Convolutional Code Constraint Length over Rayleigh Fading Channel: Performance Evaluation and Hardware Aspects

A. E. Abdelkareem
Department of Networks Engineering, College of Information Engineering, Al-Nahrain University.
e_mail: ammar.e@coie-nahrain.edu.iq

Abstract— *In this paper, the performance of convolutional code constraint length for bandwidth efficient transmission over AWGN and Rayleigh fading channels is presented. Using intensive simulation and in terms of BER error floor, we evaluate the performance of binary phase shift keying (BPSK) mapping scheme. In addition, the hardware implementation considerations based on the Digital Signal Processor (DSP) platform of convolutional decoding are discussed. To be more specific, the code constraint length effect on both memory management and clock cycles are considered.*

Index Terms— *Convolutional codes, DSP, Hardware., Processing time.*

I. INTRODUCTION

Convolutional codes were invented 5 decades ago by Peter Elias [1]. Unlike block codes, there is a memory at the encoder and the output at any time depending on the current input and the state of the encoder. Andrew Viterbi algorithm was the convolutional decoding, which is a trellis based maximum-likelihood decoder that returns hard decision bits. In practice, convolutional codes are widely used in a variety of wireless systems with several hardware implementation for encoding and decoding [2]. Furthermore, these codes are designed to increase the immunity of the system against random independent errors, whereas communication channels are affected by Fading, Multipath, switching and burst noise. Implementation of coded modulation has attracted great interest in the literature. In [3], the authors address that large memory size and long processing time are the main requirements to implement low-density parity-check (LDPC) convolutional codes. They proposed several strategies to cope with decoding delay. The researchers in [4], have implemented three channel coding techniques based on Motorola DSP56F807EVM platform. In [5], the feedback control signal is utilized to control the constraint length in the Luby transform codes (LT codes). TMS320C6713 DSP kit was adopted in [6] in order to implement adaptive channel coding with Reed-Solomon codes. The authors in [7] have suggested an implementation of Network On-Chip (NoC) for turbo decoding algorithm that reduces the BCJR decoder complexity.

In this paper, the DSP based hardware aspects in terms of storage memory size and decoding processing time are considered to propose techniques for efficient utilization of the available hardware. Two powerful platforms are considered as an example of real kit such as TMSC6713 and ADSP-SHARC. In addition, the performance of convolutional code with BPSK is evaluated.

II. PROPOSED SYTEM MODEL

The proposed system is depicted in *Fig.* 1. The transmitter consists of a **forward error correction (FEC)** with convolutional code and a BPSK mapper. In this figure, a stream of information bits are with the guidelines presented here and

Received 6 April 2017; Accepted 2 November 2017
 ISSN 1811-9212

are encoded by convolutional code to produce a coded word. Then, this coded word denoted by C_k is delivered to a modulator which maps each subsequence C_k to a complex S_k chosen from M-array signal constellation, where $M = 2^m$ and m is the number of bits per symbol. The M-array data of k_{th} symbols are transmitted over a channel described by

$$r_k = g_k . S_k + n_k \quad (1)$$

Where g_k is the fading coefficients with (for AWGN channel $g_k = 1$), and n_k is the complex white Gaussian noise with variance ${}^{No}/_{2}$ in each real dimension. The receiver uses demodulation and convolutional decoding. The received complex symbols r_k is processed by the demapper.

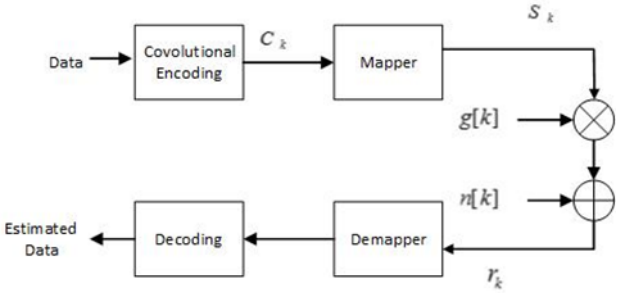

FIG. 1. SYSTEM MODEL.

