QRコードのデコードソフトウェアの開発及び高速化に関する検討

西本　篤生[*1]・世古　　忠

Development of QR Code Decode Software and Consideration of Speed Up Method

Atsuki Nishimoto and Tadashi Seko

In this paper we have newly developed QR code decode software using C language and we have measured CPU time of each functions of the software. As the experimental results, it shows that the error correction part takes 30% ～ 60% time of the software. Therefore we propose the speed up method of the error correction function to implement it using FPGA and we consider the experimental results.

1．研究目的

2 次元コードの 1 種である QR（Quick Response）コード[1,2] には様々な使用用途があり今後，様々なところで更に利用されるものと思われる．QR コード処理用のソフトウェアをハードウェア化することで処理の高速化や機器組み込みの専用システムを作成することができる．

本研究では，デコード処理のハードウェア化を行うために，初めに QR コードデコード処理のためのアルゴリズムを解析し，C 言語を用いて QR コードデコードソフトウェアを開発する．そして，各部の処理時間の評価を行い，処理時間のかかっている部分を明らかにする．次に，C 言語による記述を基にして，ハードウェア記述言語 VHDL を用いて QR コードのデコード処理アルゴリズムを記述し，ハードウェア化することによってデコード処理の高速化を図る．

2．QR コード

2.1 QR コードの構造

QR コードは，株式会社デンソーによって開発された二次元コードである．これは，「二次元コードシンボル－QR コード－基本仕様（JIS X 0510）」[1] として日本工業規格により制定されている．二次元コードは，バーコードなどの一次元コードに比べ，水平，垂直の 2 方向に情報を持つため数十倍から数百倍の情報量を扱うことができる．

QR コードには，大きさの小さい順に 1 型から 40 型まで 40 種類の型番がある．型番が増加すると格納できるデータの最大量も増加するので，QR コードに格納するデータの量にあわせ型番を選択する必要がある．

QR コードの構造図を図 1 に示す．QR コードは，QR コードの位置，方向を認識する位置検出パターン，モジュールの座標を認識するタイミングパターン，全体の歪みを補正する位置合せパターン，誤り訂正レベル，マスクパターンに関する情報である形式情報，QR コードの大きさを表す情報である型番情報，データ及び誤り訂正コード語により構成されている．

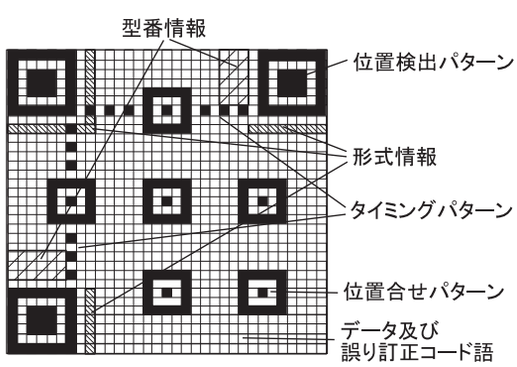

図 1 QR コードの構造図

図 2 に QR コードデコード処理の手順を示す．まず，形式情報の復号と型番の決定を行い QR コードのデータ部分の復号に必要な情報を得る．そして，データ部分のマスク処理を解除し，データ及び誤り訂正符号を復元し，誤りがある場合，データ部分の誤り訂正を行う．誤りを訂正したらデータを復号する．

[*1] 電子情報工学専攻 2 年

2.2　QR コードのデコード手順

QR コードをデコードする手順を図 2 に示す．以下で各処理の概要を説明する．なお各処理の手順は，文献 [1] で制定されている手順に従っている．まず，形式情報の復号と型番の決定を行い QR コードのデータ領域の復号に必要な情報を得る．そして，データ領域のマスク処理を解除し，データ及び誤り訂正符号を復元し，データ領域の誤り訂正を行う．誤りを訂正したら符号化の際に使用されたモードに基づいてデータを復号する．

