設 計 自 動 化 81－8
（1996. 10. 18）

論理合成用 HDL 記述への定数テーブル生成機能の組み込み

東田 基樹　　小林 直弘　　石川 淳士

三菱電機 (株) 半導体基盤技術統括部 DA 開発第二グループ

兵庫県伊丹市瑞原 4-1

TEL 0727-84-7495　　Email: higasida@lsi.melco.co.jp

あらまし

パラメータ値により不規則な変化を行なう構造の回路が存在する。従来は、このような回路をパラメータ化 HDL として表現することができなかった。本論文では、不規則な変化を定数テーブルとして表現し、パラメータ値からこの定数テーブルを生成する処理を HDL 記述中に組み込むことにより、このような回路をパラメータ化 HDL として表現する手法について述べる。本手法を適用することにより、DAT や DVD 等の応用製品に適用可能なリードソロモン誤り訂正器のパラメータ化 HDL を開発することが出来た。

キーワード　パラメータ化、HDL、再利用設計、論理合成

Synthesizable HDL Model with Constant Table Generation

Motoki Higashida　　Naohiro Kobayashi　　Junji Ishikawa

Mitsubishi Electric Corp., Semiconductor Group, Manufacturing Technology Div.

4-1 Mizuhara, Itami, Hyogo Pref., JAPAN

TEL +81-727-84-7495　　Email: higasida@lsi.melco.co.jp

Abstract

There exists a circuit whose structure changes irregularly by a parameter value. We can't represent the circuit as a parameterized HDL description with a conventional technique. In this paper, we propose a novel technique to represent it as a parameterized HDL description with constant table generation. The constant table is used to represent a irregular structure. With this technique, we could create a parameterized HDL description for Reed-Solomon Error Correcting circuit applicable to many consumer products including DAT, DVD.

key words　parameter, HDL, reuse, logic synthesis

1. はじめに

既存設計を再利用できれば、設計期間を確実に短縮できる。再利用する回路が大きいほど、設計期間の短縮効果も大きい。しかし、個々の設計には独自の構成や機能が必ず存在する。再利用可能部は共通の構成や機能をもった部分に限られ、大規模な回路の再利用は困難であった。

パラメータ化 HDL 記述とは、回路の構成や機能の多様性をパラメータとして表現し、パラメータを設定することにより、パラメータに応じた回路を合成可能とした HDL 記述のことである。回路の多様性を検討し、パラメータ化 HDL 記述として設計を行なえば、設計の再利用の可能性は大幅に向上できる [1]。

構成や機能の変化をパラメータとして表現可能な回路は多い。しかし、パラメータ化 HDL として実現されている回路は限られている。現在、パラメータ化 HDL として実現されている回路の多くは、ビットスライス構造の演算器や RAM、 FIFO 等の規則性を持った回路である [2]。パラメータ値により構成や機能が不規則に変化する回路については、パラメータ化 HDL として実現されていない。

パラメータ値により構成や機能が不規則に変化する回路をパラメータ化 HDL として記述可能にするには、HDL 記述のパラメータ化手法の進歩が必要となる。本論文では、HDL 記述の新しいパラメータ化手法として、パラメータ値から定数テーブルを計算する処理を HDL 記述中に組み込む手法を提案する。本手法を用いることにより、従来は、HDL記述の生成プログラムを用いて生成していた複雑な構成の回路や、定数テーブルを含んだ回路についても、パラメータ化 HDL として表現可能になる。

なお、本パラメータ化手法は、記述言語として VerilogHDL を、論理合成ツールとして Synopsys 社の HDL Compiler / Design Compiler を用いることを想定している。

2. 従来のパラメータ化手法

Canonical Signed Digit(CSD) 定数係数乗算器の HDL 記述を例にして、従来の HDL 記述のパラメータ化手法 [1] を簡単に説明する。