III. HARDWARE CONSIDERATIONS

A. Memory management

The encoder of a binary convolutional code with rate 1/n-bit code, consists of an M-stage shift register, thus the code rate is given by:

$$r = 1/n \quad (2)$$

The constraint length K of a convolutional code, which represents the number of shifts over a single message, affects the memory size as follows:

$$K = M + 1 \quad (3)$$

Equation (3) represents the number of shifts required for a message to enter the shifter and then go out. Refer to (2) and (3), it is noticed that the entire message length is doubled and added to the constraint length. For instance, a message sequence of length L=8 bits produce an encoded bits of length n (L+K-1) =20, assuming K=3. It is clear that the code length is proportional with the constraint length. The construction of this scenario is shown in *Fig.* 2. This necessitates implementing convolutional encoder and its decoder (Viterbi) in an approach that should tackle such challenge. Implementing a Viterbi decoder necessitates building some data structures such as arrays around which the decoder algorithm will be implemented. For super Harvard architecture (SHARC), part of these arrays are stored in data memory and the others are stored in the program memory.

Dealing is with the available memory size in most of the DSP platforms, where most of these platforms are of limited internal memory. To be more specific, how to buffer the entire message block, where partitioning the message increases complexity and spend the available bandwidth. In this paper, two suggestions are presented to deal with the memory bottleneck: involving the external memory and optimum utilization of the available on-chip memory.

Received 6 April 2017; Accepted 2 November 2017

ISSN 1811-9212

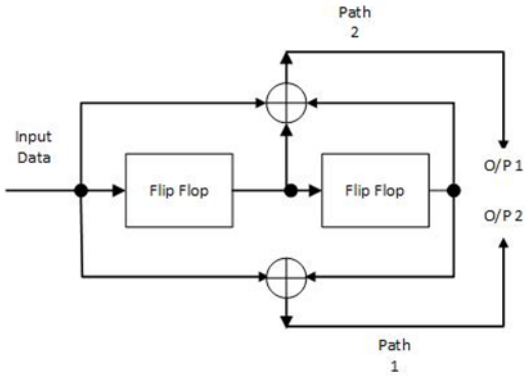

FIG. 2. CONVOLUTIONAL ENCODER WITH CONSTRAINT LENGTH 3 AND RATE ½ [7].

B. External memory allocation

Developing C code for a DSP application somehow needs to explicitly place code and data at specific addresses in memory. There are three techniques to do this.

a) The first technique involves editing the linker description file (LDF) to explicitly locate entire object files in memory segments.

b) The second technique is how to use the SECTION keyword in C source code to explicitly place individual variables and functions in specific memory segments. In such technique, for example, ADSP-21364 SHARC processor, it is possible to map the certain data arrays into the external memory. In this paper, the code below is utilized to map certain data arrays into the external memory (SRAM). It helps to control the placement of the code or data at a specific segment.

```
#pragma default_section(DATA,"seg_sram")
int dm buffer[1024];
```

These tow statements are utilized to change the default mapping of buffer into the seg_dmda(internal data memory) to seg_sram (external memory). On the TMSC6713 platform, this code can be used:

```
#pragmaDATA_SECTION(buffer, ”EXT_RAM”)
```

Where the word pragma represents the directive.

c) When memory or data storage is at premium [8], dynamic memory allocation, which is a type of memory management, offers more memory space for a program, while running or to release space when no space is required. It should be noted that with calloc() instruction, there is no return on its own. The instruction free() must be used by the programmer explicitly to release space. Dynamically allocated memory is the third technique that can be considered.

In this paper, the last two techniques are adopted. It is palpable from *Fig.* 3 that the internal memory is very tight, where it contains 4 blocks of on-chip memory with different word lengths, thus the data resolution and the memory size are tradeoff. In other words, the internal memory space entirely depends on the word length. In the platform ADSP-21364 SHARC processor, there are many formats of the word size according to the available memory shown in *Fig.* 5. For instance, the following arrangement can be used for Short word (16-bit) space. Address range 0x0010 0000 to 0x001F FFFF:

```
// Block 0 0x0010 0000 to 0x0011 FFFF Short word (16)Space(2MbitsROM)
// Block 0 0x0012 0000 to 0x0012 FFFF Reserved address space(1Mbit)
// Block 0 0x0013 0000 to 0x0013 FFFF Short word (16) Space(1MbitRAM)
// Block 1 0x0014 0000 to 0x0015 FFFF Short word (16) Space(2MbitsROM)
// Block 1 0x0017 0000 to 0x0017 FFFF Short word (16) Space(1MbitRAM)
```

Received 6 April 2017; Accepted 2 November 2017

ISSN 1811-9212

// Block 2 0x0018 0000 to 0x0018 7FFF Short word (16) Space(0.5MbitRAM)
// Block 3 0x001C 0000 to 0x001C 7FFF Short word (16) Space(0.5MbitRAM)

