Vol.2011-SLDM-150 No.9
2011/5/19

束データ方式による非同期式回路の設計支援システムの構築

飯塚 成[†1] 濱田尚宏[†1] 齋藤寛[†1]
山口良一[†2] 吉永稔[†2]

本稿では，束データ方式による非同期式回路の ASIC 設計支援システムを提案する．提案システムは，束データ方式による非同期式回路のタイミング検証の自動化，設計制約コマンドの生成の自動化，遅延素子の生成，ならびに遅延素子の調整の自動化を行うツール群からなる．提案システムと既存の市販 CAD ツールを使用することで，束データ方式による非同期式回路の ASIC 設計が容易に行える．ケーススタディを通して，提案システムを利用して合成した回路の面積，性能，消費電力，消費エネルギーを評価する．

Development of an ASIC Design Support System for Asynchronous Circuits with Bundled-data Implementation

Minoru Iizuka,[†1] Naohiro Hamada,[†1] Hiroshi Saito,[†1]
Ryoichi Yamaguchi[†2] and Minoru Yoshinaga[†2]

This paper proposes an ASIC design support system for asynchronous circuits with bundled-data implementation. The proposed system consists of a tool set which includes automatic timing verification, design constraint command generation, delay element generation, and delay adjustment. Using the proposed system with commercial CAD tools, asynchronous circuits with bundled-data implementation can be designed easily. Through the case study, this paper evaluates synthesized circuits in terms of area, performance, power consumption, and energy consumption.

†1 会津大学
The University of Aizu
†2 ルネサス マイクロシステム（株）
Renesas Micro Systems Co., Ltd.

1. はじめに

集積回路の微細化技術の向上により，回路全体をクロック信号で制御する同期式回路は以下の問題が深刻となってくる．1 つは配線遅延に起因されるクロックスキューである．クロックスキューによってデータを書き込むタイミングがずれ，誤動作を起こす可能性がある．他には高周波のクロック信号を大規模な回路に供給する際に問題となるクロックツリーの消費電力と電磁放射の問題である．

一方，要求，応答信号によるローカルなハンドシェーク信号で回路部品を制御する非同期式回路は，クロック信号がないため，同期式回路ほど上述の問題は顕著とはならない．しかしながら，ローカルなハンドシェーク信号による制御を実現することは容易ではない．性能の良い回路を得るためには，演算毎の動作タイミングを意識して設計する必要がある．また，ハザードが起きた場合，それが伝播してしまうと誤動作を起こす可能性がある．それを防ぐためハザードフリーな回路，またはハザードが伝播しないような回路を設計しなければならない．こうした問題を解決するために，非同期式回路の合成に関してこれまでに様々な研究が行われてきた．

Amde らは，同期式 DLX マイクロプロセッサを自動で非同期化する方法を提案した 4)．非同期式回路の自動設計には，Pipefitter 5) を使用する．Sotiriou は，同期式回路の設計で一般的に使用されている CAD ツールを用いて非同期式回路を設計する手法を提案した 6)．この手法は，非同期式制御回路を one-hot で符号化した Asynchronous Finite State Machine (AFSM) でモデリングし，各状態に対して 1 つの制御モジュールを割り当てる．Cortadella らは，同期式回路のレジスタとその制御をラッチベースの 4 相ハンドシェークプロトコルに置き換えることで，非同期式回路を生成する手法 (Desynchronization と呼ぶ) を提案した 7)．提案手法を Data Encryption Standard (DES) や DLX に適用し，同期式回路と比較評価している．これらの手法は，同期式回路をスタートとし，同期式回路の設計で一般的に利用されている CAD ツールや一般的なライブラリを用いて非同期式回路を設計する．しかし，リソースの共有がないパイプラインモデルを対象としているので，一般的に動作合成ツールなどが生成するリソースの共有がある同期式回路を扱うことができない．また，これらの手法を実装した CAD ツールも利用できない．

本稿では，束データ方式による非同期式回路を対象に，ASIC を実現するための設計支援システムを提案する．提案システムは，制御モジュール非最適化制約生成ツール，タイミング解析に必要なコマンドの生成ツール，タイミングレポートの解析と XML 変換のツール，タイミング制約検証ツール，遅延素子生成ツール，最大遅延制約生成ツール，遅延調整ツールからなるこれらのツールと同等の機能を持った市販ツールはなく，上記の研究からこうしたツールを開発し公開したという例もない．

提案システムは，上記で述べた研究同様，同期式回路の設計で一般的に利用されている CAD ツールと組み合わせて利用することを想定している．上記の研究との違いは，リソース共有のあるモデルを扱うことにある．しかし逆に，パイプラインモデルは現段階では扱っておらず今後の課題である．本稿は，提案システムのツール群とある設計フローにおける利用例を解説し，ケーススタディとして提案システムを使って合成した回路の面積，性能，消費電力，消費エネルギーを評価をする．

