Optimization of the "FOCUS" Inband-FEC Architecture for 10-Gbps SDH/SONET Optical Communication Channels

Afxendios Tychopoulos Odysseas Koufopavlou
Department of Electrical and Computer Engineering,
Polytechnic School of the University of Patras, Rio Patras, Greece 26500
atychopoulos,odysseas@ee.upatras.gr

Abstract

Forward-Error Correction (FEC) is of key importance to the robustness of optical communication networks. In particular, Inband-FEC is an attractive option, because it improves channel-performance without requiring an increase of the transmission bandwidth. We have devised and implemented a novel inband FEC method, dubbed FOCUS, for the electronic-mitigation of physical impairments in SDH/SONET optical networks. It is an inherently low-cost approach for both the metro and backbone network regions, scalable to any SDH/SONET rate and capable to significantly increase optical channel performance. This paper analyzes the most sophisticated ones from the plethora of optimizations that were employed to minimize the architectural complexity of FOCUS, falling in: a) Arithmetic operator design, b) Resource sharing and c) Redundant logic elimination. These optimizations were necessary to obtain a prototype, which eventually permitted the first fully successful laboratory evaluation of the FOCUS Inband-FEC method.

1. Introduction

Due to impairments that are inherent in optical channels (noise, dispersion, non-linearities, switching penalties, etc), some form of signal-regeneration must take place periodically in order to restore the signal quality. However, conventional 3R-regeneration schemes involve opto-electronic conversions (O-E-O), which quickly render the overall cost of optical networks prohibitive and therefore, retard their expansion and proliferation. Ideally, O-E-O conversions should be replaced by all-optical compensation and/or regeneration, but this technology is not yet sufficiently mature, it has therefore considerable limitations and remains expensive as well. Nevertheless, optical transparency (i.e. no O-E-O conversions) has established itself as a primary objective, enabling a higher degree of upgradeability and flexibility.

In this direction, a promising solution is based on the inclusion of advanced electronic-processing mechanisms only at the end nodes of optical links, capable to offer improved transmission efficiency characteristics and thus, to significantly increase the distance, over which the signal can be transparently transmitted. This solution is particularly desirable in metro networks, where the initial-investment cost is of vital importance.

The majority of today's metro networks are based on SDH/SONET technology, whereupon forward error correction (FEC) is employed to upgrade channel performance and assure Quality of Service (QoS). In the case of outband-FEC (oFEC), the parity (redundant) bits, introduced by the encoder at the transmitting end, can be used to detect and possibly correct channel-errors by the FEC decoder at the receiving end. This is however at the expense of an increased line-rate (bandwidth), in order to preserve the client bit-rate (SDH/SONET-signal) fixed.

Inband FEC (iFEC) can be used alternatively, to avoid the necessity for additional (external) parity-bits, leaving the line-rates unchanged. This interesting property can be obtained by utilising parts (octets) of the SDH/SONET frame-overhead (OH) sections that are defined by the widely adopted SDH/SONET standards, but remain unused in practice. From the early days of those standards, many researchers have elaborated on the idea of iFEC coding [1]. Notably, ITU-T rec. G.707 includes an iFEC method too. Backbone networks can be benefited by iFEC through its property to seamlessly integrate with oFEC, offering an increased combined coding-gain and enhanced immunity to channel impairments.

The proposed iFEC method, FOCUS, was initially introduced in [2]. It applies to the Regenerator (RS) and Multiplex-Section (MS) functionality of SDH (equally, SONET), utilising the shortened Reed-Solomon code RS(244,240) operating over Galois-Field GF(2^9). Coding can be switched (on-the-fly) between a 'strong'-mode (redundancy: 1.67%) and a 'weak'-mode (redundancy: 0.83%). The option of 'weak'-mode decoding is important for applications that require a moderate coding-gain; these should benefit from the significantly reduced implementation-complexity of this mode.