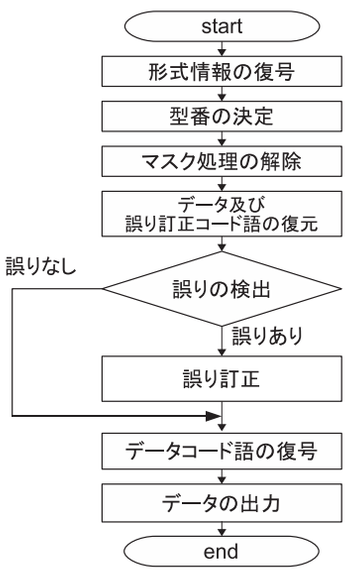

図 2　QR コードのデコード手順

2.2.1　形式情報の復号

‘ 0 ’ 及び ‘ 1 ’ のビット列として認識された QR コードから，形式情報のビット列を取得する．次に取得したビット列をマスクパターン 101010000010010 と XOR 演算を行い，形式情報のマスク処理を解除する．そしてマスク処理を解除した形式情報のビット列に誤りがある場合，誤り訂正を行い QR コードに適用されている誤り訂正レベル指示子及びマスクパターン参照子を取得する．誤り訂正レベルには L, M, Q, H の 4 段階があり，復元能力が高いほど，多くの誤り訂正符号が必要となる．具体的には，L が約 7%，M が約 15%，Q が約 25%，H が約 30% の誤り訂正能力を持っている．

2.2.2　型番の決定

QR コードの位置検出パターンの中心間の距離 D 及び 2 つのパターンの幅（W_L と W_R）を求める．次に QR コードの公称 X 寸法を計算する．

$$X = (W_L + W_R) / 14 \qquad (1)$$

そして QR コードの仮の型番 V を求める．

$$V = (D / X) - 10 / 4 \qquad (2)$$

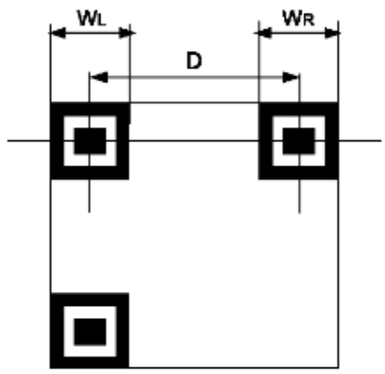

図 3　中心間の距離と位置検出パターンの幅

このとき仮の型番 V が 6 以下の場合，この値を型番として採用する．仮の型番 V が 7 以上の場合は，QR コードから型番情報のビット列を取得する．型番情報に用いる 34 種類のビット列と得られたビット列を 1 つずつ比較して，3 ビット以下の違いであれば，比較した型番の情報ビット列に対応する型番を採用する．3 ビット以上なら仮の型番 V を採用する．

2.2.3　マスク処理の解除

QR コードのデータ部分には，位置検出パターンなどと同じパターンを作成しないようにマスク処理が行われている．マスクパターンは複数あり，形式情報で得られたマスクパターンを用いて，符号化領域のビットパターンと XOR 演算し，マスク処理を解除する．

2.2.4　データおよび誤り訂正コード語の復元

データ及び誤り訂正コード語は，8 ビットを 1 ブロックとし図 4 に示すように QR コードの右下から配置され，領域の上側又は下側の境界に到達すると左側に配置され，配置される方向が反転する．そのため，1 ブロックずつ右下から順番にデータ及び誤り訂正コード語を取得する．

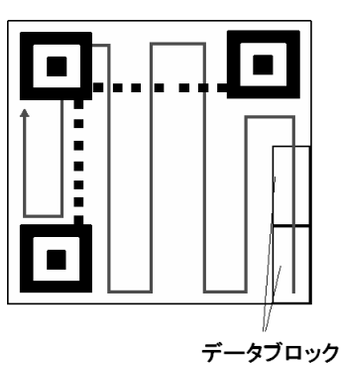

図 4　データブロックの配置

2.2.5 誤り訂正

取得したデータ及び誤り訂正コード語に誤りがあるか否か検出し，誤りが検出された場合は誤りを訂正する．

誤り訂正処理はシンドローム計算，ユークリッド処理，チェン検索の 3 つに分けられる．これらの処理については第 4 章で説明する．

2.2.6 データコード語の復号

誤り訂正を行ったデータ及び誤り訂正コード語の先頭に配置されているモード指示子と文字数指示子を取得する．そこから復号するモードと取得する文字数を決定し，データコード語を取得し，セグメントに分割する．そして，各モードに基づいてデータ文字を復号し，結果を出力する．

