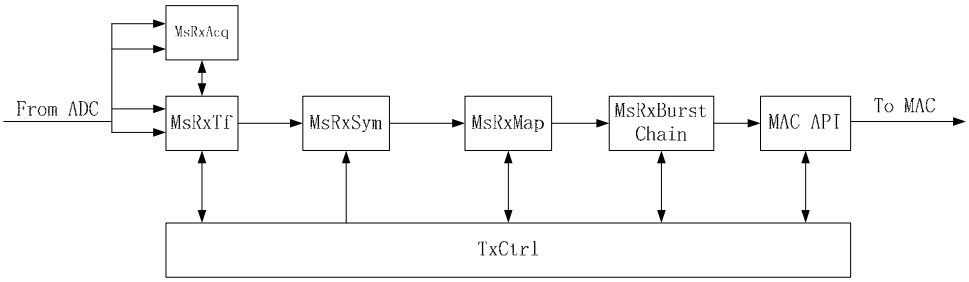


Fig. 6. MS receive process

Synchronization /radio interface:

This block is responsible for controlling the absolute and relative timing of uplink and downlink frames in the PHY and the radio. It is also responsible formultiplexing, demultiplexing and formatting data for the interface to the radio via the picoArray ADI (Asynchronous Data Interface) interface(PicoChip Company, 2008). For this reason realizing block may well be somewhat platform specific.

3.2 Link-level simulation

3.2.1 Simulation platform based on MATLAB

Before realizing the whole WiMAX PHY layer software on the picoArray DSP (PicoChip Company, 2008), a fixed-point link-level simulation is needed. First of all, it's important to make sure that the algorithms are correct and satisfy the performance demand in simulation environment. Because the development on DSP processor is time-consuming and expensive, the consequence of implementing a system that will never work in DSP processor can't be affordable. Secondly, it's very difficult to locate bugs and correct them in DSP processor. When the bugs have nothing to do with hardware, the bugs finding and correcting work can move back to simulation platform. This will save your development time and cost significantly. At last, there are varies wireless channel models in MATLAB, which are very useful for us to figure out how the system performs on different channel environment.

MATLAB simulation platforms are floating point in common situations. But this simulation platform does an extra job that it converts the calculation result from floating point to fixed-point result. That is to say, the simulation platform is a fixed-point platform which can be more approaching to fixed-point picoArray DSP processor. In this way, the performance between MATLAB simulation platform and DSP process on picoArray will be the same roughly. It makes the simulation more convincing.

3.2.2 Simulation platform development

The MATLAB simulation platform is built in the way that all blocks of the platform are map to the functions on picoArray respectively. So, the function blocks are transferred from MATLAB platform to DSP platform smoothly. Moreover, each block of the MATLAB fixed-point simulation can generate the corresponding result of this block which can be used as input of the followed block on the DSP platform directly. That is a very efficient way to verify the functions on the picoArray.

Matlab platform is built in accordance with the WiMAX physical layer protocol. The platform complies with the frame structure and resource distribution of WiMAX standard. It can generate any structure's sub-frame. Also it can provide fixed-point simulation for each sub-frame. Moreover, the platform can generate test vectors for each of the WiMAX PHY's module. The test vectors can be mapped to the AE level (picoChip processing unit), including the input control information and the input and output data of each AE.

3.3 PHY protocol implementation on picoArray DSP

In the following, the PHY protocol implementation on picoArray DSP, the chosen Multi-Core DSP for baseband application will be introduced in details.

3.3.1 PicoArray introduction

The picoArray multi-core DSP is based on a massively parallel architecture comprising large numbers of small independent processors. A DSP application is logically decomposed into a number of communicating sequential processes, each of which is assigned to a particular processor on the picoArray. The designing tools statically allocate processors and picoBus (PicoChip Company, 2008) resources for the system, so there is no need for an operating system. The static allocation of resources allows much of the system's runtime complexity to be moved back into the tool suite. It allows the hardware to be lightweight and hence allows a very high proportion of the power of the processor array to be used for the real DSP application.

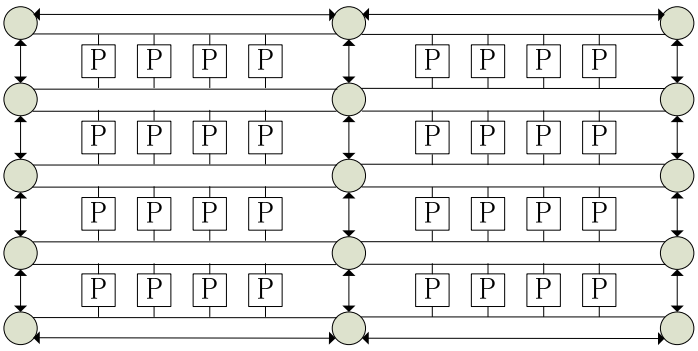


Fig. 7. A simplified representation of a picoArray

Fig.7 shows a simplified representation of the overall structure of the picoArray. Each box marked 'P' in the figure represents a single processor, referred to an Array Element(AE).

The processors are laid out in a grid, interconnected by a matrix of buses called the picoBus. Each AE is connected to two buses. The lines between the processors represent the picoBus, and the circles represent bus switches which connect buses together to provide routes between all AEs in the array. The communications between these processes, called signals, are then mapped on to physical segments of the picoBus between the assigned processors by suitable settings of the bus switches. The heavy red and blue lines illustrate two example connection paths between particular processors. Communication between AEs is time-multiplexed over the picoBus, which is a shared resource.

3.3.2 PHY implementation

A PicoArray process is composed of a number of AE which can work simultaneously. This structure is quite different from traditional single-core processor. As a result, developing work on picoArray will share nothing with that on traditional single-core processor. The major steps of developing work on picoArray are given as follows.

