Advances in Radio Science, 3, 271–276, 2005
SRef-ID: 1684-9973/ars/2005-3-271

Advances in Radio Science

Implementation and modeling of parametrizable high-speed Reed Solomon decoders on FPGAs

A. Flocke, H. Blume, and T. G. Noll

Chair of Electrical Engineering and Computer Systems, RWTH Aachen University, 52062 Aachen, Germany

Abstract. One of the most important error correction codes in digital signal processing is the Reed Solomon code. A lot of VLSI implementations have been described in literature. This paper introduces a highly parametrizable RS-decoder for FPGAs. By implementing resource-sharing and by using a fully pipelined multiplier/adder-unit in GF(2^m) it was possible to achieve high throughput rates up to 1.3 Gbit/s on a standard FPGA, while using only an attractive small amount of logical elements (LE). The implementation, written in a hardware description language (HDL), is based on an inversionless Berlekamp Algorithm (iBA), whose structure leads to a chain of identical processing elements (PE). The critical path of one PE runs only through one adder and one multiplier. A detailed description of a resource-sharing methodology for this Berlekamp Algorithm and the achievable gain are presented in this paper.

The benchmarking for the design was done for different 8bit-codes against state-of-the-art FPGA-solutions and showed a gain of up to a factor of six regarding the AT-product, compared to other implementations.

1 Introduction

Today's applications for digital signal processing become more and more demanding, aiming for higher throughputs on the one hand and for lower power and smaller devices on the other hand. Efficient algorithms combined with state-of-the-art silicon technology are essential to cope with the requirements of modern digital signal processing applications. Besides processing power and energy efficiency, flexibility is a critical factor as well. Consumer electronics have to be market-ready, while details for technical standards may still change or it is not clear, which one of two competing standards has to be supported. Reconfigurable hardware, e.g. embedded FPGA-cores in Systems on Chip (SoC), can be one attractive compromise between flexibility on the one hand and efficiency on the other hand.

Correspondence to: A. Flocke
(flocke@eecs.rwth-aachen.de)

An important objective in digital signal processing is the protection of data against errors. The Reed Solomon (RS)-Code, developed in 1958 by I. S. Reed and G. Solomon, is a key component for fault-tolerant data communication. It is particularly suitable for the correction of burst errors and hence used for example to protect data on CDs and DVDs or in network communications. RS applications cover various fields from e.g. coding graphical address stickers with six bit symbol width, over 172 kB/s audio CD decoding, up to 10 Gbit/s high speed optical cable communication.

The algorithm presented in this paper is based on a four-step approach: Transformation of the codeword to the frequency domain, computation of the error locator- and evaluator-polynomial, back transformation to the time domain and computation of the correction values, that are finally superposed with the received codeword. The decoding is done in the frequency domain by a reformulated inversionless Berlekamp Algorithm (Blahut, 1992; Sarwate and Shanbhag, 2001). This structure is highly parametrizable and thus very suitable for this implementation.

In this generic implementation the values for symbol size ($2 \leq m \leq 9$), code word length $k < 2^m - 1$ (shortened codes are supported) and error correction capability $t \leq \lfloor (2^m - 1 - k)/2 \rfloor$ (see Fig. 1) can be changed to any possible combination within the limitations given above, where larger values than m=9 are possible, but have not been analyzed in this work. The primitive polynomial can be altered as well as the generator polynomial to meet certain RS-standard requirements.

By optimizing the basic design, implementing resource-sharing and pipelining, the number of utilized LEs dropped significantly, leading to a high performance implementation. The methodology for implementing the optimized structure, using resource-sharing, and the achievable gain are described in detail in this paper.

This paper is structured as follows: Sect. 2 is sketching briefly the principle functionality of a RS-decoder. In Sect. 3 the optimization for an FPGA implementation is described and Sect. 4 presents the results and benchmarks for different RS-decoder implementations.

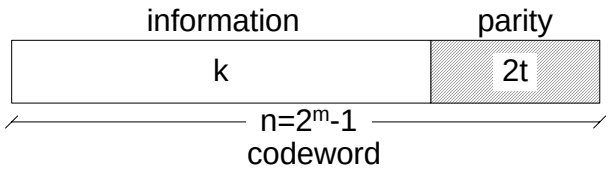

Fig. 1. RS codeword.

