High-Throughput and Low-Power Architectures for Reed Solomon Decoder

Akash Kumar
Eindhoven University of Technology
5600MB Eindhoven, The Netherlands
Email: a.kumar@tue.nl

Sergei Sawitzki
Philips Research Laboratories
5656AA Eindhoven, The Netherlands
Email: sergei.sawitzki@philips.com

***Abstract*— This paper presents a uniform comparison between various algorithms and architectures used for Reed Solomon (RS) decoder. For each design option, a detailed hardware analysis is provided, in terms of gate count, latency and critical path delay. A new low-power syndrome computation is proposed in the paper. Dual-line architecture of modified Berlekamp Massey algorithm was chosen for Ultra Wide-band (UWB) as an application example. The results obtained are very encouraging both in terms of silicon area and power. A detailed analysis of results is presented and they are also compared with other published industrial and academic designs.**

I. Introduction

Reed Solomon (RS) codes have been widely used in a variety of communication systems. Continual demand for ever higher data rates and storage capacity makes it necessary to devise very high-speed implementations of RS decoders. A number of algorithms are available and this often makes it difficult to determine the best choice due to the number of variables and trade-offs available.

For IEEE 802.15-03 standard proposal (commonly known as UWB) in particular, very high data rates for transmission are needed. Since the standard is also meant for portable devices, power consumption is of prime concern. There is no clear algorithm or architecture that can meet the low-power and high-throughput requirements of UWB. In this paper, a uniform comparison of various designs and architecture is presented. Dual-line architecture of BerleKamp Massey algorithm was implemented, with a lot of other optimisations to the conventional design.

In the next section we present an introduction to RS codes and the decoder structure, followed by syndrome computation architecture. The design space is explored in the following section. We then present the results obtained for the architecture chosen for UWB followed by some optimisations to the design. The results are then compared with existing architectures in the section on benchmarking followed by conclusions.

II. Reed Solomon Codes

RS codes are one of the most commonly used in all forms of transmission and data storage for forward error correction (*FEC*). They are a subset of Bose-Chaudhuri-Hocquenghem (*BCH*) codes and are linear block codes. [1] is one of the best references for RS Codes.

An $RS(n,k)$ code implies that the encoder takes in k symbols and adds $n-k$ parity symbols to make it an $n-$ symbol code word. Each symbol is at least of m bits, where $2^m > n$. Conversely, the longest length of code word for a given bit-size m, is $2^m - 1$. For example, $RS(255, 239)$ code takes in 239 symbols and adds 16 parity symbols to make 255 symbols overall of 8 bits each. Figure 1 shows an example of a systematic RS code word. It is called systematic code word as the input symbols are left unchanged and the parity symbols are appended to it.

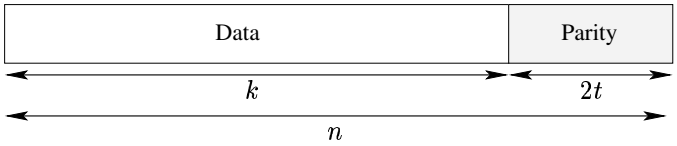

Fig. 1. A typical RS code word

Reed Solomon codes are best for burst errors. If the code is not meant for erasures, the code can correct errors in up to t symbols where $2t = n - k$. A symbol has an error if at least one bit is wrong. Thus, $RS(255, 239)$ can correct errors in up to 8 symbols or atleast upto 50 successive bit errors. It is also interesting to see, that the hardware required is proportional to the error correction capability of the system and not the actual code word length as such.

When a code word is received at the receiver, it is often not the same as the one transmitted, since noise in the channel introduces errors in the system. Let us say if $r(x)$ is the received code word, we have

$$r(x) = c(x) + e(x) \tag{1}$$

where $c(x)$ is the original codeword and $e(x)$ is the error introduced in the system. The aim of the decoder is to find $e(x)$ and then subtract it from $r(x)$ to recover original code word transmitted. It should be added that there are two aspects of decoding - error detection and error correction. As mentioned before, the error can only be corrected if there are fewer than or equal to t errors. However, the Reed Solomon algorithm still allows one to detect if there are more than t errors. In such cases, the code word is declared as *uncorrectable.*

A. Decoder Structure

A detailed explanation on Reed Solomon decoders can be found in [1] and [2]. The decoder essentially consists of four

modules. The first module computes the syndrome polynomial from the received sequence. Typically $2t$ syndromes need to be computed for a code with error correction capability of t. This is used to solve a key equation in the second block, which generates two polynomials for determining the location and value of these errors in the received code word. The next block of Chien search computes the error location, while the fourth block employs Forney algorithm to determine the value of error occurred.

