リタイミング機能による遅延最適化

3C－7

中村 敦資　河原林 政道　淺香 俊治　前田 直孝

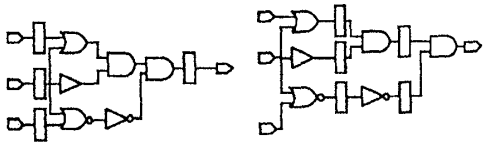
NEC

1　はじめに

論理合成における遅延最適化手法の一つとして，リタイミングを用いた手法がある。本稿では，実設計回路向きに改良されたリタイミングを用いた遅延最適化手法を提案し，論理合成システム Varchsyn[1] 上での実験結果について述べる。

2　概要

リタイミングとは，順序回路において入力信号に対する出力論理を変えることなく，DFF などのレジスタ素子を，組み合わせ論理回路をまたいで移動させる遅延最適化手法であり，レジスタ数最小化，リシンセシスとの併用等の研究もなされている [2][3]。リタイミングは，レジスタを移動して組み合わせ回路部分の遅延時間配分を均等化することにより，回路全体をより速いクロック周波数で動作可能にする。



(a) before retiming　(b) after retiming

図 1. リタイミング

従来，リタイミングを用いた遅延最適化のアルゴリズムとして，逐次処理による方法や線形プログラミングによる方法が提案されているが，実設計回路に適用するときの様々な制約により，実用的規模の回路に対し，実用的な計算時間内で良好な解を得られなかった。

本手法は，1相クロックのエッジトリガータイプ DFF から成る順序回路を対象に，与えられた面積制約を満たしつつ，遅延制約を満たすことを目標にリタイミングを用いた遅延最適化を行う。このとき，外部入出力端子を含んだパスの優先処理，大規模回路に対する組合せ論理回路部分のブロック化を行うことにより処理時間の短縮と解の高品質化を実現した。さらに，非同期セット／リセット機構付き DFF の移動も実現した。

3　従来手法

従来の逐次的手法を用いたリタイミングによる遅延最適化は，回路中で，入力端もしくは出力端が DFF であり，かつ最大遅延制約を違反しているパス（クリティカルパス）の中で，最大遅延制約を最も違反しているパス（ワーストクリティカルパス）を選択し，パスの入力端の DFF を出力方向に，または出力端の DFF を入力方向に移動することによって，パスの最大遅延を削減する。これをクリティカルパスが解消するまで繰り返す手法であった。

しかし，パスの選択順序や DFF の移動方向の決定法が単純であったため，冗長なレジスタ移動が多く発生し，大規模回路では膨大な処理時間が必要であった。

4　本手法

4.1　データ構造

本手法では，回路を重み付き有向グラフで表す。組み合わせ論理回路や入出力端子を vertex で表す。組み合わせ論理回路間の接続を vertex 間の edge で表し，edge の向きは信号の流れる方向とする。DFF はこの edge 上にあるものとし，edge には DFF の個数を重みとして持たせる。

また，vertex は方向性フラグを持ち，過去の DFF 移動履歴を保持する。これにより，DFF の移動方向を限定し，過去の移動元への逆移動の禁止／許可を制御して処理時間を短縮する。

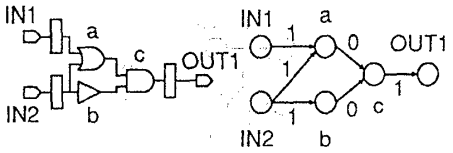


図 2. グラフ表現

4.2　遅延モデル

本手法の遅延モデルには，テクノロジ独立な回路に対してはユニット遅延をファンイン数およびファンアウト数で補正したものを採用し，テクノロジマッピング済みの回路に対しては各ブロックの内部伝播遅延と配線遅延を考慮した詳細な遅延モデルを採用した。

4.3　アルゴリズム

本手法の処理手順をアルゴリズム 1 に示す。Phase1 で外部入出力端子を含んだクリティカルパスを優先処理し，Phase2 で全てのクリティカルパスを対象として，ワーストクリティカルパスから順に改善する。

各クリティカルパスを改善する際に，パスの両端の vertex の方向性フラグをチェックし，一方の端の DFF を遅延削減の方向へ移動することが禁止されていた場合，パスの反対側の端の DFF を移動させ，新たにまたがれた vertex に方向性フラグを設定する。両端の方向性フラグが DFF 移動禁止を示していた場合そのパスの最大遅延は削減不可能であり，その時点で最適化を終了する。

DFF の移動方向によっては DFF 数が増減するため，面積制約に違反する場合は処理を終了する。

アルゴリズム 1

```
{
Phase1:
  for( 全ての外部入力 / 出力端子 ) {
    外部入力 / 出力端子を含んだクリティカルパスの
    出力 / 入力端の DFF を移動して最大遅延を削減;
```

Delay Optimization Using Retiming Technique
Atsushi NAKAMURA,
Masamichi KAWARABAYASHI,
Toshiharu ASAKA, Naotaka MAEDA,
NEC Corporation.

```
    またがれた vertex の方向性フラグを設定;
  }
Phase2:
  while( クリティカルパスがある ) {
    全クリティカルパスの中からワーストクリティカルパスを選ぶ;
    パスの両端の vertex の方向性フラグをチェック;
    if( 両端の DFF が移動不可 ) {
      exit;
    }
    else if( 片方だけが移動可 ) {
      DFF を移動して最大遅延を削減;
      またがれた vertex の方向性フラグを設定;
    }
    else if( 両方移動可 )
      出力端の DFF を移動して最大遅延を削減;
    }
  }
}
```