In designing FOCUS, particular attention has been paid to the prevention of error-multiplication [3]. This phenomenon is common to all forms of FEC, making its appearance when the channel is impaired above the code's corrective reach. It results from FEC decoding-errors and

978-3-9810801-2-4/DATE07 © 2007 EDAA

failures, which cause wrongful, damaging corrections on client data. The choice of the shortened code RS(244,240) for FOCUS, which features less than 50% data capacity (240/507 = 47.33%) – in comparison with the corresponding full code RS(511,507) – significantly reduces error-multiplication from decoding-errors, because any wrongful correction in the null-space of the shortened code is implicitly skipped. With regard to error-multiplication from decoding-failures, FOCUS 'strong'-mode decoding applies the 'root-multiplicity' criterion on the error-locator polynomial for the timely termination of wrongful corrections; notably, this criterion is degenerated in the case of 'weak'-mode and cannot be of use.

FOCUS has been modelled in VHDL and implemented in a large Xilinx® Virtex-II™ FPGA (XC2V-3000-4), situated on '10g-Tester', a PCB card designed for operation in excess of 10Gb/s [2]. At the transmitting card, an Intel® LXT16717 MUX is used to serialize the SDH frames, while at the receiving card an Intel® LXT16716 DEMUX parallelizes the frames again to allow for bulk processing at lower speed. A micro-controller, local to each card, enables Fast-Ethernet connection with a supervising PC, which configures the cards and collects the signal-integrity statistics from FOCUS. The main micro-controller functions are: switching between the 'weak' and the 'strong'-mode of FEC, SDH scrambling activation and deactivation, frame- and bit-error count.

The present paper analyzes the optimizations that were necessary to obtain an efficient implementation of FOCUS. They can be grossly classified in the following categories: a) Arithmetic operator design, b) Resource sharing and c) Redundant logic elimination. Fitting the FOCUS FPGA-implementation netlist into the target device, allowed for its first laboratory evaluation against optical impairments (at 10Gbps) [3]. This achievement is attributable to the high-degree of optimization. In the following section, these optimizations are introduced and analyzed in detail.

2. Optimizations

This section is an analytical survey of the optimizations that were applied to the VHDL-implementation of FOCUS.

2.1. Arithmetic operator design

Galois-Field arithmetic over GF(2^9) underlies the FOCUS iFEC coding [4], [5]. By prudently selecting the Field Generator Polynomial (FGP) of GF(2^9), substantial area savings can be achieved, attributable to the impact of FGP-choice on the cost of the multiplication-operator over the GF(2^9), henceforth called 'GF-mult'.

More specifically, the complexity of this iFEC design is dominated by the GF-mult operations in the FOCUS decoder. The full-scale decoder for SDH STM-64 consists of 192 identical, partial, STM-0-level decoders. Assuming a brute-force implementation (no optimizations), then every partial decoder ('strong'-mode) would involve:

- 5 GF-mult for the computation of syndrome-values.
- 3 GF-mult for the calculation of the coefficients of the error-locator polynomial (λ).
- 6 GF-mult for the 'Chien-Search' and the computation of error-values (Forney's formula).

In total, FOCUS decoder's cost in GF-mult for 10Gbps STM-64 operation (i.e. 192 STM-0 subframes) amounts to:

$$192 * (5 + 3 + 6) = 2688 \text{ GF-mult}$$

This cost is evidently prohibitive. Even by applying extensive resource-sharing (see 2.2), the above cost was eventually reduced by a factor of 12 to:

$$2688 / 12 = 224 \text{ GF-mult}$$

Nonetheless, the above cost remains unaffordable, especially with regard to FPGA implementations.

Another important consideration is the contribution to the total complexity from the division-operators over GF (henceforth called 'GF-div'). Notably, GF-div operators turn out to be approx. 8 times more costly than GF-mult. Taking, again, resource sharing into account, 32 GF-div operations are required in total by the FOCUS decoder for STM-64. The cost of these operations is equivalent to:

$$32 \text{ GF-div} * 8 = 256 \text{ GF-mult}$$

Finally, apart from the decoder, all iFEC implementations must include the RS- and MS-layer functionality of an essential SDH STM-64 deframer. The above points obviate that dropping the cost of operators to a minimum is imperative, especially for GF-mult.