2 Algorithm

Derived from BCH-codes (Clark and Cain, 1981; Hamming, 1980), RS-codes (Reed and Chen, 1999) are able to correct symbols instead of bits. As all bits in a symbol can be corrupt, the decoding algorithm not only has to identify the wrong symbols but also has to restore the original value, whereas binary codes only have to invert the corrupt bit. Thus, instead of simple Boolean operations, Galois-Field arithmetic (Blahut, 1992; Clark and Cain, 1981; Roman, 1992) has to be used to compute the two dimensional codewords of a symbol code. In Fig. 2 the complete decoding process of an RS-decoder is depicted. Three different types of information have to be extracted from a received codeword: The number of errors and, if errors have occurred, their positions in the codeword and their values. The corrupt input data r is stored in a FIFO-RAM and simultaneously sent to the syndrome computation, where the syndromes of the error pattern are separated from the information part of the codeword. With these syndromes S the error locations and magnitudes can be determined. Therefore, the key equation has to be solved. The Chien-Search locates the positions of incorrect symbols, by finding the zero points of the error locator polynomial Λ. The Forney-Algorithm finally computes the correction pattern out of the error evaluator polynomial Ω and the odd coefficients of Λ. This pattern is added to the delayed input and results in an error-free codeword, if not more than t errors have occurred. Otherwise the decoding process will fail. It depends on the error pattern, whether this failure is detectable or not.

2.1 Syndrome Computation

Assuming a corrupt transmission, the received codeword r consists of the original codeword y which is superposed by the error pattern e:

$$r_j = y_j + e_j \,.$$

The syndrome S is defined as the multiplication of the transpose of a received word r with the parity check matrix $\mathbf{H}$ (Hamming, 1980)

$$S^T = \mathbf{H} \cdot r^T = \mathbf{H} \cdot \left(y^T + e^T\right) = \mathbf{H} \cdot e^T$$

and depends only on the error pattern e. Practically, this operation is a partial transformation of the codeword to the frequency domain. The computation of the syndrome follows Horner's rule. Each syndrome is computed by:

$$S_i = r\left(\alpha^i\right) = \sum_{j=0}^{2^m-2} r_j \alpha^{ij}$$
$$= r_0 + \alpha^i \left(r_1 + \alpha^i \left(\ldots + \alpha^i \left(r_{n-1}\right)\right)\right) \mid i = 0, 2, \ldots, 2t-1 \quad .$$

This structure describes a recursive operation, that multiplies and accumulates a constant value α^i with the input data $r_{j=0\ldots2^{-2}}^m$. The $2t$ (where t is the number of maximum correctable errors) syndromes are computed in parallel by one MAC (Multiply ACcumulate)-unit each. After the whole code word is processed, all syndromes have been calculated simultaneously.

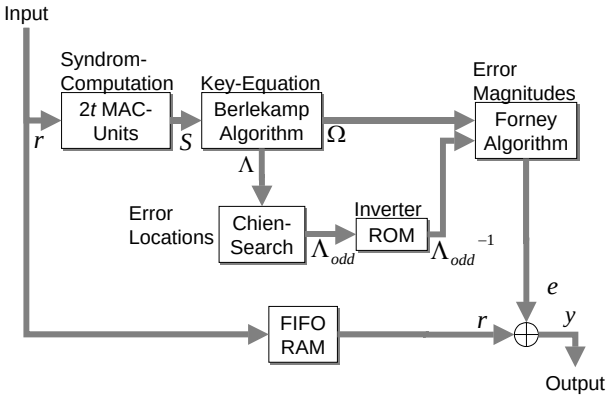

Fig. 2. Block diagram of a RS decoder .

