Cyrille Chavet · Philippe Coussy *Editors*

Advanced Hardware Design for Error Correcting Codes

Springer

Advanced Hardware Design for Error Correcting Codes

Cyrille Chavet • Philippe Coussy
Editors

Advanced Hardware Design for Error Correcting Codes

Springer

Editors
Cyrille Chavet
University of South
Brittany, Lorient, France

Philippe Coussy
University of South
Brittany, Lorient, France

ISBN 978-3-319-10568-0 ISBN 978-3-319-10569-7 (eBook)
DOI 10.1007/978-3-319-10569-7
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014951358

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

For many years, experts more expert than the rest have regularly heralded the end of research focused on the physical layer of telecommunications. Some claim that the best has already been delivered from the promises offered by the theory of communication, others say that the theoretical limits predicted will never be reached by simple means. As Costello and Forney explained in an award winning IEEE article [1], this pessimistic standpoint is nothing new and some were already proclaiming “Coding is dead” in the early 1970s, only 20 years after the pioneering work of Claude Shannon. Other experts, this time in the field of microelectronics, have also regularly announced the end of CMOS technology, starting as early as the mid-1980s when the submicron barrier for mass production seemed insurmountable to some. “CMOS is dead” was also commonly heard.

Fortunately, these doubts were swept away each time they were raised. And this was often because of the need for increasingly demanding telecommunications: farther, faster, more reliable, that microelectronics increased its efforts in the miniaturization of components. Conversely, the steady progress of the semiconductor industry has opened the way to new processing information algorithms unforeseen at the time of the first generations of integrated circuits. Information, as understood by Shannon, and the transistor were born around the same time in the legendary Bell Labs and, from that time, have continued to join hands to lead to ever more effective telecommunications systems which have become indispensable in our daily lives.

Among the processes made possible today by high density integration on silicon, distributed error correction coding (or channel coding) came to occupy a place of prime importance. To put it simply, distributed coding is to monolithic coding what a combination of small mathematical relationships is to one complex equation. It was by adopting the point of view of distributed computing that error correction coding proved able to find its best practical solutions to achieve optimality, or nearly. Rather than trying to build a codeword by a single coding operation and recover it through a single decoding step, it is wiser to adopt the strategy of “divide and rule”, not at the cost of lesser performance, quite the contrary.

In the early 1990s, the competition (all virtual) between monolithic coding and distributed coding in the race for optimality designated its winner. On the one hand, a team from the prestigious Jet Propulsion Laboratory in Pasadena was working to develop a Viterbi decoder for a 16384-state convolutional code: the Big Viterbi Decoder (BVD) [2]. It had to consist of 256 identical integrated circuits each processing 64 states, plus additional control circuits. The correction power significantly exceeded the state of the art but an entire table was needed to lay the decoder. On the other hand, in a little known French laboratory, an electronics engineer wondered whether two small convolutional codes, typically of 8 or 16 states, associated in an original way and iteratively decoded one after another could not do better than the Californian code. The answer was affirmative: three integrated circuits (same number as iterations) were enough to provide better correction power than the BVD.

A large part of the community of digital communications therefore became interested in this new way of building a redundant code that its inventor [3] later named turbo code [4] to keep it significantly shorter than "parallel concatenation of recursive systematic convolutional codes decoded iteratively". It also provided the opportunity to take a new look at and give a new impetus to LDPC codes [5, 6] to which many researchers turned their attention, whether for optimization or implementation. Different distributed structures, in parallel, in series or both together, were proposed one after the other. From the most compact of distributed codes—a turbo code with only two component codes—to more distributed—LDPC codes or the most recent and promising polar codes [7]—all kinds of solutions were possible. In addition to studies on "modern coding" [8], the philosophy of decoding by message passing was also expanding its ramifications to applications other than channel coding, for example equalization [9, 10], demodulation [11] or joint source-channel coding [12]. "Do not lose any of the pieces of information available in the receiver whatever the level" became the leitmotif of many researchers.