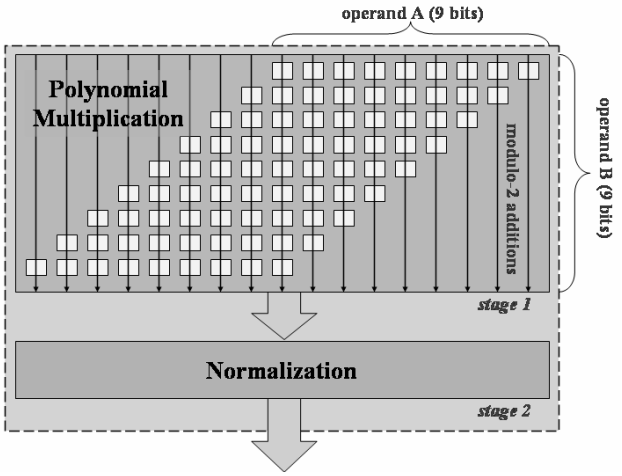

Figure 1: Stages in parallel GF-multiplication

Bit-serial operator-architectures are, in general, the least demanding in terms of area [4], [5]. However, a bit-serial architecture for the GF-mult operator would incur a delay that is unacceptable, due to the real-time character of the SDH protocol. On the contrary, only a fully parallel architecture of the GF-mult operator can meet the design-constraints for processing-speed. Such a parallel

architecture for GF-mult can be split in two stages, as shown in Figure 1.

The 1st stage is a usual polynomial multiplication over GF(2) between the operands of GF-mult (treated here as polynomials with bit-coefficients). With FOCUS operating over GF(2^9), operands are 9-bit wide and the output product is 17-bits wide. The 2nd stage normalizes the product back to 9-bits to ensure that the algebraic field GF(2^9) is closed under the operation of multiplication. Normalization is achieved by dividing the product with a constant polynomial, which is said to 'generate' the field, hence its name: Field Generator Polynomial (FGP) [5].

The 1st stage is a vector of variable length [(A•B)+(C•D)+…] operations, with + and • denoting addition and multiplication over GF(2) respectively. Evidently, the choice of FGP does not have an impact on the complexity of the 1st stage, rather only on the 2nd stage. More specifically, during the modulation of the 1st stage-output by the FGP in the 2nd stage, the non-zero FGP coefficients – apart from the most significant bit (MSB) – infer • operators, which have the cost of 1 XOR gate each. Therefore, the less non-zero coefficients the FGP contains, the simpler is the corresponding normalization-circuitry.

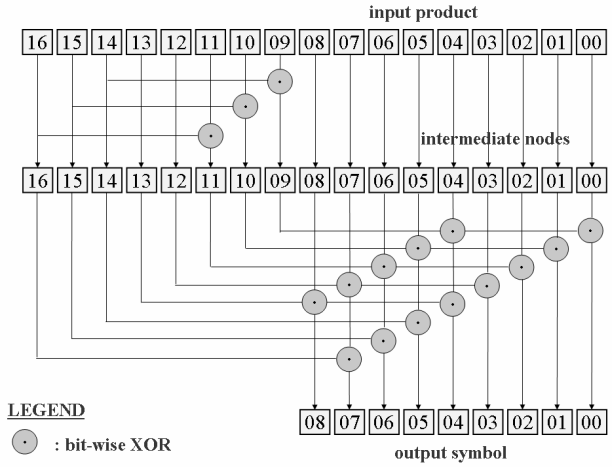

Figure 2: Minimum circuit for GF(2^9)-normalization

In examining the Hamming-sense weight of all possible FGP for GF(2^9), only one was found to have the minimum weight, which equals 3. In fact, this is the absolute minimum, because the most- and the least-significant bits of an algebraically primitive polynomial (FGP) must equal 1 and the number of non-zero coefficients must be odd. We therefore selected