2.2 Solving the Key Equation

The main component of an RS-decoder is the key equation block. It solves a set of $2t$ linearly dependent equations

$$\Omega(X) = \Lambda(X) \cdot \mathbf{S}(X) \;\Big|_{\bmod X^{2t}}$$

with the two polynomials $\Lambda(X)$, that describes the error locations, and $\Omega(X)$ that identifies the error magnitudes. There are numerous algorithms described in literature that find the minimum-degree solution to the equation above. One of the fastest and hence often preferred algorithm is the so called "Berlekamp Massey Algorithm" (BMA) that solves $\sum_{\nu}^{2t-1} \Lambda(X) \cdot \mathbf{S}(X) = 0$, where $\nu \leq t$ is the number of errors that have occurred. For this implementation a further development of the BMA, the "Berlekamp Algorithm" (BA), that concurrently computes Λ and Ω, is used.

2.2.1 Berlekamp Algorithm

The problem of finding the minimum-degree solution to the key equation is the same as trying to find the smallest Linear Feedback Shift Register (LFSR) $\Lambda(X)$, that generates the first $2t$ terms of S. The initial LFSR that predicts S_1 from S_0 is tested, whether it can predict S_2 as well. When it passes the test, it is still the best solution and the test continues with

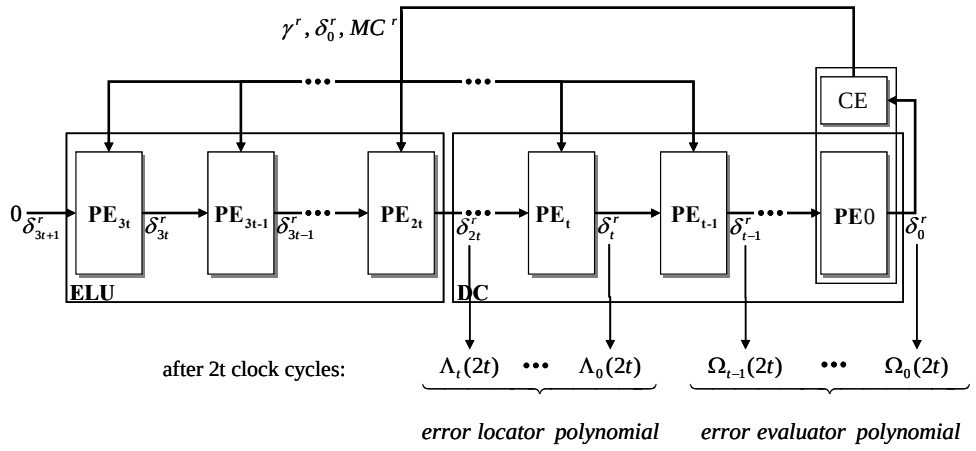

Fig. 3. iBA structure.

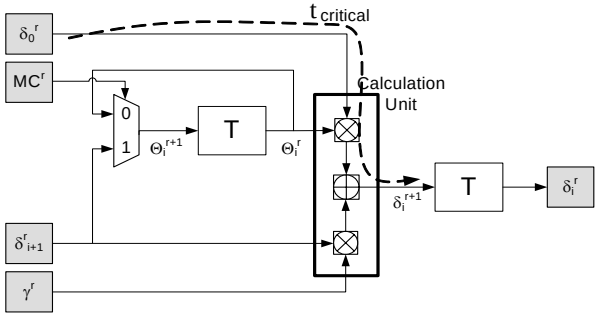

Fig. 4. Schematic of a processing element.

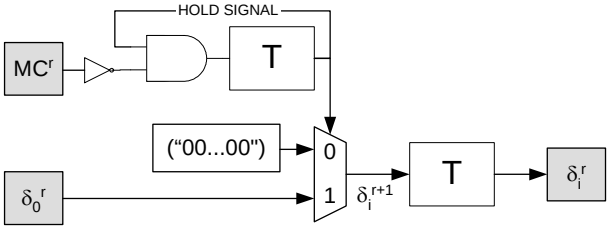

Fig. 5. Schematic of PE$_{3t}$.

the successive syndromes, until it fails. At this point, the registers have to be modified in such a manner that the next syndrome is predicted correctly and the previous predictions do not change. Furthermore, the length of the LFSR should increase by the least possible amount.

