情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-SLDM-147 No.26
2010/12/1

束データ方式による非同期式回路を対象とした動作合成とフロアプランの統合

濱 田 尚 宏[†1]　齋 藤 寛[†1]

本稿では，束データ方式による非同期式回路を対象とした動作合成とフロアプランの反復適用手法を提案する．提案手法は，動作合成手法とフロアプラン手法において，タイミング制約を考慮しつつ性能最適化を行う．ケーススタディとして，提案手法を用い EWF を合成し，提案手法の有用性について評価する．

Integration of Behavioral Synthesis and Floorplanning for Asynchronous Circuits with Bundled-data Implementation

Naohiro Hamada[†1] and Hiroshi Saito[†1]

In this paper, we propose the iterative application of behavioral synthesis and floorplanning for asynchronous circuits with bundled-data implementation. Proposed methods optimize the performance while considering timing constraints during behavioral synthesis and floorplanning. We evaluate the effectiveness of proposed methods through synthesizing EWF.

1. はじめに

LSI の設計規模の増大に対し設計者の設計能力が追いつかないという問題が発生している．この問題の解決法の一つとして，設計の抽象度を引き上げ，引き上げられた抽象度から設計を行う方法がある．現在では，回路の動作仕様からレジスタ転送レベル (RTL) モデルを自動合成する動作合成技術が広く使われるようになってきた．また，LSI の微細化に伴い，設計された回路の性能において配線遅延が支配的になってきている．これにより，レイアウト設計後の回路の性能において，多くの部分を論理遅延ではなく，配線遅延が占めてきている．そのため，動作合成で最適な回路が得られたとしても，レイアウト設計後には最適な回路ではなくなっている可能性がある．したがって，設計の早い段階で配線遅延の影響を考慮する必要がある．

グローバルクロック信号で制御する同期式回路を対象とした動作合成手法は，多くの研究者によって提案されている[1)–4)]．しかしながら，同期式回路には，クロックツリーにおける消費電力，クロックスキューによる同期の失敗などの問題がある．一方，ローカルなハンドシェーク信号によって制御する非同期式回路では，クロック信号に起因する問題はなく，低消費電力，低電磁放射などの利点がある．しかし，アプリケーションに応じて，適切なデータエンコーディング方式，制御プロトコル，遅延モデルの選択が必要で，選択によって設計手法も異なり，更にはハザードフリーな回路が要求される為，非同期式回路の設計は熟練者でもない限り極めて難しい．それに加え，非同期式回路を対象とした動作合成手法自体も限られており，また，それらの手法では，配線遅延の影響が考慮されていない[5)–7)]．

本研究では，束データ方式による非同期式回路を対象とした動作合成とフロアプランの反復適用手法の提案を行う．提案手法は，束データ方式による非同期式回路のタイミング制約を考慮しつつ性能最適化を目標に動作合成とフロアプランニングを行う．

以下に，本稿の構成を述べる．2 節では，関連研究を説明する．3 節では，本稿で対象とする束データ方式による非同期式回路，動作合成手法，フロアプランニングについて説明する．4 節，5 節と 6 節では，提案手法の概要とケーススタディの考察を述べる．最後に，7 節では，本稿のまとめを述べる．

2. 関連研究

同期式回路を対象とした動作合成手法とフロアプランニング手法を同時に解く手法は，これまで多くの研究者によって提案されている[1)–4)]．これらの手法を非同期式回路に対し，直接的に適用したとしても，性能などの面で最適な回路を得ることは難しい．なぜならば，演算の完了が次の演算の開始を意味するという非同期式回路の特性が考慮されていないためである．

非同期式回路を対象とした動作合成手法は，これまでにも多くの研究者によって提案されている[5)–7)]．しかしながら，これらの手法では，配線遅延の影響が考慮されていない．そのため，動作合成で得られた性能と実際に配置配線まで行った性能との間に大きな誤差が生じてしまうという問題がある．

