DUAL-MODE CONVOLUTIONAL/SOVA BASED TURBO CODE DECODER VLSI DESIGN FOR WIRELESS COMMUNICATION SYSTEMS

Pen-Hsin Chen, Kai-Huang, Nai-Hsuan Hsueh, and An-Yeu (Andy) Wu

Graduate Institute of Electronics Engineering, and
Dept. of Electrical Engineering,
National Taiwan University, Taipei, 106, Taiwan, R.O.C.

Abstract—**A prototype design of a dual-mode convolutional/turbo code decoder for 3rd generation wireless communication systems is proposed. By merging some similar modules exist in the convolutional code and turbo code decoder, we build one dual-mode decoder with these two functions. Besides, in order to conform to the CDMA2000 standard, the architecture can also perform as a reconfigurable Viterbi decoder. It means that our decoder meets the multi code rate and multi generator polynomial convolutional code specification. The final prototyping chip is presented with Avant! *0.35* standard cell library.**

I. INTRODUCTION

With the coming era of 2.5G/3G high-speed wireless communication systems, multi standard and multi mode in a single communication system standard has become a worldwide trend for providing a variety of service. Take *Code Division Multiple Access 2000* (*CDMA2000*) [1] system for example, there are several channels with different FEC, convolutional or turbo code [2] specifications listed in Table 1. The first column of the table shows different channel types such as power control channel, data transmission channel, *etc.* Then the FEC module and code rate used for the channel are listed in the second and third column, respectively. To implement such 3G FEC standards, most intuitional way is to prepare different sets of hardware for different specifications. However, the area and gate count will be huge. The more economical way is to merge similar sub-modules into one module, and reuse them in different specifications.

Table 1. Different FEC specifications in CDMA2000 system.

Channel Type	FEC	Code Rate
Access Channel	Convolutional	1/3
Enhanced Access Channel	Convolutional	1/4
Reverse Common Control Channel	Convolutional	1/4
Reverse Dedicated Control Channel	Convolutional	1/4
Reverse Fundamental Channel	Convolutional	1/2, 1/3, 1/4
Reverse Supplemental Code Channel	Convolutional	1/2, 1/3
Reverse Supplemental Channel	Convolutional /Turbo Code	1/2, 1/4

This work is supported by the MediaTek Inc., under NTU-MTK wireless research project.

From Table 1, we know that the convolutional code and turbo code are both adopted as the channel coding scheme in the CDMA2000 system. Therefore, our goal is to implement the circuit according to the CDMA2000 standard. Following lists two main goals to achieve:

- Dual-mode convolutional/turbo code decoder: In the CDMA2000 system, there are two kinds of FEC modules, convolutional code and turbo code. We intend to design a dual-mode decoder that can perform the both two FEC functions according to the control signals.
- Reconfigurable Viterbi decoder for multi-spec convolutional code: Even in the same type of FEC modules, multiple specifications such as multi-code rate, different generation functions are usually adopted. Like CDMA2000, the convolutional code has three kinds of different code rate (1/2,1/3,1/4). Hence, we design *BMU to ACS Routing Generator* (*BARG*) to meet the dual-mode and multi-code rate specifications.

II. CONVOLUTIONAL CODE AND VITERBI ALGORITHM

Convolutional code is a widely used FEC module nowadays. Usually, we use Viterbi algorithm to decode the convolutional code because of the performance and hardware concern. In a convolutional encoder, an output bit at any time depends on not only the input bit at that moment, but also *m* previous input bits. Normally, the encoding process of the convolutional code is much simpler than block code, but decoding process is more complicated. Generally speaking, the three major processing units in the Viterbi decoder are shown in Fig 1.

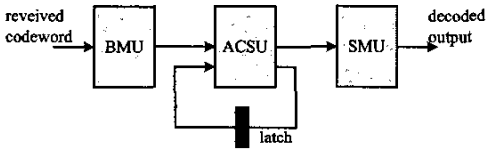


Fig 1. The architecture of the general Viterbi decoder.

