A DSP Software Architecture of Multiplexing and Channel Coding (MCC) Process for WCDMA Mobile System

Chen Chien-Yu
BENQ Mobile System Inc.
No 23 Li-Hsin Rd., Science-Based Industrial Park, HsinChiu, Taiwan, R.O.C.
Tel: +886-3-6118800#6581 Fax: +886-3-6118877
jerrycychen@benqms.com
Key Words: DSP architecture, WCDMA, Multiplexing and Channel Coding

Abstract

This paper proposes a software architecture of the channel coding and multiplexing processing for 3rd-generation WCDMA mobile system. With the properties of low internal memory usage and fixed task arrangement, this architecture is suitable for DSP implementation. It can also be shown that processing data from multi-user won't increase the task number and internal memory. Finally, this architecture has been realized in the TI's TMS320C6201 DSP and the performance simulation and memory estimation of different data rate is given.

1 Introduction

In the 3rd-generation WCDMA spec, the MCC (multiplexing and channel coding) functional block defines the encoding/decoding procedure of data stream from/to MAC layer to offer transport service over the radio transmission link. The MCC scheme combining the error detection, error correction, interleaving, rate matching and transport channel mapping from/to physical channels is defined in the 3G TS 25.212[1]. Figure1 shows the MCC procedure for uplink data, while figure2 shows the downlink part procedure. The following paragraph describes these sub modules of the MCC structure.

CRC attachment

The error detection in 3GPP spec is provided by CRC attachment in the tail of each transport block. The CRC is 24, 16, 12, 8 or 0 bits, which is decided in higher layer.

Transport block concatenation and code block segmentation

After the CRC attachment, these transport blocks are concatenated into one single block and preparing for channel coding. However, the channel coding size has its limitation. Block size larger than this size should be separated into smaller segments for further processing.

Channel coding

The error correction scheme provided in 3GPP spec contains the convolutional coding and turbo coding. The convolutional code has constraint length 9 and coding rate 1/3 and 1/2. The turbo code in 3GPP spec is the parallel concatenated convolutional code each with constraint length 3 (8-states) combining with a turbo code interleaver with rate 1/3.

DTX insertion

Since the downlink channel has the predefined spreading factor, the DTX bits are inserted if transport format with smaller transport channel size is used.

Interleaving

The interleaving prevents the channel coded data from the burst error in wireless channel. The 1st interleaving is performed before the transport channel multiplexing, and the 2nd interleaving is performed after the transport channel multiplexing.

Rate matching

The rate matching process repeats or punctures the transport channel so that they can be fit into the physical channel size, which denoted by the spreading factor of the

Transport channel multiplexing

The transport channel multiplexing multiplexes different transport channel into a coded composite transport channel (CCTrCH).

2 DSP implementation issue

To put these operations to the DSP, we have to consider some limitations of the DSP: the CPU cycle of the MCC operation and internal memory usage for each user. Table1 lists the required operation cycle of each MCC operation for TMSC6201 DSP, note that the process not listed here means the CPU cycles of this operation is not taken into consideration.

Besides, the internal memory usage is also an important issue of DSP implementation. Since the C6201 only has a limit internal data memory of 64kbytes, it's important to arrange the task and use the internal memory. Table2 shows the memory required by the MCC operations.

From table1 and table2, we can find that the DSP can process up to 17 users for 12.2k case, but the memory size limits the supported user number. Thus the memory and task arrangement for the MCC processing can optimize the DSP utilization for multi-user requirement.

3. The purposed architecture

To optimize the memory utilization, we propose architecture of MCC operation. Instead of using a single task processing MCC operation for each user, it separates the MCC operations to different tasks. Each task performs an MCC operation for all users. In the downlink side, there are 5 tasks: CRC encoding, channel encoding, rate matching, transport channel multiplexing and 2nd interleaving. In the uplink side, there are also 5 tasks: CRC decoding, channel decoding, de-rate matching, transport channel de-multiplexing and 2nd de-interleaving. These tasks are described below:

tEnCRC

This task is invoked by tDLDataIn when there are data from higher layer. It performs CRC eocoding operation for every transport channel. The data are CRC-padded for every transport block. The crc-padded block is then combined and moved to external memory through DMA for further channel coding.

tEnCC

This task is invoked by tEnCRC. It performs channel encoding operation for every transport channel. The transport block is combined by the tEnCRC task and tEnCC gets the data from external memory. It separates the data into code blocks, which are limited by the channel encoder size limitation. The channel coding operation is

performed every code block. The channel encoded data will be combined into one streams and moved to external memory for rate matching.