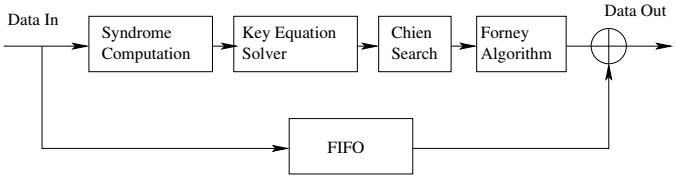

Fig. 2. *Decoder flow*

III. Low Power Syndrome Computation

If there is no error in the code word, all $2t$ syndromes are equal to zero. Interestingly, not all $2t$ syndrome coefficients are actually needed to check if indeed an error has occurred. If any t continuous coefficients are zero, this indicates that there is no error or that the code word is uncorrectable. We can therefore save power that is required in the computation of other t syndromes. Mathematically, for a correctable code word, if

$$S_j, S_{j+1}, \ldots S_{j+t-1} = 0 \ \forall \ 1 \leq j \leq t, \tag{2}$$

or

$$S_j, S_{j+1}, \ldots S_{2t}, S_1, \ldots S_{j-t-1} = 0 \ \forall \ t+1 \leq j \leq 2t, \tag{3}$$

then $S_j = 0 \ \forall \ 1 \leq j \leq 2t$. A similar idea has been proposed in [3]. However, that is only based on the inference that first t syndromes being zero imply all syndromes are zero, which is only a subset of the theory presented here.

The modified decoder flow has been presented in Figure 3. As can be seen only half the syndromes are computed in the normal flow. The other half are only computed if any of them is found to be non-zero.

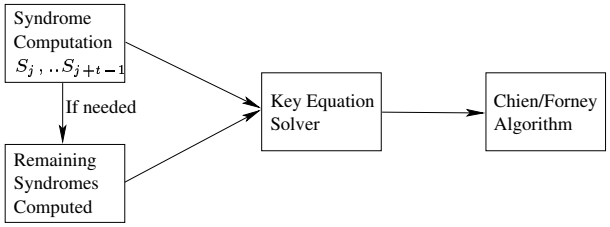

Fig. 3. *New decoder flow when only half the syndromes are computed.*

IV. Design Space

An extensive survey of design options was carried out and the architectures available are shown in Figure 4. The hardware utilized for each of the options was studied comprehensively and summarized as well.

In order to choose a good architecture for the application, various things have to be taken into account.

- Gate count: Determines the silicon area to be used for development. A one time production cost but can be critical if it is too high. Power is also determined by it to some extent.
- Latency: Latency is defined as the delay between the received code word and the corresponding decoded code word. The lower the latency, the smaller is the FIFO buffer size required.
- Critical path delay: It determines the minimum clock period, i.e. maximum frequency that the system can be operated at.

The above parameters were carefully studied and documented for each of the design options presented in Figure 4 in [6]. Table I shows a summary of all the above mentioned parameters. For our intended UWB application, speed is of prime concern as it has to be able to support data rates as high as 1.0 Gbps. At the same time, power has to be kept low, as it is to be used in portable devices as well. This implies that the active hardware at any time should be kept low. Also, the overall latency and gate count of computational elements should be low since that would determine the total silicon area of the design.

Reformulated inversion-less and dual line implementation of the modified Berlekamp Massey have the smallest critical path delay among all the alternatives of the Key Equation Solver. Inversion-less and dual-line architectures are explained in [10] and [4] respectively. When comparing inversion-less and dual-line implementations, dual line is a good compromise in latency and computational elements needed. The latency is one of the lowest and it has the least critical path delay of all the architectures summarized. Thus, dual-line implementation of the BM algorithm is chosen for the key-equation solver. Another benefit of this architecture is that the design is very regular and hence easy to implement. As for the other modules of decoder, the basic design itself was chosen.

Table II shows the various parameters for choosing this architecture with $n = 255, k = 239$ and $t = 8$. The overall critical path delay is Mul + Add. The chosen option is indicated in Figure 4 in bold boxes.

