A Parallel Convolutional Coder Including Embedded Puncturing with Application to Consumer Devices

R. Simon Sherratt, *Senior Member*, IEEE

Abstract — *As consumers demand more functionality from their electronic devices and manufacturers supply the demand then electrical power and clock requirements tend to increase, however reassessing system architecture can fortunately lead to suitable counter reductions.*

To maintain low clock rates and therefore reduce electrical power, this paper presents a parallel convolutional coder for the transmit side in many wireless consumer devices. The coder accepts a parallel data input and directly computes punctured convolutional codes without the need for a separate puncturing operation while the coded bits are available at the output of the coder in a parallel fashion. Also as the computation is in parallel then the coder can be clocked at 7 times slower than the conventional shift-register based convolutional coder (using DVB 7/8 rate). The presented coder is directly relevant to the design of modern low-power consumer devices[1].

Index Terms — Parallel Convolutional Coder, DVB, 802.11, Consumer Device, Power Reduction.

I. INTRODUCTION

Many consumers are used to being wirelessly connected to their work, home or social networks by using devices that connect using many differing types of communication (Cellular, Wireless Local Area Network (WLAN), Bluetooth, etc). Manufacturers are under great pressure to release new products to market, and often reuse their own well known and tested Intellectual Property (IP) to create the next generation of products. Such an example of a well known IP module is the convolutional coder, which is used in the transmit side of many popular digital consumer devices.

This paper will present a convolutional coder suitable for consumer devices that is clocked at a much lower rate than conventional convolutional coders thus reducing electrical power. The proposed coder also simplifies the design of the hardware by directly creating punctured data (as opposed to requiring a separate puncturing operation) to create suitable code-rates. The design is also suitable to be directly interfaced to block coders or conventional memory. As an example the application of the technology is directed to the DVB convolutional coder [1] as DVB has many differing code rates. The technology is also directly applicable to IEEE 802.11 [2] and many other popular consumer wireless technologies.

[1] This work was supported in part by the Higher Education Infrastructure Grant Scheme, UK.

R. Simon Sherratt is with the Signal Processing Laboratory, School of Systems Engineering, the University of Reading, RG6 6AY, UK (e-mail: r.s.sherratt@reading.ac.uk).

Contributed Paper
Manuscript received October 16, 2008

Section II will present the basics behind the convolutional coder and issues relating to its implementation in hardware with particular reference to puncturing. Section III presents previous convolutional coders in the literature. Section IV will present the proposed parallel convolutional coder while Section V discusses alternative structures to implement the parallel convolutional coder. The paper concludes with comments on the presented technology and the advantages for consumer devices.

II. CONVENTIONAL CONVOLUTIONAL CODER

The convolutional coder is an extremely popular scheme to add inherent error correction capability to a sequence of digital bits to be transmitted; indeed some of the early uses are in deep space probes [3]. The received bit sequence is often decoded by a Viterbi decoder with the aim to reduce the number of bits in error in the received sequence.

The coder accepts a sequence of logical bits into a shift-register structure, as depicted in Fig. 1 (in this case for DVB-T as IEEE 802.11 coded outputs are in the opposite order). The coder output is formed by a combination of the current input bit and previous input bits, hence adding memory in the code. For each input bit K output bits are generated resulting in a 1/K mother code. Puncturing can periodically remove some coded bits to allow for a tradeoff between the number of bits to be sent and the error correction capability. Puncturing the mother code creates overall code rates of N/K, with N being the number of input bits. 1/2 rate mother codes are often punctured to 1/2 (no puncturing), 2/3, 3/4, 5/6 and 7/8; while 1/3 rate mother codes are often punctured to 1/3, 1/2, 3/4, and 5/8.

Many systems retrieve the data to be coded from local memory or from the output of block coders, hence the data to be coded is often already in a parallel form. A parallel to serial process is often included to apply single bits to the input of the convolutional coder. Likewise, the conventional coder creates output data that may require some bits to be punctured requiring puncture systems to follow the convolutional coder with complicated internal state machines or requiring multiple clocks at the coder.

Due to the issues outlined above, this work presents a method to parallelize the convolutional coding process. It also implements the puncturing operation internal to the coding operation and thus requires only 1 clock to encode blocks of N/K bits. No parallel to serial coder is required and hence the design of consumer based systems can be simplified with the reduction of differing clocks and puncturing state machines.

