Schematic Driven Layout of Reed Solomon Encoders

Kari Arave, John Canaris, Lowell Miles and Sterling Whitaker
NASA Space Engineering Research Center
University of New Mexico
2650 Yale SE, Suite # 101
Albuquerque, New Mexico 87106

Abstract – **Two Reed Solomon error correcting encoders are presented. Schematic driven layout tools were used to create the encoder layouts. Special consideration had to be given to the architecture and logic to provide scalability of the encoder designs. Knowledge gained from these projects was used to create a more flexible schematic driven layout system.**

1 Introduction

The layout of custom VLSI circuits frequently requires the generation of very regular, densely packed functional blocks. These blocks require extreme attention to area consumed as well as attention to local and global routing considerations. It is often appropriate to perform all routing inside the base cells of these sections. In this case, no routing channels are needed and all routing is done with *connection by abutment.* The complex wiring task is addressed by the layout designer during the planning and cell layout stage. After the base cells are layed out they must be placed together to form the larger functional block. As all interconnections are made by abutment, little mental effort is required at this stage.

This placement stage, in the layout of regularly structured blocks, is time consuming and tedious. A schematic driven, smart tiler, coupled with a schematic capture system which allows connections by abutment, provides a significant time savings in layout design.

The two Reed Solomon (RS) error correcting encoders presented in this paper were developed for Hewlett-Packard (HP) Disk Mechanism Division using a $0.75\mu m$ triple metal CMOS technology. These encoders were layed out using a schematic driven tiler and were used, by HP, as macrocells on HP designed custom chips. This paper will first present a general review of Reed Solomon encoding followed by a description of the two specific encoders designed for HP. The paper will conclude with a description of the the schematic driven layout and the tools used to create it.

2 Reed Solomon Encoders

A Reed Solomon (RS) code is a cyclic symbol error correcting code for correcting errors introduced into data during transmission through a communication channel [1]. Information is represented as m bit binary symbols forming a finite field $GF(2^m)$ containing 2^m-1 nonzero symbols. The elements of the field are specified by an irreducible, primitive polynomial $p(x)$.

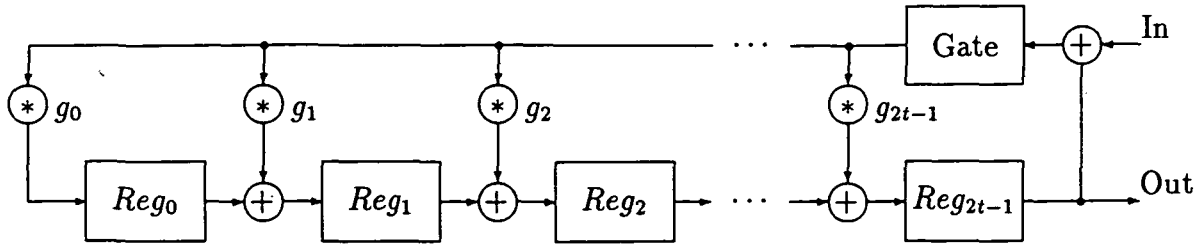

Figure 1: Parity generator block diagram.

A code block consists of $N \leq 2^m - 1$ symbols. Each code block includes both information symbols and parity symbols. In order to correct t symbol errors, $2t$ parity symbols are calculated and appended to a group of information symbols during an encoding process. An RS code is classified as a (N,k) code where k is the number of information symbols in a code block and N is the total number of information plus parity symbols. $N = 2^m - 1$ for a full length code. If $N < 2^m - 1$, the code is said to be shortened.

The RS code block can be defined as a polynomial $c(x)$, of order $N - 1$, formed from an order $k - 1$ information polynomial, $m(x)$, and an order $N - k$ generator polynomial, $g(x)$, as in Equation 1. A shift in time, due to the addition of the parity symbols, is represented by x^{2t}.

$$c(x) = x^{2t}m(x) + x^{2t}m(x) \bmod g(x). \tag{1}$$

Consequently, every valid code block is a multiple of the generator polynomial

$$g(x) = \prod_{i=s+1}^{s+2t} (x - \beta^i) = \sum_{j=0}^{2t} g_i x^i \tag{2}$$

where $\beta = \alpha^h$ is a primitive element of the field, s is an offset term and t is the error correcting ability of the code. A transmitted code block, $c(x)$, is modified by the introduction of errors in a noisy channel. The appended parity from the encoding process allows t such errors to be corrected during a decoding process.

