設 計 自 動 化 77－17
(1995. 10. 20)

リタイミングと冗長除去を用いた順序回路の簡単化

四柳 浩之　梶原 誠司　樹下 行三

大阪大学大学院 工学研究科 応用物理学専攻
〒565 吹田市山田丘２－１
Tel: 06-879-7835　Fax: 06-879-7836
E-mail : {yanagi4, kajihara, kinoshita}@ap.eng.osaka-u.ac.jp

あらまし　論理回路に含まれる冗長な信号線は回路面積やテスト容易性に悪影響を与える．本研究では回路の冗長除去にリタイミングを用いた手法を提案する．本手法ではゲート数やフリップフロップ数を削減するために組合せ回路用のテスト生成を用いた冗長除去とリタイミングを適用する．リタイミングは２回行われるが，１つは組合せ回路的冗長に変換することを目的とし，他の１つはフリップフロップ数の削減を目的とする．リタイミングを行った後に組合せ回路用の冗長除去手法を適用すると順序回路的冗長として含まれている冗長の一部も除去できるため，簡単化に有効である．ベンチマーク回路に対する実験結果より本手法により多くの順序回路的冗長を含む冗長部分の除去が可能であることを示す．

キーワード　順序回路簡単化，リタイミング，故障検出，冗長除去，縮退故障

Optimization of Sequential Circuits using Retiming and Redundancy Removal

Hiroyuki Yotsuyanagi　Seiji Kajihara　Kozo Kinoshita

Department of Applied Physics, Faculty of Engineering, Osaka University
2-1 Yamadaoka, Suita, 565 Japan
Tel : +81-6-879-7835　Fax : +81-6-879-7836
E-mail : {yanagi4, kajihara, kinoshita}@ap.eng.osaka-u.ac.jp

Abstract　The existence of sequential redundancy in logic circuits will have bad effects on chip size and testability of sequential circuits. In this paper we propose a redundancy removal method for reducing the number of gates and flip-flops. The proposed method is comprised of redundancy removal using a combinational test generator and retiming. Retiming is utilized for two purposes: One is for finding sequential redundancies and another is for reducing the number of flip-flops. Applying redundancy removal techniques after retiming, not only all combinational redundancies but also several sequential redundancies can be removed. Experimental results for ISCAS'89 benchmark circuits show that this method can remove many sequential redundancies and reduce the number of gates and flip-flops.

key words　sequential logic optimization, retiming, fault detection, redundancy removal, stuck-at faults

1. はじめに

縮退故障が存在する回路が出力系列により正常回路との区別ができないとき，その故障を冗長故障という．その冗長故障に対応する信号線やゲートを除去しても，回路の論理的なふるまいは変わらないが，回路は小さくなり，一般的に高い故障検出率が得られるようになる．この観点では論理回路内の冗長はチップサイズやテスト容易性に悪影響を与えていることになる．それゆえ，冗長除去は順序回路の再合成の主要なものの一つとなっている．

組合せ回路に対しては，すべての冗長故障は検出不能な縮退故障として判定できるため，テスト生成を用いて冗長除去を行う手法がいくつか提案されている[1-5]．一方，順序回路の冗長故障は外部出力を変化させないが，状態遷移や状態割当を変化させるため，その判定は容易ではない．すべての状態遷移を考慮して順序回路の冗長判定や除去を行う手法が提案されているが[6-8]，それらはフリップフロップ数の多い回路に対しては実用的ではない．フリップフロップ数の多い回路に対して冗長判定や除去を行う手法についての研究はそれほど多くはない．そのうちの1つとして，回路のフィードバックループをなくすように回路を任意の信号線で切断し，順序回路的冗長故障を判定・除去する手法が提案されている[9]．また順序回路的冗長故障だけでなく，部分的検出可能故障の一部も回路の通常動作を変化させずに除去することが可能であることが[10]に述べられている．

本論文では同期式順序回路を簡単化するために，リタイミングと冗長除去を用いて順序回路的冗長故障を多く削除する手法を提案する．リタイミングは同期式順序回路の再合成手法の一つで，回路内のフリップフロップの配置を変えることで回路動作の高速化[16,18]や，ゲート数やフリップフロップ数の削減[13-15]，テスト容易性の向上[12,17]などが実現できる．また，リタイミングを行うことで組合せ回路的冗長に変換される順序回路的冗長故障があることが知られている[12]．本手法においてはリタイミングにより組合せ回路的冗長故障に変換される順序回路的冗長故障があることを利用して多くの冗長を除去することを試みる．

