A POWER AND AREA EFFICIENT MULTI-MODE FEC PROCESSOR*

Yi-Chen Tseng, Chien-Ching Lin, Hsie-Chia Chang, Chen-Yi Lee

Department of Electronics Engineering, National Chiao Tung University

1001, Ta Hsueh Road, Hsinchu, 300, Taiwan, R.O.C.

Tel: +886-3-5726111 Ext.54238; Email: bryan@si2lab.org

ABSTRACT

Forward Error Correction (FEC) is a key component in communication system which mostly contains scrambler, Reed-Solomon coding, interleaving, and trellis coding. For the performance and complexity issues, design parameters are different in various applications. In this paper, a multi-mode FEC processor is presented to meet different system requirements with a power and area efficient architecture. The proposed processor is fully compliant to ITU-T J.83 cable modem system including the reconfigurable Reed-Solomon decoder and memory-based universal convolutional interleaver. With 0.18 μ m 1P6M CMOS technology, the simulation result shows the FEC decoder can work over 100MHz while costs 54.5K gate counts and two 376x8 bits embedded dual-port SRAM. The average power consumption in most critical mode is about 34.2mW at 100MHz. While running at 7MHz that meets symbol rate of cable modem, the power dissipation is 2.32mW.

1. INTRODUCTION

In communication system, channel coding is a key module which uses various types of error correcting algorithm and interleaving techniques to minimize the effect of channel noise during data transmission. They can be summarized as the following four parts in most systems: scrambler, Reed-Solomon(RS) coding, interleaving, and trellis coding. And different applications have specific parameters to achieve an optimum system. Due to the similarity in FEC sections, such as ITU-T J.83, DVB, and ATSC Digital TV, etc, a multi-mode FEC design is an important issue to lower down the design cost. As for the RS code, it is not easy to implement a decoder that meets different finite field definition and generator polynomial, and each application has its own dedicated hardware for RS decoding. Moreover, memory controller of interleaver is also difficult to generate proper addresses for multi-standard. In this paper, a multi-mode architecture of FEC decoder is proposed, which mainly contains a multi-mode RS decoder and a universal convolutional interleaver. The proposed design with the lowest overhead can support different annexes in J.83 and DVB.

The organization of this paper is as follows. In section2, the motivation of multi-mode FEC design will be described, and ITU-T J.83 cable modem system is chosen as the design platform. Reed - Solomon coding with the multi - mode

implementation methods will be addressed in Section 3. In Section 4, a novel and low complexity memory-based interleaving methodology is presented. Section 5 mentions the scrambler and trellis coding. Finally, the overall multi-mode FEC decoder and performance are discussed in Section 6.

2. MULTI-MODE FEC DESIGN

An efficient architecture for multi-mode design is an important issue and challenge to lower down the design cost. In ITU-T J.83 recommendation, there are four annexes for digital transmission system. Digital television cable networks should use one of the systems which are specified in annex A, B, C and D. A comparison of FEC section in different annexes of ITU-T J.83 is listed in table 1. There are three modes in RS codes and various parameters in convolutional interleaving. It is a challenge to design a multi-mode FEC decoder to achieve various standards while considering the complexity and power consumption. The efficient architecture of multi-mode FEC design will be proposed in later sections.

Table 1: Comparison of different specification in FEC

Item	Annex B	Annex A	Annex C	Annex D
Scrambler	$x^3 + x + \alpha^3$ over $GF(2^7)$	$1 + x^{14} + x^{15}$ for 15-bits polynomial of the PRBS		$1 + x + x^3 + x^6 + x^7 + x^{11} + x^{12} + x^{13} + x^{16}$ for 16-bits polynomial of the PRBS
Reed-Solomon coding	(128,122) extended RS codes over $GF(2^7)$, $t = 3$	(204,188) RS codes over $GF(2^8)$, $t = 8$		(207,187) RS codes over $GF(2^8)$, $t = 10$
Interleaving	Convolutional interleaving depth: $I=128,64,32,16,8$ $J=1,2,3,4,5,6,7,8,16$	Convolutional interleaving depth: $I=12$ $J=17$		Convolutional interleaving depth: $I=52$ $J=4$
Trellis coding	$G=(25,37_{octal})$	None		

