Adv. Radio Sci., 10, 175–181, 2012
www.adv-radio-sci.net/10/175/2012/
doi:10.5194/ars-10-175-2012

Advances in Radio Science

Design space exploration of high throughput finite field multipliers for channel coding on Xilinx FPGAs

C. de Schryver, S. Weithoffer, U. Wasenmüller, and N. Wehn

Microelectronic Systems Design Research Group, University of Kaiserslautern, Erwin-Schrödinger-Str., Germany

Correspondence to: C. de Schryver (schryver@eit.uni-kl.de)

Abstract. Channel coding is a standard technique in all wireless communication systems. In addition to the typically employed methods like convolutional coding, turbo coding or low density parity check (LDPC) coding, algebraic codes are used in many cases. For example, outer BCH coding is applied in the DVB-S2 standard for satellite TV broadcasting. A key operation for BCH and the related Reed-Solomon codes are multiplications in finite fields (Galois Fields), where extension fields of prime fields are used. A lot of architectures for multiplications in finite fields have been published over the last decades. This paper examines four different multiplier architectures in detail that offer the potential for very high throughputs. We investigate the implementation performance of these multipliers on FPGA technology in the context of channel coding. We study the efficiency of the multipliers with respect to area, frequency and throughput, as well as configurability and scalability. The implementation data of the fully verified circuits are provided for a Xilinx Virtex-4 device after place and route.

1 Introduction

Galois or finite field multiplications (FFMs) are key in a wide range of technical applications, for example in cryptography or channel coding. In particular BCH- and Reed-Solomon codes require a lot of Galois field (GF) multiplications to be performed in the decoding process. Nowadays, BCH codes are widely spread and employed in modern communication standards like DVB and in error correction units for flash memory devices (Liu et al., 2006), for instance. In Sect. 2 we summarize the theoretical background of FFM and BCH decoding.

However, the specific FFM requirements vary over the application range: high Galois field dimensions are used in cryptography, whereas for FFMs in channel coding the focus lies on high throughput and low latency. FFMs are key components in BCH decoder implementations and consume a significant amount of hardware resources in the overall design (Chen et al., 2009; Zhang et al., 2010).

Up to now, a lot of research has been made on area efficient and fast FFM architectures. However, Ahlquist et al. haven shown in 1999 (Ahlquist et al., 1999) that not every VLSI optimized multiplier architecture performs optimal on FPGAs.

Already in 1986, Scott et al. have proposed a scalable FFM that is easily adaptable to different GF sizes and primitive polynomials (Scott et al., 1986). Their bit-slice architecture makes efficient use of area providing flexibility for different field sizes. Building on the work of Scott et al., Kitsos et al. in 2003 proposed a flexible, reconfigurable FFM architecture for extension fields $GF(2^m)$ (Kitsos et al., 2003). Their bit-serial polynomial basis multiplier features reconfigurability for field sizes up to m and reduced power consumption due to gated clocking. The low hardware complexity promises good area performance.

In this work, we investigate the potential benefit of these published FFM architectures as well as two "standard" architectures for a high-throughput DVB-S2 standard BCH decoder. For this purpose we have implemented the selected FFMs as described in Sect. 3 and fully validated them with a hardware-software co-simulation setup. We have synthesized our implementations and compare area consumption and achievable throughput for a common Xilinx Virtex-4 FGPA target device as described in Sect. 4. Furthermore, we evaluate them with respect to their suitability for FPGA target devices, area consumption, latency and scalability. Finally, we comment on the impact of using sophisticated pipelined FFMs in a DVB-S2 BCH decoder and show which parts can profit most from those.

2 Theoretical background

For a better understanding of the finite field multiplier architectures and the context of BCH coding, we give an overview on the underlying mathematics in this section. For more details we refer to available standard literature (Bossert, 1998; Friedrichs, 1995).

2.1 Finite fields and multiplication

The basis for the finite field multiplication architectures implemented in this work is a reformulation of the finite field multiplication in polynomial representation. The derivation of the algorithm originally proposed by Scott et al. (Scott et al., 1986) is outlined in the following.

