A Parallel Architecture for Hermitian Decoders: Satisfying Resource and Throughput Constraints

Rachit Agarwal*, Emanuel M. Popovici†, Brendan O’Flynn‡ and Michael E. O’Sullivan‡

*Department of Microeletronic Engineering, University College Cork, Cork, Ireland
†Microelectronics Application Integration Group, Tyndall National Institute, Cork, Ireland
‡Department of Mathematics and Statistics, San Diego State University, San Diego, US
Email: rachit.agarwal@ue.ucc.ie, e.popovici@ucc.ie, brendan.oflynn@tyndall.ie, m.osulliv@math.sdsu.edu

Abstract—Hermitian Codes offer desirable properties such as large code lengths, good error-correction at high code rates, etc. The main problem in making Hermitian codes practical is to find a way of performing the required computations in a fast and memory efficient way so as to satisfy resource and throughput constraints imposed by the systems. We present some architectures for Hermitian Decoders which enhance their applicability in Communication Systems. Formulae and architectures for Gap Detection and Address Generation Unit for satisfying memory constraints have been presented, which amount to 50% savings in storage area and 10% savings in the number of clock cycles reported in literature. A Semi-Parallel Architecture is proposed as a solution to the latency and resource requirements trade-off, which improves the throughput about \( q \) times compared to the word-serial architecture at an expense of some \( q \) times more adders, multipliers and simple multiplexers, where the code is defined over \( GF(q^2) \). For a \( t \) error correcting code, the resource load of the parallel architectures is about \( \gamma \frac{2g^2 + (q-2)}{q} + 1 \) times our architecture, where \( \gamma \) is the resource requirement ratio of a multiplier and an inverter.

I. INTRODUCTION

Construction of block codes from irreducible projective curves over finite fields was introduced by Goppa [1]. These codes are known as Algebraic-Geometric (AG) Codes and offer desirable properties such as large code lengths over small finite fields, the potential to find a large selection of codes and good error-correction at high code rates [2]. Despite these desirable properties, AG codes are not widely used in Communication Systems due to their being resource hungry in nature. Implementation of parallel architectures for AG codes requires huge memory and resources. Serial architectures, on the other hand, do not satisfy the throughput constraints imposed by systems and applications. Hence, the main problem in making AG codes practical is to find a way of performing the required computations in a fast and efficient way so as to satisfy resource and throughput constraints.

The objective of this paper is to study the implementation aspects of architectures for decoding of a subclass of AG codes, known as Hermitian Codes and to identify the trade-offs imposed by systems and applications. We present some techniques for enhancing the performance of both, the serial and parallel architectures which significantly reduce the storage requirements and enhance the speed of decoding. The implementation aspects of Hermitian Decoders come with a trade-off between latency and resource requirements. As a solution, we propose a semi-parallel architecture which satisfies both the throughput and resource constraints. Such an architecture is of particular advantage to applications like Optical Storage & Communications and Hard-disk Drives. Hermitian Codes with smaller fields can be of great advantage for achieving high reliability in Wireless Sensor Networks [3].

The paper is organized as follows. In Section II, we discuss the preliminaries for Hermitian Codes. We derive a gap detection formula and unit, and Address Generation Unit in Section III. Section IV covers the applications of these units in serial and parallel architectures and we propose a semi-parallel architecture in Section V. We discuss the performance enhancement in Section VI and close the paper with some final remarks in Section VII.

II. HERMITIAN CODES

We briefly discuss the notions required for discussions in the paper. For an in-depth study of Hermitian codes, readers are referred to [4]. We follow the notations and construction of codes as in [5].

We consider codes from Hermitian curve
\[
\chi : x^{q+1} = y^q + y
\]
over a finite field \( \mathbb{F}_{q^2} \). The space \( L(mP_{\infty}) \) consists of all functions on \( \chi \) that have a pole of multiplicity at most \( m \) only at the point at infinity. For \( L(mP_{\infty}) \), we choose a basis
\[
L(mP_{\infty}) = \langle x^a y^b : aq + b(q + 1) \leq m, \ 0 \leq a, \ 0 \leq b < q \rangle
\]
(1)

For a given divisor \( G \), \( L(G) \) is defined as the space of all functions satisfying
\[
L(G) = \{ f : (f) + G \geq 0 \}
\]

