RC-001

論理合成における面積・遅延最適化のためのフォールスパスの活用について
Utilization of False Paths for Area/Delay Optimization in Logic Synthesis

三上 雄大†, 吉川 祐樹†, 市原 英行†, 井上 智生†
Takehiro Mikami, Yuki Yoshikawa, Hideyuki Ichihara, Tomoo Inoue

1 背景

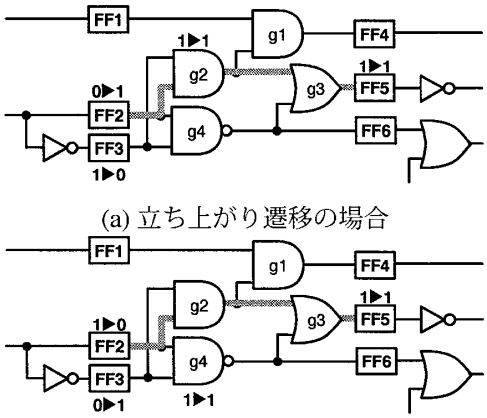
LSI の大規模化，高性能化に伴い，LSI は回路面積や信号の伝搬遅延時間など厳しい制約下での設計が求められている．本論文では，フォールスパス情報を用いてレジスタ転送レベル (RTL) 回路を論理合成することにより，生成されるゲートレベル回路の面積および遷移の伝搬遅延を最適化できることを示す．フォールスパスとは，始点での信号変化 (以降，遷移) が終点まで伝搬しないパスのことであり，フォールスパス上の遷移の伝搬遅延は回路性能に影響しない [1].

これまでに，組合せ回路もしくは順序回路の組合せ回路部分を対象としたフォールスパス判定法 [2, 3, 4] や，順序回路に対してフリップフロップの値の依存関係まで考慮したフォールスパス判定法 [5, 6] が提案されている．これらの手法はゲートレベル回路に対するフォールスパス判定法である．一方，RTL 回路に対して，回路構造とコントローラからの制御信号を解析することにより RTL でフォールスパスを判定する手法 [7] や，さらに高位の動作レベル情報を利用したフォールスパス判定手法 [8] が提案されている．これらの文献では，フォールスパス上の遅延が回路動作に影響しないことに着目し，フォールスパスを判定することにより，判定されたパス上の遅延故障に対するテスト費用を削減することを目的としている．テスト費用削減の例として，判定されたフォールスパス上の遅延故障を検出するためのテストパタン生成時間の削減や，製造テストにおいてチップにテストパタンを印加する時間の削減などが挙げられる．

本論文ではテスト費用の削減だけでなく，合成前に判定できるフォールスパス情報を利用することで，論理合成時に回路面積や遅延を最適化できることを示す．論理合成前に得ることのできるフォールスパス情報として，RTL 設計の段階で設計者が既知のフォールスパスや文献 [7, 8] の手法で判定できるフォールスパスがある．本論文では文献 [7] で判定できるフォールスパス情報を利用する．

一般に，回路にはフォールスパスが多数存在することが経験的に知られており，フォールスパスであるにも関わらずそのパスをクリティカルパスとした場合，遷移が伝搬しないパス上の伝搬遅延時間を回路の最大遅延時間として見積もってしまい，回路性能を低く見積もる可能性がある．そこで，論理合成前に判定されたフォールスパス情報を論理合成時に利用することで，このような誤った性能の見積もりを防ぐことができる．また，フォールスパス上の遅延は回路性能に影響しないことから，他のパスの遅延制約に違反しない範囲で，フォールスパス上の論理素子（ゲート）に遅延が大きく面積が小さい素子を使用できる．その結果，クリティカルパスではないパスについても，フォールスパス情報を利用することで，合成時に面積や遅延を削減することができる．実験的考察では，遅延制約下での面積最小化と面積制約下での遅延最小化について評価し，フォールスパス情報を利用することの有効性について考察する．