3．デコードソフトウェアの開発

3.1 デコードソフトウェアの構成

2.2 で述べた QR コードの復号手順に基づき QR コードデコードソフトウェアを C 言語を用いて開発した．開発環境には，Microsoft Visual C++ 6.0 を用いて開発を行い，Dell(CPU:Pentium(R)4, Memory:1.00Gbytes,OS:Windows XP Professional Edition) 上で実装した．

開発した QR コードデコードソフトウェアの構成を図 5 に示す．

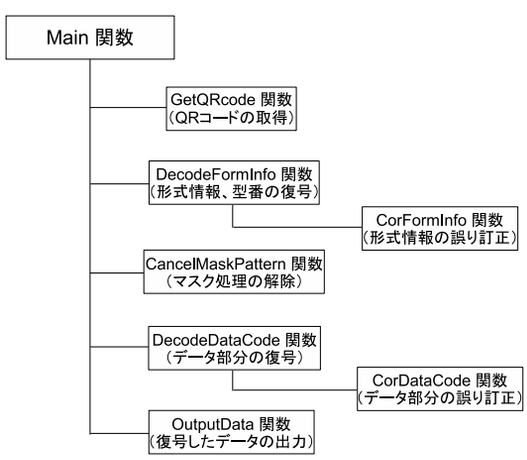

図 5 デコードソフトウェアの構成

3.2 QR コードデコードソフトウェアの各関数

以下に，図 5 に示したソフトウェアを構成する各関数の機能を簡単に説明する．

- Main 関数

  QR コードのデコード処理を行う各関数を呼び出す main 関数である．

- GetQRcode 関数

  数値化された QR コードを取得する関数である．‘0’と‘1’で数値化された QR コードをファイルから取得する．

- DecodeFormInfo 関数

  取得した QR コードから形式情報を取得し，それを復号する関数である．形式情報を取得し，マスクパターン 101010000010010 で形式情報のマスクを解除し，CorFormInfo 関数を呼び出して誤り訂正を行い，誤り訂正レベル指示子とデータ領域のマスクパターン指示子を取得する．

- CorFormInfo 関数

  形式情報の誤り訂正を行う関数である．DecodeFormInfo関数で取得した形式情報から誤りがあるか否かを検出し，誤りがあった場合は誤り訂正を行う．

- CancelMaskPattern 関数

  データ領域のマスク処理の解除を行う関数である．DecodeFormInfo 関数で取得したマスクパターン指示子に従いマスクの解除を行う．

- DecodeDataCode 関数

  データ領域の復号を行う関数である．CancelMaskPattern 関数で解除されたデータ領域を配置規則に従って，データコード語と誤り訂正符号を取得し，CorDataCode 関数を呼び出し，誤り訂正を行う．

- CorDataCode 関数

  データ領域の誤り訂正を行う関数である．DecodeFormInfo 関数で取得した誤り訂正レベル指示子に従い，DecodeDataCode 関数で取得したデータコード語と誤り訂正符号から誤りがあるか否かを検出し，誤りがあった場合は誤り訂正を行う．なお，誤り訂正手法としては，バーレカンプ・マッシィ法を用いた．

- OutputData 関数

  誤り訂正を行ったデータを復号し，復号したデータの出力を行う関数である．誤り訂正を行ったデータ及び誤り訂正コード語の先頭に配置されている，モード指示子，文字数指示子を取得し，使用モードに基づいて復号したデータを出力する．

3.3 開発結果

開発したQRコードデコードソフトウェアの動作確認として,「二次元コードシンボル－ QR コード－基本仕様（JIS X 0510）付属書G シンボルの符号化例」に掲載されているQR コードをデータブロックに誤りが1つある状態にしてデコードを行った．デコードに使用したQR コードを図6 に示す.