All the Instruction/Data memory that developers can utilized are in the core in traditional processors, so developers needn't care about how to arrange Instruction/Data memory for each functional block. But when doing developing work on picoArray, the first thing developers need to deal with is to select suitable AE for each functional block. There are some different categories of AE which have quite different abilities. Three types of AE are mostly used in our work: STAN2 (short for standard AE), MEM2 (short for memory AE) and CTRL2 (short for control AE).

Feature	CTRL2	MEM2	STAN2
Bus connections	4	2	2
Number of ports	32	12	10
Number of registers	15	15	15
Instruction/Data memory size options (bytes)	49152/16384232768/327681 16384/49152 0 default 0	6656/2048 3 4608/4096 2 2560/6144 1 512/8192 0 default 0	512/256
Byte memory accesses	Yes	Yes	No

Table 1. AE types in PicoArray processor

For example, there are 196 STAN2 AEs, 50 MEM2 AEs and 2 CTRL AEs in one piece of picoArray PC205. The ability contrast is shown in the above table. When arranging the functional blocks into AEs, developers should select a good enough type of AE to implement the block according to the need of the blocks. Moreover, developers should optimize the code of the block to fulfill the limit of different types of AE. If one piece of picoArray can't hold all the blocks, another piece of picoArray processor should be introduced in to share the burden. The communication between the two pieces of picoArray processor is accomplished by IPI (Inter PicoArray Interface).

After developers have selected AEs for each block, memory access type should be selected. There are three types of data memory for data storage: data registers inside AE, data memory inside AE and SDRAM outside AE. The data registers are very fast access memories. When storing a small amount of variables, developers ought to use the registers as far as possible. There are only 15 registers in one AE as shown in table 1. If the size of data which needs to be stored exceeds the amount of unused registers, the data should be stored in the inside data memory. The access speed of this type of memory is a little slower than registers but is much faster than SDRAM outside of AEs. Last but not least, if the size of data exceeds the capacity of the memory inside AE, SDRAM is used to store it. In this situation, developers must arrange the SDRAM access area very carefully because the SDRAM is shared among different AEs which need it for their data storage. If some of the AEs use the same area in the SDRAM, fatal error will take place unexpectedly and is difficult to discover.

Then it comes to programming step. Firstly, functional code is created carefully based on the MATLAB simulation platform for each block. Then verify the code's syntax accuracy and logical accuracy with picoTools. The next step is taking the MATLAB simulation test vector as the input vector of each block. The function of block on every AE is verified by through comparison between the AE output and the MATLAB simulation result. The throughput matching is another important process when programming. The reason is if the throughputs among the blocks don't fit for each other, they can't work when connected together. If this problem happens, you should change your design to matching the throughputs demand. One principle is that the getting/reading port rate of the AE must be faster than the rate of the AE putting/writing data.

Last but not least, debug on picoArray. All the above coding and debug work is done with picoTools on development environment. It is easy to select the result of each block in the form of text document to be compared with MATLAB simulation result for verification. When it comes to the debug work on the picoArray, it is much more difficult to get the result of each block. As some hardware-related bugs can't be discovered on software environment, it's very important to get some methods for the debug work on picoArray. Fortunately, a probe mechanism is provided. You can configure the unused AE or unused SDRAM to get the output of the block which is needed to debug. The data in the 'probe' AE or SDRAM can be transmitted into text document to compare with MATLAB simulation result of the same block.

Some tips of developing on the picoArrays are given as follows:

First, the sum of the rates of signals connected to one single AE can't exceed 2 because there are two channels connecting one AE to picoBus. For example, if one AE has three signals names sig_A, sig_B and sig_C. The rates of the three signals are @2, @2 and @1 respectively. It is easy to find out that 1/2+1/2+1/1=2, so there are no channel space for another signal to connect to this AE.

Secondly, for a process chain, the getting data rate of AE must be faster than the providing data rate of the previous AE. This is very important rule during the picoArray DSP design, because if this can't be satisfied, data flow would be blocked. Moreover, if one AE's data flow is blocked, the conjoint AE' data flow will be blocked too.

3.4 API architecture design

3.4.1 API introduction

The API interface is between the MAC and OFDMA PHY, which is defined to picoChip's IEEE 802.16e base-station PHY and optional lower MAC accelerator to perform CRC and HCS calculations. The API described in this document is based on the Wireless MAN-OFDMA PHY. The greatest feature of API is to process data and control information separately.

- The API addresses data and control plane functions. Data plane functions include the transfer of MAC PDUs in the uplink and downlink directions via the PHY service access point (SAP). Information required for uplink and downlink processing is sent separately within a frame configuration structure.
- Control-plane functions include determining the capabilities of the PHY, reconfiguration of the PHY and notification of error conditions, alarms and measurements gathered from the PHY or radio subsystem.
- The base-station PHY can be configured to perform CRC and HCS calculations to lessen the processor requirements for the MAC. Encryption and decryption is beyond the scope of this API document.
- A frame-sync interrupt is sent to indicate the start of every downlink sub-frame; this mechanism operates in parallel to this API.
- The response and indication primitives sent from the PHY to MAC can be masked at PHY configuration, allowing the MAC to select only the messages it is interested in.

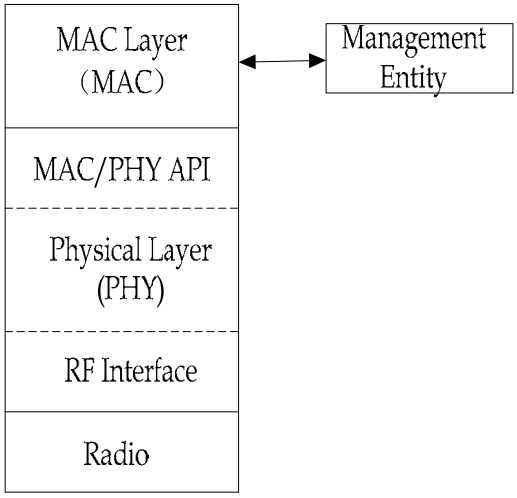


Fig. 8. 802.16e Protocol Stack

Fig.8 shows a modified version of the 802.16e reference model and the location of the API as described in this document (shown as MAC/PHY API). The 802.16e protocol stack is shown, together with the radio and management entity.