†広島市立大学大学院情報科学研究科



(a) 立ち上がり遷移の場合

(b) 立ち下がり遷移の場合

図 1: ゲートレベルフォールスパス例

2 フォールスパス

2.1 ゲートレベルフォールスパス

ゲートレベルパスとは，外部入出力やフリップフロップ (FF) とゲートからなる順序集合 $(g_0, g_1, ..., g_n)$ である．ここで，g_0 は外部入力または FF であり，g_n は外部出力または FF である．また，g_i $(1 \leq i \leq n-1)$ はゲートであり，g_i の出力は g_{i+1} の入力に接続する．これ以降，g_0 と g_n を FF として議論する．

ゲートレベルパス $p = (g_0, g_1, ..., g_n)$ 上の信号の遷移について考える．任意の連続する 2 時刻 t，$t+1$ について，時刻 t にパスの始点 g_0 の出力で遷移が起こらない，もしくは g_0 で遷移が起こったとしても時刻 $t+1$ までにその遷移がパス p の終点 g_n まで伝搬しない，もしくは g_n まで遷移が伝搬したとしても時刻 $t+1$ で g_n がその値を取り込まないとき，p はフォールスパスである．図 1 にゲートレベルフォールスパスの例を示す．この回路は順序回路の一部を抜き出したものである．ゲートレベルパス $(FF2, g2, g3, FF5)$ に着目する．このパスは，時刻 t に始点の $FF2$ で遷移が起こったとしても，その遷移は時刻 $t+1$ までに終点の $FF5$ まで伝搬しない．その理由は，$FF2$ で立ち上がりの遷移が起こる場合，$FF3$ の始点では立ち下がりの遷移が起こり，その結果，遷移の伝搬はゲート $g2$ で妨げられる．一方，$FF2$ で立ち下がりの遷移が起こる場合，その遷移の伝搬はゲート $g3$ で妨げら

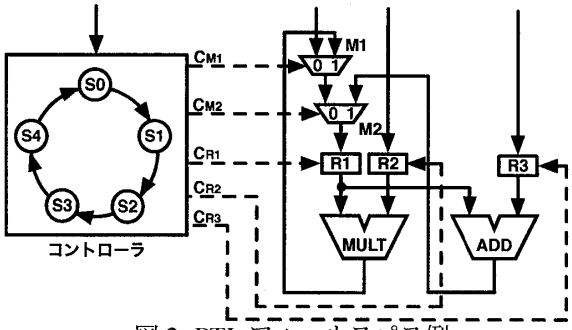


図 2: RTL フォールスパス例

れる．よって，このパスはゲートレベルフォールスパスである．

フォールスパス上の遷移は，パスの終点まで伝搬し，さらにその伝搬した値が終点の FF に取り込まれることはない．そのため，フォールスパス上の遷移の伝搬遅延時間は，回路の性能には影響しない．つまり，設計の観点から見ると，回路性能 (クロック周期) を決定する場合には，フォールスパスではないパスの中で最も伝搬遅延時間の大きいパスを基準に動作速度を決定すべきである．

2.2 レジスタ転送レベルフォールスパス

レジスタ転送レベル (RTL) パスとは，外部入出力やレジスタと組合せ回路要素からなる順序集合 $(m_0, m_1, ..., m_n)$ である．ここで，m_0 は外部入力またはレジスタであり，m_n は外部出力またはレジスタである．また，m_i $(1 \leq i \leq n-1)$ は組合せ回路要素であり，m_i の出力は m_{i+1} の入力に接続する．

RTL パス $q = (m_0, m_1, ..., m_n)$ 上の信号の遷移について考える．パス q の任意の連続する 2 時刻 t，$t+1$ について，時刻 t にパスの始点のレジスタ m_0 の出力で遷移が起こらない，もしくは m_0 で遷移が起こったとしても時刻 $t+1$ までにその遷移がパス q の終点のレジスタ m_n まで伝搬しない，もしくは m_n まで遷移が伝搬したとしても時刻 $t+1$ で m_n がその値を取り込まないとき，q は RTL フォールスパスである [7]．RTL パスは，論理合成後のゲートレベル回路の中で対応するゲートレベルパスの集合であり，RTL フォールスパスに対応するゲートレベルパスはすべてフォールスパスである．