本手法ではリタイミングを二回行うが，初めは順序回路的冗長を多く組合せ回路的冗長に変換することを目的とし，二回目はフリップフロップ数を削減することを目的とする．それぞれのリタイミングを行った後の回路に対して組合せ回路のテスト生成手法を用いた冗長除去を行う．本手法により得られる回路は元の回路に比べゲート数，フリップフロップ数が削減される．

本論文では回路はある同期化系列のような初期化系列をもち，初期化系列印加中の出力については無視すると仮定する．もし初期状態に依存する部分の出力が意味を持つ場合はリタイミングや冗長除去の際に初期状態について考慮する必要があるが，それについては[13]に述べている．

本論文は以下のように構成される．2．ではリタイミングと冗長故障との関係について考察する．3．では冗長除去に有効な到達不能状態の削除について述べる．4．ではフリップフロップ数の削減にリタイミングを用いる手法について述べる．5．で本手法による順序回路の簡単化の手順について紹介する．6．で実験結果，7．でまとめを示す．

2. リタイミングと冗長故障

2．1 縮退故障の検出可能性

ある縮退故障について故障回路と正常回路を区別する入力系列が存在しないとき，その故障は冗長であるという．順序回路においては同じ入力系列を印加しても現れる出力系列は初期状態によって異なる．状態の集合を S として，正常回路が状態 $s_i \in S$ のときに入力系列 Y を印加したときの出力系列を g_i，故障回路が状態 $s_i \in S$ のときに入力系列 Y を印加したときの出力系列を f_i とする．これらの記号を用いて，入力系列 Y を印加したときの正常回路の取りうる出力系列の集合 $G(Y)$ と故障 f の存在する回路の取りうる出力系列の集合 $F(Y)$ は次のように表される．

$$G(Y)=\bigcup_{s_i \in S} g_i \qquad F(Y)=\bigcup_{s_i \in S} f_i$$

$G(Y)$ と $F(Y)$ を用いると，順序回路の単一縮退故障はその検出可能性により以下のように分類できる[6,8]．

定義1：$G(Y) \cap F(Y) = \phi$ となる入力系列 Y が存在するとき，故障 f を検出可能故障（detectable fault）という．

検出可能故障は $G(Y)$ と $F(Y)$ に共通な出力系列がないため，入力系列 Y を印加して得られた出力系列が G と F のどちらに含まれているかを調べれば，初期状態に関わらず故障が存在しているか否かがわかる．従って入力系列 Y はテスト系列になっている．

定義2：検出可能ではないが $G(Y) \neq F(Y)$ となる入力

系列 Y が存在するような故障 f を確率的検出可能故障（probabilistically detectable fault）という．

確率的検出可能故障は入力系列 Y により検出される場合とされない場合がある．例えば出力系列 $f_j \in F(Y)$ と同じ出力系列が $G(Y)$ にないならば，その故障は初期状態 s_j で入力系列 Y を印加することで出力系列 f_j が観測されるので検出が可能である．ところが初期状態が s_j でなければ正常回路の出力系列に含まれる系列を出力するため，入力系列 Y を印加しても故障の検出は不可能である．このように確率的検出可能故障が検出されるか否かは回路の初期状態に依存している．

定義３：すべての入力系列 Y において $G(Y) = F(Y)$ である故障 f を冗長故障（redundant fault）という．

冗長故障は組合せ回路的冗長故障と順序回路的冗長故障の２つに分けられる．

定義４：すべてのフリップフロップの入出力線をそれぞれ外部出力・外部入力とした組合せ回路において検出不能である故障を組合せ回路的冗長故障（combinationally redundant fault）という．

定義５：組合せ回路的冗長故障でない冗長故障を順序回路的冗長故障（sequentially redundant fault）という．

組合せ回路の場合は定義１と定義３に従って検出不能である故障はすべて冗長故障である．ところが順序回路の場合には，組合せ回路部分で検出可能な順序回路的冗長故障が存在する．図１の回路で信号線 r の１縮退故障は出力系列に影響を与えない冗長故障である．しかし図２に示した回路の組合せ部分では検出可能故障である．よって，この故障は組合せ回路的冗長故障ではなく順序回路的冗長故障である．

