第3章

総当たり方式による エラー訂正回路の設計

――Design Wave設計コンテスト 2006 Student部門 第1位

チーム マッハ255（若林秀明）

Design Wave設計コンテスト2006のStudent部門 第1位の設計を紹介する．最初，筆者はエラー数の削減を目ざしていたが，なかなか良いアルゴリズムが見つからなかった．そこで，面積と速度の最適化へと方針転換しようとしたときに理想的な信号と総当たりで比較し，もっとも近いものを選ぶ方式（最尤復号法）を思いついた．回路設計では，はじめに最尤復号法のアルゴリズムどおりの設計を行い，比較的大きなブロックを削減していった．なお，本稿で紹介した回路の設計データは本誌のWebサイト（http://www.cqpub.co.jp/dwm/）からダウンロードできる．（編集部）

私が所属している研究室では，ハードウェアを設計して数値シミュレーションを高速化する研究などを行っています．研究の中では，HDLを用いて論理回路を設計し，FPGAに実装して動作させることもあります．

研究室では，2年前からこの設計コンテストに参加して，回路設計の勉強と力試しをしてきました．今年もコンテストを通して回路設計について学びたいと思いました．そして，何よりも1次審査を通過して沖縄での発表会を楽しみたいと思いました．そこで，課題のエラー訂正回路の設計にチャレンジすることにしました．

1．復号方式を考える

まず，設計仕様書(1)に示されていた繰り返し型デコーダ（エラー訂正回路）をそのままVHDLで記述してみました．符号の復号についてはなじみがなかったので，はじめは設計仕様書の内容を難しく感じました．しかし，「仕様書の回路例のとおりに作って実際に動かしてみれば，課題の理解が深まるだろう」と思い，設計を開始したのです．設計仕様書どおりのエラー訂正回路を記述し，シミュレーションしてみると，8クロックごとに復号結果が出力されていることが確認できました．

次に，この復号結果が正しいかどうかを検証するため，設計仕様書と同じ方法で復号を行うプログラムをC言語で作成しました．そして，設計した回路とプログラムで復号できたビット数が同じになり，正しく回路を設計できたことを確認しました．

その後，この回路をどのようにくふうしていくか考えました．良いエラー訂正回路の条件としては，コンパクトで，高速に動作し，エラー訂正能力が高いといったことなどが挙げられます．この中で，エラー訂正能力をより向上させるのは難しいと思いました．復号アルゴリズムの根本的なところを改良する必要があるからです．その一方で，今回のコンテストではだれも作らないようなおもしろい回路を作ってほかの人を驚かせてみたいとも思っていました．そこで，今回はあえてエラー訂正能力の向上を目ざすことにしました．

● エラー数の削減を目ざす

エラー数を減らす方法として，まず，外部値を計算するときに近似式を使わないことを考えました．設計仕様書の回路では，外部値の計算において，

KeyWord 最尤復号法，エラー訂正，2次元積符号，BPSK信号，確率，2乗器，アキュムレータ，比較回路，FPGA

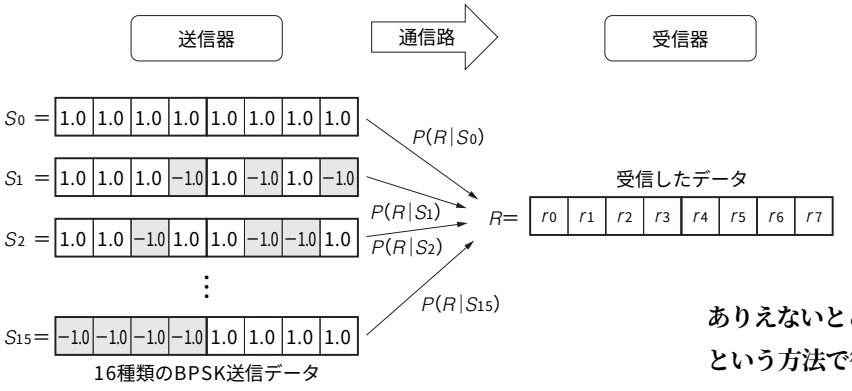

図1
復号アルゴリズムの概要
情報ビットは4ビットのため，送信器から送出される信号は16通り．そこで受信したデータにいちばん似ているものが送信されたとみなして復号する．

$$\ln\left(\frac{1+\exp(A+B)}{\exp(A)+\exp(B)}\right)\approx \mathrm{sgn}(A\cdot B)\min\left(|A|,|B|\right) \quad \cdots(1)$$

の近似を行っています．そこで，この近似を行わずに外部値を計算するようにC言語のプログラムを変更してみました．しかし，変更後も，課題の$S/N=0$dBのデータを復号したときのエラー数は700前後となり，変更前の735とそれほど変わらないことがわかりました．

また，繰り返し回数をより多くすることでエラー数が減るかどうかも試してみました．しかし，この場合もエラー数は700前後となり，思ったほどエラー数が減らないことがわかりました．このときは，がんばればエラー数が300ぐらいまで減らせるのではないか，と何の根拠もなしに思っていたのです．

その後，思いつくままにいろいろな方法を試してみました．例えば，行パリティと列パリティを使う順番を変えたり，0に近い受信値を強制的に0にして復号してみました．さらに，繰り返すごとに情報ビットのエラーが訂正されていくのと同じように，パリティの受信値も繰り返すごとに更新できないかと考えてみました．しかし，やはりエラー数は大きく減らせませんでした．図書館で符号理論の本を探してみましたが，今回のような2次元積符号の復号について詳しく解説している本は見つけられませんでした．

● 行き詰まりつつあったときにひらめく

どうやってもエラー数を減らすことができないので，行き詰まったような感じになりました．今回のコンテストは，エラー数を保ったまま，いかにして小さく高速な回路を作るかの勝負になりそうだな，と感じ始めていました．

そんな中，ふと，あるビットを‘0’か‘1’かどちらかに仮定して復号を行い，パリティに矛盾があったり確率的にありえないところがあったら最初の仮定がまちがっていた，という方法で復号できるのではないかと思いました．さらに，どうせなら送信情報を“0000”から“1111”までのすべての場合について仮定してみて，いちばん良く当てはまるものに復号すればよいのではないかと思いました．

この方法でほんとうに復号できるかどうか，最初はわかりませんでした．しかしその後，設計仕様書に記載されている式を参考にしてよく考えてみると，どうやら復号できそうだとわかりました．このアイデアでいこう，そう思い，さっそく回路設計に取りかかりました．

2. 復号アルゴリズム

図1に，今回提案する復号アルゴリズムの概要を示します．今回復号する2次元積符号では，符号内の情報ビットは4ビットなので，送信器から送出されるBPSK（binary phase shift keying）信号系列は$2^4=16$通りだけです．この16通りのBPSK信号系列を，

$$S_j=(s_{j0},\ s_{j1},\ \cdots,\ s_{j7})\ (j=0,\ 1,\ \cdots,\ 15)$$

で表すことにします．また，受信器が受信した信号系列を，

$$R=(r_0,\ r_1,\ \cdots,\ r_7)$$

で表すことにします．

通信路では，送信された信号にノイズが加えられます．そのため，受信値は送信値とは若干異なる値になります．しかし，元の送信データはS_0～S_{15}の16個のうちのどれかであるので，受信器側では，S_0～S_{15}のうちRにいちばん似ているものが送信されたとみなして復号することにします．このとき，パリティの行方向や列方向といったことは考えずに，単純に8個の受信値がどのBPSK送信系列と似ているかを考えて復号します．

この復号法は，確率の考えかたを使って次のように述べることもできます．まず，16個すべてのS_jに対して，S_jを