Multi-Symbol-Sliced Dynamically Reconfigurable Reed–Solomon Decoder Design Based on Unified Finite-Field Processing Element

Huai-Yi Hsu, *Student Member, IEEE*, Jih-Chiang Yeo, and An-Yeu Wu, *Member, IEEE*

Abstract—Reed–Solomon (RS) codes play an important role in providing the error correction and the data integrity in various communication/storage applications. For high-speed applications, most RS decoders are implemented as dedicated application-specified integrated circuits (ASICs) based on parallel architectures, which can deliver high data throughput rate. For lower-speed applications, the RS decoding operations are usually performed by using fine-grained processing elements (PE) controlled by a programmable digital signal processing (DSP) core, which provides high flexibility. In this paper, we propose a novel m -PE multi-symbol-sliced (MSS) RS datapath structure. The m -PE RS architecture is a highly scalable design and can be dynamically reconfigured at 1-PE, 2-PE, $\dots$, $m/2$ -PE, and m -PE modes to deliver necessary data throughput rate. With the help of the gated-clock scheme to turn off the idle PEs, the proposed runtime configurable ASIC design provides good tradeoff between the data throughput rate and the power consumption. Hence, it can save energy to extend the battery life of the portable devices. We demonstrate a prototyping design using 4 PEs by using UMC 0.18- μm CMOS technology. The design can be dynamically reconfigured to be operated at 1-PE, 2-PE, and 4-PE modes, with performance of 140 Mb/s at 18.91 mW, 280 Mb/s at 28.77 mW, and 560 Mb/s at 48.47 mW, respectively. Compared with existing RS designs, the proposed m -PE RS decoder has better normalized area/power efficiency than most DSP-type and ASIC-type RS designs. The reconfigurable feature makes our design a good candidate for the error control coding (ECC) unit of the storage system in power-aware portable devices.

Index Terms—Dynamically reconfigurable architectures, multi-symbol-sliced, processing elements, Reed–Solomon codes.

I. INTRODUCTION

NOWADAYS, the portable or handheld devices (PDA, Smart Phones, iPod) are very popular in our daily life and users can run a variety of multimedia/entertainment applications. As the complexity of the applications increase, the demand of storage capacity is also increasing. Currently, 60 GB of micro hard drive and 2 GB of Flash memory become standard specifications for high-end handheld devices [1]. However,

Manuscript received July 25, 2005; revised November 18, 2005. This work was supported in part by MediaTek Inc. and by National Science Council, R.O.C., under Grant NSC 92-2220-E-002-012.

H.-Y. Hsu and A.-Y. Wu are with the Graduate Institute of Electronics Engineering, Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: yuki@access.ee.ntu.edu.tw; andywu@cc.ee.ntu.edu.tw).

J.-C. Yeo was with the Graduate Institute of Electronics Engineering, Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. He is now with the Digital IC Design Department I, Realtek Semiconductor Corporation, Hsinchu 300, Taiwan, R.O.C. (e-mail: james@access.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TVLSI.2006.876102

Fig. 1. Spectrum of VLSI implementations in terms of flexibility, power efficiency, and silicon cost.

since the Flash memory and micro hard drive are manufactured in nanometer scale, the structural/manufacturing faults become inevitable. Currently, the error control coding (ECC) [2] unit is a common scheme in DRAM and Flash memory systems to provide data integrity. However, the correcting capability is limited since most of them still use BCH codec as the forward error correcting (FEC) processing engine. As the manufacturing defects of memory/storage become inevitable and the data amount and throughput rate increases, the reliability of data transmission on the mobile devices has become more and more important. Hence, a better and robust FEC mechanism should be employed in the ECC unit of the storage system in future portable/hand-held devices. Reed–Solomon (RS) codec [3]–[5] is a widely adopted FEC technique that has an excellent error correction capability against burst errors. It has been adopted by many communication/storage systems, such as computer memory storage, magnetic and optical recording, wireless mobile and satellite communications. Hence, it becomes a good candidate in the design of ECC units.

In portable/hand-held devices, power efficiency always plays the key role to extend the standby/operating period. Currently, the hardware solutions can be categorized into three types: application-specified integrated circuits (ASICs), reconfigurable designs, and digital signal processing (DSP)-type solutions [12]. From Fig. 1, we can see that they provide tradeoff among flexibility, power efficiency, and silicon area. In general, for low-speed applications (e.g., error-correcting for MP3/voice decoding), existing ASIC-type RS solutions [13], [14] provide much more computing power than required. Hence, it can be considered as an over-design while using ASIC-type designs in low-speed applications. On the other hand, for high-speed applications (e.g., error-correcting for movie decoding), the DSP-type solutions [15]–[17] need much more cycles at very high operating frequency, which can lead to high-power consumption. That is, both designs play the two extreme roles

Fig. 2. Applications of the proposed dynamically reconfigurable RS architecture.

in Fig. 1, and they usually do not have runtime-controlled power-saving mechanisms due to the constraints of their VLSI architectures.

In this paper, we propose a novel m -PE multi-symbol-sliced (MSS) RS datapath structure, which is classified as the reconfigurable Silicon IP (SIP) in Fig. 1. The m -PE RS architecture is a highly scalable design, and can be dynamically reconfigured at 1-PE, 2-PE, ..., $m/2$ -PE, and m -PE modes. That is, it can deliver different data throughput rate by activating the number of PEs based on the design flowchart of Fig. 1. The applications of the proposed design are illustrated in Fig. 2. We have one-PE, partial-PE, and full-PE modes to meet different required throughput rates of various target applications. Meanwhile, the idle PEs can be turned off through gated-clock schemes to save energy. Hence, the proposed runtime configurable ASIC design provides good tradeoff between the data throughput rate and the power dissipation. Therefore, the proposed MSS-RS design is a good candidate to extend the battery life of portable devices.

