An FPGA Interpolation Processor for Soft-Decision Reed-Solomon Decoding

Warren J. Gross
McGill University
Department of Electrical and Computer Engineering
Montreal, Quebec, Canada, H3A 2A7
wjgross@ece.mcgill.ca

Frank R. Kschischang, P. Glenn Gulak
University of Toronto
Edward S. Rogers Sr. Department of
Electrical and Computer Engineering
Toronto, Ontario, Canada, M5S 3G4
{frank@comm, gulak@eecg}.utoronto.ca

Abstract

We propose a parallel architecture for implementing the interpolation step in the Koetter-Vardy soft-decision Reed-Solomon decoding algorithm. The key feature is the embedding of both a binary tree and a linear array into a two-dimensional array processor, enabling fast polynomial evaluation operations. An FPGA interpolation processor was implemented and demonstrated at a clock frequency of 23 MHz, corresponding to decoding rates of 10–15 Mbps.

1 Introduction

Recently, there has been an interest in *soft-decision* decoding of Reed-Solomon codes, incorporating reliability information from the channel into the decoding process. In a breakthrough result, Koetter and Vardy proposed a front end which converts soft-information in the form of probabilities into integer constraints suitable for algebraic list decoding [1]. The most computationally demanding task in the soft-decision Koetter-Vardy algorithm is a weighted interpolation of a bivariate polynomial. In this paper we describe the architecture and implementation of an FPGA interpolation processor aimed at implementing decoders at real-time decoding rates.

2 Interpolation-Based Soft-Decision Decoding of Reed-Solomon codes

Encoding is performed with an evaluation-map encoder that evaluates a message polynomial of k message symbols at n evaluation symbols. The Koetter-Vardy algorithm consists of three steps: 1. **Soft-decision front end**: using soft information, derive positive integer weights (multiplicities) for each point (pair of evaluation symbol and possible transmitted symbol) proportional to the probabilities, 2. **Interpolation**: find a bivariate polynomial $P(x, y)$ with the smallest $(1, k-1)$-weighted degree that passes through all the points with the prescribed multiplicities, and 3. **Factorization**: find the list of all the factors of $P(x, y)$ of the form $y - f(x)$ with $\deg f(x) < k$. Of the three main components of the algorithm, the most computationally expensive one is the bivariate interpolation step [2].

3 Parallel Architecture for Interpolation

We propose a parallel architecture that efficiently implements the inherent parallelism in the required polynomial operations. Bivariate polynomials are written as univariate polynomials in y with coefficients that are themselves univariate polynomials in x. This enables an architecture with separate units for x and y operations. A processing element (PE) is assigned to each possible monomial in the polynomial. The x-calculations are carried out in independent x-processors while the y-calculations and top-level control are carried out in a y-processor).

The proposed *monomial-parallel* architecture updates the polynomials one at a time, but does each update in a parallel manner using L processing elements (PEs), where L is the maximum length of the polynomials. We take advantage of the fact that the length of the polynomial, L, is much larger than the number of polynomials, $d_y + 1$, where d_y is the maximum y-degree. Therefore, the degree of parallelism is high. Each PE consists of a GFADD and a GFMULT and the required storage for $d_y + 1$ monomial coefficients. The critical path is the delay of one Galois field multiplier, one Galois field adder, and one MUX.

3.1 Interconnection Network

The interconnection network embeds two topologies, a linear array and a binary tree, in a two-dimensional array

Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04)
0-7695-2230-0/04 $ 20.00 IEEE

FPGA	LUTS	% of FPGA
Virtex XCV2000E-6	27356 (FPGA0)	63%
23 MHz (measured)	26227 (FPGA1)	61%
	13175 (FPGA2)	30%
Virtex II XC2V8000-4	69518	74%
35 MHz (estimated)		

Table 1. The logic resources and clock frequency for the FPGA implementation.

of processors. Youn and Singh [3] propose a method to generate a regular two-dimensional VLSI layout that uses purely local connections, if diagonal connections are included. This scheme is more efficient than the classical H-tree layout in the case of large processing elements with narrow bus widths. Using this idea, a binary tree and linearly-connected array can be embedded into a two dimensional array of PEs by adding some extra connections. Many of the tree connections and linear connections are shared.

