PAPER

An Analysis of Time Domain Reed Solomon Decoder with FPGA Implementation

Kentaro KATO[†a)], ***Member and*** **Somsak CHOOMCHUAY**[††b)], ***Nonmember***

SUMMARY This paper analyzes the time domain Reed Solomon Decoder with FPGA implementation. Data throughput and area is carefully evaluated compared with typical frequency domain Reed Solomon Decoder. In this analysis, three hardware architecture to enhance the data throughput, namely, the pipelined architecture, the parallel architecture, and the truncated arrays, is evaluated, too. The evaluation reveals that the number of the consumed resources of $RS(255, 239)$ is about 20% smaller than those of the frequency domain decoder although data throughput is less than 10% of the frequency domain decoder. The number of the consumed resources of the pipelined architecture is 28% smaller than that of the parallel architecture when data throughput is same. It is because the pipeline architecture requires less extra logics than the parallel architecture. To get higher data throughput, the pipelined architecture is better than the parallel architecture from the viewpoint of consumed resources.

key words: *FPGA, reed solomon decoder, error correcting code, time domain*

1. Introduction

Error control coding has become an essential means of ensuring data integrity in a variety of applications including satellite and mobile communications and the storage of data in magnetic and optical media. Among the various codes available for correcting multiple errors, the Reed-Solomon code is one of the most important codes. The notation $RS(N, k)$ is commonly used to donate a Reed-Solomon code of block size N symbols with each block containing k information symbols. Each symbol is an n bit word, which is usually interpreted as an element in $GF[2^n]$. This can correct for t errors, where $t = (N - k)/2$. The time domain algorithm has been developed by Blahut [1] by taking the inverse DFT of all the sequences and operators of the Berlekamp-Massey (BM) algorithm. This operates directly on the received data and generates the error sequence in the domain in which the data is received. Hence it eliminates the need for transform and inverse transformation operators such as syndrome computation and Chien search. As a result, the complete algorithm can be implemented by the repeated application of a single operation, which is of importance for hardware implementation.

However, the main drawback of the time domain algorithm is its high computation count. This is brought about by the fact that time domain key equation solving algorithm has to operate on the complete data sequence of length N, while the frequency domain algorithm needs to work only the syndrome sequence of length $N - k$. The computational count can be reduced by modifying the algorithm or considering hardware architecture [2].

It is known that time domain Reed Solomon decoder has lower area and longer computational time in general. But it is not well analyzed quantitatively in circuit level from practical point of view.

This paper analyzes the time domain Reed Solomon Decoder with FPGA implementation. Data throughput and area is carefully evaluated compared with typical frequency domain Reed Solomon Decoder. In this analysis, three hardware architecture to enhance the data throughput, namely, the pipelined architecture, parallel architecture, and the truncated array, is carefully evaluated in addition to the analysis of the normal time domain Reed Solomon decoder.

The contribution of this paper is as follows.

- We first analyzed the performance of three different styles of the architecture of time domain Reed Solomon Decoder shown in [2] in the circuit level.
- We first revealed quantitatively that the pipelined architecture of the time domain Reed Solomon Decoder is better than the parallel architecture from the viewpoint of area cost.
- We provided the data for design of practical time domain Reed Solomon Decoder.

The rest of the paper is organized as follows. Section 2 shows the related works. Section 3 explains the detail of the partially parallel time-domain Reed Solomon decoder. Section 4 gives the evaluation results. Finally Sect. 5 concludes the paper.

2. Related Works

Among codes employed to ensure data integrity, RS code is preferably desired for burst error correction whilst convolutional code is good for random error correction. The emerging of convolution code and LDPC code has led to the intensive interest in iterative decoding. The Koetter Vardy (KV),

Manuscript received January 24, 2017.
Manuscript revised June 25, 2017.
Manuscript publicized August 23, 2017.

[†]The author is with the department of Creative Engineering, National Institute of Technology, Tsuruoka College, Tsuruoka-shi, 997–8511, Japan.

[††]The author is with Electronics Engineering Department, King Monkut's Institute of Technology, Ladkrabang, Chalongkrung Rd. Ladkrabang Bangkok 10520, Thailand.

a) E-mail: k-katoh@tsuruoka-nct.ac.jp (Corresponding author)

b) E-mail: kchsomsa@kmitl.ac.th

DOI: 10.1587/transinf.2017EDP7039

a soft-input algebraic decoding [3] is quite well known. Inspired by the research published by Yedidia et al. [4], Jiang et al. [5] has proposed a stochastic shifting based iterative decoding (SSID) scheme in decoding RS codes. They showed that Belief Propagation algorithm (BPA) can be used to decoded RS code. It should be noted that BPA and SPA [6], [7] are commonly not practical for high density parity check code such as RS code in [5]. Regardless, the extensive computation caused by Gaussian elimination (GE), Jiang's work can outperform about 2 dB (FER, at 200 outer rounds and 50 SPA) compared to HDD (Hard decision decoding). However, their method diminishes as the codeword length becomes long, i.e. With such a regard, Bellorado et al. [8] has proposed an iterative RS decoder based on reduced-density, binary, parity check matrix. The computation complexity is reduced notably as the error rate is better. They said algorithm seems to be implementable in hardware. However, for the application which FER is not as that low and energy saving is more concerned, the tradition HDD is still reveal.