3. MULTI-MODE RS DECODER

Reed-Solomon decoding process can be divided into four steps [1]. First, a finite field multiplier (FFM) for different finite field definition should be designed. Then, the syndrome calculator calculates a set of syndromes from the received codewords. The key equation solver produces the error locator polynomial $\sigma(x)$ and the error value evaluator polynomial $\Omega(x)$ from the

*Work supported by National Science Council, R.O.C. under grant NSC 91-2218-E-009-010, and Ministry of Economic Affairs, R.O.C. under grant 91-EC-17-A-03-S1-005.

syndromes. By the Chien search and the error value evaluator, we can get the error locations and error values respectively. The proposed multi-mode architecture is described in the following sub-sections. It can be used in many applications, such as ITU-T J.83, DVB system, etc.

3.1. Multi-Mode Finite Field Multiplier

For different RS codes, the different primitive polynomials will cause a challenge to design a finite field multiplier (FFM). However, FFM can be split into multiply and modular operation respectively. The primitive polynomial only has an impact on modular operation. Therefore, the complexity of programmable design just lies in the modular operation. A multi-mode FFM is proposed as shown in figure 1.

Figure 1: Multi-mode FFM over GF(2^m)

3.2. Syndrome Calculator

Figure 2(a) (b) show the two cells of different types in syndrome calculator, figure 2(a) is for GF(2⁸); figure 2(b) is for GF(2⁸) and GF(2⁷) which are decided by current mode. The architecture of multi-mode syndrome calculator is shown in figure 2(c). For different specification, a specific group of cells will be chosen.

Based on [2], moreover, the first t syndromes equal to zeros implies all syndromes are zeros, which can simplify the error detection procedure. It not only improves the power consumption, but also reduces the complexity.

Figure 2: Multi-mode syndrome calculator

3.3. Key Equation Solver

To solve the key equation,

$$\Omega(x) = \sigma(x) S(x) \text{ mode } x^{2t} \dots \dots \dots (1)$$

Berlekamp-Massey (BM) algorithm is used due to its regular operation. For different t, it needs 2t iterations to find error locator polynomial $\sigma(x)$. Based on the proposed multi-mode FFM and modified decomposed algorithm [1] [2], the multi-mode key equation solver is proposed. The computation of $\Omega(x)$ after $\sigma(x)$ results in fewer multiplications and additions than the original BM algorithm. It includes only one key equation solver with three proposed multi-mode FFMs to calculate $\sigma(x)$ and $\Omega(x)$ respectively. Hence, the hardware complexity is reduced. The architecture is depicted in figure 3.

Figure 3: Multi-mode Key equation solver

3.4. Chien Search

Similar to syndrome calculator, there are two cells of different types in Chien search as shown in figure 4(a) (b). The architecture of multi-mode Chien Search is depicted in figure 4(c). For different specifications, the sums of proper cells will be chosen. The cell of C2_L calculates the current calculating location.

Figure 4: Multi-mode Chien Search

3.5. Error Value Evaluator

Forney algorithm is a method to achieve error value evaluator. Assume β_j is the j -th root of error locator polynomial $\sigma(x)$. For annex A, C, and D, the error value:

$$e_i = \frac{\Omega(\beta_j)}{\beta_j \sigma'(\beta_j)} \quad (2)$$

For annex B, the error value:

$$e_i = \frac{\Omega(\beta_j)}{\sigma'(\beta_j)} \quad (3)$$

Figure 6 shows the proposed architecture. It will calculate $\sigma'(\beta_j)$ and $\Omega(\beta_j)$ at the same time while the left mux will choose β_j^2 , the bottom mux will choose β_j . After calculating $\sigma'(\beta_j)$, $\sigma'(\beta_j)$ will multiply β_j in annex A,C and D. In order to calculate the final error value, the bottom mux will choose the upper path.

Figure 6: Multi-mode Error value evaluator

4. UNIVERSAL CONVOLUTIONAL INTERLEAVING

The main purpose of interleaving is to resist burst errors which are induced in noisy channel. And, convolutional interleaving has better ability to spread burst errors than block interleaving. The structure of (I, J) convolutional interleaver based on Forney approach and Ramsey type III approach [3] is shown in figure 7.