4.4 パスの選択順および DFF の移動方向

図 3 において，2 個の DFF で仕切られた 3 本のパスがあり，このうちパス p2 がワーストクリティカルパスであり，パス p3 は外部出力端子を含んだクリティカルパスであるとする。

従来のパスの選択順では，まずワーストクリティカルパス p2 が選ばれる。p2 の遅延削減のためには DFF1, DFF2 のどちらを移動してもよい。ここで DFF2 を入力方向に移動した場合，p3 が新たなワーストクリティカルパスになるが，p3 の遅延削減には DFF2 を再び元の位置に戻す以外にない。

そこで，外部出力端子を含むクリティカルパス p3 を最初に選択し，DFF2 を出力方向に移動してそのパスが最大遅延制約を満たすまで遅延を削減する。さらに，これ以降 DFF2 を入力方向へ移動することを禁止し，改善済みのパス p3 が再びクリティカルパスになることを防ぐ。次にワーストクリティカルパス p2 を選択し，移動が禁止されていない DFF1 を移動して最大遅延を削減する。以上の手法によって，DFF の不要な移動を削減し，処理時間を短縮する。入力端子を含んだクリティカルパスについても同様である。

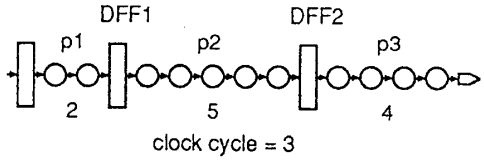


図 3. パスの選択順と DFF の移動方向

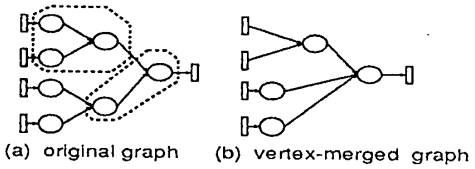


図 4. vertex のブロック化

4.5 vertex のブロック化

本手法では，グラフ上で DFF のない edge で接続された vertex 同士をブロック化することにより，グラフの規模を縮小するとともに DFF 配置可能位置を削減し，DFF 移動の試行回数を削減する。マージ後の vertex の遅延値が指定された値以内におさまる範囲でブロック化する (図 4)。

4.6 非同期セット / リセット付 DFF の扱い

本手法では 非同期セット／リセットピンを内蔵した DFF を，リセット回路部分を等価回路で展開して [4] 表現せず，リセット状態を持った DFF としてそのまま扱う。この場合，セット／リセットピンも DFF とともに移動するため，リセット状態での回路出力はリタイミング後も不変であることが要求される。そこで，DFF 移動毎に，移動後の位置での新しいリセット状態を計算し，その DFF に与える (図 5)。

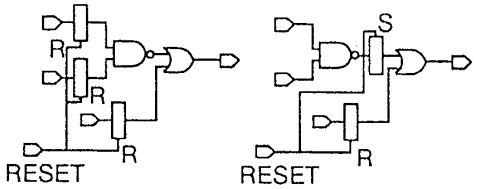


図 5. リセット状態

5 実験結果

表 1 に，実設計で用いられた 6 種類の VHDL 記述データを論理合成システム Varchsyn 上に読み込み、通常の最適化を実行した結果と、その後でリタイミングによる遅延最適化を追加実行した結果を示す。面積増加を 30% まで許容し，遅延制約としてクロックサイクルを 8ns とし，Sparc 670 MP 上で実行した。遅延は平均 5% 程度改善された。また，vertex のブロック化によって実行時間が 3 分の 1 程度に短縮することが可能なことも確認できた。

表 1 リタイミング前後の最大遅延値

回路	面積 (セル)		FF 数		最大遅延値 [ns]			CPU
	前	後	前	後	前	後	(相対値)	time [s]
C1	652	652	6	6	11.52	10.78	(0.93)	2
C2	132	150	2	4	10.81	10.42	(0.96)	1
C3	406	520	32	44	10.50	9.87	(0.94)	22
C4	1061	1370	12	39	27.06	26.17	(0.96)	32
C5	715	907	20	40	22.66	19.85	(0.87)	10
C6	1676	2190	18	69	34.12	30.36	(0.89)	245
C6(b)	1676	1960	18	44	34.12	30.28	(0.88)	80

C6(b) は回路 C6 を vertex のブロック化を行ってからリタイミングした結果。

6 おわりに

論理合成システム Varchsyn 上での，リタイミング機能を用いたタイミング最適化について述べた。今後の課題として，遅延解析の実行回数の削減，他手法との組み合わせ等がある。

参考文献

[1] 前田 直孝, 他, 「Varchsyn(1): 論理合成システムの全体構成」, 第 46 回情処全大, 1993

[2] C.E.Leiserson and J.B.Saxe, "Retiming Synchronous Circuitry", In TM 372, MIT/LCS, Oct 1988

[3] S.Malik, et al, "Retiming and Resynthesis: Optimizing Sequential Networks with Combinational Techniques", IEEE Trans. on Compter Aided Design, 10(1), Jan 1991

[4] 中田 広, 他, 「同期回路における論理を保存したラッチの挿入方法」, 電子情報通信学会論文誌, J75-A, 12, pp.1849-pp.1858, 1992