Architecture of higher performance Reed Solomon decoder has been researched for a long time, which is still one of the hot research topics. A syndrome-based RS decoder generally consists of three main blocks: a syndrome calculation (SC) block, a key equation solver (KES) block, and a Chien search and error evaluation (CSEE) block [9]. Dayal et al. [10] proposed the $RS(255, 239)$ decoder for wireless network 802.16 introduced pipelining in Chien-Search component of decoder to improve the maximum frequency of Reed-Solomon codes. Jiang et al. [11] proposed a multi-gigabit Reed-Solomon (RS) convolutional codes (CC) decoder architecture for 60 GHz systems. They introduced reformulated inversionless Berlekamp-Massey (RIBM) algorithm via double-clock methods to improve the decoding speed in the RS decoder part. Lee et al. proposed a low-complexity, high-speed $RS(255, 239)$ decoder architecture using Modified Euclidean (ME) algorithm for the high-speed optic communication systems [12]. These days, communication system is often implemented on FPGA for lower development costs and its re-configurability [13]. RS decoders for several communication systems have been implemented on FPGA [14], [15]. However because FPGA is vulnerable to soft error, soft error tolerant system design is important for communication systems on FPGA [16].

3. Time Domain Reed Solomon Decoder

This section explains the target time domain Reed Solomon decoder [2]. Either the Modified Euclidean (ME) algorithm or BM algorithm can be used to solve a key equation for an error locator polynomial and an error evaluator polynomial [9]. The target decoder uses BM algorithm. Section 3.1 explains the decoding algorithm briefly. Section 3.2 describes hardware architecture.

3.1 Decoding Algorithm

The target algorithm is Algorithm D proposed in [2]. The algorithm consists of 3 phases. In the first phase, the 9 parameters for this algorithm are initialized. After that, in the 2nd phase, the process to update the parameters following the Eqs. (1a) and (1b) is repeated $2t$ times. This algorithm does not require transform and inverse transformation operators such as syndrome computation and Chien search. As a result, the complete algorithm can be implemented by the repeated application of a single operation, which is of importance for hardware implementation. Finally, in the 3rd phase, error magnitude in each location i is calculated by the Eq. (1c).

Algorithm D:

Initialize : $\lambda_i^0 = \gamma_i^0 = \omega_i^0 = 1;$
$\zeta_i^0 = \xi_i^0 = \mu_i^0 = 0;$
$L_0 = 0;$
$K_0 = u_0 = 1;$

for $r = 1, 2, \ldots 2t$
begin
$\delta_r = 1$ if both $\Delta \neq 0$ and $2L_{r-1} \ll r - 1$; otherwise $\delta_r = 0$. If $\delta_r = 0$, $u_r = u_{r-1} + 1$;otherwise $u_r = 1$.
for $i = 0, 1, \ldots, N - 1$
begin

$$\Delta_r = \sum_0^{N-1} v_i \lambda_i^{r-1} \alpha^{i(r-1)} \tag{1.a}$$
$$L_r = \delta_r(r - L_{r-1}) + (1 - \delta_r)L_{r-1}$$
$$K_r = \delta_r \Delta_r^{-1} + (1 - \delta_r)K_{r-1}$$
$$\begin{bmatrix} \lambda_i^r \\ \gamma_i^r \\ \zeta_i^r \\ \xi_i^r \end{bmatrix} =$$
$$\begin{bmatrix} 1 & -\Delta_r K_{r-1}\alpha^{-iu_r} & 0 & 0 \\ \lambda_r & (1-\delta_r) & 0 & 0 \\ 0 & -\Delta_r K_{r-1} u_r \alpha^{-i(u_r+1-t_0)} & 1 & -\Delta_r K_{r-1}\alpha^{-iu_r} \\ 0 & 0 & \delta_r & (1-\delta_r) \end{bmatrix}$$
$$\times \begin{bmatrix} \lambda_i^{r-1} \\ \gamma_i^{r-1} \\ \zeta_i^{r-1} \\ \xi_i^{r-1} \end{bmatrix} \tag{1.b}$$
$$\begin{bmatrix} \omega_i^r \\ \mu_i^r \end{bmatrix} = \begin{bmatrix} 1 & -\Delta_r K_{r-1}\alpha^{-iu_r} \\ \lambda_r & (1-\delta_r) \end{bmatrix} \begin{bmatrix} \omega_i^{r-1} \\ \mu_i^{r-1} \end{bmatrix}$$
end
end
$$e_i = \frac{\omega_i^{2t}}{\zeta_i^{2t}} \text{ where } \lambda_i = 0 \tag{1.c}$$

3.2 Hardware Architecture

Here the hardware architecture of the time domain Reed Solomon decoder to reduce the computational time is explained. The processing module computes the arithmetic calculation formulated (1a), (1b), and (1c). The data are decoded by $2t$ times calculation with the processing modules. First the processing module, which is the basic unit for the

time domain decoder, is described in Sect. 3.2.1. Choomchuay proposed three hardware architectures to make the computational time practical, namely pipelined architecture, parallel architecture, and truncated arrays [2]. Sections 3.2.2, 3.2.3, and 3.2.4 explain the architecture, respectively.

3.2.1 Processing Module

Figure 1 depicts the block diagram of the processing module. The processing module calculates Δ_r and updates the 6 parameters λ_i, γ_i, ζ_i, ξ_i, ω_i, μ_i $(0 \le i \le N-1)$. The left side is the input-side. The right side is the output-side. The module consists of Δ_r computation unit for calculating Δ_r and the units for updating the 6 parameters, matrix coeff generation, iteration decision, and vector modification unit. N bit shift register D(N) is connected to v_i and each input line of the 6 parameters. The processing module calculates N elements of each parameter sequentially with shift operation of D(N). In addition to the N clock cycles, 1 clock cycle for initialization and 1 clock cycle for updating Δ_r are needed. Therefore, $N+2$ clock cycles are required for the calculation. The waveforms of the logic simulation of a processing module for $RS(15, 9)$ is shown in Fig. 3.