2.1 CSD 定数係数乗算器

CSD 定数係数乗算器とは、加算器と減算器とシフタを用いて定数係数乗算を実現した回路である [3]。図 1 に汎用的な CSD 定数係数乗算器の HW 構成を示す。図中、CoefW は定数係数値のビット幅である。CSD の配列 (要素数は CoefW+1) には定数係数値を CSD 表現に変換した値が設定されている。CSD 配列の要素の値は 1,0,-1 のいずれかの値をとり、定数係数値から一意に決まる。配列要素が 1 ならシフタと加算器を、-1 ならシフタと減算器を、0 なら接続のみの回路を直列に接続することにより、指定した定数係数値に対応した CSD 定数乗算器を実現できる。

2.2 パラメータ化 HDL 記述

図 2 に、入力ビット幅が 7、定数係数のビット幅が 5、出力ビット幅が 9、定数係数値が 23 のパラメータ値に対応した CSD 定数係数乗算器の HDL 記述を示す。なお、定数値 23 に対応する CSD 配列の値は、下位要素から順に、-1, 0, 0, -1, 0, 1 となる。図 2 の点線内に、ビット幅と CSD 配列の値が設定されている。これらの値を変更することにより、任意の CSD 定数係数乗算器を生成できる。従って、この記述は、ビット幅と CSD の値をパラメータとした、パラメータ化 HDL 記述である。

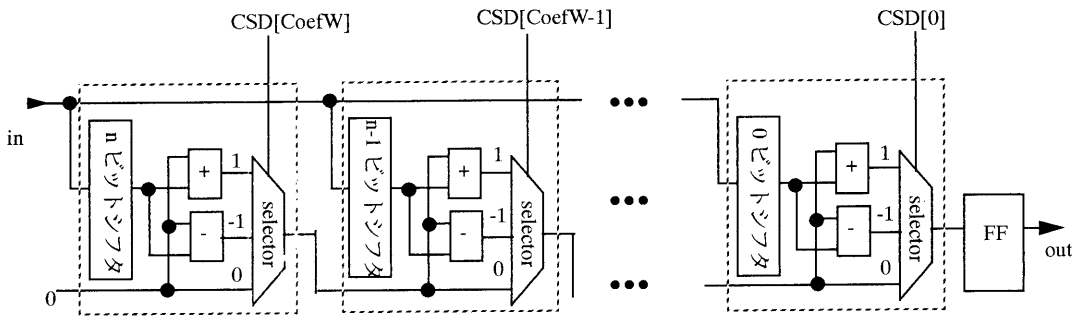


図 1 汎用的な CSD 定数係数乗算器

```
module CoefMult( in, out, clk );
  parameter InW = 7 ;      // パラメータ(ビット幅定義)
  parameter CoefW = 5 ;
  parameter OutW = 9 ;

  input [InW-1:0]  in ;       // 入力データ
  output[OutW-1:0] out ;      // 出力データ
  input            clk ;      // クロック

  reg   [OutW-1:0] out ;

  // FFを生成する記述
  always @( posedge clk )
    out <= comb_func(in) ;

  //組合せ回路部の記述
  function[OutW:0] comb_func ;
    input [InW-1:0]  in ;
    reg[InW+CoefW-1:0] outVal ;
    integer  CSD[CoefW:0] ;
    integer  i ;

    begin
    // パラメータ(CSD表記の値の定義)
      CSD[0] = -1 ;
      CSD[1] =  0 ;
      CSD[2] =  0 ;
      CSD[3] = -1 ;
      CSD[4] =  0 ;
      CSD[5] =  1 ; // CoefWに応じてエントリを増減する。

    // 汎用的なCSD定数係数乗算器の定義
      outVal = 0 ;                    // 始めは0を設定
      for( i = CoefW ; i >= 0 ; i = i-1 ) begin
          shiftVal = in << i ;        // シフタ
          // 1なら加算、-1なら減算、0なら接続のみを選択
          case (CSD[i])
            1  :     outVal = outVal + shiftVal ;
            -1 :     outVal = outVal - shiftVal ;
            default  outVal = outVal ;
          endcase
      end
      comb_func <= outVal[InW+CoefW-1:InW+CoefW-OutW] ;
                          // 不要な下位ビットを切捨て
    end
  endfunction
endmodule
```

図 2 CSD 定数係数乗算器のパラメータ化 HDL(従来例)

