(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4562136号
(P4562136)
(45)発行日　平成22年10月13日(2010.10.13)　(24)登録日　平成22年8月6日(2010.8.6)

(51)Int.Cl.　F I
G06F 17/50 (2006.01)　G06F 17/50 656A
G06F 17/50 654M

請求項の数 16　(全 53 頁)

(21)出願番号　特願2005-515662(P2005-515662)
(86)(22)出願日　平成16年11月19日(2004.11.19)
(86)国際出願番号　PCT/JP2004/017263
(87)国際公開番号　WO2005/050494
(87)国際公開日　平成17年6月2日(2005.6.2)
審査請求日　平成19年10月19日(2007.10.19)
(31)優先権主張番号　特願2003-389264(P2003-389264)
(32)優先日　平成15年11月19日(2003.11.19)
(33)優先権主張国　日本国(JP)

(73)特許権者　802000031
財団法人北九州産業学術推進機構
福岡県北九州市若松区ひびきの２番１号
(74)代理人　100121371
弁理士　石田　和人
(72)発明者　笹尾　勤
福岡県福岡市中央区鳥飼３丁目１５－１７
(72)発明者　井口　幸洋
神奈川県足柄下郡箱根町強羅１３２０－１２３１

審査官　田中　幸雄

最終頁に続く

(54)【発明の名称】グラフ幅削減装置及びグラフ幅削減方法、並びに論理回路合成装置及び論理回路合成方法

(57)【特許請求の範囲】
【請求項１】
入力変数を$X=(x_1,\cdots,x_n)$（n∈自然数）とし出力にドント・ケアを含む不完全定義関数である多出力論理関数$F(X)=(f_0(X),\cdots,f_{m-1}(X))$に対して式（１）により定義される特性関数χ(X,Y)（但し、$Y=(y_0,\cdots,y_{m-1})$（m≧2，m∈自然数）はF(X)の出力を表す変数。）を表現する特性関数二分決定グラフ（Binary Decision Diagram for Characteristic Function : BDD for CF）の幅を削減するグラフ幅削減装置であって、以下の構成を備えているグラフ幅削減装置：
多出力論理関数F(X)の特性関数二分決定グラフのそれぞれの非終端節点v_iについて、当該非終端節点に関わる変数z_i（$z_i\in(X\cup Y)$）に付与される変数ラベル並びに当該変数z_i（$z_i\in(X\cup Y)$）の値が０のとき及び１のときに遷移する先の子節点を特定する一対の枝$e_0(v_i)$，$e_1(v_i)$からなる節点データのテーブルである節点テーブルを記憶する節点テーブル記憶手段；
前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ（以下「分割の高さ」という。）levの設定を行う分割線設定手段；
前記節点テーブル記憶手段に記憶された前記節点テーブルから、前記特性関数二分決定グラフを前記分割線設定手段により設定された前記分割の高さlevで分割し関数分解することによって得られる分解表の列を表す関数である列関数を生成する列関数生成手段；
及び、前記列関数生成手段が生成する各列関数のうち、両立する列関数に対してドント

・ケアに定数を割り当てることによって同一の列関数（以下「割当列関数」という。）とするとともに、これらの新たな割当列関数を用いて前記特性関数二分決定グラフ再構成し前記節点テーブル記憶手段の節点テーブルを更新する割当ＢＤＤ再構成手段。

【数１】

$$\chi(X,Y)=\bigwedge_{i\ 0}^{m-1}\{\bar{y}_i f_{i\_0}\vee y_i f_{i\_1}\vee f_{i\_d}\} \qquad (1)$$

但し、f_{i_0}，f_{i_1}，f_{i_d}は、それぞれ、式（２）で定義されるOFF関数、ON関数、及びDC関数を表す。

【数２】

$$f_{i\_0}(X)\quad\begin{cases}1 & (X\in f_i^{-1}(0))\\ 0 & (\text{otherwise})\end{cases},\ f_{i\_1}(X)\quad\begin{cases}1 & (X\in f_i^{-1}(1))\\ 0 & (\text{otherwise})\end{cases},\ f_{i\_d}(X)\quad\begin{cases}1 & (X\in f_i^{-1}(d))\\ 0 & (\text{otherwise})\end{cases} \qquad (2)$$

【請求項２】

前記列関数を節点（以下「関数節点」という。）とするグラフであって、互いに両立する複数の列関数に対応する関数節点同士が枝（以下「両立枝」という。）により連結されたグラフである両立グラフを、前記各関数節点の列関数ラベル及び当該関数節点に連結する両立枝のデータ（以下「関数節点データ」という。）のテーブルとして記憶するための両立グラフ記憶手段；

前記両立グラフ記憶手段に記憶された前記各関数節点データに対する列関数から、両立する列関数の組を選出し、それら両立する列関数に対する関数節点データにそれらの関数節点同士を連結する両立枝を追加して前記両立グラフ記憶手段に記憶された関数節点データの更新を行う両立枝生成手段；

及び、前記両立グラフのすべての節点について、完全部分グラフ（以下「クリーク」という。）による節点被覆を行うことにより、クリークに含まれる関数節点集合であるクリーク・データを生成するクリーク生成手段；

を備え、

前記列関数生成手段は、前記節点テーブル記憶手段に記憶された前記節点テーブルから、前記分割線設定手段により設定された前記分割の高さlevの節点の各枝に対応する列関数を生成し、それら各列関数に対応した列関数ラベルを有する前記関数節点データを生成して前記両立グラフ記憶手段に保存するものであり、

前記割当ＢＤＤ再構成手段は、前記クリーク生成手段が生成する前記各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数とすることにより、前記特性関数二分決定グラフを再構成し前記節点テーブル記憶手段の前記節点テーブルを更新するものであること

を特徴とする請求項１記載のグラフ幅削減装置。

【請求項３】

前記分割線設定手段は、前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフの根節点の子の節点の高さから下位に向かって順次分割の高さlevの設定を行うものであり、

前記割当ＢＤＤ再構成手段は、前記分割線設定手段が設定する前記各分割の高さlevにおいて逐次前記特性関数二分決定グラフ再構成を行うこと

を特徴とする請求項１又は２記載のグラフ幅削減装置。

【請求項４】

入力変数を$X=(x_1,\cdots,x_n)$（n∈自然数）とする多出力論理関数$F(X)=(f_0(X),\cdots,f_{m-1}(X))$の特性関数二分決定グラフから、前記多出力論理関数F(X)に対応する論理回路を構成するためのデータであるルックアップ・テーブル（Look-Up Table : LUT）を生成する論理回路合成装置であって、以下の構成を備えている論理回路合成装置：

完全定義関数である前記多出力論理関数$F(X)=(f_0(X),\cdots,f_{m-1}(X))$に対して式（３）により定義される特性関数$\chi(X,Y)$（但し、$Y=(y_0,\cdots,y_{m-1})$（$m\geqq 2$，$m\in$自然数）は$F(X)$の出力を表す変数。）を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点v_iについて、当該非終端節点に関わる変数z_i（$z_i\in(X\cup Y)$）に付与される変数ラベル並びに当該変数z_i（$z_i\in(X\cup Y)$）の値が０のとき及び１のときに遷移する先の子節点を特定する一対の枝$e_0(v_i)$，$e_1(v_i)$からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段；

前記ルックアップ・テーブルを記憶するＬＵＴ記憶手段；

前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ（以下「分割の高さ」という。）levの設定を行う分割線設定手段；

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において２つの部分グラフB_0,B_1に分割したときに根節点を含む側の部分グラフB_0に属するものであって、出力を表す変数y_r（$\in Y$）に関わる節点v_j及びその親節点v_kの節点データについて、当該節点v_jの枝$e_0(v_j)$，$e_1(v_j)$の一方$e_a(v_j)$が$\chi(X,Y)=0$に関わる終端節点を特定する場合、当該節点v_jの親節点v_kの枝$e_0(v_k)$，$e_1(v_k)$のうち当該節点v_jを特定する枝$e_c(v_k)$を、当該節点v_jの前記枝$e_a(v_j)$以外の枝$e_b(v_j)$に置き換える短絡除去処理を行う短絡除去手段；

前記短絡除去手段により短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって前記分割の高さlevより高い高さにある非終端節点に属する枝のうち、前記分割の高さlevより低い高さの非終端節点の子節点を特定するものの個数を計数し（但し、同じ節点を特定するものは１つと数え、定数０に向かう枝は無視する。）、その個数を前記分割の高さlevの分割線における幅Ｗとして出力するＢＤＤ幅計測手段；

前記ＢＤＤ幅計測手段が出力する幅Ｗに基づき、式（４）の演算により中間変数の個数uを算出する中間変数算出手段；

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において２つの部分グラフB_0,B_1に分割したときに根節点を含む側の部分グラフB_0に属するものについて、ルックアップ・テーブルを生成しそれをＬＵＴ記憶手段に格納するＬＵＴ生成手段；

及び、前記中間変数算出手段が算出する前記中間変数の個数uと等しい制御入力数を有する二分木（binary tree）を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフB_0に属する非終端節点の節点データを、前記二分木を表す節点データで置き換えることにより、特性関数二分決定グラフを再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新するＢＤＤ再構成手段。

【数３】

$$\chi(X,Y)\quad\bigwedge_{i=0}^{m-1}(y_i\equiv f_i(X)) \tag{3}$$

【数４】

$$u=\lceil\log_2 W\rceil \tag{4}$$

【請求項５】

前記節点テーブル記憶手段には、出力にドント・ケアを含む不完全定義関数である前記多出力論理関数$F(X)=(f_0(X),\cdots,f_{m-1}(X))$に対して式（５）により定義される特性関数$\chi(X,Y)$（但し、$Y=(y_0,\cdots,y_{m-1})$（$m\geqq 2$，$m\in$自然数）は$F(X)$の出力を表す変数。）を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点v_iについて、当該非終端節点に関わる変数z_i（$z_i\in(X\cup Y)$）に付与される変数ラベル並びに当該

変数z_i（$z_i \in (X \cup Y)$）の値が０のとき及び１のときに遷移する先の子節点を特定する一対の枝$e_0(v_i)$，$e_1(v_i)$からなる節点データのテーブルである節点テーブルとして記憶されていること
を特徴とする請求項４記載の論理回路合成装置。
【数５】

$$\chi(X,Y) - \bigwedge_{i\_0}^{m-1} \{\bar{y}_i f_{i\_0} \vee y_i f_{i\_1} \vee f_{i\_d}\} \tag{5}$$

但し、f_{i_0}，f_{i_1}，f_{i_d}は、それぞれ、式（６）で定義されるOFF関数、ON関数、及びDC関数を表す。
【数６】

$$f_{i\_0}(X) \quad \begin{cases} 1 & (X \in f_i^{-1}(0)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i\_1}(X) \quad \begin{cases} 1 & (X \in f_i^{-1}(1)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i\_d}(X) \quad \begin{cases} 1 & (X \in f_i^{-1}(d)) \\ 0 & (\text{otherwise}) \end{cases} \tag{6}$$

【請求項６】
請求項１乃至３の何れか一記載のグラフ幅削減装置を備え、
前記短絡除去手段は、前記グラフ幅削減装置によって前記節点テーブル記憶手段に記憶された前記節点テーブルにより表される特性関数二分決定グラフの幅を削減し更新された前記節点テーブルについて短絡除去処理を行うものであることを特徴とする請求項５記載の論理関数合成装置。
【請求項７】
前記多出力論理関数F(X)の要素をなす論理関数$f_0(X),\cdots,f_{m-1}(X)$の順序をπ＝（π[0]，…，π[m-1]）（但し、π[i]=jは、f_jがi番目であることを表す。）とし、論理関数f_j（∈F(X)）が依存する入力変数の集合を$\mathrm{supp}(f_j)$としたとき、式（７）で表されるTの値が最小となるように前記多出力論理関数F(X)の要素の順序πを決定する出力変数順序決定手段；
前記各入力変数x_i（∈X）及び出力を表す変数y_j（∈Y）の順序を、式（８）を満たす順序Pに決定する全変数順序決定手段；
及び、前記全変数順序決定手段で決定された順序Pに従って、特性関数二分決定グラフの節点データを生成し前記節点テーブル記憶手段に格納するＢＤＤ生成手段；
を備えていることを特徴とする請求項４乃至６の何れか一記載の論理回路合成装置。
【数７】

$$T - \sum_{k\_0}^{m-1} \left| \bigcup_{l\_0}^{k} \mathrm{supp}(f_{\pi[l]}) \right| \tag{7}$$

【数８】

$$P \quad \cdots \quad \left(\mathrm{supp}(f_{\pi[0]}), y_{\pi[0]}, \mathrm{supp}(f_{\pi[1]}) - \mathrm{supp}(f_{\pi[0]}), y_{\pi[1]}, \mathrm{supp}(f_{\pi[2]}) - \left(\sum_{k-0}^{1} \mathrm{supp}(f_{\pi[k]}) \right), y_{\pi[2]}, \right.$$
$$\left. \cdots, \mathrm{supp}(f_{\pi[m-1]}) - \left(\sum_{k\_0}^{m-2} \mathrm{supp}(f_{\pi[k]}) \right), y_{\pi[m-1]} \right) \tag{8}$$

【請求項８】
入力変数をX=($x_1,\cdots,x_n$)（n∈自然数）とし出力にドント・ケアを含む不完全定義関数である多出力論理関数F(X)=($f_0(X),\cdots,f_{m-1}(X)$)に対して式（９）により定義される特性

関数χ(X,Y)（但し、Y=(y_0,…,y_{m-1})（m≧2，m∈自然数）はF(X)の出力を表す変数。）を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点v_iについて、当該非終端節点に関わる変数z_i（z_i∈(X∪Y)）に付与される変数ラベル並びに当該変数z_i（z_i∈(X∪Y)）の値が０のとき及び１のときに遷移する先の子節点を特定する一対の枝$e_0(v_i)$，$e_1(v_i)$からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段を備えたシステムにおいて、当該特性関数二分決定グラフの幅を削減するグラフ幅削減方法であって、以下の各ステップを有するグラフ幅削減方法：

前記節点テーブルによって表現される特性関数二分決定グラフを分割する高さ（以下「分割の高さ」という。）levの設定を行う分割線設定ステップ；

前記節点テーブル記憶手段に記憶された前記節点テーブルから、前記特性関数二分決定グラフを前記分割線設定ステップで設定された前記分割の高さlevで分割し関数分解することによって得られる分解表の列を表す関数である列関数を生成する列関数生成ステップ；

及び、前記列関数生成ステップにおいて生成される各列関数のうち、両立する列関数に対してドント・ケアに定数を割り当てることによって同一の列関数（以下「割当列関数」という。）にするとともに、これらの新たな割当列関数を用いて前記特性関数二分決定グラフ再構成し前記節点テーブル記憶手段の節点テーブルを更新する割当ＢＤＤ再構成ステップ。

【数９】

$$\chi(X,Y) \quad \bigwedge_{i=0}^{m-1} \{\bar{y}_i f_{i\_0} \vee y_i f_{i\_1} \vee f_{i\_d}\} \qquad (9)$$

但し、f_{i_0}，f_{i_1}，f_{i_d}は、それぞれ、式（１０）で定義されるOFF関数、ON関数、及びDC関数を表す。

【数１０】

$$f_{i\_0}(X) \quad \begin{cases} 1 & (X \in f_i^{-1}(0)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i\_1}(X) \quad \begin{cases} 1 & (X \in f_i^{-1}(1)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i\_d}(X) \quad \begin{cases} 1 & (X \in f_i^{-1}(d)) \\ 0 & (\text{otherwise}) \end{cases} \qquad (10)$$

【請求項９】

前記システムは、前記列関数を節点（以下「関数節点」という。）とするグラフであって、互いに両立する複数の列関数に対応する関数節点同士が枝（以下「両立枝」という。）により連結されたグラフである両立グラフを、前記各関数節点の列関数ラベル及び当該関数節点に連結する両立枝のデータ（以下「関数節点データ」という。）のテーブルとして記憶するための両立グラフ記憶手段を備えており、

前記列関数生成ステップにおいては、前記節点テーブル記憶手段に記憶された前記節点テーブルから、前記分割線設定ステップで設定された前記分割の高さlevの節点の各枝に対応する列関数を生成し、それら各列関数に対応した列関数ラベルを有する前記関数節点データを生成して前記両立グラフ記憶手段に保存するとともに、

前記両立グラフ記憶手段に記憶された前記各関数節点データに対する列関数から、両立する列関数の組を選出し、それら両立する列関数に対する関数節点データにそれらの関数節点同士を連結する両立枝を追加して前記両立グラフ記憶手段に記憶された関数節点データの更新を行う両立枝生成ステップ；

及び、前記両立グラフのすべての節点について、完全部分グラフ（以下「クリーク」という。）による節点被覆を行うことにより、クリークに含まれる関数節点集合であるクリーク・データを生成するクリーク生成ステップ；

を有し、

前記割当ＢＤＤ再構成ステップにおいては、前記クリーク生成ステップにおいて生成さ

れる前記各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数とすることにより、前記特性関数二分決定グラフを再構成し前記節点テーブル記憶手段の前記節点テーブルを更新すること
を特徴とする請求項８記載のグラフ幅削減方法。
【請求項１０】
前記分割線設定ステップにおける前記分割の高さlevを、前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフの根節点の子の節点の高さから最下位の高さまで順次変更しながら、前記分割線設定ステップ乃至前記割当ＢＤＤ再構成ステップを実行すること
を特徴とする請求項８又は９記載のグラフ幅削減方法。
【請求項１１】
入力変数を$X=(x_1,\cdots,x_n)$（n∈自然数）とする完全定義関数である多出力論理関数$F(X)=(f_0(X),\cdots,f_{m-1}(X))$に対して式（１１）により定義される特性関数$\chi(X,Y)$（但し、$Y=(y_0,\cdots,y_{m-1})$（m≧2，m∈自然数）は$F(X)$の出力を表す変数。）を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点v_iについて、当該非終端節点に関わる変数z_i（$z_i\in(X\cup Y)$）に付与される変数ラベル並びに当該変数z_i（$z_i\in(X\cup Y)$）の値が０のとき及び１のときに遷移する先の子節点を特定する一対の枝$e_0(v_i)$，$e_1(v_i)$からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段；
　及び、ルックアップ・テーブル（Look-Up Table ： LUT）を記憶するためのＬＵＴ記憶手段；
を備えたシステムにおいて、
　前記特性関数二分決定グラフから、前記多出力論理関数$F(X)$に対応する論理回路を構成するためのデータであるルックアップ・テーブルを合成する論理回路合成方法であって、以下のステップを有する論理回路合成方法：
　前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ（以下「分割の高さ」という。）levの設定を行う分割線設定ステップ；
　前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において２つの部分グラフB_0,B_1に分割したときに根節点を含む側の部分グラフB_0に属するものであって、出力を表す変数y_r（∈Y）に関わる節点v_j及びその親節点v_kの節点データについて、当該節点v_jの枝$e_0(v_j)$，$e_1(v_j)$の一方$e_a(v_j)$が$\chi(X,Y)=0$に関わる終端節点を特定する場合、当該節点v_jの親節点v_kの枝$e_0(v_k)$，$e_1(v_k)$のうち当該節点v_jを特定する枝$e_c(v_k)$を、当該節点v_jの前記枝$e_a(v_j)$以外の枝$e_b(v_j)$に置き換える短絡除去処理を実行する短絡除去ステップ；
　前記短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって前記分割の高さlevより高い高さにある非終端節点に属する枝のうち、前記分割の高さlevより低い高さの非終端節点の子節点を特定するものの個数を計数し（但し、同じ節点を特定するものは１つと数え、定数０に向かう枝は無視する。）、その個数を前記分割の高さlevの分割線における幅Wとして出力するＢＤＤ幅計測ステップ；
　前記幅Wに基づき、式（１２）の演算により中間変数の個数uを算出する中間変数算出ステップ；
　前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において２つの部分グラフB_0,B_1に分割したときに根節点を含む側の部分グラフB_0に属するものについて、ルックアップ・テーブルを生成しそれをＬＵＴ記憶手段に格納するＬＵＴ生成ステップ；
　及び、前記中間変数算出ステップにおいて算出された前記中間変数の個数uと等しい制御入力数を有する二分木（binary tree）を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフB_0に属する非終端節点の節点データを、前記二分木を表す節点データで置き換えることにより、特性関数二分決定グラフ

を再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新するＢＤＤ再構成ステップ。

【数１１】

$$\chi(X,Y) = \bigwedge_{i=0}^{m-1}(y_i \equiv f_i(X)) \qquad (11)$$

【数１２】

$$u = \lceil \log_2 W \rceil \qquad (12)$$

【請求項１２】

前記節点テーブル記憶手段には、入力変数を$X=(x_1,\cdots,x_n)$（n∈自然数）とし出力にドント・ケアを含む不完全定義関数である多出力論理関数$F(X)=(f_0(X),\cdots,f_{m-1}(X))$に対して式（１３）により定義される特性関数χ(X,Y)（但し、$Y=(y_0,\cdots,y_{m-1})$（m≧2，m∈自然数）はF(X)の出力を表す変数。）を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点v_iについて、当該非終端節点に関わる変数z_i（z_i∈(X∪Y)）に付与される変数ラベル並びに当該変数z_i（z_i∈(X∪Y)）の値が０のとき及び１のときに遷移する先の子節点を特定する一対の枝$e_0(v_i)$，$e_1(v_i)$からなる節点データのテーブルである節点テーブルとして記憶されていること
を特徴とする請求項１１記載の論理回路合成方法。

【数１３】

$$\chi(X,Y) = \bigwedge_{i=0}^{m-1}\{\bar{y}_i f_{i\_0} \vee y_i f_{i\_1} \vee f_{i\_d}\} \qquad (13)$$

但し、f_{i_0}，f_{i_1}，f_{i_d}は、それぞれ、式（１４）で定義されるOFF関数、ON関数、及びDC関数を表す。

【数１４】

$$f_{i\_0}(X) = \begin{cases} 1 & (X \in f_i^{-1}(0)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i\_1}(X) = \begin{cases} 1 & (X \in f_i^{-1}(1)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i\_d}(X) = \begin{cases} 1 & (X \in f_i^{-1}(d)) \\ 0 & (\text{otherwise}) \end{cases} \qquad (14)$$

【請求項１３】

前記節点テーブル記憶手段に記憶された前記節点テーブルにより表される特性関数二分決定グラフの幅を、請求項８乃至１０の何れかに記載のグラフ幅削減方法によって削減し、前記節点テーブル記憶手段に記憶された前記節点テーブルを更新した後に、前記分割線設定ステップ乃至前記ＢＤＤ再構成ステップを実行する請求項１２記載の論理回路合成方法。

【請求項１４】

前記多出力論理関数F(X)の要素をなす論理関数$f_0(X),\cdots,f_{m-1}(X)$の順序をπ＝（π[0]，…，π[m-1]）（但し、π[i]=jは、f_jがi番目であることを表す。）とし、論理関数f_j（∈F(X)）が依存する入力変数の集合を$supp(f_j)$としたとき、式（１５）で表されるTの値が最小となるように前記多出力論理関数F(X)の要素の順序πを決定する出力変数順序決定ステップ；

