USC–SIPI REPORT #299

Design of Smart Pixel Interfaces for Optical Page-Oriented Memories

by

Wei-Feng Hsu

May 1996

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 404
Los Angeles, CA 90089-2564 U.S.A.

Dedication

To Yu-Yin Hsu, Jin-Yeh Lin and Bin-Feng Hsu.

To Sy-Ying Lee.

Acknowledgments

I am deeply grateful to my thesis advisor, Dr. Alexander A. Sawchuk, for his invaluable support, guidance and encouragement through the whole path of my doctoral study. With his broad knowledge and insight, not only was I able to finish my study and dissertation, I also have found my persistent devotion in the field of optics for my future career.

Thanks are also extended to my dissertation committee members: Dr. B. Keith Jenkins, Dr. Timothy M. Pinkston, and Dr. Woldek Proskurowski, for their constructive comments. A special thanks goes to Dr. Proskurowski for his kindness to joint this committee when my former outside member, Dr. Gorden, left for good.

I would also like to thank: Dr. Mark A. Neifeld, Arizona University, for his helpful discussion and assistance in **optical, parallel error decoding**; Dr. Lloyd Welch, for his enthusiasm and generousness in sharing his resource and great insight in the part of **error correction coding**; Dr. Vijay P. Kumar, for his teachings and guidance in **Reed-Solomon codes**; and Dr. S. Gupta and Mr. Chenhuan Chiang for their assistance and advice in **VLSI circuit designs**. This work would have been much more difficult without them.

Thanks go as well to my officemate and friend, Mr. Jeng-Feng Lin, who shares his ideas and provides assistance in VLSI circuit estimations. I always enjoy working with him. I am also thankful to my colleagues and friends Dr. Charles Kuznia, Dr. Andrew Miller, Chingchu Huang, Jen-Ming Wu, Chih-Hao Chen, and Bogden Hoanca for their encouragement and support.

I also thank Gloria Halfacre and Linda Varilla for their tremendous administrative help. Thanks also to Dr. Allen Weber for his patience in providing computer assistance and in solving computer problems.

This research was supported by the National Center for Integrated Photonic Technology (NCIPT) program funded by ARPA under Contract No. MDA972-94-1-0001 and by the Joint Services Electronics Program through the Air Force Office of Scientific Research (ASOSR) under Contract F49620-94-0022. I would like to acknowledge this financial support which made this work possible.

Finally, I am sincerely grateful to my parents, Yu-Yin Hsu and Jin-Yeh Lin, and my brother, Bin-Feng Hsu, in Taiwan for their love and support. Most of all, my gratitude to my wife, Sy-Ying Lee, is far beyond any ways that I can express. She stood by with incredible patience and encouragement. With her, my life as a graduate student was more wonderful than I could have imagined; without her being in L.A. for the last three months, it would have been the hardest time that I have ever been through.

Abstract

Novel digital information services such as multimedia and video-on-demand require the storage of a large amount data at very low bit-error rate (BER), fast access to this data, and the efficient interface of the storage system to high speed (gigabit/second) networks. Optical page-oriented memory (OPOM) technology is one candidate that simultaneously provides large capacity and high data access rates. In this thesis, several designs for smart pixel (SP) interfaces for optical page-oriented memories are studied. Because of their potentially high capacity and large aggregate data transfer rate, the output of OPOMs must be directly interfaced to high-speed networks. However, the high raw BER of OPOMs severely limits some applications.

We concentrate on error correction coding/decoding and interface design using smart pixel technology. The interface contains an array of SP Reed-Solomon (RS) decoders that reduce the BER to 10^{-12} or better. The RS decoder, implementing the transfer decoding algorithm (TDA), has a pipeline structure and provides a high decoding rate. The TDA is implemented by 1-D and 2-D pipeline structures and serial and parallel finite field multipliers, resulting in six variations of the TDA RS decoder. A modified VLSI circuit simulation model was employed to estimate decoder area, power dissipation, and the maximum clock frequency.

The system analysis in this thesis was performed under two different sets of objectives. The first objective is to define system parameters for the RS coder and decoder which provide the highest aggregate output rate (throughput) of corrected information bits. The second objective is to define system parameters and the RS coder and decoder design which provide the highest code rate (defined as the fraction of total

bits that are useful information) and, in turn, achieve the largest usable capacity. The results of these two analyses are that the codeword length of the chosen RS codes tends to approach two extremes: achieving either high data throughput (shorter length codes); or high capacity (longer length codes). Finally, several methods, including advanced optoelectronic packaging and multi-dimensional array codes, are proposed to improve future system performance.

Contents

List of Figures

List of Tables

List of Symbols

A_c	Chip area of a circuit of N_g logic gates.
A_{pg}	Total allowed area of an SP interface (10 cm^2).
A_{RSde}	The estimated area of a TDA RS decoder.
B_{deW}	Number of codewords processed by an RS decoder in t_a.
B_{in}	Aggregate input data rate in A_{pg} limited by P_{pg}.
B_{info}	Aggregate output data rate in A_{pg} limited by P_{pg}.
$C_{clkwire}$	Total capacitance of the global clock wires.
C_{int}	Wiring capacitance per unit length.
C_{out}	Total capacitance of an output pin.
$C_{totalclk}$	Total capacitance of the clock distribution.
C_{tr}	Capacitance of a minimum-size CMOS transistor.
D_c	Chip dimension.
d_g	Logic gate dimension in actual unit (e.g., micron).
$d_{gintlim}$	The d_g limited to interconnection capacity.
d_{gtrlim}	The d_g limited to transistor packing density.
d_{info}	Information spatial channel density (in bits/second/cm^2).
D_{long}	The longest delay of a codeword (in seconds).
d_{scin}	Input spatial channel density of an array of the TDA RS decoders.
e_w	Wiring efficiency.
F	VLSI minimum feature size.
f_{blk}	Block rate limited to P_{pg}.
f_c	Specified circuit clock rate (in Hz).
$f_{c,max}$	Maximum clock frequency.
f_d	Duty factor.
f_g	Average fan-out of a logic gate.
f_{ld}	Logic depth.
k	Number of information symbols in an RS codeword (in symbols).

k_{driver}	Clock driver ratio.
k_g	Proportionality constant between gate area.
k_{tr}	Width/Length ratio of a minimum-size transistor.
l_{av}	Average interconnection length in actual units.
L_{clk}	Number of clock driven transistors in a logic gate.
m	The order of an RS code (= number of bits per code symbol).
M	Total capacity of an OPOM.
n	Codeword length (in symbols).
N	the natural length of the Reed-Solomon codes (= 2^m - 1).
N_{blk}	Number of bits of a data block.
N_{buf}	Number of logic gates in buffers of a TDA RS decoder.
N_{Dmax}	Number of fabricated decoders that can simultaneously operate.
$N_{D\text{-}A}$	Number of RS decoders fabricated in A_{pg}.
$N_{D\text{-}D}$	Number of RS decoders needed by N_{in} in t_a.
$N_{D\text{-}P}$	Number of RS decoders operates in A_{pg} limited by P_{pg}.
N_g	Total number of logic gates in an electrical circuit.
N_{in}	Number of bits per data page (= number of input channels).
N_{logic}	Number of logic gates for decoding processes.
N_p	Number of I/O pins of an electrical circuit.
n_w	Number of wiring layers.
p	Rent's constant for on-chip interconnection length calculation (in area/power estimation); or, Characteristic of the Reed-Solomon codes (in error correction).
$P(x)$	Primitive polynomial of the Reed-Solomon codes.
P_b	Raw (input) bit-error rate (BER).
P_{buf}	Power dissipation of buffers in an RS decoder.
P_c	Total power dissipation (in Watts).
P_{clk}	Power dissipation of the clock distribution.
P_e	Output bit-error rate.
P_{io}	Power dissipation of electrical I/O buffers.
P_{idec}	Product of the information density and the effective capacity.
P_{logic}	Power dissipation of the decoding logic.
P_{pg}	Total power dissipation in area A_{pg} (in Watts).
P_{RSde}	Power dissipation per RS decoder at f_c.

P_s	Raw symbol bit-error rate.
p_w	Wiring pitch.
r	Code rate.
$\overline{R}$	Average wire length in units of gate pitch (estimated by the Rent's rule).
R_{int}	Wiring resistance per unit length.
r_q	Required code rate.
S	Scaling factor.
t	Largest number of symbols that can be corrected per codeword.
t_a	Optical memory access time per data page.
T_g	Average gate delay.
v_c	Light speed.
V_{DD}	Supply voltage.
X_1	Number of transistors of a type-1 FFM (using 1-D systolic array).
x_{21sw}	Number of transistors of a CMOS 2-to-1 switch (using C-switches).
x_{22sw}	Number of transistors of a CMOS 2-to-2 switch (using C-switches).
x_{add}	Number of transistors of a CMOS single-bit full adder.
x_{and}	Number of transistors of a CMOS 2-input AND gate.
X_{c2}	Number of transistors of a type-c2 FFM (compound-gate, fixed P, no C).
X_{c3}	Number of transistors of a type-c3 FFM (compound-gate, fixed P/B, no C).
x_d	Number of transistors of a CMOS single-bit delay cell.
x_{dff}	Number of transistors of a CMOS D-Flip-Flop.
x_{inv}	Number of transistors of a CMOS inverter.
x_{nor}	Number of transistors of a CMOS 2-input NOR gate.
x_{rsl}	Number of transistors of a CMOS reset-set latch (= $2 \cdot X_{nor}$).
X_{RSde}	Number of transistors required to realize an (n, k) RS decoder.
X_{s2}	Number of transistors of a type-s2 FFM (2-D systolic array, fixed P).
X_{s3}	Number of transistors of a type-s3 FFM (2-D systolic array, fixed P/B).
X_{s3}	Number of transistors of a CMOS 2-input exclusive-OR gate.

List of Acronyms

BEC	Bit error constraint.
BER	Bit-error rate.
BPSP-C	Bit-parallel/symbol-parallel RS decoder with compound-circuit FFM's.
BPSP-S	Bit-parallel/symbol-parallel RS decoder with 2-D systolic FFM's.
BPSS-C	Bit-parallel/symbol-serial RS decoder with compound-circuit FFM's.
BPSS-S	Bit-parallel/symbol-serial RS decoder with 2-D systolic FFM's.
BSSP	Bit-serial/symbol-parallel RS decoder with linear systolic FFM's.
BSSS	Bit-serial/symbol-serial RS decoder with linear systolic FFM's.
EA	Euclid's algorithm.
FFM	Finite field multiplier, or finite field multiplication.
CCD	Charge couple device.
CRC	Code rate constraint.
GF	Galois field (finite field).
ITES	Inverse transform of the error sequence.
MAXC	Power/area constraint.
MINC	Buffer length constraint.
OE	Optoelectronic.
OEIC	Optoelectronic integrated circuit.
OPOM	Optical page-oriented memory.
RS	Reed-Solomon.
SNR	Signal-to-noise ratio.
SP	Smart pixel.
SUSPENS	Stanford University System Performance Simulator.
TEP	Transform of the error pattern.
TDA	Transform decoding algorithm.
XOR	Exclusive-OR.

Chapter 1

Introduction

1.1 Motivation and Objectives

Current digital processors have clock rates in excess of 100 MHz, and conventional data storage technology has difficulty in maintaining these rates. In addition to computer applications, enhanced digital information services, such as high-resolution multimedia images, video-on-demand, and high definition television, requires the storage of a large amount of data at very low bit-error rates (BER), fast access to this data, and the efficient interface of the storage system to high speed, gigabit per second networks [1]. The information bandwidth of communication networks using optoelectronic integrated devices and optical-fiber transmission has increased rapidly to the current 2.4 Gbits/second/channel and to 10 Gbits/second/channel in near future [2]. The performance of information systems is inevitably limited by the access rate of the data storage systems. Optical page-oriented memory (OPOM) [3]-[5] technology is one candidate that simultaneously provides large capacity (10^{12} bits/cm^3 theoretically) and high data access rate (10^9 bits/second or more). Unfortunately, OPOMs have a high raw BER (in the range of 10^{-4} to 10^{-7}) which presents a limitation. The use of error detection/correction is one way to reduce the BER to a desirable rate while improving the overall memory capacity [1], [6].

Interfaces between OPOMs and high speed networks must not only provide high data throughput rates to prevent I/O bottlenecks, but must also reduce the BER. Because high performance error-correction encoding/decoding requires complicated

operations and hardware, the overall data throughput may need to be reduced if extensive error correction is needed. Then, there is a tradeoff between the data throughput and error-correction capability. This study focuses on the error correction decoding in the output interface of OPOMs as shown in Fig. 1.1. The decoding logic in the interface works on the binary bit streams received from binary thresholding performed at the photodetector. Decoded binary data, whose BER is ideally 10^{-12} or better, is output to an interconnection network. While error encoding/decoding may be needed to process the data for its transmission in the network, this subject has been well studied in the field of digital data transmission and is outside the scope of this thesis.

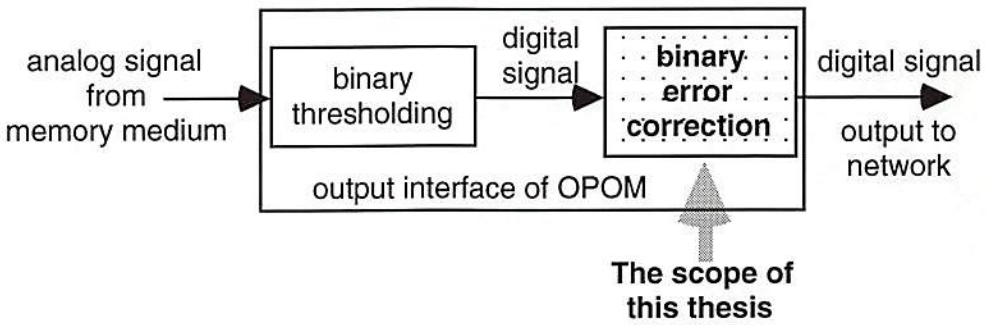


Figure 1.1 Scope of this thesis.

Error correcting codes such as Reed-Solomon (RS) codes have been successfully used to correct random and burst errors in deep space communications, and in data retrieval from mass storage devices, and optical disks [7]. Because of the complicated decoding processes of high performance codes, however, the decoding rate is limited to several tens of megabits per second [8]. In order to provide gigabit-per-second aggregate decoding rate, multiple error-correcting decoders at the output of the OPOM systems are needed. Optoelectronic (OE) smart pixel (SP) devices are one means of achieving high data rates by parallel processing and a large number of I/O. The SP technology uses VLSI processes for monolithic integration of photodetectors for data input, optical

modulators or sources for data output, and electrical circuitry for computing and logical operations [9]. Neifeld proposed an RS decoder with parallel input/output (I/O) with an information rate of 300 megabits/second, and constructed a module with an array of parallel decoders [10], [11].

The goal of this thesis is to design output interfaces for a smart pixel OPOM system of high aggregate data rates (10^{11} bits/second or 0.1 terabits/second) and large usable capacity while reducing the BER to an acceptable rate (10^{-12} or better). Different design variations of RS decoders used in the SP interfaces were studied. The RS code and decoder design that provides the highest aggregate data rate and the largest usable memory capacity were determined under the limits of given physical conditions such as the minimum feature size of VLSI processes, chip area, and power density. The scenarios developed here may also be applied to evaluate the performance of other SP designs.

1.2 Methodology

In this thesis, there are six RS decoder variations which are constructed from four decoder designs, with bitwise and/or symbolwise I/O formats, devised with two types of finite field multipliers. All these designs employed an RS decoding scheme, the transform decoding algorithm, which has a regular structure and is suitable for VLSI implementation [12]. Since there are six implementations and each with many choices of possible RS codes, an objective is to determine which implementation and RS code provides the highest data throughput under the limits of a fixed VLSI minimum feature size, chip area, and power density. For an OPOM, on the other hand, the minimum access time and the size of a data page are determined by the material characteristics. The corresponding addressing scheme is known in advance, so the data rate going into

the output interface is determined. It depends on the designers to choose the RS code and the decoder implementation which are able to best match the established data rate while achieving a large usable capacity.

Two scenarios were investigated to determine the RS code and decoder design for the highest data throughput, and the RS code and decoder design for the largest usable memory capacity. The procedures of the first scenario are described below:

i) Calculate the number of logical gates and VLSI transistors of a decoder for all of the RS codes that are capable of a desirable BER.

ii) Estimate the chip area, power dissipation, and the maximum clock frequency.

iii) Calculate the number of decoders that can be fabricated in a fixed interface area.

iv) Let all fabricated decoders perform the decoding processes at a rate limited by a given power density.

v) Calculate the data input rate, the output data throughput, and the product of data throughput and usable capacity.

Then, the RS code as well as the decoder design of best performance are determined in terms of the parameters obtained in step (v).

However, the RS code and decoder found in the first scenario might not effectively utilize the memory capacity because the code rate is not optimized. The second scenario determines the RS code and decoder design specified by system parameters (chip area, power density, desirable clock frequency, and minimum feature size) to achieve the largest usable capacity, and is described in the following:

i) Same as step (i) in the first scenario.

ii) Same as step (ii) in the first scenario.

iii) Calculate the number of decoders that can be fabricated in a fixed interface area.

iv) Calculate the number of decoders that can operate simultaneously at a desirable clock rate, depending on system specifications, subject to a fixed power density.

v) The maximum number of decoders that simultaneously process data, limited to the specified area and power density, is given by the smaller number obtained in steps (iii) and (iv).

vi) Calculate the minimum number of decoders required to decode a data page in a specified page access time. This number depends on the buffer length and the decoding delay of a specific decoder design.

vii) If the result of (v) is larger than or equal to the result of (vi), then the RS code with the largest code rate and the corresponding decoder design is the best selection; if not, no RS code and decoder design can satisfy the system specifications.

Note that part of the fabricated RS decoders operates at a specific clock rate which gives the interface design more flexibility.

1.3 Thesis Organization

This thesis contains five major sections. Chapter 2 describes the fundamentals of the optical page-oriented memories, summarizes the current smart pixel technology, and introduces a VLSI circuit simulation model. Chapter 3 studies the error control codes, particularly the Reed-Solomon codes and a decoding scheme, followed by a summary of their applications. The error correction techniques applied to optical computing are also described. In Chapter 4, the parameters of OPOMs and the SP interfaces studied in the thesis are specified, and tables of symbols used in sub-sections are individually listed. Chapter 5 describes the designs and implementations of the RS decoder in detail.

Chapter 6 shows the performance of the decoder variations and the feasibility analysis of the SP interfaces using the designed decoders. Chapter 7 concludes and summarizes key results of this dissertation. Chapter 8 describes future research extensions for related topics.

1.4 Contributions

This work describes design issues and feasibility of the output interface of OPOM systems using the smart pixel devices. The main contributions are summarized below:

1) The developed schemes propose a solution to reduce the high BER of the OPOMs while increasing the memory capacity without sacrificing the data rate.
2) To the author's knowledge, the error-correcting SP cell contains the most complicated electrical logic among the current SP applications. It was also shown that the interface design requires I/O capacity which exceeds the physical limits of electrical I/O pins.
3) Two dimensional, parallel RS decoders implementing the TDA scheme were developed.
4) A large variety of the RS codes and decoder designs were compared. It is observed that the bit-parallel implementations perform better than the symbol-parallel ones in the studies.
5) Parameters were defined to evaluate the performance of the SP devices for OPOMs.
6) The developed schemes may be applied to other designs of SP devices.

Chapter 2

Preliminaries

2.1 Optical Page-Oriented Memories

Optical page-oriented memory (OPOM) [1], [3]-[5] has potentially large storage capacity, short access time, and high data access rate to satisfy the requirements of advanced audio, video, and multimedia applications. Recent developments in materials, spatial light modulators, and solid-state lasers have revitalized the OPOM, which was proposed in 1963 [13]. An OPOM consists of an input interface, a memory medium, and an output interface, as shown in Fig. 2.1 [9]. The input interface contains a serial-to-parallel converter, parity-check encoders, and a page composer. The serial-to-parallel converter is composed of optical receivers and buffers. Examples of it are a CCD array or an array of photodetectors. The parity-check encoder adds parity-check symbols to a block of data symbols by using electrical encoding circuits. The parity-check symbols are linear combinations of the data symbols. The page composer in the input interface at the left formats the data to be input in the memory, and then a light beam reads out the data as an image, called a data page. The input optical data page is combined with a reference beam in the memory medium, and the data page is recorded as the resulting modulation of a physical property of the recording medium. When reading a data page, the recorded pattern is reconstructed by a counterpart of the reference beam, called the reconstruction beam, and is detected by the output interface at the right. The details of the output interface will be discussed in Chapter 4. In this section, we present the characteristics of these modules. Note that Fig. 2.1 shows recording material and I/O

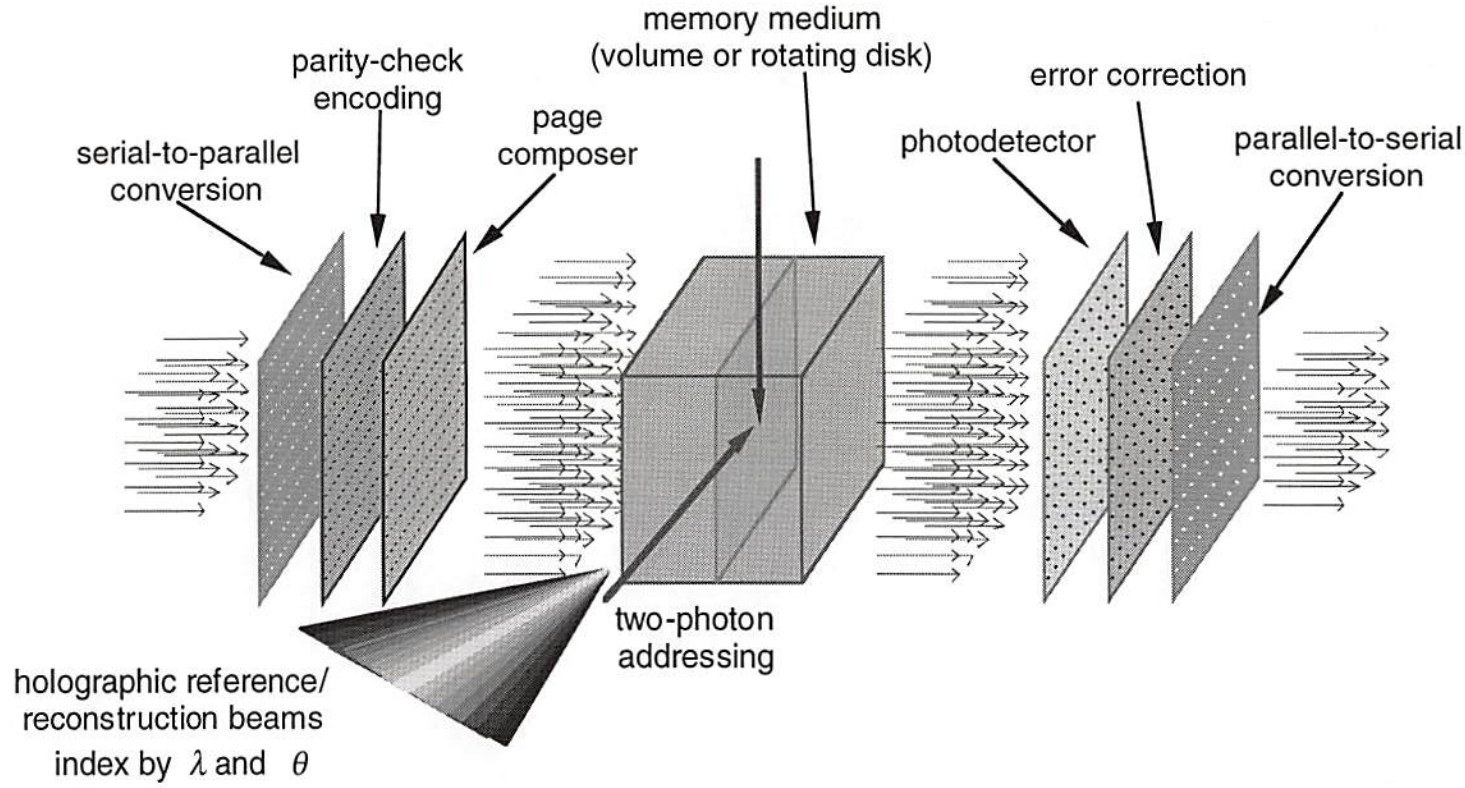


Figure 2.1 Optical page-oriented memory and input/output interfaces.

interfaces only; however, in a complete memory system, an addressing system will be included as well.

2.1.1 Input Interface

The input interface consists of a serial-to-parallel converter, an error-correction parity encoder, and a page composer. The error-correcting encoder is needed when the received data has not previously encoded by error-correcting codes. The significant improvement in OPOM performance using error-correcting codes will be described in the discussions of the noise sources in this chapter and in the error-correcting code discussion in Chapter 3. In many applications (such as archival data storage), the data input operation takes place off-line at a relatively slow speed compared to readout. The page composer formats the data to be input in the memory. The page composer consists of many independent spatial elements whose transmittance is modulated by a large number of serial electrical data channels. Data may then be read out in parallel by a light beam. Liquid-crystal spatial light modulators (SLMs) and film masks are examples of SLMs usually used for dynamic and static data input, respectively. Resolution is the major issue for the page composer. The more data bits per data page, the larger the capacity and the higher the data rate which can be achieved. The number of data bits is also limited, however, by the imaging system, noise effects of memory medium, and the capacity of the output devices. The size of a data page may be on the order of a million bits.

2.1.2 Memory Medium and the Recording/Retrieving Methods

There are several materials and technologies that have been developed for the OPOM. These include: photorefractive crystals; two-photon materials; spectral hole-

burning materials, and other materials [5]. The characteristics and their recording/ retrieving methods are described below.

Photorefractive crystals contain electron traps [1], [14]-[17]. Electrons are excited to the conduction band by exposing the crystal to light. The resulting free electrons in the conduction band drift under the applied or internal electrical field or diffuse thermally until the field completely cancels the effect of diffusion or drift. After the free electrons drift some distance under the applied field, they are re-trapped, leaving an internal electrical field due to the re-distribution electrons. The resulting space-charge distribution modifies the index of refraction of the crystal through the electrooptic effect. The crystal consequently records the input light pattern by the modulated index of refraction.

The input light pattern is in general an interference pattern (or a hologram) which results from the interference of an object beam with a reference beam. Since the object beam carries an image containing a 2-D bit array, the capacity and data access rate are potentially high. To retrieve the recorded pattern, a reconstruction beam reads out the data image from the interference pattern. The large capacity of photorefractive materials is achieved by multiplexing a number of holograms in the same volume, and each contains a large 2-D bit array. Multiplexing schemes such as angular [16]-[19], wavelength [19]-[21], phase-code [22], [23], spatial [24], fractal [25], and peristrophic [26] multiplexing have been developing to increase the number of multiplexed holograms with lower cross-talk. In order to explore further the available capacity of the holographic material, more than one multiplexing scheme may be combined, such as spatial-angular [27], fractal-spatial [28], and spatial-angular-peristrophic [29] in a volume.

The two-photon effect is based on the absorption of two optical beams for write-in or read-out of data [30], [31]. Photochromic molecules of the two-photon material

which simultaneously absorb two beams at λ_1 and λ_2 (typically $\lambda_1 < \lambda_2$) are excited from the ground state to an excited state and then stay at a stable state thus writing information to the material. The read process is achieved also by absorbing two beams both at λ_2 which excite the molecules to an unstable state. The unstable molecules immediately radiate light at λ_3 and move back to the stable state. The two-photon OPOM is constructed as a multilayer 2-D memory stack, and each layer is illuminated by two beams separated at 90°. By moving one of the readout beams to a given spatial position, the associated layer is specified. Since the energy required to change the states is small (< 10 fJ/μm^2), the data access time is also relatively small (around 30 ps).

In spectral hole-burning materials [32], [33], e.g., chlorin, the absorption coefficient and/or refractive index are spatially modulated by absorption of the incident light at a particular absorption band, say λ_1, of the molecule. After absorbing the photon energy, the excited molecules, called tautomers, no longer react to light of λ_1 and form a spectral hole. With a recorded pixel size of a limited number of molecules, this technique can possibly achieve high data density (10^{14} bits/cm^2 in theory). The spectral hole-burning material contains inhomogeneous molecules which have different absorption bands, and each reacts to a different wavelength. Therefore, large capacity is achieved by recording a large number of data pages multiplexed by different wavelengths. Data are recorded by associating a spectral hole with a binary digit '1', and its absence with a '0'. The direct recording enables random access to the data, but it also induces strong background noise when data is being read out. To reduce the background noise and increase the readout SNR, holographic techniques using frequency and electrical field multiplexing are used. At the low temperature of liquid helium, spectral hole-burning materials store data for nearly an indefinite time. The recorded data is read out by a light beam of the same exciting wavelength λ_1, but with much lower intensity to avoid additional hole burning.

2.1.3 Output Interface

The parallel output of the memory medium impinges on a photodetector array or CCD camera at the left side, and is converted to electrical form. In current OPOM systems, a CCD device consisting of one million pixels has been used [34]. However, the data is transferred at TV frame rates which are far below the rate of a high speed network. To achieve high aggregate data rate, an array of smaller CCD devices may be used to provide parallel outputs. The error correction systems next to the CCD or photodetector array decodes the data and corrects errors. At the right side of the output, a transmitter array transfers the corrected data to an output channel or network. In the following chapters, the error correcting interface will be discussed in detail.

2.1.4 Source of Noise, and Its Effect on OPOM's

The noise sources of the OPOM systems are divided into system and material noises [35]. The system noises include: input/output (I/O) device imperfection, detector noise (thermal and shot noise), lens aberrations, scattering and multiple reflections from lenses and other optical components, misalignment from transmitter to photodetector pixel, and laser non-uniformity and fluctuations. By carefully designing the components and precisely aligning the system, the bit-error rate (BER) is estimated in the range of 10^{-4} to 10^{-5}, which mainly depends on the number of pixels in the I/O devices [34], [35]. Note that even without any data recorded by the memory medium, the raw BER of an OPOM system for using large data pages could not satisfy the low BER requirement of 10^{-15} or better required for digital applications. The BER degrades rapidly to the range of 10^{-6} to 10^{-3} when the medium starts recording holograms

multiplexed by the schemes described above [1], [34], [35]. Such a high BER necessitates the use of error correcting codes in the OPOM for most applications. Examples of the material noise sources are: cross-talk between recorded holograms, inter-pixel cross-talk in a hologram, internal reflections in the medium, non-uniform diffraction efficiency, distortions due to surface imperfection, blurring due to limited spatial resolution, damage to the medium, and scattering by defects and particles. In Chapter 3, the improved BER by using error correction coding/decoding is shown.

2.2 Optoelectronic Smart Pixel Technology

Optics is better than electronics in interconnections of distance over a few hundreds microns, and electronics is better at signal amplification, processing and logic at each pixel [36]. The recently developed optoelectronic (OE) smart pixel (SP) devices have the potential to combine the merits of optics and electronics in a single device. An SP device consists of an array of SP pixels, and each has electrical circuitry to perform computations and logic; and optical I/O to transfer data to/from local pixels or other devices. The advantage of using the SP technology is three-fold: parallel operations provide a large data processing rate, optical I/O and free-space interconnections reduce the communication hardware and increase the number of available I/O ports, and electrical circuitry can perform relatively complicated computations and logic. Various SP techniques have been investigated, and each has characteristic strengths and weakness, which we attempt to summarize in Table 2.1. These SP devices can be divided into passive spatial light modulators (SLMs), active optical sources, and hybrid OEIC devices.

The first technology of the passive SP devices is SEEDs (self-electro-optic-effect devices) using multiple quantum well (MQW) technology [37]. The basic element of

SEEDs is an electrically biased optically controlled PIN diode combining photodetector, switch and modulator. In order to increase the switching speed, later SEEDs have been fabricated with field-effect transistors (FETs) in the same substrate, a combination called FET-SEEDs [38]. In the latest SEEDs, called CMOS-SEEDs, the GaAs-AlGaAs MQW modulators are flip-chip bonded to the active silicon CMOS circuits [39]. The SEED MQW modulators used for optical I/O are connected to CMOS circuits through metal wires. Liquid crystal (LC) SLMs, the second passive SP device, have been integrated on the silicon substrate, and addresses by metal and polysilicon wires [40]. The third passive SP device is the integration of Si VLSI circuits on a lead lanthanum zirconate titanate (PLZT) substrate [41].

The active SP devices are vertically integrated devices containing photo detector, bipolar switch, and optical source such as VCSEL (vertical cavity surface emitting lasers) or LED. The first technology of this type is LED-mode VSTEP (vertical to surface transmission electro-photonic) devices by NEC [42] which are a light-triggered pnpn thyristor combined with an LED. The LED-mode VSTEP can be used as a dynamic memory device for its low switching power. However, due to low electrical-to-optical conversion efficiency, the output power is low. Laser-mode VSTEP device has a structure of pnpn thyristor in a vertical cavity. Smart pixels based on this technology usually combine the vertical-cavity pnpn material with heterobipolar transistor (HBT) or heterojunction phototransistor (HPT). The third technology of vertically integrated devices includes VCSELs at the University of New Mexico [43].

Optoelectronic integrated circuits (OEICs) are the hybrid SPs consisting of separate detectors, transistors, MQW modulators, and/or optical sources in each pixel. There are various fabrication technologies and materials under considered in this type of SPs.

Table 2.1 Characteristics of optoelectronic smart pixel devices.