これらの研究とは異なり，本研究では，動作合成とフロアプランニングを繰り返し適用する手法を提案する．フロアプランニングを行う事によって，配線遅延を概算し，次の動作合成に活かす．また，提案手法は，束データ方式による非同期式回路のタイミング制約を考慮

†1 会津大学
The University of Aizu, Japan

Vol.2010-SLDM-147 No.26
2010/12/1

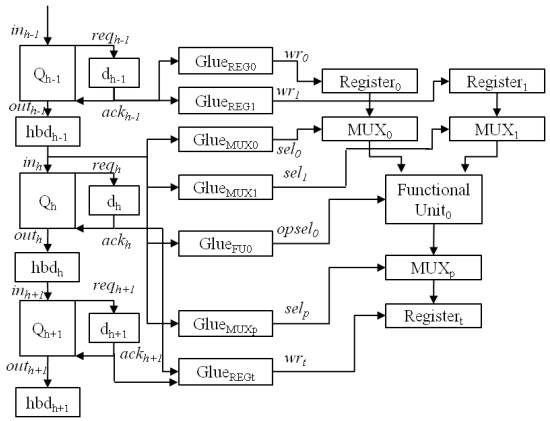

図 1 束データ方式による非同期式回路

しつつ，性能最適化を目標に動作合成とフロアプランニングを行う．

3. 基 礎 事 項

3.1 束データ方式による非同期式回路

束データ方式とは，非同期式回路におけるデータエンコーディング方式の一つである．束データ方式では，N ビットのデータ転送を $N+2$ 本の信号線で表現する．1 ビットのデータが 1 本の信号線に対応する．回路の制御には，ローカルなハンドシェーク信号（要求信号 req と応答信号 ack）を用いる．演算の完了を保証するために，要求信号 req の配線上に遅延素子を挿入する．遅延素子の遅延は演算の最大実行時間に対応する．

図 1 に本研究で対象としている束データ方式の回路モデルを示す．回路はデータパス回路と制御回路とで構成される．データパス回路は，演算を実行する演算器，入力データと演算結果を保持するレジスタと演算器とレジスタに対し適切な入力を選択するマルチプレクサで構成される．制御回路は，Q モジュール[8]，グルーロジックと遅延素子で構成される．回路内部の状態 $s_h(h=1,\ldots,\gamma)$ に一つの Q モジュールが割り当てれる．

次に，制御回路の動作を説明する．状態 s_h に対応する Q モジュール $q_h(h=1,\ldots,n)$ の入力信号 in_h が 0 から 1 に変化したときに動作が開始される．p 番目のマルチプレクサに対する選択信号 sel_p は in_h から生成される．in_h が 1 に変化した後，Q モジュール q_h は要求信号 req_h を立ち上げる．req_h は，対応する遅延素子を通過し，応答信号 ack_h として Q モジュール q_h に戻ってくる．t 番目のレジスタに対する書き込み信号 wr_t は ack_h から生成される．ack_h の立ち上がり後，Q モジュール q_h は req_h を立ち下げ，ack_h が立ち下がるのを待つ．ack_h が立ち下がったのち，レジスタに入力されている値が書き込まれる．Q モジュール q_h は，出力信号 out_h を立ち上げ，次の状態 s_{h+1} に対応する Q モジュール q_{h+1} の動作を開始させる．