前記各入力変数x_i（∈X）及び出力を表す変数y_j（∈Y）の順序を、式（１６）を満たす順序Pに決定する全変数順序決定ステップ；

及び、前記順序Pに従って、特性関数二分決定グラフの節点データを生成し前記節点テーブル記憶手段に格納するＢＤＤ生成ステップ；
を実行した後に、前記分割線設定ステップ乃至前記ＢＤＤ再構成ステップを実行すること

を特徴とする請求項１１乃至１４の何れか一記載の論理回路合成方法。

【数１５】

$$T=\sum_{k=0}^{m-1}\left|\bigcup_{l=0}^{k}\mathrm{supp}(f_{\pi[l]})\right| \tag{15}$$

【数１６】

$$P\quad\left(\mathrm{supp}(f_{\pi[0]}),y_{\pi[0]},\mathrm{supp}(f_{\pi[1]})-\mathrm{supp}(f_{\pi[0]}),y_{\pi[1]},\mathrm{supp}(f_{\pi[2]})-\left(\sum_{k=0}^{1}\mathrm{supp}(f_{\pi[k]})\right),y_{\pi[2]},\right.$$
$$\left.\cdots,\mathrm{supp}(f_{\pi[m-1]})-\left(\sum_{k=0}^{m-2}\mathrm{supp}(f_{\pi[k]})\right),y_{\pi[m-1]}\right) \tag{16}$$

【請求項１５】

コンピュータに請求項８乃至１０の何れかに記載のグラフ幅削減方法を実行させるプログラム。

【請求項１６】

コンピュータに請求項１１乃至１４に何れかに記載の論理回路合成方法を実行させるプログラム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、ルックアップ・テーブル（以下、「ＬＵＴ」という。）・カスケード論理回路やＬＵＴ型ＦＰＧＡ等において使用される、多出力論理関数ｆ（Ｘ）に対応する論理回路を構成するためのデータであるＬＵＴを生成するための論理回路合成技術に関するものである。

【背景技術】

【０００２】

近年では、種々の電子回路の設計において、ＬＵＴ型のフィールド・プログラマブル・ゲートアレイ（Ｆｉｅｌｄ　Ｐｒｏｇｒａｍｍａｂｌｅ　Ｇａｔｅ　Ａｒｒａｙ：以下、「ＦＰＧＡ」という。）が広く使用されている（例えば、非特許文献１～３参照）。ＬＵＴ型ＦＰＧＡは、多数の再構成可能論理ブロック（Ｃｏｎｆｉｇｕｒａｂｌｅ　Ｌｏｇｉｃ　Ｂｌｏｃｋ：以下、「ＣＬＢ」という。）が格子状に配列され、各ＣＬＢの間に縦横に格子状に配線された複数の接続線（ｒｏｕｔｉｎｇ　ｌｉｎｅ：繞線）を備えた構成からなる。各々の接続線の交点には、再構成可能なスイッチ・ブロック（Ｓｗｉｔｃｈ　Ｂｌｏｃｋ：以下「ＳＢ」という。）が配されており、各々の接続線の接続を自由に変更（再構成）することができる。また、各ＣＬＢと接続線とは、再構成可能な接続ブロック（Ｃｏｎｎｅｃｔｉｏｎ　Ｂｌｏｃｋ：以下、「ＣＢ」という。）により接続されている。そして、各接続線の終端には、ＦＰＧＡ外部との入出力を行う入出力部（Ｉ／Ｏ）が設けられている。各ＣＬＢは、内部に多入力１出力のＬＵＴやマルチプレクサ（ＭＵＸ）を１ないし複数個有し、このＬＵＴやＭＵＸにより論理演算が可能である。ＬＵＴ型ＦＰＧＡでは、ＳＢ、ＣＢ及びＣＬＢの内部を再構成することにより、所望の多入力１出力論理関数が格納された複数のＬＵＴの入出力を目的に応じてネットワーク状に順序接続し、様々な組み合わせ論理回路を実現することができる。

【０００３】

一方、ＦＰＧＡよりも高速な再構成可能論理回路を実現するものとして、ＬＵＴカスケード論理回路が提案されている（例えば、非特許文献４，５参照）。ＬＵＴカスケード論理回路においては、多入力多出力のＬＵＴがカスケード状に接続された構成からなる。各ＬＵＴには、外部からの入力に加えて前段のＬＵＴの出力が入力される。そして、最終段

のＬＵＴから１ないし複数の出力変数が出力される。演算を行おうとしている論理関数を複数の多出力論理関数に分解し、各多出力論理関数は各段のＬＵＴに格納される。このようにして、ＬＵＴカスケード論理回路においては多出力論理関数の演算を行うことができる。

【０００４】

上記のようなＬＵＴ型ＦＰＧＡやＬＵＴカスケード論理回路を、実際の論理回路設計に適用する場合には、まず、論理回路で実現しようとする論理関数（以下、「目的論理関数」という。）を複数の論理関数に関数分解して合成論理関数とする。ここで、合成論理関数とは、関数分解により得られる複数の論理関数（以下、「分解関数」という。）の結合論理からなる関数をいう。すなわち、目的論理関数をｆ（Ｘ）（｛Ｘ｝は入力変数の集合）で表した場合に、この目的論理関数を、ｆ（X_1，X_2）＝ｇ（ｈ（X_1），X_2）（$\{X_1\} \subset \{X\}$，$\{X_2\} \subset \{X\}$，$\{X_1\} \cap \{X_2\} = \phi$，$\{X_1\} \cup \{X_2\} = \{X\}$）の形に関数分解した場合、ｇ（ｈ（$X_1$），$X_2$）をｇとｈの合成論理関数という。

【０００５】

尚、本明細書において、Ｘ，Ｙと記す場合には、入力変数，出力変数のベクトル又は全順序集合（ｏｒｄｅｒｅｄ　ｓｅｔ）を表し、｛Ｘ｝，｛Ｙ｝と記すときは、入力変数，出力変数の非順序集合（ｕｎｏｒｄｅｒｅｄ　ｓｅｔ）を表すものとする。

【０００６】

上記関数分解により、２つの分解関数ｈ（X_1），ｇ（ｈ，X_2）が得られる。尚、かかる関数分解は常に可能であるとは限らないが、コンピュータ等で使用する制御回路や算術演算回路等の多くの実用的関数は分解可能なものが多い（非特許文献６参照）。また、分解関数ｇ（ｈ，X_2）が更に分解可能であれば同様の関数分解を繰り返す。

【０００７】

そして、各分解関数を別々のＬＵＴとして実現し、各ＬＵＴを合成論理関数に従ってネットワーク状又はカスケード状に接続する。このようにして、目的論理関数をＬＵＴ型ＦＰＧＡやＬＵＴカスケード論理回路によって実現することができる。

【０００８】

単一出力の目的論理関数を、上記関数分解の手法を用いて、ＬＵＴが結合した論理回路により実現することは比較的容易である（例えば、非特許文献４，５参照）。

【０００９】

一方、目的論理関数が多出力である場合において、目的論理関数をＬＵＴが結合した論理回路により実現する論理合成技術としては、現在のところ以下の方法によるものが知られている。

（１）多端子二分決定グラフ（Ｍｕｌｔｉ－Ｔｅｒｍｉｎａｌ　Ｂｉｎａｒｙ　Ｄｅｃｉｓｉｏｎ　Ｄｉａｇｒａｍ：ＭＴＢＤＤ）を用いる方法（非特許文献７，８参照）

（２）出力を幾つかのグループに分割して構成する方法（非特許文献７参照）

（３）分割理論を用いる方法（非特許文献９，１０参照）

（４）代入による方法（非特許文献１１参照）

（５）Ｈｙｐｅｒ　Ｆｕｎｃｔｉｏｎを用いる方法（非特許文献１２参照）

（６）非零出力符号化特性関数（Ｅｎｃｏｄｅｄ　Ｃｈａｒａｃｔｅｒｉｓｔｉｃ　Ｆｕｎｃｔｉｏｎ　ｆｏｒ　Ｎｏｎ－ｚｅｒｏ：ＥＣＦＮ）を用いた時分割実現による方法（非特許文献４，５参照）

（７）以上のうちいくつかの方法の組み合わせによる方法（非特許文献１１参照）

【非特許文献１】Ｓ．Ｂｒｏｗｎ，Ｒ．Ｊ．Ｆｒａｎｃｉｓ，Ｊ．Ｒｏｓｅ，ａｎｄ　Ｚ．Ｇ．Ｖｒａｎｅｓｉｃ，"Ｆｉｅｌｄ－Ｐｒｏｇｒａｍｍａｂｌｅ　Ｇａｔｅ　Ａｒｒａｙｓ"，Ｋｌｕｗｅｒ　Ａｃａｄｅｍｉｃ　Ｐｕｂｌｉｓｈｅｒｓ，１９９２．

【非特許文献２】Ｐ．Ｃｈｏｗ，Ｓ．Ｏ．Ｓｅｏ，Ｊ．Ｒｏｓｅ，Ｋ．Ｃｈｕｎｇ，Ｇ．Ｐａｅｚ－Ｍｏｎｚｏｎ，ａｎｄ　Ｉ．Ｒａｈａｒｄｊａ，"Ｔｈｅ　ｄｅｓｉｇｎ　ｏｆ　ａｎ　ＳＲＡＭ－ｂａｓｅｄ　ｆｉｅｌｄ－ｐｒｏｇｒａｍｍａｂｌｅ　ｇａｔｅ

array－－－Part I：Architecture，"IEEE Transactions on VLSI Systems，vol. 7，pp. 191－197，June 1999.
【非特許文献3】Chow，P.，S. Seo，J. Rose，K. Chung，G. Pez－Monzn and I. Rahardja. "The Design of an SRAM－Based Field－Programmable Gate Array，Part II：Circuit Design and Layout"，IEEE Transactions on Very Large Scale Integration（VLSI）Systems，Vol. 7 No. 3，Sept. 1999，pp. 321－330.
【非特許文献4】笹尾勤，松浦宗寛，井口幸洋，“多出力関数のカスケード実現と再構成可能ハードウェアによる実現”，電子情報通信学会ＦＴＳ研究会，ＦＴＳ２００１－８，pp. ５７－６４，三重大学（２００１－０４）.
【非特許文献5】T. Sasao，M. Matsuura，and Y. Iguchi，"A cascade realization of multiple－output fulnction for reconfigurable hardware，"International Workshop on Logic and Synthesis（IWLS01），Lake Tahoe，CA，June 12－15，2001. pp. 225－230.
【非特許文献6】T. Sasao and M. Matsuura，"DECOMPOS：An integrated system for functional decomposition"，1998 International Workshop on Logic Synthesis，Lake Tahoe，June 1998.
【非特許文献7】Y－T. Lai，M. Pedram and S. B. K. Vrudhula，"BDD based decomposition of logic functions with application to FPGA synthesis"，30th ACM／IEEE Design Automation Conference，June 1993.
【非特許文献8】T. Sasao，"FPGA design by generalized functional decomposition"，In Logic Synthesis and Optimization，Sasao ed.，Kluwer Academic Publisher，pp. 233－258，1993.
【非特許文献9】C. Scholl and P. Molitor，"Communication based FPGA synthesis for multi－output Boolean functions"，Asia and South Pacific Design Automation Conference，pp. 279－287，Aug. 1995.
【非特許文献10】B. Wurth，K. Eckl，and K. Anterich，"Functional multiple－output decomposition：Theory and implicit algorithm"，Design Automation Conf.，pp. 54－59，June 1995.
【非特許文献11】H. Sawada，T. Suyama，and A. Nagoya，"Logic synthesis for look－up table based FPGAs using functional decomposition and support minimization"，ICCAD，pp. 353－359，Nov. 1995.
【非特許文献12】J. －H. R. Jian，J. －Y. Jou，and J. －D. Huang，"Compatible class encoding in hyper－function decomposition for FPGA synthesis"，Design Automation Conference，pp. 712－

７１７，Ｊｕｎｅ １９９８．
【非特許文献１３】Ｐ．Ａｓｈａｒ ａｎｄ Ｓ．Ｍａｌｉｋ，"Ｆａｓｔ ｆｕｎｃｔｉｏｎａｌ ｓｉｍｕｌａｔｉｏｎ ｕｓｉｎｇ ｂｒａｎｃｈｉｎｇ ｐｒｏｇｒａｍｓ"，Ｐｒｏｃ．Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｃｏｍｐｕｔｅｒ Ａｉｄｅｄ Ｄｅｓｉｇｎ，ｐｐ．４０８－４１２，Ｎｏｖ．１９９５．
【非特許文献１４】Ｃ．Ｓｃｈｏｌｌ，Ｒ．Ｄｒｅｃｈｓｌｅｒ，ａｎｄ Ｂ．Ｂｅｃｋｅｒ，"Ｆｕｎｃｔｉｏｎａｌ ｓｉｍｕｌａｔｉｏｎ ｕｓｉｎｇ ｂｉｎａｒｙ ｄｅｃｉｓｉｏｎ ｄｉａｇｒａｍ"，ＩＣＣＡＤ’９７，ｐｐ．８－１２，Ｎｏｖ．１９９７．
【非特許文献１５】Ａ．Ｍｉｓｈｃｈｅｎｋｏ ａｎｄ Ｔ．Ｓａｓａｏ，"Ｌｏｇｉｃ Ｓｙｎｔｈｅｓｉｓ ｏｆ ＬＵＴ Ｃａｓｃａｄｅｓ ｗｉｔｈ Ｌｉｍｉｔｅｄ Ｒａｉｌｓ"，電子情報通信学会ＶＬＳＩ設計技術研究会，ＶＬＤ２００２－９，琵琶湖（２００２－１１）
【非特許文献１６】Ｍ．Ｒ．Ｇａｒｅｙ，Ｄ．Ｓ．Ｊｏｈｎｓｏｎ，"Ｃｏｍｐｕｔｅｒｓ ａｎｄ Ｉｎｔｒａｃｔａｂｉｌｉｔｙ：Ａ Ｇｕｉｄｅ ｔｏ ｔｈｅ Ｔｈｅｏｒｙ ｏｆ ＮＰ－Ｃｏｍｐｌｅｔｅｎｅｓｓ"，Ｗ．Ｈ．Ｆｒｅｅｍａｎ ＆ Ｃｏ．，Ｎｅｗ Ｙｏｒｋ，１９７９．

【発明の開示】
【発明が解決しようとする課題】
【００１０】
上記背景技術に挙げた（１）～（７）の方法による論理合成技術は、何れも多出力論理関数の論理合成の技術としては決定的な方法であるとはいえない。例えば、ＭＴＢＤＤを用いて多出力論理関数の関数分解を行うことは、理論上は可能である。しかしながら、ｍ出力の論理関数の場合、最大で2^m個の終端節点が存在する。従って、ＭＴＢＤＤを用いて多出力論理関数の関数分解を行った場合には、多くの場合、列複雑度（後述の〔定義６〕参照）が増大し、実用的ではない。また、他の従来の方法に関しても同様なことがいえる。
【００１１】
一方、比較器や加算器のように、すべての論理演算が終了しない段階においても、演算を進めるに従って、順次出力変数が決定していくような論理関数も多い。例えば、入力変数$P=(p_1, p_2, p_3, p_4)$（$P \in B^4$，$B=\{0, 1\}$）及び$Q=(q_1, q_2, q_3, q_4)$（$Q \in B^4$）を加算して出力変数$R=(r_0, r_1, r_2, r_3, cout)$（$R \in B^5$）を出力するような加算器では、各桁の加算が終了した時点で、すべての桁の演算が終了する前にその桁に対応する出力変数が決定する。このように、すべての論理演算が終了しない段階で決定し出力される出力変数を「中間出力変数」という。
【００１２】
目的論理関数が中間出力変数を有する場合、目的論理関数を実現するＬＵＴネットワークやＬＵＴカスケードにおいては、最終段以外の段のＬＵＴにおいて中間出力変数を出力するように構成することにより、各ＬＵＴのサイズを小さくするとともに演算速度を向上させることが可能となる。
【００１３】
しかしながら、上記背景技術に挙げた論理合成技術では、中間出力変数を有する目的論理関数に対して、中間出力を有するＬＵＴネットワークやＬＵＴカスケードを合成することは困難であった。
【００１４】
また、基数変換回路や加算回路などにおいては、各出力値は必ずしもすべての入力に対して定義されておらず、一部の入力変数に対しては出力値が０でも１でもよいようなドント・ケアを含む場合が多い。このようなドント・ケアを含む多出力論理関数をそのままＬＵＴネットワークやＬＵＴカスケードとして合成すると、メモリの冗長性が大きくなる。従って、ドント・ケアを含む多出力論理関数についてはできるだけＬＵＴネットワークや

ＬＵＴカスケードの冗長性が少なくなる形式に変更することが好ましい。

【００１５】

そこで、本発明の第１の目的は、一般的な多出力論理関数に対して汎用的に論理合成が可能であり、また、中間出力を有するＬＵＴネットワークやＬＵＴカスケードを合成することが可能な論理回路合成技術を提供することにある。

【００１６】

また、本発明の第２の目的は、ドント・ケアを含む多出力論理関数を、ＬＵＴネットワークやＬＵＴカスケードの冗長性が少なくなる形式に変更する一手法であるグラフ幅削減技術を提供することにある。

【課題を解決するための手段】

【００１７】

以下の説明では、まず本明細書で使用する主要な用語について定義し、本発明の基本的原理について説明した後、本発明の構成及び作用について説明する。

【００１８】

〔１〕用語の定義等

〔定義１〕（サポート変数）

関数Ｆが変数ｘに依存するとき、ｘをＦの「サポート変数（ｓｕｐｐｏｒｔ　ｖａｒｉａｂｌｅ）」という。関数Ｆのサポート変数の集合をｓｕｐｐ（Ｆ）と記す。

〔定義終わり〕

【００１９】

〔定義２〕（完全定義関数）

ブール変数をｘと記し、ｘ全体の集合をＢ＝｛０，１｝と記す。論理変数Ｘ（$\in B^n$，ｎは任意の自然数）の関数Ｆ（Ｘ）：$B^n \rightarrow B^m$（ｍは任意の自然数）を「完全定義関数（ｃｏｍｐｌｅｔｅｌｙ　ｓｐｅｃｉｆｉｅｄ　ｆｕｎｃｔｉｏｎ）」という。特に、ｍが２以上の自然数の場合を「完全定義多出力関数（ｃｏｍｐｌｅｔｅｌｙ　ｓｐｅｃｉｆｉｅｄ　ｍｕｌｔｉｐｌｅ－ｏｕｔｐｕｔ　ｆｕｎｃｔｉｏｎ）」といい、ｍ＝１の場合を「完全定義単一出力関数（ｃｏｍｐｌｅｔｅｌｙ　ｓｐｅｃｉｆｉｅｄ　ｓｉｎｇｌｅ－ｏｕｔｐｕｔ　ｆｕｎｃｔｉｏｎ）」という。

〔定義終わり〕

【００２０】

〔定義３〕（不完全定義関数）

ある関数ｆ（Ｘ）の関数値が０でも１でもよいことを「ドント・ケア（ｄｏｎ’ｔ　ｃａｒｅ）」といい、ｄと記す。論理変数Ｘ（$\in B^n$，ｎは任意の自然数）の関数Ｆ（Ｘ）：$B^n \rightarrow \{0, 1, d\}^m$（ｍは任意の自然数）を「不完全定義関数（ｉｎｃｏｍｐｌｅｔｅｌｙ　ｓｐｅｃｉｆｉｅｄ　ｆｕｎｃｔｉｏｎ）」という。特に、ｍが２以上の自然数の場合を「不完全定義多出力関数（ｉｎｃｏｍｐｌｅｔｅｌｙ　ｓｐｅｃｉｆｉｅｄ　ｍｕｌｔｉｐｌｅ－ｏｕｔｐｕｔ　ｆｕｎｃｔｉｏｎ）」といい、ｍ＝１の場合を「不完全定義単一出力関数（ｉｎｃｏｍｐｌｅｔｅｌｙ　ｓｐｅｃｉｆｉｅｄ　ｓｉｎｇｌｅ－ｏｕｔｐｕｔ　ｆｕｎｃｔｉｏｎ）」という。

〔定義終わり〕

【００２１】

〔定義４〕（変数の分割）

論理関数の入力変数を$X=(x_1, x_2, \cdots, x_n)$とする。Ｘの変数の集合を｛Ｘ｝で表す。$\{X_1\} \cup \{X_2\} = \{X\}$かつ$\{X_1\} \cap \{X_2\} = \phi$のとき、$X=(X_1, X_2)$を「Ｘの分割（ｐａｒｔｉｔｉｏｎ）」という。また、Ｘの変数の個数を｜Ｘ｜で表す。

〔定義終わり〕

【００２２】

〔定義５〕（分解表）

論理関数Ｆ（Ｘ）に対して、Ｘの分割を$X=(X_1, X_2)$とする。また、$|X_1| =$

n_1，$|X_2|=n_2$とする。このとき、2^{n1}列2^{n2}行の表で、各行、各列に２進符号のラベルを持ち、その要素がＦの真理値であるような表を、Ｆの「分解表（ｄｅｃｏｍｐｏｓｉｔｉｏｎ　ｃｈａｒｔ）」という。このとき、X_1を「束縛変数（ｂｏｕｎｄ　ｖａｒｉａｂｌｅ）」、X_2を「自由変数（ｆｒｅｅ　ｖａｒｉａｂｌｅ）」という。ここで、分解表の列、行は、それぞれn_1，n_2ビットのすべてのパターンを有する。

〔定義終わり〕

【００２３】

〔定義６〕（列複雑度、列関数）

分解表の異なる列パターンの数を「列複雑度（ｃｏｌｕｍｎ　ｍｕｌｔｉｐｌｉｃｉｔｙ）」といい、μと記す。列パターンが表す関数を「列関数（ｃｏｌｕｍｎ　ｆｕｎｃｔｉｏｎ）」という。

〔定義終わり〕

【００２４】

〔例１〕

（表１）に４入力１出力不完全定義関数の分解表を示す。すべての列パターンが異なるので、列複雑度μは４となる。

【００２５】

【表１】

$X_1=\{x_1,x_2\}$

$X_2=\{x_3,x_4\}$	00	01	10	11
00	0	0	d	1
01	1	1	d	d
10	d	1	0	d
11	0	d	0	0
	Φ_1	Φ_2	Φ_3	Φ_4

〔例終わり〕

【００２６】

〔定義７〕（ＯＮ関数、ＯＦＦ関数、ＤＣ関数）

ｆ（Ｘ）＝０，１，ｄを満たすすべてのＸの集合を、それぞれ、「ＯＦＦ集合（ＯＦＦ－ｓｅｔ）」、「ＯＮ集合（ＯＮ－ｓｅｔ）」、「ＤＣ集合（ＤＣ－ｓｅｔ）」といい、$f^{-1}(0)$，$f^{-1}(1)$，$f^{-1}(d)$と記す。論理変数Ｘ（$\in B^n$）の不完全定義関数をｆ（Ｘ）とする。ＯＦＦ集合$f^{-1}(0)$に属するすべてのＸに１を対応づけ、それ以外のＸに０を対応づける関数を「ＯＦＦ関数（ＯＦＦ－ｆｕｎｃｔｉｏｎ）」といい、f_0と記す。

