MATHSTAR™

Fast Silicon, Faster Solutions

RAID 6 Q Encoder Implementation Using MathStar's FPOA Technology

Silicon Objects Application Note **041210**

Introduction

RAID 6 spreads storage data across multiple hard drives for redundancy. The storage patterns and encodings allow up to two drives to be lost, yet all storage data can be recovered. (In contrast, RAID 5 allows only one drive to be lost.)

The MathStar FPOA technology supports 16 Gb/s RAID 6 Q encode using six ALU objects. This represents 1.5% of the objects of either of two available medium-sized FPOA chips, SOA13C40-01 and SOA13D40-01.

Algorithm

Two redundancy codes are calculated from two or more data drives. The redundancy codes are stored on drives other than where the data is stored (else the reliability benefit is forfeited). Thus the minimum number of drives for RAID 6 is four drives.

We'll choose to span only two data drives for any particular stripe of data - two drives for data, two drives for redundancy codes. (Spanning more than two data drives is possible. This increases data efficiency (amount of data stored vs. amount of redundancy codes), but reduces reliability and performance. Interacting with more drives increases the likelihood of failure for any given stripe of data, and also requires more drives to be read in order to write, reconstruct, or verify data.) A RAID 6 drive array can consist of any number of drives; but (for this implementation) any particular stripe relates data sectors across precisely four of the drives. Array sizes that are not multiples of four drives are accommodated by distributing the stripes across the total number of drives with a fair hash.

One redundancy code, often called "p" (for parity), is the logical XOR of the two data drives. If only one drive is lost, this code can construct the missing drive (because XOR is self-inverting, the lost drive is the XOR of the two remaining drives). The FPOA ALU directly supports this XOR operation at a rate of 16 Gb/s with a single instruction. The other redundancy code, often called "q", is a Reed-Solomon error-correcting code, which requires more effort to generate. However, this code is what allows two drives to be lost (you can reconstruct two data drives from knowing only p and q).

Implementation

Q Calculation

Call the two data drives dx and dy. Calculate q = dy*g + dx in Galois field GF(2^8) with irreducible polynomial 0x11D and g=2. A single ALU can encode 16 bits (two 8-bit values) in six instructions:

- Copy the dy data into a working register.
- Test for carry out of the lower byte.
- Shift left one (which also tests for carry out of the upper byte).
- Compensate for upper byte carry.
- Compensate for lower byte carry.
- XOR the dx data and update the output register.

Pipeline

Six ALUs are ganged to provide full-rate processing. The full-rate dx and dy data are presented to two six-deep parallel pipelines. The six ALUs operate simultaneously, sampling their data out of their time slot in the pipeline. Six clock cycles pass, allowing the data pipelines to fill with fresh data. On the sixth clock cycle, the six ALUs set their outputs and acquire new inputs. The entire pipeline is replaced with the ALU output values. The forward movement of the pipeline pushes out the results as new data inputs are marched in. The net effect is that full-rate inputs enter the six-ALU pipeline, and full-rate outputs exit.

Design in Visual Elite

The single-ALU encoding program is in the block diagram calc_q_alu. It encodes 16 bits every 6 clock cycles, and is tested in calc_q_alu_test.

The pipeline combines six ALUs together in calc_q_pipe. Mux objects manage the pipeline content, choosing to step the pipeline forward five out of six cycles, replacing the pipeline content with fresh ALU results in one out of five cycles. Each ALU sends a control signal to the closest Mux object to establish proper timing. The pipeline encodes 16 bits every clock cycle, and is tested in calc_q_pipe_test.

Map to FPOA

The design calc_q_pipe_test is mapped into the FPOA chip SOA13D40-01. Six adjacent ALUs on the north-most row of the array implement the six-ALU encoding pipeline. The results are routed to GPIO object gpio_w_13 for external visibility.

Extensibility

Note that if redundancy codes across more than two data drives are desired, the calculation of q is recursive. For example, for three drives, q = (dz*g + dy)*g + dx. This can be calculated by running the calculation twice deep - i.e., Q(Q(dz,dy),dx). The impact of choosing three data drives (instead of two) would be a storage efficiency improvement to 60% (from 50%), a vulnerability increase[1], and a performance decrease for data writes (requiring two reads instead of one to fetch the other data in the stripe).

Results

The described encoder has been modeled, mapped, and executed on a SOA13D40-01 chip. Arbitrary sequences and all Galois Field corner conditions (all combinations of the upper- and lower-byte carries) were tested.

Operating at 1 GHz frequency with 1.2V supply, the six ALUs will consume approximately 0.23 amps, or 0.27 watts. The pipeline calculates 16 bits per clock cycle, yielding functional power efficiency of 16 Gb / 0.27 W = 58 Gb/W.

1. For an array of n drives, changing the stripe size from four to five drives increases the likelihood of a single drive failure affecting a given stripe to 1-(n-5)/n (from 1-(n-4)/n). For example, for an array of six drives, losing a single drive causes 83% of the data to have to be reverse engineered (instead of 67%); for an array of ten drives, losing a single drive affects 50% of the data (instead of 40%).

Fast Silicon, Faster Solutions