The nonzero elements of the extension field $GF(2^m)$ can be constructed by powers of the primitive element α, where α is a root of a primitive polynomial $P(x) = x^m + p_{m-1}x^{m-1} + \cdots + p_1 x + p_0$ over $GF(2)$. Since $P(\alpha) = 0$ (α being root of P) $\alpha^m = p_{m-1}\alpha^{m-1} + \cdots + p_1\alpha + p_0$ and therefore nonzero the elements can be written in the canonical base representation:

$$\{a_{m-1}\alpha^{m-1} + \cdots + a_1\alpha + a_0 | a_i \in GF(2) \text{ for } 0 \le i \le m-1\}$$

2.1.1 Multiplication algorithm

Based on the work of Scott et al., an algorithm for FFM can be derived similar to the design proposed by Kitsos et al. as follows: Let A, $B \in GF(2^m)$ in canonical base representation and P primitive polynomial over $GF(2)$. Then let

$$C = A(x) \cdot B(x) \bmod P \tag{1}$$

$$= \sum_{k=0}^{m-1} A(x) \cdot b_k x^k \bmod P \tag{2}$$

$$= (\underbrace{(0 \cdot x + b_{m-1}A(x))}_{K^0(x)} x^{m-1} + \cdots + b_0 A(x)) \bmod P(x) \tag{3}$$

$$= (\underbrace{\left(K^0(x)x + b_{m-2}A(x)\right)}_{K^1(x)} x^{m-2} + \cdots + b_0 A(x)) \bmod P(x) \tag{4}$$

$$\vdots$$

$$= K^{m-1}(x) \bmod P(x) \tag{5}$$

Thus, the product $C(x)$ can be obtained by performing iteratively $m-1$ times:

$$K^i(x) = \left(K^{i-1}(x)x + b_{m-1-i}A(x)\right) \bmod P(x) \tag{6}$$

where $K^{-}1(x) = 0$.

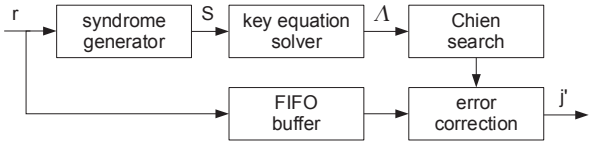


Fig. 1. Block Diagram of an BCH Decoder.

Equation (6) can be simplified by using the fact that $x^m \bmod P(x) = p_{m-1}x^{m-1} + \cdots + p_1 x + p_0$ and by writing $K^{i-1}(x)$ and $A(x)$ as $k_{m-1}^{i-1}x^{m-1} + \cdots + k_0^{i-1}$ and $a_{m-1}x^{m-1} + \cdots + a_0$ respectively and one obtains:

$$K^i(x) = \sum_{j=0}^{m-1} k_j^i x^j \tag{7}$$

with

$$k_j^i = k_{m-1}^{i-1} p_j + k_{j-1}^{i-1} + b_{m-1-i}a_j \quad \text{and} \quad k_{-1}^{i-1} = 0 \tag{8}$$

With Eqs. (6) and (8), the multiplication algorithm over $GF(2^m)$ is given, that is used for the architectures of the Scott- and Kitsos-multiplier.

2.2 BCH decoding

BCH codes have been invented in 1959 by Hocquenghem (Hocquenghem, 1959), and independently in 1960 by Bose and Ray-Chaudhuri (Bose and Ray-Chaudhuri, 1960). These codes provide high flexibility and predictable error correction ability, and hence are used in a wide range of technical applications. Details about their construction and decoding can be found in literature (Bossert, 1998; Friedrichs, 1995).

Two main categories of BCH codes exist: so-called *primitive* and *non-primitive* ones. Code words of primitive BCH codes are always of fixed length $n = 2^m - 1$, with n being the length of the code word and m the dimension of the Galois field 2^m. Non-primitive BCH codes provide flexibility in choosing the appropriate code word size. In contrast to code puncturing, where one or more positions in a code word are omitted, the minimum distance in a reduced code is not changed (Bossert, 1998). Non-primitive BCH codes are used in standards like DVB-T2, DVB-S2, DB-C2, DVB-H, ITU-T H.261 or ITU-T G.975.1, for example.