本稿の構成は以下の通りである．2 節では束データ方式による非同期式回路の動作とタイミング制約について解説する．3 節では提案システム，4 節ではケーススタディについて解説する．最後に 5 節で結論を述べる．

Vol.2011-SLDM-150 No.9
2011/5/19

2. 束データ方式による非同期式回路

2.1 回路モデル

束データ方式とはデータエンコーディング方式の 1 つで，N ビットのデータを束と呼ばれる N+2 本の信号で表わす (+2 は要求, 応答信号)．データパス回路は同期式回路のものとほぼ同じものを用い，制御回路は要求，応答信号をベースとした非同期式制御回路である．データパスの最大遅延より大きい遅延を遅延素子として実現し，要求信号線上に付加することによって演算の完了，並びに制御タイミングを保証する．こうしたことより，遅延素子を含む制御回路の遅延が性能を決める．

図 1 に，本稿で対象とする束データ方式による非同期式回路の回路モデルを示す．右側はデータパス回路を表わし，左側は制御回路を表わす．データパス回路は，レジスタ (REG)，マルチプレクサ (MUX)，演算器 (FU)，グルーロジック ($glue$) により構成される．データパス回路のグルーロジックは，レジスタの書き込み信号，マルチプレクサの制御信号，演算器の制御信号を生成する．マルチプレクサの存在からリソースの共有がある．制御回路は幾つかの制御モジュール $CTRL_i(1 \leq i \leq m)$ から構成される．1 つの制御モジュール $CTRL_i$ は，Q モジュール Q_i2)，二つの遅延素子 (SD_i, $BD_{i,k}$)，グルーロジックから構成される．SD_i はデータパス回路の演算の完了を保証するために用いられ，$BD_{i,k}$ は $CTRL_i$ の後 l $(1 \leq k \leq l)$ 方向への分岐動作を保証するために用いられる．制御モジュールのグルーロジックは，C 素子 (C) や分岐判定論理 (図 1 では Q_3 と Q_4 の直前にある AND ゲート) から構成される．C 素子は記憶素子の一種で，**図 2** に 2 入力の C 素子の動作を示す．分岐判定論理は，制御モジュールからの信号と条件信号により分岐方向を決定する．

次に，制御動作について説明する．外部からの $start$ 信号が 1 になることで，動作を開始する．制御モジュール $CTRL_i$ の Q モジュール Q_i は，Q モジュール Q_{i-1} から出力された in_i の立ち上がり遷移によって動作を開始する．データパスのマルチプレクサや，演算器の制御信号は，この in_i よりグルーロジックを介して生成する．in_i が立ち上がると，Q_i は要求信号 req_i を立ち上げる．req_i の立ち上がり遷移は，遅延素子 SD_i を通り，応答信号 ack_i を立ち上げる．その後，Q_i は req_i を立ち下げる．req_i の立ち下がり遷移は，SD_i を通り，ack_i を立ち下げる．ack_i が立ち下がると，Q_i は次の Q モジュール Q_{i+1} への入力信号となる out_i を立ち上げる．レジスタへの書き込み信号は，SD_i の出力信号 (ack_i) の立ち下がり遷移を用いる．

以後の説明では，一つの $CTRL_i$ で処理が終わるデータパスをシングルサイクル，複数を要する場合をマルチサイクルと呼ぶ．

2.2 タイミング制約

前節で解説した回路モデルは，以下の 3 種類のタイミング制約を満たす必要がある．

2.2.1 セットアップ制約

書き込み先のレジスタに書き込み信号が到着するより一定の時間 (セットアップ時間) 前に，レジスタの入力データが安定していなければならない．これをセットアップ制約と呼ぶ．セットアップ制約は制御モジュール $CTRL_i$ 毎に考える．**図 3** を例に解説する．ここでは，SD_1 の出力の立ち下がり遷移から，データパス (太線) を通り，書き込み先のレジスタ REG_3 のデータ入力ピンまでの最大遅延を T_{maxsdp} とする．これに対し，SD_1 の出力の立ち下がり遷移から，次の Q モジュール Q_2 の遅延素子 SD_2 の出力の立ち下がり遷移を通過し，REG_3 のクロックピンまでのパス (破線) の最小遅延を T_{minscp} とする．T_{setup} をレジスタのセットアップタイム，$R_{sm}(R_{sm} \geq 1)$ を T_{maxsdp} に対するマージンとすると，セットアップ制約は以下の不等式で表すことができる．

$$T_{minscp} > T_{maxsdp} * R_{sm} + T_{setup} \quad (1)$$

2.2.2 ホールド制約

レジスタにデータが書き込まれた後，一定の時間 (ホールド時間) レジスタの入力は安定していなければならない．これをホールド制約と呼ぶ．ホールド制約はレジスタ毎に考える．**図 4** を例に解説する．ここでは，SD_4 の出力の立ち下がり遷移により REG_3 にデータを書き込んでいる．SD_4 の出力の立ち下がり遷移から REG_3 のクロックピンまで (太線) の最大遅延を T_{maxhcp}，SD_4 の出力の立ち下がり遷移から REG_3 のデータ入力ピンまで (破線) の最小遅延を T_{minhdp} とする．T_{hold} をホールドタイム，$R_{hm}(R_{hm} \geq 1)$ を T_{maxhcp} に対するマージンとすると，ホールド制約は以下の不等式で表すことができる．