TABLE II
Summary of hardware utilization for Dual-line architecture

Architecture	Add.	Mult.	Muxes	Latches	Latency
Syndrome	2t	2t	2t	4t	n
Key Equation	2t	4t + 1	2t	4t + 1	3t + 1
Chien/Forney	2t	2t + 2	2t + 2	2t + 10	4
Total	6t	8t + 3	6t + 2	10t + 11	3t + n + 5
For RS(255, 239)	48	67	50	91	284

V. Results

This section covers the results of various synthesis experiments conducted. Resource utilization, timing analysis and the power consumption were used as benchmarking parameters.

A. Area Analysis

Ambit - an ASIC synthesis tool, was run for $0.12\mu m$ and $0.18\mu m$ CMOS technology. The silicon area required was

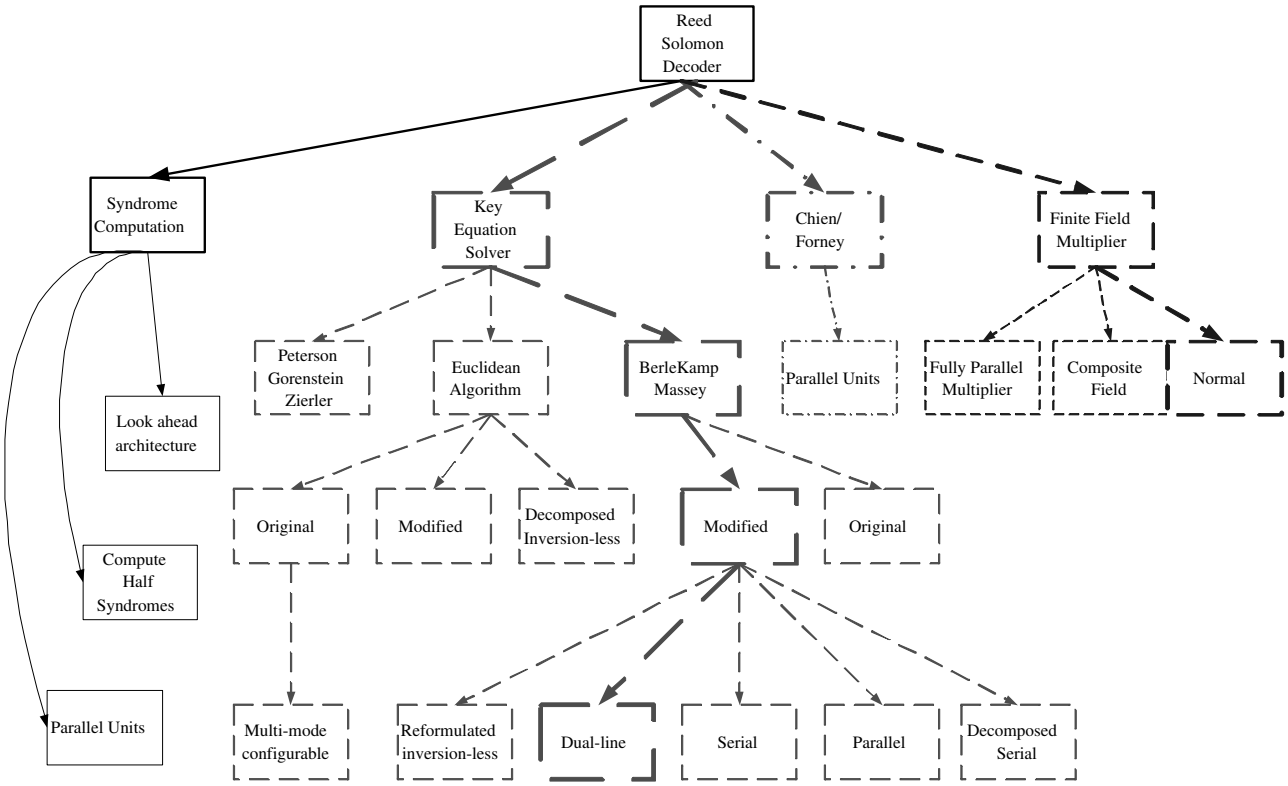

Fig. 4. *Design Space Exploration*

TABLE I

SUMMARY OF HARDWARE UTILIZATION OF VARIOUS ARCHITECTURES

Architecture	Blocks	Adders	Multipliers	Muxes	Latches	Latency	Critical Path Delay
Syndrome Computation [5]	2t	1	1	1	2		
Total		2t	2t	2t	4t	n	Mul + Add
Look ahead architecture (x units)	2t	x	x	1	2		
Total		2xt	2xt	2t	4t	n/x	Mul + Add
Original Euclidean [7]							
Divider Block	2t	1	1	3	2		
Multiply Block	t	2	1	3	3		
Total (Estimates)		4t	3t	9t	7t		
Actual [8]		4t + 1	3t + 1	11t + 4	14t + 6	4t - 3	ROM+2×Mul+Add+2×Mux
Modified Euclidean [7]							
Degree Computation Block	2t	2	0	7	7		
Polynomial Arithmetic Block	2t	2	4	8	19		
Total (Estimates)		8t	8t	30t	52t		
Actual [8]		8t	8t	40t + 2	78t + 4	10t + 8	Mul + Add + Mux
Decomposed inversion-less [9]		1	3	1	3t + 1	2t×(t+1)	Mul + Add + Mux
Modified BerleKamp Massey							
Serial		1	3	4	3t + 2	2t×(2t+2)	Mul + Add + Mux
Decomposed inversion-less [10]		2	3	2	5	2t×(t+1)	Mul + Add + Mux
Parallel		t	3t + 2	t	3t + 1	2t	2×Mul + 2×Add + Mux
Dual-line [4]		2t	4t + 1	2t	4t + 1	3t + 1	Mul + Add
Reformulated inversion-less [11]		3t + 1	6t + 2	3t + 1	6t + 2	2t	Mul + Add
Chien/Forney		2t	2t + 2	2t + 2	2t + 10	4	max(Mul + Add, ROM)

analysed for various timing constraints. A comparison for area of the decoder is shown in Table III. This table shows the area requirement when the constraint was set to 5 ns, which can support 200 MHz frequency, i.e. 1.6 Gbps. The total number of design cells used, including the memory, were 12,768 and 12,613 for $0.12\mu m$ and $0.18\mu m$ respectively.

TABLE III
RESOURCE UTILIZATION FOR THE DECODER

