Parallel Processing Design for LTE PUSCH Demodulation and Decoding Based on Multi-Core Processor

Zhang Ziran, Li Jun, Li Changxiao
(ZTE Corporation, Shenzhen 518057, P. R. China)

Abstract:
The Long Term Evolution (LTE) system imposes high requirements for dispatching delay. Moreover, very large air interface rate of LTE requires good processing capability for the devices processing the baseband signals. Consequently, the single-core processor cannot meet the requirements of LTE system. This paper analyzes how to use multi-core processors to achieve parallel processing of uplink demodulation and decoding in LTE systems and designs an approach to parallel processing. The test results prove that this approach works quite well.

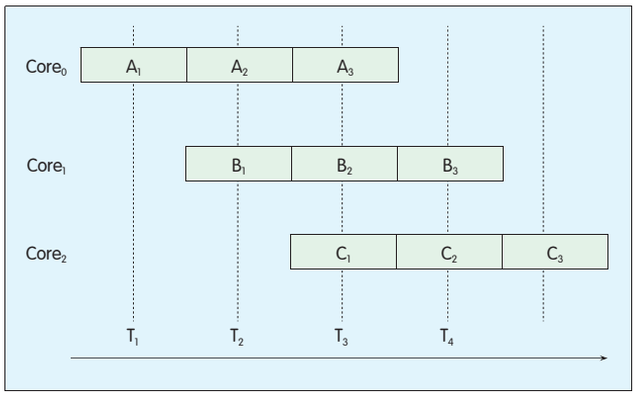
Long Term Evolution (LTE) refers to the long term evolution of 3G systems. According to LTE protocols, the uplink Hybrid Automatic Repeat-Request (HARQ) delay, from the time eNodeB finishes receiving Physical Uplink Shared Channel (PUSCH) subframes to the time the Acknowledge (ACK) or Non-Acknowledge (NACK) message is sent via the downlink, should be no more than 3 ms. In order to verify such delay in case of single-core processor, tests have been conducted under the following conditions: single core processors are used to process demodulation and decoding serially on the uplink, the User Equipment (UE) type is category 5, the Transport Block (TB) size is 75,056 bits, virtual Multiple-Input Multiple-Output (MIMO) is used on the uplink and the rate at the air interface reaches 150 Mb/s. The test results show that the total processing delay cannot meet the requirements specified in the protocol. As a result, multi-core processors are introduced to perform demodulation and decoding parallelly on the uplink so as to shorten the processing time, allowing the delay to meet the minimum requirement of the protocol, i.e. 3 ms.[1-6]

1 Architecture and Principle of Multi-Core Processor

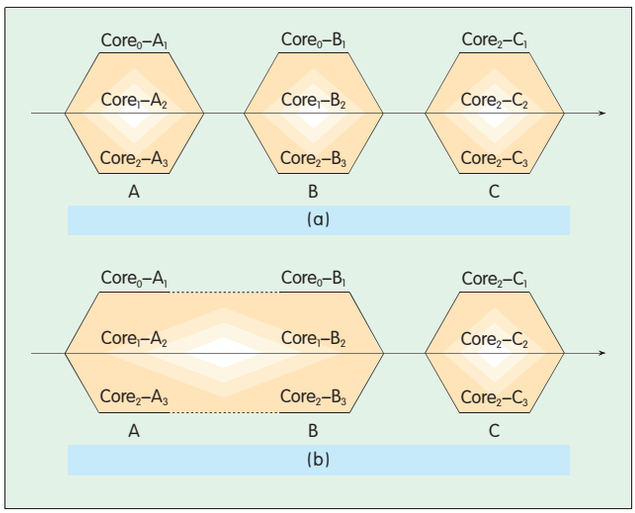
1.1 Architecture

Multi-core processor technology is a new technology recently introduced in Central Processing Unit (CPU) design. It allows two or more processor cores integrated onto one chip to enhance the processing capability of the chip.