Finally, we demonstrate a prototyping VLSI design using four processing elements (4-PE MSS-RS data-path architecture) by using UMC 0.18- μ m CMOS standard-cell library. The design can be dynamically reconfigured to be operated at 1-PE, 2-PE, and 4-PE modes, with performance of 140 Mb/s at 18.91 mW, 280 Mb/s at 28.77 mW, and 560 Mb/s at 48.47 mW, respectively. Compared with existing RS designs, the proposed m -PE RS decoder has better performance indices, such as normalized efficiency (NAE) and normalized power efficiency (NPE), than most DSP- and ASIC-type RS designs.

The rest of this paper is organized as follows. In Section II, we investigate the syndrome-based RS decoding algorithm and design three different basic finite-field operators (FFO). In Section III, we deliver a new coarse-grained finite-field MSS-PE by using the FFOs. Moreover, we propose a dynamically reconfigurable RS decoding architecture. The detailed operations of the proposed MSS-RS decoder are discussed in Section IV. Section V shows the performance and comparisons. Finally, we conclude our works in Section VI.

II. PROPOSED MSS PROCESSING ELEMENT

The syndrome-based RS decoding process consists of syndrome calculation (SC), key equation solving (KES), and error

Fig. 3. Block diagram of a syndrome-based RS decoder.

correction (EC), as shown in Fig. 3 [4]–[6]. In this work, we employ the Modified Euclidean (ME) algorithm [18] to solve the key equation. The advantage of the ME-based RS decoder is its pipelinability for high-throughput decoding. In this section, we analyze the three major DSP modules of Fig. 3. By the algorithmic analysis of each module, we derive three new finite-field operators (FFO) that correspond to the basic operation kernels of these three modules. The FFOs will be applied to our MSS Processing Element (PE) design in Section III.

A. Syndrome Calculation

The encoded RS codeword consists of n finite-field symbols, $c_0, c_1, c_2, \dots, c_{n-1}$, and $c(x)$ denotes the polynomial representation of this codeword. We also define the received code polynomial and the error value polynomial as $r(x)$ and $e(x)$, respectively. Then, $r(x) = c(x) + e(x)$. The syndrome calculation (SC) module receives the RS code from channel, and computes the syndrome values from the $(2t - 1)$ -degree syndrome polynomial

$$S(x) = \sum_{i=0}^{2t-1} S_i x^i \quad (1)$$

where S_i is the syndrome values and t denotes the error correcting capability. Then, S_i can be calculated by

$$S_i = r(\alpha^i) = \sum_{j=0}^{n-1} r_j \alpha^{ij}, \quad i = 0, 1, \dots, 2t - 1 \quad (2)$$

where $\alpha^0, \alpha^1, \dots, \alpha^{2t-1}$ are the roots of the generation polynomial in RS encoder. Equation (2) can be represented in a recursive format as

$$S_i = r(\alpha^i) = (\dots ((r_{n-1} \alpha^i + r_{n-2}) \alpha^i + r_{n-3}) \dots) \alpha^i + r_0. \quad (3)$$

Equation (3) is an iterative process and can be implemented with the hardware architecture of Fig. 4(a). We can see that the basic operator of the syndrome calculation consists of one finite-field multiplier (FFM) and one finite-field adder (FFA). The basic finite-field operator (FFO) corresponding to the syndrome calculation module (we call it SC-FFO) is shown in Fig. 4(b). Each iteration of the SC module takes $2t$ SC-FFO operations.

B. Modified Euclidean Algorithm for Solving Key Equation

The key equation solving (KES) module is the most critical/complicated part of the RS decoder. Assume that there are ν

Fig. 4. (a) Over architecture of the SC module. (b) The corresponding basic FFO (SC-FFO).

error symbols in the received code block with $\nu \leq t$. We can define the error location polynomial $\sigma(x)$ as

$$\sigma(x) = \sum_{i=0}^{\nu} \sigma_i x^i. \quad (4)$$

If there is an error at the coefficient r_k of the received code polynomial $r(x)$, we have $\sigma(\alpha^{-k}) = 0$. We can also define the error magnitude polynomial $\omega(x)$ as

$$\omega(x) = \sum_{i=0}^{\nu-1} \omega_i x^i. \quad (5)$$

The error location polynomial and the error magnitude polynomial can be obtained by solving the key equation below

$$S(x)\sigma(x) = \omega(x) \bmod (x^{2t}). \quad (6)$$

In this paper, we employ the modified Euclidean (ME) algorithm [18] to solve the key equation. The ME algorithm can be explained by following iteration equations.

1) Initial conditions:

$$\begin{cases} R_0(x) = x^{2t}, \\ Q_0(x) = S(x), \end{cases} \begin{cases} L_0(x) = 0; \\ U_0(x) = 1. \end{cases} \quad (7)$$