$$T_{minhdp} > T_{maxhcp} * R_{hm} + T_{hold} \quad (2)$$

2.2.3 分岐制約

分岐制約は分岐に関するタイミング制約で，分岐毎に考える．**図 5** を例に解説する．ここでは，Q_2 から Q_3 か Q_4 のどちらかに分岐する．分岐の条件判定には，SD_2 の出力の立ち下がり遷移により書き

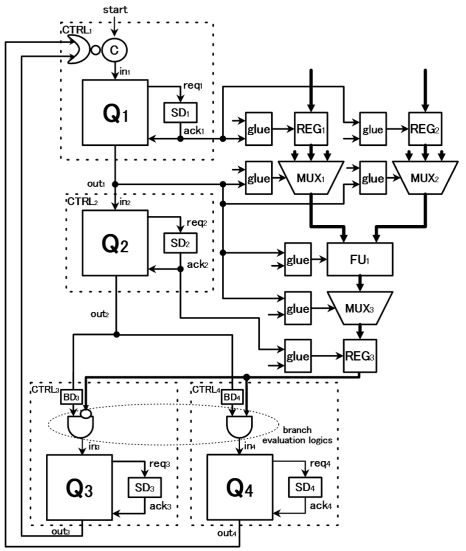
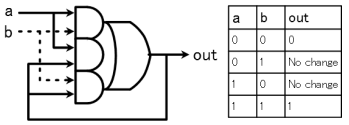


図 1 束データ方式による非同期式回路モデル.
Fig. 1 A circuit model of asynchronous circuits with bundled-data implementation.



a	b	out
0	0	0
0	1	No change
1	0	No change
1	1	1

図 2 C 素子.
Fig. 2 C-element.

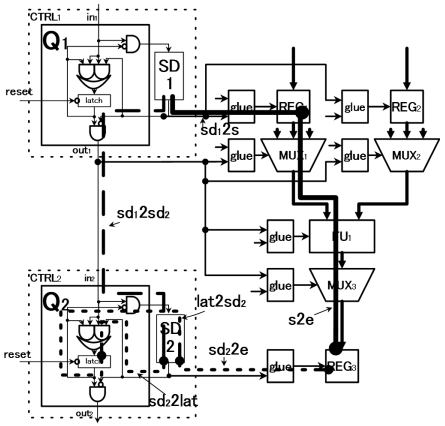


図 3 セットアップ制約に関するパスの例.
Fig. 3 Paths related to a setup constraint.

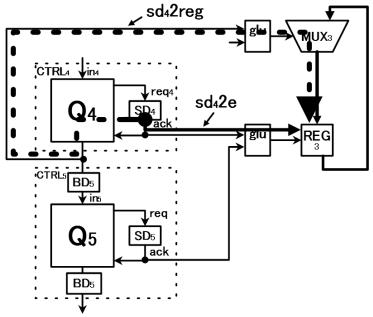


図 4 ホールド制約に関するパス.
Fig. 4 Paths related to a hold constraint.

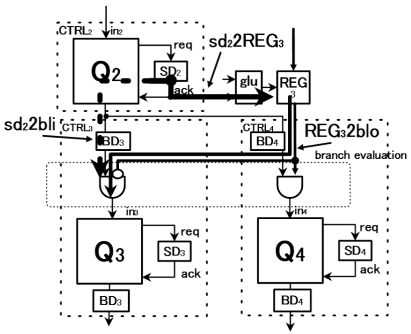


図 5 分岐制約に関するパス.
Fig. 5 Paths related to a branch constraint.

込まれるレジスタ REG_3 の出力信号を使用している．このとき，SD_2 の出力の立ち下がり遷移から，REG_3 を通り $CTRL_3$ にある分岐判定論理の出力ピンまで (太線) の最大遅延を T_{maxbdp} とする．一方，SD_2 の出力の立ち下がり遷移から，分岐判定論理の入力ピンまで (破線) の最小遅延を T_{minbcp} とする．$R_{bm}(R_{bm} \geq 1)$ を T_{maxbdp} に対するマージンとすると，分岐制約は以下の不等式で表すことができる．

$$T_{minsd2bli} > T_{maxsd2blo} * R_{bm} \quad (3)$$

3. 提案システム

3.1 概　　要

提案システムは，束データ方式による非同期式回路の ASIC 設計を支援するツール群からなる．これらのツールは，制御モジュール非最適化制約生成ツール (CTRCONSTGEN)，タイミング解析に必要なコマンドの生成ツール (REPTIMING)，タイミングレポートの解析と XML 変換のツール (STA2XML)，タイミング制約検証ツール (TIMINGCHECKER)，遅延素子のネットリスト生成ツール (DELAYGEN)，最大遅延制約生成ツール (MAXCONSTGEN)，遅延調整ツール (DELAYADJUST) からなる．こうしたツールを利用しない部分は，同期式回路で利用される市販ツールを用いる．なお，同等の機能を持った市販ツールはなく，またこれまでの非同期式回路の合成に関する研究からこうしたツールを開発し公開したという例もない．