【００２７】

【数１】

$$f_0(X)=\begin{cases}1 & (X\in f^{-1}(0))\\ 0 & (\text{otherwise})\end{cases} \qquad (1)$$

【００２８】

ＯＮ集合f^{-1}（１）に属するすべてのＸに１を対応づけ、それ以外のＸに０を対応づける関数を「ＯＮ関数（ＯＮ－ｆｕｎｃｔｉｏｎ）」といい、f_1と記す。

【００２９】

【数２】

$$f_{\_1}(X)=\begin{cases}1 & (X\in f^{-1}(1))\\ 0 & (\text{otherwise})\end{cases} \qquad (2)$$

【００３０】

ＤＣ集合ｆ－１（ｄ）に属するすべてのＸに１を対応づけ、それ以外のＸに０を対応づける関数を「ＤＣ関数（ＤＣ－ｆｕｎｃｔｉｏｎ）」といい、ｆ＿ｄと記す。

【００３１】

【数３】

$$f_{\_d}(X)=\begin{cases}1 & (X\in f^{-1}(d))\\ 0 & (\text{otherwise})\end{cases} \qquad (3)$$

【００３２】

ｆ（Ｘ）が完全定義関数の場合にも、ＯＦＦ関数ｆ＿０，ＯＮ関数ｆ＿１を同様に定義する。特にｆ（Ｘ）が完全定義単一出力関数の場合、

【００３３】

【数４】

$$f_{\_0}=\bar{f}(X),\quad f_{\_1}=f(X) \qquad (4)$$

である。

〔定義終わり〕

【００３４】

〔定義８〕（完全定義関数の特性関数）

入力変数をＸ＝（ｘ１，ｘ２，…，ｘｎ）（∈Ｂｎ）とし、完全定義多出力関数をＦ（Ｘ）＝（ｆ０（Ｘ），ｆ１（Ｘ），…，ｆｍ－１（Ｘ））（ｍは自然数）とする。このとき、以下の式（５）ように定義される関数ｘ（Ｘ，Ｙ）を多出力論理関数Ｆの「特性関数（ｃｈａｒａｃｔｅｒｉｓｔｉｃ　ｆｕｎｃｔｉｏｎ）」という（非特許文献１４参照）。ここで、Ｙ＝（ｙ０，…，ｙｍ－１）は出力を表す変数である。

【００３５】

【数５】

$$\chi(X,Y)=\bigwedge_{i=0}^{m-1}(y_i\equiv f_i(X)) \qquad (5)$$

〔定義終わり〕

【００３６】

ｎ入力ｍ出力関数の特性関数は（ｎ＋ｍ）変数の二値論理関数であり、入力変数ｘｉ（ｉ＝１，２，…，ｎ）の他に、各出力ｆｊ（ｊ＝０，１，…，ｍ－１）に対して出力変数ｙｊを用いる。Ｂ＝｛０，１｝とする。Ｘ∈Ｂｎかつ Ｆ（Ｘ）＝（ｆ０（Ｘ），ｆ１（Ｘ），…，ｆｍ－１（Ｘ））∈Ｂｍとする。いま、Ｙ∈Ｂｍとすると、論理関数Ｆ（Ｘ）の特性関数ｘ（Ｘ，Ｙ）の値は式（６）のようになる。

【００３７】

【数６】

$$\chi(\boldsymbol{X},\boldsymbol{Y})=\begin{cases}1 & (\boldsymbol{Y}=F(\boldsymbol{X}))\\ 0 & (\boldsymbol{Y}\neq F(\boldsymbol{X}))\end{cases} \qquad (6)$$

【００３８】
完全定義多出力関数の特性関数は、入力の組み合わせと出力の組み合わせのうち許されているものを表す。式（７）のようにおくと、特性関数ｘは式（８）のように表現することができる。
【００３９】
【数７】

$$f_{i\_0}(X) = \bar{f}_i(X), \quad f_{i\_1}(X) = f_i(X) \tag{7}$$

【００４０】
【数８】

$$\chi(X,Y) = \bigwedge_{i=0}^{m-1} (\bar{y}_i \cdot f_{i\_0} \vee y_i \cdot f_{i\_1}) \tag{8}$$

【００４１】
一方、不完全定義関数では、ドント・ケアの場合、その関数値が０でも１でもよい。従って、特性関数では、出力変数ｙ$_{i}$の値が０でも１でも真となる。そこで、ドント・ケアを表現する不定値関数ｆ$_{i_d}$に対して、次式が成立する。
【００４２】
【数９】

$$\bar{y}_i(f_{i\_0} \vee f_{i\_d}) \vee y_i(f_{i\_1} \vee f_{i\_d}) = \bar{y}_i f_{i\_0} \vee y_i f_{i\_1} \vee f_{i\_d} \tag{9}$$

【００４３】
式（８）、式（９）より、完全定義多出力関数の特性関数を拡張して、不完全定義関数の特性関数ｘを次のように定義する。
【００４４】
〔定義９〕（不完全定義関数の特性関数）
入力変数Ｘ＝（ｘ$_{1}$，ｘ$_{2}$，…，ｘ$_{n}$）（∈Ｂ^{n}）の不完全定義多出力関数をＦ（Ｘ）＝（ｆ$_{0}$（Ｘ），ｆ$_{1}$（Ｘ），…，ｆ$_{m-1}$（Ｘ））とする。このとき、以下の式（１０）により定義される関数ｘ（Ｘ，Ｙ）を不完全定義多出力関数Ｆの特性関数という。ここで、Ｙ＝（ｙ$_{0}$，…，ｙ$_{m-1}$）は出力を表す変数である。
【００４５】
【数１０】

$$\chi(X,Y) = \bigwedge_{i=0}^{m-1} \{\bar{y}_i f_{i\_0} \vee y_i f_{i\_1} \vee f_{i\_d}\} \tag{10}$$

〔定義終わり〕
【００４６】
〔例２〕
（表２）に示した４入力２出力の不完全定義関数を考える。
【００４７】

【表２】

x_1	x_2	x_3	x_4	f_0	f_1
0	0	0	0	d	1
0	0	0	1	d	1
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	d	d
0	1	0	1	d	d
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	1	d
1	1	0	1	1	d
1	1	1	0	d	0
1	1	1	1	d	1

（表２）より、各ＯＦＦ関数、ＯＮ関数、ＤＣ関数は次のように記述される。

【００４８】

【数１１】

$$
\begin{aligned}
f_{0\_0} &= \bar{x}_1\bar{x}_2x_3 \vee x_1\bar{x}_2\bar{x}_3 \\
f_{0\_1} &= \bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3 \\
f_{0\_d} &= \bar{x}_1\bar{x}_3 \vee x_1x_2x_3 \\
f_{1\_0} &= \bar{x}_1\bar{x}_2x_3 \vee x_1\bar{x}_2x_3 \vee x_2x_3\bar{x}_4 \\
f_{1\_1} &= \bar{x}_2\bar{x}_3 \vee x_2x_3x_4 \\
f_{1\_d} &= x_2\bar{x}_3
\end{aligned}
\qquad (11)
$$

従って、この関数の特性関数は、次式のようになる。

【００４９】

【数１２】

$$
\begin{aligned}
\chi = {} & \{\bar{y}_0(\bar{x}_1\bar{x}_2x_3 \vee x_1\bar{x}_2\bar{x}_3) \vee y_0(\bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3) \vee (\bar{x}_1\bar{x}_3 \vee x_1x_2x_3)\} \cdot \\
& \{\bar{y}_1(\bar{x}_1\bar{x}_2x_3 \vee x_1\bar{x}_2x_3 \vee x_2x_3\bar{x}_4) \vee y_1(\bar{x}_2\bar{x}_3 \vee x_2x_3x_4) \vee (x_2\bar{x}_3)\}
\end{aligned}
\qquad (12)
$$

〔例終わり〕

【００５０】

〔定義１０〕（特性関数二分決定グラフ）

多出力論理関数Ｆ（Ｘ）＝（ｆ$_0$（Ｘ），ｆ$_1$（Ｘ），…，ｆ$_{m-1}$（Ｘ））の「特性関数二分決定グラフ（Ｂｉｎａｒｙ Ｄｅｃｉｓｉｏｎ Ｄｉａｇｒａｍ ｆｏｒ Ｃｈａｒａｃｔｅｒｉｓｔｉｃ Ｆｕｎｃｔｉｏｎ：ＢＤＤ ｆｏｒ ＣＦ）」とは、多出力論理関数Ｆ（Ｘ）の特性関数ｘ（Ｘ，Ｙ）を表現する二分決定グラフ（Ｂｉｎａｒｙ Ｄ

ｅｃｉｓｉｏｎ Ｄｉａｇｒａｍ：ＢＤＤ）をいう。但し、二分決定グラフの変数は、根節点を最上位としたとき、変数y_jはf_jに依存する入力変数$x_i \in supp(f_j)$よりも下位に置く。ここに、ｓｕｐｐ（f_j）はf_jの依存変数の集合を表す。（非特許文献１３，１４参照）
〔定義終わり〕
【００５１】
〔例３〕
図１（ａ）の真理値表によって表現される多出力論理関数の特性関数二分決定グラフは、図１（ｂ）により表される。ここで、○は変数節点を表し、□は特性関数値の節点を表す。
〔例終わり〕
【００５２】
〔定義１１〕（短絡除去）
特性関数二分決定グラフにおいて、出力を表す節点y_j（∈Ｙ）から出る枝（ｅｄｇｅ）のうち、定数０に接続する枝を取り除き、節点y_jの親節点（ｐａｒｅｎｔ ｎｏｄｅ）から節点y_jの定数０以外の子節点（ｃｈｉｌｄ ｎｏｄｅ）へ直接枝を繋ぐ。この作業を、y_jを表現する総ての節点に対して実行する操作を、「出力変数y_jの短絡除去（ｒｅｍｏｖａｌ ｏｆ ｔｈｅ ｏｕｔｐｕｔ ｖａｒｉａｂｌｅｙ＿ｉ ｂｙ ｓｈｏｒｔｉｎｇ）」という。
〔定義終わり〕
【００５３】
具体的に図２を用いて短絡除去の説明をする。図２（ａ）に示したような出力を表す節点y_j（∈Ｙ）を考える。節点y_jの親節点をz_p、節点y_jの子節点をz_qとする。まず、特性関数値０に接続する節点を除去すると、図２（ｂ）のようになる。次に、節点y_jの親節点z_pから子節点Ｚz_qへ直接枝を繋ぐことにより、図２（ｃ）のようにグラフが変形される。節点y_jに関する短絡除去では、以上のような操作を、特性関数二分決定グラフの中にある出力を表す節点y_jのすべてについて行うものである。
【００５４】
〔定義１２〕（特性関数二分決定グラフの幅）
（z_{n+m}，z_{n+m-1}，…，z_1）を特性関数二分決定グラフの変数順序とする。ここで、z_{n+m}は、根節点に対応する。特性関数二分決定グラフの高さｋでの「幅」とは、変数z_kと変数z_{k+1}との間の枝の本数をいう。ここで、同じ節点に接続している枝は一つと数え、定数０に向かう枝は無視する。また、特性関数二分決定グラフの高さ０での幅を１と定義する。
〔定義終わり〕
【００５５】
変数z_iの節点の高さ（ｌｅｖｅｌ）とは、二分決定グラフにおいて順序づけられた変数において、終端節点から数えた順番をいう。但し、終端節点の高さは０とする。すなわち、ｎ＋ｍ個の変数｛z_i；ｉ＝１，…，ｎ＋ｍ｝を有する二分決定グラフにおいて根節点から終端節点にかけて順序づけられた変数順序をＰ＝（z_{n+m}，z_{n+m-1}，…，z_1）とすると、変数z_iの節点の高さはｉとなる。このとき、変数z_i，z_jについてｉ＞ｊのとき、変数z_iは変数z_jよりも高位（ｈｉｇｈ ｌｅｖｅｌ）である（又は変数z_jは変数z_iよりも低位である）という。また、高さｉでの幅は、変数z_iと変数z_{i+1}との間の枝の本数である。但し、同じ節点を特定するものは１つと数え、定数０に向かう枝は無視する。
【００５６】
〔例４〕
図１（ｂ）の特性関数二分決定グラフにおいては、終端節点（特性関数値がラベル付けられた節点）の高さが０、変数y_2の高さが１、変数y_1の高さが２、変数x_4の高さが３、…、変数x_1の高さが７となる。また、高さ１での幅は、変数y_1の節点から変数ｙ

$_2$の節点へ向かう枝の本数（但し、終端節点０へ向かう枝は無視するものとする。）で２である。また、高さ２での幅は、変数ｘ$_4$の節点から変数ｙ$_1$の節点へ向かう枝の本数で４である。

〔例終わり〕

【００５７】

ＢＤＤを用いた関数分解において、ＢＤＤの幅は列複雑度μに等しい。（〔注意〕列複雑度を考えるときは、元の多出力関数を考える（表３の分解表参照）。ＢＤＤを考えるときは、多出力関数の特性関数を考える。単一出力論理関数の時は、必然的に、出力変数は、一番下に来る。定数１に至るＢＤＤの部分が、もとの関数のＢＤＤと殆ど変わらない。）変数の分割（Ｘ$_1$，Ｘ$_2$）において、変数Ｘ$_1$の節点に直接接続されている変数Ｘ$_2$の節点は、分解表の列パターンに対応する。

【００５８】

関数分解では、ｆ（Ｘ$_1$，Ｘ$_2$）＝ｇ（ｈ（Ｘ$_1$），Ｘ$_2$）と表現する。

【００５９】

【数１３】

$$\lceil \log_2 \mu \rceil < |X_1| \qquad (13)$$

が成立するとき、Ｘ$_1$を入力とし列関数の符号を生成するｈ（Ｘ$_1$）の回路と、ｇ（ｈ，Ｘ$_2$）の回路を個別に実現することによって、関数ｆを実現する（図３参照）。関数分解は分解後の関数ｇの入力数が少ないほど効果的である。ＢＤＤの幅を削減できれば、列複雑度が減り、関数分解の効果が上がる。

【００６０】

〔定義１３〕（ドント・ケアへの定数の割り当て）

２つの不完全定義列関数をＦ$^{(1)}$＝（ｆ$_0{}^{(1)}$，ｆ$_1{}^{(1)}$…，ｆ$_{m-1}{}^{(1)}$），Ｆ$^{(2)}$＝（ｆ$_0{}^{(2)}$，ｆ$_1{}^{(2)}$，…，ｆ$_{m-1}{}^{(2)}$）とする。このとき、ｆ$_i{}^{(1)}$＝ｄ又はｆ$_i{}^{(2)}$＝ｄであるＦ$^{(1)}$，Ｆ$^{(2)}$の各要素に対して、次のような論理積演算ｆ$_i{}^{(1)}$・ｆ$_i{}^{(2)}$を行うことを「ドント・ケアへの定数の割り当て（ｄｏｎ’ｔ　ｃａｒｅ　ａｓｓｉｇｎｍｅｎｔ）」という。

【００６１】

【数１４】

$$f_i^{(1)} \cdot f_i^{(2)} = \begin{cases} 1 & (\text{if } [f_i^{(1)} = 1 \wedge f_i^{(2)} = d] \vee [f_i^{(1)} = d \wedge f_i^{(2)} = 1]) \\ 0 & (\text{if } [f_i^{(1)} = 0 \wedge f_i^{(2)} = d] \vee [f_i^{(1)} = d \wedge f_i^{(2)} = 0]) \\ d & (\text{if } f_i^{(1)} = d \wedge f_i^{(2)} = d) \end{cases} \qquad (14)$$

〔定義終わり〕

【００６２】

〔定義１４〕（関数の両立）

二つの不完全定義関数ｆ$_a$とｆ$_b$にドント・ケアへの定数の割り当てを行うことで、同じ関数にできる場合、ｆ$_a$とｆ$_b$は「両立する（ｃｏｍｐａｔｉｂｌｅ）」といい、ｆ$_a$～ｆ$_b$と記す。

〔定義終わり〕

【００６３】

〔補題１〕

三つの不完全定義関数の特性関数をｘ$_a$，ｘ$_b$とする。ｘ$_a$～ｘ$_b$且つｘ$_c$＝ｘ$_a$ｘ$_b$のとき、ｘ$_c$～ｘ$_a$且っｘ$_c$～ｘ$_b$が成立する。

〔補題終わり〕

【００６４】

〔例５〕

（表１）の分解表で、列関数の対｛Φ$_1$，Φ$_2$｝，｛Φ$_1$，Φ$_3$｝，｛Φ$_3$，Φ$_4$｝は両立する。列Φ$_1$とΦ$_2$の論理積をとり、Φ$_1$＊とΦ$_2$＊と置き換え、列Φ$_3$とΦ$_4$の論理積をとり、Φ$_3$＊とΦ$_4$＊と置き換えると（表３）のようになり、μ＝２となる。

【００６５】

【表３】

$X_1 = \{x_1, x_2\}$

$X_2 = \{x_3, x_4\}$	00	01	10	11
00	0	0	1	1
01	1	1	d	d
10	1	1	0	0
11	0	0	0	0
	Φ_1^*	Φ_2^*	Φ_3^*	Φ_4^*

〔例終わり〕

【００６６】

〔定義１５〕（両立グラフ）

関数を節点とし、両立する関数同士を枝で接続したグラフを「両立グラフ（ｃｏｍｐａｔｉｂｌｅ　ｇｒａｐｈ）」という。

〔定義終わり〕

【００６７】

〔２〕本発明の基本的原理

（１）不完全定義多出力論理関数の特性関数二分決定グラフの幅の削減

最初に、不完全定義多出力論理関数を表現する特性関数二分決定グラフの幅を削減する方法について説明する。一つの方法としては、特性関数二分決定グラフの１つの節点に着目し、その節点の２つの子節点が両立する場合、２つの子節点を併合して一つにすることを繰り返し、節点数を削減する方法が考えられる。この方法のアルゴリズムは、以下のアルゴリズム１のようになる。

【００６８】

〔アルゴリズム１〕

特性関数二分決定グラフの根節点から、再帰的に以下の処理を行う。

（１）節点ｖがドント・ケアを含まなければ終了；

（２）節点ｖの２つの子節点ｖ$_0$，ｖ$_1$が両立するか調べる；

・両立しなければ、ｖ$_0$，ｖ$_1$に対して再帰する；

・両立すれば、ｖ$_0$，ｖ$_1$をｖ$_{new}$に置き換える。ｖ$_{new}$に対して再帰する。但し、ｖ$_0$，ｖ$_1$，ｖ$_{new}$を表す特性関数をｘ$_0$，ｘ$_1$，ｘ$_{new}$とすれば、ｘ$_{new}$＝ｘ$_0$・ｘ$_1$。

【００６９】

〔例６〕

（表４）に４入力２出力不完全定義関数の真理値表を示す。

【００７０】

【表４】

x_1	x_2	x_3	x_4	f_0	f_1
0	0	0	0	d	1
0	0	0	1	d	1
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	d	d
0	1	0	1	d	d
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	1	d
1	1	0	1	1	d
1	1	1	0	d	0
1	1	1	1	d	1

【００７１】

（表４）の真理値表によって表される不完全定義多出力関数を特性関数二分決定グラフにより表すと、図４（ａ）のようになる。図４（ａ）では、節点１，２が両立する子節点を持っている。この特性関数二分決定グラフに対してアルゴリズム１を適用すると、図４（ｂ）のようになる。この関数では、節点１を置き換える節点と、節点２を置き換える節点が同じになるので、図４（ｂ）では、節点１，２が節点３に置き換えられている。図４の特性関数二分決定グラフの左にあるＷｉｄｔｈは、各高さでの幅である。最大幅は８から５に削減され、非終端節点の数は１５から１２に削減されている。

〔例終わり〕

【００７２】

アルゴリズム１のグラフ幅削減方法は、局所的な節点数の削減には効果がある。しかし、一つの節点の子節点同士でしか両立性を考えていないため、全体的な特性関数二分決定グラフの幅の削減にはあまり効果的ではない。そこで、次に、特性関数二分決定グラフの幅を構成するすべての部分関数の両立性を調べ、マッチングを行うことで直接幅を削減するグラフ幅削減方法を示す。

【００７３】

関数分解において、すべての列関数の両立性を調べ、両立グラフを作成し、最小数の完全部分グラフ（クリーク）による節点被覆を行うことで、列複雑度μの最小化が可能である。しかしながら、この節点被覆の問題はＮＰ困難であることが知られている（非特許文献１６参照）。そこで、ここでは次のアルゴリズム２で表される発見的手法を使用する。

【００７４】

〔アルゴリズム２〕（クリーク集合による節点被覆）

与えられた両立グラフのすべての節点の集合をS_aとする。Ｃを節点集合の集合とする。枝を持たない節点をS_aから除去し、要素が１つの節点集合（クリーク）としてＣの要素とする。

$S_a \neq \phi$の間、以下の（１）～（３）の操作を繰り返す：

（１）S_aの中で枝数最小の節点を調べ、これをv_iとする。$S_i \leftarrow \{v_i\}$とする。S_aの中でv_iに接続している節点の集合をS_bとする；
（２）$S_b \neq \phi$の間、以下の（ａ），（ｂ）の操作を繰り返す；
（ａ）S_bの中で枝数最小の節点v_jを調べ、$S_i \leftarrow S_i \cup \{v_j\}$，$S_b \leftarrow S_b - \{v_j\}$とする；
（ｂ）v_jに接続していない節点をS_bから除去する；
（３）$C \leftarrow C \cup \{S_i\}$，$S_a \leftarrow S_a - S_i$。

【００７５】

両立グラフ上におけるクリーク集合による節点被覆を利用して、次のアルゴリズム３により、特性関数二分決定グラフの幅の削減を行うことができる。

【００７６】

〔アルゴリズム３〕（特性関数二分決定グラフの幅削減）
与えられた特性関数二分決定グラフの高さをｔとし、定数節点の高さを０とする。高さｔ－１から１までの間、以下の操作を繰り返す：
（１）各高さにおけるすべての列関数の集合を作る；
（２）列関数の集合から両立グラフを作成する；
（３）クリーク集合による節点被覆（例えば、アルゴリズム２）を行う；
（４）各クリークに被覆されているすべての節点に対応する列関数に対して、ドント・ケアへの定数の割り当てを行い、一つの列関数（割当列関数）にする；
（５）各列関数を（４）で作成した割当列関数に置換して、特性関数二分決定グラフを再構成する。

【００７７】

このグラフ幅削減方法によれば、特性関数のすべての部分関数の両立性を全体的に評価して最適なドント・ケアへの定数の割り当てができる。従って、特性関数二分決定グラフの幅を効果的に削減することが可能となる。