2) Updating equations in each iteration:

$$\begin{cases} R_i(x) = sw_{i-1} \left(\begin{bmatrix} b_{i-1}R_{i-1}(x) \\ a_{i-1}Q_{i-1}(x) \end{bmatrix} + x^{|l_{i-1}|} \begin{bmatrix} a_{i-1}Q_{i-1}(x) \\ b_{i-1}R_{i-1}(x) \end{bmatrix} \right) \\ Q_i(x) = sw_{i-1} \left(\begin{bmatrix} Q_{i-1}(x) \\ R_{i-1}(x) \end{bmatrix} \right) \end{cases} \quad (8)$$

$$\begin{cases} L_i(x) = sw_{i-1} \left(\begin{bmatrix} b_{i-1}L_{i-1}(x) \\ a_{i-1}U_{i-1}(x) \end{bmatrix} + x^{|l_{i-1}|} \begin{bmatrix} a_{i-1}U_{i-1}(x) \\ b_{i-1}L_{i-1}(x) \end{bmatrix} \right) \\ U_i(x) = sw_{i-1} \left(\begin{bmatrix} U_{i-1}(x) \\ L_{i-1}(x) \end{bmatrix} \right) \end{cases} \quad (9)$$

where

$$\begin{cases} l_{i-1} = \deg(R_{i-1}(x)) - \deg(Q_{i-1}(x)) \\ sw_{i-1} = \begin{cases} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, & \text{if } l_{i-1} \geq 0; \\ \begin{bmatrix} 0 & 1 \end{bmatrix}, & \text{if } l_{i-1} < 0. \end{cases} \end{cases} \quad (10)$$

3) Stop condition:

$$\deg(R_{i-1}(x)) < t. \quad (11)$$

4) Output assignments:

$$\omega(x) = Q_{i-1}(x), \quad \sigma(x) = U_{i-1}(x) \quad (12)$$

where “ a_{i-1} ” and “ b_{i-1} ” denote the leading coefficients of $R_{i-1}(x)$ and $Q_{i-1}(x)$, respectively.

Equations (8) and (9) can be implemented with the hardware architecture shown in Fig. 5(a) and (b), respectively. The basic operations of Fig. 5(a) and (b) are two FFMs and one FFA. Since (8) and (9) are calculated concurrently, we use four FFMs and two FFAs to construct the basic finite-field operator for computing the ME algorithm (we call it ME-FFO) as shown in Fig. 5(c). Each iteration of the ME algorithm takes $2t$ ME-FFO operations.

C. Error Correction (EC)

The error correction (EC) module finds the locations of errors by checking if $\sigma(\alpha^{-k}) = 0$, for $k = 0, 1, 2, \dots, n-1$. This is called Chien’s search, which evaluates

$$\sigma(\alpha^{-k}) = \sum_{i=0}^{\nu} \sigma_i \cdot (\alpha^{-k})^i, \quad \text{where } k = 0, 1, \dots, n-1. \quad (13)$$

Since the maximum number of errors ν equals to the error correcting capability t , we can replace ν with t and set $\sigma_i = 0$ for $\nu < i \leq t$ in (13). It can be represented in an iterative form as

$$\sigma(\alpha^{-k}) = \sum_{i=0}^t \sigma_i \cdot (\alpha^i)^{-k} = \sum_{i=0}^t \sigma_i \cdot (\alpha^i)^{255-k} \quad (14)$$

where $k = 0, 1, \dots, n-1$.

Then, the values of $\sigma(\alpha^{-k})$ can be calculated sequentially by the following iterative operations.

1) Initial conditions:

$$\Delta_{i(k=n)} = \sigma_i \cdot (\alpha^i)^{255-n}, \quad i = 0, 1, \dots, t. \quad (15)$$

2) Iterative operations:

$$\begin{cases} \Delta_{ik} = \Delta_{i(k+1)} \cdot \alpha^i, & i = 0, 1, \dots, t \\ \sigma(\alpha^{-k}) = \Delta_{0k} + \Delta_{1k} + \dots + \Delta_{tk} \end{cases} \quad (16)$$

where $k = n-1, n-2, \dots, 0$.

Fig. 5. (a) and (b): Architectures of the ME algorithm. (c) Basic FFO corresponding to the ME algorithm (ME-FFO).

To calculate the error value, the Forney algorithm is used. If there is error at the coefficient r_k in $r(x)$, the error value at r_k can be calculated by

$$\begin{aligned} Er(k) &= \frac{\omega(\alpha^{-k})}{\sigma'(\alpha^{-k})} \\ &= \frac{\omega_0 + \omega_1 \cdot (\alpha^{-1})^k + \dots + \omega_{t-1} \cdot (\alpha^{-(t-1)})^k}{\sigma_1 \cdot (\alpha^{-1})^k + \sigma_3 \cdot (\alpha^{-3})^k + \dots + \sigma_{t-1} \cdot (\alpha^{-(t-1)})^k}. \end{aligned} \quad (17)$$

The denominator sums half of the Δ_{ik} , which are odd parts of the coefficients of $\sigma(\alpha^{-k})$. This is part of the computation in the Chien's search. In considering the hardware sharing, we do not need additional hardware to evaluate this denominator. The

numerator evaluates the error magnitude polynomial by setting $x = \alpha^{-k}$.

Similarly, we can evaluate the error location polynomial by

$$\omega(\alpha^{-k}) = \sum_{i=0}^{t-1} \omega_i \cdot (\alpha^{-i})^k = \sum_{i=0}^{t-1} \omega_i \cdot (\alpha^i)^{255-k} \quad (18)$$

where $\alpha^{255} = 1$.

The values of $\omega(\alpha^{-k})$ can be calculated sequentially by the following iterative operations.

1) Initial conditions:

$$W_{i(k=n)} = \omega_i \cdot (\alpha^i)^{255-n}, \quad i = 0, 1, \dots, t-1. \quad (19)$$

Fig. 6. (a) Architecture of EC. (b) Basic FFO corresponding to EC (EC-FFO).

2) Iterative operations:

$$\begin{cases} W_{ik} = W_{i(k+1)} \cdot \alpha^i, & i = 0, 1, \dots, t-1 \\ \omega(\alpha^{-k}) = W_{0k} + W_{1k} + \dots + W_{(t-1)k} \end{cases} \quad (20)$$

where $k = n-1, n-2, \dots, 0$.