Starting from the initial values $\Lambda_0(X)$=1 and $\Omega_0(X)$=0 the BA computes the solution for the key equation in r=2t steps by calculating a so called discrepancy polynomial

$$\Delta_r(X) = \sum_{j=0}^{\deg \Lambda_{r-1}} \Lambda_{r-1}^j \cdot S_{r-j}$$

and then updating Λ and Ω as necessary by solving

$$\begin{bmatrix} \Lambda_r(X) \\ B_r(X) \end{bmatrix} = \begin{bmatrix} 1 & -\Delta_r X \\ \Delta_r^{-1}\delta & (1-\delta)X \end{bmatrix} \cdot \begin{bmatrix} \Lambda_{r-1}(X) \\ B_{r-1}(X) \end{bmatrix}$$
$$\begin{bmatrix} \Omega_r(X) \\ A_r(X) \end{bmatrix} = \begin{bmatrix} 1 & -\Delta_r X \\ \Delta_r^{-1}\delta & (1-\delta)X \end{bmatrix} \cdot \begin{bmatrix} \Omega_{r-1}(X) \\ A_{r-1}(X) \end{bmatrix},$$

(for a detailed description refer to Blahut (1992), Clark and Cain (1981) and Berlekamp (1968). The problem with both algorithms (BMA and BA) is the multiplication with the inverse discrepancy polynomial $\Delta_r^{-1}(X)$ that has to be performed in every step of the computation, when the polynomials Λ_r and Ω_r are updated. This division in the GF-domain needs a lot of resources, and furthermore, the BA leads to an irregular structure that is very hard to implement in a parametrizable RS decoder. Sarwate and Shanbhag (2001) introduced an architecture for the BA, where the division is replaced by multiplications. Thus the BA can be described in a very regular structure of identical processing elements with a small control unit. This composition leads to a parametrizable and efficient implementation and hence was used for this work.

2.3 Chien Search and Forney Algorithm

With the known error locator polynomial it is possible to determine the error locations by finding the zero positions of

$$\lambda_i = \sum_{j=0}^{2t} \Lambda_j \cdot \alpha^{j\cdot(n-i)} = \sum_{j=0}^{2t} \Lambda_j \cdot \alpha^{jk}\ , k = 0, 1, 2, \ldots, (n-1)\,.$$

To finally compute the correction pattern, the error evaluator polynomial is analyzed. The error pattern e is derived from the following formula:

$$e_i = \begin{cases} 0 & if\ \Lambda\left(\alpha^i\right) \neq 0 \\ \alpha^{-ik} \cdot \frac{\Omega(\alpha^i)}{\Lambda_{odd}(\alpha^i)} & if\ \Lambda\left(\alpha^i\right) = 0 \end{cases}.$$

The division can be performed by inverting the signal Λ_{odd}, using a ROM, and a succeeding multiplication of the two signals.

Table 1. Calculation for constant data rate.

	syndrome computation block	**key equation block**
data rate	$D\ {}^{symbols}/_{s}$	$D\cdot\frac{2t}{2^m-1}\ {}^{syndromes}/_{s}$
computational complexity	$2t\cdot(\otimes+\oplus)\ {}/_{symbol}$	$(3t+1)\cdot(2\otimes+\oplus)\ {}/_{syndrome}$
computing power	$D\cdot 2t\cdot(\otimes+\oplus)$	$D\cdot\frac{2t}{2^m-1}\cdot(3t+1)\cdot(2\otimes+\oplus)$
PE computing power	$PE_{syndrome}:=(\otimes+\oplus)$	$PE_{key}:=(2\otimes+\oplus)$
# of PEs per block	$\#PE_{syndrome}=2t$	$\#PE_{key}=\left\lceil\frac{2t\cdot(3t+1)}{2^m-1}\right\rceil$

3 Optimization for FPGA

To map the theoretical formulas of an RS-decoder on an implementation feasible for FPGAs, the inversionless BA, mentioned previously, has been used for this design. Furthermore, the regular structure of the iBA has been modified for resource-sharing. By implementing a fully pipelined multiplier/adder-unit (also referred to as the calculation unit), the maximum clock frequency could be increased.

The optimizations described in this work are examplarily shown for an Altera APEX20KE FPGA with LEs (logic elements), containing primarily a four bit input LUT (look-up-table) and an optional register.