$$f(x) = x^9 + x^4 + 1,\ W^H_{f(x)} = 3$$

for FGP, resulting in the low implementation cost of 50 Look-Up Tables (LUT) per GF-mult. This result is readily achievable by automated digital-synthesis with commercial tools, such as Mentor Graphics® Leonardo Spectrum™ and Precision Synthesis™ i.e. without custom design of the GF-mult. The optimized normalization circuit is depicted in Figure 2.

The above architecture will be henceforth called 'GF-mult architecture A'. Although an achievement in its own right, the total area-cost in LUT of the above parallel GF-mult architecture remained unaffordable:

(224 + 256) GF-mult * 50 LUT/GF-mult = 24000 LUT

This is particularly true, bearing in mind that the target FPGA device (Xilinx XC2V-3000-4) contains 28672 LUT in total and that many are consumed for routing purposes.

Consequently, other architectures had to be examined as well, aiming at the exploitation of additional FPGA resources, such as the embedded RAM & embedded multipliers.

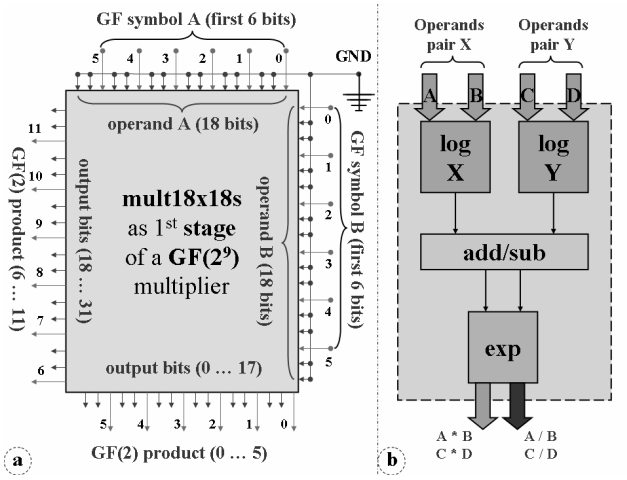

Figure 3: (a) using embedded multipliers as 1st stage of parallel GF-mult, (b) most efficient parallel GF-mult & GF-div architecture, using embedded RAM

Remarkably, there is one primitive element already in the target device that partially implements the 1st stage of the parallel GF-mult: the embedded multiplier. The device contains 96 embedded multipliers (mult18x18s), each with 18-bit long operands, for signed fixed-point multiplication. There is however one important distinction between an embedded multiplier and a GF-mult: the former incorporates carry-propagation. To exploit this pool of mult18x18s, the carry propagation must be disabled and this can be done by grounding the 2 subsequent input bits for every usable input bit, as shown in Figure 3 (part a). Accordingly, the 2 subsequent output bits for every usable output bit are to be ignored.

From Figure 3 (part a) can be easily deduced that a single mult18x18s replaces only 6 out of the 9 bits that make one GF-mult operand (i.e. GF-symbol). Nonetheless, the introduction of 1 mult18x18s primitive in the parallel GF-mult operator architecture decreases the LUT-cost to 40 (GF-mult architecture B_1). The same cost drops to only

25 LUT by dedicating 2 mult18x18s per GF-mult (GF-mult architecture B_2), one at 6 LSB & another at 6 MSB.

In an effort to achieve further area-savings, taking also the embedded RAM of the target device into advantage, the GF-mult operation was analyzed in elementary operations, according to the following formula:

$$a * b = \exp[\log(a) + \log(b)]$$

This analysis is reasonable, because the exponentiation (GF-exp) and logarithm (GF-log) operations act upon a single operand (symbol) and can be implemented by tables. Conveniently, the GF-symbol bit-width (9 bits) matches the dimensions of the embedded-RAM primitive element of the target-device (RAMB16), making the use of RAMB16 very attractive.