2.3 従来のパラメータ化手法

図 2 の HDL 記述を例にして、従来のパラメータ化手法を説明する。

(1) ビット幅のパラメータ化

VerilogHDL では、任意のビット幅の入出力ポートや reg 変数、wire を定義できる。また、VerilogHDL の演算 (論理演算、算術演算) は変数のビット幅に応じて、不要ビットの除去や不足ビットの拡張を行なってくれる。図 2 の HDL 記述では、この手法を用いて入出力ポート幅のパラメータ化、及び、加算、減算演算に対応する演算器のビット幅のパラメータ化を実現している。

(2) 規則構造の繰り返し回数のパラメータ化

規則構造を構成している基本セルの論理を動作記述し (構造記述ではない)、その動作記述を for ループを用いて必要な回数分呼び出す書式の HDL 記述を合成すれば、規則構造を持った回路を実現できる。この書式の for ループのループ回数をパラメータとすれば、規則構造の繰り返し回数をパラメータ化できる。CSD 定数係数乗算器 (図 1) は、繰り返し回数が CoefW+1 である規則構造を持つ。図 2 では、パラメータ (CoefW) をループ回数とした for 文により、これを表現している。

(3) 機能選択のパラメータ化

論理合成ツールには、合成回路中の常に定数となる信号線を探索し、冗長な回路を除去する機能 (定数伝搬機能) がある。多くの機能をもつ汎用的な回路を用意し、パラメータにより必要な機能を選択可能にした HDL 記述を作成する。この HDL 記述を論理合成すれば、定数伝搬機能により、パラメータで選択されなかった機能のみに必要な論理回路は除去される。この機能を用いることにより、回路の機能選択のパラメータ化が実現できる。

CSD 定数係数乗算器の HW 構成 (図 1) の点線枠の回路は、加算器と減算器と接続のみの回路をセレクタで選ぶ構成をしている。対応する HDL 記述 (図 2) でも、case 文によりこのセレクタを実現している。本 HDL を合成すると、セレクタの入力 (CSD 配列の値) は定数であるから、定数伝搬機能により、CSD 配列の値に応じて、加算器か減算器か接続のうちの 1 つの回路のみが実現される。このようにして、図 2 の HDL 記述では機能選択のパラメータ化を実現している。

以上の 3 種類が、従来使用されている HDL 記述の主なパラメータ化手法である。

3. 定数テーブル生成を組み込んだパラメータ化 HDL

図 2 の CSD 定数係数乗算器の HDL 記述では、定数係数を CSD 表現の値に変換し、CSD の値としてパラメータを設定する必要がある。この変換作業の必要性は、HDL 記述の利用を困難にしていた。もし、定数係数値から CSD 表現の値に変換する処理を HDL 記述中に組み込むことができれば、HDL 記述の利用が容易になる。本章では、このような HDL 記述の作成を可能にする、定数テーブルの生成処理を組み込んだパラメータ化 HDL の作成手法について述べる。