3.1 Inversionless BA

The iBA architecture, presented in Sarwate and Shanbhag (2001), replaces the division by multiplications and leads to an arrangement of a chain of identical processing elements (PE) and only one irregular control element (CE). A chain of $3t+1$ PEs, consisting of an error locator update (ELU) and a discrepancy computation (DC) block, compute the values for Λ and Ω. In order to avoid the inversion, instead of Λ and Ω shifted polynomials $\beta\cdot\Lambda$ and $\beta\cdot\Omega$ are computed. The result of the Chien Search will not be affected by that and the Forney Algorithm can be adapted by implementing a constant multiplication to the formula. Hence, the polynomials will still be referred to as Λ and Ω. The structure of the iBA is shown in 2.3. The PEs are initialized with a polynomial

$$\delta_i=\begin{cases}S_i\ for\ 0\le i\le 2t-1\\ 0\ for\ 2t\le i\le 3t-1\\ 1\ for\quad i=3t\\ 0\ for\quad i=3t+1\end{cases}$$

and compute the error locator and the error evaluator polynomials in $2t$ clock cycles. Every clock cycle the computations

$$\delta_i^{r+1}=\gamma^r\cdot\delta_{i+1}^r+\Theta_i^r\cdot\delta_0^r$$

$$\Theta_i^{r+1}=\begin{cases}\delta_{i+1}^r\ if\ MC^r=1\\ \Theta_i^r\ \ if\ MC^r=0\end{cases}$$

(Θ is a help polynomial) are carried out for $0\le i\le 3t$, where the initial values are $\Theta_i^0=\delta_i^0$ for $0\le i\le 3t$ and $\gamma^0=1$. The control element computes γ^r and k^r:

$$\begin{matrix}MC^r=1\Rightarrow\gamma^{r+1}=\delta_0^r\text{ and }k^{r+1}=-(k^r+1)\\ MC^r=0\Rightarrow\gamma^{r+1}=\gamma^r\text{ and }k^{r+1}=k^r+1\end{matrix}.$$

The control signal MC^r is 1, if $\delta_0^r\neq 0$ and $k^r\ge 0$, and 0 if otherwise. The internal counter k^r of the CE is initialized with $k^0=0$. The Control Element is combined with the first PE (PE_0), as the internal signal δ_0^{r+1} is needed to compute the control signal MC^{r+1}.

The processing element and its main part, the calculation unit, are shown in Fig. 4. The critical path of the complete iBA algorithm passes through one $GF(2^m)$ multiplier and one $GF(2^m)$ adder of this unit.

The last processing element can be simplified, because the second multiplication is always zero ($\delta_{3t+1}^0=0$). As shown in Fig. 5, the computation of Θ_i^{r+1} is reduced to a hold signal, that becomes and stays low, when MC^r is high for the first time. This simplified design uses less than 10% of the number of LEs that a normal PE uses. The area-reduction is noticeable in the complete design, if only a few PEs are used (see Sect. 3.2).

3.2 Resource-sharing

For an optimal implementation of the decoding process, it is important, that the data rates of the four consecutive blocks match. Otherwise, blocks that can run at a higher data rate will be idle for a certain amount of time. Let the data rate for the syndrome computation block be D symbols/s. For every codeword, $2t$ syndromes are computed, so the data rate for the key equation block is $D\cdot\frac{2t}{2^m-1}\ {}^{syndromes}/_{s}$. With the computational complexity according to Sect. 2.1 and Sect. 3.1 ($\oplus$ denotes a GF addition, $\otimes$ a GF multiplication), the required computing power for an operation is the product of computational complexity and data rate. Finally, the required number of PEs can be derived as shown in Table 1.