図6 デコードに使用したQR コード

デコードの結果, 復号されたデータは図7 となった.

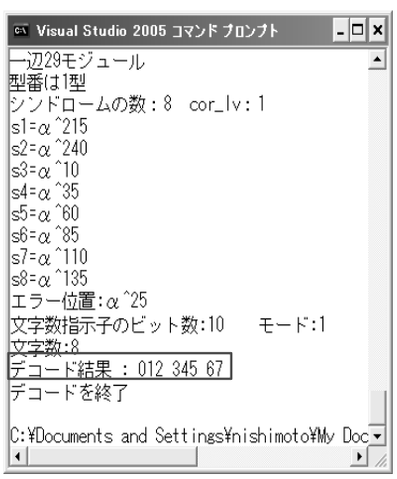

図7 デコード結果

図7より, デコード結果は“ 012 345 67 ”と出力されているので, 開発したデコードソフトウェアは正しい動作を行っているといえる.

3.4 デコードの処理時間の測定結果

QR コードデコード処理の中で, どの処理が最も時間がかかるのかを調査するため, 開発したデコードソフトウェアの処理時間を計測した.

例として, 誤り個数を1, データ数を18 文字, 誤り訂正レベルをＬにして, 型番を変化させQR コードがデコードされるまでの処理時間を計測した結果を表1 に示す.

表1 型番を変化させたときの処理時

			処理内容			
型番	形式情報, 型番の復号	マスク処理の解除	データ領域の抽出	データの誤り訂正	データの復号	全体の処理時間
1	57.2	52.9	45.1	117.9	78.9	352.0
3	59.1	83.0	56.0	176.0	79.9	454.0
5	53.9	105.1	68.9	280.9	77.0	585.8
7	54.8	140.9	96.8	39.1	78.2	764.8
9	57.0	170.9	118.0	587.0	79.0	1011.9

表1 より型番が大きくなるにつれて誤り訂正処理が大きくなっており, デコード処理の中で最も処理時間がかかっている.

また, この例以外にも各パラメータを変化させて, 処理時間を測定した結果, 最も処理時間がかかるのは誤り訂正処理で, 30 ～ 60% を占めていた.

表2 に各パラメータを変化させたときの測定結果を示す.

表2 の結果から, 多項式への代入処理や, 複数の多項式の計算をハードウェア化し, 誤り訂正処理を並列に動作させれば, 処理の高速化を行うことが可能であると考えられる.

表2 QR コードデコードの処理時間

処理内容	処理時間の割合[%]
形式情報, 型番の復号	5~15
マスク処理の解除	10~30
データ領域の抽出	5~15
データの誤り訂正	30~60
データの復号	5~20

4. デコード処理の高速化

4.1 高速化の実現

QR コードデコード処理の高速化の具体的な実現方法を図8 に示す.

QR コードデコード処理の内, 最も処理時間がかかるのは誤り訂正処理であり, 他の処理は誤り訂正に比べて処理時間が少ない．よって, 誤り訂正処理をハードウェア (FPGAで実現) で行い, その他の部分の処理はソフトウェアで行うシステムを提案する.

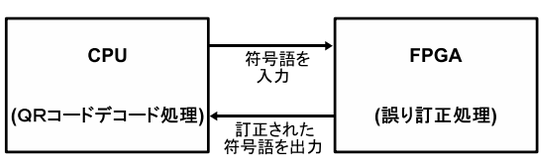

図8 高速化の実現方法

4.2 シンドローム計算

誤り訂正処理ではまず，シンドローム計算 [3-5] を行う．シンドローム計算の処理では受信語($r_{n-1}, r_{n-2}, \cdots, r_1, r_0$)から，式 (3) を用いてシンドローム多項式を生成する．ここで，受信語($r_{n-1}, r_{n-2}, \cdots, r_1, r_0$)の値はガロア体である GF ($2^8$)の元($\alpha^0$～$\alpha^{255}$)とする．

$$s(x) = \sum_{k=0}^{m-1} \sum_{i=0}^{n} r_{n-i} \cdot (\alpha^k)^{n-i} \cdot x^{k-1} \qquad (3)$$