In this paper, we take example of three-core processor, i.e. three cores integrated onto one chip. The three cores, called $Core_0$, $Core_1$ and $Core_2$ respectively, share the memory and other peripheral devices, and each of them is configured with a high-speed L2 cache to decrease the bottleneck effect which may occur when the three cores access the memory at the same time. Any shared resource (e.g. a segment of codes or data in Double Data Rate Two (DDR2)) can be shared or excluded via semaphores. All cores run their own Real-Time Operating Systems (RTOSs) and communicate with each other through semaphores. A task on a core can communicate with another task on the same core. Hence, specific applications are realized with this cooperative communication between tasks. By dividing an application into



◀Figure 1.
Assembly line processing of multi-core processor.



◀Figure 2.
Distributed processing of multi-core processor.

several tasks that can run parallelly on different cores, the data can be processed simultaneously, thus the system's processing capability is enhanced.

1.2 Principle of Parallel Processing of Multi-Core Processor

A critical issue in applying multi-core technology is how to convert the serial processing of a single-core processor into the parallel processing of a multi-core processor. Suppose a serial processing task involves three modules A, B and C, and their processing times are T_1, T_2 and T_3 respectively. It can achieve parallel processing with a multi-core processor in one of the following two ways.

1.2.1 Assembly Line Processing

The first way is called assembly line processing, with which the tasks of three modules A, B and C are processed by $Core_0$, $Core_1$ and $Core_2$ respectively.

The assembly line processing goes as follows: First, divide each module into three sub-processes respectively. That is to say, divide module A into A_1, A_2 and A_3, module B into B_1, B_2 and B_3, and module C into C_1, C_2 and C_3. After a sub-process of module A is completed, it will be sent to module B for processing; similarly, after a sub-process of module B is completed, it will be sent to module C for processing, thus all sub-processes are processed in an assembly line way, as shown in Figure 1. The core in the rear does not have to wait for all sub-processes of the core in front of it to complete before it begins processing; instead, after the core in the front completes a sub-process, it gives the sub-process to the one behind it for processing. In this way, parallel processing is realized.

1.2.2 Distributed Processing

In the distributed processing method, each core processes module A, B and C simultaneously. For a UE, if a single-core processor can meet the delay requirement for processing module A, B or C, a three-core processor (i.e. $Core_0$, $Core_1$ and $Core_2$) can be used to process the three modules in a distributed way. As each core processes module A, B and C parallelly, more than one UE can be distributed on different cores for load sharing, thus parallel processing is achieved. The processing is illustrated in Figure 2.

The difference between Figure 2 (a) and (b) is as follows: In 2(a), module B has to wait for processing until all sub-processes of module A are completed; while in 2(b), module A and B can be processed at the same time.

2 Analysis and Design of Parallel Processing of LTE PUSCH Based on Multi-Core Processor

2.1 PUSCH Overview

PUSCH is used to transmit service data. It is shared by multiple UEs and dispatched with Media Access Control (MAC) dispatcher.

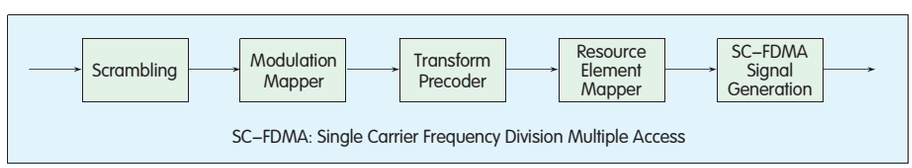
At the UE side, the processing of PUSCH is illustrated in Figure 3.

The demodulation and decoding flowchart of PUSCH is shown in Figure 4.