【００７８】

（２）目的論理関数の関数分解とＬＵＴ回路の生成
次に、特性関数二分決定グラフからＬＵＴ論理回路を合成する手法について、その原理を説明する。本発明におけるＬＵＴ論理回路の合成は、特性関数二分決定グラフの分割と短絡除去とを利用して行う。特性関数二分決定グラフを分割して２つの回路を構成した場合、両回路を接続する接続線数は、次の〔定理１〕により算出することができる。

【００７９】

〔定理１〕
X_1，X_2を入力変数の集合、Y_1，Y_2を出力変数の集合とする。特性関数二分決定グラフの変数順序を（X_1，Y_1，X_2，Y_2）とするとき、特性関数二分決定グラフの高さ｜X_2｜＋｜Y_2｜における幅をＷとする。ここで、Ｗを数える際、出力を表現する変数から、定数０へ向かう枝は無視する。多出力論理関数を図３の回路で実現する場合、二つの回路ＨとＧの間の必要かつ十分な接続線数ｕは、式（１５）により表される。

【００８０】

【数１５】

$$u = \lceil \log_2 W \rceil \qquad (15)$$

〔定理終わり〕

【００８１】

（証明）
特性関数二分決定グラフの構成法から、出力関数Y_1とY_2が、図３の回路で表現できることは明らかである。特性関数二分決定グラフにおいて、Y_1の出力を表現する節点を短絡除去すると、Y_1以外の関数を表現する特性関数二分決定グラフが得られる。また、この操作によって、特性関数二分決定グラフの幅が増えることはない。高さ｜X_2｜＋｜

Y_2｜におけるＢＤＤの幅をＷとする。いま、関数分解ｇ（ｈ（X_1），X_2）の分解表を考えると、その列複雑度はＷに等しい。従って、回路Ｈと回路Ｇの間の配線数は、

【００８２】

【数１６】

$$\lceil \log_2 W \rceil \tag{16}$$

だけあれば十分である。また、分解表の列複雑度はＷなので、二つのブロック間の配線は少なくとも式（１６）の本数だけ必要である。

（証明終わり）

【００８３】

特性関数二分決定グラフの分割及び短絡除去と上記〔定理１〕とを利用して、次のように目的論理関数の関数分解が行われる。

【００８４】

まず、目的論理関数Ｆ＝（f_0，f_1，…，f_{m-1}）の特性関数二分決定グラフの変数順序が（X_1，Y_1，X_2，Y_2），Y_1＝（y_0，y_1，…，y_{k-1}），Y_2＝（y_k，y_{k+1}，…，y_{m-1}）であるとする。この場合、$y_j = f_j$（X_1）（ｊ＝０，１，…，ｋ－１）は図３の回路Ｈで直接実現する。一方、高さ｜X_2｜＋｜Y_2｜における特性関数二分決定グラフの幅をＷとする。Ｗ本の枝にｕビット（ｕは式（１５）により得られる数。）の異なる二進数を割り当てる。二つのブロックＨ，Ｇ間を接続する枝が実現する関数をｈ$_1$，ｈ$_2$，…，ｈ$_u$とすると、ブロックＧの出力関数Y_2は（ｈ$_1$，ｈ$_2$，…，ｈ$_u$，X_2）を入力変数とする論理関数として表現可能であり、その特性関数二分決定グラフは図５のようになる。

【００８５】

〔例７〕

本例では、２つの２ビットの２進数を加算する回路（ＡＤＲ２）を、中間出力を有する関数分解を用いて実現する場合について説明する。ＡＤＲ２の入出力の関係は式（１７）のように定義できる。これより、ｓ$_0$，ｓ$_1$，ｓ$_2$は、それぞれ式（１８）により表される。また、真理値表は図６（ａ）により表される。

【００８６】

【数１７】

$$\begin{array}{cccc} & & a_1 & a_0 \\ +) & & b_1 & b_0 \\ \hline & s_2 & s_1 & s_0 \end{array} \tag{17}$$

【００８７】

【数１８】

$$\begin{aligned} s_0 &= a_0 \oplus b_0 \\ s_1 &= a_0 b_0 \oplus (a_1 \oplus b_1) \\ s_2 &= a_0 b_0 (a_1 \vee b_1) \vee a_1 b_1 \end{aligned} \tag{18}$$

【００８８】

変数の分割として、X_1＝（ａ$_0$，ｂ$_0$），Y_1＝（ｓ$_0$），X_2＝（ａ$_1$，ｂ$_1$），Y_2＝（ｓ$_1$，ｓ$_2$）とおく。このとき、変数順序は、（X_1，Y_1，X_2，Y_2）＝（ａ$_0$，ｂ$_0$，ｓ$_0$，ａ$_1$，ｂ$_1$，ｓ$_1$，ｓ$_2$）となる。従って、ＡＤＲ２の特性関数二分決定グラフは、図６（ｂ）により表される。Ｚ＝（Z_A，Z_B），Z_A＝（X_1，Y_1），Z_B＝（X_2，Y_2）のように分割し、Z_Aを図３の回路Ｈ、Z_Bを図３の回路Ｇにより構成する。このとき、高さ｜Z_B｜での特性関数二分決定グラフの幅Ｗは２となる。従

って、回路Ｈと回路Ｇとを連結する線は、式（１５）より、１本あればよい。出力s_0は、X_1のみの関数として表現することが可能である。

【００８９】

また、変数組Z_Aにより構成される特性関数二分決定グラフは、図７（ａ）のようになる。これは、図６の特性関数二分決定グラフの分割線よりも根節点側（上側）の部分を切り取ったものである。尚、図７（ａ）では特性関数値０につながる枝は省略してある。回路Ｈと回路Ｇとを連結する線は１本なので、図７（ａ）のグラフの終端節点には、１ビットの中間変数h_1を導入して、各終端節点に割り当てる。なお、変数h_1への符号の割り当ては任意である。この特性関数二分決定グラフは、容易に多端子二分決定グラフ（ＭＴＢＤＤ）に変換することができる。図７（ａ）のグラフを変換することにより得られるＭＴＢＤＤは、図７（ｂ）のようになる。図７（ｂ）のＭＴＢＤＤから回路Ｈに相当するＬＵＴを図７（ｃ）のように生成することができる。

【００９０】

次に、図６の特性関数二分決定グラフについて、先に導入した新しい中間変数h_1を用いて、図６（ｂ）に示した特性関数二分決定グラフの分割線よりも上部を、h_1を入力変数とする決定木（ｄｅｃｉｓｉｏｎ　ｔｒｅｅ）で置き換える。これにより、特性関数二分決定グラフは図８（ａ）のように変形される。尚、図８（ａ）でも特性関数値０につながる枝は必要ないので省略してある。この特性関数二分決定グラフは、容易にＭＴＢＤＤに変換することができる。図８（ａ）のグラフを変換することにより得られるＭＴＢＤＤは、図８（ｂ）のようになる。更に、図８（ｂ）のＭＴＢＤＤから回路Ｇに相当するＬＵＴを図９（ａ）のように生成することができる。従って、結局、図９（ｂ）に示したようなＬＵＴカスケードを構成することができる。

〔例終わり〕

【００９１】

〔３〕本発明の構成及び作用

本発明に係るグラフ幅削減装置の第１の構成は、入力変数を$X=(x_1, \cdots, x_n)$（n∈自然数）とし出力にドント・ケアを含む不完全定義関数である多出力論理関数$F(X)=(f_0(X), \cdots, f_{m-1}(X))$に対して式（１９）により定義される特性関数$x(X, Y)$（但し、$Y=(y_0, \cdots, y_{m-1})$（$m \geqq 2$，$m$∈自然数）はＦ（Ｘ）の出力を表す変数。）を表現する特性関数二分決定グラフの幅を削減するグラフ幅削減装置であって、以下の構成を備えていることを特徴とする：

多出力論理関数Ｆ（Ｘ）の特性関数二分決定グラフのそれぞれの非終端節点v_iについて、当該非終端節点に関わる変数z_i（$z_i \in (X \cup Y)$）に付与される変数ラベル並びに当該変数z_i（$z_i \in (X \cup Y)$）の値が０のとき及び１のときに遷移する先の子節点を特定する一対の枝$e_0(v_i)$，$e_1(v_i)$からなる節点データのテーブルである節点テーブルを記憶する節点テーブル記憶手段；

前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ（以下「分割の高さ」という。）ｌｅｖの設定を行う分割線設定手段；

前記節点データ記憶手段に記憶された前記節点テーブルから、前記特性関数二分決定グラフを前記分割線設定手段により設定された前記分割の高さｌｅｖで分割し関数分解することによって得られる分解表の列を表す関数である列関数を生成する列関数生成手段；

及び、前記列関数生成手段が生成する各列関数のうち、両立する列関数に対してドント・ケアに定数を割り当てることによって同一の列関数（以下「割当列関数（ａｓｓｉｇｎｅｄ　ｃｏｌｕｍｎ　ｆｕｎｃｔｉｏｎ）」という。）とするとともに、これらの新たな割当列関数を用いて前記特性関数二分決定グラフ再構成し前記節点テーブル記憶手段の節点テーブルを更新する割当ＢＤＤ再構成手段。

【００９２】

【数１９】

$$\chi(X,Y)=\bigwedge_{i=0}^{m-1}\{\bar{y}_i f_{i\_0}\vee y_i f_{i\_1}\vee f_{i\_d}\} \qquad (19)$$

但し、ｆｉ＿０，ｆｉ＿１，ｆｉ＿ｄは、それぞれ、式（２０）で定義されるＯＦＦ関数、ＯＮ関数、及びＤＣ関数を表す。

【００９３】

【数２０】

$$f_{i\_0}(X)=\begin{cases}1 & (X\in f_i^{-1}(0))\\ 0 & (\text{otherwise})\end{cases},\ f_{i\_1}(X)=\begin{cases}1 & (X\in f_i^{-1}(1))\\ 0 & (\text{otherwise})\end{cases},\ f_{i\_d}(X)=\begin{cases}1 & (X\in f_i^{-1}(d))\\ 0 & (\text{otherwise})\end{cases} \qquad (20)$$

【００９４】

この構成により、分割線設定手段が分割の高さｌｅｖを設定すると、列関数生成手段が特性関数二分決定グラフを分割の高さｌｅｖで分割し関数分解することによって得られる列関数を生成する。そして、割当ＢＤＤ再構成手段は、各列関数のうち、両立する列関数に対してドント・ケアに定数を割り当てることによって同一の割当列関数とする。それとともに、これらの新たな割当列関数を用いて特性関数二分決定グラフ再構成し節点テーブル記憶手段の節点テーブルを更新する。これにより、分割の高さｌｅｖにおける特性関数二分決定グラフの幅が削減される。

【００９５】

一般に、二分決定グラフを用いて関数分解を行う場合、二分決定グラフの幅が小さいほど、関数分解後の論理回路も小さくなる。従って、上述のようなドント・ケアへの定数の割り当てを実行することによって、二分決定グラフの幅を削減し、関数分解後の論理回路を小さくすることができる。

【００９６】

本発明に係るグラフ幅削減装置の第２の構成は、前記第１の構成において、

前記列関数を節点（以下「関数節点」という。）とするグラフであって、互いに両立する複数の列関数に対応する関数節点同士が枝（以下「両立枝」という。）により連結されたグラフである両立グラフを、前記各関数節点の列関数ラベル及び当該関数節点に連結する両立枝のデータ（以下「関数節点データ」という。）のテーブルとして記憶するための両立グラフ記憶手段；

前記両立グラフ記憶手段に記憶された前記各関数節点データに対する列関数から、両立する列関数の組を選出し、それら両立する列関数に対する関数節点データにそれらの関数節点同士を連結する両立枝を追加して前記両立グラフ記憶手段に記憶された関数節点データの更新を行う両立枝生成手段；

及び、前記両立グラフのすべての節点について、完全部分グラフ（以下「クリーク」という。）による節点被覆を行うことにより、クリークに含まれる関数節点集合であるクリーク・データを生成するクリーク生成手段；

を備え、

前記列関数生成手段は、前記節点データ記憶手段に記憶された前記節点テーブルから、前記分割線設定手段により設定された前記分割の高さｌｅｖの節点の各枝に対応する列関数を生成し、それら各列関数に対応した列関数ラベルを有する前記関数節点データを生成して前記両立グラフ記憶手段に保存するものであり、

前記割当ＢＤＤ再構成手段は、前記クリーク生成手段が生成する前記各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数とすることにより、前記特性関数二分決定グラフを再構成し前記節点テーブル記憶手段の前記節点テーブルを更新するものであることを特

徴とする。

【００９７】

この構成によれば、列関数生成手段が、節点データ記憶手段に記憶された節点テーブルから、分割の高さｌｅｖの節点の各枝に対応する列関数を生成する。そして、それら各列関数に対応した列関数ラベルを有する前記関数節点データを生成して前記両立グラフ記憶手段に保存する。両立枝生成手段は、両立グラフ記憶手段に記憶された各関数節点データに対する列関数から、両立する列関数の組を選出する。そして、選出された両立する列関数に対する関数節点データに、それらの関数節点同士を連結する両立枝を追加して関数節点データの更新を行う。次に、クリーク生成手段は、両立グラフのすべての節点について、最小数のクリークによる節点被覆を行い、クリーク・データを生成する。最後に、割当ＢＤＤ再構成手段は、各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数とする。そして、割当列関数により、特性関数二分決定グラフを再構成し、節点テーブルを更新する。

【００９８】

以上のようにして、上述のアルゴリズム３の（１）～（５）を実行し、列関数の両立グラフのクリーク被覆による特性関数二分決定グラフの幅の削減を行うことができる。本発明によれば、特性関数二分決定グラフの幅を構成するすべての部分関数の両立性を、両立グラフを用いて全体的に評価して、ドント・ケアへの定数の割り当てを行うことができるため、極めて効果的に特性関数二分決定グラフの幅を削減することが可能となる。

【００９９】

ここで、クリーク生成手段は、クリークを生成する場合に、両立グラフのすべての節点について、最小数のクリークにより節点被覆を行うことが望ましい。しかしながら、前述したように、最小数のクリークによる節点被覆問題はＮＰ困難であり実用的ではない。従って、実際にはクリーク生成手段は、発見的方法によってできるだけ少ないクリーク数で節点被覆を行うことができるようにクリークを生成するように構成するのがよい。

【０１００】

本発明に係るグラフ幅削減装置の第３の構成は、前記第１又は第２の構成において、前記分割線設定手段は、前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフの根節点の子の節点の高さから下位に向かって順次分割の高さｌｅｖの設定を行うものであり、前記割当ＢＤＤ再構成手段は、前記分割線設定手段が設定する前記各分割の高さｌｅｖにおいて逐次前記特性関数二分決定グラフ再構成を行うことを特徴とする。

【０１０１】

この構成によれば、特性関数二分決定グラフの根節点の高さｔの子節点の高さｔ－１から定数節点の上の節点の高さ１まで、上述したアルゴリズム３による特性関数二分決定グラフの幅の削減を行うことができる。従って、多出力論理関数のすべてのサポート変数に対して効果的に特性関数二分決定グラフの幅を削減することが可能となる。

【０１０２】

本発明に係る論理回路合成装置の第１の構成は、入力変数を$X=(x_1, \cdots, x_n)$（n∈自然数）とする多出力論理関数$F(X)=(f_0(X), \cdots, f_{m-1}(X))$の特性関数二分決定グラフから、前記多出力論理関数Ｆ（Ｘ）に対応する論理回路を構成するためのデータであるルックアップ・テーブルを生成する論理回路合成装置であって、以下の構成を備えていることを特徴とする：

完全定義関数である前記多出力論理関数$F(X)=(f_0(X), \cdots, f_{m-1}(X))$に対して式（２１）により定義される特性関数ｘ（Ｘ，Ｙ）（但し、$Y=(y_0, \cdots, y_{m-1})$（$m \geqq 2$，m∈自然数）はＦ（Ｘ）の出力を表す変数。）を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点v_iについて、当該非終端節点に関わる変数z_i（$z_i \in (X \cup Y)$）に付与される変数ラベル並びに当該変数z_i（$z_i \in (X \cup Y)$）の値が０のとき及び１のときに遷移する先の子節点を特

定する一対の枝ｅ$_0$（ｖ$_i$），ｅ$_1$（ｖ$_i$）からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段；

前記ルックアップ・テーブルを記憶するＬＵＴ記憶手段；

前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ（以下「分割の高さ」という。）ｌｅｖの設定を行う分割線設定手段；

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さｌｅｖの分割線において２つの部分グラフＢ$_0$，Ｂ$_1$に分割したときに根節点を含む側の部分グラフＢ$_0$に属するものであって、出力を表す変数ｙ$_r$（∈Ｙ）に関わる節点ｖ$_j$及びその親節点ｖ$_k$の節点データについて、当該節点ｖ$_j$の枝ｅ$_0$（ｖ$_j$），ｅ$_1$（ｖ$_j$）の一方ｅ$_a$（ｖ$_j$）がｘ（Ｘ，Ｙ）＝０に関わる終端節点を特定する場合、当該節点ｖ$_j$の親節点ｖ$_k$の枝ｅ$_0$（ｖ$_k$），ｅ$_1$（ｖ$_k$）のうち当該節点ｖ$_j$を特定する枝ｅ$_c$（ｖ$_k$）を、当該節点ｖ$_j$の前記枝ｅ$_a$（ｖ$_j$）以外の枝ｅ$_b$（ｖ$_j$）に置き換える短絡除去処理を行う短絡除去手段；

前記短絡除去手段により短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって前記分割の高さｌｅｖより高い高さにある非終端節点に属する枝のうち、前記分割の高さｌｅｖより低い高さの非終端節点の子節点を特定するものの個数を計数し（但し、同じ節点を特定するものは１つと数え、定数０に向かう枝は無視する。）、その個数を前記分割の高さｌｅｖの分割線における幅Ｗとして出力するＢＤＤ幅計測手段；

前記ＢＤＤ幅計測手段が出力する幅Ｗに基づき、式（２２）の演算により中間変数の個数ｕを算出する中間変数算出手段；

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さｌｅｖの分割線において２つの部分グラフＢ$_0$，Ｂ$_1$に分割したときに根節点を含む側の部分グラフＢ$_0$に属するものについて、ルックアップ・テーブルを生成しそれをＬＵＴ記憶手段に格納するＬＵＴ生成手段；

及び、前記中間変数算出手段が算出する前記中間変数の個数ｕと等しい制御入力数を有する二分木（ｂｉｎａｒｙ　ｔｒｅｅ）を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフＢ$_0$に属する非終端節点の節点データを、前記二分木を表す節点データで置き換えることにより、特性関数二分決定グラフを再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新するＢＤＤ再構成手段。

【０１０３】

【数２１】

$$\chi(X,Y)=\bigwedge_{i=0}^{m_t-1}(y_i\equiv f_i(X)) \qquad (21)$$

【０１０４】

【数２２】

$$u=\lceil \log_2 W \rceil \qquad (22)$$

【０１０５】

この構成によれば、

（ａ）まず、論理回路合成を行う目的論理関数の特性関数二分決定グラフの節点データを節点テーブル記憶手段に格納しておく。短絡除去手段は、節点テーブル記憶手段に格納された節点データで構成される特性関数二分決定グラフについて、所定の高さｌｅｖの分割線に対して根節点を含む側の部分グラフＢ$_0$について短絡除去処理を行う。短絡除去処理については、上記「〔２〕本発明の基本的原理」の欄において説明した通りである。ＢＤ

Ｄ幅計測手段は、上記短絡除去処理が行われた結果得られる特性関数二分決定グラフについて、上記分割線における幅Ｗを計測する。そして、中間変数算出手段は、式（２２）によって中間変数の個数ｕを算出する。

【０１０６】

（ｂ）次に、ＬＵＴ生成手段は、節点テーブル記憶手段に格納された節点データで構成される特性関数二分決定グラフについて、所定の高さｌｅｖの分割線に対して根節点を含む側の部分グラフ$Ｂ_{0}$について、ルックアップ・テーブルを生成する。この生成方法の原理については、上記「〔１〕本発明の基本的原理」の欄において説明した通りである。このＬＵＴの出力は、部分グラフ$Ｂ_{0}$に属する出力変数$Ｙ_{0}$＝（$ｙ_{0}$，…，$ｙ_{k-1}$）及びｕ個の中間変数Ｈ＝（$ｈ_{1}$，…，$ｈ_{u}$）となる。生成されたルックアップ・テーブルは、ＬＵＴ記憶手段に格納される。中間変数Ｈの節点に対しては、適当な符号を割り当てることができる。

【０１０７】

（ｃ）最後に、ＢＤＤ再構成手段は、ｕ個の制御入力数を有する二分木を生成し、各制御入力数に対して上記中間変数Ｈ＝（$ｈ_{1}$，…，$ｈ_{u}$）を対応させる。そして、ＢＤＤ再構成手段は、この処理により再構成された特性関数二分決定グラフの各節点の節点データを節点テーブル記憶手段に格納し、節点テーブル記憶手段が保持する特性関数二分決定グラフを更新する。

【０１０８】

以上のような（ａ）～（ｃ）の処理を繰り返せば、目的論理関数は複数の部分関数に関数分解され、ＬＵＴカスケード論理回路が構成される。また、ＬＵＴ生成手段により生成されるルックアップ・テーブルを特性関数二分決定グラフにして、更に同様の処理によって関数分解を行うことによって、ＬＵＴネットワーク論理回路が構成される。

【０１０９】

本構成による論理回路合成装置では、分割線の位置を、部分グラフ$Ｂ_{0}$の大きさが過度に大きくならないように適度な高さに決めることで、部分グラフ$Ｂ_{0}$に関する変数を束縛変数とした分解表の列複雑度が極度に増大することを防止することが可能である。従って、比較的少容量のメモリを用いて論理回路合成装置を実現することが可能である。また、部分グラフ$Ｂ_{0}$に関する変数を束縛変数とした分解表の列複雑度を抑えることができるため、計算処理量やメモリ・アクセス回数を少なくすることが可能であり、論理関数分解処理の演算処理速度を高速化することができる。

【０１１０】

尚、短絡除去手段による短絡除去処理において、実際に節点テーブルの構造自体を書き換えることによって短絡除去処理を行ってもよいが、節点テーブルのデータ構造は変化させず、特性関数に分決定グラフを解釈するプログラムにおいて或る節点に短絡処理がなされたものとみなすようにすることもできる。例えば、短絡処理がなされた出力変数の節点については節点ラベルを特定の値に書き換える。そうすることにより、節点ラベルを調べることで、その変数が出力か否かが判定できるようにして、事実上短絡処理がなされたと見なすことができるようにすることができる。