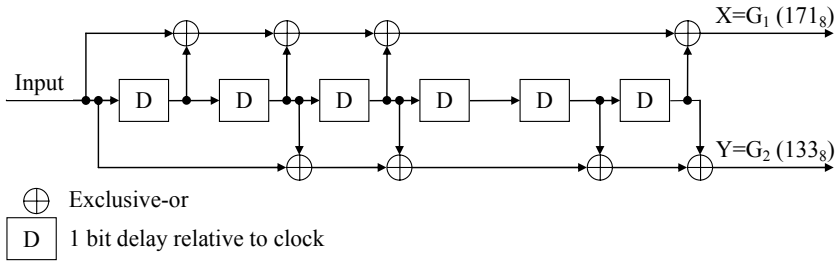

Fig. 1. Standard 1/2 rate mother code DVB convolutional coder, also similar to IEEE 802.11a/b/d/e/g/h/i/j, with IEEE 802.11 outputting Y before X. For each bit clocked into the input, 2 bits are computed, X and Y. Puncturing not shown.

The application of this technology lends itself to any convolutional coding operation mainly performed in hardware but software coders can also benefit.

III. Previous Convolutional Coders

Tang was the first to present a parallel convolutional coder [4]. Tang's coder accepted N parallel input bits and calculated K output bits directly creating a punctured N/K code. Although Tang maintained the shift register approach, the advantage of Tang's approach was to reduce the number of shift register elements to the minimum required to create the punctured code. However, as the number of inputs and outputs are fixed for each code-rate then a different coder is needed for each code-rate leading to a bank of coders and the need to select the appropriate coder. Also as each code-rate uses different taps in the shift register (due to puncturing) then there is little commonality in the structure between each coder. As the number of output bits is a function of the code-rate then interfacing the output of the parallel coder with differing number of output bits requires further hardware to interface to successive operations.

El-Rayis *et al* [5] expanded Tang's work using reconfigurable hardware, but again the number of inputs and outputs varied depending on the code-rate.

It is clear from the literature that a parallel convolutional, coder that accepts a fixed number of input bits and creates a fixed number of output bits would be desirable for low power consumer devices.

IV. Proposed Parallel Convolutional Coder Including Embedded Puncture

This Section will first present a 1/2 rate convolutional coder without puncturing and then various common schemes with embedded puncturing will be detailed.

A. Parallel 1/2 Rate Convolutional Coder

The parallel coder receives as input a M-bit word to code from local memory or from a prior operation. In this case, M=8 will be used, but clearly M=16 or M=32 could be common and the presented system can easily be expanded to account for various values of memory width M. Also, the coded result will be kept at the same value of M to ease interfacing to further operations after the coder.

Fig. 2 presents the structure of the parallel 1/2 rate coder. As can be expected, if 8-bits are output from a 1/2 rate coder then 4-bits need to be applied. The input control function separates the M=8-bit input into 2 4-bit coding operations creating 2 8-bit outputs in 2 clock cycles, or in other words 4 blocks of N/K per clock cycle

The coder requires 6 1-bit memory elements (the same as the conventional coder in Fig. 1), but their implementation is just as a 6-bit register, R, not a shift register. R is a pre-load register that contains the previous 6 input bits, or 0 upon initialization. The operation of the coder is to apply the current 4-bits to be coded I_0, I_1, I_2, I_3 into an exclusive-or (EXOR) array along with the previous 6 input bits stored in register R and to compute all 8 output bits in parallel in the same clock cycle. As 8 bits are directly computed and they need to be the same value as what the conventional coder would output over 4 clocks, then the output of the coder needs to be X_0, Y_0, X_1, Y_1, X_2, Y_2, X_3, Y_3 (with subscript representing each state increment of the conventional convolutional coder). By examining Fig. 1, it can be seen that the outputs are only computed as the EXOR of the current input and previous inputs, but if all 4 inputs bits are present then the 8 outputs can be directly computed. The computation of X_0 and Y_0 can be achieved in the same fashion as Fig. 1 with the current status of R containing the same state as the shift register, and the first input bit, I_0. To compute X_1 and Y_1, the conventional coder clocks the shift register along one place, but in the parallel case, no shift is required as the correct inputs to the EXOR function are already present and only need to be taken from the data already present, likewise for X_2, Y_2, X_3, Y_3. Therefore the 8 output bits to be latched into the output register O, can directly computed all at the same time, i.e.:

$$\begin{aligned}
O_0 &= X_0 = I_0 \oplus R_0 \oplus R_1 \oplus R_2 \oplus R_5 \\
O_1 &= Y_0 = I_0 \oplus R_1 \oplus R_2 \oplus R_4 \oplus R_5 \\
O_2 &= X_1 = I_1 \oplus I_0 \oplus R_0 \oplus R_1 \oplus R_4 \\
O_3 &= Y_1 = I_1 \oplus R_0 \oplus R_1 \oplus R_3 \oplus R_4 \\
O_4 &= X_2 = I_2 \oplus I_1 \oplus I_0 \oplus R_0 \oplus R_3 \\
O_5 &= Y_2 = I_2 \oplus I_0 \oplus R_0 \oplus R_2 \oplus R_3 \\
O_6 &= X_3 = I_3 \oplus I_2 \oplus I_1 \oplus I_0 \oplus R_2 \\
O_7 &= Y_3 = I_3 \oplus I_1 \oplus I_0 \oplus R_1 \oplus R_2
\end{aligned} \tag{1}$$

Where $\oplus$ denotes EXOR operation. After the output has been latched into the output register O, R can be updated in parallel by:

$$\begin{aligned} R_5 &= R_1 \\ R_4 &= R_0 \\ R_3 &= I_0 \\ R_2 &= I_1 \\ R_1 &= I_2 \\ R_0 &= I_3 \end{aligned} \quad (2)$$

On the next clock cycle, the input control selects the other 4 input bits to be coded I_4, I_5, I_6, I_7 from the input register and the coder is ready to compute the second set of 8 coded bits. The coder can be continually clocked to compute the coded bits providing that the input bits are available.

B. Parallel 3/4 Rate Convolutional Coder

Conventionally, puncturing requires the removal of bits already computed. In the parallel coder, the bits that are to be punctured are never computed. Using the DVB 3/4 rate puncturing scheme [1] of X[1 0 1], Y[1 1 0] (it should be noted that IEEE 802.11 puncturing schemes are different [2], but structurally similar), then the first 8 outputs of the coder need to be of the form X_0, Y_0, Y_1, X_2, X_3, Y_3, Y_4, X_5 and they can be directly computed without the need to remove any unwanted bits because again all the required input bits and previous input bits are all available to directly compute the punctured output, i.e.:

$$\begin{aligned} O_0 &= X_0 = I_0 \oplus R_0 \oplus R_1 \oplus R_2 \oplus R_5 \\ O_1 &= Y_0 = I_0 \oplus R_1 \oplus R_2 \oplus R_4 \oplus R_5 \\ O_2 &= Y_1 = I_1 \oplus R_0 \oplus R_1 \oplus R_3 \oplus R_4 \\ O_3 &= X_2 = I_2 \oplus I_1 \oplus I_0 \oplus R_0 \oplus R_3 \\ O_4 &= X_3 = I_3 \oplus I_2 \oplus I_1 \oplus I_0 \oplus R_2 \\ O_5 &= Y_3 = I_3 \oplus I_1 \oplus I_0 \oplus R_1 \oplus R_2 \\ O_6 &= Y_4 = I_4 \oplus I_2 \oplus I_1 \oplus R_0 \oplus R_1 \\ O_7 &= X_5 = I_5 \oplus I_4 \oplus I_3 \oplus I_2 \oplus R_0 \end{aligned} \quad (3)$$

After the result is latched into the output register, the 6 current input bits ($I_5..I_0$) are loaded into R as before, but it should be noted that as a consequence of having a fixed 8-bit input, then 2 input bits (I_6, I_7) have not been used and need to be latched to be used with 4 bits ($I_{11}..I_8$) of the next input word forming the next 6-bit input word to the coder. Upon computation then the remaining 4 input bits that have not been used ($I_{15}..I_{12}$) can be latched to be used with 2 bits ($I_{17}..I_{16}$) of the next input word, to form the next input set. Lastly, the final 6 bits of the current input word ($I_{23}..I_{18}$) may be used and the sequence repeated.