The three processing units are:

- *Branch Metric Unit* (*BMU*): Compute branch metrics from the received symbols.
- *Add-Compare-Select Unit* (*ACSU*): The function of this mod-

module is as its name, add, compare, and then select; it operates for each state to update their path metrics, respectively.

- *Survivor Memory management Unit* (*SMU*): Record all the decision results from the ACSU and back trace the survivor path to find the oldest data as decoded bits. Both of these actions should be taken in place simultaneously, so some properly designed pipeline strategies should be applied.

III. TURBO CODE

Turbo code was proposed in 1993 and is well known for its performance. Based on the knowledge in the previous section, we will next explain how a turbo code operates. Then we extend the Viterbi algorithm to the *Soft Output Viterbi Algorithm* (*SOVA*) [3] *that* is used as the component decoder in turbo code system.

A. Turbo code encoder

Fig 2 shows a turbo code encoder, which is composed of two parallel *Recursive Systematic Convolutional* (*RSC*) encoders and one interleaver. The RSC encoder encodes the input sequence and the interleaver permutes the input sequence. The difference between the RSC encoder and the traditional convolutional encoder is that the RSC encoder has a recursive feedback loop and one of the outputs X_n is the same as the input d_k, called *"systematic bit"*. Although the two RSC encoders have the same architecture, the encoding outputs are different respective because the input sequence of the second encoder is scrambled by the interleaver. By doing this, the burst errors occurred during the transmission can be eliminated in one of the encoding output, thus improves the decoding performance.

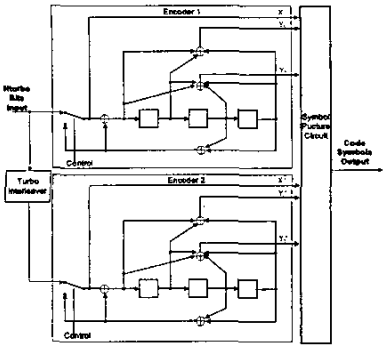


Fig 2. Turbo code encoder

B. Turbo code decoder

The general architecture of a turbo code decoder is shown in Fig 3. The whole decoder is composed of two component decoders, interleaver and de-interleaver. The component decoders decode the sequence encoded by the RSC encoders in Fig 2 relatively, and the interleaver/de-interleaver also have the same algorithm as the interleaver in the turbo code encoder. When the decoding process starts, the input data X and Y_{1k} are put into the component decoder one. After decoder 1 finishing the decoding process, the results will be sent out to the interleaver and the data will be permuted. Then, the permuted data is sent to the component decoder two together with Y_{2k}. After decoding process is done, the data will be put into the de-interleaver to recover the order of data. By repeating the above actions, we achieve the iteration decoding process, and the soft output decisions are more and more reliable as the iteration number increases. Finally, we get the hard decisions by a slicer or other simple modules [4].

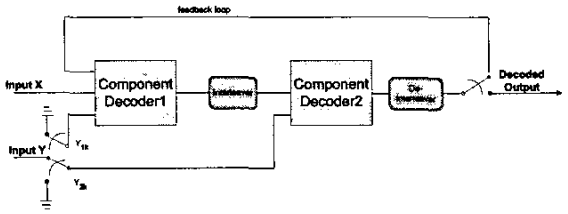


Fig 3. Turbo code decoder.

But from the previous sections, we know that the decoding output of a Viterbi decoder is only the hard decision symbols and can not be used in such iterative decoding scheme. Hence, we need to add some extra reliability information to the output of the Viterbi decoder. SOVA decoder is such a decoder that meets the requirement and can be used in a turbo code system.