図 2 に RTL フォールスパスの例を示す．この RTL 回路は評価実験で使用するベンチマーク回路 TSENG[9] の一部を示している．図中の C_{R1}，C_{R2}，C_{R3}，C_{M1}，C_{M2} はそれぞれレジスタ，マルチプレクサの制御信号を示している．コントローラの各状態の制御出力は，$S0 : (C_{R1}, C_{R2}, C_{R3}, C_{M1}, C_{M2}) = (H, L, H, 0, 0)$，$S1 : (L, H, H, 1, 0)$，$S2 : (L, L, H, 0, 1)$，$S3 : (H, H, L, 0, 0)$，$S4 : (L, H, H, 0, 0)$ である．ここで，H はレジスタのホールド信号，L はロード信号，0 および 1 はマルチプレクサの選択信号を示す．RTL パス $(R3, ADD, M1, R2)$ に着目する．状態 $S0$ から $S1$ の状態遷移について，状態 $S0$ のとき，パスの始点レジスタ $R3$ の制御信号 C_{R3} が H であるため，$R3$ の出力で遷移が起こらない．同様に，状態 $S1$ から $S2$，状態 $S2$ から $S3$ の状態遷移についても，パスの始点レジスタ $R3$ の制御信号 C_{R3} が H であるため，$R3$ の出力で遷移が起こらない．また，状態 $S3$ から $S4$ の状態遷移について，状態 $S4$ のとき，マルチプレクサ $M2$ の選択信号 C_{M2} が 0 であるため，遷移がパスの終点のレジスタ $R3$ まで伝搬しない．状態 $S4$ から $S0$ の状態遷移についても，同様である．このように，すべての状態遷移について $(R3, ADD, M1, R2)$ のパスでは遷移が発生しない．つまり，任意の 2 時刻で始点のレジスタで遷移が起こらない，もしくは，終点のレジスタまで遷移が伝搬しない，もしくは，終点のレジスタが遷移を取り込まない．したがって，RTL パス $(R3, ADD, M1, R2)$ は RTL フォールスパスである．

文献 [7] では，上述のようにコントローラの制御を解析することにより，データ転送の起こらない RTL パスを RTL フォールスパスと判定する手法を提案している．

3 フォールスパス情報を利用した論理合成

本論文では，論理合成時に RTL フォールスパス情報を用いることにより，面積や遅延を最適化できることに着目する．本節では，まず論理合成について述べ，続いてフォールスパス情報を利用した論理合成システム，論理合成時にフォールスパス情報を利用した場合の面積や遅延への効果について説明する．

3.1 フォールスパス情報を利用した論理合成システム

論理合成は，RTL 回路記述から論理式への変換，論理式の最適化，テクノロジマッピングの 3 つの工程により，RTL 回路をゲートレベル回路へ変換する．このとき，RTL 回路記述以外に，チップの動作環境などの使用条件，クロック周期や入力遅延値などの制約が与えられる．

論理式への変換は，if 文や case 文，処理や演算を解析し RTL 回路記述を論理式へ変換する．論理式の最適化では，論理の共有化や多段化によって論理を圧縮する．テクノロジマッピングでは，最適化された論理式に対して半導体テクノロジで使用できる論理素子を割り当てる．マッピングを行う際，論理合成ツールは論理素子の情報が記載されたセルライブラリを用いる．セルライブラリには，論理素子の素子面積や遅延時間の計算に用いられる固有値 (intrinsic)，抵抗値 (resistance) などが定義されている．表 1 にセルライブラリの例 (class ライブラリの一部) を示す．一般に，同じ機能を持つ論理素子でも素子面積や遅延時間の異なるものが複数存在する．例えば，論理式 $\overline{A+B}$ を表す NOR ゲートに着目すると，NR2 は，素子面積は NR2P と比較して小さいが，遅延時間は立ち上がり遷移，立ち下がり遷移ともに NR2P より大きい値となっている．テクノロジマッピングでは，与えられた制約を満たすように論理素子を割り当てる．

