Vol.2016-SLDM-176 No.4
2016/5/11

# 東データ方式による非同期式回路の遅延調整に関する考察

吉見　宗真[1,a)]　齋藤　寛[1,b)]

**概要**：本稿では，束データ方式による非同期式回路の遅延調整法を考察する．束データ方式による非同期式回路では，タイミング制約を満たすための遅延調整が回路面積，性能，設計時間に影響を与える．そのため本稿では，タイミング制約毎の遅延挿入位置，調整順番を考察する．また，タイミング違反を減らす試みとして，パスに最大・最小遅延制約を与え回路を合成する．　実験では，複数のベンチマーク回路に考察した遅延調整法を適用し，調整回数，回路面積，および消費エネルギーを評価する．

**キーワード**：非同期式回路，遅延調整，タイミング検証

# Consideration of Delay Adjustment for Asynchronous Circuits with Bundled-data Implementation

Kazumasa Yoshimi[1,a)]　Hiroshi Saito[1,b)]

***Abstract:*** In this paper, we consider a delay adjustment method for asynchronous circuits with bundled-data implementation. In bundled-data implementation, delay adjustment to satisfy timing constraints affects circuit area, performance, and design time. Therefore, in this paper, we consider position to insert delay elements and sequence to adjust delay elements. In addition, we synthesize circuits with maximum and minimum delay constraints in order to reduce timing violations. In the experiments, we evaluate then number of delay adjustments, circuit area, and energy consumption for benchmark circuits applying considered delay adjustment method.

***Keywords:*** Asynchronous circuits, delay adjustment, timing verification

## 1. はじめに

半導体微細化技術の向上により，VLSI はその性能を飛躍的に向上させてきた．こうした VLSI のほとんどは，グローバルクロック信号にて回路部品を制御する同期式回路である．しかし，微細化技術が進化するにつれ，同期式回路ではクロックツリーによる消費電力の増加やクロックスキューによる同期の失敗が顕著となる．

非同期式回路は，要求・応答信号からなるハンドシェーク信号により回路部品を制御するため，クロック信号に関する問題は発生しない．しかしながら，非同期式回路の設計は，同期式回路と比べ困難である．アプリケーションに応じて，適切な遅延モデルやデータエンコーディング法を選択する必要がある．また，選択に応じて設計手法やタイミング検証が異なる．利用できる設計自動化ツールも，合成や検証の一部分と限定的である．

1　会津大学
University of Aizu, Aizu-Wakamatsu, Fukushima 965–8580, Japan
a)　m5201119@u-aizu.ac.jp
b)　hiroshis@u-aizu.ac.jp

束データ方式による非同期式回路は，非同期式回路の 1 つであり，N ビットのデータを N+2 本の信号で表す．余分な 2 本の信号は，要求・応答信号である．データパス回路は，同期式回路と同じものを用いる事が出来る．演算の完了やレジスタへの書き込みは，要求信号線上に付加した遅延素子によって保証する．そのため，回路の性能は，遅延素子を含む制御回路の遅延に依存する．

束データ方式による非同期式回路では，タイミング制約を満足するために遅延調整が必要となる．遅延調整は回路面積，性能，および設計時間に影響を与えるので，不適切な調整では，設計に時間がかかり，かつ期待通りの回路面積や性能が得られない．そのため，遅延調整を効率よく行うことが求められる．

本稿では，束データ方式による非同期式回路の遅延調整法の考察を行う．まず，タイミング制約毎に遅延挿入位置を定義する．次に，タイミング制約式より，調整の順番を考察する．また，タイミング違反を減らすために，パスに最大遅延制約だけでなく，最小遅延制約を与え合成する．

本研究は，[1] をベースとしている．しかし，[1] と比べ，ホールド制約に関する遅延素子の挿入位置，遅延素子の調整の順番が異なる．また，[1] では，回路合成に最大遅延制

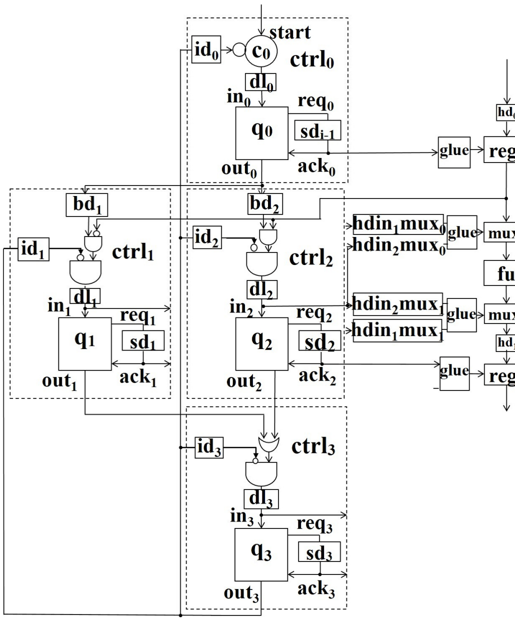



図 1: 束データ方式による非同期式回路モデル.