Finally, with both Chien's search and the Forney algorithm, the EC module can be implemented using the hardware architecture of Fig. 6(a). The basic operator consists of one FFM and one FFA except for the finite-field divisor (FFD). The basic FFO corresponding to the EC module (we call it EC-FFO) is shown in Fig. 6(b). Each iteration of EC takes $2t$ EC-FFO operations.

III. DYNAMICALLY RECONFIGURABLE REED–SOLOMON DECODER BASED ON PROPOSED MSS DATA-PATH ARCHITECTURE

A. Unified Finite-Field PE Design

By investigating the three major modules of the RS decoding procedure—SC, ME, and EC—we can modify each of them into an iterative form. Then, using the three basic operators (SC-FFO, ME-FFO, and EC-FFO), we can construct the system architecture of a parallel RS decoder as shown in Fig. 7(a). All three stages use iterative computation. In each iteration, the RS decoder reads in one symbol for syndrome calculation and outputs one symbol by error correction.

Since the number of SC-FFOs, the number of ME-FFOs, and the number of EC-FFOs in Fig. 7(a) are all $2t$, the parallel RS decoder can be divided into $2t$ horizontal hardware slices. By applying the bit-sliced datapath concept [24], we can define a symbol-sliced datapath. Each slice consists of one SC-FFO, one ME-FFO, and one EC-FFO. Fig. 7(b) shows the unified processing element (PE) design and the PE can simultaneously perform the SC, ME, and EC operations. We can use $2t$ PEs to construct the parallel RS decoding architecture in Fig. 7(a).

B. Dynamically Reconfigurable RS Decoder Design Based on Unified PE

Based on the newly derived Unified-PE, we propose a scalable design methodology to construct a dynamically reconfigurable RS decoder. Fig. 8 shows the system block diagram of a generalized reconfigurable RS decoder. It consists of the m -PE array, the data register bank, the datapath units, and the controller. The detailed operations are given below.

- 1) *m*-PE array: This array consists of m MSS-PEs which can perform m SC operations, m ME operations, and m EC operations simultaneously. Based on requirement of the system data throughput rate, we can decide the number of PEs within the chip. With more PEs running in parallel, the decoding throughput can be higher.
- 2) *Data register bank*: The data register bank includes $2t$ register blocks based on the error correcting capability t . Each register block consists of six symbols of data register: one for the SC operation, four for the ME operation, and one for the EC operation.
- 3) *Datapath units*: The datapath units read the data of those selected register blocks as the input signals of the working PEs, and write back to those selected register blocks with the output signals of those working PEs.
- 4) *Controller*: The controller is in charge of the address generator and the coefficient generator for controlling the datapath access and generating the multipliers' coefficients.

C. Direct-Mapping Data Path RS Architecture

Assume that the number of processing elements is m . A direct-mapping reconfigurable architecture of RS decoder design is shown in Fig. 9. For each PE, this design uses $2t$ -to-1 multiplexers to select one register block from the whole $2t$ register blocks to be the inputs of the PE. For each register block, this design uses 1-to- $2t$ demultiplexers to select one signal block from the m outputs of PEs to refresh the data of the register block. The total delay on data-path control is $2\log_2(t+3)$.

Fig. 7. (a) System architecture of parallel RS decoder using FFOs. (b) The proposed unified finite-field processing element.

Fig. 8. Block diagram of a MSS-PE based reconfigurable RS decoder.

2-to-1 multiplexers delay. The total data-path control circuit takes $(24mt - 6m)$ 2-to-1 symbol-wise multiplexers.

While in single-PE mode, all $2t$ register blocks are processed by PE_0 sequentially. PE_0 uses its own $2t$ -to-1 symbol-wise multiplexers to choose the data register block to process, and output back to the same register block through the 1-to- $2t$ output data-path symbol-wise demultiplexers. While in 2-PE mode, t register blocks are processed by PE_0 sequentially, and the other t register blocks are processed by PE_1 , sequentially. Other cases are similar to the single-PE and 2-PE modes. This is a very straightforward solution to reconfigure the PE array. However,

the loading of all PEs is not balanced, and the datapath design is complicated. This will cause the higher hardware cost and slow down the processing speed.

D. Proposed Multisymbol-Sliced (MSS) Data-Path Architecture

The direct-mapping architecture always uses the same PEs to process all data. Hence, the PEs need to choose one register block as the input from all register blocks at all the time. As the result, each PE needs $2t$ -to-1 multiplexers at its input and each register needs 1-to- $2t$ demultiplexers to choose one output

Fig. 9. Direct-mapping method of dynamically reconfigurable RS decoder using the PE.

Fig. 10. MSS datapath architecture of the dynamically reconfigurable m -PE RS decoder.

signal block from all the m processing elements. To simplify the datapath design, we propose the multisymbol sliced (MSS) datapath architecture. By splitting all the $2t$ data register blocks into m groups, and each group consists of $2t/m$ data register blocks, we can modify the direct-mapping architecture to a more efficient format—the MSS data-path RS architecture.