Figure 2 shows the sequential decoding architecture using single Processing Module. The time domain architecture decodes the encoded codeword by repeating computation of (1.a) and (1.b) $2t$ times. The precomputed Δ_r and the 8 parameters are applied to the Processing Module and the Processing Module generates the updated parameters. The updated parameters are looped back to the inputs of the

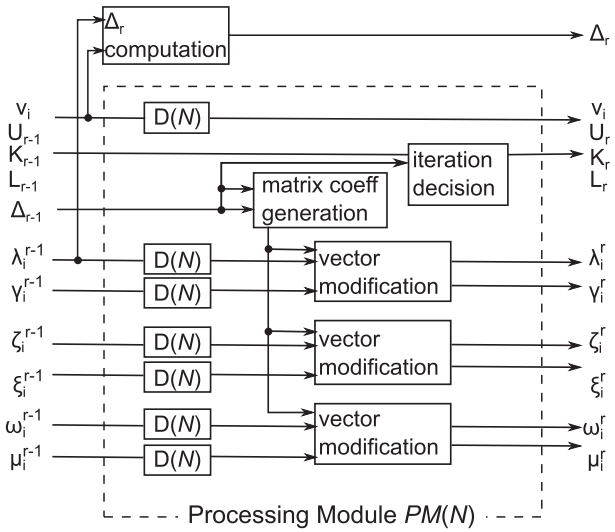

Fig. 1 Processing module

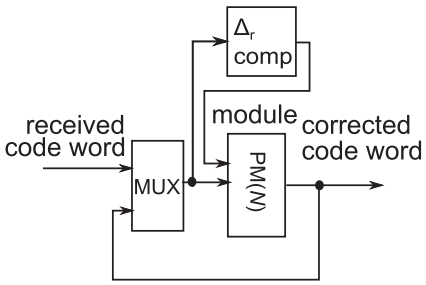

Fig. 2 Sequential architecture.

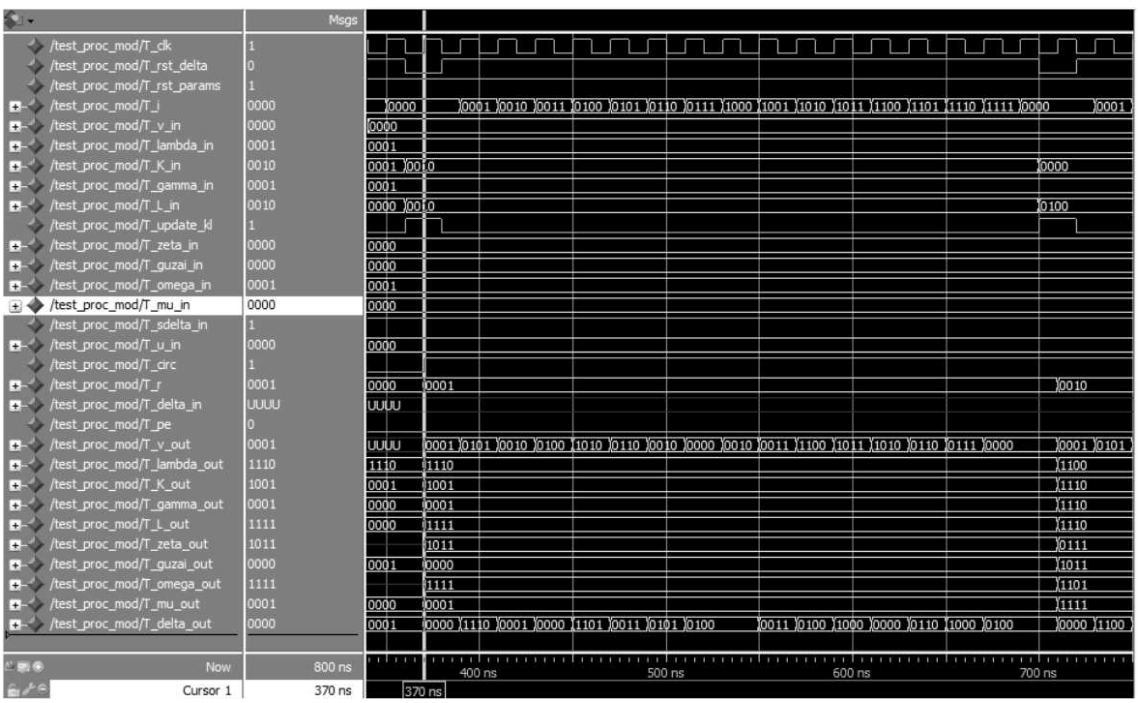

Fig. 3 Waveform of logic simulation of $RS(15, 9)$.

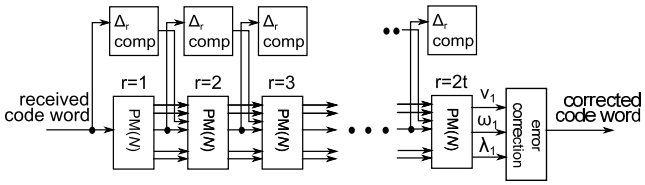

Fig. 4 Pipelined architecture.

Processing Module via MUX. This process is repeated $2t$ times. The value of $\Delta_{(r+1)}$ is calculated simultanously. This calculation requires N clock cycles. Here, let PM(N) be the Processing Module including shift registers whose length is common N.