C. Parallel 7/8 Rate Convolutional Coder

In an identical fashion to 3/4 rate, the 7/8 rate coder directly computes 8 output bits from 7 input bits without the need for separate puncturing. Given the puncturing scheme X[1 0 0 0 1 0 1], Y[1 1 1 1 0 1 0], then the 8 punctured outputs are directly calculated from:

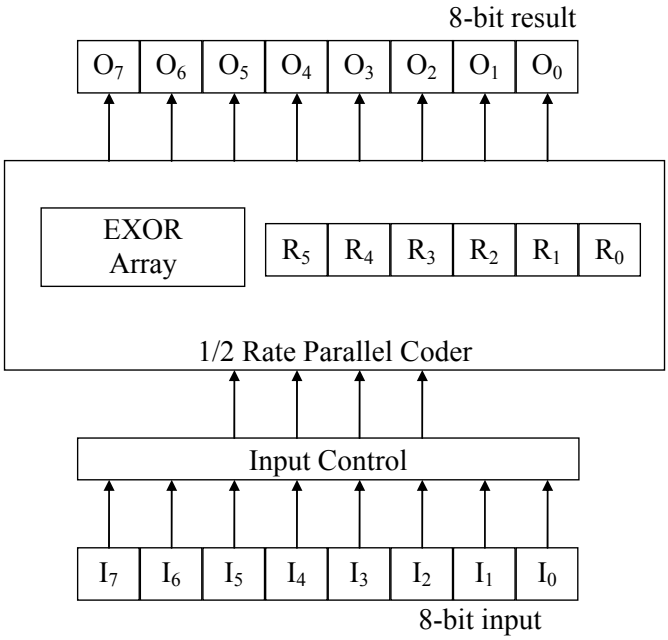

Fig. 2. Architecture of 1/2 rate Parallel Convolutional Coder with 8 bit input word an 8 bit output word.

$$\begin{aligned} O_0 &= X_0 = I_0 \oplus R_0 \oplus R_1 \oplus R_2 \oplus R_5 \\ O_1 &= Y_0 = I_0 \oplus R_1 \oplus R_2 \oplus R_4 \oplus R_5 \\ O_2 &= Y_1 = I_1 \oplus R_0 \oplus R_1 \oplus R_3 \oplus R_4 \\ O_3 &= Y_2 = I_2 \oplus I_0 \oplus R_0 \oplus R_2 \oplus R_3 \\ O_4 &= Y_3 = I_3 \oplus I_1 \oplus I_0 \oplus R_1 \oplus R_2 \\ O_5 &= X_4 = I_4 \oplus I_2 \oplus I_1 \oplus R_0 \oplus R_1 \\ O_6 &= Y_5 = I_5 \oplus I_3 \oplus I_2 \oplus I_0 \oplus R_0 \\ O_7 &= X_6 = I_6 \oplus I_5 \oplus I_4 \oplus I_3 \oplus I_0 \end{aligned} \quad (4)$$

Similar to 3/4 rate input control, 7 of the 8 input bits are used with the unused bits latched to be used with the next input word.

D. Parallel 2/3 and 5/6 Rate Convolutional Coders

Coding 2/3 and 5/6 is a similar structure, but codes to 6 parallel output bits. Given the 2/3 puncturing scheme X[1 0], Y[1 1] and for 5/6 X[1 0 1 0 1], Y[1 1 0 1 0], then output bits may be computed as:

2/3 Rate:

$$\begin{aligned} O_0 &= X_0 = I_0 \oplus R_0 \oplus R_1 \oplus R_2 \oplus R_5 \\ O_1 &= Y_0 = I_0 \oplus R_1 \oplus R_2 \oplus R_4 \oplus R_5 \\ O_2 &= Y_1 = I_1 \oplus R_0 \oplus R_1 \oplus R_3 \oplus R_4 \\ O_3 &= X_2 = I_2 \oplus I_1 \oplus I_0 \oplus R_0 \oplus R_3 \\ O_4 &= Y_2 = I_2 \oplus I_0 \oplus R_0 \oplus R_2 \oplus R_3 \\ O_5 &= Y_3 = I_3 \oplus I_1 \oplus I_0 \oplus R_1 \oplus R_2 \end{aligned} \quad (5)$$