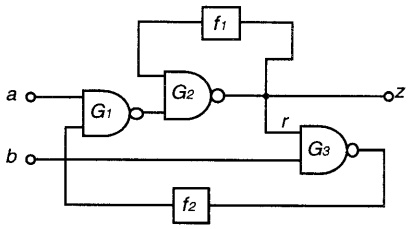

図１　順序回路的冗長故障の例

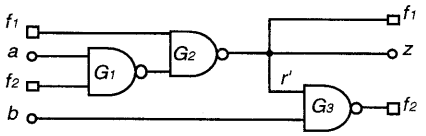

図２　図１の回路の組合せ部分

本論文では与えられた回路は同期化系列のような初期化系列が存在し，その初期化系列印加中の回路動作については無視できるものとする．故障が冗長となる信号線は回路の出力論理を変えずに除去することが可能である．また，確率的検出可能故障のうち，故障回路が初期化系列をもつ場合は，初期化系列印加後の回路の出力論理を変えずに除去することが可能である[10]．

２．２　リタイミング

同期式順序回路の再合成手法の１つとして，回路内のフリップフロップの位置を移動することで再配置するリタイミングが提案されている[11-19]．リタイミングはフリップフロップを出力方向に再配置する前方への再配置と入力方向に再配置する後方への再配置の組合せによって行われる．

フリップフロップがゲートの出力に配置されているとき，図３(a) のようにそのフリップフロップを除き，そのゲートのすべての入力線上にフリップフロップを配置することができる．これをゲート部の後方への再配置という．逆に図３(b) のようにあるゲートの全入力線上にフリップフロップが配置されているときにはそれらをゲートの出力線に再配置できる．これをゲート部の前方への再配置という．

また，フリップフロップが分岐の幹に配置されているときは，それをすべての枝に再配置することができる（図３(c)）．これを分岐部の後方への再配置という．逆に分岐のすべての枝にフリップフロップが配置されているときはそれらを幹へと再配置できる（図３(d)）．これを分岐部の前方への再配置という．

リタイミングを行った後の回路は元の回路と完全に等価ではないことがある．ゲート部分の前方・後方

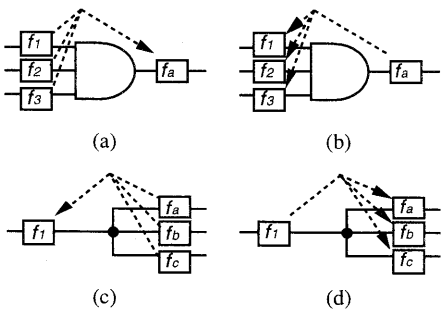

図３　リタイミングの基本操作
a)ゲート部の前方への再配置　b)ゲート部の後方への再配置
c)分岐部の後方への再配置　d)分岐部の前方への再配置

への再配置では等価な状態を持たない状態が現れることはない．図3(a)，(b)において状態$(f_1, f_2, f_3)=(1, 1, 1)$は状態$(f_a)=(1)$と等価であり，その他の状態はすべて$(f_a)=(0)$と等価である．

一方で，分岐部の後方への再配置を行うと，どの状態からも遷移できない到達不能状態は等価な状態を持たないことがある．図3(c)において状態$(f_1, f_2, f_3)=(1, 1, 1)$は状態$(f_a)=(1)$と等価であり，状態$(f_1, f_2, f_3)=(0, 0, 0)$は$(f_a)=(0)$と等価であるが，$(f_1, f_2, f_3)$に示されるその他の状態に等価な状態が存在するとはいえない．例えば，$(f_1, f_2, f_3)=(1, 1, 0)$と等価な状態は存在しない．到達不能状態に依存する出力は初期化系列印加中にしか現れないため，正常回路の回路動作には影響しない．しかし，故障の検出可能性に変化を与えるため，冗長除去に有効に用いることができる．

分岐部の前方への再配置を行うと，元の回路にある状態に等価な状態が必ず存在している．しかし，元の回路に等価な状態が存在しない状態が付け加わることがある．図3(d)において状態$(f_a)=(1)$は状態$(f_1, f_2, f_3)=(1, 1, 1)$と等価であり，状態$(f_a)=(0)$は$(f_1, f_2, f_3)=(0, 0, 0)$と等価である．しかし，その他の状態（$(f_1, f_2, f_3)=(0, 1, 1)$など）は必ずしも状態$(f_a)=(0)$，$(1)$のどちらかに等価であるとはいえない．このときは元の回路に等価な状態を持たない状態がリタイミングにより付け加わっていることになる．