フォールスパス情報を利用した論理合成を行うために構築したシステムについて説明する．図 3 に論理合成システムの概要を示す．システムの入力はフォールスパスリスト，制約 (面積もしくは遅延時間)，RTL 回路記述であり，出力はゲートレベル回路である．このシステムは，

表 1: class ライブラリ

論理式	素子名	素子面積	遅延時間 [ns]	
			立ち上がり遷移	立ち下がり遷移
A·B	AN2	2	0.6243	0.8223
	AN2P	2	0.6118	0.8747
A + B	OR2	2	0.6243	0.8223
	OR2P	2	0.5253	0.9747
$\overline{(A \cdot B)}$	ND2	1	0.6377	0.2154
	ND2P	2	0.5623	0.2053
$\overline{(A + B)}$	NR2	1	0.8089	0.3089
	NR2P	2	0.6882	0.1931
$\overline{A}$	IV	1	0.5243	0.2089
	IVP	1	0.4253	0.1931
	IVAP	2	0.2931	0.1747

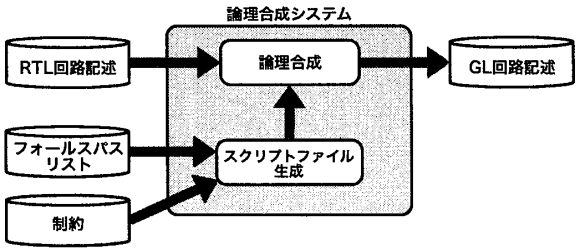


図 3: システムの概要

フォールスパスリストと制約から実行スクリプトファイルを自動生成し，論理合成を行う．論理合成には，論理合成ツールとして Design Compiler(Synopsys)，合成ライブラリとして class ライブラリを使用する．

3.2 論理合成におけるフォールスパス情報の効果

論理合成を行う際にフォールスパス情報を利用することによる効果を，回路例を用いて示す．この回路例では，特にマッピングに対する効果に着目している．使用する回路は，入力 A,B,C,D,E,F,G,H,I，出力 $Output1, Output2$ を持ち，以下のような演算を行うものとする．

$$Output1 = \overline{\overline{\overline{\overline{(A \cdot B) + C} \cdot D} + E}} \quad (1)$$

$$Output2 = \overline{\overline{\overline{\overline{(F \cdot G) + H} + I} \cdot E}} \quad (2)$$

3.2.1 遅延制約下での面積最小化合成

遅延時間が制約として与えられた場合を考える．遅延制約下での最小化項目は回路面積である．式 (1) および (2) を実現する回路に対して，遅延制約（4.07 [ns]）の論理合成を行うと，例えば図 4 のゲートレベル回路が得られる．各ゲートには，ゲートの種類と素子面積を記載している．この回路のクリティカルパスは，フリップフロップ H から ND2，OR2，ND2P，フリップフロップ $Output2$ へのパスである．

一方で，論理合成前に RTL フォールスパス情報が与えられ，その RTL フォールスパスに対応する合成後のゲートレベルパスが図 4 のフリップフロップ A,B,C,D,E から $Output1$ へのパスであると仮定する．このフォールスパス情報を用いて同じ遅延制約（4.07[ns]）の下で論理合成を行うと，図 5 のゲートレベル回路が得られる．この回路のクリティカルパスは，図 4 の回路と同様にフリップフロップ H から ND2，OR2，ND2P，フリップフロップ $Output2$ へのパスである．図 4 と図 5 の回路を比較すると，フォールスパス上のマッピングが変化している．$g3$ が素子面積 2 の NR2P でマッピングされていたものが，NR2P より素子面積が小さく遅延時間が大きい素子である NR2 でマッピングされている．フォールスパス情報がない図 4 では，この回路の 2 番目に遅延が大きいパスの遅延時間が 4.06[ns] であるため，$g3$ に素子の遅延時間が大きい NR2 を使用できない．しかし，フォールスパス情報がある図 5 では，フォールスパス上の遅延時間を考慮する必要がなくなり，遅延時間が大きく素子面積の小さい素子が使用できる．また，$g4$，$g5$，$g6$ の論理式が $\overline{(A + B)}$ から $C + E$ となり，OR2 がマッピングされている．これも同様の理由である．この結果，回路面積を 2 削減できる．