4 FPGA Implementation

We implemented the interpolation array processor on three Xilinx Virtex 2000E FPGAs on the TM3 FPGA prototyping system [4]. The logic resource requirements for each FPGA are tabulated in Table 1. The fastest clock frequency measured was 23 MHz. We also synthesized the design to a single Xilinx Virtex II 8000. The place-and-route software reported a maximum clock frequency of 35 MHz.

Table 2 summarizes the hardware requirements and throughput for a maximum multiplicity of $m = 4, 6, 8$. The above analysis is for the worst case, for the maximum possible number of iterations for a given maximum multiplicity m. However, in practice, the actual number of iterations might be lower. We ran simulations to measure the average number of cycles needed to find an interpolation polynomial for RS(255,239) code with maximum multiplicity m. The channel was AWGN and the value of E_b/N_0 was varied from 5 to 12 dB. Figure 1 plots the maximum and average throughput for an FPGA implementation clocked at 35 MHz. At $E_b/N_0 = 8$ dB, and $m = 4$, a throughput of 30 Mbps for an FPGA at 35 MHz can be achieved. For $E_b/N_0 = 8$ dB, and $m = 8$, a throughput of 4.5 Mbps is achieved.

5 Conclusions

We proposed a parallel array architecture for bivariate interpolation that realizes two different topologies, a linear array, and a binary tree, to match the communications patterns of the various polynomial operations. An interpolation engine was implemented using an FPGA prototyping system and was demonstrated at 23 MHz, corresponding to throughputs of between 10 and 15 Mbps. Using modern FPGAs, the processor could be integrated onto a single device and clocked at 35 MHz.

m	memory	GFMULTs	GFADDs	max. cycles
4	1.8 Kbytes	188	531	3947
6	4.5 Kbytes	386	1125	9694
8	9.1 Kbytes	655	1911	19665

Table 2. The hardware requirements and maximum number of cycles for the array processor for RS(255,239).

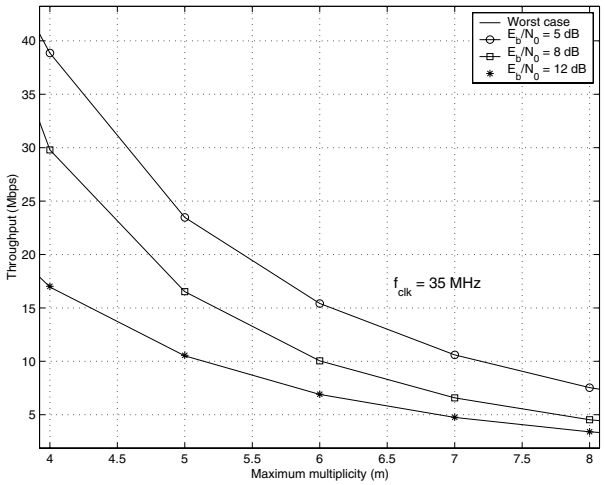

Figure 1. The worst case and average throughput as a function of the maximum multiplicity m for an FPGA implementation. The average case was measured on an AWGN channel.

References

[1] R. Koetter and A. Vardy. Algebraic soft-decision decoding of Reed-Solomon codes. *IEEE Transactions on Information Theory*, 49(11):2809–2825, November 2003.

[2] W. J. Gross, F. R. Kschischang, R. Koetter, and P. Gulak. Towards a VLSI architecture for interpolation-based soft-decision Reed-Solomon decoders. Accepted for publication in the *Journal of VLSI Signal Processing*, November 2003.

[3] H. Y. Youn and A. D. Singh. On implementing large binary tree architectures in VLSI and WSI. *IEEE Transactions on Computers*, 38(4):526–537, April 1989.

[4] `http://www.eecg.utoronto.ca/~tm3`.