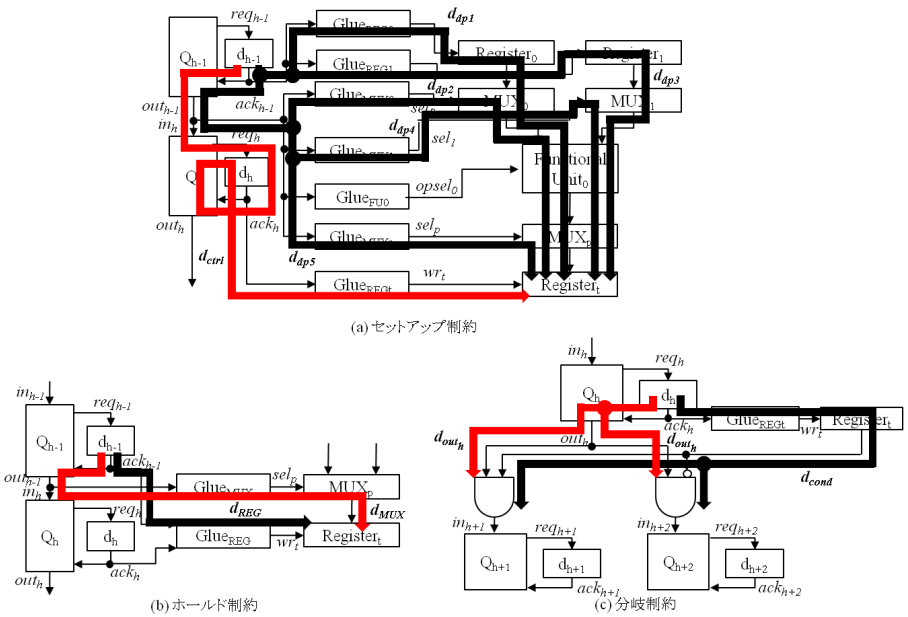

図 2 束データ制約

この回路が正しく動作するためには，以下の三つのタイミング制約を満たす必要がある．

- セットアップ制約: $d_{ctrl} > max(d_{dp1}, d_{dp2}, d_{dp3}, d_{dp4}, d_{dp5})$
- ホールド制約: $d_{MUX} > d_{REG}$
- 分岐制約: $d_{out_h} > d_{cond}$

セットアップ制約 (図 2(a)) とは，レジスタに値が到着したのちにレジスタ書き込み信号 wr_t が到着しなければならないという制約である．セットアップ制約が満たされない場合，レジスタに不正な値が書き込まれてしまう．ホールド制約 (図 2(b)) とは，状態 s_h とその次状態 s_{h+1} において同一のレジスタに値を書き込む際に，状態 s_h で生成されるレジスタ書き込み信号 wr_t が到着する前にレジスタへの入力データが変化してはいけないという制約である．状態 s_h からの状態遷移が条件信号 $cond$ に依存する場合，分岐制約 (図 2(c)) を満た

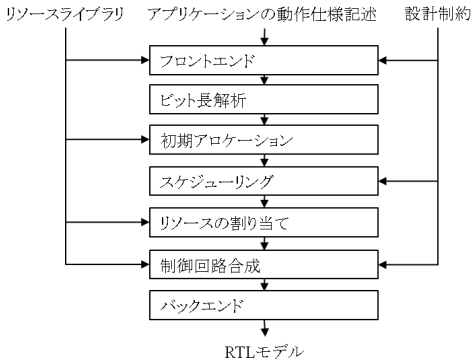

図 3 動作合成の流れ

す必要がある．分岐制約とは，対応する Q モジュール q_h の出力信号 out_h が立ち上がるよりも先に条件信号 $cond$ が次状態を決定する論理に到着していなければいけないという制約である．out_h が先に到着した場合，回路にレースが発生してしまう．

セットアップ制約が満たされない場合，対応する遅延素子の長さ d_h が短いということを意味する．したがって，セットアップ制約を満たすためには，遅延素子 d_h の長さを長くする必要がある．ホールド制約や分岐制約が満たされない場合，Q モジュール q_h の出力信号 out_h の信号線上に遅延素子を新たに挿入する必要がある．この新たに挿入された遅延素子は，回路の性能を悪化させてしまう．

3.2 動作合成

動作合成とは，プログラミング言語やその拡張 (C 言語，SystemC[9] など) で記述された動作仕様記述から RTL モデルを自動生成する手法である．動作合成では，与えられた設計制約下でスケジューリング，リソースの割り当てと制御回路合成を行いながら，設計空間の探索を行い，最適な RTL モデルを生成する．本研究では，著者らが以前提案した動作合成手法[11] を拡張する．図 3 に，動作合成の流れを示す．