In this case, data throughput TH (bit/s) of $RS(N,k)$ is expressed by the following formula.

$$TH = \frac{fnk}{2t(N+2)}, \tag{2}$$

where f is the cock frequency of the decoder, n is the bit width of a code word, k is the number of information symbols, and N is the block size.

3.2.2 Pipelined Architecture

As shown in Fig. 4, fully pipelined architecture consists of $2t$ processing modules connected serially each other. Each processing module carries out each iteration from 1 to $2t$ of the algorithm. The Δ_r comp unit of each stage calculates the value of Δ_{r+1} used in the next stage. In this case, data throughput TH (bit/s) of $RS(N,k)$ increases $2t$ times compared with the sequential architecture. Accordingly, the data throughput is expressed by the following formula.

$$TH = \frac{fnk}{(N+2)}. \tag{3}$$

This architecture requires $2t$ processing modules. Therefore, the required hardware resources increase linearly as the iteration time increases.

3.2.3 Parallel Architecture

Figure 5 shows the p-parallel processing architecture. The decoding architecture is based on the sequential decoding architecture shown in Fig. 2. However in the parallel architecture, the single Processing Module is replaced by the p parallel Processing Modules. Let $B^{r-1}(i)$ be the set of parameters of the location i in the loop $r-1$. The Processing Modules calculate $B^r(i)\cdots B^r(i+p-1)$ simultaniously. Thus, the p parallel processing reduces the number of the loop back operation and the length of shift registers of each Processing Modules from N to $(N+1)/p$. The number of inputs of Δ_r comp unit increases p times compared with that of the sequential architecture because the data of p locations are processed simultaniously. The required resources for the Δ_r comp unit is p times theoretically, too. On the other hand, the clock cycles are reduced from N to $(N+1)/p$.

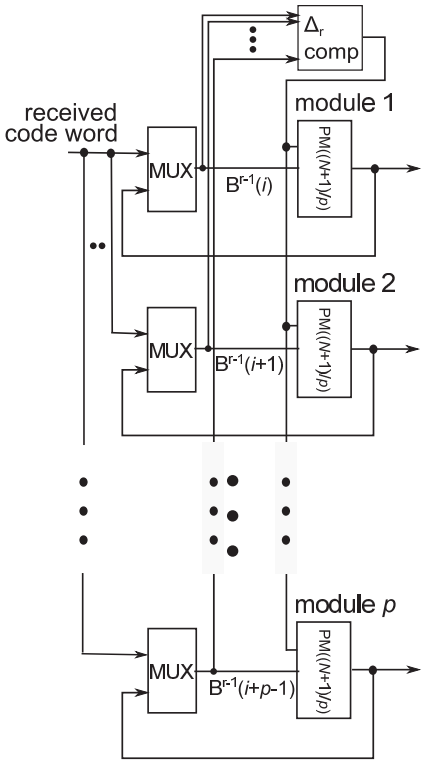

Fig. 5 p parallel architecture.

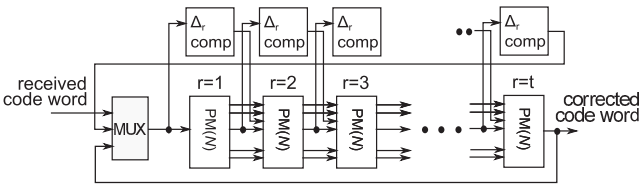

Fig. 6 Truncated arrays.

In this case, data throughput TH (bit/s) is expressed by the following formula.

$$TH = \frac{fnk}{2t((N+1)p^{-1}+1)}. \tag{4}$$

The p parallel architecture requires p Processing Modules and p MUXs. It causes area overhead. The number of whole registers for storing the parameters is not changed because the number of parameter values is not changed even if the architecture is modified for parallel processing theoretically.

3.2.4 Truncated Arrays

The pipelined architecture can be truncated. An array consisting of only t processing modules can be used and the data recycled twice through the array as shown in Fig. 6. Data throughput and hardware are halved in this case. It is possible to truncate the array further.

4. Evaluation

Performance of time domain Reed Solomon decoder is analyzed compared with frequency domain Reed Solomon decoder on FPGA. The performance of Reed Solomon decoder depends on the number of the information symbols per block size. Section 4.1 evaluates the sequential architecture. Sections 4.2, 4.3, and 4.4 analyze the performance of the pipeline architecture, the parallel architecture, and the truncated arrays, respectively. In this evaluation, the block size N is fixed to 255. The bit width of an information symbol n is 8 bit. Each hardware architecture of the time domain Reed Solomon decoder is described by VHDL. QuartusII 13.1 Web Edition and ModelSim-Altera 10.1d are used for the implementation. The target devices are CycloneIII, and CycloneIVGX. The clock frequency of CycloneIII and CycloneIVGX are 50MHz and 100 MHz, respectively. Altera Reed-Solomon II IP Core [17] is used as the frequency domain Reed Solomon Decoder for this evaluation. Here, the two parameters for the evaluation, the ratio of data throughput R_{TH} and the ratio of consumed resources R_N, are defined as follows.

$$R_{TH} = \frac{TH_T}{TH_F}, \tag{5}$$

where TH_T and TH_F are data throughput of the evaluated time and frequency domain Reed Solomon decoder, respectively.

$$R_N = \frac{N_T}{N_F}, \tag{6}$$

where N_T and N_F are the number of logic elements used for the syntheses of the time and frequency domain Reed Solomon decoder, respectively.