An encoder can also represent data in a dual basis such that

$$[z_0, z_1, \ldots, z_{m-1}] = [u_{m-1}, u_{m-2}, \ldots, u_0]\, T \tag{3}$$

where $[z_0, z_1, \ldots, z_{m-1}]$ is the symbol represented by the dual basis, the symbol representing the normal basis is $[u_{m-1}, u_{m-2}, \ldots, u_0]$ and where T is an m by m transform matrix. Normal data can be derived from data represented in the dual basis using an inverse transform, T^{-1}.

A dual field is simply a different representation of the original field. The coefficients of $g(x)$ are linear operators. An operator O in the original representation of the field can be used in the dual representation by applying the following transform.

$$O_{dual} = TO_{original}T^{-1} \tag{4}$$

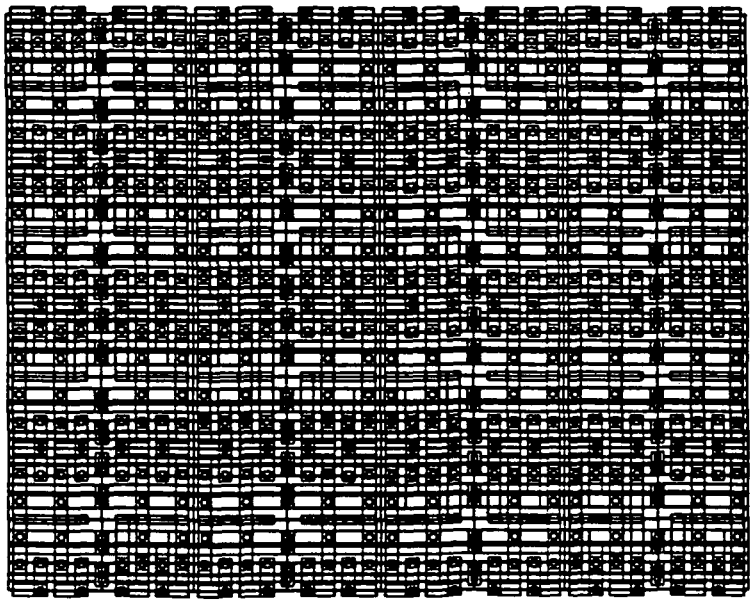

Figure 2: Layout of constant multiplier.

Figure 1 shows a block diagram of the logic required to perform the parity generation function. The parity generator organization can be viewed as a set of slices. Each slice consisting of a multiply/add structure and a register. Each multiplier multiplies data by a constant, g_i, which is a coefficient of the generator polynomial. Interleaving of two or more encoded messages allows higher burst error correction capabilities by spreading out the effect of burst errors over more than one codeword. The desired interleaving depth, I is controlled by the depth of the register shown in Figure 1. Each register is therefore an m-bit wide, I-bit deep shift register.

The two encoders designed for HP operated on 8-bit symbols ($m = 8$) forming a field GF(2^8). The first, RS1, corrected a single symbol error ($t = 1$) and had no interleaving ($I = 1$). The second, RS3, corrected three errors ($t = 3$) and had a variable interleave depth of $I = 2, 3, 5$ or 9. The code blocks were of variable length, $2t + 1 \leq N \leq 255$ where 255 is the full length code ($2^m - 1$). These encoders operated in the normal basis and required no data transformation. The primitive polynomials were identical, but the two used different generator polynomials since the correcting capabilities were different. Both were designed for a sustained 30 MHz (240 Mbit/sec) data rate and included circuitry in the parity generator to detect zeros.