BCH codes are usually decoded by algebraic syndrome decoding. Figure 1 shows the generic structure of a syndrome based BCH decoder.

It is important to notice that the syndrome $S(x)$ only depends on a possible error vector e and not on the correct code word that has been sent. The decoder therefore has to find a possible error vector e with a minimum weight, that means an e with a minimum number of coefficients unequal to zero. In order to calculate e algebraically, the search for minimum

weight is transformed into a search for a polynomial with minimum degree (Bossert, 1998).

The syndrome generator unit (SGU) computes the syndrome $S(x)$ that is fed into a so-called *key equation solver* (KES). The KES computes the *error locator polynomial* $\Lambda(x)$ out of the syndrome. The Chien search unit (CSU) afterwards checks for all elements in the specified Galois field $GF(2^m)$ if $\Lambda(x) = 0$. In this case, the error vector e has a 1 at the corresponding position; the error correction then flips the bits at all recognized error positions in the received code word r.

3 Hardware implementation

The hardware implementation of the FFMs should be flexible and generic with respect to the primitive polynomial and field size. Furthermore, they should exploit the underlying structures of the Virtex-4 FPGA-platform well to yield small area usage while providing sufficient throughput.

For the target DVB-S2 standard, BCH codes based on the Galois fields 2^{14} and 2^{16} are used. As this research was performed in the context of a more complex communication system, the common clock frequency of the whole system was predefined to 200MHz.

The hardware implementations considered in our work are based on (Scott et al., 1986) and (Kitsos et al., 2003). Furthermore, we give insight into the architecture of the plain combinatorial FFM implementations used.

3.1 Scott-Multiplier

The architecture of the Scott-multiplier resembles the multiplication algorithm derived in Sect. 2.1. Equation (8) is mapped to a basic cell (see Fig. 2). It consists of two 2-input AND-gates, two 2-input XOR gates and a flip-flop. Additionally, our implementation contains flip-flops for storing the coefficients of the A-operand during a calculation and control logic for that. Still, such cells fit comfortably into a Virtex-4 FPGA slice that offers two 4-input LUTs, two multiplexers, two 1-bit registers and arithmetic logic.

For a multiplier over $GF(2^m)$, m basic cells are connected into an array that calculates the product $C = A \cdot B \bmod P$ in m clock cycles. This array can be viewed as a linear feedback shift register and is illustrated in Fig. 3. The primitive polynomial of the Galois field can arbitrarily be set at design time.

3.2 Kitsos-multiplier

This architecture extends the Scott-multiplier by a reconfigurable feedback path, gated clocking and is also based on Eq. (8). The reconfigurable feedback path allows for a runtime reconfiguration of the multiplier. This reconfigurability aims to support not only arbitrary primitive polynomials but also field sizes $1 \leq i \leq m$ for an array length m. The gated clocking leads to an increased power efficiency when the multiplier is reconfigured to lower dimension fields.

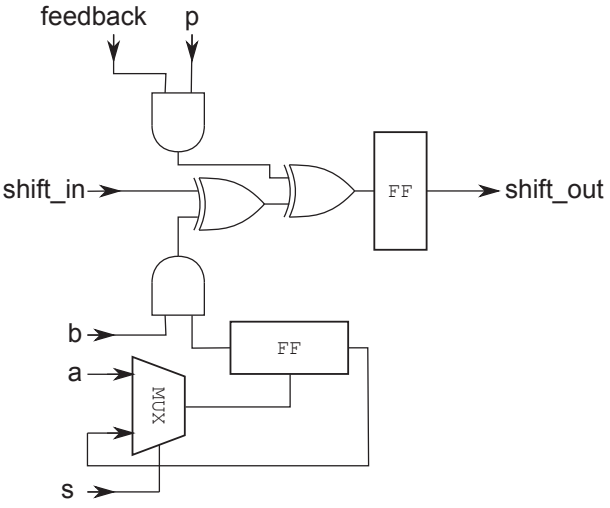


Fig. 2. Scott-Multiplier Basic Cell.