R_{TH} indicates the ratio of the data throughput of the frequency domain decoder against the time domain decoder.

4.1 Sequential Architecture

This section evaluates the basic sequential architecture. In this evaluation, the block size is fixed to 255. First, the data throughput and the number of consumed resources, is evaluated when the size of information symbols k = 249, 243, 239, 233, and 223. Data throughput is calculated from the Eq. (2). The number of consumed resources is obtained from the synthesis result.

Table 1 shows the evaluation result. As size of information symbol increases, data throughput increases. The column CIII is the results of Cyclone III. The column CIV is the results of Cyclone IV. According to the result, data throughput of time domain decoder is lower than that of frequency domain decoder. The frequency domain decoder does require no iteration of calculation. On the other hand the time domain decoder requires $2t$ iterations. Therefore as the size of information symbols decreases the data throughput becomes lower. The value of R_{TH} is 0.08 when k=243.

Table 1 Data throughput of sequential architecture of time domain Reed Solomon Decoder (Mbit/s).

N	k	freq domain		time domain		R_{TH}	
		CIII	CIV	CIII	CIV	CIII	CIV
255	249	390.6	781.2	64.6	129.2	0.17	0.17
255	243	381.2	762.4	31.5	63.0	0.08	0.08
255	239	374.9	748.8	23.2	46.5	0.06	0.06
255	233	365.5	731.0	16.5	33.0	0.05	0.05
255	223	349.8	699.6	10.8	21.7	0.03	0.03

Table 2 Resources used for synthesizes of frequency and time domain decoder (N = 255).

k	res	freq domain		time domain		R_N	
		CIII	CIV	CIII	CIV	CIII	CIV
249	lelm	1,585	1,585	1,742	2,471	1.10	1.56
	comb	1,478	1,478	1,697	1,687	-	-
	reg	645	645	96	852	-	-
	mem	5,120	5,120	795	120	-	-
243	lelm	2,218	2,218	1,742	2,471	0.65	0.94
	comb	2,083	2,083	1,697	1,687	-	-
	reg	945	945	96	852	-	-
	mem	5,120	5,120	795	120	-	-
239	lelm	2,664	2,623	1,742	2,471	0.65	0.94
	comb	2,509	2,508	1,697	1,687	-	-
	reg	1,148	1,138	96	852	-	-
	mem	5.120	5,120	795	120	-	-
233	lelm	3,312	3,312	1,742	2,471	0.52	0.75
	comb	3,133	3,133	1,697	1,687	-	-
	reg	1,437	1,437	96	852	-	-
	mem	5,120	5,120	795	120	-	-
223	lelm	4,383	4,342	1,742	2,471	0.39	0.57
	comb	4,160	4,160	1,697	1,687	-	-
	reg	1,928	1,918	96	852	-	-
	mem	5,120	5,120	795	120	-	-

It indicates that data throughput of time domain decoder is less than 10%. The results of Cyclone IV is twice of those of Cyclone III. It is because the clock frequency of Cyclone IV is twice of that of Cyclone III.

Table 2 shows the evaluation result of the number of the consumed resources for the implementation. The rows, lelm, comb, reg, and mem, are the number of used logic elements, combinational functions, logic registers and memory bits, respectively. The number of the used logic elements of the frequency domain decoder increases as the size of information symbols increases. It indicates that larger amount of resources are required as t increases. On the other hand, the number of the logic elements of the time domain decoder is constant. It is because the time domain decoding uses the common processing modules even when t is changed. The value R_N is larger than 1 when k=249, which means that the area of the time domain decoder requires larger resources than the frequency decoder.

In case of Cyclone III, R_N is smaller than 1 when k is smaller than 243. It indicates that the area of processing module is lower than that of frequency domain decoder. In case of Cyclone III, the area of the time domain decoder is 65% of that of frequency decoder when k = 239. In case of Cyclone IV, the number of the used logic elements of the time domain decoder is 94% of that of frequency decoder

Table 3 Resources used for synthesis of fully pipelined architecture ($N = 255$).

k	res	time domain		R_N	
		CIII	CIV	CIII	CIV
249	lelm	5,815	6,422	3.7	4.1
	comb	5,511	4,358	-	-
	reg	495	2,292	-	-
	mem	4,770	720	-	-
243	lelm	10,639	19,260	4.7	8.7
	comb	10,047	10,028	-	-
	reg	985	10,156	-	-
	mem	9,540	1,440	-	-
239	lelm	13,809	25,353	5.2	9.7
	comb	13,025	13,049	-	-
	reg	1,317	13,548	-	-
	mem	12,720	1,920	-	-

when $k = 239$. Like this, R_N depends on the implemented devices. However as a whole, the number of the used logic elements of the time domain decoder tends to be smaller than that of the frequency domain decoder according to the evaluaton result. The number of used memory bits of the synthesis result of Cyclone III is larger than that of Cyclone IV. On the other hand, the number of used logic registers of the synthesis result of Cyclone IV is larger than that of Cyclone III. It is because the shift registers of D(N) of Fig. 1 are constructed using memory bits in case of Cyclone III, and those are constructed using logic registers in case of Cyclone IV.

4.2 Pipelined Architecture

This section evaluates the number of the consumed resources for the fully pipelined architecture when the size of information symbols $k = 249$, 243, and 239.

Theoretically, the data throughput of the typical single input channel Reed Solomon Decoder including Alteara *Reed* Solomon II IP Core is expressed by fnk/N (bit/s), where f is the clock frequency, n is the bit width of a code word, k is the number of information symbols, and N is the block size. On the other hand, the data throughput of the pipelined architecture of the time domain Reed Solomon Decoder is express by Eq. (3). When N is enough larger than 2, then we can approcimate that $N + 2 \simeq N$. Because $257 \simeq 255$, the data throughput of the time domain Reed Solomon Decoder can be approximated to be equal to that of the frequency domain Reed Solomon Decoder.