Fig. 10 shows the proposed dynamically reconfigurable MSS-RS decoder design. The total $2t$ register blocks are sliced into m multisymbol groups. For each data register group, we assign one PE to be responsible for all the operations performed on the data register blocks in this group. While in single-PE mode ($m = 1$), each time only one PE is in operation, which is similar to the direct-mapping architecture. However, we do not reuse the same PE to process all data. We make every PE operate sequentially and each PE needs to operate $2t/1 = 2t$ times, for processing its own $2t/1 = 2t$ data register blocks. While in 2-PE mode ($m = 2$), each time only two PEs are active and every PE still needs to operate $2t/2 = t$ times to process its own $2t/2 = t$ data register blocks. Other cases are similar to these two operating modes. The difference between these two operating modes is the operating timing for the PEs

TABLE I
COMPARISON BETWEEN THE DIRECT-MAPPING AND THE PROPOSED MSS DATA-PATH ARCHITECTURE

Datapath Architecture	Critical Path Delay (2-to-1 multiplexer's delay)	Hardware Cost (2-to-1 symbol-wise multiplexer)
Direct Mapping	$2\log_2 t + 3$	$24mt - 6m$
Proposed MSS for m -PE	$2\log_2 t - 2\log_2 m + 3$	$24t - 6m$
Reduced amount in hardware	$2\log_2 m$	$24t(m - 1)$

processing data, which will be discussed in detail in the next section.

For a given m value, each PE always operates $2t/m$ times to processing its own $2t/m$ data register blocks. As the result, each PE only needs $(2t/m)$ -to-1 multiplexers to select one register block from $2t/m$ register blocks as its input signals. The output signals will go through a 1-to- $(2t/m)$ demultiplexer to one of the PE's $2t/m$ data registers. The total delay on data-path control is $(2\log_2 t - 2\log_2 m + 3)$ 2-to-1 multiplexers delay. The total datapath control circuit takes $(24t - 6m)$ symbol-wise multiplexers. Compared with the direct-mapping architecture, the proposed MSS data-path architecture can save $(2\log_2 m)$ 2-to-1 multiplexers critical path delay, and $(24mt - 24t)$ 2-to-1 symbol-wise multiplexers hardware area cost, as shown in Table I.

There are two extreme cases of the MSS data-path architecture. When $m = 2t$, each data group consists of only one register block. The datapath multiplexers will be needless and the hardware architecture becomes a fully expanding parallel architecture. Another special case happens when $m = 1$. The MSS data-path architecture with only one PE will be the same as the single-PE data-multiplexing architecture.

IV. OPERATIONS OF THE DYNAMICALLY RECONFIGURABLE REED–SOLOMON DECODER

The proposed MSS data-path structure can be dynamically reconfigured to operate at different modes for various RS specification requirement. In this section, we show the detailed operations of the proposed dynamically reconfigurable RS decoding architecture.

A. Dynamically Reconfigurable Property for d -PE Mode

Assume the m embedded PEs are $PE_0, PE_1, \dots, PE_{m-2}$, and PE_{m-1} , and the $2t$ register blocks are $Reg_0, Reg_1, \dots, Reg_{2t-2}$, and Reg_{2t-1} . For any m value, every PE just operates $2t/m$ times for processing its own $2t/m$ data register blocks. For d -PE runtime mode, it takes m/d steps to complete one iteration and each step takes $2t/m$ clock cycles. At the first step, PE_0 spends $2t/m$ clock cycles to complete the processes of $Reg_0, Reg_1, \dots, Reg_{(2t/m)-2}$, and $Reg_{(2t/m)-1}$. Meanwhile, $PE_{m/d}$ spends $2t/m$ clock cycles to complete the processes of $Reg_{2t/d}, Reg_{(2t/d)+1}, \dots, Reg_{(2t/d)+(2t/m)-2}$, and $Reg_{(2t/d)+(2t/m)-1}$ at the first step, also.

Fig. 11. Timing diagram of 4-PE RS decoder in full-run (4-PE) mode ($t = 8$, $m = 4$).

Fig. 12. Timing diagram of 4-PE RS decoder in 2-PE mode ($t = 8$, $m = 4$).

$PE_{2(m/d)}, PE_{3(m/d)}, \dots$, and $PE_{(d-1)(m/d)}$ also spend $2t/m$ clock cycles to complete the processes of their own $(d-2)(2t/m)$ data register blocks at the first step.

Similarly, at the second step, $PE_1, PE_{(m/d)+1}, \dots$, $PE_{(d-2)(m/d)+1}$, and $PE_{(d-1)(m/d)+1}$ spend $2t/m$ clock cycles to complete the processes of their own $d \cdot (2t/m)$ data register blocks. The steps continue till the $(m/d - 1)$ th step are performed. Finally, at the $(m/d - 1)$ th step, $PE_{(m/d)-1}, PE_{2(m/d)-1}, \dots, PE_{(d-1)(m/d)-1}$, and PE_{m-1} spend $2t/m$ clock cycles to complete the processes of their own $d \cdot (2t/m)$ data register blocks at the (m/d) -th step. Hence, d register blocks are always processed by d PEs in every clock cycle and $d \cdot (2t/m)$ register blocks are processed during each step. Therefore, it takes $(m/d) \cdot (2t/m) = 2t/d$ clock cycles to finish processing the total $2t$ register blocks in one iteration. After each iteration, the RS decoder delivers one symbol of output data. Thus, the data throughput rate of d -PE mode is $d/2t$ symbols per clock cycle.

B. Operations of Dynamically Reconfigurable RS Decoder Using Four PEs

To demonstrate our design, we use $t = 8$ and $m = 4$ in the following discussion. With four PEs, our design can be dynamically reconfigured to operate at 1-PE (single-PE), 2-PE, and 4-PE (full-run) modes. In the full-run mode, the RS decoder can achieve the best data throughput rate. For lower speed system requirement, we can use 2-PE or 1-PE modes, to deliver the necessary data throughput rate.