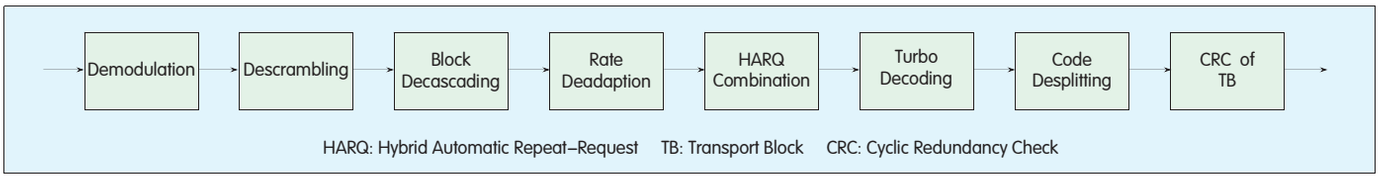
2.2 System Delay Requirement

2.2.1 Delay Requirement for Uplink HARQ in LTE System

The minimum delay requirement for uplink HARQ in LTE systems is 3 ms. The delay refers to the duration when eNodeB begins to process data packets received from Uplink Shared Channel (UL-SCH) and responds with ACK or NACK



▲Figure 3. Processing of PUSCH.



▲Figure 4. Demodulation and decoding flowchart of PUSCH.

▼Table 1. Processing time of single-core processor

Function	Processing Time (ms)
Channel Estimation	0.63
MIMO Decoding	0.94
IDFT	0.60
Demodulation	0.30
Descrambling	0.33
Demultiplexing	0.30
Rate De-Adaption	2.62
CRC of Code Block, Block Connection, CRC of TB	0.2
Total	5.92

MIMO: Multiple-Input Multiple-Output
CRC: Cyclic Redundancy Check

message via the air interface. The processing at eNodeB includes channel estimation, MIMO decoding, Inverse Discrete Fourier Transform (IDFT), demodulation and decoding.

2.2.2 Delay Test Based on Single-Core Processor

The delay test of a single-core processor (CPU frequency: 1 GHz) is conducted under the following conditions: a cell with 20 MHz bandwidth, 1,200 sub-carriers, 4 receiving antennas at eNodeB side, 2 transmitting antennas at UE side, virtual MIMO for uplink receiving, 2 code words, TB size of 75,056 bits, 64 Quadrature Amplitude Modulation (64QAM) scheme, and a peak rate of 150 Mb/s. The test results are listed in Table 1, which show the total processing time of 5.92 ms, more than the given 3 ms. Therefore, the single-core processor cannot meet the system's delay requirement.

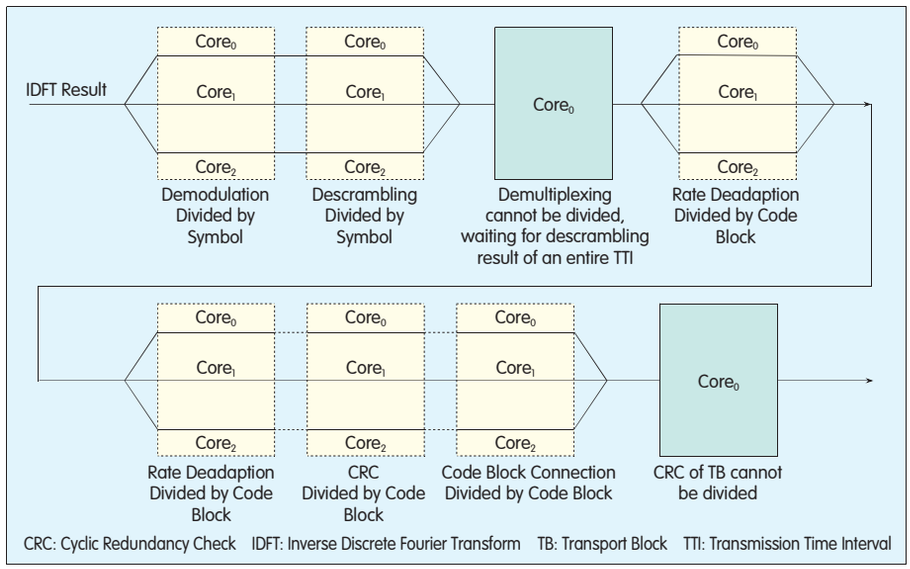
2.3 Feasibility Analysis of Parallel Processing