C. SOVA decoder

In a Viterbi decoder, we choose the survivor path from the two paths leading to the same state and terminate the other competing path. This operation is executed by ACSU, and the decision results are saved in SMU to keep track of the decision history. After processing a specific amount of input data, we can find the maximum likelihood path and obtain the output sequence according to the data saved in SMU. Similar to the notation in, we can write the probabilities of the competing path at state s as

$$Pr\{path\ (m), state\ (s)\} \sim e^{-\Gamma^{(m,s)}}, \qquad (1)$$

where $\Gamma^{(m,s)}$ is the summation metric of the mth path that ends at state s. For a code with code rate *1/2*, the decision is taken between two competing path $\Gamma^{(1,s)}$ and $\Gamma^{(2,s)}$ which leading to the same state s. We assume that path one is selected as the survivor path for state s. Then, the probability of selecting the wrong path as the survivor path is

$$P_{s,k} = \frac{e^{-\Gamma^{(2,s)}}}{e^{-\Gamma^{(1,s)}} + e^{-\Gamma^{(2,s)}}} = \frac{1}{1+e^{\Delta_s}} \le \frac{1}{2}, \quad \text{where}$$

$$\Delta_s = \Gamma^{(2,s)} - \Gamma^{(1,s)} \ge 0. \qquad (2)$$

Hence, we can update the total probability $\hat{P}_{s,k-1}$ of the symbol in Viterbi algorithm to be

$$\hat{P}_{s,k-j} \leftarrow \hat{P}_{s,k-j}(1-P_{r,k}) + (1-\hat{P}_{s,k-j})P_{r,k}. \quad (3)$$

In practice, the update can be performed directly on the likelihood ratio $\hat{L}_{s,k-j}$ instead of probabilities for easier calculation,

$$\hat{L}_{s,k-j} \leftarrow Min(\hat{L}_{s,k-j}, \frac{\Delta_S}{\alpha}), \quad \hat{P}_{s,k-j} = \frac{1}{\alpha} log \frac{1-\hat{P}_{s,k-j}}{\hat{P}_{s,k-j}}, \quad (4)$$

where α is related to the encoder, modulation, SNR, and the equation above will be used to perform the update operations. We substitute the notation Δ_S / α with Δ_S below. The update operation is required for all of the states along the survivor path. Moreover, the update length should be at least two times of the constraint length [4].

IV. Dual - Mode Convolutional/Turbo Code Decoder

In this section, a methodology of designing the dual-mode SOVA-based decoder will be presented. We will show the overall architecture and describe the theorems of our design, which can perform two different functions but does not occupy double area. Finally, we discuss some design issues of the dual-mode decoder.

A. Overall architecture

Fig 4 shows the overall architecture of our dual-mode Viterbi/turbo code decoder. There are six processing modules: *Shared BMU, Radix-4 ACS, BMU to ACS Routing Generator* (*BARG*), *Interleavr and Deinterleaver, Update Unit* (*UU*), and *Control Unit*. And there are two memory modules: *Path Metric Memory* and *Survivor, Update, Interleavr/Deinterleaver Memory*. The system will choose the turbo code decoder output or the Viterbi decoder output according to the signal from the control unit.

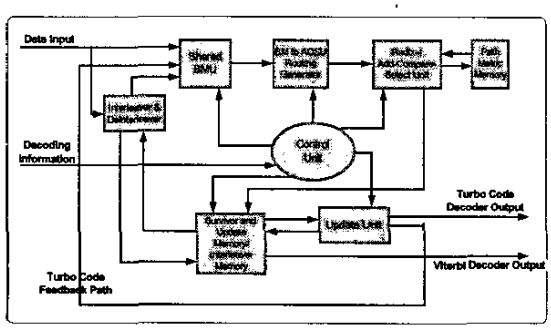


Fig 4. Overall architecture of the dual-mode decoder.