Figure 7: Structure of (I, J) convolutional interleaving

It is not efficient for implementing so many pieces of FIFO for hardware which consumes lots of power, area and routing difficulty. Hence, a better solution is to use a two-port SRAM to reduce the area, and power consumption. The key issue becomes how to generate the correct address for each input and output data. As a result, a novel and low-complexity memory-based method to implement the multi-mode convolutional interleaver and de-interleaver is proposed, which is induced from [4] [5].

4.1. A Low Complexity Algorithm for Interleaving

Take an example for $(12, 17)$ convolutional de-interleaver which is adopted in ITU-T J.83A, C and DVB system. In figure 8, the datum received are $0 \times \dots \times 12 \times \dots \times 204 \times \dots \times 2244$ $2041 \dots 11 \dots$, where the number means the input index from interleaver and $\times$ means don't care symbol at the beginning.

2244	2040	600 ... 420	408	396 ... 216	204	192 ... 12	0
2041	1837	...	397 ... 217	205	193 ... 13	1	X ... X X
1838	1634	194 ... 14	2	X ... X X	X ... X X	X ... X X	X
:	:	:	:	X ... X X	X ... X X	X ... X X	:
406 ... 226	214	202 ... 22	10	:	:	:	:
203 ... 23	11	X ... X X	:	:	:	:	:

Figure 8: Received symbols in $(12, 17)$ de-interleaver

When de-interleaving, in figure 9(a), after writing 0 to memory, the interval between 0 and the next writing address is $(I-1) \times J = 187$ (here is x). The interval between previous address and the next address is $(I-2) \times J = 170$, ..., until $2J = 34$. These numbers are the same as the numbers of FIFO on branches. When writing 12 to the memory, it needs go back to the address of "initial writing address - 1" and does the previous operation again. After writing 202 to the memory, the data stored in memory is like figure 9(b). Then we can see that the interval between 0 and 1 is $(I-2) \times J = 170$. The distance between 1 and 2 is $(I-3) \times J = 153$... etc. At this time, the memory size in figure 9(b) is $J \times I \times (I-1)/2$, just the same as the minimum memory requirement in figure 7. Because there is no space to write 2244 into memory, it must increase memory sizes. By the observation, it needs more J memory size. As shown in figure 9(c), when 0 is read out from memory, 2244 is written into memory. Then, 1 is read out, 2041 is written to the original position of 0, etc. Therefore, the required memory size is $J \times I \times (I-1)/2 + J$. The maximum size is 65032 bytes for $(128, 8)$ in ITU-T J.83B, and it just needs more 8 bytes than the original structure.

However, when the address is out of the memory size, it must modulo the address by the memory size.

Figure 9: Behavior of the novel algorithm in $(12, 17)$ de-interleaving

The detail operations in universal de-interleaver are described as pseudo codes in figure 10. Interleaver is just the inverse of de-interleaver, which can be easily formulated too.

```

Initial condition:
w_addr = w_ini_addr = 0;    branch_addr = (I-1)*J;
r_addr = r_ini_addr = (I-1)*J;    counter = 1;
mem_bound = J*I*(I-1)/2 + J;

While ( in != NULL )
{
  if (counter == I)    /* In last branch, input will pass to output directly*/
  {
    out = in;
    branch_addr = (I-1)*J;    /* branch_addr goes back to initial condition */
    counter = 1;    /* reset the counter */
    w_ini_addr = w_ini_addr - 1;    /* reset the writing and reading address */
    r_ini_addr = r_ini_addr - 1;
    w_ini_addr = w_ini_addr mod mem_bound;    /* mod the address */
    r_ini_addr = r_ini_addr mod mem_bound;
    w_addr = w_ini_addr;    /* set writing address to w_ini_addr */
    r_addr = r_ini_addr;
  }
  else
  {
    out = mem[r_addr];    /* read out from memory */
    mem[w_addr] = in;    /* write data into memory */
    w_addr = w_addr + branch_addr;
    branch_addr = branch_addr - J;
    r_addr = r_addr + branch_addr;
    w_addr = w_addr mod mem_bound;    /* mod the address */
    r_addr = r_addr mod mem_bound;
    counter = counter + 1;
  }
}

```

Figure 10: Pseudo codes of universal convolutional deinterleaver

5. SCRAMBLER AND TRELLIS CODING

The circuit complexity of randomization is simple, and it is suggested to use dedicated hardware for different annexes. Trellis coding is only included in J.83B. Using hard-decision Viterbi decoder with 16 states can fit the requirement of ITU-T J.83.