【０１１１】

本発明に係る論理回路合成装置の第２の構成は、前記第１の構成において、前記節点テーブル記憶手段には、出力にドント・ケアを含む不完全定義関数である前記多出力論理関数Ｆ（Ｘ）＝（$ｆ_{0}$（Ｘ），…，$ｆ_{m-1}$（Ｘ））に対して式（１９）により定義される特性関数ｘ（Ｘ，Ｙ）（但し、Ｙ＝（$ｙ_{0}$，…，$ｙ_{m-1}$）（ｍ≧２，ｍ∈自然数）はＦ（Ｘ）の出力を表す変数。）を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点$ｖ_{i}$について、当該非終端節点に関わる変数$ｚ_{i}$（$ｚ_{i}$∈（Ｘ∪Ｙ））に付与される変数ラベル並びに当該変数$ｚ_{i}$（$ｚ_{i}$∈（Ｘ∪Ｙ））の値が０のとき及び１のときに遷移する先の子節点を特定する一対の枝$ｅ_{0}$（$ｖ_{i}$），$ｅ_{1}$（$ｖ_{i}$）からなる節点データのテーブルである節点テーブルとして記憶されていることを特徴とする。

【０１１２】
この構成により、不完全定義多出力関数についても、完全定義多出力関数と同様、ＬＵＴカスケード論理回路やＬＵＴネットワーク論理回路を構成することが可能となる。
【０１１３】
本発明に係る論理回路合成装置の第２の構成は、前記第２の構成において、前記第１乃至３の何れか一の構成のグラフ幅削減装置を備え、前記短絡除去手段は、前記グラフ幅削減装置によって前記節点テーブル記憶手段に記憶された前記節点テーブルにより表される特性関数二分決定グラフの幅を削減し更新された前記節点テーブルについて短絡除去処理を行うものであることを特徴とする。
【０１１４】
この構成により、不完全定義関数の特性関数二分決定グラフの幅を削減した上でＬＵＴカスケードの合成を行うため、不完全定義多出力関数の冗長性を削減して、ＬＵＴカスケードのメモリ使用効率を改善することができる。
【０１１５】
本発明に係る論理回路合成装置の第４の構成は、前記第１乃至３の何れか一の構成において、
前記多出力論理関数Ｆ（Ｘ）の要素をなす論理関数f_0（Ｘ），…，f_{m-1}（Ｘ）の順序をπ＝（π［０］，…，π［ｍ－１］）（但し、π［ｉ］＝ｊは、f_jがｉ番目であることを表す。）とし、論理関数f_j（∈Ｆ（Ｘ））が依存する入力変数の集合をｓｕｐｐ（f_j）としたとき、式（２３）で表されるＴの値が最小となるように前記多出力論理関数Ｆ（Ｘ）の要素の順序πを決定する出力変数順序決定手段；
前記各入力変数x_i（∈Ｘ）及び出力を表す変数y_j（∈Ｙ）の順序を、式（２４）を満たす順序Ｐに決定する全変数順序決定手段；
及び、前記全変数順序決定手段で決定された順序Ｐに従って、特性関数二分決定グラフの節点データを生成し前記節点テーブル記憶手段に格納するＢＤＤ生成手段；
を備えていることを特徴とする。
【０１１６】
【数２３】

$$T = \sum_{k=0}^{m-1} \left| \bigcup_{l=0}^{k} \mathrm{supp}(f_{\pi[l]}) \right| \qquad (23)$$

【０１１７】
【数２４】

$$P = \left(\mathrm{supp}(f_{\pi[0]}), y_{\pi[0]}, \mathrm{supp}(f_{\pi[1]}) - \mathrm{supp}(f_{\pi[0]}), y_{\pi[1]}, \mathrm{supp}(f_{\pi[2]}) - \left(\sum_{k=0}^{1} \mathrm{supp}(f_{\pi[k]}) \right), y_{\pi[2]}, \right.$$
$$\left. \cdots, \mathrm{supp}(f_{\pi[m-1]}) - \left(\sum_{k=0}^{m-2} \mathrm{supp}(f_{\pi[k]}) \right), y_{\pi[m-1]} \right) \qquad (24)$$

【０１１８】
実用的な多出力論理関数の特性関数二分決定グラフは、通常はかなりの大きさとなる。そのため、出力を分割する方法が必要である。そこで、上記構成によれば、出力変数順序決定手段が、論理関数f_0（Ｘ），…，f_{m-1}（Ｘ）の順序を式（２３）で表されるＴの値が最小となるように決定することにより、ＢＤＤ生成手段が生成する特性関数二分決定グラフの節点データの数を削減できる。特性関数二分決定グラフにおいて、各出力変数$y_j = f_j$（Ｘ）に対して、その出力変数が依存する入力変数が、その出力変数y_jよりも上位に置かれるため、依存変数がなるべく増えないような順序に並べられ、一つずつ出力を増やしながら特性関数二分決定グラフが作られる。このようにして、出力を最適に分

割する。このとき、特性関数二分決定グラフの節点データの数が削減できるので、その後の論理回路合成に要する処理時間が短縮され、高速な論理回路合成が可能となる。

【０１１９】

本発明に係るグラフ幅削減方法の第１の構成は、入力変数をＸ＝（x_1，…，x_n）（ｎ∈自然数）とし出力にドント・ケアを含む不完全定義関数である多出力論理関数Ｆ（Ｘ）＝（f_0（Ｘ），…，f_{m-1}（Ｘ））に対して式（１９）により定義される特性関数χ（Ｘ，Ｙ）（但し、Ｙ＝（y_0，…，y_{m-1}）（ｍ≧２，ｍ∈自然数）はＦ（Ｘ）の出力を表す変数。）を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点v_iについて、当該非終端節点に関わる変数z_i（z_i∈（Ｘ∪Ｙ））に付与される変数ラベル並びに当該変数z_i（z_i∈（Ｘ∪Ｙ））の値が０のとき及び１のときに遷移する先の子節点を特定する一対の枝e_0（v_i），e_1（v_i）からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段を備えたシステムにおいて、当該特性関数二分決定グラフの幅を削減するグラフ幅削減方法であって、以下の各ステップを有することを特徴とする：

前記節点テーブルによって表現される特性関数二分決定グラフを分割する高さ（以下「分割の高さ」という。）ｌｅｖの設定を行う分割線設定ステップ；

前記節点データ記憶手段に記憶された前記節点テーブルから、前記特性関数二分決定グラフを前記分割線設定ステップで設定された前記分割の高さｌｅｖで分割し関数分解することによって得られる分解表の列を表す関数である列関数を生成する列関数生成ステップ；

及び、前記列関数生成ステップにおいて生成される各列関数のうち、両立する列関数に対してドント・ケアに定数を割り当てることによって同一の列関数（以下「割当列関数」という。）にするとともに、これらの新たな割当列関数を用いて前記特性関数二分決定グラフ再構成し前記節点テーブル記憶手段の節点テーブルを更新する割当ＢＤＤ再構成ステップ。

【０１２０】

本発明に係るグラフ幅削減方法の第２の構成は、前記第１の構成において、前記システムは、前記列関数を節点（以下「関数節点」という。）とするグラフであって、互いに両立する複数の列関数に対応する関数節点同士が枝（以下「両立枝」という。）により連結されたグラフである両立グラフを、前記各関数節点の列関数ラベル及び当該関数節点に連結する両立枝のデータ（以下「関数節点データ」という。）のテーブルとして記憶するための両立グラフ記憶手段を備えており、

前記列関数生成ステップにおいては、前記節点データ記憶手段に記憶された前記節点テーブルから、前記分割線設定ステップで設定された前記分割の高さｌｅｖの節点の各枝に対応する列関数を生成し、それら各列関数に対応した列関数ラベルを有する前記関数節点データを生成して前記両立グラフ記憶手段に保存するとともに、

前記両立グラフ記憶手段に記憶された前記各関数節点データに対する列関数から、両立する列関数の組を選出し、それら両立する列関数に対する関数節点データにそれらの関数節点同士を連結する両立枝を追加して前記両立グラフ記憶手段に記憶された関数節点データの更新を行う両立枝生成ステップ；

及び、前記両立グラフのすべての節点について、完全部分グラフ（以下「クリーク」という。）による節点被覆を行うことにより、クリークに含まれる関数節点集合であるクリーク・データを生成するクリーク生成ステップ；

を有し、

前記割当ＢＤＤ再構成ステップにおいては、前記クリーク生成ステップにおいて生成される前記各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数とすることにより、前記特性関数二分決定グラフを再構成し前記節点テーブル記憶手段の前記節点テーブルを更新することを特徴とする。

【０１２１】

本発明に係るグラフ幅削減方法の第３の構成は、前記第１又は第２の構成において、前記分割線設定ステップにおける前記分割の高さｌｅｖを、前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフの根節点の子の節点の高さから最下位の高さまで順次変更しながら、前記分割線設定ステップ乃至前記割当ＢＤＤ再構成ステップを実行することを特徴とする。

【０１２２】

本発明に係る論理回路合成方法の第１の構成は、入力変数をＸ＝（x_1，…，x_n）（ｎ∈自然数）とする完全定義関数である多出力論理関数Ｆ（Ｘ）＝（f_0（Ｘ），…，f_{m-1}（Ｘ））に対して式（２１）により定義される特性関数ｘ（Ｘ，Ｙ）（但し、Ｙ＝（y_0，…，y_{m-1}）（ｍ≧２，ｍ∈自然数）はＦ（Ｘ）の出力を表す変数。）を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点v_iについて、当該非終端節点に関わる変数z_i（z_i∈（Ｘ∪Ｙ））に付与される変数ラベル並びに当該変数z_i（z_i∈（Ｘ∪Ｙ））の値が０のとき及び１のときに遷移する先の子節点を特定する一対の枝e_0（v_i），e_1（v_i）からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段；

及び、ルックアップ・テーブルを記憶するためのＬＵＴ記憶手段；

を備えたシステムにおいて、前記特性関数二分決定グラフから、前記多出力論理関数Ｆ（Ｘ）に対応する論理回路を構成するためのデータであるルックアップ・テーブルを合成する論理回路合成方法であって、以下のステップを有することを特徴とする：

前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ（以下「分割の高さ」という。）ｌｅｖの設定を行う分割線設定ステップ；

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さｌｅｖの分割線において２つの部分グラフB_0，B_1に分割したときに根節点を含む側の部分グラフB_0に属するものであって、出力を表す変数y_r（∈Ｙ）に関わる節点v_j及びその親節点v_kの節点データについて、当該節点v_jの枝e_0（v_j），e_1（v_j）の一方e_a（v_j）がｘ（Ｘ，Ｙ）＝０に関わる終端節点を特定する場合、当該節点v_jの親節点v_kの枝e_0（v_k），e_1（v_k）のうち当該節点v_jを特定する枝e_c（v_k）を、当該節点v_jの前記枝e_a（v_j）以外の枝e_b（v_j）に置き換える短絡除去処理を実行する短絡除去ステップ；

前記短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって前記分割の高さｌｅｖより高い高さにある非終端節点に属する枝のうち、前記分割の高さｌｅｖより低い高さの非終端節点の子節点を特定するものの個数を計数し（但し、同じ節点を特定するものは１つと数え、定数０に向かう枝は無視する。）、その個数を前記分割の高さｌｅｖの分割線における幅Ｗとして出力するＢＤＤ幅計測ステップ；

前記幅Ｗに基づき、式（２２）の演算により中間変数の個数ｕを算出する中間変数算出ステップ；

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さｌｅｖの分割線において２つの部分グラフB_0，B_1に分割したときに根節点を含む側の部分グラフB_0に属するものについて、ルックアップ・テーブルを生成しそれをＬＵＴ記憶手段に格納するＬＵＴ生成ステップ；

及び、前記中間変数算出ステップにおいて算出された前記中間変数の個数ｕと等しい制御入力数を有する二分木を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフB_0に属する非終端節点の節点データを、前記二分木を表す節点データで置き換えることにより、特性関数二分決定グラフを再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新するＢＤＤ再構成ステップ。

【０１２３】

本発明に係る論理回路合成方法の第２の構成は、前記第１の構成において、前記節点テーブル記憶手段には、入力変数をＸ＝（x_1，…，x_n）（ｎ∈自然数）とし出力にドント・ケアを含む不完全定義関数である多出力論理関数Ｆ（Ｘ）＝（f_0（Ｘ），…，f_m

$_{-1}$（Ｘ））に対して式（１９）により定義される特性関数ｘ（Ｘ，Ｙ）（但し、Ｙ＝（y_0，…，y_{m-1}）（ｍ≧２，ｍ∈自然数）はＦ（Ｘ）の出力を表す変数。）を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点v_iについて、当該非終端節点に関わる変数z_i（z_i∈（Ｘ∪Ｙ））に付与される変数ラベル並びに当該変数z_i（z_i∈（Ｘ∪Ｙ））の値が０のとき及び１のときに遷移する先の子節点を特定する一対の枝e_0（v_i），e_1（v_i）からなる節点データのテーブルである節点テーブルとして記憶されていることを特徴とする。

【０１２４】

本発明に係る論理回路合成方法の第３の構成は、前記第２の構成において、前記節点テーブル記憶手段に記憶された前記節点テーブルにより表される特性関数二分決定グラフの幅を、前記第１乃至３の何れかの構成のグラフ幅削減方法によって削減し、前記節点テーブル記憶手段に記憶された前記節点テーブルを更新した後に、前記分割線設定ステップ乃至前記ＢＤＤ再構成ステップを実行することを特徴とする。

【０１２５】

本発明に係る論理回路合成方法の第４の構成は、前記第１乃至３の何れか一の構成において、前記多出力論理関数Ｆ（Ｘ）の要素をなす論理関数f_0（Ｘ），…，f_{m-1}（Ｘ）の順序をπ＝（π［０］，…，π［ｍ－１］）（但し、π［ｉ］＝ｊは、f_jがｉ番目であることを表す。）とし、論理関数f_j（∈Ｆ（Ｘ））が依存する入力変数の集合をｓｕｐｐ（f_j）としたとき、式（２３）で表されるＴの値が最小となるように前記多出力論理関数Ｆ（Ｘ）の要素の順序πを決定する出力変数順序決定ステップ；

前記各入力変数x_i（∈Ｘ）及び出力を表す変数y_j（∈Ｙ）の順序を、式（２４）を満たす順序Ｐに決定する全変数順序決定ステップ；

及び、前記順序Ｐに従って、特性関数二分決定グラフの節点データを生成し前記節点テーブル記憶手段に格納するＢＤＤ生成ステップ；

を実行した後に、前記分割線設定ステップ乃至前記ＢＤＤ再構成ステップを実行することを特徴とする。

【０１２６】

本発明に係るプログラムの第１の構成は、コンピュータに第１乃至３の何れかの構成のグラフ幅削減方法を実行させることを特徴とする。

【０１２７】

本発明に係るプログラムの第２の構成は、コンピュータに第１乃至４の何れかの構成の論理回路合成方法を実行させることを特徴とする。

【０１２８】

本発明に係るプログラマブル論理回路の構成は、第１乃至４の何れかの構成の論理回路合成方法により論理合成されたことを特徴とする。

【発明の効果】

【０１２９】

以上のように、本発明に係るグラフ幅削減装置及びグラフ幅削減方法によれば、特性関数二分決定グラフの幅を構成するすべての部分関数の両立性を調べてマッチングを行うことができるため、効果的に特性関数二分決定グラフの幅を削減することが可能となる。

また、本発明に係る論理回路合成装置及び論理回路合成方法によれば、特性関数二分決定グラフを分割する分割線の位置を、部分グラフB_0の大きさが過度に大きくならないように適度な高さに決めることで、部分グラフB_0に関する変数を束縛変数とする分解表の列複雑度が極度に増大するのを防止することが可能となる。その結果、比較的小容量のメモリを用いて論理回路合成を実現でき、また、論理関数分解処理の演算処理速度を高速化することができる。従って、現在まで有効な論理回路合成手段が知られていなかった、多出力論理関数の分解によるＬＵＴ論理回路の生成を、現実的なハードウェアを使用して短時間で実行することが可能となる。また、本発明は、任意の多出力論理関数のＬＵＴ論理回路生成に対して汎用的に適用可能である。

【図面の簡単な説明】

【０１３０】
【図１】特性関数二分決定グラフの一例を示す図である。
【図２】出力変数の短絡除去を説明するための図である。
【図３】多出力論理関数を２つの関数に関数分解した場合の論理回路の構成を表す図である。
【図４】表４の不完全定義関数の特性関数二分決定グラフに対してアルゴリズム１による幅の削減を行う例を表す図である。
【図５】短絡除去を行った後の特性関数二分決定グラフの構成を表す図である。
【図６】ＡＤＲ２の特性関数二分決定グラフを表す図である。
【図７】ＡＤＲ２の特性関数二分決定グラフを分割したときの根節点側の部分グラフをＬＵＴに変換する処理を説明するための図である。
【図８】ＡＤＲ２の特性関数二分決定グラフを短絡除去し、ＬＵＴに変換する処理を説明するための図である。
【図９】ＡＤＲ２の特性関数二分決定グラフを短絡除去し、ＬＵＴに変換する処理を説明するための図である。
【図１０】本発明の実施例１に係る論理回路合成装置及びその周辺装置の構成を表す図である。
【図１１】実施例１における論理回路合成方法の全体の流れを示すフローチャートである。
【図１２】変数順序の決定処理の流れを表すフローチャートである。
【図１３】節点テーブルのデータ構造を表す図である。
【図１４】式（３１）の既約な特性関数二分決定グラフ及びその節点テーブルを表す図である。
【図１５】式（３１）の特性関数二分決定グラフ及びその節点テーブルを表す図である。
【図１６】ＬＵＴカスケード論理回路の合成処理の流れを表すフローチャートである。
【図１７】本発明の論理回路合成方法によるＬＵＴカスケード論理回路の合成についての実験結果を示す図である。
【図１８】本発明の実施例２に係る論理回路合成装置及びその周辺装置の構成を表す図である。
【図１９】図１８のグラフ幅削減装置２０の構成を表す図である。
【図２０】グラフ幅削減装置２０のグラフ幅削減処理の全体の流れを表すフローチャートである。
【図２１】クリーク集合による両立グラフの節点被覆処理の流れを表すフローチャートである。
【図２２】例１０の不完全定義関数の特性関数二分決定グラフの幅削減処理を説明する図である。
【図２３】例１０の不完全定義関数の特性関数二分決定グラフの幅削減処理を説明する図である。
【図２４】例１０の不完全定義関数の特性関数二分決定グラフの幅削減処理を説明する図である。
【図２５】ベンチマーク関数５－７－１１－１３　ＲＮＳについてＬＵＴカスケード論理回路の合成を行った結果を比較する図である。
【符号の説明】
【０１３１】
１，１’　論理回路合成装置
２　論理仕様記憶手段
３　入力装置
４　出力装置
５　出力変数順序決定手段
６　全変数順序決定手段

７　ＢＤＤ生成手段
８　節点テーブル記憶手段
９　変数順序最適化手段
１０　分割線設定手段
１１　短絡除去手段
１２　ＢＤＤ幅計測手段
１３　中間変数算出手段
１４　ＬＵＴ生成手段
１５　ＢＤＤ再構成手段
１６　ＬＵＴ記憶手段
２０　グラフ幅削減装置
２１　ＢＤＤ高さ計測手段
２２　分割線設定手段
２３　列関数生成手段
２４　列関数記憶手段
２５　両立枝生成手段
２６　両立グラフ記憶手段
２７　クリーク生成手段
２８　クリーク・データ記憶手段
２９　割当ＢＤＤ再構成手段

【発明を実施するための最良の形態】

【０１３２】

以下、本発明を実施するための最良の形態について、図面を参照しながら説明する。

【実施例１】

【０１３３】

図１０は本発明の実施例１に係る論理回路合成装置及びその周辺装置の構成を表す図である。実施例１に係る論理回路合成装置１は、論理仕様記憶手段２に格納された論理回路の論理仕様から、ＬＵＴ論理回路を合成し、出力装置４に出力する。論理仕様記憶手段２には、入力装置３により、目的論理関数がネットリストなど論理仕様データとして格納される。

【０１３４】

尚、本実施例においては、論理回路設計用のＣＡＤ装置において論理回路合成装置１を使用する場合を想定している。論理回路合成装置１は、プログラム形態として提供され、そのプログラムをコンピュータに読み込んで実行することによって、図１０に示したような論理回路合成装置１の機能構成がコンピュータで実現される。尚、図１０において、入力装置３は、キーボード、マウス、ＣＤ－ＲＯＭドライブ等の各種データ入力装置を表している。出力装置４は、ディスプレイ、プリンタ、磁気記録装置等の各種データ出力装置を表している。また、出力装置４として、プログラマブル論理回路に生成されたＬＵＴ論理回路を直接書き込むためのプログラム書込装置を使用することもできる。

【０１３５】

論理回路合成装置１は、出力変数順序決定手段５、全変数順序決定手段６、ＢＤＤ生成手段７、節点テーブル記憶手段８、変数順序最適化手段９、分割線設定手段１０、短絡除去手段１１、ＢＤＤ幅計測手段１２、中間変数算出手段１３、ＬＵＴ生成手段１４、ＢＤＤ再構成手段１５、及びＬＵＴ記憶手段１６を備えた構成からなる。

【０１３６】

出力変数順序決定手段５は、多出力論理関数ｆ（Ｘ）の要素の順序π＝（π［０］，…，π［ｍ－１］）を決定する。全変数順序決定手段６は、各入力変数x_i（∈Ｘ）及び出力を表す変数y_j（∈Ｙ）の順序を、所定の順序Ｐに決定する。ＢＤＤ生成手段７は、全変数順序決定手段６で決定された順序Ｐに従って、特性関数二分決定グラフの節点テーブルを生成する。生成された節点テーブルは、節点テーブル記憶手段８に格納される。変数

順序最適化手段９は、節点テーブル記憶手段８に格納された節点テーブルの変数順序を更に最適化する。

【０１３７】

分割線設定手段１０は、節点テーブル記憶手段８に格納された特性関数二分決定グラフの節点データに対して、当該特性関数二分決定グラフを分割する分割線の高さｌｅｖの決定を行う。短絡除去手段１１は、節点テーブル記憶手段８に格納された特性関数二分決定グラフに対し、高さｌｅｖの分割線より上の部分グラフの短絡除去処理を行う。ＢＤＤ幅計測手段１２は、短絡除去処理がされた特性関数二分決定グラフの分割線における幅Ｗとして出力する。中間変数算出手段１３は、ＢＤＤ幅計測手段１２が出力する幅Ｗに基づき中間変数の個数ｕを算出する。ＬＵＴ生成手段１４は、分割線において２つの部分グラフB_0，B_1に分割したときの根節点を含む側の部分グラフB_0に属するものについて、ＬＵＴを生成しそれをＬＵＴ記憶手段１６に格納する。ＢＤＤ再構成手段１５は、部分グラフB_0を中間変数の個数ｕと等しい制御入力数を有する二分木に置き換えて、特性関数二分決定グラフを再構成し、節点テーブル記憶手段８を更新する。