元の回路に等価な状態を持たない場合は初期化系列をそのまま用いることができない可能性がある．しかし，数クロック分入力を印加した後に元の回路の初期化系列を印加すれば初期化できることが知られている[19]．

2．3　リタイミングと除去可能な故障

リタイミングにより変換された回路において，元の回路で検出可能である故障は変換後においても検出可能であることが知られている[19]．それ以外の故障については検出可能性が変化することがある．またリタイミングにより順序回路的冗長が組合せ回路的冗長に変換されることがある[12]．例として図1の回路を考える．信号線rの1縮退故障は順序回路的冗長故障であった．この回路に対してゲートG_3の後方への再配置を行い，分岐部を2つに分けて得られる回路と，この回路の組合せ回路部分を図4，図5に示す．図1の信号線rに対応する信号線をr'とすると，図5の組合せ回路の信号線r'の1縮退故障は，故障の影響をゲートG_2より先に伝搬させることができないため，冗長である．従って，図4の順序回路の信号線r'の1縮退故障は組合せ回路的冗長故障である．

順序回路的冗長故障だけではなく，確率的検出可能故障が組合せ回路的冗長故障に変換されることもある．図6と図7に例を示す．図6の回路の信号線rの1縮退故障は確率的検出可能故障である．なぜなら，入力$a=1$に対して$G=\{0, 1\}, F=\{1\}$となるからである．つまり$a=1$を入力して出力に0が観測されれば回路内にこの故障が存在しないことがわかる．この回路に分岐部の後方への再配置によるリタイミングを行うと図7の回路になる．信号線rに対応する信号線r'の1縮退故障は組合せ回路的冗長故障になってい

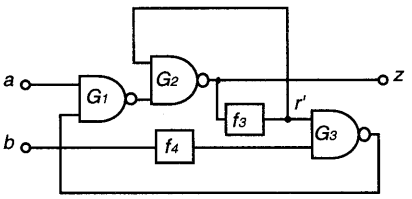

図4　図1の回路にリタイミングを行った後の回路

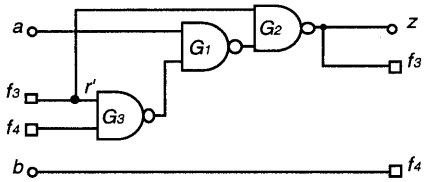

図5　図4の回路の組合せ部分

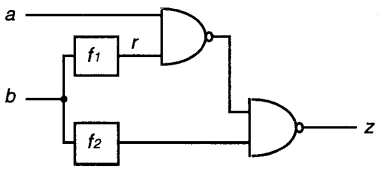

図6　確率的検出可能故障の例

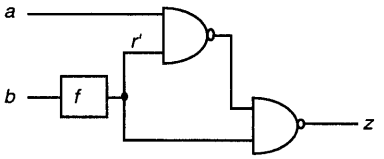

図7　図6の回路にリタイミングを行った後の回路

る．このようにリタイミングにより組合せ回路的冗長に変換される故障があれば，組合せ回路用の冗長除去手法を用いてそれらを除去することが可能になる．

3　到達不能状態の削除

組合せ回路用のテスト生成では，フリップフロップの値はすべて独立に制御できることを仮定している．そのため，組合せ回路部分では検出可能な故障でも順序回路的には検出不能故障となる可能性がある．例えば，組合せ回路のテストパターンで到達不能状態に相当するフリップフロップの値が必要である故障は，順序回路的冗長故障か確率的検出可能故障である．なぜなら，任意の状態から到達不能状態へ遷移させることのできる入力系列は存在しないからである．

2．2節で述べたように到達不能状態のいくつかは分岐部の後方への再配置により対応する状態が存在しなくなる．本論文ではこのことをリタイミングによる到達不能状態の削除という．もし組合せ回路で検出するのに必要な状態に到達不能状態があり，それがリタイミングにより削除されたならば，その故障は組合せ回路的冗長故障へと変換される．