1) *Full-Run (4-PE) Mode*: Assume the four embedded PEs are PE_0, PE_1, PE_2 , and PE_3 , and the $2t = 16$ register blocks are $Reg_0, Reg_1, \dots, Reg_{14}$, and Reg_{15} . While in full-run (4-PE) mode, each time all four PEs are active. Every PE still needs to operate $2t/m = 4$ times for processing its own $2t/m = 4$ data register blocks. As shown in Fig. 11, to complete one iteration with throughput of one symbol/cycle, the decoder needs to spend $2t/m = 4$ clock cycles for processing all the $2t = 16$ data register blocks. The data throughput rate in full-run mode is $m/2t = 1/4$ symbol per clock, which is the best rate that a 4-PE decoder can deliver.

2) *Half-Run (2-PE) Mode*: While in half-run (2-PE) mode, each time two PEs are active. For the 2-PE mode, there are only

Fig. 13. Timing diagram of 4-PE RS decoder in single-PE mode ($t = 8$, $m = 4$).

TABLE II
COMPARISON TABLE OF m -PE DESIGN IN DIFFERENT RUN-TIME MODES

	Data throughput rate (symbol/clock)	Latency time (clock cycle)	Hardware usage each clock cycle			
			reg	mult	add	2-to-1 mux
m -PE mode	$m / 2t$	$(2t/m) \cdot (n+2t)$	$6m$	$6m$	$4m$	$24t - 2m$
$(m/2)$ -PE mode	$m / 4t$	$(4t/m) \cdot (n+2t)$	$3m$	$3m$	$2m$	$12t - m$
d -PE mode	$d / 2t$	$(2t/d) \cdot (n+2t)$	$6d$	$6d$	$4d$	$(24t/m) \cdot d - 2d$
2-PE mode	$1 / t$	$t(n+2t)$	12	12	8	$(48t/m) - 4$
1-PE mode	$1 / 2t$	$2t(n+2t)$	6	6	4	$(24t/m) - 2$

Fig. 14. Data throughput rates of RS decoder with different numbers of PEs for $t = 1, 2, \dots, 8$, respectively.

two PEs operating at the same time, and it takes $m/2 = 2$ steps to complete one iteration to throughput one symbol. As well as the full-run mode, every PE still needs to operate four times for processing its own four data register blocks. As shown in Fig. 12, to complete one iteration with throughput of one symbol/cycle, the decoder needs to spend eight clock cycles for processing all data register blocks. The data throughput rate in half-run mode is $1/8$ symbol per clock.

3) *Single-PE Mode*: While in signal-PE mode, each time there is only one PE which is active, and it takes $m/1 = 4$ steps to complete one iteration to throughput one symbol. As well as the full-run mode, every PE still needs to operate $2t/m = 4$ times for processing its own $2t/m = 4$ data register blocks. As shown in Fig. 13, to complete one iteration with throughput of one symbol/cycle, the decoder needs to spend 16 clock cycles for

Fig. 15. Summary of 4-PE dynamically reconfigurable prototyping RS decoder design.

Fig. 16. Data throughput rates and dynamic power consumptions of the 4-PE RS decoder prototyping design in 1-PE, 2-PE, and 4-PE operating modes.

processing all data register blocks. The data throughput rate in 1-PE mode is 1/16 symbol per clock, which is the lowest rate can deliver.

V. PERFORMANCE AND COMPARISON

A. Decision of m -PE MSS Data-Path Architecture

1) *Theoretical Performance Analysis of m -PE design in Different Run-Time Modes:* Based on the data-path controlling method we have discussed, we can analyze the theoretical performance of the m -PE design in different run-time modes in Table II. The data throughput rate of the proposed dynamically reconfigurable RS decoder is directly proportional to the run-time mode d . The latency time is inversely proportional to d . The hardware utilization rate, which is related to the dynamic power consumption, is also proportional to d .

2) *Synthesized Data Throughput Rate Analysis:* Based on synthesis results by using UMC 0.18- μm CMOS standard-cell library, we can obtain the critical path delay of the proposed MSS data-path architecture. It consists of register delay, data-path delay and PE delay. Fig. 14 shows the data throughput rates

that the m -PE RS decoder can achieve by varying numbers of PEs and error correcting capability. According to Fig. 14, we can decide a suitable number of PE for RS specification of error correcting capability.

B. Prototyping RS Decoder Implementation With 4-PE MSS Data-Path Architecture

We implement a dynamically reconfigurable RS decoder with four PEs for $t = 8$. By using UMC 0.18- μm CMOS standard-cell library, this prototyping RS decoder design only requires 24 000 gate counts and can achieve 560 Mb/s in full-run mode. The layout and specification is summarized in Fig. 15.

1) *Dynamic Power Analysis:* The 4-PE RS decoder prototyping design can be dynamically reconfigured to the low-power single-PE mode if system needs a data throughput rate less than 140 Mb/s. For higher symbol rate systems, we can dynamically reconfigure the 4-PE RS decoder to 2-PE mode or 4-PE mode, and pay more power to deliver a higher data throughput rate to achieve the system requirement. Fig. 16 shows the data throughput rates and the dynamic power consumption of the 4-PE RS decoder prototyping architecture design in 1-PE, 2-PE,

Fig. 17. Applications of the prototyping 4-PE RS decoder design.

and 4-PE operating modes. We can see that the dynamic power consumption is a linear function of the data throughput rate and the number of used PEs. Furthermore, by using the gated-clocking scheme, we can enhance the dynamic power saving by 28%–56% in the different PE modes.