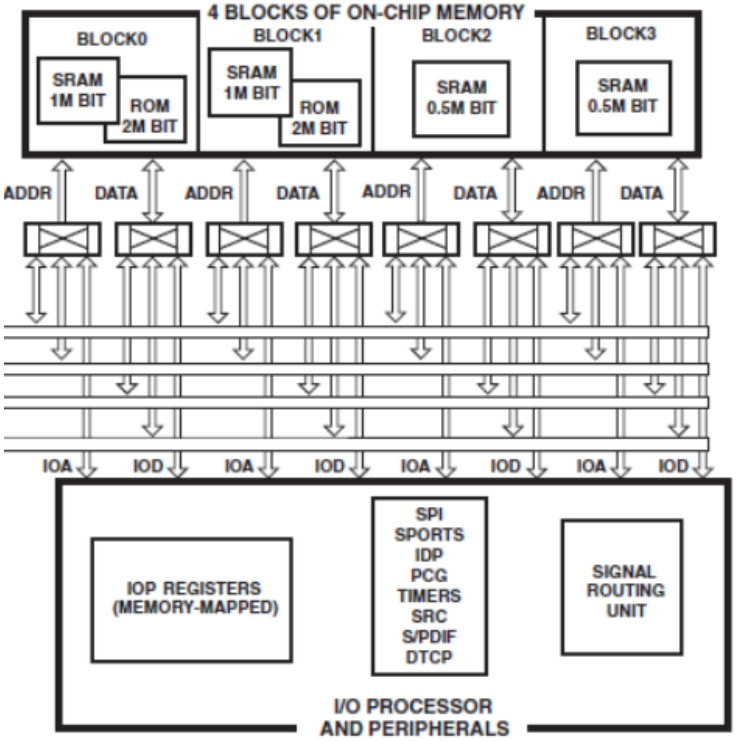

FIG. 3. FUNCTIONAL BLOCK DIAGRAM OF SHARC –PROCESSOR CORE [9].

In this paper, 48-bit word size is utilized to extend the precision. The available SRAM is divided as:

Block 0 0x0009 0000 to 0x0009 5554
Block 1 0x000B 0000 to 0x000B 5554
Block 2 0x000C 0000 to 0x000C 2AA9
Block 3 0x000E 0000 to 0x000E 2AA9

It can be concluded from this arrangement that we exploit 3 Mbit of the SRAM as shown in *Fig.* 4, where it gives 65530 locations of 48-bit which is equivalent to 3Mbit. It is worth pointing out that the sizes of SRAM blocks are not equal as shown in *Fig.*3. The main challenge in designing and implementing such coded system is the calculation of the main arrays of Alpha α, which is called the forward metric, and Beta β, which is denoted as the backward metric [10], where they require a large memory space. In addition, these two parameters spend all computing resources that cause to put the processor offline.

C. Processing time

In order to achieve an optimization in the clock cycles managements, there is a key component needs to be considered which is input/output module of the audio signal. This key is crucial to manage the processing time. There are two types of codec in the DSP. For instance, the ADSP-21364 development board has a built-in module for sampling audio signal. The task is handled by the integrated Analog Device AD183x CODEC family [9]. Whereas AIC23 CODEC [11] is used for Texas Instruments TMS320C6713 platform. Variable Data transfer word lengths with sampling rates from 8 kHz to 96 kHz, are both supported.

Received 6 April 2017; Accepted 2 November 2017

ISSN 1811-9212

Double buffering technique is commonly used in real time signal processing systems to obtain maximum processor utilization. The DSP platform has a direct memory access (DMA) whose function is to transfer a block of data in\out of memory while the processor core works on another block of memory. When the core has finished processing its current block of memory, it waits for DMA to issue an interrupt indicating that it has filled the other block with new data (or output the data). The core now starts to process the new data while the DMA fills/outputs the other buffer. This process is illustrated in *Fig*. 4. The DSP code to implement the double buffering depends on the platform and the timing constraints for convolutional decoding should be considered. However, this technique is very useful for such type of forward error correction (FEC).

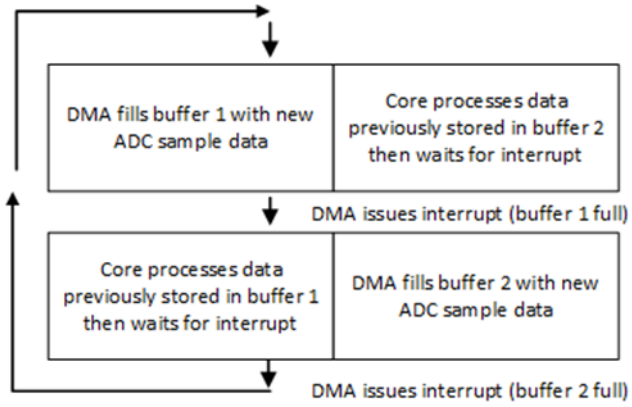

FIG. 4. DOUBLE BUFFERING CONCEPT FOR ADC SAMPLE PROCESSING.