tEnRM

This task is invoked by tEnCC. It performs downlink rate matching operation for every transport channel. Downlink rate matching matches the size of physical channel for largest size of all transport format for each transport channel (fix position) or largest size of all transport format combination (flexible position). After rate matching, the 1st DTX bit insertion and 1st interleaving is performed. Then the transport channel data can be divided into 'TTI' parts and are ready for transport channel multiplexing. Every part presents data which will be transferred in 10 ms. These data will be combined with other transport channels to be one physical channel.

tTrchMux

This task is also invoked by its previous task - tEnRM. Before transport channel muxtiplexing, the data is processed in the unit of one transport channel. This task multiplexes transport channel that should be combined into one physical channel. After transport channel multiplexing, the 2nd DTX insertion is performed.

t2ndil

This task is invoked by the frame ISR every frame. It receives the data from tTrchMux in the external memory and finds the physical channel data for next frame. If there exists the data for next frame, 2nd interleaving operation will be performed. After this operation, this task invokes the output task called tDlDataOut and output for further processing.

tDe2ndil

The same as t2ndil, this task is invoked by the frame ISR every frame. It receives the physical channel data form tULDataIn task and performs the 2nd de-interleaving operation. The data will be moved to external memory after the process.

tDeRM

This task performs the de-rate matching operation for the physical channel data every frame. After the rate matching, these data will be separated to every transport channel per frame. These segments will also be moved to external memory and tTrchMux will be invoked to process these data.

tTrchDemux

The task collects data of 'TTI' frames from the tDeRM task. The de-multiplexing process combines the data of 'TTI' frames into one transport channel. After the combining the 1st de-interleaving is also performed. The data are then moved to external memory for channel decoding.

tDeCC

The channel decoding is performed every transport channel and is invoked by the tTrchDemux task. After channel decoding, the code blocks are then combined together for CRC decoding process. Since this operation is most DSP cycle consuming, it is possible to use an ASIC to accelerate the speed. Then this task can call the relative function call to activate the ASIC for the channel decoding processing.

tDeCRC

The last task of the MCC decoding process is the De-CRC operation. This task is also invoked by the it previous task - tDeCC. It performs CRC decoding for each transport block and checks if there are CRC errors being reported to higher layer.

From the above description, we can find that each task processes the MCC data from its previous

task, and passes the data to next task. Since the internal memory may not be enough for all the users, the data will be put to external memory after the task finishing its operation. The next task has to move the data from external memory to internal memory and processes data. Since the data moving between external memory and internal memory is through the DMA channel, it doesn't increase much load. In this mechanism, the data passing between the tasks are the coding information of the data, i.e., the data length, coding type, parameters, … etc, and the real data are passing through internal memory and external memory. Thus there is always one user's data keep in internal memory, no matter how many users the DSP processing now at the same time. Figure3 and figure4 show the downlink and uplink tasks of this architecture.

The MCC operations described in the prior section are distributed into the 10 tasks. Considering the worst case of 12.2k the case, if all the tasks are processing in parallel, the data memory requirement is the same as the value shown in the total cost of table2, i.e., 18912, no matter how many users it supported. Hence the DSP has a good utilization instead of the internal memory size limitation. On the other hand, if we consider the 64k case, since the DSP can support only one user, it is impossible for these tasks to process in parallel. Thus the maximum memory usage is not the sum of the memory of all these tasks, but the maximum data requirement of a single task. In this way the internal memory requirement can be reduced from 57324 bytes to 15600 bytes.

Beside of the data heap, the stack of each task also uses the internal memory. The stack required by each task depends on the program itself. Since the tasks are statically allocated before compiling stage, the stack size is fixed and it doesn't increase according to the number of supported users.

Conclusion

This paper proposes a DSP architecture for MCC processing. This architecture has been implemented on the Innovative Integration's M62 DSP board. With careful resource arrangement, a single 64kbps or multiple 12.2kbps MCC processing can be achieved. With other channel decoding ASIC's help, more radio link and higher bit rate service can be supported with the same architecture.

References

[1] 3G TS 25.212 V4.4.0, "3rd generation partnership project; Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD)"

[2] 3G TS 25.141 V3.9.0, "3rd generation partnership project; Technical Specification Group Radio Access Network; Basestation conformance testing (FDD)"

[3] Henry Yiu, "Implementing V.32bis Viterbi Decoder on the TMS320C62xx DSP", *Application Report SPRA444*, Customer Applications Center, Texas Instruments Hong Kong Ltd.

[4] D.E. Cress, and W.J. Ebel, "Turbo Code Implementation Issues for Low Latency, Low Power Applications", *1998 Symposium on Wireless Personal Communications*, MPRG, Virginia Tech, June 10-12, 1998.