6. DISCUSSION

Base on the proposed multi-mode RS decoder, universal de-interleaver, descrambler and trellis decoder, the overall multi-mode FEC decoder is illustrated in figure 11. Implemented with 0.18um 1P6M CMOS technology, the simulation result shows the FEC decoder can work over 100MHz while costs 54.5K gate counts, two 376x8 bits embedded dual-port SRAM and 65032 bytes external memory for de-interleaver with only 8 bytes overhead. In fact, 7 MHz has met the requirement. The detail gate counts of each module are listed in table 2. Table 2 also shows the gate counts of RS Decoder in ITU-T J.83D which is the most complex RS code in ITU-T J.83. The proposed multi-mode RS decoder is only larger about 1.1K gate count than that specified in J.83D. Besides, the (12, 17) interleaver in [6] needs two 128-byte RAM and four 256-byte RAM. On the other hand, it requires memory size of 1280 bytes. For the proposed algorithm and architecture in the same interleaver, it needs only one 1139-byte RAM and a low complexity controller. In [4], [5] and [6], they can only meet for suitable standard using the same component, but the proposed FEC processor can be used in many standards, such as ITU-T J.83, DVB, ATSC Digital TV, etc. The average power consumptions for each mode in postlayout simulation are listed in table 3. The floorplan of layout is shown in figure 12, and the chip size is 1892x1892um².

Figure 11: Platform of FEC system in ITU-T J.83

Table 2: Gate Count for each module

Module	Logic gate count
Multi-Modes RS Decoder	19051
Universal deinterleaver	8306
Viterbi Decoder	9883
Trellis decoder(contains 2 Viterbi Decoder)	24632
Scrambler	1190
Overall FEC Decoder	54542
J.83D RS Decoder	17963

Table 3: Power Consumption for each mode

Mode	Avg. power(mW)@100MHz	Avg. power(mW)@7Mhz
J.83A & C	18	1.27
J.83B	34.2	2.32
J.83D	19.5	1.34

	Technology	0.18um, 1P6M CMOS
	Max Frequency	100MHz
	Gate Count	54542
	Embedded memory	752 bytes
	Area	3.58 mm ²
	Avg. Power in critical	34.2mW@100MHz 2.32mW@7MHz

Figure 12: Layout of Multi-Mode FEC Processor

7. REFERENCES

- [1] H. C. Chang, C. B. Shung, and C. Y. Lee, "A Reed-Solomon Product-Code (RS-PC) Decoder Chip for DVD Applications," *IEEE J. Solid-State Circuits*, Vol. 36, No.2, pp. 229 -238, Feb. 2001.
- [2] H. C. Chang, C. C. Lin, and C. Y. Lee, " A Low-Power Reed-Solomon Decoder For STM-16 Optical Communications," in *IEEE Asia-Pacific Conf. ASIC*, Aug. 2002.
- [3] J. L. Ramsey, "Realization of Optimum Interleavers," *IEEE Trans. on Inform. Theory*, vol. IT-16, no. 3, May 1970.
- [4] Y. X. You, J. X. Wang, and X. R. Piao, "Design and Implementation of Concatenated Encoder," in *Int. Conf. ASIC*, Oct. 2001.
- [5] H. Yang, Y. Zhong, and L. Yang, "An FPGA Prototype of A Forward Error Correction (FEC) Decoder For ATSC Digital TV," *IEEE Trans. on Consumer Electron*, vol. 45, no. 2, pp. 387 -395, May 1999.
- [6] J. B. Kim, Y. J. Lim, and M. H. Lee, "A Low Complexity FEC Design for DAB," in *ISCAS*, May 2001.