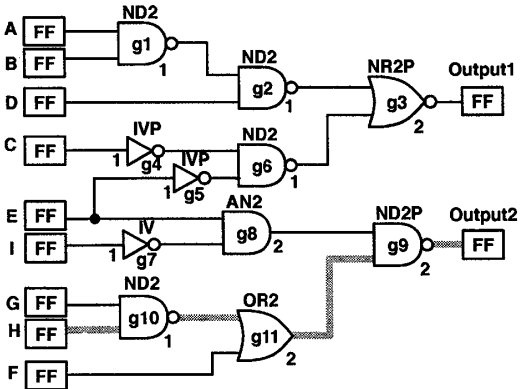


図 4: 面積最小化合成（フォールスパス情報なし）

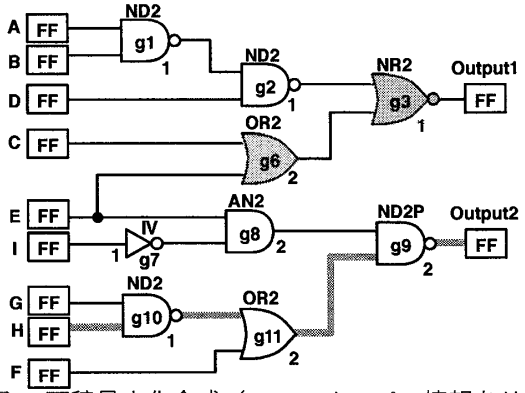


図 5: 面積最小化合成（フォールスパス情報あり）

以上のように，フォールスパス情報を利用することにより，そのパス上の遅延時間を考慮する必要がなくなり，素子面積の小さい素子を使用し回路面積を削減できる．

3.2.2 面積制約での遅延最小化合成

次に，回路面積が制約として与えられた場合を考える．面積制約下での最小化項目は遅延時間である．この条件

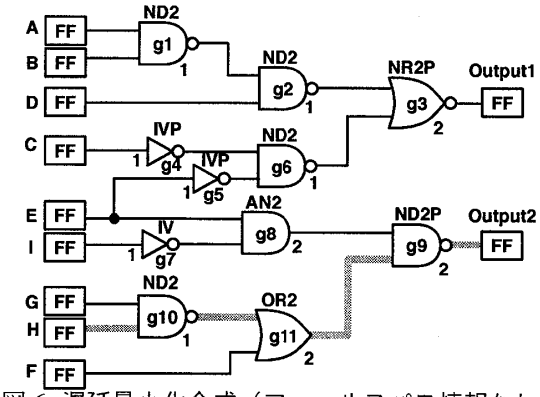


図 6: 遅延最小化合成（フォールスパス情報なし）

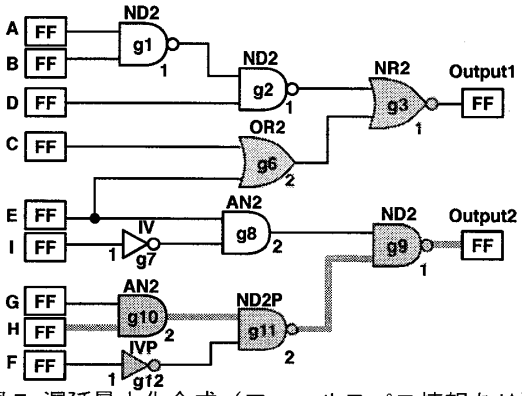


図 7: 遅延最小化合成（フォールスパス情報あり）

の下で論理合成を行う場合について，回路性能を決定するクリティカルパスがフォールスパスリストに含まれている場合と含まれていない場合の 2 つの場合を考える．

まず，クリティカルパスがフォールスパスリストに含まれる場合を考える．この場合，そのクリティカルパス上の伝搬遅延は回路性能に影響しないため，フォールスパスではないパスの中で最も遅延時間の大きいパスを基準に動作速度を決定できる．その結果，回路の最大の伝搬遅延時間が短縮され，回路性能が向上する．