IV. SIMULATION RESULTS

Several simulation results, which have been conducted using MATLAB for Convolutional code over AWGN and Rayleigh channels, are presented. In simulating the system, the generation polynomial was assumed to be [5 7], [23 35], [133 171], and [561 753] for the constraint length of 3, 5, 7, and 9, respectively. The code rate r was 1/2. The transmitted signal is passed through the channel. The complex fading channel g_k is generated as a Gaussian random process and then normalized. The influence of code length on the performance of coded (BPSK) constellation is investigated. In *Fig*. 5, it can be seen that, a convolutional code with 4 states has higher error floor than 256 states at high SNR. However, the complexity is increased. The reason behind this result is due to a small hamming distance and insufficient diversity order. In *Fig.* 6, it is shown that the convolutional code is well performing under Rayleigh channel, where about 4dB gain is obtained when using constraint length of 9 compared with 3. In addition, both figures show that at low SNR, a convolutional code with k=3 (4-sate) outperforms a convolutional code with a higher constraint length. This is because the error multiplicity is increased as the constraint length (number of states) is increased. Rayleigh fading channel illustrates higher level of BER which is due to the fact that Doppler power spectrum occurs and interference of other signals from the environment degrades the performance.

In terms of hardware aspects, there are memory requirements that should be considered to achieve successful implementation of convolutional decoding. It is shown that the primary five arrays we need for the Viterbi decoder are shown in Table I, where SOI denotes size of integer and N_f is the frame length. It is worth mentioning that the first three arrays should be initialized before starting the

Received 6 April 2017; Accepted 2 November 2017

ISSN 1811-9212

decoding. It can be shown that although the authors in [12] have optimized built-in code for turbo decoder, our system memory outcome is very close to their results, where the internal RAM required is 21 N_f.

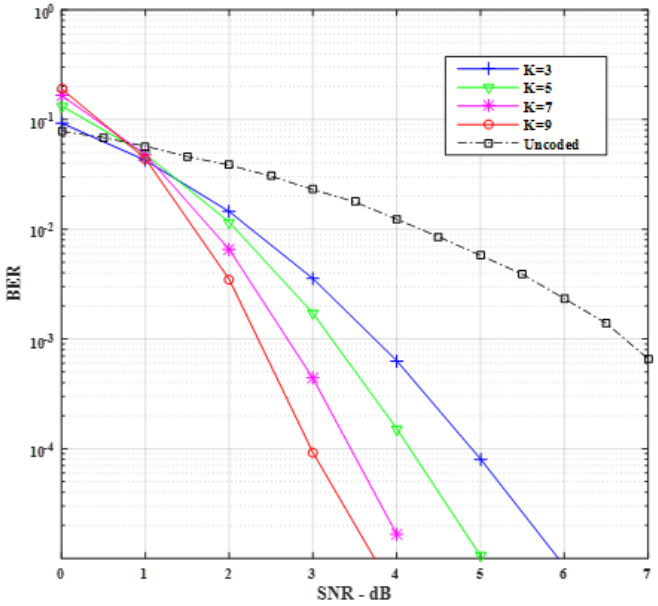

FIG. 5. BER PERFORMANCE OF CODED SYSTEM OVER AWGN CHANNEL.

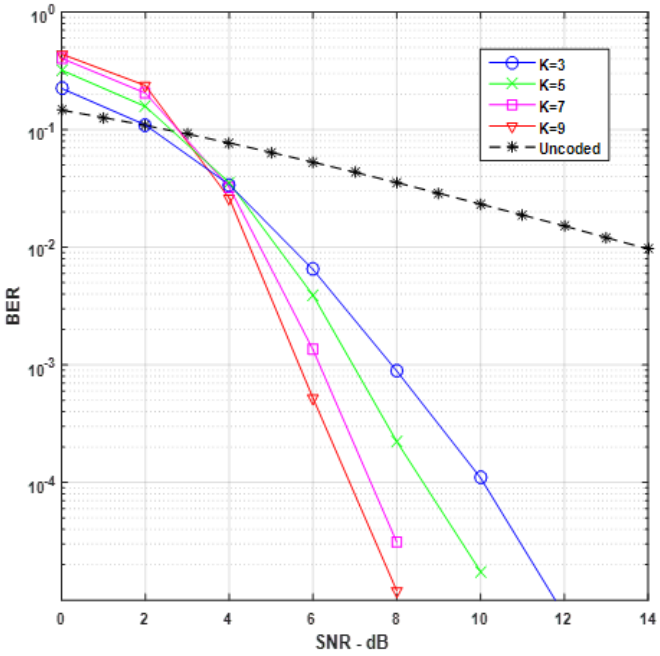

FIG. 6. BER PERFORMANCE OF CODED SYSTEM OVER RAYLEIGH CHANNEL.