3.4.2 The block diagram of API

According to the function, the API is divided into two parts: communication with the MAC and the data processing chain of PHY layer. The first part includes getting messages from MAC, sending data to the SDRAM and regrouping the receive message to MAC. The messages from MAC include control-plane messages, which are used to perform configuration and reconfiguration of the PHY, and data-plane messages. Both messages have the same message format which includes message header, which descript message type and PHY entity, and message body. The receive messages to MAC are called response or indication messages. Then it comes to communication with PHY. The input signals of each module in the data processing chain can be classified into two types: control and data signal. Among them, all control signals come from the API; data signal is the output of the previous module. However, the beginning of the data also derives from the API. So, API separately process control and data information to produce the two signals. Their modules are Control system and Data processing, which are shown in Fig. 9.

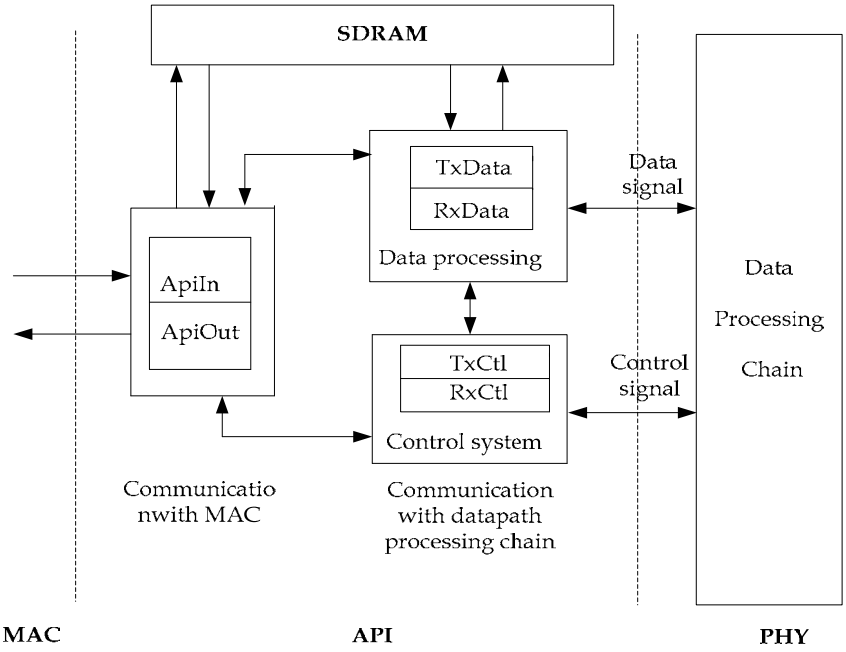


Fig. 9. The block diagram of API

4. MAC protocol implementation

4.1 Introduction of MAC functions

According to the OSI seven-layer network protocol，MAC lays between the PHY layer and the network layer, responsibility for the data convergence and resource scheduling. So there

should be appropriate interfaces between the PHY and network layers. In order to highlight the implementation of the MAC layer, for the following description in the chapter, the network layer is designed simply. Its main functions and features are shown in Fig.10 (Du, 2010).

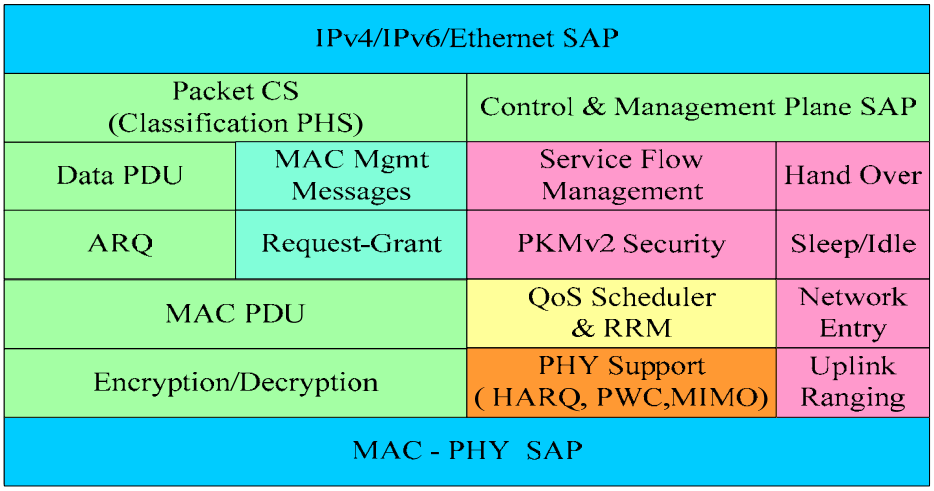


Fig. 10. The structure of MAC functions

4.2 Implementation of MAC layer

The embedded ARM-Linux operation system is chosen as the development environment of the MAC layer. Considering the requirement of the processing time, the multi-threads techniques are designed so that the MAC layer can packet and parse messages in time. Besides that, the algorithms adopted by the system are needed optimizing to achieve the compromise between system performances and the complexity. Because the message process procedures are different at the different BS/MS state, the state machine is designed to track the state of the BS/MS. The implementation details are introduced in the following part.

4.2.1 State machine

This section will introduce the BS state machine and MS state machine briefly. For BS, the state machine of BS from the startup to normal and the state of the accessed MS to the current BS are designed. For MS, the corresponding state machine is also designed. The conditions of the state transfer are defined.