With a multi-core processor, parallel processing can be achieved. The feasibility of parallel processing is analyzed as follows:

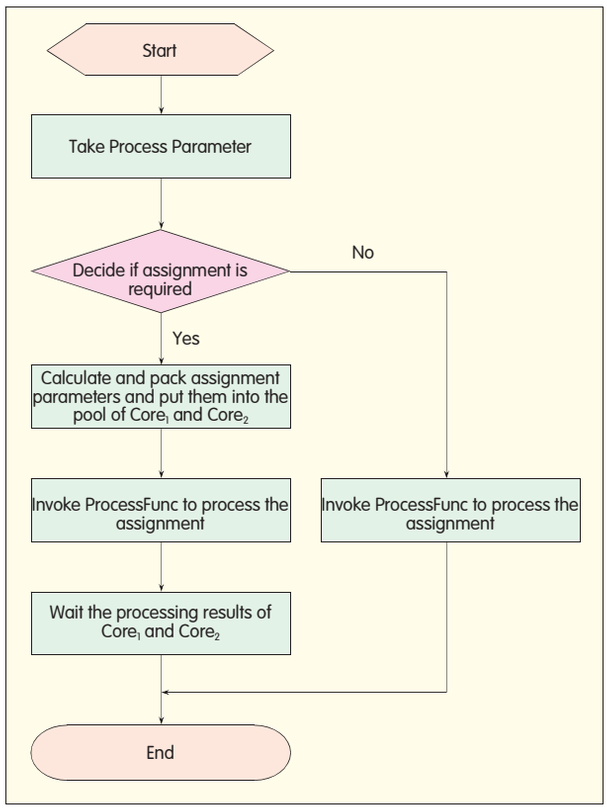
- Demodulation: This process can be divided among multiple cores based on modulation symbols and every core needs to generate the entire scrambling sequence. Therefore, the cores can start demodulation before getting all modulation symbols.
- Descrambling: This process can be divided among multiple cores by soft bit. Hence, it can start before all soft bits are given. Each core must generate the entire scrambling sequence.
- Control and data demultiplexing: This process can only be divided among multiple cores by the user. Various users' demultiplexing tasks can be processed by different cores. As it is quite complicate to divide the demultiplexing task of a user among multiple cores and deinterleaving of different Single Carrier Frequency Division Multiple Access (SC-FDMA) symbols within a Transmission Time Interval (TTI) is involved, the demultiplexing process cannot begin until all symbols of a TTI are collected.
- Rate de-adaption: This process can be divided among multiple cores by code block. That is to say, the code blocks of different users can be processed by different cores, and the code blocks of a user can also be processed by different cores.
- CRC of code block: This process can be divided among multiple cores by code block. That is to say, the code blocks of different users can be processed by different cores, and the code blocks of a user can also be processed by different cores. Once a code block rather than all code blocks is decoded, the CRC check can be performed for the decoded block.
- Code block connection: This process can only be divided among multiple cores by the user. The task of one user can only be processed by one core. Once a code block rather than all code blocks is decoded, this process can be performed for the decoded block.
- CRC of TB: This process can only be divided among multiple cores by the user. The task of one user can only be processed by one core. It can start only after all TBs are decoded.

2.4 Design of Parallel Processing Based on Multi-Core Processor

The parallel processing of demodulation



▲Figure 5. Parallel processing of demodulation and decoding.



◀Figure 6. Processing flow of dispatching core.

and decoding is illustrated in Figure 5.

In Figure 5, $Core_0$ acts as the dispatching core, and $Core_1$ and $Core_2$ are non-dispatching cores. The principles for dispatching a dispatching core are as follows: If the processing resources required by new users can be provided by a single core, the processes of all the new users will be shared among all cores by UE; otherwise, the process of the user occupying the most resources will be broken down into sub-processes for parallel processing.