動作合成手法の入力は，アプリケーションの動作仕様記述，リソースライブラリと設計制約である．フロントエンドでは，与えられた動作仕様記述の解析を行い，中間表現である CDFG(Control Data Flow Graph)[10] を生成する．CDFG 生成後，各演算と各変数のビット長の解析を行い，初期アロケーションを行う．初期アロケーションでは，演算器の割り当てを行い，演算の遅延を概算する．スケジューリングでは，時間制約，または資源制約の下，概算された演算の遅延を使い演算の開始時間を決定する．スケジューリング後，リソースの割り当てを行いデータパス回路を生成する．制御回路合成では，スケジューリング結果より状態を決定し，各状態に対し，一つの Q モジュールを割り当てる．そして，データパス回路を制御するためのグルーロジックを生成し，対応する Q モジュールの要求信号線上に遅延素子を挿入する．ここで，遅延素子の長さは，生成されたデータパス回路を基に各状態で必要となるリソースの遅延の総和を計算し，設計制約として与えられたマージン値を加えた値より決定する．最後に，バックエンドで Verilog HDL で記述された合成可能な RTL モデルを出力する．

なお，この動作合成手法[11] は，同期式回路を対象とした動作合成手法に対し，二つの変更を加えている．非同期式回路では演算の完了が次の演算の開始を表すため，動作合成で用いられる制御ステップは演算の開始時間より決定する．また，状態空間は，演算の開始時間より計算する．

3.3 フロアプラン

フロアプランニングとは，配置配線に先立って回路内のモジュールの位置と形状を決定する手法である．フロアプランニングを行うことで，モジュール間の配線遅延を概算することができる．本研究では，フロアプランニング手法として擬似焼きなまし法 (SA)[12] を用いる．

提案手法では，スライシングフロアプランを対象としてフロアプランニングを実行する．スライシングフロアプランとは，再帰的に回路を水平，または垂直に分割して得られたフロアプランのことである．

4. 性能最適化

本節では，提案する束データ方式による非同期式回路の性能向上について述べる．提案手法では，フロアプランニングにおいて，束データ制約を考慮しつつ，制御回路の遅延の最小化を行い，合成された回路の性能最適化を行う．また，動作合成でホールド制約と分岐制約の緩和を行うことで，遅延素子の挿入を抑える．遅延素子の挿入が抑えられることにより，合成された回路の性能の低下を抑えることが出来る．

4.1 制御回路の遅延の最小化

本研究で用いている回路モデルのレイテンシは，最初の Q モジュールの入力 in から最後の Q モジュールの ack の立ち下り遷移によるレジスタへのデータ書き込みまでの遅延となる．本稿では，この遅延を制御回路の遅延と呼び，また，次の式で定式化する．

$$\begin{aligned} Latency = & \sum_{i=1,\ldots,\gamma-1} (d_{i2r_i} + d_{a2r_i} + d_{a2o_i} + 2 * (dw_{q_i 2 d_i} + d_i + dw_{d_i 2 q_i})) \\ & + \sum_{i=1,\ldots,\gamma-1} (dw_{q_i 2 hbd_i} + hbd_i + dw_{hbd_i 2 q_{i+1}}) \\ & + (d_{i2r_\gamma} + d_{a2r_\gamma} + 2 * (dw_{q_\gamma 2 d_\gamma} + d_\gamma + dw_{d_\gamma 2 q_\gamma})) \end{aligned}$$