Module	Module Area(μm^2) CMOS18	CMOS12	CMOS12 Optimised
Syndromes	34,754	17,828	17,719
Key Equation	186,404	89,602	89,587
Chien	15,675	7,663	7,655
Forney	52,936	21,608	13,408
FIFO	148,684	83,183	108,906
Top View	438, 472	219, 913	237, 414

B. Power Analysis

Diesel - an internal tool developed within Philips which estimates the power for the simulated design, was used for power analysis. The power estimates provided in this section are for design operation at 125 MHz, which translates to data rate of 1Gbps.

1) Variation With Number of Errors: Figure 5 shows the variation of power with the number of errors found in the codeword for $0.12\mu m$ technology. As can be seen from the graph obtained, the power dissipated for the FIFO and syndrome computation block is independent of the number of errors as expected. For the key-equation solver, it is clearly seen that the power dissipated increases linearly with the number of errors. The Chien search block also shows a linear increase in the power dissipated.

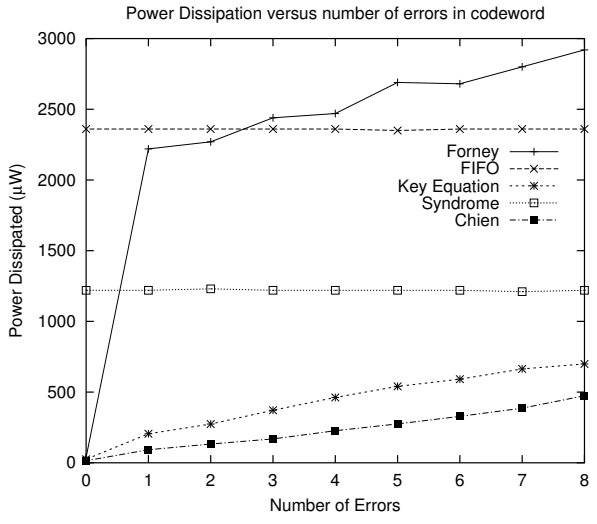

Fig. 5. Graph showing variation of power dissipated with number of errors for different modules.

The behaviour of Forney evaluator is a bit different from the other modules. We see that the power dissipated for the codeword with an even number of errors is not significantly larger than the one with the previous number of errors. The reason lies in the fact that the degree of Error Evaluator Polynomial for codeword with one error is often the same as the one with two errors, and so on and so forth. However, as a general rule, there is still an increase in the power dissipation, because of some computation that is done for each error found.

2) Distribution of Power in Different Modules: Figure 6 shows a distribution of power when there are maximum number of errors correctable in the received code word, while Figure 7 shows the distribution when the code word is received intact. As can be seen, in the case of no errors, bulk of the power is consumed in computing syndromes, apart from the memory. In the event of maximum errors detected, the Forney block consumes the maximum power. The total power consumption varies from 14mW to 17mW with the former corresponding to no-error case, and the latter to maximum errors.