約のみしか用いていない．Chakraborty と Dill は [2] で，非同期式 finite state machine の最大・最小タイミング解析手法を提案した．しかし，本稿で述べるような遅延素子の調整手法や最大・最小遅延制約を用いた回路合成は言及していない．

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について解説する．3 節では提案手法，4 節では提案手法を含んだ設計フロー，5 節では実験と結果，6 節では結論を述べる．

## 2. 束データ方式による非同期式回路

### 2.1 回路モデル

図 1 は，本研究で使用する束データ方式による非同期式回路の回路モデルを表す．この回路は，制御回路とデータパス回路で構成される．制御回路は，複数の制御モジュール $ctrl_i$ より構成される．1 つの制御モジュールは，1 つの状態を制御する．$ctrl_i$ は C 素子 [3] か AND ゲート，Q 素子 $q_i$ [4]，遅延素子 $sd_i$，$bd_i$，$id_i$，ラッチ $dl_i$ より構成される．データパス回路は，レジスタ（$reg_k$），マルチプレクサ（$mux$），関数ユニット（$fu$），遅延素子（$hd_k$, $hd_{in_i mux}$）やグルーロジックにより構成される．グルーロジックは，$in_i$ 信号を受け取りマルチプレクサの制御信号を生成するものと，$ack_i$ 信号よりレジスタの制御信号を生成するものがある．

遅延素子 $sd_i$ は，セットアップ制約を満たすため，$bd_i$ は分岐制約を満たすため，$id_i$ は休止相制約を満たすため，$hd_{in_i mux}$，$hd_k$ はホールド制約を満たすために用いられる．$bd_i$ 以外の遅延素子はバッファーを直列につなぐことで実現し，$bd_i$ は 2 入力の AND ゲートを直列につなぐことで実現する．

次に, この回路モデルの動作を解説する．ここでは，信号 $signal$ が 0 から 1 へ変化する場合は $signal+$，信号が 1 から 0 へと変化する場合は $signal-$ と表記する．最初の動作は，制御回路の $start$ に $strat+$ が到着したところから始まる．制御モジュール $ctrl_i$ は，前の $ctrl_{i-1}$ からの出力信号 $out_{i-1}+$ が $ctrl_i$ に到着した時にデータパス回路の制御を開始する．ラッチ $dl_i$ の出力は $in_i$ となり，グルーロジックを通りマルチプレクサを制御する．$q_i$ に $in_i+$ が到着すると $req_i+$ となる．$req_i+$ は，$sd_i$ を通り $ack_i+$ となり $q_i$ に戻る．次に，$q_i$ は $req_i-$ を生成する．その後、再び $sd_i$ を通り $ack_i-$ となり $q_i$ に戻る．$ack_i-$ はグルーロジックを通りレジスタを制御する．$ack_i-$ の後，$q_i$ は $out_i+$ を生成し，次の制御モジュール $ctrl_{i+1}$ に制御を移す．最後の制御モジュール $ctrl_n$ （図 1 の場合，$ctrl_3$）が $out_n+$ を生成した後，全ての制御モジュールは $in_i-$ と $out_i-$ を生成し、制御モジュールを初期化する．

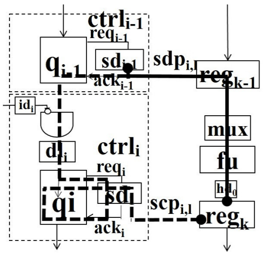



図 2: セットアップ制約に関するデータパス $sdp_{i,l}$ と制御パス $scp_{i,l}$.

### 2.2 タイミング制約

本稿で使用する束データ方式による非同期式回路は，以下の 4 つ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

デスティネーションレジスタにデータが書き込まれるより一定の時間（セットアップ時間）前に，書き込まれるデータは安定しなければならない．これをセットアップ制約と呼ぶ．

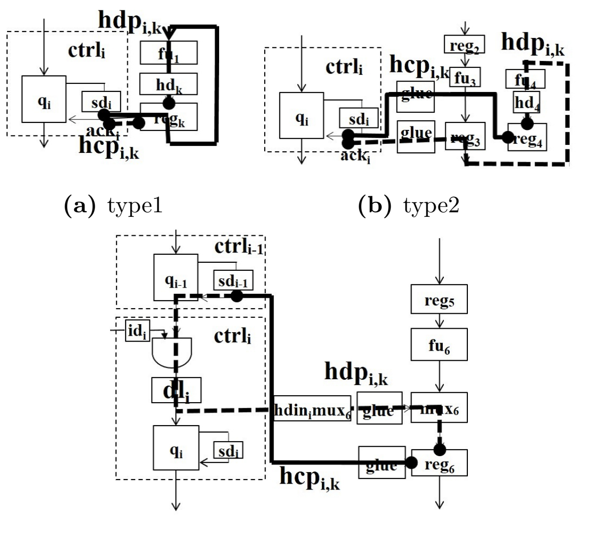
図 2 を用いてセットアップ制約を解説する．$ctrl_{i-1}$ の $sd_{i-1}$ の出力からソースレジスタを通り $ctrl_i$ によって制御されるデスティネーションレジスタまでのデータパスを $sdp_{i,l}$ とする．一方，$ack_{i-1}$ から $dl_i$ を経由しデスティネーションレジスタまでの制御パスを $scp_{i,l}$ とする．$t_{maxsdp_{i,l}}$ を $sdp_{i,l}$ における最大遅延，$t_{minscp_{i,l}}$ を $scp_{i,l}$ における最小遅延，$t_{setup_{i,l}}$ をセットアップ時間，$t_{maxsdp_{i,l}}$ に対するマージンを $sm_{i,l}$ ($sm_{i,l} > 0$) 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．