Vol.2010-SLDM-147 No.26
2010/12/1

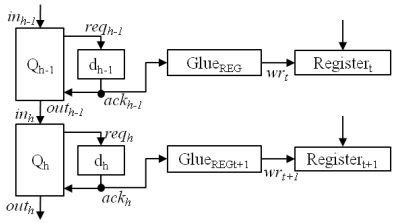

図 4 ホールド制約の緩和

ここで，d_{i2r_i}，d_{a2r_i}，d_{a2o_i} は，それぞれ，Q モジュール q_i の入力 in の立ち上がりから出力 req の立ち上がりまでの遅延，入力 ack の立ち上がりから req の立ち下りまでの遅延，入力 ack の立ち下りから出力 out の立ち上がりまでの遅延を表す．d_i は、遅延素子 d_i の遅延を表す．$dw_{q_i2d_i}$ と $dw_{d_i2q_i}$ は，Q モジュール q_i から遅延素子 d_i までの遅延と遅延素子 d_i から Q モジュール q_i までの配線遅延を表す．また，$dw_{q_i2hbd_i}$，hbd_i，$dw_{hbd_i2q_{i+1}}$ は，それぞれ，Q モジュール q_i から遅延素子 hbd_i までの配線遅延，遅延素子 hbd_i の遅延，遅延素子 hbd_i から Q モジュール q_{i+1} までの配線遅延を表す．上記の式において第一項，第二項，第三項は，それぞれ，最初の Q モジュールから最後の Q モジュールまでの Q モジュールと遅延素子にかかる遅延の総和，Q モジュール間の遅延，最後の Q モジュールでの遅延を表す．

なお，モジュール間の配線は以下のようにして概算する．まず，一方のモジュールの角からもう一方のモジュールの角までのマンハッタン距離を 16 通り列挙する (一つのモジュールは 4 つの角がある)．次に，各マンハッタン距離に対し，実際の配線遅延から導出した概算式を用いて配線遅延を概算する．概算された配線遅延の中で，最大となる配線遅延をモジュール間の配線遅延とする．

レイテンシを向上するために，上記の式を最小化するように，タイミング制約を考慮しつつ，フロアプランニングを行う．セットアップ制約より，制御回路の遅延はレジスタ間遅延よりも長くなくてはいけない．したがって，制御回路の遅延の最小化は，レジスタ間遅延の最小化を意味する．また，セットアップ制約以外のタイミング制約も考慮することにより，遅延のサイズは必要最小限となり，面積の最適化にもつながる．

4.2 ホールド制約の緩和

状態 s_h とその次状態 s_{h+1} において同一のレジスタに値を書き込むとき，ホールド制約を満たさなければいけない．ホールド制約が満たされない場合，レジスタに正しい値が書き込まれる前に値が変化してしまう．ホールド制約を満たすため，Q モジュール q_h の出力信号 out_h に新たに遅延素子を挿入する必要がある．しかしながら，新たに挿入された遅延素子は回路の性能を悪化させる．

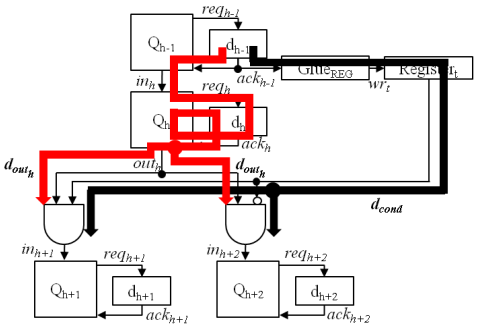

図 5 分岐制約の緩和

ホールド制約を緩和するために，提案手法では，レジスタの割り当て時に図 4 のように，連続した二つの状態 s_h と s_{h+1} で同一のレジスタを割り当てず新たなレジスタを割り当てる．よって，状態 s_{h+1} においてレジスタの入力信号が変化しなくなるため，ホールド制約の違反がなくなる．この方法では，レジスタ数が増加する可能性があるが，新たに遅延素子を挿入する必要はなくなる為，性能低下を抑えることができる．

4.3 分岐制約の緩和

分岐制約が満たされない場合，回路にレースが発生する．したがって，分岐制約を満たすためには，条件信号 $cond$ に依存して分岐をする Q モジュール q_h の出力信号 out_h に新たに遅延素子を挿入しなければならない．分岐制約を緩和するために，スケジューリングにおいて，条件信号 $cond$ を計算する演算は可能な限り早くスケジュールする．図 5 に示されるように，条件信号 $cond$ が状態 s_{h-1} で計算されることで，分岐制約を満たすことが容易になる．新たに遅延素子を挿入する必要がなくなり，性能低下を抑えることができる．