提案システムが含むツール群と市販ツールを組み合わせた設計フロー例を**図 6** に示す．フローの入力は，同期式回路の RTL モデル，テクノロジライブラリ，レイテンシ制約であり，出力は 2 で示した束データ方式による非同期式回路のレイアウト設計である．束データ方式による非同期式回路は同期式回路とほぼ同じデータパス回路を使うということから，同期式回路との明確な違いは制御回路のみである．同期式回路を非同期式回路に置き換える作業を Desynchronization 7) と呼ぶが，このフローでも Desynchronization を行う．なお，本稿は 7) とは扱う回路モデルが違う (リソースの共有がある)．提案システムを用いることによって束データ方式による非同期式回路の設計のほとんどが自動化される．

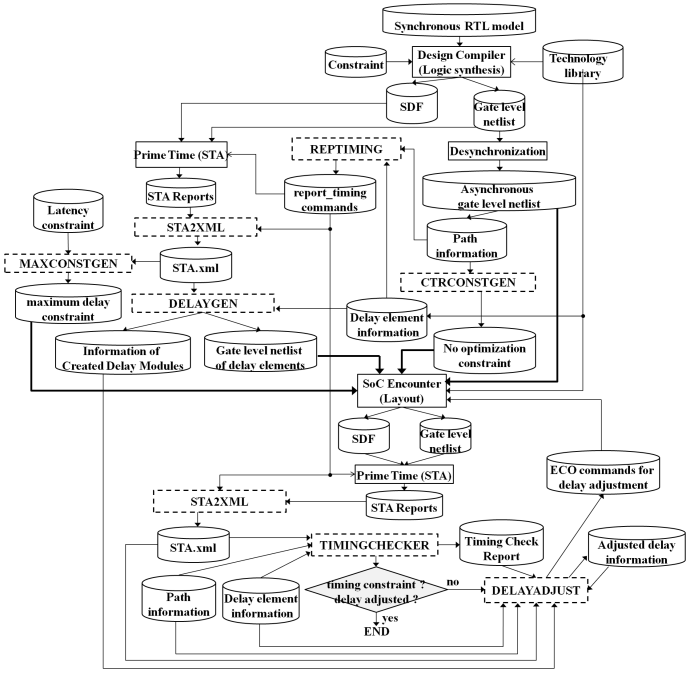


図 6 設計フローの例.
Fig. 6 Example of design flow.

また，違うフローで提案システムに含まれるツール群を使うことも可能である．本節の以下では，図 6 で示したフローを基に，提案システムが含むツール群の解説を行う．

3.2 準　　備

フローの入力は，同期式回路の RTL モデル，テクノロジライブラリ，レイテンシ制約である．モデルのデータパス回路は，図 1 で示したように REG，MUX，FU から構成されていると仮定する．制御回路に関しては，Finite State Machine(FSM) によって制御されているものと仮定する．始めに，目標クロックサイクルタイムを与えて論理合成を行い，ゲートレベルのネットリストと Standard Delay Format(SDF) ファイルを生成する．論理合成には Synopsys 社の Design Compiler を用いる．

論理合成後のネットリストに対して，非同期化 (Desynchronization) を行う．まず，FSM の各状態に対して，**図 7** に示した Q モジュール Q_i を割り当てる．Q モジュール内の C 素子の出力にはラッチがおかれており，グローバルリセット信号の否定でオープンとなる．ラッチを置く理由は，組み合わせループの回避のためである．次に，必要に応じてレジスタを置き換える．レジスタは，入力ピン，出力ピン，クロックピン，リセットピンからなるフリップフロップとする．したがって，論理合成の後，

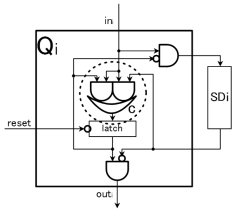


図 7 ラッチ付き Q モジュール.
Fig. 7 Q-module with a latch.

表 1 パス情報ファイルの一部.
Table 1 A part of path information file.

Start	Through	End	Source CTRL	Destination CTRL
FF0	ADD0	FF2	CTRL0	CTRL1
FF1	ADD0	FF2	CTRL0	CTRL1

表 2 遅延素子用セルファイルの一部.
Table 2 A part of resource information file.