$$t_{minscp_{i,l}} > t_{maxsdp_{i,l}} + t_{setup_{i,l}} + sm_{i,l} \qquad (1)$$

#### 2.2.2 ホールド制約

デスティネーションレジスタに入力信号が書き込まれてから一定の時間（ホールド時間），入力信号の値は変化してはならない．これをホールド制約と呼ぶ．

図 3 を用いてホールド制約を説明する．$sd_i$ の出力から $reg_k$ の入力ピンまでのデータパスを $hdp_{i,k}$，$sd_i$ の出力から $reg_k$ のクロックピンまでの制御パスを $hcp_{i,k}$ とする．$hdp_{i,k}$ は，3 種類に分類することができる．ディスティネーションレジスタの出力が入力に至る type1 (図 3(a))，$ack_i$ によって他のレジスタ $reg_y$ を経由して $reg_k$ に至る type2 (図 3(b))，$ack_i$ から次の制御モジュール $ctrl_{i+1}$ の $in_{i+1}$ を経由して、$reg_k$ に至る type3 (図 3(c)) である．$hdp_{i,k}$ の最小遅延を $t_{minhdp_{i,k}}$ とし，$hcp_{i,k}$ の最大遅延を $t_{maxhcp_{i,k}}$，ホールド時間を $t_{hold_{i,k}}$，$t_{maxhcp_{i,k}}$ に対するマージンを $hm_{i,k}$ ($hm_{i,k} > 0$) とすると，ホールド制約は以下の不等式で表す事ができる．




**(a)** type1 **(b)** type2

**(c)** type3

**図 3:** ホールド制約に関するデータパス $hdp_{i,k}$ と制御パス $hcp_{i,k}$．(a) ディスティネーションレジスタの出力が入力に至る場合，(b) 同じ制御モジュールに制御された他のレジスタの値がディスティネーションレジスタの入力に至る場合，(c) 制御モジュールからマルチプレクサを経由してディスティネーションレジスタの入力に至る場合．

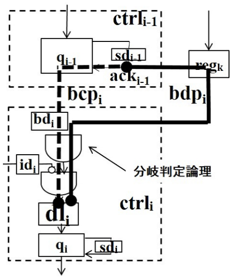



**図 4:** 分岐制約に関するデータパス $bdp_i$ と制御パス $bcp_i$.

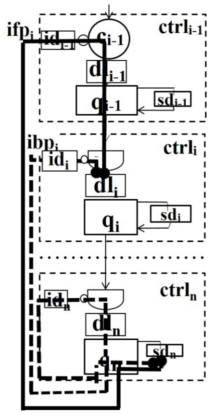



**図 5:** 休止相制約に関するパス $ibp_i$ と $ifp_i$.

$$t_{minhdp_{i,k}} > t_{maxhcp_{i,k}} + hm_{i,k} + t_{hold_{i,k}} \qquad (2)$$

#### 2.2.3 分岐制約

レジスタの値によって制御が別れる場合，レジスタの値は制御信号より先に分岐判定論理に到達していなければならない. これを分岐制約と呼ぶ.

図 4 を用いて分岐制約を解説する．ここで，$sd_{i-1}$ の出力から分岐判定の AND ゲートの入力までのパスを $bcp_i$，$sd_{i-1}$ の出力よりレジスタ $reg_k$ を経由し，レジスタ $reg_k$ から分岐判定 AND ゲートの出力までのデータパスを $bdp_i$，$bcp_i$ の最小遅延を $t_{minbcp_i}$，$bdp_i$ の最大遅延を $t_{maxbdp_i}$，$t_{maxbdp_i}$ に対するマージンを $bm_i$ ($bm_i > 0$) とすると，分岐制約は，以下の不等式で表すことができる．

$$t_{minbcp_i} > t_{maxbdp_i} + bm_i \qquad (3)$$

#### 2.2.4 休止相制約

休止相制約とは，次のデータ処理のために制御モジュールを正しく初期状態に戻すための制約である．

図 5 を用いて休止相制約を解説する．フィードバック信号を生成する最後の制御モジュール $ctrl_n$ の $sd_n$ より $ctrl_n$ の $dl_n$ を通り，制御モジュール $ctrl_i$ の $dl_i$ に至るパスを $ibp_i$ とする．一方，$ctrl_n$ の $sd_n$ より $ctrl_{i-1}$ を通り $ctrl_i$ の $dl_i$ に至るパスを $ifp_i$ とする．$ifp_i$ の最大遅延を $t_{maxifp_i}$，$ibp_n$ の最小遅延を $t_{minibp_n}$，$t_{maxifp_i}$ に対するマージンを $im_i$ ($im_i > 0$) とすると，休止相制約は，以下の不等式で表すことができる．

$$t_{minibp_n} > t_{maxifp_i} + im_i \qquad (4)$$

## 3. 提案手法

束データ方式による非同期式回路は，遅延素子にてタイミングの保証を行う．遅延素子の挿入は，回路面積や性能に影響を及ぼすだけでなく，設計時間にも影響を及ぼす．そのため，遅延素子の挿入数や調整回数を最小化することが求められる．提案手法は，2.2 節で解説したタイミング制約式を基に，遅延素子の挿入位置や調整法を決める．また，調整回数を抑えるために，パスに最大遅延制約のみならず最小遅延制約を与え回路を合成する．