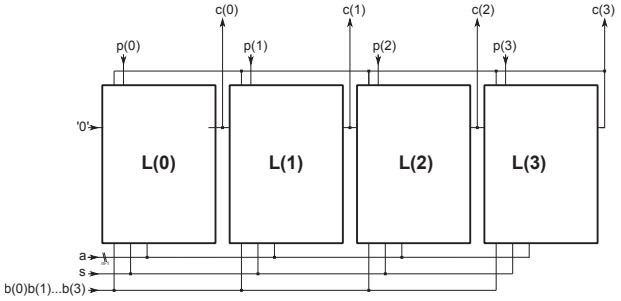


Fig. 3. Scott-Multiplier Array of Four Cells for $m = 4$.

In addition to the two 2-input AND-gates, two 2-input XOR gates and flip-flop required due to Eq. (8), the basic cells of the Kitsos-multiplier contain a demultiplexer and a two-input XOR and AND gate as shown in Fig. 4.

Similar to the Scott-multiplier, m basic cells form an array to allow for multiplication over $GF(2^m)$. The Kitsos-multiplier can be reconfigured at run time to support any field dimension j with $1 \leq j \leq m$ by selecting a shorter feedback path and disabling part of the flip-flops. Our implementation also features built-in default primitive polynomials and further control logic to be able to use an arbitrary primitive polynomial. An example for $m = 4$ is given in Fig. 5.

3.3 Plain multiplier architectures

In the original BCH decoder implementation, the multipliers used were based on a pure polynomial division algorithm. As a first step, plain unrolling of this polynomial division led to pure combinatorial FFMs. In the second step we have

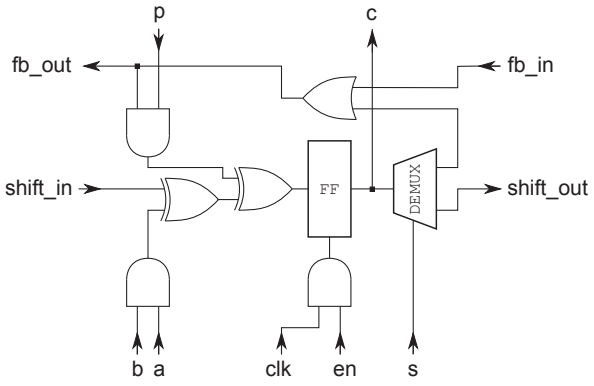


Fig. 4. Kitsos-Multiplier Basic Cell.

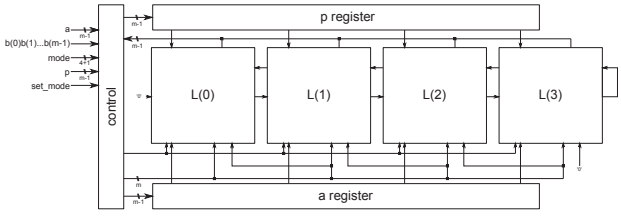


Fig. 5. Kitsos-Multiplier Array for Maximum $m = 4$.

introduced pipeline registers to shorten the critical path of the circuit.

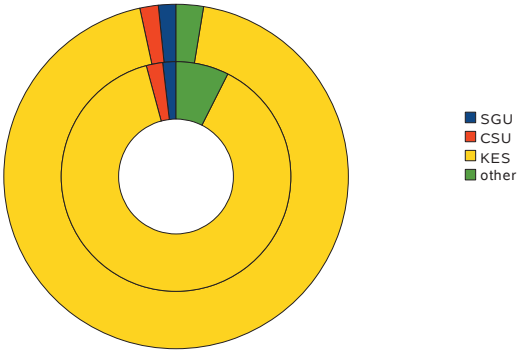
We use both plain FFM types as a comparison for our results. Detailed numbers for all multipliers are given in Fig. 2 and Table 2.