For example for an 8bit-Code with t=8 the number of MAC units for the syndrome computation is $\#PE_{syndrome}$=16. For the key equation with the iBA,

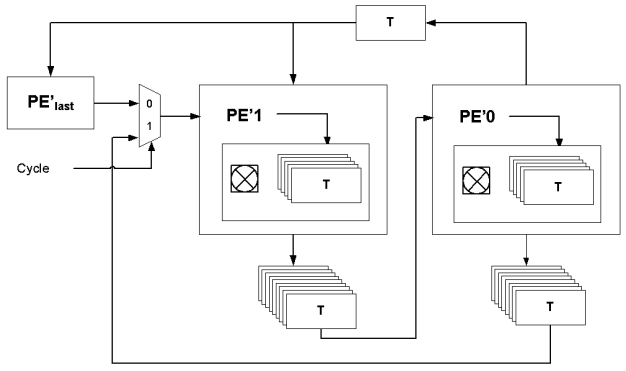

Fig. 6. Schematic of iBA (resource-sharing).

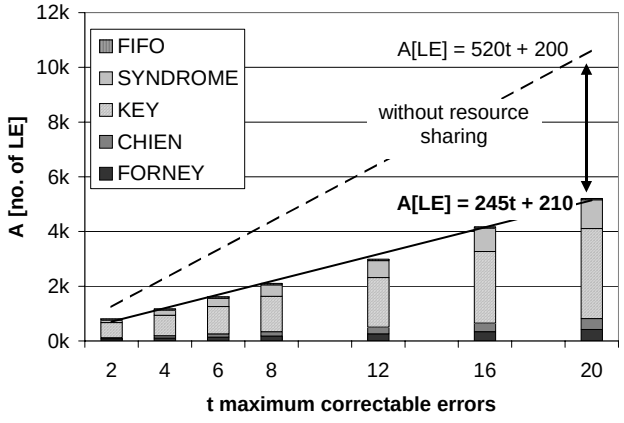

Fig. 7. Reduction of utilized LEs for 8-bit codes.

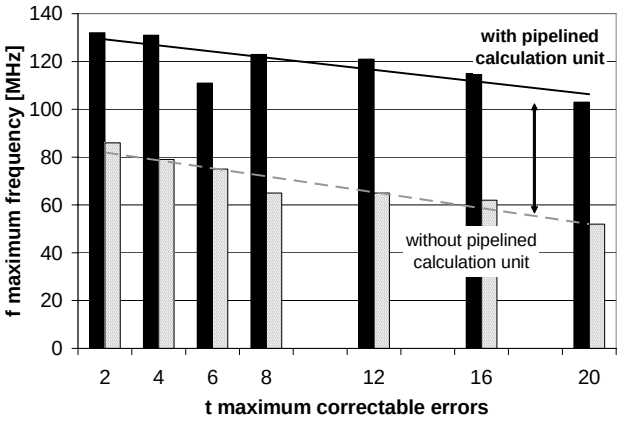

Fig. 8. Maximum clock frequency (8-bit-codes).

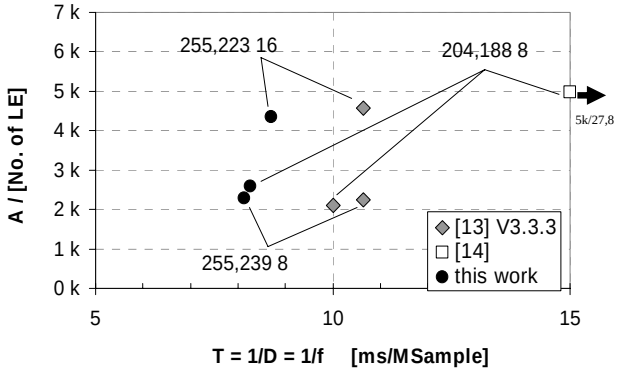

Fig. 9. Comparison with other FPGA IP-cores.

$\#PE_{key}$=2 processing elements are required for a constant data rate. So instead of $\#PE_{key,\,iBA}$=25, there will be only two PEs, calculating the equations given in Sect. 3.1. The complete implementation is shown in Fig. 6. The intermediate results have to be stored in a register chain. Some of these registers can be used to pipeline the critical path through the calculation unit. The small processing element PE'$_{last}$ is added to the chain in order to save a normal PE' in some other codes of the generic design space. The control unit is integrated in the PE'0, the control signals are then fed back to the other PE's.