There are two ways of defining a Hermitian code. The functional code \( C_L(D, G) \) is defined as
\[
C_L(D, G) = \{ (f(P_0), f(P_1), \ldots, f(P_{n-1}) : f \in L(G) \}
\]
where \( P_0, P_1, \ldots, P_{n-1} \) are the \( n \) points on the curve under consideration. The residual code \( C_{\Omega1}(D, G) \) is defined as
\[
C_{\Omega1}(D, G) = (C_L(D, G))^\perp
\]
The parameters of \( C_{\Omega1}(D, G) \) are, length \( n = q^3 \), dimension \( k = n - deg(G) + g - 1 \) and designed minimum distance \( d^* = deg(G) - 2g + 2 \) for \( n > deg(G) > 2g - 2 \) [4].
Any integer $s$ has a unique representation as

$$s = aq + b(q + 1)$$

with $a$ a non-negative integer and $b \in \{0, 1, \ldots, q - 1\}$. $s$ is said to be a gap if it does not have an $a$ and $b$ satisfying the above condition.

Let a vector $y$ of length $n$ be given such that

$$y = c + e,$$

$c \in C_{\Omega}(D,G)$ where $e$ is an error vector with Hamming weight $t < d^{*}(C_{\Omega})/2$. For the Hermitian Curve, genus is given by

$$g = q^2 - q/2,$$

which is also the number of gaps in the code. All the coefficients in error locator polynomial and update polynomial corresponding to the gaps are zero. Also, syndrome $s_{a,b}$ is zero if $a$ OR $b$ is a gap. The key to eliminating these stored zeros is an efficient mechanism for gap detection that can be easily incorporated in the control unit of the decoder. We devise such a mechanism in the next section.

### III. GAP DETECTION FORMULATION

Notice from (2), $s$ is a gap if and only if for all values of $b$, corresponding $a < 0$. Also, following conclusions can be drawn from the formation of the equation:

$$\left\lfloor \frac{s}{q} \right\rfloor = a + b$$

$$s \mod q = b$$

It follows from the above two equations that $s$ is a gap if and only if

$$\left\lfloor \frac{s}{q} \right\rfloor < s \mod q$$

Since $q$ is a power of 2, the formula can be easily translated into circuitry[6]. It essentially consists of a substraction unit. In Figure 1, an example is shown for $q = 16$ and $s = s_8s_7s_6 \ldots s_1s_0$ a binary representation of a 9 bit integer. This method of gap detection can be successfully used in architectures such as the one presented in [5][8][9]. The gap formula has to be followed by a formula that gives the address for a particular non-gap $r < q^2$ and this is given by:

$$A = r \mod q + 1/2 \left\lfloor \frac{r}{q} \right\rfloor \left(\left\lfloor \frac{r}{q} \right\rfloor + 1 \right)$$

This can be implemented in hardware using a square function and some adders as presented in Figure 2.

### A. Saving Iterations Using Gap Detection

The efficient gap detection formula can be used to save some iterations. If $1 \leq i \leq q$, then the integers in the interval $[(i-1)(q+1)+1, iq-1]$ are gaps. Iterations corresponding to such numbers can be skipped. For example, we detect if the next iteration $l + 1$ is a gap and if so, the next valid iteration is $l + 1 + q - (l + 1) \mod q$. Since $q$ is a power of two, all these operations have a simple hardware equivalent.

### IV. HERMITIAN DECODER ARCHITECTURE

#### A. Serial Architecture

A word-serial algorithm and architecture of a Hermitian decoder was first introduced in [5]. It consists of $q$ Berlekamp-Massey units [7] that communicate with each other. We will consider two approaches to improve the performance of these blocks. The first is the word serial approach, in which each block can be implemented as in Figure 3. The other is a semi-parallel architecture that will be presented in the next section.

For storing the coefficients of $b_0$ and $b_1$ we use an SRAM memory. The simplicity of this architecture resides in the fact that the control unit follows the mathematical algorithm. In other words, the functions are stored as coefficients corresponding a monomial of a particular pole order, that is, at address 0, we store coefficient of $\phi_{m_0}$ at address 1, we store the coefficient of $\phi_{m_1}$, and so on. We have eliminated the storage of the zero components (corresponding to gaps) by using circuitry that alters the shifting of the registers to account for the irregularity of the non-gaps.
B. Parallel Architectures

The main disadvantage of serial architecture is that the latency is too high. Another way to improve the decoder’s performance is to use a more parallelled architecture. A fully parallel architecture based on Feng-Rao algorithm is proposed in [8]. The architecture requires \((t + \lfloor (g - 1)/2 \rfloor + 1)\) finite field inverters and \((\lfloor (g - 1)/2 \rfloor + (t + 1)/2)\) finite field multipliers, where \(g\) is the genus of the curve and \(t\) is the error correcting capability of the code. For codes over a large field, \(\text{GF}(2^8)\) for example, this implementation consumes a very large area. A solution to this is proposed in the next section, a semi-parallel architecture, which requires \(3q^2\) multipliers and \(q\) inverter, about \(\gamma(q^2 + (g+3)/4 + \lfloor (g-3)/4 \rfloor + 1)\) times lesser resource load when compared to parallel architecture, where \(\gamma\) is the ratio of resource requirements of a finite field multiplier and inverter (depending on serial or parallel implementation). The advantage of a parallel architecture is, however, the high speed of operation. The time complexity for the code \(C_m\) is \(m+g+1\).