Table 3 shows the result of the evaluation. The number of all the required resources increases as k decreases. When k=239, R_N of Cyclone III and Cyclone IV is 5.2 and 9.7. It indicates that the area of the fully pipelined time domain decoder are 5.2 times of the frequency domain decoder in case of Cyclone III, and it is 9.7 times of the frequency domain decoder in case of Cyclone IV.

4.3 Parallel Architecture

This section evaluates the number of the consumed resources for the parallel architecture $RS(255, 239)$ when the parallel size p is 1, 2, 4, 8, and 16. Table 4 shows the evaluation result of data throughput. The data throughput increases linearly because k is fixed to 239. Therefore, we can conclude that data throughput is increase by p times when p parallel architecture is applied.

Table 4 Data throughput of $RS(255, 239)$ (Mbit/s).

p	CIII	CIV
1	23.2	46.5
2	46.3	92.6
4	91.9	183.8
8	181.1	362.1
16	351.5	702.9

Table 5 Resources used for synthesis of parallel architecture of $RS(255, 239)$.

device	p	lelm	comb	reg	mem
CIII	1	1,742	1,697	96	795
	2	3,095	2,983	147	1,590
	4	5,673	5,465	249	3,180
	8	11,018	10,618	453	6,360
	16	21,323	20,539	861	12,720
CIV	1	2,471	1,687	852	120
	2	4,559	3,007	1,668	240
	4	8,551	5,463	3,300	480
	8	16,700	10,540	6,564	960
	16	32,706	20,402	13,092	1,920

Table 5 shows the consumed resources. The data throughput of $RS(255, 239)$ of p parallel architecture can be approximated to be equal to the pipelined architecture of $RS(255, 239)$. The required resources of the p parallel architecture is larger than those of the pipelined architecture. It is because the p parallel architecture requires extra control logics. Those logics result in the increase of the consumed resources. The number of the consumed resources of the pipelined architecture is 35% and 22% smaller than those of the 16 parallel architecture when Cyclone III and Cyclone IV are targeted, respectively.

The Eq. (3), which is used for calculation of the data throughput of the fully pipelined architecture, can be approximated to fnk/N when N is enough larger than 2. On the other hand, Eq. (4) is transformed to $fnk/(N + 1 + 1)$ when $2t$ is substituted to p. When N is enough larger than 2, the data throughput is approximated to fnk/N. Accordingly, the data throughput of the fully pipelined architecture and $2t$ parallel architecture can be approximated to be equal to that of the frequency domain decoder.

Table 6 shows the consumed resources of the fully pipelined architecture and $2t$ parallel architecture of $RS(255, 253)$, $RS(255, 251)$, $RS(255, 247)$, and $RS(255, 239)$. The ratio of the number of the used resources for synthesis of the fully pipelined architecture against that of the $2t$ parallel architecture R_{N2} is defined as follows.

$$R_{N2} = N_{FP}/N_P, \tag{7}$$

where N_{FP} is the number of used resources for synthesis of the pipelined architecture and N_P is that of the $2t$ parallel architecture. R_{N2} of lelm and comb is smaller than 1.

Table 6 Data throughput (Mbit/s) and area of pipelined architecture and parallel architecture.

N	k	*data throughput*		*resource*	pipelined		parallel		R_{N2}	
		CIII	CIV		CIII	CIV	CIII	CIV	CIII	CIV
255	253	397	794	lelm	2,562	3,983	3,095	4,559	0.83	0.87
				comb	2,450	2,431	2,983	3,007	0.82	0.81
				reg	179	1,700	147	1,668	1.22	1.02
				mem	1,590	240	1,590	240	1,00	1.00
255	251	394	787	lelm	4,204	7,052	5,673	8,551	0.74	0.82
				comb	3,996	3,964	5,465	5,463	0.73	0.73
				reg	345	3,396	249	3,300	1.39	1.03
				mem	3,180	480	3,180	480	1.00	1.00
255	247	388	775	lelm	7,439	13,168	11,018	16,700	0.65	0.78
				comb	7,039	7,008	10,618	10,540	0.63	0.64
				reg	677	6,788	453	6,564	1.53	1.03
				mem	6,360	960	63,600	960	1.00	1.00
255	239	375	750	lelm	13,809	25,353	21,323	32,706	0.65	0.78
				comb	13,025	13,049	20,539	20,402	0.63	0.64
				reg	1,317	13,548	861	13,092	1.53	1.03
				mem	12,720	1,920	12,720	1,920	1.00	1.00
average of R_{N2}				lelm	-	-	-	-	0.72	0.82
				comb	-	-	-	-	0.71	0.71
				reg	-	-	-	-	1.41	1.03
				mem	-	-	-	-	1.00	1.00

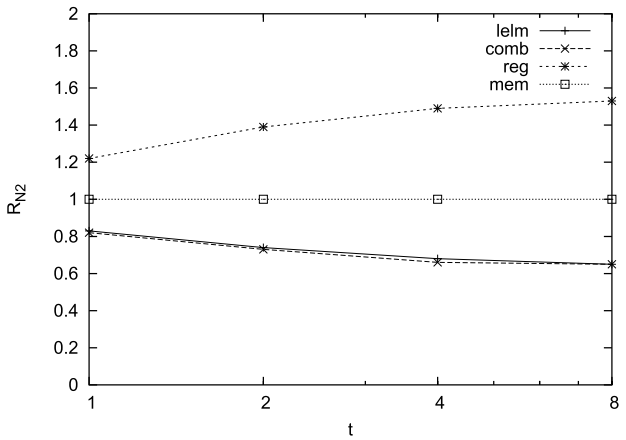

Fig. 7 t specification of R_{N2} of lelm, comb,reg, and mem.