次に，クリティカルパスがフォールスパスリストに含まれない場合を考える．式 (1) および (2) を実現する回路に関して，面積制約（125）の論理合成を行うと，例えば図 6 のゲートレベル回路が得られる．この回路のクリティカルパスは，フリップフロップ H から ND2，OR2，ND2P，フリップフロップ $Output2$ へのパスであり，回路の遅延時間は，4.07[ns] である．

一方で，論理合成前に RTL フォールスパス情報が与えられ，その RTL フォールスパスに対応する合成後のゲートレベルパスが遅延制約時と同様に図 6 のフリップフロップ A,B,C,D,E から $Output1$ へのパスであるとする．このフォールスパス情報を用いて同じ面積制約（125）下での論理合成を行うと，図 7 のゲートレベル回路が得られる．この回路のクリティカルパスは，フリップフロップ H から AN2，ND2P2，ND2，フリップフロップ $Output2$ へのパスであり，回路の遅延時間は 4.02[ns] である．このクリティカルパスはフォールスパスではない．図 6 と図 7 の回路を比較すると，回路の論理式およびマッピングが変化している．フォールスパス上は，遅延制約時と同様に論理式，マッピングの変化が起きており面積が 2 削減されている．これにより，面積制約に対し余裕が生まれる．そのため，削減された面積分をクリティカルパス上で使用できるようになり，使用素子の多い論理式への変形や素子面積が大きく遅延時間の小さい素子を使用できる．その結果，フォールスパス情報を利用することにより，遅延時間を短縮でき，回路性能が向上する．

3.3 削減効果の高いフォールスパス

論理合成時に用いるフォールスパス情報に関して，より回路面積や遅延時間削減の効果の高いパスについて考える．

遅延制約下では，遅延時間の大きいパスが回路面積削減の効果が高いと考えられる．フォールスパスに対しては遅延時間を考慮する必要がないため，そのパス上の素子に素子面積が小さく遅延時間の大きい素子を使うことにより回路面積が削減される．このとき，遅延時間の小さいパスは遅延制約に対して余裕があるため，フォールスパスであるかどうかに関わらず，パス上には遅延が大きく面積の小さい素子が配置されることが期待できる．一方で，遅延時間の大きいパスは遅延制約に対して余裕がないため，フォールスパスとして指定されない限り，面積の小さい素子が使用されることはない．そのため，フォールスパス情報を用いる場合，遅延時間の大きいパスのほうがより回路面積の削減効果が高いと考えられる．

面積制約の場合について考えると，クリティカルパスがフォールスパスリストに含まれる場合，遅延時間は当然削減される．クリティカルパスが含まれない場合でも，遅延時間の大きいパスが遅延時間削減の効果が高いと考えられる．これは，前述の通り遅延時間の大きいパスは回路面積を削減できる可能性が高いため，クリティカルパスなどの遅延時間の大きいパスの遅延時間を削減できる可能性が高まると考えられるためである．このように遅延制約，面積制約どちらの場合も遅延時間の大きいパスがより効果が高いと考えられる．

4 実験的評価

論理合成時にフォールスパス情報を利用した場合の有効性を確認するため，遅延制約下での面積最小化，面積制約下での遅延最小化を行う論理合成に関して評価実験を行った．ベンチマーク回路として PAULIN[10]，TSENG[9] を対象とした．論理合成は DELL Power Edge2950(CPU:2.3GHz, メモリ:4GB) 上で行い，論理合成には Synopsys 社の Design Compiler を使用し，セルライブラリは class ライブラリを使用した．論理合成の入力となるフォールスパスリストは，文献 [7] の手法により判定されたフォールスパスリストを使用した．