1. State machine for BS

According to the functions of BS, the implementation of the BS state machine is divided into two modules. The first one is BS-State-Machine which manages the BS own state and state transition. The second one is MS-State-Machine which manages the states of MSs which have already accessed or attempted to access to the current cell. The BS-State-Machine is responsible for tracking the states of accessed MSs.

- BS-State-Machine module

The main functions of this state machine are to hold the current state of BS, parse the messages which are received from PHY layer, packet the messages which are sent to the PHY layer and transfer to another state. The BS state machine is designed as three states, namely STATE_CONFIG, STATE_NORMAL, STATE_NULL. The BS's initialization state is STATE_NULL. After the startup, the MAC layer entity of BS will configure the PHY layer with specific parameters and change the current state to STATE_CONFIG. The state will transfer to STATE_NORMAL when succeeding the configuration. The state of STATE_NORMAL indicates that the BS is working normally and any MS can try to access to it.

Fig. 11. The state of BS-State-Machine

- BS-MS-State-Machine module

The main functions of BS-MS-State-Machine are to hold the current state of MSs, parse the messages received from the PHY layer, packet the messages, send to the PHY layer and transfer to other state. According to the possible state of MS, the BS-MS-State-Machine is defined as having nine states, which are depicted in Fig.12 and Fig.13.

After the success of the initial ranging, BS will create a corresponding state machine for the MS. The BS-MS-State-Machine transfers from STATE_NULL to STATE_CONNECTED. The MS will execute the process of registration. Once registration succeeds, the state will transfers to STATE_NORMAL. During the STATE_NORMAL, MSs can establish the connection with BS to transmit the traffic of video, voice and data services.

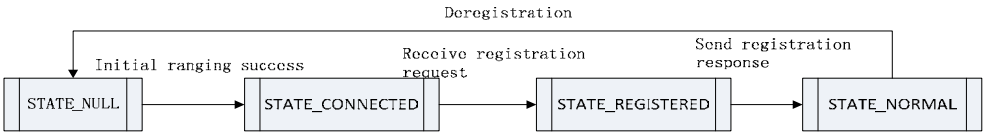


Fig. 12. The state of BS-MS-State-Machine

When the quality of the signals MS received is poor for a long time or the traffic capacity of the BS is saturated, the MS will consider to handover to another BS. Before the handover, the BS-MS-State-Machine will transfer to STATE_HOSCAN state to scan other BSs. After negotiating with target BSs, the MS will select the best one to process the handover confirm. Once receiving the allowance of the target BS, the BS-MS-State-Machine will transfer to STATE_HOPROCESS state to execute the handover. The target BS will initial a corresponding BS-MS-State-Machine and transfer to STATE_HOACCESSED state. The target BS changes into the serving BS. The target BS transfers the state to STATE_NORMAL and initial serving BS transfer the state to STATE_HOCOMPLETE. At this point, the scan and handover process is over.

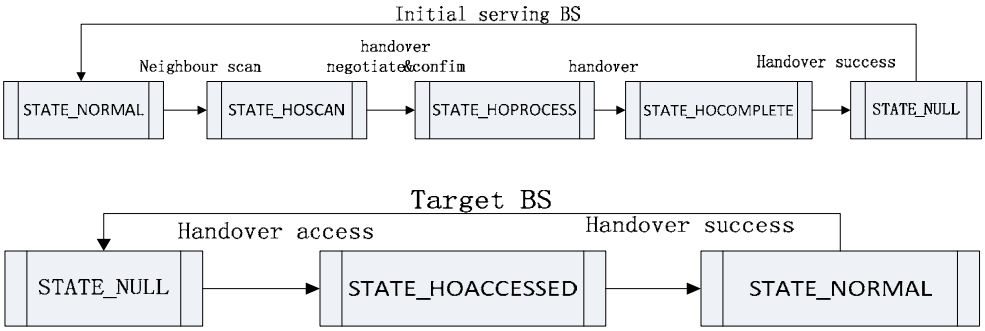


Fig. 13. The handover state of BS-MS-State-Machine

2. the design of MS state machine

The design principles of MS state machine are the same with those of BS. Similarly, the main functions of MS state machine are saving the current state, parsing the messages receiving from PHY layer, sending the packet messages to PHY layer and transferring the state.

- MS-State-Machine module

According to the behaviors of MS, the MS-State-Machine includes several states depicted as Fig.14. During the initial process of MS state machine, the state transfers from STATE_NULL to STATE_IDLE. When the MS receives the accessing indication from the upper layer, the MAC entity will process the PHY configuration and transfer the state to STATE_CONFIG. In order to synchronize with BS, the MS will scan the downlink channels to get downlink synchronization with the BS at the state of STATE_DL_SYN and achieve the uplink channel transmitting parameters to send the initial ranging request at the state of STATE_UL_SYN. After the synchronization, MS state machine transfers to STATE_RANGING and executes the process of registration at STATE_REGISTRATION. The state transfers to STATE_NORMAL after receiving the successful registration response from BS. The details are shown in Fig.14.

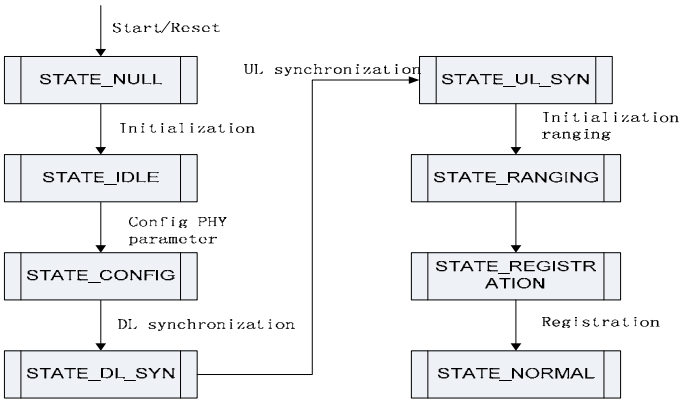


Fig. 14. The normal state setup of MSStaeMachine

When the quality of the MS received signals is poor for a long time or the traffic capacity of the BS is saturated, the MS will consider to handover and send the scan request to serving BS. After receiving the scan response, the MS state will transfer to STATE_Scanning_timerM. The MS will set the PHY configuration parameters at STATE_Scanning_PHYSyn according to the target BSs, synchronize with the target BS and send the scan result to the serving BS. When all the targets have been scanned, the state transfers to STATE_MOB_SCN_REP.