以下に組合せ回路的冗長に変換できる故障をなるべく多く変換するためのリタイミング手法について述べる．この手法では元の回路に含まれる一部の到達不能状態の削除を目的とする．

本手法では到達不能状態を見つけるためにある1つの分岐の幹での信号値とフリップフロップの信号値との関係に注目する．ある分岐の幹 s において以下のような記号を用いる．$F(s)$ を s から他の分岐を通らずに接続している経路のあるフリップフロップの集合とする．そのうち，s の信号値が0（1）であるときに一意的に値の定まるフリップフロップの集合を $F_0(s)$（$F_1(s)$）とする．

もし，$F_0(s)$ と $F_1(s)$ の要素数が両方とも1以上であれば，到達不能状態が存在している．なぜなら，$F_0(s)$ 内のフリップフロップ f が信号線 s に0を与えることで一意的に信号値 v に定まるならば，f の信号値が $\bar{v}$ であるときは必ず信号線 s の値は1である．このとき $F_1(s)$ 内のフリップフロップの信号値は一意的に定まるので，その値の示す状態以外はすべて到達不能状態である．そのような到達不能状態は $F_0(s)$ と $F_1(s)$ に含まれるフリップフロップを s に配置されるようにリタイミングを行うと削除することができる．

到達不能状態を削除するためのリタイミングの手順を手順ＲＤＵとする．手順ＲＤＵの概略を示す．

［手順ＲＤＵ (Retiming for Deleting Unreachable states)］

1）回路内の分岐の幹の集合を S とする．
2）S から未処理の分岐の幹 s を取り出す．
3）$F(s)$ を求める．
4）$F(s)$ 内のフリップフロップのうち，s の値を0（1）にして一意的に値の定まるフリップフロップの集合 $F_0(s)$（$F_1(s)$）を求める．
5）$F_0(s)$ と $F_1(s)$ の要素数が両方とも1以上であれば6へ．それ以外は8へ．
6）$F_0(s)$ と $F_1(s)$ に含まれるフリップフロップをゲートの後方への再配置を繰り返して分岐の枝へ配置する．
7）フリップフロップのある枝のみが枝となる分岐をつくり，その幹へフリップフロップを再配置する．
8）$S \neq \phi$ ならば2へ．$S = \phi$ ならば終了．

図8に例を示す．図8(a)の回路には到達不能状態 $(f_a, f_b) = (0, 1)$ がある．分岐 s について $F(s) = \{f_a, f_b\}$ である．s の値を0にすると f_a の値は一意的に1に定まるので，$F_0(s) = \{f_a\}$ となる．s の値を1にすると f_b の値は一意的に0に定まるので，$F_1(s) = \{f_b\}$ となる．$F_0(s)$ と $F_1(s)$ の要素数が両方とも1以上であるため，到達不能状態があることがわかる．そこで，$F_0(s)$ と $F_1(s)$ に含まれるフリップフロップを再配置して，s にフリップフロップを配置する．再配置を行ったあとの回路を図8(b)に示す．図8(a)の回路の到達不能状態 $(0, 1)$ はリタイミングにより削除されるため，図8(b)の回路では対応する状態がない．

このリタイミングを行うと回路内のフリップフロップ数は増加することがあるが，多くの到達不能状態を削除することができる．それゆえ多くの故障を組合せ回路的冗長故障に変換することができる．

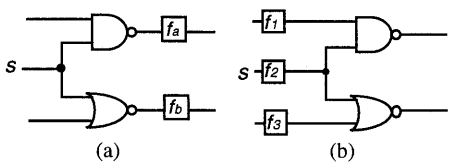

図8　到達不能状態の削除
a)回路例　b)後方への再配置を行った後の回路

4　フリップフロップ数の削減

リタイミングを行うことによりフリップフロップ数を削減できることがある．あるゲートの全入力にフリップフロップが配置されているときは後方への再配置を行うことでフリップフロップ数は削減される（図3(a)）．ある分岐についてすべての枝にフリップフロップが配置されているときは前方への再配置によりフリップフロップ数が削減できる（図3(c)）．

ゲート出力のフリップフロップをその入力へ再配置するとフリップフロップ数は増加するが，分岐部でフリップフロップ数が削減されるならばフリップフロップ数の総数は削減できる場合がある．分岐部で効率的にフリップフロップ数を削減するため，分岐の枝 i からフリップフロップまでの距離 d_i を次のように定義する．