Gate	Delay	input pin	output pin
delay1	0.10	I	O
delay2	0.20	I	O
delay3	0.30	I	O

レジスタイネーブル信号やクロックゲーティングに対するゲーティング回路が含まれるレジスタに対して置き換えを実施する．最後に，データパスリソースや分岐に対してグルーロジックを挿入する．なお，この段階で遅延素子 SD_i は，入力信号を出力信号に代入する代入文からなるダミーモジュールである．遅延素子 $BD_{i,k}$ は，分岐制約違反があるときにだけ必要となるので，この段階では生成しない．

次に，タイミング検証が必要な全てのパスをパス情報ファイルとして登録する．登録すべき情報は，**表 1** に示す通り，パスの開始点，中継点，終了点，ソースレジスタ，ディスティネーションレジスタを制御する制御モジュール，パスに対する最大遅延制約である．また，テクノロジライブラリから遅延素子で使うセルを決定し，遅延素子用セルファイルとして登録する．登録すべき情報は，**表 2** に示す通り，セル名，遅延値，入出力ピン名である．

以上が提案システムを使うまでの準備項目である．このうち，非同期化，パス情報ファイル，遅延素子用セルファイル生成の自動化は今後の課題である．

3.3 CTRCONSTGEN

束データ方式の非同期式回路を合成 (このフローではレイアウト合成) するとき，制御モジュールが最適化されないように制約を設ける必要がある．さもないと，タイミング違反，あるいはハザードが生ずる可能性がある．CTRCONSTGEN は，各制御モジュール $CTRL_i$ に対して Synopsys Design Constraint(SDC) にある非最適化制約 *set_dont_touch* を生成し，SDC ファイルとして出力する．

3.4 静的タイミング解析

3.4.1 REPTIMING

REPTIMING は，パス情報ファイルからタイミング解析コマンド *report_timing* を生成し，SDC ファイルとして出力する．

セットアップ制約の場合，データパスは図 3 のように，ソースレジスタ (REG_1) へ書き込みをする $CTRL_1$ の遅延素子 SD_1 の出力からソースレジスタのクロックピンまでのパス $sd_1 2s$ とソースレジスタの出力データピンからディスティネーションレジスタ (REG_3) の入力データピンまでのパス $s2e$ の二つに分けられ，それらの最大遅延 $T_{maxsd_1 2s}$ と T_{maxs2e} を解析する *report_timing* コマンドを生成する．制御パスは図 3 のように，ソースレジスタへ書き込みをする $CTRL_1$ の遅延素子 SD_1 の出力から，ディスティネーションレジスタへ書き込みをする $CTRL_2$ の遅延素子 SD_2 の出力までのパス $sd_1 2sd_2$, SD_2 の出力から Q モジュール Q_2 のラッチの入力データピンまでのパス $sd_2 2lat$, Q_2 のラッチの出力データピンから SD_2 の出力までのパス $lat2sd_2$, SD_2 の出力からディスティネーションレジスタのクロックピンまでのパス $sd_2 2e$ の四つに分けられ，それらの最小遅延 $T_{minsd_1 2sd_2}$, $T_{minsd_2 2lat}$, $T_{minlat2sd_2}$, $T_{minsd_2 2e}$ を解析する *report_timing* コマンドを生成する．**図 8** は $T_{maxsd2s}$ に対するコマンドを表す．

ホールド制約の場合，図 4 のように遅延素子 SD_4 の出力からレジスタ REG_3 の入力データピンまでの最小遅延 $T_{minsd_4 2reg}$ を解析する *report_timing* コマンドを生成する．遅延素子 SD_4 の出力からレジスタ REG のクロックピンまでの最大遅延 $T_{maxsd_4 2e}$ を解析する *report_timing* コマンドは，セットアップ制約を解析するコマンドの生成の際に生成される．

分岐制約の場合，データパスは図 5 のように，条件信号を保持するレジスタ (REG_3) に書き込む $CTRL_2$ の遅延素子 SD_2 の出力から条件信号を保持するレジスタのクロックピンまでのパス $sd_2 2REG_3$ と条件信号を保持するレジスタの出力データピンから分岐判定論理の出力までのパス $REG_3 2blo$ の二つに分け，そのうちの最大遅延 $T_{maxREG_3 2blo}$ を解析する *report_timing* コマンドを生成する．$sd_2 2REG_3$ の最大遅延 $T_{maxsd_2 2e}$ を解析する *report_timing* コマンドは，セットアップ制約を解析するコマンドの生成の際に生成される．制御パスは SD_2 の出力から分岐判定論理の入力までのパスの最小遅延 $T_{minsd_2 2bli}$ を解析する *report_timing* コマンドを生成する．

3.4.2 静的タイミング解析

論理合成，もしくは，レイアウト合成後に生成されたネットリスト，SDF ファイル，REPTIMING にて生成した *report_timing* コマンドを Synopsys 社の Prime Time に渡し静的タイミング解析（STA）を行う．STA の結果はテキスト形式で保存する．