5. 動作合成とフロアプランニングの反復適用

図 6 に，動作合成とフロアプランニングを反復適用する手法の流れを示す．反復適用では，始めに C 言語で記述された動作仕様記述から RTL モデルを合成し，フロアプランニングを実行する．4 節で述べた性能最適化手法は，動作合成とフロアプランニングにおいて次のように適用する．4.1 節の手法はフロアプランニング時に，4.2 節の手法はリソースの割り当て時に，4.3 節の手法はスケジューリング時に，それぞれ適用する．

一回目の反復適用後，配線遅延の値が動作合成やフロアプランに反映される．初期アロケーションにおいて，直前のフロアプランからモジュール間の配線遅延の概算を行い，リ

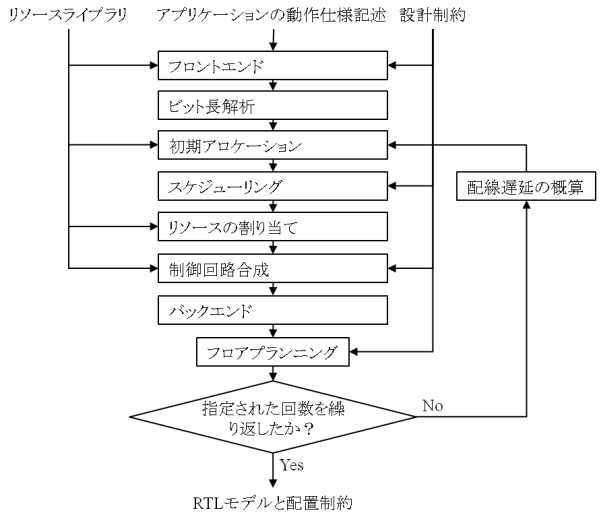

図 6 提案手法の流れ

ソースの遅延と配線遅延を用いて演算の最大実行時間を概算する．ここで概算された演算遅延を用いて，動作合成，フロアプランニングを行う．

提案手法では，出力として，Verilog HDL で記述された合成可能な非同期式回路の RTL モデルとフロアプラン結果を反映した配置制約を生成する．

6. ケーススタディ

本節では，提案手法を用いて EWF の合成を行い，提案手法の有用性を評価する．EWF は演算数 34 の分岐を含まないプログラムである．提案手法は Java を用いて実装を行い，デュアルコアプロセッサ (2.66GHz) と 2GB のメモリを搭載した Windows マシン上で実験を行った．

表 1 に，ケーススタディで用いたリソースライブラリの一部を示す．各リソースは，Verilog HDL を用いて記述され，Virtex5 (xc5vlx330-2ff1760) FPGA を対象デバイスとし Xilinx ISE 10.1 Foundation[13] を用いて合成を行った．表 1 の各行は，それぞれ，リソース名，リソースの入出力のビット長，利用されるスライス数，遅延を示す．スライスは，4 つのフリップフロップと 4 つの 6 入力ルックアップテーブルから構成される．通常，乗算器は FPGA に搭載されている専用乗算器を利用するが，本実験では専用乗算器を利用せず，スライスを用いて構成された乗算器を利用した．

表 1 リソースライブラリの一部

リソース名	ビット長	面積 [スライス数]	遅延 [ns]
mux2	32	9	0.08
mux8	32	16	0.35
register	32	8	0.45
adder	32	8	1.33
multiplier	32	144	6.80

表 2 配置配線後の面積と性能の結果

束データ制約考慮	面積 [スライス数]					レイテンシ [ns]	
	DP	CTRL	GL	DELAY	total	実測値	概算値
あり	702	51	32	174	959	286.622	263.171
なし	734	35	25	146	940	293.595	237.585
提案手法なし	1697	44	59	110	1910	278.728	54.040