定義6：分岐の枝 i から他の分岐点を通らずにフリップフロップに接続する経路があるとき，一番近いフリップフロップまでの経路上の2入力以上のゲート数に1を加えたものを i からフリップフロップまでの距離 d_i とする．それ以外の場合の距離 $d_i = 0$ とする．

ある分岐点において各枝での距離を求めることで，後方への再配置を行ってフリップフロップ数の削減が可能かどうかを判定できる．$d_i > 0$ となる i の数を n とすると，後方への再配置を繰り返すことで n 個のフリップフロップが分岐の枝に配置できる．この再配置によりフリップフロップ数は (Σd_i) 個に増加する．その後，分岐部の後方への再配置を行うことでフリップフロップ数が $(n-1)$ 個削減できる．したがって n 個のフリップフロップがリタイミングにより $(1 - n + \Sigma d_i)$ 個となる．よってこの数の比較を行い，フリップフロップ数が増加しないときにはリタイミングを行うことにすればよい．

フリップフロップ数の総数を削減するためのリタイミングの手順を手順ＲＲＦとする．手順ＲＲＦの概略を示す．

［手順ＲＲＦ (Retiming for Reducing the number of Flip-flops)］

1）分岐の幹の集合を S とする．
2）S からある未処理の分岐の幹 s を取り出す．
3）s の各枝 i に対してフリップフロップまでの距離 d_i を求める．
4）$d_i > 0$ となる i の数を n とし，$1 - n + \Sigma d_i \leqq n$ であれば5へ．不等式がなりたたなければ6へ．
5）$d_i > 0$ である分岐の枝にフリップフロップが配置されるように後方への再配置を行い，フリップフロップのある枝のみが枝となる分岐をつくり，その幹へフリップフロップを再配置する．
6）$S \neq \phi$ ならば2へ．$S = \phi$ ならば終了．

図9に例を示す．図9(a)の回路の分岐 s の各枝での距離を求めると，枝 b_1 からフリップフロップへの経路上には2入力ANDゲートがあるので距離 $d_{b1} = 2$，枝 b_2 と b_3 ではフリップフロップまでの経路上に2入力以上のゲートはないので $d_{b2} = d_{b3} = 1$ である．枝 b_4 からは別の分岐点までにフリップフロップがないので $d_{b4} = 0$ である．よって $n = 3$．このとき，$1 - n + \Sigma d_i = 2 \leqq n$ であるので，リタイミングによりフリップフロップ数が増加しないことがわかる．リタイミングを行った後の回路が図9(b)の回路である．このリタイミングによりフリップフロップ数は4から3に削減される．

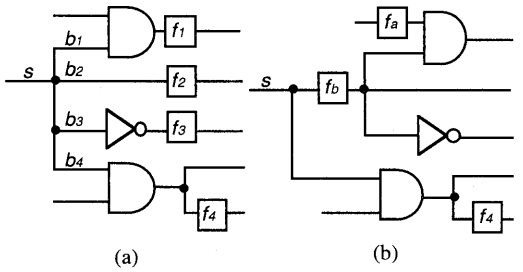

図9　フリップフロップ数の削減
a)回路例　b)リタイミングを行った後の回路

5．　簡単化の手順

3章と4章で述べたリタイミングの手順に組合せ回路用の冗長除去手法を組み合わせ，以下のように順序回路を簡単化する．

不要なゲートを取り除くために，1入力AND・ORゲートと連続する2つのNOTゲートを各処理の前後に除去する．

処理の概要は以下の通りである．

1）回路の組合せ部分の冗長除去を行う．
2）回路内のフリップフロップの位置を記憶する．
3）到達不能状態の削除のためのリタイミングを行う．（手順ＲＲＵ）
4）組合せ部分の冗長除去を行う．
5）2で記憶した位置へフリップフロップを戻す．
6）フリップフロップ数の削減のためのリタイミング