More specifically, one RAMB16_S36_S36 fully implements 2 independent GF-log or 2 independent GF-exp. Consequently, 3 RAMB16 plus 2 general-purpose adders can implement 2 parallel GF-mult, as depicted in Figure 3 (part b) i.e. the average cost of a parallel GF-mult operator drops to as low as 1½ RAMB16 plus one adder. Adders have almost negligible cost.

Analogously, the GF-div operation can be analyzed as:

$$a / b = \exp[\log(a) - \log(b)]$$

Due to the close similarity of the above formulas, a GF-mult architecture (that relies on RAM16) can be easily extended to encompass the GF-div operation. The only modification required for this purpose is optionally negating the second operand of the adder. In conclusion, the architecture shown in Figure 3 (part b) can very efficiently implement either 2 GF-mult or 2 GF-div operators and will be henceforth called 'GF-mult/div architecture C'. The target device contains 96 RAMB16 primitive elements. Consequently, 64 GF-mult can be readily implemented using the above architecture.

2.2. Resource sharing

Since each STM-0 requires a dedicated, partial FOCUS encoder-decoder pair, 192 such pairs are presumably required to protect STM-64 signals against channel-errors. In the previous subsection, it was evidenced that the resources for a brute-force implementation of such an iFEC method are huge. Fortunately, the internal operations of FOCUS can be parallelized, giving rise to substantial area-savings through resource sharing.

Bulk-processing, such as iFEC-coding and SDH functionality, has to take place at significantly lower clock-speed, in comparison with the very high rates of serial transmission. In particular, the prototype of FOCUS processes data in 128-bit parallel-form to have the internal bulk-processing speed reduced to 77.76 MHz accordingly. 16 STM-0 (out of 192 in STM-64) are processed in parallel, each with one data-octet (16 * 8 = 128) at a time. Processing one data-octet from all 192 STM-0 within STM-64 lasts therefore 12 clock-cycles (192 / 16 = 12).

In other words, the STM-0's with serial-numbers that are equal modulo 16 will be assigned to the same processing sub-channel. There are 16 sub-channels and each one of them processes 12 STM-0 in a circular fashion. Processing within each sub-channel can be classified in:

- Operations (e.g. arithmetic, logical, etc)
- Updating of state-variables.

Clearly, STM-0 cannot share their state-variables, but sharing the operations is fully legitimate, since processing is identical for each and every STM-0 subframe within STM-64. Furthermore, no two STM-0 within a sub-channel can have their state simultaneously updated. Therefore, partial (STM-0) FOCUS decoder state can be conveniently stored in RAM, rather than flip-flops (FF). Moreover, RAM can be organized in rows and columns, with each row corresponding to a single STM-0 within this sub-channel. Notably, the target device contains an abundance of LUT-RAM with depth 16 > 12 i.e. a single primitive LUT-RAM element suffices for a 1-bit state-variable for all 12 STM-0 within a sub-channel. Resource sharing is crucial for the feasibility of FOCUS, because it allows a reduction of complexity by a factor of 12.

Another crucial observation is that some costly operators in the FOCUS decoder are not used concurrently. More specifically, FOCUS decoding cycles continuously between the following steps, once for each codeword [5]:

1. **(S)** : Syndrome values computations
2. **(L)** : Error-Locator polynomial (λ) formation
3. **(C)** : 'Chien'-Search determination of the roots of the 'λ'-polynomial and application of required corrections

The 1st step (S) can be split further in two parts: S_1) the computation of syndrome values over the message-portion of codewords and S_2) the computation of syndrome values over the parity-portion of codewords. The former (S_1) requires 4 dedicated GF-mult operators that are continuously busy. The later (S_2) requires 1 GF-mult operator, which is however occupied only for 16 clock-cycles after all relative parity-symbols are extracted from the SDH overhead and become available to the unit for computations. On the other hand, the 2nd and 3rd steps (L, C) are executed in succession, requiring 1 GF-div each.