3.1 定数テーブル生成のパラメータ化

図 3 に、定数係数値 (CoefVal) から CSD 配列の値を計算する処理を組み込んだ CSD 定数係数乗算器の

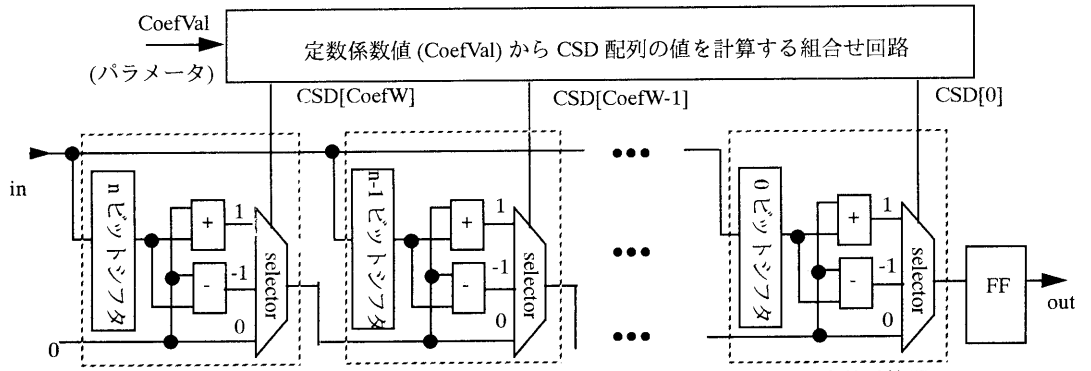


図 3 CSD 配列の計算処理を組み込んでパラメータ化した CSD 片側定数乗算器

```
module CoefMult( in, out, clk );

  parameter InW = 7 ;      // パラメータ (ビット幅定義)
  parameter CoefW = 5 ;
  parameter OutW = 9 ;
  parameter CoefVal = 23 ;

  input [InW-1:0]  in ;       // 入力データ
  output[OutW-1:0] out ;      // 出力データ
  input            clk ;      // クロック

  reg   [OutW-1:0] out ;

  // synopsys translate_off
  reg sim_first ;
  initial begin
     sim_first = 1 ;
  end
  // synopsys translate_on

  // FF の生成記述
  always @( posedge clk )
    out <= comb_func(in) ;

  // 組合せ回路の生成記述
  function[OutW:0] comb_func ;
    input [InW-1:0]  in ;
    reg[InW+CoefW-1:0] outVal ;
    integer  CSD[CoefW:0] ;
    integer  i ;

    begin
      // 定数係数値から定数テーブルの計算処理
        // synopsys translate_off
        if ( sim_first ) begin
        // synopsys translate_on

           <CoefVal から、CSD の値の計算処理の記述 >

        // synopsys translate_off
           sim_first = 0 ;
        end
        // synopsys translate_on

      // 汎用的な CSD 定数係数乗算器の定義
        outVal = 0 ;                          // 始めは 0 を設定
        for( i = CoefW ; i >= 0 ; i = i-1 ) begin
            < 省略 >
        end
        comb_func <=
              outVal[InW+CoefW-1:InW+CoefW-OutW] ;
                              // 不要な下位ビットを切捨て
    end
  endfunction
endmodule
```

図 4 CSD 定数係数乗算器のパラメータ化 HDL(新手法)

HW 構成を示す。図 1 の回路に、定数係数から CSD 配列の値を計算するための組合せ回路が付加されている。

図 3 の HW 構成に対応するパラメータ化 HDL 記述は図 4 のようになる。図 2 の HDL 記述と比べ、図 4 の HDL 記述は、定数係数値 (CoefVal) を直接設定できる parameter 文が追加され、また、CSD 配列の値を設定する記述が、定数係数値 (CoefVal) から CSD 配列を計算するための記述に置き換えられている。また、HDL 記述の検証時のシミュレーション時間の短縮のために、initial 文等が付加されている。これについては、次節で述べる。

定数係数値から CSD 配列の値を計算するための組合せ回路の入力は定数値 (パラメータ値:CoefVal) のみである。従って、論理合成ツールの定数伝搬機能により、この組合せ回路は除去され、CSD 配列に相当する接続線に CSD の値が残る。この回路は、図 1 の回路に相当する。

このようにして、パラメータ値 (定数値) から定数テーブルを生成する処理を組み込んだ HDL 記述を作成することができる。

VerilogHDL は C 言語並の文法と演算子を持っているので、C 言語で作成されたプログラムを VerilogHDL 記述へ変換することは容易である。フィルタの係数テーブルのような定数テーブルは、従来は生成プログラムによって作成していたが、この計算処理を VerilogHDL にて記述し HDL 記述に組み込むことにより、利便性の高いパラメータ化 HDL 記述とすることができる。

3.2 問題点と解決策

定数テーブルの生成処理を HDL 記述中に含める

と、(1) 論理合成時の処理時間の増加、及び、(2)HDL 記述の検証時のシミュレーション時間の増加という問題が発生する。

これらの問題は、パラメータ化を行なっていない通常の HDL 記述においても、回路が大規模になると必ず発生する問題である。パラメータ化 HDL の大きな目的は、大規模な回路の再利用性の向上であるため、これらの問題を無視することはできない。