	SEEDs		SLMs with silicon		Vertically Integrated Devices			OEICs	
	FET-SEED	CMOS-SEED	liquid crystal SLMs	Si/PLZT	VSTEP/DOES (LED)	VSTEP/DOES (laser)	CELL/VCSEL	OE Cellular Array	Optically Powered Smart Pixel
developer	AT&T	AT&T	Colorado/BNS	UCSD	NEC/AT&T	NEC/AT&T	UNM, Sandia	Honeywell	USC/Princeton
contrast ratio	3.5:1	4:1	10:1	> 10:1 (15V)	(> 200 : 1)	(> 200 : 1)	> 200 : 1	> 200:1	> 100:1
write λ (nm) read λ (nm)	853 (10V), 860 (6.5V)	850	830/633	514	(LED λ)	955	680-870 850 (958-983)	infrared or 830	1300 1300
applied voltage (V)	6 - 10	5 & 10	6	> 15	3.5	4.1	8 (3 - 5)	5	4
switching energy per μm^2	-	-	< 7.7 pJ	2 pJ	-	100 pJ	3.8 pJ [1]	-	-
switching energy per pixel	0.9 pJ	60 fJ	< 3.6 nJ	200 pJ	10 pJ	100 nJ, (10 fJ?)	7.5 nJ (?)	-	3.8 pJ
switching time (modulation rate)	< 1 GHz	375 MHz	> 50 μs	1 MHz	> 2.5 ns	10 - 25 ns, 400 MHz	1 - 5 ns 0.2 - 1 GHz	1 ns	25 ns 40 MHz
switching power per pixel	0.9 mw	3.5 mw	72 μw (20 kHz)	0.2 mw	< 4 mw	< 1 mw	N/A	100 μw (36 mW)	5.3 mw
static (holding) power/pixel	-	0.1 pw	-	-	20 μw	2 - 5 mw	2 - 5 mw	2 mw	-
insertion loss (?)	(0.3)	-	0.86	~ 0.25	-	-	-	-	-
readout mode	readout beam modulation	readout beam modulation	readout beam modulation	readout beam modulation	LED	laser	laser	LED	laser - not yet integrated
pixel size (μm × μm)	40 × 40, 10×10 (win)	20 × 45 (modulator)	21.6 × 21.6	10×10 (win)	30 × 30	10 × 10	50 dia. (1 - 5.5 dia)	250 × 280	550 × 550
integrated array size or packing density	6 × 6	10 × 10	256 × 256	32 × 32	32 × 32	-	$2\times10^4/\mu m^2$	8 × 8	$300/cm^2$
electrical/optical control	elect./opt.	elect.	elect.	elect.	elect	elect	elect/optical (w/ HPT)	elect	optical
wavelength cascadability	Yes	Yes	-	Yes	Yes	Yes	Yes	Yes	Yes

Two examples are the optoelectronic cellular array by Honeywell [44] and optically powered SP at USC and Princeton [45]. OEICs usually have higher contrast ratio and signal bandwidth, but higher switching energy.

2.3 A VLSI Circuit Simulation Model and An Ideal Scaling Rule

2.3.1 The Modified SUSPENS Circuit Simulation Model

This section introduces a CMOS simulation model that will be used to estimate circuit parameters for smart pixels at various scale sizes.

This circuit model is originated from the SUSPENS [46] model and modified by Liu and Svensson's model [47]. The SUSPENS model is a system-level circuit simulator for central processing units (CPUs). The model estimates the clock frequency, power dissipation, and chip/module sizes of general-purpose processors by emphasis on the interactions among devices, circuits, logic, packaging, and architecture. In order to apply the SUSPENS model to the TDA decoder, we modified it using Liu and Svensson's model in which the on-chip SRAM and the logic gates are estimated using different sets of parameters and the power consumption in clock distribution is also taken into account. The TDA decoder does not have on-chip SRAM; instead, there are a large number of shift registers which have very different properties from the decoding logic gates. Therefore, two sets of parameters and equations are used by the modified model to predict the TDA decoders.

In the modified circuit model, first, an upper limit to average wire length $\overline{R}$ (in units of gate pitch) is defined by

$$\overline{R} = \begin{cases} \frac{2}{9}\left(7\frac{N_g^{p-0.5}-1}{4^{p-0.5}-1} - \frac{1-N_g^{p-1.5}}{1-4^{p-1.5}}\right)\cdot\frac{1-4^{p-1}}{1-N_g^{p-1}}, & p \neq 0.5 \\ \frac{2}{9}\left(7\log_4 N_g - \frac{1-N_g^{p-1.5}}{1-4^{p-1.5}}\right)\cdot\frac{1-4^{p-1}}{1-N_g^{p-1}}, & p = 0.5. \end{cases} \quad (2.1)$$

It is obtained by applying Rent's rule to calculate the number of interconnections in a circuit block. Here, N_g is the number of logic gates in the block and p is the empirical Rent's constant for on-chip interconnection length calculation. In Eq. (2.1), Rent's constant p has been modified from the p calculated from Rent's rule, for example, $p = 0.4$ rather than $p = 0.6$ - 0.7 which is obtained directly from circuit layouts. On the other hand, the theoretical p is used when a factor of 0.54 is applied to the computed $\overline{R}$ [48], i.e.,

$$\overline{R} \leftarrow 0.54\overline{R}(\text{real } p) \quad (2.2)$$

Then, the average interconnection length in actual units is

$$l_{av} = \overline{R} \cdot d_g, \quad (2.3)$$

where d_g is the logic gate dimension in, for example, microns.

The logic gate dimension is limited by transistors when all gates can be placed right next to each other. In TDA decoders, the shift registers used for pipelining and buffers are transistor packing density limited. The logic gate dimension is computed as

$$d_{gtrlim} = \sqrt{k_g} \cdot F, \quad (2.4)$$

where k_g is a proportionality constant between gate area and F, the minimum feature size of CMOS technology used. In our simulation, $k_g = 67$ for a D-flip-flop consisting of 16 transistors.

Another case of the logic gate dimension happens in logic-intensive chips where area is normally limited by wiring capacity. In TDA decoders, the finite field multipliers, mod-2 adders, and other logic are characterized into the interconnection-capacity limit, and the logic gate dimension is given as

$$d_{gintlim} = \frac{f_g \overline{R} p_w}{e_w n_w}, \tag{2.5}$$

where f_g is the fan-out of a typical gate, p_w is wiring pitch, e_w is wiring efficiency, and n_w is the number of wiring levels. In our simulations, p_w is chosen as 3 times F although a larger factor, for examples, 4 or 5, is usually used in sub-micron VLSI technology. The wiring efficiency e_w is typically 0.4, and the wiring level n_w is assumed to be 3.

The logic gate dimension is the maximum of d_{gtrlim} and $d_{gintlim}$, i.e.,

$$d_g = \max\left(d_{gtrlim}, d_{gintlim}\right). \tag{2.6}$$

The dimension and area of the VLSI chip are then

$$D_c = d_g \sqrt{N_g}, \text{ and} \tag{2.7}$$

$$A_c = D_c^2 = d_g^2 N_g, \tag{2.8}$$

respectively.

The maximum clock frequency that can be achieved is estimated by

$$f_{c,max} = \left(f_{ld} T_g + R_{int} C_{int} \frac{D_c^2}{2} + \frac{D_c}{v_c} \right)^{-1}, \tag{2.9}$$

where f_{ld} is the logic depth, T_g is the average gate delay, v_c is the light speed, and R_{int} and C_{int} are the wiring resistance and capacitance per unit length, respectively. Due to the pipeline design of the TDA decoders and FFMs, the logic depth is much shorter (e.g., 4 to 8) than most general-purpose computing processors (from 8 to 30), which implies high operating speed for the TDA decoders.

The total power dissipation is estimated as

$$P_c = \frac{1}{2} f_c f_d N_g f_g \left(l_{av} C_{int} + 3 k_{tr} C_{tr} \right) V_{DD}^2 + \frac{1}{3} \frac{1}{2} N_p f_c f_d C_{out} V_{DD}^2 + f_c C_{totalclk} V_{DD}^2, \tag{2.10}$$

where f_c is the clock frequency used, f_d is the duty factor, C_{tr} is the capacitance of the minimum-size transistor, k_{tr} is the width/length (W/L) ratio of VLSI transistor, N_p is the

number of inputs/outputs, C_{out} is the total capacitance at an output pin (= 50 pF), $C_{totalclk}$ is the total capacitance of clock distribution, and V_{DD} is the supply voltage. The first and second terms estimates the power consumed in performing logic functions and in the input/output buffers, respectively. The third term accounts for the power consumption in clock distribution. The total capacitance of clock distribution $C_{totalclk}$ is given as

$$C_{totalclk} = \left(1 + k_{driver}\right)\left(L_{clk} N_g C_{tr} + C_{clkwire}\right), \tag{2.11}$$

where k_{driver} the clock driver ratio (= 0.3), L_{clk} is the number of clock driven transistors in a logic gate, and $C_{clkwire}$ is the global clock wire capacitance and is approximated by

$$C_{clkwire} = 24 C_{int} D_c. \tag{2.12}$$

The original Eq. (2.10) in [47] contains a term of SRAM capacitance which is not used in TDA decoders, and then does not appear here. Finally, in a TDA decoder, the total power consumption P_c can be expressed as a combination of powers consumed in the logic part P_{logic}, buffers P_{buf}, input/output buffers P_{io}, and clock distribution P_{clk}, i.e.,

$$P_c = P_{logic} + P_{buf} + P_{io} + P_{clk}, \tag{2.13}$$

where

$$P_{logic} = \frac{1}{2} f_c f_d N_{logic} \left(\frac{d_{gintlim}^2 n_w e_w C_{int}}{p_w} + 3 k_{tr} f_g C_{tr} \right) V_{DD}^2, \tag{2.14.a}$$

$$P_{buf} = \frac{1}{2} f_c f_d f_g N_{buf} \left(d_{gtrlim} \overline{R} C_{int} + 3 k_{tr} C_{tr} \right) V_{DD}^2, \tag{2.14.b}$$

$$P_{io} = \frac{1}{3}\frac{1}{2} N_p f_c f_d C_{out} V_{DD}^2, \text{ and} \tag{2.14.c}$$

$$P_{clk} = f_c C_{totalclk} V_{DD}^2. \tag{2.14.d}$$

Here, N_{logic} and N_{buf} are the number of logic gates of the logic and buffer circuits, and $N_g = N_{logic} + N_{buf}$.

2.3.2 Ideal Scaling Rule

The properties of a circuit in different fabrication scales can roughly be estimated by using an ideal scaling theory. Let the scaling factor of physical dimension be S. For example, $S = 2$ if the minimum feature size is shrunk from 0.8 μm to 0.4 μm. When the scaling theory is applied, all horizontal and vertical dimensions of transistors are scaled down by $1/S$, substrate doping is increased by S, and all voltages are decreased by $1/S$. These result in an approximately constant internal electrical field. In consequence, the scaled-down circuit has higher operating speed, lower power dissipation, and higher packing density. These scaling properties of CMOS transistors are listed in Table 2.2. Note that the above scaling theory is used as a rough estimation to the scaled CMOS circuits . There are a number of second-order effects and fabrication specifications that require modifications in the scaling rules, and they are omitted at this moment.

Table 2.2 Scaling rule of MOS transistors.

Parameter	Scaling Factor
dimensions (W, L, t_{gox}, …)	$1/S$
substrate doping	S
voltages (V_{DD}, …)	$1/S$
gate capacitance C_g ($\propto C_{tr}$)	$1/S$
intrinsic gate delay	$1/S$
power-dissipation per gate	$1/S^2$
power-dissipation per gate (fixed f_c)	$1/S^3$
area per device	$1/S^2$
resistance per unit length (R_{int})	S^2
capacitance per unit length (C_{int})	1

Chapter 3

Error Correction Using Reed-Solomon Codes

Error correction/detection techniques have been widely used in applications such as digital communications, mass data storage, optical compact disk data storage and computers, to improve the reliability of information processing [7]. Error correction techniques add or encode a small portion of redundant information into the digital messages before they are transmitted or stored. At the receiver, the received data is decoded, and certain types of errors occurring due to system noise and material defects can be corrected by recombining the error-correcting information.

Many error correction codes have been invented and studied since Claude Shannon initiated the error correcting idea in 1948. A major topic in the error correction/detection is to find 'good' error correction codes in which the codewords are maximally separated in a code space, or to find so called maximum distance separable (MDS) codes [49]. In 1959, a new class of algebraic MDS codes was proposed and named Reed-Solomon (RS) codes after their inventors. The RS codes belong to the family of linear block codes and, more precisely, they are a special class of the BCH (Bose-Chaudhuri-Hocquenghem) codes. The RS codes are used in deep space telecommuncations and compact disk data storage because of their ability to correct both burst and random errors and their great flexibility in code length and properties. The message block of RS codes can have various lengths while retaining the same error-correcting capability, while many other error-correcting codes are restricted to their length.

In this chapter, a brief description of the Reed-Solomon codes is given in Section 3.1. The details of the RS codes are given in [49]-[51]. In the rest of this chapter, several implementations of the RS codes and the uses of error correction codes (including the RS codes and other codes) are reviewed. A decoding scheme that uses the transform decoding algorithm and its implementation is described in Chapter 5.

3.1 Reed-Solomon Codes

3.1.1 Finite Field Arithmetic

Reed-Solomon codes are defined on finite field called a Galois field which is denoted as $GF(p^m)$, where p is a prime integer and m is a positive integer. There are p^m distinct elements, including 0, in $GF(p^m)$, and p and m are called the characteristic and the order of the field. Each finite field element can be represented in by an exponential, a polynomial of degree (m-1), and an m-tuple vector. Addition and multiplication defined in $GF(p^m)$ imply modulo-p addition and multiplication. In digital applications, the finite fields with characteristic $p = 2$ are mostly used because of its binary implementation. Table 3.1 shows the three representations for the elements of $GF(2^4)$ generated by a primitive polynomial $P(x) = 1 + x + x^4$. Note that α is an element in $GF(2^4)$. To add α^5 and α^7, for example, the polynomial representation is used as

$$\begin{aligned}\alpha^5 + \alpha^7 &= (\alpha + \alpha^2) + (1 + \alpha + \alpha^3) \\ &= 1 + \alpha^2 + \alpha^3 \\ &= \alpha^{13},\end{aligned}$$

and to multiply α^5 and α^7 the exponential representation is used as

$$\alpha^5 \cdot \alpha^7 = \alpha^{12} = 1 + \alpha + \alpha^2 + \alpha^3.$$

Table 3.1 Three representations for the elements of $GF(2^4)$ generated by a primitive polynomial $P(x) = 1 + x + x^4$.

Exponential representation	Polynomial representation	4-Tuple representation
0	0	(0 0 0 0)
1	1	(1 0 0 0)
α	α	(0 1 0 0)
α^2	α^2	(0 0 1 0)
α^3	α^3	(0 0 0 1)
α^4	$1 + \alpha$	(1 1 0 0)
α^5	$\alpha + \alpha^2$	(0 1 1 0)
α^6	$\alpha^2 + \alpha^3$	(0 0 1 1)
α^7	$1 + \alpha + \alpha^3$	(1 1 0 1)
α^8	$1 + \alpha^2$	(1 0 1 0)
α^9	$\alpha + \alpha^3$	(0 1 0 1)
α^{10}	$1 + \alpha + \alpha^2$	(1 1 1 0)
α^{11}	$\alpha + \alpha^2 + \alpha^3$	(0 1 1 1)
α^{12}	$1 + \alpha + \alpha^2 + \alpha^3$	(1 1 1 1)
α^{13}	$1 + \alpha^2 + \alpha^3$	(1 0 1 1)
α^{14}	$1 + \alpha^3$	(1 0 0 1)

3.1.2 Reed-Solomon Codes

A Reed-Solomon code from $GF(2^m)$ that corrects at most t symbol errors has the following parameters:

Codeword length: $n = 2^m - 1$,

Number of parity-check symbols: $n - k = 2t$,

Minimum distance: $d_{min} = 2t + 1$.

In practical applications, each $GF(2^m)$ element or code symbol contains m binary bits, and each RS codeword consists of mn binary bits which include $2mt$ parity-check bits.

The code rate r is defined as the ratio of the number of information symbols to codeword length n, i.e., k/n. One way to construct (encode) the t-error-correcting RS code is to use a generator polynomial

$$g(x) = \prod_{j=1}^{2t} (x - \alpha^j) \tag{3.1}$$

with roots of $2t$ consecutive powers of α. Assume that there is a block of k information symbols $(u_0, u_1, ..., u_{k-1})$ each of which is an element of $GF(2^m)$. This block of symbols is also expressed in polynomial from as $u(x) = u_0 + u_1x + ... + u_{k-1}x^{k-1}$. Then, a code polynomial $v(x)$ is constructed as the product of the information polynomial $u(x)$ and the generator polynomial $g(x)$. That is

$$v(x) = u(x) \cdot g(x) = v_0 + v_1x + \cdots + v_{n-1}x^{n-1}, \tag{3.2}$$

where $n = k + 2t$. However, the RS codes usually appear in a systematic form, rather than the above. A systematic RS codeword contains two parts, k information symbols and $2t$ parity-check symbols. The $2t$ parity-check symbols are the coefficients of the remainder resulting from dividing the information polynomial $x^{2t}u(x)$ by the generator polynomial $g(x)$.

When decoding a retrieved RS code from a communication channel or a data storage, a syndrome containing $2t$ symbols is calculated from the retrieved codeword. Each syndrome corresponds to an error pattern which corrupts the original codeword during transmission. The error pattern results from an error-location polynomial which is obtained from the syndrome using a modified Euclid's algorithm. The location of errors is the reciprocal of the roots of the error-location polynomial. A decoding scheme, the transform decoding algorithm [12], as well as its implementation will be described in details in Chapter 5. Another decoding scheme is to record all the error patterns in a look-up table which is accessed by the calculated syndrome. However, the look-up

table scheme is only applied to RS codes with short length because of the extensive hardware involved in building a huge table for long RS codes.

The Reed-Solomon codes provide wide flexibility to define different length codes which have a good capability in error correction. The previously discussed codeword length of $n = 2^m - 1$ is referred to as the natural length of the RS codes, and the RS code of that length is called the primitive RS code. In many applications, shortened codes, in which the higher order information symbols are simply replaced by zeros, are used. Along with $2t$ parity-check symbols the shortened codes can still correct up to t symbol errors with the same decoding scheme. For example, two shortened RS codes, (32, 28) and (28, 24) RS codes in $GF(2^8)$ are cross interleaved and used in compact disks to correct 4 symbol errors. Different from the shortened RS codes, extension RS codes are also used. The extension RS codes add one more parity-check symbol over all the information symbols.

3.1.3 Performance of Error-Correction Using the Reed-Solomon Codes

The performance of the Reed-Solomon codes is evaluated by the output bit-error probability, or bit-error rate (BER). An upper bound of the output BER for a t-error-correcting (n, k) RS code in $GF(2^m)$ is given by [52]

$$P_e \le \frac{2^{m-1}}{2^m - 1} \sum_{j=t+1}^{n} \frac{j+t}{n} \binom{n}{j} \cdot P_s^{\,j} \left(1 - P_s\right)^{n-j}, \tag{3.3}$$

where P_b is the raw (input) bit-error probability and $P_s = 1 - (1 - P_b)^m$ is the symbol-error probability. In Fig. 3.1, the vertical axis shows the decoded BER at the output of an RS decoder for the primitive RS codes (i.e., $n = 2^m - 1$) with code rate r around 0.8, and the horizontal axis shows the raw BER at the input. For a fixed r, the performance curves of the codes with longer codewords drop rapidly showing that they have better

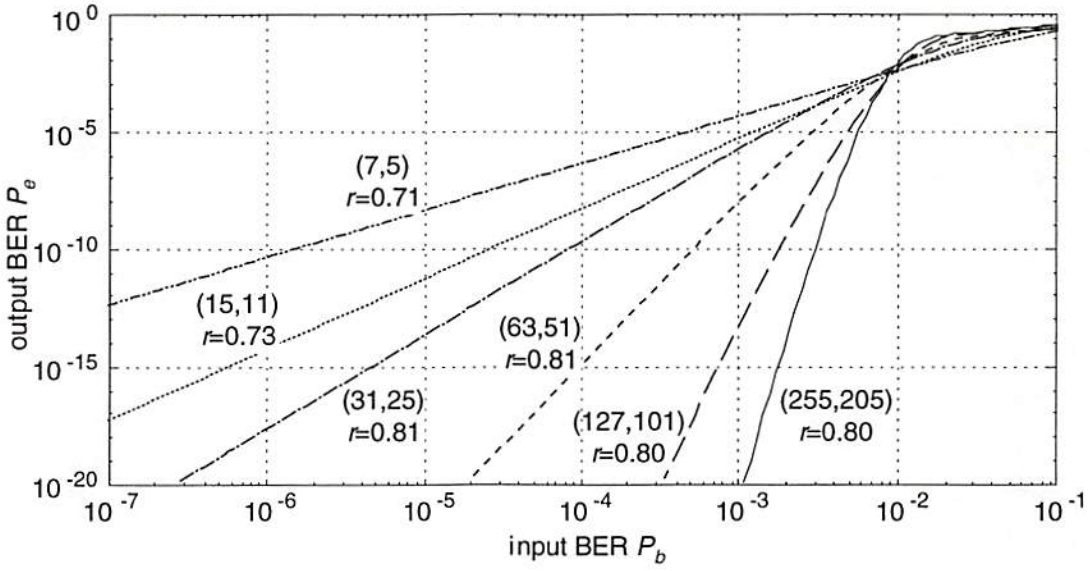


Figure 3.1 The output bit-error rate (P_e) versus the raw (input) bit-error rate (P_b) for the primitive Reed-Solomon codes with code rate ≈ 0.8.

error-correcting capability. Figure 3.2 shows the performance of the primitive RS codes which reduce the output BER down below 10^{-12} and 10^{-15} at a raw BER 10^{-4} as the dotted lines shown. The parameters of these RS codes are listed in Table 3.2 showing that, for the codes of similar capability, the code rate of the RS codes with longer codewords is always larger. It shows that RS codes with short codeword length are inefficient in the use of information space and bandwidth. In Fig. 3.3, the code rate to satisfy the required output BERs of 10^{-9}, 10^{-12}, and 10^{-15} at the raw BER of 10^{-4} is calculated for the primitive RS codes (shown by dotted lines), the RS codes over $GF(2^5)$, and over $GF(2^8)$, respectively. Note that the r values of the RS codes shown in Fig. 3.2 are depicted by two of the dotted lines in Fig. 3.3.

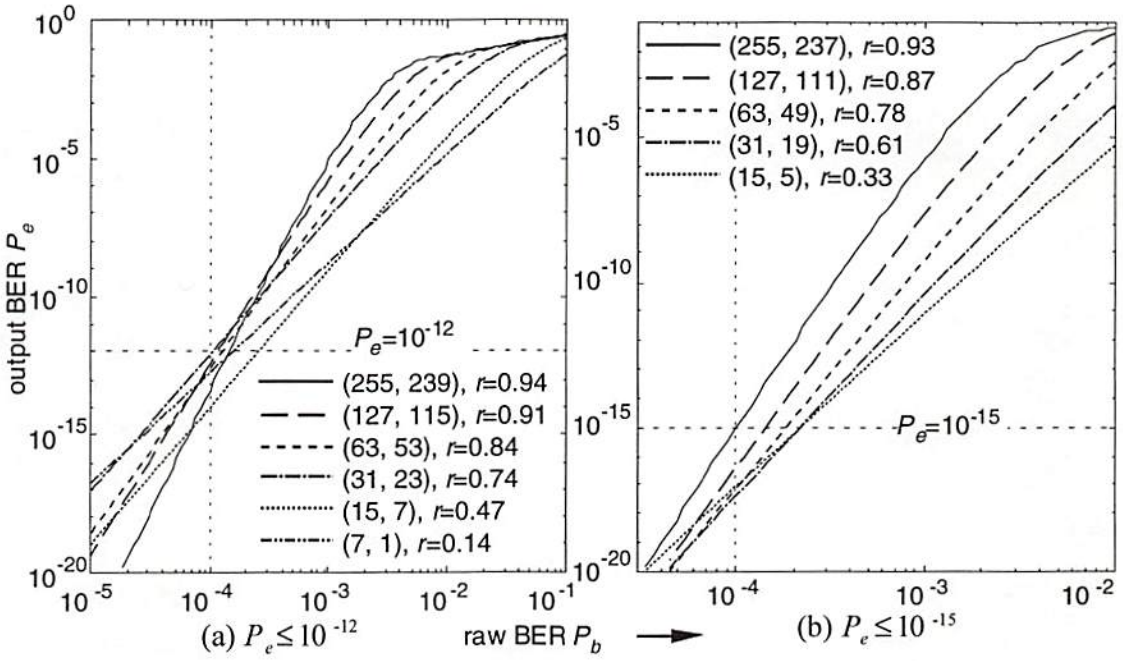


Figure 3.2 Performance of the primitive RS codes for (a) $P_e \leq 10^{-12}$ and (b) $P_e \leq 10^{-15}$ at $P_b = 10^{-4}$.

Table 3.2 Parameters for the Reed-Solomon codes which reduce BER from 10^{-4} to 10^{-12} and 10^{-15}.

$10^{-4} \rightarrow 10^{-12}$				$10^{-4} \rightarrow 10^{-15}$			
(n, k)	m	t	r	(n, k)	m	t	r
(7, 1)	3	3	0.14	(non existent)	3	-	-
(15, 7)	4	3	0.47	(15, 5)	4	5	0.33
(31, 23)	5	4	0.74	(31, 19)	5	6	0.61
(63, 53)	6	5	0.84	(63, 49)	6	7	0.78
(127, 115)	7	6	0.91	(127, 111)	7	8	0.87
(255, 239)	8	8	0.94	(255, 237)	8	9	0.93

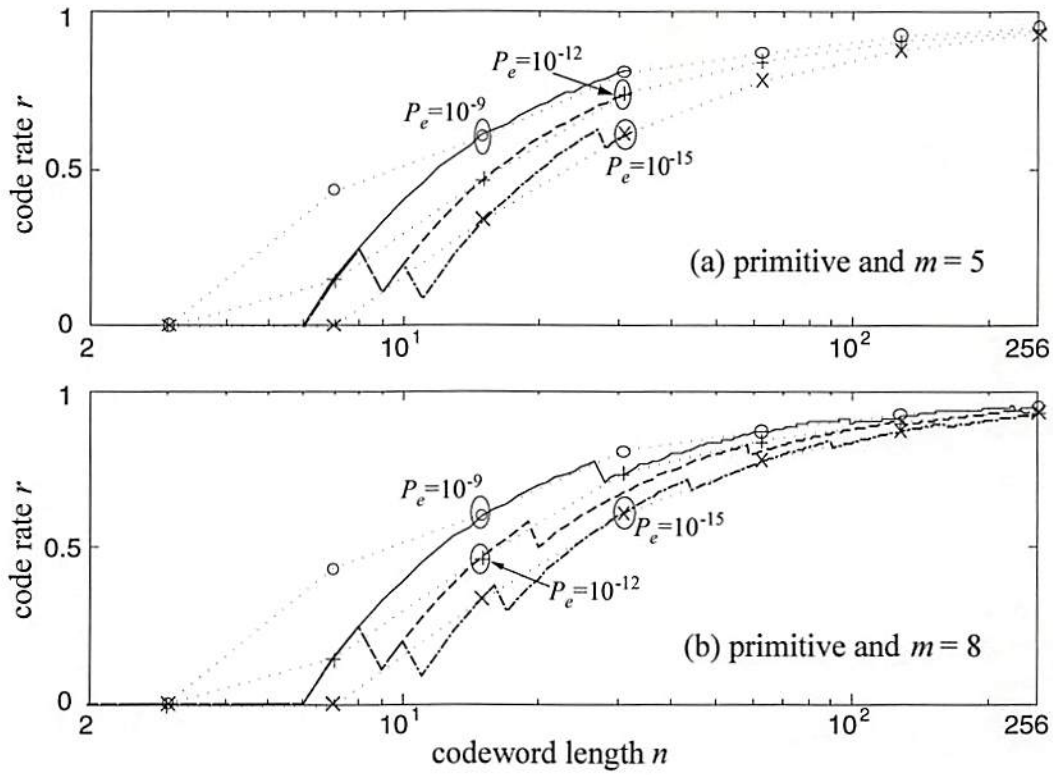


Figure 3.3 Code rate r versus codeword length n for the primitive RS code (dotted lines), (a) $m = 5$, and (b) $m = 8$ RS codes at $P_b = 10^{-4}$.

3.2 Transform Decoding Algorithm (TDA) for the Reed-Solomon Codes

The transform decoding algorithm utilizes a modified form of Euclid's algorithm to decode RS codes [12]. The goal of the algorithm is to compute an error sequence corresponding to a codeword so that the corrected codeword is obtained by adding the error sequence to the codeword. The decoding diagram shown in Fig. 3.4 is described in the following four major steps.

Step 1, syndrome computation. Let a received codeword be

$$R = (r_{n-1}, r_{n-2}, \ldots, r_1, r_0) = r_{n-1}x^{n-1} + r_{n-2}x^{n-2} + \cdots + r_1x^1 + r_0.$$

Then, the syndrome S_j of the received codeword are computed as

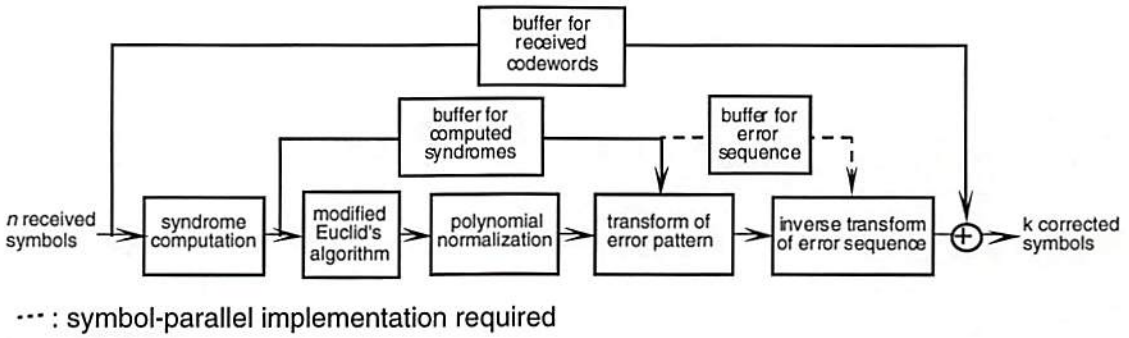


Figure 3.4 Transform decoding algorithm (TDA) for the Reed-Solomon codes.

$$S_j = \sum_{i=0}^{n-1} r_i \alpha^{ij}, \text{ for } 1 \le j \le 2t, \tag{3.4}$$

where α is a primitive element in $GF(2^m)$. The first $2t$ elements of the error sequence are equal to the syndromes, i.e., $E_j = S_j$ for $1 \le j \le 2t$. The $2t$ syndrome symbols are stored in a buffer waiting for the error-locator polynomial to be computed in the next step.

Step 2, determination of error-locator polynomial using modified Euclid algorithm. The modified Euclid algorithm is performed on x^{2t} and the syndrome polynomial

$$S(x) = \sum_{j=1}^{2t} S_j x^{2t-j} \tag{3.5}$$

to obtain the polynomial

$$\lambda(x) = \lambda_0 x^t + \lambda_1 x^{t-1} + \cdots + \lambda_t. \tag{3.6}$$

A short version of the modified Euclid's algorithm is given as

$$R_{i+1}(x) = \sigma_i \left[b_i R_i(x) - x^{|l_i|} a_i Q_i(x) \right] + \overline{\sigma}_i \left[a_i Q_i(x) - x^{|l_i|} b_i R_i(x) \right], \tag{3.7.a}$$

$$Q_{i+1}(x) = \sigma_i Q_i(x) + \overline{\sigma}_i R_i(x), \tag{3.7.b}$$

$$\lambda_{i+1}(x) = \sigma_i \left[b_i \lambda_i(x) - x^{|l_i|} a_i \mu_i(x) \right] + \overline{\sigma}_i \left[a_i \mu_i(x) - x^{|l_i|} b_i \lambda_i(x) \right], \tag{3.7.c}$$

$$\mu_{i+1}(x) = \sigma_i \mu_i(x) + \overline{\sigma}_i \lambda_i(x), \tag{3.7.d}$$

where a_i and b_i are the leading coefficients of $R_i(x)$ and $Q_i(x)$, respectively,

$$l_i = \deg\big(R_i(x)\big) - \deg\big(Q_i(x)\big), \text{ and} \tag{3.7.e}$$

$$\sigma_i = \begin{cases} 1, & \text{if } l_i \geq 0 \\ 0, & \text{if } l_i < 0. \end{cases} \tag{3.7.f}$$

The initial conditions are $R_0(x) = x^{2t}$, $Q_0(x) = S(x)$, $\lambda_0(x) = 0$, and $\mu_0(x) = 1$, and the algorithm stops when $\deg(R_i(x)) < t$. Note that σ_i is a switch control: when $\sigma_i = 1$, the first term of the RHS of (3.7.a) and (3.7.c) is computed; otherwise, the second term is computed.