Some were also questioning what once had been considered absolute certainties: but no, on balance, coding is not only a matter of mathematics. Because information theory was built on non-trivial mathematical concepts, such as entropy and mutual information, it was indeed long believed that practical solutions would be exclusively provided by mathematics. But math is especially used to justify and set parameters, rarely to create and build. While algebra, probability and graph theory continue to be part of the arsenal of skills of engineers and researchers in communication technologies, other knowledge and skills have also become indispensable: computer science, electronics, and, in particular, parallel architectures needed to obtain high throughputs. The days when it was legitimate to invent a code or any algorithm without proposing practical ways of decoding or implementation are over. Not only must processes be compatible with what electronics can provide but other significant constraints such as energy consumption in embedded systems or the speed of information transfer in highly distributed structures can be crucial.

The proliferation of new applications (very high throughput cellular systems, sensor networks, Internet of Things, etc.) and the demand for improved performance are still ongoing challenges. It is no longer just a matter of bits per second per hertz or bit error rate; now it is also necessary to consider other criteria such as joules per bit (in transmission as in reception processing), flexibility and interoperability. A new generation of researchers has emerged, mastering at the highest levels the interdisciplinarity necessary to cope with these multiple constraints. Some of these researchers have come together to write this book with the latest ideas and developments in the design of circuits for error correction encoding and decoding. Tomorrow's telecommunications are in their hands and we can certainly say alongside them: "Coding and CMOS are still alive, and for a long time to come".

[1] Costello DJ, Forney GD (2007) Channel coding: the road to channel capacity. Proc IEEE 95(6):1150–1177
[2] Statman J, Rabkin J, Siev B (1989) Big Viterbi decoder (BVD) results for (7,1/2) convolutional code. TDA Progress Report 42–99, JPL, November 1989
[3] Berrou C (1991/1995) Error-correction coding method with at least two systematic convolutional codings in parallel, corresponding iterative decoding method, decoding module and decoder. Patents no 9,105,280 (France, April 1991), no 5,446,747 (USA, August 1995)
[4] Berrou C, Glavieux A, Thitimajshima P (1993) Near Shannon limit error-correcting coding and decoding: turbo-codes. In: Proceeding of IEEE ICC '93, Geneva, pp 1064–1070, May 1993
[5] Gallager RG (1962) Low-density parity-check codes. IRE Trans Inf Theory IT-8:21–28
[6] MacKay DJC, Neal RM (1995) Good codes based on very sparse matrices. In: Boyd C (ed) Cryptography and coding 5th IMA Conference, Lecture notes in computer science, no 1025. Springer, Berlin, pp 100–111
[7] Arikan E (2009) Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. IEEE Trans Inf Theory 55(7):3051–3073
[8] Richardon T, Urbanke R (2008) Modern coding theory. Cambridge University Press, New York
[9] Douillard C, Picard A, Didier P, Jézéquel M, Berrou C, Glavieux A (1995) Iterative correction of intersymbol interference: turbo-equalization. Eur Trans Telecom 6(5):507–511 (special issue on turbo decoding)
[10] Laot C, Glavieux A, Labat J (2001) Turbo equalization: adaptive equalization and channel decoding jointly optimized. IEEE J Select Areas Commun 19(9):1744–1752
[11] Hoeher P, Lodge J (1999) Turbo DPSK: iterative differential PSK demodulation and channel decoding. IEEE Trans Commun 47(6):837–843
[12] Hagenauer J, Görtz N (2003) The turbo principle in joint source channel coding. In: Proceeding of ITW 2003, Paris, pp 275–278, April 2003