### 3.1 遅延素子の挿入位置

遅延素子の挿入位置は，[1] をベースとする．$sd_i$，$bd_i$，および $id_i$ は，[1] と全く同じ場所で，図 2，図 4，図 5 の位置に挿入する．これらの遅延素子が置かれた信号線は 1 ビットなので，面積を抑えることができる．なお，$bd_i$ の挿入位置は，セットアップ制約のデータパス $sdp_{i,l}$，および制御パス $scp_{i,l}$ 上になる．そのため，3.2 節で解説する通り調整の順番を工夫することで，調整回数を抑える．一方，$id_i$ の位置は他のどのタイミングパスにも影響を及ぼさない．

[1] では，ホールド違反の遅延素子 $hd_k$ は，全てレジスタの直前に置くようにしていた．しかし，データ幅が増えると $hd_k$ もデータ幅分必要になるため，面積が増えてしまう．そのため，本稿では，ホールドパスを 2.2 節で解説した通り 3 種類に分類し，type1 と type2 は $hd_k$ としてレジスタの直前に置き，type3 は $hd_{in_imux}$ として，$in_i$ からマルチプレクサへの制御信号線上に置く．$in_i$ は 1 ビットなので，面積を抑えることがきる．なお，$hd_k$，$hd_{in_imux}$ 共にセットアップ制約のデータパス $sdp_{i,l}$ 上になる．そのため，調整の順番を工夫することで，調整回数を抑える．

### 3.2 遅延素子の調整法

3.1 節で確認した通り，あるタイミング制約に関する遅延素子の挿入は他のタイミング制約に影響を及ぼす．遅延素子と影響を受けるタイミングパスの関係は以下のとおりである．

- $sd_i \rightarrow$ 影響を与えない
- $hd_k, hd_{in_imux} \rightarrow sdp_{i,l}$
- $bd_i \rightarrow bdp_i, bcp_i$
- $id_i \rightarrow$ 影響を与えない

このため，ホールド違反に対する調整はセットアップ違反に対する調整より前に，分岐制約に対する調整もセットアップ違反に対する調整より前に行う．

各タイミング制約共に調整は，遅延の追加，および遅延の削除からなる．セットアップ制約の場合，$t_{maxsdp_{i,l}}$ が，以下の不等式を満足するように調整する．

$$t_{maxsdp_{i,l}} + t_{setup_{i,l}} + sm_{i,l} < t_{minscp_{i,l}} < t_{maxsdp_{i,l}} + t_{setup_{i,l}} + sm_{i,l} + margin_{scp} \quad (5)$$

$t_{minscp_{i,l}}$ が，$t_{maxsdp_{i,l}} + t_{setup_{i,l}} + sm_{i,l}$ より小さい場合，セットアップ違反なので，差の絶対値分の値を持つよう $sd_i$ に遅延素子のセルを追加する．一方，大きい場合，セットアップ制約は満足となるが，性能を悪くする恐れがあるので，$t_{maxsdp_{i,l}} + t_{setup_{i,l}} + sm_{i,l} + margin_{scp}$ を超えないよう，$sd_i$ より遅延素子のセルを削除する．ここで，$margin_{scp}$ は，制御パス $scp$ に対するマージンを表す．ホールド制約，分岐制約，休止相制約についても同様に調整を行う．$margin_{hdp}$，$margin_{bcp}$，$margin_{ibp}$ はそれぞれ，ホールド制約のデータパス $hdp$ に対するマージン，分岐制約の制御パス $bcp$ に対するマージン，休止相制約のパス $ibp$ に対するマージンを表し，$margin_{scp}$ と同様に，遅延素子のセルの削除基準として用いる．なお，ホールド制約の type1 と type2 に関しては，レジスタのデータ入力が複数ビットになる可能性があるが，全ビット同様に行うのではなく，ビット毎に調整を行うことで無駄なセルの追加や削除を行わないようにする．一方，休止相制約では，$ifp_i$ は $ctrl_i$ の直前の制御モジュール $ctrl_{i-1}$ を通ることになる．$ctrl_{i-1}$ の $id_{i-1}$ を調整すると，(4) 式の右辺の $t_{maxifp_i}$ に影響を与えるため，$id_{i-1}$ の調整の後に $id_i$ の調整を行う．