3 Layout

One restriction on most schematic driven layout is the requirement that the schematic exactly match the layout. The schematic becomes a floor plan for the tiler. Rotations, mirrorings, hierarchy, intermediate node labels, port positions, names and locations of all cells must exactly match the layout. Since schematics are usually captured early in the design process, so that schematic driven simulation can be used to verify the logic design, physical layout

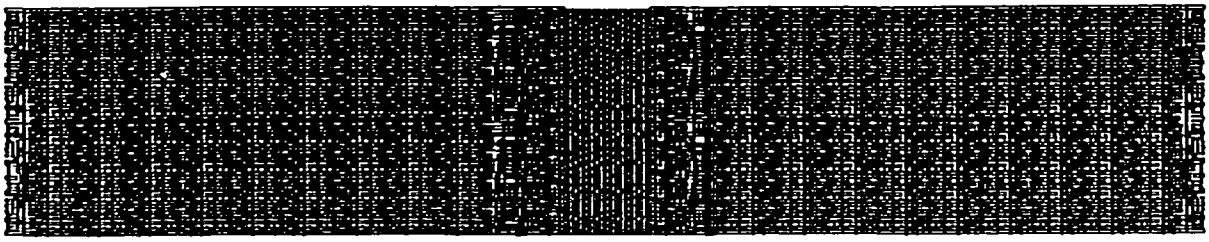

Figure 3: Layout of two adjacent RS3 slice details.

information is not available at the time schematics are originally captured. This results in additional schematic capture iterations, following the design phase, when schematics are redrawn to match the layout, and logic simulations are rerun to verify that no errors have been introduced during the modifications. To overcome these restrictions, tiling parameters were added to the schematic to avoid this rework and to improve the readability of the encoder schematics. A slice in the encoder parity generator will be used to illustrate this improvement.

A slice consists of a multiplier, adder and register. The multiplier cell is a precharged exclusive or (XOR) chain. Eight of these chains form a constant multiplier. The input data word is multiplied by a constant, g_i, programmed into the multiplier as XOR cells or interconnect (ZERO) cells. The XOR cell consists of 4 NMOS transistors. The ZERO cell is a modified XOR cell which acts as an interconnect block.

The multiplication constant can be programmed with a single mask layer defining the pattern of XOR and ZERO cells in the XOR chains. The final layout of a constant multiplier is shown in Figure 2. For maximum operating speed the XOR chain is precharged from both ends. The addition function is folded into the evaluate structure for the multiplier. The XOR cell was designed in layout to consume minimum area and the registers were designed to match the pitch of two XOR cells. Half the registers were placed above the multiplier and half below.

Since the registers are twice as wide as the XOR chain, the outputs of the columns in the multiplier alternate between top and bottom. The higher order nibble is output on one end of the constant multiplier and the lower order nibble on the opposite end. This requires the columns of the multiplier matrix to be rearranged such that the columns in the matrix are $[C_0C_4C_1C_5C_2C_6C_3C_7]$. Also, in order to avoid long interconnect runs between registers on the top and bottom to drive the adder inputs in the multiply/add structure, a second slice detail was drawn such that the top and bottom sections were reversed. These two slice details were then alternated in the parity core allowing connection of the adjacent slices by abutment. Figure 3 shows the final layout of two adjacent RS3 slices including routing between the slices. There is no interconnect required between any of the leaf cells. The entire structure is connected by abutment and thus is an ideal candidate for tiling.

Figure 4 is the logic diagram for the top of the slice drawn during logic design. Figure 5 is the same diagram modified to drive the tiler. Notice that the cell mirrorings and rotations cause a loss of readability and that the hierarchy changes hide the logical functionality,