【０１３８】

以上のように構成された本実施例に係る論理回路合成装置１について、以下その動作を説明する。

【０１３９】

図１１は実施例１における論理回路合成方法の全体の流れを示すフローチャートである。全体的な処理の流れでは、最初に、出力変数順序決定手段５と全変数順序決定手段６が、論理仕様記憶手段２に格納された論理仕様を読み出す。そして、読み出された論理仕様から、入力変数Ｘと出力変数Ｙとを抽出し、これら各変数の初期順序の決定を行う（Ｓ１）。次に、ＢＤＤ生成手段７は、論理仕様記憶手段２から論理仕様データを読み出し、上記決定された変数の初期順序に従って特性関数二分決定グラフの節点テーブルを生成する。生成された節点テーブルは、節点テーブル記憶手段８に格納される（Ｓ２）。次に、変数順序最適化手段９は、節点テーブル記憶手段８に格納された節点テーブルにおいて、特性関数二分決定グラフの幅の総和がより小さくなるように、更に変数順序の入れ替えを行い、変数順序を最適化する（Ｓ３）。このとき、変数順序の入れ替えは、出力変数の順序集合$Y=(y_0, y_1, \cdots, y_{m-1})$における出力変数$y_j$（$j=0, 1, \cdots, m-1$）に対して、その出力変数$y_j$が依存する入力変数$x_i$（$\in supp(f_j)$）は、出力変数$y_j$よりも常に高位であるという条件のもとで、変数の入れ替えが行われる。最後に、節点テーブル記憶手段８に格納された特性関数二分決定グラフの節点テーブルに基づいて、ＬＵＴ論理回路の合成が行われる（Ｓ４）。以下、上記各段階での処理について詳述する。

【０１４０】

（１）変数Ｘ，Ｙの初期順序の決定

図１２は変数順序の決定処理の流れを表すフローチャートである。まず、出力変数順序決定手段５は、内部変数ｉ，ｊ，ｒｅｎｅｗを０に初期化する（Ｓ１０）。ｉ，ｊは置換する変数のインデックスを表す内部変数であり、ｒｅｎｅｗは順序の置換が行われたことを検出するための更新フラグである。また、出力変数順序決定手段５は、出力変数Ｙの変数順序を表すｍ個の要素を持つ順序$\pi_0=(\pi_0[0], \pi_0[1], \cdots, \pi_0[m-1])$を（０，１，…，ｍ－１）に初期化する（Ｓ２）。順序π_0は、出力変数Ｙの変数順序を表す配列である。出力変数の順序は、順序π_0により、式（２５）のように順序づけされる。

【０１４１】

【数２５】

$$Y=(y_{\pi_0[0]}, y_{\pi_0[1]}, \cdots, y_{\pi_0[m-1]}) \qquad (25)$$

【０１４２】

次に、出力変数順序決定手段５は、式（２６）の演算を行うことにより、順序π_0に対する出力変数順序評価関数を算出し、これを変数T_{min}に格納する（Ｓ１２）。ここで、出力変数順序評価関数とは、式（２６）の右辺の関数である。この出力変数順序評価関数は、各出力変数Ｙの順序π_0が、依存変数の数の少ない順に出力変数Ｙを順序づけするものであるときに最小となる。従って、出力変数順序評価関数を最小化することにより、出力変数Ｙの順序は最適化されることとなる。また、ｓｕｐｐ（f_j）は、論理関数f_j（∈ｆ（Ｘ））が依存する入力変数の集合を表す。例えば、$f_j = f_j(x_1, x_2, x_5, x_{10})$であれば、｜ｓｕｐｐ（$f_j$）｜＝４である。また、｜ｓｕｐｐ（$f_j$）｜はｓｕｐｐ（$f_j$）の変数の個数である（〔定義４〕参照）。

【０１４３】

【数２６】

$$T_{min} = \sum_{k=0}^{m-1} \left| \bigcup_{l=0}^{k} \mathrm{supp}(f_{\pi_0[l]}) \right| \qquad (26)$$

【０１４４】

次に、出力変数順序決定手段５は、ｉ≠ｊであるかどうかを判定する（Ｓ１３）。ｉ＝ｊの場合には、ステップＳ１９に行く。これは、ｉ＝ｊの場合には置換が行われないので、ｉ＝ｊの場合を除外する必要があるからである。

【０１４５】

ステップＳ１３において、ｉ≠ｊであれば、まず、変数置換後の順序π_1を初期順序π_0に初期化した後（Ｓ１４）、順序π_1の要素π_1［ｉ］と要素π_1［ｊ］との置換を行う（Ｓ１５）。そして、出力変数順序決定手段５は、式（２７）の演算を行うことにより、順序π_1に対する出力変数順序評価関数Ｔを算出する（Ｓ１６）。

【０１４６】

【数２７】

$$T = \sum_{k=0}^{m-1} \left| \bigcup_{l=0}^{k} \mathrm{supp}(f_{\pi_1[l]}) \right| \qquad (27)$$

【０１４７】

ここで、置換後の出力変数順序評価関数Ｔが置換前の出力変数順序評価関数T_{min}よりも小さい（Ｔ＜T_{min}）場合には（Ｓ１７）、順序π_1のほうが出力変数の順序としてはより適当であると判断されるため、出力変数順序決定手段５は、順序π_0をπ_1に更新し、T_{min}をＴとする。そして、順序π_0の更新がされたことを表す更新フラグｒｅｎｅｗを１とする（Ｓ１８）。

【０１４８】

次に、ｊ＜ｍ－１であれば（Ｓ１９）、ｊを１だけインクリメントして（Ｓ２０）、ステップＳ１３に戻り、上記出力変数の順序の最適化処理を繰り返す。

【０１４９】

ステップＳ１８でｊ＝ｍ－１ならば、ｉ番目の出力変数の順序を固定したときの順序の入れ替えに対する評価がすべて終了したことになるので、ｊを０に初期化する（Ｓ２１）。

【０１５０】

次に、ｉ＜ｍ－１であれば（Ｓ２２）、次の出力変数の順序を固定したときの順序の入れ替えに対する評価を行うべく、ｉを１だけインクリメントして（Ｓ２３）、ステップＳ１３に戻る。

【０１５１】

ステップＳ２２において、ｉ＝ｍ－１の場合、ｉ番目の出力変数の順序を固定した場合についての評価は一通り終了したことになる。但し、順序入れ替えの総数は、ｍ！通りで

あるが、この段階では、順序入れ替えの評価はｍ×（ｍ－１）通りしか行われていない。そこで、残りの場合についても評価するかどうかを判断するため、上記ｍ×（ｍ－１）通りの評価において順序π_0の更新が行われたかどうかを判定する。すなわち、変数ｒｅｎｅｗが１であるかどうかを参照する（Ｓ２４）。ここで、ｒｅｎｅｗ＝１であれば、ｉを０、ｒｅｎｅｗを０に初期化して（Ｓ２５）、再びステップＳ１３の処理に戻る。尚、ステップＳ２４でｒｅｎｅｗ＝０であれば、終了する。

【０１５２】

以上のようにして、出力変数Ｙの順序π_0は、出力変数順序評価関数T_{min}の値が極小化する順序に最適化される。出力変数順序決定手段５は、出力変数Ｙの順序を順序π_0に並べ替えて出力する。

【０１５３】

尚、上記処理では、順序入れ替えの評価は必ずしもｍ！回行われないので、必ずしも出力変数順序評価関数T_{min}の値が最小となるように最適化されるとは限らないことに注意しておく。ｍ！回のすべてについての評価を行わないこととしたのは、処理時間を短縮するためである。しかしながら、上記変数並べ替えのアルゴリズムは一例であり、計算処理速度が十分速い場合や、出力変数及び入力変数の数が比較的少ない場合には、ｍ！回のすべてについて総当たりで評価を行うことにより、出力変数順序評価関数T_{min}の値が最小となるように最適化してもよい。

【０１５４】

出力変数の順序が決定された後、全変数順序決定手段６は、入力変数Ｘ及び出力変数Ｙの順序を式（２８）の順序Ｐに従って決定する。

【０１５５】

【数２８】

$$P = \left(\mathrm{supp}(f_{\pi[0]}), y_{\pi[0]}, \mathrm{supp}(f_{\pi[1]}) - \mathrm{supp}(f_{\pi[0]}), y_{\pi[1]}, \mathrm{supp}(f_{\pi[2]}) - \left(\sum_{k=0}^{1} \mathrm{supp}(f_{\pi[k]}) \right), y_{\pi[2]}, \cdots, \mathrm{supp}(f_{\pi[m-1]}) - \left(\sum_{k=0}^{m-2} \mathrm{supp}(f_{\pi[k]}) \right), y_{\pi[m-1]} \right) \tag{28}$$

ここで、式（２８）の条件を満たす限りにおいては、入力変数Ｘの順序の決め方は任意である。したがって、ここでは、出力変数

【０１５６】

【数２９】

$$y_{\pi[j]} \quad (j = 0, 1, \cdots, m-1) \tag{29}$$

の前にある入力変数の集合

【０１５７】

【数３０】

$$(x_{\theta[p]}, x_{\theta[p+1]}, \cdots, x_{\theta[p+q-1]}) = \mathrm{supp}(f_{\pi[j]}) - \left(\sum_{k=0}^{j-1} \mathrm{supp}(f_{\pi[k]}) \right) \tag{30}$$

の要素間の順序については、x_iのインデックスｉが小さい順に順序づけることとする。尚、式（３０）において（θ［１］，θ［２］，…，θ［ｐ］，θ［ｐ＋１］…，θ［ｐ＋ｑ－１］，…，θ［ｎ］）は、入力変数Ｘの順序を表し、θ［ｋ］＝ｉの場合、x_iは入力変数のｋ番目に位置することを表す。

【０１５８】

（２）特性関数二分決定グラフの節点テーブルの作成

変数Ｘ，Ｙの初期順序Ｐの決定がされた後、ＢＤＤ生成手段７は、決定された初期順序Ｐに従って、論理仕様記憶手段２に記憶された論理仕様に従って、特性関数二分決定グラフの節点テーブルを生成する。この節点テーブルは、節点テーブル記憶手段８に格納される。尚、論理仕様から特性関数二分決定グラフを生成するアルゴリズムに関しては、すでに種々のアルゴリズムが公知であるため、ここでは説明を省略する。

【０１５９】

図１３は節点テーブルのデータ構造を表す図である。１つの節点に対応する節点データは、（変数ラベル，０枝の子節点のアドレス，１枝の子節点のアドレス）の３つのデータ組からなる。変数ラベルには、各節点に対応する入力変数又は出力変数を特定する符号が格納される。通常は、計算処理の便宜のため、ｍ＋ｎ個の各変数x_1，…，x_n，y_0，…，y_{m-1}には２進数の符号が割り当てられるが、ここでは説明の都合上、変数x_1，…，x_n，y_0，…，y_{m-1}の記号をそのまま使用する。「０枝」（０－ｅｄｇｅ）とは、その節点に対応する変数の値（リテラル）が０であるときの節点間の遷移を表す枝（ｅｄｇｅ）をいい、「１枝」（１－ｅｄｇｅ）とは、その節点に対応する変数の値（リテラル）が１であるときの節点間の遷移を表す枝をいう。「０枝の子節点」（「１枝の子節点」）とは、その節点に対して０枝（１枝）に沿って遷移した先の節点をいう。「節点のアドレス」とは、節点テーブル記憶手段８において、その節点データが格納されている記憶装置の論理アドレスをいう。

【０１６０】

従って、ある節点に対応する変数の値が与えられた場合に、その変数値に対する遷移先の節点を参照する場合には、変数値０，１に応じて０枝の子節点のアドレス，１枝の子節点のアドレスを参照すれば、特性関数二分決定グラフの径路（ｐａｔｈ）を辿ることができる。

【０１６１】

〔例８〕

入力変数Ｘ＝（x_1，x_2，x_3）に対して式（３１）により表される多出力論理関数ｆ（Ｘ）について、変数順序Ｐは、上記アルゴリズムによってＰ＝（x_1，x_2，y_0，x_3，y_1）となる。

【０１６２】

【数３１】

$$\begin{aligned} F(X) &= (f_0(X),\ f_1(X)) \\ &= (x_1x_2,\ x_1 \vee x_3) \end{aligned} \qquad (31)$$

【０１６３】

ここで、式（３１）の多出力論理関数ｆ（Ｘ）の特性関数二分決定グラフは、図１４（ｂ）により表される。従って、図１４（ｂ）の特性関数二分決定グラフに対応する節点テーブルは、図１４（ａ）のようになる。ここで、終端節点のうち、特性関数値０に対応する終端節点にはアドレス０、特性関数値１に対応する終端節点にはアドレス１が割り当てられる。

〔例終わり〕

【０１６４】

（３）変数Ｘ，Ｙの順序の最適化

次に、変数Ｘ，Ｙが初期順序により順序づけられた特性関数二分決定グラフの節点テーブルが節点テーブル記憶手段８に格納されると、変数順序最適化手段９は、変数Ｘ，Ｙの間の順序の入れ替えを実行し、変数Ｘ，Ｙ間の順序の最適化を行う。このとき、変数Ｘの順序の入れ替えには式（３２）の条件（すなわち、「集合ｓｕｐｐ（f_j）に属するすべての入力変数x_iはy_jよりも高位である。」という条件）が課される。

【０１６５】

【数３２】

$$x_i \succ y_j \qquad (\forall x_i \in \mathrm{supp}(f_j)) \tag{32}$$

(但し、$z_p \succ z_q$は、変数z_pの高さが変数z_qの高さよりも高い（z_pはz_qより高位である）ことを表す。)

【０１６６】

〔例９〕

式（３１）の多出力論理関数Ｆ（Ｘ）で表現される論理仕様に対して、上記の変数順序Ｐ＝（x_1，x_2，y_0，x_3，y_1）による特性関数二分決定グラフの節点テーブルが、図１４のように生成され、節点テーブル記憶手段８に格納されているとする。変数順序最適化手段９は、この節点テーブルに対して、上記条件のもとでの変数順序Ｐの入れ替えを行う。そして、入れ替え後の変数順序Ｐ’により、特性関数二分決定グラフの再構築を行い、その結果、特性関数二分決定グラフの幅の総和が小さくなった場合にのみ、変数順序Ｐ’の節点テーブルで節点テーブル記憶手段８の内容を更新する。

【０１６７】

例えば、変数順序最適化手段９が上記の変数順序Ｐを順序Ｐ’＝（x_1，x_2，x_3，y_0，y_1）のように入れ替えを行った場合、特性関数二分決定グラフは図１５（ｂ）のようになる。また、この特性関数二分決定グラフに対応する節点テーブルは図１５（ｂ）のようになる。図１５（ａ）の節点テーブルの大きさと図１４（ａ）の節点テーブルの大きさは同じである。従って、この場合、節点テーブル記憶手段８の内容の更新は行われない。

【０１６８】

このような変数順序の入れ替えによる節点テーブルの再構築を総当たり（又は、適当な条件を課した試行）で試みることにより、変数順序の最適化が行われる。尚、式（３１）の多出力論理関数の場合、図１４の特性関数二分決定グラフが最小であるため、節点テーブル記憶手段８内の節点テーブルの更新は行われない。

〔例終わり〕

【０１６９】

（４）ＬＵＴカスケード論理回路の合成

以上のように、最適化された特性関数二分決定グラフの節点テーブルが節点テーブル記憶手段８に格納されると、続いてＬＵＴカスケード論理回路の合成処理が行われる。そこで、次に、ＬＵＴカスケード論理回路の合成処理について説明する。

【０１７０】

図１６はＬＵＴカスケード論理回路の合成処理の流れを表すフローチャートである。論理回路合成装置１は、分割線の高さを表す内部変数ｉ（ｉ－１が分割線の高さを表す。）をｎ＋ｍ＋１（ｎは入力変数Ｘの大きさ（ｎ＝｜Ｘ｜）、ｍは出力変数Ｙの大きさ（ｍ＝｜Ｙ｜）。）とする。また、中間変数の集合Ｈを空集合φとする。また、分割線より高位の変数の集合Z_aを空集合φとする。また、関数分解を行う特性関数二分決定グラフ$B_{CF}^{current}$を関数ｆの特性関数二分決定グラフ$B_{CF}(f)$とし、$B_{CF}^{current}$に含まれる変数の集合Z_tを全変数の集合Ｚ（＝Ｘ∪Ｙ）に初期化する（Ｓ３０）。

【０１７１】

次に、分割線設定手段１０は、変数ｉを１だけ減少する（Ｓ３１）。すなわち、分割線の高さを１だけ下げる。そして、集合Z_{temp}を$Z_a \cup \{z_i\}$とする（Ｓ３２）。ここで、$\{z_i\}$は高さｉの変数z_iの集合を表す。すなわち、分割線を下げたので、集合$\{z_i\}$を分割線よりも高位の変数の集合Z_{temp}に追加する。

【０１７２】

次に、分割線設定手段１０は、関数分解可能性を判定する（Ｓ３３）。関数分解の可能性の判定は、分割線で分割した場合の列複雑度μ及び集合Z_{temp}の要素の数とＬＵＴ

の入力数の最大値ｋを比較することにより行われる。すなわち、関数分解可能な条件は、式（３３）で表される。従って、式（３３）の条件を満たすか否かで関数分解の可能性が判定される。

【０１７３】

【数３３】

$$\lceil \log_2 \mu \rceil < k \ \wedge \ |Z_{temp}| \leqq k \qquad (33)$$

【０１７４】

上記ステップＳ３４の判定の結果、関数分解が可能であれば、次に、変数ｉが１であるか否かを判定する（Ｓ３５）。ｉ＝１ならば、これ以上関数分解はできないので、ＬＵＴ生成手段１４は関数ｆ（Z_{temp}）のＬＵＴを生成し、ＬＵＴ記憶手段１６に格納し、終了する（Ｓ３６）。また、ステップＳ３５において、分割線の高さｉが１よりも大きい場合には、分割線設定手段１０は、集合Z_aを集合Z_{temp}に更新して（Ｓ３７）、ステップＳ３１に戻る。

【０１７５】

一方、ステップＳ３４において、関数分解が不可能であると判定される場合、分割線設定手段１０は、Z_aの大きさが｜Z_a｜＝｜Ｈ｜かどうかを判定する（Ｓ３８）。もし、｜Z_a｜＝｜Ｈ｜ならば、これ以上分割線を下げても関数分解が可能となる可能性はない。従って、この場合には、分割線設定手段１０は、ＬＵＴ論理回路が実現不可能である旨のメッセージを出力装置４に出力し（Ｓ３９）、処理を終了する。

【０１７６】

ステップＳ３８において、｜Z_a｜＞｜Ｈ｜であれば、短絡除去手段１１は、分割線の高さ以下の高さにある変数の集合Z_bを$Z_t - Z_a$とする（Ｓ４０）。次に、短絡除去手段１１は、分割線よりも高位の部分グラフに対して短絡除去を行う。次いで、ＢＤＤ幅計測手段１２は、短絡除去がされた特性関数二分決定グラフについて、分割線における幅μ_aを計測する。次に、中間変数算出手段１３は、レイル数u_aを式（３４）により計算する（Ｓ４１）。

【０１７７】

【数３４】

$$u_a = \lceil \log_2 \mu_a \rceil \qquad (34)$$

【０１７８】

次に、ＬＵＴ生成手段１４は、（Z_a，Z_b）を変数の分割として、ｆ（Ｚ）＝ｇ（ｈ（Z_a），Z_b）の形で関数分解したときの関数ｈ（Z_a）のＬＵＴを生成し、ＬＵＴ記憶手段１６に格納する（Ｓ４２）。すなわち、上記Ｓ４１の短絡処理が行われていない特性関数二分決定グラフについて、変数Z_aに関する部分グラフをB_a、Z_bに関わる部分グラフをB_bとする。また、ステップＳ４１において短絡除去がされた特性関数二分決定グラフについて、X_a（$\subseteq Z_a$）に関わる部分グラフをB_a’、Z_bに関わる部分グラフをB_bとする。ＬＵＴ生成手段１４は、部分グラフB_aから、集合Z_aに属するすべての入力変数x_i（$\in Z_a$）を制御入力とし、集合Z_aに属するすべての出力変数y_j（$\in Z_a$）を出力するＬＵＴを生成する。次いで、ＬＵＴ生成手段１４は、ステップＳ４１において短絡除去がされた特性関数二分決定グラフについて、部分グラフB_a’から直接接続される部分グラフB_b内の節点の各々にu_aビットの符号（h_1，…，h_{u_a}）を割り当てる。この符号（h_1，…，h_{u_a}）を中間変数とする。そして、部分グラフB_a’から、集合Z_aに属するすべての入力変数X_a（$\in Z_a$）を制御入力とし、中間変数（h_1，…，h_{u_a}）を出力するＬＵＴを生成する。そして、これらのＬＵＴをＬＵＴ記憶手段１６に格納する。

【０１７９】

次に、ＢＤＤ再構成手段１５は、（Z_a，Z_b）を変数の分割として、ｆ（Ｚ）＝ｇ（ｈ（Z_a），Z_b）の形で関数分解したときの関数ｇ（ｈ，Z_b）についての特性関数二分決定グラフを生成する（Ｓ４４）。すなわち、ＢＤＤ再構成手段１５は、ステップＳ４１において短絡除去がされた特性関数二分決定グラフについて、X_a（$\subseteq Z_a$）に関わる部分グラフを、中間変数（h_1，…，h_{u_a}）を制御入力とする二分木に置き換える。

【０１８０】

次に、中間変数の集合Ｈを、u_a個の中間変数（h_1，…，h_{u_a}）に更新する。そして、集合Z_tを集合Z_bと集合Ｈとの和集合$Z_b \cup H$とする（Ｓ４４）。

【０１８１】

次に、$|Z_t|$がｋ以下かどうかを判定する（Ｓ４５）。$|Z_t| \leqq k$ならば、関数ｇのＬＵＴを生成し、ＬＵＴ記憶手段１６に格納し（Ｓ４６）、処理を終了する。

【０１８２】

ステップＳ４４において、$|Z_t| > k$ならば、分割線の高さｉを$|Z_t|+1$，$B_{CF}{}^{current}$を$B_{CF}(g)$，集合Z_tを集合Z_bと集合Ｈとの和集合$Z_b \cup H$とした後（Ｓ４７）、更に、集合Z_aを集合Ｈ，関数ｆを関数ｇとして（Ｓ４８）、ステップＳ３１に戻る。

【０１８３】

以上の処理により、論理関数の分解が行われ、ＬＵＴカスケード論理回路が構成される。

【０１８４】

以上のように、本実施例の論理回路合成装置１によれば、比較的少容量のメモリを用いてＬＵＴ論理回路の合成を実現できる。また、論理関数分解処理の演算処理速度を高速化することができる。

【０１８５】

また、現在まで有効な論理回路合成手段が知られていなかった、多出力論理関数の分解による途中出力を有するＬＵＴ論理回路の生成を、現実的なハードウェアを使用して短時間で実行することが可能となる。

【０１８６】

（５）実験結果

最後に、本発明の効果を示すため、実際に幾つかのベンチマーク関数を用いてＬＵＴカスケード論理回路の合成を行った結果を図１７に示す。実験は上記実施例１のアルゴリズムをＣ言語で実装し、ＭＣＮＣ８９ベンチマーク関数に適用した。図１７は、ＬＵＴの入力数ｋを１０に設定した場合の結果である。

