リードソロモンCODEC 演算エンジン　設計仕様書

琉球大学工学部情報工学科　和田　知久

[0]　はじめに

今回、RAID (Redundant Array of Inexpensive Disks)で用いられる、データのエンコーダーおよびデコーダを設計する。このエンコーダ／デコーダではリードソロモンという聞きなれない符号化を用いて、データよりチエックサムという冗長データを生成し、これによりデータおよびチエックサムの一部のデータが失われても元のデータをデコーダにて復元することができる。

リードソロモン符号はコンパクトディスクやデジタル通信でのエラー訂正に用いられている符号であり、数学的には、カロア体演算を取り扱うが、そのような符号理論や数学の知識なしに設計できるように以下の仕様書は工夫されている。聞きなれない言葉も多いが、設計すべき内容は比較的単純であるので、自信をもって課題に取り組んでいただきたい。

[1]　RS-RAID システムオーバービュー

今回は、Prof. James S. Plank 氏の"A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like System" に述べられている RAID (Redundant Array of Inexpensive Disks)で用いられる Reed Solomon CODEC のエンジンであるガロア体での行列乗算器を HDL を用いて設計する。RAID とは文字どうり低価格なハードディスクを冗長に装備し、データを分散して記憶することで、一部のハードディスクに故障が生じてもシステムの正常動作を保証するものである。
RAID システムの概要を図１に示す。D1, D2, D3 はデータであり、それぞれデータデバイスに記憶される。ENCODER はチェックサム C1, C2, C3 を生成し、チェックサムデバイスに記憶される。図１では D2, D3, C3 を記憶するディスクが故障した場合であり、D1, C1, C2 より元データである D1, D2, D3 を再生成する。

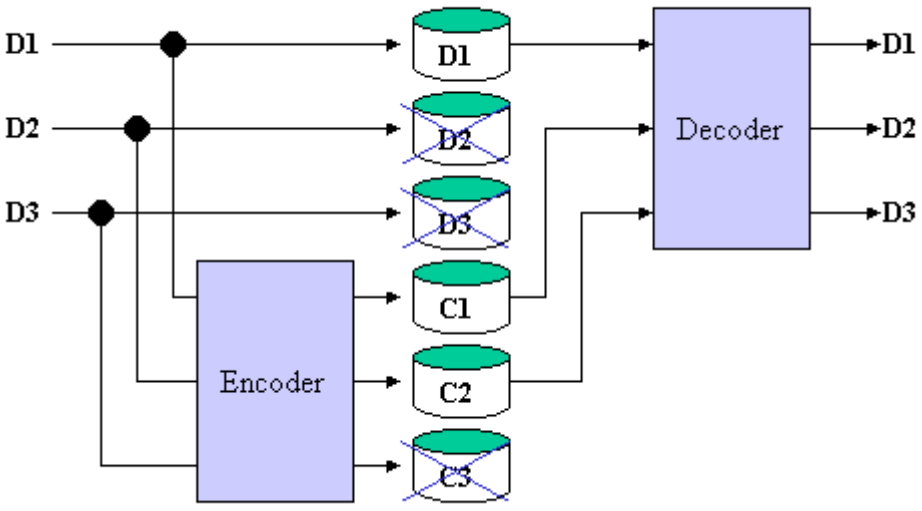

図１．RAID システム

[2]　ENCODER と DECODER

ENCODER と DECODR はどちらも、下記(1),(2)式の行列演算を行う。したがって、同一のハードウエアで構成できる。式 (1) の行列は規則的に自然数の指数乗の要素からなる行列であり、Vandermonde 行列と呼ばれるものである。また、ここでの数学はカロア体での計算を行う必要がある。例として右下の行列要素３の２乗が９ではなく５になっていることに注目してほしい。