Claude Berrou
May 2014

Contents

Chapter 1
User Needs

David Gnaedig

TurboConcept is an industry-reference provider of Intellectual Property Cores (IP Cores) for advanced Forward Error Correction (FEC) techniques (turbo codes and LDPC codes). We propose IP Core products, which offer to our customers the best trade-offs between error correction capability, throughput, silicon cost, and power consumption. Since 2007, TurboConcept is part of the Newtec group, specializing in satellite communications equipments and systems. We have developed IP cores implementing encoders and decoders addressing most of the families of error correcting codes:

– Turbo codes: since the development of the first DVB-RCS turbo decoder in 1999 (a duo-binary turbo code), we have developed IPs for all flavors of convolutional turbo codes that are specified in standards such as the CCSDS, WCDMA, Homeplug AV, WiMAX, LTE.
– LDPC codes since their adoption in 2002 by the DVB-S2 standard. We have naturally extended this IP to cover also DVB-T2, DVB-C2, and have then developed products for WiMAX, G.hn, and more recently WiFi 802.11ac.
– Turbo product codes: two- and three-dimensional product codes.
– Convolutional codes for WiFi, LTE, WCDMA.
– BCH codes, that are used in concatenation with LDPC codes in DVB-S2 standard for instance but also for other proprietary concatenation schemes.

We also propose proprietary error correcting codes based on either turbo or LDPC codes for various applications: satellite communication, wireless backhaul, …

Our FEC IP cores target both ASIC and FPGA designs. During the first years of the company most IP cores were designed for FPGA devices. But since 2005 with

D. Gnaedig (✉)
TurboConcept, 185 rue Joseph Fourier, 29280 Plouzane, France
e-mail: david.gnaedig@turboconcept.com

C. Chavet, P. Coussy (eds.), *Advanced Hardware Design for Error Correcting Codes*, DOI 10.1007/978-3-319-10569-7_1

the emergence of the WiMAX market and later on with the throughput increase of 3G cellular systems, and in particular the evolution toward LTE, we have developed a range of IPs that are integrated into ASIC designs. Our sales volume is today a balance between FPGA and ASIC users. It is also worth noting that for some markets (for instance LTE base station) we propose two different IP cores, one optimized for ASIC designs and another optimized for FPGA design. For ASIC, the IP is optimized for best area and specifically memory area and low power consumption. For FPGA designs, the IP architecture takes advantage of specific resources available for "free" in FPGA devices: dual port RAMs, a large number of registers enabling high pipe-lining and thus very high clock frequency, multiple clock domains, …

Our story starts with turbo codes and related iterative decodable codes, back in the second half of the 1990s. At that time, it was a real challenge to have an FPGA or an ASIC hosting such a complicated function as a turbo decoding, especially when compared to legacy convolutional codes. The first prototype chip initiated by turbo codes inventors was based on a "one-chip-per-iteration" pipelining [5]. Moore's law is obviously a great enabler in the wide adoption of turbo and LDPC codes: it helped greatly to minimize the complexity overhead of iterative decoding, as compared to legacy FECs, even in the context of ever-higher-throughput applications. Algorithmic advances also helped significantly: the transposition of the probabilistic decoding equations into the logarithmic domain led to low complexity algorithms to decode efficiently turbo codes (log-MAP [1]) or LDPC codes (min-sum [2]). At the architectural level, significant breakthroughs have enabled to develop low complexity and high throughput decoder architectures. First, sliding window algorithms [3] enabled to reduce drastically the required memory of the BCJR algorithm while maintaining acceptable performance. Second, the concept of parallel soft-in–soft-out implementations accessing in concurrence to a shared memory, applicable to both turbo and LDPC code decoding enabled to reach several tens of Mbits/s and up to several Gbits/s for LDPC codes using actual technologies. Additionally, the concept of "shuffled scheduling" (also referred to as "layered" or "turbo" scheduling) of LDPC codes has also contributed to almost dividing by two the required number of iterations and thus increasing the throughput equivalently. Finally, resolving memory access issues has been a critical issue in massively parallel architectures. It has been tackled by taking into account the constraints of the architecture early in the design of the code. Such architecture-aware code design techniques have been used for:

- Turbo codes through the design of a structured interleaver (e.g., DVB-RCS code with the ARP interleaver, or quadratic interleaver used in the LTE specification. This structure of the interleaver can be exploited by the parallel decoding architecture to enable collision free memory accesses.
- LDPC codes through the design of a prototype parity matrix which specifies the complete parity matrix of the code in a condensed way. The expansion factor that enables to derive the binary parity matrix from the prototype matrix offers a natural level of parallelism that is then exploited when designing high throughput architectures.