V. SEMI-PARALLEL ARCHITECTURE

Here we propose a semi-parallel architecture which uses \(3q^2\) multipliers and \(q\) finite field inverters. The proposed architecture is about \(q\) times faster than in terms of latency compared to the serial architecture. For each polynomial \(b = b_0^{(i)}\), where \(i = 0, 1, \ldots, q - 1\), we compute discrepancies \(\mu = \mu_i\) using the formula:

\[
\mu = \sum_{k=0}^{a(b)} b_k s_{c,k} = \sum_{l=a(b)} b_{l-a(b)-k} s_{c,0-l} - k
\]

(5)

where \(c = \text{span}(b) = l - a(b)\). Figure 4 presents the architecture for computation of the discrepancy. The coefficients are stored in SRAM memories which are read on the diagonal as in the figure. For small fields, the addressing unit can be replaced by multiplexers. As the code length increases, e.g. to 256, the shift registers can be implemented as dedicated memories. For example, for a code over the field \(\text{GF}(2^4)\), if the current iteration is \(l = 34\), \(a(b_0) = 20\) and \(e = \text{span}(b_0) = l - a(b_0)\), then the syndrome is read as represented by the diagonal in the figure. The corresponding error locator coefficients that participate in the calculation of the discrepancy are given by the two lines (reading is from left to the right). The addressing units \(A_1\) and \(A_2\) give the address to read. For each row \(i\), the starting addresses are \(A_2(i) = (l - qi)/q\) and \(A_1(i) = A_2(i) - (e+i)/q\) and each clock these addresses are decreasing until the whole syndrome is scanned. If the address is a negative number (gap), then the output of the corresponding memory block is set to zero. The results of the reading are then mixed through the switch so that the corresponding pairings are obtained. For our example, the pairings (syndrome, error locator coefficient) are \((32, 18), (33, 19), (34, 20), (35, 21)\).

In order to get the multiplications right, the output register of the memories in which the error locator coefficients are stored has to be cyclically shifted by \(e \mod q\) positions. This could be achieved by \(q\) multiplexers.

The switch can be synthesized as shown in Figure 5.
The address $A$ has the form $110$ is we want to shift by two positions, $111$ for three positions, etc. The address is established at the beginning of each iteration and is kept over the iteration. In order to achieve the best clock speed, the switch can be pipelined. The hardware complexity for this block is $(q - 1) 2 \times 1$ multiplexers.

For the second part, namely updating the polynomials, the architecture is presented in Figure 6. The addressing of the memories is much simpler since we have to shift only along the x-axis.

Similar remarks as for the serial architecture can be applied. The advantage of using this architecture is that the throughput can be improved about $q$ times at an expense of some $q$ times more adders, multipliers and simple multiplexers when compared to the serial architecture.

VI. PERFORMANCE

For long codes, these formulae can amount to a significant hardware cost savings. For example, for a code over $\text{GF}(2^8)$ the number of gaps is $120$ and we can save over $30k$ bits. This is $50\%$ of the storage size reported in [9]. A greater impact of the gap detection formula is in the number of cycles required to complete the algorithm. For the above code, we save $g = 120$ cycles every iteration by skipping the gaps. This results in saving 27000 clock cycles, or approximately $10\%$ when correcting 60 errors.

VII. CONCLUSION

In this paper, we have studied issues related to satisfying resource and throughput constraints in practical implementations of Hermitian Decoder architectures. Architectures and Formulae for Gap Detection and Address Generation Unit have been presented for minimizing the storage requirements. These units also increase the speed of the decoder architectures significantly. For satisfying the trade-off between resource requirements and latency, we have proposed a semi-parallel architecture for Hermitian Decoders.

ACKNOWLEDGEMENT

This work was funded in part by the Irish Marine Institute Body (Smartcoast Project), and the Informatics Commercialisation Initiative of Enterprise Ireland (D-Systems project, ATRP-02/227A)

REFERENCES