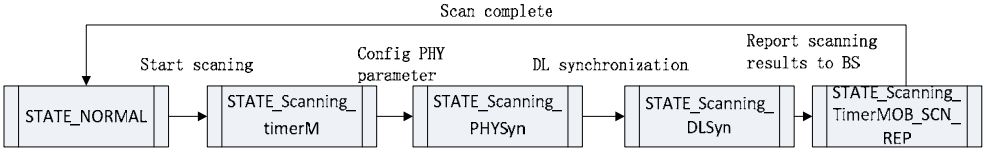


Fig. 15. MS scanning flow chart

Serving BS will consider the factors of signals strength and so on to select the best target BS to process the handover. The MS state will transfer to STATE_HO_Request. The MS will process the downlink synchronization with the target BS at the state of STATE_HO_PHY_SYN and execute the uplink synchronization by ranging mechanism at the state of STATE_HO_RANGING. Once having received the successful ranging response, the process of handover finishes. The MS sets the best target BS as the serving BS and transfers the state to STATE_NORMAL.

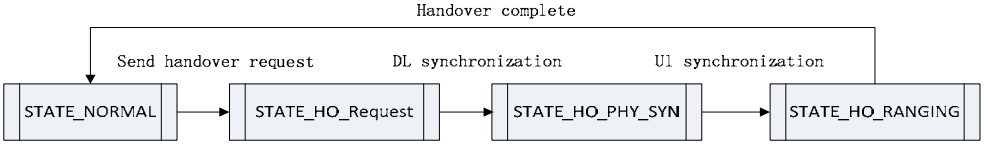


Fig. 16. MS handover flow chart

4.2.2 Multi-threading

Three threads for MAC protocol are designed, which are time-thread, MAC-thread and APP- thread to keep the system running. The functions performed by the three threads are slightly different at BS and MS sides. The following will make a brief introduction.

1. three threads at BS side

Time-thread is mainly used for timing, polling the DMA channels to get the interruption to trigger the next operation. If the interruption type represents data arrival, the MAC layer will get the data from the PHY layer through the API for the further processing. If the interruption type represents frame beginning, then the frame number will increase by 1and the MAC entity will send the packet messages to the PHY entity through the API. Such a send-receive method is aimed to accommodate the requirements of limited time processing. The details can be referred to section 4.3. Because there are many timers in the MAC layer module, the time-thread will also be responsible for the timing and overtime processing.

MAC-thread is mainly responsible for processing the messages according to the current state, sending the API messages to PHY layer or getting the API messages from PHY layer.

APP-thread mainly executes the tasks of converging the uplink traffic data and sending the downlink traffic to the MAC-thread which will deal with PDUs forming. The triggers and relations of the three threads are shown in Fig.17

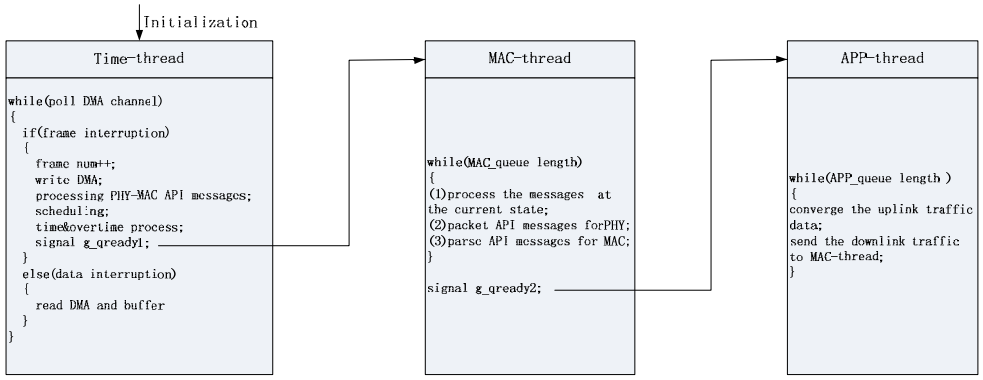


Fig. 17. The triggers and relations of the three threads at BS

2. three threads at MS side

The functions of the three threads at the MS side are similar with BS's. Once the frame interruption arrives, the time-thread will execute the overtime process. And then, the MAC-thread will process the API messages received from PHY layer according to current state. The MAC-thread will also packet the management messages and the uplink traffic data into PDUs that are sent to PHY layer. The APP-thread is mainly used to converge the downlink traffic data and send the uplink traffic data to the MAC-thread. The triggers and relations of the three threads are shown in Fig.18.