The choice of a given code for a given application is usually driven by the requirements of the application in terms of latency, SNR operation range, target BER, flexibility (block size, code rates, …). But we have also encountered cases where the choice of an error correcting code is driven by marketing objectives rather than technical reasons. For instance LDPC codes are seen sometimes as a "new" technology while turbo codes are present since many years into various applications and therefore LDPC may be selected by customers even if turbo codes may have superior error decoding performance for this application. A single code family that would outperform other codes over all possible applications does not exist yet and will to our opinion never exist. Therefore, a trade-off shall be made depending on the most important application requirement. Usually, turbo codes have superior BER performance for small block sizes and low code rates and enable a large flexibility both in block size (using a parametric interleaver) and code rates (through puncturing). Turbo product codes have very good performance in the high code rate region typically around above 0.80 with a very low complexity. They are attractive for very high throughput applications, owing their inherent parallelism ability, but they offer a poor flexibility. Convolutional codes decoded by the Viterbi algorithm have their interest for very short block size (typically a few tens of bits) and/or for applications where the critical factor is the lowest latency due to the fact that they do not require an iterative decoding scheme. LDPC codes have better performance for very large (typically a few tens of thousands bits) to medium block sizes (around a few thousands bits) and have the advantage of enabling high throughput parallel implementation with an affordable complexity. They lack however in a large flexibility as each block size-code rate combination requires to specify one parity check matrix. Also encoding complexity of LDPC codes grows quadratically with the block size while the encoding complexity of convolutional code and convolutional turbo code grows only linearly with the block size. This apparent complexity drawback can however be greatly mitigated by introducing specific structures in the parity check matrix of LDPC code like a so-called "staircase" structure of the parity bits sub-matrix.

These general trends are continuously evolving due to large research efforts devoted to code design. LDPC codes are getting more and more efficient with small block sizes especially when considering the non-binary LDPC codes. New interleaver design techniques for turbo code bring significant improvement in the error floor region over previous generation, especially for high code rates, one of the turbo code weaknesses. In addition to the evolution of the now "old" turbo and LDPC code families, brand new code structures are being introduced: spatially couples LDPC codes, polar codes which are proven to achieve (and not only approach) capacity of a given channel.

Once the code family is selected, to define a set of codes suitable for the application, other key parameters have to be determined in light of their impact on the implementation complexity. For turbo codes, this includes the choice of the recursive systematic convolutional code (larger memory induces lower error floor but at the cost of higher complexity), the design of the interleaver that influences greatly the performance in the error floor region, the puncturing scheme. For an

LDPC code, the parity check matrix density has an influence on the error floor but also on the convergence and on the complexity. Also a specific structure in the parity bits region of the parity check matrix is helpful to enable simple encoding scheme.

With standardized applications, the choice of the code itself is obviously not part of our degrees of freedom, but a constraint to which the designed IP core product must comply with. In light of our implementation expertise, we see however how choices made on code design may be very helpful to reduce implementation complexity without sacrifying the error decoding performance. For example, for DVB-S2 codes, there exists the well-known issue of double-diagonal events present in the protograph matrix. Resolving this issue can be performed with various architectural solutions that have an impact on the implementation complexity, throughput, and/or performance. Therefore, if double-diagonal events can be avoided when designing the code, it would be beneficial for enabling low complexity LDPC decoder architectures. An active participation to standardization bodies through the proposition of specific coding schemes is the natural way to influence the choice of the channel coding scheme. To this end, TurboConcept has participated to several standardization processes: DVB-RCS, DVB-S2, and more recently to DVB-SX (as part of Newtec).