### 3.3 最大・最小遅延制約の生成

非同期式回路ではクロック信号がないため，目標性能を達成するためにはパス遅延制約を与える必要がある．[1] では，非パイプライン回路ではレイテンシ制約から，パイプライン回路ではサイクルタイム制約からセットアップ制約に関するパス $sdp_{i,l}$ と $scp_{i,l}$ に最大遅延制約を与え回路合成を行っている．

本研究では，[1] 同様，セットアップ制約に関する全てのパスに最大遅延制約を与える．一方，[1] とは異なり，制御パスには最大遅延制約の他に最小遅延制約も与える．また，休止相制約のパスの一部に最大遅延制約を与える．セットアップ制約に関しては，制御パスの最小遅延制約値よりデータパス回路の最大遅延制約値が小さくなるように制約を生成する．合成の段階でタイミング制約を満足しやすくすることで，調整回数の削減や面積削減，および性能向上を試みる．

非パイプライン回路でレイテンシ制約を考慮して回路合成を行う場合，状態毎に適切な遅延を実現することで，性能や面積の最適化を行う．そのために，最大・最小遅延制約の生成の前に，レイアウト合成まで行い，静的タイミング解析 (Static Timing Analysis, STA) を行い，各データパス $sdp_{i,l}$ の最大遅延を求める．ある状態 $i$ における，データパスの最大遅延を $t_{maxsdp_i}$ とする．全状態の遅延の和に対する，状態 $i$ の遅延の割合を $R_i$ とすると，$R_i$ は以下の式で求めることができる．

$$R_i = \frac{t_{maxsdp_i}}{\Sigma_{i=0}^{n} t_{maxsdp_i}} \quad (6)$$

なお，分岐で制御パスが複数に分かれる場合，パス毎に $R_i$ を求める．同様に繰り返しが存在する場合，繰り返し内部の制御パスとそれ以外の制御パスで分けて $R_i$ を求める．

状態 $i$ における，制御パスとデータパスの最大遅延制約値 $c_{maxscp_i}$，$c_{maxsdp_i}$ を，以下の式で求める．

$$c_{maxscp_i} = L * CR_{max} * R_i \quad (7)$$

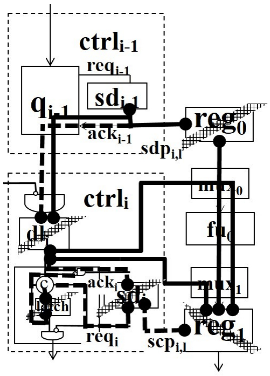



図 6: セットアップ制約に関するデータパス $sdp_{i,l}$ と制御パス $scp_{i,l}$ の分割.

$$c_{maxsdp_i} = L * DR_{max} * R_i \quad (8)$$

一方，状態 $i$ における，制御パスの最小遅延制約値 $c_{minscp_i}$ を，以下の式で求める．

$$c_{minscp_i} = L * CR_{min} * R_i \quad (9)$$

$L$ はレイテンシ制約値を，$CR_{max}$ はレイテンシ制約値のうち制御回路の最大遅延の割合を，$CR_{min}$ はレイテンシ制約値のうち制御回路の最小遅延の割合を，$DR_{max}$ はレイテンシ制約値のうちデータパス回路の最大遅延の割合を表す．$CR_{max}$，$CR_{min}$，$DR_{max}$ は，$CR_{max} > CR_{min} > DR_{max}$ となるように設定する．$CR_{min} > DR_{max}$ より，セットアップ制約の左辺が右辺より大きな値を持つように制約を生成する．なお，$CR_{max} = 1$ の場合，レイテンシ制約値を目標に回路合成を行うことを意味する．

(7)，(8)，(9) 式を用いてセットアップ制約に関するパスに対する，最大・最小遅延制約を生成する．データパス $sdp_{i,l}$ は，図 6 に示す通り，直前の制御モジュール $ctrl_{i-1}$ の遅延素子 $sd_{i-1}$ の出力からソースレジスタを経由し，ディスティネーションレジスタに至るまで，あるいは制御モジュール $ctrl_i$ のラッチ $dl_i$ を経由し，ディスティネーションレジスタに至るまでの 2 種類が存在する．今，データパス $sdp_{i,l}$ をソースレジスタ，あるいはラッチ $dl_i$ で分割する．$t_{maxsdp_{i,l}}$ のうち $sd_{i-1}$ の出力からソースレジスタ，あるいは $dl_i$ までの遅延の重みを $w_{sdp_{sd2reg}}$，ソースレジスタ，あるいは $dl_i$ からディスティネーションレジスタまでの遅延の重みを $w_{sdp_{reg2reg}}$ とする．$sd_{i-1}$ からソースレジスタ，あるいはラッチ $dl_i$ までの最大遅延制約値は，$c_{maxsdp_i} * w_{sdp_{sd2reg}}$ とする．一方，ソースレジスタ，あるいはラッチ $dl_i$ からディスティネーションレジスタまでの最大遅延制約値を $c_{maxsdp_i} * w_{sdp_{reg2reg}}$ とする．制御パス $scp_{i,l}$ は，図 6 に示す通り，直前の制御モジュール $ctrl_{i-1}$ の遅延素子 $sd_{i-1}$ の出力からラッチ $dl_i$，ラッチ $dl_i$ から遅延素子 $sd_i$，遅延素子 $sd_i$ から Q モジュール内部のラッチ，Q モジュール内部のラッチから遅延素子 $sd_i$，遅延素子 $sd_i$ 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 5 つに分割する．$t_{minscp_{i,l}}$ に対する 5 つの部分制御パスそれぞれの遅延の重みを $w_{scp_{sd2dl}}$，$w_{scp_{dl2sd}}$，$w_{scp_{sd2q}}$，$w_{scp_{q2sd}}$，$w_{scp_{sd2reg}}$ とする．各重みを (7) 式に掛けることで各部分制御パスの最大遅延制約値を求める．同様に，各重みを (9) 式に掛けることで各部分制御パスの最小遅延制約値を求める．なお，同一部分パスに対して，複数の制約値を生成した場合，値が厳しい方を利用する．なお，$w_{sdp_{sd2reg}}$ のような重みの値は，最初の合成後の STA で得