式（２）の行列は[d1, c1, c2]から[d1, d2, d3]の再生行列であり、この求め方は上記英語論文(PDF, 2MB)、およびその和訳に別途示す。

$$\begin{bmatrix} c1 \\ c2 \\ c3 \end{bmatrix} = \begin{bmatrix} 1^0 & 2^0 & 3^0 \\ 1^1 & 2^1 & 3^1 \\ 1^2 & 2^2 & 3^2 \end{bmatrix} \begin{bmatrix} d1 \\ d2 \\ d3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 5 \end{bmatrix} \begin{bmatrix} d1 \\ d2 \\ d3 \end{bmatrix} \quad \cdots \quad (1)$$

$$\begin{bmatrix} d1 \\ d2 \\ d3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 1 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} d1 \\ c1 \\ c2 \end{bmatrix} \quad \cdots \quad (2)$$

図２．基本式

[3]　ガロア体演算

ガロア体は有限な数の要素からなる集合であり、各要素間の四則演算の結果も要素の一つとなる。2 の w 乗要素ガロア体は$GF(2^w)$と示され、１から（２のw乗-１）までの整数要素から構成される。

- 加算と減算は単純で XOR となる。W=4 の場合、以下のようになる。

$$11+7 = 1011 \oplus 0111 = 1100 = 12$$
$$11-7 = 1011 \oplus 0111 = 1100 = 12$$

- 掛算と割算はもっと複雑であり、２つの logarithm テーブルを用いて計算される。この２つのテーブルは"gflog"と"gfilog"であり、gflog はインデックスをガロア体での logarithm に対応させ、gfilog は inverse logarithm に対応させる。また、gflog[gfilog[i]] = i と gfilog[glog[i]] = i が成り立つ。２つの数の掛算はそれぞれの gflog の結果に対して通常の加算を行い、それを gfilog することで得られる。割算は通常加算の代わりに、通常の引き算を行う。通常加算の結果、それが（２のw乗-１）に等しいかもしくはそれを超えると、（２のw乗-１）を引き算し、結果を０から（２のw乗-２）の範囲にする。同様に、通常の引き算でその結果が-1 以下になれば、（２のw乗-１）を加算して、結果を０から（２のw乗-２）の範囲にする。
- 以下に W=4 の場合の例を示す。また、テーブル１に W=4 の Logarithm テーブルを示す。

```
3*7 = gfilog[gflog[3]+gflog[7]]=gfilog[4+10]=gfilog[14]= 9
13*10 = gfilog[gflog[13]+gflog[10]]=gfilog[13+9]=gfilog[7]= 11
13/10 = gfilog[gflog[13]-gflog[10]]=gfilog[13-9]=gfilog[4]= 3
3/7 = gfilog[gflog[3]-gflog[7]]=gfilog[4-10]=gfilog[9]= 14
```

I	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
gflog[i]	-	0	1	4	2	8	5	10	3	14	9	7	6	13	11	12
gfilog[i]	1	2	4	8	3	6	12	11	5	10	7	14	15	13	9	-

テーブル１．Logarithm テーブル　(w=4)

上記英語論文の Figure3、Figure4 に C コードでの実現方法が示されており、参考になる。特に 0 の取り扱いは例外であり注意を要する。

[4] 外部仕様

W=4 に対応するガロア体の行列乗算器を設計する。図３にシンボルを示す。この乗算器を ENCODER にも DECODER にも共用するために、各行列要素はアドレス信号(ADD)に指定された要素に係数(ELE)を書き込む。この書き込みはクロック信号(CLK)の立ち上がりエッジで書き込みコントロール信号(WE)をアサートすることで行われる。各要素とアドレスの関係は図４に示す。

その後、３入力（それぞれ４ビット）IN1,IN2,IN3 を与え、GET のアサートされたクロック信号の立ち上がりで３入力がガロア行列乗算器に取り込まれで演算が開始される。

演算は任意のサイクル数で行う**（設計者の設計の自由とする）**。一般的には、１サイクルで実行する場合素子数が増加し、複数サイクルで実行すれば素子数を減らせるトレードオフがある。但し、出力信号 OUT1, OUT2, OUT3 を外部デバイスに取り込ませるために、１サイクル期間アサートされる DONE 信号を出力信号確定時にアサートする。