They indicate the number of the logic elements, the combinational functions used for synthesis of the pipelined architecture is smaller than that of the $2t$ parallel architecture. It is because the p parallel architecture requires extra control logics. Those logics result in the increase of the consumed resources. On average, the number of used logic element of the fully pipelined architecture of Cyclone III and Cyclone IV are 28% and 18% smaller than those of the $2t$ parallel architecture, respectively. On the other hand, the number of the logic registers used for synthesis of the fully pipelined architecture is larger than that of the $2t$ parallel architecture. It indicates the number of the logic registers used for synthesis of the pipelined architecture is larger than that of the $2t$ parallel architecture. The number of the registers for D(N) is proportional to the number of the pipeline stages of the fully pipelined architecture. On the other hand the number of the registers for D(N) does not depend on p of the parallel architecture. The result is affected by the difference. The number of the memory bits is constant. Figure 7 plots t specification of R_{N2} of lelm, comb, reg, and mem. Although R_{N2} of the logic register increases as t increases, R_{N2} of logic elements decreases.

Table 7 Used resources for synthesis of truncated arrays.

res	CIII	CIV
lelm	7,472	13,206
comb	7,072	7,046
reg	677	6,788
mem	6,360	960

4.4 Truncated Arrays

Table 7 shows the result of the evaluation of the consumed resources of the truncated arrays of $RS(255, 239)$. The number of the consumed resources is about the half of the fully pipelined architecture. Concretely, the number of the consumed resources of the truncated arrays of $RS(255, 239)$ are 43% and 46% smaller than those of the fully pipelined architecture. However the data throughput is halved, too.

4.5 Discussion

The merit of the time domain Reed Solomon Decoder comapred with typical frequency doman Reed Solomon Decoder is as follows.

Universality: The time domain Reed Solomon Decoder is an universal Reed Solomon Decoder [1]. This is a good point. The universal decoder can decode the encoded codeword including arbitrary information symbols unlike the frequency domain Reed Solomon Decoder. Especially the sequential architecture shown in Fig. 2 can decode arbitrary encoded codeword with single Processing Modules. We can conclude that it is area efficient compared with typical frequency domain Reed Solomon Decoder. The evaluation result of the area

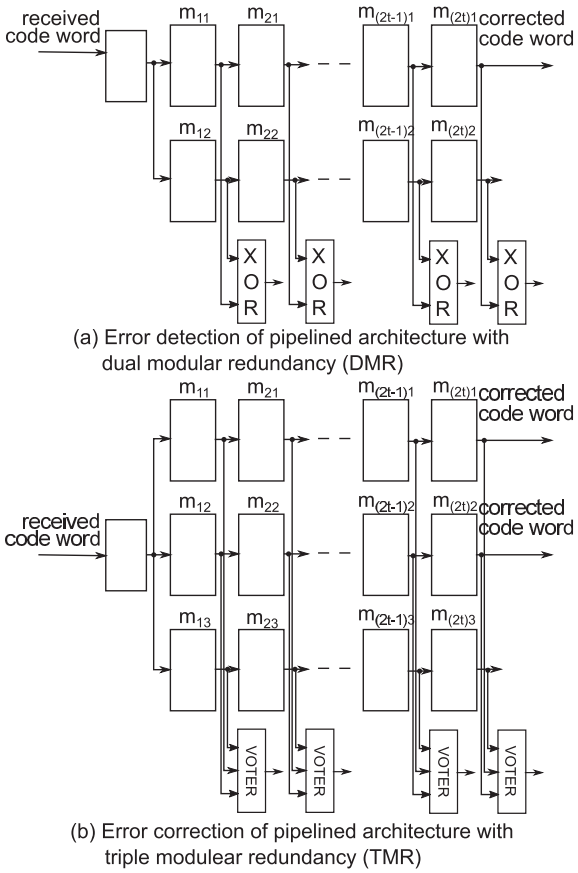

Fig. 8 Pipelined architecture of time domain decoder applied DMR (a) and TMR (b).

efficiency of the sequential architecture of the time domain Reed Solomon decoder is shown in Table 2.

Simplicity of hardware architecture: Because the algorithm can be implemented by the repeated application of a single operation, it is easy to implement on hardware. Especially, the devices with regular structure such as FPGA and GPGPU [18] is suitable for the implementation.

Ease to apply techniques for dependable design: Because hardware architecture consisits of simple controller and simple and regular Processing Modules, it is easy to apply fault tolerant techniques such as dual modular redundancy (DMR), or triple modular redundancy (TMR) [19]. Especially, it is good point for the implementation using SRAM-based FPGA, which is vulnerble to soft error occuring during normal operation [16]. Figure 8 depicts the pipelined architecture of the time domain Reed Solomon Decoder applied DMR and TMR. The above statements are added to page 4, right, 3rd line of the revised manuscript, too.