3.4.3 STA2XML

STA 結果を保存したテキストファイルには，必要な遅延値以外の部分も含まれてしまうため，STA2XML を用いて必要な情報のみを自動で抽出し XML 形式に変換する．

3.5 DELAYGEN

パス情報ファイルに指定した各データパスの最大遅延制約，もしくはその値が設定されていない場合，非同期化後の STA 結果より各制御モジュール $CTRL_i$ の SD_i に必要な遅延値を計算する．レジスタにデータを書き込むまで SD_i を 2 回通ることを考慮して，SD_i の遅延値には $CTRL_i$ で処理が終わるシングルサイクルのデータパスの最大遅延 T_{maxs2e_i} の半分を使用する．この遅延値をみたすように，遅延素子用セルファイルに指定した遅延素子用セルを用いて遅延素子を生成し，Verilog によるネットリストを出力する．出力される SD_i の Verilog ネットリストの例を**表 3** に示す．マルチサイクルのデータパスは，レイアウト合成後に遅延素子を調整することで対応する．

分岐制約のための遅延モジュール $BD_{i,k}$ はここでは生成しない．レイアウト合成後，分岐制約違反が無ければ，$BD_{i,k}$ は必要ないからである．もし，分岐制約が起きる場合は，Engineering Change Order (ECO) で挿入する．

3.6 レイアウト合成

3.6.1 MAXCONSTGEN

現段階で提案システムは非パイプラインの回路合成のみを支援する．性能のよい回路，あるいはレイテンシ制約を満たす回路を合成するように，MAXCONSTGEN は各パスに対して最大遅延制約

```
redirect TimingReport/Tmaxsd2s { report_timing ¥
        -fall_from { SD_i/out  } ¥
        -to        { REG_1/CLK } ¥
        -delay_type max }
```

図 8 $Tmaxsd2s$ に対する STA コマンド.
Fig. 8 STA command for $Tmaxsd2s$.

```
set_max_delay 5 ¥
        -from    { sd_l/out   } ¥
        -to      { sd_2/out   }
```

図 9 最大遅延制約の例.
Fig. 9 An example of max delay constraint.

表 3 SD_i のネットリストの例.
Table 3 Example of SD_i netlist.

```
module SD_i (in, out);
    input  in;
    output out;
    wire w0, w1, w2, w3, w4, w5,
      w6, w7, w8, w9, w10, w11,
      w12, w13, w14, w15, w16, w17;
    delay2 d0(.I(in), .O(w0));
    delay3 d1(.I(w0), .O(w1));
    delay3 d2(.I(w1), .O(w2));
            :
    delay3 d15(.I(w14), .O(w15));
    delay3 d16(.I(w15), .O(out));
endmodule
```

set_max_delay を生成する．なおレイテンシは，制御が一周するのに要する時間と仮定する．

束データ方式による非同期式回路では，タイミング制約から制御パスの最小遅延がデータパスの最大遅延より大きくなければならない．レイテンシ制約に対し，ばらつきなどによる遅延を差し引いた制御回路の最大遅延の割合を制御割合 CR とし，制御回路の最大遅延に対し，データパス回路の最大遅延の割合をデータパス割合 DR とする．これらのパラメータは，入力として与える．

想定している設計フローでは，パス情報ファイルに指定したデータパスの最大遅延制約，もしくは指定していない場合，非同期化後の STA 結果より最大遅延制約コマンドを SDC ファイルとして生成する．MAXCONSTGEN は，始めに各制御モジュール $CTRL_i$ で制御されるデータパス遅延の最大値 T_{maxs2e_i} を求める．これらの値は，遅延素子の生成時に使用した値と同じである．制御が一周するときの T_{maxs2e_i} の和より各制御モジュール $CTRL_i$ における T_{maxs2e_i} の割合 R_i $(0 < R_i < 1)$ を以下の式で算出する．

$$R_i = \frac{T_{maxs2e_i}}{\sum_{j=k}^{l} T_{maxs2e_j}} \tag{4}$$

MAXCONSTGEN は次に，制御パスに対する制約生成を行う．STA では $CTRL_i$ における制御パスを四つに分けたので (図 3 では $sd_1 2sd_2$, $sd_2 2lat$, $lat2sd_2$, $sd_2 2e$)，それぞれのパスに対して最大遅延制約を生成する．STA の結果より，4 つのパスの最小遅延の和を求め，各パスの最小遅延をその和で割ることにより，重み $\alpha, \beta, \gamma, \delta$ $(\alpha + \beta + \gamma + \delta \leq 1)$ を求める．なお，非同期化後では遅延素子はただのワイヤなので，SD_i を含むパス (図 3 では $sd_1 2sd_2$ と $lat2sd_2$) の最大遅延は，対応するデータパスの論理合成後の最大遅延の半分とする．それぞれのパスの最大遅延制約の値は，レイテンシ制約 $\times$ CR $\times$ R_i の値にそれぞれの割合 $\alpha, \beta, \gamma, \delta$ をかけたものである．**図 9** は生成される制約コマンドの例を示す．