を行う．（手順ＲＲＦ）

7）組合せ部分の冗長除去を行う．

まず与えられた回路の組合せ回路的冗長故障を除去する．組合せ回路的冗長故障の中にリタイミングにより順序回路的冗長故障に変換されてしまうものがあるためである．その後，多くの故障を組合せ回路的冗長故障に変換するために３章で述べたリタイミングを行う．そして変換された組合せ回路的冗長の除去を行う．ここではフリップフロップ数が増加するため，このリタイミングを行う前にフリップフロップの位置を記憶し，冗長除去後に元の位置へと戻す．更にフリップフロップ数を削減するために４章で述べたリタイミングを行う．このリタイミングによっても新たに組合せ回路的冗長故障となる故障がありうるため，最後にもう一度組合せ回路部分の冗長除去を行う．

６．　実験結果

本手法をＣ言語で記述し，富士通 S-4/CL ワークステーション上で実験を行った．対象回路はISCAS'89ベンチマーク回路[20]である．本手法のリタイミングにより組合せ回路的冗長故障に変換された冗長部分のあった７つの回路に対しての結果を表１に示す．表中のORIGINALの欄はベンチマーク回路から連続する２つの NOT ゲート，１入力 AND・OR ゲートを除去した後のゲート数・フリップフロップ数を示している．ＲＲ１，ＲＲ２，ＲＲ３の欄はそれぞれ初めの冗長除去，手順ＲＲＵを行った後の冗長除去，手順ＲＲＦを行った後の最後の冗長除去を行った後の回路について，ゲート数・フリップフロップ数・除去された故障数を示している．

これらの結果から本手法によりゲート数・フリップフロップ数が元の回路の組合せ回路的冗長の除去を行った後にも削減されていることがわかる．

表２に処理時間を示す．組合せ回路用のテスト生成を用いた冗長除去が処理の多くの時間を占めている．リタイミングに要する時間は複雑な処理を必要とするテスト生成に比べてかなり短いことがわかる．s386 の最後の冗長除去は，２回目のリタイミングにより回路が変化しなかったため行っていない．

表３は文献[9]での手法による結果と本手法での結果との比較を示す．文献[9]ではベンチマーク回路を修正して用いているものがあり，元のベンチマーク回路よりフリップフロップ数が少ない回路で実験を行っている．それについては回路名に＊印をつけている．

表１　実験結果

	ORIGINAL		RR1			RR2			RR3		
circuit	#gate	#FF	#gate	#FF	#red	#gate	#FF	#red	#gate	#FF	#red
s386	133	6	133	6	0	138	25	12	130	6	0
s5378	1625	179	1596	176	29	1633	198	33	1459	158	0
s9234	2583	228	2387	228	258	2592	826	1	2151	200	16
s13207	3179	669	3014	667	412	3483	1152	111	2665	533	0
s15850	4470	597	4305	594	354	4808	1685	0	3951	579	1
s38417	12217	1636	12176	1636	137	13450	3446	29	12013	1578	0
s38584	15783	1452	14298	1435	455	15629	3580	16	13758	1426	0

表２　処理時間（sec.）

circuit	RR1	RDU	RR2	RRF	RR3	total
s386	1.96	0.05	3.18	0.05	—	5.28
s5378	90.6	0.42	46.3	0.88	39.6	179
s9234	2900	0.93	94.3	0.58	1539	4534
s13207	4812	2.37	519	5.65	150	5491
s15850	3285	3.80	227	7.13	209	3736
s38417	2555	15.62	1570	37.42	1484	5701
s38584	28754	58.04	1904	89.88	1370	32195

表３　本手法と文献[9]との比較

	本手法			文献[9]		
circuit	#gate	#line	#FF	#gate	#line	#FF
s386	130	352	6	159	359	6
s5378	1459	3528	158	1363	3096	119
s9234*	2151	4943	200	2644	5174	198
s13207*	2665	6803	533	2166	4227	318
s15850*	3951	9247	579	5023	10082	466
s38417	12013	26883	1578	18399	33692	1620
s38584	13758	30905	1426	——	——	——

s38417 に対してはゲート数・信号線数・フリップフロップ数ともに本手法のほうが多く削減されている．また，s386 ではフリップフロップ数は変化していないが，ゲート数・信号線数は多く削減されていることが分かる．

7． まとめ

本論文では同期式順序回路の再合成手法としてゲート数・フリップフロップ数の削減による簡単化を行う手法を提案した．本手法においては２つの異なるリタイミングの手順を用いた．１つは到達不能状態を削除するためにリタイミングを用い，更にフリップフロップ数を削減するためにリタイミングを用いた．初めのリタイミングにより順序回路的冗長故障や確率的検出可能故障が組合せ回路的冗長故障へと変換され，その除去により回路内のゲート数・フリップフロップ数を削減することができた．しかし，本手法適用後の回路にもまだ冗長部分が残っていることが考えられるため，今後さらに冗長部分を発見・除去するために有効なリタイミング手法について考察する必要がある．

参考文献

[1] D. Bryan, F. Brglez, R. Lisanke, "Redundancy Identification and Removal " MCNC Workshop on Logic Synthesis, May 1989.