本節では、これらの問題を軽減するための工夫について述べる。

(1) シミュレーション時間

シミュレーション時間の増加に関する問題は、論理合成用 HDL 記述では always 文あるいは関数文中に定数テーブル生成部を記述する必要があるので、シミュレーション時に何度も定数テーブルの計算処理を実行してしまうことに起因する。この問題を回避するために、図 4 の HDL 記述では、 sim_first の変数を使用して、定数テーブル計算処理を 1 回だけの実行に押えている。

図 4 の HDL 記述のシミュレーションは次のように実行される。まず最初に initial 文にて sim_first 変数が 1 にセットされる。続いて、クロックの立ち上がりイベント毎に always 文、及び、comb_func 関数が実行される。一回目の実行では、sim_first 変数は 1 なので、if 文の内部が実行され、パラメータ値(CoefVal)から CSD 配列の値の計算処理が実行される。if 文の最後では、sim_first 変数が 0 にリセットされる。2 回目以降の com_func 関数の実行では、sim_first 変数は 0 であり、if 文の内部動作である CSD 配列の計算処理は実行されない。このようにして、定数テーブル計算部の複数回の実行を回避している。

なお、変数 sim_first と if 文、及び initial 文は、シミュレーションの高速化が目的であり、論理合成の対象ではない。従って、translate_off/on の合成指示子を用いて合成の対象外としてある。

(2) 論理合成時間

定数テーブル生成部を組み込んだ HDL 記述からの論理合成では、図 3 に示すような定数テーブル生成の処理に対応した組合せ回路が論理合成ツール内で生成され、その後、定数伝搬機能により除去される。これらの処理に必要な時間が、合成時間の増加の原因である。ここでは、合成時間の増加を少なくする工夫について述べる。

論理合成ツールは、HDL 記述が入力されると、まず HDL 記述中から FF やラッチのような順序素子になるべき reg 変数を特定し、続いて HDL 記述に対応する組合せ論理を抽出し、その後、定数伝搬処理が行なわれる。これらの処理の中で、順序素子になるべき reg 変数の特定に最も多くの時間を要している。

図 3 からも分かるように、定数テーブルの生成部は、全て組合せ回路になる必要がある。従って、定数テーブルの生成記述中の reg 変数が順序素子になることはない。論理合成ツールは、function 文による記述部については全て組合せ回路になるものと解釈し、順序素子になるべき reg 変数の特定を行なわない。図 4 では、定数生成部を function 文中に記述することにより、順序素子の特定処理のための時間をなくし、合成時間の短縮を図っている。

定数テーブルの生成記述では、定数テーブル自身を筆頭に、一時的な変数として多くの reg 変数を使用する。このため、順序素子の特定処理を回避することによる合成時間の短縮効果は大きい。

4. リードソロモン誤り訂正器への応用

本パラメータ化手法を用いて、大規模かつ複雑な回路のパラメータ化を実現した例として、リードソロモン誤り訂正器 (RS 訂正器) のパラメータ化 HDL について述べる。

4.1 リードソロモン誤り訂正符号

リードソロモン誤り訂正符号 (RS 符号) は、図 5 のような構成となる。原データを、(n-2m) 個のシンボル単位に分割し、これを情報シンボルとする。情報シンボルに、2m 個のパリティシンボルを付加して、1 ブロックの RS 符号が構成される。但し、m は誤り訂正可能なシンボル数であり、1 シンボルは b ビット、1 ブロックの RS 符号は n シンボルからなる。パリティシンボルは情報シンボルと生成多項式から計算できる。また、生成多項式は原始多項式から決定できる。従って、任意の RS 符号は、(1) ブロックのシンボル長 (n)、(2) シンボルのビット長 (b)、(3) 誤り訂正可能シンボル数 (m)、(4) 原始多項式 (p) の 4 種類のパラメータにより決定できる [1]。

RS 符号の生成に用いる演算は通常の四則演算では

1. 1 つの原始多項式には複数の生成多項式が存在するので、厳密に任意の RS 符号を構成できる訳ではない。しかし、通常の応用分野においては、原始多項式が決定すれば生成多項式も決まる。