Figure 4 displays the timing-relationships between busy-periods of the aforementioned GF-mult and GF-div operations, using codeword (row-triplet) No 1 as an example. More specifically, point [1] signifies the beginning of syndrome-values computations over parity-data (S_2), which is also the deadline for error corrections of the previous codeword (i.e. row-triplet No 3). At point [2], the syndrome-values are available to the 2nd step (L) for the formation of λ(x). Label [3] points at the beginning of the 3rd step (C) during which all necessary corrections are applied. Finally, point (4) indicates the time-slot, when the last error-correction within this codeword takes place.

This timing diagram obviates that the GF-arithmetic operators involved in steps S_2, L and C are used consecutively, not concurrently. It is sensible therefore

that a GF-arithmetic unit is shared between these three steps, in order to minimize area-requirements. A parallel GF-mult/div architecture (of type 'C') can switch at will between the role of multiplier and divider, as explained in the previous subsection. By employing this architecture, the area-cost of 1 GF-mult and 2 GF-div is reduced to the cost of just 1 GF-mult. The importance of the above optimization can be further appreciated by noting that these savings extend over all 16 parallel-processing subchannels i.e. the benefit scales by a factor of 16.

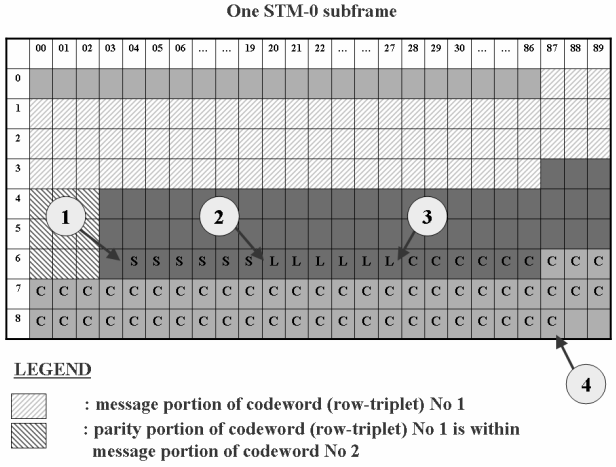

Figure 4: Timing chart for the shared GF-arith. unit

A subtle aspect of the above optimization is that normally, decoding step (S_2) requires as many GF-mult as (S_1) i.e. 4. However, only 1 GF-mult operator was mentioned in the preceding paragraphs for S_2 and this is precisely what made possible the sharing of this arithmetic unit with decoding-steps L & C, which involve 1 GF-div operator each. The cost-reduction within (S_2) from 4 GF-mult to 1 was achieved by serializing the computation of syndrome-values at the expense of a quadruple (4x) computational time delay ('strong'-mode). In other words, the dedicated GF-mult operator is not only shared between 12 STM-0's in the same subchannel, but also between 4 syndromes in the same partial decoder i.e. same STM-0.

The challenge in this case was to assure that the increased processing-time in decoding steps S_2, L and C, incurred by sharing the GF-mult in (S_2) as explained above, does not exceed the duration of one codeword (3 SDH rows), which constitutes the ultimate deadline. As indicated by point [4] in Figure 4, corrections are indeed completed before the deadline, which for codeword No 1 is at SDH row 0 and column 3. This is achievable, because FOCUS processes 9-bit GF-symbols, in contrast to the 8-bit octets of the SDH protocol. More specifically, the FOCUS decoder has been customized, such that when the 2nd decoding step (L) is completed, the 3rd one (C) initiates a burst-correction at 9/8 the speed of SDH processing. Although non-obvious, the timing-condition holds indeed and the optimization is possible.