When this dual-mode decoder operates at Viterbi decoding mode, at initial stage, control unit will send control signals to turn off some modules belong to the turbo code mode, like the Interleaver and Deinterleaver module and the Update Unit module. Besides, the Survivor, Update, and Interleavr/Deinterleaver Memory part is totally configurated for survivor memory unit. Then, the remaining modules compose a complete Viterbi decoder and can move to the standard Viterbi decoding phase. After decoding the received signal, the decoded data is send out by the Viterbi decoder output path.

When the dual-mode decoder operates in the turbo decoding mode, all sub-modules are enabled by control unit, and the Survivor, Update, and Interleavr/ Deinterleaver Memory part is now shared by Update Unit module and Interleaver/Deinterleaver module. After decoding the received signal, the decoded data goes back to the BMU module by the Turbo Code Feedback Path for the next iteration until all decoding iterations are finished. Then, the decoded data is driven into the next stage by the Turbo code decoder output path.

B. BMU sharing methodology

The function of the BMU module is to compute the distance between the received signal and the ideally transmitted signals. For the system architecture with multi-code rate *1/2* and *1/3*, the calculation equations can be simplified as Table 2. In this table, BM calculation for code rate *1/2* is listed on the left side and code rate *1/3* is on the right side. The marked rectangles in the table indicate the same part during calculation on both sides. Hence, we can find that if we want to calculate BM for both code rates, we just need to compute the right side of the table because the BM value of code rate *1/2* can be extract from the BM value of code rate *1/3*. By using this sharing methodology, there is no need for us to design two different set of hardware for the multi-code rate system to execute the BM calculation function. Then, we can implement this BMU module by only inverters and adders.

Table 2. BM calculation for multi-code rate system.

Code Rate	*1/2*	*1/3*
BM Calculation Equations	*bm00= (-x)+(-y)* *bm01= (-x)+ y* *bm10= x +(-y)* *bm11= x + y*	*bm000 = (-x)+(-y)+ (-z)* *bm001 = (-x)+(-y)+ z* *bm010 = (-x)+ y+(-z)* *bm011 = (-x)+ y+ z* *bm100 = x +(-y)+ (-z)* *bm101 = x +(-y)+ z* *bm110 = x + y+ (-z)* *bm111 = x + y+ z*

C. ACSU reuse methodology

We take an *8*-state trellis diagram in Fig 5 as an example. The left of the figure is the original trellis diagram with four butterflies inside. Each butterfly takes one operation of a single *Processing Element* (*PE*). When we use four parallel PE's to handle the trellis, it takes one operating cycle to run through one stage. But if we use only one PE to do this job, four processing steps are needed to perform one trellis stage. It means that this reuse method takes quadruple processing time comparing with the previous full PE method, but the area is reduced to one-fourth. In the 3G systems applications, traditional convolutional code is

used for voice transmission, so the PE reuse method can be applied due to the low data rate [6][7].

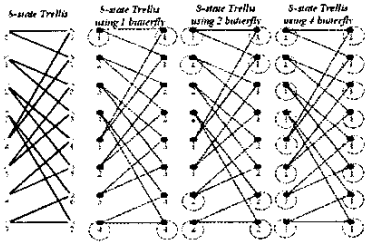


Fig 5. ACSU PE reuse methodology.

D. BARG module

Every time when the generation polynomial function of a turbo code or a convolutional code encoder is changed, it results in a totally different encoded data sequence. However, in the decoder, different generation polynomials only change the branch metrics values put into the ACSU module. Thus, the only thing should be done is to put appropriate BM values to the ACSU. So, we design a new module called BMU to ACSU Routing Generator. Once we know the specification of the current encoder and send control signals to the BARG module, a reconfiguration process will be performed. That means our decoder can fit any encoder with different generation polynomials.

E. UU Module Implementation Issue

About the update unit, from the operating principle, the LLR value at time *t* must compare with the LLR value at time *t-1*. That means we must store LLR values at each timing stage and do a series of comparison operations. In order to reduce hardware consumption, we propose an update unit design with hardware sharing scheme shown in Fig 6 [5] [8].