2) *Applications of the Prototyping Design:* For Digital Video Broadcasting-Terrestrial (DVB-T), Digital Versatile Disc (DVD) - 10X and Compact Flash (CF) Card - 100X systems, the required data throughput rates are lower than 140 Mb/s (see table of Fig. 17). In these cases, we can configure the prototyping RS decoder to 1-PE operating mode to achieve low power consumption. On the other hand, the data throughput rates for Memory Stick (MS) Card and Extreme Digital (XD) Card systems are 160 and 256 Mb/s, respectively. Hence, the 2-PE mode (280 Mb/s) is enough to support MS/XD card rate. For Fast Flash Disk (FFD) system, it needs 360 Mb/s data throughput rate. We can dynamically reconfigure the RS decoder to 4-PE operating mode (560 Mb/s) to support the high-performance requirement. Fig. 17 summarizes the data throughput rates of the 4-PE RS decoder in 1-PE, 2-PE, and 4-PE operating mode, and requirements of some systems using RS code with $t = 8$.

3) *Performance Comparison:* Table III shows the comparison of performance between the existing RS decoders and the proposed 4-PE MSS-RS decoder. First, we define new technical index (normalized area-efficiency, NAE) as

$$NAE = \frac{\text{ThroughputRate}}{(f_{\max})(\# \text{ of Gates})} (\text{bps}/(\text{MHz} \cdot \text{K gate})) \quad (21)$$

which denotes the maximum data throughput rate per gate normalized to maximum operation frequency. The NAE can clearly demonstrate the area efficiency of proposed VLSI architecture.

TABLE III
COMPARISON WITH EXISTING RS DECODING ARCHITECTURES

	Tech. (μm)	Area (Kgate)	$f_{\max}$ (MHz)	Latency (cycles)	Data Rate (Mbps)	Power (mW)	NAE	NPE
Our proposed	0.18	24	280	$4n + 64$	560	48.47	0.083	22.10
J. H. Lee <i>et al.</i> [15]	0.25	—	130	$4n + 114$	228.1	—	—	—
Ji [16]	—	60	200	$4n + 236$	304.5	—	0.025	—
Koohi <i>et al.</i> [19]	0.13	5,000	450	—	2,534	4,000	0.001	0.63
Chen <i>et al.</i> [20]	0.35	49.5	50	$n + 5t + 2$	400.0	1,200	0.162	2.39
H. Lee [22][23]	0.13	$\frac{4.3}{9.3+8.7}$	770	$n+4t^2$	6,160	—	0.359	—

From the NAE value of Table III, the proposed 4-PE RS decoder can deliver higher data throughput rate than the DSP-type RS decoders while with similar latency time.

Furthermore, we define another performance index [normalized power-efficiency (NPE)] as

$$NPE = \frac{\text{Throughput Rate}}{\text{Power Consumption}} \times \left(\frac{\text{technology}}{0.13 \mu\text{m}} \right)^2. \quad (22)$$

The NPE value describes how much data throughput rate one VLSI design can deliver based on unit power consumption. From Table III, we can see that the NPE of our design is also higher than those of other existing programmable or reconfigurable solutions in [19], [20]. Note that [20] has better NAE value than our design but its NPE value is only 10% of our design. On the other hand, the ASIC-type the pipelined recursive RS architecture [22], [23] is very area-efficient with high data throughput rate (6160 Mb/s). Hence, it has a very high

NAE value. However, this solution is a not reconfigurable design (also, we cannot obtain the NPE value for fair comparison). The high-speed design may be an over-design for low data rate applications, such as voice/MP3 decoding in power-saving domain. On the contrary, our design can be run-time reconfigured to less PE modes to deliver the necessary (no more, no less) throughput rate at high power efficiency and area efficiency.

VI. CONCLUSION

In this paper, we have developed a dynamically reconfigurable RS decoding architecture based on the multi-symbol-sliced processing element. Our proposed m -PE dynamically reconfigurable RS decoder has good flexibility to trade off between the data throughput rate and the dynamic power consumption. For that reason, the proposed architecture is very suitable for the ECC unit of memory system in the portable devices. Moreover, we implement a 4-PE prototyping RS decoder to demonstrate the effectively of our MSS-RS VLSI architecture.

ACKNOWLEDGMENT

The authors would like to thank the support given by National Chip Implementation Center (CIC) in IC design flow.