3.4 BCH decoder architecture

Our original decoder implementation is derived from the generic template shown in 1. The key equation solver is based on a fully parallel Euclidean implementation that uses non-pipelined combinatorial FFMs as described in Sect. 3.3. SGU and CSU use the same type of FFM. Since we intend to enhance our decoder to Reed-Solomon codes in the future, we have decided to go for a fully parallel Euclidean based key equation solver. The decoder can be configured at design time with the following parameters:

- n: length of the code word
- m: dimension of the extension field 2^m
- t: numbers of errors that can be corrected
- p: number of bits that are processed in parallel in the syndrome generator unit and the Chien search unit

In all three units in the upper part of Fig. 1, FFMs are the key components (Chen et al., 2009; Zhang et al., 2010).



inner: m=8, n=128, t=3, p=1 / outer: m=13, n=4096, t=7, p=1

Fig. 6. Slice Distribution of BCH Decoder Components.

However, the complexity of some FFMs can be reduced at design time, since they have one input fixed to a constant value out of the specific Galois field used. This is the case for all $2 \times t$ multipliers in the syndrome computation unit and for all $t \times p$ multipliers in the Chien search unit.

The key equation solver only contains full FFMs with non-constant inputs. To support both GF dimensions (2^{14} and 2^{16}) in the DVB-S2 BCH decoder, every FFM instance internally consists of two FFMs, one for each GF dimension. They share the same wires and are selected via multiplexers at the inputs and outputs.

Figure 6 clearly shows for two explicit configuration cases that these $8 \times t + 2$ FFMs in total consume the vast majority of the overall area required by the complete BCH decoder.

InTable 1 the latency of the single decoder stages is given for combinatorial and Scott FFMs based on the decoder parameters. Due to the high parallelism in the KES, the majority of decoding time is spend in SGU and CSU. Nevertheless, these units can easily be speeded up by increasing p, reducing the latency in these units by $1/p$. However, this will also increase the consumed area by a factor of nearly p.

4 Results

4.1 Validation

With increasing complexity in current hardware designs, permanent evaluation and verification is crucial throughout the design process. To speed up the otherwise very time consuming simulation process, the VHDL models of the FFMs have been exhaustively co-simulated with C++ reference code for GF-multiplication. This co-simulation has been carried out using Mentor Graphics ModelSim and a SystemC test bench as illustrated in Fig. 7. As only field sizes up to 2^{16} are of interest for our BCH decoder, all meaningful stimuli have been applied and the models have therefore been fully verified.

Table 1. Latency Distribution of BCH Decoder Components.

Component	Latency Combinatorial FFMs	Latency Scott FFMs
Key Equation Solver	$4 \times t + 2$	$(4 \times t + 2) \times m$
Syndrome Generator Unit	$n/p - 1$	$(n/p - 1) \times m$
Chien Search Unit	$n/p + 1$	$(n/p + 1) \times m$

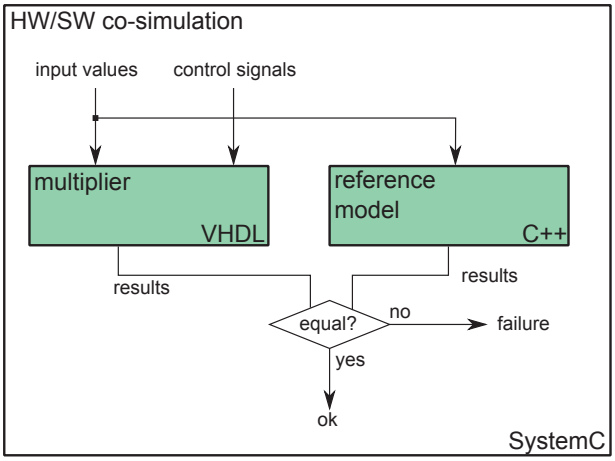


Fig. 7. HW/SW Co-Simulation Schematic.

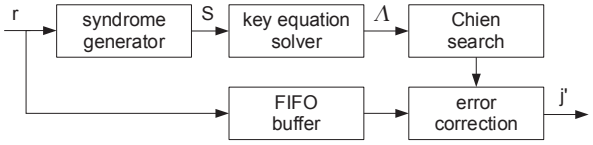


Fig. 8. Throughput per Slice @200 MHz.