4 Results

The design flow, used for the following results, involves mainly two programs: Scirocco from Synopsys for logical simulation of the VHDL-code on a functional level and Quartus II from Altera for synthesis, place and route, gate level simulation and device programming. The code was tested on hardware with an APEX 20K300EQC240-1, that was analyzed with an 16702A logic analyzer from HP. All values concerning the number of maximum clock frequency and the number of utilized logic elements are evaluated by the Quartus software.

In real systems maximum throughput rate is not always a key objective. Usually, a given specification has to be met (e.g. real time applications or a certain bit rate). Nevertheless, the maximum achievable throughput rate can be a valuable information as time sharing or parallelization-concepts can be applied to the implementation.

4.1 Optimized RS-Decoder with Resource-sharing

The improvement, achieved by the implementation of resource sharing is shown exemplarily for 8-bit codes, which are mostly used in practice.

Compared to the code without resource-sharing, the AT-product increases in average by a factor of four (Fig. 7, Fig. 8). The maximum clock frequency increased only by a factor of two, though the critical path through the LEs could be fully pipelined. This is due to the fact, that the critical path of the whole decoder was then determined by the inverter-ROM, which can not be further pipelined.

The maximum achievable throughput rate is roughly 1Gbit/s on an APEX 20KE device. On a standard Stratix FPGA the maximum throughput for an 8-bit code with eight correctable errors (255,239,8) is about 1.3 Gbit/s.

When the optimized code is compared against other Reed-Solomon decoder implementations, the realization introduced in this paper shows comparable qualities in terms of LE utilization and maximum clock frequency.

In Fig. 9, three different codes (m=8) with t=8, 16 are compared with other state-of-the-art solutions (Website Altera, Website Amphion). Two codes (255,239,8), (255,223,16) are compliant to CCSDS (Consultative Committee for Space Data Systems) standards and a shortened code (204,188,8) is compliant to the DVB (Digital Video Broadcasting) standard. All values for Website Altera and the implementation presented in this paper have been compiled for an APEX 20 KE device. The values for the DVB code have been taken from the data sheet (Website Amphion).

5 Summary

In this paper a highly parametrizable RS-Decoder for FPGAs has been introduced. The implementation, based on an inversionless Berlekamp Algorithm for solving the key equation, achieves high throughput rates while consuming, compared with other FPGA-implementations, only a small amount of logical elements. The maximum throughput was about 1 Gbit/s on an APEX 20 K300E device with speed grade 1.

References

Altera Corporation: Integrating Product-Term Logic in APEX20K Devices, Application Note 112, http://www.altera.com/literature/an/an112.pdf, 1999a.

Altera Corporation: Designing with ESBs in APEX II Devices, Application Note 179, http://www.altera.com/literature/an/an179.pdf, 1999b.

Berlekamp, E. R.: Algebraic Coding Theory, McGraw-Hill, 1968.

Blahut, R. E.: Algebraic Methods for Signal Processing and Communications Coding, Springer-Verlag, 1992.

Clark, G. C. and Cain, J. B.: Error-Correction Coding for Digital Communications, Plenum Press, 1981.

Hamming, R. W.: Coding and Information Theory, Prentice-Hall, 1980.

Reed, I. S. and Chen, X.: Error-Control Coding for Data Networks, Kluwer Academic Publishers, 1999.

Roman, S.: Coding and Information Theory, Springer-Verlag, 1992.

Sarwate, D. V. and Shanbhag, N. R.: High-Speed Architectures for Reed-Solomon Decoders, IEEE Transactions VLSI Systems, Vol. 9, No. 5, October 2001.

Takagi, N., Yoshiki, J., and Takagi, K.: A Fast Algorithm for Multiplicative Inversion in GF(2^m) Using Normal Basis, IEEE Transactions on Computers, Vol. 50, No. 5 Mai, 394–398, 2001.

Veendrick, H.: Deep-Submicron CMOS ICs From Basics to ASICs, Kluwer academic publishers, 338–339, 2000.

Website Altera: http://www.altera.com/products/ip/ipm-index.html.

Website Amphion: http://www.amphion.com/cs3210.html.

Wilhelm, W.: A New Scalable VLSI Architecture for Reed-Solomon Decoders, IEEE Journal of Solid-State Circuits, Vol. 34, No. 3, 388–396, March 1999.