MAXCONSTGEN は最後に，データパスの最大遅延制約を生成する．各データパスの最大遅延制約は，レイテンシ制約 $\times$ CR $\times$ R_i $\times$ DR である．なお，表 1 で指定したデータパスの最大遅延制約をそのまま使うのではなく，レイテンシ制約に対する重みを計算して制約を生成している．そのため，実際には指定した値より小さい値が設定される．

3.6.2 レイアウト合成の実行

想定しているフローでは，Cadence 社の SoC Encounter に遅延素子 SD_i を含んだ非同期化後のネットリストと *set_max_delay* コマンドを与えてレイアウト合成を行う．

3.7 TIMINGCHECKER

レイアウト合成後，Prime Time に Verilog ネットリスト，SDF ファイル，*report_timing* コマンドを与え STA を行い，STA2XML を用いて STA 結果からタイミング検証に必要な情報を抜き出した XML ファイルを生成する．

TIMINGCHECKER は，この XML ファイルとパス情報ファイルを用いて，全てのタイミング制約を検証する．具体的には，2 節で示した各タイミング制約式に値を代入し，制約式の差 (左辺 − 右辺) を計算することで検証する．差が負の場合は制約違反，正の場合は制約を満たしているということになる．

3.8 DELAYADJUST

タイミング検証で制約違反がある場合，あるいはセットアップ制約は全て満たしているけれども制約式の左辺の値が極端に大きい場合は DELAYADJUST，遅延素子用セルファイルを用いて遅延調整を行う．遅延調整は遅延素子を構成するバッファ数を増減することによって行う．そのために DELAYADJUST は ECO コマンドを生成する．

タイミング制約違反がある場合，DELAYADJUST はホールド制約，分岐制約，セットアップ制約の順番で遅延を調整していく．ホールド制約違反に対する遅延調整では，ホールド制約違反のあるレジスタのデータ入力ピンの直前に，ホールド制約の差の絶対値の遅延だけバッファを挿入する．1 つのレジスタに対して複数のホールド違反がある場合，差のうち最大となるものの分だけバッファを挿入する．レジスタの直前にバッファを挿入するとデータパスの遅延が伸びるため，セットアップ制約のデータパスに影響を与える．このため，セットアップ制約違反の調整より先に行う．分岐制約違反に対する遅延調整では，分岐判定論理の直前で制御モジュールからの信号線に分岐制約の差の絶対値の分だけバッファを挿入する．これは遅延素子 $BD_{i,k}$ に対応する．

最後にセットアップ制約違反に対する遅延調整をおこなう．これは遅延素子 SD_i のセル数を調整することに相当する．$CTRL_i$ に関するすべてのセットアップ制約の中に違反が有る場合，違反した制約式の差のうち最大となるものの半分に相当するバッファを SD_i に挿入する．一方，$CTRL_i$ に関するすべてのセットアップ制約の中に違反が無い場合，制約式の差の最小値の半分に相当するバッファを SD_i から削除する．SD_i 調整のためのコマンドの例を**図 10** に示す．

4. ケーススタディ

ケーススタディとして，提案システムを用いて Elliptic Wave Filter (EWF) を合成し，面積，性能，消費電力，消費エネルギーを評価する．なお，比較対象として同期式回路の EWF も合成する．提案システムに含まれるツール群は Java を用いて実装した．

```
# add a delay element delay1
# to CTRL1/SD1 as the instance name d10
addInst -cell delay1 -inst CTRL1/SD1/d10
addNet CTRL1/SD1/w9
attachTerm CTRL1/SD1/d9  O CTRL1/SD1/w9
attachTerm CTRL1/SD1/d10 I CTRL1/SD1/w9
attachTerm CTRL1/SD1/d10 O CTRL1/SD1/out
# delete the instance d5
ecoDeleteRepeater -inst d5
```

図 10 セットアップ制約に対する ECO コマンドの例.
Fig. 10 An example of ECO command for a setup constraint.

始めに，EWF の RTL モデル，テクノロジライブラリ，レイテンシ制約を設定する．EWF は加算と乗算からなるフィルタで，このケーススタディはどちらの演算も 1 クロックサイクルとし，as soon as possible スケジュールの RTL モデルを生成する．制御 FSM は 14 状態である．テクノロジライブラリは Rohm 社の 0.18um テクノロジを使用する．レイテンシ制約は同期式回路において最速となるレイアウトより決める．このときのクロックサイクルタイムは 3.8 ns で，14 状態が直列に実行されることよりレイテンシ制約を 53 (3.8*14=53.2≈53) ns とする．また，同期式回路の最速なものより速い非同期式回路を実現するために，レイテンシ制約を 50 ns とした場合も試す．これは同期式回路の場合全てのサイクルが最悪遅延から決められたクロックサイクルで動作する反面，束データ方式による非同期式回路の場合サイクル毎の最悪遅延で各遅延素子を決めるため，レイテンシにおいて優位な回路を実現できる可能性があるからである．