表 2 に遅延制約下での面積最小化合成の結果を示す．各列は左から順に，回路名，与えた遅延制約，フォールスパスリストの有無，回路面積，合成時間 (LS_time) を示している．回路面積は NOT ゲートの素子面積を 1 として算出している．PAULIN では，フォールスパスリストを与えた場合の面積は 18,327 であり，与えない場合の面積

表 2: 遅延制約面積最小化の論理合成結果

回路名	遅延制約 [ns]	FP 情報	回路面積	*LS_time*[s]
PAULIN	67.50	なし	18546	246.29
		あり	18327	668.70
TSENG	62.10	なし	12274	125.03
		あり	12000	131.57

表 3: RTL フォールスパスの選択と合成結果の変化（遅延制約 67.50 [ns]）

フォールスパス指定本数	降順		昇順	
	回路面積	*LS_time*[s]	回路面積	*LS_time*[s]
0	18546	246.29	18546	246.29
5	18367	498.92	18610	446.14
10	18327	668.70	18529	473.64

18,546 に比べて 219 (1.2%) 減少している．Tseng では，フォールスパスリストを与えた場合の面積は 12,000 であり，与えない場合の面積 12,274 に比べて 274 (2.2%) 減少している．ただし，どちらの回路も合成時間は延びている．特に，PAULIN に関してフォールスパスリストを与えた場合，与えない場合と比較して約 2 倍の論理合成時間を要した．これは，フォールスパス情報の利用により，フォールスパス上の遅延時間制約を満たす必要がなくなるため，より最適な解を探索したためと考えられる．

表 3 は，PAULIN に関して論理合成時に与えるフォールスパス情報を変化させたときの遅延制約下での面積最小化合成の結果を示している．PAULIN は文献 [7] の手法で 28 本の RTL パスがフォールスパスと判定されている．この実験では，面積の削減効果の高い RTL フォールスパスを調べるために，28 本の RTL フォールスパスそれぞれについて，対応するゲートレベルパスの中で最も遅延が大きいパスを調べ，その遅延を RTL フォールスパスの遅延とし，遅延の大きいものから RTL フォールスパスを降順にソートした．クリティカルパスの遅延は 67.5 ns であり，表 3 にはクリティカルパスとの遅延値の差が 1%未満の遅延の大きい RTL フォールスパスを 5 本，10 本与えたときのそれぞれの合成結果を降順の列に示している．また，遅延の小さいパスから 5 本，10 本与えたときの合成結果を昇順の列に示している．遅延の小さいパスは，クリティカルパスに比べて遅延値が 6%から 30%程度小さい．

遅延の大きいパスを与えた場合，回路面積は 5 本与えたときに約 200 減少し，さらに 5 本追加した場合，回路面積は約 240 減少している．一方，遅延の小さいパスから与えた場合，5 本与えたときは約 60 増加，さらに 5 本追加した場合，約 20 減少している．このことから，遅延の大きいフォールスパスを論理合成に利用したほうが効果が高いことがわかる．ただし，与えるパス数が増加すると合成時間が延びる傾向にあるため，面積削減の効果と合成時間のトレードオフを考慮する必要がある．そのトレードオフを考慮した効果的なフォールスパスの選択は今後の課題である．

表 4 に面積制約下での遅延最小化合成の結果を示す．遅延制約下での面積最小化と同様に，フォールスパスリストを与えることにより，PAULIN，TSENG ともに 1%未満ではあるがクリティカルパスの遅延が削減されている．表 5 は表 4 と同様に，PAULIN について，与えるフォールスパス情報を変化させたときの面積制約下での遅延最

表 4: 面積制約遅延最小化の論理合成結果

回路名	面積制約	FP 情報	遅延時間 [ns]	*LS_time*[s]
PAULIN	17400	なし	69.70	163.21
		あり	69.40	269.56
TSENG	11800	なし	62.70	107.74
		あり	62.40	119.06

表 5: RTL フォールスパスの選択と合成結果の変化（面積制約 17400）

フォールスパス指定本数	降順		昇順	
	遅延時間 [ns]	*LS_time*[s]	遅延時間 [ns]	*LS_time*[s]
0	69.70	163.21	69.70	163.21
5	69.40	269.56	69.60	242.73
10	69.50	264.31	69.70	232.19