5/6 Rate:

$$
\begin{aligned}
O_0 &= X_0 = I_0 \oplus R_0 \oplus R_1 \oplus R_2 \oplus R_5 \\
O_1 &= Y_0 = I_0 \oplus R_1 \oplus R_2 \oplus R_4 \oplus R_5 \\
O_2 &= Y_1 = I_1 \oplus R_0 \oplus R_1 \oplus R_3 \oplus R_4 \\
O_3 &= X_2 = I_2 \oplus I_1 \oplus I_0 \oplus R_0 \oplus R_3 \\
O_4 &= Y_3 = I_3 \oplus I_1 \oplus I_0 \oplus R_1 \oplus R_2 \\
O_5 &= X_4 = I_4 \oplus I_3 \oplus I_2 \oplus I_1 \oplus R_1
\end{aligned}
\tag{6}
$$

V. Alternative Structures

Section III presented the definitions of the parallel outputs for various coders without any optimization or reduction.

A. Logic Reduction

Considering (1-6), it can be seen that individual outputs are calculated from 5 inputs with 4 operators, but that the calculation for each output is also derived from sub-operations calculated as part of neighbor calculations, therefore gate minimization can be applied across the calculation of the M outputs within each coder.

Further minimization can be achieved when combining all the coders together because the calculation of X_0 and Y_0 is the same irrespective of code-rate, and there exists subsets of identical calculations in all the schemes.

B. Common Width Output Register

As has been presented, this paper selects a common output width of the parallel coder of M=8-bits. However the code-rates of 2/3 and 5/6 only use 6 of the 8 available bits. The smallest common denominator for all of the code-rates is 24. Hence if the output register is extended to M=24-bits then all the code rates fully populate the output register. Also as 8 is a common factor of 24, then 8-bit bytes can easily be extracted from the output of the parallel convolutional coder as would be expected from DVB block codes.

VI. Conclusion

Convolutional coders are used in transmitters or in the transmit chain of many digital wireless devices. This paper has presented a method to compute blocks of Convolutional coded output bits all in parallel. There are advantages of such computation including reducing the number of clocks in the device, no need to have puncturing and indeed remove the need for state machine puncturing. Assuming that the data to be coded comes from memory or a block coder then also no parallel to serial conversion is necessary. Other advantages relate to reducing clock frequencies and hence reduce electrical power, all of which contribute to smaller consumer devices with longer battery life.

References

[1] Standard ETSI EN 300 744, "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television",V1.5.1 (2004-06)

[2] Standard IEEE 801.11-2007, "IEEE Standard for information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements", Part 11, 2007.

[3] J. L. Massey and D. J. Costello, Jr., "Nonsystematic Convolutional Codes for Sequential Decoding in Space Applications", IEEE Transactions on Communication Technology, Volume 19(5), October 1971, pp. 806-813.

[4] B. Tang, "Parallel Punctured Convolutional Coder", US Patent 6598203 22 July 2003

[5] A.O. El-Rayis, T. Arslan and A.T. Erdogan, "High Performance Embedded Reconfigurable Concatenated Convolution-Puncturing Fabric for 802.16", IEEE Second NASA/ESA Conference on Adaptive Hardware and Systems, 5-8 Aug. 2007, pp. 190-194

R. Simon Sherratt (M'97-SM'02) was born in Heswall, UK. He received his B.Eng. in Electronic Systems and Control Engineering from Sheffield City Polytechnic UK in 1992, M.Sc. in data telecommunications in 1994 and Ph.D. in video signal processing in 1996 both from the University of Salford UK. Since 1996, he has been a Lecturer in Electronic Engineering at the University of Reading where he is now a senior lecturer in consumer electronics and currently head of Electronic Engineering. His research topic is signal processing in consumer electronic devices concentrating on equalization, communications layer 1, DSP architectures and adaptive signal processing. Eur Ing Dr. Sherratt is a senior member of the IEEE, IEEE Consumer Electronics Society and currently Vice President (Conferences) of the IEEE Consumer Electronics Society, member of the Society AdCom (2003-2008), Society awards chair (2006 and 2007), member of the IEEE Transactions on Consumer Electronics publications committee (2004-), IEEE International Conference on Consumer Electronics vice-technical chair 2007, technical chair 2008 and general chair 2009 and IEEE International Symposium on Consumer Electronics general chair 2004. He received the IEEE Chester Sall 1st place best Transactions paper award in 2004 and the best paper in the IEEE International Symposium on Consumer Electronics 2006.