【０１８７】

図１７において、「Ｎａｍｅ」は関数名、「Ｉｎ」は入力数、「Ｏｕｔ」は出力数、「ＬＵＴ」は生成されるＬＵＴの総数、「Ｃａｓ」はカスケードの個数を表し、「段数」はカスケードの段数を表す。実行環境は、ＩＢＭ　ＰＣ／ＡＴ互換機，Ｐｅｎｔｉｕｍ（登録商標）４２．０ＧＨｚ，メモリ５１２ＭＢｙｔｅ，ＯＳはＷｉｎｄｏｗｓ（登録商標）２０００，ｃｙｇｗｉｎ上でｇｃｃを用いてコンパイルした。本アルゴリズムでは、出力のグループ分けを行うため、一つずつの出力をグループに加えて特性関数二分決定グラフを構成し、変数順序最適化を行う。ｋが大きくなると、ＬＵＴカスケードに対応する特性関数二分決定グラフが大きくなる。ＬＵＴカスケードで構成可能な限り、グループの出力を一つずつ増やしながら変数順序最適化を行うため、大きな特性関数二分決定グラフの変数順序最適化を行う回数が増える。このため、ｋが大きくなると多くの実行時間が必要となる。実行時間の殆どは、特性関数二分決定グラフの変数順序最適化のための時間である。

【０１８８】

本発明の効果を確認するため、非特許文献１５に記載の方法との比較を行った。非特許文献１５の方法は、ＭＴＢＤＤに基づいており、多出力論理関数の出力を幾つかのグループに分割してＭＴＢＤＤで表現し、ＬＵＴカスケードで実現する。ＭＴＢＤＤの幅が大き

すぎるため分解不可能な場合は、ＯＲ分割を用いてカスケードを分割する。図１７では、非特許文献１５の方法については、一つのグループの出力数は８として、出力を分割している。

【０１８９】

非特許文献１５の方法では、実現したＬＵＴカスケードのＬＵＴの個数や段数にかかわらず、分割するグループの出力数を固定しているため、カスケードの個数は多くなり、段数は小さくなる傾向にある。一方、本発明に係る方法では、ＬＵＴカスケードは可能な限り、一つのグループの出力の個数を増やすので、段数は多くなり、カスケードの個数は少なくなる。

【実施例２】

【０１９０】

図１８は、本発明の実施例２に係る論理回路合成装置及びその周辺装置の構成を表す図である。図１８において、論理仕様記憶手段２、入力装置３、出力装置４、出力変数順序決定手段５、全変数順序決定手段６、ＢＤＤ生成手段７、節点テーブル記憶手段８、変数順序最適化手段９、分割線設定手段１０、短絡除去手段１１、ＢＤＤ幅計測手段１２、中間変数算出手段１３、ＬＵＴ生成手段１４、ＢＤＤ再構成手段１５、及びＬＵＴ記憶手段１６は、実施例１と同様のものである。本実施例の論理回路合成装置１’は、グラフ幅削減装置２０を備えている点を特徴としている。

【０１９１】

尚、本実施例においても、実施例１と同様、論理回路設計用のＣＡＤ装置において論理回路合成装置１’を使用する場合を想定している。論理回路合成装置１’は、プログラム形態として提供され、そのプログラムをコンピュータに読み込んで実行することによって、図１８に示したような論理回路合成装置１’の機能構成がコンピュータで実現される。

【０１９２】

グラフ幅削減装置２０は、節点テーブル記憶手段８に記憶された節点テーブルにより表される特性関数が不完全定義関数である場合、その特性関数二分決定グラフの幅の削減を行う。

【０１９３】

図１９は、図１８のグラフ幅削減装置２０の構成を表す図である。グラフ幅削減装置２０は、ＢＤＤ高さ計測手段２１、分割線設定手段２２、列関数生成手段２３、列関数記憶手段２４、両立枝生成手段２５、両立グラフ記憶手段２６、クリーク生成手段２７、クリーク・データ記憶手段２８、及び割当ＢＤＤ再構成手段２９を備えている。

【０１９４】

ＢＤＤ高さ計測手段２１は、節点テーブル記憶手段８に記憶されている節点テーブルによって表現される特性関数二分決定グラフの根節点の高さｔを計測する。分割線設定手段２２は、高さｔ－１から１まで、分割の高さｌｅｖを順次設定する。

【０１９５】

列関数生成手段２３は、節点データ記憶手段８に記憶された節点テーブルから、分割の高さｌｅｖの節点の各枝に対応する列関数を生成し、それら各列関数に対応した列関数ラベルを有する関数節点データを生成して両立グラフ記憶手段２６に保存する。列関数記憶手段２４は、列関数生成手段２３が生成する列関数を一時的に記憶する。

【０１９６】

両立枝生成手段２５は、両立グラフ記憶手段２６に記憶された各関数節点データに対する列関数から、両立する列関数の組を選出し、それら両立する列関数に対する関数節点データにそれらの関数節点同士を連結する両立枝を追加して、両立グラフ記憶手段２６に記憶された関数データの更新を行う。

【０１９７】

クリーク生成手段２７は、両立グラフ記憶手段２６に記憶された両立グラフの全節点について、完全部分グラフ（クリーク）による節点被覆を行う。そして、クリークに含まれる関数節点集合であるクリーク・データを生成し、クリーク・データ記憶手段２８に記憶

する。

【０１９８】

割当ＢＤＤ再構成手段２９は、クリーク・データ記憶手段２８に記憶された各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数を生成する。そして、当該クリーク・データに含まれる各関数節点の列関数を、割当列関数に置換することによって、特性関数二分決定グラフの再構成を行い、節点テーブル記憶手段８の節点テーブルの更新を行う。尚、割当ＢＤＤ再構成手段２９は、各関数節点に対応する列関数を参照する際、節点テーブル記憶手段８の節点テーブルから必要に応じて列関数を合成するようにしてもよいが、列関数記憶手段２４に一時保存されている列関数を参照するようにしてもよい。

【０１９９】

以上のように構成された本実施例に係る論理回路合成装置１’について、以下その演算処理動作を説明する。

【０２００】

全体の流れとしては、最初に、図１１のステップＳ１～Ｓ３の処理により、節点データ記憶手段８に、不完全定義関数の特性関数二分決定グラフの節点テーブルが保存される。次に、グラフ幅削減装置２０による特性関数二分決定グラフの幅削減処理が実行される。そして、最後に、図１１のステップＳ４のＬＵＴ論理回路の合成が行われる。ここで、本実施例の論理回路合成装置１’は、グラフ幅削減装置２０による特性関数二分決定グラフの幅削減処理以外の演算処理は、実施例１と同様である。そこで、ここではグラフ幅削減装置２０の演算処理についてのみ説明する。

【０２０１】

図２０は、グラフ幅削減装置２０のグラフ幅削減処理の全体の流れを表すフローチャートである。最初に、ＢＤＤ高さ計測手段２１が、節点テーブル記憶手段８に記憶された節点テーブルを探索して、特性関数二分決定グラフの根節点の高さｔを決定する。根節点の高さｔが決定されると、分割線設定手段２２は、分割の高さｌｅｖをｔ－１に設定する（Ｓ５０）。

【０２０２】

次に、列関数生成手段２３は、分割の高さｌｅｖにおけるすべての列関数の集合を作成する（Ｓ５１）。生成された列関数に対しては、関数節点の節点ラベルが付与され、両立グラフ記憶手段２６に関数節点データとして保存される。また、特性関数二分決定グラフの大きさが小さい場合には、計算速度を向上させるために、列関数生成手段２３が生成した各列関数を、列関数記憶手段２４に保存するようにしてもよい。

【０２０３】

次に、両立枝生成手段２５は、列関数生成手段２３が生成する各列関数から、両立する列関数の組を選出する。そして、両立グラフ記憶手段２６に記憶された両立する列関数に対する関数節点データに、それらの関数節点同士を連結する両立枝を追加する。両立枝が追加された関数節点データは、両立グラフ記憶手段２６に保存され、関数節点データの更新が行われる（Ｓ５２）。この処理は、列関数生成手段２３が生成するすべての列関数に対して、すべての２つの列関数組の組み合わせについて実行される。この処理によって、両立グラフ記憶手段２６に、両立グラフを表現する関数節点データのテーブルが生成される。

【０２０４】

次に、クリーク生成手段２７は、両立グラフ記憶手段２６に記憶された関数節点データのテーブルが表す両立グラフに対して、クリーク集合による節点被覆処理を実行する（Ｓ５３）。この処理の詳細については後述するが、これによって、両立グラフのすべての節点について、完全部分グラフ（クリーク）による節点被覆が行われ、クリーク・データが生成される。生成されたクリーク・データは、クリーク・データ記憶手段２８に保存される。

【０２０５】

尚、「完全部分グラフ（クリーク）」とは、両立グラフの部分グラフであって、当該部分グラフの任意の一つの節点が他のすべての節点と枝により連結されているものをいう。また、クリーク・データとは、クリークに含まれる関数節点の集合のデータをいう。

【０２０６】

次に、割当ＢＤＤ再構成手段２９は、節点テーブル記憶手段８に記憶された節点テーブルに基づき、クリーク・データ記憶手段２８に記憶された一つのクリーク・データについて、当該クリーク・データに含まれる関数節点に対応する列関数を生成し、列関数記憶手段２４に一時的に保存する。そして、当該クリーク・データに属するすべての関数節点に対応する列関数について、ドント・ケアへの定数の割り当てを行い、同一の割当列関数を生成する。そして、当該クリーク・データに属するすべての関数節点に対応する列関数を、生成した割当列関数により置換した特性関数二分決定グラフとなるように、節点テーブル記憶手段８内の節点テーブルの書き換えを行う。この割当列関数を用いた節点テーブルの書き換え処理を、すべてのクリーク・データに対して実行することにより、特性関数二分決定グラフの再構成を行う（Ｓ５４）。

【０２０７】

次に、分割線設定手段２２は、分割の高さｌｅｖを１だけ減少させる（Ｓ５５）。そして、分割の高さｌｅｖが１以上であれば（Ｓ５６）、ステップＳ５１に戻る。分割の高さｌｅｖが０となった場合には（Ｓ５６）、グラフ幅削減処理を終了する。

【０２０８】

以上のようなグラフ幅削減処理によって、上述したアルゴリズム３が実行され、効果的な特性関数二分決定グラフの幅の削減が実行される。

【０２０９】

次に、上記ステップＳ５３におけるクリーク集合による両立グラフの節点被覆処理について説明する。図２１は、クリーク集合による両立グラフの節点被覆処理の流れを表すフローチャートである。

【０２１０】

クリーク生成手段２７は、内部変数として、クリーク・データのリストを表す変数Ｃ（以下「集合Ｃ」という。）、クリーク被覆が行われていない関数節点のリストを表す変数S_a（以下「節点集合S_a」という。）、一つのクリークに含まれる関数節点のリストを表す変数S_i（以下「節点集合S_i」という。）、及びクリーク変数に追加する関数節点の候補のリストを表す変数S_b（以下「節点集合S_b」という。）を有する。

【０２１１】

まず、クリーク生成手段２７は、クリーク・データの集合Ｃを空集合に初期化する（Ｓ６０）。そして、両立グラフのすべての関数節点を節点集合S_aに追加して初期化を行った後、節点集合S_aから枝を持たない関数節点を除去するとともに、これらの関数節点を、要素１のクリーク集合として、集合Ｃの要素に追加する（Ｓ６１）。

【０２１２】

次に、クリーク生成手段２７は、節点集合S_aが空集合でないか否かを判定する（Ｓ６２）。

【０２１３】

節点集合S_aが空集合でない場合、クリーク生成手段２７は、節点集合S_aの中で枝数が最小である関数節点を探索し、これを関数節点v_iとする（Ｓ６３）。次に、クリーク生成手段２７は、節点集合S_iを$\{v_i\}$に初期化する（Ｓ６４）。そして、節点集合S_aの中で関数節点v_iに接続している関数節点を探索し、これを節点集合S_bとする（Ｓ６５）。

【０２１４】

次に、クリーク生成手段２７は、節点集合S_bが空集合か否かを判定する（Ｓ６６）。節点集合S_bが空集合でなければ、節点集合S_bに属する関数節点の中で、枝数が最小のものを探索し、これを関数節点v_jとする。そして、節点集合S_iに$\{v_j\}$を追加するとともに、節点集合S_bから関数節点v_jを除去する（Ｓ６７）。そして、節点集合S_b

に属する関数節点のうち、関数節点ｖ$_{j}$に接続していない関数節点を節点集合Ｓ$_{b}$から除去する（Ｓ６８）。そして、ステップＳ６６に戻る。

【０２１５】

ステップＳ６６において、節点集合Ｓ$_{b}$が空集合の場合、集合Ｃの要素に節点集合Ｓ$_{i}$をクリークとして追加するとともに、未被覆の節点集合Ｓ$_{a}$から節点集合Ｓ$_{i}$を除去する（Ｓ６９）。そして、ステップＳ６２に戻る。

【０２１６】

ステップＳ６２において、未被覆の節点集合Ｓ$_{a}$が空集合となった時点で、集合Ｃをクリーク・データとしてクリーク・データ記憶手段２８に保存し、クリーク集合による両立グラフの節点被覆処理を終了する。

【０２１７】

以上のような処理によって、両立グラフのすべての関数節点が、クリークにより被覆され、クリーク・データが生成される。

【０２１８】

〔例１０〕

真理値が例６の（表４）により表される４入力２出力不完全定義関数について、特性関数二分決定グラフの幅を削減する。（表４）により表される４入力２出力不完全定義関数の特性関数二分決定グラフは、図４（ａ）で表される。特性関数二分決定グラフの高さｔは６である。これを、本実施例のグラフ幅削減装置２０によって特性関数二分決定グラフの幅の削減処理を行う。

【０２１９】

まず、最初に分割の高さｌｅｖはｔ－１＝５、すなわち、変数ｘ$_{2}$の高さに設定される。ｌｅｖ＝５において、分解表は（表５）に示したようになる。（表５）の列関数は両立しないため、ドント・ケアへの定数の割り当てはできない。

【０２２０】

【表５】

$X_1 = \{x_1\}$

$X_2 = \{x_2, x_3, x_4\}$	0	1
000	$d1$	01
001	$d1$	01
010	00	10
011	00	10
100	dd	$1d$
101	dd	$1d$
110	10	$d0$
111	11	$d1$
	Φ_1	Φ_2

【０２２１】

次に、分割の高さをｌｅｖ＝４とする（図２２（ａ）参照）。ｌｅｖ＝４の分解表は、（表６）のようになる。

【０２２２】

【表６】

$X_1 = \{x_1\}$

$X_2 = \{x_2, x_3, x_4\}$	00	01	10	11
00	$d1$	dd	01	$1d$
01	$d1$	dd	01	$1d$
10	00	10	10	$d0$
11	00	11	10	$d1$
	① Φ_1	② Φ_2	③ Φ_3	④ Φ_4

【０２２３】

（表６）より、４つの列関数Φ$_1$～Φ$_4$が得られる。列関数Φ$_2$と列関数Φ$_4$は両立する。従って、両立グラフは、図２２（ｂ）のように表される。図２２（ｂ）の両立グラフについて、クリークによる節点被覆処理を行うと、図２２（ｂ）の点線で示した３つのクリークＳ$_1$，Ｓ$_2$，Ｓ$_3$が生成される。そこで、クリークＳ$_3$に含まれる２つの列関数Φ$_2$，Φ$_4$に対して定数の割り当てを行うと、（表７）のようになる。これらの列関数から特性関数二分決定グラフを生成すると、図２３（ａ）のようになる。図２２（ａ）のグラフの節点２，４は、図２３（ａ）では、節点１１に置き換えられている。

【０２２４】

【表７】

$X_1 = \{x_1\}$

$X_2 = \{x_2, x_3, x_4\}$	00	01	10	11
00	$d1$	$1d$	01	$1d$
01	$d1$	$1d$	01	$1d$
10	00	10	10	10
11	00	11	10	11
	① Φ_1	② Φ'_2	③ Φ_3	④ Φ'_4

【０２２５】

次に、分割の高さをｌｅｖ＝３（変数ｙ$_1$の高さ）とする（図２３（ａ）参照）。図２３（ａ）より、分割の高さをｌｅｖ＝３を横切る各枝に対応して、６つの列関数Φ$_1$～Φ$_6$が得られる。図２３（ａ）では、各列関数Φ$_1$～Φ$_6$の添字に対応して、それぞれの枝に番号が付されている。列関数Φ$_1$，Φ$_5$、列関数Φ$_3$，Φ$_4$，Φ$_6$は両立する。従って、両立グラフは、図２３（ｂ）のように表される。図２３（ｂ）の両立グラフについて、クリークによる節点被覆処理を行うと、例えば、図２３（ｂ）の点線で示した４つのクリークＳ$_1$，Ｓ$_2$，Ｓ$_3$，Ｓ$_4$が生成される。そこで、クリークＳ$_1$に含まれる２つの列関数Φ$_1$，Φ$_5$に対して定数の割り当てを行い、クリークＳ$_2$に含まれる２つの列関数Φ$_3$，Φ$_4$に対して定数の割り当てを行う。そして、生成された割当列関数を用いて、特性関数二分決定グラフの再構成を行うと、グラフ構造は図２４のようになる。図２３（ａ）の節点６，８と節点７，１０を、図２４では、それぞれ節点１２，１３で置き換えている。

【０２２６】

分割の高さをｌｅｖ＝２，１では、各列関数は両立しないので、これ以上特性関数二分決定グラフの再構成は行われない。

【０２２７】

図２２（ａ）と図２４とを比較すると、グラフ幅削減処理の結果、最大幅は８から４に削減され、非終端節点数は１５から１２に削減されたことが分かる。これを、上述したア

ルゴリズム１の結果と比較すると、アルゴリズム１を適用した結果（図４（ｂ）参照）に比べ、最大幅がより削減されていることが分かる。
〔例終わり〕
【０２２８】
最後に、幾つかのベンチマーク関数を用いて、本実施例に係るグラフ幅削減装置１’の性能を試験した結果について説明する。（表８）は、各ベンチマーク関数について、本実施例の論理回路合成装置１’によりＬＵＴカスケード論理回路の合成を行った結果を表す。
【０２２９】
【表８】

Name	$DC=0$			Alg.3		
	# Cells	# LUTs	# Cas	# Cells	# LUTs	# Cas
5-7-11-13 RNS	6	35	3	4	29	2
7-11-13-17 RNS	9	53	3	8	52	3
11-13-15-17 RNS	19	118	5	18	108	5
4-digit 11-nary to binary	4	29	2	4	28	2
4-digit 13-nary to binary	4	31	2	4	30	2
5-digit decimal to binary	8	50	3	7	48	2
6-digit 5-nary to binary	6	44	2	5	36	2
6-digit 6-nary to binary	5	38	2	5	29	2
6-digit 7-nary to binary	10	68	3	9	58	3
10-digit ternary to binary	9	59	3	6	48	2
3-digit decimal adder	7	37	2	3	17	1
4-digit decimal adder	10	64	2	4	24	1
2-digit decimal multiplier	8	51	3	7	48	3
Total	105	677	35	84	555	30
Ratio	1.00	1.00	1.00	0.80	0.82	0.86

【０２３０】
それぞれのベンチマーク関数は以下の通りである。
【０２３１】
（１）剰余数表現（Ｒｅｓｉｄｕｅ Ｎｕｍｂｅｒ Ｓｙｓｔｅｍ，ＲＮＳ）を２進表現に変換する回路（Ｒｅｓｉｄｕｅ Ｎｕｍｂｅｒ ｔｏ Ｂｉｎａｒｙ Ｎｕｍｂｅｒ Ｃｏｎｖｅｒｔｅｒｓ）
・５－７－１１－１３ＲＮＳ（１４－ｉｎｐｕｔ １３－ｏｕｔｐｕｔ，ＤＣ ６９．５％：４－ｄｉｇｉｔ ＲＮＳ ｎｕｍｂｅｒ，ｗｈｅｒｅ ｔｈｅ ｍｏｄｕｌｉ ａｒｅ ５，７，１１，ａｎｄ １３）
・７－１１－１３－１７ ＲＮＳ（１６－ｉｎｐｕｔ １５－ｏｕｔｐｕｔ，ＤＣ ７４．０％）
・１１－１３－１５－１７ ＲＮＳ（１７－ｉｎｐｕｔ １６－ｏｕｔｐｕｔ，ＤＣ ７２．２％）
【０２３２】
（２）ｉ桁のｋ進数を２進数に変換する回路（ｋ－ｎａｒｙ ｔｏ Ｂｉｎａｒｙ Ｃｏｎｖｅｒｔｅｒｓ）
・４－ｄｉｇｉｔ １１－ｎａｒｙ ｔｏ ｂｉｎａｒｙ（１６－ｉｎｐｕｔ １４－ｏｕｔｐｕｔ，ＤＣ ７７．７％）
・４－ｄｉｇｉｔ １３－ｎａｒｙ ｔｏ ｂｉｎａｒｙ（１６－ｉｎｐｕｔ １５－ｏｕｔｐｕｔ，ＤＣ ５６．４％）
・５－ｄｉｇｉｔ ｄｅｃｉｍａｌ ｔｏ ｂｉｎａｒｙ（２０－ｉｎｐｕｔ １７－ｏ

ｕｔｐｕｔ，ＤＣ　９０．５％）
・６－ｄｉｇｉｔ　５－ｎａｒｙ　ｔｏ　ｂｉｎａｒｙ（１８－ｉｎｐｕｔ　１４－ｏｕｔｐｕｔ，ＤＣ　９４．０％）
・６－ｄｉｇｉｔ　６－ｎａｒｙ　ｔｏ　ｂｉｎａｒｙ（１８－ｉｎｐｕｔ　１６－ｏｕｔｐｕｔ，ＤＣ　８２．２％）
・６－ｄｉｇｉｔ　７－ｎａｒｙ　ｔｏ　ｂｉｎａｒｙ（１８－ｉｎｐｕｔ　１７－ｏｕｔｐｕｔ，ＤＣ　５５．１％）
・１０－ｄｉｇｉｔ　ｔｅｒｎａｒｙ　ｔｏ　ｂｉｎａｒｙ（２０－ｉｎｐｕｔ　１６－ｏｕｔｐｕｔ，ＤＣ　９４．４％）

【０２３３】

（３）ｉ桁の１０進数の加算回路、乗算回路（Ｄｅｃｉｍａｌ　Ａｄｄｅｒｓ　ａｎｄ　ａ　Ｍｕｌｔｉｐｌｉｅｒ）
・３－ｄｉｇｉｔ　ｄｅｃｉｍａｌ　ａｄｄｅｒ（２４－ｉｎｐｕｔ　１６－ｏｕｔｐｕｔ，ＤＣ　９４．０％）
・４－ｄｉｇｉｔ　ｄｅｃｉｍａｌ　ａｄｄｅｒ（３２－ｉｎｐｕｔ　２０－ｏｕｔｐｕｔ，ＤＣ　９７．７％）
・２－ｄｉｇｉｔ　ｄｅｃｉｍａｌ　ｍｕｌｔｉｐｌｉｅｒ（１６－ｉｎｐｕｔ　１６－ｏｕｔｐｕｔ，ＤＣ　８４．７％）