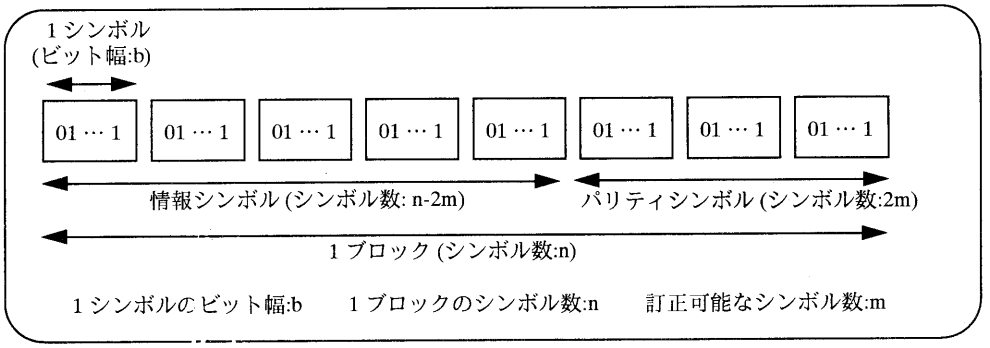


図 5 リードソロモン誤り訂正符号の符号構成

なく、ガロア体演算と呼ばれる演算規則を用いる[4]。ガロア体演算では、演算の入出力は全て同じビット幅となる。ガロア体加算はビット毎の XOR として、ガロア体乗算は原始多項式 p をパラメータとする多段の XOR として、ガロア体除算は原始多項式 p により変化する逆数テーブルとガロア体乗算を組み合わせた回路として実現できる。

4.2 パラメータ化 HDL

RS 符号を決定する 4 種類のパラメータ (n,b,m,p) の全てをパラメータにすれば、任意の RS 符号に対応可能な RS 訂正器を実現できる。図 6 に、パラメータによる構成の変化を考慮した RS 訂正器の HW 構成の一部 (全体構成とチェンサーチモジュール) を示す。

図 6 の回路は、パラメータ値により、次に示すような構成の変化がある。

(1) モジュール/ ブロック間のバス信号のビット幅、及び Reg, Sel, XOR のビット幅
(2) RAM のワード数
(3) チェンサーチモジュールにある定数乗算器の個数
(4) チェンサーチモジュールにある奇数部/ 偶数部の XOR の入力線
(5) (ガロア体) 乗算器 (ビット幅 b 及び原始多項式 p により変化する)
(6) (ガロア体) 除算器 (除算器は逆数変換のための定数テーブルと乗算器により構成される。)
(7) 定数乗算器の定数値

(1),(2) はビット幅やワード数の変化であり、従来のパラメータ化手法を使用して対応できる。(3),(4) も、規則構造の繰り返し回数の変化であり、従来のパラメータ化手法により対応できる。

(5) ～ (7) については、今回提案した定数テーブル生成処理を組み込む手法によりパラメータ化できる。

(5) は、XOR の接続構成がパラメータ値 p により複雑に変化する回路である。この回路は、CSD 定数係数乗算器と同様の形態のパラメータ化 HDL 記述として作成できる。

(6) はビット幅と原始多項式によって変化する定数テーブルを使用している。定数テーブルの計算プログラムは C 言語記述で 20 行程度のプログラムになる。このテーブル生成プログラムを VerilogHDL 記述に変換し、除算器本体の HDL 記述に組み込んで、パラメータ化 HDL 記述を作成した。