Then, the standard error-locator polynomial is obtained by normalizing the nonzero leading coefficient of $\lambda(x)$ as

$$\sigma(x) = x^t + \sigma_1 x^{t-1} + \cdots + \sigma_t \text{ , where } \sigma_i = \lambda_i / \lambda_0 \text{ for } 1 \leq i \leq t, \tag{3.8}$$

assuming $\lambda_0 \neq 0$.

Step 3, transform of error pattern. The unknown elements of an error sequence E_j, for $(2t+1) \leq j \leq N$, where $E_N = E_0$, are computed from the coefficients σ_t of the standard error-locator polynomial and the $2t$ computed error sequence E_j ($= S_j$, $1 \leq j \leq 2t$) by the equation

$$E_{2t+j} + \sum_{i=1}^{t} (-1)^t \sigma_t E_{2t+j-i} = 0 \text{ , for } j \geq 1. \tag{3.9}$$

Note that $N = 2^m - 1 \geq n$.

Step 4, inverse transform of the error sequence. The error pattern (e_0, e_1, ..., e_{N-1}) which is the inverse transform the error sequence E_j, for $0 \leq j \leq N-1$, is computed as

$$e_i = \sum_{j=0}^{N-1} E_j \alpha^{-ij} \text{ , for } 0 \leq i \leq N-1. \tag{3.10}$$

In practice, only the k ($= n - 2t$) information symbols are of interest, and therefore only e_i, for $2t \leq i \leq n-1$, have to be computed.

The original codeword (if the number of error symbols is less than or equal to t) is then recovered by adding the error pattern e_i to the received codeword r_i stored in a buffer, i.e.,

$$\overline{u}_i = r_i + e_i, \text{ for } 0 \le i \le k\text{-}1. \tag{3.11}$$

3.3 Applications of the Reed-Solomon Codes

When Reed-Solomon codes are applied to digital systems, simple, regular RS codes are seldom used. In practical applications, interleaving and combining of the RS codes is frequently used. The interleaving of several codewords breaks up a burst error into several shorter ones and, thus, makes correction of the burst easier. The combining of two codes improves the error-correction capability and makes the decoder design simpler. In Table 3.3, the construction and properties of RS codes for various applications [7], [8], [49], [53] is summarized for reference.

Burst errors resulting from material defects or particle noise may corrupt hundreds to thousands of data bits during transmission or recording. To break a huge burst error into smaller ones, a group of codewords is transmitted in such a way that the code symbols of the same order are sent into the channel in sequence, than the next order symbols, and so on. The number of codewords in the group is called the interleaving depth, and its selection depends on the size of the expected burst error.

Combining two smaller codes can simplify the decoder design and improve the error correcting capability. In turn, the code efficiency or the code rate is increased while maintaining the same error correcting capability. Concatenated, cross-interleaved, and product codes belong to this category. When encoding, for example, the data of a product code are arranged in a $k_1 \times k_2$ array. The k_2 columns each of k_1 elements are first encoded individually by an (n_1, k_1) encoder which results in an $n_1 \times k_2$ array. Then,

the rows are encoded by an (n_2, k_2) encoder resulting in an $n_1 \times n_2$ code array. The decoding is performed in reverse order by decoding the rows of the code array followed the columns.

Table 3.3 Applications of Reed-Solomon codes.

application	(n, k) code	t (symbols)	Galois field	r	structure	data rate
deep space communications	(255, 223)	16	$GF(2^8)$	0.88	concatenated + interleaved	-
magnetic-disk mass storage (IBM 3370/80)	(174, 171)	1	$GF(2^8)$	0.98	interleaved	-
Digital Cypress (IBM photodigital mass storage)	(61, 50)	5	$GF(2^6)$	0.82	(shortened)	-
digital HDTV	(255-l,235-l)	10	$GF(2^8)$	≤ 0.92	regular	80 Mbps
audio CD	(32, 28) + (28, 24)	4	$GF(2^8)$	0.75	cross-interleaved	1.41 Mbps
540 MBytes CD-ROM	(45, 43) + (26, 24)	2	$GF(2^8)$	0.88	cyclic redundancy check (CRC)	≤ 7.2 Mbps
2.6 GBytes CD-ROM	(39, 35)	2	$GF(2^8)$	0.90	interleaved + CRC	≤ 7.2 Mbps

3.4 Error-Correction in Optical Computing

As used in digital electrical processors, error correction codes have been used in optical computing systems, such as systolic array processors [54], optical associative memories [55], and matrix-vector multipliers [56], [57], to improve the accuracy of the systems. In optical matrix-vector multipliers, extra columns or rows of the codes are appended to the original matrix. By evaluating the redundant elements at the output

vector, errors resulting from device defects and system noise can be either corrected or detected and, thus, the accuracy is improved. In these applications, binary Hamming codes are in general employed because of the easier encoding and decoding.

An all-optical encoding/decoding scheme was presented for error correction of binary linear codes [58]. The encoding process was achieved using a binary-amplitude mask in an imaging system used as an optical vector-matrix multiplier. The decoding process was performed by such an optical multiplier for syndrome computations. Then, the corresponding error pattern was retrieved from an optical lookup table which is followed by an optical modulo-2 adder working with a large intensity dynamic range. In particular, the all-optical scheme was applied to a (7, 4) Hamming code which corrects a bit error in a block of 7 binary bits. The best data throughput was estimated to reach 56 gigabits per second. The optical coding/decoding scheme, however, can only apply to simple binary linear codes. For error-correction codes requiring complicated encoding/ decoding processes such as the BCH and the RS codes, electrical circuits are needed.

3.5 Parallel Reed-Solomon Decoder: The Previous Work

Error correction coding techniques have been used to reduce the bit-error rate at the output of optical page-oriented memories. In addition, the use of error correction can increase the usable capacity of the memory [6]. The output signal-to-noise (SNR) ratio and so the bit-error rate (BER) is degraded when more data pages are recorded in a memory medium. At a desired BER, only a certain number of data pages can be recorded. However, using a small portion of the memory capacity for the error-correction codes, more data pages can be recorded since the degraded output BER is decreased by decoding the readout data. The output SNR, for example, was about 20

for 40 holograms recorded in a Cu-doped KNSBN crystal. The SNR was degraded to 3 when 100 holograms were recorded. As a Reed-Solomon code of code rate 0.8 was used to encode the input data, the decoded SNR of the 100 holograms was improved to 20 again. The effective memory capacity, equal to 80 (= 100×0.8) holograms, was twice of the uncoded memory.

A parallel electronic RS decoder with 60 optical inputs for a (15, 9) RS code in $GF(2^4)$ was proposed, and it provided an effective data throughput of 300 megabits per second [10], [11]. The syndrome and error correction were performed by electrical vector-matrix multipliers structured as a programmable logic array (PLA). The error-location polynomial was obtained by a 2-D processor array in which finite field multiplications were performed by accessing a table-lookup RAM. In a fixed VLSI area, 10 cm^2, with an output BER below 10^{-12} at a raw BER of 10^{-4}, the optoelectronic (OE) parallel RS decoder for the $GF(2^6)$ RS code (primitive) of codeword length 63 provided the largest effective data throughput. The OE parallel RS decoders were able to provide a data throughput up to 10^{12} bits per second if a 0.1-μm CMOS process were used in the fixed area. Compared with an array of conventional symbol-serial RS decoders (called bit-parallel/symbol-serial (BPSS) in Chapter 5), the OE parallel RS decoder provides larger data throughput, and more efficient utilization of VLSI area. In addition, the syndrome and error correction modules can be replaced by optical vector-matrix multipliers to reduce the fabrication area and power dissipation so as to increase the data throughput. The optical vector-matrix multiplier used in decoding the RS code was implemented using a binary mask which was followed by an optical modulo-2 device and an electrical circuit for finite field logic.

Chapter 4

Performance Analysis Scenarios of Smart Pixel Interfaces for Optical Page-Oriented Memories

Optical page-oriented memories (OPOMs) require input/output (I/O) interfaces to transfer large amounts of information without resulting in an I/O bottleneck. This chapter discusses the application of smart pixel techniques to construct interfaces between OPOM systems and data transmission networks which provide satisfactory aggregate data rates. The following sections describe a performance analysis and outline system characteristics, assumptions and symbols used in the design of an OPOM system.

4.1 Smart Pixel Interfaces of the Optical Page-Oriented Memory Systems

A schematic diagram of an OPOM and its components was given in Section 2.1. Here, we summarize additional functions of the interface components:

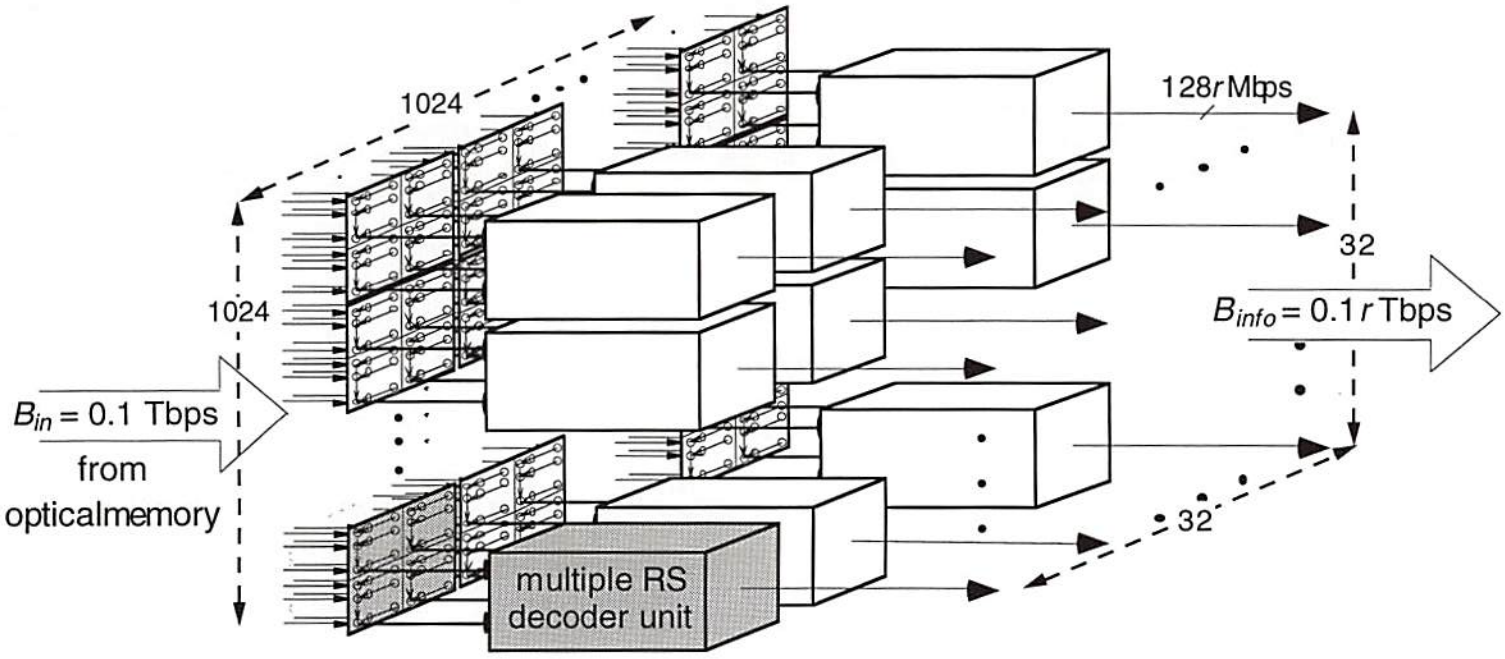
i) Format conversion. The conversions include serial-to-parallel/parallel-to-serial (spatial) and wavelength conversions. Because the number of I/O channels in OPOMs differs from that in physical network and interconnection hardware by several orders of magnitude, the OPOMs need spatially demultiplexing and multiplexing at the input and output interfaces, respectively. Wavelength conversion is needed because of different optical wavelengths used in OPOM storage materials and optoelectronic SP devices.

ii) Interface to wave guides and free-space input/outputs. Direct coupling of optical signals from/to array of waveguides (e.g., optical fibers) requires precise alignment, and relay lenses and lenslet arrays are required to transfer optical signals.

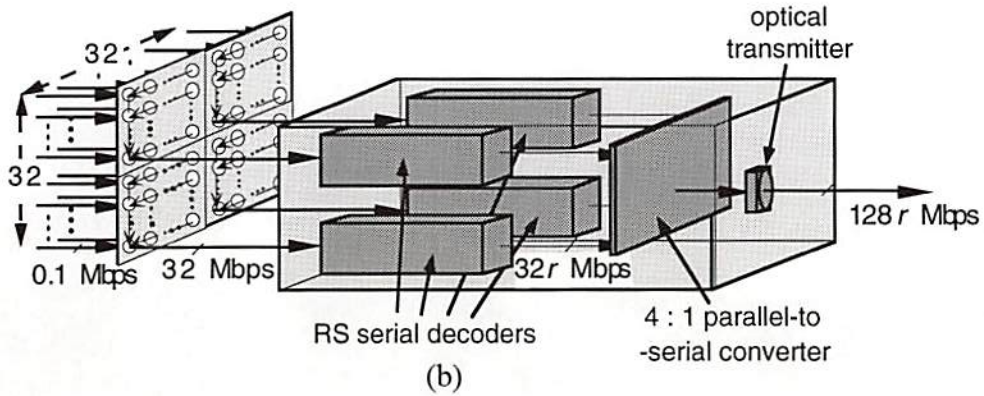
iii) Error encoding/decoding. Due to the inherent noise and crosstalk in memory materials and OE components (Section 2.1), errors will occur at the memory output. The SP interface needs error control coding techniques to correct errors, increase the reliability of the retrieved data, and decrease output BER.

Because the decoding processes are more complicated than the encoding process, we concentrate on designs of the output interfaces. Figure 4.1 shows a conceptual structure of an output interface performing the photo detection, error correction, and parallel-to-serial conversion. A CCD array or an array of photodetectors receives an optical data page retrieved from the memory medium. Error-correction decoders implemented by electrical circuitry are connected to the photo detector array and decode the retrieved data. The decoding of retrieved data requires the most hardware and results in the longest delays, as we describe in the following sections. An array of many-to-one spatial multiplexers performs the parallel-to-serial conversion, and an optical transmitter array converts the electrical signal into optics. The multiplexing can be achieved using shift registers. An array of shift registers are grouped, and each group is associated with an optical transmitter. The shift registers first buffer the decoded data bits. Then the transmitter converts the decoded data bits into an optical signal and transfers them to output port in sequence.

The array of error-control decoders in the output interface is required to provide high data throughput and low output bit-error probability. We assume that each data page containing $1{,}024 \times 1{,}024$ bits is accessed in 10 μs. The data throughput is then 10^{11} bits per second, i.e., 0.1 terabit per second. These projected limits will be used by



(a)



(b)

Figure 4.1 (a) Conceptual structure of the smart pixel interface using serial decoders, and (b) a multiple decoder unit. We assume 1024×1024 inputs (0.1 Mbps/channel) and 32×32 outputs ($128r$ Mbps/channel). The input page is divided into 32×32 blocks, and each contains 32×32 bits. Each block requires 4 serial RS decoders, for example, to decode the 1024 bits in 10 μs.

the OPOM systems in the next few years. The uncoded BER is assumed at 10^{-4}, and the output BER is required to reduce down to the range of 10^{-9} to 10^{-15}. Our goals are to develop design schemes for the SP interfaces that satisfy the two requirements simultaneously under limits of fixed chip area (assumed 10 cm^2) and fixed power dissipation (assumed 1 to 5 Watts per cm^2), and to define parameters to evaluate the performance. These specifications are summarized in Table 4.1.

In Fig. 4.1, the serial Reed-Solomon decoders, for example, are assumed to decode retrieved data bits in this structure. The input data page contains 1,024 × 1,024 bits. The data page is divided into 32 × 32 blocks. Each block is further divided into 4 sub-blocks and so each contains 256 input channels. The number of sub-blocks is determined by the number of data bits which can be decoded by a single decoder in a memory access period, i.e., 10 μs. In each multiple RS decoder unit, there are 4 serial RS decoders, and each is electrically connected to a sub-block of input channels. The outputs of the four RS decoders are multiplexed by a 4-to-1 parallel-to-serial converter, and finally the electrical signal is converted to an optical output by an optical transmitter.

4.2 Maximum-Throughput Scenario

Each smart pixel node is an electrical RS decoder with optical I/O. In this section, we define the parameters that determine the performance of the RS decoder.

The numbers of logical gates and transistors of a decoder for an (n, n-2t) RS code on GF(2^m) can be determined when the I/O format and finite-field multiplier (FFM) are specified (see Chapter 5). The modified SUSPENS model estimates the decoder area A_{RSdc} from the number of logical gates and other VLSI parameters. Then, in a fixed area A_{pg}, there are

$$N_{D-A} = \left\lfloor \frac{A_{pg}}{A_{RSde}} \right\rfloor \tag{4.1}$$

RS decoders fabricated in a decoder array, where $\lfloor x \rfloor$ represents the largest integer less than or equal to x. The input spatial channel density d_{scin} is defined as the number of input channels per unit area (channels per cm^2), and is given as

$$d_{scin} = \frac{N_{D-A} \cdot [\text{no. of inputs per decoder}]}{A_{pg}}. \tag{4.2}$$

The numerator in Eq. (4.2) presents a number of bits N_{blk} which are simultaneously input to the SP interface, and called a data block,. For example, a data block in an OPOM, as shown in Fig. 4.1, contains 4,096 bits because there are 32×32 multiple RS decoder units and each unit has 4 input channels. To compute the block rate, an arbitrary clock frequency f_c ($\leq f_{c,max}$, Eq. (2.9)) is assumed in Eq. (2.13), and the total power P_{RSde} dissipated by a decoder at f_c is obtained using Eqs. (2.13) and (2.14). Then the block rate f_{blk} as limited by a given power density P_{pg} is calculated by

$$f_{blk} = \frac{f_c \cdot P_{pg}}{N_{D-A} \cdot P_{RSde}}. \tag{4.3}$$

The data input rate, defined as the total number of data bits input to the decoder array per time unit, is given by the product of the data block N_{blk} and the block rate f_{blk}, i.e.,

$$B_{in} = N_{blk} \cdot f_{blk}. \tag{4.4}$$

Because there are k information symbols in a RS codeword of n symbols (code rate $r = k/n$), only a fraction r ($0 < r < 1$) of the input data carries information, and $(1 - r)$ of it is for parity-check data, the effective information rate at the output of the decoder array is then given by

$$B_{info} = r \cdot B_{in} = r \cdot N_{blk} \cdot f_{blk}. \tag{4.5}$$

The information spatial channel density d_{info} is then defined as the number of bits which are obtained at the output of the decoder array in a unit time and a unit area (bits per second per cm^2), and is expressed as

$$d_{info} = \frac{B_{info}}{A_{pg}} = r d_{scin} f_{blk}. \tag{4.6}$$

The d_{info} parameter is one of the most important in evaluating the performance of the RS decoder designs. Another important parameter P_{idec} is defined as the product of d_{info} and the effective capacity of the memory medium, i.e.,

$$P_{idec} = d_{info} \cdot rM, \tag{4.7}$$

where M is the total capacity of the memory medium and rM is the capacity available for storing information. Therefore, P_{idec} is used to evaluate the overall performance of an OPOM system with error correction coding.

4.3 Largest-Capacity Scenario Using Four Interface Constraints

When designing an output interface which consists of an array of Reed-Solomon decoders for the OPOMs, we consider four constraints: the bit error constraint (BEC); the code rate constraint (CRC); the buffer length constraint (the minimum number of required decoders) (MINC); and the power/area constraint (the maximum number of decoders) (MAXC). These constraints are classified into two categories, code-dependent and VLSI-dependent constraints. The first category includes the BEC and CRC which only depend on properties of the RS codes. The second category includes the MINC and MAXC constraints and, the VLSI specifications and parameters that are imposed to determine the possible RS codes and decoders. Feasible RS codes and decoders are then specified as limited by the decoding delay, buffer length, interface area, and power density. In the following, these constraints are described in detail.

Bit Error Constraint (BEC)

The RS codes achieving a desired BER P_e given a raw input BER P_b satisfy Eq. (3.3) which is rewritten here as

$$P_e \le \frac{2^{m-1}}{2^m - 1} \sum_{j=t+1}^{n} \frac{j+t}{n} \binom{n}{j} \cdot P_s^{\,j} \left(1 - P_s\right)^{n-j}, \tag{4.8}$$

where $P_s = 1 - (1 - P_b)^m$ is the symbol error rate. For given P_b and P_e, there are a number of (n, m, t) sets satisfying Eq. (4.8). In addition, for each pair of n and m, the smallest t is selected as the optimal solution because it corresponds to the highest code rate in the pair of n and m. Note that Eq. (4.8) can be applied to primitive codes as well as all the shortened and extension RS codes.

Code Rate Constraint (CRC)

The code rate r $(= k / n)$ is an important measure of both the memory and the data throughput efficiency. Substituting $n - 2t$ for k yields

$$r = 1 - \frac{2t}{n}. \tag{4.9}$$

A minimum code rate r_q required to provide high efficiency and large usable capacity can then be specified as

$$r = 1 - \frac{2t}{n} \ge r_q. \tag{4.10}$$

Buffer Length (Minimum Number of Decoders) Constraint (MINC)

The minimum number of RS decoders required to provide high data throughput and to prevent access bottlenecks depends on the size of the decoder buffers and the longest decoding delay of a codeword.

Given a codeword delay D_{long} (discussed in Section 5.4) and a memory access time t_a, the number of codewords that are processed by an RS decoder in t_a is

$$B_{deW} = \left\lfloor \frac{t_a}{D_{long}} \right\rfloor \text{ codewords,} \tag{4.11}$$

corresponding to mnB_{deW} binary bits. Note that B_{deW} must be greater than 0, i.e., $t_a \geq D_{long}$, otherwise the selected decoder fails to provide the necessary data rate, and results in an extra access delay. In this case, either other decoder designs are considered or the data-page access time t_a is increased. Because a data page contains N_{in} bits, an interface needs at least

$$N_{D-D} = \left\lceil \frac{N_{in}}{mnB_{deW}} \right\rceil = \left\lceil \frac{N_{in}}{mn\left\lfloor t_a / D_{long} \right\rfloor} \right\rceil \tag{4.12}$$

decoders to process a retrieved data page in a memory access cycle. Here, $\lceil x \rceil$ is the smallest integer that is larger than or equal to x. $N_{D\text{-}D}$ is the minimum number of RS decoders needed by the specified interface.

Power/Area (Maximum Number of Decoders) Constraint (MAXC)

Given an (n, k) RS code over GF(2^m), the decoder area A_{RSde} and the power dissipation P_{RSde} can be estimated using Eqs. (2.8) and (2.13). If the allowed chip area of the interface is A_{pg}, at most a certain number of RS decoders can be fabricated on the chip. With a limited power density only a certain number of RS decoders can operate at the selected clock rate. The maximum number of RS decoders that can simultaneously operate is given by the smaller of the two. In a given area A_{pg},

$$N_{D-A} = \left\lfloor \frac{A_{pg}}{A_{RSde}} \right\rfloor \tag{4.13}$$

RS decoders can be fabricated. Since the power consumed by a decoder is P_{RSde} at a clock rate f_c, the number of decoders that can operate at the same time without excess power dissipation is

$$N_{D-P} = \left\lfloor \frac{P_{pg}}{P_{RSde}} \right\rfloor, \tag{4.14}$$

where P_{pg} is the power that can be dissipated in area A_{pg}. Therefore, the number of RS decoders that can simultaneously process a data page at a fixed clock rate f_c is given by

$$N_{Dmax} = \min(N_{D\text{-}A}, N_{D\text{-}P}). \tag{4.15}$$

Figure 4.3 illustrates the relationship between the four interface constraints. The code-dependent constraints, BEC and CRC, specify some RS codes that fulfill the requirements on the fidelity of retrieved data and throughput efficiency. The use of the two constraints is independent of the VLSI technology. The VLSI-dependent constraints, MINC and MAXC, specify some RS codes and the decoder designs that satisfy the limits on buffer length, power dissipation, and allowed area. If $N_{D\text{-}D} \leq \min(N_{D\text{-}A}, N_{D\text{-}P})$, then there exist some sets of code parameter that satisfy the two constraints; otherwise, no decoders and RS codes can achieve the system requirements.

4.4 System Characteristics, Assumptions and Notations

In this section, the parameters used in the thesis are tabulated according to their use in individual sections. Table 4.1 lists the characteristics of the OPOM system and the specifications of the SP interfaces which are summarized from Section 4.1.

Table 4.2 tabulates the notation, the descriptions, and the reference sections, of the parameters which are used to analyze the performance of the decoder design and the SP interface in Sections 4.2 and 4.3.

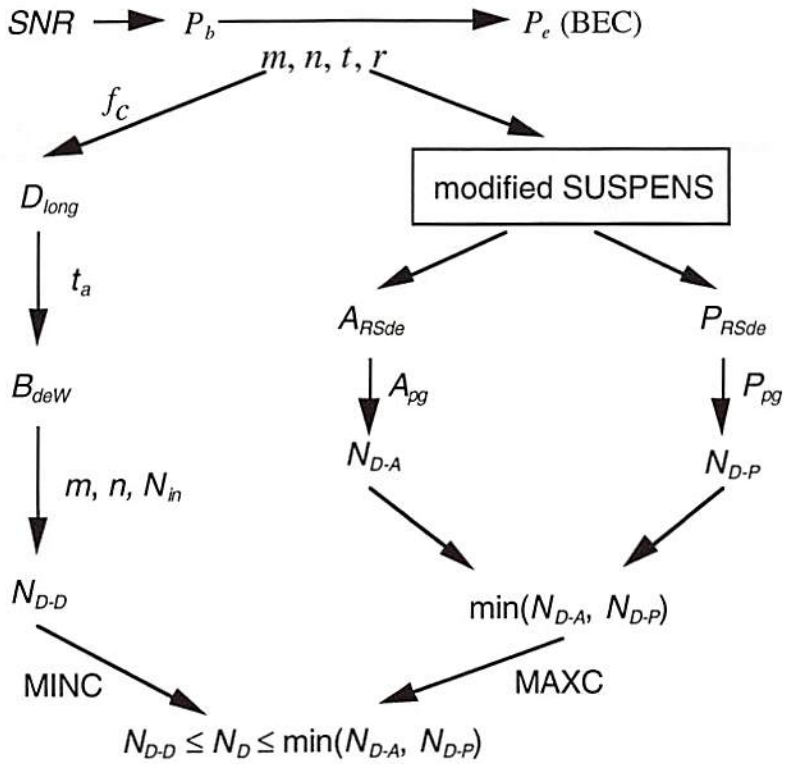


Figure 4.2 Relationship between the four interface constraints for the largest capacity scenario.

Table 4.1 Characteristics and specifications of smart-pixel interfaces.

data page size N_{in}	1,024 × 1,024 bits
page access time or page rate t_a	10^{-5} sec. or 100 KHz
electrical circuit clock rate f_c	100 MHz (0.8-μm CMOS)
raw bit-error probability P_b	10^{-4}
total area of interface A_{pg}	10 cm^2
power dissipation P_{pg} in A_{pg}	1 - 5 Watts/cm^2
aggregate input data rate	10^{11} bits per second
output bit-error probability P_e	10^{-9} - 10^{-15}

Table 4.2 Notation used in the RS decoder and the SP interface implementation.

symbol	definition or description	reference
A_{pg}	total allowed area of an SP interface (10 cm^2)	Sect. 4.1
A_{RSde}	the estimated area of a TDA RS decoder	Sect. 2.3
B_{deW}	no. of codewords per decoder in t_a	Sect. 4.3
B_{in}	aggregate input data rate in A_{pg} limited by P_{pg}	Sect. 4.2
B_{info}	output data rate in A_{pg} limited by P_{pg}	Sect. 4.2
d_{info}	information spatial channel density (bits/second/cm^2)	Sect. 4.2
D_{long}	longest delay time of a codeword (second)	Sect. 4.3/5.4
d_{scin}	input spatial channel density of the TDA decoder	Sect. 4.2
f_{blk}	block rate limited to P_{pg}	Sect. 4.2
f_c	specified circuit clock rate (Hz)	specified
m	order of an RS code (no. of bits per code symbol)	Sect. 3.2/4.3
M	total capacity of an OPOM	Sect. 4.2
n	codeword length (symbols)	Sect. 3.2/4.3
N	natural length of Reed-Solomon codes (= 2^m - 1)	Sect. 3.1/5.2
N_{blk}	no. of bits of a data block	Sect. 4.2
N_{Dmax}	no. of decoders that can simultaneously operate	Sect. 4.3
$N_{D\text{-}A}$	no. of RS decoders fabricated in A_{pg}	Sect. 4.3
$N_{D\text{-}D}$	no. of RS decoders needed by N_{in} in t_a	Sect. 4.3
$N_{D\text{-}P}$	no. of RS decoders operates in A_{pg} at f_c	Sect. 4.3
N_{in}	no. of bits per data page (no. of input channels)	Sect. 4.1
P_b	raw (input) BER	Sect. 4.1
P_e	desirable output BER	Sect. 3.2/4.3
P_{idec}	product of information density and effective capacity	Sect. 4.2
P_{pg}	total power dissipation in area A_{pg}	Sect. 4.1
P_{RSde}	power dissipation per RS decoder at f_c	Sect. 2.3/4.2
P_s	raw symbol bit-error rate	Sect. 4.3
r	code rate	Sect. 3.2/4.3
r_q	required code rate	Sect. 4.3
t	no. of symbols that can be corrected per codeword	Sect. 3.2/4.3
t_a	optical memory access time per data page	Sect. 4.1

Before analyzing the two scenarios, we calculate the numbers of logical gates and transistors of an RS decoder that implements the transform decoding algorithm (TDA) based on the formulas of the six implementations listed in Table 5.2, Section 5.3. Table 4.3 lists the notation, the description, and assumed number of transistors in logical gates used in the TDA RS decoders, and these parameters are used in Sections 5.2 and 5.3.

Table 4.3 Notation, description, and their value for implementations of RS decoder.

symbol	definition or description (trs. = transistors)	no. of trs.
X_1	no. of trs. of a type-1 FFM (using 1-D systolic array)	(Sect. 5.1)
x_{21sw}	no. of trs. of a CMOS 2-to-1 switch (using C-switches)	6
x_{22sw}	no. of trs. of a CMOS 2-to-2 switch (using C-switches)	10
x_{add}	no. of trs. of a CMOS single-bit full adder	24
x_{and}	no. of trs. of a CMOS 2-input AND gate	4
X_{c2}	no. of trs. of a type-c2 FFM (compound-gate, fixed P, no C)	(Sect. 5.1)
X_{c3}	no. of trs. of a type-c3 FFM (compound-gate, fixed P/B, no C)	(Sect. 5.1)
x_d	no. of trs. of a CMOS single-bit delay cell	4
x_{dff}	no. of trs. of a CMOS D-Flip-Flop	16
x_{inv}	no. of trs. of a CMOS inverter	2
x_{nor}	no. of trs. of a CMOS 2-input NOR gate	4
x_{rsl}	no. of trs. of a CMOS reset-set latch (= $2 \cdot X_{nor}$)	8
X_{RSde}	no. of CMOS trs. required to realize an (n, n-2t) RS decoder	(Sect. 5.2)
X_{s2}	no. of trs. of a type-s2 FFM (2-D systolic array, fixed P)	(Sect. 5.1)
X_{s3}	no. of trs. of a type-s3 FFM (2-D systolic array, fixed P/B)	(Sect. 5.1)
x_{xor}	no. of trs. of a CMOS 2-input exclusive-OR gate	10

For a TDA RS decoder with a known number of logic gates, the chip area, power dissipation, and the maximum clock frequency of the decoder circuit are estimated by using the modified SUSPENS model described in Section 2.3. The VLSI parameters vary with parallelism of the implemented decoders, and thus are not specified until Chapter 6. Table 4.4 tabulates the notation and their description for the use of the model discussed in Section 2.3 and Chapter 6. Note that when estimating the chip area, power dissipation, and clock frequency of RS decoder circuits in various fabricated sizes, a simple ideal scaling rule is used (Section 2.3). In order to estimate precisely, however, the simple rule needs modifications when the minimum feature size is scaled down to the submicron domain. However, these further modifications are outside the scope of this research.

Table 4.4 Notation and Definition used in VLSI simulations.

symbol	definition and description	symbol	definition and description
A_c	chip area of a circuit of N_g logic gates	l_{av}	average interconnection length in actual units
$C_{clkwire}$	total capacitance of the global clock wires	L_{clk}	no. of clock driven transistors in a logic gate
C_{int}	wiring capacitance per unit length	N_{buf}	no. of logic gates in buffers
C_{out}	total capacitance of an output pin	N_g	no. of logic gates in a circuit
$C_{totalclk}$	total capacitance of the clock distribution	N_{logic}	no. of logic gates for decoding processes
C_{tr}	capacitance of a minimum-size CMOS transistor	N_p	no. of I/O pins of a circuit
D_c	chip dimension	n_w	number of wiring layers
d_g	logic gate dimension in actual unit (e.g., micron)	p	Rent's constant for on-chip interconnection length calculation
$d_{gintlim}$	d_g limited to interconnection capacity	P_{buf}	power dissipation of buffers
d_{gtrlim}	d_g limited to transistor packing density	P_c	total power dissipation
e_w	wiring efficiency	P_{clk}	power dissipation of clock dist.
F	VLSI minimum feature size	P_{io}	power dissipation of I/O
f_c	circuit clock frequency	P_{logic}	power dissipation of logic part
$f_{c,max}$	maximum clock frequency	p_w	wiring pitch
f_d	duty factor	$\overline{R}$	average wire length in units of gate pitch
f_g	average fan-out of a logic gate	R_{int}	wiring resistance per unit length
f_{ld}	logic depth	S	scaling factor
k_{driver}	clock driver ratio	T_g	average gate delay
k_g	proportionality constant between gate area	v_c	light speed
k_{tr}	W/L ratio of min.-size transistor	V_{DD}	supply voltage

Chapter 5

Transform Decoding Algorithm of Reed-Solomon Codes

The most complicated function in the smart-pixel interface of OPOMs is the decoding of retrieved codewords. The decoding scheme adopted here is the transform decoding algorithm (TDA) which is suitable for the VLSI implementation discussed in Section 3.2. This algorithm has been implemented using a linear pipeline structure [12], and is implemented using 2-D parallel pipeline structures in this thesis. Before we describe the implementation of the transform decoding algorithm, we first describe the finite field multiplier (FFM), which is a main component of the RS decoder.