Proprietary applications give a larger degree of freedom in the code design and the adaptation of the code design to the target hardware implementation. These include satellite communication systems, wireless backhaul, magnetic storage (hard disk drives), military and governmental... We have developed coding schemes (association of code and modulation) for some of these applications.

When designing products incorporating error correcting codes we need to take into account three typical constraints: throughput, area, and power consumption. First, in terms of throughput we saw the demand for increasing throughput from a few Mbits/s in the early 2,000 years (e.g., in cellular, satellite communications applications) to today's several hundreds of Mbits/s in most wireless applications. Our latest products are scaled to offer several Gbits/s in a wireless physical layer system. Optical links and other markets have even higher throughput demands, but we are not addressing them specifically up to now. Second, for a given throughput requirement, a low implementation area is always desirable for obvious cost reasons. The area constraint greatly varies on the market. Indeed the area being mostly a cost issue, the impact of the area is essentially linked to the other cost elements from the application (e.g., cost of the radio bandwidth, number of users, other operational costs). As an example, if we consider the satellite market, the hub operating the network can afford a large (and expensive) FPGA and therefore a code of higher implementation complexity for the return link. The gain in signal-to-noise ratio translates into more available capacity and thus additional users for the same satellite frequency band. This naturally induces an increased profitability. On the other side, for a consumer equipment where the cost of implementation is of primary interest, the constraint on the area is more stringent. On an ASIC target, the area is mainly driven by the memory area and therefore, the size of the code impacting largely the memory area is an important trade-off between the implementation complexity and the error decoding performance of the code. For an FPGA implementation, the decoder needs to fit into a low-end FPGA with limited resources (logic and memory). Assuming a throughput from a few tens of Mbits/s to a few hundreds of Mbit/s turbo codes, LDPC code can today be implemented even in low cost FPGAs. Finally, power consumption is obviously getting more and more important, and the relative importance of low power aspects is increased for mobile equipment. We characterize our products by actual numbers (technology dependant) but also by using some design rules and guidelines that ensure the core is not wasting power useless (systematic use of enable signals propagated along the data path, no free-running logic, minimal access to memory blocks, …).

When designing our products especially when targeting very high throughput architectures we faced several algorithmic and architecture problems that needed to be solved efficiently. On the algorithmic side, increasing the throughput of decoder requires specific techniques in order to maintain good error decoding performance and fast convergence. For example for convolutional turbo code, dealing with very high code rates induces specific algorithm design as the conventional sliding window BCJR algorithms using acquisition for initializing border state metrics are not efficient (and even useless) [4] when code rates grow above 0.95, as it is the case of HSPA. For LDPC codes, higher throughput requires a higher level of parallelism that makes more challenging the selection of a good scheduling for layered decoding architectures. On the architectural side, developing high throughput decoders means that the interfaces and the interleaver (in a BICM scheme) shall support the same level of throughput. Therefore the requirement for high throughput induces to design high speed parallel interleavers. With the increase of the throughput requirement in the future this issue is getting more and more complex to solve because contrary to the code design that have been performed in light of parallelism constraints, it is rarely the case for the associated external interleaver. One last constraint is related to the validation of the performance for FER as low as 10^{-11}. This performance validation is necessary since implementation and especially fixed point representation may introduce a floor, even if the code itself has no floor, This is usually achieved by using an FPGA board able to simulate at throughput of several Gbits/s.