次に，3 節で示した設計フローを用いて束データ方式による非同期式回路を設計する．論理合成にはクロックサイクルタイムを指定するが，ここでは 3.0 ns として合成する．合成された回路を非同期化し，パス情報ファイル，遅延素子用セルファイルを準備する．データパスの最大遅延制約は，同期式回路の論理合成後の STA 結果を基に設定する．加算を通るパスに対して，乗算を通るパスの遅延はその倍とする．MAXCONSTGEN の生成におけるパラメータ CR と DR はともに 0.9 とする．最終的に得られたレイアウト設計に対して，Synopsys 社の VCS を用いてシミュレーションを行い，動作確認 (機能検証) を行った．

表 4 は，合成された回路の面積，性能，消費電力，消費エネルギーを表わす．面積は Cadence 社の SoC Encounter が生成した値，性能は任意のテスト入力を与えた時のシミュレーション時間，消費電力はシミュレーションから Switching Activity Interchange Format(SAIF) ファイルを生成し，Synopsys 社の Prime Time にて出力された値，消費エネルギーはシミュレーション時間*消費電力である．

表 4 より，レイテンシ制約を満たす 2 つの非同期式回路が生成されたことがわかる．前者は同期式回路と同等の性能を目標としたが，実際はずいぶんと速い回路となった．そのため，消費電力がほぼ同等となってしまっている．これは，最大遅延制約の値が指定した値より小さくなったからである．なお，同期式回路はクロックゲーティングを行っている．レイテンシ制約をもう少し緩めて回路合成をすれば，性能が同期式回路と同等となり消費電力が同期式回路よりもよいものが期待できる．後者は，期待通り同期式回路よりも速い回路となった．この結果は，束データ方式による非同期式回路のレイテンシに対する優位性を表わしている．一方，これらの回路と同期式回路の面積を比較すると，面積は若干良くなっている．これは同期式回路にはゲーティング回路が挿入されていることと，クロックツリー合成によるバッファの挿入に起因する．

表 4 実験結果.
Table 4 Result of case study.

	面積 [μm^2]	性能 [ns]	消費電力 [W]	消費エネルギー [nJ]
同期式	329,163	53.2	0.0297	1.58
非同期式 (レイテンシ制約 53 ns)	281,396	43.1	0.0321	1.38
非同期式 (レイテンシ制約 50 ns)	296,856	42.3	0.0347	1.47

5. 結　　論

本稿では，束データ方式による非同期式回路の ASIC 設計支援システムを提案した．提案システムは，非同期式回路設計において市販 CAD ツールが支援しない部分をツール化したツール群からなる．提案システムと市販 CAD ツールを組み合わせることによって，束データ方式による非同期式回路の ASIC 設計が容易になる．また，ケーススタディを通じて最速な同期式回路より速い回路を設計した．

今後の課題は，パイプライン回路への対応，フロアプランを含んだ最適化，非同期化やパス情報ファイルの自動化があげられる．

謝辞 本研究は東京大学大規模集積システム設計教育研究センターを通し，シノプシス株式会社，日本ケイデンス株式会社，ローム株式会社の協力で行われたものである．

参 考 文 献

1) N.Hamada, T.Konishi, H.Saito, T.Yoneda, and T.Nanya, "Proposal of a behavioral synthesis method for asynchronous circuits in bundled-data implementation", *IEICE technical report (VLD2006-63)*, vol.106, no.387, pp.71-76, Nov., 2006.
2) F.U.Rosenberger, C.E.Molnar, T.J.Chaney, and TP.Fang, "Q-Modules: Internally Clocked Delay Insensitive Modules", *IEEE Transaction of Computer*, vol.C-37, no.9, pp.1005-1018, 1988.
3) A. Kondratyev and K. Lwin, "Design of Asynchronous Circuits using Synchronous CAD Tools", *IEEE Design and Test of Computer, pp.107-117, July/Aug, 2002* .
4) M. Amde, I. Blunno, C. P. Sotiriou, "Automating the Design of an Asynchronous DLX Microprocessor", *Design Automation Conference, pp.502-507, June 2-6, 2003.*
5) I. Blunno and L. Lavagno, "Automated synthesis of micro-pipelines from behavioral Verilog HDL". *In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, pages 84-92, IEEE Computer Society Press, Apr. 2000.*
6) C. P. Sotiriou, "Implementing Asynchronous Circuits using a Conventional EDA Tool-Flow", *Design Automation Conference, pp.415-418 June 10-14, 2002.*
7) J. Cortadella, A. Kondratyev, L. Lavagno, and C. P. Sotiriou, "Desynchronization: Synthesis of Asynchronous Circuits From Synchronous Specifications", *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems. vol.25, no.10, pp. 1904-1921 Oct., 2006.*