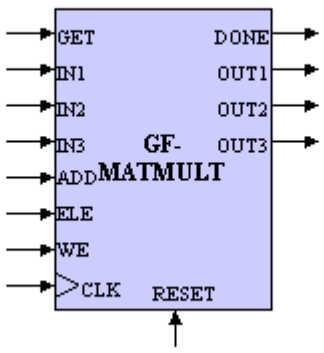

図３．ガロア行列乗算器のシンボル

$$\begin{bmatrix} OUT1 \\ OUT2 \\ OUT3 \end{bmatrix} = \begin{bmatrix} ELE[Add=0] & ELE[Add=1] & ELE[Add=2] \\ ELE[Add=3] & ELE[Add=4] & ELE[Add=5] \\ ELE[Add=6] & ELE[Add=7] & ELE[Add=8] \end{bmatrix} \begin{bmatrix} IN1 \\ IN2 \\ IN3 \end{bmatrix}$$

図４．ガロア行列乗算

[5]　ピンアサイン

各ピンのビット幅／機能等をテーブル２に示す。

信号名	入出力	ビット幅	説明	信号名	入出力	ビット幅	説明
GET	IN	1	H で入力データを取込み計算開始	CLK	IN	1	クロック信号
IN1	IN	4	入力データ１	RESET	IN	1	H でリセット
IN2	IN	4	入力データ 2	DONE	OUT	1	H で出力データ確定
IN3	IN	4	入力データ 3	OUT1	OUT	4	出力データ１
ADD	IN	4	行列要素の位置を指定	OUT2	OUT	4	出力データ 2
ELE	IN	4	行列要素入力	OUT3	OUT	4	出力データ 3
WE	IN	1	H で行列要素書き込み				

テーブル２．Logarithm テーブル　(w=4)

[6]　システム動作検証

図１に示す RS-RAID システムの動作検証のために以下図５のシステムモデルを用いて、検証を行う。

- EWE は ENCODER 用の WE、DWE は DECODER 用の WE であり、ADD/ELE は共有されている。EWE/DWE を用いて ENCODER/DECODER に式（1）、（2）に示される適切な行列要素を設定する。
- D1IN/D2IN/D3IN に適当な入力を入れ、内部信号 C1/C2/C3/DONE 等をモニターし、最終的に D1OUT/D2OUT/D3OUT に再生された同じデータが出力されることを観測する。
- 図６／図７にセットアップシーケンスと動作シーケンスの動作波形例を示す。