Advanced error correcting codes have made significant progress over the last 20 years and are now used in a large range of applications. There are still areas that need substantial improvements. First, the choice of a FEC coding scheme is often based on some simplified channel modeling (AWGN being the simplest commonality). More progress can be made by considering a refined channel model of the application and to optimize the code in light of this channel model, which is not a simple task. Indeed, refining the channel model often results in a (much) larger design space (e.g., phase noise parameters, multi-path characteristics, non-linearity models). More theoretical methodologies need to be developed in order to find good codes in this context. Second, another important aspect is to find techniques for predicting and designing codes for high order modulations especially in the error floor region. Bit interleaved coded modulation has been used in the past as a mean to design independently the code and the modulation and to achieve good performance on high order modulations. But this technique does not address the optimization of the performance in the error floor region. Finally, techniques for finite length optimization of codes are still an area that needs to be explored. The techniques that are usually used to characterize code ensembles assume infinite block sizes and ideal BP decoder that does not suffer from correlation due to cycles in a real iterative decoder. Finding optimal code for finite length code is still a challenging problem.

The future challenge that needs to be tackled by the next generation codes and architectures is flexibility. Not flexibility in the sense of an universal decoder that would supports all types of codes of all possible standards. This kind of universal decoder does not seem today to be an industrial requirement, and it is sometimes more complex than using dedicated and optimized cores, one per application. By flexibility, we mean the ability for the code and the decoder to adapt dynamically to changing channel conditions in order to always obtain the best error decoding performance while minimizing power consumption. One mean to achieve this objective is to design codes that can be decoded algebraically when the SNR is high thus enabling high throughput and low power. When the SNR is low an iterative decoder would be used in order to achieve performance close to the Shannon limit. Algorithms such as the bit-flipping algorithm for LDPC codes seem to be very promising in this sense.

In conclusion, we envisage the evolution of error correction coding driven by three main requirements. First, continuous increase of throughput requirements: next generation broadband wireless access systems target maximal data rate in the order of 1 Gbits/s on a handheld device. Second, reducing power consumption will become a major requirement even for non-battery powered applications, and finally, improved error decoding performance (closer to Shannon capacity) by taking into account refined channels models during the code design stage.

References

1. Robertson P, Villebrun E, Hoeher P (1995) A comparison of optimal and sub-optimal decoding algorithm in the log domain. Proceedings IEEE international conference on communications, Seattle, June 1995, pp 1009–1013
2. Fossorier M, Mihaljevic M, Imai H (1999) Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. IEEE Trans Commun 47:673–680
3. Viterbi AJ (1998) An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes. IEEE J Sel Areas Commun 16:260–264
4. Boutillon E, Sanchez-Rojas J-L, Marchand C (2013) Compression of redundancy free trellis stages in Turbo-decoder. Electron Lett 49(7):460–462
5. CAS 5093 40 Mb/s Turbo code decoder. December Rev 4.1. Comatlas. (May 1995)

Chapter 2
Challenges and Limitations for Very High Throughput Decoder Architectures for Soft-Decoding

Norbert Wehn, Stefan Scholl, Philipp Schläfer, Timo Lehnigk-Emden, and Matthias Alles

2.1 Motivation

In modern communications systems the required data rates are continuously increasing. Especially consumer electronic applications like video on demand, IP-TV, or video chat require large amounts of bandwidth. Already today's applications require throughputs in the order of Gigabits per second and very short latency. Current mobile communications systems achieve 1 Gbit/s (LTE [1]) and wired transmission enables even higher data rates of 10 Gbit/s (e.g., Thunderbolt [2], Infiniband [3]) up to 100 Gbit/s. For the future it is clearly expected that even higher data rates become necessary. Early results show throughputs in the order of 100 Tbit/s [4] for optical fiber transmissions.

Satisfying these high date rates poses big challenges for channel coding systems. Software solutions usually achieve only very small data rates, far away from the required speed of most applications. Therefore dedicated hardware implementations on ASIC and FPGA are mandatory to meet the requirements for high speed signal processing. To achieve speeds of Gigabits per second, these architectures need large degrees of parallelism.

Parallelism and speed can easily be increased by running several single decoders in parallel. This is however mostly an inefficient solution, because area and power increase linearly with parallelism. Moreover it implies a large latency.