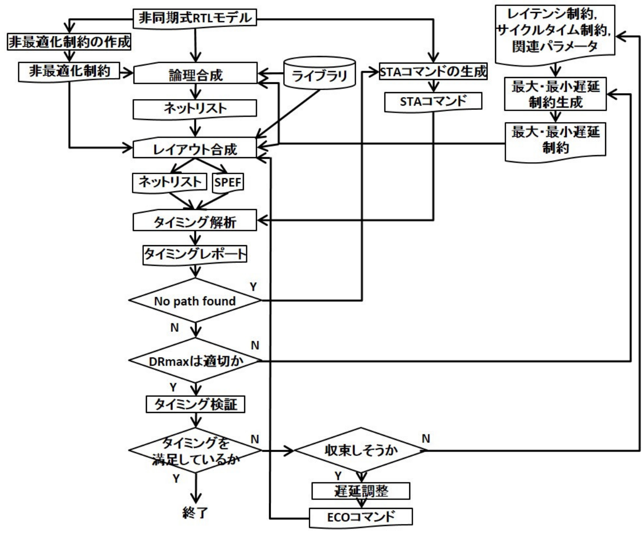



図 **7:** 設計フロー.

たパス遅延の内の部分パス遅延の割合より求める.

パイプライン回路の場合,(7),(8),(9) 式の L がサイクルタイム制約値 $CT$ となる.また,全てのパイプラインステージで同じ遅延になることが望ましいので,$R_i$ を用いずに最大・最小遅延制約を生成する.一方,データパスがマルチサイクルの場合,(8) 式にサイクル数を掛けた値がデータパスの最大遅延制約となる.

休止相制約に関しては,最後の制御モジュール $ctrl_n$ から各制御モジュール $ctrl_i$ のラッチ $dl_i$ までの遅延が支配的になる.そのため,最初の合成後,最後の制御モジュール $ctrl_n$ の遅延素子 $sd_n$ の出力から,各制御モジュール $ctrl_i$ のラッチ $dl_i$ までの遅延を STA で求め,その時の最小値を制約値として用いる.

## 4. 設計フロー

本稿では,[1] で提案された設計フローをベースに,設計を行う.図 7 は,本稿で用いる設計フローを表す.[1] との大きな違いは,対象を FPGA ではなく ASIC にしているということ,および $CR_{max}$,$CR_{min}$,$DR_{max}$ の決め方を,STA をベースにしているという点である.

フローの入力は,束データ方式による非同期式回路の Verilog RTL モデル,レイテンシ制約やサイクルタイム制約とそれらに関するパラメータ,およびテクノロジーライブラリである.初期的に,各制御モジュールの遅延素子 $sd_i$ は 1 つのバッファーで実現する.それ以外の遅延素子は,入力信号を出力信号に接続したダミーモジュールとし,モデルの中に含めておく.

合成で制御モジュール内の Q モジュール $q_i$ と全ての遅延素子に最適化が掛かってしまうと,ハザードやタイミング違反の原因となるので,これらに対して非最適化制約を生成する.また,タイミング検証が必要な全てのパスに対する STA コマンドを生成する.STA コマンドでは,パスの始点や終点を指定するが,最適化でピンやネットがリネームされる恐れがある.そのため,STA でパス遅延が解析できない場合(例えば,no path found),始点や終点名を変更し STA コマンドを再生成する.

レイテンシ制約やサイクルタイム制約とそれらに関わるパラメータは,レイテンシ制約値 $L$,サイクルタイム制約値 $CT$,および最大・最小遅延制約を生成するために用いる

表 **1:** 各手法における遅延調整回数と遅延素子のセル数.

| DIFFEQ | 調整回数 | セル数 |
|---|---|---|
| 非同期式回路 [1] | 10 | 412 |
| 提案手法 w/o min | 5 | 97 |
| 提案手法 with min | 3 | 45 |
| EWF | 調整回数 | セル数 |
| 非同期式回路 [1] | 22 | 1,404 |
| 提案手法 w/o min | 6 | 373 |
| 提案手法 with min | 7 | 240 |

$CR_{max}$,$CR_{min}$,$DR_{max}$ である.この他にも,タイミング制約式の右辺のパスに対するマージン $sm_{i,l}$,$hm_{i,k}$,$bm_i$,$im_i$ や,タイミング制約式の左辺のパスに対するマージン $margin_{scp}$,$margin_{hdp}$,$margin_{bcp}$,$margin_{ibp}$ を含む.