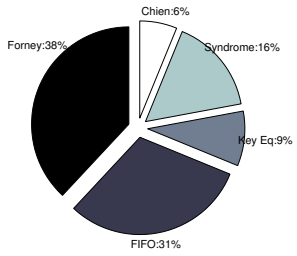

Fig. 6. Power consumed by various blocks when 8 errors are found

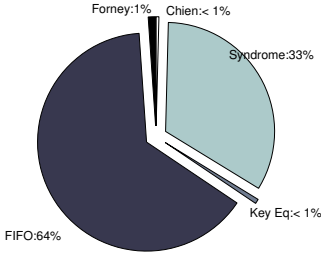

Fig. 7. Power consumed by various blocks when no errors are found

VI. OPTIMIZATIONS TO DESIGN

From the results, it was observed that the FIFO and the Forney block consumed most of the power. These blocks were investigated further and redesigned to improve the performance. The original FIFO design involved a serial arrangement of shift-registers. This design was the most compact in terms of area but consumed more power since at every cycle all the elements were shifted by one. The design was hence, modified to have only one read and write every clock cycle - like a RAM with pointers. This increased the design area, but significantly reduced the power. Area of the new design of FIFO is now 109,000 μm^2 (with $0.12\mu m$ technology), while the power consumed is only 970 μW, 60% lower than the earlier design. This results in power savings even in the case when no errors are found.

For the Forney block, design was optimised by combining two table lookups into one for computing the inverse of elements. This led to a better circuit in terms of area and also decreased the power significantly. The optimised design for Forney now occupies an area of 13,400 μm^2, about 38% lower than original design. The power consumption is lower by at least 1.5 mW for all cases. Table III shows the area distribution of the decoder after optimizing the design.

Figure 8 shows the power distribution in various modules when there are 8 errors in the received codeword, while Figure 9 shows the distribution when the codeword is received intact. As we can see, the FIFO now takes less than half the power in no-error case, as compared to two-thirds in the original design. In the case of 8-errors, the power consumption of Forney has

now reduced to about a quarter as compared to one-third in the original design.

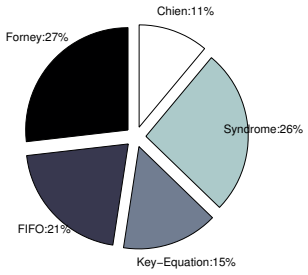

Fig. 8. Power consumed when 8 errors are found in optimised design

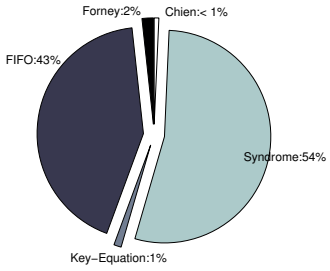

Fig. 9. Power consumed when no errors are found in optimised design

Figure 10 shows the variation of power with the number of errors. The total power consumption of the design now lies between 12mW to 14mW depending upon the no-error case to when maximum errors are found. It should be noted that 9.5mW of power is consumed in driving the input. Thus, only about 2.5mW to 4.5mW is actually consumed in the transitions in the design. The embedded memory AMDC C12ESRAM - developed internally in Philips, only consumes 900μW of power overall and has an area of $0.02mm^2$. As can be noticed, FIFOs take about 45% of the overall area in the design. Thus, the input power dissipation will be lower by 45% when using embedded memory - savings of $0.45 \times 9.5 = 4.275$mW. The input power dissipation, therefore, becomes only about 5.3mW. Thus, the total power consumed is 8 for the best case and 10mW for the worst. The overall design area is $0.16mm^2$. Voltage scaling measures can also be applied on the design to further lower the power consumption.