【０２３４】

（表８）において、「ＤＣ＝０」は、すべてのドント・ケアに０を割り当てた場合を示し、「Ａｌｇ．３」は、本実施例に係る論理回路合成装置１’によりアルゴリズム３を適用した場合を示している。「＃Ｃｅｌｌｓ」の欄は、カスケード内のセル数を表す。「＃ＬＵＴｓ」の欄は、ＬＵＴの出力数の総和を表す。「＃Ｃａｓ」の欄は、カスケードの数を表す。（表８）より、本実施例に係る論理回路合成装置１’により、カスケード内のセル数とＬＵＴの出力数の総和を効果的に削減することが可能であることが分かる。

【０２３５】

例えば、５－７－１１－１３　ＲＮＳを考える。この関数は、入力の組み合わせに対し６９．５％のドント・ケアを含む。この関数についてＬＵＴカスケード論理回路を合成する。条件として、各セルの入力の本数は制限されており、入力数が最大１２まで許されるものとする。

【０２３６】

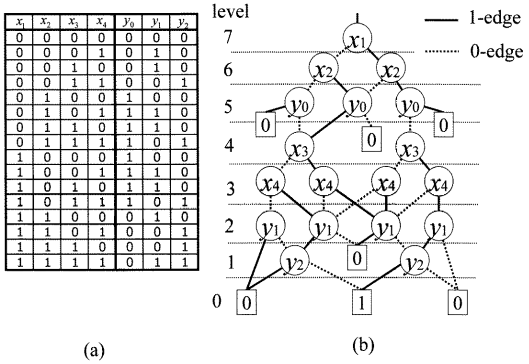
ドント・ケアに０を割り当ててＬＵＴカスケード論理回路を合成すると、図２５（ａ）のようになる。これに対し、本実施例に係る論理回路合成装置１’によりＬＵＴカスケード論理回路を合成すると、図２５（ｂ）のようになる。明らかに、本実施例に係る論理回路合成装置１’によりＬＵＴカスケード論理回路の回路規模を減少させることが可能であることが分かる。

【産業上の利用可能性】

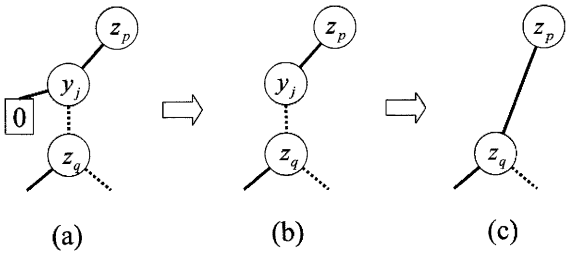
【０２３７】

本発明は、ＬＵＴ型の論理回路を自動設計する論理回路設計用のＣＡＤ装置、ＬＵＴ型ＦＰＧＡの論理合成ツール等に適用することができる。

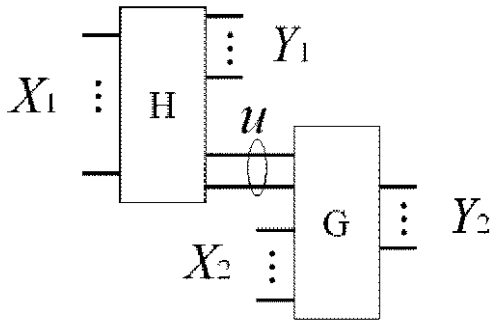
【図１】



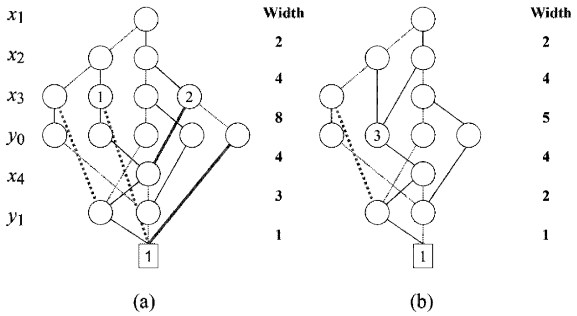
【図２】



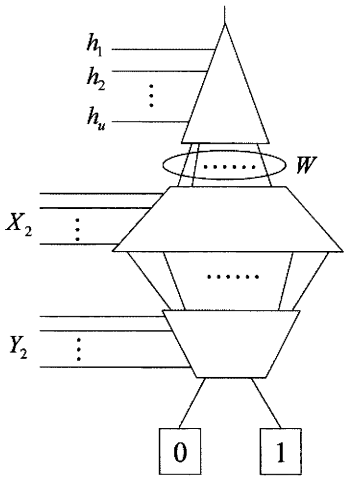
【図３】



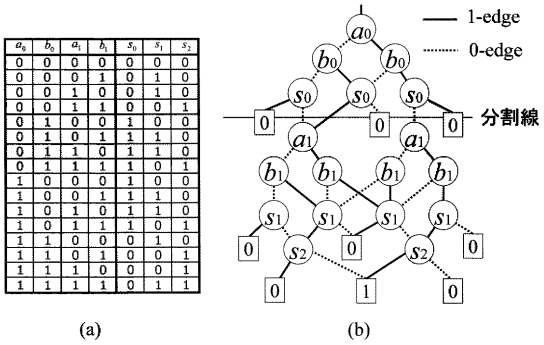
【図４】



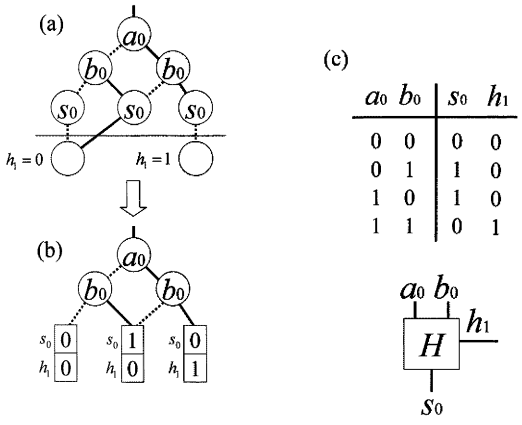
【図５】



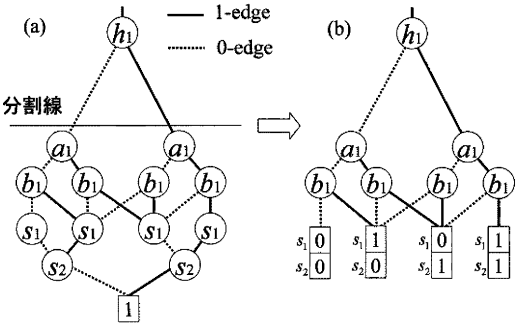
【図６】



【図７】



【図８】

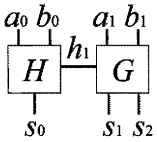


【図９】

(a)

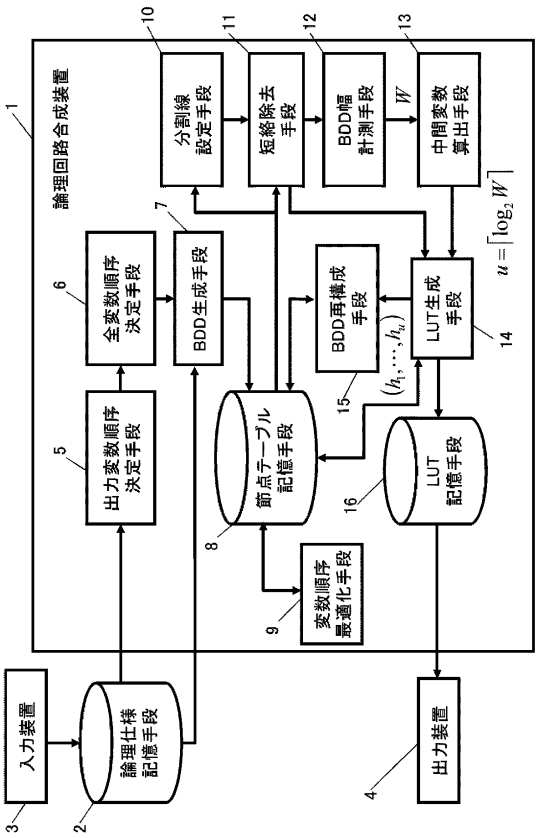
h_1	a_1	b_1	s_1	s_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(b)

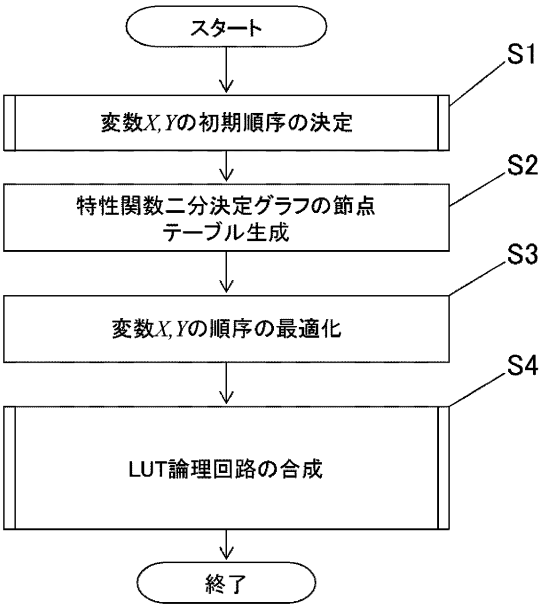


LUT数: 4
段数: 2

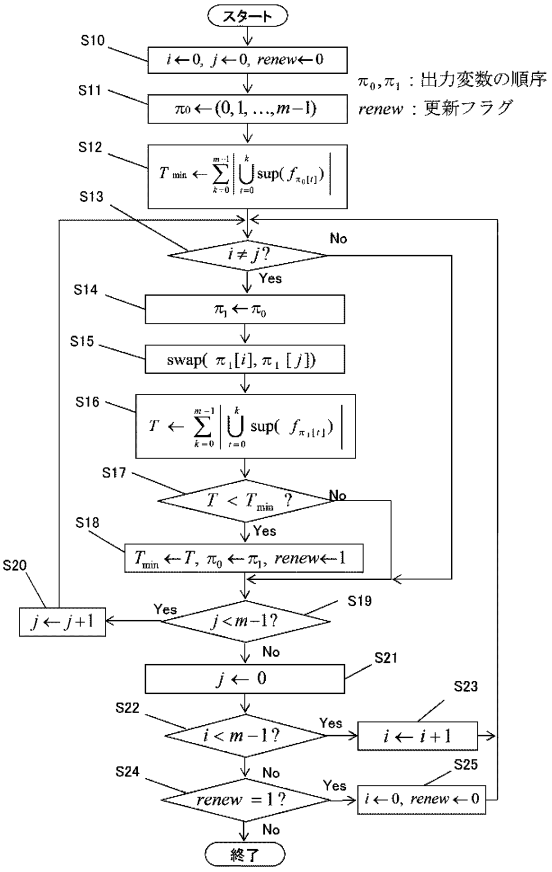
【図１０】



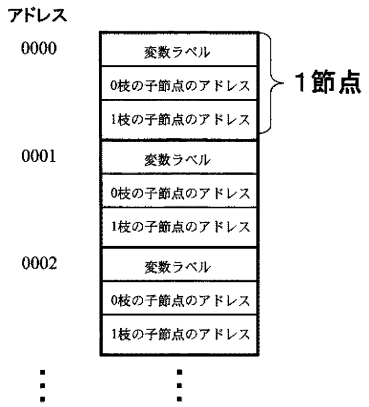
【図１１】



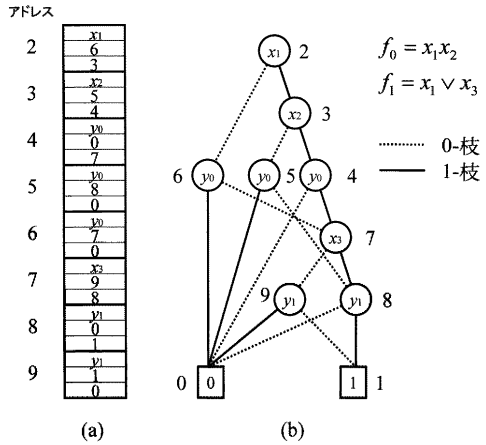
【図１２】



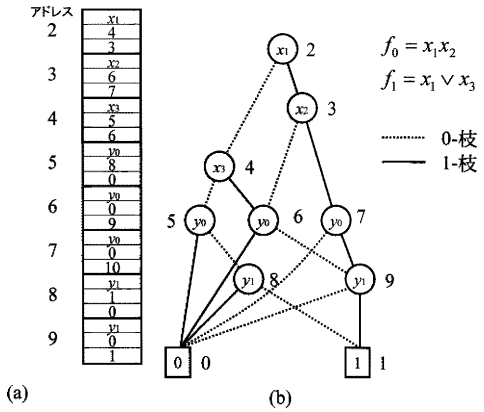
【図１３】



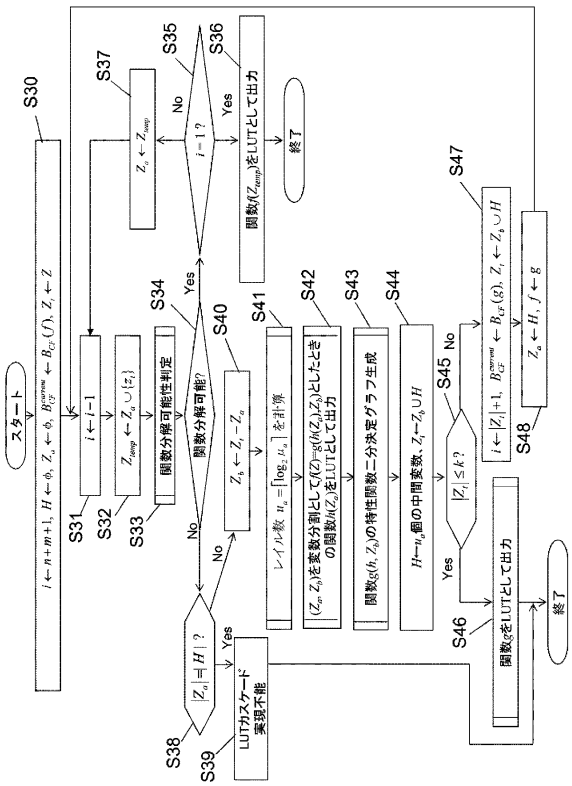
【図１４】



【図１５】



【図１６】

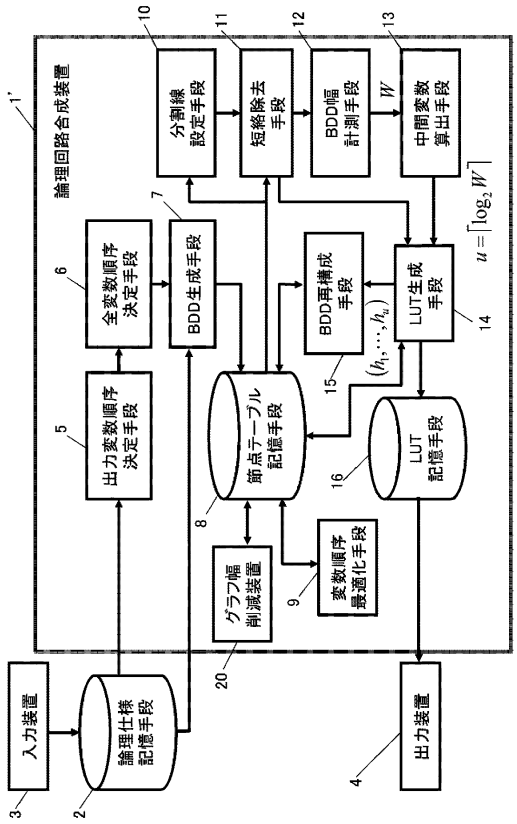


【図１７】

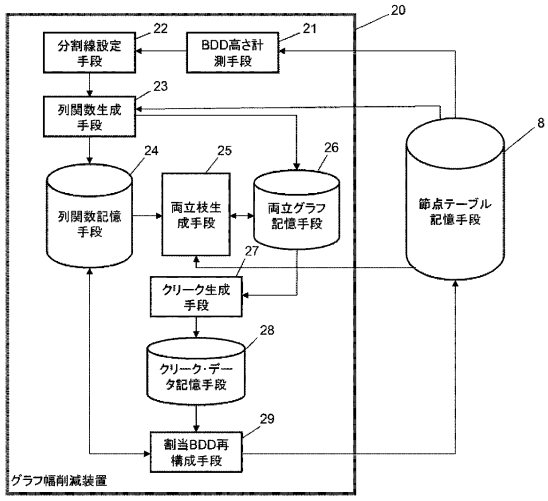
Name	In	Out	本発明の方法			Mishchenko [15]		
			LUT	段数	Cas	LUT	段数	Cas
C1908	33	25	320	11	5	3015	18	30
apex1	45	45	115	12	1	213	10	5
apex6	135	99	385	18	4	728	18	18
duke2	22	29	46	4	1	51	3	4
term1	34	10	50	8	1	37	6	2
ttt2	24	21	54	6	1	47	4	3

LUTの入力数をk=10とした場合

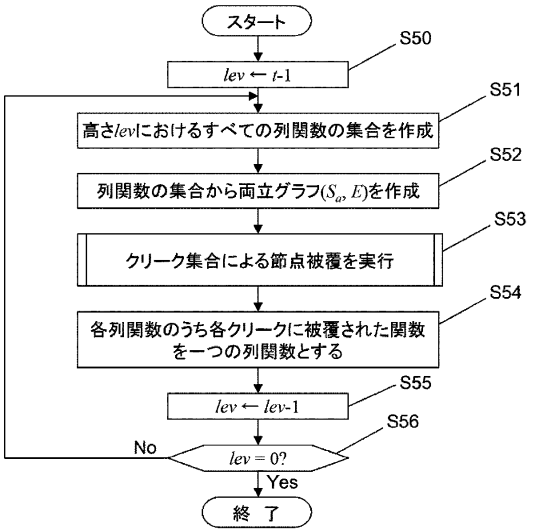
【図１８】



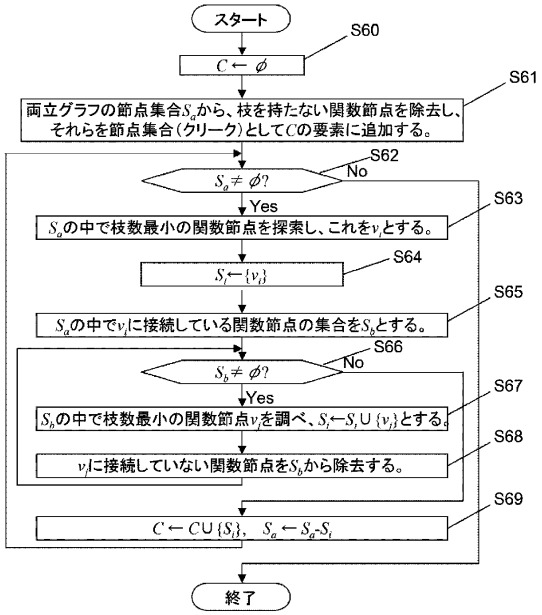
【図１９】



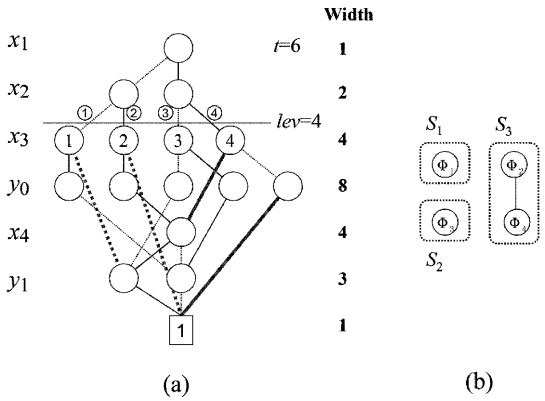
【図２０】



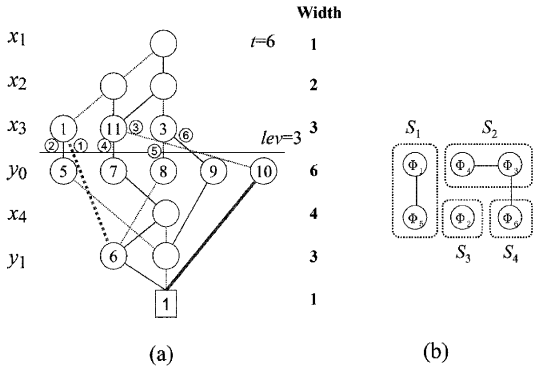
【図２１】



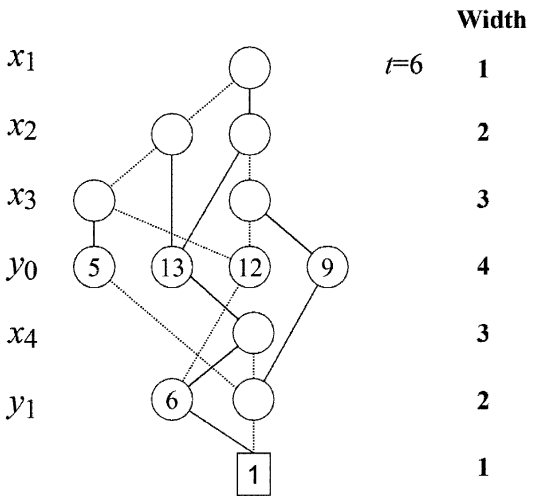
【図２２】



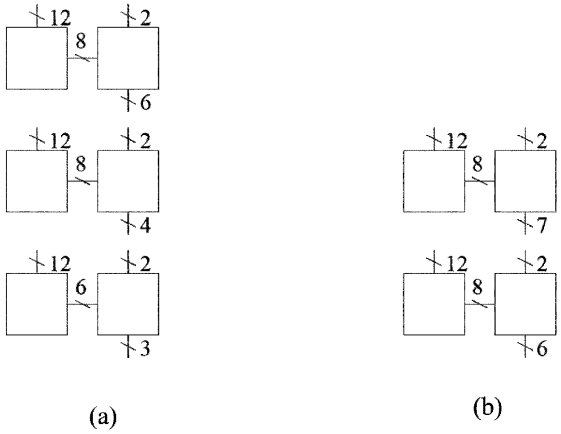
【図２３】



【図２４】



【図２５】



フロントページの続き

(56)参考文献　笹尾勤ほか，多出力関数のカスケード実現と再構成可能ハードウェアによる実現，電子情報通信学会技術研究報告，日本，社団法人電子情報通信学会，２００１年　4月　6日，Ｖｏｌ．１０１　Ｎｏ．３，５７－６４頁，ＦＴＳ２００１－８

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 17/50