このシンドローム計算を C 言語で行うと，ループ処理を多用するため処理に時間がかかる．しかし，シンドローム多項式の項の演算には依存関係がない．よって，シンドローム多項式の$\sum_{i=0}^{n} r_{n-i} \cdot (\alpha^k)^{n-i}$の部分はそれぞれ並列に演算することが可能である．

4.3 ユークリッド処理

4.2 のシンドローム計算の処理で求めたシンドローム多項式から多項式の乗算，除算を繰り返し，誤り位置多項式 L(x) と誤り数値多項式 E(x) を求める．この処理をユークリッド処理 [5,6] という．ユークリッド処理のフローチャートを図 9 に示す．ここで，t は訂正できる誤りの個数である．これら二つの多項式を求めることによって符号語の誤りがある部分と，その部分の正しい値を求めることができる．

ユークリッド処理を C 言語で行うと，処理全体のループ以外に多項式の乗算，除算の際，ループ処理を用いて項を一つずつ求めるため，非常に処理に時間がかかる．

これをハードウェアで行うと，多項式の乗算，除算の結果の項を並列に求めることができる．

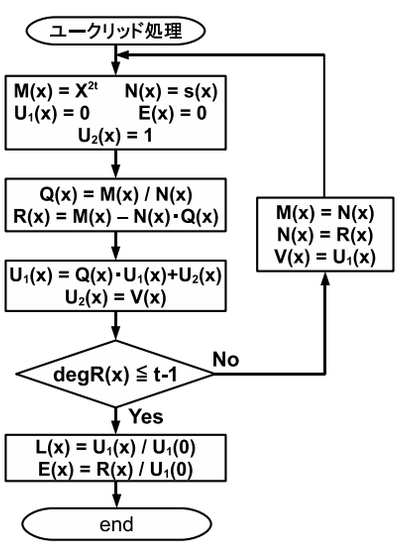

図 9 ユークリッド処理

4.4 チェン検索

ユークリッド処理から求まった誤り位置多項式 L(x) と誤り数値多項式 E(x) に GF (2^8)の元(α^0～α^{255})を一つずつ代入する．この処理をチェン検索 [5,6] という．これによって多項式の解が 0 であったときに代入した元に対応した位置に誤りがあることが分る．

本研究で設計した回路は，図 10 に示すように，多項式への代入回路を二つ作成し，GF(2^8)の元を半分ずつ代入し，2 並列で行った．

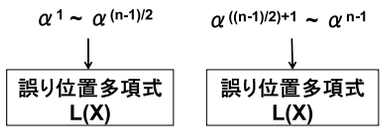

図 10 チェン検索の並列化

5. 研究結果

5.1 開発結果

ハードウェア記述言語 VHDL を使用して QR コードデコード処理を記述し，開発環境には，QuartusII Web Edition を用いた．そして，多項式の乗算，除算に必要であるガロア体 GF (2^8)の乗算回路と逆元演算回路 [7] を設計，また，第 4 章に示した処理のうちシンドローム計算回路とチェン検索回路を設計し，これらの回路について評価を行った．設計した回路の評価結果を表 3 に示す．なお，表 3 の遅延時間とは 1 クロック当たりの遅延時間を表す．

表 3 よりガロア体逆元演算回路の遅延時間はガロア体乗算回路に比べ，約 2.3 倍程度であった．また，チェン検索回路の遅延時間はシンドローム計算回路に比べ，約 2.8 倍程度であった．

表 3 実験結果 (1)

	ロジックエレメント数	遅延時間[ns]	消費電力[mW]
ガロア体乗算回路	52	17.8	31.4
ガロア体逆元演算回路	232	40.3	30.5
シンドローム計算回路	115	20.5	37.7
チェン検索回路	1911	57.7	45.5