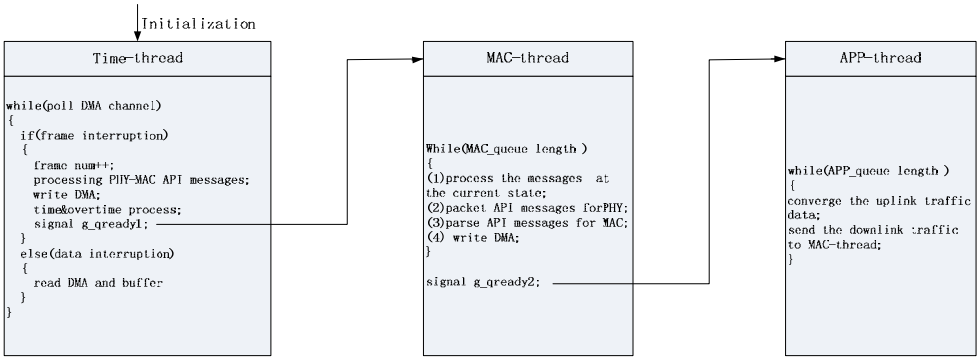


Fig. 18. The triggers and relations of the three threads at MS

4.3 Synchronization between MAC and PHY

The interaction between MAC layer and PHY layer is through API entity. The MAC layer parses the API messages received from the API entity and sends the packet API messages to the API entity. The API messages of txstart.request and rxstart.request at the BS side consist of DL-MAP and UL-MAP which are sent in the downlink subframe. The common place of DL-MAP and UL-MAP is that they all contain the frame number. If the frame number is the same, the DL-MAP indicates the current downlink subframe resource allocation and the UL-MAP indicates the current uplink subframe resource allocation. This is referred as minimum-time relevance. If the frame number in the UL-MAP is larger than that in the DL-MAP by1, the DL-MAP indicates the current downlink subframe resource allocation but the UL-MAP indicates the next uplink subframe resource allocation. This is referred as maximum time relevance. (Zeng, 2006).

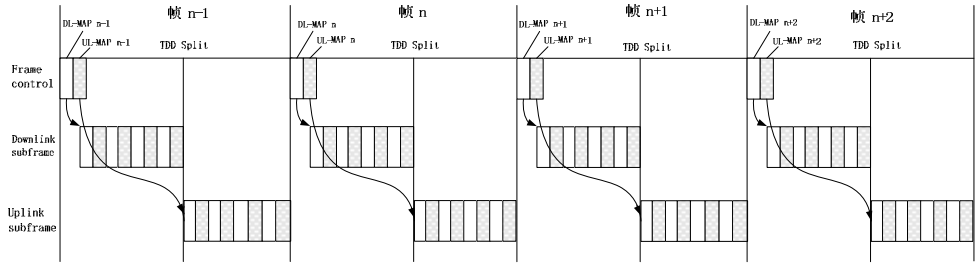


Fig. 19. Minimum time relevance of DL-MAP and UL-MAP

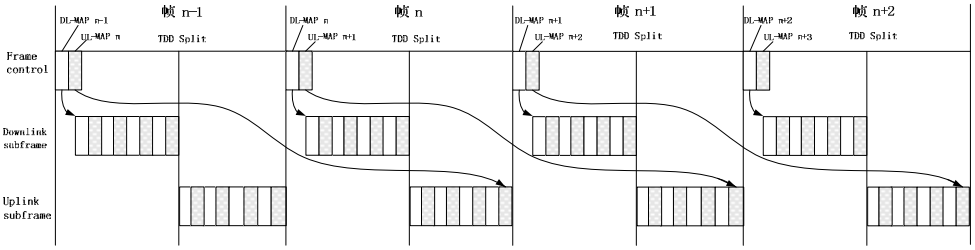


Fig. 20. Maximum time relevance of DL-MAP and UL-MAP

The API messages should be sent to the API entity before the transmission of the air interface. The processes of downlink and uplink transmission will be briefly introduced in the next section.

4.3.1 Downlink transmission

The construction and transmission of a downlink subframe at BS occurs as follows:

1. The BS MAC issues a TXSTART.request which includes the TXVECTOR describing the subframe structure according to the schedule. For a valid request, TXSTART.response returns the frame number(N) which it received from the MAC in the TXVECTOR structure.

2. The BS MAC issues one MACPDU.request for each downlink PHY burst(i.e. each burst described in TXVECTOR). It is not necessary for the BS MAC to wait for TXSTART.response before issuing MACPDU.request. Each MACPDU.request may contain multiple actual MAC PDUs. The MACPDU.request messages must be issued in the burst order specified by the TXVECTOR provided with the TXSTART.request and issued before transmission of the subframe is indicated by TXSTART.indication. An error is returned in MACPDU.response if a PDU is submitted too late.