N. Wehn (✉) • S. Scholl • P. Schläfer
Microelectronic System Design Research Group, University of Kaiserslautern, Kaiserslautern, Germany
e-mail: wehn@eit.uni-kl.de; scholl@eit.uni-kl.de; schlaefer@eit.uni-kl.de

T. Lehnigk-Emden • M. Alles
Creonic GmbH, Kaiserslautern, Germany
e-mail: info@creonic.com

C. Chavet, P. Coussy (eds.), *Advanced Hardware Design for Error Correcting Codes*, DOI 10.1007/978-3-319-10569-7_2

Thus it is more advantageous to investigate efficient architectures specialized to high throughput. This may also include modifications to the decoding algorithm itself.

An important metric for analyzing high throughput architectures is area efficiency. Area efficiency is defined as throughput per chip area ($[(\mathrm{Gbit/s})/\mathrm{mm}^2]$). The area efficiency can be increased significantly by new architectural approaches.

We present high throughput decoders for different application relevant coding schemes, such as Reed–Solomon, LDPC and Turbo codes and point out their benefits compared to state-of-the-art architectures.

2.2 Architectures for Soft Decision Reed–Solomon Decoders

2.2.1 Introduction

Reed–Solomon (RS) codes are utilized in many applications and communication standards, either as a stand-alone code or in concatenation with convolutional codes, e.g., DVB. They are traditionally decoded using hard decision decoding (HDD), using, e.g., the well-known Berlekamp–Massey algorithm. However, using also the probabilistic information—so-called soft information—on the received bits can lead to large improvement of frame error rate (FER) in comparison to HDD.

Numerous algorithms have been proposed for soft decision decoding of RS codes. They are using different approaches to achieve a gain in FER over HDD with different complexities. A selection of interesting algorithms can be found in [5–11]. This chapter will focus on the RS(255,239) and RS(63,55) codes, because they are widely used in many applications.

Up to now, only few hardware implementations for ASIC and FPGA have been proposed for soft decoding of RS codes, especially the RS(255,239). One trend becoming apparent are implementations based on Chase decoding [7] and the closely related low-complexity chase (LCC) algorithm [8]. Hardware implementations based on LCC exhibit low hardware complexity [12–14], but this low complexity comes at the expense of a poor FER gain. Implementations based on LCC provide only little FER gain over HDD of about 0.3–0.4 dB.

The design of hardware architectures for a larger gain in FER is more challenging. Architectures and implementations based on adaptive belief propagation and stochastic Chase decoding exhibit a larger FER gain (0.75 dB), but having a low throughput [15, 16].

In this chapter a third approach for soft decoding is described that enables a large gain in FER *and* high throughput. It is based on a variant of the information set decoding algorithm, for which an efficient architecture is presented. This architecture shows a uncompromised gain in FER of 0.75 dB and a high throughput that exceeds 1 Gbit/s on a Xilinx Virtex 7 FPGA [17].

2.2.2 *Information Set Decoding*

This section introduces the algorithm, which is the basis of the high throughput hardware architecture. First, the used variant of information set decoding called ordered statistics decoding (OSD) algorithm [10] is reviewed. Then, a reduced complexity version of OSD using the syndrome weight [18] is presented.

2.2.2.1 Original OSD

OSD has been proposed in [10] and belongs to the class of information set decoders [19].

Basically information set decoding works as follows: First, divide the N received bits $\bar{\mathbf{y}}$ into two groups according to their reliability. The bit reliability is determined by the absolute value of the corresponding log likelihood ratio (LLR). The first group contains the set of K reliable bits, called the *information set*. The second group contains the $M = N - K$ unreliable bits, also referred to as low reliable bit positions (LRPs).