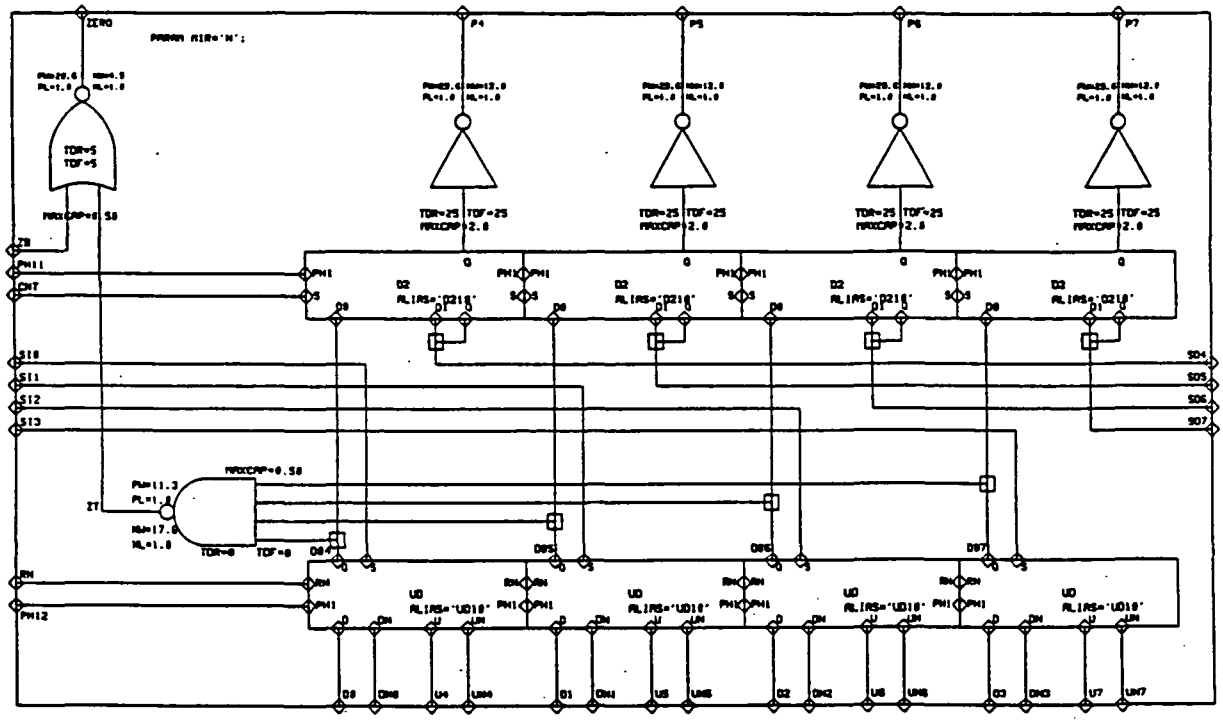

Figure 4: Original schematic diagram for top of RS1 slice.

however, the block can drive the tiler and labels are automatically placed on the artwork from the schematic. This makes labeling a one time manual task.

Figure 6 is the original logic diagram for the multiplier core. Figure 7 is the multiplier diagram modified to drive the tiler. The cell mirrorings and rotations again cause a loss of readability. The schematic entry program was then modified to allow tiling parameters to be added. This allowed the original logic diagram to drive the tiler directly without the need to redraw the schematic. Figure 8 is the original logic diagram modified at the time of layout with parameters to drive the tiler.

4 Acknowledgements

This research was supported in part by NASA under Space Engineering Research Grant NAGW-1406 and by the NSF under Research Initiation Grant MIP-9109618.

References

[1] G. C. Clark and J. B. Cain *Error Correcting Coding For Digital Communications*, New York NY, Plenum Press, 1981