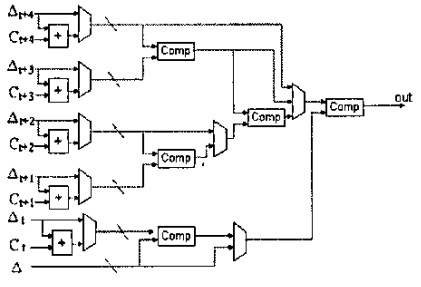
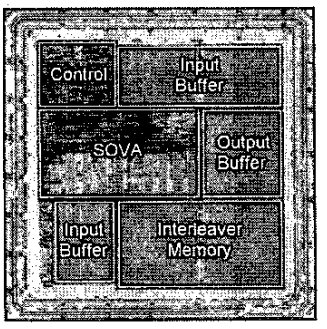


Fig 6. The architecture of UU for our SOVA decoder.

V. Chip Result

Fig 7 shows the chip layout and a summary of our design, the layout is verified by *Design Rule Check* (*DRC*) and *Layout Versus Schematic* (*LVS*) functions in Dracula. The chip die size is *3.5* mm*3.5 mm (core size is *2.7* mm*2.7 mm), uses about *52,000* gate count and *27* kb memory cells. The power consumption is about *746* mW (operating in the turbo code mode) and can work at *83* MHz clock rate to achieve *3.3* Mb/sec data rate when the supply voltage is *3.3* v. It has *40* I/O pads and we use Avant! *0.35 1P4M* standard cell library.



Cell Library	Avant! 0.35 um 1P4M
Voltage	3.3V
Die Size	3.5mmX3.5mm
Core Size	2.7mmX2.7mm
Gate Count	52,000
Memory Size	27,000 bits
Power	746mW@83MHz 449mW@50MHz
I/O Pad	40
Data Rate	3.3Mb/s 2Mb/s

Fig 7. Chip Summary.

VI. Conclusions

A practical prototype design of a dual-mode convolutional/turbo code decoder for CDMA 2000 is proposed. Although this work is developed for the CDMA2000 standard, the basic principle of dual-mode and multi specification for channel coding design can also be adopted to other systems. The methodologies we use here like sharing, reuse, and reconfigurable architecture not only save power consumption but area. With the dual-mode decoder, it is convenient for system designers to plan the whole baseband architecture because the control of our design is simple and explicit.

References

[1] TIA/EIA/CDMA2000, Physical Layer Standard for CDMA2000 Standards for Spread Spectrum Systems, June ,2000.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding. Turbo codes," in Proc. Int. Conf. Communications, May 1993, pp. 1064–1070.

[3] J. Hagenauer, P. Hoeher, "A Viterbi Algorithm with soft-decision outputs and its applications," Proceedings of IEEE Globecom Conference, pp. 1680-1686, 1989.

[4] J. P. Woodard and L. Hanzo, "Comparative study of turbo decoding techniques: an overview," IEEE Trans. on Vehicular Technology, vol. 49, pp. 2208–2233, Nov. 2000.

[5] C. Berrou, P. Adde, E. Angui, and S. Faudeil, "A low complexity soft-output Viterbi decoder architecture," in Proc. Int. Conf. Communications, May 1993, pp. 737–740.

[6] P. J. Black and T. H. Meng, "A 140-Mb/s, 32-state radix-4 Viterbi decoder," IEEE J. Solid-State Circuits, vol. 27, pp. 1877–1885, Dec. 1992.

[7] P. J. Black and T. H. Y. Meng, "Hybrid Survivor Path Architecture for Viterbi Decoders," in Proc. ICASSP 93, 1993, pp. I-433-I-436.

[8] S. Benedetto,a D. Divsalar,b G. Montorsi,a and F. Pollara, "Soft-Output Decoding Algorithms in Iterative Decoding of Turbo Codes" in TDA Progress Report 42-124, pp63-87, Feb. 1996.