ケーススタディにおいて，動作合成では時間制約としてクリティカルパス遅延を，遅延素子へのマージンはなし，という制約を与えて合成を行った．フロアプランでは，SA における次の温度を計算する係数と解のリジェクト率を 0.8 とした．提案手法の繰り返しの回数を 10 とし，提案手法を適用した．提案手法の評価は，提案手法の各繰り返しで得られた解のうち，最も良い性能を得られた回路を用いて行う．

動作合成およびフロアプランによって得られた回路に対し，Verilog HDL でテストパターンを記述し，Mentor Graphics 社の ModelSim 6.5a[14] を用いてシミュレーションを行う．動作の正しさを確認するため，C 言語からコンパイルした結果より得られた値との比較を行い，合成された回路が正しく動作していることを確認する．また，Virtex5 FPGA を対象デバイスとし Xilinx ISE 10.1 Foundation を用いて配置配線と STA を行い，配置配線後の回路の検証を行う．

表 2 に束データ制約を考慮しつつ提案手法を適用した結果と束データ制約を考慮しない場合の結果を示す．表の二行目に，束データ制約を考慮しつつ提案手法を適用した結果を，三行目に束データ制約を考慮せずに提案手法を適用した結果を示す．また，比較のため，提案手法を用いず，動作合成を適用後，任意に遅延素子を配置した結果を四行目に示す．ISE を用いて配置配線を行う場合，遅延素子が最適化され無くなってしまうことを防ぐために，少なくとも遅延素子のみを配置する必要がある．二列目から六列目に，それぞれ，データパス回路，制御回路，グルーロジック，遅延素子，回路全体の面積を示す．七列目と八列目に，それぞれ，配置配線後の回路において，回路の演算にかかる時間，ISE に入力する前に提案手法適用後に見積もられた回路の演算にかかる時間を示す．

表 2 より，合成された回路の性能は，束データ制約を考慮した場合，わずかではあるが性

表 3 束データ制約違反数

束データ制約考慮	セットアップ制約	ホールド制約
あり	11/216	0/0
なし	14/189	5/9
提案手法なし	13/212	7/8

能のよい回路が得られている．また，提案手法で概算された性能と実際に配置配線を行い得られた性能との誤差では，束データ制約を考慮した場合は，8%程度の誤差であるが，束データ制約を考慮しない場合には，19%程度の誤差が発生している．したがって，束データ制約を考慮することにより，合成後の回路の性能がよい精度で見積もれる ．

提案手法を用いずに，動作合成のみを適用した場合も，提案手法を適用した場合と同等の性能の回路が得られることが分かる．しかし，この場合，ISE は面積を増加させて，性能の良い回路を得ている．こうしたことを考えると，提案手法は面積を増やすことなく，ISE で任意に配置配線を行った時と同等の性能を持った回路を生成することが出来る．

表 3 に束データ制約考慮しながらフロアプランニングを行った場合と，束データ制約を考慮せずにフロアプランニングを行った場合のタイミング制約の違反数を示す．ケーススタディで利用した EWF には，分岐が存在しないため，セットアップ制約とホールド制約の違反数を示す．表の二行目に，束データ制約を考慮しつつ提案手法を適用した結果を，三行目に束データ制約を考慮せずに提案手法を適用した結果を示す．また，比較のため，提案手法を用いず，動作合成を適用後，任意に遅延素子を配置した結果を四行目に示す．列 “セットアップ制約” と列 “セットアップ制約” は，それぞれ，セットアップ制約の違反数とホールド制約の違反数を示す．また，表中の数字において左側の数字と右側の数字は，それぞれ，違反したパスの数，解析したパスの総数を示す．表 3 において，解析したパスの総数が異なるのは，提案手法で得られた各繰り返しで得られた解のうち，最も性能の良い回路を用いて比較したため，回路構造が束データ制約を考慮した場合，考慮しない場合と提案手法を用いなかった場合で異なるためである．