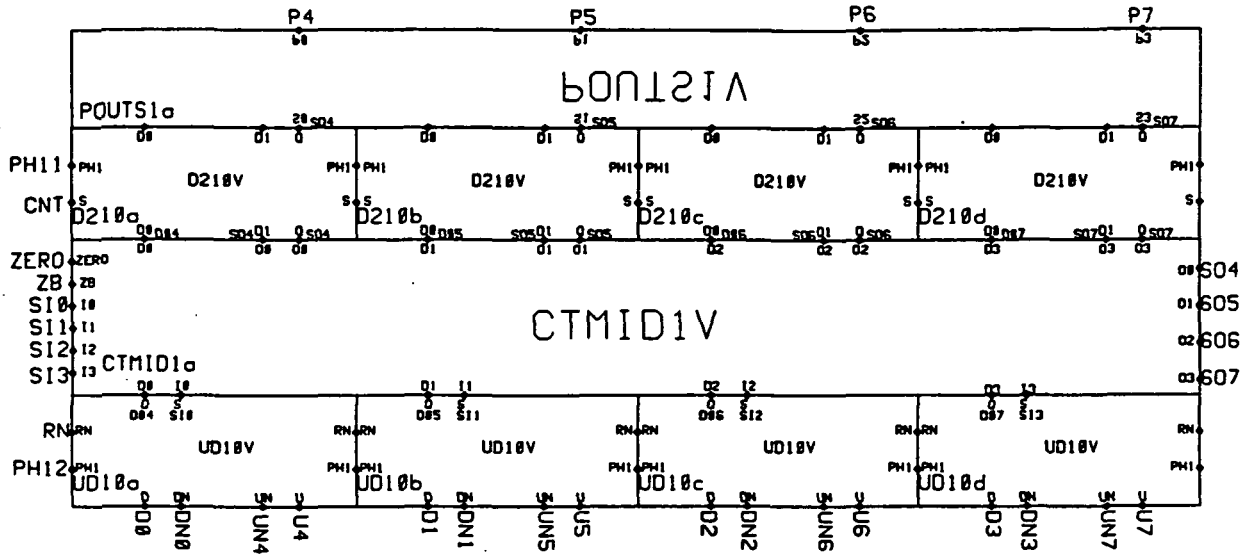

Figure 5: Schematic diagram for tiling.

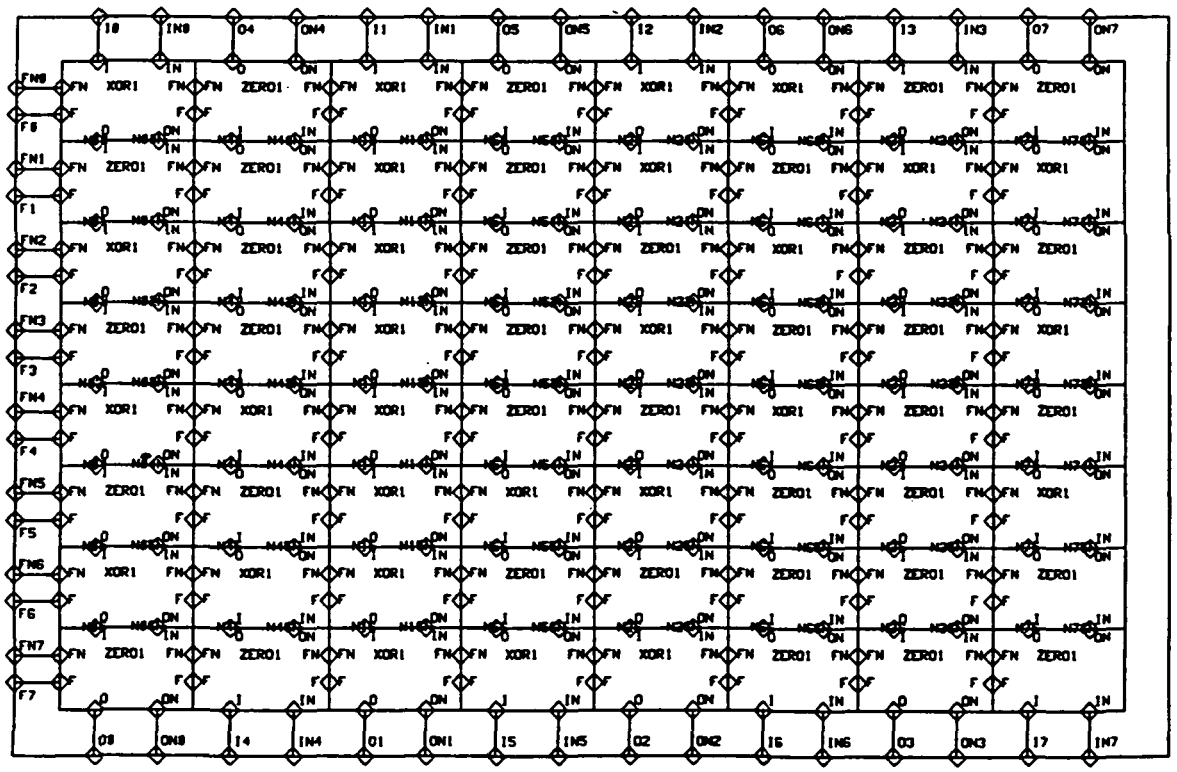

Figure 6: Original schematic diagram for multiplier core.