Before actual decoding starts the M LRPs are erased. Then the information set is used to reconstruct the M erased bits using the M parity checks in the parity check matrix $\mathbf{H}$. To do so, $\mathbf{H}$ has to be put in a diagonalized form $\hat{\mathbf{H}}$ via Gaussian elimination. If the K bits of the information set are correct, all errors in the M LRPs can be corrected. This is referred to as order-0 reprocessing in OSD or OSD(0).

To perform successful correction in the case of one error in the information set, the reconstruction process is repeated several times, each time with exactly one bit of the information set flipped. This results in a list of $K+1$ possible codewords from which the best codeword is selected by evaluating the Euclidean distance to the received LLRs. This improved decoding is called order-1 reprocessing or OSD(1).

A key part for understanding information set decoding is the reconstruction process. To successfully reconstruct the M erased bits, the rows of the parity check matrix are used, as mentioned before. It is required that the parity check equation in each row covers mostly one of the erased M bits. To fulfill this requirement, $\mathbf{H}$ is put into a diagonalized form by Gaussian elimination, such that each row covers not more than one erased bit. Note that the Gaussian elimination does not change the channel code itself, because an RS code is a linear code.

2.2.2.2 Reduced Complexity Algorithm for Hardware

The computational bottleneck of the original algorithm are the reconstructions of the M erased bits. For example, in case of decoding an RS(255,239) with OSD(1) this operation is required 2,041 times. To overcome this problem a reduced complexity algorithm is utilized which makes use of the syndrome $\hat{\mathbf{s}}$ and its weight [18] that enables fast and efficient reconstruction.

The reduced complexity algorithm starts (as the original OSD) with determining the information set according to the bit reliabilities and diagonalization of $\mathbf{H}$ by Gaussian elimination.

Original OSD then evaluates the parity check equations given by the rows of $\hat{\mathbf{H}}$ whereas the considered low-complexity algorithm merely uses the syndrome to correct the corrupted bits.

Moreover, if the syndrome vector is calculated using the diagonalized parity check matrix, i.e., $\hat{\mathbf{s}} = \hat{\mathbf{H}}\bar{\mathbf{y}}^{\mathbf{T}}$, two distinct cases for the binary weight of the syndrome vector can be observed:

- The syndrome weight is small: In this case, it is assumed that only errors in the M bits are present, i.e., OSD(0) processing is sufficient.
- The syndrome weight is large: In this case, it is assumed that also errors in the information set are also present. Then OSD(1) processing is performed.

A fixed weight threshold to decide between the two cases is denoted by $\Theta \in \mathbb{N}$ and determined by simulation.

OSD(0) (small syndrome weight) is performed by simply flipping the M LRPs that have led to the 1s in the syndrome vector. Conducting OSD(1) (large syndrome weight) to correct one error inside the information set is done by first flipping the bit position

$$j = \underset{i=0,\ldots,N-1}{argmin}\ wgt\left(\hat{\mathbf{s}} \oplus \hat{\mathbf{h}}_{\mathbf{i}}\right)$$

where $\hat{\mathbf{h}}_{\mathbf{i}}$ denotes the ith column of $\hat{\mathbf{H}}$. After flipping the error inside the information set at position j, the syndrome is calculated again and the remaining errors outside the information set (i.e., among the LRPs) are corrected by performing OSD(0).

Note that this algorithm inherently determines the best codewords among the possible candidates only by looking at the syndrome weight. It is sufficient to select the candidate with smallest syndrome weight. In case of original OSD the Euclidean distance between candidate and received LLRs had to be evaluated many times.

For more detailed information on the syndrome weight OSD please refer to [18] or [17].

2.2.2.3 HDD Aided Decoding

One disadvantage of OSD over other soft decision decoding algorithms is the tendency for a weak FER performance if SNR increases. To improve FER OSD is extended with a conventional HDD, whose result is output if OSD fails.

A failure in OSD can be easily detected again by looking at the syndrome weight. If after OSD(1) reprocessing the updated syndrome still has a large weight, OSD can be considered as unsuccessful.