4.2 Synthesis results

All multiplier architectures described in 3 have been implemented in VDHL. They have been synthesized for low area consumption on a Xilinx Virtex 4 FPGA (XC4VLX100, package FF1148). The applied tool chain was Xilinx ISE IDE Release 11.3 and the optimization effort for place and route was set to high. Figure 2 gives detailed synthesis results (post place and route).

Due to the given system requirements, the main focus lay on a resulting small footprint on the FPGA for given throughputs. Therefore the metric considered most important was throughput per slice at a fixed clock frequency. Figure 2 shows that the required target clock frequency of 200 MHz could be achieved by all implementations, except the Kitsos-multiplier for field dimensions $m \geq 15$.

For the Scott-multiplier, we can achieve possible clock frequencies of more than 1 GHz due to the short critical path (see Fig. 2). Furthermore, the Scott-multiplier outperforms the other multiplier architectures and even the plain combinatorial multiplier implementation with respect to the metric introduced above. This is shown in Fig. 2.

When comparing the Scott-multiplier and the plain combinatorial multiplier, the area saving outweighs the lesser throughput (factor $\frac{1}{m}$) by a factor from 14.75 (for $m = 12$) up to 18.1 (for $m = 16$).

The overhead of the Kitsos-multiplier implementation heavily impacts the area usage. While still less area is needed as for either one of the polynomial-division-based multipliers, the Kitsos-multiplier requires roughly ten times more area than the Scott-multiplier yet providing the same throughput. A BCH decoder for DVB-S2 requires FFMs for two different GF dimensions: 2^{14} and 2^{16}. As a result, two instances of the Scott-multiplier (one for each dimension) outperform the Kitsos-multiplier clearly.

4.3 Impact on BCH decoder

With respect to the consumed area,Table 2 shows that the the total number of slices occupied by the Scott-multiplier is 9 for GF 2^{14}. That are 5.4 % of the 168 slices required by the plain combinatorial FFM for this GF dimension. For GF 2^{16}, the Scott-FFM only consumes 5.5 % of area. For the FFM pairs in the DVB-S2 decoder as explained in Sect. 3.4, this results in a total area saving of around 94 % for the FFMs. With respect to Fig. 6, we thus expect a very high potential for area saving by employing Scott-FFMs in the key equation solver part. Nevertheless, the area occupied by the multiplexers and registers will not be reduced. Furthermore, the area reduction of FFMs with constant inputs as in the SGU and CSU will also be significantly lower.

By using pipelined multipliers, the latency in each decoder component is increased by a factor of m (see Table 1). For the configurations shown in Fig. 6, the latencies of SGU and CSU are already massively dominating over the KES latency. Using pipelined FFMs in SGU and CSU would there lead to a significant decrease of the overall decoder throughput, by only small area reduction in total.

However, Fig. 6 shows that the KES is the dominating part in the decoder with respect to area, but hardly contributes to the overall latency. This means that adding additional latency in the KES will only insignificantly increase the overall latency of the decoder. On the other hand, smaller FFMs in the KES will show a considerable impact on the overall

Table 2. Detailed Synthesis Result.

Type	m	slice-FFs	4-input LUTs	slices total	throughput $[\frac{results}{sec}]$@200MHz	throughput per slice $[\frac{results}{sec \cdot slice}]$@200MHz	max. freq. [MHz]
Scott	12	12	15	8	1.66×10^7	2.08×10^6	1034.1
	13	13	16	8	1.53×10^7	1.92×10^6	1177.9
	14	14	17	9	1.42×10^7	1.58×10^6	1189.1
	15	15	16	8	1.33×10^7	1.66×10^6	1116.1
	16	16	19	10	1.25×10^7	1.25×10^6	1083.4
Kitsos	12	59	136	83	1.66×10^7	0.2×10^6	226.50
	13	64	160	100	1.53×10^7	0.15×10^6	212.27
	14	69	186	110	1.42×10^7	0.12×10^6	209.03
	15	74	191	114	$*$	$*$	169.81
	16	79	169	104	$*$	$*$	198.77
Plain (pipelined)	12	392	177	200	20×10^7	10^6	833.33
	13	457	205	246	20×10^7	0.81×10^6	840.34
	14	527	236	270	20×10^7	0.74×10^6	889.68
	15	602	240	323	20×10^7	0.61×10^6	871.84
	16	682	302	348	20×10^7	0.57×10^6	701.26
Plain (combinatorial)	12	53	169	115	20×10^7	1.73×10^6	373.13
	13	61	182	130	20×10^7	1.53×10^6	394.01
	14	75	237	168	20×10^7	1.19×10^6	365.63
	15	66	220	148	20×10^7	1.35×10^6	401.61
	16	71	281	181	20×10^7	1.1×10^6	340.14