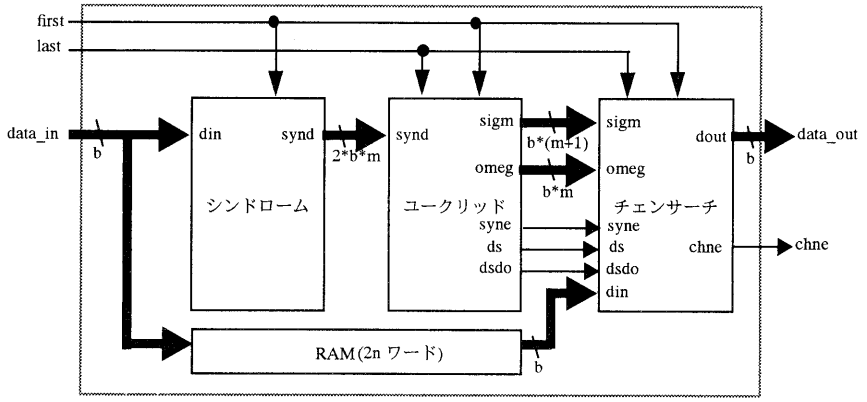
(7) は、(3) のパラメータ化で生成される複数の定数乗算器に与える定数値に関するものである。それぞれの定数乗算器に与えられる定数値は、単純には計算できない。このため、定数乗算器の HDL 記述内にループ変数の値から定数値を検索できるような定数テーブルを組み込んだ。この定数テーブル生成にも、定数テーブル作成のパラメータ化手法を用いた。

以上のようなパラメータ化手法を適用することにより、RS 符号の 4 種類のパラメータ全てをパラメータ化した HDL 記述を作成できた。次節で述べるように、HW アーキテクチャ上の限界により任意の RS 符号を実現可能な HDL 記述ではないが、広い適用範囲をもつパラメータ化 HDL 記述とすることができた。