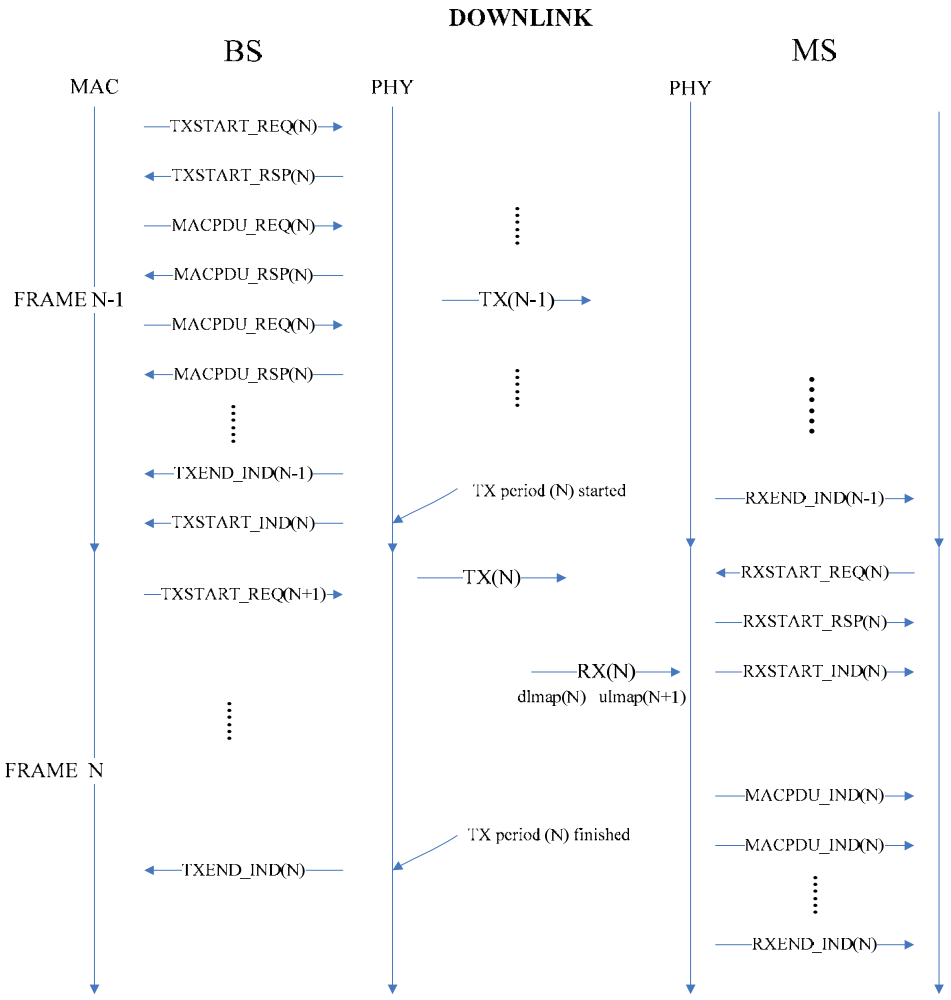


Fig. 21. API primitives for downlink transmission

3. A MACPDU.response is returned for each request indicating whether the PDU has been successfully queued for transmission. The TXEDN.indication message indicates whether the downlink subframe has been successfully transmitted or if transmission was aborted due to an error. If a TXSTART.request is not received then nothing is transmitted. Therefore to ensure continuous transmission, a TXSTART.request must be submitted for each and every downlink subframe (PicoChip Company, 2007).

The reception of a downlink subframe at MS occurs as follows:

1. The RXSTRAT.request message instructs the PHY to start reception of subframe(N). To allow the PHY to parse the DL-MAP the MAC must send a RXSTART.request message which includes the CIDs to decode. When reception of subframe(N) begins the RXSTRAT.indication message is send to MAC.
2. When the MS is registered with the BS and not in idle or sleep mode, it should issue a RXSTRAT.request for every subframe.
3. MAC PDUs received by the BS are transferred via the MACPDU.indication message. Each MACPDU.indication may contain multiple MACPDUs. Each received PDU is associated with the downlink frame number.
4. The first MACPDU.indicatiion for burst#0 will contain the FCH, the second for burst#1 will contain the DL-MAP. However, the PHY will parse the FCH and DL-MAP so it is not necessary for the MAC to send any downlink frame structure information to PHY.
5. The end of decoding for each downlink subframe is signaled via the RXEND.indication message. The downlink bursts for the MS may occur early in a subframe so RXEND.indication can be issued before the end of the downlink subframe (PicoChip Company, 2007).

4.3.2 Uownlink transmission

The construction and transmission of a uplink subframe at MS occurs as follows:

1. The MS MAC issues a TXSTART.request which includes the TXVECTOR describing the uplink subframe structure according to schedule. For a valid request, TXSTART. response returns the frame number (N) which it received from the MAC in the TXVECTOR structure.
2. The MS MAC issues one MACPUD.request for each uplink PHY burst described in TXVECTOR, there should be only one normal uplink burst allocated to the MS per frame. Each MACPDU.request may contain multiple actual MAC PDUs. The MACPDU.request message must be issued before transmission of the subframe is indicated by TXSTART.indicaation.An error is returned in MACPDU.response if a PDU is submitted late.
3. A MACPDU.response is returned for each request indication whether the PDUs have been successfully queued for transmission, The TXEDN.indication message indicates whether the uplink subframe has been successfully transmitted or if transmission was aborted due to an error. If a TXSTRAT.request is not received then nothing is transmitted. The MAC should only issue the TXSTART.request when it needs to transmit uplink data, perform ranging or send information on the ACK channel or fast-feedback channel (PicoChip Company, 2007).

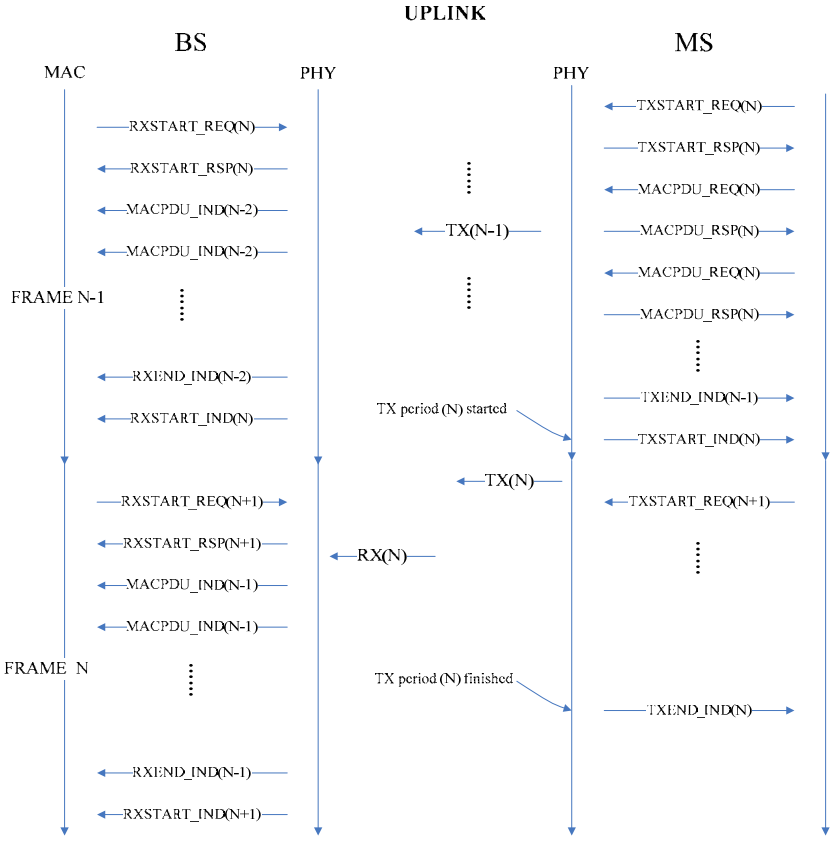


Fig. 22. API primitives for uplink transmission

The construction and transmission of an uplink subframe at BS occurs as follows:

1. The RXSTART.request message provides the RXVECTOR which describes the bursts on the uplink for frame N. The UL-MAP is transmitted at the start of downlink subframe N.
2. The RXSTRAT.request is issued against a particular downlink subframe and must be sent after the associated TXSTART.requset. A successful RXSTRAT.request returns the frame number N which is received from the MAC in the RXVECTOR structure. The RXSTRAT.request must also be issued before the start of downlink subframe N.
3. If a RXSTART.request is not received then the receiver is effectively disabled. Therefore to ensure continuous reception, a RXSTART.request must be submitted for each uplink subframe.
4. MAC PDUs received by the BS are transferred via the MACPDU.indication message. Each MACPDU.indication may contain multiple MAC PDUs. Each received PDU is associated with uplink frame number and the burst number specified in RXVECTOR. The end of decoding for each uplink subframe is signaled via the RXEND.indication(PicoChip Company, 2007).

5. Conclusions

In this chapter, the design and implementation of a WiMAX wireless baseband communication system are presented. Based on the discussion of the typical baseband hardware schemes, we adopt the Picochip multi-core DSP processor as the base of the baseband platform. The hardware, PHY protocol and MAC protocol are introduced in terms of design and implementation other than research aspect, the tradeoff between complexity and performances has been taken into account to meet the requirements. The PHY-MAC interface and API, link-level simulation and debugging method are also mentioned in this chapter, which may provide users better understanding of the development procedure.

6. References

Du, Y. (2010), The MAC layer, In: *IEEE 802.16m Broadband Wireless Technology and System Design*, pp. (181-220), Posts & Telecom Press, Peking, 978-7-115-22754-6.

Dohler, M. Lerau, C. &Hardouin, E. (2005). *UMTS FDD multi-Antenna Receiver Complexity Estimation.* France Telecom R&D, internal report. pp. 158-175.

Fathi, L. (2004). *Complexity of MPIC receiver for HSDPA R5.* France Telecom R&D, internal report. pp. 204-210.

Foschini. G.J.(1996).*Layered Space-Time Architecture for Wireless Communicationin a Fading Environment When Using Multi-Element Antennas* · Bell Labs Technical Journal, 1996, V01.1(2).pp.41-59.

Goldfarb, M. Palmer, W. & Murphy,T. (2000).*Analog baseband IC for use in direct conversion W-CDMA receivers.*Radio Frequency Integrated Circuits (RFIC) Symposium, 2000. Digest of Papers. 2000 IEEE, pp. 79-82.

Jan Mietzner, Robert Schober, Senior Member, Lutz Lampe, Wolfgang H. Gerstacker and Peter A. Hoeher.(2009). *IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 11, NO. 2, SECOND QUARTER 2009, pp.* 87-105.

Jusslia, J. Ryynanen, J. &Kivekas,K. (2001).A 22-ma 3.0-db NF direct conversion receiver for 3g WCDMA.IEEE journal of solid-state Circuits, vol. 36, pp. 2025-2029.

LAN/MAN Standards Committee of the IEEE Computer Society and the IEEE Microwave Theory and Techniques Society.(December 2007). *Part 16: Air interface for Broadband Wireless Access System*, pp. (619-1082), P802.16Rev2/D2.

Parssinen, A. Jussila, J. &Ryynanen, J. (1999).*A 2-GHZ wide-band direct conversion receiver for WCDMA applications.*IEEE Journal of Solid-state Circuits, vol. 34, pp. 1893-1903.

Parssinen, A.Jussila, J.&Ryynanen, J. (2000).*A wide-band direct conversion receiver with on-chip A/D converters.*VLSI Circuits, 2000. Digest of Technical Papers. 2000 Symposium, pp. 32-33.

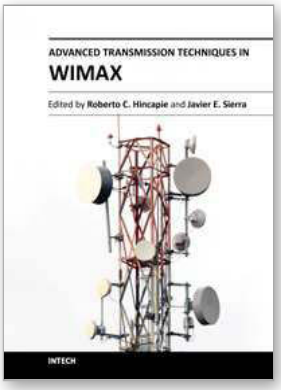
PicoChip Company. (July27,2007). BS MAC-PHY API Definition. pp. 11-13

PicoChip Company. (May21,2007). MS MAC-PHY API Definition. pp. 16-17

PicoChip Company. (September 17,2008).*Tools_Userdoc_Fullman_7.4.5.* pp. 123-138

S. Lin and P. Yu. (July 1982). *A Hybrid ARQ Scheme with Parity Retransmission for Error Control of Satellite Channel , IEEE Trans on Communications.* pp. 1701–1719.

Zeng, C. (2006), The support from MAC layer to PHY layer, In: The Principles and Applications of WIMAX/802.16, pp. (113-115), China Machine Press, Peking, 7-11120111-6.



Advanced Transmission Techniques in WiMAX

Edited by Dr. Roberto Hincapie

ISBN 978-953-307-965-3
Hard cover, 336 pages
Publisher InTech
Published online 18, January, 2012
Published in print edition January, 2012

This book has been prepared to present the state of the art on WiMAX Technology. The focus of the book is the physical layer, and it collects the contributions of many important researchers around the world. So many different works on WiMAX show the great worldwide importance of WiMAX as a wireless broadband access technology. This book is intended for readers interested in the transmission process under WiMAX. All chapters include both theoretical and technical information, which provides an in-depth review of the most recent advances in the field, for engineers and researchers, and other readers interested in WiMAX.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821