REFERENCES

- [1] Apple [Online]. Available: <http://www.apple.com/ipod/>.
- [2] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli, "On-chip error correcting techniques for new-generation flash memories," *Proc. IEEE*, vol. 91, no. 4, pp. 602–616, Apr. 2003.
- [3] L. Song, K. K. Parhi, I. Kuroda, and T. Nishitani, "Hardware/software codesign of finite-field datapath for low-energy Reed-Solomon codecs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 2, Apr. 2000.
- [4] H.-C. Chang, C. B. Shung, and C.-Y. Lee, "A Reed-Solomon product-code (RS-PC) decoder chip for DVD applications," *IEEE J. Solid-State Circuits*, vol. 36, no. 2, pp. 229–238, Feb. 2001.
- [5] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [6] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applications*. Piscataway, NJ: IEEE Press, 1994.
- [7] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
- [8] Annex B to ITU-T Recommendation J.83, *Digital Multi-Programme Systems for Television Sound and Data Services for Cable Distribution*, ITU-T J.83, Oct. 1995.
- [9] W. Y. Chen, *DSL: Simulation Techniques and Standards Development for Digital Subscriber Line Systems*. New York: Macmillan, 1998.
- [10] D. J. Rauschmayer, *ADSL/VDSL Principles: A Practical and Precise Study of Asymmetric Digital Subscriber Lines and Very High Speed Digital Subscriber Lines*. New York: Macmillan, 1999.
- [11] *Digital Video Broadcasting (DVB): Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television*, ETSI Standard EN 300 744 V1.4.1, Jan. 2001.
- [12] R. Tessier and W. Burleson, "Reconfigurable computing for digital signal processing: A survey," *J. VLSI Signal Process. Syst.*, vol. 28, no. 1-2, pp. 7–27, May-Jun. 2001.
- [13] H. Lee, M.-L. Yu, and L. Song, "VLSI design of Reed-Solomon decoder architectures," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2000, pp. 705–708.
- [14] T. K. Matsushima, T. Matsushima, and S. Hirasawa, "Parallel architecture for high-speed Reed-Solomon codec," in *Proc. SBT/IEEE Int. Telecomm. Symp.*, 1998, pp. 468–473.
- [15] J. H. Lee, J. Lee, and M. H. Sunwoo, "Design of application-specific instructions and hardware accelerator for Reed-Solomon codes," *EURASIP J. Appl. Signal Process.*, vol. 20, no. 13, pp. 1346–1354, Dec. 2003.
- [16] H. M. Ji, "An optimized processor for fast Reed-Solomon encoding and decoding," in *Proc. IEEE Int. Circuits Syst. Symp.*, 2002, pp. 3097–3100.
- [17] J. S. Lee, M. H. Sunwoo, and S. K. Oh, "Design of DSP instructions and their hardware architecture for a Reed-Solomon codec," in *IEEE Workshop Signal Process. Syst. (SIP)*, 2002, pp. 103–108.
- [18] H.-Y. Hsu and A.-Y. Wu, "VLSI design of a reconfigurable multi-mode Reed-Solomon codec for high-speed communication systems," *Proc. IEEE Asia-Pacific Conf. ASIC*, pp. 359–362, 2002.
- [19] A. Koochi, N. Bagherzadeh, and C. Pan, "A fast parallel Reed-Solomon decoder on a reconfigurable architecture," *IEEE Conf. Hardware/Software Codesign Syst. Synthesis*, pp. 59–64, Oct. 1-3, 2003.
- [20] Z. X. Chen, N.-Y. Shen, O. T.-C. Chen, Y.-F. Hsu, Y.-J. Tsen, and D. Y. Perng, "A programmable Reed-Solomon codec processor," in *Proc. 44th IEEE Midwest Symp. Circuits Syst.*, 2001, pp. 359–362.
- [21] J. Villasenor and B. Hutchings, "The flexibility of configurable computing," *IEEE Signal Process. Mag.*, vol. 15, no. 5, pp. 67–84, Sep. 1998.
- [22] H. Lee, "High-speed VLSI architecture for parallel Reed-Solomon decoder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 2, pp. 288–294, Apr. 2003.
- [23] H. Lee and A. Azam, "Pipelined recursive modified Euclidean algorithm block for low-complexity, high-speed reed-solomon decoder," *Electron. Lett.*, vol. 39, no. 19, pp. 1371–1372, Sep. 2003.
- [24] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits—A Design Perspective*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2003.

Huai-Yi Hsu (S'03) was born in Taiwan, R.O.C., in 1975. He received the B.S. degree from Tam-Kang University, Taiwan, and the M.S. degree from National Central University, Taiwan, both in electrical engineering, in 1999 and 2001, respectively. He is currently pursuing the Ph.D. degree from the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan, in the field of DSP-based high-performance VLSI designs.

His research interests are in the areas of VLSI implementation of DSP algorithms, error control coding, reconfigurable architecture, and digital communication systems.

Jih-Chiang Yeo received the B.S. and M.S. degrees in electrical engineering from the National Taiwan University, Taipei, Taiwan, R.O.C., in 2003 and 2005, respectively.

He is currently an Engineer at Realtek Semiconductor Corporation, Hsinchu, Taiwan, where he is working on communication IC designs.

An-Yeu (Andy) Wu (S'91–M'96) received the B.S. degree from National Taiwan University, Taipei, Taiwan, R.O.C., in 1987, and the M.S. and Ph.D. degrees from the University of Maryland, College Park, in 1992 and 1995, respectively, all in electrical engineering.

From August 1995 to July 1996, he was a Member of the Technical Staff at AT&T Bell Laboratories, Murray Hill, NJ, working on high-speed transmission IC designs. From 1996 to July 2000, he was with the Electrical Engineering Department of National Central University, Taiwan. In August 2000, he joined the faculty of the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, where he is currently a Professor. His research interests include low-power/high-performance VLSI architectures for DSP and communication applications, adaptive/multirate signal processing, reconfigurable broadband access systems and architectures, and SoC platform for software/hardware co-design. He served as an Associate Editor for *EURASIP*

Journal Of Applied Signal Processing from 2001 to 2004, and acted as the leading Guest Editor for a Special Issue on "Signal Processing for Broadband Access Systems: Techniques and Implementations" of the same journal (published in December 2003). He has served on the Technical Program Committees of many major IEEE International Conferences, such as ICIP, SiPS, AP-ASIC, ISCAS, ISPACS, ICME, SOC, and A-SSCC.

Dr. Wu was a four-time recipient of the A-Class Research Award from the National Science Council between 1997–2000. He was also a recipient of the Macronix International Corporation (MXIC) Young Chair Professor Award in 2003. In 2004, he received the Distinguished Young Engineer Award from The Chinese Institute of Engineers (CIE), Taiwan. In 2005, he received two research awards, the Ta-you Award (Young Scholar Award) and the President Fu Si-nien Award, from the National Science Council and National Taiwan University, respectively, for his research works in VLSI system designs. He served as the Associate Editor of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS from 2003 to 2005.