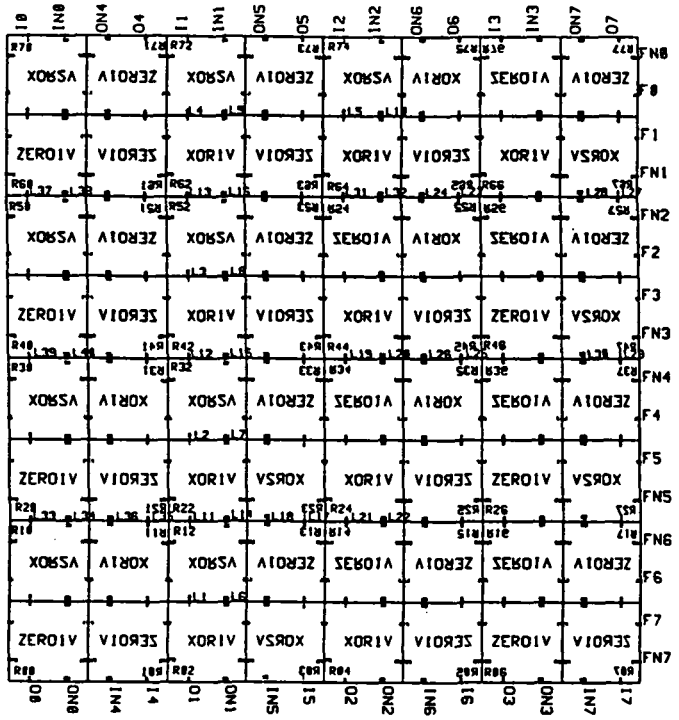

Figure 7: Multiplier diagram for tiling.

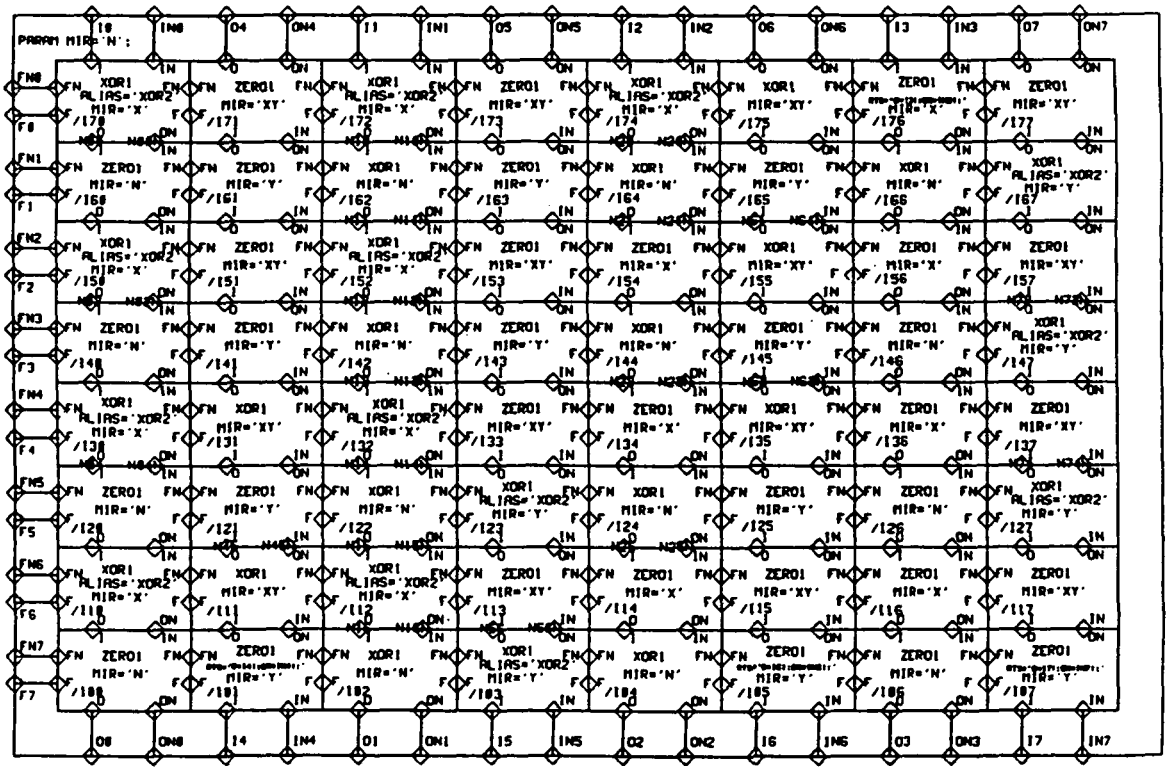

Figure 8: Parameterized schematic diagram for multiplier core.