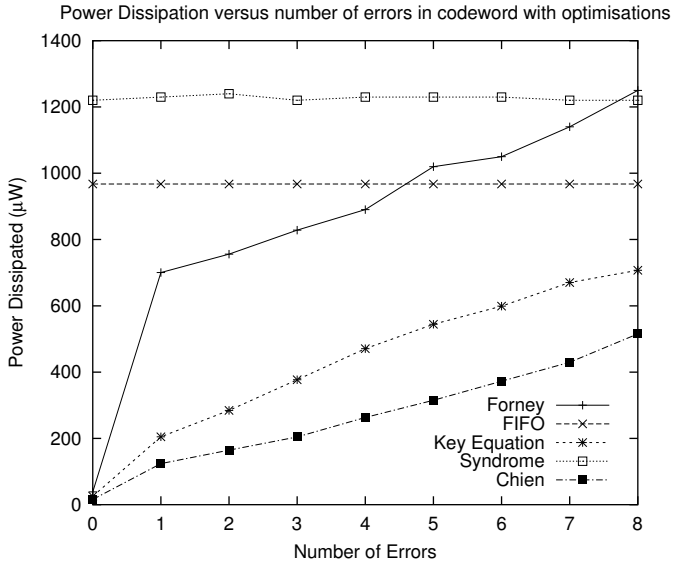

Fig. 10. Variation of power dissipated with number of errors for different modules with modifications in the design.

VII. Benchmarking

Please note that for all the designs $RS(255, 239)$ code has been used for benchmarking. The design proposed in [5] uses roughly 115K gates for $0.12\mu m$ CMOS technology operating at 6.0 Gbps excluding memory. The proposed design only uses 12K cells including memory in both $0.12\mu m$ and $0.18\mu m$ technology. The results are better even when they are normalised for throughput and technology. The latency of the design is only 284 cycles when compared to 355 cycles in [5].

A design was proposed in [3] for low power. In that design, 62mW of power is used in the best case, including memory, using $0.25\mu m$ CMOS technology, and 100mW are consumed in the worst case. In our design, only 10mW of power is used in the worst case using $0.12\mu m$ technology. The area of the chip proposed in [3] using $0.25\mu m$ CMOS technology is $5mm^2$, while the area of the proposed design is $0.16mm^2$ with $0.12\mu m$ technology. Even after scaling our design to $0.25\mu m$ CMOS technology, our results are better.

VIII. Conclusions

A uniform comparison was drawn for various algorithms that have been proposed in literature. This helped in selecting the appropriate architecture for the intended application. Modified Berlekamp Massey algorithm is chosen for the VHDL implementation. A dual-line architecture is used, which is as fast as serial and has low latency as that of a parallel approach.

The decoder implemented is capable of running at 200 MHz in ASIC implementation, which translates to 1.6 Gbps and requires only about 12K design cells and an area of $0.16mm^2$ with CMOS12 technology. The system has a latency of only 284 cycles for RS(255, 239) code. The power dissipated in the worst case is 10mW including the memory block when operating at 1.0 Gbps data rate. The results are better than other proposed designs and are suitable for use in UWB.

References

[1] S. B. Wicker and V. K. Bhargava; *Reed Solomon Codes and Their Applications*, Piscataway, NJ: IEEE Press, 1994.

[2] R. E. Blahut; *Theory and Practice of Error Control Codes*, Addison-Wesley, 1983.

[3] H. C. Chang, C. C. Lin and C. Y. Lee; *A low-power Reed-Solomon decoder for STM-16 optical communications* IEEE Asia-Pacific Conference on ASIC 2002

[4] H. J. Kang and I. C. Park; *A high-speed and low-latency Reed-Solomon decoder based on a dual-line structure* IEEE Intl. Conference on Acoustics, Speech, and Signal Processing, 2002

[5] H. Lee; *High-speed VLSI architecture for parallel Reed-Solomon decoder* IEEE transactions on VLSI Systems 2003.

[6] A. Kumar; High-throughput Reed-Solomon decoder for Ultra Wide band. Masters Thesis 2004, Technical University of Eindhoven and National University of Singapore, http://www.ics.ele.tue.nl/∼akash.

[7] H. Lee, M. L. Yu, and L. Song; *VLSI design of Reed-Solomon decoder architectures* IEEE Intl. Symposium on Circuits and Systems 2000.

[8] H. Lee; *An area-efficient Euclidean algorithm block for Reed-Solomon decoder* IEEE Computer Society Annual Symposium on VLSI, 2003

[9] H. C. Chang and C. Y. Lee; *An area-efficient architecture for Reed-Solomon decoder using the inversion less decomposed Euclidean algorithm* IEEE Intl. Symposium on Circuits and Systems 2001.

[10] H. C. Chang and C. B. Shung; *A (208,192;8) Reed-Solomon decoder for DVD application* IEEE Intl. Conference on Communications, 1998.

[11] D. V. Sarwate and N. R. Shanbhag; *High-speed architectures for Reed-Solomon decoders*, IEEE transactions on VLSI Systems 2001.