5.1 Finite Field Multiplier

We have chosen Reed-Solomon (RS) codes for error-control in this thesis because they are capable of random/burst error correction. A key element of RS codes is the finite field multiplier. This section discusses three possible ways to implement the FFM. The first two implementations perform a multiplication and an addition of three finite field elements for any irreducible polynomials [59]. The first method implements a serial FFM using a linear systolic array, while the second method uses a 2-D systolic array. The third implementation performs the finite field multiplication with a fixed primitive polynomial which is implemented using VLSI compound circuits. In the following sections, 'multiplier' is used for 'finite field multiplier', and a multiplier for two binary numbers will be called 'binary multiplier'.

5.1.1 Formulas for Finite Field Addition and Multiplication

A finite field multiplication is much like polynomial multiplication except that FFM is perform modulo (p, $P(x)$), where p is the characteristic and $P(x)$ is an irreducible polynomial of the finite field $GF(p^m)$. Let A, B, C, and $T \in GF(p^m)$, and $T = A \cdot B + C$. In order to perform the product-sum operation, the polynomial expression of these elements are used and given by

$$A = a_{m-1}\alpha^{m-1} + \cdots + a_1\alpha^1 + a_0\alpha^0,$$

$$B = b_{m-1}\alpha^{m-1} + \cdots + b_1\alpha^1 + b_0\alpha^0,$$

$$C = c_{m-1}\alpha^{m-1} + \cdots + c_1\alpha^1 + c_0\alpha^0, \text{ and}$$

$$T = t_{m-1}\alpha^{m-1} + \cdots + t_1\alpha^1 + t_0\alpha^0,$$

where α is a root of $P(x) = x^m + p_{m-1}x^{m-1} + \cdots + p_1x^1 + p_0x^0$, and a_i, b_i, c_i, and $t_i \in GF(p)$, i.e., a_i, b_i, c_i, and $t_i = \{0, 1, \ldots, p - 1\}$. Since the finite field of the characteristic p of 2 is used in most practical applications, the prime field $GF(2)$ and its extended fields $GF(2^m)$ are used in the following discussion.

Addition of two finite field elements is performed as a regular addition of two polynomials, and then each coefficient of the sum carries out a modulo-p (or mod-p) operation. For $p = 2$, the addition of two $GF(2)$ elements is equivalent to an exclusive-OR (XOR) and the addition '+' is the same as the subtraction '-'. Therefore,

$$A \pm B = \sum_{i=0}^{m-1} \left(a_i \oplus b_i\right) \cdot \alpha^i, \tag{5.1}$$

where $\oplus$ denotes the mod-2 addition or XOR, and it is equivalent to the plus sign '+'.

To simplify the following discussion of FFMs, let $C = 0$. Hence,

$$T = A \cdot B$$

$$= \sum_{i=0}^{m-1} \left(\sum_{k=0}^{m-1} a_i^{(k)} b_k \right) \cdot \alpha^i, \tag{5.2.a}$$

where $a_0^{(k)} = a_{m-1}^{(k-1)} \cdot p_0$, and

$$a_i^{(k)} = a_{i-1}^{(k-1)} + a_{m-1}^{(k-1)} \cdot p_i, \text{ for } 1 \le k, i \le m\text{-}1. \tag{5.2.b}$$

The coefficients of the product T are then expressed as

$$t_i = a_i^{(0)} b_0 + a_i^{(1)} b_1 + \cdots + a_i^{(m-1)} b_{m-1}. \tag{5.3}$$

Another way to compute an FFM is to perform a regular polynomial multiplication of A and B, and then substitute α^j, for $i \ge m$ with $\alpha^m = p_{m-1}\alpha^{m-1} + \cdots + p_1\alpha^1 + p_0\alpha^0$ since $P(\alpha) = 0$. Hence, m binary linear functions are obtained

$$t_i = f_i(a_0, \cdots, a_{m-1}, b_0, \cdots, b_{m-1}, p_0, \cdots, p_{m-1}), \text{ for } 0 \le i \le m\text{-}1. \tag{5.4}$$

Note that the multiple functions can be simplified using a single function with the input sequence circularly shifted by a position for each t [60].

5.1.2 Implementations of the Finite Field Multiplier and Adder

The finite field multiplier has been implemented by a look-up table, a systolic array [59], and by VLSI programmable logic arrays [10]. Multiplications in RS codes can be simplified by using different bases, such as the normal basis [60] or the dual basis [61]. In this section, the three implementations of the multiplier use the standard bases [47]. Simplified versions with constant operands, such as $P(x)$ and/or B, are also presented, and will be used in the following implementations. Because the implementation of the compound-circuit multiplier with a fixed $P(x)$ depends on the number and position of non-zero coefficients of $P(x)$, an example of an $m = 4$ FFM is used for simplicity.

The first implementation of FFM uses a linear systolic array with serial inputs and outputs. Figure 5.1 shows both the linear systolic multiplier for $GF(2^m)$ and the design of a unit cell. It needs m cells and 2 flip-flops to compute an FFM. Note that 'FF' denotes a flip-flop here and in the following figures. The result of $T = A \cdot B + C$ is serially output at the top port of the rightmost cell, as shown in Fig. 5.1 (a). Two control signals, Start and End, are needed to specify the beginning and the end of a multiplication. After $2m$ time units from the first input bit, the first output is obtained at the right. Note that the inner loop in Fig. 5.1 (b), constructed with a flip-flop and a switch, is used to compute the intermediate term $a_i^{(k)}$ in Eq. (5.2.b). The linear systolic multiplier will be referred to as type-1 FFM in the next section. (Note that the term 'FFM' is used for both the finite field multiplication and the finite field multiplier. However, it should not cause confusion since the former definition is used only in the discussion of finite field arithmetic, and the latter is used only in the application of Reed-Solomon decoding.)

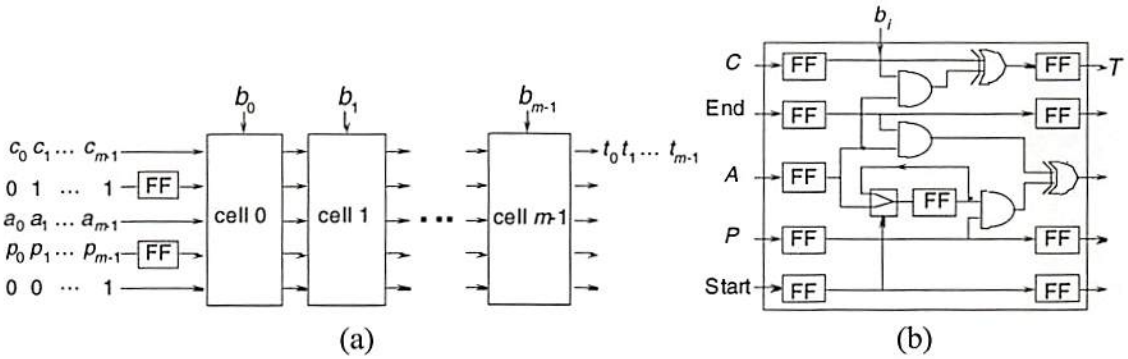


Figure 5.1 Bit-serial finite field multiplier (type-1 FFM) of order m using (a) a $1 \times m$ linear systolic array, and (b) the circuit diagram of a unit cell.

The second FFM implementation uses a 2-D systolic array so all bits are input and output in parallel. Figure 5.2 shows a 2-D systolic multiplier for the finite field $GF(2^m)$ and the designs of unit cells with different specifications. The 2-D FFM needs m^2 cells

and ($3m^2$-$2m$) unit delays for an FFM as shown in Fig. 5.2 (a). The result is obtained at the bottom ports of the cells at the rightmost column. The 2-D multiplier takes $2m$ time units from the first input to the first output, which is the same as the linear implementation. However, because the next multiplication starts right after its previous one enters the multiplier, the computation rate is high and there is no control signal needed.

To carry out the FFM with a fixed $P(x)$, the p_i lines can be either connected to V_{dd} or grounded depending on the values of p_i. The diagram of a such cell is shown in Fig. 5.2 (c), similar to a standard cell in Fig. 5.2 (b), except the flip-flops and delays of p are not needed. Further simplification is achieved by fixing b_i of another operand B, as shown in Fig. 5.2 (d). This implementation is frequently used in an RS decoder. The two simplified versions, fixing p_i and fixing both p_i and b_i, are referred as type-s2 and type-s3 FFM, respectively.

The third design carries out a set of functions of Eqs. (5.4) by using VLSI compound circuits. Figure 5.3 shows two compound circuits of the FFM for $GF(2^4)$ with the primitive polynomial $P(x) = x^4 + x + 1$. Although the multiplier contains only the basic gates (inverters, NAND's, and XOR's in the example), it is expected that compound logic circuits, which require less hardware, will be used in the future. The circuits typically need m^2 2-input ANDs, (m^2-1) 2-input XORs, and $3m$ registers, except in FFMs where $m = 8$ which has 5 terms in its primitive polynomial. When one of the operands, say A, is fixed, the circuit is reduced to $m(m$-1) XORs and $2m$ registers. In this case, the inputs are either connected to b_i or grounded depending on A. Since there are no buffers, data takes 1 time unit from input to output. The compound FFMs are referred as type-c2 and -c3 bit-parallel FFM, respectively.

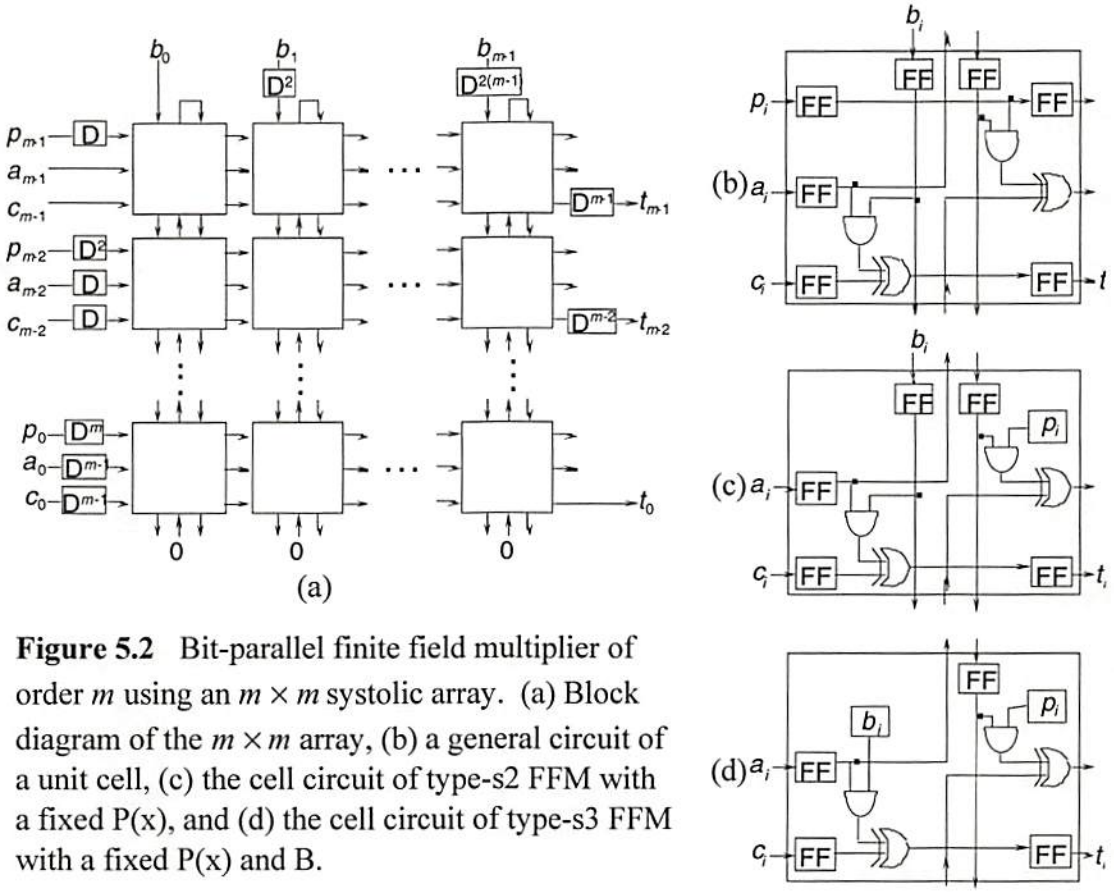


Figure 5.2 Bit-parallel finite field multiplier of order m using an $m \times m$ systolic array. (a) Block diagram of the $m \times m$ array, (b) a general circuit of a unit cell, (c) the cell circuit of type-s2 FFM with a fixed P(x), and (d) the cell circuit of type-s3 FFM with a fixed P(x) and B.

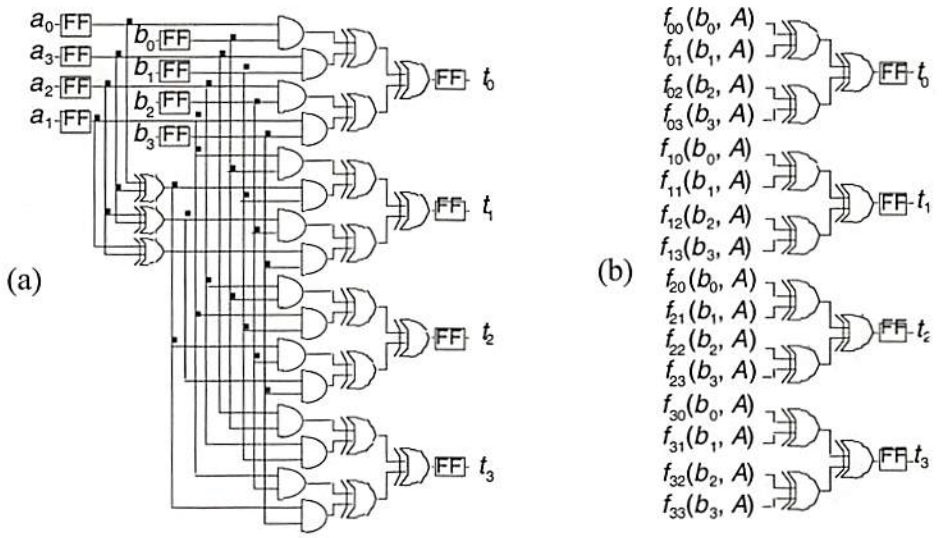


Figure 5.3 (a) Bit-parallel finite field multiplier using a compound circuit for $GF(2^4)$ with the primitive polynomial $P(x) = x^4 + x + 1$ (type-c2 FFM), and (b) the circuit for the same FFM with a fixed B (type-c3 FFM).

5.1.3 Comparisons of the Implementations of Finite Field Multiplier

Table 5.1 lists the properties of the multipliers. The first row lists the type of FFMs used in the following discussion. The second row briefly describes the structure used in the implementation. The average computation time, defined as the number of average time units needed to compute an FFM, is listed in the third row. The reciprocal of the average computation time represents the computation throughput of an FFM. Row 4 lists the total delay from the first input to the first output. Although the 2-D systolic multipliers have longer total delay, their throughput is still m times higher than the linear systolic FFM because of the short computation time per multiplication. The numbers of inputs and outputs are shown in the next row. Then, the number of logical gates per multiplier, obtained directly from observing Figs. 5.1 to 5.3, are listed in the sixth row. The last row gives a closed form formula for the number of VLSI transistors, using basic logic gates, needed to implement the FFM using Table 4.3.

The numbers of transistors needed to implement these FFM designs, as functions of m, are shown in Fig. 5.4. Note that the numbers of transistors for logical gates listed

Table 5.1 Properties of the finite field multipliers.

type	type-1		type-s2	type-s3	type-c2	type-c3
structure	1-D systolic multiplier	2-D systolic multiplier	2-D systolic multiplier - fixed P	2-D systolic multiplier - fixed P/B	compound-gate multiplier - fixed P - no C	compound-gate multiplier - fixed P/B - no C
min. avg. time per computation	m	1	1	1	1	1
delay from 1st-in to 1st-out	$2m$	$2m$ ($3m$)	$2m$ ($3m$)	$2m$ ($3m$)	2	2
numbers of I/O	4/1	$4m/m$	$3m/m$	$2m/m$	$2m/m$	m/m
gates per multiplier	$16m+2$	$2m(7m-1)$	$\frac{1}{2}m(23m-5)$	$\frac{1}{2}m(19m-3)$	$2m^2+3m-1$, ($+19$ for $m=8$)	$m(m+1)$
transistors per multiplier	$218m+36$	$166m^2-8m$	$124m^2-10m$	$98m^2-6m$	$14m^2+54m-10$ ($+190$ for $m=8$)	$10m^2+26m$

in Table 4.3 come from some common commercial CMOS libraries. Notice also that type-c2 and -c3 FFMs use fewer transistors than type-s2 and -s3 FFMs, and fewer transistors than type-1 FFMs for some m. The reason is that the systolic (linear and 2-D) FFMs require buffers or flip-flops between unit cells to synchronize the decoding operations. The linear and 2-D systolic structures have their most efficiency when all the operands, A, B, and C, arrive simultaneously. However, some situations, e.g., B is pre-loaded and C arrives later than A, neither require nor prefer synchronization. On the other hand, the compound FFMs, as shown in Fig. 5.3, work better when operand A arrives earlier or is pre-loaded. Therefore, different implementations of FFM may be used to reduce the hardware and increase performance.

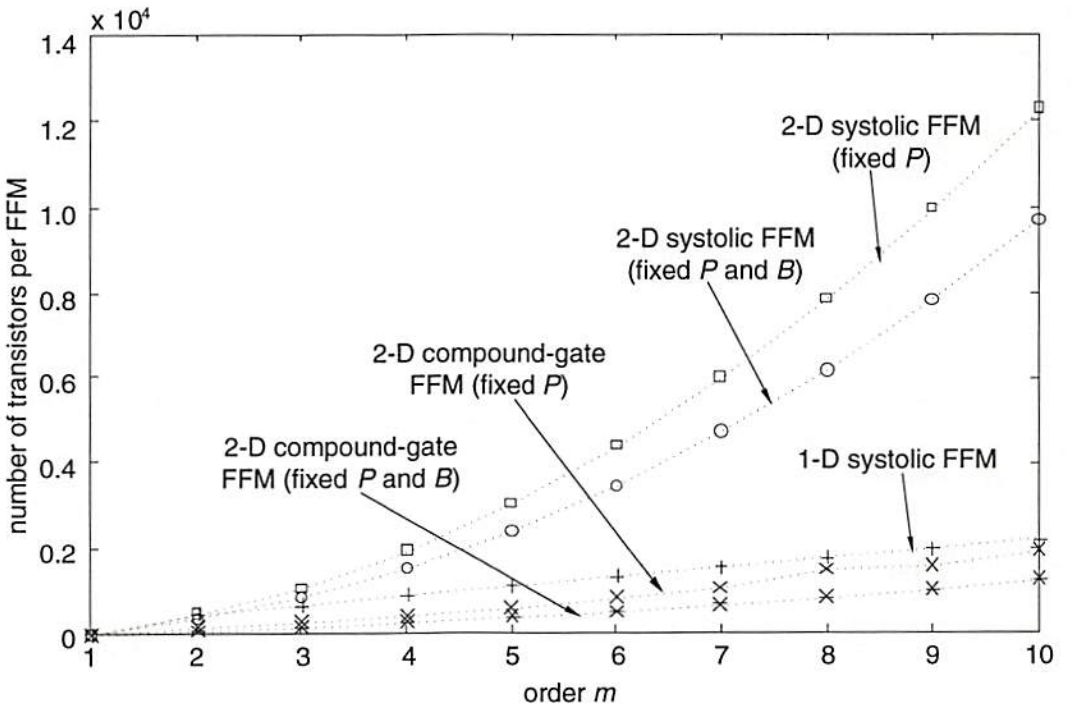


Figure 5.4 The number of CMOS transistors for the implementations of the finite field multiplier of order m.

The advantages of compound FFMs are that they use minimum hardware. The drawbacks are that; unlike the systolic FFMs using pipeline processes, some input gates need to be carefully designed because of large fan-out, likewise, the complexity of the design depends on the number of terms in the primitive polynomial $P(x)$ for $GF(2^m)$. The number of logical gates is directly related to the number of terms in $P(x)$. For example, the design of the FFM for $GF(2^8)$, with $P(x) = x^8 + x^4 + x^3 + x^2 + 1$, is much more complicated than the others which normally have three terms in their $P(x)$'s. Notice also that the computation rates of compound FFMs are limited because of m or $\log_2 m$ gate delays between the I/O, compared with two gate delays for each multiplication in systolic FFMs.

5.2 Implementations of RS Decoder Using the Transform Decoding Algorithm

The transform decoding algorithm (TDA), discussed in Section 3.2, was implemented using a pipeline structure in which code symbols are serially input and output [12]. We extended this implementation so that code symbols are input and output in parallel. The serial and parallel implementations can be devised with bit-serial and bit-parallel FFMs. This section discusses these combinations in detail.

5.2.1 Symbol-Serial Pipeline RS Decoders

Some modifications on the decoding equations of the TDA scheme are needed in order to implement the RS decoder. This occurs in four steps. The **first step** to decode the RS codes is to compute the syndromes from a received codeword (or code pattern) as in Eq. (3.4). To realize a symbol-serial decoding process, Eq. (3.4) is rewritten in an iterative form as

$$S_j = \left(\cdots\left(\left(r_{n-1}\alpha^j + r_{n-2}\right)\alpha^j + \cdots + r_1\right)\alpha^j + r_0\right), \text{ for } 1 \le j \le 2t, \tag{5.5}$$

and its implementation is shown in Fig. 5.5. The high-order code symbol enters the circuit first, and each input symbol r_i is distributed to $2t$ cells simultaneously. Each cell computes a term in a pair of parentheses of Eq. (5.5) in a delay, called a symbol delay, which is needed to perform an addition and a multiplication. The 2×2 switch in each cell is initially set to a straight connection, and the temporal result is stored in Y_i registers. After (n - 1) symbol delays, the switches change into a cross connection so the syndrome symbols appear at Z registers while the Y registers are reset. Then the switch is switched back to the straight state, and the syndromes are sequentially shifted to the right, and the syndrome module starts processing the next incoming codeword.

In the **second step**, each set of syndromes is associated with an error pattern recognized by the decoder, and the error pattern is obtained by solving an error-locator polynomial from the syndromes. The modified Euclid's algorithm, as shown in Eq. (3.7), is one way to find the error-locator polynomial. As shown in Fig. 5.6, this algorithm is implemented by $2t$ processing cells which are serially connected. The input polynomials are x^{2t} and the syndrome polynomial, $S(x)$. Since the degree of $S(x)$ is at most $2t$-1 and each cell reduces the degree by 1 on either of two input polynomials, in the worst case it takes $2t$ cells to reduce the degree of $R_i(x)$ less than t where the processes stop. Let the output polynomial be expressed as

$$\lambda(x) = \lambda_0 x^t + \lambda_1 x^{t-1} + \cdots + \lambda_t,$$

then the error-locator polynomial $\sigma(x)$ is obtained by dividing the coefficients of $\lambda(x)$ by the non-zero leading coefficient as $\sigma_i = \lambda_i / \lambda_0$, for $1 \le i \le t$, assuming $\lambda_0 \ne 0$.

In the **third step**, instead of finding the roots of $\sigma(x)$, the TDA calculates an error sequence using Eq. (3.9), an iterative equation which comes from $\sigma(x)$ and the

syndromes. Figure 5.7 shows an implementation of Eq. (3.9) in which t cells iteratively compute the error sequence. Initially, the switch at the left is open and the 1×2 switch in each cell is switched to the upper output. After the first $2t$ elements, equal to the syndromes, enter the module, the 2-to-1 switches are routed to the lower output and the left switch is closed. Then the following elements in an error sequence are sequentially calculated as Eq. (3.9).

In the **fourth and final step**, the computation of an error pattern, Eq. (3.10), is completed by inversely transforming the error sequence using a similar implementation as used in syndrome computations. Eq. (3.10) is first multiplied by a constant finite element α^N, and becomes

$$e_i = \sum_{j=0}^{N-1} E_j \alpha^{(N-j)i}, \text{ for } 0 \le i \le N\text{-}1. \tag{5.6}$$

Note that $\alpha^N = \alpha^0 = 1$ and $E_N = E_0$. According to step 3, E_1 arrives first, then E_2 ,and so on. Therefore, Eq. (5.6) is rewritten in an iterative form as

$$e_i = \left(\cdots\left(\left(E_1 \alpha^i + E_2\right)\alpha^i + \cdots + E_{N-1}\right)\alpha^i + E_0\right), \text{ for } 0 \le i \le N\text{-}1. \tag{5.7}$$

A similar implementation to realize Eq. (5.7) is shown in Fig. 5.8. Since there are only k code symbols carrying the information in systematic encoding (Section 3.1), a total of k cells are used in the 1-D implementation. After N symbols input to the module, an error pattern associated with the received codeword is obtained.

Finally, the estimated codeword is obtained by adding the error pattern e_i to the received codeword stored in a buffer as Eq. (3.11). This only needs an XOR (for bit-serial case) or m XORs (for bit-parallel case) following the inverse transform of the error sequence.

There are two first-in-first-out (FIFO) buffers in the RS decoder for the computed syndromes and the received codewords. One buffer, called the syndrome buffer, needs

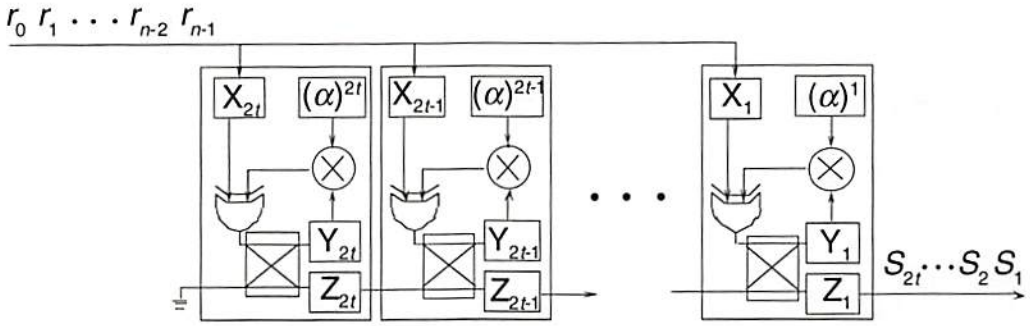


Figure 5.5 Syndrome computation for the symbol-serial RS decoder.

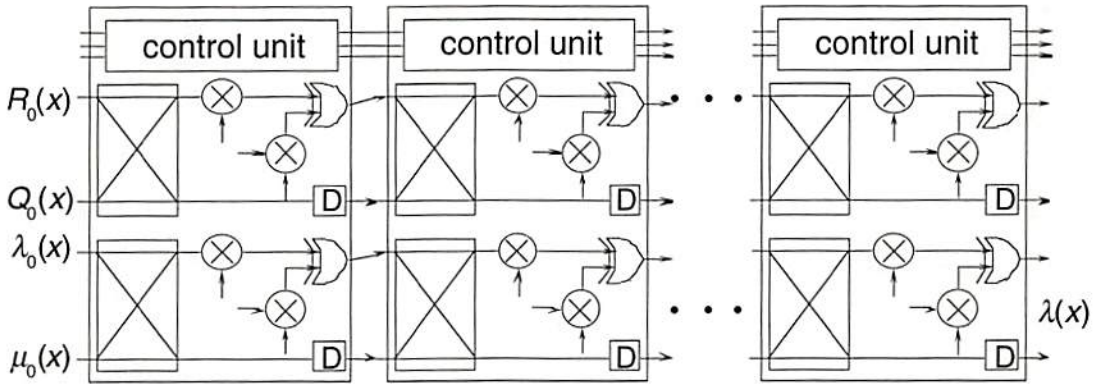


Figure 5.6 Modified Euclid's algorithm for the symbol-serial RS decoder.

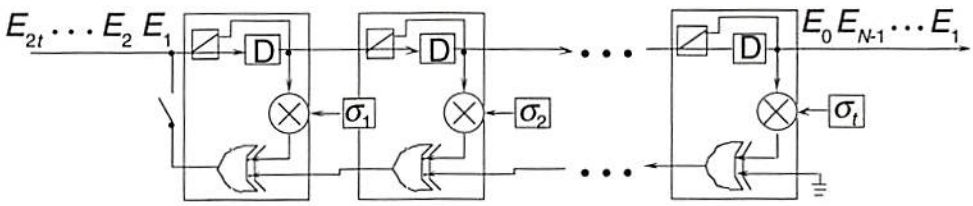


Figure 5.7 Transform of the error pattern for the symbol-serial RS decoder.

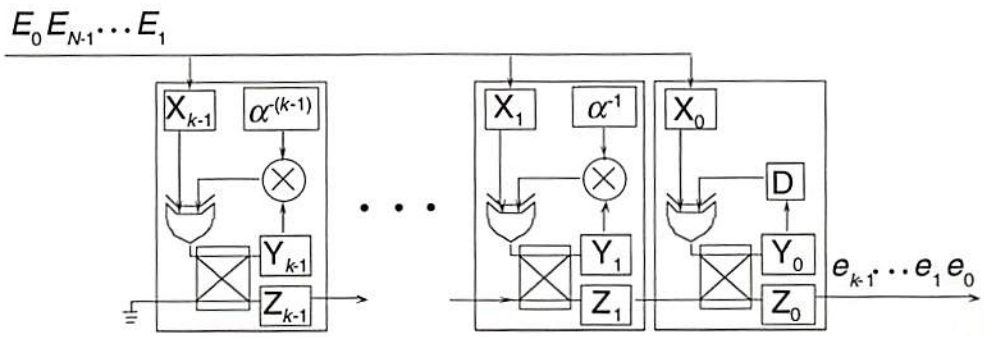


Figure 5.8 Inverse transform of the error sequence for the symbol-serial RS decoder.

$2t$ symbol registers or $2mt$ bit registers for an error-locator polynomial to be computed. Note that a symbol register contains m bit registers. The other buffer, called the codeword buffer, has a length of three codewords for reasons: (i) the syndrome module sequentially reads n symbols (assume that n is in the same order as N), (ii) the modified Euclid's algorithm needs $2t$ symbol delays and the transform of an error pattern needs N-$2t$ time delays, and (iii) the inverse transform needs N symbol delays. Therefore, there are three codewords being processed in each time unit, and the codeword buffer has a capacity of $3k$ symbols or $3mk$ bits.

5.2.1.1 Bit-Serial/Symbol-Serial (BSSS) Reed-Solomon Decoder

When the symbol-serial RS decoder employs the bit-serial FFM, i.e., type-1 FFM, every symbol and every bit enters the decoder in a series. This design is thus called the bit-serial/symbol-serial or BSSS RS decoder. In the BSSS RS decoder, there is one input port and one output port.

Implementation of the BSSS RS decoder is straightforward because *two* elements in a cell, the multiplier and mod-2 adder, are replaced by *one* type-1 FFM (Figs. 5.5 - 5.8). Each register contains m flip-flops connected in a row which store m bits of a symbol, and each link carries a single data bit. The syndrome buffer stores $2t$ syndromes or $2mt$ bits, and the codeword buffer has a capacity of 3 k-symbol codewords which is equal to $3mk$ bits. Note that all the unit registers are serially connected.

5.2.1.2 Bit-Parallel/Symbol-Serial (BPSS) Reed-Solomon Decoder

The symbol-serial RS decoder is called bit-parallel/symbol-serial (or BPSS) RS decoder when the bit-parallel (systolic or compound-circuit) FFM is used. There are m

inputs and m outputs to the BPSS decoder because the m bits of a code symbol arrive and leave in parallel.

When 2-D systolic FFMs are used, the symbol-serial decoder is denoted as a BPSS-S decoder. It is called BPSS-C when compound-circuit multipliers are used. Although, a BPSS-C decoder needs fewer transistors than a BPSS-S decoder, we will discuss and compare two implementations: (i) RS decoders using linear and 2-D FFMs and (ii) RS decoders using different 2-D FFMs. In both BPSS implementations, the modules of the syndrome computation (Fig. 5.5) and the inverse transform of the error sequence (Fig. 5.8) use type-x3 FFMs (type-x3 represents either type-s3 or type-c3 FFMs) because one of the operands α^i is always pre-determined and can be stored in the multipliers. The modules of the modified Euclid's algorithm, the polynomial normalization, and the transform of the error pattern require type-x2 FFMs (type-x2 represents either type-s2 or type-c2 FFMs) due to two varying operands. Each link, buffer, and switch operates on a symbol of bits at a time. Also, each switch block, link, and symbol register consists of m switches, parallel channels, and unit registers connected in parallel, respectively. The BPSS decoder requires the same capacity of syndrome and codeword buffers as the BSSS decoder does. However, the registers are arranged as m parallel linked buffers, and each link contains $2t$ and $3k$ unit registers connected in sequence for the syndrome and codeword buffers, respectively.