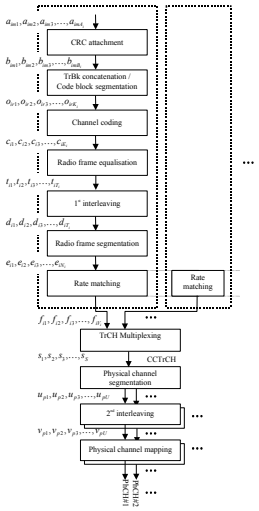

Fig1: The DL MCC procedure

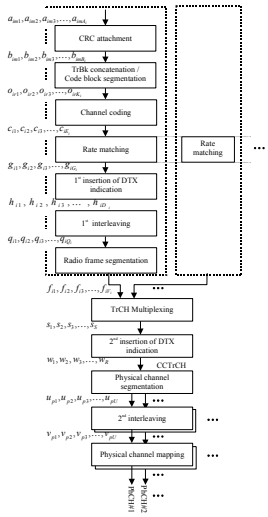

Fig2: The UL MCC procedure

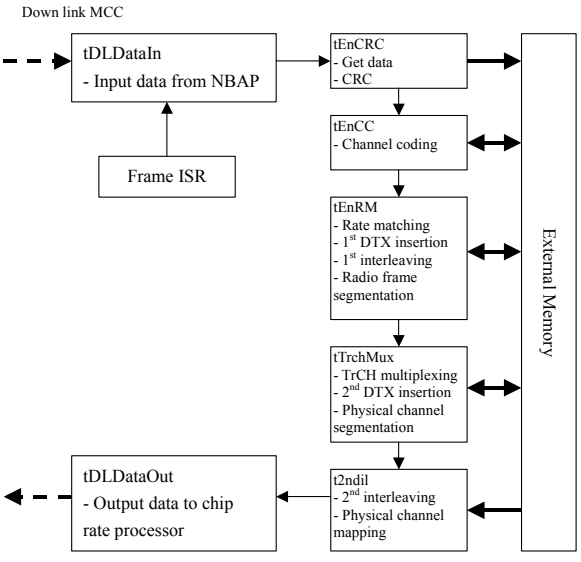

Fig3: Proposed downlink MCC architecture

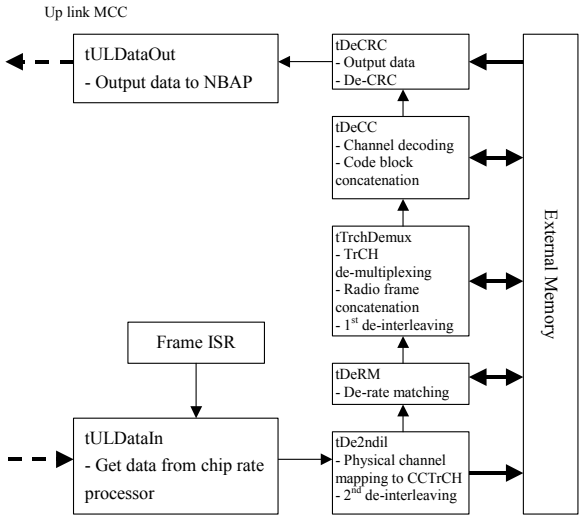

Fig4: Proposed downlink MCC architecture

MCC operations	Cycles/bit	# of bit/sec for 12.2k	Cost cycles/sec for 12.2k	# of bit/sec for 64k	Cost cycles/sec for 64k
CRC/De-CRC	1	12200 * 2	24400	64000 * 2	128000
Convolutional encoding	8	13000	104000	0	0
Turbo encoding	15	0	0	64800	972000
1st interleaving / 1st de-interleaving	1	40200 * 2	80400	195000 * 2	390000
Rate matching / de-rate matching	6.5	40200 * 2	522600	195000 * 2	2535000
2nd interleaving / 2nd de-interleaving	1	60000 * 2	120000	240000 * 2	480000
Viterbi decoding	800	13000	10400000	0	0
Turbo-decoding	1850	0	0	64800	119880000
Total cost			11251400		124385000
Support user (C6201@200MHz)			17. 87		1.61

Table1: the operation cycle estimation of the MCC processing for 12.2k and 64k case

MCC operations	Data required for 12k	Data required for 64k
CRC/De-CRC	1008	5152
Convolutional encoding	1064	0
Turbo encoding	0	7788
Rate matching / de-rate matching	3216	15600
TrCH Mux / TrCH De-mux	1608	8400
2nd interleaving / 2nd de-interleaving	1608	8400
Viterbi decoding	10408	0
Turbo-decoding	0	11984
Total cost	18912	57324
Support user (64 kbyte)	3.3	1.1

Table2: the internal memory estimation of the MCC processing for 12.2k and 64k case