5. Conclusion

This paper has analyzed the time domain Reed Solomon Decoder with FPGA implementation. Data throughput and area is carefully evaluated compared with typical frequency domain Reed Solomon Decoder. In this analysis, three hardware architecture to enhance the data throughput, namely, the pipelined architecture, the parallel architecture, and the truncated arrays, have been evaluated, too. The evaluation reveals that the number of the consumed resources of $RS(255, 239)$ is about 20% smaller than those of the frequency domain decoder although data throughput is less than 10% of the frequency domain decoder. The number of the consumed resources of the pipelined architecture is 28% smaller than that of the parallel architecture when data throughput is same. It is because the pipeline architecture requires less extra logics than the parallel architecture. To get higher data throughput, the pipelined architecture is better than the parallel architecture from the viewpoint of consumed resources.

References

[1] R.E. Blahut, "A universal reed-solomon decoder," IBM Journal of Research and Development, vol.28, no.2, pp.150–158, March 1984.
[2] S. Choomchuay and B. Arambepola, "Time domain algorithms and architechures for reed-solomn decoding," IEE Proceedings-I, vol.140, no.3, pp.150–158, June 1984.
[3] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of reed-solomon codes," IEEE Trans. Inf. Theory, vol.49, no.11, pp.2809–2825, Nov. 2003.
[4] J.S. Yedidia, W.T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," IEEE Transactions on Information Theory, vol.51, no.7, pp.2282–2312, 2005.
[5] J. Jiang and K.R. Narayanan, "Iterative soft decoding of reed solomon codes," IEEE Commun. Lett., vol.8, no.4, pp.244–246, May 2004.
[6] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," IEEE Trans. Inf. Theory, vol.42, no.2, pp.429–445, March 1996.
[7] R. Lucas, M. Bossert, and M. Breitbach, "On iterative soft-decision decoding of linear binary block codes and product codes," IEEE J. Sel. Areas Commun., vol.16, no.2, pp.276–296, Feb. 1998.
[8] J. Bellorado, A. Kavcic, M. Marrow, and L. Ping, "Low-complexity soft-decoding algorithms for reed–solomon codes—part II: Soft-input soft-output iterative decoding," IEEE Trans. Inf. Theory, vol.56, no.3, pp.960–967, March 2010.
[9] S.D. Mhaske, U. Ghodeswar, and G.G. Sarate, "Design of area efficient reed solomon decoder," IEEE 2nd International Conference on Devices, Circuits and Systems (ICDCS'14), pp.1–4, March 2014.
[10] P. Dayal and R.K. Patial, "Implementation of reed-solomon codec for IEEE 802.16 network using VHDL code," IEEE International Conference on Reliability Optimization and Information Technology (ICROIT'14), pp.452–455, Feb. 2014.
[11] P. Jiang, B. Gao, Z. Xiao, L. Su, and D. Jin, "Multigigabit rs-cc decoder for 60-GHz systems," IEEE International Conference on Computational Problem-Solving (ICCP'13), pp.207–210, Oct. 2013.
[12] P. Jiang, B. Gao, Z. Xiao, L. Su, and D. Jin, "Multigigabit rs-cc decoder for 60-GHz systems," IEEE International Conference on Computational Problem-Solving (ICCP'13), pp.207–210, Oct. 2013.
[13] P. Parvathi and P.R. Rasad, "FPGA based design and implementation of reed-solomon encoder & decoder for error detection and correction," IEEE Conference on Power, Control, Communication and Computational Technologies for Sustainable Growth (PCCCTSG'15), pp.261–266, Dec. 2015.
[14] T.C. Barbosa, R.L. More, T.C. Pereira, and L.H.C. Ferreira, "FPGA implementation of a reed-solomon codec for otn g.709 standard

with reduced decoder area," IEEE 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM'10), pp.1–4, Sept. 2010.
[15] B. Tiwari and R. Mehra, "Design and implementation of reed solomon decoder for 802.16 network using FPGA," IEEE International Conference on Signal Processing, Coumputing and Control (ISPCC'12), pp.1–5, March 2012.
[16] M.T. Leipnitz, L.H. Geferson, and G.L. Nazar, "A fault injection platform for FPGA-based communication systems," IEEE 7th Latin American Symposium on Circuits and Systems (LASCAS'16), pp.59–62, March 2016.
[17] Altera Inc., http://www.altera.com, Reed-Solomon II IP Core User Guide, UG-01090 ed., May 2016.
[18] Nvidia Inc., http://www.nvidia.com.
[19] D. Pradhan, Fault Tolerant Computer System Design, Prentice Hall, New Jersey, 1995.

Kentaro Kato received the B.E. and M.E. degrees from Nagoya University, Nagoya, Japan, in 1997 and 1999, respectively, and the Ph.D. degree from Chiba University, Chiba, Japan, in 2009. In 1999, he joined Fujitsu Limited and engaged in the development of the embedded control system of HDD from 1999 to 2001. He joined Chiba University, in 2001. Since 2011, he has been a member of National Institute of Technology, Tsuruoka College. He is currently an Associate Professor with the Creative Engineering Department. He was a visiting researcher of Gunma University, Japan, in 2013. He was a visiting scholar of King Monkut's Institute of Technology, Ladkrabang, Thailand, in 2015. His research interests include design for logic testing method, testability of mixed signal SoC, and fault-tolerant design of reconfigurable hardware, SoC, and Biomedical equipments. He is a member of the IEEE.

Somsak Choomchuay received M.Phil. and Ph.D. from Imperial college, University of London, United Kingdom in 1994. Since then he had joined King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. He was promoted to an Associate Professor in Electronic Engineering in 2000. His current research interests are data security, error control codes, and applications of signal processing in Biomedical Engineering. Dr. Somsak is a member of ECTI (Thailand) and IEEE (USA).