5.2.2 Symbol-Parallel Pipeline RS Decoders

This section presents a pipeline RS decoder which receives and outputs all the symbols of a codeword simultaneously. This design is expanded from the symbol-serial pipeline structure, and it will be referred to as the symbol-parallel RS decoder. Two

combinations of the symbol-parallel decoder devised with serial and parallel FFMs will be described.

In the syndrome computation module, because all the n code symbols arrive at the decoder simultaneously, Eq. (3.4) is written in a vector-matrix multiplication

$$\left(S_1, S_2, \cdots, S_{2t}\right) = \left(r_0, r_1, \cdots, r_{n-1}\right) \cdot \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \left(\alpha^2\right) & \left(\alpha^2\right)^2 & \cdots & \left(\alpha^2\right)^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \left(\alpha^{2t}\right) & \left(\alpha^{2t}\right)^2 & \cdots & \left(\alpha^{2t}\right)^{n-1} \end{bmatrix}^T . \qquad (5.8)$$

The superscript T denotes the transpose of a matrix, so the syndromes are calculated in parallel. The vector-matrix multiplication is achieved by a group of $2t$ FFMs. Each FFM group is followed by an n-input mod-2 adder, as shown in Fig. 5.9. The n symbols of a received codeword are simultaneously multiplied by the finite field constants α^i from the matrix of Eq. (5.8). The product terms are then summed by an n-input mod-2 adder. Each n-input mod-2 adder can be implemented by (n-1) 2-input mod-2 adders (or 2-input XORs) which are organized in a binary tree structure.

The modified Euclid's algorithm is performed by $2t$ cells as shown in Fig. 5.10. Each cell contains a control unit and $3t + 1$ sub-cells ($2t$ sub-cells for Eqs. (3.7.a-b) and (t+1) sub-cells for Eqs. (3.7.c-d)). In each sub-cell, the 1-to-2 switches are controlled by difference of the degrees of R and Q polynomials, and the 2-to-1 switch by σ_i as in Eq. (3.7.f). The error polynomial $\lambda(x)$ is obtained at the output of the $2t$-th cell. A polynomial normalization module, following the modified Euclid algorithm cells, finds the nonzero leading coefficient, inverts it, and then multiplies the inverse to all the other coefficients to yield the error-locator polynomial $\sigma(x)$.

As shown in Fig. 5.11, the transform of error pattern, Eq. (3.9), was implemented by a pipeline with (N - $2t$) stages. This pipeline accepts the input of t syndrome symbols while it shifts the other t symbols directly to the error-sequence buffer at the

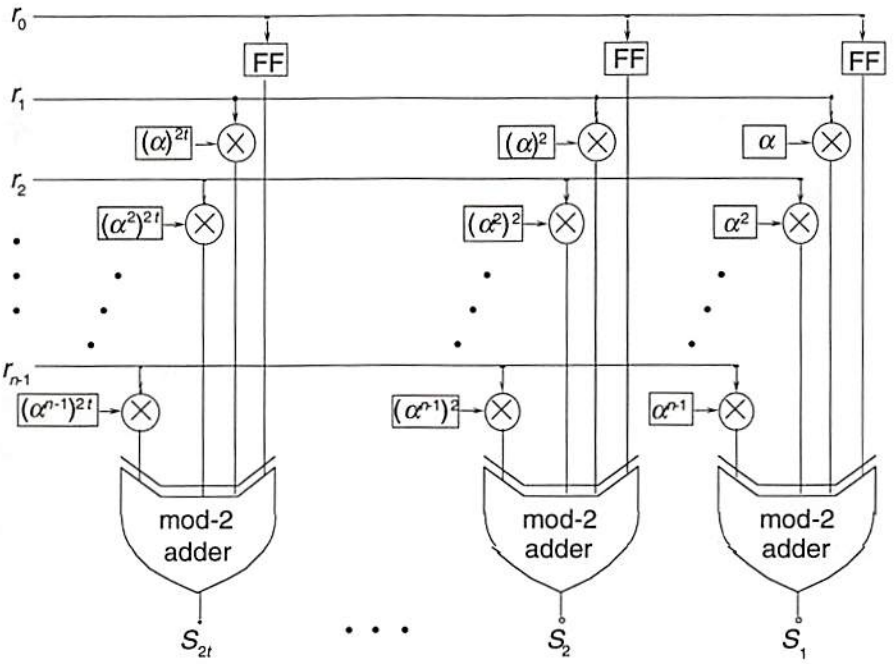


Figure 5.9 Syndrome computation for the symbol-parallel RS decoder.

right of Fig. 5.11. The jth stage calculates E_{2t+j} using a parallel vector-matrix multiplier, as previously described, and outputs E_{2t+j} at the left-most output. E_{2t+j} and other (t - 1) symbols, $E_{2t+j-1} \ldots E_{t+j+1}$, are input to the $(j + 1)$th stage to compute the next element in the error sequence. Note that, again, the first $2t$ elements in an E_j sequence are equal to the $2t$ syndrome symbols. After $(N - 2t)$ stages, the error sequence appears simultaneously at the bottom of the pipeline.

The error sequence obtained from the last module is then inversely transformed using a similar implementation to the first step. Again, Eq. (3.10) is rewritten in a vector-matrix multiplication, similar to Eq. (5.8), as

$$(e_0, e_1, \cdots, e_{N-1}) = (E_0, E_1, \cdots, E_{N-1}) \cdot \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & (\alpha^{N-1}) & (\alpha^{N-1})^2 & \cdots & (\alpha^{N-1})^{N-1} \\ 1 & (\alpha^{N-2}) & (\alpha^{N-2})^2 & \cdots & (\alpha^{N-2})^{N-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha & (\alpha)^2 & \cdots & (\alpha)^{N-1} \end{bmatrix}^T . \quad (5.9)$$

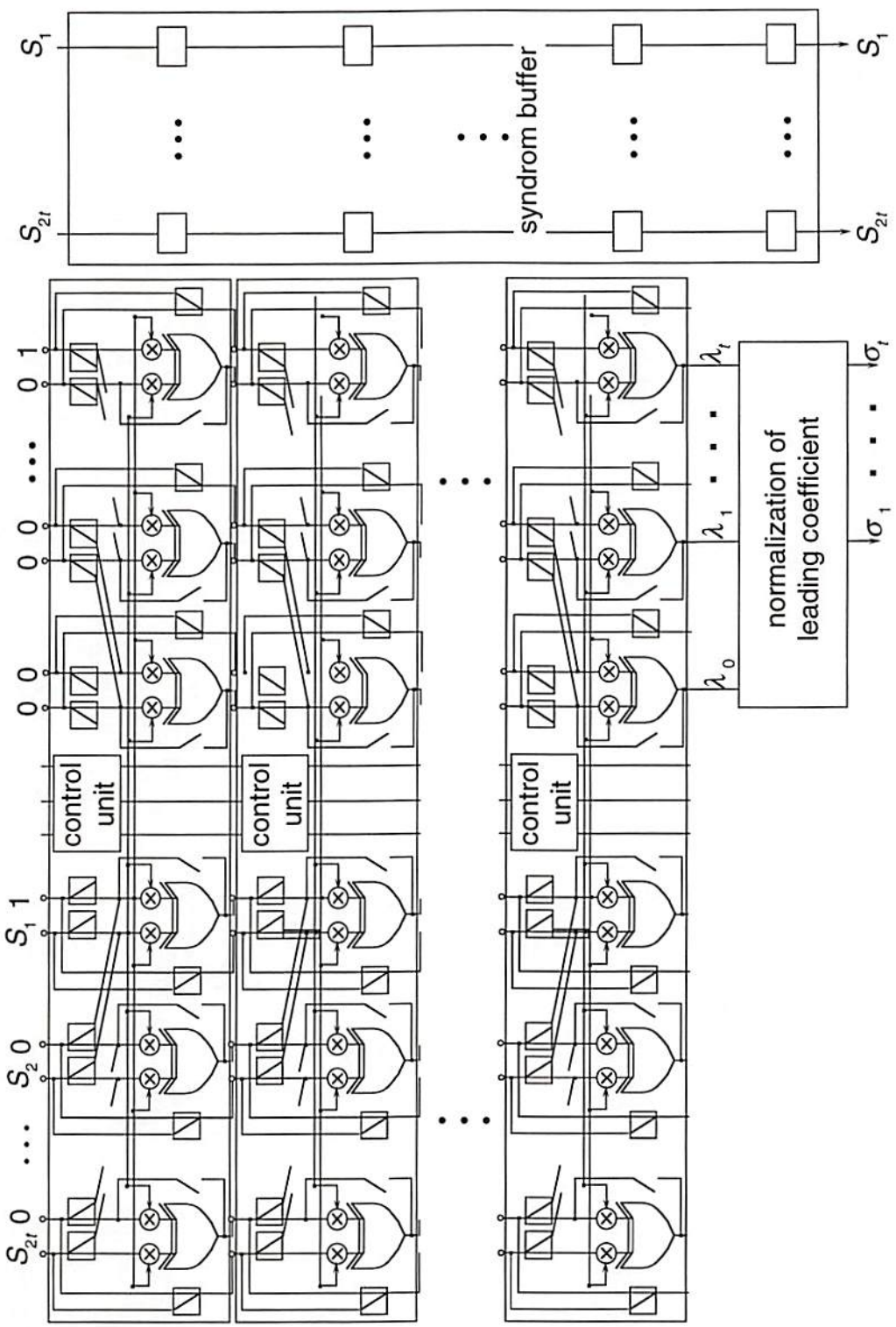


Figure 5.10 Modified Euclid's algorithm and the syndrome buffer for the symbol-parallel RS decoder.

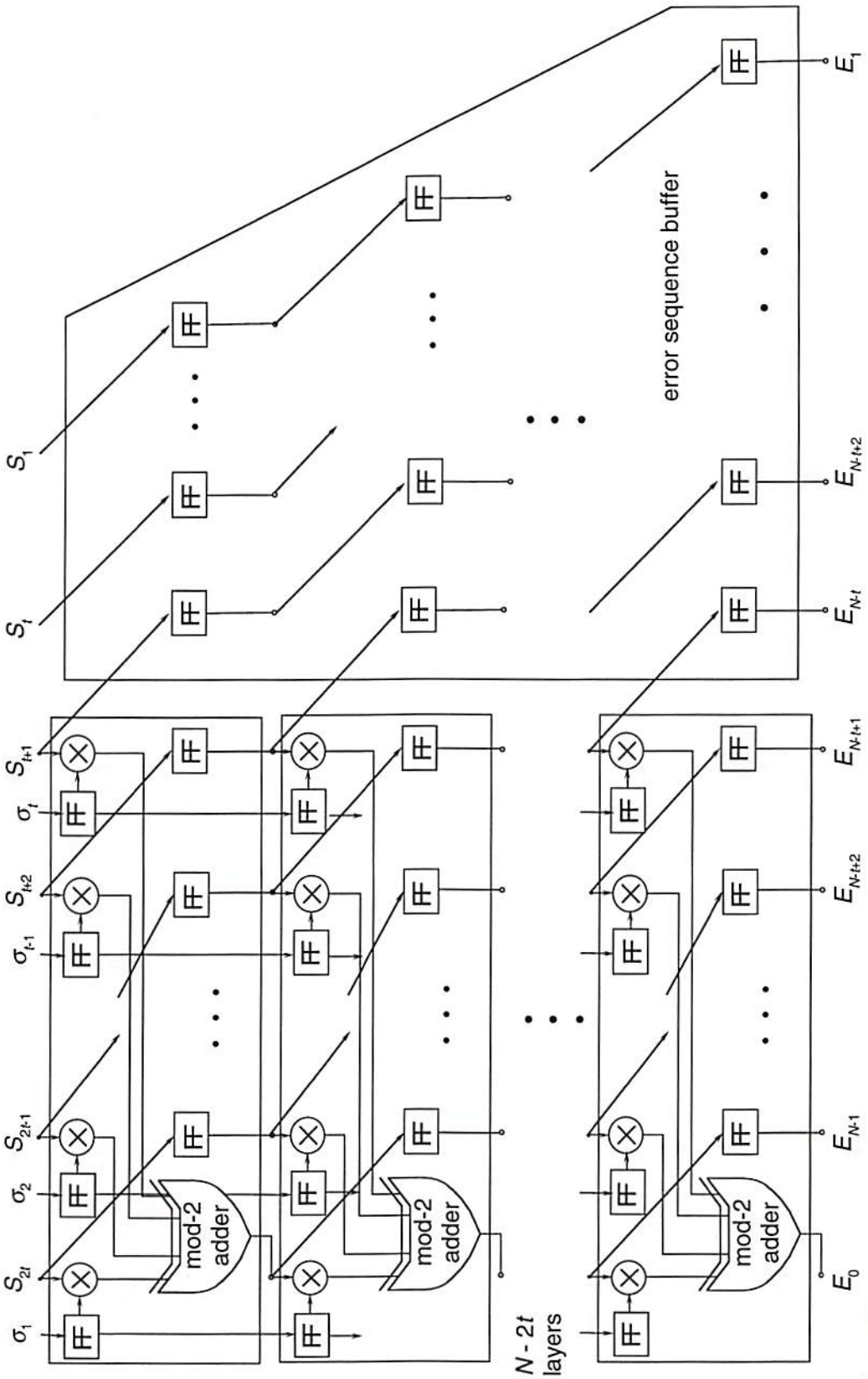


Figure 5.11 Transform of the error pattern and the error-sequence buffer for the symbol-parallel RS decoder.

Each element e_i in the error pattern is computed as an inner product of the E vector and a row of the matrix in Eq. (5.9). This inner product is implemented using an array of N multipliers and an N-input mod-2 adder, as shown in Fig. 5.12. Since there are k error-pattern elements needed to correct the k information symbols, k inner products are necessary (not all N inner products are needed).

Finally, the error pattern is added to the k symbols of the received codeword to correct errors. This addition requires either an XOR for a bit-serial case or m XORs for a bit-parallel case.

The symbol-parallel implementation of the RS decoder needs three FIFO buffers to store received codewords and intermediate results, as shown in Fig. 3.4. As shown at

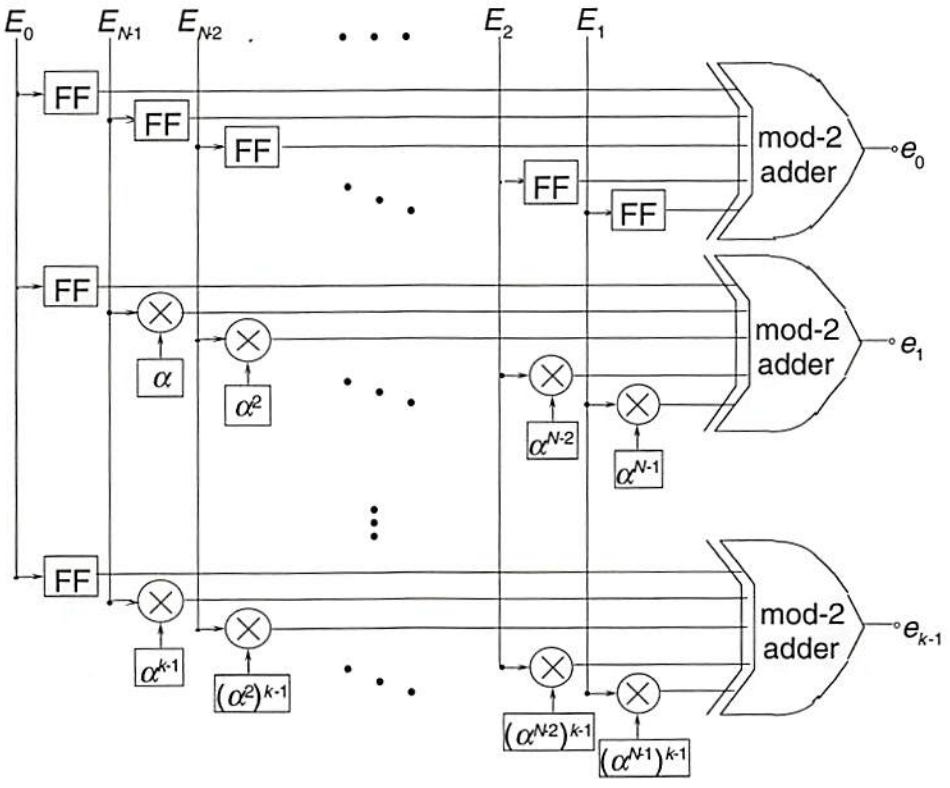


Figure 5.12 Inverse transform of the error sequence for the symbol-parallel RS decoder.

the right of Fig. 5.10, the first (syndrome) buffer has a parallel input of $2t$ syndrome symbols and contains $(2t+1) \times 2t$ symbol registers. This is because there are $2t$ stages in the modified Euclid's module and one stage in the polynomial normalization module. The second buffer is the error sequence buffer, as shown in Fig. 5.11, which contains $(N - 2t)$ stages of shift registers. The first stage has $t+1$ symbol registers which receive the $t+1$ lower-order syndrome symbols simultaneously. The number of symbol registers is increased by one as each new error-sequence element is calculated from the left pipeline in Fig. 5.11. In consequence, the error-sequence buffer needs $(N + 1)(N - 2t)/2$ symbol registers. The final buffer stores the received codewords as they await their associated error pattern. In the symbol-parallel implementation, this codeword buffer needs a capacity of $N+3$ codewords because there are a total number of $N+3$ pipeline stages in the four decoding steps discussed earlier (1 for syndrome computation, $2t+1$ for modified Euclid's algorithm and polynomial normalization, $N - 2t$ for transform of error sequence, and 1 for the inverse transform). Note that only k code symbols in a codeword need to be stored; therefore, the codeword buffer contains $k(N - 2t)$ symbol registers.

5.2.2.1 Bit-Serial/Symbol-Parallel (BSSP) Reed-Solomon Decoder

The bit-serial/symbol-parallel (or BSSP) RS decoder is an implementation of the symbol-parallel design which is devised with the bit-serial FFMs. The BSSP decoder contains n input channels and k output channels, and each I/O channel serially inputs or outputs m bits of a symbol.

The BSSP decoder is implemented using the diagrams shown in Figs. 5.9 - 12 in which the multipliers are replaced by the type-1 FFM. In the modules of the syndrome computation, the transform of the error pattern, and the inverse transform of the error

sequence, the addition in type-1 FFM is not used because the summation of the product terms is carried out by a mod-2 adder with multiple, parallel input. Also, each symbol register in the three buffers (syndrome, error-sequence and codeword) consists of m bit registers that are serially connected.

5.2.2.2 Bit-Parallel/Symbol-Parallel (BPSP) Reed-Solomon Decoder

A fully parallel RS decoder is constructed by combining the bit-parallel FFMs and the symbol-parallel RS implementation. It is called bit-parallel/symbol-parallel, or BPSP RS decoder. In addition, it will be denoted by the BPSP-S decoder when the bit-parallel systolic FFMs are used, and the BPSP-C decoder when using the compound-circuit FFM. Note that a BPSP decoder contains mn input and mk output channels.

The modules of the syndrome computation and the inverse transform of the error sequence of the BPSP implementation use type-x3 FFMs because one operand is fixed. On the other hands, the modules of the modified Euclid's algorithm and the transform of the error pattern use type-x2 FFMs. All the links, switches and registers contain m bit registers connected in parallel in order to support the parallel processes. The syndrome, error-sequence, and received codeword buffers contain $k(N+3)$ symbol registers, the same number of registers as for the BSSP decoder, while the bit registers of a symbol are connected in parallel.

5.3 Summary and Comparisons of the TDA RS Decoder

So far we have described six implementations of the TDA RS decoder, and they are the BSSS, BPSS-S, BPSS-C, BSSP, BPSP-S, and BPSP-C decoders. The first and the third letters in the acronym stand for 'bit' and 'symbol', respectively; 'S'/'P' in the second

and the fourth letters stand for 'serial'/'parallel'; and the appending 'S'/'C' stand for the systolic/compound-circuit FFMs. Table 5.2 summarizes these implementations in terms of the hardware properties, including the I/O format, the codeword delay, total delay between the input and output of the first symbol of a codeword, numbers of I/O, and numbers of logic gates per RS decoder.

The first three rows of Table 5.2 list the acronym, the I/O formats of an input codeword, and the types of multipliers used in the implementations.

The codeword delay in the fourth row is defined as the longest delay needed by the slowest module or cell in the implementation for a codeword. The reciprocal of the codeword delay is proportional to data rate of the output interface which affects the total data throughput. In the BSSS and BPSS designs, the slowest module is the inverse transform of the error sequence (**ITES**) which needs N ($=2^m-1$) symbol delays because of the sequential input of N symbols. Unfortunately, the delay of this module does not decrease for any shortened codes because an error sequence always contains N symbols. Since each type-1 FFM consists of m cells and each cell needs a unit delay, a symbol delay of the BSSS implementation corresponds to m unit delays. In consequence, the BSSS needs mN unit delays to process a RS codeword. On the other hand, the bit-parallel FFMs simultaneously process m bits of a symbol, and thus need only one unit delay for each symbol, i.e., 1 symbol delay = 1 unit delay (BPSS). Therefore, the BPSS implementations which process a code symbol in parallel need N unit delays between two consecutive codewords. In the symbol-parallel implementations, the slowest modules are the polynomial normalization and/or the ITES modules. Each cell in the latter module needs N parallel FFMs followed by an N-input m-stage adder which results in about 2 symbol delays. The module of polynomial normalization needs 2 symbol delays by assuming one symbol delay to the computation of the reciprocal of a finite field element. Therefore, a codeword delay of the symbol-parallel implementation

contains two symbol delays, or corresponding to $2m$ and 2 unit units. Note that if more stages of registers are placed in the buffers of the symbol-parallel implementations, then only one symbol delay is needed between two codewords.

The delay between the first input symbol of a received codeword and the first output of its correction is listed in Row 5. This delay is the part of memory access time that is consumed by the output interface. Note that these formulas are simplified by assuming $\log_2 N \approx m$ and $\log_2 t \approx 1$. Also, note that this delay does not vary as different FFMs being used in a certain implementation because the total delay for the three systolic FFMs are the same, and so are the compound-circuit FFMs (Row 4, Table 5.1).

The following rows of Table 5.2 show the number of VLSI logical gates needed in these implementations. Since there is no simple expression for them, the numerical results using MATLAB are discussed and shown in the following figures.

Figure 5.13 shows the number of logic gates and VLSI transistors needed by these implementations of the primitive RS codes ($n = N = 2^m - 1$) to reduce the raw BER of 10^{-4} to 10^{-12} or lower. Notice that an average of 10 transistors per logical gate is required here while 6 transistors per logical gate are normally required in microprocessor chips. These implementations employ a large number of D-flip-flops for synchronization and XOR gates for finite field arithmetic, and both contain more transistors than the average (Table 4.3). This situation necessitates the individual estimates of VLSI area and power dissipation, as described in Section 2.3. Secondly, as n increases, the six lines apparently separate into two groups. The lines of the symbol-parallel implementations increase more rapidly than the symbol-serial. This seems to imply that the TDA scheme is inefficient in hardware for the symbol-parallel implementations. However, this will be discussed from another point of view in Fig. 5.15.

Table 5.2 Properties of the implementations of the RS decoder using the transform decoding algorithm.

RS decoder	BSSS	BPSS-S	BPSS-C	BSSP	BPSP-S	BPSP-C
I/O format	bit-serial symbol-serial	bit-parallel symbol-serial	bit-parallel symbol-serial	bit-serial symbol-parallel	bit-parallel symbol-parallel	bit-parallel symbol-parallel
FFM type	type-1	type-s2/s3	type-c2/c3	type-1	type-s2/s3	type-c2/c3
codeword delay	mN	N	N	$2m$	2	2
delay from 1st input to 1st output	$2m(2N+n+2)$	$2m(2N+n+2)$	$2(2N+n+2)$	$2m(N+4)+N-2t+2$	$2m(N+4)+N-2t+2$	$3N+10$
numbers of I/O	1/1	m/m	m/m	n/k	mn/mk	mn/mk
FFMs	$n+9t+2$	$n+9t+2$	$n+9t+2$	$N(n-t-1)+n(2t-1)$ $+2(5t^2+1)$	$N(k-1)+n(2t-1)+$ $10t^2+5t+2$	$N(k-1)+n(2t-1)+$ $10t^2+5t+2$
x_{dff}	$m(8n+3t+2)$	$2m(2n-2t+1)$	$2m(2n-2t+1)$	!	!!	!!!
x_{xor}	1	m	$(n+5t)(m-1)+m$	$(n-t)(N+2t+1)$ $-(N+t)$	$mN(n-t-1)+$ $2t(n+m-mt-1)$	$mN(n-t-1)+$ $2t(n+2m-mt-2)$
x_{rsl}	0	$m(2n+5t)$	0	0	$m(5t+1)$	0
x_d	$5mt$	$5mt$	$5mt$	0	0	0
x_{22sw}	$n+6t$	$m(n+4t)+2t$	$m(n+4t)+2t$	$2t$	$2t$	$2t$
x_{21sw}	t	mt	mt	$6t(3t+1)$	$6mt(3t+1)$	$6mt(3t+1)$
x_{sw}	1	m	m	$2t(3t+1)$	$2mt(3t+1)$	$2mt(3t+1)$
x_{add}	$2t$	$2t$	$2t$	$2t$	$2t$	$2t$

$! = 2nmt + 4t + m(t+1) + m(2t-1)(N-2t) + Nmk + 2mt(2t+1) + \frac{m}{2}(N+1)(N-2t) + mk(N+3)$

$!! = 2mt + m(2t-1)(N-2t) + [mN+m(k-1)] + 2mt(2t+1) + \frac{m}{2}(N+1)(N-2t) + mk(N+3)$

$!!! = 2mt + m(t-1)(N-2t) + [mN+m(k-1)] + 2mt(2t+1) + \frac{m}{2}(N+1)(N-2t) + mk(N+3) = !! - mt(N-2t)$

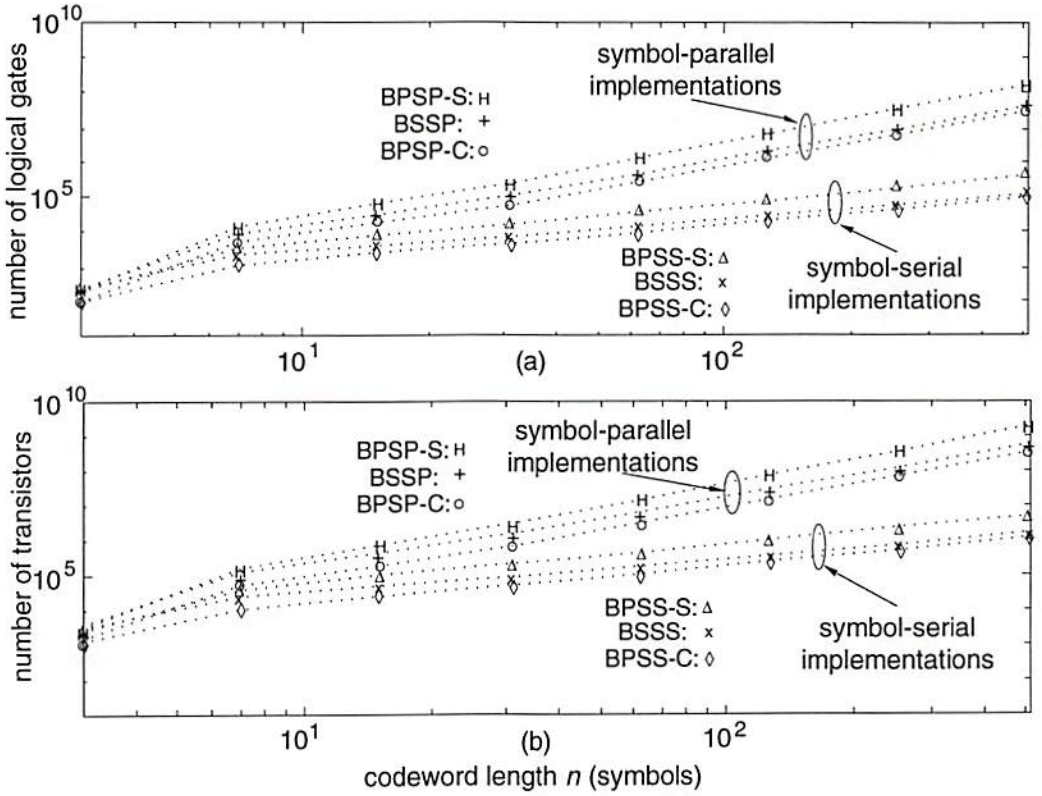


Figure 5.13 (a) The number of logical gates and (b) the number of CMOS transistors per RS decoder versus codeword length for the primitive RS codes which reduce the BER from 10^{-4} (raw BER P_b) to 10^{-12} (desirable BER P_e) or better.

The number of transistors per decoder for two families of RS codes, $m = 5$ and 8, is shown in Fig. 5.14. The selected shortened and extension codes here have the same performance as the primitive codes which are discussed previously and shown by the dotted lines. As n decreases in the shortened codes, the number of transistors per decoder only reduces slightly because the number of transistors of the ITES mainly depends on the natural length N ($= 2^m - 1$) and the ITES module needs the most transistors among these decoding modules (also refer to Section 6.1). In addition, the number of transistors of other modules depends on t and m more than on n. Therefore, we conclude that the TDA scheme is not suitable for the implementations of shortened RS codes in all the decoder designs studied.

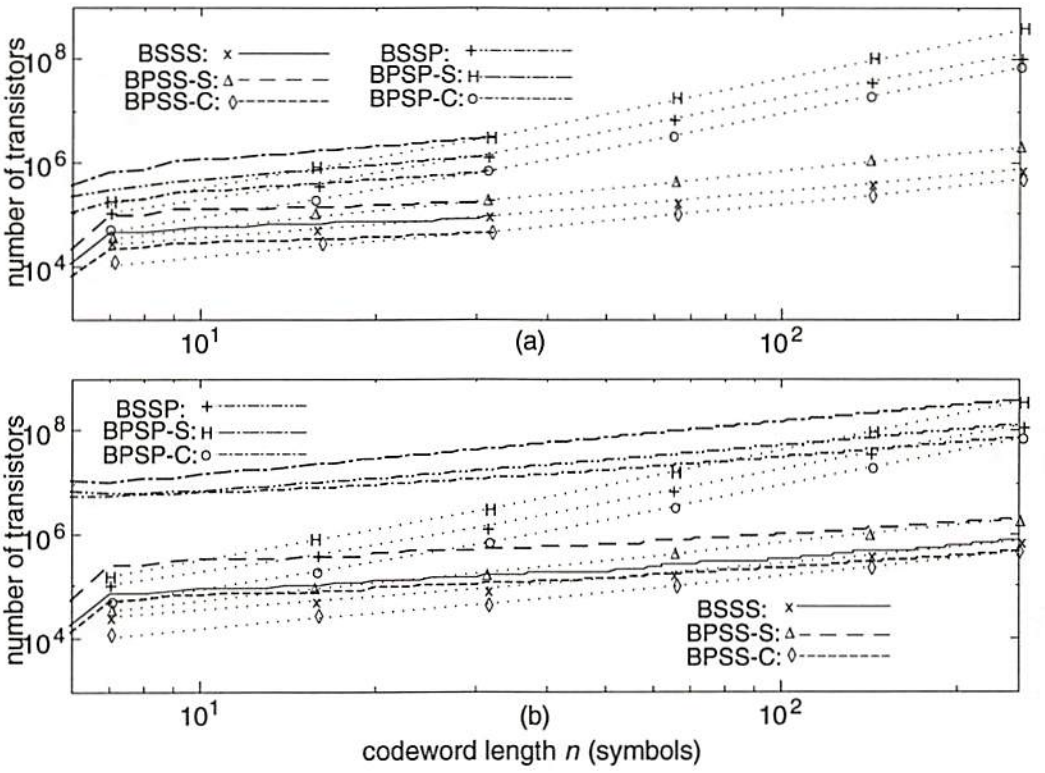


Figure 5.14 The number of CMOS transistors per RS decoder versus codeword length for the primitive RS codes (dotted lines), (a) $m = 5$ RS codes, and (b) $m = 8$ RS codes which reduce the BER from 10^{-4} to 10^{-12} or better.