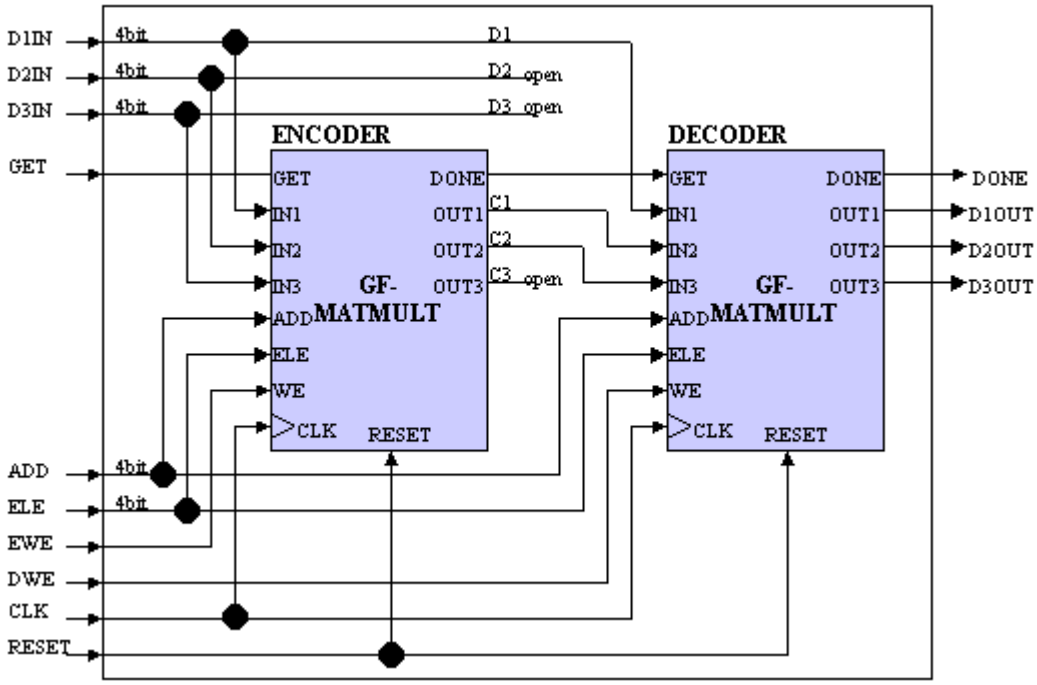

図５．システムモデル

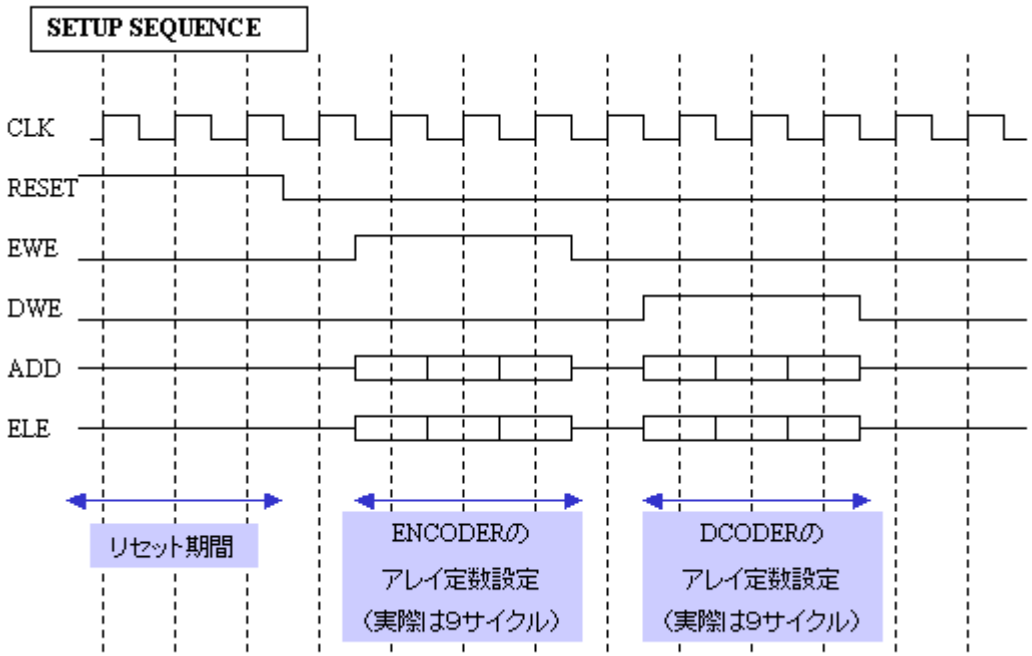

図６．セットアップシーケンス

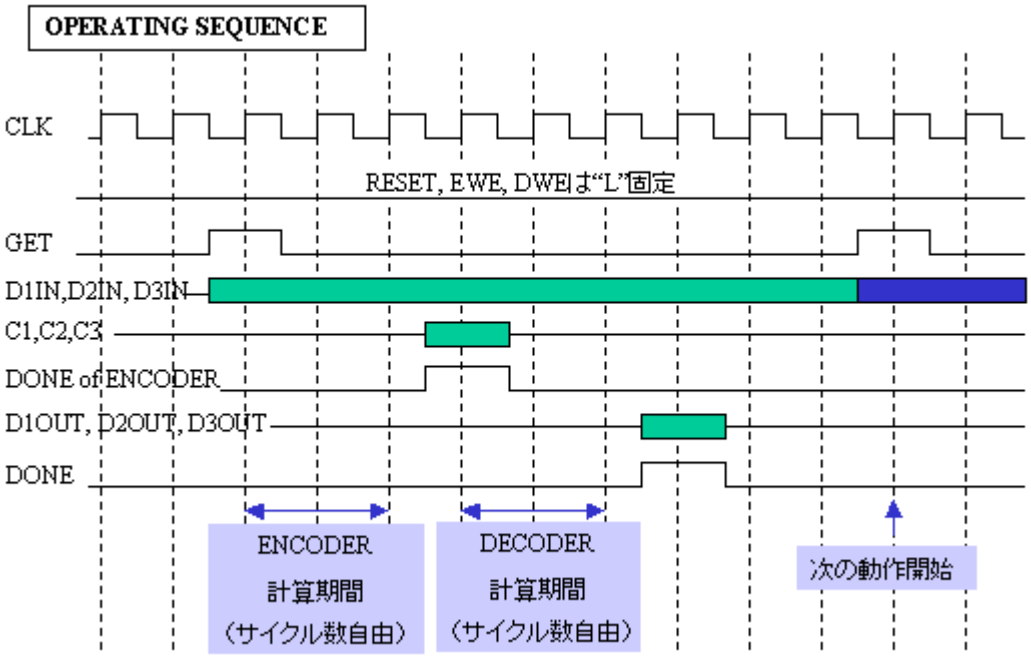

図７．動作シーケンス

[7]　スピードの測定単位

琉球大学以外からの参加の場合、同一のシノプシスデザインコンパイラの論理合成用ライブラリを使用することが困難であるので、以下のような多段 EXOR 回路を例に従って合成し最適化して頂き、その１段あたりの遅延時間を単位時間としてスピードの単位とする。

多段 EXOR 回路の VHDL ソース例：５０入力の EXOR 回路

合成語の回路図　PDF,　PS

この例では6段の EXOR 段が合成され、クリティカルパス遅延は report_timing コマンドにより7.１７であったので、７．１７/６＝１．１９５を単位（UNIT）とする。

- 例えば、ある遅延が２０ならば、２０/１．１９５＝**１７．７４UNIT 遅延**とする。
- 琉大情報のライブラリ使用者は上記値で換算で換算すればよい。

ちなみに面積は report_timing コマンドの total cell area にある。

[8]　レポート

レポートには以下の内容を含めること。また、ページ数を少なめにコンパクトにまとめること。