小化合成の結果を示している．遅延最小化に関しても，遅延の大きいパスを与えたときのほうが，遅延の小さいパスを与えたときに比べてクリティカルパスの遅延が削減されていることがわかる．ただし，5 本と 10 本の結果からフォールスパス情報を多く与えたとしても，合成ツールが途中で探索を打ち切ることにより，少ない本数を与えた場合より質の悪い解を出力する場合があることがわかった．

これらの評価実験より，遅延の大きいゲートレベルパスを含む RTL フォールスパスが，面積／遅延最小化に有効であることがわかった．一般に，遅延の大きなゲートレベルパスに関して，そのパス上のゲートを多く共有する周辺パスの遅延も同様に大きいと考えられる．そのため，1 本の遅延の大きなゲートレベルフォールスパス情報を論理合成時に与えたとしても，周辺パスの遅延制約を満たすために，面積や遅延を削減できない可能性がある．

一方，RTL パスは複数のゲートレベルパスの集合であり，RTL フォールスパス情報を論理合成時に利用することにより，RTL フォールスパスに対応する全てのゲートレベルパスをフォールスパスとして扱うことができる．つまり，遅延の大きなゲートレベルパスの集合をまとめて遅延制約から除外できる可能性がある．そのため，RTL フォールスパス情報を利用した論理合成は，面積，遅延の削減に効果的であると考える．

5 まとめ

本論文では，論理合成においてフォールスパス判定法により判定されたフォールスパス情報を利用することの有効性に着目した．実験結果より，フォールスパス情報利用によって論理合成時に回路面積や遅延時間を削減できることを示した．また，フォールスパス情報利用の有効性の解析のために，ゲートレベルパスの遅延時間を調査し，それに基づいたフォールスパスの選択を行った．これを踏まえ，論理合成前にフォールスパスの遅延時間を予測することができれば，論理合成に利用する RTL フォールスパスを適切に選択できると考えられる．今後の課題として，適切な RTL フォールスパス選択手法の考案が挙げられる．

参考文献

[1] G.D. Micheli, Synthesis and Optimization of Digital Circuit, *McGraw-Hill*, 1994.

[2] K.T. Cheng and H.C. Chen, "Classification and identification of nonrobust untestable path delay faults," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol.15, no.8, pp.854–853, Aug. 1996.

[3] S. Kajihara, K. Kinoshita, I. Pomeranz, and S.M. Reddy, "A method for identifying robust dependent and functionally unsensitizable paths," *Proc. International Conf. on VLSI Design*, pp.82–87, 1997.

[4] S.M. Reddy, S. Kajihara, and I. Pomeranz, "An efficient method to identify untestable path delay faults," *Proc. 10th IEEE Asian Test Symp.*, pp.233–238, 2001.

[5] A. Krstic, S.T. Chakradhar, and K.T. Cheng, "Testable path delay fault cover for sequential circuits," *Proc. European Design Automation conf.*, pp.220–226, 1996.

[6] R. Tekumalla and P.R. Menon, "Identifying redundant path delay faults in sequential circuits," *Proc. 9th International Conf. VLSI Design*, pp.406–411, 1996.

[7] Y. Yoshikawa, S. Ohtake, T. Inoue, and H. Fujiwara, "Fast false path identification based on functional unsensitizability using RTL infomation," *Proc. 14th Asia and South Pacific Design Automation Conf.*, pp.660–665, 2009.

[8] S. Ohtake, N. Ikeda, M. Inoue, and H. Fujiwara, "A method of unsensitizable path identification using high level design information," *International Conf. on Design & Technology of Integrated Systems in nanoscale era*, 2010.

[9] C.J. Tseng and D.P. Siewiorek, "Automated synthesis of datapaths in digital systems," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol.5, no.3, pp.379–395, July 1986.

[10] P.G. Paulin and J.P. Knight, "Force directed scheduling for the behavioral synthesis of asics," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol.8, no.6, pp.661–679, June 1988.