In Fig. 5.15, the number of transistors of a single BPSP decoder is compared with m BSSP decoders, n BPSS decoders, and mn BSSS decoders because they all have mn input channels. It is reasonable that the BSSS design needs the most transistors and the BPSP design needs the least. mn BSSS decoders require mn sets of modules such as mn modules of the modified Euclid's algorithm which contain $2t \cdot mn$ cells. On the other hand, the module of modified Euclid's algorithm in a BPSP decoder contains $2t \cdot (3t+1)$ cells in total. Since t is usually much smaller than n, the symbol-parallel design is more efficient in hardware than the symbol-serial design. In addition, by observation, the bit-parallel implementation is also more efficient in hardware than the bit-serial. The

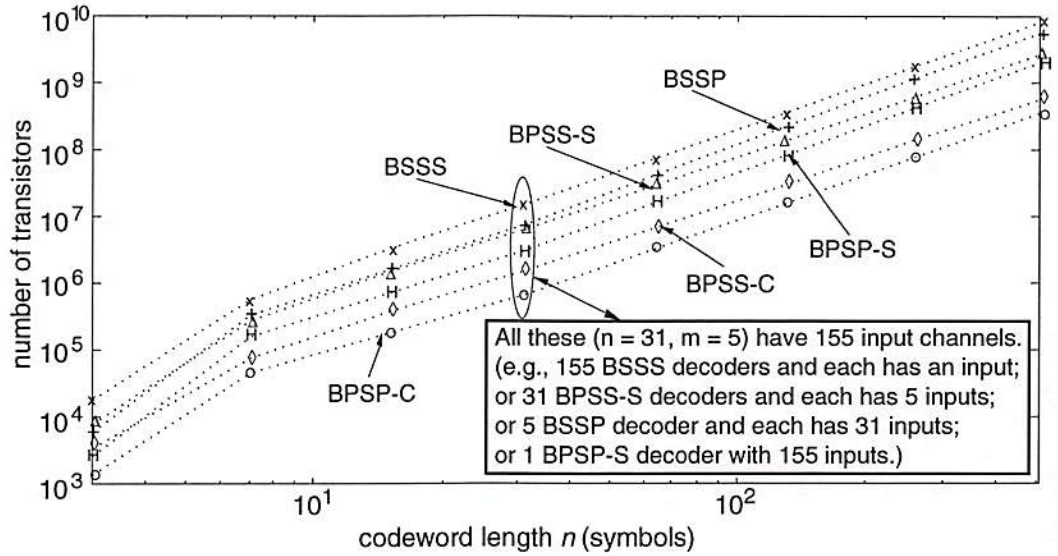


Figure 5.15 The number of transistors of decoders of the same number of input channel (for the primitive RS codes, $P_b = 10^{-4}$, $P_e = 10^{-12}$).

relation between bit and symbol parallelism is not clear, however, according to Fig. 5.15, bit-parallel is more efficient in hardware than symbol-parallel for the TDA.

The conventional parameters used here do not describe these implementations clearly, as seen in Figs. 5.13 and 5.15. Therefore, these designs need new parameters such as those defined in Chapter 4. In the next chapter, the decoder variations and the SP interface are analyzed based on these parameters.

Chapter 6

System Analysis of Decoder Implementations

The previous two chapters presented various implementations of the Reed-Solomon decoders using the transform decoding algorithm (TDA) and designs for the SP interface containing these TDA RS decoders. In this chapter, the performance of the decoders and the feasibility of the SP interfaces is analyzed using computer simulations.

Given a set of RS code parameters (m, n, k, t, r), the number of logical gates and transistors of a TDA RS decoder are calculated from Tables 4.3 and 5.2. Then, the chip area, power dissipation, and maximum clock frequency of the VLSI decoding implementations are estimated using the modified SUSPENS model discussed in Section 3.3. The parameters used in the modified SUSPENS are listed in Table 6.1, which is obtained from the current VLSI CMOS processes with proper modifications. For example, the Rent's constant p for high speed microprocessor chips is typically from 0.6 to 0.7, depending on the interconnection complexity of the circuits. Note that p is empirical. Therefore, we assumed $p = 0.6$ for the BSSS implementation because of serially connected modules and cells, and $p = 0.68$ for the BPSP-C because of the more global interconnections in the parallel decoding modules and compound-circuit FFMs. The p's of other implementations are then specified with values between these two extremes. In particular, the p of the buffers is 0.5 because D-flip-flops are sequentially connected to each other, and, therefore, only local connections are used. To simplify the analysis, we assume that the wiring pitch p_w is $3F$ in all scales, although the practical

p_w is larger (e.g., $4F$ to $6F$) in sub-micron VLSI processes. In Table 6.1, the values of the parameters marked with an asterisk are obtained by assuming 0.8-μm CMOS, and they are scaled when different CMOS feature sizes are applied (Section 3.3).

Section 6.1 describes the properties of various implementations of the TDA RS decoder in terms of the chip area, power dissipation, and maximum clock frequency. In addition, the area and power dissipation of individual modules of the decoder are shown. In Section 6.2, all the fabricated TDA decoders in a given area are assumed to operate simultaneously at the maximum allowable clock frequency limited by a given power density. The results of two parameters, defined in Section 4.2, are used to determine

Table 6.1 Parameters for the modified SUSPENS model (* for 0.8-μm CMOS).

parameters	buffers	BSSS	BPSS-S	BPSS-C	BSSP	BPSP-S	BPSP-C
p_w (μm)	$3F$	$3F$	$3F$	$3F$	$3F$	$3F$	$3F$
e_w	0.4	0.4	0.4	0.4	0.4	0.4	0.4
n_w	2	3	3	3	3	3	3
p	0.5	0.6	0.63	0.64	0.64	0.67	0.68
f_g	2	2	2	3	3	4	4
f_{ld}	2	4	4	4	m	m	m
T_g (ns)	2	2	2	2	2	2	2
R_{int} (Ω/μm)*	9×10^{-4}	9×10^{-4}	9×10^{-4}	9×10^{-4}	9×10^{-4}	9×10^{-4}	9×10^{-4}
C_{int} (fF/μm)*	0.2	0.2	0.2	0.2	0.2	0.2	0.2
k_{tr}	3	4	4	4	4	4	4
C_{ox} (fF/μm^2)*	2.3	2.3	2.3	2.3	2.3	2.3	2.3
V_{DD} (volts)*	5	5	5	5	5	5	5
f_d	0.5	0.2	0.2	0.2	0.2	0.2	0.2

the RS code and the decoder design that have the best performance limited to the given physical conditions. Section 6.3 shows the intersection of two code-dependent constraints, bit error (BEC) and code rate (CRC) constraints (Section 4.3). Then, the results of VLSI-dependent constraints, buffer length constraint (MINC) and power/area constraint (MAXC), are combined with the BEC and CRC in Section 6.4. The RS code that has the largest code rate and with the necessary data rate is then specified using the four constraints.

6.1 Performance of An Individual TDA RS Decoder

The number of logic gates and transistors needed for the TDA RS decoders was shown in Section 5.4. Here, using the modified SUSPENS VLSI simulation model and the circuit parameters, the chip (decoder) area, power dissipation of a decoder and modules, average switching energy, and the maximum clock frequency of these decoders are compared. In the following estimates, the parameters assumed include 0.25-μm minimum feature size (F), 10^{-4} raw BER (P_b), and 10^{-12} output BER (P_e), except for those listed in Table 6.1.

In Fig. 6.1, the decoder area needed for the primitive RS codes ($n = 2^m$-1) are shown by dotted lines and symbols at the discrete positions where they exist, while the other lines show the area of the TDA decoders for the shortened and extended RS codes of m = 5 and 8. Similar to Fig. 5.13, the decoder area needed for the TDA decoders separate into two groups, the symbol-serial (BSSS, BPSS-S, and BPSS-C) and the symbol-parallel (BSSP, BPSP-S, and BPSP-C), as n increases. For the primitive codes of small n, the area of the symbol-parallel decoders is an order of magnitude larger than that of the symbol-serial decoders, and it is three orders larger for large n. However, the area changes slightly for the RS codes with the same m.

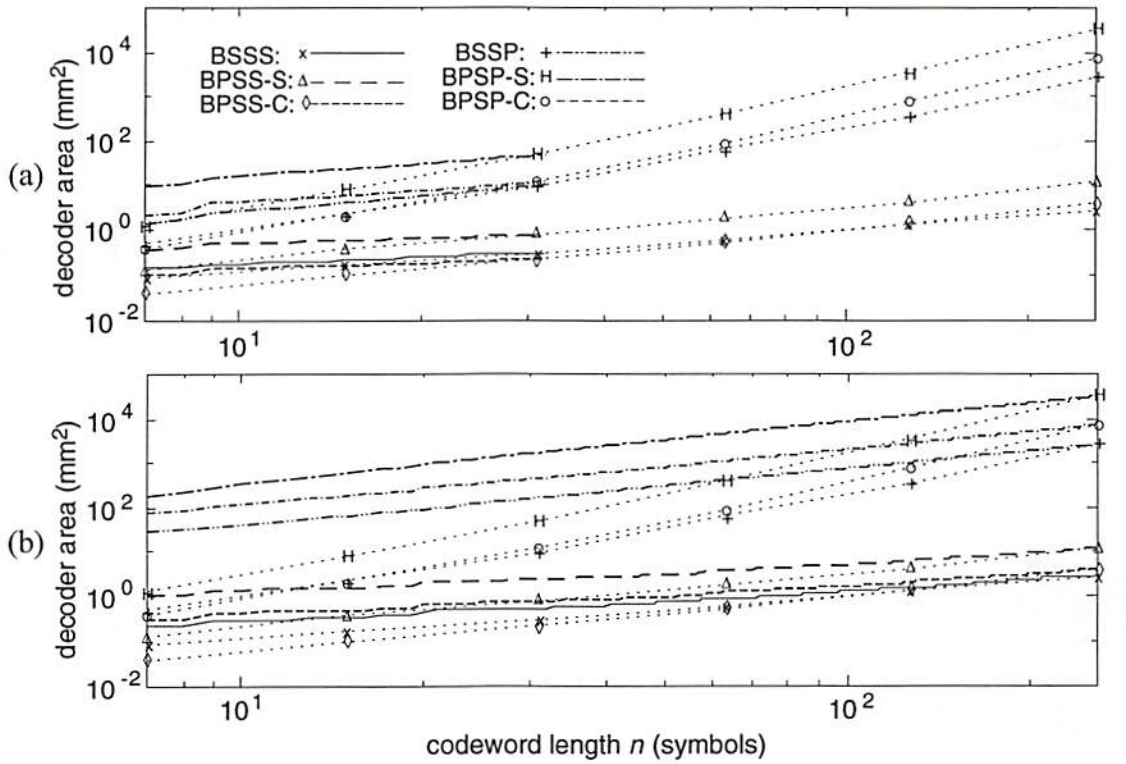


Figure 6.1 Decoder area (mm^2) for the primitive RS codes (dotted lines), (a) $m = 5$, and (b) $m = 8$ RS codes ($F = 0.25$ µm, $P_b = 10^{-4}$, $P_e = 10^{-12}$). (BSSS: Bit-Serial/Symbol-Serial; BSSP: Bit-Serial/Symbol-Parallel; BPSS-S: Bit-Parallel/Symbol-Serial (using the 2-D systolic FFMs); BPSP-S: Bit-Parallel/Symbol-Parallel (using the 2-D systolic FFMs); BPSS-C: Bit-Parallel/Symbol-Serial (using the compound-circuit FFMs); and BPSP-C: Bit-Parallel/Symbol-Parallel (using the compound-circuit FFMs))

Figure 6.2 shows the maximum clock frequency $f_{c,max}$ achieved in a decoder circuit, estimated by Eq. (2.9). Because the logic depth f_{ld} and the average gate delay T_g are identical for the symbol-serial implementations, the $f_{c,max}$ is constant ($\approx$ 400 MHz) for all n's. Note that the second and the third terms of Eq. (2.9) contribute insignificantly when the decoder area D_c is small. The $f_{c,max}$ of the symbol-parallel decoders is reduced dramatically as both m and n increase due to the effect of rapidly increased D_c to the second term.

The power dissipated by these implementations was estimated by using Eq. (2.13) and is shown in Fig. 6.3. These lines distribute similar to the lines of decoder area in

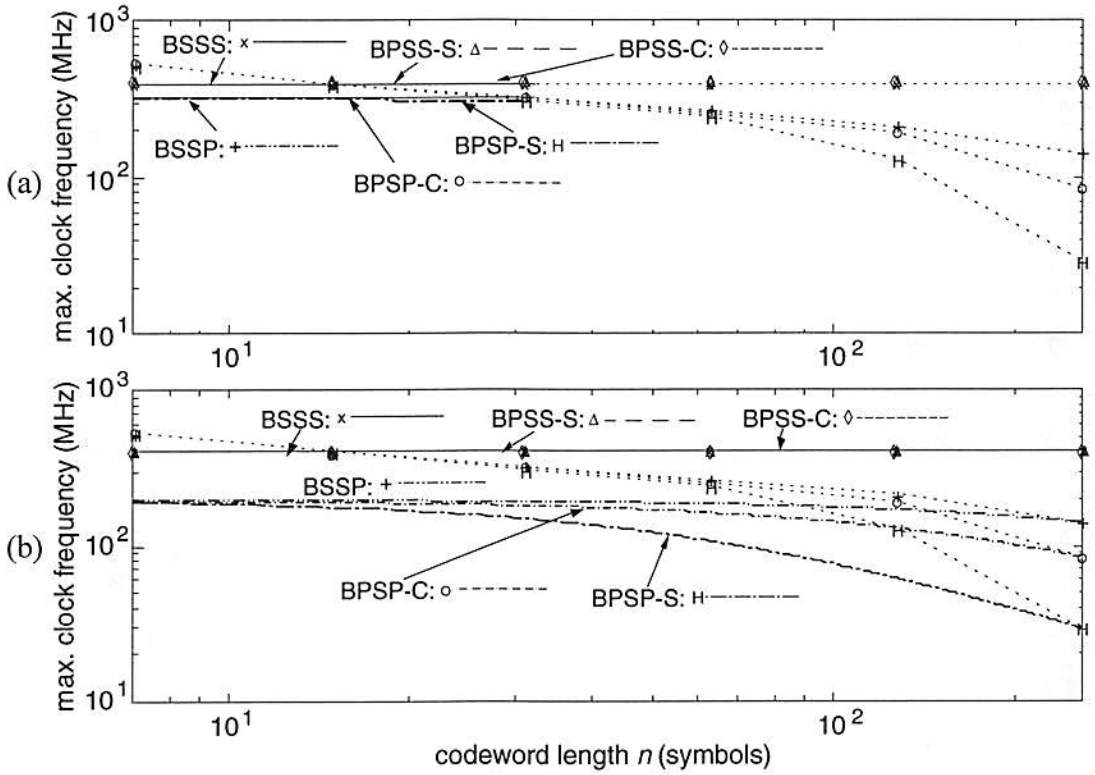


Figure 6.2 Maximum achieved clock frequency for the primitive RS codes (dotted lines), (a) $m = 5$, and (b) $m = 8$ RS codes ($F = 0.25$ μm, $P_b = 10^{-4}$, $P_e = 10^{-12}$).

Fig. 6.1. We notice that the power dissipation of the three symbol-parallel decoders and the primitive RS codes of long n's, e.g., 255, which normally yield high code rate are intolerably large.

For such large area and power dissipation of VLSI circuitry, it is interesting to show the detail of the individual modules in these decoders. Figure 6.4 shows the module area of the syndrome computation (Syn), the Euclid's algorithm (EA) (including the polynomial normalization), the transform of error pattern (TEP), the inverse transform of the error sequence (ITES), and the buffers (Buff) of the TDA decoders. In the symbol-serial implementations, the Buff and ITES modules have the largest area, shown in Fig. 6.4 (a), (c) and (e). On the other hand, in the symbol-parallel decoders, because of the long global connected wires, the ITES module dominates the decoder area.

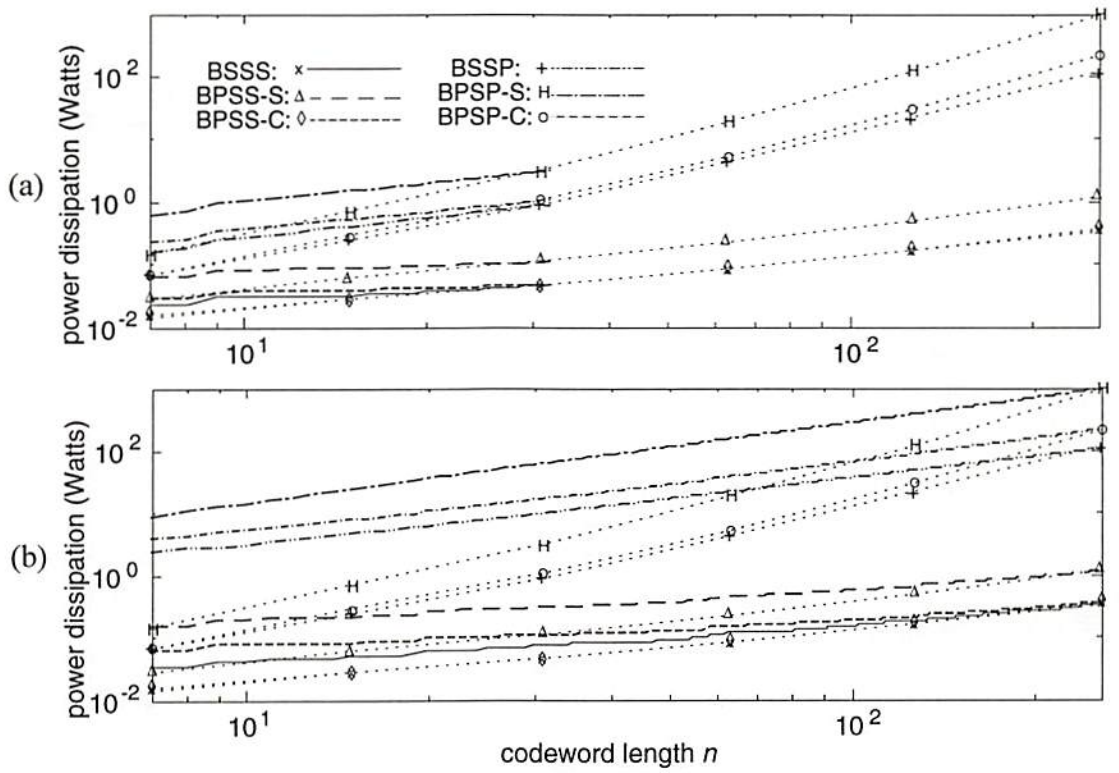


Figure 6.3 Power dissipation per decoder for the primitive RS codes (dotted lines), (a) $m = 5$, and (b) $m = 8$ RS codes ($F = 0.25$ μm, $P_b = 10^{-4}$, $P_e = 10^{-12}$).

The buffers of the BPSP-C decoder requires a smaller area compared with other types of decoders because the compound-circuit FFMs use less flip-flops for pipelined operations.

Figure 6.5 shows the estimated power dissipation of individual modules of the implementations. In the three symbol-serial decoders shown in Figs. 6.5 (a), (c) and (e), the power dissipation of the clock distribution (Clk) is the largest among these parts, and then the ITES and Buff modules. This situation is again due to the long global interconnections and pipeline nature of the TDA decoder designs. A great amount of power is dissipated when the clock signal is distributed to most logic gates and every elements in the buffers during decoding processes. For the symbol-parallel decoders, the ITES module dissipates the most power, and the power dissipation of Clk is still

significant. In conclusion, the large power dissipation in most parts of the symbol-parallel TDA decoders inhibits the use of the RS codes of long codewords, unless smaller VLSI feature size or other packaging techniques are used.

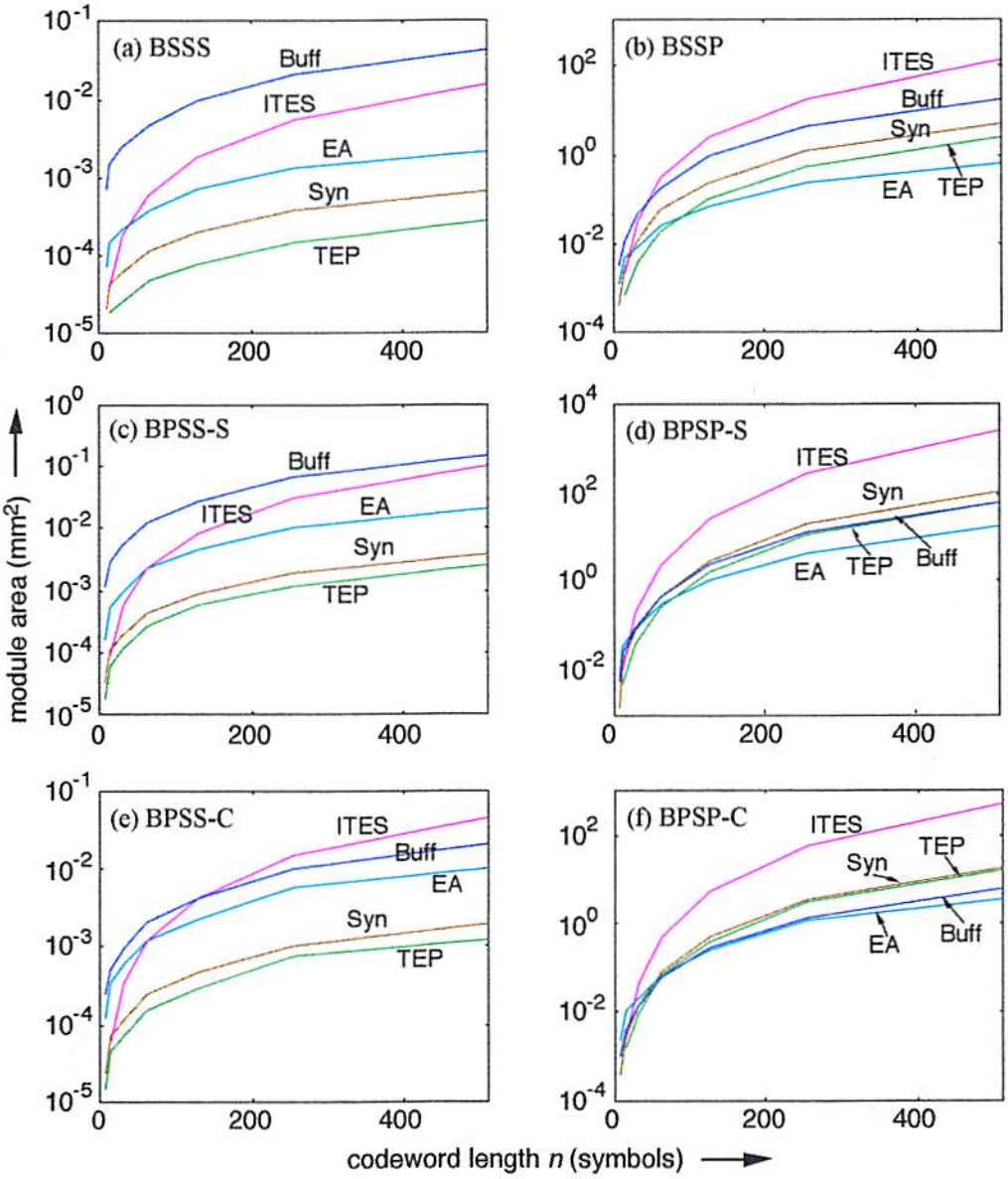


Figure 6.4 Area (mm^2) of modules of the (a) BSSS, (b) BSSP, (c) BPSS-S, (d) BPSP-S, (e) BPSS-C, and (f) BPSP-C decoders for the primitive RS codes ($F = 0.25$ μm, $P_b = 10^{-4}$, $P_e = 10^{-12}$, $n = 2^m - 1$). (Syn: syndrome computation; EA: Euclid's algorithm+polynominal normalization; TEP: transform of the error pattern; ITES: inverse transform of the error sequence; Buff: buffers)

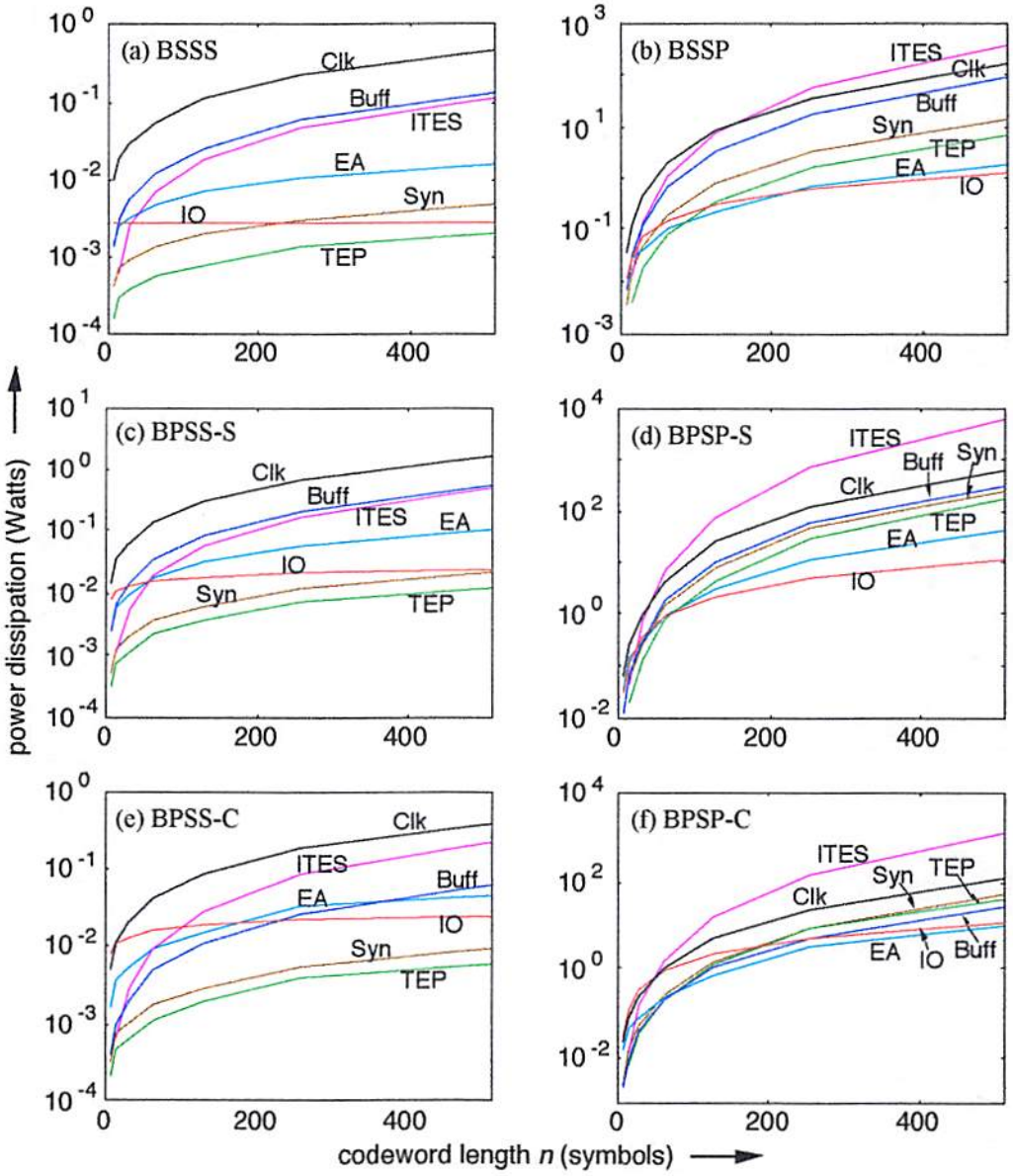


Figure 6.5 Power dissipation of modules of the (a) BSSS, (b) BSSP, (c) BPSS-S, (d) BPSP-S, (e) BPSS-C, and (f) BPSP-C decoders for the primitive RS codes ($F = 0.25$ µm, $P_b = 10^{-4}$, $P_e = 10^{-12}$, $f_c = 320$ MHz). (Clk: clock distribution)

6.2 The Optimal Implementation of TDA RS Decoder

In this section, the properties defined in Section 4.2 are used to evaluate the performance of the implementations. The same VLSI parameters are used as in the

previous section. In addition, the implementation is confined to a fixed area A_{pg} (10 cm^2) and a fixed power density P_{pg} (2 Watts/cm^2), and all the codes shown here reduce a raw BER from 10^{-4} to 10^{-12} or lower.

The first property of the TDA decoders is the input spatial channel density, d_{scin}, which is defined as the number of input channels in a unit area as Eq. (4.2). Figure 6.6 shows the d_{scin} for the primitive, $m = 5$, and $m = 8$ RS codes. Note that the $m = 2$ RS codes do not provide error correction capability of reduction the BER from 10^{-4} to 10^{-12} and thus are not shown. The RS codes with large n result in small d_{scin} due to the logarithmic increase of D_c. In Fig. 6.6 (b), the lines of the BSSP and the BPSP-C stop at $n = 128$ and the BPSP-S at 64 is because the area of a single decoder increases larger than A_{pg} (Fig. 6.1). Therefore, no BSSP, BPSP-C, or BPSP-S decoders can be implemented for those n's. The two horizontal dashed lines in Fig. 6.6 show 1-D and 2-D electrical limits given by the maximum numbers of input pins on the edge of the chip and through the chip, respectively. The 1-D electrical input is limited to is 20 channels per cm^2 (or 40 I/O channels per cm^2 in total), and the 2-D electrical limit is 50 input channels per cm^2 (or 100 I/O channels per cm^2).

Fixing the power density, the estimated input rate of a data block, f_{blk}, given by Eq. (4.3) is Shown in Fig. 6.7 (a). The zigzag lines of the symbol-parallel implementations of $m = 8$ result from the decrease of the number of fabricated decoders in a decoder array, $N_{D\text{-}A}$, by 1. As n increases for a fixed m, the area of a decoder as well as the area and power dissipation of a decoder array are increased. f_{blk} is then decreased limited by the power density P_{pg}. When the area of a decoder array exceeds the given area A_{pg}, $N_{D\text{-}A}$ is reduced by 1. Because the effect of increased n is less than the effect of decreased $N_{D\text{-}A}$, all the decoders then operate at a higher rate. The relationship between the decreased $N_{D\text{-}A}$ and abruptly increased f_{blk} is illustrated by the double-arrow lines crossing Fig. 6.7

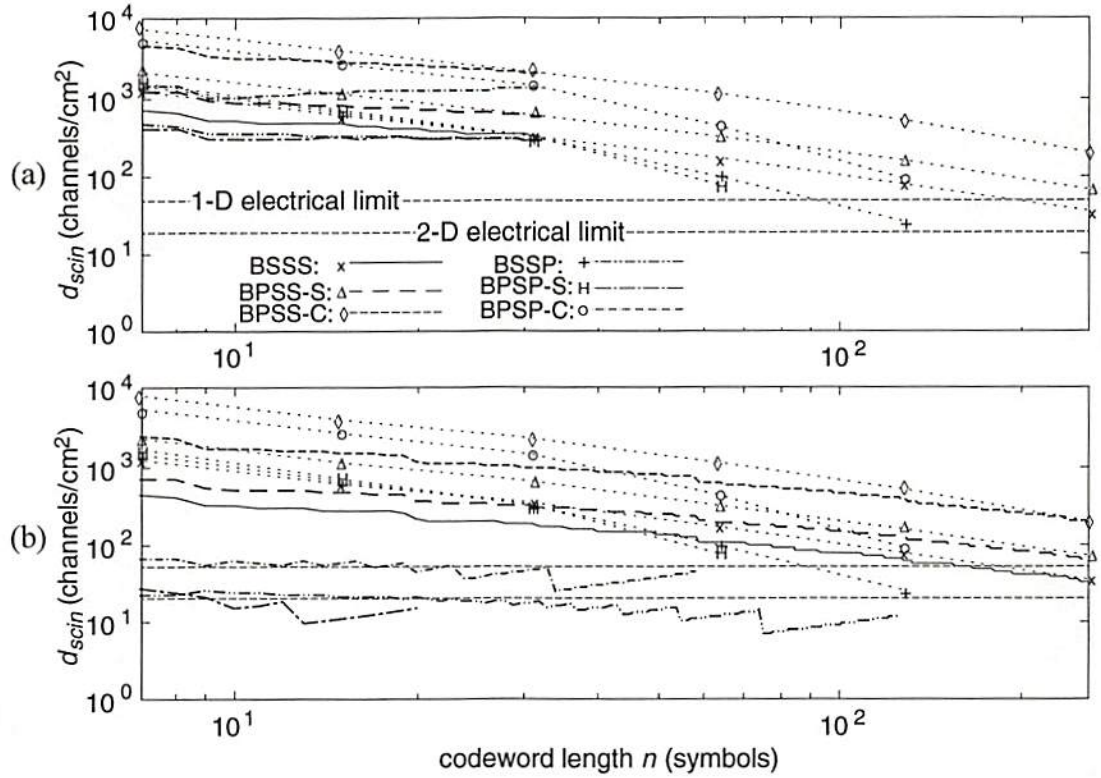


Figure 6.6 Input spatial channel density d_{scin} for the primitive RS codes (dotted lines), (a) $m = 5$, and (b) $m = 8$ RS codes.

(a) and (b). Note that this phenomenon is apparent when $N_{D\text{-}A}$ is small (≤ 10), i.e., the area of a decoder is large.

Figure 6.8 shows the data rate input to the decoder array, B_{in}, which is the product of the number of bits of a data block and the block rate, f_{blk}, as in Eq. (4.4). It is shown that B_{in} decreases as n and/or m increase. However, the B_{in} of the symbol-parallel implementations decreases slightly because the ITES module, consisting of $(N - 2t)$ cells, dominates the area and the power dissipation (Figs. 6.4 and 6.5) and $N \gg t$. Note that because the area of a decoder exceeds the given area A_{pg}, the lines of the BPSP-C, the BPSP-S, and the BSSP of $m = 8$ end at $n = 55$, 18, and 127, respectively. The dashed line at 10^{11} represents a data throughput required by the output interface input from a memory medium and will be used in Section 6.4.