[2] P. R. Menon, and H. Ahuja, "Redundancy Removal and Simplification of Combinational Circuits," IEEE VLSI Test Symposium, pp. 268-273, April. 1992.

[3] M. Abramovici, and M. A. Iyer, "One-Pass Redundancy Identification and Removal," 1992 ITC, pp. 807-815, Sept. 1992.

[4] R. Jacoby, P. Moceyunas, H. Cho, and G. Hachtel, "New ATPG techniques for logic optimization," International Conference on Computer-Aided Design, pp. 548-551, Nov. 1989.

[5] S. Kajihara, H. Shiba and K. Kinoshita, "Removal of Redundancy in Logic Circuits under Classification of Undetectable Faults," 22th International Symposium on Fault-Tolerant Computing, pp. 263-270, July 1992.

[6] I. Pomeranz, and S. M. Reddy, "Classification of Faults in Synchronous Sequential Circuits," IEEE Trans. on Comput., Vol. 42, No. 9, pp. 1066 - 1077, Sept. 1993.

[7] H. Cho, G. D. Hachtel, and F. Somenzi, "Redundancy Identification/Removal and Test Generation for Sequential Circuits Using Implicit State Enumeration," IEEE Trans. on CAD, Vol. 12, No. 7, pp. 935 - 945, July 1993.

[8] I. Pomeranz and S. M. Reddy, "On Identifying Undetectable and Redundant Faults In Synchronous Sequential Circuits," 12th IEEE VLSI Test Symposium, pp.8-14, May 1994

[9] K. -T. Cheng, "Redundancy Removal for Sequential Circuits without Reset States," IEEE Trans. on CAD, Vol. 12, No. 1, pp. 13 - 24, Jan. 1993.

[10] I. Pomeranz and S. M. Reddy, "On Achieving Complete Testability of Synchronous Sequential Circuits with Synchronizing Sequences," 1994 ITC, pp.1007-1016, Oct. 1994.

[11] C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Circuitry," Algorithmica, 6:5-35, 1991.

[12] S. Dey and S. T. Chakradhar, "Retiming Sequential Circuits to Enhance Testability," 12th IEEE VLSI Test Symposium, pp. 28 - 33, May 1994.

[13] H. Yotsuyanagi, S. Kajihara and K. Kinoshita, "Resynthesis for Sequential Circuits Designed with a Specified Initial State," 13th IEEE VLSI Test Symposium, pp. 152-157, May 1995.

[14] S. Malik, E. M. Sentovich, R. K. Brayton and A. Sangiovanni- Vincentelli, "Retiming and Resynthesis: Optimizing Sequential Networks with Combinational Techniques," IEEE Trans. on CAD, Vol. 10, No. 1, pp. 74 - 84, Jan. 1991.

[15] S. Lejmi, B. Kaminska and E. Wagneur, "Resynthesis and Retiming of Synchronous Sequential Circuits," ISCAS, pp. 1674 - 1677, 1993.

[16] G. De Micheli, "Synchronous Logic Synthesis: Algorithms for Cycle-Time Minimization," IEEE Trans. on CAD, Vol. 10, No. 1, pp. 63 - 73, Jan. 1991.

[17] D. Kagaris and S. Tragoudas, "Partial Scan with Retiming," 30th DAC, pp. 249 - 254, June 1993.

[18] K. Bartlett, G. Borriello and S. Raju, "Timing Optimization of Multiphase Sequential Logic," IEEE Trans. on CAD, Vol. 10, No. 1, pp. 51-62, Jan. 1991.

[19] V. Singhal, C. Pixley, R. L. Rudell and R. K. Brayton, "The Validity of Retiming Sequential Circuits," 32nd DAC, pp.316-321, June 1995.

[20] F. Brglez, D. Bryan and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," ISCAS, pp. 1929 - 1934, May 1989.