2.4.1 Processing of Dispatching Core

In the dispatching core, each process function (e.g. ProcessFunc) is invoked by the function ProcessDispatch. The flow of ProcessDispatch function of $Core_0$ is illustrated in Figure 6.

$Core_0$ first dispatches a process. If it can complete the process by itself, it directly invokes the ProcessFunc function to do the processing. If this process has to be assigned to other cores, $Core_0$ packs the parameters and puts them in the parameter pools of $Core_1$ and $Core_2$, notifying $Core_1$ and $Core_2$ of processing. Meanwhile $Core_0$ invokes ProcessFunc to process the sub-process assigned to it. Finally, $Core_0$ waits and collects the processed data from $Core_1$ and $Core_2$ to do the final processing.

2.4.2 Processing of Non-Dispatching Core

Unlike the dispatching core, the non-dispatching core has a high priority task DispatchTsk, which is specially used for processing the sub-process assigned by $Core_0$ and triggered by the dispatching core $Core_0$. DispatchTsk takes input parameters of ProcessFunc from the parameter pool, and then invokes ProcessFunc to process the assigned sub-process. After a non-dispatching core completes a sub-process, it labels the sub-process with a completion tag and notifies the dispatching core. To reduce the waiting time of the dispatching core, the priority of DispatchTsk is set to be higher than the service processing tasks of the non-dispatching core itself, enabling DispatchTsk to be responded timely. When DispatchTsk does not take control over a none-dispatching core, the core will process the tasks of its own. Figure 7 shows such a processing flow.

3 Result Analysis

To evaluate the performance of parallel

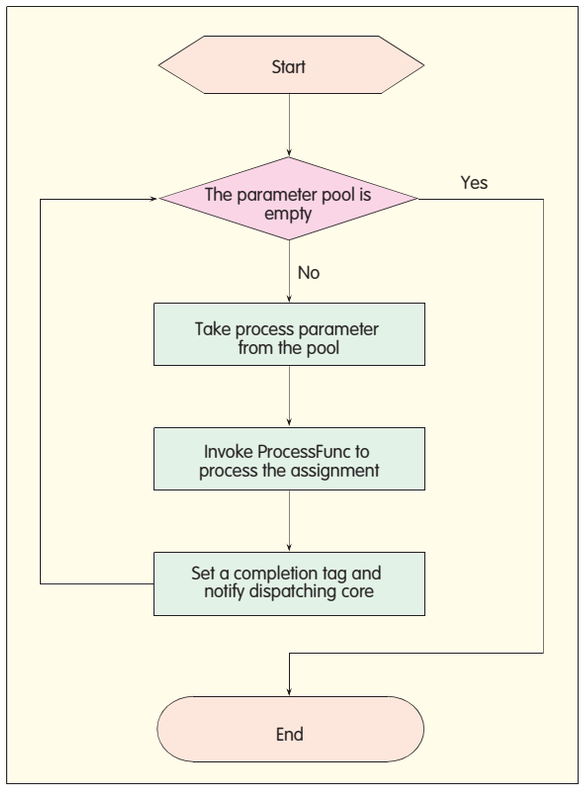


Figure 7.▶ Processing flow of non-dispatching core.

▼Table 2. Serial processing vs. parallel processing in rate de-adaption

Transport Block Size (bit)	Quantity of Code Blocks	Processing Time of Single-Core Processor (Cycles)	Processing Time of 3-Core Processor (Cycles)	Efficiency Ratio
75,056	13	2,600,447	1,001,607	2.6
48,944	9	1,867,429	625,102	2.98
24,472	5	905,015	544,364	1.67
12,236	3	395,125	133,879	2.95

processing, we have conducted related tests and measured the processing times of rate de-adaption when a single-core processor (for serial processing) and a multi-core processor (for parallel processing) are used respectively. The test results are shown in Table 2.