$*$: Maximum clock frequency was below 200 MHz.

consumed area, since those FFMs use up most of the hardware resources (see Sect. 3.4).

In order to achieve a well-balanced decoder design, we therefore suggest to employ two different FFM types in the decoder: combinatorial FFMs in the SGU and CSU, and small pipelined FFMs in the KES. This setup will exploit the benefits of both FFM types and reduce the overall area of the decoder, by only adding an insignificant amount of total latency.

5 Conclusions

Finite field multiplications are key in many technical applications, particularly in decoding BCH codes. In this work we have investigated available pipelined architectures with respect to scalability, throughput and area consumption on FPGAs. We have implemented and synthesized those designs for a common target device, a Xilinx Virtex-4 XC4VLX100 FPGA. Our implementations have been fully verified in a hardware-software co-simulation. We have evaluated and compared the selected architectures and show that for the choosen effiency metric the Scott-multiplier features in principle the best performance. For application of the considered multipliers in a DVB-S2 BCH decoder the Kitsos-multiplier is outperformed by the other multipliers, despite the inherent architectural flexibility of the Kitsos-multiplier. Furthermore, we conclude that the key equation solver part can profit most from pipelined finite field multipliers, and that simple combinatorial multipliers provide a higher benefit in the syndrome computation and the Chien search.

References

Ahlquist, G. C., Nelson, B. E., and Rice, M.: Optimal Finite Field Multipliers for FPGAs, in: FPL '99: Proceedings of the 9th International Workshop on Field-Programmable Logic and Applications, pp. 51–60, Springer-Verlag, London, UK, 1999.

Bose, R. C. and Ray-Chaudhuri, D. K.: On a class of error correcting binary group codes, Inform. Control, 3, 68–79, doi:10.1016/S0019-9958(60)90287-4, 1960.

Bossert, M.: Kanalcodierung, B. G. Teubner Stuttgart, 2nd edn., 1998.

Chen, Z., Zhang, Y., Ying, Y., Wu, C., and Zeng, X.: An Area-Efficient and Degree-Computationless BCH Decoder for DVB-S2, in: Proceedings of the IEEE 8th International Conference on ASIC, (ASICON) 2009. , 489–492, doi:10.1109/ASICON.2009.5351625, 2009.

Friedrichs, B.: Kanalcodierung, Springer Verlag Berlin Heidelberg, 1995, ISBN 3-540-59353-5.

Hocquenghem, A.: Codes correcteurs derreurs, Chiffres, 2, 147–56, 1959.

Kitsos, P., Theodoridis, G., and Koufopavlou, O.: An efficient reconfigurable multiplier architecture for Galois field GF (2m), Microelectr. J., 34, 975–980, 2003.

Liu, W., Rho, J., and Sung, W.: Low-Power High-Throughput BCH Error Correction VLSI Design for Multi-Level Cell NAND Flash Memories, in: Proceedings of the IEEE Workshop on Signal Processing Systems Design and Implementation, (SIPS) 2006. , 303–308, doi:10.1109/SIPS.2006.352599, 2006.

Scott, P., Tavares, S., and Peppard, L.: A Fast VLSI Multiplier for GF(2m), in: IEEE Journal on Selected Areas in Communications (J-SAC) 1986, 4, 62–66, 1986.

Zhang, B., Liu, D., Wang, S., Chen, X., and Liu, H.: Design and Implementation of Area-Efficient DVB-S2 BCH Decoder, in: Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, 3, V3–179–V3–184, doi:10.1109/ICCET.2010.5485823, 2010.