Intel(R) Pen-tium(R) 4 上でソフトウェアで処理を行った場合との比較結果を表 4 に示す．表 4 より，Pentium 4 上のソフトウェアの処理時間と比べて Cyclone Ⅱで処理を行った場合，シンドローム計算は約 34.2%，ユークリッド処理は約 10.1%，チェン検索は約 48.9%，全

体としては約 15.0% の時間で処理できることが分り，高速化が行えることを確認した．

表 4　実験結果 (2)

誤り訂正処理	(a) Pentium 4 [μs]	(b) Cyclone II [μs]	比率 [%] (b/a)
シンドローム計算	3.8	1.3	34.2
ユークリッド処理	105.8	10.7	10.1
チェン検索	13.1	6.4	48.9
全体	122.7	18.4	15.0

5.2　考察

チェン検索回路では，多項式への代入処理を 2 並列で行った結果，遅延時間は 47.7[ns] であったが，ユークリッド処理回路の評価の結果次第ではより高速な回路を設計できると考えられる．

ユークリッド処理回路を設計する際には多項式の乗算，除算処理を多く行うため，チェン検索回路よりも多くの回路が生成されると考えられる．ユークリッド処理回路で生成される回路をチェン検索回路の代入処理に用いることで，3 並列以上の代入回路を設計することが可能になる．このようにすることでチェン検索の処理を更に高速化することができる．

6. 結論

本研究では，デコード処理のハードウェア化を行うための基礎として，初めに QR コードデコード処理のアルゴリズムを解析し，C 言語を用いて QR コードデコードソフトウェアを開発した．QR コードのデコード処理は，形式情報の復号と型番の決定，データ領域のマスク処理の解除，データ領域の誤り検出と訂正，データの復号から構成されている．そして，開発したデコードソフトウェアの中で最も時間がかかる処理を明らかにするために，各部のデータ処理時間の評価を行った．その結果，データ領域の誤り訂正に関する処理は多項式計算のためにループ処理を多用しており，全体の約 30% から 60% と，各処理の中で最も処理時間がかかっていることが分った．

次に，本研究ではデコード処理において，最も時間のかかる誤り訂正処理についてハードウェア化を行った．誤り訂正処理はシンドローム計算，ユークリッド処理，チェン検索の 3 つで構成されている．これら 3 つの処理は C 言語で行った場合，ループ処理を多用することで処理時間がかかるが，ハードウェアで並列に処理することによって，ループ回数を減少させたり，ループ 1 回の処理を高速に行うことが可能になる．これらの処理のうち，多項式の乗算，除算に必要であるガロア体 GF (2^8) の乗算回路と逆元演算回路，シンドローム計算回路とチェン検索回路をハードウェア記述言語 VHDL を用いて記述した．そして，FPGA 上で実現することを想定して論理合成を行い，ロジックエレメント数，遅延時間，消費電力の評価を行った．その結果，ハードウェア化により処理の高速化が見込まれることが分った．

今後の課題としては，誤り訂正処理以外の部分のハードウェア化が挙げられる．

参考文献

[1] 日本工業標準調査会審議，“ 2 次元コードシンボル − QR コード − 基本仕様，JIS X0510，” 日本規格協会 (2006).

[2] 標準化研究学会，“ ＱＲコードのおはなし，” 日本規格協会，2002.

[3] 西村芳一，“ ディジタルエラー訂正技術入門，” CQ 出版社，2004.

[4] 三谷政昭，“ やり直しのための工業数学，” CQ 出版社，2001.

[5] 江藤良純，金子敏信，“ 誤り訂正符号とその応用，” オーム社，1996.

[6] 小田島賢和，小西徹也，神戸尚志，“ C 言語設計によるリードソロモン符号復号化回路の最適化，” 情報処理学会研究報告，2006-SLDM-126(18), pp.99-104, 2006.

[7] 伊藤利哉，辻井重男，“ 正規基底を用いた有限体における高速逆元算出アルゴリズム，” 電子情報通信学会論文誌 (A), Vol.J70-A, pp.1637-1645, 1987.