Furthermore, the processing time is crucial for coded system. For example, in a sampling interval of 20 μs, and a frame length of 1024 samples, the processor necessitates a frame acquisition interval of 21.33 ms. Actually, the DMA has to complete all the required processing tasks for that frame of data within 21.33 ms and before the next interrupt. However, in some applications such as convolutional coding and its Viterbi decoding, it is required to transpose the processing time according to the acquisition time in addition to the utilization of double

Received 6 April 2017; Accepted 2 November 2017

ISSN 1811-9212

buffering.

TABLE I. ARRAYS NEEDED FOR VITERBI DECODER

Array Name	Purpose	Size
Beta	Copy of the next state table and the state transition table of the encoder.	$2^{K*1} * N_f$
Out0	encoder output table for 0 bit	$2^{K*1} * N_f$
Out1	encoder output table for 0 bit	$2^{K*1} * N_f$
State0	encoder current state and next state of bit 0	$2^{K*1} * SOI$
State1	encoder current state and next state of bit 1	$2^{K*1} * SOI$

V. CONCLUSIONS

In this paper, the performance of convolutional code has been investigated over AWGN and Rayleigh channels. In addition, the hardware considerations in terms of memory management and processing time are presented. One finding that can be drawn from simulation is that the performance of convolutional code is hugely influenced by the constraint length. In conclusion, longer constraint code leads to better performance but with higher complexity. Also, a smaller constraint length accelerates convergence, however, it results in relatively higher error floor, higher processing time and less memory requirements.

REFERENCES

[1] C. Berrou, Codes and Turbo Codes. Springer, 2010.
[2] T. K. Moon, Error correction coding mathematical and algorithms. Wiley, 2005.
[3] Ali Emre Pusane, B. Mulgrew, and Alberto Jim´enez Feltstr¨om, “Implementation Aspects of LDPC Convolutional Codes,” IEEE Trans. Communications, vol. 56, pp. 1060–1069, Jul. 2008.
[4] T. A. K. Opperman, L. Staphorst and S. Sinha, “A hardware implementation of an adaptive channel coding system,” in Proc. AFRICON 2007, pp. 1-6.
[5] J. Hu; H. Gao; and Shi Ge, “The Implementation of Encoder and Decoder of LT Codes based on DSP,” in Conf. Rec. 2012 IEEE Int. Conf. Automation and Logistics, pp. 559 - 562.
[6] V. Gala, H. Nishar, H. Dave and Y. S. Rao, “Real-Time Implementation of an Adaptive Channel Coding System using Reed-Solomon Codes,” in Conf. Rec. 2012 IEEE Int. Conf.
[7] AL-DUJAILY, RAAED, Li, An, Maunder, Robert, Mak, Terrence, Al-Hashimi, Bashir and Hanzo, Lajos, “A Scalable Turbo Decoding Algorithm for High-Throughput Network-on-Chip Implementation,” IEEE Access, Volume 4, pp. 9880 – 9894, 2016.
[8] S. Haykin, and M. Moher, Modern Wireless Communications. Pearson, 2005.
[9] I. P. AL Kelly, *A Book on C*, Fourth Edition. Addison-Wesley, 1998, ch 2.
[10] A. Devices, "ADSP-21364 EZ-KIT Lite Evaluation System Manual," Revision 3.2 ed: Analog Devices, Julay, 2007.
[11] S. Mishra. Gao; H. Shukla, S. Madhekar “Implementation of Turbo decoder using MAX-LOGMAP algorithm in VHDL,” in Conf. Rec. 2015 India Conference (INDICON), 2015 Annual IEEE.
[12] R Chassaing, and D. Reay, Digital signal processing and applications with TMS320C6713 and TMS320C6416 DSK. John Wiley & Sons, 2008.
[13] R. Kothandaraman, M. J. Lopez,”An efficient implementation of turbo decoding on ADI TigerSHARC TS201 DSP,” IEEE SPCOM Inter. Conf., pp. 344-348, 2004.

Received 6 April 2017; Accepted 2 November 2017

ISSN 1811-9212