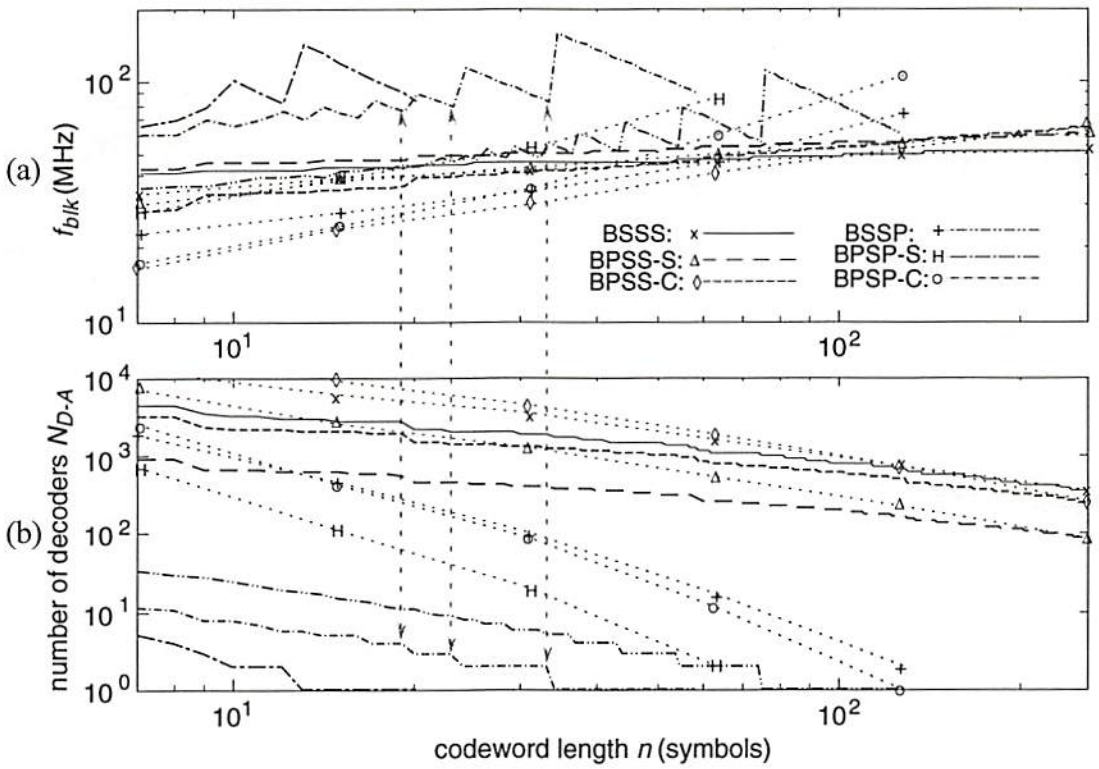


Figure 6.7 (a) Block rate f_{blk} and (b) the number of decoders fabricated in 10 cm^2 for the primitive RS codes (dotted lines) and the $m = 8$ RS codes.

The information rate at the output of the decoder array (or the aggregate output rate), B_{info}, is the product of the code rate r and B_{in}, as in Eq. (4.5). The information spatial channel density, d_{info}, defined in Eq. (4.6), is the aggregate output rate in a unit area and is shown in Fig. 6.9. Note that the zigzags in the lines of $m = 5$ and 8 implementations result from the discontinuities of r, as shown in Fig. 3.3. For the four implementations using the systolic FFMs (BSSS, BPSS-S, BSSP, and BPSP-S), the peak of d_{info} occurs at $n = 15$ or 31, and the BPSS-S has the largest d_{info} for all n and designs (for the primitive codes). On the other hand, among the four symbol-parallel implementations (BPSS-S, BPSP-S, BPSS-C, and BPSP-C), the BPSS-C and BPSP-C are able to provide better d_{info}. Both peaks of d_{info} of the two TDA decoders occur at $n = 31$, and, the BPSS-C decoder has the highest d_{info} among the six implementations. The

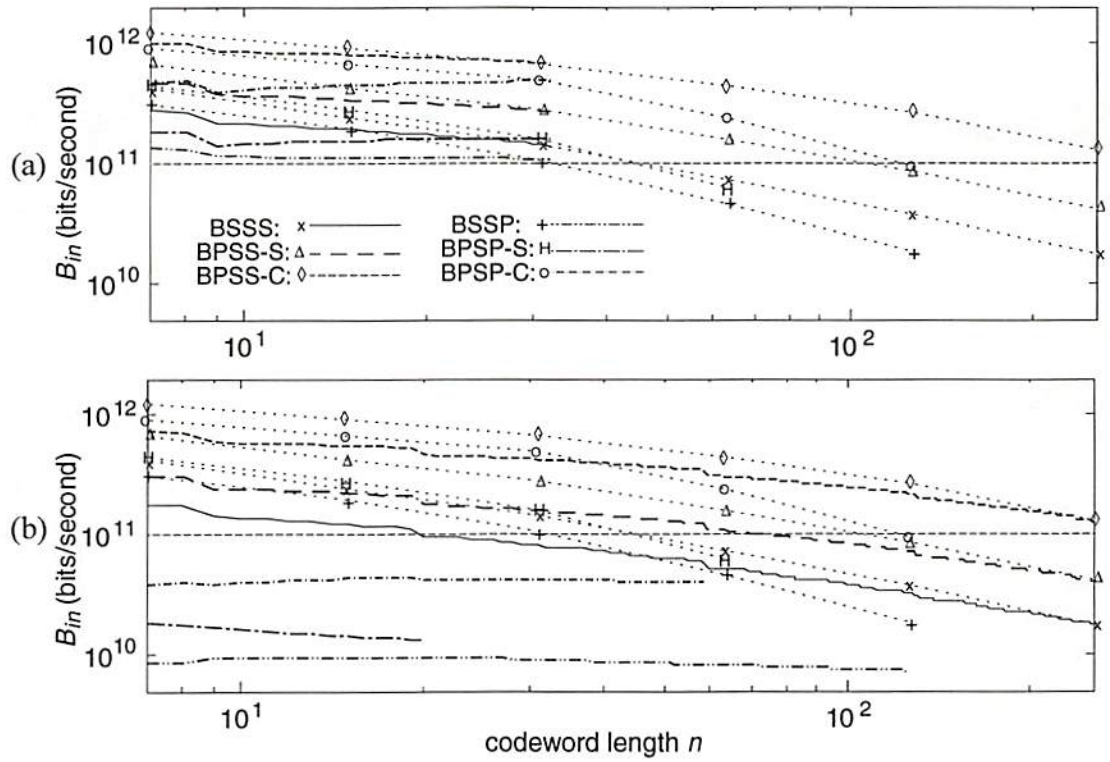


Figure 6.8 Aggregate input data rate B_{in} for the primitive RS codes (dotted lines), (a) $m = 5$, and (b) $m = 8$ RS codes.

d_{info} of the TDA decoders for $m = 5$ and 8 RS codes are also shown, and the peak d_{info} are listed in Table 6.2. In addition, the d_{info} for the codes that reduce the BER to 10^{-15} are shown. In all the cases, the highest d_{info} was obtained by the BPSS-C design at different n, but not at the primitive values, i.e., 2^m-1.

Since the objective of the TDA decoder is to improve the BER for the OPOMs, the decoder design should take into account the effective capacity of the memory, rM, where M is the total capacity of the memory. Combining d_{info} and rM yields the product of the information density and the effective capacity, P_{idec}, as defined in Eq. (4.7). In Fig. 6.10, the peaks of the normalized P_{idec}, i.e., P_{idec}/M, all occur at $n = 32$ that is the best RS code ($n = 32$, $k = 24$, $t = 4$, and $r = 0.75$) for the TDA decoders. Table 6.3 lists the peak values of P_{idec}/M of the implementations for the $m = 5$, 6 and 8 RS

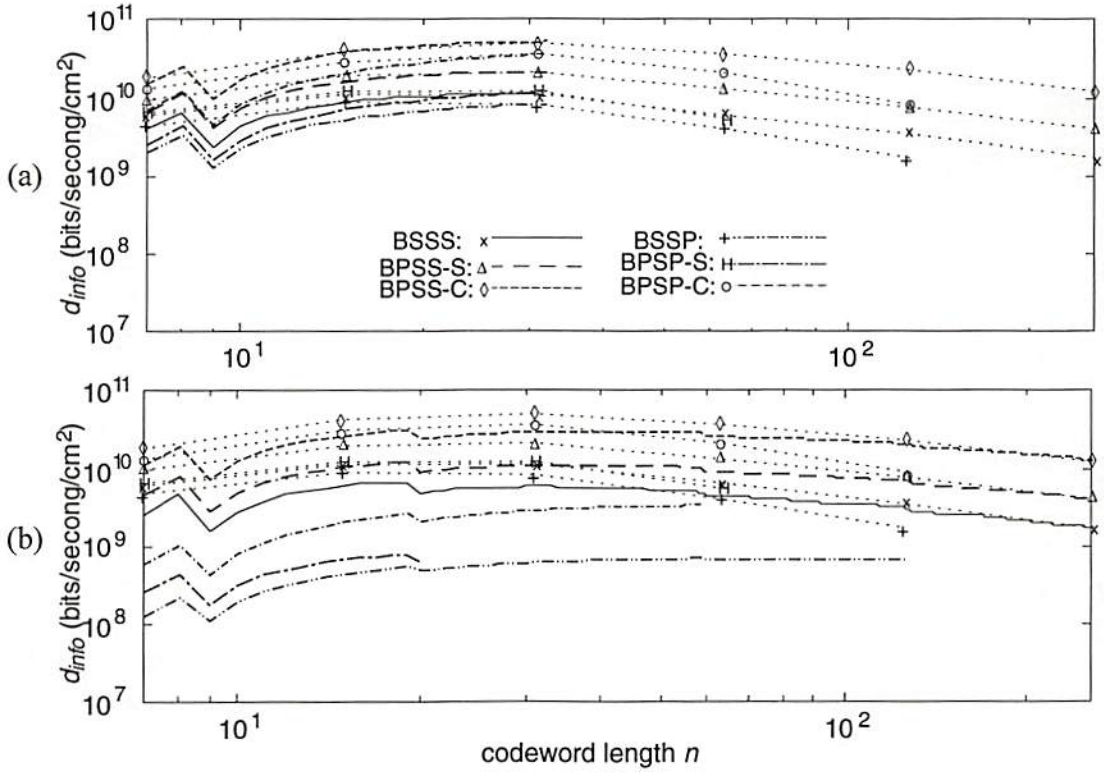


Figure 6.9 Information spatial channel density d_{info} for the primitive RS codes (dotted lines), (a) m = 5, and (b) m = 8 RS codes.

Table 6.2 The highest information spatial channel density, d_{info}, of the TDA RS decoders for the m = 5 and 8 RS codes.

	$P_b = 10^{-4} \rightarrow P_e = 10^{-12}$				$P_b = 10^{-4} \rightarrow P_e = 10^{-15}$			
	$m = 5$		$m = 8$		$m = 5$		$m = 8$	
	n	d_{info}	n	d_{info}	n	d_{info}	n	d_{info}
BSSS	26	1.08×10^{10}	19	6.68×10^{9}	27	8.68×10^{9}	38	4.77×10^{9}
BPSS-S	30	2.10×10^{10}	19	1.25×10^{10}	27	1.65×10^{10}	43	9.38×10^{9}
BPSS-C	32	5.16×10^{10}	19	3.26×10^{10}	27	4.11×10^{10}	43	2.63×10^{10}
BSSP	32	8.25×10^{9}	58	7.09×10^{8}	27	6.38×10^{9}	89	6.81×10^{8}
BPSP-S	32	1.22×10^{10}	19	7.83×10^{8}	27	8.97×10^{9}	16	5.24×10^{8}
BPSP-C	32	3.67×10^{10}	32	3.44×10^{9}	27	2.71×10^{10}	43	2.97×10^{9}

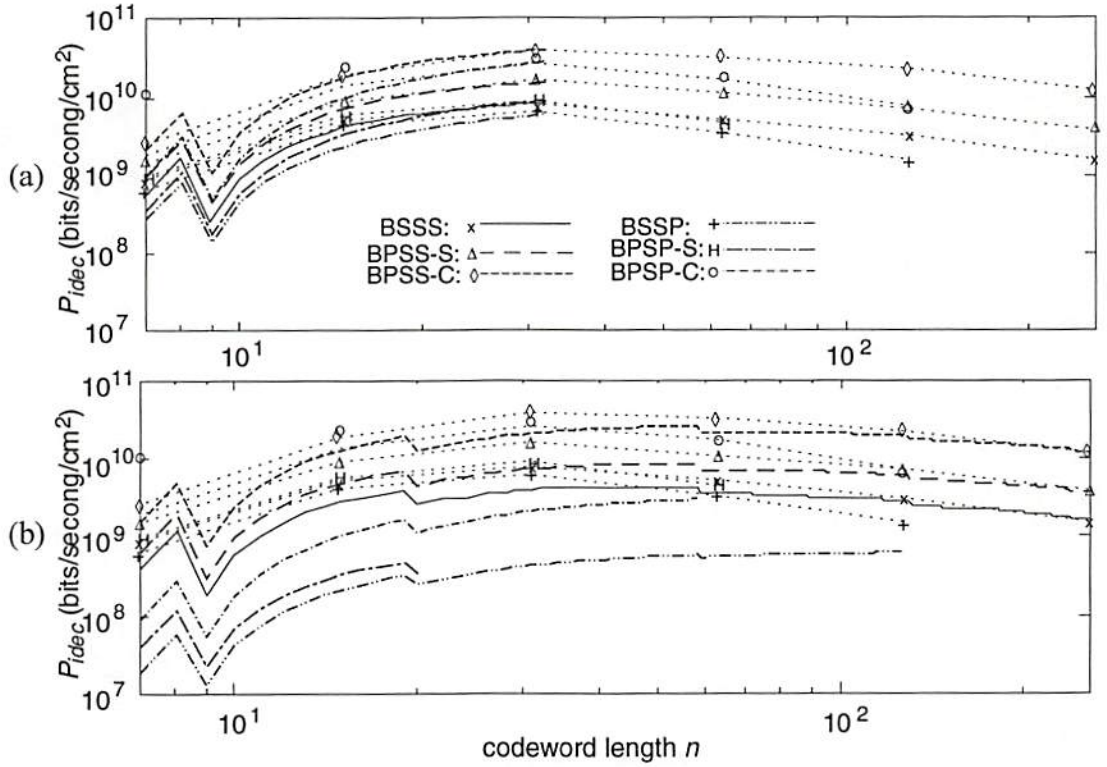


Figure 6.10 Product of the information density and the effective capacity, P_{idec}, for the primitive RS codes (dotted lines), (a) $m = 5$, and (b) $m = 8$ RS codes.

codes which reduce the BER from 10^{-4} to 10^{-12} and 10^{-15}. Note that the best TDA design for $P_e = 10^{-15}$ in terms of P_{idec} is still the BPSS-C decoder, and the best RS code is (58, 44) code in $GF(2^6)$ of $t = 7$ and $r = 0.76$.

6.3 Code Dependent Analysis

In this section, the RS codes which simultaneously satisfy a specified output BER at a raw BER and a specified code rate are selected by using the code dependent constraints, BEC and CRC (Section 4.3). In order to depict the constraints, an (n, t) code plane is used in which each grid point specifies an RS code.

Table 6.3 The highest normalized figure of merit, P_{idec}/M, of the TDA decoders for the $m = 5$, 6, and 8 RS codes. Note that the P_{idec}/M peak for $P_e = 10^{-15}$ occurs at $m = 6$, rather than $m = 5$.

	$P_b = 10^{-4} \rightarrow P_e = 10^{-12}$				$P_b = 10^{-4} \rightarrow P_e = 10^{-15}$			
	$m = 5$		$m = 8$		$m = 6$		$m = 8$	
	n	P_{idec}/M	n	P_{idec}/M	n	P_{idec}/M	n	P_{idec}/M
BSSS	32	7.97×10^{9}	46	4.20×10^{9}	55	4.65×10^{9}	43	3.41×10^{9}
BPSS-S	32	1.57×10^{10}	53	8.66×10^{9}	58	9.54×10^{9}	43	6.76×10^{9}
BPSS-C	32	3.87×10^{10}	58	2.48×10^{10}	58	2.68×10^{10}	43	1.90×10^{10}
BSSP	32	6.19×10^{9}	126	6.28×10^{8}	58	2.88×10^{9}	125	5.80×10^{8}
BPSP-S	32	9.17×10^{9}	19	4.53×10^{8}	58	3.84×10^{9}	20	2.06×10^{8}
BPSP-C	32	2.76×10^{10}	58	2.84×10^{9}	58	1.45×10^{10}	55	2.16×10^{9}

The bit error constraint (BEC) specifies the minimum number of parity-check symbols in a codeword that is required to achieve the desirable BER. For an RS code, the number of parity-check symbols is equal to $2t$ where t is the maximum number of error symbols that can be corrected. An upper bound of the output BER P_e of an $(n, n-2t)$ RS code with raw BER, P_b, is given in Eq. (4.8). In turn, given m, n, P_b, and P_e, the smallest t satisfying Eq. (4.8) can be calculated. Figure 6.11 shows the minimum t that is needed to reduce the BER from 10^{-4} to 10^{-9}, 10^{-12}, and 10^{-15} for the primitive, $m = 4$, 5, and 8 RS codes. Note that the t value grows rapidly at small n and becomes steady at large n. This confirms that the long-codeword codes have higher code rate than the short-codeword codes of the same error correction capability.

The code rate constraint (CRC) specifies the maximum t that ensures the code rate r of an $(n, n-2t)$ RS code is larger than a required code rate r_q as given in Eq. (4.10). In Fig. 6.11, two dotted lines that specify the maximum t achieving $r = 0.6$ and 0.75 are shown.

The RS codes in the intersection of the upper half plane of BEC and the lower half plane of the CRC simultaneously satisfy the code dependent constraints. For example, the RS codes in the shaded regions in Fig. 6.11 reduce the BER from 10^{-4} to 10^{-12} and have code rate higher than 0.75. Note that no RS codes of $m = 4$ is found, and only the extension RS code in GF(2^5) satisfies the code dependent constraints. In addition, the RS codes on the P_e curve have the highest code rate in the region because of the smallest t (at a fixed n).

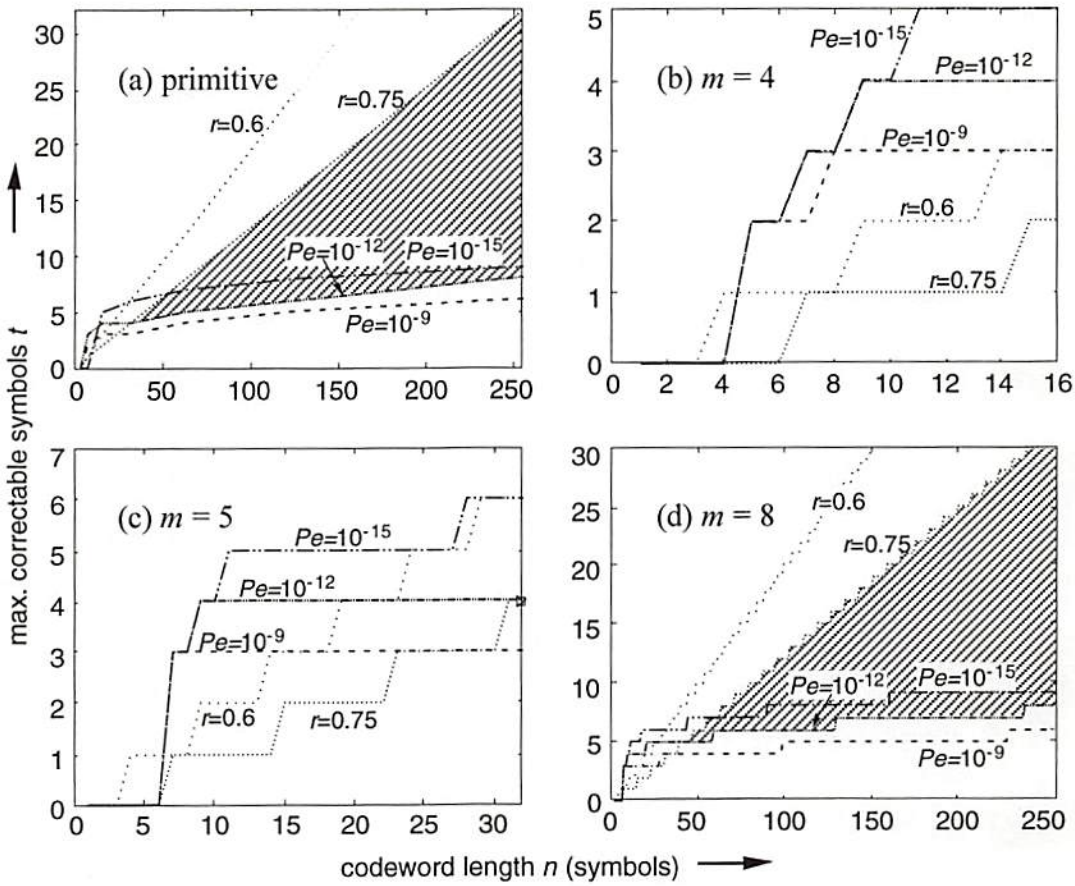


Figure 6.11 The bit error constraint (BEC) and the code rate constraint (CRC) for (a) the primitive RS codes, (b) $m = 4$, (c) $m = 5$, and (d) $m = 8$ RS codes with $P_b = 10^{-4}$. Note that the RS codes in the shaded region simultaneously satisfy $r \geq 0.75$ and $P_e \leq 10^{-12}$.

6.4 VLSI Dependent Analysis

The two interface constraints, MINC and MAXC discussed in Section 4.3, use the code parameters m, n, and t as inputs to compute the corresponding number of the TDA decoders which satisfy the VLSI physical requirements, the VLSI area A_{pg} and the power density P_{pg}. In Section 6.2, the block rate was determined by letting all the fabricated decoders operate at the largest possible frequency that is limited by the given power density. In this section, the clock rate f_c here, however, was specified (as long it is lower than the maximum clock rate $f_{c,max}$) to match the data transfer rate at the output channels. The specified clock rate was in general larger than the block rate and, then, only some of the fabricated decoders operated at that rate. Since the input data rate is fixed, choice of the highest code rate becomes the issue, and thanks to the highest code rate determined, the capacity of the optical page-oriented memory is effectively utilized.

In the buffer length constraint (MINC), the factors of the buffer length and the longest decoding delay that affect the decoding throughput are considered. For a TDA decoder implementing an (n, k) RS in $GF(2^m)$, the number, $N_{D\text{-}D}$, of the decoders to achieve the required memory throughput is determined by Eq. (4.12). The power/area constraint (MAXC), described by Eqs. (4.13) to (4.15), is used to determine the number, N_{Dmax}, of the TAD decoders which are possibly fabricated or operated at a specified clock rate limited to the given VLSI area and power density. The $N_{D\text{-}D}$ and N_{Dmax} specified for the primitive RS codes are shown in Fig. 6.12 which is commented as following:

(1) The interface implementation and the RS codes that satisfy all the conditions are determined when $N_{Dmax} \geq N_{D\text{-}D}$. The result shown here agrees with the result shown in Fig. 6.8 in which a dotted line shows the minimum required input rate.

(2) In (a), $N_{D\text{-}D}$ stops at $n = 255$ indicating that the decoding delay D_{long} is larger that the memory access period t_a. Therefore, no minimum number of decoders is specified unless larger t_a is specified.

(3) In figures (b), (d) and (f), N_{Dmax} stops at some n which shows that, beyond that n, either the power dissipation and/or the area of a single decoder are larger than the interface area and the power density, and, therefore, no decoders can be fabricated or operated.

(4) As shown in (d) and (f), the decoding throughput of a single decoder for the codes of large n is larger than the input throughput to the interface. Therefore, only a BPSP decoder is needed. However, the large decoder area and high power dissipation inhibit their fabrication in finite physical conditions.

6.5 Interface Feasibility Analysis

To conclude this section , two examples are used to illustrate the design scenario of the smart-pixel error-correcting interface. Both assume: 0.25-μm CMOS process; VLSI area, 10 cm^2; power density, 2 Watts/cm^2; data page size, 1,024 × 1,024 bits; clock rate, 320 MHz (100 MHz for 0.8-μm process); data page access period, 10 μs; raw BER, 10^{-4}; and output BER, 10^{-12} or better.

The first example shows the result obtained from applying the four constraints to the BPSS-S and BPSP-S implementations for the RS codes in $GF(2^5)$. Figure 6.13 (a) shows the number of TDA BPSS-S decoders in the decoder array in terms of various pairs of parameter (n, t) by applying the MINC and MAXC,. Note that the (n, t) can be used for the SP interface when the feasible decoders are more than the required ones, i.e., N_{Dmax} (MAXC) ≥ $N_{D\text{-}D}$ (MINC). The intersection of the MINC and the MAXC planes is projected onto an n-t plane, as shown in Fig. 6.13 (b), in which the lines of the

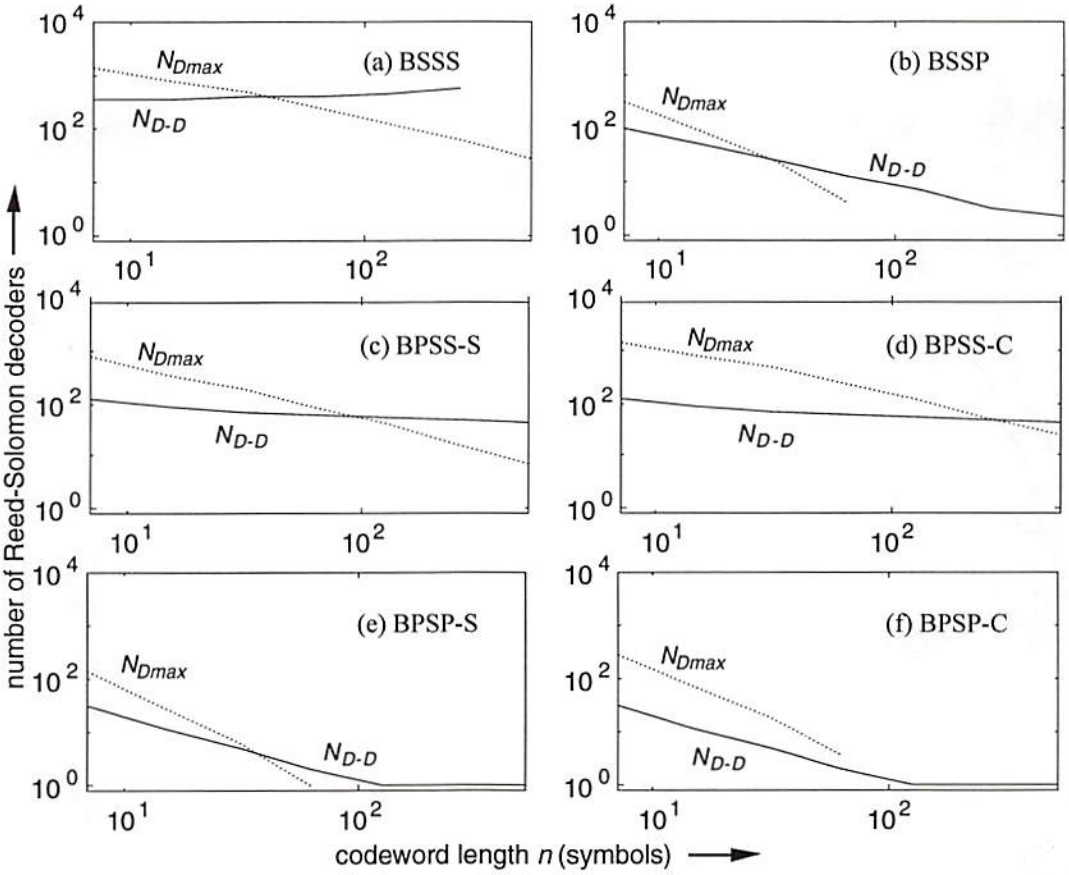


Figure 6.12 The $N_{D\text{-}D}$ from the buffer length constraint (MINC) and the N_{Dmax} from the power/area constraint (MAXC) for the implementations of the (a) BSSS, (b) BSSP, (c) BPSS-S, (d) BPSP-S, (e) BPSS-C, and (f) BPSP-C for the primitive RS codes ($F = 0.25$ μm, $P_b = 10^{-4}$, $P_e = 10^{-12}$, $f_c = 320$ MHz, $A_{pg} = 10$ cm^2, $P_{pg} = 2$ W/cm^2, $N_{in} = 10^6$ bits).

maximal t for $r = 0.6$ and 0.75 and the minimal t for $P_e = 10^{-12}$ and 10^{-15} are also shown. It shows that the RS codes of $n \geq 23$ satisfy both $r = 0.6$ and $P_e = 10^{-12}$, and only the (32, 24) code satisfies both $r = 0.75$ and that P_e. Figure 6.13 (c) presents the same result in an n-r plane. Note that no RS codes in $GF(2^5)$ has r greater than 0.75 and reduces the BER from 10^{-4} to 10^{-15}. Figures 6.13 (b) and (c) also shows the intersection of MINC and MAXC of the BPSP-S interface. Although their d_{info} is lower than the BPSS-S decoders (Section 6.2), the intersection is still higher than the r and P_e lines and the BPSP-S decoders can also be used in the SP interface.

In Fig. 6.13 (c), the RS codes in the upper half plane of the intersection curve satisfy the MINC and the MAXC, and the lower half plane of the P_e curve satisfies the BEC. Note that the BPSS-S starts at the top (actually it is from $r = 1$, but not shown here) and the BPSP-S starts at $r = 0$. At small n, the BPSS-S decoder has limited input channels and, hence, the interface requires much more decoders than the area and power can offer. Therefore, limited to the timing and buffer length, no RS codes can be used. On the contrary, the BPSP-S decoder has shorter decoding delay and, hence, achieves high data rate at small n. Therefore, any choices of t are acceptable even when $2t \geq n$. The maximum r achieved by the BPSS-S and the BSPS-S interfaces is 0.75 due to the code dependent constraints. When P_e was required at 10^{-15} or better, the r achieved by the two decoders for $m = 5$ RS codes was merely above 0.6.

In order to achieve higher r, the RS codes of long n, e.g., 255, have to be used, and, as shown in Fig. 6.8, the BPSS-C is the only design that can implement such long RS codes. In the second example, the BPSS-C design and the $m = 8$ RS codes were analyzed using the result obtained from the four constraints. Figure 6.14 shows the intersection of the MINC and MAXC in the n-t and r-t planes. The smallest n values that satisfy $P_e = 10^{-12}$ (10^{-15}) were obtained at 117 (150) which corresponds to $r = 0.897$ (0.893). The largest r for $P_e = 10^{-12}$ (10^{-15}) was obtained at $n = 237$ (256) with $r = 0.941$ (0.930) which is much higher than the r using the BPSS-S and BPSP-S designs for the $m = 5$ codes.

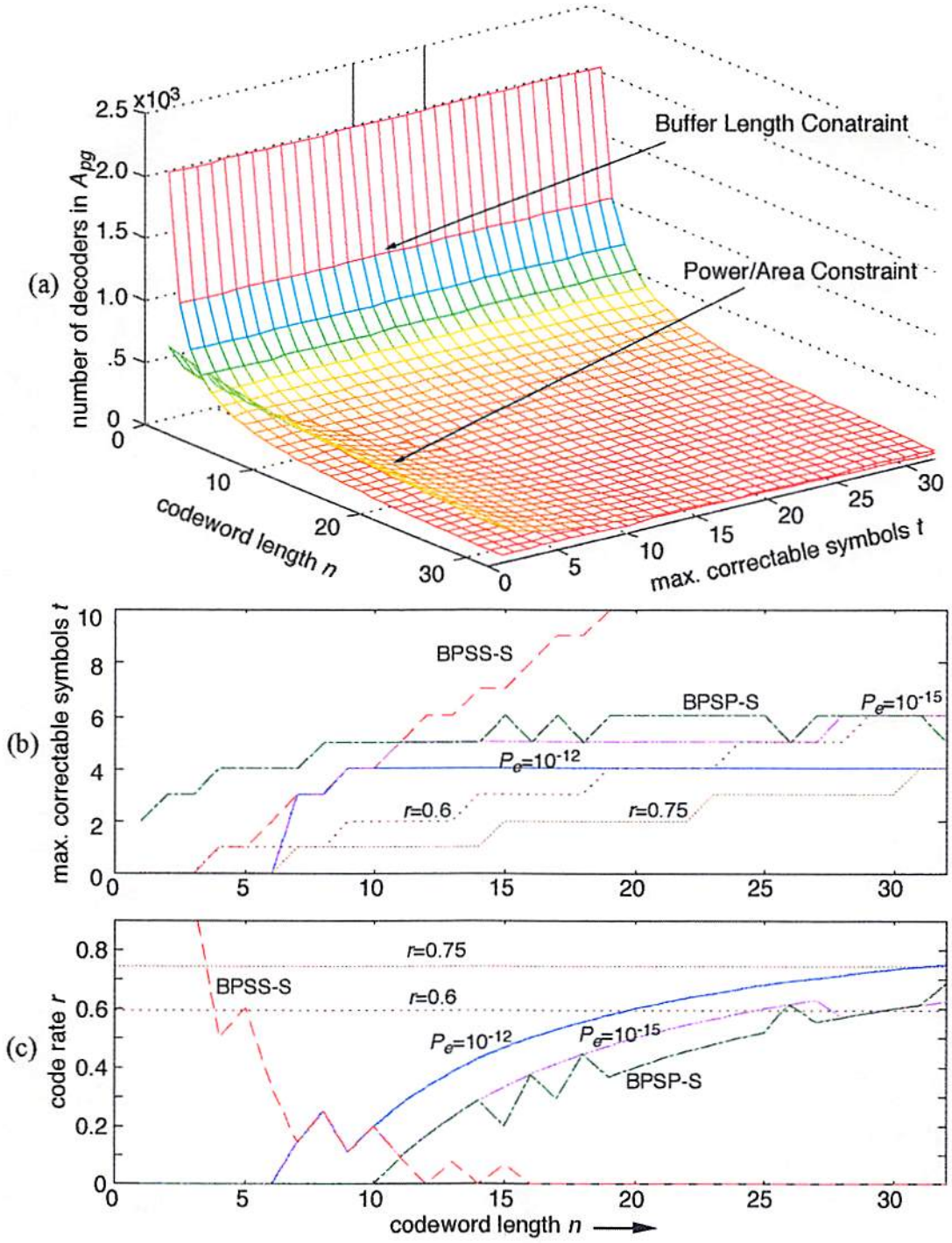


Figure 6.13 Feasibility analysis of the BPSS-S and the BPSP-S decoders for the $m = 5$ RS codes. (a) the MINC (buffer length constraint) and MAXC (power/area constraint) of the BPSS-S interface; the BEC, CRC, MINC, and MAXC (b) in the n-t plane, and (c) in the n-r plane ($F = 0.25$ μm, $P_b = 10^{-4}$, $A_{pg} = 10$ cm^2, $P_{pg} = 2$ W/cm^2, $N_{in} = 10^6$ bits, $t_a = 10$ μs, $f_c = 320$ MHz).