The data in Table 2 indicate parallel processing with a three-core processor is over 2.6 times faster than serial processing with a single-core processor in most cases. Therefore, parallel processing with multi-core processor can greatly shorten the processing time of a LTE system during uplink demodulation and decoding. This no doubt offers a new approach for wireless communication system design.

4 Conclusion

The future communication systems require much higher peak rate for the air interface but very short processing delay. One critical issue of communication systems is how to improve the processing speed and capability and decrease the processing delay[7-12]. This paper analyzes parallel processing and suggests an approach to use multi-core processors to process uplink demodulation and decoding of LTE systems in parallel. The test results demonstrate that this approach does work quite well.

References

[1] 3GPP TSG RAN1#50bis .Timing and HARQ [S]. 2007.
[2] 3GPP TS36.212 v8.1.0. Multiplexing and Channel Coding [S]. 2007.
[3] 3GPP TS36.211 v8.1.0. Physical Channels and Modulation [S]. 2007.
[4] 3GPP TS36.201 v8.1.0. Physical Layer: General Description [S]. 2007.
[5] 3GPP TS36.204 v8.0.0. Base Station (BS) Radio Transmission and Reception [S]. 2007.
[6] 3GPP TS36.306 v8.0.0. User Equipment (UE) Radio Access Capabilities [S]. 2007.
[7] 佟学俭, 同涛. OFDM移动通信技术原理与应用 [M]. 北京: 人民邮电出版社, 2000.
[8] 曹志刚, 钱亚生. 现代通信原理 [M]. 北京: 清华大学出版社, 1992.
[9] TANENBAUM A S. 现代操作系统 [M]. 陈向群, 等译. 北京: 机械工业出版社, 1999.
[10] HOLMA H, TOSKALA A. WCDMA for UMTS-HSPA evolution and LTE [M]. 4th ed. New York, NY, USA: John Wiley & Sons ltd, 2007.
[11] 王念旭. DSP 基础与应用系统设计 [M]. 北京: 北京航天航空大学出版社, 2001.
[12] 申敏, 郑建宏, 刘栋. DSP原理及其在移动通信中的应用 [M]. 北京: 人民邮电出版社, 2001.

Biographies

Zhang Ziran

Zhang Ziran graduated from University of Electronic Science and Technology of China. He is now a senior engineer in ZTE Corporation, mainly engaged in research of frontier technologies of such wireless communication systems as UMTS and LTE. He has published 5 papers and applied for 2 patents.

Li Jun

Li Jun graduated from Huazhong University of Science and Technology. He is now a senior engineer in ZTE Corporation, mainly engaged in research of frontier technologies of such wireless communication systems as UMTS and LTE.

Li Changxiao

Li Changxiao graduated from University of Electronic Science and Technology of China. He is now an engineer in ZTE Corporation, mainly engaged in research of frontier technologies of such wireless communication systems as UMTS and LTE.

Roundup

ZTE Rolls Out New Unified Open Environment Platform

ZTE Corporation (ZTE), a leading global provider of telecommunications equipment and network solutions, on February 12, 2009, released a unique Unified Open Environment (UOE) platform that will make it easier for telecom operators to develop and offer new mobile services and stay competitive in their respective markets. The ZTE UOE service platform is aimed at addressing future global demand for network expansion and application development.

The new ZTE UOE network solution comprises an open module service capability and service development environment, two of its key features that allow modular deployment of different service networks. With an open module service capability, it provides telecom operators a unified management platform for integrated service management operation and monitoring, and unified content management. UOE defines a common API for several international organizations, including Parlay/OSA and JCP, two technical industry consortiums that specify APIs for the telephone network. UOE's service development environment offers an integrated development environment and an open service interface, providing external IT developers and telecom operators a wide range of development tools to meet their specific business and individual requirements.

With the extraordinary features of UOE platform, telecom operators can effectively implement in-house developed network services in a converged infrastructure, as well as partner with equipment vendors to develop unique value added services, such as mobile newspaper, mobile stock exchange, instant mobile news, video conferencing, voice mail and BlackBerry applications.