束データ方式による非同期式回路の場合,遅延素子を含む制御回路の遅延や面積は,データパス回路に依存する.そのため,面積や性能の面で良い回路を得るために,$DR_{max}$ を決める必要がある.設計フローでは,最初に $DR_{max}$ を固定し,3.3 節で解説した方法でデータパスの最大遅延制約のみを生成し,論理合成,レイアウト合成,STA までを行う.データパスの STA 値と最大遅延制約値の差,および回路面積の増減を確認した上で $DR_{max}$ の値を調整する.調整後,3.3 節で解説した方法で制御パスや休止相に関するパスに対して最大・最小遅延制約を生成する.

最大・最小遅延制約を用いて再合成を行い,全てのタイミング制約が満足するまで遅延調整を行う.遅延調整は,3.2 節で解説した方法を用いる.この時,タイミング制約に違反した全てのパスを 3.2 節で説明した順番で調整し,Engineering Change Order (ECO) コマンドを生成する.調整回数が多く,収束する見込みがない時は,タイミング制約式の左辺のパスに対するマージン($margin_{scp}$ など)を大きくし,最大・最小遅延制約を変えたうえで再合成を行う.

## 5. 実験

実験では,提案手法を用いて differential equation (DIFFEQ) と Elliptic Wave Filter (EWF) を合成する.本稿では,レイテンシ制約を与え,非パイプライン回路を合成する.提案手法のうち,最小遅延制約を生成しなかった非同期式回路を(提案手法 w/o min),最小遅延制約を生成した非同期式回路(提案手法 with min)とする.また,比較として,同期式回路と [1] を用いて合成した非同期式回路(非同期式回路 [1])も合成する.[1] では FPGA を対象としていたため,プリミティブの利用やコマンドの生成を ASIC 向けに修正した.始めに,合成した非同期式回路の調整回数や遅延素子で用いたセル数を評価する.次に,回路面積と消費エネルギーを評価する.提案手法のうち,遅延素子の調整など一部は Perl や Excel を用いて自動化した.論理合成は Synopsys の Design Compiler (I-2013.12-SP2),レイアウト合成は Cadence の EDI (14.2),論理シミュレーションは Synopsys の VCS (I-2014.03),STA は Synopsys の PrimeTime (H-2013.06-SP3-5) を使用した.また,eShuttle 65nm のセルライブラリを使用した.

レイテンシ制約やパラメータを決めるにあたって,STA やデザインルールチェックを満足し,クロックサイクルタイムが最小となる同期式回路を探索した.DIFFEQ,EWF 共に 2,000ps であった.レイテンシ制約値 $L$ は,クロックサイクルタイム×状態数で,DIFFEQ は 8,000ps(4 状態),

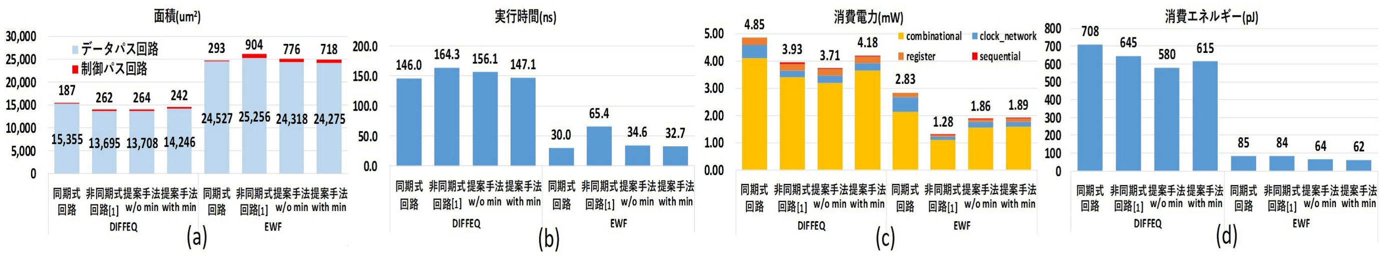



**図 8:** 実験結果．(a) 回路面積，(b) 実行時間，(c) 動的消費電力，(d) 消費エネルギー．

EWF は 30,000ps（15 状態）である．次に，$DR_{max}$ を 1 から 0.05 づつ減らしていき，データパスに対して最大遅延制約を生成し，レイアウト合成を行う．データパスの 8 割以上でレジスタ間遅延の STA 値が最大遅延制約を満足する最小の $DR_{max}$ を探索した．実験では，全てのケースで $DR_{max}$ は 0.8 となった．初期的に，$sm_{i,l}$，$hm_{i,k}$，$bm_i$，$im_i$ を，クロックサイクルタイムの 10%（200ps）とした．$margin_{scp}$，$margin_{hdp}$，$margin_{bcp}$，$margin_{icp}$ は，クロックサイクルタイムの 5%（100ps）とした．$CR_{min}$ は 0.95（0.80+0.10+0.05），$CR_{max}$ は 1.00（0.95+0.05）とした．DIFFEQ の合成において，[1] の手法では，調整が数十回となったため，最終的に $CR_{min}$ を 1.00（0.80+0.10+0.10），$CR_{max}$ を 1.05（1.00+0.05）として合成した．