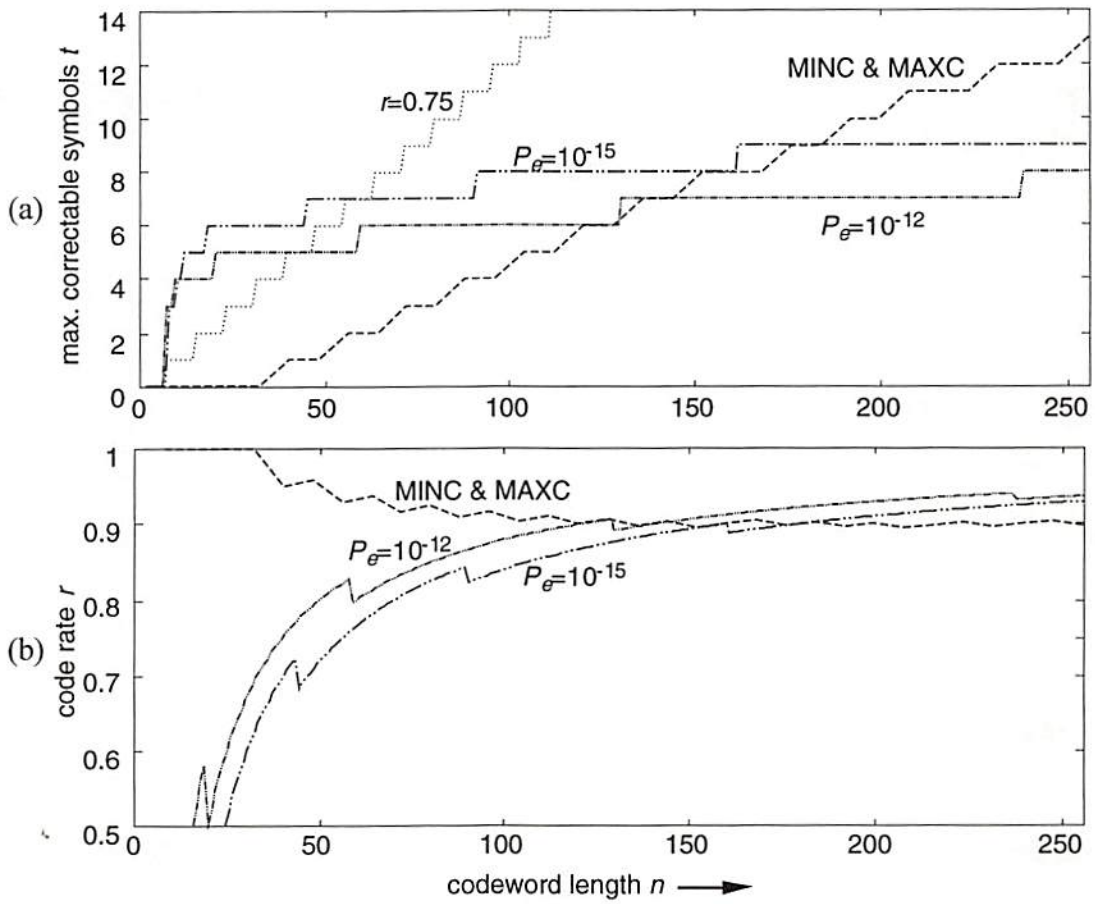


Figure 6.14 Feasibility analysis of the BPSS-C interface for the $m = 8$ RS codes. (a) BEC, CRC, MINC, and MAXC on the n-t plane, and (b) BEC, MINC, and MAXC on the n-r plane ($F = 0.25$ μm, $P_b = 10^{-4}$, $A_{pg} = 10$ cm^2, $P_{pg} = 2$ W/cm^2, $N_{in} = 10^6$ bits, $t_a = 10$ μs, and $f_c = 320$ MHz).

Chapter 7

Discussion and Conclusion

Optical page-oriented memory (OPOM), employing advanced photonic materials and optoelectronic devices, provides the large capacity and the high data access rate required by novel digital information applications. Unfortunately, uncoded OPOMs have a high raw bit error rate (BER) which presents a limitation. The use of error detection/correction is one way to reduce the BER to an acceptable level and improve overall memory capacity. Reed-Solomon (RS) codes are frequently used for error correction because they can effectively correct both random and burst errors. Likewise, RS codewords have a variety of lengths, and they are separated at the largest possible distance in the code space. We discussed the construction, specifications, and requirements of the output interface of OPOMs containing an array of RS decoders implemented using smart pixel (SP) technology. Each SP cell consists of an electrical RS decoder and an optical parallel I/O. Because of the large number of parallel I/O channels and a high processing rate, the SP interface simultaneously reduces the BER to a desirable rate and provides a high aggregate data throughput. In this thesis, six SP implementations of the RS decoder using the transform decoding algorithm were analyzed to find the most effective implementation.

We summarize the results based on the two scenarios defined in Chapter 4. The first scenario was developed to evaluate the performance of the six implementations in terms of: the input spatial channel density (d_{scin}); the aggregate input data throughput, the information spatial channel density (d_{info}); and the product of the information

density and effective capacity (P_{idec}). In this scenario, d_{info} and P_{idec} are optimized as functions of data page size, memory access time, and other physical conditions. It was shown that RS decoding processes need smart pixel technology to provide a large number of I/O because the d_{scin} of most implementations exceeds electrical limits, as shown in Fig. 6.6. The BPSS-C decoder for the (32, 24) RS code in $GF(2^5)$ provides the largest d_{info} for an SP interface where $P_e = 10^{-12}$. In an SP interface where $P_e = 10^{-15}$, the (27, 17) RS code also implemented by the BPSS-C decoder provides the largest d_{info}. The largest P_{idec} comes from BPSS-C design implementing the (32, 24) RS code for an SP interface where $P_e = 10^{-12}$. However, when $P_e = 10^{-15}$, the (58, 44) RS code in $GF(2^6)$ provides the largest P_{idec}. These results assume that area = 10 cm^2, power density = 2 Watts/cm^2, and $P_b = 10^{-4}$.

The second scenario was developed to determine the largest memory capacity by optimizing the code rate of RS codes when the maximal data access rate, specified by the access time and the size of a data page, is known. This scenario determines which RS codes can satisfy the four following constraints: the bit error requirement; the code rate requirement; the buffer length limitation; and the VLSI codes that meet these requirements, the code of the highest code rate is then selected, because such a code would utilize the memory capacity most effectively. For the example discussed, the (237, 223) RS code ($r = 0.941$) in $GF(2^8)$ implemented by the BPSS-C design was the best selection when $P_e = 10^{-12}$. The (256, 238) RS code provided the highest r (= 0.930) when $P_e = 10^{-15}$. These results assume that page access time = 10 μs, page size = 1,024 × 1,024 bits, and the same parameters as in the first scenario.

Figure 7.1 shows the relationship between code rate r and d_{info} for $P_e = 10^{-12}$ and 10^{-15}, and Fig. 7.2 shows the relationship between r and the normalized P_{idec}. The first scenario determines the RS code denoted by S_1 which represents the highest d_{info} (Fig. 7.1) and the largest P_{idec} (Fig. 7.2), and the second scenario determines S_2 representing

the largest r. Note that r is proportional to the usable capacity of the OPOMs. The zigzags of the lines of fixed m come from discontinuities of r for various n at a fixed m, as shown in Fig. 3.3. When physical conditions are changed to increase the data throughput (e.g., smaller VLSI feature size, larger area, and larger power density), these lines move to the right without changing r.

From Figs. 7.1 and 7.2, the RS codeword length n tends to approach two extremes: achieving either high data throughput (shorter n), or large capacity (longer n). One possible way to extend the envelops of these conflicting requirements is to use 3-D VLSI packaging to implement long-length RS decoders. The 3-D packaging technique connects the multiple stacked substrates with electrical circuitry through optical vias. Individual modules of the TDA decoder are fabricated on separated substrates, and the substrates are aligned and interconnected to perform the pipelined decoding scheme. Another possibility is the product codes in which two RS codes with shorter n are combined. The product codes provide higher combined code rate than a regular RS code of the same error-correcting capability. In addition, the design of decoder for short-length codes is easier.

There are two other results discovered in this study. First, the VLSI circuit simulation model, SUSPENS, was originally developed for electronic general-purpose microprocessor chips. It was modified for the SP decoder array so that the buffers were estimated separately from the decoding logic and more proper parameters were used. However, two intuitive problems exist. First, the average number of transistors per logical gate is 50% larger than the average number of the general-purpose chips which might affect the use of the modified model. Secondly, the Rent's rule was obtained empirically from electronic circuitry where the pins are on the edge of a chip. For

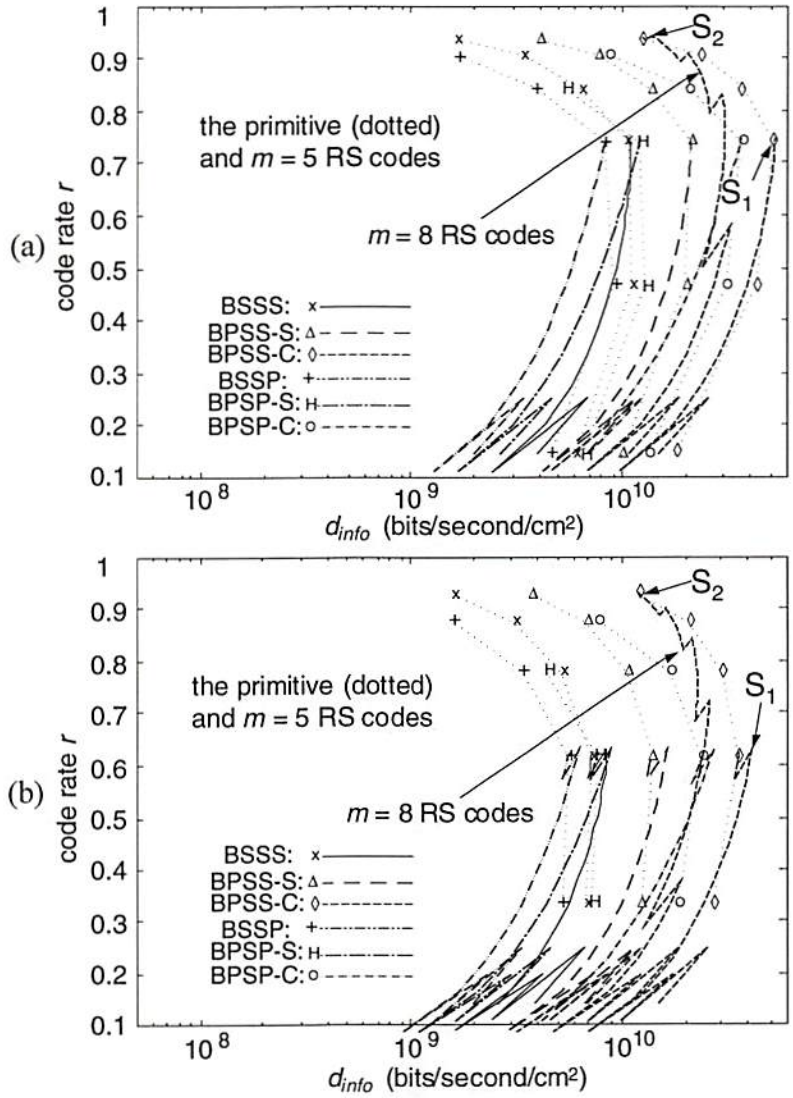


Figure 7.1 Relationship between the code rate r and the information spatial channel density d_{info} for (a) $P_e = 10^{-12}$ and (b) $P_e = 10^{-15}$. Point S_1, obtained from the first scenario, denotes the highest d_{info}, and point S_2, obtained from the second scenario, denotes the largest code rate, i.e., the largest usable storage capacity.

optoelectronic SP devices, the optical sources and receivers can be mounted together with the electronic components. This planar arrangement also affects the application of the modified SUSPENS to the estimation of SP devices.

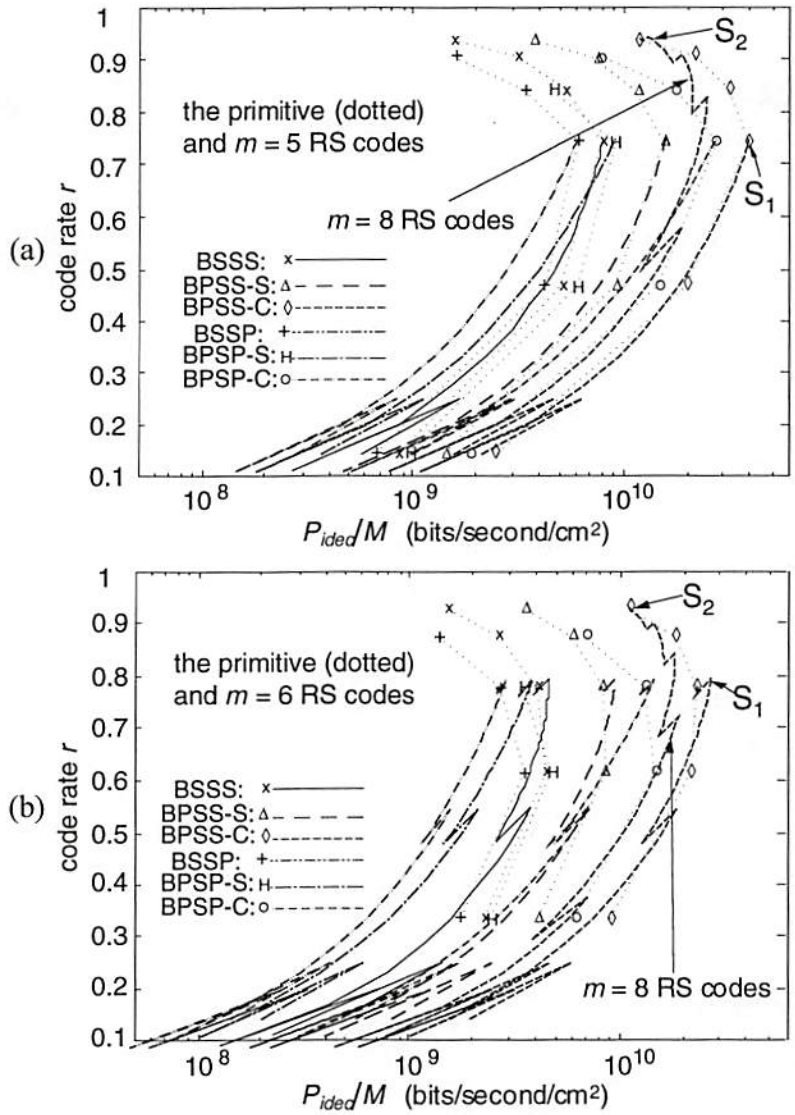


Figure 7.2 Relationship between the code rate r and the normalized information spatial channel density P_{idec} for (a) $P_e = 10^{-12}$ and (b) $P_e = 10^{-15}$. Point S_1, obtained from the first scenario, denotes the highest P_{idec}, and point S_2, obtained from the second scenario, denotes the largest code rate, i.e., the largest usable capacity.

The second result is that the TDA is not a 'good' scheme for shortened RS codes, nor for the RS codes with short n. As shown in Figs. 5.14, 6.1, and 6.8, the decoding hardware and power dissipation changed slightly as n decreases for a fixed m. In the implementations of the TDA, the ITES module needs many more logical gates than the

other modules, and the number of logical gates is proportional to $m(N - 2t)$, where $N = 2^m - 1$. In order to achieve better performance design, different decoding schemes will be studied and applied.

Figure 7.3 shows the performance of the implementations studied in this thesis. The horizontal axis shows the input spatial channel density, and the vertical axis shows the information rate per channel. The three dashed lines represent the information spatial channel density at 10^{-6}, 10^{-9}, and 10^{-12} bits per second per cm^2, respectively. The d_{info} of the implementations of the SP interface for a large number of RS codes which reduce the BER for 10^{-4} to 10^{-12} and 10^{-15} is in the range of 10^8 to 10^{11} bits/second/cm^2. This results in the aggregate information rate up to 1 terabit per second in 10 cm^2.

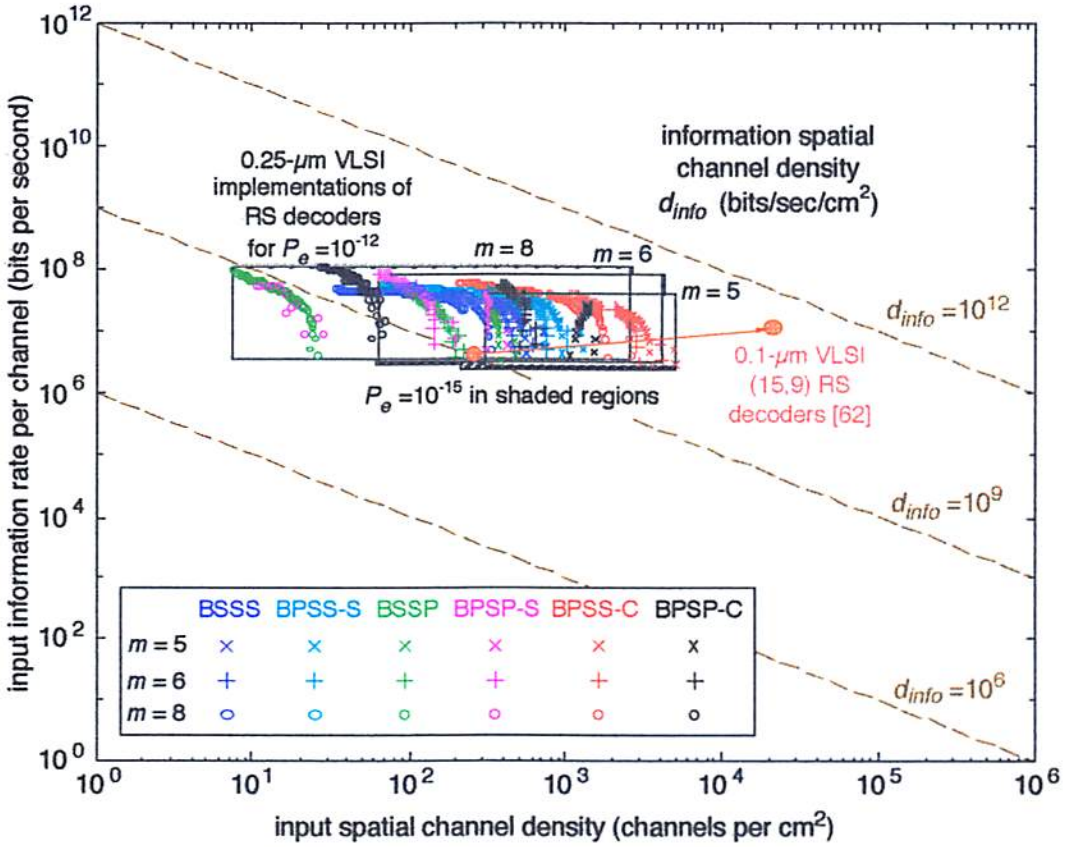


Figure 7.3 Performance of the optoelectronic smart pixel error-correction interfaces for optical page-oriented memories.

Chapter 8

Future Extensions

The study described this dissertation has suggested many additional areas of future research. Some interesting ideas are summarized below:

- This work highlights the use of the smart pixel technology in optical page-oriented memories. Fabrication and setup of an experimental SP implementation with complicated decoding circuitry will be an extremely interesting and challenging project.
- The clock distribution in the pipeline decoding algorithm consumes the most power (Fig. 6.5). Different approaches can be studied to synchronize the pipelined processes and reduce the power dissipation.
- The analyses techniques developed in this work to the design of other complex logic smart pixel systems.
- When long-n RS codes are implemented, 3-D VLSI packaging techniques are needed. The details of alignment and assembly of 3-D packages is non-trivial and interesting.
- For practical designs, product codes combining two short-n RS codes are employed. One code is used to correct random and/or small burst errors, and the other code is used with an interleaving scheme to correct large burst errors that may extend over thousands of bits. The interleaving can be realized by using diffractive optical devices.
- 2-D error-correction codes (array codes) can be applied to OPOMs.

- Different decoding schemes can be studied, especially for RS codes having small t. When $t \leq 3$, the roots of the error-location polynomial can be found using compound circuits with optimal design which could reduce the hardware up to 40%.
- This work suggests modifications in the application of Rent's rule, used in the prediction of electrical circuits, to optoelectronic devices. Rent's constant p is 0.6 - 0.7 for electrical circuits because of the pin-outs on the edge of the chip. For OE devices, p could be larger due to the optical I/O on the chip.

References

[1] J. F. Heanue, M. C. Bashaw, and L. Hesselink, "Volumeholographic storage and retrieval of digital data," *Science* **265**, 749-752 (1994).

[2] H. Ishio, "Next-generation communications networks and optical fiber technologies," *Optoelectronics-Devices and Technologies* **10**, 3-14 (1995); and Special Issue on Optical Interconnections for Information Processing, *IEEE J. Lightwave Technol.* December 1995.

[3] D. Chen and J. D. Zook, "An overview of optical data storage technology," *Proc. IEEE* **63**, 1207-1230 (1975).

[4] T. Parish, "Crystal clear storage," *Byte*, 283-288, November 1990.

[5] S. Jutamulia and G. M. Storti, "Three-dimensional optical digital memory," *Optoelectronics-Devices and Technologies* **10**, 343-360 (1995).

[6] M. A. Neifeld and M. McDonald, "Eror correction for increasing the usable capacity of photorefractive memories," *Opt. Lett.* **19**, 1483-1485 (1994).

[7] T. R. N. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[8] S. R. Whitaker, J. A. Canaris, and K. B. Cameron, "Reed Solomon VLSI codec for advanced television," *IEEE Trans. Circuits and Systems for Video Technol.* **1**, 230-236 (1991).

[9] A.A. Sawchuk, "Smart Pixel Devices and Free-Space Digital Optics Applications," *LEOS '95 Conference Proceedings,* IEEE Lasers and Electro-Optics Society, 1995 Annual Meeting, San Francisco, November 1995, pp. 268-269, (invited paper).

[10] M. A. Neifeld and J. D. Hayes, "Parallel error correction for optical memories," *Opti. Mem. Neur. Netw.* **3**, 87-98 (1994).

[11] M. A. Neifeld and J. D. Hayes, "Error-correction schemes for volume optical memories," *Appl. Opt.* **34**, 8183-8191 (1995).

[12] H. M. Shao, T. K. Trung, L. J. Deutsch, J. H. Yuen, and I. S. Reed, "A VLSI design of a pipeline Reed-Solomon Decoder," *IEEE Trans. Comput.* **C-34**, 393-403 (1985).

[13] P. J. van Heerden, "Theory of optical information storage in solids," *Appl. Opt.* **2**, 393-400 (1963).

[14] S. Boj, G. Pauliat, and G. Roosen, "Dynamic holographic memory showing readout, refreshing, and updating capabilities," *Opt. Lett.* **17**, 438-410 (1993).

[15] L. Hesselink and M. C. Bashaw, "Optical memories implemented with photorefractive media," *Opt. Quantum Electron.* **25**, 611-661 (1993).

[16] F. H. Mok, M. G. Tackitt, and H. M. Stoll, "Storage of 500 high-resolution holograms in a $LiNbO_3$ crystal," *Opt. Lett.* **16**, 605-607 (1991).

[17] K. Rastani, "Storage capacity and cross talk in angularly multiplexed holograms: two case studies," *Appl. Opt.* **32**, 3772-3778 (1993).

[18] C. Gu, J. Hong, I. McMichael, R. Saxena, and F. Mok, "Cross-talk-limited storage capacity of volume holographic memory," *J. Opt. Soc. Am. A* **9**, 1978-1983 (1992).

[19] J. F. Heanue, M. C. Bashaw, and L. Hesselink, "Sparse selection of reference beams for wavelength- and angular-multiplexed volume holography," *J. Opt. Soc. Am. A* **12**, 1671-1676 (1995).

[20] G. A. Rakuljic, V. Leyva, and A. Yariv, "Optical data storage by using orthogonal wavelength-multiplexed volume holograms," *Opt. Lett.* **17**, 1471-1473 (1992).

[21] S. Yin, H. Zhou, F. Zhao, M. Wen, Z. Yang, and F. T. S. Yu, "Wavelength multiplexed holographic storage in a sensitive photorefractive crystal using a visible-light tunable diode laser," *Opt. Comm.* **101**, 317-321 (1993).

[22] H. Sasaki, J. Ma, Y. Fainman, S. H. Lee, and Y. Taketomi, "Fast update of dynamic photorefractive optical memory," *Opt. Lett.* **17**, 1468-1470 (1992).

[23] C. Denz, G. Pauliat, and G. Roosen, "Volume hologram multiplexing using a deterministic phase encoding method," *Opt. Comm.* **85**, 171-176 (1991).

[24] D. Psaltis, M. Levene, A. Pu, G. Barbastathis, and K. Curtis, "Holographic storage using shift multiplexing," *Opt. Lett.* **20**, 782-784 (1995).

[25] D. Brady and D. Psaltis, "Control of volume holograms," *J. Opt. Soc. Am. A* **9**, 1167-1182 (1992).

[26] K. Curtis, A. Pu, and D. Psaltis, "Method for holographic storage using peristrophic multiplexing," *Opt. Lett.* **19**, 993-994 (1994).

[27] L. Hesselink and S. Redfield, "Photorefractive holographic recording in strontium barium niobate fibers," *Opt. Lett.* **13**, 877-879 (1988).

[28] F. H. Mok, "Angle-multiplexed storage of 5000 holograms in lithium niobate," *Opt. Lett.* **18**, 915-917 (1993).

[29] D. Psaltis and A. Pu, "Holographic 3-D disks," *Optoelectronics-Devices and Technologies* **10**, 333-342 (1995).

[30] D. A. Parthenopoulos and P. M. Rentzepis, "Three-dimensional optical storage memory," *Scinece* **245**, 843-845 (1989).

[31] S. Hunter, F. Kiamilev, S. Esener, D. A. Parthenopoulos, and P. M. Rentzepis, "Potentials of two-photon based 3-D optical memories for high performance computing," *Appl. Opt.* **29**, 2058-2066 (1990).

[32] C. De Caro, A. Renn, and U. P. Wild, "Hole buring, Stark effect, and data storage: 2: holographic recording and detection of spectral holes," *Appl. Opt.* **30**, 2890-2898 (1991).

[33] H.-J. Muschenborn and U. P. Wild, "holographic image storage and molecular computing using spectral hole-burning," *Optoelectronics-Devices and Technologies* **10**, 311-332 (1995).

[34] M.-P. Bernal, H. Coufal, R. K. Grygier, J. A. Hoffnagle, C. M. Jefferson, R. M. Macfarlane, R. M. Shelby, G. T. Sincerbox, P. Wimmer, and G. Wittmann, "A precision tester for studies of holographic optical storage materials and recording physics," submitted to *Applied Optics*.

[35] A. Pu and D. Psaltis, "High density recording in photopolymer-based holographic 3-D disks," to be published in *Applied Optics*, May 1996.

[36] D. A. B. Miller, "Optics for low-energy communication inside digital processors: quantum detectors, sources, and modulators as efficient impedance converters," *Opt. Lett.* **14**, 146-148 (1989).

[37] D. A. B. Miller, "Quantum-well self-electro-optic effect devices," *Opt. Quantum Electron.* **22**, S61-S98 (1990).

[38] A. L. Lentine, *et al.*, "Field-effect-transistor self-electro-optic effect device (FET-SEED) electrically addressed diffeential modulator array," *Appl. Opt.* **33**, 2849-2855 (1994).

[39] A. V. Krishnamoorthy, *et al.*, "3-D integration of MQW modulators over active submicron CMOS circuits: 375 Mb/s transimpedance receiver-transmitter circuit," *IEEE Photon. Technol. Lett.* **7**, 1288-1290 (1995).

[40] D. J. McKnight, K. M. Johnson, and R. A. Serati, "256 × 256 liquid-crystal-on-silicon spatial light modulator," *Appl. Opt.* **33**, 2775-2784 (1994).

[41] A. Ersen, S. Krishnakumar, V. Ozguz, J. Wang, C. Fan, S. Esener, and S. H. Lee, "Design issues and development of monolithic silicon/lead lanthanum zirconate titanate integration technologies for smart spatial light modulators," *Appl. Opt.* **31**, 3950-2965 (1992).

[42] K. Kasahara, ".VSTEP-based smart pixels," *IEEE J. Quantum Electron.* **29**, 757-768 (1993).

[43] J. L. Jewell, Y. H. Lee, A. Scherer, S. L. McCall, N. A. Olsson, J. P. Harbison, and L. T. Florez, "Surface-emitting microlasers for photonic switching and interchip connections," *Opt. Eng.* **29**, 210-214 (1990).

[44] M. Hibbs-Brenner, S. Mukherjee, J. Skogen, B. Grung, E. Kalweit, and M. Bendett, "Design, fabrication and performance of an integrated optoelectronic cellular array," *Proc. Optical Enhancements to Computing Technology, SPIE* **1563**, 10-20 (1991).

[45] J. J. Brown, J. T. Gardner, and S. R. Forrest, "An integrated optically powered, optoelectronic "smart" logic pixel for interconnection and computing applications," *IEEE J. Quantum Electron.* **29**, 715-726 (1993).

[46] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, (VLSI Systems Series), Addison Wesley, Reading, MA, 1990.

[47] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE J. Solid -State Circuits* **29**, 663-670 (1994).

[48] W. E. Donath, "Placement and average interconnection lengths of computer logic," *IEEE Trans. Circuit and Systems* **CAS-26**, 272-277 (1979).

[49] S. B. Wicker and V. K. Bhargava, Ed., *Reed-Solomon Codes and Their Applications*, IEEE Press, New York, 1994.

[50] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*, North-Holland, Amsterdam, 1977.

[51] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[52] G. C. Clark, Jr. and J. B. Cain, *Error-Correction Coding for Digital Communications*, Plenum Press, New York, 1981.

[53] H. Imai, Ed., *Essentials of Error-Control Coding Techniques*, Academic Press, New York, 1990.

[54] H. J. Caulfield and R. S. Putnam, "Fault tolerance and self-healing in optical systolic array processors," *Opt. Eng.* **24**, 65-67 (1985).

[55] S. Liebomitz and D. Casasent, "Error-correction coding in an associative processor," *Appl. Opt.* **26**, 999-1006 (1987).

[56] S. Oh, D. C. Park, R. J. Marks II, and L. E. Atlas, "Error detection and correction in multilevel algebraic optical processors," *Opt. Eng.* **27** 289-294 (1988).

[57] S. A. Ellett, J. F. Walkup, and T. F. Krile, "Error-correction coding for accuracy enhancement in optical matrix-vector multipliers," *Appl. Opt.* **31**, 5642-5653 (1992).

[58] W. Kawakami and K.-I. Kitayama, "Optical error-correction coding encoder and decoder: design considerations," *Appl. Opt.* **34**, 5064-5073 (1995).

[59] C.-S. Yeh, I. S. Reed, and T. K. Truong, "Systolic multipliers for finite fields $GF(2^m)$," *IEEE Trans. Comput.* **C-33**, 357-360 (1984).

[60] C. C. Wang, T. K. Truong, H. M. Shao, L. J. Deutsch, J. K. Omura, and I. S. Reed, "VLSI architectures for computing multiplications and inverses in $GF(2^m)$," *NASA, JPL TDA Prog. Rep.* **42-75**, 52-64, July-Sept. 1983.

[61] I.-S. Hsu, I. S. Reed, T. K. Truong, K. Wang, C.-S. Yeh, and L. J. Deutsch, "The VLIS implementation of a Reed-Solomon encoder using Berlekamp's bit-serial multiplier algorithm," *IEEE Trans. Comput.* **C-33**, 906-911 (1984).

[62] S. K. Sridharan and M. A. Neifeld, "Parallel error correction for page access optical memories," *OSA Annual Meeting*, TuE5 (Post deadline paper), Portland, September 10-15, 1995.