表 3 より，束データ制約を考慮することにより，セットアップ制約では，束データ制約を考慮しない場合と比べて違反数は減少している．4.2 節で述べた手法を用いることにより，束データ制約を考慮した場合，ホールド制約を違反するパスがなくなるため，0 となっている．また，提案手法を用いない場合と比べても，提案手法ではセットアップ制約の違反数は減少していることが分かる．

7. ま と め

本稿では，束データ方式による非同期式回路の性能最適化手法と束データ方式による非同期式回路を対象とした動作合成とフロアプランの反復適用手法の提案を行った．提案手法を用いることにより，商用ツールを用いた場合と同等の性能と半分近い規模の面積の回路を得ることが出来ることが分かった．また，束データ制約を考慮することにより，高い精度で性能を見積もることが出来，タイミング制約の違反数を減少させることができた．

今後の課題として，より多くのベンチマークでの実験を通して提案手法の評価を行うこと，パイプライン回路合成への拡張などがあげられる．

謝辞 本研究は，一部，文部科学省科学技術研究補助金 若手研究 (B) と科学技術振興事業団 (JST) の戦略的基礎研究推進事業 (CREST) の支援を受けて行われたものである．

参 考 文 献

1) Weng, J. and Parker, A.: 3D Scheduling: High-Level Synthesis with Floorplanning, *Proc. DAC*, pp.668–673 (1991).
2) Um, J., Kim J. and Kim, T.:Layout-Driven Resource Sharing in High-Level Synthesis, *Proc. ICCAD* , pp.614–618 (2002).
3) Xu, M. and Kurdahi F.J.: Layout-Driven RTL Binding Techniques for High-Level Synthesis Using Accurate Estimators, *ACM Trans. Design Automation Electronic Systems*, Vol.2, No.4, pp.312–343 (1997).
4) Gu, Z., Wang, J., Dick, R.P. and Zhou, H.: Unified Incremental Physical-Level and High-Level Synthesis, *IEEE Trans. Computer-Aided Design*, Vol. 26, No.9, pp.1576–1588 (2007).
5) Badia, R.M. and Cortadella, J.: High-Level Synthesis of Asynchronous Systems: Scheduling and Process Synchronization, *Proc. European Conference on Design Automation (EDAC)*, IEEE Computer Society Press, pp.70–74 (1993).
6) Bachman, B.M.: Architectural Synthesis of Timed Asynchronous Systems, *ICCD '99: Proceedings of the 1999 IEEE International Conference on Computer Design*, Washington, DC, USA, IEEE Computer Society, p.354 (1999).
7) Sacker, M., Brown, A.D., Rushton, A.J. and Wilson, P.R.: A behavioral synthesis system for asynchronous circuits, *IEEE Trans. Very Large Scale Integr. Syst.*, Vol.12, No.9, pp. 978–994 (2004).
8) Rosenberger, F., Molnar, C., Chaney, T. and Fang, T.-P.: Q-Modules: Internally Clocked Delay-Insensitive Modules, *IEEE Transactions on Computers*, Vol.37, No.9, pp.1005–1018 (1988).
9) Open SystemC Initiative, http://www.systemc.org/.
10) Ranganathan, N., Namballa, R. and Hanchate, N.: CHESS: A Comprehensive Tool for CDFG Extraction and Synthesis of Low Power Designs from VHDL, *ISVLSI '06* (2006).
11) Hamada, N., Shiga, Y., Konishi, T., Saito, H., Yoneda, T., Myers, C. and Nanya, T.: A Behavioral Synthesis System for Asynchronous Circuits with Bundled-data Implementation, *IPSJ Trans. SLDM*, Vol.2, pp.64–79 (2009).
12) Sait, S. M. and Youssef, H.: *VLSI Physical Design Automation: Theory and Practice*, McGraw-Hill, Inc. (1994).
13) Xilinx Inc., ISE Foundation 10.1, http://www.xilinx.com/.
14) Mentor Graphics Corp., ModelSim 6.5a, http://www.mentor.com/.