Another valuable optimization relates to the evaluation of polynomials over GF(2^9), as for instance in the case of the 'ω' polynomial [4] in Forney's formula (3rd decoding step). A polynomial P(x) of degree d:

$$P(x) = c_d * x^d + c_{d-1} * x^{d-1} + \cdots + c_1 * x^1 + c_0$$

involves d GF-mult (*) operations between the coefficients c_i and the powers of the independent variable x^i, i=1,2,…,d. A parallel GF-mult architecture is assumed for each of the above multiplications; it can therefore be broken into a 'bit-polynomial multiplication' 1st stage and a subsequent 'normalization' 2nd stage, as of Figure 1. Normalization, however, turns out to be distributive over addition and as such, can be shared among all GF-mult operations within the evaluation of P(x). Then:

$$P(x) = \left\| c_d \bullet x^d + c_{d-1} \bullet x^{d-1} + \cdots + c_1 \bullet x^1 + c_0 \right\|$$

where (●) denotes 'bit-polynomial multiplication' (1st stage) and (||…||) denotes 'normalization' (2nd stage).

2.3. Redundant logic elimination

The nominal maximum degree of the error-corrector polynomial (ω) equals the number of syndrome values in use [5], which are 4 in the 'strong'-mode of FOCUS and 2 in the 'weak'-mode. This polynomial is in fact the one with the highest degree that needs to be evaluated as part of the FOCUS decoding. It has been verified by means of extensive digital-simulations that in the absence of decoding failures, the actual degree of ω(x) is half as much at most, the most-significant coefficients of ω(x) being zeroes. This observation relieves the implementation of FOCUS 'strong'-mode from:

- 2 GF-mult' 1st stage (●) operators, and
- 2 constant-exponent GF-exp (x^i) operators

per processing-subchannel, in total: 32 GF-mult and 32 GF-exp needless operators.

More redundant logic was found and eliminated in the backend component dealing with decoding failures. As stated already in the introductory section, decoding failures are a common property of all FEC methods. They manifest themselves, when the channel generates more errors than the FEC system is designed to handle. Their effect is to even further deteriorate the signal quality (error-multiplication) through damaging corrections. A major criterion to detect and possibly, avoid decoding failures is to disallow multiplicity of the roots of λ(x).

Generally, the detection of decoding failures in Reed-Solomon decoders takes place as part of the backend 'Chien-Search' algorithm [5], which compares the roots' count with the degree of λ(x). It is noteworthy however that a degree-2 polynomial over GF(2^9) with a single double-root (instead of 2 different ones) takes the form:

$$\lambda_{DF}(x) = (a * x + b)^2 = a^2 * x^2 + b^2$$

i.e. the middle power coefficient is null, which is in contrast with real-number arithmetic, where:

$$p(x) = (a \cdot x + b)^2 = a^2 \cdot x^2 + 2 \cdot a \cdot b \cdot x + b^2$$

$\lambda_{DF}(x)$ is the case of FOCUS 'strong'-mode; decoding failures are therefore immediately recognizable by a zero 1st power coefficient and a non-zero 2nd power coefficient. In the 'weak'-mode of FOCUS, the $\lambda_{DF}(x)$ polynomial is degenerated to degree-0, which does not have roots and cannot be of use anyway.

By immediately identifying decoding-failures, the correction-algorithm is not initiated at all, sparing the damaging corrections. This is a distinctive feature of FOCUS 'strong'-mode. On the contrary, general Reed-Solomon decoders [6] have to dully count the roots of $\lambda(x)$, while 'Chien-Search' is in progress, which implies that:

- Either: damaging corrections are applied, before the criterion has a chance to indicate the failure
- Or: the client-data must be subjected to double the ordinary delay, waiting for the criterion to decide

FOCUS has been designed to avoid both of the above.

A primary benefit in terms of complexity is the elimination of the middle 1st power coefficient in $\lambda_{DF}(x)$, which translates to 1 redundant GF-mult operator (b=1 is assumed to be constant). Furthermore, the detection of decoding-failures reduces to simple comparisons with zero and lastly, a counter is no-more required for the roots. The benefit is for once again scaled by a factor of 16, due to the existence of 16 parallel-processing subchannels.