表 1 は，各手法における遅延調整回数，および遅延素子のセル数を表す．なお，DIFFEQ にのみ分岐があるが，分岐制約違反がなかったため $bd_i$ の調整はなかった．そのため，遅延素子のセル数は，$sd_i$，$hd_k$，$hd_{in_i max}$，$id_i$ で用いられたバッファー数を表す．

表 1 では，最小遅延制約がない場合（提案手法 w/o min）もある場合（提案手法 with min）も [1] と比べ，調整回数や遅延素子のセル数が少ない．提案手法による遅延素子の挿入位置，調整の順番が有効であることを示している．セル数が大きく減った理由は，ホールド制約違反をビットレベルで行ったためである．最小遅延制約がある場合は，最小遅延制約がない場合と比べ遅延素子のセル数は更に減少した．調整回数に関しては，DIFFEQ は 2 回減ったが，EWF は 1 回多かった．$CR_{min}$ と $CR_{max}$ の差は 0.15 だが，遅延で換算すると 300ps と非常に小さく，タイミング違反数の減少につながらなかったことが増加の原因である．

次に，合成した回路の回路面積と消費エネルギーを示す．消費エネルギーは，実行時間と動的消費電力の積として表すため，実行時間と動的消費電力も表す．図 8(a) は回路面積を，図 8(b) は実行時間を，図 8(c) は動的消費電力を，図 8(d) は消費エネルギーを表す．回路面積は EDI のレポートより得た値，実行時間は任意のテスト入力を与え VCS でシミュレーションした時の値，動的消費電力はシミュレーション時に生成した回路のスイッチング情報を PrimeTime に与えた時の値である．

回路面積は [1] と比べ，平均で DIFFEQ は 4.2%，EWF は 4.5%削減された．遅延素子の面積が削減されたことが原因である．実行時間は [1] と比べ，DIFFEQ では，最小遅延制約がない場合は 5.0%，ある場合は 10.4%改善した．EWF では，最小遅延制約がない場合は 47.1%，ある場合は 49.9%改善した．[1] では，ホールド違反で多くの遅延素子がレジスタの直前に入り，それに伴いセットアップ制約も影響を受け $sd_i$ が大きくなったため性能が悪化した．最小遅延制約がある場合は，ない場合と比べどちらも改善した．挿入した遅延素子数が少ないことが原因である．消費電力は [1] と比べ，DIFFEQ では，最小遅延制約がない場合は 5.6%改善したが，ある場合は 6.5%増加した．最小遅延制約による遅延素子の減少より演算器の増加が上回ったことで combinational の消費電力が増加したことが原因である．EWF では，最小遅延制約がない場合 45.3%，ある場合は 47.7%増加した．増加の原因は，実行時間の長さが挙げられる．最小遅延制約がある場合は，ない場合と比べどちらも増加した．先に述べたとおり，最小遅延制約による遅延素子の減少より演算器の増加が上回ったことが原因である．消費エネルギーは [1] と比べ，平均で DIFFEQ は 16.9%、EWF は 15.4%削減された．最小遅延制約がある場合はない場合と比べ，DIFFEQ は 6.0%増加し，EWF は 3.1%減少した．最小遅延制約がない方が動的消費電力が小さかったのが増加の原因である．

実験より，提案手法によって調整回数，回路面積，消費エネルギーの全てを改善した．しかし，最小遅延制約の与え方によっては，消費電力が増加する可能性がある．

## 6. 結論

本稿では，束データ方式による非同期式回路の遅延調整法を考察をした．タイミング制約式を基に遅延素子の挿入位置，調整順番を考察するとともに，最大遅延制約のみならず最小遅延制約を与え回路を合成する．実験では，提案手法によって，調整回数，回路面積，消費エネルギーの全てにおいて削減を確認した．今後は，規模の大きな回路やパイプライン回路で実験を行う．

**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，eShuttle の協力で行われたものである．また，本研究は JSPS 科研費 15K00080 の助成による．

### 参考文献

[1] 滝澤，齋藤，“束データ方式による非同期式回路の FPGA 設計支援環境の構築”, 2015-SLDM-171, 5, pp.1–6, 2015.

[2] S. Chakraborty, D. I. Dill, K. Y. Yun, ”Min-max timing analysis and an application to asynchronous circuits”, Proceedings of the IEEE , Vol:87, Issue:2, pp.332 - 346, 1999.

[3] J.　Sparso and S.　Furber,“Principles of asynchronous circuit design: a systems perspective”, Springer, 2001.

[4] F. U. Rosenberger, C. E .Molnar, T. J. Chaney, and T P. Fang, ” QModules: Internally Clocked Delay Insensitive Modules”, IEEE Transaction of Computer, vol.C-37, no.9, pp.1005-1018, 1988.