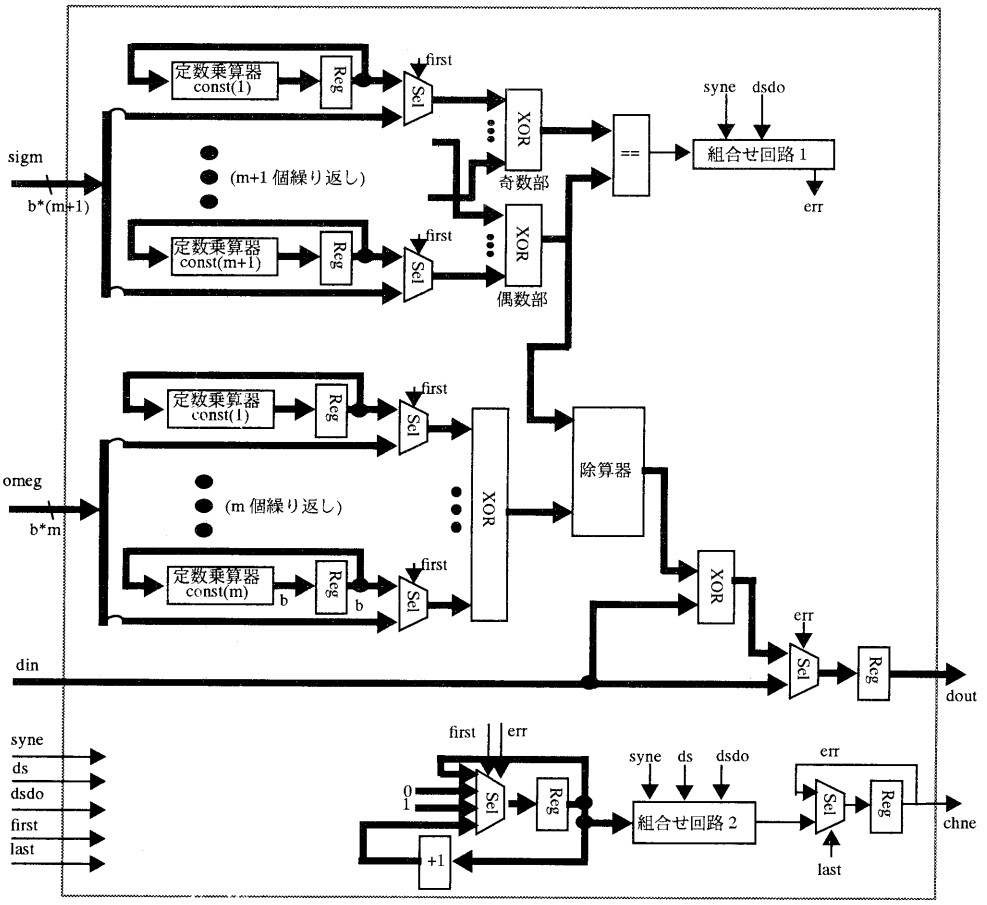
4.3 適用範囲

本 RS 訂正器ではパイプラインアーキテクチャを採用している。具体的には、シンドローム、ユークリッド、チェンサーチの 1 ブロック分の処理を、それぞれ n クロック (n は 1 ブロックのシンボル数) で実行させ、この 3 つの処理に対応したモジュールをパイプライン実行させている。

シンドローム、チェンサーチの処理は入力データに無関係に n クロックで処理が完了する。しかし、ユークリッドの処理に必要なクロック数は入力データ



(a) RS 訂正器全体の HW 構成



(b) チェンサーチモジュールの HW 構成

図 6 パラメータを考慮した RS 訂正器の HW 構成

表 1 RS 訂正器の実現可能範囲と RS 訂正器の応用製品

誤り訂正可能数 (m)	ブロックシンボル数 (n)	RS 訂正器の応用製品
1	>= 6	CD-ROM(26, 45)
2	>= 16	CD (28, 32), DAT(32), HDTV-VTR(64)
3	>= 32	DAT (32), HDTV-VTR(110)
4	>= 54	D-3VTR(95, 136)
5	>= 82	DVD(182)
6	>= 116	
7	>= 156	
8	>= 202	CATV(204), DVD(208)
9	>= 254	

に依存し、最悪のクロック数は訂正可能なシンボル数 m によって決まる。このため、ユークリッドの処理に必要なクロック数が n を越えるとパイプラインが異常動作する。ユークリッドの処理に必要な最悪のクロック数が n を越えない場合が、本 RS 訂正器が正常に動作する範囲である。表 1 に、本 RS 訂正器が正常に動作する範囲 (誤り訂正可能数に対するブロックシンボル数の下限 [1]) を示す。表には、RS 訂正器を使用している応用製品を誤り訂正可能数で分類して載せた。製品名の右の数字は、ブロックシンボル数である。RS 訂正器を複数使用している製品については、複数のエントリや複数のブロックシンボル数の記載がある。これらの RS 訂正器の全ては、本 HDL 記述の適用範囲内である。ここに載せた応用製品例は文献 [5] に紹介されている RS 訂正器の使用例の全てを含んでいる。本 HDL が、代表的な RS 訂正器をカバーする高い汎用性を持つ記述になっていることがわかる。

5. おわりに

本論文では、HDL 記述の新しいパラメータ化手法として、HDL 記述に定数テーブル生成処理を組み込む手法を提案した。

本手法を利用することにより、パラメータ値により不規則に変化する構成の回路についても、パラメータ化 HDL として記述できる。これにより、RS 訂正器のような、従来は HDL 記述の生成プログラムにより生成していた複雑かつ大規模な HDL 記述についても、パラメータ化 HDL として表現できるようになった。

本手法の最大の問題点は、定数テーブルの計算処理が複雑になると、論理合成時間が非常に長くなることである。定数テーブルの計算は、定数伝搬機能を用いなくとも、論理シミュレーション技術を用いれば、簡単かつ高速に計算できる。今後は、論理合成ツールと論理シミュレータを組み合わせ、高速に定数テーブルを計算できるようにしたいと考えている。

1. 上限は RS 符号の理論上の上限と等しく、2^b-1(b はビット幅) である。

参考文献

[1] J.Scott Runner, "Considerations when implementing a design reuse methodology", Synopsys Methodology Note, Synopsys (1996).

[2] Synopsys, Design Ware Componetnts Databook, version 3.2b, Synopsys, (1995).

[3] R.Hartley, "Optimization of canonic signed digit multipilers for filter design," Proc. ISCS., pp.1992-1995, june (1991).

[4] 今井監、 誤り訂正符号化技術の要点、 日本工業技術、(1986).

[5] NHK 放送技術研究所編、マルチメディア時代のディジタル放送技術事典、丸善、(1994).