Through the optimizations, listed and explained in the preceding sections and paragraphs, a prototype of FOCUS 'strong'-mode was eventually obtained, using a large FPGA as the target device. Nevertheless, provision has been made also for applications that cannot tolerate the complexity of FOCUS 'strong'-mode. In such cases, a trade-off can be made between complexity and performance. More specifically, the corrective performance of FOCUS can be deliberately halved, by ignoring the upper half of the available syndrome-values. In return, the degree of all polynomials involved in FOCUS decoding is halved and the overall complexity is halved as well. This trade-off gives rise to the so-called 'weak'-mode of FOCUS operation [2], [3].

3. Conclusions

The efforts to obtain a prototype for our inband-FEC, FOCUS, have yielded substantial complexity-reductions through numerous, sophisticated optimizations. This paper described in detail these optimizations. Firstly, particular attention has been paid to the design of arithmetic operators: Itself the underlying Galois-Field arithmetic has been carefully chosen as to minimize the cost of operators, primarily multiplication and division, and the primitive-elements of the target-device have been exploited to the maximum possible extent. Additionally, resource-sharing has been heavily employed in many aspects of the design: For SDH STM-64 signals, every partial FOCUS decoder is shared between 12 constituent STM-0 subframes. Within each FOCUS decoder, operators are shared between the decoding-steps. In polynomial evaluations, parallel multipliers share their normalization stage. Finally, redundant logic was discovered and eliminated: Evaluation of the highest-degree polynomial (ω) has been drastically simplified. Detection of decoding-failures has been improved, bringing further benefits in terms of area.

To the best of our knowledge, this is the first implementation of iFEC for SDH/SONET networks that investigates trade-offs on complexity. It is noteworthy that no evidence on implementations of the corresponding ITU-T G.707 rec. iFEC could be found, except of a brief mention in [6]. In addition, we could not find feedback on the implementation-cost of GF(2^9) arithmetic operators. We do not have, therefore, a suitable reference to compare our results with. Nevertheless, the efficiency of the final FOCUS design can be demonstrated through the results from the implementation-software, which is the Xilinx® ISE™ version 8.1.2 MAP-tool. The above cost represents not only the full-scale 'strong'-mode FOCUS-decoder for SDH STM-64, but also an essential STM-64 deframer, the micro-controller I/F and a 128-bit parallel verification unit for PRBS(2^{31}-1) test-sequences (in the payload). The final implementation-cost per partial FOCUS-decoder turns out to be impressive (Table 1, 4th column):

Table 1: Implementation cost of the FOCUS Inband-FEC decoder for SDH STM-64 / SONET OC-192

primitive	total	utilization	/ 192
LUT	20832	72%	108.50
B-RAM	96	100%	0.50
Mult18x18s	84	87%	0.44
T-BUF	1282	17%	6.68

4. References

[1] M. Tomizawa et Al. "Forward Error Correcting Codes in Synchronous Fiber Optic Transmission Systems". Journal of Lightwave Technology, Vol. 15, Jan 1997, pp. 43-52.

[2] A. Tychopoulos, O. Koufopavlou, "In-Band Coding Technique to Promptly Enhance SDH/SONET Fiber-Optic Channels with FEC Capabilities", European Trans. Telecomm, Vol. 15, April 2004, pp. 117 – 133.

[3] A. Tychopoulos et Al, "Demonstration of a Low-cost Inband FEC Scheme for STM-64 Transparent Metro Networks", in Proc. IEEE ICTON 2006, Tu.A3.4.

[4] Key-papers in the Development of Coding Theory, E. R. Berlekamp, 1974, IEEE Press, New York.

[5] Algebraic Coding Theory, Elwyn R. Berlekamp, 1968, McGraw-Hill, ISBN 07-004903-3.

[6] K. Azadet et Al, "Equalization and FEC Techniques for Optical Transceivers", IEEE Journal of Solid State Circuits, Vol. 37, Mar 2002, pp 317-327.