表紙	1	代表者の氏名、チーム名、大学院修士/大学学部生/高専生の区別
	2	共同設計者（最高３名まで）全員の名前、学籍番号、学年、学校名、住所、電話、email 等連絡先
	3	全員の **T シャツの希望サイズ**
内容	1	設計した回路ブロックの構成説明（ブロック図と説明、**特にガロア体乗算器の回路図や構成**）
	2	設計した回路ブロックの動作説明（何サイクルで出力がでるか？など）
	3	工夫した点、オリジナリティを出した点（アピールが重要！）
	4	クリティカルパスのスピード、論理合成後の回路規模
	5	VHDL もしくは Verilog のコード
	6	正常動作している VHDL/Verilog シミュレーション波形
	7	その他自由意見など

紙でのコピー３部を下記へ送付のこと：

和田　知久　（わだ　ともひさ）
903-0213　沖縄県西原町字千原１番地　琉球大学　工学部　情報工学科
PHONE: 098-895-8713

締め切りは 2000 年２月１８日（金）必着です。

[9]　審査のポイント

- 速度、回路規模だけでなく、アーキテクチャのユニークさ、アイデア、面白さを十分考慮して審査します。
  （ちゃんとアピールしてね！）

- 大学院修士、学部生、高専生のレベルに応じて審査をします。

仕様書に従ってまじめに作るのも良いが、オモロイアイデアを歓迎します。他人と違ったことをしよう！

仕様の部分変更など、柔軟に受け付けます。

ENJOY　HDL！　沖縄で会おう！